Software Test Plan

IEEE Standard 829-1998 Format

Project Name: CareConnect - Electronic Visit Verification (EVV) System and Communication Test Plan

Document Version: 1.0

Date: February 14, 2026

Prepared By: Parthav Dani, QA Lead

Approved By: Project Manager - Team A

Organization: University of Maryland Global Campus - Team A

Document Control

Version	Date	Author	Description of Changes
0.1	02/07/2026	Parthav Dani	Initial Draft
0.2	02/10/2026	Parthav Dani	Added test environment details
1.0	02/14/2026	Parthav Dani	Final version for approval

Table of Contents

12. Remaining Test Tasks
13. Environmental Needs
14. Staffing and Training Needs
15. Responsibilities
16. Schedule
17. Planning Risks and Contingencies
18. Approvals
19. Glossary

---

1. Test Plan Identifier

Test Plan ID: TP-CARECONNECT- INTEGRATED-2026-001

Version: 1.0

Status: Approved

---

2. References

List all documents referenced in this test plan:

- **Requirements Specification:** SRS-CARECONNECT-001, Version 2.1, 01/15/2026
- **System Design Specification:** SDS-CARECONNECT-001, Version 1.5, 01/22/2026
- **Project Plan:** PP-CARECONNECT-001, Version 1.3, 01/10/2026
- **Quality Assurance Plan:** QAP-CARECONNECT-001, Version 1.0, 01/18/2026
- **Configuration Management Plan:** CMP-CARECONNECT-001, Version 1.0, 01/12/2026
- **IEEE Std 829-1998:** Standard for Software Test Documentation
- **HIPAA Compliance Guidelines:** Health Insurance Portability and Accountability Act
- **HHAeXchange API Documentation:** Version 5, Flat File
- **Virginia DMAS EVV Requirements:** Version 2.0, Department of Medical Assistance Services
- **AWS ECS Fargate** Documentation
- **PostgreSQL** Documentation
- **Amazon Chime SDK** Documentation
- **AWS WebSocket & ALB** Documentation

- **Sentiment Analysis** Architecture (TDD Section 5.4)
- **WebRTC Security** Documentation

---

3. Introduction

3.1 Purpose

This Test Plan describes the testing approach and overall framework for testing the CareConnect Electronic Visit Verification (EVV) system. The plan identifies the items to be tested, features to be tested, testing tasks to be performed, personnel responsible for each task, and any risks requiring contingency planning.

CareConnect is a comprehensive EVV solution designed to track and verify home healthcare visits for Medicaid reimbursement. The system integrates with HHAeXchange platform to submit rendered service data for Virginia Department of Medical Assistance Services (DMAS) compliance.

CareConnect also includes a real-time communication (chat and audio/video), Amazon Chime SDK-based meeting sessions, and live sentiment analysis for caregiver-patient interactions.

3.2 Scope

In Scope:

- Flutter-based mobile application (iOS and Android)
- Flutter-based web application
- Java Spring Boot backend REST API
- PostgreSQL database operations
- XML generation and validation for HHAeXchange submission
- OAuth 2.0 authentication and authorization
- GPS-based check-in/check-out functionality
- File upload and processing (Excel/CSV)
- Real-time data synchronization
- HHAeXchange API integration
- Dashboard and reporting features
- User management and role-based access control
- Real-time chat messaging (WebSocket + REST)
- Audio/Video calling via Amazon Chime SDK
- Call lifecycle management (initiate, accept, decline, terminate)

- Sentiment analysis with Trend Pulse Graph visualization
- Post-call sentiment summary generation and retrieval
- Call metadata persistence

Testing Levels:

- Unit Testing
- Integration Testing
- System Testing
- User Acceptance Testing (UAT)
- Security Testing
- Performance Testing

Out of Scope:

- Third-party HHAeXchange platform internal testing
- AWS infrastructure configuration (managed by DevOps)
- Third-party Flutter plugin internal testing
- Future Phase 2 features (Advanced analytics, AI-powered scheduling)
- Legacy system migration from previous EVV solutions
- Amazon Chime SDK internal media processing
- AI model internal training or tuning

3.3 Objectives

The primary objectives of the testing effort are:

- Verify that all specified requirements from SRS-CARECONNECT-001 are implemented correctly
- Validate that the system meets Virginia DMAS EVV compliance requirements
- Ensure successful integration with HHAeXchange API for data submission
- Verify GPS accuracy and reliability for visit verification
- Ensure HIPAA compliance for Protected Health Information (PHI)
- Validate cross-platform functionality (Web, iOS, Android)
- Identify and document defects before production deployment
- Assess system performance under expected load conditions
- Verify data integrity throughout the XML generation and submission workflow

- Ensure user interface meets usability standards for field caregivers

---

4. Test Items

Item ID	Component/Module Name	Version	Description	Documentation Reference
TI-001	Mobile Application (Flutter)	1.0.0	iOS and Android mobile app for caregivers	SRS-3.1, SRS-3.2
TI-002	Web Application (Flutter)	1.0.0	Web-based admin and scheduler interface	SRS-3.3
TI-003	Backend REST API	1.0.0	Java Spring Boot microservices	SRS-4.1
TI-004	Authentication Service	1.0.0	OAuth 2.0 implementation	SRS-4.2
TI-005	Database Layer	1.0.0	PostgreSQL database and ORM	SRS-4.3
TI-006	XML Processing Service	1.0.0	XML generation and validation	SRS-5.1
TI-007	HHAeXchange Integration	1.0.0	API integration layer	SRS-5.2
TI-008	GPS Service	1.0.0	Location tracking and verification	SRS-6.1
TI-009	File Upload Service	1.0.0	Excel/CSV processing	SRS-6.2
TI-010	Dashboard & Reports	1.0.0	Analytics and reporting module	SRS-7.1
TI-011	Chat Messaging Service	1.0.0	Real-time messaging between caregivers and patients	SRS-8.1
TI-012	Audio/Video Calling Module	1.0.0	Amazon Chime SDK integration for meeting sessions	SRS-8.2
TI-013	Sentiment Processing Service	1.0.0	Real-time emotional trend analysis engine	SRS-8.3
TI-014	Call Signaling & WebSocket Layer	1.0.0	WebSocket signaling and call state management	SRS-8.4
TI-011	In-Home Residential Support Module	1.0.0	ADL/IADL logging, behavioral incident reports, client profile, DD Waiver compliance	SRS-8.5

4.1 Programs

Frontend Applications:

- CareConnect Mobile App (iOS) - Version 1.0.0
- CareConnect Mobile App (Android) - Version 1.0.0
- CareConnect Web Portal - Version 1.0.0

Backend Services:

- Authentication Service (Spring Boot) - Version 1.0.0
- Visit Management Service - Version 1.0.0
- XML Processing Service - Version 1.0.0
- HHAeXchange Integration Service - Version 1.0.0
- Reporting Service - Version 1.0.0

4.2 Databases

- CareConnect Production Database (PostgreSQL 14.x)
- CareConnect Test Database (PostgreSQL 14.x)
- User credential store (Flutter Secure Storage)

4.3 Documentation

- User Manual for Caregivers (Mobile App)
- Administrator Guide (Web Portal)
- API Documentation (Swagger/OpenAPI)
- System Installation Guide
- Troubleshooting Guide

---

5. Software Risk Issues

Risk ID	Risk Description	Probability	Impact	Mitigation Strategy
R-001	HHAeXchange API changes or downtime during testing	Medium	High	Maintain mock API server; coordinate testing schedule with HHAeXchange; implement circuit breaker pattern
R-002	GPS accuracy issues in certain geographic areas	Medium	High	Test in multiple locations; implement fallback mechanisms; add manual override capability

Risk ID	Risk Description	Probability	Impact	Mitigation Strategy
R-003	Cross-platform Flutter compatibility issues	Low	Medium	Test early on all target platforms; maintain platform-specific test suites
R-004	HIPAA compliance violations in data handling	Low	Critical	Conduct security audit; implement encryption at rest and in transit; train team on PHI handling
R-005	PostgreSQL performance degradation under load	Medium	High	Conduct performance testing early; optimize queries; implement caching strategy
R-006	OAuth token expiration during long sessions	Low	Medium	Implement token refresh logic; conduct session management testing
R-007	XML schema validation failures	Medium	High	Validate against HHAeXchange XSD early; implement comprehensive unit tests
R-008	Mobile device compatibility issues	Medium	Medium	Test on wide range of devices; use device farm for testing
R-009	Data synchronization conflicts	Medium	High	Implement conflict resolution strategy; test offline scenarios
R-010	Schedule delays due to resource availability	High	Medium	Cross-train team members; identify backup resources; prioritize critical testing
R-011	Amazon Chime SDK service outage	Medium	High	Implement graceful error handling; retry logic
R-012	Sentiment latency exceeding SLA	Medium	High	Horizontal scaling; monitoring alerts
R-013	Unauthorized meeting join attempts	Low	Critical	Strict RBAC; backend credential issuance
R-014	WebSocket saturation under load	Medium	High	Load testing; horizontal scaling

