Project Plan Document

CareConnect

University of Maryland Global Campus

SWEN 670 – Software Engineering Capstone

Dr. Mir Assadullah

Spring 2026

Contributors: Parthav Dani, Pemon Kouadio, Mawuko Kpelevi, Jordene Downer, Melissa Burgos

Revision History 6
1. Executive Summary 7
1.1 Project Name 7
1.2 Purpose 7
1.3 Scope 7
2. Objectives and Goals 8
2.1 Goals 8
2.2 Objectives 8
3. Scope of Work 8
3.1 In-Scope 8
3.1.1 Requirements Analysis and Documentation 9
3.1.2 Architectural Design for Communication Features 9
3.1.3 Development of Real-Time Chat Messaging Functionality 9
3.1.4 Development of Audio and Video Calling Capabilities 9
3.1.5 Implementation of Role-Based Communication Permissions 9
3.1.6 UI Development for EVV and In-Home Support Features 9
3.1.7 Integration with AWS-Hosted Services 10
3.2 Out-of-Scope 10
3.2.1 Payment Processing and Subscription Management 10
3.2.2 Regulatory Certification and Compliance Audits 10
3.2.3 Emergency Response and Clinical Decision Systems 10
3.2.4 Full User Experience (UX) Redesign 10
3.2.5 Long-Term Operational Monitoring and Support 11
3.2.6 Third-Party Device and Wearable Integrations 11
4. Stakeholders 11
4.1 Internal Stakeholders 11

Project Team Members 11
Project Lead 11
Product Owner 11
Course Instructor 12
4.2 External Stakeholders 12
Patients 12
Caregivers 12
Cloud Service Providers 12
5. Timeline & Milestones 12
5.1 Timeline and Milestone 13
5.2 Milestone Descriptions 13
Milestone One – Project Initiation 13
Milestone Two – System Design and Test Planning 13
Checkpoint Meeting – Midpoint Review 13
Milestone Three – Implementation and Deployment Preparation 14
Milestone Four – Validation and Final Documentation 14
5.3 Gantt Chart and Work Breakdown structure 14
6. Technical Architecture 18
6.1 Architectural Objectives 18
6.2 Logical System Architecture 19
6.3 High-Level System Architecture Diagram 19
6.4 Architectural Rationale 19
6.5 Runtime Flows 20
6.6 Technology Stack Summary 20
6.7 Deployment Topology Highlights 21
6.8 Frontend 21
6.9 Backend 22

6.10 Database 22
6.11 APIs 23
6.11.1 External Health and Device APIs 23
6.12 AI Services 23
6.12.1 Purpose and User Flow 24
6.12.2 Current State and Motivation for Migration 24
6.12.3 Target Architecture Using AWS Bedrock 25
6.12.4 Security, Privacy, and Data Protection 25
6.12.5 Technical Approach 25
6.12.6 Cost Management and Operational Considerations 26
6.12.7 AI Migration to AWS Bedrock 27
6.12.8 Governance, Compliance, and Non-Clinical Boundaries 27
6.12.9 Summary and Future Enhancements 28
6.13 Hosting 28
6.13.1 Cloud Deployment Model 28
6.13.2 Containerization and Orchestration 28
6.13.3 Serverless-First and Scale-to-Zero 29
6.13.4 Security and Reliability 29
6.14 Security 29
7. Project Deliverables 29
8. Risk & Mitigation 30
8.1 Risk Probability Matrix 30
8.2 Risk Identification, Probability, and Mitigation 31
9. Budget Estimate 32
9.1 Labor Cost 32
9.2 Materials Cost 33
9.3 Infrastructure Cost 33

9.3.1 AWS Core Services 33
9.3.2 Third-Party / One-Time Fees 34
9.3.3 Total Budget for MVP Phase (12 Weeks) 34
9.3.4 Cost-Control and Optimization Plan 34
9.3.5 Alignment to Financial Policies 35
9.3.6 Budget Summary for Project Plan 35
10. Roles & Responsibilities 35
10.1 Roles 35
10.2 RACI Matrix 36
11. Monitoring and Reporting 37
11.1 Intra-Team Monitoring and Reporting 37
11.2 Project Communications 38
12. Assumptions 38
13. Constraints 38
14. Change Management Procedures 39
14.1 Change Identification and Initiation 39
14.2 Impact Analysis 39
14.3 Review and Approval 40
15.4 Implementation and Documentation 40
16. Appendices 41
16.1 Glossary of Terms 41
16.2 References 43

Revision History

Team Name	Date	Reason for Changes	Version
CareConnect Team	01/24/2026	Initial Document Submission	1.0

1. Executive Summary

1.1 Project Name

CareConnect – Communication, EVV, In House Residential Support and Platform Enhancements

1.2 Purpose

The purpose of this project is to enhance the CareConnect mobile application by implementing critical communication and platform improvements that support real-time interaction, scalability, and readiness for production deployment. The project focuses on strengthening communication between patients and caregivers while modernizing supporting services to align with cloud-based best practices.

This project is conducted as part of an academic software engineering initiative and emphasizes real-world system design, implementation, and validation practices. The enhancements are intended to improve CareConnect's ability to support in-home care scenarios, increase responsiveness between users, and prepare the application for potential publication in mobile app marketplaces.

1.3 Scope

The scope of this project includes the design, implementation, testing, and validation of selected enhancements to the CareConnect application. These enhancements focus on communication features, in-home care support functionality, and cloud service modernization.

Specifically, this project includes:

- Person-to-person chat messaging between patients and caregivers
- Patient/caregiver audio and video calling functionality
- UI enhancements related to EVV and in-home residential support
- Migration and evaluation of AI services using AWS Bedrock
- Backend and frontend integration within an AWS-hosted environment

The project does not attempt to redesign the entire CareConnect platform, but instead targets specific enhancements that can be completed within a single academic term.

2. Objectives and Goals

2.1 Goals

The primary goal of this project is to improve communication and coordination between patients and caregivers using the CareConnect platform. By introducing real-time messaging and calling capabilities, the system aims to support timely interaction, reduce reliance on external communication tools, and provide a more integrated care experience.

Additional goals include improving system scalability, enhancing cloud readiness, and ensuring that new features adhere to security and performance expectations appropriate for a healthcare-related application.

2.2 Objectives

To achieve the stated goals, the project defines the following objectives:

- Replace polling-based messaging mechanisms with real-time communication approaches suitable for live interaction
- Implement role-based communication rules that govern who may initiate and participate in conversations
- Enable audio and video calling capabilities that support both one-to-one and conference communication
- Integrate EVV and in-home residential support features using existing data models
- Evaluate and integrate AWS-based AI services while considering cost and scalability
- Deliver a functional, testable application suitable for demonstration and evaluation at the end of the semester

Each objective is measurable through functional testing, system validation, and project deliverables.

3. Scope of Work

3.1 In-Scope

This section identifies the project activities and deliverables that are included within the scope of the CareConnect enhancement project. Each subsection represents a major work item that will be completed during the project's lifecycle.

3.1.1 Requirements Analysis and Documentation

- The project shall include the analysis, refinement, and documentation of functional and non-functional requirements for the assigned CareConnect enhancements.
- Requirements will be captured in formal artifacts such as a Software Requirements Specification (SRS) and supporting documentation.

3.1.2 Architectural Design for Communication Features

- The project shall include the design of a high-level and logical architecture to support real-time communication features.
- Architectural decisions will consider scalability, security, AWS compatibility, and integration with existing CareConnect components.

3.1.3 Development of Real-Time Chat Messaging Functionality

- The project shall include the implementation of person-to-person chat messaging between patients and caregivers.
- Messaging functionality will support real-time delivery and enforce role-based communication rules.

3.1.4 Development of Audio and Video Calling Capabilities

- The project shall include the implementation of audio and video calling functionality between patients and caregivers.
- Calling capabilities will support session initiation, acceptance, termination, and multi-party communication where applicable.

3.1.5 Implementation of Role-Based Communication Permissions

- The project shall include the enforcement of communication rules that govern who may initiate or participate in chat and calling features.
- Role-based permissions will be applied consistently across all communication functionalities.

3.1.6 UI Development for EVV and In-Home Support Features

- The project shall include user interface development to support EVV and in-home residential support functionality.

- UI components will be implemented in alignment with existing data models and system workflows.

3.1.7 Integration with AWS-Hosted Services

- The project shall include the integration of backend services hosted within an AWS environment.
- AWS services will be leveraged to support real-time communication, scalability, and secure system operation.

3.2 Out-of-Scope

This section identifies activities and system capabilities that are explicitly excluded from the scope of the CareConnect enhancement project. Excluding these items helps prevent scope creep and ensures the project remains achievable within the defined timeline.

3.2.1 Payment Processing and Subscription Management

- The project shall not include the implementation or modification of payment processing or subscription management functionality.
- Integration with third-party or native mobile payment platforms is excluded from this project.

3.2.2 Regulatory Certification and Compliance Audits

- The project shall not include formal regulatory certification or compliance audits.
- Legal, clinical, or governmental approval processes are outside the scope of this project.

3.2.3 Emergency Response and Clinical Decision Systems

- The project shall not include automated emergency response systems or clinical decision-making logic.
- Communication features are not intended to replace emergency services or medical professionals.

3.2.4 Full User Experience (UX) Redesign

- The project shall not include a complete redesign of the CareConnect user experience.
- UI changes are limited to necessary enhancements supporting assigned features.

3.2.5 Long-Term Operational Monitoring and Support

- The project shall not include long-term system monitoring, maintenance, or operational support beyond the project timeline.
- Post-deployment operational responsibilities are excluded.

3.2.6 Third-Party Device and Wearable Integrations

- The project shall not include new integrations with external devices or wearable platforms.
- Existing integrations, if any, will not be expanded or modified.

4. Stakeholders

This section identifies the key stakeholders involved in or affected by the CareConnect enhancement project. Stakeholders include individuals and groups who have an interest in the project outcomes, influence project direction, or are impacted by the system once implemented.

4.1 Internal Stakeholders

Internal stakeholders are individuals or groups directly involved in the planning, development, oversight, and evaluation of the project.

Project Team Members

Project team members are responsible for executing the project activities, including requirements analysis, design, implementation, testing, and documentation. They collaborate to ensure that project objectives are met within the defined scope and timeline.

Project Lead

The project lead oversees project execution, coordinates team activities, and ensures alignment with project objectives and academic requirements. The project lead also serves as the primary point of contact for project-related communication.

Product Owner

The product owner provides domain knowledge, clarifies requirements, and prioritizes features based on project goals. The product owner ensures that enhancements align with the intended vision of the CareConnect platform.

Course Instructor

The course instructor provides academic guidance, evaluates project deliverables, and ensures that the project meets course learning objectives and assessment criteria.

4.2 External Stakeholders

External stakeholders are individuals or entities that are not directly involved in project execution but are impacted by the system or provide supporting services.

Patients

Patients are end users of the CareConnect application who rely on the platform for communication, care coordination, and support. Their interaction with the system influences usability and effectiveness.

Caregivers

Caregivers are primary end users responsible for managing patient care and communication. They rely on the application to coordinate tasks, communicate with patients, and monitor care-related activities.

Cloud Service Providers

Cloud service providers, such as AWS, supply the infrastructure and services required to host and operate the CareConnect platform. Their availability and reliability impact system performance and scalability.

5. Timeline & Milestones

This section outlines the timeline and major milestones for the CareConnect enhancement project. The project spans a twelve-week period and is structured around four primary milestones, with an additional midpoint checkpoint used to assess progress. Each milestone represents the completion of a significant phase of work and includes the production of key project artifacts.

5.1 Timeline and Milestone

Table 1 - Timelines and milestones

Milestone #	Milestone Name	Due Date
1	Milestone One – Project Initiation	January 24
2	Milestone Two – System Design and Test Planning	February 7
—	Checkpoint Meeting (Midpoint Review)	—
3	Milestone Three – Implementation and Deployment Preparation	March 21
4	Milestone Four – Validation and Final Documentation	March 31

5.2 Milestone Descriptions

Milestone One – Project Initiation

Milestone One establishes the foundation for the project by defining its scope, objectives, and requirements. During this phase, project planning activities are completed, and a comprehensive understanding of system expectations is documented. The project plan outlines how the work will be organized and managed, while the Software Requirements Specification (SRS) formally captures functional and non-functional requirements. Completion of this milestone ensures that all stakeholders have a shared understanding of the project direction before design and implementation begin.

Milestone Two – System Design and Test Planning

Milestone Two focuses on translating documented requirements into a technical solution. During this phase, the system architecture is designed, including communication components, integration points, and deployment considerations. Design decisions are documented to provide a clear blueprint for implementation. In parallel, test planning activities are conducted to define the testing approach, scope, and strategies that will be used to verify system behavior. This milestone ensures that both the system design and validation approach are clearly defined prior to extensive development work.

Checkpoint Meeting – Midpoint Review

The checkpoint meeting occurs at approximately the halfway point of the project timeline and serves as an informal review of project progress. Although no formal submission is required, this meeting provides an opportunity to evaluate completed work, identify risks or challenges, and make adjustments to the project plan if necessary. Feedback gathered during this checkpoint helps guide the remaining implementation and documentation activities.

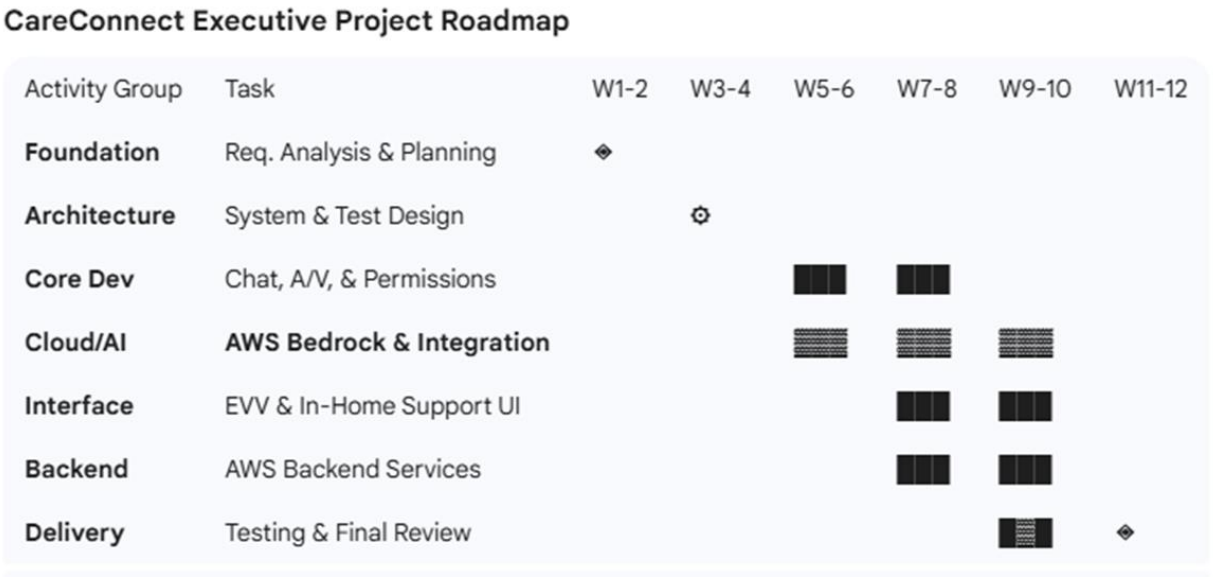
Milestone Three – Implementation and Deployment Preparation

Milestone Three emphasizes the completion of core development activities and preparation for system deployment. During this phase, communication features and supporting functionality are implemented and integrated. Documentation is produced to support developers and system operators, including guidance on code structure, configuration, and deployment procedures. Testing activities continue to validate implemented features and refine test coverage as needed. Completion of this milestone indicates that the system is functionally complete and ready for final validation.

Milestone Four – Validation and Final Documentation

Milestone Four concludes the project with comprehensive system validation and final documentation. During this phase, testing results are consolidated and analyzed to confirm that system requirements have been met. End-user documentation is finalized to support application usage and feature understanding. This milestone represents the formal completion of the project and readiness for final evaluation and submission.

5.3 Gantt Chart and Work Breakdown structure



1.0 Phase 1: Foundation (Weeks 1–2)
1.1 Requirements Elicitation
1.1.1 Define functional requirements for Real-Time Chat and A/V calling
1.1.2 Define functional requirements for EVV & In Home Residential Support module
1.1.3 Define technical requirements for AWS Bedrock migration epic
1.2 Project Setup & Governance
1.2.1 Initialize Git repositories and branching strategy.
1.2.3 Finalize the Project Management Plan and SRS documentation
2.0 Phase 2: Architecture & Design (Weeks 3–4)
2.1 System Architecture Design
2.1.1 Create high-level microservices diagram (Auth, Chat, AI, Billing, EVV).
2.1.2 Design and confirm the database schema for visits and other EVV specific data.

2.2 Communication & Security Design
2.2.1 Design WebSocket protocol for real-time messaging.
2.2.2 Define JWT-based authentication and Role-Based Access Control (RBAC) levels.
2.3 Quality Assurance Planning
2.3.1 Define unit, integration, and user acceptance test strategy and plan
2.3.2 Setup automated testing framework (JUnit).
3.0 Phase 3: Core Feature Development (Weeks 5–8)
3.1 Real-Time Communication Suite
3.1.1 Develop WebSocket server for Chat.
3.1.2 Implement message history and read receipts.
3.1.3 Integrate WebRTC for Audio/Video calling logic.
3.2 Security & Permissions
3.2.1 Implement access controls on EVV and In-house Residential support module based on the RBAC strategy
3.2.2 Conduct security audit on patient data access points.
3.3 EVV & In House Residential Support module
3.3.1 Develop the UI pages to support viewing, modification and auditing of the visit data
3.3.2 Develop the backend interface to submit the visits data in the recommended format to EVV Aggregator
3.3.3 Develop the backend to support add, update and maintain IADLs and ADLs related activities for visits in In house Residential support module

4.0 Phase 4: AI & AWS Integration (Weeks 5–10) [CRITICAL PATH]
4.1 AWS Bedrock Implementation
4.1.1 Evaluate AWS Bedrock or any other AWS service to replace Deepseek, openrouter etc.
4.1.2 Develop OpenRouterService / BedrockService integration.
4.2 UI/UX Development
4.2.1 Build Electronic Visit Verification (EVV) check-in/check-out screens.
4.2.2 Develop the In-Home Support dashboard for caregivers.
4.3 AWS Bedrock migration
4.3.1 Provision AWS S3 buckets for document storage.
5.0 Phase 5: Quality, Documentation & Review (Weeks 9–12)
5.1 Comprehensive Testing
5.1.1 Create and execute end-to-end (E2E) testing of the Communication Module
5.1.2 Create and execute unit and integration tests for In House Residential support and EVV module
5.1.3 Conduct User Acceptance Testing (UAT) with mock clinical data.
5.2 Finalization & Deployment
5.2.1 Compile technical documentation and API Swagger docs.
5.2.2 Final Stakeholder Review and Project Sign-off.

6. Technical Architecture

This section describes the technical architecture for the CareConnect enhancement project. The architecture is designed to support real-time communication, modular development, scalability, and secure deployment within an AWS-hosted environment. The focus of this section is on high-level structure and design intent rather than low-level implementation details.

6.1 Architectural Objectives

The technical architecture for this project is guided by several key objectives that align with both system requirements and project constraints.

The primary architectural objective is to support **real-time communication** between patients and caregivers, including chat messaging and audio/video calling. The architecture must enable low-latency interaction while remaining reliable under varying network conditions.

Additional objectives include:

- Supporting real-time chat and audio/video communication without polling-based mechanisms
- Ensuring persistent connectivity for signaling and messaging
- Avoiding serverless technologies unsuitable for long-lived sessions
- Leveraging AWS-hosted services where appropriate
- Minimizing operational complexity and infrastructure cost
- Enforcing security, privacy, and compliance requirements

These objectives influence architectural decisions across all system components.

6.2 Logical System Architecture

The CareConnect system follows a **client–server architecture** in which mobile clients interact with backend services hosted within AWS. The logical architecture separates responsibilities across distinct layers to improve maintainability and scalability.

At a high level, the system consists of:

- Mobile client applications used by patients and caregivers
- Backend services responsible for authentication, authorization, and business logic
- Real-time communication services supporting chat and call signaling
- Data storage services for persistent system data

Communication between components occurs through secure APIs and real-time communication channels. This separation of concerns allows individual components to evolve independently while maintaining integration across the system.

6.3 High-Level System Architecture Diagram

The high-level system architecture illustrates how frontend clients, backend services, and AWS infrastructure interact to support system functionality. The diagram highlights major components and communication paths without exposing internal implementation details.

The diagram includes:

- Mobile clients (patient and caregiver applications)
- API layer for backend communication
- Real-time communication services
- Data storage components
- AWS hosting environment

This diagram serves as a conceptual reference for understanding system structure and interactions.

6.4 Architectural Rationale

The chosen architecture emphasizes modularity and cloud compatibility. By separating communication features from core business logic, the system can scale and adapt more easily as requirements evolve.

Using AWS-hosted services allows the system to leverage managed infrastructure for reliability, security, and scalability. Real-time communication components are designed to operate independently from traditional request–response APIs, enabling efficient handling of live interactions.

This architectural approach balances performance needs with development simplicity and aligns with modern cloud-based application design principles.

6.5 Runtime Flows

Runtime flows describe how system components interact during normal operation. Key runtime scenarios include:

- User authentication and authorization prior to accessing features
- Initiation of chat messaging sessions
- Real-time message exchange between clients
- Call initiation, signaling, acceptance, and termination
- Persistence of messages and call metadata

These flows demonstrate how frontend clients and backend services cooperate to deliver communication functionality while enforcing system rules and constraints.

6.6 Technology Stack Summary

The project utilizes a technology stack selected to support scalability, security, and cloud deployment.

At a high level, the technology stack includes:

- Mobile frontend technologies for patient and caregiver applications
- Backend services hosted within AWS
- Real-time communication mechanisms for messaging and signaling
- Relational or structured data storage for system data
- AWS Bedrock for AI-related services

Specific technology choices may evolve during implementation, but all components are selected to remain compatible with AWS hosting requirements.

6.7 Deployment Topology Highlights

The system is designed to be deployed within an AWS-hosted environment using a modular and scalable topology. Deployment considerations include:

- Separation of environments (development, test, and production)
- Secure network communication between components
- Support for scaling communication services independently
- Integration with monitoring and logging mechanisms

This deployment approach supports both project evaluation and potential future production use.

6.8 Frontend

The frontend component of the CareConnect enhancement project consists of the mobile application interfaces used by patients and caregivers to interact with the system. The frontend is responsible for presenting information to users, capturing user input, and communicating with backend services to support application functionality.

The frontend is designed to support modular development, allowing individual features such as communication, EVV, and in-home support to be developed and updated independently. This modular approach improves maintainability and reduces coupling between user interface components and backend logic.

Key responsibilities of the frontend include:

- Providing user interfaces for chat messaging and audio/video calling
- Displaying EVV and in-home residential support information
- Enforcing basic client-side validation and interaction flows
- Communicating securely with backend APIs and real-time services
- Handling notifications for incoming messages and calls

The frontend does not contain business logic related to authorization or rule enforcement. Instead, it relies on backend services to validate permissions and determine allowable actions. This separation ensures consistent enforcement of system rules and reduces the risk of unauthorized access.

Frontend development focuses on functional correctness and integration rather than full user experience redesigning. UI components are implemented to align with existing CareConnect workflows and layouts while supporting the new communication and EVV features introduced by the project.

6.9 Backend

The backend component of the CareConnect enhancement project is responsible for implementing the core business logic, enforcing system rules, and coordinating communication between frontend clients and supporting services. The backend manages authentication integration, authorization checks, data processing, and interaction with AWS-hosted services.

A key responsibility for the backend is the enforcement of **role-based communication rules**. All requests related to chat messaging and audio/video calling are validated on the backend to ensure that only authorized users can initiate or participate in communication sessions. This design prevents reliance on client-side enforcement and ensures consistent application of system rules across all users.

The backend also supports real-time communication by coordinating with AWS-compatible messaging and signaling services. It manages the creation and lifecycle of communication sessions, routes messages and signaling events, and persists in relevant metadata for audit and retrieval purposes. For audio and video calling, the backend supports call setup, participant management, and session termination workflows.

Additional backend responsibilities include:

- Integration with existing CareConnect authentication and user profile services
- Handling multi-tenant data isolation using organization identifiers
- Persisting chat messages and call session metadata
- Supporting API endpoints used by frontend components
- Integrating with monitoring and logging services for visibility and diagnostics

The backend is designed to be modular and scalable, allowing communication services to evolve independently from other system components. This approach supports incremental enhancement and aligns with cloud-based best practices for maintainability and reliability.

6.10 Database

The database component of the CareConnect enhancement project is responsible for the persistent storage of system data required to support communication features, EVV functionality, and in-home residential support. The database design supports a **multi-tenant architecture**, ensuring that data associated with one organization is logically isolated from others.

The database stores structured data such as:

- User and profile identifiers

- Patient–caregiver relationship references
- Chat message metadata and content references
- Audio and video call session metadata
- EVV and in-home support records as defined by the existing data dictionary

The database is designed to support reliable data persistence, efficient retrieval, and secure access. Sensitive data access is controlled through backend authorization logic, and database interactions are performed exclusively through backend services to maintain data integrity and consistency.

6.11 APIs

Application Programming Interfaces (APIs) provide the primary mechanism for communication between frontend clients and backend services. APIs are designed to expose necessary functionality while maintaining security, validation, and abstraction of internal system logic.

The APIs support:

- User authentication and session validation
- Retrieval and update of user and patient-related data
- Initiation and management of chat messaging sessions
- Coordination of audio and video call signaling
- Access to EVV and in-home support data

All APIs are designed to be secure, versioned, and compatible with deployment within an AWS-hosted environment. API usage is authenticated, and access is restricted based on user roles and permissions.

6.11.1 External Health and Device APIs

While the CareConnect platform acknowledges the existence of external health and device APIs (such as fitness trackers or smart home integrations), these integrations are not a primary focus of the current project. Any references to such APIs are considered contextual and are not expanded or modified as part of this enhancement effort.

6.12 AI Services

This section describes the role of Artificial Intelligence (AI) services within the CareConnect enhancement project and defines the planned migration from existing external AI providers to AWS Bedrock. The intent of this section is to establish architectural direction, governance

expectations, and project level considerations for AI usage, rather than to prescribe detailed model implementations.

AI services are treated as supporting capabilities that enhance system functionality while remaining strictly nonclinical in nature. All AI related functionality is designed to operate within the constraints of healthcare data protection, cost control, and system reliability.

6.12.1 Purpose and User Flow

AI services within the CareConnect platform are intended to support automation and intelligent processing of application data to improve operational efficiency and user experience. AI functionality is assistive and informational, operating behind the scenes to augment existing workflows rather than replacing human judgement or clinical decision making.

Examples of AI supported use cases may include, but not limited to:

- Processing or summarizing user generated content
- Supporting transcription or structured data extraction
- Enhancing search, filtering, or categorization of system data
- Providing nonclinical insights derived from application data

AI services do not provide medical advice, clinical recommendations, or autonomous decision making. All outputs generated by AI components are subject to system defined constraints, and user reviews where applicable.

6.12.2 Current State and Motivation for Migration

At the outset of the project, CareConnect relies on externally hosted AI providers accessed through third party APIs. While functional, this approach introduces several challenges relevant to a healthcare system:

- Limited control over data residency and lifecycle
- Increased complexity in enforcing consistent security and access policies
- Fragmented cost management across multiple vendors
- Dependency on external availability and service level guarantees

To address these challenges, the project includes a planned migration of AI functionality to AWS Bedrock, leveraging AWS managed foundation models and services hosted within the same cloud environment as the rest of the CareConnect platform.

This migration aligns with broader project goals related to cloud readiness, governance, and platform consolidation.

6.12.3 Target Architecture Using AWS Bedrock

In the target architecture, AI services are integrated into the CareConnect backend as managed, cloud native components using AWS Bedrock. Backend services act as the sole access point to AI capabilities, ensuring that AI usage is centrally controlled, authenticated, and audited.

Key architectural characteristics include:

- AI requests are initiated by backend services only; client applications do not interact directly with AI models
- All AI interactions occur within the AWS environment
- AI services are treated as stateless, on demand resources
- AI outputs are returned to backend services for validation before being exposed to users

This architecture supports modularity and allows AI components to evolve independently of frontend or core business logic.

6.12.4 Security, Privacy, and Data Protection

Given the sensitive nature of CareConnect data, AI services are subject to the same security and privacy requirements as other system components.

Key considerations include:

- Encryption of all data in transit between backend services and AWS Bedrock
- No long-term storage of Protected Health Information (PHI) by AI services beyond the duration of processing
- Enforcement of role-based access control (RBAC) at the backend layer
- Restriction of AI service access to authorized system components only

AI services do not retain conversational history or reuse data for training purposes within the scope of this project. Data handling practices are designed to align with HIPAA principles, even though formal regulatory certification is outside the project's scope.

6.12.5 Technical Approach

Phase 1 – Conversational AI

- Implement AI-assisted chatbot for Q&A

- Lowest complexity, highest user value

Phase 2 – Invoice Processing

- Automated invoice extraction
- Structured output review workflow

Phase 3 – Video Analysis

- Video upload and asynchronous analysis
- Subject to time and cost constraints

6.12.6 Cost Management and Operational Considerations

AI services are evaluated not only for functional capability but also for cost predictability and operational impact. AWS Bedrock provides usage-based pricing models that allow the project to monitor and control AI related expenses.

Cost management strategies include:

- Limiting AI usage to clearly defined system workflows
- Monitoring invocation frequency and data volume
- Favoring consolidated AI services over fragmented, multi-vendor usage
- Designing fallback behaviors that allow core system functionality to continue if AI services are unavailable or disabled

These strategies help ensure that AI integration remains sustainable within the constraints of an academic project and does not introduce uncontrolled operational risk.

AI services may incur variable cost based on usage. Mitigation includes phased implementation, usage limits, and evaluation of specialized services for document extraction where appropriate.

Table 2: Cost and feasibility analysis

Capability	Potential Services	Cost Risk	Notes
Chatbox	Bedrock LLM	Medium	Token based pricing
Invoice Processing	Bedrock or Textract	Mow-Medium	Textract is cheaper for structured OCR

Video Analysis	Bedrock & Media services	High	Compute & storage heavy

6.12.7 AI Migration to AWS Bedrock

As part of the CareConnect enhancement project, a dedicated effort is focused on evaluating and planning the migration of AI functionality to AWS Bedrock. This enhancement establishes a foundation for consistent, secure, and governable AI usage across the platform.

The migration effort includes:

- Identification of existing AI-dependent workflows
- Evaluation of AWS Bedrock foundation models and managed AI services for suitability
- Assessment of tradeoffs between using a single generalized model versus multiple specialized services
- Documentation of cost, scalability, and integration considerations
- Definition of system behaviors in the event of AI service unavailability

The migration is designed to be incremental and non-disruptive. Core system workflows must remain operational regardless of AI availability, ensuring that AI enhances but does not gate essential functionality.

This enhancement does not mandate final model selection or tuning. Instead, it provides architectural direction and governance boundaries that support future AI evolution within a controlled framework.

6.12.8 Governance, Compliance, and Non-Clinical Boundaries

AI usage within CareConnect is governed by strict boundaries to prevent misuse or overreach. These boundaries are enforced through architectural design and backend validation rather than relying on user discretion.

Key governance principles include:

- AI outputs are informational only and do not constitute clinical guidance
- AI services do not initiate actions autonomously
- AI generated content may be reviewed, filtered, or suppressed by backend logic
- All AI usage is subject to logging and auditability

While formal regulatory audits are outside the scope of this project, the architecture is designed to support future compliance efforts by maintaining transparency, traceability, and controlled access.

6.12.9 Summary and Future Enhancements

This project establishes a robust foundation for AI integration within the CareConnect platform by migrating toward AWS Bedrock and defining clear architectural and governance principles.

Future enhancements may include:

- Expansion of AI supported workflows based on demonstrated value
- Optimization of model selection and cost efficiency
- Enhanced monitoring and analytics for AI usage
- Refinement of AI driven features based on user feedback

By addressing AI migration at the architectural planning level, the project ensures that CareConnect is positioned for scalable, secure, and responsible for AI adoption beyond the scope of the current academic term.

6.13 Hosting

The CareConnect enhancement project is deployed within an **AWS-hosted environment**, leveraging cloud infrastructure to support scalability, reliability, and security.

6.13.1 Cloud Deployment Model

The system uses a cloud-based deployment model that supports flexible resource allocation and environmental separation. Development, testing, and production environments are logically separated to support safe validation and deployment.

6.13.2 Containerization and Orchestration

Where appropriate, containerization may be used to package backend services for consistent deployment. Orchestration mechanisms are considered to support scaling and service management.

6.13.3 Serverless-First and Scale-to-Zero

Serverless services are evaluated for components that benefit from event-driven execution and cost efficiency. This approach supports scaling based on demand while minimizing idle resource usage.

6.13.4 Security and Reliability

Hosting configurations include network security controls, access management, and monitoring to ensure system reliability and protection against unauthorized access.

6.14 Security

Security is a core consideration across all aspects of the CareConnect enhancement project. The system architecture incorporates security measures at multiple layers to protect sensitive data and ensure proper access control.

Key security considerations include:

- Encryption of data in transit
- Role-based access control enforced by backend services
- Secure API endpoints with authentication and authorization
- Infrastructure-level security controls within AWS
- Monitoring and logging to detect and respond to issues

The security approach is designed to support both project evaluation and potential future production use.

7. Project Deliverables

This section describes the primary deliverables produced as part of the CareConnect enhancement project. Deliverables represent tangible outputs that demonstrate progress, support validation, and enable evaluation of project success.

Throughout the project lifecycle, deliverables are produced incrementally and aligned with the defined milestones. These deliverables include planning artifacts, design documentation, implementation of outputs, testing materials, and user-facing documentation. Collectively, they

provide evidence that the system requirements have been addressed and that the project objectives have been met.

Key deliverables produced during the project include formal documentation such as the Project Plan and Software Requirements Specification, technical and architectural documentation supporting system implementation, testing artifacts that demonstrate verification and validation activities, and user-oriented materials that support system usage. Final deliverables reflect a complete, testable system suitable for academic evaluation and demonstration.

8. Risk & Mitigation

This section identifies potential risks that could impact the successful completion of the CareConnect enhancement project. Each risk is evaluated based on its likelihood of occurrence and potential impact on the project. Mitigation strategies are defined to reduce the probability of occurrence or minimize the impact should the risk materialize.

- High: Scope creep → Mitigation: Strict change control.
- Medium: Cloud costs exceed budget → Mitigation: Monitor AWS billing weekly.
- Low: Team member bandwidth issues → Mitigation: Assign backup reviewer.

8.1 Risk Probability Matrix

The following matrix categorizes identified project risks according to their probability and impact levels.

Table 2 - Risk Probability matrix

Impact ↓ / Probability →	**Low**	**Medium**	**High**
High Impact	Security breach	Communication system failure, Backend service outage	Cloud cost overrun, Schedule slippage
Medium Impact	AI processing errors	File upload failures, Accessibility noncompliance, API incompatibility	Feature integration delays, Team workload imbalance
Low Impact	UI cosmetic defects	Documentation delays, Tooling learning curve	Minor test failures

This matrix provides a high-level view of risk exposure and helps prioritize mitigation efforts.

8.2 Risk Identification, Probability, and Mitigation

Table 3- Risk probability and mitigation table

Risk ID	Risk Description	Probability	Impact	Mitigation Strategy
R1	Real-time chat or call failures due to network instability	Medium	High	Use WebSockets for signaling and messaging, WebRTC with reconnect logic, limit MVP to 1:1 calls
R2	Backend service outage	Medium	High	Deploy using containerized services, enable health checks and monitoring
R3	Database connection or schema failures	Medium	High	Use managed DB, version schemas, implement backups, test migrations
R4	Sensitive data exposure (security breach)	Low	High	Enforce HTTPS, restrict roles, avoid logging PHI, encrypt secrets
R5	Cloud service costs exceed budget	High	High	Cap media quality, restrict session length, use S3 lifecycle rules, monitor usage
R6	File uploads fail or become too expensive	Medium	Medium	Use S3 for attachments, limit file size, compress images
R7	Frontend and backend APIs become incompatible	Medium	Medium	Define API contracts, version endpoints, mock APIs
R8	AI or automation features produce incorrect results	Low	Medium	Treat AI as assistive, allow manual override, limit usage
R9	Accessibility issues reduce usability	Medium	Medium	Apply WCAG 2.1 AA, reduce text density, use iconography
R10	Device/browser incompatibility	Medium	Medium	Test on multiple platforms, use responsive layouts
R11	Uneven team workload distribution	High	Medium	Assign clear roles, track tasks, rotate responsibilities
R12	Knowledge silos form around key components	Medium	Medium	Require documentation, conduct walkthroughs
R13	Schedule delays due to academic workload	High	High	Break work into milestones, add buffer time, prioritize core features
R14	Git merge conflicts disrupt development	Medium	Medium	Use feature branches, enforce pull requests

Risk ID	Risk Description	Probability	Impact	Mitigation Strategy
R15	Insufficient test coverage leads to undetected defects	Medium	Medium	Write unit tests, automate builds
R16	Environment configuration inconsistencies	Medium	Medium	Use environment variables, provide setup docs
R17	External service dependency outage (email, cloud API)	Low	Medium	Abstract services, provide fallback modes

9. Budget Estimate

This section provides a high-level estimate of the costs associated with the CareConnect enhancement project. The budget is intended for **planning and academic evaluation purposes only** and does not represent a finalized or contractual cost commitment. Due to the exploratory nature of the project and the use of managed cloud services, cost estimates are based on reasonable assumptions rather than precise usage data.

9.1 Labor Cost

Labor costs represent the estimated effort required to complete the project activities over the twelve-week timeline. As this is an academic project, labor costs are estimated using notional hourly rates to reflect industry-standard roles and responsibilities.

Estimated labor effort includes:

- Requirements analysis and documentation
- Architectural design
- Frontend and backend development
- Testing and validation
- Documentation and project management activities

Table 4- Labor cost

Role	Estimated Hours	Notional Rate ($/hr)	Estimated Cost ($)
Project Lead / Developer	50	64.00	3200.00
UX Design	40	46.00	1840.00

Role	Estimated Hours	Notional Rate ($/hr)	Estimated Cost ($)
Frontend Development	150	46.00	6900.00
Backend Development	150	64.00	9600.00
Testing & Documentation	150	52.00	7800.00
Project Management	60	48.00	2880.00
Totals	600		32220.00

Note: Rates are illustrative and used solely for project estimation purposes.

9.2 Materials Cost

The project does not require significant physical materials. Development is performed using personal computing equipment and commonly available software tools.

Table 5-Material cost summary

Item	Estimated Cost ($)
Development Tools & IDEs	0 (Free tier licenses)
Documentation Tools	0 (Free tier licenses)
Total Materials Cost	**0**

9.3 Infrastructure Cost

Infrastructure costs are primarily associated with the use of AWS-hosted services to support backend processing, real-time communication, data storage, and AI services.

9.3.1 AWS Core Services

AWS services are used on a limited scale, suitable for development and testing environments. Costs are estimated conservatively based on low usage assumptions.

Table 6-AWS core service cost

Service	Purpose	Estimated Monthly Cost
API Gateway WebSocket	Chat + call signaling	$1 – $15
ECS Fargate	Chat + signaling backend	$15 – $60

Amazon S3	Attachments + recordings	$1 – $20
WebRTC (client-side)	Audio/video media	$0
AWS Textract	Invoice OCR	$15 – $150
AWS Transcribe	Audio transcription	$10 – $150
AWS Bedrock (on-demand)	Chatbot + AI	$10 – $100
CloudWatch	Logs & metrics	$1 – $10
Estimated Monthly Total (MVP)		**$50 – $500**

9.3.2 Third-Party / One-Time Fees

The project does not include significant third-party licensing fees. All tools and services used are either open-source or covered under existing academic or free-tier usage.

Table 7-Third party licensing fees

Item	Estimated Cost ($)
Third-Party Services	0
Total Third-Party Cost	0

9.3.3 Total Budget for MVP Phase (12 Weeks)

Table 8- Total budget breakdown

Cost Category	Estimated Cost ($)
Labor	32220.00
Materials	0.0 (free tier licenses)
Infrastructure	50 - 500.00/month
Total Estimated Budget	32370.00 - 33720.00

9.3.4 Cost-Control and Optimization Plan

To manage costs effectively, the project adopts the following strategies:

- Use of AWS free-tier and low-cost service options, where available
- Limiting infrastructure usage to development and testing needs
- Monitoring service usage and disabling unused resources

- Favoring serverless or scale-to-zero services where appropriate

These measures help ensure that infrastructure costs remain predictable and controlled.

9.3.5 Alignment to Financial Policies

All budget estimates align with academic project guidelines and emphasize responsible resource usage. No costs are incurred without prior review and justification within the project context.

9.3.6 Budget Summary for Project Plan

The overall project budget reflects a conservative estimate that balances realism with academic constraints. Labor represents the majority of project cost due to the emphasis on design, development, and documentation. Infrastructure costs are kept intentionally low by leveraging managed cloud services and development-scale usage patterns.

10. Roles & Responsibilities

This section defines the roles involved in the CareConnect enhancement project and clarifies responsibility and accountability for key project activities. Clearly defining roles and responsibilities supports effective coordination, accountability, and project execution.

10.1 Roles

The following table describes the primary roles involved in the project and their associated responsibilities.

Table 91- Roles & Responsibilities Table with Assigned Team Members

Role	Description	Key Responsibilities
Project Lead	Oversees overall project execution and coordination	Project planning, milestone tracking, stakeholder communication, final deliverable review
Backend Developer	Responsible for backend system design and implementation	Business logic implementation, role-based authorization, API development, AWS service integration

Role	Description	Key Responsibilities
Frontend Developer	Responsible for mobile application interfaces and client-side integration	UI development, integration with backend APIs, communication feature interfaces
QA / Tester	Responsible for validation and verification activities	Test planning, test case execution, defect tracking, test reporting
Documentation Lead	Responsible for project documentation artifacts	SRS, design documents, user guides, deployment and operations documentation

10.2 RACI Matrix

The RACI matrix below defines who is **Responsible (R)**, **Accountable (A)**, **Consulted (C)**, and **Informed (I)** for major project activities.

Table 10-RAI Matrix table

Key:

- R – Responsible
- A – Accountable
- C – Consulted
- I - Informed

Milestone / Activity	Project Lead	Backend Developer	Frontend Developer	QA / Tester	Documentation Lead
Milestone One					
Project Planning	A	I	I	I	C
Requirements Analysis	A	C	C	I	R
Software Requirements Specification (SRS)	C	I	I	I	R
Milestone Two					
Technical Architecture Design	A	R	C	I	C
Communication Feature Design	A	R	C	I	C
Software Test Plan Development	C	I	I	R	C

Milestone / Activity	Project Lead	Backend Developer	Frontend Developer	QA / Tester	Documentation Lead
Checkpoint					
Progress Review	A	R	R	C	I
Risk and Issue Assessment	A	C	C	R	I
Plan Adjustments	A	C	C	I	I
Milestone Three					
Backend Implementation	I	R	C	I	I
Frontend Implementation	I	C	R	I	I
AWS Integration	C	R	C	I	I
Programmer Guide Development	I	C	C	I	R
Deployment and Operations Guide	I	C	C	I	R
Milestone Four					
System Testing and Validation	I	C	C	R	I
Test Report Preparation	I	I	I	R	C
User Guide Development	I	I	I	C	R
Final Review and Submission	A	C	C	C	R

Note. RACI Matrix created with reference from previous 2025 SWEN 670 CareConnect project (CareConnect, 2025).

11. Monitoring and Reporting

This section describes how project progress, risks, and issues are monitored and communicated throughout the project lifecycle. Effective monitoring and reporting ensure transparency, accountability, and early identification of potential problems.

11.1 Intra-Team Monitoring and Reporting

Project progress is monitored through regular internal check-ins among team members. These check-ins focus on reviewing completed tasks, identifying blockers, and confirming alignment

with milestone objectives. Progress is evaluated against the project timeline and scope defined in earlier sections.

Status updates are documented informally and used to track completion of planned activities. Any deviations from the schedule or scope are discussed and addressed promptly to minimize impact on overall project delivery.

11.2 Project Communications

Project communications are conducted through scheduled meetings, written updates, and shared documentation repositories. Key communications include milestone reviews, checkpoint discussions, and clarification of requirements or design decisions. Communication with stakeholders ensures that expectations remain aligned, and that feedback is incorporated as appropriate.

12. Assumptions

This section lists assumptions made during project planning that influence scope, schedule, and execution. These assumptions are considered valid unless explicitly changed during the project.

- Team members have consistent access to required development tools and environments.
- AWS services required for development and testing are available and accessible.
- Users involved in testing have compatible devices and network connectivity.
- Requirements remain stable after formal approval, except through defined change management procedures.
- The project is executed within the constraints of an academic semester.

13. Constraints

This section identifies constraints that limit or influence how the project is executed. Constraints are conditions that cannot be changed without significant impact on the project.

- The project must be completed within a twelve-week academic term.
- The system must be deployable within an AWS-hosted environment.
- Development effort is limited by academic workload and availability.
- The project scope is restricted to assigned enhancements and excludes full platform redesign.
- Budget estimates are constrained to development-scale usage and academic assumptions.

14. Change Management Procedures

This section defines the formal process used to manage changes to project scope, requirements, schedule, or technical approach throughout the CareConnect enhancement project. The purpose of change management is to ensure that all proposed changes are evaluated systematically, approved appropriately, and implemented in a controlled manner that minimizes disruption to project objectives and timelines.

Given the fixed duration of an academic semester and the technical complexity of the system, disciplined change control is essential to prevent scope creep and maintain delivery feasibility.

14.1 Change Identification and Initiation

A change request may be initiated by any project stakeholder, including team members, the product owner, or the course instructor. Changes may arise due to clarification of requirements, technical constraints, integration challenges, or feedback received during milestone reviews.

Each change request must clearly describe:

- The nature of the proposed change
- The reason for the change
- The affected project artifacts (requirements, design, schedule, or deliverables)

All change requests are documented before any analysis or implementation of work.

14.2 Impact Analysis

Upon submission, each change request undergoes an impact analysis conducted by the project team and led by the Project Lead. The analysis evaluates the potential effects of the proposed change across the following areas:

- Project scope and alignment with stated objectives
- Schedule and milestone impact
- Resource availability and workload distribution
- Technical complexity and architectural implications
- Risk exposure, including cost and performance considerations

If a change introduces significant scope expansion or jeopardizes milestone commitments, alternative solutions or phased approaches are considered.

14.3 Review and Approval

Following impact analysis, the change request is reviewed by the Project Lead in consultation with relevant team members. Approval authority depends on the nature of the change:

- Minor changes (documentation updates, clarifications, or low-impact technical adjustments) may be approved directly by the Project Lead.
- Major changes (scope expansion, architectural shifts, or schedule modifications) require agreement from the full project team and alignment with academic requirements.

Unapproved changes are deferred or rejected to preserve project feasibility.

15.4 Implementation and Documentation

Approved changes are incorporated into the project in a controlled manner. Relevant project artifacts such as the Project Plan, Software Requirements Specification, architectural documentation, or test plans are updated to reflect the approved modification.

Implementation follows standard development and validation practices, including testing and documentation updates where applicable. All implemented changes are traceable to their original change request.

Project Name	CareConnect
Team	A
Date of Change Request Submitted	
Requested By	
Change Description	
Change Reason	
Impact of Change	
Proposed Action	
Status (Submitted, Approved, Rejected)	
Decision Reason	
Approval Date	
Approved By	

Note. CareConnect Change Request Form Template created with reference from the previous 2025 SWEN 670 CareConnect project (CareConnect, 2025).

All change requests are recorded using a standardized Change Request Form to ensure traceability and accountability.

Maintaining formal change records supports transparency and provides an audit trail for project evaluation.

16. Appendices

This section provides supporting information referenced throughout the project plan. The appendices include a glossary of terms used in the document and a list of references cited or consulted during project planning and design.

16.1 Glossary of Terms

The following table defines key terms and acronyms used throughout this project plan to ensure consistent understanding.

Term / Acronym	Definition
API (Application Programming Interface)	A set of rules and protocols that allows software components to communicate with one another.
AWS (Amazon Web Services)	A cloud computing platform that provides infrastructure, platform, and software services.
AWS Bedrock	An AWS service that provides access to foundation models for building AI-powered applications.
Backend	The server-side portion of the application responsible for business logic, data processing, and integrations.
Chat Messaging	Real-time exchange of messages between users within the CareConnect application.
Communication Module	The system component responsible for chat messaging and audio/video calling functionality.
EVV (Electronic Visit Verification)	A system used to electronically document the time, location, and services provided during in-home care visits.
Frontend	The client-side portion of the application that users interact with directly.
Gantt Chart	A visual representation of a project schedule showing tasks over time.

Term / Acronym	Definition
Milestone	A significant point or event in the project timeline marking completion of a major phase.
Multi-Tenant Architecture	A system design in which multiple organizations share the same infrastructure while keeping data logically isolated.
RACI Matrix	A responsibility assignment model identifying who is Responsible, Accountable, Consulted, and Informed.
Real-Time Communication	Data exchange that occurs with minimal delay, enabling live interaction between users.
SRS (Software Requirements Specification)	A formal document that defines functional and non-functional system requirements.
WebRTC	A technology that enables real-time audio, video, and data communication between browsers or applications.
WebSocket	A communication protocol that enables persistent, bidirectional communication between a client and server.
WBS (Work Breakdown Structure)	A hierarchical decomposition of project work into manageable components.

16.2 References

The following references were consulted during the planning and design of the CareConnect enhancement project. References are formatted according to **APA (7th edition)** guidelines.

Amazon Web Services. (2024). *Amazon API Gateway documentation*. https://docs.aws.amazon.com/apigateway/

Amazon Web Services. (2024). *AWS Bedrock documentation*. https://docs.aws.amazon.com/bedrock/

Amazon Web Services. (2024). *Working with WebSocket APIs*. https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-websocket-api-overview.html

Amazon Web Services. (2024). *Amazon Connect Web and mobile SDK documentation*. https://docs.aws.amazon.com/connect/

CareConnect. (2025). *CareConnect Project Plan Document (Version 3.0)* [Project documentation]. University of Maryland Global Campus.

GeeksforGeeks. (2025, March 10). *What is Amazon Bedrock?* GeeksforGeeks. https://www.geeksforgeeks.org/cloud-computing/amazon-bedrock-aws-bedrock/

IEEE Computer Society. (2018). *ISO/IEC/IEEE 29148:2018 — Systems and software engineering—Life cycle processes—Requirements engineering*. https://standards.ieee.org/

Project Management Institute. (2021). *A guide to the project management body of knowledge (PMBOK® Guide)* (7th ed.). PMI.

Wang, B. (2026, January 21). *Amazon Bedrock Review (2026): Is It Production Ready?* Truefoundry.com; Truefoundry. https://www.truefoundry.com/blog/our-honest-review-of-amazon-bedrock-2026-edition

W3C. (2023). *WebRTC 1.0: Real-time communication between browsers*. https://www.w3.org/TR/webrtc/

Project Plan Document

CareConnect Team B

Version 2.1

University of Maryland Global Campus

SWEN 670 - Software Engineering Capstone

Dr. Mir Assadullah

January 24, 2026

Contributors: Eduardo Estrada, Gary Jurado, Yismaw Tilaye, Joseph Wojcik, Chastity Sapp

Table of Contents

Revision History *3*
1. Executive Summary *5*
1.2 Purpose **5**
1.3 Scope **5**
2. Objectives and Goals *6*
2.1 Goals **6**
2.2 Objectives **6**
3. Scope of Work *6*
3.1 In-Scope **7**
3.2 Out-of-Scope **8**
4. Stakeholders *9*
4.1 Internal Stakeholders **9**
4.2 External Stakeholders **9**
5. Timeline & Milestones *11*
5.2 Project Schedule **11**
5.4 Planning **13**
5.5 Definition of Done **13**
6. Technical Architecture *13*
6.1 Architectural Objectives **14**
6.2 Logical System Architecture **14**
6.3 High-Level Architecture Diagram **16**
6.4 Architectural Rationale **17**
6.5 Runtime Flows **19**
6.6 Technology Stack Summary **19**
6.7 Deployment Topology Highlights **20**
6.8 Frontend **20**
6.9 Backend **21**
6.10 Database **22**
6.11 APIs **23**
6.12 AI Services **25**
6.13 Hosting **25**
6.14 Security **26**
7. Project Deliverables *27*

8. Risk & Mitigation *28*
8.1 Risk Probability Matrix **28**
8.2 Risk Identification, Probability, and Mitigation **29**
8.3 External API Risk Register **30**
9. Budget Estimate *30*
9.1 Labor / Effort Estimate **31**
9.2 Materials Cost **32**
9.3 Infrastructure Cost **32**
10. Roles & Responsibilities *37*
10.1 Team Roles **37**
10.2 RACI Matrix **37**
11. Monitoring and Reporting *39*
11.1 Objectives **39**
11.2 Intra-Team Monitoring and Reporting **40**
11.3 Project Communications **40**
11.4 Quality Plan **41**
12. Assumptions *41*
12.1 Technical and Operational Assumptions **41**
13. Constraints *42*
13.1 Technical and Project Constraints **42**
14. Work Breakdown Structure (WBS) *43*
15. Change Management Procedures *43*
15.1 Initiation **44**
15.2 Review (Change Control Board) **44**
15.3 Authorization & implementation **44**
15.4 Change Request Form **45**
15.5 UI/UX wireframe preview **45**
16. Appendices *45*
16.1 Glossary and Terms **46**
16.2 References **48**

Revision History

Team Name	Date	Reason for Changes	Version
CareConnect Team B	1/24/2026	Initial document submission.	1.0
CareConnect Team B	2/7/2026	Revisions based on 1.0 feedback	2.0
CareConnect Team B	2/20/2026	Updates based on requirement updates and system architecture updates	2.1

1. Executive Summary

CareConnect is a mobile-first, cloud-native application designed to bridge patients and caregivers through scheduling, health tracking, and secure communication. This Project Management Plan (PMP) defines Team B's delivery scope for the capstone phase, and documents the implementation of three required technical capabilities that elevate the existing CareConnect baseline into a production-ready MVP. This Project Management Plan (PMP) outlines team B's phase of development, expanding upon the functional baseline and architectural requirements established in the CareConnect Project Plan v3.0 [4, 9].

Team B will deliver:

- **Offline-First Robustness (BNS 5):** Implement an offline persistence and synchronization framework, then fully enable it for two MVP modules to validate reliability in low-connectivity conditions while keeping scope feasible alongside codebase normalization.
- **Infrastructure as Code (BNS 6):** Provision the backend environment using AWS CloudFormation, ensuring repeatable, version-controlled deployments with no configuration drift.
- **Advanced Observability (BNS 7):** Implement privacy-preserving telemetry, centralized logging, and distributed tracing to validate feature usage and system health while scrubbing PII and PHI at the point of capture.

This PMP expands upon prior CareConnect planning artifacts and updates the delivery approach to reflect the current Team B technical design and architecture.

1.2 Purpose

The purpose of CareConnect is to engineer a secure, HIPAA-eligible mobile application that unifies the caregiver and patient experience. The platform provides a centralized interface for care-plan adherence, symptom and wellness logging, and operational monitoring. The system is engineered to remain reliable during intermittent connectivity and to support operational excellence through privacy-safe observability.

1.3 Scope

CareConnect facilitates a collaborative care environment through a bidirectional mobile ecosystem.

For caregivers, the platform provides tools for patient profile management, care task scheduling, and oversight of patient-reported data. For patients, the application supports structured wellness logging, medication adherence tracking, and care-related entries through a mobile-first interface.

The platform is deployed on AWS, and is designed to:

- Support offline operation for two validated modules through encrypted local persistence and sync-on-reconnect mechanics.
- Use CloudFormation-managed infrastructure as the only source of truth for environments.
- Capture logs, traces, and feature-usage telemetry using a privacy-first approach that prevents PII and PHI from entering observability streams.

2. Objectives and Goals

2.1 Goals

The primary goal of this phase is to evolve CareConnect into a resilient, production-ready MVP by implementing Team B's required technical capabilities while preserving the functional baseline. This phase prioritizes reliability, reproducibility, and auditability across the mobile client and backend. Building upon the requirements established in previous development [4, 9], this development phase focuses on the following strategic goals:

2.2 Objectives

The objectives for this project phase are to execute the technical modernization required to meet production-grade benchmarks and satisfy BNS compliance mandates.

- **Care Coordination Foundation:** Deliver a shared patient and caregiver experience that supports scheduling, reminders, and high-priority alerting, with a target of 99.9% service availability through cloud-native deployment and operational monitoring.
- **Offline-First Data Integrity (BNS 5):** Deploy an encrypted local data persistence layer and synchronization workflow that ensures users can create and retain records while offline, and reliably reconcile changes when connectivity returns. The offline framework must be extensible, but full synchronization is implemented only for two modules in the current MVP.
- **Infrastructure Automation (BNS 6):** Provision 100% of the backend environment using AWS CloudFormation. The deployment must be reproducible, version-controlled, and verified as drift-free in a sandbox environment.
- **Mechanical Telemetry Capture (BNS 7):** Integrate centralized logging, distributed tracing, and anonymous feature telemetry to validate performance and feature usage. Implement a privacy middleware layer that scrubs PII and PHI before any telemetry is emitted or stored.

3. Scope of Work

The scope of work for CareConnect team B focuses on migrating legacy infrastructure [4, 9] and implementing new architectural capabilities. All deliverables defined below are developed to a production-ready standard, as governed by the quality requirements in the Definition of Done (Section 5.5). To ensure project feasibility within a 12-week cycle, the MVP boundary is strictly maintained by focusing these production standards on the following specific functional areas:

3.1 In-Scope

3.1.1 Core User & Profile Management [4, 9]

- The system shall allow caregivers to register, authenticate, and manage patient profiles.
- The system shall support Role-Based Access Control (RBAC) to ensure data security between caregiver and patient views.

3.1.2 Essential Care Modules [4, 9]

- **Tasking & Medication:** Caregivers shall assign daily tasks (medication/meal reminders); patients shall mark tasks as complete.
- **Health Data Logging:** Manual recording of vital signs and health data with report generation.
- **Calendar:** A shared calendar for manual entry of events, tasks, and reminders.

3.1.3 Offline-First Data Integrity (BNS 5)

- The system shall implement a data persistence abstraction layer to enable offline functionality for two prioritized high-value modules, selected based on implementation complexity and risk reduction goals.
- The offline framework shall be architected to support future extension to additional modules, but only two modules will be fully synchronized within the current MVP timeline.
- The system shall implement a sync-on-reconnect mechanism to ensure data consistency with the cloud-hosted backend.
- The system shall provide a Manual Offline Toggle within the application settings, allowing users to explicitly enable or disable offline mode.

Note: Offline synchronization scope was intentionally reduced from five modules to two in response to instructor guidance prioritizing the "Start Normalization of Codebase" requirement. This ensures delivery of a stable framework without compromising architectural integrity.

3.1.4 Infrastructure Automation (BNS 6)

- The system shall transition infrastructure provisioning to an AWS-native Infrastructure-as-Code (IaC) framework.
- The system shall use automated templates to standardize workflows, ensuring reproducible and version-controlled deployments.
- The migration shall preserve all existing networking, compute, storage, and security configurations established in legacy documentation [4, 9], while aligning deployment to the current Team B runtime architecture (for example, ECS/Fargate-based services where applicable).

3.1.5 Observability & Privacy-Preserving Analytics (BNS 7)

- **Operational Monitoring:** The system shall implement distributed tracing and centralized logging to enable rapid troubleshooting and performance monitoring.
- **Anonymous Feature Analytics:** The system shall implement a mechanical capture mechanism to track feature usage patterns.
- **Privacy Constraint:** Analytics collection shall be strictly anonymous; a middleware layer shall be implemented to ensure PII/PHI is not collected for telemetry.

3.1.6 Engineering Continuity

- The system shall include documented rationales for the architectural designs to support future scalability and knowledge transfer.

3.1.7 Start Normalization of Codebase (Graded Requirement)

- Team B will initiate normalization of the CareConnect codebase to establish a stable and maintainable foundation. This includes standardizing API pathing, reducing duplicate controllers and DTOs, consolidating project structure, and aligning shared frameworks across teams. This effort is intentionally scope-limited and does not include full refactoring of all legacy modules.

3.2 Out-of-Scope

The following items are explicitly excluded from the current CareConnect development cycle to ensure project feasibility, minimize third-party integration risks, and focus labor on team B's core technical requirements (BNS 5, 6, and 7).

3.2.1 Clinical Diagnosis & Decision Support [4, 9]

- The system shall not offer medical diagnoses, prescribe treatments, or provide clinical decision support. All AI responses remain informational.

3.2.2 Accessibility & Language Enhancements [4, 9]

- The application shall not include specialized accessibility features such as voice control.
- The system shall not support American Sign Language (ASL) translation.

3.2.3 Home Automation & Environmental Control [4, 9]

- The system shall not link to or monitor smart home devices, cameras, or home sensors (e.g., motion detectors, smart lighting, or climate control).

3.2.4 Custom Hardware & Wearables Integration [4, 9]

- The project shall not involve the development of new wearables or proprietary medical hardware.
- Direct integration with Fitbit, Apple HealthKit, and Google Health Connect is excluded.

3.2.5 Financial, Billing, & Mail Management [4, 9]

- The system shall not include tiered subscription billing or external bill assistant features (e.g., invoice scanning).
- Integration with USPS Informed Delivery or any mail-digitization services is excluded.

3.2.6 Full Offline Application Parity (BNS 5)

- The application shall not provide full offline functionality for all features. Offline synchronization is strictly limited to two validated modules in the current MVP. Expansion to additional modules is deferred to future development phases once codebase normalization is complete.

3.2.7 Non-AWS or Multi-Cloud Environments

- Infrastructure-as-Code (IaC) is restricted to AWS-native tools; support for Terraform or multi-cloud providers is excluded.

3.2.8 Advanced Business Intelligence

- The CareConnect application shall not include predictive analytics or user behavior modeling based on telemetry data. Telemetry is used strictly for operational health and anonymous feature usage counting.

3.2.9 AI Feature Enhancement

- The system shall not include new prompt engineering, model fine-tuning, or natural language processing (NLP) development.
- Development is strictly limited to the infrastructure migration to AWS Bedrock.

4. Stakeholders

4.1 Internal Stakeholders

The Internal Stakeholders involved in this project are listed below. The details about the roles and responsibilities of the internal stakeholders are outlined in Section 10 of this document.

- Client (Roy Gordon)
- Faculty/Mentor (Dr. Assadullah)
- Project Lead (Colyn Mahoney)
- Team B Technical Lead/Architect (Eduardo Estrada)
- Team B Developers, BA, QA, UI/UX (Gary Jurado, Yismaw Tilaye)
- Team B DevOps / Infrastructure Engineer (Chasity Sapp, Joseph Wojcik)

4.2 External Stakeholders

The following are the key external stakeholders for the CareConnect project:

- **Patients:** End users of the application who will manage their health tasks, receive reminders, and request emergency help via the platform.
- **Caregivers:** Individuals, whether professional or family-based, will utilize the platform to manage patient schedules, tasks, and wellness data.
- **Health Professionals:** Licensed providers such as nurses or case managers who may use CareConnect for remote monitoring and coordination of care tasks.
- **Family Members:** Extended family who support the care of patients and may be secondary users of the system to stay informed or provide supplemental care.
- **Clients/Mentors:** Advisors providing project development feedback and insight into real-world healthcare technology integration.
- **University Faculty (Professor):** Faculty overseeing and evaluating the capstone project deliverables and adherence to software engineering best practices.
- **Platform Providers:** External service providers including Apple App Store and Google Play Store that distribute the CareConnect application. Third-party service providers are limited to AWS-managed services defined in the system architecture.

5. Timeline & Milestones

5.1 Timeline and Milestone Table

Table 5.1.1

Milestone and Timeline Table

Milestone	Deliverables Due	Due Date
Milestone 1	• Project Plan • Software Requirements Specification	January 24, 2026
Milestone 2	• Technical Design Document • Software Test Plan	February 7, 2026
Checkpoint	• PowerPoint of Progress	February 27, 2026
Milestone 3	• Programmer Guide • Deployment and Operations Guide • Software Test Plan	March 10, 2026
Milestone 4	• User Guide • Test Report	March 31, 2026

5.2 Project Schedule

This WBS outlines the tasks required to expand the CareConnect functional baseline [4, 9] into the Team B technical framework. It balances parallel workstreams to ensure infrastructure (BNS 6), offline logic (BNS 5), and telemetry (BNS 7) mechanics are delivered within the 12-week window.

Table 5.2.1

Task-Level Work Breakdown Structure (WBS)

Task ID	Task Description	Wk	Milestone	Est. Hours	Deliverable
1.0	**Phase 1: Planning & Design**	**1-5**	**M1 / M2**	**300**	**PMP, SRS, TDD, STP**
1.1	PMP & SRS Maintenance	1-5	M1	100	Approved Governance Baseline.
1.2	Technical Design (TDD)	2-5	M2	120	BNS 5, 6, 7 Architecture. Includes architectural updates required to support codebase normalization and shared framework alignment.

1.3	**Software Test Plan (STP)**	2-5	M2	80	Test Case Definitions.
2.0	**Phase 2: Infrastructure (BNS 6)**	**4-7**	**Checkpt**	**220**	**Environment Live**
2.1	CloudFormation & VPC	4-7	Checkpt	110	Network Stack Automation.
2.2	Data Tier & IAM Hardening	4-7	Checkpt	110	RDS/Security Audit.
3.0	**Phase 3: Core Logic (BNS 5/7)**	**5-9**	**M3**	**320**	**Verified BNS Mechanics. Includes normalization of shared synchronization, telemetry, and API structures.**
3.1	Offline Persistence	5-9	M3	120	Encrypted local persistence framework, implemented for two MVP modules (Mood & Symptom Tracking, Manual Calendar).
3.2	Sync Engine & Redaction	5-9	M3	150	Sync-on-reconnect engine, conflict handling, and PII/PHI-safe telemetry redaction, validated on two MVP modules.
3.3	Mechanical Telemetry	5-9	M3	50	Telemetry hooks and tracing for sync performance and feature events, with privacy scrubbing enforced.
4.0	**Phase 4: Engineering Guides**	**5-9**	**M3**	**180**	**Continuity Suite**
4.1	Programmer Guide	5-9	M3	90	Repo & API Specs.
4.2	Deployment & Ops Guide	5-9	M3	90	SOPs & OpenTelemetry Exporter Setup (deployment-configurable backends).

5.0	Phase 5: Final Validation	9-12	M4	180	Handover Package
5.1	End-to-End Test & Report	9-12	M4	90	**Final Test Report**.
5.2	User Guide & Onboarding	9-12	M4	50	User Manuals.
5.3	Final Demo	12	M4	40	Final Delivery.
TOTAL				**1,200**	

5.4 Planning

- **Methodology:** Agile-Scrum with 2-week sprints; weekly retrospectives.
- **Tools:** GitHub (repo & CI/CD), Teams (collaboration), SharePoint (document management).
- **Schedule Baseline:** 12-week academic timeline

5.5 Definition of Done

To ensure all work packages meet the project's quality and security standards, a task is considered "Done" only when it satisfies the following criteria:

Table 5.5.1

Completion Definitions

Category	Requirement for Completion
Code & Security	Code is merged into the main branch, passed peer review, and verified to contain zero PII in telemetry/logs.
Infrastructure	CloudFormation templates are validated, deployed to a sandbox environment, and verified as "drift-free."
Testing	Unit tests for the specific logic (e.g., sync triggers) have passed, and the feature is verified in an offline state.
Documentation	The Programmer's Guide and README are updated to reflect any changes in logic, schema, or deployment.
Final Approval	The feature or infrastructure change has been demonstrated in the weekly sync and "signed off" by the Project Lead for inclusion in the final demo.

6. Technical Architecture

The purpose of this section is to translate the functional requirements defined in the legacy documentation [4, 9] into a modernized, cloud-native architecture. This design modernizes CareConnect into an AWS-native, container-based architecture using stateless Spring Boot services deployed on AWS ECS Fargate. By leveraging a decoupled, event-driven approach, the system improves resilience through Offline-First data integrity (BNS 5) while maintaining predictable runtime behavior suitable for synchronization and auditing workflows. Infrastructure is managed exclusively via AWS CloudFormation (BNS 6) to ensure reproducible deployments,

while a privacy-preserving Telemetry Layer (BNS 7) provides operational observability without compromising PII/PHI.

6.1 Architectural Objectives

The following objectives prioritize the "Team B" technical requirements while maintaining the functional core inherited from the previous design phase [4, 9].

Table 6.1.1

Architectural Objectives

Objective	Rationale
Resilience (BNS 5)	The system must maintain 100% data integrity for core health logs during network-constrained states using localized persistence.
Operational Efficiency (BNS 6)	Managed AWS-native services (ECS Fargate, API Gateway, Cognito) are utilized to reduce server maintenance and minimize "Ops Toil."
Regulatory Compliance	HIPAA/GDPR standards drive the design for encryption-at-rest, automated PII scrubbing in logs, and isolated VPC architecture.
Cost-Efficiency	The architecture minimizes idle compute costs through right-sized ECS task baselines, environment-specific scaling, and usage-based AWS pricing to maintain the MVP under the $300/month operating target.
Observability (BNS 7)	Distributed tracing and mechanical telemetry enable rapid troubleshooting and data-driven verification of feature usage.

6.2 Logical System Architecture

The logical architecture has been modernized from the legacy design [4, 9] to a cloud-native, event-driven framework. While maintaining the Spring Boot application framework for business logic, the infrastructure will be re-engineered for AWS-native deployment using containerized services on AWS ECS Fargate, supporting controlled horizontal scaling and Offline-First (BNS 5) resilience.

Table 6.2.1

Logical System Architecture Table

Tier	Logical Component	Responsibilities	Tech Choice
Client [4. 8] (BNS 5)	Local Persistence Layer	Offline synchronization is implemented for two validated modules, while the persistence and synchronization framework is designed to support future expansion without architectural refactoring.	Dart / Flutter / Drift (SQLite)

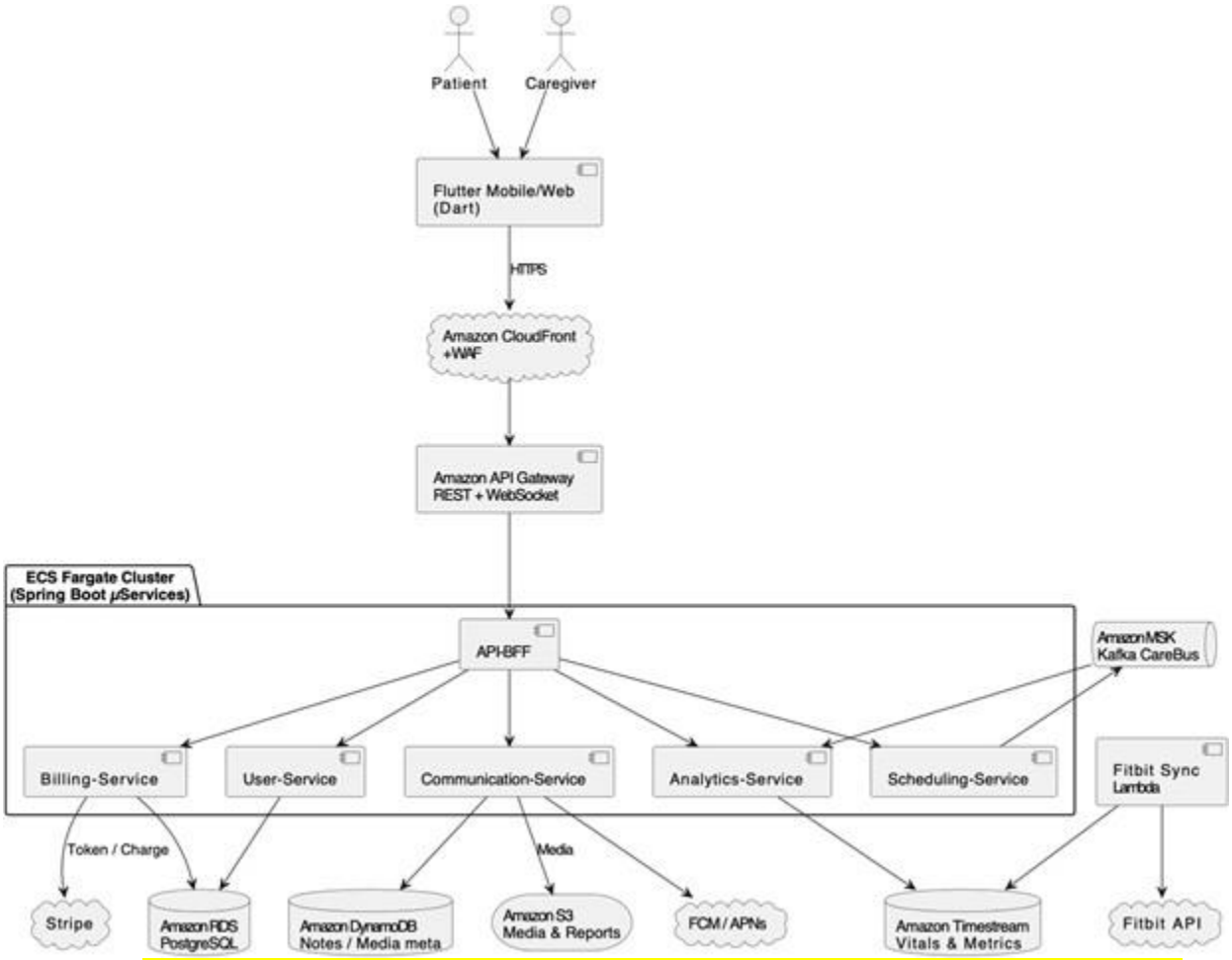
Edge [4. 8]	Security & Entry	Identity management (Cognito), request routing, and JWT validation.	Amazon API Gateway
Compute [4. 8]	Backend Sync Engine	Spring Boot microservices managing Health Logs, Scheduling, and SOS alerts. [4. 8]	**Spring Boot** (AWS ECS Fargate)
Integration [4. 8]	Cloud Intelligence	Consolidated AI note-taking and informational caregiving tips.	Amazon Bedrock (AWS Native)
Data BNS 5	Persistent Store	Relational source of truth for patient records and synchronized logs.	Amazon RDS (PostgreSQL)
BNS 7	Telemetry Mechanics	Distributed tracing, centralized logging, and PII-scrubbed usage metrics.	OpenTelemetry (OTel) SDK with deployment-configurable exporters (backend determined per environment).
BNS 6	Automation Layer	Automated provisioning of the Spring Boot environment and VPC.	AWS CloudFormation

Note. This table identifies the key architectural objectives and rationale for the CareConnect platform.