---

6. Features to be Tested

Feature ID	Feature Name	Description	Priority	Requirements Reference
F-001	User Authentication	OAuth 2.0 login with JWT tokens	High	REQ-AUTH-001
F-002	GPS Check-In	Location capture at visit start	High	REQ-GPS-001
F-003	GPS Check-Out	Location capture at visit end	High	REQ-GPS-002
F-004	Visit Recording	Manual visit data entry	High	REQ-VISIT-001
F-005	Patient Search	Search and select patients	High	REQ-PATIENT-001
F-006	Service Type Selection	Select from predefined service codes	High	REQ-SERVICE-001
F-007	Visit History	View past visits and status	Medium	REQ-HISTORY-001
F-008	XML Generation	Generate HHAeXchange-compliant XML	High	REQ-XML-001
F-009	XML Validation	Validate against HHAeXchange schema	High	REQ-XML-002
F-010	HHAeXchange Submission	Submit XML via HTTPS POST	High	REQ-API-001
F-011	Response Processing	Parse and handle API responses	High	REQ-API-002
F-012	File Upload	Upload Excel/CSV visit data	Medium	REQ-UPLOAD-001
F-013	File Processing	Parse and validate uploaded files	Medium	REQ-UPLOAD-002
F-014	Dashboard Metrics	Display visit statistics	Medium	REQ-DASH-001
F-015	Visit Reports	Generate compliance reports	Medium	REQ-REPORT-001
F-016	User Management	Admin user CRUD operations	Medium	REQ-ADMIN-001
F-017	Role-Based Access	Permission-based feature access	High	REQ-AUTH-002
F-018	Data Encryption	Encrypt PHI at rest and in transit	High	REQ-SEC-001
F-019	Offline Mode	Record visits without connectivity	Medium	REQ-OFFLINE-001
F-020	Data Synchronization	Sync offline data when connected	Medium	REQ-SYNC-001

Feature ID	Feature Name	Description	Priority	Requirements Reference
F-021	Chat Messaging	Real-time caregiver-patient messaging	High	REQ-COMM-001
F-022	Message Delivery Status	Sent/Delivered indicators	Medium	REQ-COMM-002
F-023	Call Initiation	User initiates call	High	REQ-CALL-001
F-024	Call Acceptance/Decline	Accept or reject call	High	REQ-CALL-002
F-025	Meeting Creation	Backend creates Chime meeting session	High	REQ-CALL-003
F-026	Attendee Credential Issuance	Authorized users receive join credentials	High	REQ-CALL-004
F-027	Media Encryption	SRTP media encryption validation (Amazon Chime SDK)	High	REQ-SEC-003
F-028	Live Sentiment Updates	Trend Pulse Graph real-time updates	High	REQ-SENT-001
F-029	Sentiment SLA	Latency < 2s P95	High	REQ-SENT-002
F-030	Sentiment Summary Persistence	End-of-call structured summary	High	REQ-SENT-003
F-031	ADL Documentation	Record Activities of Daily Living during an active EVV visit	High	REQ-IHRS-001
F-032	IADL Documentation	Record Instrumental Activities of Daily Living during an active EVV visit	High	REQ-IHRS-003
F-033	Care Notes Entry	Free-text caregiver notes attached to a visit record	High	REQ-IHRS-003
F-034	Behavioral Incident Reporting	Create, save, and retrieve behavioral incident reports linked to a visit	High	REQ-IHRS- 004
F-035	Client Profile View	Caregiver read access to client profile information within the IHRS module	Medium	REQ-IHRS-006

Feature ID	Feature Name	Description	Priority	Requirements Reference
F-037	Client Profile Management	Admin/Supervisor ability to create and update client profiles	Medium	REQ-IHRS-007
F-038	IHRS Data Persistence	All ADL/IADL, notes, and incident data retained after visit completion	High	REQ-IHRS-008
F-039	IHRS Audit Loggin	Log all create/edit actions on IHRS records with user ID, timestamp, and change type	High	REQ-IHRS-010

---

7. Features Not to be Tested

Feature/Area	Reason for Exclusion
HHAeXchange Internal Logic	Third-party system; tested by HHAeXchange; only integration points tested
AWS Infrastructure Configuration	Managed by DevOps team; infrastructure testing separate from application testing
Flutter Framework Core	Open-source framework maintained by Google; assume framework stability
PostgreSQL Database Engine	Mature RDBMS; only test queries and data integrity, not database engine itself
Third-Party Plugins (File Picker, Charts)	Community-maintained plugins; functional testing only, not internal plugin code
iOS/Android OS Functionality	Operating system features; test app behavior only
Network Infrastructure	Network team responsibility; assume network availability
Future Phase 2 Features	AI scheduling, predictive analytics - deferred to next release
Legacy Data Migration	One-time activity; separate migration testing plan
Email/SMS Notifications	Phase 2 feature; not in current release
Amazon Chime internal WebRTC stack	AWS-managed service

Feature/Area	Reason for Exclusion
AI model training internals	External model management
WebSocket library internals	Framework-level component

---

8. Approach

8.1 Testing Strategy

Testing Levels:

1. Unit Testing

- **Scope:** Individual functions, classes, and methods in backend services and Flutter widgets
- **Responsibility:** Development Team (Parthav, Jordene, Mawuko)
- **Tools:**
    - JUnit 5 for Java Spring Boot
    - Mockito for mocking dependencies
    - Flutter Test for Dart/Flutter
- **Entry Criteria:**
    - Code completed and peer-reviewed
    - Unit test cases written
- **Exit Criteria:**
    - Minimum 80% code coverage achieved
    - All unit tests pass
    - Zero critical defects

2. Integration Testing

- **Scope:** API endpoints, database interactions, third-party integrations
- **Responsibility:** QA Team (Parthav, Pemon)
- **Tools:**
    - Postman for API testing
    - Spring Boot Test for integration tests
    - PostgreSQL test database

- **Entry Criteria:**
    - Unit testing complete
    - Test environment ready
    - Integration test cases approved
- **Exit Criteria:**
    - All integration test cases executed
    - Critical integration points verified
    - API contracts validated

3. System Testing

- **Scope:** End-to-end workflows, cross-platform testing, non-functional requirements
- **Responsibility:** QA Team (All members)
- **Tools:**
    - Manual testing
    - Appium for mobile automation
    - Selenium for web automation
    - JMeter for performance testing
- **Entry Criteria:**
    - Integration testing complete
    - System deployed to test environment
    - Test data prepared
- **Exit Criteria:**
    - All high-priority test cases pass
    - Performance benchmarks met
    - Security testing complete

4. User Acceptance Testing (UAT)

- **Scope:** Business workflows, usability, DMAS compliance validation
- **Responsibility:** Business stakeholders, pilot caregivers
- **Tools:**
    - Production-like environment

    - Real devices (iOS/Android)
    - UAT checklist
- **Entry Criteria:**
    - System testing complete
    - UAT environment ready
    - User training completed
- **Exit Criteria:**
    - UAT sign-off received
    - No critical defects outstanding
    - User satisfaction criteria met

8.2 Testing Types

Functional Testing:

- Verify all functional requirements from SRS
- Test business logic and workflows
- Validate data input/output
- Test error handling and validation
- Verify GPS capture accuracy
- Call lifecycle validation
- Unauthorized join testing
- Sentiment degradation mode testing

Performance Testing:

- **Load Testing:** Simulate 100 concurrent users
- **Stress Testing:** Test system limits (500+ concurrent users)
- **Scalability Testing:** Verify horizontal scaling capability
- **Response Time:** API responses < 2 seconds, Page loads < 3 seconds
- **Database Performance:** Query optimization, connection pooling
- Concurrent calls (50+ active sessions)
- Sentiment SLA validation (<2s P95)
- Combined chat + call load testing

Security Testing:

- **Authentication/Authorization:** OAuth 2.0 token validation, JWT expiration, role-based access
- **Vulnerability Scanning:** OWASP Top 10 vulnerabilities
- **Penetration Testing:** Simulated attacks on API endpoints
- **Data Encryption:** Verify AES-256 encryption at rest, TLS 1.2+ in transit
- **HIPAA Compliance:** PHI protection, audit logging, access controls
- Unauthorized call join attempts
- Token replay attempts
- Meeting credential reuse validation

Usability Testing:

- Field caregiver workflow efficiency
- Mobile app navigation intuitiveness
- Web dashboard clarity
- Error message helpfulness
- Accessibility compliance (WCAG 2.1 Level AA)

Compatibility Testing:

- **Browser Compatibility:** Chrome, Safari, Firefox, Edge (latest 2 versions)
- **Mobile OS:** iOS 14+, Android 10+
- **Device Testing:** iPhone 12/13/14, Samsung Galaxy S20/S21/S22, Google Pixel
- **Screen Resolutions:** 375x667 to 428x926 (mobile), 1920x1080 (desktop)

Regression Testing:

- **Automated Regression Suite:** Critical path test cases automated
- **Manual Regression:** High-priority scenarios after each build
- **Smoke Testing:** Basic functionality verification after deployment

8.3 Test Data Management

Test Data Strategy:

- **Synthetic Data:** Generate realistic patient and caregiver data using Faker library
- **Anonymized Data:** Use de-identified production data subset for realistic testing
- **Boundary Data:** Min/max values, edge cases for validation testing

- **Invalid Data:** Test error handling with malformed inputs
- **GPS Coordinates:** Sample locations across Virginia service areas
- **IHRS Data:** Synthetic ADL/IADL checklists, behavioral incident reports, client profiles, and care notes covering valid, boundary, and invalid scenarios linked to active and completed visit records

Data Refresh:

- Test database reset weekly
- On-demand refresh for integration testing
- Automated seed scripts for consistent data state

8.4 Test Automation

Automation Strategy:

- Automate repetitive regression test cases (60% automation coverage target)
- Focus manual testing on exploratory and usability testing
- Continuous integration with automated test execution

Tools:

- **Backend API:** Postman/Newman for API test automation
- **Web UI:** Selenium WebDriver with TestNG
- **Mobile:** Appium for cross-platform mobile automation
- **CI/CD:** GitHub Actions for automated test execution
- **Reporting:** Allure Framework for test reporting