6.3 High-Level Architecture Diagram

Figure 6.3.1

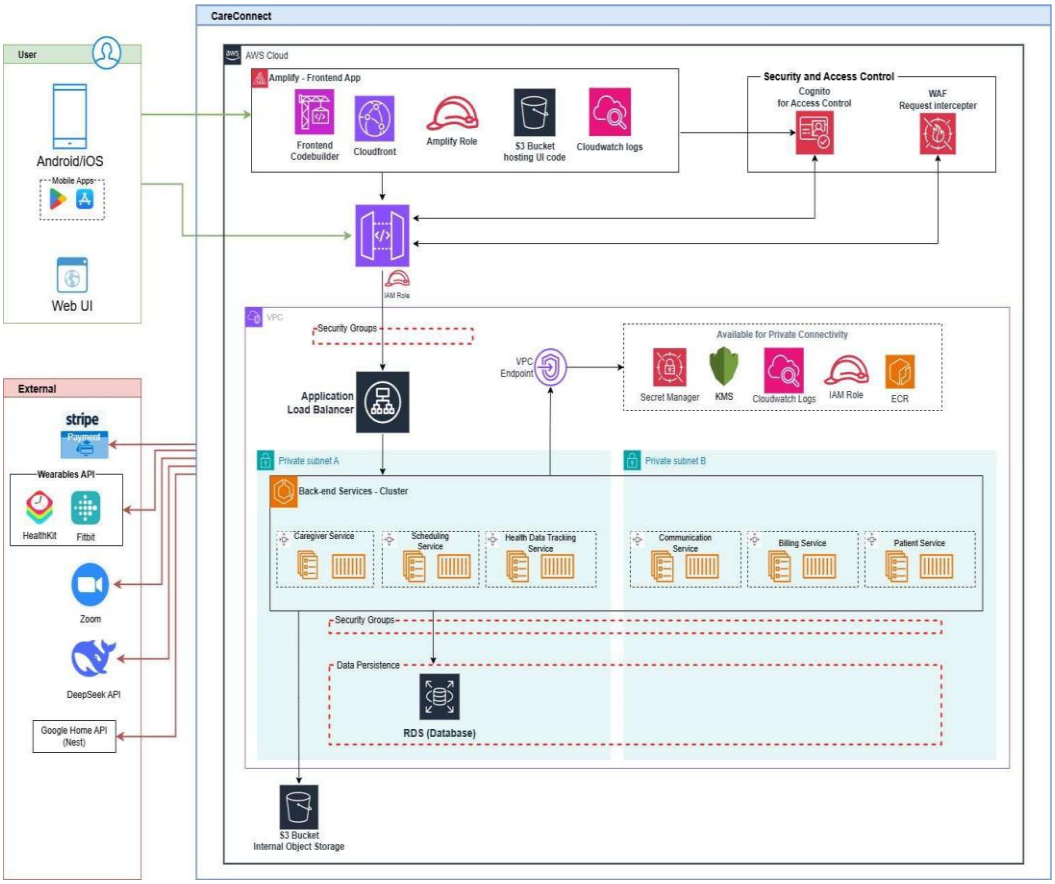
High-Level System Architecture Diagram 1



Note. This figure depicts the high-level CareConnect system architecture and is adapted from legacy documentation [4, 9].

Figure 6.3.2

High-Level System Architecture Diagram 2



6.4 Architectural Rationale

The following table justifies the technical decisions made to transition CareConnect to a production-ready, cloud-native framework. These decisions prioritize the three technical requirements: Offline-First (BNS 5), Infrastructure Automation (BNS 6), and Observability (BNS 7).

Table 6.4.1

Architectural Rationale Table

Concern	Design Decision	Justification
Scalability	Stateless Spring Boot microservices deployed as containerized services on AWS ECS Fargate.	Enables controlled horizontal scaling with predictable runtime behavior, stable authorization flows, and consistent database connection management, which supports offline synchronization, auditability, and compliance requirements.

Latency	API-BFF aggregates service calls; local-first data access via Drift/SQLite.	Reduces mobile round-trips and perceived latency; provides near-instant UI response regardless of network state.
Offline Reliability (BNS 5)	Decoupled abstraction layer using Flutter-compatible local storage with automated sync-on-reconnect logic.	Prevents data loss for critical health logs and medication markers during intermittent connectivity.
Security & HIPAA	Cognito JWT authentication, AES-256 encryption at rest, and PII-scrubbing middleware.	Ensures HIPAA/GDPR compliance while protecting sensitive patient metadata during telemetry collection.
Observability (BNS 7)	Centralized logging and distributed tracing instrumented via OpenTelemetry, exported via environment-specific backends (deployment-configurable).	Enables rapid troubleshooting and performance monitoring without the overhead of external BI tools.
Cost Control	Container-based architecture utilizing ECS Fargate service autoscaling and S3 lifecycle tiering.	Controls operational spend by scaling task counts based on demand and minimizing unnecessary idle capacity, supporting the targeted $300 monthly MVP budget.
Extensibility	Modular domain logic and infrastructure provisioned via AWS CloudFormation (BNS 6).	Allows future development to replicate the environment or extend offline support without re-engineering the foundational stack.
Dev & Ops Consistency	Standardized Infrastructure-as-Code (IaC) templates for all environments.	Ensures reproducible deployments, governance alignment, and eliminates "works on my machine" inconsistencies.

Note. This table details the logical system architecture, including component tiers, responsibilities, and technology choices.

6.5 Runtime Flows

The following flows illustrate how the system executes the core caregiver and patient features defined in the legacy requirements [4, 9], updated for an AWS-native, container-based execution model.

Secure Patient Log Entry (Legacy Feature [4, 9])

- Patient Log Entry → **Spring Boot Service (ECS Fargate) → PII/PHI Scrubber (BNS 7) → Amazon RDS Commit → (Optional) Amazon Bedrock Informational Analysis → UI Update.**

Offline Data Capture & Synchronization (BNS 5)

- Patient/Caregiver (Offline) → **Local Device Store (Drift/SQLite)** → Connectivity Restored → **Sync Manager** detects changes → **API Gateway** → **Spring Boot Backend** → **Amazon RDS** Global Commit → Sync ACK returned to client

Automated Incident Alerting (Legacy SOS Feature [4, 9])

- SOS Trigger → **Spring Boot Logic** → **Amazon Pinpoint** (Push/SMS) → Designated Caregiver Notification → **OpenTelemetry trace emitted and exported to AWS-managed tracing backend (BNS 7).**

Infrastructure Provisioning (BNS 6)

- Developer Updates Template → **AWS CloudFormation → Automatic update of ECS clusters/services and task definitions, VPC settings, and RDS schema.**

6.6 Technology Stack Summary

The following table summarizes the modernized technology stack for the CareConnect platform. This selection prioritizes AWS-managed services to ensure the implementation of the three (Team B) core technical requirements.

Table 6.6.1

Technology Stack Summary Table

Layer	Tech Choice	BNS / Legacy Alignment
Front-end	Flutter 3.x (Dart 3)	Single codebase for iOS/Android; includes Drift (SQLite) for BNS 5 offline persistence.
Services	Java 21, Spring Boot 3	Core business logic from [4, 9] containerized and deployed on AWS ECS Fargate for predictable execution and controlled scaling.
Data	RDS (Postgres 15), DynamoDB	Relational master records and high-speed metadata storage with used-based, on-demand pricing.
Integration	Amazon Bedrock & Pinpoint	Consolidates legacy AI and notification requirements into secure, native AWS services.
IaC (BNS 6)	AWS CloudFormation	Automated provisioning of the entire AWS environment and VPC networking.

Observability (BNS 7)	OpenTelemetry (OTel)	Standards-based instrumentation with deployment-configurable telemetry backends, supports portability while meeting BNS 7 requirements.
Security	AWS Cognito, KMS, IAM	HIPAA-compliant OIDC authentication and least-privilege resource access.

Note. This table identifies the specific technologies used to realize the architectural objectives, transitioning from the legacy infrastructure [4, 9] to a modernized, container-based, and offline-capable stack

6.7 Deployment Topology Highlights

- All Spring Boot services are deployed as containerized tasks on AWS ECS Fargate within private subnets of a dedicated VPC [4, 9].
- The entire topology, including VPC networking, IAM least-privilege roles, and database schemas, will be provisioned and managed via AWS CloudFormation (BNS 6).
- Public access is restricted to the Amazon API Gateway, which handles JWT validation via AWS Cognito. All downstream services remain isolated from the public internet [4, 9].
- Structured JSON logs, metrics, and traces are instrumented using OpenTelemetry and exported to deployment-configurable observability backends for BNS 7 compliance.
- The topology specifically supports the "Sync-on-Reconnect" flow. Encrypted local storage on Flutter devices interacts with the API Gateway to reconcile data with the Amazon RDS master instance once connectivity is restored (BNS 5).

6.8 Frontend

The CareConnect frontend is a cross-platform mobile application developed using Flutter and Dart. This architecture ensures a single codebase for iOS and Android while supporting the BNS 5 (Offline-First) requirement through local state management and a persistent local database (Drift/SQLite). The UI is architected with role-based access control (RBAC) to provide distinct experiences for patients and caregivers, as established in the legacy functional requirements [4, 9].

MVP Page & Feature Directory

- Welcome / Authentication Page (Both Roles) [4, 9]
  - Status: Critical MVP
  - Function: Secure login and role-based routing via AWS Cognito.
- Patient Home Dashboard (Patient Role) [4, 9]
  - Status: Critical MVP
  - Function: Centralized status view, SOS trigger, and Sync Manager status, tracking the two offline-synchronized MVP modules (Mood & Symptom Tracking, Manual Calendar).
- Mood & Symptom Tracking (Patient Role) [4, 9]
  - Status: BNS 5 Enabled (Offline-First)

    - Function: Offline capture of daily wellness and physical symptoms; syncs to cloud upon reconnection.
- Medication Logging (Patient Role) [4, 9]
    - Status: Online-First (No Offline Sync in current MVP)
    - Function: Tracking adherence and dosage, requires connectivity in the current MVP (offline sync deferred).
- Manual Calendar (Patient Role) [4, 9]
    - Status: BNS 5 Enabled (Offline-First)
    - Function: Local entry of appointments or care milestones; data resides in local Drift DB until synced.
- Caregiver Home Dashboard (Caregiver Role) [4, 9]
    - Status: Critical MVP
    - Function: Real-time alert feed, prioritized patient list, and consolidated AI caregiver tips.
- Patient Health Logs & Reports (Caregiver Role) [4, 9]
    - Status: Critical MVP
    - Function: Read-access to synced mood, symptom, and med data; includes consolidated AI trend analysis.
- Electronic Visit Verification (EVV) (Caregiver Role) [4, 9]
    - Status: Team A Module (Not Offline-Synced by Team B in current MVP)
    - Function: Location-stamped check-ins; online-first behavior in Team B scope unless cross-team integration is approved.

6.9 Backend

The backend architecture will be modernized from the legacy containerized prototype [4, 9] to an AWS-native, container-based model that fulfills Team B mandates for cost-efficient scaling and Infrastructure-as-Code (BNS 6).

- **AWS CloudFormation (BNS 6) – Infrastructure-as-Code**
    - **Purpose:** The single source of truth for the entire backend environment.
    - **Features:** Defines, provisions, and updates VPC networking, ECS services/task definitions, and RDS schemas to ensure consistent and repeatable deployments across all project phases.
- **Amazon Cognito – User Authentication & Authorization** [4, 9]
    - **Purpose:** Manages secure user identity and role-based access control (RBAC).
    - **Features:** Issues JWT tokens for API Gateway validation; enforces secure session management for both Patient and Caregiver roles.
- **Amazon API Gateway – Managed Entry Point** [4, 9]
    - **Purpose:** Acts as the secure interface for the Flutter mobile frontend.
    - **Features:** Routes requests to backend services, manages rate limiting, and serves as the sync endpoint for **BNS 5 (Offline-First)** data reconciliation.
- **AWS ECS Fargate – Managed Container Compute [4, 9]**
    - **Purpose:** Hosts containerized Spring Boot services with predictable runtime behavior suitable for authorization, synchronization, and auditing workflows.

    - **Features:** Supports controlled horizontal scaling via service autoscaling policies to handle demand peaks while maintaining a defined baseline capacity for service availability and operational stability.
- **Spring Boot 3 – Core Business Logic** [4, 9]
    - **Purpose:** The primary framework for health data processing and Caregiver insights.
    - **Features:** Containerized for ECS Fargate deployment; handles synchronization logic for the two offline-synchronized MVP modules (Mood & Symptom Tracking and Manual Calendar). Other modules remain online-first or local-capture only, with future expansion supported by the shared framework.
- **Amazon RDS (PostgreSQL) – Managed Relational Database** [4, 9]
    - **Purpose:** Stores structured health records and caregiving schedules.
    - **Features:** Encrypts data at rest via AWS KMS; serves as the master global store for data synced from the local mobile Drift DB.
- **OpenTelemetry (BNS 7) – Observability**
    - **Purpose:** Provides standards-based instrumentation for mechanical telemetry, distributed tracing, and metrics collection.
    - **Features:** Emits privacy-scrubbed logs, metrics, and traces to environment-defined backends, validating system health and synchronization latency without exposing PII/PHI.
- **AWS IAM – Least Privilege Governance** [4, 9]
    - **Purpose:** Secures communication between all cloud resources.
    - **Features:** Enforces strict execution roles; Backend functions are granted only the specific permissions needed to write to RDS or S3.

6.10 Database

The CareConnect data layer will utilize a hybrid-persistence model. This architecture bridges the gap between the centralized Amazon RDS (PostgreSQL) master and the BNS 5 local device storage, ensuring continuous operation for critical modules during network-constrained states.

6.10.1 Dual-Tier Storage Strategy

- **Cloud Master (Amazon RDS):** The authoritative relational store for all health records, user identities, and system metadata.
- **Local Persistence (BNS 5 - Drift/SQLite):** A secure, encrypted local database on the Flutter mobile client. It is strictly limited to two approved offline modules in the current MVP implementation: **Mood & Symptom Tracking, and Manual Calendar.**

6.10.2 Key Schema Entities

- **Health Logs:** BNS 5 Enabled (Offline-First) for Mood & Symptom Tracking, encrypted local persistence with sync-on-reconnect.
- **Medication & Tasks:** Online-first storage; no offline synchronization in current MVP.
- **Scheduling (BNS 5 Enabled):** Tracks Manual Calendar entries with encrypted local storage and sync-on-reconnect.

- **System Telemetry (BNS 7):** Tracks synchronization metadata and system performance without storing PII/PHI.

6.10.3 Offline Data Handling & Sync Logic

- **Encryption at Rest:** Contrary to legacy prototypes, all BNS 5 local data is **encrypted on the device** to maintain HIPAA compliance for sensitive health markers.
- **Conflict Resolution:** The system uses a "Last-In-Wins" or "Timestamp-Based" reconciliation logic to merge offline entries with the cloud master upon reconnection.
- **Data Integrity:** Unique identifiers (UUIDs) are generated client-side to prevent record duplication during the synchronization handshake.

6.10.4 Security & Compliance

- **Encryption:** AES-256 encryption at rest (KMS) and TLS 1.3 in transit.
- **PII/PHI Isolation:** Clinical data is logically separated from system logs to facilitate **BNS 7** privacy-safe monitoring.

6.11 APIs

The CareConnect platform integrates with a focused set of APIs and platform services that directly support Team B's approved business needs: offline data storage and synchronization, infrastructure standardization, and observability with anonymous feature analytics. API usage is intentionally scoped to these needs to reduce complexity, avoid unnecessary external dependencies, and support maintainability for future cohorts. For a complete list of all APIs used by the application, refer to the legacy documentation [4, 9].

6.11.1 Offline Data Storage and Synchronization APIs (BD5)

Enable selected high-value application features to function when network connectivity is unavailable by persisting data locally on the device and synchronizing changes with the backend once connectivity is restored.

Technologies / APIs

- **Local Storage API (Flutter-compatible, e.g., Drift or SQLite-based abstraction)**
- **Connectivity Monitoring APIs**
  - Detect online/offline state transitions
- **Backend Synchronization APIs**
  - REST-based backend endpoints supporting incremental data sync

Supported Modules

Offline support with synchronization is explicitly limited to:

- Mood & Symptom Tracking
- Manual Calendar

All other modules operate in online-first mode or support local capture without synchronization in the current MVP.

Integration Model

- Data is written to a local persistence layer when offline.
- An abstraction layer decouples application features from storage and synchronization logic.
- When connectivity is restored, queued changes are synchronized to the backend using defined API endpoints.
- Conflict handling and retry logic follow documented offline-first design patterns to ensure reliability and data consistency.

Considerations

- Local storage is limited to essential data only to keep the application lightweight.
- The design is based on research into industry-standard offline-first architectures and available Flutter libraries.
- The abstraction layer enables future cohorts to extend offline support without duplicating core implementation logic.

6.11.2 Observability and Anonymous Feature Analytics APIs (BD7)

Provide operational visibility into application behavior while enabling anonymous feature usage analysis without collecting personally identifiable information.

Technologies / APIs

Observability APIs

- Centralized logging
- Metrics collection
- Distributed tracing

Anonymous Analytics APIs

- Event-based feature usage tracking
- Aggregated, non-identifying metrics

Anonymous Analytics APIs

- Event-based feature usage tracking
- Aggregated, non-identifying metrics

Integration Model

- Application components emit logs, metrics, and traces to a centralized observability platform.
- Feature usage events are captured using anonymized identifiers with no association to patient or caregiver identity.
- Operational telemetry and feature analytics are handled as separate data streams to maintain privacy boundaries.

Considerations

- Observability data supports faster troubleshooting, performance monitoring, and reliability improvements.
- Analytics are limited to feature usage patterns (e.g., frequency, entry points, duration).
- No personally identifiable or protected health information is collected as part of analytics.
- The design supports healthcare privacy expectations and compliance constraints.

6.12 AI Services

The "Ask AI" feature provides informational support and care-plan inquiry functionality defined in the legacy requirements **[4, 9]**. High-risk responses are defined as AI outputs containing medication dosage recommendations, emergency instructions, or diagnostic claims. These responses are flagged via keyword detection and require caregiver approval prior to patient display.

- **Functional Requirement:** As per [4, 9], the system provides patients with immediate responses to routine queries regarding care plans and medication schedules to reduce caregiver burden.
- **Architecture:** The feature is implemented using an AWS-native inference service integrated directly with the Spring Boot backend. This ensures all data processing remains within the project's secure VPC boundary.
- **Data Grounding:** Responses are generated using the patient's specific records stored in Amazon RDS, ensuring accuracy and relevance to the individual's care plan.
- **Safety & Review:** Consistent with the safety protocols in [4, 9], a human-in-the-loop (HITL) review process is enforced. Any response deemed "high-risk" is routed to the caregiver for manual approval before the patient can view it.

6.13 Hosting

6.13.1 Cloud Deployment Model

- **Managed Compute Environment:** The backend utilizes a managed container environment to host core domain services originally scoped in [4, 9].
- **Network Isolation:** To protect sensitive health data (PII/PHI), all backend services are deployed within private VPC subnets, as required by the security standards established in [4, 9].

6.13.2 Compute Optimization (BNS 5)

A hybrid compute approach is utilized to balance the baseline requirements of **[4, 9]** with modern cost-efficiency:

- **Background Workloads:** On-demand container execution is used for background tasks, including BNS 5 offline data synchronization for the two MVP modules (Mood & Symptom Tracking and Manual Calendar).

6.14 Security

Security is the foundational layer of CareConnect, evolving the safety standards established in the legacy documentation [4, 9] into a cloud-native framework. This approach integrates industry best practices to protect sensitive health information while fulfilling the BNS 7 (Privacy-Safe Monitoring) mandate.

6.14.1 Data Protection

- All Protected Health Information (PHI) is encrypted at-rest using AES-256 and in-transit via TLS 1.3. This ensures that the health records scoped in [4, 9] remain confidential across all system tiers.
- Identifiable information is replaced with tokens where possible.
- **Offline Data Security (BNS 5):** To support the two offline-synchronized MVP modules (Mood & Symptom Tracking and Manual Calendar), all locally stored data is encrypted at rest on the mobile device. Other modules remain online-first in the current MVP.

6.14.2 Authentication and Access Control

- Secure authentication is managed via AWS Cognito, supporting MFA and social login as per the identity requirements in [4, 9].
- Access to patient data and caregiver dashboards is strictly determined by user roles—Patient, Caregiver, or Admin—preserving the role-specific functionality established in earlier project phases [4, 9].
- Managed through AWS IAM, ensuring that backend services and human users only have the minimum permissions required to perform their tasks [4, 9].

6.14.3 Infrastructure Hardening & Monitoring (BNS 7)

- System components are deployed within private VPC subnets with no direct public internet exposure [4, 9].
- **Web Application Firewall (WAF):** Provides protection against SQL injection and XSS at the API Gateway level [4, 9].
- **Privacy-Safe Telemetry:** In alignment with team B's (BNS 7) goals, the architecture includes the technical mechanics required to collect system telemetry. These collection mechanics must ensure that system-level performance data, error traces, and synchronization logs are captured for troubleshooting without persisting sensitive patient information in the telemetry layer

6.14.4 Infrastructure as Code (BNS 6)

- **CloudFormation Security:** All infrastructure provisions will follow security best practices. Version-controlled templates ensure that security groups and IAM policies are applied consistently, preventing *"configuration drift"* and unauthorized modifications.

7. Project Deliverables

Fully Functional Mobile Application

A cross-platform mobile application developed in Flutter, providing role-specific functionality for patients and caregivers as established in the legacy requirements [4, 9].

- **(Offline-First) Mechanics (BNS 5):** Supports a production-grade offline framework with full synchronization implemented for two approved modules.
- **Telemetry Integration (BNS 7):** Includes embedded hooks for mechanical telemetry to track feature usage and system performance without capturing PII/PHI.

Backend Infrastructure & Service Layer

- **Cloud-Native Backend:** A consolidated AWS-native backend managing authentication, persistence, and communication [4, 9].
- **Integrated AI Services:** Deployment of the legacy "Ask AI" feature, transitioned to **AWS Bedrock** for AWS-native inference (BNS 6).
- **Refactored API Connectors:** Production connectors for OpenFDA and Jitsi Meet, utilizing localized error handling and caching to mitigate external dependency risk.

Cloud Infrastructure as Code (BNS 6)

- **AWS CloudFormation Templates:** Comprehensive IaC artifacts defining the VPC, container compute (ECS/Fargate), and RDS/DynamoDB resources.
- **Migration Records:** Detailed technical records documenting the transition from legacy configurations [4, 9] to a standardized AWS-native framework.

Telemetry and Observability Mechanics (BNS 7)

- **Data Pipeline Artifacts:** Implementation of the mechanical capability to collect logs, metrics, and traces, ensuring privacy-safe redaction of all health-related data at the point of ingestion.
- **System Performance Baselines:** Technical artifacts demonstrating the system's ability to track operational health and synchronization latency.

Technical Documentation

- **Project Management Plan (PMP):** The authoritative project governance document covering scope, labor-hour justification, budget, and risk registers.
- **Software Requirements Specification (SRS):** The technical requirements baseline defining the MVP boundaries for BNS 5, 6, and 7.
- **Developer Documentation:** Inline code comments and an updated GitHub repository structure, including a README with deployment instructions.
- **Technical Design Document (TDD):** Comprehensive architecture definitions, including the **BNS 5 Sync Data Model** and **BNS 7 Privacy Middleware** logic.
- **Software Test Plan & Test Report:** Formal strategy and execution records for unit, integration, and **HIPAA-compliance** validation testing.
- **Programmer's Guide:** Technical onboarding documentation, API specifications, and repository management protocols for future cohorts.
- **Deployment & Operations Guide:** SOPs for environment provisioning, secret management, and OpenTelemetry-based observability configuration with AWS-managed backends.

8. Risk & Mitigation

8.1 Risk Probability Matrix

- High: Scope creep → Mitigation: Strict change control.
- Medium: Cloud costs exceed budget → Mitigation: Monitor AWS billing weekly.
- Low: Team member bandwidth issues → Mitigation: Assign backup reviewer.

Table 8.1.1

Risk Probability Matrix Table

Probability				
Impact	**1 = High(80% ≤ x ≤ 100%)**	**2 = Medium High (60% ≤ x ≤ 80%)**	**3 = Medium Low (30% ≤ x ≤ 60%)**	**4 = Low (0% ≤ x ≤ 30%)**
A = High (Rating 100)	100 –Very High Exposure	80 –Very High Exposure	60 –High Exposure	30 – Moderate Exposure
B = Medium (Rating 50)	50 – High Exposure	40 – Moderate Exposure	30 – Moderate Exposure	15 – Low Exposure
C = Low (Rating 10)	10 – Low Exposure	8 – Low Exposure	6 – Low Exposure	3 – Very Low Exposure

Note. Risk Probability Matrix table created with reference from previous 2023 SWEN 670 AlphaSoft project (AlphaSoft, 2023).

8.2 Risk Identification, Probability, and Mitigation

Table 8.2.1

Risk Probability and Mitigation Table

Risk Description	Prob. (1-5)	Impact (A-E)	Score	Mitigation Strategy
Team Capacity: Members not completing assigned work due to labor hour constraints.	2	A	60	Utilize the RACI matrix and weekly sprints to track progress. Reassign tasks early if milestones are missed.
Technical Gap: Team unfamiliarity with Flutter or AWS-native AI services.	3	B	50	Allot "Sprint 0" for targeted R&D. Utilize existing AWS documentation and peer code reviews to accelerate learning.
Regulatory Risk: Failure to meet HIPAA/GDPR data privacy standards.	1	A	30	**BNS 7 implementation:** Engineering privacy-safe telemetry and end-to-end encryption [4, 9]. Conduct internal compliance audits.
Scope Creep: Client or stakeholders request features outside the 12-week MVP.	3	B	45	Strict adherence to the Change Management Process. Defer non-team B items (like USPS Mail Assist) to post-MVP phases.
Integration Complexity: Failure to integrate 8+ external APIs (Legacy Risk).	4	A	60	**Critical Mitigation:** Reduced API scope to strictly **OpenFDA and Jitsi Meet**. Implemented failure handling and local caching for BNS 5 features.
Security Breach: Unauthorized access to PII/PHI or data exposure.	2	A	40	Implementation of MFA via AWS Cognito, RBAC, and encryption of all data at rest and in transit [4, 9].
Data Synchronization Failure: Local data (BNS 5) failing to reconcile with the cloud.	3	B	45	Use of UUIDs and timestamp-based conflict resolution logic. Comprehensive testing of the synchronization pipeline.
Infrastructure Deployment: Errors in migrating from Terraform to CloudFormation (BNS 6).	2	B	30	Use of version-controlled templates and incremental deployments to validate infrastructure-as-code stability.
AI Hallucination/Accuracy: Inaccurate summaries	2	A	80	**Escalation Workflow:** Implement "Human-in-the-Loop" review; all AI notes are flagged as "Draft"

generated by legacy AI logic via Amazon Bedrock.				until manual caregiver verification/approval.
AI Cost Overruns: Amazon Bedrock token usage exceeding the $50/mo MRC estimate (Table 9.3.1)	3	B	30	Implement AWS billing alarms and usage-based alerts at 80% of budget and apply rate-limiting to the Bedrock API endpoint.
AI Data Leakage: PII/PHI being transmitted to the AI model without proper scrubbing.	4	A	30	**BNS 7 Compliance:** Force all Bedrock traffic through a privacy middleware layer to strip identifying data before model ingestion.

8.3 External API Risk Register

The following table identifies the dependencies on third-party services and the specific mitigation strategies to prevent "Integration Bloat" or external outages from stalling the MVP.

API / Service	Status in MVP	Key Risk	Mitigation Strategy
Stripe Connect	**In-Scope (Legacy)**	Compliance/KYC Failure	**Manual Fallback:** Retain manual payment verification if Stripe onboarding fails for a user; restrict to legacy stable API versions.
AWS Bedrock	**In-Scope (BNS 6)**	Token Cost/Latency	**Caching & Throttling:** Implement a local cache for common summary requests and set hard monthly dollar caps in AWS Billing.
Jitsi Meet	**In-Scope (Legacy)**	Connection Drops	**Graceful Degradation:** The UI shall provide a "Rejoin" button and fallback to a simple text-based status update if the video socket fails.
OpenFDA	**In-Scope (Legacy)**	Schema Changes	**Persistence Abstraction:** Use the BNS 5 abstraction layer to cache medicine data locally, ensuring the feature works even if the API is down.
USPS Informed Delivery	**Out-of-Scope**	Privacy/PHX Leakage	**Exclusion:** Explicitly removed from MVP (Section 3.2.5). No API keys will be provisioned or tested to prevent data exposure risks.
Fitbit / HealthKit	**Out-of-Scope**	Sync Complexity	**Exclusion:** Deferred to future cohorts. Data input remains "Manual Entry Only" to ensure 100% reliability for BNS 5 modules.

9. Budget Estimate

9.1 Labor / Effort Estimate

The primary cost driver for the CareConnect project is professional labor. Labor rates are derived from the 2024 U.S. Bureau of Labor Statistics for Software Developers and System Analysts. This estimate assumes a consistent effort of 20 hours per week per team member over the 12-week project duration, totaling 1,200 cumulative labor hours.

To ensure project feasibility and the delivery of a production-ready documentation suite, labor is allocated as follows:

Table 9.1.1
Labor Allocation by Workstream

Workstream	Labor %	Est. Hours	Activities & Deliverables
Project Governance	15%	180	Maintenance of PMP and SRS; Weekly Sprints; RACI management.
Technical Design	15%	180	Development of the TDD; BNS 5 Data Modeling; BNS 6 IaC Architecture.
Feature Engineering	35%	420	BNS 5 (Offline Logic), BNS 6 (AWS Migration), BNS 7 (Privacy Middleware).
Quality & Validation	20%	240	Software Test Plan & Test Report; HIPAA compliance auditing; Bug fixing.
Technical Guides	15%	180	Programmer's Guide, Deployment/Ops Guide, and User Guides.
TOTAL	**100%**	**1,200**	

Summary Budget:

- **Labor (5 roles, 12 weeks):** $70,862.40
- **Infrastructure (AWS Managed Services):** $1,320.00
- **Materials:** $0.00 (Student-provided hardware)
- **Total Estimated Project Cost: $72,182.40**

Table 9.1.2

Budget Estimate for Labor Costs

Team Member	Project Role	Cost / Hour	Work Hours / Week	Weekly Salary	Monthly Salary	3-Month Cost
Eduardo Estrada	Team Lead / Software Developer	$69.50	20	$1,390.00	$5,560.00	$16,680.00
Joseph Wojcik	DevOps / Software Developer	$65.34	20	$1,306.80	$5,227.20	$15,681.60

Gary Jurado	UI/UX Designer / Developer	$53.58	20	$1,071.60	$4,286.40	$12,859.20
Yismaw Tilaye	System Analyst / Developer	$53.83	20	$1,076.60	$4,306.40	$12,919.20
Chasity Sapp	QA / Test Engineer / Developer	$53.01	20	$1,060.20	$4,240.80	$12,722.40
Totals			**100**	**$5,905.20**	**$23,620.80**	**$70,862.40**

Note. Cost per hour rates retrieved from the U.S. Bureau of Labor Statistics (U.S. Bureau of Labor Statistics, 2024)

9.1.2 Labor Allocation Rationale

The 1,200 total labor hours (20 hrs/week per member) are allocated across the following activities to ensure production-grade delivery:

- **Architectural Migration (BNS 6 & 7):** 30% (360 hrs) – Dedicated to CloudFormation templating, IAM security hardening, and observability middleware.
- **Feature Development (BNS 5):** 40% (480 hrs) – Implementing the offline persistence framework and validating sync-on-reconnect logic for two MVP modules (Mood & Symptom Tracking and Manual Calendar), while establishing a reusable foundation for future expansion.
- **QA & Regression Testing:** 20% (240 hrs) – Ensuring legacy features (Stripe, Jitsi) remain stable post-migration.
- **Documentation & Compliance:** 10% (120 hrs) – Maintaining the PMP, SRS, and HIPAA-compliance audits.

9.2 Materials Cost

Given this project is to be completed in the scope of a capstone course, team members will be responsible for their hardware technology used for development (laptops, keyboards, mice, etc.). Thus, hardware costs are estimated to be $0.

9.3 Infrastructure Cost

This subsection converts the target technical architecture (Section 6) into a quantified operating expense forecast so that project sponsors can gauge cash burn and approve the required AWS spend-limit. The estimate follows a "bottom-up, usage-based" methodology:

1. **Reference Architecture Mapping** – Each logical component (API Gateway, AWS EWS/Fargate, RDS, S3, etc.) is mapped to its corresponding billable SKU in the AWS Pricing Catalog (Amazon Web Services, 2024). This reflects the transition to managed container compute to ensure cost-efficiency.
2. **Sizing Assumptions** – Workload drivers—concurrent users, sync frequency for **BNS 5** offline modules, and telemetry ingest volume for **BNS 7**—are derived from the SRS and

Performance NFRs. Conservative headroom (+20%) is applied to prevent cost overruns during load spikes.

3. **AI Inference Modeling** – Per reviewer feedback, this estimate explicitly includes AWS-native AI service costs. Pricing is modeled on token-based consumption for the "Ask AI" feature [4, 9], ensuring all generative AI workloads are accounted for within the AWS budget.
4. **Unit-Price Collection** – 2026 on-demand prices for the *us-east-1* region were validated against the AWS Pricing Calculator. High-complexity external integrations (Stripe, Nest, Fitbit) have been removed from the MVP scope to reduce integration risk and financial exposure.
5. **Estimation Technique** – For recurring services, costs are projected as:

   MRC = Unit Price X Monthly Usage Qty
   Project Burn = MRC X 3 months

6. **Contingency Allocation** – A **32% buffer** is included to cover data-egress spikes and high-frequency synchronization events, ensuring the project remains within the approved financial envelope.

Infrastructure Cost Summary:

- **Monthly Recurring Cost (MRC):** ≈ $333.33 (Consolidated Compute, Database, Storage, and AI Inference)
- **3-Month MVP Base Total:** ≈ $1,000.00
- **Contingency (32% for Data Egress/Spikes):** ≈ $320.00
- **Total Infrastructure Budget: $1,320.00**

Budget Alignment:

The infrastructure cost represents approximately 1.8% of the total project budget, affirming the decision to utilize managed AWS services (ECS Fargate, RDS, API Gateway) to control idle capacity and operational overhead.

Financial Policy Alignment: OpEx model, HIPAA BAA eligible services.

- **Total Team B Project Budget: $72,182.40** (Labor: $70,862.40 + Infrastructure: $1,320.00).

Table 9.3.1

Infrastructure Cost Table

#	Service	Sizing Assumption	Unit Price*	MRC (USD)
1	**AWS ECS Fargate (App Service)**	1 service, 2 tasks baseline, 0.25 vCPU, 0.5GB RAM, 730h/mo	$/vCPU-h + $/GB-h (on-demand)	TBD

2	**API Gateway (REST)**	3M API calls/mo	$3.50 per 1M	$11.00
3	**AWS Cognito**	6,000 Monthly Active Users (MAU)	$0.0055/MAU (after free tier)	$33.00
4	**Amazon RDS (PostgreSQL)**	db.t4g.medium Multi-AZ	$0.101/h × 730h	$74.00
5	**RDS Storage & I/O**	40GB gp3 + 3M I/O	$0.115/GB + $0.20/M I/O	$8.00
6	**Amazon DynamoDB**	15GB + 2M WCUs/RCUs	$1.25/GB + $1.25/M WCU	$25.00
7	**S3 Standard**	200GB data + 1M operations	$0.023/GB + API fees	$8.00
8	**CloudFront Data Out**	400GB/mo egress	$0.085/GB (Tier 1)	$34.00
9	**Observability Backend (OTel log/trace storage and processing)**	50GB logs/traces (BNS 7 Compliance)	$0.50/GB	$30.00
10	**Amazon Bedrock**	5M Tokens (Informational AI Inquiry)	Avg $0.01 / 1k tokens	$50.00
11	**Route 53 / Certificates**	DNS & SSL Management	Flat fee / Monthly	$12.00
12	**WAF / Shield**	Basic Security Protections	Usage-based	$40.00
Subtotal – AWS Recurring				**$333.00**

9.3.1 AWS Core Services

Pricing Basis: AWS public on-demand rates as of 2026, *us-east-1* region. To maintain the $333/month target, the architecture utilizes AWS ECS Fargate for compute and Amazon Bedrock (on-demand tokens) for AI services, leveraging the AWS Free Tier for initial development and testing.

9.3.2 Third-party / One-time Fees

Table 9.3.2.1

Fees Table

Service	Type	Cost USD
Apple Developer Program	Annual	**99**
Google Play Console	One-time	**25**
Stripe – Standard Plan	Usage-based	< 0.3 % per txn (pass-through)
Sectigo TLS Wildcard (if not using ACM)	1-year cert	**80**

9.3.3 Total Budget for MVP Phase (13 weeks ≈ 3 months)

Table 9.3.3.1

Total Infrastructure Table

Category	Monthly	3-Month	Notes
AWS Core (Table 9.3.1 subtotal)	$333	$999	Includes Managed Container Compute (ECS Fargate) + AI Inference
App-store & TLS fees	—	$204	$99 + $25 + $80
Subtotal		**$1,203**	
Contingency 10 %		**$117**	Adjusted to align with $1,320 target
Grand Total (Infrastructure)		**≈ $1,320 USD**	

9.3.4 Cost-Control & Optimization Plan

To ensure the project remains within the $1,320 infrastructure budget, the following cost-control levers will be implemented. These focus on reducing waste during the development and testing phases.

Table 9.3.4.1

Cost-Control Tables

Lever	Action	Expected Savings
Compute	Right-size ECS task CPU/memory; scale task counts by environment; auto-stop RDS Development instances during non-working hours.	15–20%
Storage	Implement S3 Lifecycle policies for logs >30 days; compress telemetry log streams at ingestion/export.	5–7%
Traffic	Enable CloudFront caching to reduce data egress from the origin database.	3–5%
Reserved Instances	Evaluate 1-year RDS Reserved Instance (RI) commitment after the Milestone 2 architecture is finalized.	35–40% on DB Layer
Serverless Tuning	Optimize ECS task sizing (CPU/memory) and autoscaling thresholds to reduce idle capacity and throttle-induced latency.	10–15% on Compute

9.3.5 Alignment to Financial Policies

To ensure the project meets corporate and regulatory standards, the following financial guidelines are applied:

- **Cap-Ex vs. Op-Ex:** All infrastructure spend is classified as **OpEx** (Pay-as-you-go), aligning with U.S. GAAP guidance for cloud-service costs (ASC 350-40).
- **Cost Recognition:** Expenses are booked monthly via a consolidated AWS invoice. Budget tracking is monitored through **AWS Budgets** with automated alerts set at 50%, 75%, and 90% of the $333/month threshold. This includes specific monitoring for **Amazon Bedrock** token consumption to prevent AI cost overruns [4, 9]
- **Audit Controls:** Detailed billing reports are exported to an immutable S3 bucket for audit transparency, ensuring compliance with internal IT-R&D cost-center reporting.
- **HIPAA Compliance:** All selected AWS services (RDS, S3, Cognito, ECS/Fargate) are covered under the **AWS Business Associate Addendum (BAA)**, ensuring HIPAA eligibility with no additional licensing surcharges.

9.3.6 Budget Summary for Project Plan

The following table consolidates all estimated expenditures for the CareConnect 12-week MVP development cycle. This budget assumes a five-person cross-functional team and a fully AWS-native, managed-services infrastructure.

Table 9.3.6

Budget Summary Table

Line Item	Section Reference	Amount (USD)
Labor Cost	Section 9.1	$70,862.40
Materials Cost	Section 9.2	$0.00
Infrastructure Cost	Section 9.3	$1,320.00
Total Project Budget		**$72,182.40**

Note. This summary includes the 32% contingency buffer applied to infrastructure to account for variable data egress, BNS 7 telemetry volume, and Amazon Bedrock (AI) [4, 9] token consumption.

Financial Validation: The infrastructure allocation represents approximately 1.8% of the total project spend. This confirms the feasibility of a AWS-native, pay-as-you-go strategy, which minimizes capital exposure while strictly adhering to HIPAA security and operational compliance requirements established in the legacy documentation [4, 9]. This low infrastructure-to-labor ratio demonstrates "Cloud-Native Thinking" by shifting the project's value focus toward software engineering rather than server maintenance.

10. Roles & Responsibilities

10.1 Team Roles

Table 10.1.1

Roles & Responsibilities Table with Assigned Team Members

Role	Team Members	Responsibilities
Client	Dr. Assadullah, Roy Gordon	Provide product requirements and feedback on product progress
Project Mentor	Roy Gordon	Advise student team members; review deliverables upon request and provide constructive feedback
MS Teams & GitHub Mentor	Roy Gordon	Act as Admin for MS Teams and GitHub; schedule MS Teams meetings; provide GitHub access for course
Class Project Manager	Colyn Mahoney	Organize team project progress for all teams; provide updates to course professor
Team B Lead	Eduardo Estrada	Organize project team deliverables; provide project progress to Class Project Manager; manage team member communications
Quality Assurance	Chastity Sapp	Design project architecture; lead developers in implementation
Technical Lead/Architect	Joseph Wojcik	Designs the project architecture
Lead Business Analyst	Yismaw Tilaye	Lead business analysts; identify use cases; confer with leads to determine functional capabilities; meet with stakeholders
Lead UI/UX Designer	Gary Jurado	Lead UI/UX Designers; confer with leads on designs; help design wireframes for product mockups

10.2 RACI Matrix

Table 10.2.1 RACI Matrix Table Key:

- R – Responsible
- A – Accountable
- C – Consulted
- I – Informed

Table 10.2.1

RACI Matrix

Workstream/Deliverables	PL	BA	QA	BE	UX	APP	OPS	PSC
Milestone 1: Project Plan (method, scope, schedule, grading alignment)	A	R	C	C	C	C	C	C

Weekly status reports & sprint demos	A	R	C	C	C	C	C	C
RAID (risk/issue/action/decision) registers	A	R	C	C	C	I	C	C
Change control (impact, MoSCoW, approvals)	A	R	C	C	C	I	C	C
Traceability matrix (requirements→tests→commits)	A	C	R	C	C	I	C	C
Milestone sign-offs & Canvas submissions	A	R	C	I	I	I	I	I
SRS (Software Requirements Specification)								
1 Introduction (purpose, scope, refs, glossary)	A	C	C	C	C	I	I	C
2 Overall Description (users, constraints, assumptions)	C	A/R	C	C	C	C	I	C
3 External Interfaces (API, UI, device, comms)	C	C	C	A/R	C	C	C	C
4 System Features & Use Cases + Acceptance Criteria	C	C	C	C	A/R	C	I	C
5 Non-Functional Requirements (perf, accessibility, etc.)	C	C	A/R	C	C	I	C	C
6 Data Model & Entities (ERD, schema, PII tagging)	C	C	C	A/R	C	I	I	C
7 Constraints & Compliance	C	A/R	C	C	C	I	I	C
Design & Architecture								
UX research, personas, user flows, wireframes, prototypes	C	C	C	C	A/R	C	I	I
UI design system (components, accessibility specs)	C	C	C	C	A/R	C	I	I
Architecture & API design (contracts, versioning)	C	C	C	A/R	C	C	C	C
Privacy & security design (threat model, roles, encryption, retention)	C	C	C	C	C	I	C	A/R
Build								
Backend/API implementation	I	C	C	A/R	C	C	C	C
Mobile app client implementation	I	C	C	C	C	A/R	C	C
Dev environments & CI/CD (branching, pipelines, secrets, IaC)	I	C	C	C	C	C	A/R	C
Test & Quality								
Test strategy & plan (system, integration, performance, accessibility)	C	C	A/R	C	C	I	C	C

Test cases & execution; defect triage	I	C	A/R	C	C	I	I	C
UAT planning & facilitation with customer	A	R	C	C	C	C	I	C
Release & Ops								
Release management (cut, tag, deploy to staging/prod)	I	C	C	C	I	C	A/R	C
Runbook, support & handover; training materials	A	R	C	C	C	C	C	C
Compliance & Course								
Privacy & security review (PII inventory, access controls, policy)	C	C	C	C	C	I	C	A/R
Peer reviews & individual contribution logs (grading)	A	R	R	R	R	R	R	R

Note. RACI Matrix created with reference from previous 2025 SWEN 670 AlphaSoft project (AlphaSoft, 2023).

11. Monitoring and Reporting

11.1 Objectives

The need for monitoring and reporting comes from having to track project progress as well as team performance. The continuous tracking of these aspects of the project will allow for the earliest indication of deviation from the schedule and project goals as well as allowing for corrective actions to be taken sooner. The data used in the reporting will be transparent and continuously updated to allow project stakeholders to have the most accurate information to make informed decisions.

- GitHub + Excel Master Tracker for progress vs plan.
- Weekly dashboards (variance, defects, pass/fail rates).
- Peer reviews, CI/CD gates, HIPAA compliance checks.
- Decision Tracker integrated for scope traceability.
- Observability dashboards derived from OpenTelemetry metrics exported to AWS-managed monitoring services.

11.2 Intra-Team Monitoring and Reporting

The CareConnect team will set up structured monitoring and reporting mechanisms to ensure timely progress, scope alignment, and quality assurance. Task ownership, completion, and status updates will also be tracked using GitHub Projects and synchronized with a Microsoft Excel Master Tracker document. Weekly dashboards will highlight planned vs. actual progress, hours logged, and quality metrics (e.g., defect counts, test pass/fail rates). Variance commentary will be documented, and unresolved variances will be escalated to the Risk Register.

1. Weekly Sprint Reviews – Each week, the team will review completed tasks, discuss blockers, and reprioritize backlog items.
2. Decision Tracker Integration – All major project decisions will be logged with rationale, alternatives, and impact, ensuring traceability across PMP, SRS, TDD, and subsequent deliverables.
3. CI/CD Quality Gates – In the CI/CD pipeline, automated test suites will execute with each commit. Dashboards will be used to record results and assess coverage patterns, defect recurrence, and acceptance criteria compliance. If something goes wrong, escalations and variance reports will be started

11.3 Project Communications

11.2.1 Internal Meetings

The team will meet internally twice a week to discuss task status, quality assurance progress, and upcoming milestones. The Master Tracker shall retain a record of these meetings, as well as any actions that were discussed.

11.2.2 Internal-External Meetings

Weekly meetings with external stakeholders, such as faculty, mentors, and product owners, will be held with the goal of demonstrating incremental functionality in an isolated testing environment, updating progress, and validating requirements. For traceability, comments will be recorded and entered into the Decision Tracker.

11.2.3 Cross-Team Meetings

Cross-team communications will occur on an as-needed basis to address dependencies (e.g., shared architecture, DevOps pipelines, or client interactions). All outcomes from cross-team discussions will be summarized in the Decision Tracker and reported back to internal stakeholders.

11.2.4 Quality and Risk Reporting

Testing results will be connected to project reporting. A Defect Log will track the severity, status, and resolution time of all defects. The difference between expected and actual quality

metrics will be summarized in weekly dashboards. Assumptions that are invalidated during testing are escalated to the Risk Register for proactive mitigation.

11.4 Quality Plan

The CareConnect team will ensure that quality is built into all project activities and deliverables through structured reviews, automated checks, and compliance monitoring. Quality management will focus on process quality, product quality, and regulatory adherence.
All major deliverables (PMP, SRS, TDD, Test Plan) will undergo peer review for clarity, accuracy, and completeness. Issues will be logged in the Decision Tracker and tracked to closure.

Incremental deliverables will be validated against customer expectations and verified against requirements. Automated CI/CD quality gates will include unit testing, static code analysis, and security scans.

A Defect Log will track severity, ownership, and resolution time. Weekly dashboards will summarize defect trends, test coverage, and pass/fail metrics to guide corrective actions.
Requirements will be mapped to acceptance criteria in the RTM. No deliverable will be closed until all criteria are satisfied and test evidence recorded.
HIPAA and GDPR standards will guide data handling, encryption, and privacy. Compliance checkpoints will be conducted at each milestone.

12. Assumptions

The following assumptions are foundational to the CareConnect project plan. These assumptions serve to resolve the contradictions between legacy features **[4, 9]** and the current MVP constraints. Any disproven assumptions will be escalated to the Risk Register (Section 8).

12.1 Technical and Operational Assumptions

- **Device Compatibility:** It is assumed that all end-users (patients and caregivers) will access CareConnect using modern smartphones (iOS 15+ / Android 11+). This aligns with the exclusion of dedicated hardware or proprietary medical devices.
- **Language & Accessibility Scope Consolidation:** To ensure a feasible 12-week delivery, the MVP assumes an English-only user base. **Multilingual support, ASL translation, and advanced screen-reader compatibility are explicitly Out-of-Scope**, as noted in the refined Section 3.1.15.
- **Data Entry & API Consolidation:** It is assumed that medication and health logs will be manually entered. While **OpenFDA** is utilized for lookup validation, integration with third-party portals or wearables (Fitbit/Apple Health) is deferred to post-MVP phases to mitigate integration complexity.
- **Notification Delivery:** It is assumed that in-app and local system notifications are sufficient for the MVP. External SMS and Email gateways are Out-of-Scope to maintain the **$333/month infrastructure budget**.
- **BNS 5 Connectivity Assumptions:** It is assumed that users will experience intermittent connectivity. This necessitates BNS 5 offline persistence and synchronization for two

MVP modules (Mood & Symptom Tracking and Manual Calendar), with additional modules deferred until after codebase normalization.

- **Cloud Infrastructure Parity:** It is assumed that the **AWS-native managed stack (ECS Fargate, RDS, Bedrock)** provides sufficient performance parity with the legacy container-based prototype **[4, 9]** while offering superior cost-control.

13. Constraints

13.1 Technical and Project Constraints

- **Time Constraint (12-Week Lifecycle):** The entire development lifecycle, from documentation recovery to final delivery, must be completed within 12 weeks. This fixed timeline dictates the prioritization of core BNS features over the expansive feature list in the legacy documentation.
- **Fiscal Constraint (Cloud & Open Source):** Development must prioritize open-source tools and AWS managed services. All infrastructure expenditures are strictly capped at the **$1,320 infrastructure budget**. No proprietary third-party licenses are permitted.
- **Architectural Constraint (AWS-Native, Managed Compute):** The system must utilize AWS-native managed services and a container-based runtime (ECS Fargate) aligned with the Team B Technical Design Document. The architecture must support cost-efficient scaling and remain within the $1,320 infrastructure budget.
- **Regulatory Constraint (Compliance Deferment):** While the architecture implements **HIPAA-eligible** services (RDS, Cognito, Bedrock) and privacy-safe telemetry (BNS 7), formal third-party regulatory certification is deferred to the post-MVP production roadmap.
- **Technology Stack Constraint:** Development is strictly limited to the chosen stack (**Flutter/Dart** and **AWS-Native**). This prevents "technology sprawl" and ensures the 5-person team remains focused on a single implementation path.
- **Accessibility and ASL Constraint:** To ensure the delivery of functional offline mechanics (BNS 5), full conformity to WCAG 2.1 and **ASL translation features** are deferred. The UI will focus on high-contrast simplicity rather than specialized accessibility frameworks.

14. Work Breakdown Structure (WBS)

The CareConnect WBS organizes the project into manageable work packages, ensuring that all **BNS (5, 6, and 7)** are mapped to specific engineering tasks. This hierarchical decomposition accounts for the 1,200 total labor hours (20 hrs/week × 5 members × 12 weeks) budgeted in Section 9.1.

Table 14.1

WBS Dictionary and Work Packages

Phase	Work Package	Task ID	Deliverable / Goal
1.0 Discovery	1.1 Requirements Audit	1.1.1	Finalized SRS and MVP boundary; resolve [4, 9] contradictions.
	1.2 Tech Mapping	1.2.1	**BNS 6:** AWS CloudFormation templates and VPC architecture.
2.0 Infrastructure	2.1 Backend Services	2.1.1	Provisioning of RDS, S3, and **Amazon Bedrock** (AI) endpoints.
	2.2 Security Layer	2.2.1	IAM least-privilege roles, Cognito MFA, and encryption-at-rest.
3.0 Offline Sync	3.1 Local Persistence	3.1.1	**BNS 5:** SQLite/Drift schema for encrypted local storage.
(BNS 5)	3.2 Sync Mechanics	3.2.1	Conflict resolution logic for bidirectional cloud-sync.
	3.3 Core Modules	3.3.1	Offline-enablement for two selected MVP modules: **Mood & Symptom Tracking, and Calendar.** Framework supports future expansion.
4.0 AI & Analytics	4.1 Informational AI	4.1.1	Implementation of "Ask AI" inquiry feature via Bedrock.
	4.2 Telemetry	4.2.1	**BNS 7:** Privacy-safe telemetry pipeline with PII redaction.
5.0 Delivery	5.1 QA & Validation	5.1.1	End-to-end UAT and cross-platform mobile performance testing.
	5.2 Project Closure	5.2.1	**Definition of Done** validation and final technical guides.

15. Change Management Procedures

Any modifications to the CareConnect project plan—whether they involve scope updates, schedule shifts, or infrastructure resource changes—must follow a controlled change process. This ensures that BNS 5, 6, and 7 requirements are not compromised by ad hoc adjustments.

15.1 Initiation

Any team member who identifies a required change—such as a shift in CloudFormation templates (BNS 6) or a logic change in the Offline Sync engine (BNS 5)—must submit a standardized **Change Request (CR)**. The request must detail:

- **The Purpose:** Why the change is necessary.
- **Impact Analysis:** How it affects the 12-week timeline and the $1,320 budget.
- **Resource Requirements:** Which team member's hours will be redirected.

15.2 Review (Change Control Board)

The proposed change is reviewed by the Change Control Board (CCB), which consists of the Course Instructor, the Team Lead (Eduardo), and the relevant Lead Developer for the impacted module.

The board assesses:

1. **Risk Alignment:** Does this change introduce clinical liability or security risks?
2. **Infrastructure Impact:** Does this change deviate from the approved AWS-native runtime architecture (ECS/Fargate) or materially increase infrastructure cost?
3. **Schedule Slack:** Can the change be absorbed without pushing the Week 12 final delivery?

15.3 Authorization & implementation

Once approved, changes are documented in the test plan baseline, including who approved them and when. The updated plan is then communicated to all stakeholders. Approved changes are implemented in development and testing environments using version control, and validation is performed through a brief smoke test or regression cycle. For major rollouts, a partial or full test pass may be scheduled. All changes and their outcomes are logged for auditability and future review.

By following this structured yet adaptable change workflow, including initiation, review, and authorization, we will ensure the CareConnect testing remains aligned with project goals while minimizing disruption.

Table 15.2.1

Change Control Board Members

CCB Member	Project Role	CCB Responsibilities
Dr. Assadullah	Course Instructor / Client	Final approval or rejection of high-impact scope changes; validates alignment with course objectives.
Eduardo Estrada	Team Lead / Software Developer	Primary gatekeeper; determines if change requests are technically viable before full board review.

Joseph Wojcik	DevOps / Infrastructure Lead	Evaluates impact on **AWS CloudFormation (BNS 6)** and the $1,320 budget.
Yismaw Tilaye	System Analyst	Assesses impact on project requirements, SRS documentation, and **BNS 5/7** compliance.
CareConnect Team	Developers / QA (Gary, Chasity)	Provide technical analysis of CRs; estimate Level of Effort (LOE) and impact on the 12-week timeline.

15.4 Change Request Form

The CareConnect Change Request Form can be located in the Teams section of the Joint Collab Microsoft Teams channel linked here: CareConnect Change Request Form Template CareConnect Change Request Form Template. A figure has been provided below to illustrate what the template for the Change Request Form looks like for reference.

Figure 15.4.1

CareConnect Change Request Form Template

Project Name	CareConnect
Team	A
Date of Change Request Submitted	
Requested By	
Change Description	
Change Reason	
Impact of Change	
Proposed Action	
Status (Submitted, Approved, Rejected)	
Decision Reason	
Approval Date	
Approved By	

Note. CareConnect Change Request Form Template created with reference from previous 2023 SWEN 670 AlphaSoft project (AlphaSoft, 2023).

15.5 UI/UX wireframe preview

A detailed discussion of the CareConnect UI/UX wireframes can be found in the CareConnect Technical Design Document. The Technical Design Document is included in the overall group of documents delivered to the intended audience.

16. Appendices

16.1 Glossary and Terms

Table 16.1.1

Glossary Table

Term	Definition
AI (Artificial Intelligence)	The capability of a system to perform tasks that typically require human intelligence, such as recognizing speech, making decisions, or learning from data.
Android OS	An open-source mobile operating system developed by Google, used for running applications on smartphones, tablets, and other devices.
CareConnect	Application that will be created to help manage the care receiver healthcare needs
Care Giver	A person (e.g., family member, nurse, or aide) who provides assistance, monitoring, or support to a Care Receiver through the app.
Care Receiver	The individual who requires support, assistance, or monitoring, is the primary beneficiary of the app's features.
Dart Tool	The primary programming language and toolset used to develop Flutter applications.
Diarization	The process of partitioning an audio stream containing human speech into homogeneous segments according to the identity of each speaker.
Flutter	An open-source UI software development kit (SDK) created by Google, used for building natively compiled applications for mobile (Android, iOS), web, and desktop from a single codebase.
HIPAA (Health Insurance Portability and Accountability Act)	U.S. legislation that provides data privacy and security provisions for safeguarding medical information.
iOS	A mobile operating system created by Apple Inc. for iPhones and iPads, used to run applications developed specifically for Apple devices.
Machine Learning (ML)	A subset of AI that uses algorithms and statistical models to enable systems to improve performance on tasks through experience.
NLP (Natural Language Processing)	A branch of AI that enables computers to understand, interpret, and generate human language.
User Interface (UI)	The visual and interactive components of the application which allows users (care givers and receivers) to interact with the system.
User Experience (UX)	The overall experience and satisfaction a user has when interacting with the application, including ease of use, efficiency, and accessibility.

Table 16.1.2

Document Acronyms

Acronyms	Definitions
AC-###	Acceptance criterion tested for a requirement/use Case
AI	Artificial Intelligence
API	Application Programming Interface
App	Application (mobile or web-based software)
BNS	Business Need Statement
BFF	Backend for Frontend
DB	Database
EHR	Electronic Health Record
GUI	Graphical User Interface
HIPAA/GDPR	U.S. healthcare privacy law / EU privacy regulation
IoT	Internet of Things
IaC	Infrastructure as code
ML	Machine Learning
NLP	Natural Language Processing
PHI/PII	Protected Health Information
PII	Personally Identifiable Information
QA	Quality Assurance
RBAC	Role-Based Access Control
REQ-###	Requirement ID
REST)	Representational State Transfer (API standard
RTO/RPO	Recovery Time/Point Objective
SRS	Software Requirements Specification
TC-###	Test case that verifies an AC
UC-###	Use case
UI	User Interface
UX	User Experience
SLA/SLO	Service Level Agreement/Objectives

16.2 References

1. AlphaSoft. (2023, August 5). Short-Term Memory System (STeMS) project plan (Version 4.0) [Project documentation]. University of Maryland Global Campus.
2. Amazon Web Services. (2026). AWS pricing catalog. https://aws.amazon.com/pricing/
3. Apple Inc. (2025). Apple Developer Program FAQ. https://developer.apple.com/support/programs/
4. CareConnect Developer Team. (2025). CareConnect project documentation and programmer guides. GitHub Repository. https://github.com/umgc/2025_fall/tree/developer/careconnect2025/docs/project-docs
5. Gemini 3 Flash. (2026). *CareConnect System Architecture Diagram* [AI-generated image]. Google AI. https://gemini.google.com/
6. OpenAI. (2025). ChatGPT (May 30 version) [Large language model]. https://chat.openai.com/
7. PubNub. (n.d.). Configuration: Authentication key (Dart SDK API Reference). https://www.pubnub.com/docs/sdks/dart/api-reference/configuration#authentication-key
8. U.S. Bureau of Labor Statistics. (2024, May). *Occupational Employment and wage statistics*. Bureau of Labor Statistics. Retrieved January 23, 2026, from https://data.bls.gov/oes/#/industry/000000
9. Wagner, S., Lord, M., Rocha, L., Pingel, M., Maloney, M., Osterneck, A., & Adams, T. (2025). *CareConnect Project Plan Document 2025*. [Project documentation]. https://umgc-cappms.azurewebsites.net/download/c77df719-c5d0-42ef-ada8-7c87e8f2566b----SWEN670_CareConnect_2025_Fall2024_PojectPlan.pdf

Project Plan Document

CareConnect

University of Maryland Global Campus

SWEN 670 - Software Engineering Capstone

Dr. Mir Assadullah

January 18th, 2026

Contributors: Spring 2026 class team C.

Revision History 5
1. Executive Summary 6
1.1 Project Name 6
1.2 Purpose 6
1.3 Scope 6
1.4 Acronyms and Definitions 7
1.5 Software Development Methodology 10
2. Objectives and Goals 10
2.1 Goals 10
2.2 Objectives 10
3. Scope of Work 11
3.1 In-Scope 11
3.1.1 Migrate to New AWS Serverless Host 11
3.1.2 Design and Implement New Testing Norms 11
3.1.3 Redesign and Implement new Roll-Based Access Control 12
3.2 Out-of-Scope 12
3.2.1 Migrate to New AWS Serverless Host 12
3.2.2 Design and Implement New Testing Norms 12
3.2.3 Redesign and Implement new Roll-Based Access Control 12
4. Stakeholders 12
4.1 Internal Stakeholders 13
4.2 External Stakeholders 13
5.1 Timeline and Milestone Table 13
5.2 Gantt Chart 14
6. Technical Architecture 14
6.1 Architectural Objectives 14
6.2 Logical System Architecture 15
6.3 High-Level System Architecture Diagram 16
6.4 Architectural Rationale 17

6.5 Runtime Flows 18
6.6 Technology Stack Summary 18
6.7 Deployment Topology Highlights 19
6.8 Frontend 20
6.9 Backend 21
6.10 Database 22
6.11 APIs 24
6.11.1 Fitbit Web API 24
6.11.2 Apple HealthKit API 24
6.11.3 Google Health Connect API 24
6.11.4 Google Smart Device Management (SDM) API 25
6.11.5 Alexa Smart Home Skill API 25
6.11.6 OpenFDA API 25
6.12 AI Services 26
6.12.1 Purpose and User Flow 26
6.12.2 Technical Architecture 27
6.12.3 Security and Data Protection 27
6.12.4 Compliance and Governance 28
6.12.5 Summary and Future Enhancements 29
6.13 Hosting 29
6.13.1 Cloud Deployment Model 29
6.13.2 Containerization and Orchestration 30
6.13.3 Serverless-First & Scale-to-Zero 31
6.13.4 Security and Reliability 32
6.14 Security 33
6.14.1 Data Protection 33
6.14.2 Authentication and Access Control 34
6.14.3 Secure APIs 35
6.14.4 Infrastructure Hardening 35

6.14.5 Monitoring and Logging 36
6.14.6 Vulnerability Management 37
6.14.7 Compliance Measures 38

---

7. Project Deliverables 40
7.1 Planning and Management Deliverables 40
7.2 Architecture and Design Deliverables 40
7.4 Testing and Quality Assurance Deliverables 41
7.5 Documentation and Handoff Deliverables 41
7.6 Academic and Course Deliverables 41
7.7 Traceability and Validation 41
8. Risk & Mitigation 42
8.1 Risk Probability Matrix 42
8.2 Risk Identification, Probability, and Mitigation 42
9. Budget Estimate 43
9.1 Labor Cost 43
9.2 Materials Cost 44
9.3 Infrastructure Cost 44
9.3.1 AWS Core Services 45
9.3.2 Third-party / One-time Fees 45
9.3.3 Total Budget for This Phase 45
9.3.4 Cost-Control & Optimization Plan 46
9.3.5 Alignment to Financial Policies 46
9.3.6 Budget Summary for Project Plan 46
10. Roles & Responsibilities 46
10.1 Roles 47
10.2 RACI Matrix 48
11. Monitoring and Reporting 49
11.1 Intra-Team Monitoring and Reporting 49
11.2 Project Communications 50

12. Assumptions 50
13. Constraints 51
14. Work Breakdown Structure (WBS) 52
14.1 Structure 52
15. Change Management Procedures 53
15.1 Initiation 53
15.2 Review 54
15.3 Authorization and Implementation 54
15.4 Change Request Form 54
16. Appendices 54
16.1 References 54

Revision History

Team Name	Date	Reason for Changes	Version
CareConnect Team C Spring 2026	01/18/2026	Initial document submission.	1.0
CareConnect Team C Spring 2026	02/07/2026	Milestone 2 Revision	2.0

1. Executive Summary

The CareConnect project is a mobile-first, cloud-native application that bridges the gap between patients and caregivers by providing scheduling, health tracking, secure communication, and AI assisted tools for notes and telemedicine.

Our purpose is to deliver a viable, user-friendly platform that enhances healthcare coordination, supports caregiver efficiency, and empowers patients with autonomy while ensuring HIPAA/GDPR compliance.

This Project Management Plan (PMP) defines objectives, scope, architecture, risk, roles, and budget for CareConnect, aligning deliverables to academic rigor and industry standards (as cited in the CareConnect PMP, Aug 30, 2025).

1.1 Project Name

The name of this project is CareConnect.

1.2 Purpose

The purpose of this project is to develop a viable mobile application that provides a functional and user-friendly interface that connects caregivers and patients on a single platform.

The purpose of the CareConnect project is to design and develop a robust, intuitive mobile application that seamlessly connects caregivers and patients through a unified digital platform. This application aims to enhance the quality of care by enabling caregivers to efficiently monitor, organize, and manage the day-to-day health-related activities of their patients (as cited in the CareConnect PMP, Aug 30, 2025).

1.3 Scope

The scope of this new project is to enhance the existing CareConnect application with a set of three new distinct requirements; Deploy to a new production host, redesign and implement new testing norms, and redesign and implement a new structure for roll-based access control.

The development team will investigate different Amazon Web Services (AWS) offerings and choose a new cloud service to host the CareConnect application. Previous teams have narrowed the different possibilities to Fargate and App Runner. Once the team identifies the best option, they will begin migration to the new host.

Automated unit testing is vital to ensure the different pieces of a system are functioning as expected. Team C will work to understand the existing unit testing structure and design and implement improvements to streamline testing.

Team C will also redesign and implement a new roll-based access control system. The team will take time to understand the current system and implement changes to streamline it. This will provide an easier implementation of access controls to future development efforts, allowing teams to focus more on the feature and less on access c

1.4 Acronyms and Definitions

The definitions, acronyms, and abbreviations used in this document are specified in Table 1 & 2. Some acronyms and definitions were inherited from CareConnect SRS, Aug 30, 2025.

Table 1 Document Conventions

Term	Definition
AI (Artificial Intelligence)	A system capable of doing human like tasks involving pattern recognition, speech recognition, generating content
Amazon Web Services (AWS)	A suite of different cloud services offered by amazon including different database and server services needed for CareConnect.
Android OS	An open-source mobile operating system developed by Google.
App Runner	AWS service for hosting applications.
CareConnect	A web/mobile application developed by SWEN 670 cohorts to assist caregivers and patients with healthcare management.
Caregiver	An individual who assists with the care of an individual with some extraordinary needs.
Care Receiver	An individual with some extraordinary healthcare needs.
Dart Tool	Programming language and toolset used to create Flutter applications.
Fargate	AWS service for hosting applications.

Flutter	An open-source UI software development kit (SDK) created by Google, capable of building natively compiled applications for mobile, web, and desktop from a single codebase.
HIPAA (Health Insurance Portability and Accountability Act)	U.S. regulation intended to protect an individual's healthcare related information
Roll based Access Control	A way of controlling which screens and data a user has access to according to the roll they are assigned in the system.
iOS	A mobile operating system developed by Apple for iPhone
Unit testing	Refers to automated testing used in software development to help catch errors in compile time.
User Interface (UI)	The visual and interactive components of the application which allows users (Caregivers and receivers) to interact with the system.
User Experience (UX)	The overall experience and satisfaction a user has when interacting with the application, including ease of use, efficiency, and accessibility.

Table 2 Acronyms

Acronyms	Definitions
AC-###	Acceptance criterion tested for a requirement/use case
AI	Artificial Intelligence
API	Application Programming Interface
App	Application (mobile or web-based software)
AWS	Amazon Web Services

BNS	Business Need Statement
DB	Database
GUI	Graphical User Interface
HIPAA/GDPR	U.S. healthcare privacy law / EU privacy regulation
IoT	Internet of Things
MVP	Minimum Viable Product
PHI/PII	Protected Health Information / Personally Identifiable Information
PMP	Project Management Plan
QA	Quality Assurance
RBAC	Role-Based Access Control
RDS	Relational Database Service
REQ-###	Requirement ID
REST	Representational State Transfer (API standard)
SRS	Software Requirements Specification
TC-###	Test case that verifies an AC
UC-###	Use case
UI	User Interface
UX	User Experience
SLA/SLO	Service Level Agreement/Objectives
MA	Medicaid Number

EOR	Employer of Record

1.5 Software Development Methodology

For this development effort, Team C will follow an Agile development methodology. This will allow the team greater flexibility to absorbed change requests approved by the CCB. The development effort will be organized into two-week sprints with weekly meetings. The backlog will be ordered in priorities agreed upon by the stakeholders and the development team.

2. Objectives and Goals

This section defines the strategic direction of the project and the outcomes it aims to achieve.

2.1 Goals

The goal of the CareConnect application is to provide a user-friendly interface to users in different role-based categories to assist in the care of individuals with extraordinary health situations. Compliance with healthcare related regulations (HIPAA/GDPR) is of utmost importance for this to be a viable application.

The CareConnect application MVP has already been created with a lot of its core components in place. The purpose of this project is to continue the work of past development teams with three new distinct requirements including investigate a new cloud-based hosting service for the application in AWS, design and implement new unit testing norms for more streamlined testing, and design and implement a new streamlined system for role-based access control.

2.2 Objectives

The requirements for this project are centered around efficiency gains and cost savings.

New unit testing norms and a new more streamlined role-based access control system will save future development teams time and effort by making these two tasks easier to implement for new development. Another gain from better unit testing will be more reliable for test results, allowing teams to more quickly identify issues and correct them. A more streamlined role-based access system will shorten the access control part of future tasks, giving teams more time to focus on the feature itself.

The goal of a new cloud host has several potential benefits. First, the host could be cheaper than what is used by CareConnect now.

Another benefit when going with a more flexible host like Fargate is the option to in the future containerize the whole backend, including the database potentially eliminating the need for an additional cloud service like RDS. A major requirement of this project is to investigate and select the best option from AWS for hosting to realize these potential cost savings.

3. Scope of Work

3.1 In-Scope

3.1.1 Migrate to New AWS Serverless Host

The development team shall research different serverless options offered by AWS to host the CareConnect application. Once the team has chosen the best option, they will take the necessary steps to deploy CareConnect to the new host.

The short list of options provided to Team C are AWS Fargate and App Runner. The following criteria will be used to evaluate the options:

- Cost
- Flexibility
- Maintenance level of effort

After considering all three of these criteria Team C has identified Fargate as the better option. While App Runner has a smaller level of effort to maintain, Fargate provides better flexibility through control over the dockerfile and is far cheaper than App Runner after additional fees outside of compute time associated with having an App Runner instance.

3.1.2 Design and Implement New Testing Norms

The development team shall design and implement new more streamlined testing norms for existing front-end and back-end code, which future teams will also implement for their development effort. This will enable more efficient unit testing implementation and more reliable results.

Currently, the CareConnect application has unit test however there is no framework for mock data implemented. This caused the original development teams to rely on third party services to run unit test which do not work in this context. This problem caused the test to just fail which lead to the unit tests being skipped all together when building the backend package.

3.1.3 Redesign and Implement new Roll-Based Access Control

The development team shall design and implement a new more streamline centralized system for handling roll-based access control. This will include handling access control on the front-end of the application for controlling which pages are available to different types of users in addition to locking down end points in the back-end to ensure users only have access to data appropriate to their role. At the conclusion of this phase the business logic that processes roll-based access control should be centralized into classes with methods exposed for the component level to use.

To this point the CareConnect application relies on decentralized business logic at the component level for roll-based access control. This adds code complexity and a lot of maintenance overhead whener something changes with roll-based access control.

3.2 Out-of-Scope

3.2.1 Migrate to New AWS Serverless Host

A recommendation for future scope Team C has that is not in the scope of these requirements is to investigate containerizing a production version of CareConnect complete with the front-end, back-end, and database. This would streamline the production deployment process and remove extra services and costs.

3.2.2 Design and Implement New Testing Norms

Team C shall not implement unit test cases following the new norms or flow for any development effort outside of their own occurring now or in the future. All necessary information needed to set up unit testing will be communicated to the other development teams for them to set up their own unit tests.

3.2.3 Redesign and Implement new Roll-Based Access Control

Team C shall not implement the new roll-based access control system for any development effort outside their own occurring now or in the future. All necessary information needed to set up roll-based access control will be communicated to the other development teams, so they can set it up for their endpoints and views.

---

4. Stakeholders

This section identifies all individuals and groups with an interest in or influence over the project.

4.1 Internal Stakeholders

1. Dr. Assadullah
2. Colyn Mahoney
3. Obeng Buo
4. Ja'Lisa Hawkins
5. Henry Stewart
6. Zechariah Hillman

4.2 External Stakeholders

1. Roy Gordon
2. Professor Minagar
3. Sukhmeet Khalar
4. Sudarshan Neupane5. Timeline & Milestone

This section outlines the project schedule and key delivery checkpoints.