Test Cases to be Automated:

- Login/authentication flows
- API endpoint validation
- Critical business workflows (visit creation, XML generation)
- Data validation scenarios
- Regression test suite

Automation Framework:

- Page Object Model (POM) for UI tests
- Behavior-Driven Development (BDD) with Cucumber for readability
- Modular design for maintainability

---

9. Item Pass/Fail Criteria

9.1 Pass Criteria

General Pass Criteria:

- All high-priority test cases executed and passed (100%)
- Medium-priority test cases: ≥95% pass rate
- Low-priority test cases: ≥90% pass rate
- No critical or high-severity defects remain open
- All medium-severity defects have approved workarounds or fix dates
- Code coverage ≥80% for backend services
- Performance benchmarks met:
    - API response time < 2 seconds (95th percentile)
    - Page load time < 3 seconds
    - GPS accuracy within 50 meters
- Security requirements validated (no critical vulnerabilities)
- HIPAA compliance verified
- HHAeXchange integration successful with ≥98% submission success rate
- UAT sign-off obtained from business stakeholders
- Call setup time < 3 seconds (95th percentile)
- Sentiment update latency < 2 seconds (P95)
- No unauthorized call joins permitted
- No PHI exposed via chat or sentiment summary

9.2 Fail Criteria

General Fail Criteria:

- Any critical defect discovered and not resolved
- More than 10% of high-priority test cases fail
- Performance below acceptable thresholds
- Critical security vulnerabilities identified
- Data integrity issues detected (data loss, corruption)

- GPS accuracy consistently beyond 100 meters
- HHAeXchange API submission success rate < 95%
- HIPAA compliance violations
- System crashes or data loss scenarios
- Unable to complete critical business workflows

9.3 Defect Severity Definitions

Severity	Definition	Example	Response Time	Resolution Time
Critical	System crash, data loss, security breach, HIPAA violation	Application crashes on launch; PHI exposed; GPS data lost	1 hour	24 hours
High	Major functionality broken, no workaround available	Unable to submit visits to HHAeXchange; login fails for all users	4 hours	3 days
Medium	Functionality impaired, workaround exists	Incorrect visit duration calculation; manual correction possible	1 day	1 week
Low	Minor issue, cosmetic problem, minimal impact	Misaligned text on dashboard; typo in error message	3 days	2 weeks

---

10. Suspension Criteria and Resumption Requirements

10.1 Suspension Criteria

Testing will be suspended if any of the following conditions occur:

- **Critical defect rate exceeds 3 defects per day** for two consecutive days
- **Test environment becomes unavailable** for more than 4 hours during testing window
- **Build stability is below 70%** (more than 30% of smoke tests fail)
- **More than 40% of test cases are blocked** due to defects or dependencies
- **Critical resources (QA lead, key developers) become unavailable** without backup
- **HHAeXchange API unavailable** for more than 24 hours

- **Database corruption** or data integrity issues discovered
- **Security breach or HIPAA violation** discovered requiring immediate remediation
- **Major requirement changes** requiring test plan revision

10.2 Resumption Requirements

Testing can resume when the following conditions are met:

- **Critical defects are resolved** and fix verified through regression testing
- **Test environment is restored** and validated through smoke tests
- **Build stability reaches ≥85%** pass rate for smoke tests
- **Blocking issues are resolved** and workarounds documented
- **Required resources are available** or adequate substitutes identified
- **HHAeXchange API connectivity restored** and integration verified
- **Database restored** from backup or data integrity confirmed
- **Security issues remediated** and validated through security scan
- **Updated requirements reviewed** and test plan updated with approvals
- **Go-ahead approval** from Test Manager and Project Manager

---

11. Test Deliverables

11.1 Before Testing

- **Test Plan** (this document) - TP-CARECONNECT-EVV-2026-001
- **Test Cases** - TC-CARECONNECT-001 through TC-CARECONNECT-250
- **Test Scripts** - Automated test scripts repository
- **Test Data Sets** - Synthetic patient/caregiver data, GPS coordinates
- **Test Environment Setup Documentation** - ENV-SETUP-001

11.2 During Testing

- **Test Logs** - Daily execution logs with timestamps
- **Defect Reports** - Jira defect tracking system
- **Test Incident Reports** - Critical issues escalation documents
- **Test Status Reports** - Weekly status updates to stakeholders
- **Test Execution Reports** - Daily test run summaries

- **Code Coverage Reports** - SonarQube analysis reports

11.3 After Testing

- **Test Summary Report** - TSR-CARECONNECT-001
- **Defect Analysis Report** - DAR-CARECONNECT-001
- **Test Metrics Report** - TMR-CARECONNECT-001 (pass/fail rates, coverage, defect density)
- **Lessons Learned Document** - LL-CARECONNECT-001
- **Test Completion Certificate** - TCC-CARECONNECT-001
- **Traceability Matrix** - Requirements to test cases mapping
- **Performance Test Results** - Load test reports with graphs
- **Security Test Results** - Vulnerability scan reports
- **UAT Sign-off Document** - Business stakeholder approval

---

12. Remaining Test Tasks

Task ID	Task Description	Responsible	Target Date	Status
TT-001	Complete integration test case design (50 remaining)	Pemon	02/18/2026	In Progress
TT-002	Set up Appium mobile test environment	Jordene	02/16/2026	Not Started
TT-003	Prepare GPS test data for 20 Virginia locations	Mawuko	02/17/2026	In Progress
TT-004	Install JMeter and configure performance scenarios	Melissa	02/19/2026	Not Started
TT-005	Configure HHAeXchange test API credentials	Parthav	02/15/2026	Complete
TT-006	Create automated regression test suite	Jordene	02/25/2026	Not Started
TT-007	Prepare UAT environment with production-like data	Mawuko	02/28/2026	Not Started
TT-008	Schedule security vulnerability scan	Parthav	02/20/2026	Scheduled
TT-009	Obtain test devices (iPhone 14, Samsung S22)	Melissa	02/16/2026	In Progress

Task ID	Task Description	Responsible	Target Date	Status
TT-010	Conduct test team training on HHAeXchange requirements	Parthav	02/17/2026	Scheduled

---

13. Environmental Needs

13.1 Hardware Requirements

Component	Specification	Quantity	Purpose
Application Server	AWS ECS Fargate (Test & UAT)	2	Backend API (Test & UAT)
Load Balancer	Application Load Balancer (WSS enabled)	1	Traffic routing
Media Service	Amazon Chime SDK	Managed	Audio/Video transport
Database Server	AWS RDS PostgreSQL (db.t3.medium)	2	Database (Test & UAT)
Test Workstation	Windows 11, 16GB RAM, i7 processor	3	Test execution
Mac Mini	macOS Monterey, 16GB RAM, M1	1	iOS testing
iPhone 14	iOS 16+	2	Mobile app testing
iPhone 12	iOS 15+	1	Compatibility testing
Samsung Galaxy S22	Android 12+	2	Mobile app testing
Google Pixel 6	Android 12+	1	Compatibility testing
Samsung Galaxy Tab	Android 11+	1	Tablet testing

13.2 Software Requirements

Software	Version	Purpose
Operating System (Server)	Ubuntu 22.04 LTS	Application hosting
Java Runtime	OpenJDK 17	Spring Boot execution

Software	Version	Purpose
PostgreSQL	14.7	Database management
Node.js	18.x	Build tools
Flutter SDK	3.16+	Mobile/web app development
Android Studio	2023.1+	Android testing
Xcode	15.0+	iOS testing
Chrome Browser	Latest	Web testing
Safari Browser	Latest	Web testing
Firefox Browser	Latest	Web testing
Postman	Latest	API testing
JMeter	5.5+	Performance testing
Appium	2.0+	Mobile automation
Selenium	4.x	Web automation
Git	Latest	Version control
Docker	24.x	Container management
SonarQube	9.x	Code quality analysis

13.3 Network Requirements

- **Network Bandwidth:** Minimum 100 Mbps for test environment
- **Network Configuration:** Isolated VLAN for test environment
- **VPN Access:** Required for remote testers to access test environment
- **Firewall Rules:**
    - Allow HTTPS (443) to HHAeXchange API endpoints
    - Allow PostgreSQL (5432) from application servers only
    - Allow SSH (22) for administrative access
- **DNS Configuration:** test.careconnect.local domain

13.4 Test Data

Patient Data:

- 500 synthetic patient records with Medicaid IDs
- Mix of active and inactive patients
- Various service authorization types

Caregiver Data:

- 100 caregiver accounts with different roles
- Valid NPI numbers for providers
- Mix of full-time and part-time schedules

Visit Data:

- 2,000 historical visit records
- Various service types (Personal Care, Companion, etc.)
- Mix of completed, pending, and rejected statuses

GPS Coordinates:

- 20 valid patient addresses in Virginia
- Corresponding GPS coordinates (latitude/longitude)
- Boundary test locations (edge of service area)

Invalid/Error Data:

- Missing required fields
- Invalid date/time combinations
- Out-of-range GPS coordinates
- Malformed Medicaid IDs
- Expired service authorizations

---

14. Staffing and Training Needs

14.1 Staffing

Role	Name	Allocation (%)	Duration	Key Responsibilities
Test Manager	Parthav Dani	100%	Weeks 5-12	Overall test coordination, risk management, stakeholder communication
QA Lead	Parthav Dani	100%	Weeks 5-12	Test case review, execution oversight, defect triage
Test Engineer	Jordene	70%	Weeks 6-11	Mobile app testing, automation development
Test Engineer	Mawuko	70%	Weeks 6-11	Web app testing, integration testing
Test Engineer	Pemon	50%	Weeks 6-11	API testing, performance testing
UAT Coordinator	Melissa	40%	Weeks 9-11	UAT planning, user training, acceptance testing
Developer (Support)	Parthav	30%	Weeks 5-12	Defect fixes, test environment setup
Developer (Support)	Jordene	30%	Weeks 5-12	Mobile defect fixes
Developer (Support)	Mawuko	30%	Weeks 5-12	Backend defect fixes