5.1 Timeline and Milestone Table

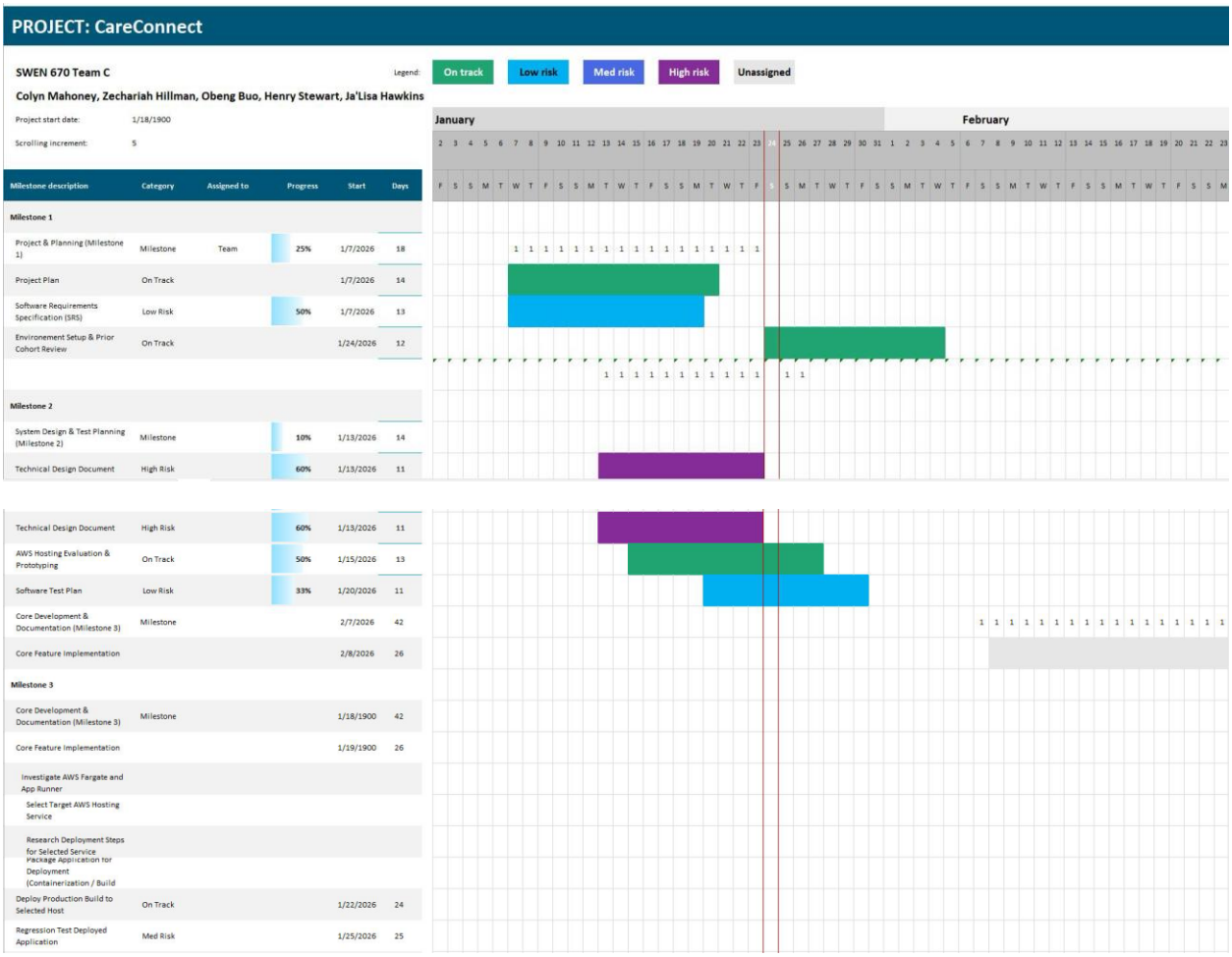
Tabular view of project phases, deliverables, and due dates:

Table 3 Milestones

Phase / Milestone	**Primary Deliverables**	**Due Date**
Milestone 1	• Project Plan • Software Requirements Specification (SRS)	Jan 24, 2026
Milestone 2	• Technical Design Document • Software Test Plan	Feb 7, 2026
Milestone 3	• Programmer Guide • Deployment and Operations Guide	Mar 21, 2026
Milestone 4	• User Guide • Test Report	Mar 31, 2026
Ongoing Activities	• Weekly Peer Evaluations • Milestone Presentations	Weekly / Per Milestone

5.2 Gantt Chart

Visual representation of the project schedule, dependencies, and task sequencing.



---

6. Technical Architecture

This section describes the system's architectural design and technical decisions.

6.1 Architectural Objectives

Scalability

The system must be able to handle growing numbers of users across multiple role-based categories without degradation in performance. Hosting the application on AWS, with options such as Fargate or containerized deployments, allows dynamic scaling based on demand.

Security

Protecting sensitive health information is a critical priority for CareConnect. The architecture includes multiple security measures to safeguard data, including:

- **Role-Based Access Control (RBAC):** Ensures users can only access data relevant to their assigned roles.
- **Encryption:** All data is encrypted both **in transit** (using TLS/SSL) and **at rest** (using database and storage encryption mechanisms).
- **Authentication and Authorization:** Strong mechanisms are in place to verify user identity and enforce permissions.
- **Monitoring and Logging:** All access and system activity is logged to detect and respond to potential security threats.

Reliability

The system should provide high uptime and fault tolerance. By leveraging AWS cloud services, load balancing, and containerized microservices (if applicable), the system can recover gracefully from failures.

Compliance

CareConnect is designed to comply with healthcare regulations to protect user data and maintain trust, including:

- **HIPAA (Health Insurance Portability and Accountability Act):** Ensures the confidentiality, integrity, and availability of protected health information (PHI).
- **GDPR (General Data Protection Regulation):** Protects the personal data of EU users and ensures proper handling of sensitive information.

6.2 Logical System Architecture

The CareConnect system is organized into the following logical components:

Application Layer (Backend Services)

- Implements core business logic, including health data management, notifications, and user workflows
- Handles role-based access control and enforces security policies
- Provides endpoints for unit testing and integration with automated test suites.

Data Layer

- Manages storage of user data, health records, logs, and audit trails
- Utilizes secure databases hosted on AWS, potentially containerized in future releases to reduce dependency on additional cloud services.

Infrastructure Layer

- Includes cloud hosting on AWS using services like Fargate or App Runner, load balancers, and container orchestration
- Handles scaling, monitoring, backups, and deployment automation.

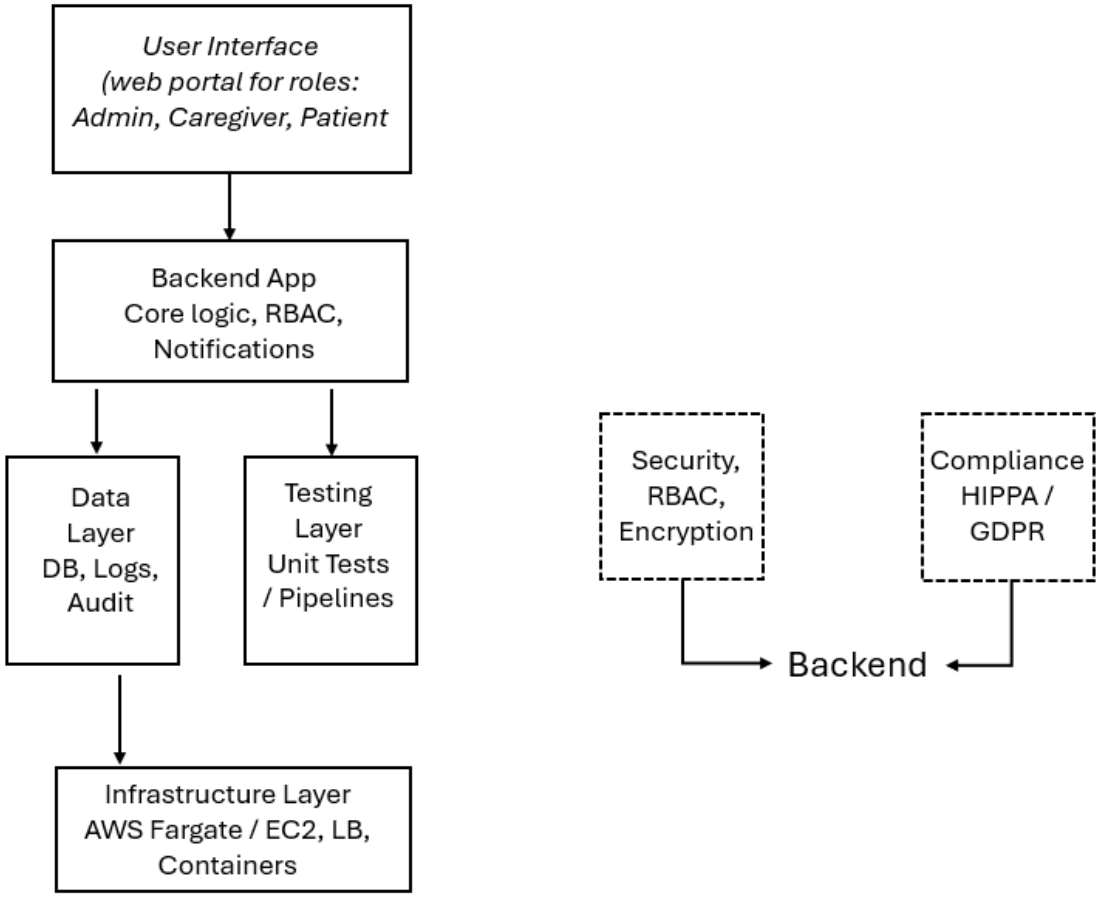
Testing Layer

- Integrates unit testing norms and automated test pipelines
- Ensures system reliability and quick identification of errors.

6.3 High-Level System Architecture Diagram

Visual diagram illustrating major system components and data flow.

The diagram shows CareConnect's architecture, where users interact with the Web UI, the Backend manages logic and access control, data is securely stored, automated testing ensures reliability, and AWS infrastructure supports scalability, all under strict Security and Compliance controls.



6.4 Architectural Rationale

Explanation of why specific architectural choices were made.

The architectural decisions for CareConnect were made based on the goals and objectives of the project:

1. Cloud Hosting on AWS:
   Selected to provide flexibility, scalability, and cost savings. Options such as Fargate allow potential future containerization, which could consolidate backend services and eliminate dependency on separate database services like RDS.
2. Role-Based Access Control:
   A new, streamlined access control system ensures users only see data relevant to their role, increasing security and compliance while simplifying development and maintenance.

3. Unit Testing Norms:
   Standardized testing ensures reliability of new features, faster error detection, and easier onboarding for future development teams.
4. Separation of Concerns:
   Logical layering (UI, Application, Data, Infrastructure) allows each component to evolve independently, improving maintainability and extensibility.
5. Security and Compliance:
   Encryption, strict access controls, and adherence to HIPAA/GDPR were included as non-negotiable architectural requirements to protect sensitive health data.

6.5 Runtime Flows

- **Payment Flow**:
  A **patient** initiates a payment through the app → **API Gateway** routes the request to **Billing Service** → **Stripe** processes the payment → Confirmation is sent back to the user via **RDS commit** and marked as "Active".
- **Real-time Vital Update**:
  Fitbit syncs data → **Lambda** processes the sync and stores data in **Timestream** → **Analytics Service** publishes the event → The data is pushed to the client in real-time via **WebSocket** (under 5 seconds).
- **Nightly Batch Processing**:
  EventBridge triggers a **Glue** job at 2:00 AM UTC → Data is aggregated into **Timestream** → **QuickSight** is refreshed with updated metrics.

(as referenced in the CareConnect Project Plan, July 26, 2025)

6.6 Technology Stack Summary

The **CareConnect** technology stack includes a range of industry-leading technologies:

- **Frontend**:
  Flutter 3.x (Dart) supports mobile and web interfaces, ensuring cross-platform compatibility.
- **Backend**:
  Spring Boot 3 with **Java 17** provides a robust, secure backend. **Micrometer** and **Resilience4j** are used for metrics and fault tolerance.
- **Data**:
  RDS (PostgreSQL), **DynamoDB**, and **Timestream** provide scalable and reliable data storage.

- **Messaging**:
  MSK (Kafka 3.x) is used for event streaming and data transfer between services.
- **CI/CD**:
  GitHub Actions for continuous integration and deployment via **AWS CodeBuild** and **CodeDeploy** with **Blue-Green** deployments.
- **Security**:
  AWS Cognito, **KMS**, **WAF**, and **IAM** ensure robust security controls across the system.

(as referenced in the CareConnect Project Plan, July 26, 2025)

6.7 Deployment Topology Highlights

The CareConnect application is deployed using a cloud-native architecture on Amazon Web Services, leveraging serverless and containerized components to ensure scalability, reliability, and cost efficiency. The deployment topology is designed to support the needs of caregivers and patients while maintaining strict compliance with healthcare regulations.

Production Environment Architecture:

d single-AZ deployments to reduce costs while maintaining architectural consistency

Frontend Deployment: The Flutter web application is hosted on AWS Amplify or CloudFront with S3, providing global content delivery with low latency. Mobile applications are distributed through Apple App Store and Google Play Store.

Backend Services: Spring Boot microservices are containerized and deployed on AWS Fargate or App Runner, providing serverless compute capacity with automatic scaling based on demand. API Gateway serves as the entry point for all client requests, handling routing, throttling, and authentication.

Database Layer: Production data is stored in Amazon RDS for PostgreSQL with Multi-AZ deployment for high availability. DynamoDB provides NoSQL storage for session data and real-time updates. Amazon Timestream stores time-series health metrics from connected devices.

Event Streaming: Amazon MSK (Managed Streaming for Kafka) handles real-time event processing and data synchronization between microservices.

Load Balancing: Application Load Balancers distribute incoming traffic across multiple container instances, with health checks ensuring only healthy instances receive requests.

Availability Zones: All critical components are deployed across multiple AWS availability zones to ensure resilience against infrastructure failures.

Development and staging environments mirror the production topology but use smaller instance sizes an

6.8 Frontend

Description of client-side technologies, frameworks, and responsibilities.

The CareConnect frontend is built using Flutter 3.x, which enables a single codebase to deliver native experiences across iOS, Android, and web platforms. This approach ensures consistency in user experience while reducing development and maintenance overhead.

Key Frontend Technologies:

Framework: Flutter 3.x with Dart programming language provides a reactive UI framework with excellent performance and native compilation capabilities.

State Management: Provider or Riverpod manages application state, ensuring efficient data flow and reactivity throughout the user interface.

Authentication: Integration with AWS Cognito provides secure user authentication with support for multi-factor authentication and social identity providers.

API Communication: HTTP client libraries facilitate secure communication with backend REST APIs over HTTPS, with automatic token refresh and request retry logic.

Real-time Updates: WebSocket connections enable push notifications and live data updates for vital signs, messages, and appointment changes.

Offline Support: Local data persistence using SQLite or Hive allows limited functionality when network connectivity is unavailable.

Accessibility: WCAG 2.1 compliance ensures the application is usable by individuals with disabilities, including screen reader support and keyboard navigation.

Frontend Responsibilities:

User Interface Rendering: Displays schedules, health metrics, medication lists, and communication interfaces tailored to caregiver and patient roles.

Client-side Validation: Performs initial input validation before sending data backend services, improving user experience, and reducing server load.

Role-based UI Rendering: Dynamically adjusts visible screens and features based on user roles retrieved from the authentication token.

Session Management: Maintains user sessions securely and handles automatic logout after periods of inactivity.

Error Handling: Provides clear error messages and guidance when issues occur, with automatic retry for transient failures.

6.9 Backend

The CareConnect backend is built on Spring Boot 3 with Java 17, providing a robust, secure, and scalable foundation for the application's business logic and data processing. The backend follows microservices architecture principles, with each service responsible for a specific domain of functionality.

Core Backend Services:

User Service: Manages user accounts, authentication, and authorization. Integrates with AWS Cognito for identity management and enforces role-based access control policies.

Scheduling Service: Handles appointment creation, modification, and cancellation. Sends notifications and reminders to caregivers and patients.

Health Tracking Service: Processes and stores health metrics from connected devices and manual entries. Provides analytics and trend analysis.

Communication Service: Manages secure messaging between caregivers and patients, with support for text, images, and file attachments.

Medication Service: Tracks medication schedules, refill reminders, and adherence reporting.

Billing Service: Processes payments through Stripe integration, manages invoices, and handles subscription tiers.

Notification Service: Delivers push notifications, emails, and SMS messages using Amazon SNS and SES.

Analytics Service: Aggregates data for reporting and dashboard visualizations using Amazon QuickSight.

Backend Technical Capabilities:

RESTful API: Exposes well-defined REST endpoints following OpenAPI specifications for client integration.

Business Logic Implementation: Enforces all business rules, data validation, and workflow orchestration.

Data Access Layer: Uses Spring Data JPA for relational data access and AWS SDK for NoSQL and time-series databases.

Event Processing: Consumes and publishes events via Kafka for asynchronous communication between services.

Resilience Patterns: Implements circuit breakers, retries, and timeouts using Resilience4j to handle service failures gracefully.

Observability: Integrates with Micrometer for metrics collection, distributed tracing with AWS X-Ray, and structured logging with CloudWatch Logs.

Security: Enforces endpoint-level authorization, validates all inputs, and sanitizes outputs to prevent injection attacks.

6.10 Database

CareConnect employs a polyglot persistence strategy, utilizing multiple database technologies optimized for specific data access patterns and use cases. This approach balances performance, scalability, and cost while maintaining data integrity and security.

Primary Database Systems:

Amazon RDS for PostgreSQL: Serves as the primary relational database for structured data including user profiles, appointments, medications, and care plans. Configured with Multi-AZ deployment for high availability and automated backups with point-in-time recovery. Encryption at rest is enabled using AWS KMS, and all connections use TLS encryption. Database credentials are managed through AWS Secrets Manager with automatic rotation.

Amazon DynamoDB: Provides low-latency NoSQL storage for session data, user preferences, and real-time notification queues. Auto-scaling adjusts read and write capacity based on demand. DynamoDB Streams enables event-driven processing for data changes.

Amazon Timestream: Stores time-series health metrics from connected devices such as Fitbit, Apple HealthKit, and Google Health Connect. Optimized for high-throughput ingestion and efficient querying of temporal data. Data retention policies automatically archive older data to reduce storage costs.

Database Schema Design:

Normalized Relational Schema: PostgreSQL tables follow third normal form to eliminate data redundancy while maintaining referential integrity through foreign key constraints.

Audit Tables: All sensitive data modifications are tracked with audit tables recording who made changes, when, and what was changed.

Soft Deletes: Critical records are never physically deleted but marked as inactive to maintain historical integrity.

Data Partitioning: Large tables are partitioned by date ranges to improve query performance and simplify data archival.

Data Management Practices:

Backup and Recovery: Automated daily backups with 30-day retention. Transaction logs enable point-in-time recovery to any moment within the retention period. Backup testing occurs quarterly to validate recovery procedures.

Data Encryption: All databases use encryption at rest with AWS KMS managed keys. Data in transit is encrypted using TLS 1.2 or higher.

Access Control: Database access is restricted to authorized services using IAM roles and security groups. Direct database access by developers is prohibited in production environments.

Performance Optimization: Database indexes are strategically placed on frequently queried columns. Query performance is monitored using Performance Insights, with slow queries automatically flagged for optimization.

Data Retention: PHI and PII are retained according to HIPAA requirements and purged securely when retention periods expire.

6.11 APIs

CareConnect integrates with multiple third-party APIs to provide comprehensive health tracking, smart home integration, and medication information. These integrations extend the platform's capabilities while maintaining security and compliance sta ndards.

6.11.1 Fitbit Web API

The Fitbit Web API enables CareConnect to retrieve health and activity data from Fitbit devices, including steps, heart rate, sleep patterns, and exercise sessions. Integration occurs through OAuth 2.0 authentication, with users granting explicit permission for data access. Data is synchronized at regular intervals and stored in Amazon Timestream for trend analysis. Real-time updates are pushed via webhooks when significant health events occur, such as irregular heart rate patterns. All data transfers are encrypted in transit and comply with HIPAA requirements through a Business Associate Agreem ent with Fitbit.

6.11.2 Apple HealthKit API

Apple HealthKit API provides access to health data stored on iOS devices, including vital signs, workout data, nutrition tracking, and clinical records. CareConnect uses the HealthKit framework within the iOS application to request specific data types with user authorization. Data is synced locally on the device and transmitted securely to CareConnect's backend when the user is authenticated. Integration respects Apple's privacy requirements, requesting only necessary health data types and providing clear explanations of how data will be used. The iOS app implements background synchronization to keep health data current without requiring manual user intervention.

6.11.3 Google Health Connect API

Google Health Connect API aggregates health and fitness data from multiple Android applications and wearable devices. CareConnect integrates with Health Connect to access standardized health metrics across the Android ecosystem, including data from Google Fit, Samsung Health, and other compatible apps. Users grant permissions through Health Connect's centralized interface, maintaining control over which data types are shared. The Android application reads data using Health Connect SDK and synchronizes it with the backend in

batches to optimize battery life and network usage. All data exchanges follow Google's health data requirements and maintain end-to-end encryption.

6.11.4 Google Smart Device Management (SDM) API

The Google Smart Device Management API enables CareConnect to integrate with Google Nest devices for environmental monitoring and safety features. This integration allows caregivers to monitor temperature, humidity, and detect smoke or carbon monoxide alerts in the care receiver's home. API access is granted through OAuth 2.0 with user consent, and device data is retrieved via RESTful endpoints. CareConnect subscribes to event notifications for critical alerts such as smoke detection or unusual temperature changes, triggering immediate caregiver notifications. Device commands can be issued through the API to adjust thermostats or activate cameras when authorized by the care receiver.

6.11.5 Alexa Smart Home Skill API

The Alexa Smart Home Skill API provides voice-based interaction capabilities for CareConnect users with Amazon Echo and Alexa-enabled devices. The custom skill allows care receivers to check medication schedules, confirm appointments, request caregiver assistance, and receive health reminders through natural voice commands. Account linking connects Alexa to the user's CareConnect account using OAuth 2.0, ensuring secure authentication. The skill implements intent handlers for common caregiving tasks and integrates with CareConnect's backend APIs to retrieve real-time information. Voice interactions are logged for audit purposes while respecting privacy requirements, with options to disable voice recording if desired.

6.11.6 OpenFDA API

The OpenFDA API provides access to public data from the U.S. Food and Drug Administration, including drug labels, adverse event reports, and recall information. CareConnect queries this API to provide medication information, including usage instructions, side effects, and interactions when users add medications to their profiles. The API is accessed without authentication as it provides public data, with rate limiting handled through request throttling and caching. Medication recall alerts are checked daily against the user's medication list, with caregivers notified immediately if any prescribed medications appear in recall databases. Drug interaction checking uses OpenFDA data combined with clinical algorithms to identify potentially dangerous medication combi nations.

6.12 AI Services

CareConnect leverages artificial intelligence to enhance caregiver productivity and improve patient outcomes through intelligent automation and predictive analytics. AI features are designed with transparency and user control as primary considerations, ensuring caregivers remain the ultimate decision-makers in care delivery.

6.12.1 Purpose and User Flow

AI services in CareConnect serve three primary purposes: documentation assistance, health monitoring, and care coordination optimization.

Documentation Assistance: Caregivers can dictate visit notes using speech-to-text powered by Amazon Transcribe Medical, which is specifically trained on medical terminology. The AI automatically structures the transcription into standard care note formats, extracting key information such as vital signs, symptoms, and care activities. Caregivers review and edit the generated notes before saving, maintaining full control over documentation accuracy.

Health Monitoring: Machine learning models analyze time-series health data to detect anomalies and trends that might indicate deteriorating health conditions. For example, gradual increases in resting heart rate combined with decreased activity levels may trigger early intervention alerts. The system highlights concerning patterns for caregiver review rather than making autonomous medical recommendations.

Care Coordination: Natural language processing analyzes appointment notes and medication schedules to suggest optimal visit timing and identify potential conflicts. The AI can recommend when medication refills will be needed based on prescription history and usage patterns, helping caregivers plan ahead.

Telemedicine Support: During virtual visits, real-time transcription and note-taking assistance allow caregivers to focus on patient interaction rather than documentation. The AI can surface relevant medical history during consultations, providing context without interrupting the conversation flow.

6.12.2 Technical Architecture

The AI architecture utilizes AWS managed services to minimize operational complexity while maintaining HIPAA compliance. All AI components operate within the secure AWS environment with encrypted data pipelines.

Core AI Components:

- Amazon Transcribe Medical: Processes audio recordings of care notes and visit summaries, converting speech to text with medical terminology accuracy. Results are post-processed to structure content into standardized note templates.
- Amazon Comprehend Medical: Extracts medical entities such as medications, dosages, symptoms, and diagnoses from unstructured text in notes and messages. Identifies relationships between entities to build structured data from narrative documentation.
- Amazon SageMaker: Hosts custom machine learning models trained on historical health data to predict health deterioration risks. Models are trained on de-identified aggregate data and regularly retrained to improve accuracy.
- AWS Lambda: Executes AI processing workflows in response to events such as new health readings or note creation. Serverless execution ensures cost efficiency and automatic scaling.
- Amazon EventBridge: Orchestrates AI workflows by routing events to appropriate processing functions based on data type and user preferences.

Data Pipeline: Health data and care notes are temporarily stored in S3 with encryption before AI processing. Lambda functions retrieve data, invoke AI services, and store structured results in DynamoDB for real-time access and RDS for long-term analysis. All temporary data is deleted after processing, with retention limited to the minimum necessary for model improvement.

6.12.3 Security and Data Protection

AI operations handle highly sensitive protected health information, requiring stringent security measures at every processing stage.

- Data Encryption: All data is encrypted in transit using TLS 1.2 or higher and at rest using AWS KMS with customer-managed keys. AI services are configured to use HIPAA-eligible endpoints with Business Associate Agreements in place.
- Access Control: AI processing functions operate with least-privilege IAM roles, accessing only the specific data required for their function. No human access to raw PHI is permitted during AI processing.

- Data Minimization: Only necessary data elements are sent to AI services. For example, transcription only receives audio without accompanying patient identifiers unless required for medical context.
- Audit Logging: All AI processing activities are logged to CloudWatch with details of what data was processed, which services were invoked, and who initiated the processing. Logs are retained for compliance auditing.
- De-identification: Data used for model training is de-identified according to HIPAA Safe Harbor method, removing all 18 categories of identifiers before aggregation for analytics.
- User Consent: Users explicitly opt-in to AI features with clear explanations of how their data will be processed. AI can be disabled at any time, with all AI-generated content retained or deleted per user preference.

6.12.4 Compliance and Governance

AI features are governed by strict policies ensuring ethical use, regulatory compliance, and transparency.

- HIPAA Compliance: All AI services are configured in HIPAA-eligible mode with executed Business Associate Agreements. Regular risk assessments validate that AI processing meets HIPAA Security Rule requirements.
- GDPR Compliance: For EU users, AI processing is documented in the Article 30 record of processing activities. Users can exercise their right to explanation for AI decisions and opt out of automated processing.
- Algorithmic Transparency: AI suggestions display confidence scores and explain the data points that influenced recommendations. Users can view which inputs led to specific outputs.
- Bias Monitoring: Model performance is regularly evaluated across demographic groups to detect and mitigate potential bias. Training data is reviewed to ensure representative coverage of patient populations.
- Human Oversight: All AI outputs are clearly labeled as machine-generated and require caregiver review before being acted upon. The system never makes automated clinical decisions without human approval.
- Model Governance: Machine learning models undergo version control and testing before deployment. Model changes are documented with performance metrics and approved through change control processes.

6.12.5 Summary and Future Enhancements

Current AI capabilities focus on reducing documentation burden and surfacing actionable health insights while maintaining caregiver control. The system successfully processes thousands of care notes monthly, saving caregivers an average of 15 minutes per visit in documentation time.

Planned enhancements for future releases include:

- Predictive Analytics: Advanced models to predict hospital readmission risk and identify patients who may benefit from increased intervention intensity.
- Conversational AI: Natural language chatbot to answer common patient questions about medications, appointment scheduling, and general health information, reducing caregiver workload for routine inquiries.
- Image Analysis: Computer vision to analyze photos of wounds, skin conditions, or medication packaging for tracking healing progress and verifying medication adherence.
- Care Plan Optimization: Recommendation engine to suggest evidence-based care interventions based on patient conditions and historical outcomes.
- Multi-language Support: Real-time translation of care notes and patient communications to support caregivers and patients who speak different languages.

All future AI enhancements will maintain the core principles of transparency, user control, and rigorous privacy protection that guide current implementations.

6.13 Hosting

CareConnect's hosting infrastructure leverages AWS cloud services to provide scalable, reliable, and cost-effective application delivery. The architecture prioritizes serverless and containerized deployment models to minimize operational overhead while maintaining performance and security standards.

6.13.1 Cloud Deployment Model

CareConnect is deployed entirely on Amazon Web Services using a multi-region architecture with primary operations in US East and failover capabilities in US West. The cloud deployment model provides several strategic advantages:

- Elastic Scalability: Infrastructure automatically scales to meet demand fluctuations, handling peak usage during business hours, and scaling down during off-peak periods to optimize costs.
- Geographic Distribution: CloudFront CDN serves static content from edge locations worldwide, reducing latency for users regardless of physical location.
- Managed Services: AWS handles underlying infrastructure maintenance, security patching, and upgrades, allowing the development team to focus on application features.
- Cost Optimization: Pay-per-use pricing model means costs scale with actual usage rather than requiring fixed capacity provisioning. Reserved instances and Savings Plans reduce costs for predictable baseline loads.
- Compliance Certifications: AWS maintains HIPAA eligibility, SOC 2, and ISO 27001 certifications, providing a compliant foundation for healthcare applications.

Deployment Environments: Three environments support different stages of the software lifecycle. Development environment provides individual developer workspaces with full infrastructure stack for testing. The staging environment mirrors production configuration for pre-release validation and load testing. The production environment serves live users with redundancy across availability zones and automated failover.

6.13.2 Containerization and Orchestration

The CareConnect backend is containerized using Docker, with container images built during the CI/CD pipeline and stored in Amazon Elastic Container Registry. Containerization provides consistent runtime environments across development, staging, and production, eliminating environment-specific bugs.

Container orchestration is handled by AWS Fargate, which provides serverless container execution without managing underlying EC2 instances. This investigation into Fargate versus App Runner is a primary objective of the current project phase.

Key containerization benefits:

- Service Isolation: Each microservice runs in its own container with dedicated resources and dependencies, preventing conflicts and simplifying troubleshooting.
- Resource Efficiency: Container-level resource allocation ensures optimal CPU and memory usage for each service based on actual demand.
- Rapid Deployment: Container images are immutable and versioned, enabling quick rollbacks and blue-green deployments to minimize downtime.

- Developer Productivity: Developers can run exact production container images locally, reducing environment discrepancies and speeding debugging.

Health checks monitor container status, automatically replacing unhealthy instances. Task definitions specify CPU and memory requirements, network configuration, and environment variables. Service auto-scaling adjusts container counts based on CPU utilization and request rates.

6.13.3 Serverless-First & Scale-to-Zero

CareConnect adopts a serverless-first philosophy where appropriate, utilizing AWS Lambda for event-driven processing and AWS Fargate or App Runner for containerized services. This approach delivers significant cost advantages by eliminating idle capacity costs.

Serverless components include:

- AWS Lambda Functions: Process events such as new health data ingestion, notification delivery, scheduled report generation, and image processing. Functions are triggered by API Gateway, S3 events, DynamoDB streams, and EventBridge schedules. Cold start optimization uses provisioned concurrency for latency-sensitive functions.
- API Gateway: Provides serverless API endpoint management with automatic scaling, request throttling, and API key management. WebSocket APIs enable real-time bidirectional communication for notifications and live updates.
- AWS Step Functions: Orchestrate complex multi-step workflows such as user onboarding, data migration, and batch processing jobs with automatic retry and error handling.
- Fargate/App Runner: Container services scale to zero during periods of no traffic in non-production environments, reducing costs while maintaining instant availability when requests arrive.

The serverless architecture delivers cost efficiency through granular per-request billing rather than continuous server rental. Development and staging environments incur minimal costs outside active testing hours. Production scales automatically during usage spikes without manual intervention or cap acity planning.

6.13.4 Security and Reliability

Hosting infrastructure incorporates multiple layers of security and reliability controls to protect patient data and ensure continuous availability.

Security Measures:

- Network Isolation: Application components run in private subnets within Amazon VPC, accessible only through controlled entry points. Security groups restrict traffic to only necessary ports and protocols.
- DDoS Protection: AWS Shield Standard provides automatic protection against common network attacks. CloudFront and Route 53 absorb attack traffic before it reaches application infrastructure.
- Web Application Firewall: AWS WAF rules protect against SQL injection, cross-site scripting, and other OWASP Top 10 vulnerabilities. Rate limiting prevents abuse and credential stuffing attacks.
- Secrets Management: AWS Secrets Manager stores database credentials, API keys, and encryption keys with automatic rotation. Applications retrieve secrets at runtime rather than storing them in configuration files.
- Patch Management: Containerized applications receive security updates through regular base image rebuilds. AWS managed services apply security patches automatically without service disruption.

Reliability Measures:

- Multi-AZ Deployment: Critical services run across multiple availability zones with automatic failover. Database replication ensures data availability even if an entire data center fails.
- Health Monitoring: CloudWatch monitors application and infrastructure metrics with automated alerts for anomalies. Target tracking auto-scaling responds to traffic patterns and resource utilization.
- Load Balancing: Application Load Balancers distribute traffic across healthy instances with connection draining during deployments. Health checks remove unhealthy targets automatically.
- Disaster Recovery: Automated backups occur daily with cross-region replication for critical data. Recovery procedures are tested quarterly to validate RTO and RPO targets.

- Incident Response: CloudWatch alarms notify on-call engineers of critical issues through PagerDuty integration. Runbooks document response procedures for common failure scenarios.

The hosting architecture achieves a target availability of 99.9 percent, accounting for planned maintenance windows and unplanned incidents.

6.14 Security

Security is foundational to CareConnect's design, with multiple defensive layers protecting patient data and ensuring compliance with healthcare regulations. The security architecture follows defense-in-depth principles, if any single control may fail and therefore implement overlapping safeguards.

6.14.1 Data Protection

All patient data is protected through encryption, access controls, and data minimization principles.

- Encryption at Rest: All databases and storage services use AES-256 encryption with keys managed by AWS KMS. Customer-managed keys allow key rotation and access auditing. Database encryption includes tablespaces, backups, and snapshots. S3 buckets storing PHI enforce encryption by default, rejecting unencrypted uploads.

- Encryption in Transit: All network communication uses TLS 1.2 or higher with strong cipher suites. API endpoints reject non-HTTPS requests. Certificate management uses AWS Certificate Manager with automatic renewal. Mobile applications implement certificate pinning to prevent man-in-the-middle attacks.

- Data Minimization: Applications request and store only data necessary for their function. User queries include only required columns rather than selecting entire records. Temporary processing data is deleted immediately after use. Search and analytics use aggregated or de-identified data when possible.

- Data Retention: PHI is retained according to state and federal requirements, typically seven years for medical records. Automated processes purge expired data securely, overwriting storage locations to prevent recovery. Users can request data deletion in compliance with GDPR right to erasure, with exceptions for legal retention requirements.

- Backup Security: Database backups are encrypted with the same keys as production data. Backup restoration procedures are tested quarterly to validate both technical capability and security controls. Backup access is logged and requires multi-person authorization for restoration.

6.14.2 Authentication and Access Control

Strong authentication and granular authorization ensure that only authorized users can access CareConnect and that each user sees only data appropriate to their role.

- User Authentication: AWS Cognito manages user identity with support for username/password, social identity providers, and SAML federation for enterprise customers. Password policies enforce minimum length, complexity requirements, and prohibition of previously used passwords. Multi-factor authentication is required for caregivers accessing sensitive functions and optional for patients.

- Session Management: JWT tokens encode user identity and roles with short expiration times, typically 15 minutes for access tokens and 30 days for refresh tokens. Token refresh occurs automatically in the background to maintain user sessions without interruption. Concurrent session limits prevent account sharing, logging out older sessions when limits are exceeded.

- Role-Based Access Control: The new RBAC system being implemented in this project phase defines granular permissions for different user types including caregivers, patients, family members, and administrators. Roles determine which API endpoints can be called, which screens are visible in the mobile app, and what data queries are permitted. Permission checks occur at multiple layers including API Gateway, application services, and database queries.

- Data-Level Authorization: Beyond role checks, the system enforces data-level permissions ensuring users can only access records they are authorized to view. Caregivers see only patients assigned to them. Patients see only their own health records. Family members see data shared explicitly by the patient. Database queries include authorization filters automatically to prevent accidental data leakage.

- Account Lockout: Failed authentication attempts trigger progressive delays and eventual account lockout after five consecutive failures. Lockouts require password reset or administrator intervention to clear, preventing automated credential guessing attacks.

6.14.3 Secure APIs

API security controls protect backend services from unauthorized access and malicious requests.

- Authentication Requirements: All API endpoints except public health checks require valid JWT tokens in authorization headers. Token validation verifies signature, expiration, and issuer before processing requests. API Gateway rejects requests with missing or invalid tokens before they reach application code.

- Input Validation: All API inputs are validated against strict schemas defining allowed data types, formats, and value ranges. Parameterized queries prevent SQL injection in database operations. Request size limits prevent resource exhaustion attacks. Content-type validation ensures requests match their declared format.

- Rate Limiting: API Gateway enforces request rate limits per user and per IP address to prevent abuse. Burst capacity allows legitimate usage spikes while sustained high rates trigger throttling. Rate limit headers inform clients of remaining quota and reset times.

- CORS Configuration: Cross-origin resource sharing policies restrict which domains can call APIs from web browsers. Allowed origins are explicitly whitelisted, with credentials restricted to trusted domains.

- API Versioning: APIs use versioned endpoints allowing deprecation of old versions with advance notice while maintaining backward compatibility during transition periods. Deprecated endpoints log warnings before eventual removal.

6.14.4 Infrastructure Hardening

Infrastructure security measures protect the underlying compute, network, and storage resources from compromise.

- Network Segmentation: Application components run in private VPC subnets without direct internet access. Public-facing load balancers and API gateways reside in public

subnets, forwarding traffic to private resources. Network ACLs and security groups implement least-privilege access between subnets and services.

- Bastion Hosts: Administrative access to private resources requires connection through hardened bastion hosts with strict access logging. Bastion security groups allow SSH only from known IP ranges. Session manager provides audited terminal access without exposing SSH ports.

- Container Security: Container images are scanned for vulnerabilities during build processes using tools such as Amazon Inspector. Base images are regularly updated to include security patches. Container runtime is hardened with read-only file systems and non-root user execution. Secrets are injected at runtime rather than baked into images.

- Resource Access Controls: IAM roles provide temporary credentials to services using least-privilege principles. Each service has a dedicated role granting only required permissions. Cross-account access requires explicit trust relationships with external ID verification. Service control policies prevent privilege escalation and enforce security guardrails.

- Infrastructure as Code: All infrastructure is defined in Terraform templates under version control. Changes undergo code review before deployment. Drift detection identifies manual modifications to infrastructure. Production deployments require approval workflows preventing unauthorized changes.

6.14.5 Monitoring and Logging

Comprehensive logging and monitoring provide visibility into system behavior, enabling rapid detection and response to security incidents.