Total Team Size: 5 members with overlapping responsibilities

14.2 Training Needs

Training Topic	Target Audience	Duration	Delivery Method	Target Date
CareConnect System Overview	All testers	4 hours	Instructor-led	02/15/2026
HHAeXchange EVV Requirements	All testers	3 hours	Online webinar	02/16/2026
HIPAA Compliance & PHI Handling	All testers	2 hours	Online course	02/14/2026

Training Topic	Target Audience	Duration	Delivery Method	Target Date
Appium Mobile Test Automation	Jordene, Mawuko	8 hours	Self-study + lab	02/20/2026
JMeter Performance Testing	Pemon	4 hours	Online tutorial	02/19/2026
GPS Testing Best Practices	All testers	2 hours	Instructor-led	02/17/2026
Postman API Testing	Pemon, Parthav	3 hours	Self-study	02/18/2026
Defect Management in Jira	All testers	1 hour	Documentation	02/15/2026
Amazon Chime SDK Architecture	QA Team	2 hours	Amazon Chime SDK Architecture	

---

15. Responsibilities

15.1 Test Manager (Parthav Dani)

- Overall test planning and strategy
- Resource allocation and scheduling
- Risk identification and mitigation
- Stakeholder communication and reporting
- Budget management
- Final test sign-off and go/no-go decision
- Escalation point for critical issues

15.2 QA Lead (Parthav Dani)

- Test case design and review
- Test execution oversight and monitoring
- Defect triage and prioritization
- Daily status reporting
- Test environment coordination
- Metrics collection and analysis
- Quality gate enforcement

15.3 Test Engineers (Jordene, Mawuko, Pemon)

- Test case execution (manual and automated)
- Defect identification and detailed reporting
- Test data preparation and management
- Test documentation updates
- Automation script development
- Cross-browser/device testing
- Participate in defect triage meetings

15.4 UAT Coordinator (Melissa)

- UAT planning and scheduling
- User training and onboarding
- UAT test case preparation
- Coordinate with business stakeholders
- Collect and document UAT feedback
- UAT sign-off facilitation

15.5 Development Team (Parthav, Jordene, Mawuko)

- Unit test creation and execution
- Defect analysis and fixes
- Build delivery to test environment
- Technical support for testing team
- Code review and quality improvements
- Integration support

15.6 Business Stakeholders

- Requirements clarification
- Acceptance criteria validation
- UAT participation
- Final acceptance sign-off
- Provide domain expertise

15.7 DevOps (External Support)

- Test environment provisioning

- CI/CD pipeline configuration
- Infrastructure monitoring
- Database backup and restore
- Security patch management

---

16. Schedule

16.1 Test Phases

Phase	Start Date	End Date	Duration	Dependencies
Test Planning	02/07/2026	02/14/2026	1 week	Requirements complete
Test Design	02/14/2026	02/28/2026	2 weeks	Test plan approved
Test Environment Setup	02/17/2026	02/24/2026	1 week	Infrastructure ready
Unit Testing	02/21/2026	03/07/2026	2 weeks	Code development complete
Integration Testing	03/03/2026	03/17/2026	2 weeks	Unit testing 80% complete
System Testing	03/10/2026	03/31/2026	3 weeks	Integration testing complete
Performance Testing	03/17/2026	03/24/2026	1 week	System stable build
Security Testing	03/20/2026	03/27/2026	1 week	System testing 70% complete
UAT Preparation	03/24/2026	03/28/2026	1 week	System testing complete
User Acceptance Testing	03/31/2026	04/11/2026	2 weeks	UAT environment ready
Defect Resolution	Ongoing	04/11/2026	Throughout	Defects identified
Regression Testing	04/07/2026	04/14/2026	1 week	Fixes deployed
Test Closure	04/14/2026	04/18/2026	1 week	All criteria met

16.2 Milestones

Milestone	Target Date	Deliverable	Status
Test Plan Approval	02/14/2026	Approved Test Plan	Complete
Test Cases Complete	02/28/2026	250+ Test Cases in Repository	In Progress
Test Environment Ready	02/24/2026	Environment Sign-off Document	Scheduled
Unit Testing Complete	03/07/2026	Unit Test Report (80% coverage)	Scheduled
Integration Testing Complete	03/17/2026	Integration Test Report	Scheduled
System Testing Complete	03/31/2026	System Test Summary Report	Scheduled
Performance Benchmarks Met	03/24/2026	Performance Test Results	Scheduled
Security Testing Complete	03/27/2026	Security Assessment Report	Scheduled
UAT Sign-off	04/11/2026	UAT Acceptance Certificate	Scheduled
Regression Complete	04/14/2026	Regression Test Report	Scheduled
Go-Live Approval	04/18/2026	Production Release Authorization	Scheduled

Current Status: Week 5 of 12-week project timeline

---

17. Planning Risks and Contingencies

Risk	Probability	Impact	Contingency Plan
Test environment delays	Medium	High	Use Docker containers for local testing; escalate to AWS support; allocate buffer time
HHAeXchange API unavailability	Medium	Critical	Implement mock API server; coordinate scheduled maintenance windows; have rollback plan
Resource unavailability (illness, turnover)	Low	High	Cross-train Jordene and Mawuko on all modules; document critical procedures; identify contractor backup
Requirement changes mid-testing	High	Medium	Implement change control process; assess impact before accepting; re-prioritize test cases; extend timeline if critical

Risk	Probability	Impact	Contingency Plan
GPS testing limitations (indoor/weather)	Medium	Medium	Use GPS simulation tools; test in multiple outdoor locations; implement manual override testing
Mobile device procurement delays	Low	Medium	Use cloud-based device farms (BrowserStack, Sauce Labs); borrow devices from team members temporarily
Performance bottlenecks discovered late	Medium	High	Conduct performance testing early (Week 7); allocate developer time for optimization; implement caching strategy
HIPAA compliance issues found	Low	Critical	Engage security consultant immediately; halt testing until resolved; implement audit trail review
Schedule compression due to development delays	High	High	Prioritize critical path tests; increase automation; add weekend shifts; negotiate scope reduction
Defect backlog exceeds capacity	Medium	High	Daily defect triage; enforce severity-based SLAs; add developer resources; automate regression
HHAeXchange API schema changes	Low	High	Maintain communication with HHAeXchange; version control XSD schemas; implement schema validation in CI/CD
Data synchronization issues in offline mode	Medium	Medium	Dedicate testing time to offline scenarios; implement comprehensive sync testing; use conflict simulation tools

---

18. Approvals

By signing below, the approvers acknowledge that they have reviewed this Test Plan and approve its contents.

Role	Name	Signature	Date
Test Manager	Parthav Dani	_____	_____
Project Manager	Team A Lead	_____	_____
Development Manager	Parthav Dani	_____	_____
Quality Assurance Manager	Parthav Dani	_____	_____

Role	Name	Signature	Date
Business Owner	UMGC Instructor	_____	_____

---

19. Glossary

Term	Definition
Acceptance Testing	Formal testing conducted to determine whether the CareConnect system satisfies DMAS EVV requirements and business needs
Actual Result	The behavior produced/observed when a test case is executed on CareConnect
API (Application Programming Interface)	Interface used to communicate between CareConnect and HHAeXchange
Defect	A flaw in CareConnect code or configuration that can cause failure to meet requirements
DMAS	Virginia Department of Medical Assistance Services - regulatory authority for EVV
EVV (Electronic Visit Verification)	System to electronically verify home healthcare visit delivery
Expected Result	The predicted observable behavior of CareConnect based on requirements
GPS	Global Positioning System used to capture caregiver location at check-in/check-out
HIPAA	Health Insurance Portability and Accountability Act - healthcare data privacy regulation
Integration Testing	Testing performed to expose defects in interfaces between CareConnect components
JWT (JSON Web Token)	Compact token format used for authentication in CareConnect
HHAeXchange	Third-party EVV aggregator platform that receives CareConnect submissions
NPI (National Provider Identifier)	Unique 10-digit identifier for healthcare providers

Term	Definition
OAuth 2.0	Industry-standard protocol for authorization used in CareConnect
PHI (Protected Health Information)	Patient health data protected under HIPAA
Regression Testing	Testing to confirm that recent code changes have not adversely affected existing CareConnect features
REST API	Representational State Transfer API architecture used in CareConnect backend
Test Case	A set of input values, execution preconditions, expected results, and execution postconditions for testing CareConnect
Test Environment	AWS-hosted environment containing hardware, software, network, and data needed for CareConnect testing
Unit Testing	Testing of individual CareConnect software components or modules
Validation	Confirmation that CareConnect meets the needs of caregivers and DMAS requirements
Verification	Confirmation that CareConnect has been built according to its specifications
XML (eXtensible Markup Language)	Structured data format used for submitting visit data to HHAeXchange

Appendix A – EVV Requirements: Test Case Details

This appendix documents all automated test cases written for the Electronic Visit Verification (EVV) requirements implemented in the CareConnect system. Test cases are organised by layer: Backend (Java Spring Boot / JUnit 5 + Mockito) and Frontend (Flutter widget tests). All tests are executed as part of the CI/CD pipeline and must pass before a build is promoted.