- Centralized Logging: CloudWatch Logs aggregates application logs, access logs, and audit trails from all services. Log retention policies maintain security logs for one year to support incident investigation and compliance audits. Logs containing PHI are encrypted and access controlled.

- Audit Logging: All access to PHI is logged with user identity, timestamp, action performed, and resources accessed. Authentication events including logins, logouts, and failed attempts are recorded. Administrative actions such as permission changes and

configuration updates generate audit entries. Audit logs are immutable once written, preventing tampering.

- Security Monitoring: CloudWatch alarms detect suspicious patterns such as repeated authentication failures, unusual access patterns, or elevated error rates. AWS GuardDuty analyzes CloudTrail logs and VPC flow logs for indicators of compromise. Security alerts route to the on-call team through PagerDuty for immediate response.

- Performance Monitoring: Application performance metrics track response times, error rates, and resource utilization. Distributed tracing with AWS X-Ray identifies bottlenecks and failed requests across microservices. Dashboard visualizations provide real-time system health visibility.

- Log Analysis: Regular log reviews identify trends and potential security issues before they become incidents. Automated analysis flags anomalies such as access from unusual locations or elevated privilege usage. Compliance reports aggregate access logs to demonstrate appropriate PHI handling.

6.14.6 Vulnerability Management

Proactive vulnerability management identifies and remediates security weaknesses before they can be exploited.

Dependency Scanning: Automated tools scan application dependencies for known vulnerabilities during every build. Critical vulnerabilities block deployment until remediated through updates or mitigation. Dependency updates are tested in staging before production deployment.

Static Code Analysis: Automated security scanning analyzes source code for common vulnerabilities such as injection flaws, insecure cryptography, and hardcoded secrets. Pre-commit hooks prevent checking in code with critical security issues. Code review processes include security considerations as standard checklist items.

Penetration Testing: Annual third-party penetration tests assess application and infrastructure security. Test scope includes web application, APIs, mobile applications, and cloud infrastructure. Findings are prioritized by severity with remediation timelines defined. Retesting validates that identified issues have been properly addressed.

Patch Management: Security patches for operating systems and middleware are applied within 30 days of release, or sooner for critical vulnerabilities. Containerized architecture simplifies patching through image rebuilds rather than runtime updates. Patch deployment follows change control processes with staging validation before production.

Vulnerability Disclosure: A security contact email and responsible disclosure policy encourage external security researchers to report vulnerabilities. Reported issues are acknowledged within 48 hours and investigated promptly. Reporters are credited in security advisories when fixes are released.

6.14.7 Compliance Measures

CareConnect implements technical and administrative controls to maintain compliance with healthcare privacy regulations.

HIPAA Compliance Controls:

- Administrative Safeguards: Security policies and procedures document how PHI is handled, who has access, and what security controls are in place. Regular risk assessments identify potential vulnerabilities in systems and processes. Security awareness training educates team members on HIPAA requirements and their responsibilities. Incident response procedures define how security breaches are detected, contained, and reported.
- Physical Safeguards: AWS data centers provide physical security for infrastructure hosting CareConnect. Access to production systems requires multi-factor authentication and is logged for audit. Workstation security policies require encryption and screen locks on devices accessing PHI.
- Technical Safeguards: Access controls ensure only authorized individuals can access PHI based on their role. Audit logging tracks all PHI access for compliance reporting and incident investigation. Encryption protects PHI both in storage and during transmission. Automatic logoff terminates idle sessions to prevent unauthorized access.
- Business Associate Agreements: BAAs are executed with all third-party vendors and service providers who handle PHI on behalf of CareConnect. AWS, Stripe, and other cloud services operate under executed BAAs ensuring shared compliance responsibility.

GDPR Compliance Controls:

- Lawful Basis for Processing: Data processing occurs under contract with users for service delivery and with explicit consent for optional features like AI assistance. Privacy policies clearly explain what data is collected and why.
- Data Subject Rights: Users can access their personal data through data export functions. The right to rectification is supported through account settings and data correction interfaces. The right to erasure allows users to request data deletion with retention only for legal requirements. The right to portability provides data in machine-readable formats.
- Data Protection Impact Assessments: DPIAs are conducted for new features involving personal data processing, especially AI and analytics. High-risk processing undergoes additional scrutiny and mitigation measures.
- Cross-Border Data Transfers: EU user data remains within EU regions using AWS European data centers. Standard Contractual Clauses govern any necessary data transfers outside the EU. Data localization preferences allow users to specify data residency requirements.
- Breach Notification: Procedures ensure data breaches are detected and assessed within 72 hours. Affected users are notified when breaches create high risk to their rights. Regulatory authorities receive breach notifications as required by law.

-

Ongoing Compliance Activities:

- Regular compliance audits validate that security controls remain effective, and that processes align with regulatory requirements. Internal audits occur quarterly while external audits by third-party assessors occur annually. Audit findings drive continuous improvement of security and compliance programs.
- Documentation maintenance ensures policies, procedures, and technical documentation remain current with system changes and regulatory updates. Change control processes require compliance review for changes affecting PHI handling or user privacy.
- Compliance training for all team members covers HIPAA, GDPR, and CareConnect-specific security procedures. New team members complete training during onboarding with annual refresher courses for all personnel. Training effectiveness is measured through assessments and incident metrics.

The comprehensive security and compliance framework ensures CareConnect protects patient privacy while delivering valuable healthcare coordination services. Ongoing vigilance and continuous improvement maintain security posture as threats evolve and regulations change.

---

7. Project Deliverables

This section defines the formal deliverables produced during the CareConnect Spring 2026 project. Deliverables are aligned with the approved project scope, SWEN 670 course requirements, and the objective of enhancing deployment architecture, testing practices, and role-based access control within the existing CareConnect system. All deliverables are intended to support technical validation, academic assessment, and future team handoff.

7.1 Planning and Management Deliverables

The Project Management Plan (PMP) defines the project's scope, objectives, schedule, risks, budget, roles, and governance structure. The PMP serves as the baseline document used to guide execution and track progress throughout the project lifecycle.

The Software Requirements Specification (SRS) is updated to reflect Spring 2026 scope constraints and non-functional requirements related to hosting, testing, and access control. All changes are traceable to prior versions and aligned with stakeholder expectations.

7.2 Architecture and Design Deliverables

Technical architecture documentation describes the system’s logical architecture, deployment topology, hosting model, and security posture. Architectural decisions are justified to ensure scalability, compliance, and maintainability.

A high-level system architecture diagram visually represents major components, data flows, security boundaries, and AWS infrastructure elements. The diagram provides a shared reference for developers and stakeholders.

The RBAC framework design specification defines user roles, access rules, and enforcement points across frontend and backend components. The design adheres to the principle of least privilege and supports consistent future implementation.

7.3 Development and Configuration Deliverables

The CareConnect application is deployed to the selected AWS serverless hosting platform. The deployed environment demonstrates secure configuration, availability, and cost-aware operation.

Infrastructure configuration documentation describes deployment assumptions, environment setup, and operational constraints. These artifacts support reproducibility and future maintenance.

7.4 Testing and Quality Assurance Deliverables

A standardized unit testing framework is defined for frontend and backend components. The framework establishes testing conventions, directory structures, and tooling standards for reuse by future teams.

The Software Test Plan (STP) documents the testing strategy, scope, responsibilities, and validation approach for the Spring 2026 objectives. The plan ensures testing activities are structured and repeatable.

Test execution evidence demonstrates the successful application of the defined testing framework to the existing codebase. Results validate the effectiveness of the testing approach.

7.5 Documentation and Handoff Deliverables

The Programmer or Developer Guide provides technical guidance for future developers. The guide includes setup instructions, testing execution details, RBAC integration notes, and deployment considerations.

The Deployment and Operations Guide documents runtime behavior, deployment procedures, monitoring assumptions, and known operational constraints. This guide supports system handoff and continuity.

7.6 Academic and Course Deliverables

All milestone deliverables required by the SWEN 670 course are submitted according to the approved schedule. Draft and final submissions follow instructor guidance and course grading criteria.

The final project presentation summarizes project objectives, technical decisions, delivered outcomes, risks, and lessons learned. The presentation supports academic evaluation and stakeholder review.

7.7 Traceability and Validation

Each deliverable is traceable to project objectives, scope elements, and course requirements. Deliverable completion is validated through milestone reviews, instructor feedback, and demonstration of working artifacts where applicable.

8. Risk & Mitigation

This section identifies potential risks and strategies to minimize their impact.

8.1 Risk Probability Matrix

Visual matrix evaluating likelihood and impact of risks.

Table 4 Risk Matrix

	Low Impact	Medium Impact	High Impact
High Probability	Minor UI inconsistencies; Minor documentation lag.	Inter-team dependency delays; Resource availability.	**Tight Schedule (12-week MVP / 4-week Beta)**
Medium Probability	Unit testing initial setup delays; Linter errors.	RBAC Complexity; AWS configuration challenges.	**AWS Migration Hurdles (Fargate/App Runner)**
Low Probability	Minor API versioning issues.	Loss of historical project documentation.	**HIPAA/GDPR Compliance Failure; Data Breach**

8.2 Risk Identification, Probability, and Mitigation

Detailed list of risks with mitigation strategies.

- Schedule Risk: The 12-week development cycle with a beta version due by Week 4.
    - Mitigation: Prioritizing a Minimum Viable Product (MVP) and using serverless/managed services to speed up deployment.
- Budget Risk: The mandate to use only open-source and free tools with no budget for paid solutions.
    - Mitigation: Leveraging AWS credit-eligible services and "scale-to-zero" architecture to minimize costs.

- Integration Risk: Limitations on wearable integration (Fitbit only) and smart home integration (Nest only).
    - Mitigation: Explicitly defining these as the only supported integrations for the MVP to prevent scope creep.
- Scope Risk: "Functional Scope Constraint" ensures no new user-facing features are added.
    - Mitigation: Strict adherence to the existing feature set to ensure the infrastructure and RBAC goals are met within the 12-week window.
- Dependency Risk: "Inter-Team Dependency Constraint" notes that parallel development is occurring.
    - Mitigation: Designing the system to be independent of other teams' deliverables to avoid bottlenecks.
- Data Security Risk: Addressed in SRS Section 6.1 (Data Encryption).
    - Mitigation: TLS 1.3 for data in transit and AES-256 for data at rest.
- Regulatory/Legal Risk: Addressed in SRS Section 6.2 (Regulatory Compliance).
    - Mitigation: Use of HIPAA-eligible services and comprehensive audit logging.
- System Failure/Data Loss Risk: Addressed in Section 6.6 (Backup & Disaster Recovery).
    - Mitigation: Automated backups in geographically separate locations and defined RTO/RPO targets.
- Resource Risk: Dependency on the availability of the existing CareConnect source code and access to AWS resources.
    - Mitigation: Early verification of repository access and AWS account permissions.

---

9. Budget Estimate

This section outlines projected costs and financial planning.

9.1 Labor Cost

The biggest expense to this phase of the CareConnect project will be Labor. Please see the table below for detailed breakdown.

Labor (7 roles, 12 weeks, 20 hrs/week): $57,609

Materials: $0 (student-provided) Infrastructure (AWS, Stripe, Stores): $1,320 (As cited in Summer/Fall 2025 PMP)

Contingency (10%): Included Total: ≈ $64,821.90

Budget Estimate for Labor Costs.

Table 5 Labor Costs

Team Member Name	Project Role	Cost Per Hour	Work Hours / Week	Weekly Salary	Monthly Salary	3- Month Cost
Colyn Mahoney	Project Manager	$48.85	5	$244.25	$977	$2,931
Colyn Mahoney	Team Lead	$82.31	15	$1,234.65	$4,938.60	$14,815.80
Obeng Buo	Technical Lead/Architect	$62.69	15	$940.35	$3,761.40	$11,284.20
Henry Stewart	Software Developer	$63.20	15	$948	$3,792	$11,376
Ja'Lisa Hawkins	Business Analyst	$48.65	15	$729.75	$2,919	$8,757
Zechariah Hillman	Test Engineer	$63.20	15	$948	$3,792	$11,376
Total						$60,540

Note. Cost per hour rates retrieved from the U.S. Bureau of Labor Statistics

9.2 Materials Cost

This project will be completed as part of the SWEN 670 course. Students are expected to provide their own hardware, so project cost is estimated to be $0.

9.3 Infrastructure Cost

This section is intended to describe the expected infrastructure costs to inform stakeholders about the costs of services needed for this project. The following table includes each service Team C has identified as needed to complete their requirements.

Table 5 Infrastructure Costs

#	Service	Sizing Assumption	Unit Price*	Monthly Qty	MRC USD
1	AWS Fargate	2 tasks × (0.25 vCPU / 0.5 GB) avg 1 hour	60.83 tasks x 0.50 vCPU x 4 hours x 0.04048 USD	2 pods x 1 Hour	$6.00

			per hour = 4.92 USD for vCPU hours 60.83 tasks x 1.00 GB x 4 hours x 0.004445 USD per GB per hour = 1.08 USD for GB hours		
2	RDS	db.t4g.medium Multi-AZ	0.015 $/h × 730 h	730	$21.90
3	API Gateway	3 M API calls/mo	$1.00 per 1 M	3	$3
Total					$30.90

9.3.1 AWS Core Services

AWS public on-demand rates as of January 2026, us-east-1.

9.3.2 Third-party / One-time Fees

There have not been any external or one-time fees identified at this time.

9.3.3 Total Budget for This Phase

Table 6 Total Budget

Category	Monthly	3-Month	Notes
AWS	$30.90	$182.82	Fargate pricing based on 1 hour for 2 pods
Subtotal		92.18	
Contingency +10%		$9.27	
Total		$101.97	

9.3.4 Cost-Control & Optimization Plan

Table 7 Cost Control

Lever	Action	Expected Savings
Compute	Keep Q&A environments on smaller instances of compute services	15%-20%
DB	Investigate cost of running container image of PSQL vs. RDS for production environment.	Unknown

(CareConnect Summer/Fall 2025 was referenced in creating this plan)

9.3.5 Alignment to Financial Policies

- Cap-Ex vs. Op-Ex. All spend is Op-Ex (pay-as-you-go), aligning with U.S. GAAP guidance for cloud-service costs (ASC 350-40).
- Cost Recognition. Expenses booked monthly via consolidated AWS invoice to Cost Center IT-R&D-CareConnect.
- SOX Controls. AWS accounts integrated with enterprise AWS Organizations. Detailed billing reports exported to an immutable S3 bucket for audit.

HIPAA Compliance Fees. AWS BAA services carry no surcharge; Stripe's PCI scope remains with the vendor. No additional license cost foreseen.(As cited in Summer/Fall 2025 CareConnect PMP)

9.3.6 Budget Summary for Project Plan

Table 8 Budget Summary

Line Item	Amount
Labor Cost (9.1)	$60,540
Materials Cost (9.2)	$0.00
Infrastructure (9.3)	$101.97
Total Project Budget	$60,741.11

---

10. Roles & Responsibilities

This section defines accountability and ownership across the project team to ensure clear communication, efficient execution, and successful delivery of project objectives.

10.1 Roles

Team Lead

The Team Lead is responsible for overall project coordination, leadership, and communication with stakeholders, ensuring that project objectives are met within scope and schedule.

Responsibilities:

- Oversee project planning, scheduling, and milestone delivery
- Serve as the primary point of contact with faculty and stakeholders
- Facilitate team meetings and track action items
- Resolve project-level issues and risks
- Ensure alignment between project goals, deliverables, and timelines

Technical Lead / Architect

The Technical Lead/Architect is responsible for defining the system architecture and guiding technical decisions to ensure scalability, security, and compliance.

Responsibilities:

- Design and maintain the overall system architecture
- Evaluate and select AWS cloud hosting solutions
- Ensure security, scalability, and compliance requirements are met
- Guide development standards and technical best practices
- Review and approve technical implementations

Software Developer

The Software Developer is responsible for implementing application features and ensuring code quality in alignment with project requirements and technical standards.

Responsibilities:

- Develop and enhance application features
- Implement role-based access control functionality
- Refactor existing code to improve maintainability
- Collaborate with the Test Engineer to resolve defects
- Contribute to technical documentation and the Programmer Guide

Business Analyst

The Business Analyst is responsible for gathering, analyzing, and documenting business and user requirements to ensure the system meets stakeholder needs.

Responsibilities:

- Elicit and document functional and non-functional requirements
- Develop and maintain the Software Requirements Specification (SRS)
- Validate requirements with stakeholders
- Ensure alignment between requirements, project goals, and compliance needs
- Support acceptance of delivered features

Test Engineer

The Test Engineer is responsible for defining testing standards and validating the quality, reliability, and correctness of the CareConnect application.

Responsibilities:

- Design and implement unit testing norms
- Develop and execute automated and manual test cases
- Document test results and defects
- Validate system reliability and performance
- Contribute to the Software Test Plan and Test Report

10.2 RACI Matrix

Matrix defining who is Responsible, Accountable, Consulted, and Informed.

Project Activity	Team Lead	Technical Lead	Software Developer	Business Analyst	Test Engineer
Project Planning & Scheduling	A	C	I	C	I
Requirements Gathering & SRS	I	C	I	A/R	I
System Architecture & Design	I	A/R	C	C	I
AWS Hosting Evaluation	I	A/R	C	I	I
Software Development	I	C	A/R	I	C
Role-Based Access Control Implementation	I	A	R	C	C
Unit Testing Strategy	I	C	C	I	A/R
Test Execution & Reporting	I	I	C	I	A/R
Documentation (Guides & Reports)	A	C	R	R	R
Final Delivery & Presentation	A	C	C	C	C

11. Monitoring and Reporting

This section describes how progress and performance are tracked and communicated.

11.1 Intra-Team Monitoring and Reporting

To maintain project momentum and ensure deliverables align with requirements, our team will employ a multi-faceted approach to track work and measure contributions:

- Sprint Retrospectives: Weekly review sessions will be conducted to assess sprint completion, identify blockers, and adjust workflows as needed
- Project Management Integration: GitHub Projects will serve as our primary task management system, providing visibility into individual assignments, status updates, and dependencies

- Analytics Dashboard: A key performance indicator (KPI) dashboard will be developed to monitor feature adoption rates and user engagement metrics for data-driven decision making

11.2 Project Communications

Internal team meetings are scheduled weekly to facilitate information sharing, coordinate efforts, and maintain alignment among all team members.

Weekly sessions between internal team members and external stakeholders will be established to:

- Share progress updates and milestones
- Review deliverables and gather feedback
- Discuss requirement modifications and scope adjustments

Coordination with other teams will be conducted on an ad-hoc basis when cross-functional collaboration is necessary for project success.

---

12. Assumptions

The following are assumptions by the development team, Team C and validated with the scope of work seen in section 3 of this PMP. Any assumption proven false will be escalated to a risk (Summer/Fall 2025 PMP was referenced).

- New hosting environment: The team assumes that with the requirement of choosing an AWS cloud service to host CareConnect that there is a service with the necessary capabilities to host an application of the size and complexity of CareConnect.
- Unit Testing: Team C assumes that the design new unit testing norms will involve setting up the new unit tests for existing code base, but any concurrent/future development efforts will write their own tests following the new norms.
- RBAC: Team C assumes that any other current or future development efforts will be using the norms or systems developed for RBAC and implement access control for their feature.
- Previous Work: Team C assumes that all documentation provided as part of the hand off and used in the development of this PMP and other related project documentation is consistent with the existing code base.

- Device Compatibility: It is assumed that all end-users will access the CareConnect application through a modern SmartPhone with iOS or Android as an operating system (Summer/Fall PMP referenced).
- Existing Code Base: It is assumed that the code base is working, free of any extra feature branches, free of syntax errors, and is consistent with handoff documentation.
- AWS: It is assumed that AWS services in use operate as described in the documentation and that AWS Fargate works as described in it's documentation.

13. Constraints

Identifies limitations affecting scope, schedule, budget, or technology.

- Timeline: 12-week development cycle; beta version required within 4 weeks.
- Budget: Only open-source and free tools; AWS services limited to credit-eligible, scale-to-zero options.
- Architecture: Scale-to-zero and serverless architecture prioritized.
- Functional Scope: No new user-facing features; scope limited to migration, unit testing redesign, and RBAC enhancement.
- Technology Stack: Must work within existing Flutter/Dart frontend and current backend architecture.
- Regulatory Compliance: Full HIPAA/GDPR verification deferred to external division; HIPAA-eligible AWS services required.
- Accessibility: Full WCAG 2.1 Level AA compliance deferred to future releases.
- Inter-Team Dependencies: Minimized dependencies on other Spring 2026 teams; no integration with newly developed components outside scope.
- Deployment: Limited to evaluating and migrating to AWS Fargate or App Runner; broader infrastructure changes excluded.
- Testing Scope: Limited to redesigning unit testing paradigm for existing functionality; excludes testing new features from other teams.
- RBAC Scope: Limited to refactoring existing roles, screens, and API endpoints; excludes new roles or permissions for new components.

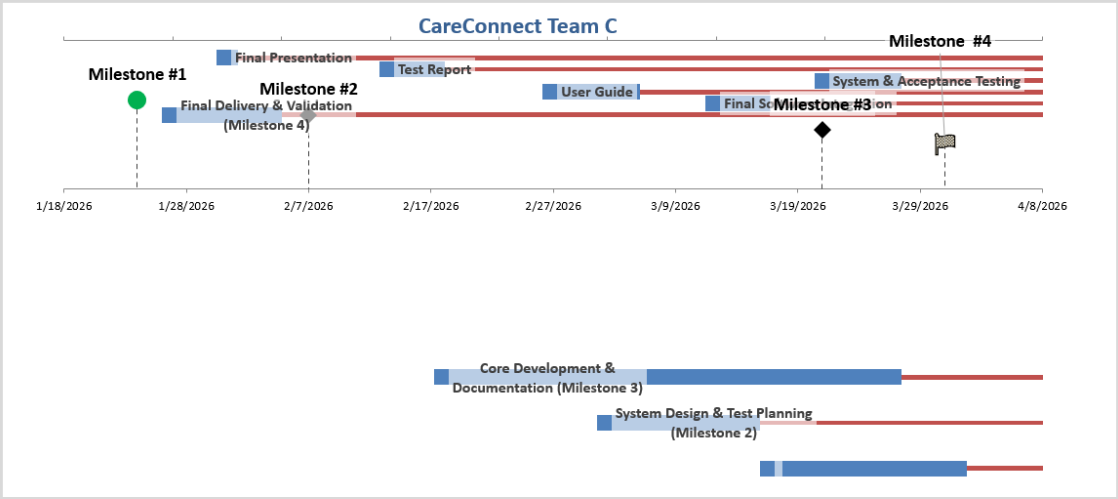
---

14. Work Breakdown Structure (WBS)

The Work Breakdown Structure (WBS) breaks down the CareConnect project into manageable and well-defined tasks. The WBS organizes the total scope of work into hierarchical levels, ensuring clear ownership, traceability to project objectives, and effective progress tracking.

14.1 Structure

Overview and explanation of the WBS organization.



Tasks

Task Index	Start	Start Offest	End	Duration	Label	Vert. Position	Vert. Line
1	1/7/2026	1/3/2026	1/24/2026	18	Project & Planning (Milestone 1)	-25	-25
2	1/7/2026	1/6/2026	1/20/2026	14	Project Plan		
3	1/10/2026	1/3/2026	1/22/2026	13	Software Requirements Specification (SRS)		
4	1/7/2026	1/21/2026	1/18/2026	12	Environment Setup & Prior Cohort Review		
5	1/25/2026	1/21/2026	2/7/2026	14	System Design & Test Planning (Milesto	-40	-15

6	1/25/2026	1//28/2026	2/4/2026	11	Technical Design Document		
7	1//28/2026	1/24/2026	2/6/2026	13	AWS Hosting Evaluation & Prototyping		
8	1/28/2026	2/4/2026	2/7/2026	11	Software Test Plan		
9	2/8/2026	2/4/2026	3/21/2026	42	Core Development & Documentation (M	-55	-15
10	2/8/2026	2/11/2026	3/5/2026	26	Core Feature Implementation		
11	2/8/2026	2/4/2026	2/12/2026	5	Investigate AWS Fargate and App Runner		
12	2/13/2026	2/12/2026	2/14/2026	2	Select Target AWS Hosting Service		
13	2/15/2026	2/14/2026	2/19/2026	5	Research Deployment Steps for Selected Service		
14	2/20/2026	2/19/2026	2/26/2026	7	Package Application for Deployment (Containerization / Build Artifacts)		
15	2/27/2026	2/26/2026	3/2/2026	4	Deploy Production Build to Selected Host		
16	3/3/2026	3/2/2026	3/7/2026	5	Regression Test Deployed Application		
17	2/15/2026	2/16/2026	3/10/2026	24	Role-Based Access Control Enhancements		
18	2/15/2026	2/11/2026	2/18/2026	4	Identify Current RBAC Inefficiencies		
19	2/19/2026	2/18/2026	2/23/2026	5	Design Streamlined RBAC Model		
20	2/24/2026	2/23/2026	3/6/2026	11	Implement Updated RBAC in Codebase		
21	3/7/2026	3/6/2026	3/10/2026	4	Validate RBAC Behavior Across User Roles		
22	2/20/2026	2/25/2026	3/15/2026	25	Unit Testing Implementation		
23	2/20/2026	2/16/2026	2/23/2026	4	Identify Unit Testing Pain Points		
24	2/24/2026	2/23/2026	2/27/2026	4	Develop New Unit Testing Standards and Norms		
25	2/28/2026	2/27/2026	3/10/2026	11	Refactor Existing Tests to New Standards		
26	3/11/2026	3/10/2026	3/17/2026	7	Implement New Unit Tests for Core Components		
27	3/1/2026	3/1/2026	3/21/2026	21	Programmer Guide		
28	3/1/2026	3/1/2026	3/5/2026	5	Code Review for Readability and Maintainability		
29	3/6/2026	3/5/2026	3/14/2026	9	Refactor Code for Clarity and Consistency		
30	3/15/2026	3/14/2026	3/21/2026	7	Update Inline Comments and Documentation		
31	3/5/2026	3/18/2026	3/21/2026	17	Deployment & Operations Guide		
32	3/22/2026	3/18/2026	3/31/2026	10	Final Delivery & Validation (Milestone 4	-80	-80
33	3/22/2026	3/18/2026	3/26/2026	5	Final Software Integration	-30	-30
34	3/22/2026	3/20/2026	3/29/2026	8	User Guide	-45	-15
35	3/24/2026	3/23/2026	3/30/2026	7	System & Acceptance Testing	-20	-20
35	3/24/2026	3/23/2026	3/30/2026	7	System & Acceptance Testing	-20	-20
36	3/27/2026	3/27/2026	3/31/2026	5	Test Report	-60	-40
37	3/31/2026	3/31/2026	3/31/2026	1	Final Presentation	-75	-15
					Insert new rows above this one		

Milestones

Date	Label	Position
1/24/2026	Due, Jan 24	30
2/7/2026	Milestone #2	25
3/21/2026	Milestone #3	20
3/31/2026	Deliver, Mar 31	15
	Insert new rows above this one	

15. Change Management Procedures

Any change request will follow a standard approval flow. Initiation -> CCB review -> Approval -> Implementation

15.1 Initiation

Any team member or stake holder can initiate a change request by filling out a standard change request form. This form will have various pieces of information including a description, a justification and priority of the change.

15.2 Review

Once a change request form is submitted, it will go to the CCB for review. Participants of the CCB and their roles are defined in the following table.

Member	Role	Responsibility
Dr. Assadullah Roy Gordon Prof. Minagar Sukhmeet Khalar Sudarshan Neupane	Course Instructor/ Client Client Client Client Client	Approval and setting priority
Colyn Mahoney	Project Manager/ Team Lead	Determines if change will be brought to the board
Henry Stewart/Obeng Buo	Software Developer/Technical Lead	Determines if change is technically possible
Team C Members	Development team	Provides analysis and LOE of the change

15.3 Authorization and Implementation

Once approved, changes will be added to the appropriate documentation, and scope will be updated. Once that has happened, the development team will take the necessary steps to implement those changes. Once development work is completed, the team will carry out regression testing.

15.4 Change Request Form

All CareConnect teams will be using a change request form that was designed by Team D and agreed upon during a team lead meet up.

16. Appendices

16.1 References

Requirements document provided by stakeholder mentors.

Project Management Plan document from Summer/Fall 2025 cohort https://umgc-cappms.azurewebsites.net/download/c77df719-c5d0-42ef-ada8-7c87e8f2566b----SWEN670_CareConnect_2025_Fall2024_PojectPlan.pdf

Amazon Web Services. (2024). *AWS App Runner*. Retrieved from https://aws.amazon.com/apprunner/

JUnit Team. (2024). *JUnit user guide (Version 6.0.2)*. Retrieved from https://docs.junit.org/6.0.2/overview.html

JUnit User Guide https://docs.junit.org/6.0.2/overview.html

Project Management Institute. (2021). A Guide to the Project Management Body of Knowledge (PMBOK® Guide) (7th ed.). PMI.

IEEE Computer Society. (2014). ISO/IEC/IEEE 29148: Systems and software engineering — Life cycle processes — Requirements engineering.

International Institute of Business Analysis. (2015). A Guide to the Business Analysis Body of Knowledge (BABOK® Guide) (3rd ed.).

NIST. (2020). Security and Privacy Controls for Information Systems and Organizations (SP 800-53 Rev. 5).

Amazon Web Services. (2024). AWS Well-Architected Framework. Retrieved from https://aws.amazon.com/architecture/well-architected/

Amazon Web Services. (2024). Architecting for HIPAA Security and Compliance on AWS. AWS Whitepaper. Retrieved from https://docs.aws.amazon.com/whitepapers/latest/architecting-hipaa-security-and-compliance-on-aws/

Amazon Web Services. (2024). AWS Fargate Documentation. Retrieved from https://aws.amazon.com/fargate/

Tanenbaum, A. S., & Van Steen, M. (2017). Distributed Systems: Principles and Paradigms (3rd ed.). Pearson.

Google. (2024). Flutter Documentation (v3.x). Retrieved from https://docs.flutter.dev/

Apple Inc. (2024). HealthKit Framework Documentation. Retrieved from https://developer.apple.com/documentation/healthkit

Pivotal/VMware. (2024). Spring Boot 3.x Documentation. Retrieved from https://docs.spring.io/spring-boot/

Newman, S. (2021). Building Microservices: Designing Fine-Grained Systems (2nd ed.). O'Reilly Media.

Richardson, C. (2018). Microservices Patterns: With Examples in Java. Manning Publications.

Apache Software Foundation. (2024). Apache Kafka Documentation. Retrieved from https://kafka.apache.org/documentation/

PostgreSQL Global Development Group. (2024). PostgreSQL 15 Documentation. Retrieved from https://www.postgresql.org/docs/15/

Amazon Web Services. (2024). Amazon RDS for PostgreSQL User Guide. Retrieved from https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_PostgreSQL.html

Amazon Web Services. (2024). Amazon DynamoDB Developer Guide. Retrieved from https://docs.aws.amazon.com/amazondynamodb/

Amazon Web Services. (2024). Amazon Timestream Developer Guide. Retrieved from https://docs.aws.amazon.com/timestream/

Kleppmann, M. (2017). Designing Data-Intensive Applications. O'Reilly Media.

Stripe, Inc. (2024). Stripe API Reference & HIPAA Compliance. Retrieved from https://stripe.com/docs/api

Fitbit, Inc. (2024). Fitbit Web API Reference. Retrieved from https://dev.fitbit.com/build/reference/web-api/

Google. (2024). Health Connect Documentation. Retrieved from https://developer.android.com/guide/health-and-fitness/health-connect

OWASP. (2021). OWASP Top 10 - 2021. Retrieved from https://owasp.org/Top10/

Amazon Web Services. (2024). Amazon Cognito Developer Guide. Retrieved from https://docs.aws.amazon.com/cognito/

Project Plan Document

CareConnect

University of Maryland Global Campus
SWEN 670 - Software Engineering Capstone
Instructor / Sponsor (if applicable)
January 24, 2026

Contributors:
Justinah Bashua (Technical Lead/Architect)
Corey Bayliss (Lead Tester)
Matthew Lester (Lead Software Developer)
James Stevens (Team Lead)
Brandon Sutan (Lead Business Analyst)

Revision History

Date	Version	Reason for Change	Team/Author
1/24/2026	1.0	Initial Document	All team members
2/4/2026	1.1	Incorporated feedback	James Stevens
2/28/2026	1.2	Incorporated feedback	James Stevens

Table of Contents

1. EXECUTIVE SUMMARY 5

2. OBJECTIVES AND GOALS 6

2.1 Goals 6
2.2 Objectives 6

3. SCOPE OF WORK 8

3.1 In-Scope 8
3.2 Out-of-Scope 9

4. STAKEHOLDERS 10

4.1 Internal Stakeholders 10
4.2 External Stakeholders 11

5. TIMELINE & MILESTONES 13

5.1 Milestone Table 13
5.2 Phase-Based Schedule 14
5.3 Gantt Chart 15

6. TECHNICAL ARCHITECTURE 16

6.1 Architectural Objectives 17
6.2 Logical Architecture 17
6.3 High-Level Diagrams 19
6.4 Architecture Decisions & Justifications 20
6.6 Technology Stack Summary 24
6.7 Deployment Topology 24
6.8 Frontend 25
6.9 Backend 25
6.10 Database 27
6.11 APIs 28
6.12 AI Services 30
6.13 Hosting 31
6.14 Data Protection 32

7. PROJECT DELIVERABLES 35

8. RISK & MITIGATION 37

8.1 Risk Probability Matrix 38
8.2 Risk Identification & Mitigation 38

9. BUDGET ESTIMATE 39

9.1 Labor Cost 39
9.2 Materials Cost 40
9.3 Infrastructure Cost 41
9.4 Total Estimated Cost 42
10. ROLES & RESPONSIBILITIES 43
10.1 Roles 43
10.2 RACI Matrix 44
11. MONITORING & REPORTING 46
11.1 Intra-Team Monitoring & Reporting 46
11.2 Project Communications 47
12. ASSUMPTIONS 48
13. CONSTRAINTS 49
13.1 Schedule Constraints 49
13.2 Technology Constraints 49
13.3 Platform and Compliance Constraints 49
13.4 Quality and Security Constraints 49
13.5 Resource Constraints 49
13.6 Dependency Constraints 50
13.7 Tooling and Environment Constraints 50
14. WORK BREAKDOWN STRUCTURE (WBS) 51
15. CHANGE MANAGEMENT PROCEDURES 53
15.1 Initiation 53
15.2 Review 56
15.3 Authorization and Implementation 56
16. APPENDICES 57
Appendix A - Glossary of Terms 57
Appendix B - References 62
Appendix C - Change Request Form 63

1. Executive Summary

This project focuses on strengthening the engineering foundation of the application by enforcing automated quality controls, secure DevOps practices, platform-compliant payment systems, and centralized notification infrastructure. Rather than introducing new end-user features, the project prioritizes reliability, security, maintainability, and compliance.

Key efforts include integrating industry-standard static analysis and security scanning tools, enforcing CI/CD quality gates that prevent insecure or failing builds from progressing, migrating payments to native Apple and Google billing systems to meet platform requirements, and implementing a unified notification and reminder system using AWS-managed services.

By emphasizing automatic enforcement, clear governance, and tooling outputs that can be understood by future cohorts without specialized security expertise, the project reduces technical debt, improves delivery discipline, and establishes a sustainable foundation for continued development.

Success Criteria

This project will be considered successful when CI/CD pipelines automatically block merges and deployments that fail quality or security gates; all Critical and High severity findings are resolved or formally documented in a carry-forward log; platform billing is validated through successful end-to-end Apple and Google native billing flows; and centralized notifications successfully deliver, log, and escalate reminders through AWS-managed services.

Success will be demonstrated through CI/CD pipeline artifacts, security scan reports, and milestone demonstrations executed from validated builds.

2. Objectives and Goals

2.1 Goals

The primary goals of this project are to improve software quality, enforce secure delivery practices, and ensure compliance with healthcare and mobile platform expectations. The project aims to prevent insecure or unstable software from being released, reduce long-term maintenance risk, and establish a delivery process that future teams can confidently extend.

Additional goals include aligning payment workflows with Apple and Google platform policies and providing a centralized, scalable notification system that supports reliable reminders and caregiver oversight.

2.2 Objectives

To achieve these goals, the project will:

- Integrate industry-standard static analysis and security scanning tools to identify bugs, vulnerabilities, and maintainability issues early in development.
- Ensure all critical and high-severity issues are resolved before release, with medium-severity issues addressed or formally justified.
- Enforce CI/CD quality gates that automatically block builds, merges, and deployments when tests or security checks fail.
- Achieve a “Green” status for code quality and vulnerability reporting prior to release.
- Migrate payment processing to Apple and Google native billing systems to ensure mobile platform compliance.
- Implement a centralized notification and reminder system using AWS SNS and SES.
- Ensure tooling outputs, reports, and documentation can be understood and reproduced by future cohorts without requiring security specialization.

Static Analysis and Security Tooling Justification

A multi-tool static analysis and security approach is required to achieve comprehensive coverage of software quality and risk. All selected static analysis, dependency scanning, container scanning, and dynamic testing tools are open source, ensuring transparency, cost neutrality, and long-term accessibility for future cohorts.

For the Flutter frontend, local static checks are enforced using `flutter analyze` **(Dart Analyzer)** to identify type errors, correctness warnings, and maintainability issues before code is committed. Formatting and linting rules (such as `dart format` and standard Flutter lint sets) are used to ensure consistent code style and reduce review overhead. These checks are executed locally and mirrored in CI to ensure consistent behavior across environments.