A.1 Backend Unit Tests (Java Spring Boot)

Framework: JUnit 5 · Mockito · AssertJ | Source root: backend/core/src/test/java/com/careconnect/ | Coverage target: ≥ 80 %

TC ID	Test Class	Test Method	Description	Expected Result	Test Type	Status
TC-EVV-BE-001	EvvServiceTest	createRecord_patientNotFound_throwsIllegalArgument	Create EVV record for a patient ID that does not exist in the repository.	IllegalArgumentException thrown with 'Patient not found' message.	Unit	Pass
TC-EVV-BE-002	EvvServiceTest	createRecord_noLocation_createsRecordWithoutLocation	Create an EVV record where no location fields are provided.	Record persisted; EvvLocationService.saveLocation() is never called.	Unit	Pass
TC-EVV-BE-003	EvvServiceTest	createRecord_withGpsCheckinAndCoords_savesLocation	Create EVV record with GPS check-in source and valid lat/lng coordinates.	Record persisted; saveLocation() called once with GPS coordinates.	Unit	Pass
TC-EVV-BE-004	EvvServiceTest	createRecord_withGpsCheckinNoCoords_skipsSave	Create EVV record with GPS check-in source but no lat/lng values.	Record persisted; saveLocation() is never called (no coordinates).	Unit	Pass
TC-EVV-BE-	EvvServiceTest	createRecord_withPatientAddressCheckin_savesLocation	Create EVV record with PATIENT_ADDRESS check-in source.	Record persisted; saveLocation() called once with PATIENT_ADDRESS type.	Unit	Pass

005						
TC-EVV-BE-006	EvvServiceTest	createRecord_withLegacyGpsSource_convertsToGps	Legacy 'gps' location source on the root request is converted to GPS check-in.	saveLocation() called once; location type resolved to GPS.	Unit	Pass
TC-EVV-BE-007	EvvServiceTest	createRecord_withLegacyManualSource_convertsToPatientAddress	Legacy 'manual' location source converted to PATIENT_ADDRESS check-in.	saveLocation() called once; location type resolved to PATIENT_ADDRESS.	Unit	Pass
TC-EVV-BE-008	EvvSubmissionServiceTest	destinationFor_maryland_returnsMarylandInfoOnly	State code 'MD' maps to correct EVV integration destination.	Returns 'maryland-info-only'.	Unit	Pass
TC-EVV-BE-009	EvvSubmissionServiceTest	destinationFor_dc_returnsDcSandata	State code 'DC' maps to correct EVV integration destination.	Returns 'dc-sandata'.	Unit	Pass
TC-EVV-BE-010	EvvSubmissionServiceTest	destinationFor_virginia_returnsVirginiaMco	State code 'VA' maps to correct EVV integration destination.	Returns 'virginia-mco'.	Unit	Pass
TC-EVV-BE-011	EvvSubmissionServiceTest	destinationFor_lowercase_mapsCorrectly	Lowercase state code 'md' is normalized and routed correctly.	Returns 'maryland-info-only' (case-insensitive routing).	Unit	Pass
TC-EVV-BE	EvvSubmissionServiceTest	destinationFor_unsupportedState_throwsIllegalArgument	Unsupported state code 'TX' raises an exception.	IllegalArgumentException with 'Unsupported state code: TX'.	Unit	Pass

-012						
TC-EVV-BE-013	EvvSubmissionServiceTest	queueForSubmission_callsOutboxEnqueueAndAuditLog	Submitting an EVV record for MD calls outbox enqueue and audit logging.	outbox.enqueue() called with 'maryland-info-only'; audit.log() called with SUBMISSION_QUEUED.	Unit	Pass
TC-EVV-BE-014	EvvOutboxServiceTest	enqueue_withPatientAndMaNumber_usesMANumber	Enqueue EVV record outbox entry where patient has a Medicaid number.	JDBC update executed with MA number as member identifier.	Unit	Pass
TC-EVV-BE-015	EvvOutboxServiceTest	enqueue_withPatientNoMaNumber_usesPatientId	Enqueue EVV record where patient has no MA number; falls back to patient ID.	JDBC update executed using patient.getId() as identifier.	Unit	Pass
TC-EVV-BE-016	EvvOutboxServiceTest	enqueue_withNullPatient_usesUnknown	Enqueue EVV record with null patient reference.	JDBC update executed; member ID defaults to 'UNKNOWN'.	Unit	Pass
TC-EVV-BE-017	EvvOutboxServiceTest	enqueue_patientGetterThrows_fallsBackToUnknown	Patient accessor throws DB error during enqueue; service is resilient.	No exception propagated; JDBC update still executed.	Unit	Pass
TC-EVV-BE-018	EvvOutboxServiceTest	enqueue_withAllLocationFields_includesAllInPayload	Enqueue with check-in and check-out GPS coordinates both populated.	JDBC update executed; payload includes both location sets.	Unit	Pass
TC-EVV-	EvvLocationServiceTest	saveLocation_evvRecordNotFound_throwsAppException	Save GPS location for an EVV record ID	AppException thrown with 'EVV record not found'.	Unit	Pass

BE-019			that does not exist.			
TC-EVV-BE-020	EvvLocationServiceTest	saveLocation_GPS_withCoords_newLocation_savesAndReturnsResponse	Save new GPS check-in location with coordinates for an existing EVV record.	EvvRecordLocation persisted; response contains correct lat/lng and role.	Unit	Pass
TC-EVV-BE-021	EvvLocationServiceTest	saveLocation_GPS_withCoords_existingLocation_updatesAndReturnsResponse	Update an already-stored GPS check-in location with new coordinates.	Existing EvvRecordLocation updated; response returned with new values.	Unit	Pass
TC-EVV-BE-022	EvvOfflineSyncServiceTest	syncOfflineRecords_noPendingItems_doesNothing	Offline sync triggered when the queue has no pending items.	No save calls made; method completes silently.	Unit	Pass
TC-EVV-BE-023	EvvOfflineSyncServiceTest	syncOfflineRecords_itemSyncSucceeds_savesRecord	Offline queue item with UPDATE operation syncs successfully.	EVV record saved with updated status; queue item marked SYNCED.	Unit	Pass
TC-EVV-BE-024	EvvOfflineSyncServiceTest	syncOfflineRecords_itemSyncFails_marksFailedAndSaves	Offline queue item references a record ID that no longer exists.	Queue item status set to FAILED; save called at least once.	Unit	Pass
TC-EVV-BE-025	EvvOfflineSyncServiceTest	syncOfflineRecords_outerException_doesNotPropagate	Database is unavailable when sync is invoked.	RuntimeException swallowed; no exception escapes the service method.	Unit	Pass
TC-EV	EvvOfflineSyncServiceTest	syncQueueItem_createOperation_marksSyncedAndSavesAsPendingReview	Sync a CREATE offline queue item for an	Record status set to PENDING_REVIEW;	Unit	Pass

V-BE-026			existing record.	queue item status set to SYNCED.		
TC-EVV-BE-027	EvvOfflineSyncServiceTest	syncQueueItem_createOperation_confirmedStatus_queuesForSubmission	Sync a CREATE item where the record status is already CONFIRMED.	Record queued for EVV submission via EvvSubmissionService.	Unit	Pass
TC-EVV-BE-028	AuditLoggerTest	log_savesAuditEventWithCorrectFields	Log an EVV audit event with actor, event type, and details.	EvvAuditEvent persisted with correct evvRecord, actorUserId, eventType, deviceInfo and details.	Unit	Pass
TC-EVV-BE-029	AuditLoggerTest	log_withNullDetails_savesAuditEvent	Log an audit event where details and device info are null.	EvvAuditEvent persisted; details and deviceInfo fields are null.	Unit	Pass
TC-EVV-BE-030	DcSandataAltEvvClientTest	destination_returnsDcSandata	Verify the DC Sandata client identifies itself with the correct destination key.	Returns 'dc-sandata'.	Unit	Pass
TC-EVV-BE-031	DcSandataAltEvvClientTest	submit_successfulResponse_doesNotThrow	Submit an EVV record to the DC Sandata AltEVV API; mock returns HTTP 200.	No exception thrown; submission accepted.	Unit	Pass
TC-EVV-BE-032	DcSandataAltEvvClientTest	submit_nonSuccessResponse_throwsRuntimeException	Submit an EVV record; mock API returns HTTP 500.	RuntimeException thrown with 'Sandata submission failed'.	Unit	Pass

TC-EVV-BE-033	VirginiaMcoClientTest	destination_returnsVirginiaMco	Verify the Virginia MCO client reports the correct destination.	Returns 'virginia-mco'.	Unit	Pass
TC-EVV-BE-034	VirginiaMcoClientTest	submit_doesNotThrow	Submit an EVV record via the Virginia MCO stub client.	No exception thrown (stub implementation acknowledged).	Unit	Pass
TC-EVV-BE-035	MarylandInfoOnlyClientTest	destination_returnsMarylandOnlyInfo	Verify the Maryland Info-Only client reports the correct destination.	Returns 'maryland-only-info'.	Unit	Pass
TC-EVV-BE-036	MarylandInfoOnlyClientTest	submit_doesNotThrow	Submit an EVV record via the Maryland Info-Only stub client.	No exception thrown (stub implementation acknowledged).	Unit	Pass
TC-EVV-BE-037	EvvIntegrationClientTest	destination_mockReturnsExpectedValue	Verify the EvvIntegrationClient interface contract for destination().	Mock returns the expected destination string.	Unit	Pass
TC-EVV-BE-038	EvvIntegrationClientTest	submit_mockCanBeInvoked	Verify the EvvIntegrationClient interface contract for submit().	submit() invoked on mock without exception; invocation verified.	Unit	Pass
TC-EVV-BE-	EvvRecordTest	noArgConstructor_createsInstance	Verify no-arg constructor initialises EvvRecord with correct defaults.	isOffline=false, eorApprovalRequired=false, isCorrected=false; all nullable fields null.	Unit	Pass