Static Application Security Testing (SAST) tools are used to analyze source code and bytecode without executing the application. **SpotBugs** detects bytecode-level defects and runtime bug patterns, **PMD** enforces coding best practices and complexity limits, and **Checkstyle** ensures consistent formatting and readability. **SonarQube** aggregates results from these analyses to provide consolidated visibility into code quality trends, security hotspots, and technical debt.

Software Composition Analysis (SCA) is performed using **OWASP Dependency-Check** to identify known vulnerabilities in third-party libraries and transitive dependencies. Container image scanning is conducted using **Trivy** to detect vulnerabilities in packaged artifacts and base images prior to deployment.

Dynamic Application Security Testing (DAST) is conducted using **OWASP ZAP** to identify runtime vulnerabilities in deployed environments. OWASP guidance is used as the authoritative baseline for evaluating web-related security risks across all analysis stages.

This layered SAST, SCA, and DAST strategy ensures vulnerabilities are detected early in the development lifecycle, results remain understandable to non-specialists, and enforcement can be automated within CI/CD pipelines.

3. Scope of Work

This project focuses on strengthening the engineering foundation of the CareConnect platform through improved quality enforcement, secure DevOps practices, compliant payment processing, and reliable notification infrastructure. The scope emphasizes process, tooling, and infrastructure enhancements rather than the introduction of new end-user features. All in-scope activities are aligned with project milestones and are intended to improve reliability, security, maintainability, and compliance while ensuring continuity for future development cohorts.

3.1 In-Scope

The following activities are included within the scope of this project:

- Integration and configuration of static code analysis tools (SpotBugs, PMD, Checkstyle).
- Integration of a security vulnerability scanner (SonarQube or SonarCloud) with documented justification.
- Dependency vulnerability scanning using OWASP Dependency-Check.
- Container and image vulnerability scanning using Trivy.
- Dynamic application security testing using OWASP ZAP.
- CI/CD enforcement using GitHub Actions to block builds, merges, and deployments on test or analysis failures.
- Implementation of branch protections enforcing pull request → development → master/production workflows.
- Migration of payments to Apple native billing for iOS and Google native billing for Android and web.
- Optional Stripe integration only if time allows after platform billing is complete.
- Implementation of centralized notifications and reminders using AWS SNS and SES, supporting SMS, email, and in-application delivery.
- Storage of notification and reminder history for caregiver oversight and analytics.
- Documentation of all tooling, pipelines, reports, and unresolved issues for future cohorts.

3.2 Out-of-Scope

The following items are explicitly excluded:

- Development of new end-user application features.
- Manual deployments or bypassing CI/CD enforcement.
- Security decisions requiring undocumented exceptions.
- Non-platform payment providers beyond conditional Stripe support.
- Legacy or hardcoded notification mechanisms.
- Feature-driven UI redesigns unrelated to payment compliance.

3-1. Scope of Work Summary

Category	In Scope	Out of Scope
Static Analysis	SpotBugs, PMD, Checkstyle	Ad-hoc/manual reviews
Security Scanning	SonarQube/SonarCloud, OWASP tools	Ignoring medium+ findings
CI/CD	Enforced GitHub Actions gates	Manual bypass
Payments	Apple & Google native billing	Non-platform providers
Notifications	AWS SNS/SES centralized messaging	Legacy hardcoded paths
Documentation	Tooling, reports, justifications	Undocumented exceptions
Features	Infrastructure & quality only	New end-user features

4. Stakeholders

This section identifies the key stakeholders involved in or affected by the project. Stakeholders include individuals and groups with decision-making authority, implementation responsibility, oversight roles, or direct usage of the system. Understanding stakeholder roles and expectations is essential for effective communication, governance, and project success.

4.1 Internal Stakeholders

Internal Stakeholders are individuals directly involved in project oversight, guidance, implementation, testing and delivery. Roles and detailed responsibilities for internal stakeholders are defined in Section 10 (Roles and Responsibilities).

- **Dr. Mir Assadullah**
  Course Instructor; provides academic oversight, evaluates deliverables, and serves as final authority for course-related decisions.

- **Roy Gordon**
  Mentor and escalation point; provides guidance, feedback, and support when issues require escalation beyond the team or product owners.

- **Sukhmeet Khalar (Project Owner)**
  Represents business and product requirements; provides clarification, prioritization, and acceptance guidance.

- **Sudarshan Neupane (Project Owner)**
  Represents business and product requirements; collaborates on feature scope, quality expectations, and validation criteria.

- **James Stevens (Team Lead)**
  Responsible for overall coordination, technical direction, stakeholder communication, and adherence to project governance and timelines.

- **Brandon Sutan (Business Analyst)**
  Responsible for requirements analysis, documentation alignment, and ensuring business needs are accurately reflected in technical and planning artifacts.

- **Matthew Lester (Programming Lead)**
  Responsible for development activities, code quality, and implementation coordination.

- **Justinah Bashua (Technical Lead/Architect)**
  Responsible for architectural decisions, technical standards, and alignment between design and implementation.

- **Corey Bayliss (Testing Lead)**
  Responsible for testing strategy, validation activities, and ensuring quality and acceptance criteria are met.

4.2 External Stakeholders

External stakeholders are individuals, groups, or organizations that interact with, depend on, or are impacted by the system but are not directly responsible for implementation.

- **Care Recipients**
  Primary beneficiaries of the system; rely on accurate reminders, notifications, and system reliability.

- **Caregivers**
  Healthcare Providers - Use the system to support care delivery, monitoring, and coordination.
  Family Members - Support care recipients and rely on notifications, reminders, and system visibility.

- **University Faculty**
  Faculty members review, evaluate, and provide feedback on project deliverables and demonstrations.

- **Platform Providers**
  Mobile platform providers (e.g., Apple, Google) that define application distribution, billing, and compliance requirements.
  Cloud service providers (e.g. AWS) support hosting, messaging, and infrastructure services.

4-1. Stakeholder Influence / Interest Matrix

Stakeholder	Influence	Interest	Notes
Instructor (Dr. Assadullah)	High	High	Final authority for grading + acceptance
Product Owners	High	High	Define requirements + approve deliverables
Mentor (Roy Gordon)	Medium	Medium	Guidance + escalation support
Team Lead	High	High	Execution ownership
Technical Lead	High	High	Architecture + enforcement decisions
Testing Lead	Medium	High	Validation + test readiness
Caregivers / Care Recipients	Low	High	Indirect beneficiaries
Apple / Google	High	Medium	Compliance constraints

5. Timeline & Milestones

This section presents the high-level project schedule aligned with defined milestones. The timeline illustrates major project phases, sequencing, and overlap rather than day-to-day task execution. The schedule is intended to communicate overall project flow, milestone alignment, and delivery expectations; detailed task breakdowns may evolve as the project progresses.

5.1 Milestone Table

Milestone	Deliverables Due	Due Date
Milestone 1	Project Plan Software Requirements Specification	January 24, 2026
Milestone 2	Technical Design Document Software Test Plan	February 7, 2026
Checkpoint	Assess progress Course correct Guidance	Febuary 21, 2026
Milestone 3	Programmer Guide Deployment and Operations Guide	March 21, 2026
Milestone 4	User Guide Test Report	March 31, 2026

Milestone dates reflect course-defined submission and presentation guidelines.

5.2 Phase-Based Schedule

The project schedule is organized into major execution phases. Activities are intentionally overlapping to reflect parallel work streams and iterative development rather than a strictly sequential waterfall approach.

Phase	Key Activities	Start Date	End Date	Related Milestone
Project Initiation & Planning	Project planning Stakeholder alignment Process definition	January 7	January 24	Milestone 1
Requirements & Design	SRS refinement Architecture design Test planning	January 14	February 7	Milestone 2
Quality & DevOps Implementation	Static analysis setup CI/CD enforcement Security scanning integration	January 28	February 21	Checkpoint
Implementation & Core Documentation	Code/config updates Payment integration Notification setup Programmer & Ops guides	February 8	March 21	Milestone 3
Testing & Validation	Test execution Security validation Deployment verification	March 1	March 31	Milestone 4
Final Review & Delivery	User guide finalization Test reporting Final demos	March 15	March 31	Milestone 4

5.3 Gantt Chart

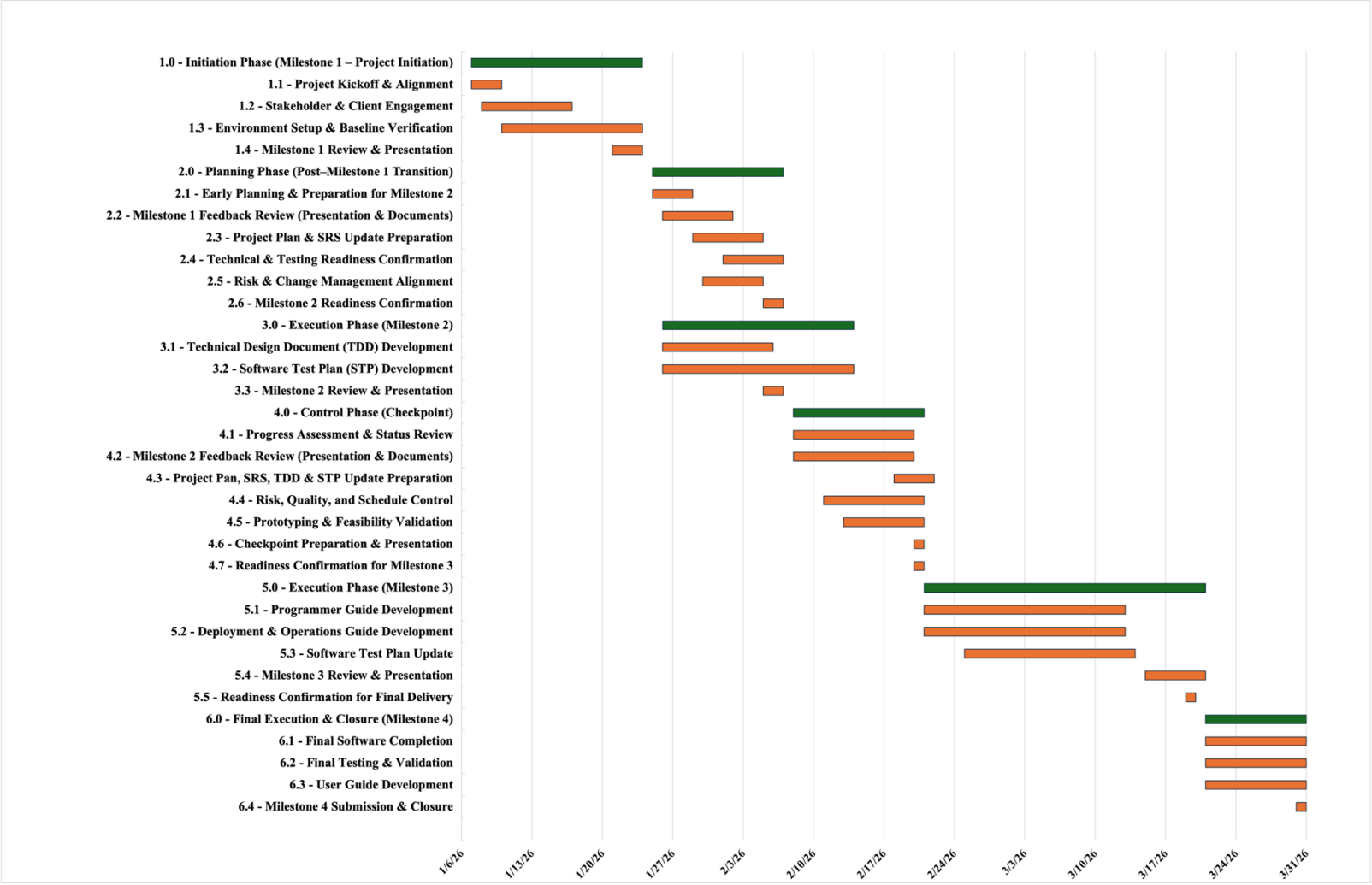


Figure 5.1 - Project Plan

A Gantt chart illustrating the phase-based project schedule and milestone alignment is included below. The chart presents major project phases only and does not enumerate task-level execution details.

6. Technical Architecture

This section defines the technical blueprint that converts the requirements outlined in the Software Requirements Specification (SRS) into a practical, cloud-based system design for CareConnect. The proposed architecture is intended to improve system robustness, enforce security controls, and support compliance-driven features while enabling the application to scale and evolve within a healthcare-adjacent environment.

The architectural design directly addresses four priority business needs established by the product owners: the introduction of standardized static code analysis with remediation requirements, enforcement of deployment controls through continuous integration pipelines, adoption of native billing mechanisms for Apple and Google platforms, and implementation of a centralized notification and reminder framework using AWS Simple Notification Service (SNS). Together, these focus areas ensure that CareConnect delivers core patient and caregiver functionality while embedding quality assurance, security validation, and operational safeguards into the system.

To achieve these goals, the architecture separates user-facing applications, backend business logic, third-party service integrations, and data storage into distinct layers. Mobile subscription payments are handled through platform-native billing services to meet app store compliance requirements, while subscription records and billing metadata are managed by backend services using a relational database. Automated pipelines are structured to run tests, static analysis, and security scans as part of every change, preventing code promotion when high-risk defects or vulnerabilities are present. Long-term maintainability is a key architectural consideration, with an emphasis on managed cloud services, consistent documentation, and tooling outputs that can be interpreted by future teams without specialized security training.

The supporting diagrams included in this section present a clear view of system components, runtime interactions, and deployment layout within AWS, providing future cohorts with the context needed to confidently operate and enhance the CareConnect platform.

6.1 Architectural Objectives

Objective	Rationale
Scalability	Must support adding thousands of patients without downtime while ingesting real-time wearable data.
Regulatory Compliance	HIPAA/GDPR drives our choices for encryption, audit logging, and isolated VPC design.
Time-to-Market	The eleven-week delivery window requires managed services (Fargate, Cognito, MSK) to minimize ops toil.
Cost-Efficiency	MVP phase should remain <$300 / month; autoscaling and spot pricing mitigate idle spend.
Extensibility	Event-driven, domain-oriented services allow future modules (e.g., Tele-health Bridge) to subscribe without direct coupling.

6.2 Logical Architecture

Tier	Logical Component	Responsibilities	Tech Choice
Client	Flutter Mobile + Web	UI, device APIs (camera, microphone), local cache, Stripe Elements, WebSocket listener	Flutter (Dart)
Platform Billing	Apple In-App Purchases, Google Play Billing	Native subscription purchase, renewal, cancellation, payment retries	StoreKit 2 (iOS), Google Play Billing (Android)
Edge	Amazon CloudFrontWAF	TLS termination, SPA hosting, DDoS & OWASP rules	AWS CloudFront, AWS WAF
API Gateway	REST & WebSocket endpoints	Routing, JWT validation (Cognito), request throttling	Amazon API Gateway

Tier	Logical Component	Responsibilities	Tech Choice
BFF (Service Mesh)	Spring Boot API-BFF	Aggregates downstream calls for mobile performance	Spring Boot
Domain Micro-Services	User-Service, Billing-Service, Scheduling-Service, CommunicationService, Analytics- Service	Each owns its schema; publishes Kafka topics	Spring Boot -> ECS Fargate
Messaging	Notification Service	Publish reminders and alerts across channels	AWS SNS (SMS, push, email triggers)
Data	Relational DB, Document DB, Time-series DB, Object Storage	Persistent storage for operational, analytical, and audit data	RDS (PostgreSQL), DynamoDB, Timestream, S3
Integretion	Stripe, Fitbit, Nest, FCM/APNs	External REST / Webhooks	HTTPS, JWT mutual-TLS
Streaming	Event Bus (“CareBus”)	Domain events, audit trail	SNS
Analytics	Athena / QuickSight	Ad-hoc queries, dashboards	Nightly ETL via Glue-Jobs
Security & Ops	Identity, Secrets, Monitoring	AuthN/Z, secrets, logs, scheduled jobs	Cognito, KMS, IAM, CloudWatch, EventBridge
CI/CD & Quality	Build & Enforcement Pipeline	Testing, static analysis, vulnerability scanning, deployment blocking	GitHub Actions, SpotBugs, PMD, Checkstyle, SonarQube
Observability	Logging & Monitoring	Application logs, alerts, health checks	AWS CloudWatch

6.3 High-Level Diagrams

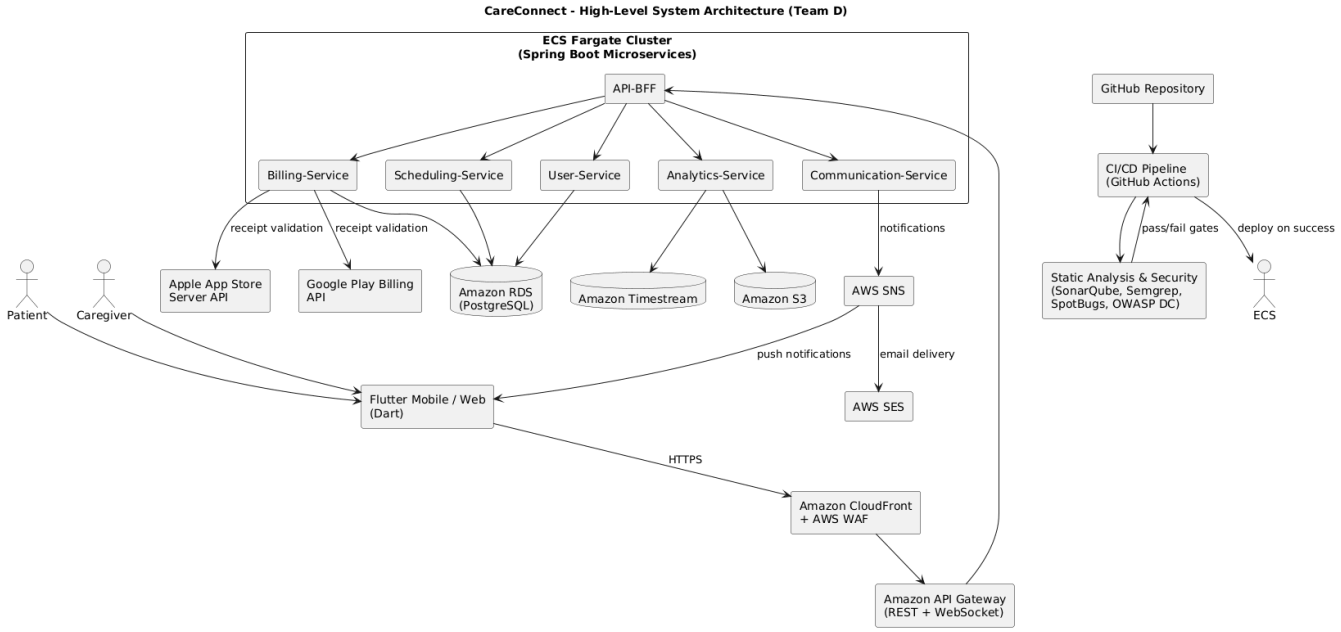


Figure 6.1 - High-Level System Architecture Diagram

6.4 Architecture Decisions & Justifications

Architectural Objective	Implementation Decision	Implementation Alignment
Scalability	Stateless Spring Boot services on ECS Fargate; event-driven messaging	Implements the Scalability objective by enabling horizontal scaling without service downtime through stateless compute and asynchronous event processing.
Latency	API-BFF aggregation, CloudFront caching, WebSockets	Supports responsiveness goals by reducing mobile roundtrips and improving perceived performance for client applications.
Software Quality & Security	Mandatory static analysis, SAST, and dependency scanning in CI	Enforces the Quality and Security objectives by preventing insecure or unstable builds from being promoted through automated CI/CD gates.
Mobile Platform Compliance	Native Apple and Google billing with backend receipt validation	Aligns with the Platform Compliance objective by ensuring all subscription flows meet App Store and Play Store policy requirements.
Regulatory Compliance (HIPAA/GDPR)	TLS, AES-256 encryption, IAM least privilege, audit logging	Implements regulatory requirements through encryption, access controls, and auditable system activity.
Cost Efficiency	Serverless-first services, autoscaling, managed infrastructure	Supports the Cost-Efficiency objective by minimizing idle resources and operational overhead during MVP and demo-scale usage.
Extensibility	Domain services and event-based communication	Enables future capabilities without tight coupling, directly supporting the Extensibility objective.
Development Velocity	Unified repository, automated pipelines, reproducible tooling	Improves developer efficiency and future cohort onboarding through consistent tooling and automated enforcement.

Architectural Objective	Implementation Decision	Implementation Alignment
Deployment Safety	CI/CD deployment blocking and blue/green releases	Prevents unstable or insecure releases by ensuring only validated builds reach deployed environments.
User Engagement & Reliability	Centralized notifications with retries and escalation	Implements reliability goals by ensuring reminders are delivered, tracked, and escalated consistently.

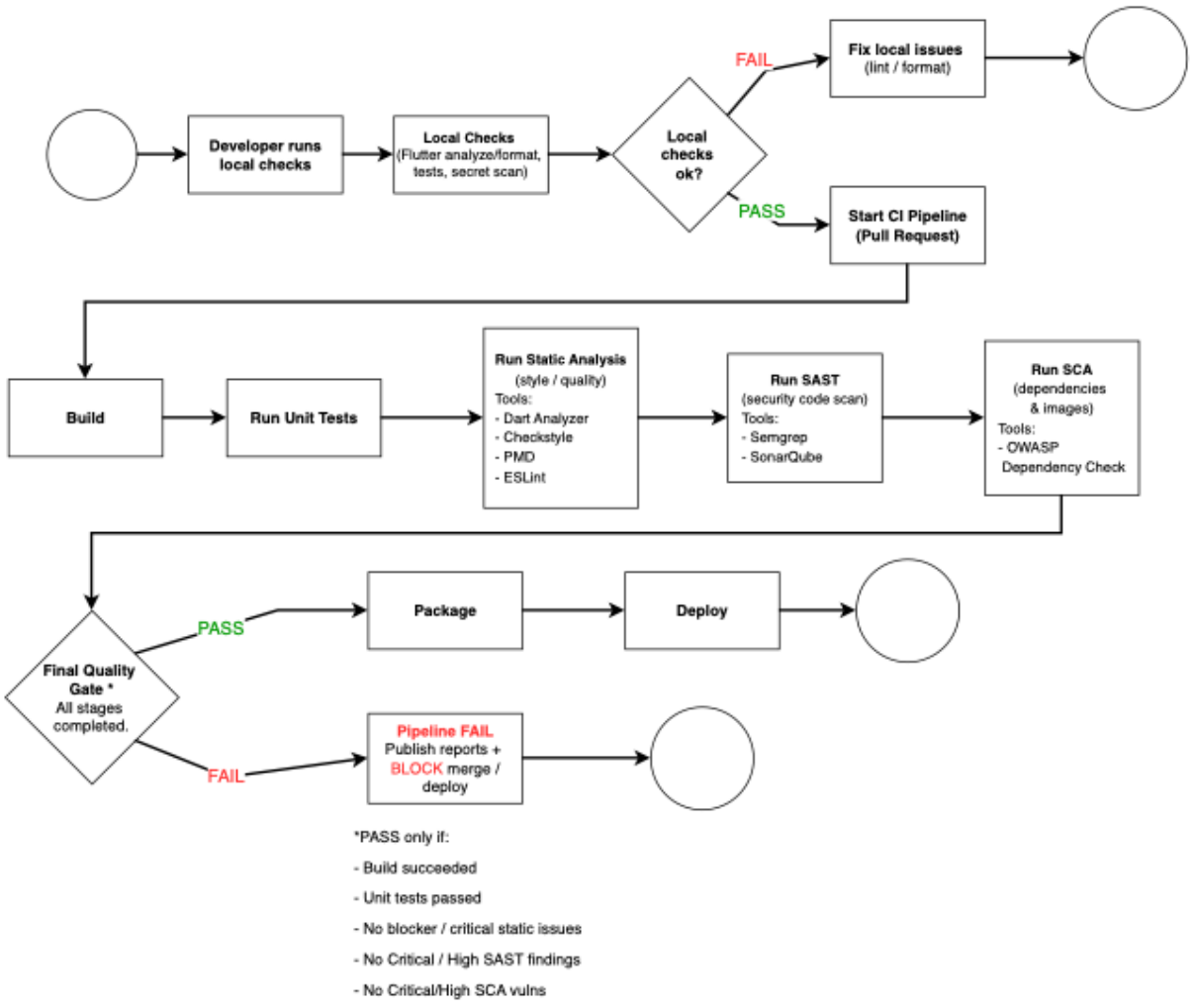


Figure 6.2 - CI/CD Gating Activity

Ensures each code change executes builds, tests, static analysis, security, and dependency checks to completion, producing a full set of reports, and only deploys artifacts when all required quality and security gates are satisfied.

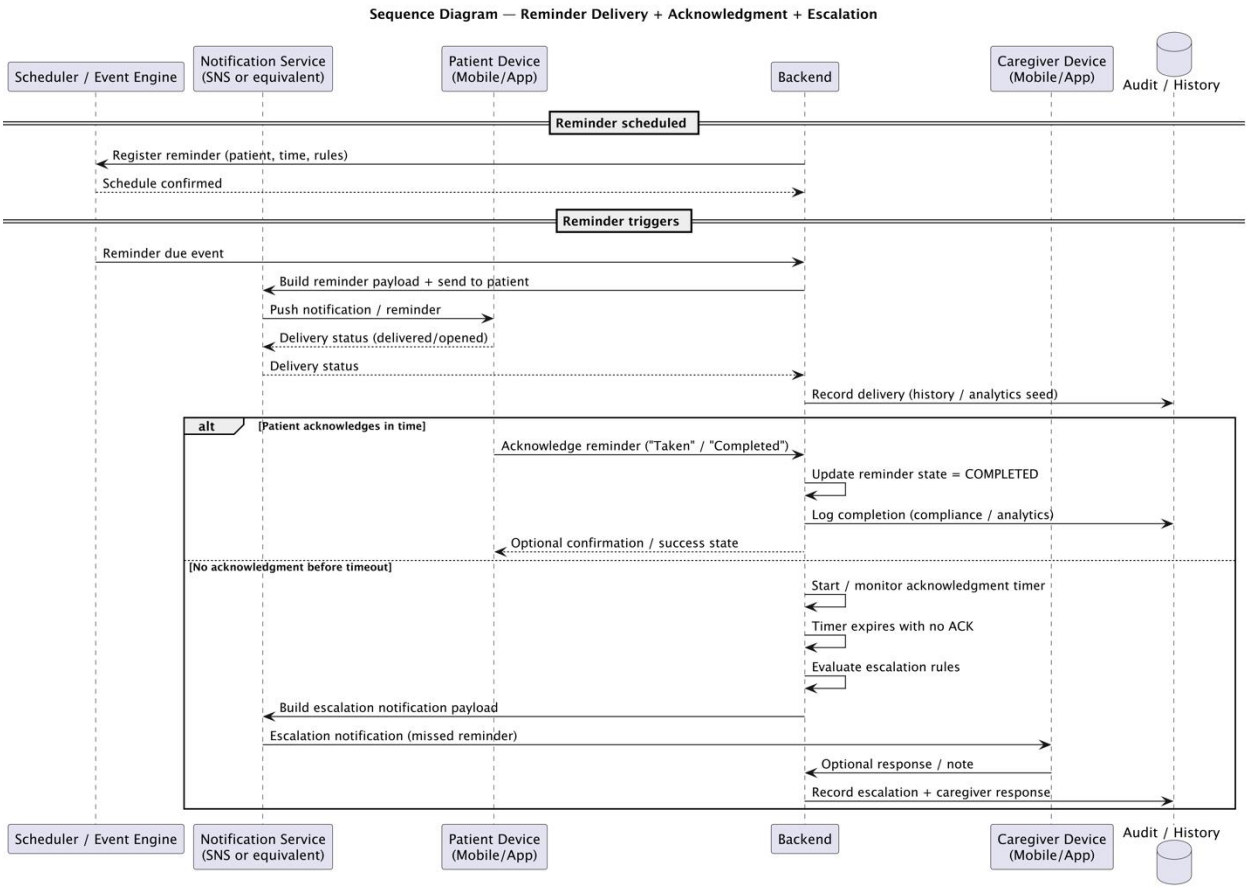


Figure 6.3 - Reminder Delivery, Acknowledgement and Escalation

Delivers reminders to patients, captures acknowledgments, and escalates missed reminders to caregivers while recording events for compliance.

6.6 Technology Stack Summary

Layer	Tech	Notes
Frontend	Flutter (Dart)	Single codebase for iOS, Android, Web
Backend	Java 17, Spring Boot	REST APIs, domain services
Database	PostgreSQL (RDS)	Transactional and subscription data
Messaging	AWS SNS	Notifications and event fan-out
CI/CD	GitHub Actions	Tests, scans, deployment blocking
Observability	CloudWatch	Logs, metrics, alarms
Security	Cognito, KMS, IAM	Authentication, encryption, access control

6.7 Deployment Topology

- All Spring services run as container images in an ECS Fargate cluster inside a private subnet; only API Gateway and CloudFront are publicly routable.
- Secrets (DB passwords, Stripe keys) are stored in AWS Secrets Manager; tasks assume IAM roles with the least-privilege policy generated via AWS SAM.
- Observability: structured JSON logs are shipped to CloudWatch; metrics and traces are exported via OpenTelemetry to a managed Grafana workspace.
- Blue/Green deployments are orchestrated by CodeDeploy. Rollback is automatic on canary health-check failures (<2 min).
- CI/CD pipelines enforce deployment blocking based on test failures and security scan results.
- Notification services are deployed as stateless consumers subscribed to CareBus events.
- Payment verification logic is isolated within the Billing Service to reduce platform coupling.

This section outlines a secure, scalable, serverless and cloud-native technical architecture for CareConnect. We will be using Amazon Web Services (AWS) to build the infrastructure and Flutter to build the user interface. This setup is well-suited for modern web and mobile applications that require secure access control, high availability while being able to scale to zero and integrate with internal AWS services and external third-party seamlessly.

6.8 Frontend

CareConnect's frontend is primarily designed for ease-of-use by both patients and caregivers, with configurable features integrated in the code to allow for user preference of the UX. The frontend will be comprised of code using Flutter and Dart, allowing for support from a wide range of platforms. The UI was designed with role-specific functionality in mind. This allows for separate features dependent on the user's role – patient or caregiver. Native Google Play Billing and Apple In-App Purchase flows are handled directly within the mobile client, with backend validation for subscription state consistency. Push notification listeners are registered to receive reminders and system alerts via SNS-backed delivery.

Main Pages:

- Welcome/Login page
- Caregiver/Patient dashboard
  Admin panel

6.9 Backend

The backend will constitute a set of AWS services for the core infrastructure and integrate external third-party resources and API(s). We will be using Terraform for infrastructure as code (IaC) management that would help to spin up, update, and take down all the resources needed in our infrastructure.

a) Amazon Cognito – *User Authentication and Authorization*

- Purpose: Manages user sign-up, sign-in, and access control.
- Features:
  - Supports OAuth2.0, SAML, and OpenID Connect.
  - Integrates with API Gateway to secure endpoints.
  - Enables multi-factor authentication (MFA) and social identity providers (Google, Apple, etc.).

b) Amazon API Gateway

- Purpose: Act as the entry point for all requests
- Features:
  - Routes requests to backend services.
  - Integrate with Cognito for authentication and authorization.
  - Supports rate limiting, throttling, and monitoring.

c) Amazon Elastic Container Service (ECS)

- Purpose: Host and orchestrate containers for the application, the backend.
- Features:
    - Supports Fargate for serverless type of launch
    - Integrate with Application Load Balancer (ALB) for traffic distribution
    - Scalability
    - Secure networking to block all external communications

d) Spring Boot – For backend development

- Purpose: Develop Rest API with Java
- Feature:
    - Robust and secure
    - Uses Java
    - Supports and integrates countless external technologies for all aspects.

e) Amazon Relational Database Service (RDS) – Managed SQL Database Server

- Purpose: Stores structured applications data.
- Features:
    - Supports engines like MySQL and PostgreSQL.
    - Offers automated backups, and read replicas
    - Encrypted at rest and in transit

For CareConnect, we will be using PostgreSQL.

f) Amazon S3 – for Object Storage

- Purpose: Stores static assets like image, documents.
- Features:
    - Cheap
    - Durable
    - Scalable

g) Amazon CloudWatch – for Monitoring and Logging

- Purpose: Provides observability into application and infrastructure
- Features:
    - Collects logs and metrics from ECS, API Gateway, RDS.

h) AWS Identity and Access Management (IAM) – Security and Permissions

- Purpose: Manage permission between AWS services, resources, and users.
- Features:
    - Enforces least privilege access (LPA)
    - ECS tasks assume IAM roles to access RDS, S3 and more

i) Static Analysis & Deployment Enforcement

- Backend services participate in CI/CD pipelines that execute unit tests, static analysis, and dependency scans
- Deployments are automatically blocked when quality thresholds are not met.

6.10 Database

The CareConnect database is designed to securely store and manage application data related to users, tasks, health records, and system operations. The database will ensure data integrity, support HIPAA compliance, and provide reliable access for both mobile and backend components.

Database Technology Stack:

- Database Type: Relational (SQL-based)
- Preferred Platform: PostgreSQL (hosted on AWS RDS, a cloud service)
- ORM/Access Layer: Dart packages compatible with Flutter, such as sqflite or drift for local caching if offline support is needed.

Key Database Components:

- Users Table
    - Stores account information for patients, caregivers, and admins. Includes authentication credentials, contact details, and role-based metadata.
- Patients Table
    - Contains patient-specific profiles, medical notes, emergency contacts, and linked caregiver relationships.
- Care Tasks Table
    - Stores both pre-defined and custom tasks related to care, such as medications, meals, and appointments, with timestamps, frequency, and completion status.
- Health Logs Table
    - Records manually entered data like vitals, symptoms, and mood tracking for patients, linked by patient ID and timestamped.
- Notifications Table
    - Logs system-generated alerts and reminders for both patients and caregivers, tracking delivery and response status.
- Billing Table
    - Manages billing events, payment history, subscription status, and Stripe transaction references.
- Subscription Metadata
    - Stores platform-neutral subscription state, renewal status, and references to Apple/Google transactions.

Security & Compliance Features:

- Role-based access enforced through middleware
- Data encryption at rest and in transit
- Regular backups and automated failover support
- Data retention and purging policies for compliance

Scalability Considerations:

- Support for multi-tenancy for different organizations or caregiver groups
- Optimized indexes for common query patterns (e.g., task retrieval by date or caregiver)
- Future-readiness for migration to a distributed database if user volume scales

6.11 APIs

The CareConnect platform requires robust and secure API integrations to support its modular feature set. Each subsection describes the purpose, integration model, and relevant considerations for the respective API.

6.11.1 Fitbit Web API

- Purpose: Retrieve heart rate and step count data from Fitbit devices.
- Authentication: OAuth 2.0
- Data Accessed: Heart rate, step count
- Platform Support: Android, iOS
- Considerations: Requires user account linking and periodic data sync.

6.11.2 Apple HealthKit API

- Purpose: Retrieve heart rate and step count data on Apple Health on iOS devices.
- Authentication: Native iOS permission-based access
- Data Accessed: Heart rate, step count
- Platform Support: iOS only
- Considerations: Requires explicit user consent for each metric.

6.11.3 Google Health Connect API

- Purpose: Retrieve heart rate and step count data on Android devices through a unified API for Google Fit and partner apps.
- Authentication: Android permission-based access
- Data Accessed: Heart rate, step count
- Platform Support: Android only
- Considerations: Health Connect to be installed and permitted.

6.11.4 Google Smart Device Management (SDM) API

- Purpose: Interact with Google Nest smart home devices for environmental monitoring.
- Authentication: OAuth 2.0
- Data Accessed: Motion events, device status, live camera availability
- Platform Support: Cloud-based
- Considerations: Must be carefully managed for privacy.

6.11.5 Alexa Smart Home Skill API

- Purpose: Enable control and monitoring of Alexa-compatible smart home devices.
- Authentication: OAuth 2.0
- Data Accessed/Controlled: Device state, control directives.
- Platform Support: Cloud-based
- Considerations: Integration requires Smart Home Skill certification and user account linking.

6.11.6 OpenFDA API

- Purpose: Retrieve official medication information using NDC codes.
- Authentication: None (public API)
- Data Accessed: Medication name, dosage, manufacturer, instructions
- Platform Support: Web-based
- Considerations: Used to auto-populate medication entries from scanned NDC codes.

6.11.8 Jitsi API

- Purpose: Use audio and video calls for various communication functions in CareConnect.
- Authentication: JWT Token
- Data Accessed: Voice, video, calendar schedule.
- Platform Support: Web-based. Apple iOS. Android.
- Considerations: Used for in-app messaging services. HIPPA compliant with set-up.

6.11.9 Apple App Store & Google Play Billing APIs

- Purpose: Validate and reconcile native mobile subscription transactions.
- Authentication: Platform-specific credentials.
- Considerations: Required for App Store and Play Store compliance.

6.12 AI Services

6.12.1 Purpose and User Flow

- Patients use the "Ask" feature in the app to type or ask questions about their treatment plans or symptoms.
- Utilize DeepSeek, a HIPAA/GDPR-compliant LLM, to power the "Ask" feature for patients to pose personalized care questions.
- If a patient's question looks risky or outside the AI's safe zone, it's held back for a real clinician to review before the patient sees it.

6.12.2 Technical Architecture

6.12.2.1 Frontend Integration

- Secure REST calls from the mobile app to an AI Gateway.

6.12.2.2 AI Gateway

- Handles authentication/authorization, rate limiting, and request routing to DeepSeek.

6.12.2.3 DeepSeek Service

- Retrieves patient context from the backend, invokes LLM, and returns structured JSON answers.

6.12.2.4 Review & Escalation

- Flag any high-risk queries for human clinician review before delivering to the patient.