039						
TC-EVV-BE-040	EvvRecordTest	builder_allFields	Build EvvRecord with all fields via Lombok builder.	All field getters return the expected values.	Unit	Pass
TC-EVV-BE-041	EvvRecordTest	markUnderReview / markApproved / markRejected	Validate status-transition helper methods on EvvRecord.	Status set to UNDER_REVIEW, APPROVED, or REJECTED respectively; updatedAt refreshed.	Unit	Pass
TC-EVV-BE-042	EvvRecordTest	markOffline / markSynced / markSyncFailed	Validate offline sync state helper methods on EvvRecord.	isOffline, syncStatus, and lastSyncAttempt updated correctly for each transition.	Unit	Pass
TC-EVV-BE-043	EvvCorrectionTest	noArgConstructor_createsInstance	Verify EvvCorrection default state on construction.	approvalRequired=false; all nullable fields null.	Unit	Pass
TC-EVV-BE-044	EvvCorrectionTest	builder_allFields	Build EvvCorrection with all fields including original/corrected values maps.	All field getters return expected values; originalValues and correctedValues maps accessible.	Unit	Pass
TC-EVV-BE-045	EvvCorrectionTest	onCreate_setsCorrectedAtWhenNull / doesNotOverwrite	JPA @PrePersist lifecycle method sets correctedAt only when null.	correctedAt populated on first persist; existing timestamp preserved on subsequent calls.	Unit	Pass
TC-EVV-BE	EvvOfflineQueueTest	noArgConstructor_createsInstance	Verify EvvOfflineQueue defaults: syncAttempts=0,	Default values applied; id and other nullable fields are null.	Unit	Pass

-046			syncStatus=PENDING, priority=1.			
TC-EVV-BE-047	EvvOfflineQueueTest	builder_allFields	Build EvvOfflineQueue with all fields including recordData map.	All getters return expected values.	Unit	Pass
TC-EVV-BE-048	EvvOfflineQueueTest	onCreate / markSyncing / markFailed	Lifecycle and state-transition helpers for offline queue items.	queuedAt set by @PrePersist; syncAttempts incremented; FAILED status set on failure.	Unit	Pass
TC-EVV-BE-049	EvvLocationRequestTest	noArgConstructor_defaults	Default EvvLocationRequest has null evvRecordId and no coordinates.	evvRecordId null; CoordinatesDto builder returns expected values.	Unit	Pass
TC-EVV-BE-050	EvvLocationRequestTest	allArgsConstructor / builder	Construct EvvLocationRequest via all-args constructor and Lombok builder.	evvRecordId, role, type, and coordinates populated correctly by both paths.	Unit	Pass

A.2 Frontend Widget & Unit Tests (Flutter)

Framework: flutter_test / WidgetTester | Source root: frontend/test/features/evv/ | Coverage target: ≥ 80 %

TC ID	Test File	Test Group	Test Scenario	Expected Result	Test Type	Status
TC-EVV-FE-001	evv_dashboard_test.dart	EvvDashboard – initial loading state	Dashboard renders without crashing; AppBar shows 'EVV Dashboard'; CircularProgressIndicator visible while loading; Scaffold present.	Widget mounts; AppBar title, loading spinner, and Scaffold all found by widget finder.	Widget	Pass

TC-EVV-FE-002	evv_dashboard_test.dart	EvvDashboard – after loading completes	After async data load fails (no server), loading spinner removed and dashboard body rendered.	CircularProgressIndicator gone; main dashboard body widgets visible.	Widget	Pass
TC-EVV-FE-003	patient_selection_page_test.dart	PatientSelectionPage – initial render	Patient selection page renders; AppBar shows 'Select Patient'; loading spinner shown while fetching; no ListView during load.	Widget mounts; spinner present; ListView absent during load.	Widget	Pass
TC-EVV-FE-004	start_visit_page_test.dart	StartVisitPage – initial render	Start Visit page renders; AppBar shows 'Start Visit'; spinner shown while loading; different patientId values accepted.	Widget mounts with any patientId; Scaffold, AppBar, and loading state all present.	Widget	Pass
TC-EVV-FE-005	checkin_location_page_test.dart	CheckinLocationPage – initial render / loading	Check-in location page renders; AppBar shows 'Check-In Location'; loading spinner displayed.	Widget finds CheckinLocationPage; AppBar title and loading indicator found.	Widget	Pass
TC-EVV-FE-006	visit_in_progress_page_test.dart	VisitInProgressPage – initial render	Visit-in-progress page renders; handles null user (unauthenticated) error path; timer lifecycle managed.	VisitInProgressPage mounts via GoRouter; null-user path shows auth error without crash.	Widget	Pass
TC-EVV-FE-007	checkout_location_page_test.dart	CheckoutLocationPage – initial loading & error state	Check-out page renders; AppBar shows 'Check-Out Location'; error state displayed after API failure with Try Again button.	Loading spinner replaced by error icon and 'Error Loading Patient' text; Try Again button triggers reload.	Widget	Pass
TC-EVV-FE-008	checkout_location_page_test.dart	CheckoutLocationPage – location selection state	Patient address and GPS location card options rendered; EVV compliance banner shown; address formatted correctly.	Both location cards visible; Select Patient Address and Get My GPS Location buttons present; address formatting handles null/partial data.	Widget	Pass
TC-EVV-FE-009	checkout_location_page_test.dart	CheckoutLocationPage – constructor parameter variations	Page renders with optional lat/lng, scheduledVisitId, notes, and custom duration parameters.	Widget mounts without crash for all parameter combinations.	Widget	Pass
TC-EVV-	visit_complete_page_test.dart	VisitCompletePage – initial render & error state	Visit Complete page renders; loads patient data; shows error state on	Scaffold and AppBar with 'Visit Complete' title;	Widget	Pass

FE-010			API failure with Try Again button; Cancel button navigates to /evv.	error icon and Try Again button after API fails; Cancel navigates correctly.		
TC-EVV-FE-011	visit_complete_page_test.dart	VisitCompletePage – location type & duration variations	Page renders correctly for GPS, PATIENT_ADDRESS, and mixed location types; handles zero to 8-hour duration range.	Widget mounts for all location type combinations and duration values without crash.	Widget	Pass
TC-EVV-FE-012	visit_completed_success_page_test.dart	VisitCompletedSuccessPage – initial render	Success page renders; AppBar shows 'Visit Completed'; Close button navigates to /dashboard; loading indicator shown initially.	Widget mounts; Close and back arrow buttons present; loading spinner visible.	Widget	Pass
TC-EVV-FE-013	visit_completed_success_page_test.dart	VisitCompletedSuccessPage – error state & auth	Error state shows after API failure; null user triggers auth error path; Try Again button restarts load.	Error heading, error_outline icon, and Try Again button rendered; null-user error shown without crash.	Widget	Pass
TC-EVV-FE-014	visit_completed_success_page_test.dart	VisitCompletedSuccessPage – edge cases and dark theme	Widget handles special characters in notes, zero/large durations, negative coordinates, same check-in/out time, narrow/wide layouts, and dark theme.	Widget renders without crash for all edge case inputs and screen sizes.	Widget	Pass
TC-EVV-FE-015	evv_record_review_test.dart	EvvRecordReviewPage – initial render & status filter	Record review page renders; AppBar shows 'All EVV Records'; loading spinner visible; status filter dropdown defaults to 'All Statuses'.	Spinner, AppBar, Scaffold, DropdownButtonFormField with correct default found.	Widget	Pass
TC-EVV-FE-016	evv_record_review_test.dart	EvvRecordReviewPage – record list from HTTP mock	Records loaded from mocked HTTP; UNDER_REVIEW/APPROVED/REJECTED/OFFLINE status badges displayed; service type, date, and time range shown in subtitle.	ListTile items rendered with correct status badges, state code, and record details.	Widget	Pass
TC-EVV-FE-017	evv_record_review_test.dart	EvvRecordReviewPage – status filter interaction	Filtering by APPROVED/UNDER_REVIEW/REJECTED shows only matching records; no-match filter shows helper text; switching back to ALL restores full list.	ListView filtered correctly for each status option; empty-filter message shown when no matches.	Widget	Pass
TC-EVV-FE-018	evv_record_review_test.dart	EvvRecordReviewPage – review dialog (approve / reject / export EDI)	Tapping a record opens review dialog; Approve/Reject/Cancel/Export EDI buttons present; approval calls reviewRecord API; rejection shows snackbar.	Dialog opens with record details; all action buttons functional; comment field present;	Widget	Pass

				success/error snackbar shown.		
TC-EVV-FE-019	evv_record_review_test.dart	EvvRecordReviewPage – location sections in dialog	Dialog shows GPS coordinates, PATIENT_ADDRESS with address text, 'Not recorded' for null source.	Location section renders correctly for each location source variant.	Widget	Pass
TC-EVV-FE-020	evv_record_review_test.dart	EvvRecordReviewPage – EVV data model tests	EvvRecord, EvvSearchRequest, EvvSearchResult, EvvCorrectionRequest, EvvOfflineQueue, EvvCorrection, EvvRecordRequest all construct correctly and serialise/deserialise JSON.	All model assertions pass; fromJson parses correctly; toJson includes required fields.	Unit	Pass
TC-EVV-FE-021	evv_visit_history_test.dart	EvvVisitHistoryPage – initial render & search filters	Visit history page renders; search filter form shows Patient Name, Service Type, Caregiver ID, Date Range, State, and Status fields.	All form fields and labels present; text can be entered in text fields.	Widget	Pass
TC-EVV-FE-022	evv_visit_history_test.dart	EvvVisitHistoryPage – empty state, filtering, and clear	Empty state with 'No records found' shown after failed search; Clear Filters button resets all fields; dropdowns openable.	Empty state icon and text rendered; text fields cleared on tap; no spinner after load.	Widget	Pass
TC-EVV-FE-023	evv_visit_history_test.dart	EvvVisitHistoryPage – error handling	Error snackbar shown when search API call fails.	Snackbar displayed with error message prefix.	Widget	Pass
TC-EVV-FE-024	evv_corrections_test.dart	EvvCorrectionsPage – initial render	Corrections page renders; AppBar shows 'EVV Corrections & Approvals'; loading spinner visible; no ListView during load.	Scaffold, AppBar, CircularProgressIndicator present; ListView absent.	Widget	Pass
TC-EVV-FE-025	evv_offline_sync_test.dart	EvvOfflineSyncPage – initial render	Offline sync page renders; AppBar shows 'Offline Sync'; loading spinner shown; no list during load.	Scaffold, AppBar with correct title, CircularProgressIndicator present; ListView absent.	Widget	Pass
TC-EVV-FE-026	schedule_page_test.dart	SchedulePage – initial render	Schedule page renders; AppBar shows 'EVV Visit Schedules'; loading spinner shown; refresh button in AppBar.	Widget mounts; all required AppBar elements and spinner present.	Widget	Pass
TC-EVV-	schedule_page_test.dart	SchedulePage – empty visits state	After successful HTTP response with no visits, empty state rendered with Schedule New Visit	Empty state text, Schedule New Visit button, and	Widget	Pass

FE-027			button and zero-count summary cards.	summary cards with zero counts all visible.		

A.3 Test Execution Summary

Layer	Test Files	Total Test Cases	Automated	Pass	Fail	Blocked
Backend (Java/JUnit)	EvvServiceTest, EvvSubmissionServiceTest, EvvOutboxServiceTest, EvvLocationServiceTest, EvvOfflineSyncServiceTest, AuditLoggerTest, DcSandataAltEvvClientTest, VirginiaMcoClientTest, MarylandInfoOnlyClientTest, EvvIntegrationClientTest, EvvRecordTest, EvvCorrectionTest, EvvOfflineQueueTest, EvvLocationTypeTest, EvvLocationRoleTest, EvvAuditEventTest, EvvRecordLocationTest, EvvLocationRequestTest	50	50	50	0	0
Frontend (Flutter/Widget)	evv_dashboard_test, patient_selection_page_test, start_visit_page_test, checkin_location_page_test, visit_in_progress_page_test, checkout_location_page_test, visit_complete_page_test, visit_completed_success_page_test, evv_record_review_test, evv_visit_history_test, evv_corrections_test, evv_offline_sync_test, schedule_page_test	27	27	27	0	0
TOTAL	31 test files	77	77	77	0	0

A.4 Requirements Traceability Matrix (EVV)

Requirement ID	Feature	Test Case IDs	Status
REQ-GPS-001	GPS Check-In (F-002)	TC-EVV-BE-003, TC-EVV-BE-004, TC-EVV-BE-019, TC-EVV-BE-020, TC-EVV-BE-021, TC-EVV-FE-005, TC-EVV-FE-007, TC-EVV-FE-008	Covered
REQ-GPS-002	GPS Check-Out (F-003)	TC-EVV-FE-007, TC-EVV-FE-008, TC-EVV-FE-009, TC-EVV-FE-010, TC-EVV-FE-011	Covered

REQ-VISIT-001	Visit Recording (F-004)	TC-EVV-BE-001 – TC-EVV-BE-007, TC-EVV-FE-006, TC-EVV-FE-010, TC-EVV-FE-011	Covered
REQ-PATIENT-001	Patient Search / Selection (F-005)	TC-EVV-FE-003, TC-EVV-FE-004	Covered
REQ-API-001 / REQ-API-002	HHAeXchange / State EVV Submission (F-010 / F-011)	TC-EVV-BE-008 – TC-EVV-BE-013, TC-EVV-BE-030 – TC-EVV-BE-038	Covered
REQ-OFFLINE-001	Offline Mode (F-019)	TC-EVV-BE-022 – TC-EVV-BE-027, TC-EVV-BE-046 – TC-EVV-BE-048, TC-EVV-FE-025	Covered
REQ-SYNC-001	Data Synchronisation (F-020)	TC-EVV-BE-014 – TC-EVV-BE-018, TC-EVV-BE-022 – TC-EVV-BE-027	Covered
REQ-HISTORY-001	Visit History (F-007)	TC-EVV-FE-021, TC-EVV-FE-022, TC-EVV-FE-023	Covered
REQ-DASH-001	Dashboard Metrics (F-014)	TC-EVV-FE-001, TC-EVV-FE-002	Covered
REQ-AUDIT	EVV Audit Logging	TC-EVV-BE-028, TC-EVV-BE-029	Covered
REQ-EVV-CORR	EVV Corrections & EOR Approval	TC-EVV-BE-043 – TC-EVV-BE-045, TC-EVV-FE-018, TC-EVV-FE-019, TC-EVV-FE-024	Covered

---

End of Test Plan Document

Document History:

Version	Date	Author	Description
0.1	02/07/2026	Parthav Dani	Initial draft based on IEEE 829-1998 template
0.2	02/10/2026	Parthav Dani	Added environmental needs and test data specifications
1.0	02/14/2026	Parthav Dani	Final version incorporating stakeholder feedback, ready for approval

---

Notes:

- This test plan follows IEEE Standard 829-1998 for Software Test Documentation
- All referenced documents (SRS, design specs) should be consulted for detailed requirements
- Test execution will begin Week 6 (02/17/2026) pending test plan approval

- Updates to this plan require change control approval from Test Manager and Project Manager
- Contact Parthav Dani (parthav.dani@student.umgc.edu) for questions or clarifications

Software Test Plan

System: CareConnect

Document Version: 3.0

Prepared by: CareConnect Team B

University of Maryland Global Campus | SWEN 670

Instructor: Dr. Mir Assadullah

Date: March 31, 2026

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

Table of Contents

Document Revision History

Author	Date	Reason for Changes	Version
CareConnect Team B	02/07/2026	Initial IEEE 829-2008 formatted STP for Milestone 2	1.0
CareConnect Team B	03/11/2026	Refined Section 5.9 to address feedback regarding "terse" descriptions by removing the procedure column from Table 2 and providing detailed test cases. Updated test priorities and scope to reflect current project implementation and testing objectives. Adjusted testing approach to align with the latest system functionality and project milestones.	2.0
CareConnect Team B	3/31/2026	Modified section 5.10 with the actual test results and added Appendix A supplement	3.0

1. Introduction

1.1 Test Plan Identifier

STP-02072026-001

This plan conforms to IEEE Std 829 test documentation structure

1.2 Purpose

The purpose of this Software Test Plan (STP) is to define the testing strategy used to verify that the CareConnect system satisfies its documented requirements. This plan outlines the scope of testing, testing approach, and the relationship between requirements and planned test activities to ensure quality, reliability, and consistency with project constraints.

This test plan is derived from the CareConnect Software Requirements Specification (SRS) and the Technical Design Document (TDD) developed for Milestone 2. Testing efforts described in this document focus on validating functional and non-functional requirements, including offline functionality, data synchronization, infrastructure migration, security, and observability.

1.3 Scope

The scope of this test plan includes in-depth testing of Team B–owned enhancements as defined in the SRS. In addition, a targeted regression test suite is included for selected inherited MVP workflows to ensure core system stability. The inherited workflows included for regression testing are user registration, login and logout, calendar assistant, patient dashboard, caregiver dashboard, Electronic Visit Verification (EVV), invoice assistant, and USPS mail assistant. Functionality outside Team B enhancements and the listed inherited workflows is considered out of scope for this milestone. This document also establishes requirements traceability to ensure that all tested requirements are verifiable and mapped to one or more test cases. Features not planned for testing under this milestone are listed in Section 4; Features Not to Be Tested.

Summary of Scope Validation for Team B:

- **BNS 5 (Offline):** Covered by "offline functionality" and "data synchronization."
- **BNS 6 (Infra):** Covered by "infrastructure migration."
- **BNS 7 (Telemetry):** Covered by "observability."
- **Legacy Issues:** Covered by the "targeted regression test suite."

1.4 Definitions, Acronyms, and Abbreviations

A comprehensive list of acronyms, definitions, and abbreviations used in this document is provided in Tables 1.4.1 and 1.4.2.