6.12.3 Security and data Protection

- Encryption: All PHI (Personal Health Information) in transit and at rest must use AES-256 encryption, and TLS 1.2+ for network calls.
- Data Minimization: Only supply DeepSeek with the minimal necessary attributes (e.g., plan ID, symptom entries) and discard logs after compliance retention periods.
- Audit Logging: Maintain immutable logs of AI interactions, including timestamps, user IDs, and query transcripts, stored in a secure, access-controlled ledger.

6.12.4 Compliance and Governance

- Privacy by Design: Embed compliance controls into each component (e.g., tokenization, PII redaction) to ensure HIPAA and GDPR adherence from day one.
- Human-in-the-Loop: Escalate any AI responses outside predefined safety envelopes to clinicians for manual approval.
- Periodic Review: Conduct quarterly risk assessments and model audits to validate compliance and performance.

6.12.5 Summary and Future Enhancements

6.12.5.1 AI Service Summary

- The ability for patients to easily and efficiently get the information they need without having to wait for a human caregiver that might not be available is something that could significantly increase patient quality of life. The AI integration piece for the ‘Ask’ function is a great way to achieve this, though takes some careful planning to ensure we are implementing strict security best practices and making sure we keep the quality-of-care high through human review when necessary.

6.12.5.2 Future Enhancements

- Potential avenues for further AI integration would be automated care alerts for caregivers based on aggregating information on vitals and medicine reception by the patients and running LLM inference against the data to find anomalies. These anomalies would be reported back to caregivers in real-time as they are found to ensure the quickest triage and remedy.

6.13 Hosting

6.13.1 Cloud Deployment Model

- Adopt API Gateway as the entry point for all client requests, enforcing authentication and throttling.
- Use AWS Fargate with ECS to run containerized services (backend API, AI Gateway) in a serverless-managed cluster, eliminating server management overhead.
- CI/CD pipelines are integrated into the hosting lifecycle, preventing container promotion when quality gates fail.

6.13.2 Containerization and Orchestration

- Docker Images: Build stateless microservices, each with immutable Docker images stored in Amazon's ECR (Elastic Container Registry)
- ECS Task Definitions: Define CPU/memory limits and Amazon IAM (Identity and Access Management) roles per task, ensuring least-privilege access to AWS resources.
- Auto Scaling: Configure ECS Service Auto Scaling to adjust desired tasks based on CPU/memory metrics.

6.13.3 Serverless-First & Scale-to-Zero

- Leverage AWS Lambda to achieve scale-to-zero and cost efficiency.

6.13.4 Security and Reliability

- Containers live inside a private network segment with no direct public access, and outbound calls go through controlled gateways.
- We protect the front door with a web firewall and standard defenses against denial-ofservice attacks.
- Data backups and multi-zone deployments ensure that, even if one data center goes down, the service stays up, and data stays safe.

6.14 Data Protection

6.14.1.1 Encryption

- All Personal Health Information (PHI) is encrypted both when stored and during transmission, using the strong industry encryption standards of AES 256 for data being stored and TLS 1.2 for any data in transit between systems.

6.14.1.2 Data Minimization

- We limit the collection and use of personal data to what is necessary for providing services. Identifiable information is replaced with tokens where possible, and logs are anonymized to protect user privacy.

6.14.2 Authentication and Access Control

6.14.2.1 User Authentication

- Secure authentication mechanisms are in place, supporting multi-factor authentication (MFA) and social login options, to verify user identities effectively.

6.14.2.2 Role-Based Access Control (RBAC)

- Access to system features and data is determined by user roles (such as patient, caregiver, or administrator), ensuring users can only access information pertinent to their responsibilities.

6.14.2.3 Principle of Least Privilege

- Access permissions are granted based on the minimum necessary access required for a user's role, reducing potential security risks, and will be managed through AWS IAM (Identity & Access Management)

6.14.3 Secure APIs

6.14.3.1 API Gateway Protection

- All API traffic is managed through AWS API Gateway, a secure gateway that enforces rate limiting and input validation to prevent misuse and attacks.

6.14.4 Infrastructure Hardening

6.14.4.1 Network Isolation

- System components are deployed within isolated virtual networks, preventing direct exposure to the internet and reducing the attack surface.

6.14.4.2 Web Application Firewall (WAF)

- A firewall is in place to protect against common web exploits such as SQL injection and cross-site scripting (XSS).

6.14.4.3 Firewall Rules

- Strict firewall configurations control traffic between system components, allowing for only necessary communication.

6.14.5 Monitoring and Logging

6.14.5.1 Real-Time Monitoring

- CareConnect will leverage the capabilities of AWS CloudWatch to monitor threats to the system continuously and actively, monitoring administrators whenever an anomaly is detected.

6.14.5.2 Audit Logging

- All critical actions, including logins, data modifications, and AI interactions, are recorded in secure and unchangeable logs to support auditing and compliance efforts.

6.14.6 Vulnerability Management

6.14.6.1 Vulnerability Management

- Static application security testing (SAST) is executed automatically during CI.
- Vulnerability findings are logged, tracked, and must be resolved before deployment.

6.7.6.1 Interval Scanning

- The system undergoes routine scans to identify and address vulnerabilities in both backend and mobile components.

6.14.6.2 Automated Patching

- Security patches are applied automatically to system components as part of our continuous integration and deployment processes, ensuring timely updates.

6.14.7 Compliance Measures

6.14.7.1 Regulatory Compliance

- The system is designed with privacy and security considerations from the outset, aligning with HIPAA and GDPR requirements.

6.14.7.2 Data Retention and Deletion

- Personal data is retained only as long as necessary for service provision, and users can request data deletion in accordance with legal guidelines.

7. Project Deliverables

The project will produce the following deliverables:

- Configured static analysis, dependency scanning, container scanning, and dynamic security tools integrated into CI/CD.
- CI/CD pipelines with enforced quality and security gates.
- Remediation of all critical and high-severity findings, or documented justification for unresolved issues.
- A maintained record of unresolved vulnerabilities carried forward for future cohorts.
- Apple native billing implementation for iOS and Google native billing for Android and web.
- Centralized AWS SNS/SES-based notification and reminder infrastructure.
- Messaging and CI/CD architecture diagrams.
- Updated repository documentation, including local setup and validation instructions.
- Deployed environments used for milestone demonstrations.

Table 7-1. Project Deliverables

Deliverable	Description
Static Analysis Integration	SpotBugs, PMD, Checkstyle configured locally and in CI
Security Scanning	Vulnerability, dependency, container, and DAST scans
CI/CD Pipelines	GitHub Actions with enforced quality gates
Vulnerability Remediation	Critical/high issues fixed or justified
Vulnerability Record	Carry-forward log for future cohorts
Payment Integration	Apple & Google native billing
Notification System	AWS SNS/SES-based messaging
Architecture Diagrams	CI/CD and messaging architecture
Documentation	README, setup, validation steps
Deployed Demos	Demonstrations from deployed environments

Definition of Done (DoD) for Deliverables

The Definition of Done (DoD) below applies to all deliverables listed in Table 7-1 and is used to determine milestone readiness and acceptance.

- **Static Analysis Integration:** tools run locally and in CI with documented setup steps and example outputs
- **Security Scanning:** SAST/SCA/DAST reports generated and uploaded as artifacts
- **CI/CD Pipelines:** merge blocking enforced for failed tests or critical/high findings
- **Vulnerability Remediation:** all critical/high issues fixed or justified in a carry-forward log
- **Payment Integration:** Apple/Google billing flow demonstrated with backend validation behavior documented
- **Notification System:** SNS/SES delivery flow demonstrated with retry/escalation behavior documented
- **Documentation:** README includes build + test instructions and demo evidence references

8. Risk & Mitigation

Key project risks and mitigation strategies include:

- **Risk:** Static analysis tools produce false positives that block progress
- **Mitigation:** Tune rulesets, prioritize severity thresholds, and document justified exclusions.
- **Risk:** CI/CD enforcement disrupts development velocity
- **Mitigation:** Stage enforcement, provide clear documentation, and validate locally before commits.
- **Risk:** Payment implementation fails platform compliance checks
- **Mitigation:** Follow Apple and Google billing guidelines and validate workflows prior to release.
- **Risk:** Notification escalation logic is misconfigured
- **Mitigation:** Use centralized AWS services, logging, retries, and clear escalation rules.
- **Risk:** Future cohorts cannot interpret security results
- **Mitigation:** Use non-specialist-friendly tools, standardized severity levels, and clear documentation.

A probability–impact risk matrix will be maintained to track exposure and ensure timely mitigation.

8.1 Risk Probability Matrix

Table 8-1. Risk Probability Matrix

Impact \ Probability	High (≥80%)	Medium (60–79%)	Low (30–59%)	Very Low (<30%)
High Impact	Very High Risk	High Risk	Moderate Risk	Moderate Risk
Medium Impact	High Risk	Moderate Risk	Low Risk	Low Risk
Low Impact	Moderate Risk	Low Risk	Very Low Risk	Very Low Risk

8.2 Risk Identification & Mitigation

Table 8-2. Risk Register

Risk ID	Description	Probability	Impact	Risk Level	Mitigation
R-01	Static analysis false positives block delivery	Medium	High	High	Tune rules, severity thresholds
R-02	CI/CD enforcement disrupts velocity	Medium	Medium	Moderate	Stage enforcement
R-03	App store rejection due to billing errors	Low	High	Moderate	Follow platform billing rules
R-04	Security findings unclear to future cohorts	Medium	Medium	Moderate	Documentation, tool choice
R-05	Notification escalation misconfigured	Low	Medium	Low	Centralized SNS logic

9. Budget Estimate

The budget estimate reflects a time-boxed academic project delivered by a student team, primarily using open-source tooling and AWS Free Tier services. Costs represent an estimated real-world equivalent value, not actual cash expenditure by the team.

9.1 Labor Cost

Labor costs are calculated using industry-standard blended rates to represent the effort required if the project were delivered in a professional environment.

Role	Estimated Hours	Rate (USD/hr)	Subtotal
Project Manager	60	$48.85	$2,931.00
Team Lead	180	$82.31	$14,815.80
Business Analyst	105	$48.65	$5,108.25
Technical Lead / Architect	180	$62.69	$11,284.20
Software Development	270	$63.20	$17,064.00
Testing & Validation	150	$63.20	$9,480.00
Total Labor	**945**	**---**	**$60,683.25**

9.1.1 Labor Cost Assumptions

Labor cost estimates represent an approximate real-world equivalent for a time-boxed academic project delivered by a student team. Hour allocations assume an estimated **12 week project duration**, with effort distributed across roles based on responsibility and project phase rather than continuous full-time engagement.

Leadership roles (Project Manager, Team Lead, Technical Lead) reflect higher coordination, governance, and oversight effort across the project lifecycle, while development and testing roles focus on implementation and validation activities aligned with milestone deliverables.

Testing and validation effort is scoped to **targeted functional, security, and deployment verification** required for demonstrations and acceptance, rather than full production-scale quality assurance. Labor estimates are not intended to represent contractual staffing levels or operational support costs.

9.2 Materials Cost

Most tooling leverages open-source or academic/free-tier licenses.

Tool / Resource	Cost Basis	Estimated Cost
SpotBugs, PMD, Checkstyle	Open Source	$0
SonarQube Community	Free Tier	$0
OWASP Dependency-Check	Open Source	$0
Trivy	Open Source	$0
OWASP ZAP	Open Source	$0
GitHub Action	Free Tier	$0
Total Materials Cost	**---**	**$0**

9.3 Infrastructure Cost

Infrastructure costs are estimated assuming Low volume, demo-scale usage aligned with the project.

Service	Estimated Monthly Cost
AWS ECS Fargate	$40
AWS RDS	$50
AWS SNS / SES	$10
CloudWatch Logging & Metrics	$15
S3 Storage	$5
Estimated Monthly Total	**$120**

Estimated project duration: 3 months

Total Infrastructure Cost: $360

Infrastructure Cost Assumptions

- Demo-scale usage (limited concurrent users)
- Single staging environment only (no full production redundancy)
- Minimal outbound SMS/email volume during demos
- AWS Free Tier assumed where applicable
- Costs reflect estimated baseline services only (not peak scaling)

9.3.1 Infrastructure Cost Assumptions

Infrastructure costs are estimated assuming **low-volume, demo-scale usage** aligned with an academic project context. Estimates assume a **single non-production (staging) environment**, limited concurrent users, and minimal external traffic during milestone demonstrations.

AWS Free Tier usage is assumed where applicable, and costs reflect **baseline service consumption only**, not peak load, redundancy, or production-grade scaling. Infrastructure estimates are provided for planning transparency and do not represent guaranteed or actual cloud expenditures.

9.4 Total Estimated Cost

Category	Cost
Labor	$60,683.25
Materials & Tools	$0
Infrastructure (3 months)	$360
Total Estimated Project Cost	$61,043.25

10. Roles & Responsibilities

10.1 Roles

Roles & Responsibilities Table with Assigned Team Members

Member	Role	Responsibility
Dr. Mir Assadullah	Client	Provide product requirements and feedback on product progress
Roy Gordon	Advisor	Advise student team members
Sukhmeet Khalar	Project Owner	Advise Students on Technical Nature of Project
Sudarshan Neupane	Project Owner	Advise Students on Technical Nature of Project
James Stevens	Team Lead	Organize project team deliverables Provide project progress to project owner
Justinah Bashua	Technical Lead/Architect	Design Project Architecture and handle Technical Issues
Brandon Sutan	Business Analyst	Identify Use Cases Coordinate with Leads to determine functional capabilities of use cases Meet with stakeholders to discuss functional requirements
Matthew Lester	Lead Software Developer	Lead Team Members in implementing code Ensure components are implemented correctly
Corey Bayliss	Lead Tester	Create Test Matrix for unit, integration, and system testing
Justinah Bashua	Backup Team Lead	Assume Team Lead when Team Lead is unavailable
Brandon Sutan	Backup Technical Lead/Architect	Assume Technical Lead/Architect when Technical Lead/Architect is unavailable

Member	Role	Responsibility
Matthew Lester	Backup Business Analyst	Assume Business Analyst when Business Analyst is unavailable
Corey Bayliss	Backup Software Developer Lead	Assume Lead Software Developer when Lead Software Developer is unavailable
James Stevens	Backup Tester	Assume Lead Tester when Lead Tester is unavailable

10.2 RACI Matrix

Workstream/Deliverables	TL	BA	Dev	Tech	Test
Milestone 1: Project Plan	R/A	R/A	R/A	R/A	R/A
Weekly Status reports	R/A	I	I	I	I
SRS	R/A	R/A	R/A	R/A	R/A
Design & Architecture					
UX research, personas, user flows, wireframes, prototypes					
UI design system (components, accessibility specs)					
Architecture & API design (contracts, versioning)					
Privacy & security design (threat model, roles, encryption, retention)					
Build					
Backend/API implementation					
Mobile app client implementation					

Workstream/Deliverables	TL	BA	Dev	Tech	Test
Dev environments & CI/CD (branching, pipelines, secrets, IaC)					
Test & Quality					
Test strategy & plan (system, integration, performance, accessibility)					
Test cases & execution; defect triage					
UAT planning & facilitation with customer					
Release & Ops					
Release management (cut, tag, deploy to staging/prod)					
Runbook, support & handover; training materials					
Compliance & Course					
Privacy & security review (PII inventory, access controls, policy)					
Peer reviews & individual contribution logs (grading)					

Matrix Table Key:

- R - Responsible
- A - Accountable
- C - Consulted
- I - Informed

11. Monitoring & Reporting

Effective monitoring and reporting are essential to ensure that project progress, quality enforcement, and risk exposure remain visible and controlled throughout the project lifecycle. This project combines automated monitoring through CI/CD pipelines with structured human oversight to ensure alignment with project objectives, product owner expectations, and academic requirements.
Monitoring focuses on build and test execution, static analysis findings, security scan results, and deployment readiness. Reporting ensures that issues are escalated through a defined chain of command and that stakeholders receive timely, accurate information to support decision-making.

11.1 Intra-Team Monitoring & Reporting

The team will monitor project progress and software quality using a combination of automated tooling and shared project tracking mechanisms. Source control, pull requests, and continuous integration execution will be managed through GitHub, which serves as the authoritative system of record for build status, test results, static analysis outputs, and security scan outcomes.
Each commit and pull request will automatically trigger unit tests, static analysis, and security scans. Results will be visible through CI/CD pipeline outputs, and failures will block merges or deployments in accordance with enforced quality gates.
Task progress, ownership, and status will be tracked using a shared tracking mechanism selected by the team (e.g., GitHub Projects, shared tracker, or equivalent). Regular internal reviews will be conducted to identify blockers, unresolved vulnerabilities, or deviations from plan.
When CI/CD failures, blocked merges, or critical findings occur, escalation will follow a defined chain of command:

1. Initial review and response by team leads
2. Escalation to product owners or professor if unresolved or release-impacting
3. Formal documentation through the change management process when required

Key monitoring artifacts include:

- CI/CD pipeline status and logs
- Unit test results
- Static analysis and security scan reports
- Pull request approval status
- Deployment status for staging and production environments

Local developer environments are used for development and validation only and are not considered authoritative sources for milestone demonstrations or acceptance decisions.

11.2 Project Communications

Clear and consistent communication channels are maintained to ensure transparency and timely decision-making. **Email is the primary and official communication method** for interactions with product owners and the professor. This ensures discussions, decisions, and approvals are documented and traceable.

Internal team coordination may use collaboration tools as appropriate; however, any decisions affecting scope, quality enforcement, security posture, or deployment must be recorded in project documentation or the Decision Tracker.

Both **weekly status updates and milestone-based reporting** will be used:

- Weekly updates summarize progress, CI/CD health, and open risks
- Milestone updates focus on deliverable readiness and quality gate status

Reporting Frequency & Format

- **Weekly Status Update (Email):** every week (summary of progress, risks, CI/CD health)
- **CI/CD Reporting:** automatic per push/pull request (GitHub Actions logs + artifacts)
- **Milestone Reporting:** submission package including deliverable PDFs + CI/CD proof
- **Escalation Reporting:** within 24 hours for blocked merges, critical/high findings, or billing compliance issues

Escalations related to blocked deployments, unresolved critical or high-severity findings, or platform compliance concerns will be communicated promptly, starting with team leads and proceeding through the established chain of command.

Milestone demonstrations will be performed against deployed environments (e.g., staging) using builds that have passed all required CI/CD quality and security gates. Developer workstations may be used solely as presentation interfaces (e.g., browser or client access) and not as execution environments for demonstrated functionality. CI/CD pipeline evidence will be shown for the exact commit/build being demonstrated. Demonstration evidence will include:

- Application access details (such as deployed URLs, endpoints, or runtime configuration information, as applicable)
- CI/CD pipeline results associated with the demonstrated build
- Confirmation that all required quality and security gates passed prior to demonstration

This approach ensures that demonstrated functionality reflects a validated build state while accommodating evolving deployment and hosting arrangements during the project lifecycle.

12. Assumptions

The following assumptions have been identified for the CareConnect project and evaluated for alignment with the defined Scope of Work (see Section 3). Assumptions that fall outside the approved scope are clearly identified to maintain transparency. If any assumption is later disproven, it will be escalated to the project Risk Register in accordance with quality management procedures.

- **Device Compatibility** - It is assumed that all end users, including patients, caregivers, and administrators, will access CareConnect via modern smartphones capable of running current iOS or Android operating systems. This aligns with the Scope of Work, which excludes dedicated hardware or wearable device support for the MVP.
- **Language Support** - The initial release will support English only. Multilingual functionality is considered out of scope for the MVP to ensure consistent usability testing and baseline measurements.
- **Medication and Task Management** - Medication lists, tasks, and related documentation will be entered manually by caregivers and patients. Integration with third-party medication management systems is out of scope for the MVP. API integrations are limited to other areas such as wearables and scheduling. Usability will be monitored, and this assumption may be escalated to a project risk if manual entry negatively impacts user adoption.
- **Notifications** - Notifications at launch will be limited to in-app alerts, consistent with the Scope of Work. Email and SMS notifications are explicitly excluded from the MVP.
- **User Technical Proficiency** - It is assumed that a portion of users will have limited technical skills. Advanced accessibility features, including screen reader support and high-contrast modes, are out of scope for the MVP. As a result, the user interface must prioritize simplicity, clarity, and minimal learning requirements to support a broad user base.

Summary of Assumptions

- Users will have access to modern smartphones.
- The MVP will support English only.
- Notifications will be delivered in-app only.
- Medications and tasks will be manually entered by caregivers and patients.
- Accessibility support will be limited to basic functionality.

13. Constraints

The project is subject to the following constraints, which influence planning, execution, and delivery decisions. These constraints are considered fixed conditions and are not expected to change without formal change management approval.

13.1 Schedule Constraints

The project must adhere to course-defined milestones, submission deadlines, and presentation dates. Deliverables are required at specific points in the academic term, limiting schedule flexibility and requiring parallel execution of activities.

13.2 Technology Constraints

The project is constrained to the existing CareConnect technology stack, including Flutter for the frontend, Java-based backend services, PostgreSQL for data persistence, AWS-managed services for infrastructure, and GitHub Actions for CI/CD. The introduction of alternative platforms, frameworks, or languages is out of scope unless explicitly approved through formal change management.

13.3 Platform and Compliance Constraints

Payment processing must comply with Apple and Google platform requirements for native billing systems. The project must also comply with applicable healthcare and data protection regulations, including HIPAA, and applicable privacy standards governing the handling of PII and PHI.

13.4 Quality and Security Constraints

Automated quality enforcement mechanisms—including static analysis, security scanning, and CI/CD gating—must be applied consistently across all environments. Builds, merges, or deployments may not proceed when defined quality or security thresholds are not met, constraining development velocity in favor of reliability, security, and compliance.

13.5 Resource Constraints

The project is executed by a fixed-size student team with limited availability and no dedicated operational or support staff. Team members balance project responsibilities with academic workloads and external commitments.

13.6 Dependency Constraints

The project depends on third-party platforms and services, including Apple and Google payment systems and AWS-managed services. Changes to external APIs, platform policies, or service availability may impact implementation and delivery timelines.

13.7 Tooling and Environment Constraints

Development, testing, and deployment activities are constrained by the availability of local development environments, CI/CD infrastructure, and cloud resources. Demonstrations and validations must be executed from deployed environments rather than individual developer machines.

14. Work Breakdown Structure (WBS)

The WBS decomposes the project into manageable, verifiable work packages.

14.1 Project Management

- Project Planning & scheduling
- Stakeholder coordination
- Risk management & reporting

14.2 Requirements & Documentation

- Business needs analysis
- SRS development
- Acceptance criteria definition
- Documentation standards alignment

14.3 Static Analysis & Security Tooling

- SpotBugs integration
- PMD rule configuration
- Checkstyle enforcement
- SonarQube/SonarCloud setup
- Dependency scanning (OWASP)
- Container scanning (Trivy)
- DAST integration (OWASP ZAP)

14.4 CI/CD Enforcement

- GitHub Actions pipeline design
- Quality gate enforcement
- Branch protection rules
- Deployment blocking logic

14.5 Vulnerability Remediation

- Critical vulnerability remediation
- High Severity remediation
- Justification documentation for deferred issues.

14.6 Payment Migration

- Apple In-App Purchase integration
- Google Play Billing integration
- Backend receipt validation
- Stripe depreciation (conditional)

14.7 Notification System

- AWS SNS configuration
- AWS SES email integration
- Reminder scheduling logic
- Notification history persistence

14.8 Testing & Validation

- Unit testing
- Integration testing
- Security validation
- Deployment verification

14.9 Final Documentation & Delivery

- Programmer guide
- Deployment & operations guide
- User guide
- Final demonstrations

14.10 WBS Alignment to Schedule and Roles

Each WBS workstream maps directly to the phase-based schedule in Section 5.2 and the role assignments defined in Section 10. For example, Static Analysis & Security Tooling (14.3) and CI/CD Enforcement (14.4) align primarily with the “Quality & DevOps Implementation” phase, while Testing & Validation (14.8) aligns with the “Testing & Validation” phase. Responsibility for each workstream is governed by the RACI matrix in Section 10.2.

15. Change Management Procedures

Given the project's emphasis on software quality enforcement, security compliance, and deployment control, all changes must follow a formal change management process. Uncontrolled changes to tooling, CI/CD enforcement, payment workflows, or notification mechanims could introduce security risk, violate platform requirements, or weaken delivery guarantees.

This project follows a three-step change control process:

> Initiation → Review → Authorization and Implementation
> All change requests are logged in a centralized Decision Tracker to ensure traceability, accountability, and continuity for future cohorts.

15.1 Initiation

Any team member may initiate a change request by submitting a standardized Change Request Form (CC-CR-001). A change request is required for any modification that affects scope, documentation, tooling, CI/CD enforcement, security posture, payment workflows, notification behavior, or deployment controls.Change initiation may occur through **documentation-driven** or **code/configuration-driven paths**. While the initiation steps differ, **all changes converge into the same review, approval, and tracking process**.

15.1.1 Documentation-Initiated Change Path

Documentation-initiated changes apply to updates of project plans, architectural diagrams, policies, procedures, governance artifacts, carry-forward records, or explanatory documentation that defines or constrains system behavior.

Initiation steps:

1. The initiator identifies the documentation requiring change (e.g., Project Plan section, diagram, appendix, or policy).
2. The initiator drafts the proposed documentation update or clearly describes the required change.
3. The initiator completes the Change Request Form (CC-CR-001), specifying:
   - Affected Area(s): Documentation and any related domains (e.g., CI/CD, Security).
   - Affected Artifacts: Specific document names, sections, or diagrams.
   - Reason for Change: Clarification, correction, alignment with product owner guidance, or carry-forward requirement.
4. The completed form is submitted to the Change Control Board for review.

Special handling considerations:

- Documentation-only changes that do not alter system behavior, enforcement rules, or approval requirements may be eligible for expedited review.
- Documentation changes that redefine processes, approval chains, quality thresholds, or security requirements are treated as high-impact changes, even if no code is modified.
- Approved documentation changes must be version-controlled and referenced in the Decision Tracker to ensure traceability for future cohorts.

15.1.2 Code- or Configuration-Initiated Change Path

Code- or configuration-initiated changes apply to application source code, CI/CD pipelines, static analysis tooling, security scanners, infrastructure definitions, deployment configurations, or runtime behavior.

Initiation steps:

1. The initiator identifies the need for change through development work, failed tests, security findings, scan results, or operational issues.
2. A pull request or proposed configuration change is prepared in the appropriate repository or branch.
3. The initiator completes the Change Request Form (CC-CR-001), specifying:
    - Affected Area(s): Code, CI/CD, Security, Payments, Notifications, or Infrastructure.
    - Affected Artifacts: Repositories, pipeline files, configuration scripts, or services.
    - Impact Analysis: Expected effects on security, quality gates, deployment behavior, or compliance.
4. The change request is submitted for review and linked to the corresponding pull request or configuration change.

Special handling considerations:

- All code- or configuration-initiated changes must pass automated testing, static analysis, and security scans as part of continuous integration.
- Changes affecting quality gates, deployment blocking behavior, or production environments require explicit approval prior to merge or release.
- Code changes may not be promoted to master or production branches without satisfying both technical validation and formal approval requirements.

15.1.3 Convergence and Governance

Regardless of initiation path, all change requests:

- Use the same Change Request Form (CC-CR-001)
- Are reviewed by the same Change Control Board
- Are approved or rejected based on impact, not artifact type
- Are logged in the centralized Decision Tracker

This approach ensures consistent governance while providing clear, practical guidance on how changes are initiated in day-to-day project work.

15.1.4 Change Initiation and Approval Flow

The change management workflow follows a single, unified path regardless of whether a change originates from documentation or code:

1. **Change Identified**
   A need for change is identified through documentation review, development activity, failed tests, security findings, product owner feedback, or compliance requirements.

2. **Initiation Path Selected**
   The initiator follows either the documentation-initiated or code/configuration-initiated path to prepare the proposed change.

3. **Change Request Submitted**
   The initiator completes and submits the standardized Change Request Form (CC-CR-001), identifying affected areas, artifacts, severity, and impact.

4. **Change Control Board Review**
   The Change Control Board evaluates the request for alignment with project objectives, security posture, compliance requirements, and delivery risk.

5. **Approval Decision**
   The change is approved, approved with conditions, or rejected. High-impact changes require explicit authorization.

6. **Implementation and Validation**
   Approved changes are implemented using controlled, version-tracked updates. Code and configuration changes are validated through automated testing, static analysis, and security scanning.

7. **Closure and Recordkeeping**
   Outcomes are documented, implementation status is updated, and the change is logged in the Decision Tracker for auditability and future cohort reference.

15.2 Review

Change requests are reviewed by a Change Control Board (CCB) consisting of the course instructor (final decision authority), product owners, project leadership, and relevant technical leads. The CCB evaluates alignment with project objectives, healthcare data sensitivity, security posture, and delivery risk. Changes affecting production deployment, quality gates, or billing behavior require explicit approval. Low-impact changes may be fast-tracked; high-impact changes require formal approval and updated documentation.

15.3 Authorization and Implementation

Approved changes are documented, version-controlled, and implemented through controlled updates to source code, pipeline definitions, or configuration artifacts. Continuous integration pipelines must validate changes through automated testing, static analysis, and security scanning. Merges to master or production branches require designated reviewer approval, and demonstrations must be executed from deployed environments rather than developer machines. All approved changes and outcomes are recorded to support auditing and future cohort handoff.

16. Appendices

Appendix A - Glossary of Terms

Americans with Disabilities Act (ADA)
A United States civil rights law that prohibits discrimination against individuals with disabilities and establishes accessibility requirements for digital systems and services.

Android OS
Google's mobile operating system used on Android devices, supporting native Google payment and application services.

Application (App)
A software program designed to perform specific tasks for end users, typically on mobile, web, or desktop platforms.

Application Programming Interface (API)
A defined set of rules and protocols that enable communication and data exchange between software components.

Artificial Intelligence (AI)
The simulation of human intelligence in software systems capable of tasks such as reasoning, prediction, pattern recognition, and decision-making.

Amazon Web Services (AWS)
A cloud computing platform providing infrastructure and managed services such as compute, storage, networking, and messaging.

Amazon Simple Email Service (AWS SES)
An AWS-managed service used for sending and receiving email notifications at scale.

Amazon Simple Notification Service (AWS SNS)
An AWS-managed messaging service used to deliver notifications via SMS, email, push notifications, or other endpoints.

American Sign Language (ASL)
A complete, natural language used by the Deaf community, referenced in accessibility and communication feature considerations.

Business Needs Statement (BNS)
A documented description of a business problem and the required solution or capabilities needed to address it.

CareConnect
The healthcare application being developed to support care coordination, payments, and notifications.

Caregiver
An individual responsible for providing care or assistance to a care recipient within the application context.

Care Recipient
An individual who receives care or services facilitated through the application.

Continuous Integration / Continuous Deployment (CI/CD)
Automated processes used to build, test, validate, and deploy software changes consistently and reliably.

Dart Tooling
Development, build, formatting, and static analysis tools used for writing and maintaining Dart code.

Database (DB)
A structured system used to store, manage, and retrieve application data.

Electronic Health Record (EHR)
A digital record of a patient's medical history maintained by healthcare providers.

Flutter
Google's UI toolkit for building cross-platform applications using the Dart programming language.

Gantt Chart
A project management visualization used to represent schedules, phases, and dependencies over time.

General Data Protection Regulation (GDPR)
A European Union regulation governing the collection, processing, storage, and protection of personal data, emphasizing consent, transparency, and individual data rights.

Graphical User Interface (GUI)
A visual interface that allows users to interact with an application through graphical elements such as buttons, menus, and icons.

Health Insurance Portability and Accountability Act (HIPAA)
A United States federal law that establishes standards for the protection, privacy, and secure handling of Protected Health Information (PHI) by healthcare entities and their business associates.

Identity and Access Management (IAM)
Security processes and technologies used to manage user identities, authentication, and authorization within a system.

Internet of Things (IoT)
A network of physical devices that collect and exchange data through embedded sensors and connectivity.

iOS
Apple's mobile operating system used on iPhones and iPads, requiring native Apple payment systems for in-app purchases.

Key Performance Indicator (KPI)
A measurable value used to evaluate system performance or business objectives.

Machine Learning (ML)
A subset of artificial intelligence that enables systems to learn and improve from data without explicit programming.

Natural Language Processing (NLP)
A branch of artificial intelligence focused on enabling systems to understand, interpret, and process human language.

Personally Identifiable Information (PII)
Information that can be used to identify an individual, directly or indirectly, such as names, contact details, or government-issued identifiers.

PostgreSQL
An open-source relational database management system used for persistent application data storage.

Protected Health Information (PHI)
Individually identifiable health information, including medical records and treatment data, subject to strict privacy and security requirements under healthcare regulations such as HIPAA.

Quality Assurance (QA)
Processes and activities used to ensure software meets defined quality standards through testing and validation.

RACI (Responsible, Accountable, Consulted, Informed)
A responsibility assignment matrix used to clarify roles and responsibilities in project management.

Recovery Point Objective (RPO)
The maximum acceptable amount of data loss, measured in time, that may occur due to a system failure or disruption.

Recovery Time Objective (RTO)
The maximum acceptable duration that a system or service can be unavailable following a disruption before operations must be restored.

Representational State Transfer (REST)
An architectural style for designing scalable web APIs using standard HTTP methods.

Role-Based Access Control (RBAC)
A security model that restricts system access based on assigned user roles and permissions.

Service Level Agreement (SLA)
A formal agreement defining expected service performance, responsibilities, and penalties between service providers and stakeholders.

Service Level Objective (SLO)
A specific, measurable performance target defined within a Service Level Agreement, such as uptime or response time.

Software Requirements Specification (SRS)
A formal document that defines system functionality, constraints, and requirements.

Test Case (TC)
A documented set of inputs, actions, and expected outcomes used to verify a specific system behavior or requirement.

User
Any individual who interacts with the application, including caregivers, care recipients, administrators, or support personnel.

User Acceptance Testing (UAT)
Testing performed by stakeholders or end users to validate that the system meets business and operational requirements.

User Interface (UI)
The visual and interactive elements through which users interact with an application.

User Experience (UX)
The overall usability, accessibility, and satisfaction a user experiences when interacting with an application.

Web Application Firewall (WAF)
A security control that monitors, filters, and blocks malicious web traffic to protect applications.

Work Breakdown Structure (WBS)
A hierarchical decomposition of project work into manageable components.

Wireframes
Low-fidelity visual representations of application layouts used during the design phase.

Appendix B - References

AlphaSoft. (2023, August 5). Short-Term Memory System (STeMS) project plan (Version 4.0) [Project documentation]. University of Maryland Global Campus.

Amazon Web Services. (2025, April). AWS pricing catalog. https://aws.amazon.com/pricing/

Apple Inc. (2025). Apple Developer Program FAQ. https://developer.apple.com/support/programs/

CareConnect High Level Requirements (n.d.). UMGC. Care Connect High Level Requirements.docx

OpenAI. (2025). ChatGPT (May 30 version) [Large language model]. https://chat.openai.com/

PubNub. (n.d.). Configuration: Authentication key (Dart SDK API Reference). https://www.pubnub.com/docs/sdks/dart/api-reference/configuration#authentication-key

Setting up Apple Pay | Apple Developer Documentation. (n.d.). Apple Developer Documentation. https://developer.apple.com/documentation/passkit/setting-up-apple-pay

Stripe, Inc. (2025). Stripe pricing guide. https://stripe.com/pricing

U.S. Bureau of Labor Statistics. (2023). Computer and information systems managers. Occupational Outlook Handbook. https://www.bls.gov/ooh/management/computer-and-information-systems-managers.htm

U.S. Bureau of Labor Statistics. (2023). Computer programmers. Occupational Outlook Handbook. https://www.bls.gov/ooh/computer-and-information-technology/computer-programmers.htm

U.S. Bureau of Labor Statistics. (2023). Project management specialists. Occupational Outlook Handbook. https://www.bls.gov/ooh/business-and-financial/project-management-specialists.htm

U.S. Bureau of Labor Statistics. (2023). Software developers. Occupational Outlook Handbook. https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm

U.S. Bureau of Labor Statistics. (2023). Web developers. Occupational Outlook Handbook. https://www.bls.gov/ooh/computer-and-information-technology/web-developers.htm

Zoom Video Communications. (n.d.). OAuth for user authorized apps. Zoom Developer Platform. https://developers.zoom.us/docs/integrations/oauth/

Appendix C - Change Request Form

Purpose:

This form is used to formally request, evaluate, approve, and document changes to scope, tooling, CI/CD enforcement, security posture, payment workflows, or notification infrastructure.

Change Request Form

Change Request ID: CR-_____ **Date Submitted**: _____

Submitted By: _____ **Role**: _____

Affected Area(s): ☐ Scope ☐ CI/CD ☐ Static Analysis ☐ Security ☐ Payments

☐ Notifications ☐ Other: _____

Description of Change:

_____

_____

_____

Reason for Change: (Business, security, compliance, or technical rationale)

_____

Severity / Priority: ☐ Critical ☐ High ☐ Medium ☐ Low

Impact Analysis: (Schedule, risk, security, deliverables)

_____

Affected Artifacts:

(Documents, pipelines, code modules, configurations)

_____

Proposed Implementation Approach:

_____

_____

Testing / Validation Required (Tests, scans, demos, verification steps): _____

_____

Rollback Plan: _____

_____

Approval Required: _____ ☐ Professor ☐ Product Owner ☐ Team Lead ☐ Technical Lead

Approval Decision: ☐ Approved ☐ Approved with Conditions ☐ Rejected

Approval Notes: _____

_____

Date Approved / Rejected: _____

Implementation Status: ☐ Not Started ☐ In Progress ☐ Completed ☐ Not Applicable

Closure Notes (Summary of outcome & verification): _____

_____

_____

Form CC-CR-001