Table 1.4.1

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 1.4.2

Acronyms

Acronyms	Definitions
AC-###	Acceptance criterion associated with a requirement or use case, verified by one or more test cases
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 (Architectural style for web APIs)
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

1.5 References

1. Angeles, D., Bias, T., Gaucin, J., Chen, F., Curran, L., Grbreegziabhere, A., Harding, A. M.,Malla-Paudel, A., Ramirez, M., Raphael, E., Truong, D., Vecchioni, A., & Yawn, C. (2025). Software requirements document: CareConnect. University of Maryland Global Campus. https://umgc-cappms.azurewebsites.net/previousprojects
2. CareConnect Developer Team. (2025). CareConnect project documentation and programmer guides. GitHub Repository. https://github.com/umgc/2025_fall/tree/developer/careconnect2025/docs/project-docs
3. IEEE, IEEE Std 829-2008: Standard for Software and System Test Documentation, 2008. [Online]. Available: https://standards.ieee.org/ieee/829/3787/
4. Mir, A. (2025). CareConnect high-level requirements. University of Maryland Global Campus.
5. Mir, A. (2026). SWEN 670 fall Team B BNS distribution. University of Maryland Global Campus.
6. Software Requirements Specification (SRS). (2025). CaPPMS. https://umgc-cappms.azurewebsites.net/previousprojects
7. University of Maryland Global Campus, CareConnect Software Requirements Specification (SRS), (Team B) 2026.
8. University of Maryland Global Campus, CareConnect Project Management Plan (PMP), (Team B), 2026.
9. University of Maryland Global Campus, CareConnect Technical Design Document (TDD), (Team B), 2026.
10. World Wide Web Consortium. (2025). Web content accessibility guidelines (WCAG) 3.0. https://www.w3.org/TR/wcag-3.0/

2. Test Items

The following software items and configurations comprise the test target for Milestone 3. All items are maintained within the *umgc/2026_spring_careconnect* github repository specifically under the *team_b* branch unless otherwise specified.

2.1 Software Components and Versioning

- **CareConnect Client Application:** Flutter mobile/web client builds (Revision: **Commit SHA** *6db20e276d01d4282282a3a158c6f95dd6dad22f*).
- **Offline Persistence Layer (BNS 5):** Local Drift/SQLite schema (Version 1.0.0) and encrypted storage modules.
- **Synchronization Service:** Background sync logic, retry handlers, and Outbox Queue management.
- **Backend Services:** CareConnect APIs (Staging Environment) supporting EVV, Invoices, and Mail Assistant.
- **Infrastructure as Code (BNS 6):** AWS CloudFormation templates and associated deployment pipelines.
- **Telemetry & Observability Stack (BNS 7):** Structured logging handlers, OpenTelemetry instrumentation and correlation logic, metric/log processing in the selected observability backend, and alerting/notification thresholds (for example CloudWatch/X-Ray and SNS when enabled)

2.2 Transmittal Media & Hardware Requirements

Test items are transmitted via GitHub Version Control. Before testing can begin, the following logical transformations are required:

- **Environment Provisioning:** AWS CloudFormation stacks must be successfully "up" in the staging environment.
- **Local Build:** The Flutter client must be compiled for the target platform (iOS/Android/Web) using the flutter build command with specific environment variables for the staging API.

2.3 References to Item Documentation

The behavior and configuration of the test items are defined in the following documents:

- **Requirements:** CareConnect Team B Software Requirements Specification (SRS) v.3.0.
- **Design:** CareConnect Team B Technical Design Document (TDD) v.2.0.
- **Users:** CareConnect Team B Programmer’s Guide v.1.0.
- **Operations/Installation:** CareConnect Team B Deployment and Operations Guide v.1.0.

2.4 Incident Reports and Exclusions

- **Exclusions:** Legacy modules not modified by Team B are excluded from primary testing unless they impact regression workflows.

3. Features to Be Tested

Features are traced to team B's SRS requirement identifiers (REQ-SB-###) and verified by test cases documented in team B's Requirements Traceability Matrix (RTM). Each feature set corresponds to a specific Test Design Specification (TDS) as outlined below.

3.1 Team B Enhancements

Table 3.1.1

Team B TDS Mapping

Feature Set	Requirements	Test Design Specification (TDS)
Offline Persistence	REQ-SB-101, 102, 103, 108	**TDS-BNS5-DATA**: Verification of Drift/SQLite schema, encryption at rest, and local dataset limitations.
Connectivity & Sync	REQ-SB-104, 105, 106, 107	**TDS-BNS5-SYNC**: Verification of network state transitions, retry logic, and conflict resolution (Outbox Pattern).
Infrastructure as Code (IaC)	REQ-SB-201 through 207	**TDS-BNS6-AWS**: Verification of CloudFormation provisioning, stack reproducibility, and rollback triggers.
Telemetry & Logging	REQ-SB-109, 110, 301..307	**TDS-BNS7-TEL**: Verification of log ingestion, metrics generation, distributed tracing (via OpenTelemetry exporters when enabled), alerting thresholds, and PHI exclusion.
Analytics Collection	REQ-SB-401 through 407	**TDS-BNS7-ANA**: Verification of anonymous data collection, retention policies, and PII-free data pipelines.

3.2 Targeted Regression Features

These inherited features are validated for regression stability and integration impact. The goal is to ensure that the introduction of BNS 5, 6, and 7 does not degrade existing baseline functionality.

- **Identity Management:** User registration, Login, and Logout.
- **Core Logistics:** Calendar assistant, EVV workflow, and Invoice assistant.
- **Navigation/Support:** Patient dashboard (data access), Caregiver dashboard (data access), and USPS mail assistant.
- **Test Design Specification:** All regression items are covered under **TDS-REG-MVP**, which focuses on end-to-end workflow integrity during environment transitions.

4. Features Not to Be Tested

The following are not planned for testing under this STP (unless required for defect reproduction):

Table 4.1

Features not tested

Feature / Combination	Reason for Exclusion
Full-Scale Load & Stress Testing	Production-level endurance and high-concurrency stress testing are outside the scope of the staging environment and current project constraints.
UI/UX Design of Legacy Dashboards	Team B is responsible for the data collection and telemetry logic (BNS 7) only. Testing is limited to data availability/integration; visual layout and UI bugs in legacy dashboards are out of scope.
Long-Term Analytics Trends	Testing is focused on schema integrity, PHI/PII privacy, and data collection correctness. Longitudinal trend analysis requires data durations exceeding the project timeline.
Production Emergency Integrations	The "SOS" and emergency notification features are simulated for a non-production environment and will not be tested against real-world emergency dispatch systems.
New Accessibility Features	Development and validation are restricted to "no regression" checks for existing behavior; no new accessibility standards are being implemented in this milestone.
External Payment & GPS Hardware	Third-party payment gateways and specific external GPS hardware drivers are considered out of scope as they are not touched by Team B enhancements.

5. Approach

5.1 Test Strategy Overview

Team B’s testing strategy combines structured manual validation with targeted automated checks to verify correctness, reliability, data privacy, and requirements traceability. The test design is derived directly from the SRS and TDD to ensure that all BNS enhancements (5, 6, and 7) are validated across local, staging, and AWS environments that mirror the final deployment state.

Comprehensiveness and Completion Criteria:

- To ensure adequate coverage, 80% of "High" and "Medium" priority requirements from the RTM must have at least one associated test case executed.
- Testing is considered complete when 80% of planned test cases have been executed, and all Priority 1 (Blocker) and Priority 2 (Critical) defects have been resolved or mitigated with a documented workaround.
- Requirements are traced via a **RTM**, mapping each SRS identifier to specific test cases and execution results to ensure no functional gaps exist.

Significant Constraints:

- Access to the AWS Staging environment is required for BNS 6 and BNS 7 validation.
- Testing activities are scheduled to run concurrently with development sprints, with final verification and regression cycles concluding by the end of the 12-week project lifecycle.

5.2 Test Levels

- **Functional Testing:** Individual validation of Team B features against specific acceptance criteria. This includes verifying local Drift database CRUD operations, encryption at rest, and UI feedback for offline states.
- **Integration Testing:** Validation of the handshake between the Flutter client, the Synchronization Service, and AWS API Gateway. Key focus areas include offline-to-online transitions, outbox queue processing, and conflict resolution during data reconciliation.
- **System Testing:** End-to-end validation of the data capture lifecycle. This ensures that the application logic correctly identifies system events (errors, sync status, user actions) and successfully generates the telemetry packets required for BNS 7.
- **Targeted Regression Testing:** Execution of high-priority test cases for inherited MVP workflows (Registration, Login, EVV) to ensure that BNS 5/6/7 integrations have not introduced side effects into the legacy baseline.

5.3 Test Types and Techniques

- **Requirements-Based Testing:** Mapping every SRS requirement to a Test Case via the RTM to ensure 100% functional coverage.
- **Scenario-Based Testing:** Execution of real-world "Caregiver Workflows," focusing on how the app captures data during "Dead Zone" transitions (loss of 5G/Wi-Fi) and partial failure conditions.

- **Negative Testing:** Verification of system boundaries, including attempting to access restricted offline modules and ensuring that the capture logic correctly logs "Denied Access" events.
- **Security/Privacy Validation:** Manual payload inspection of the captured data packets (before they leave the device) to verify that no PHI/PII is included in the telemetry or analytics strings.
- **Infrastructure Validation:** Testing of BNS 6 CloudFormation templates to ensure that the *definitions* for the data pipelines (metrics, traces, and alert thresholds) are correctly provisioned and ready to receive data.

5.4 Automation Strategy

Automation is used where it provides high confidence and repeatability, primarily in CI/CD for infrastructure validation (template validation, stack deployment, rollback checks) and in scripted checks for telemetry/analytics payload inspection. Manual execution is used for exploratory and UX-dependent flows (dashboards, assistants) and for complex offline scenarios. In addition to infrastructure validation, automated backend unit tests validate telemetry services, event persistence, RBAC access controls, and telemetry schema behavior. Test coverage reports are generated through the CI pipeline to ensure telemetry components remain verifiable across builds.

5.5 Test Data Strategy

- Use synthetic test users and synthetic patient records; do not use real PHI/PII.
- Seed baseline datasets for regression workflows (dashboards, invoices, EVV).
- Create controlled conflict datasets for sync conflict resolution tests.
- Log/trace/analytics validation uses generated events containing representative fields but excludes sensitive content.

5.6 Requirements Traceability

Traceability is maintained using the RTM in Section 5.8. Every requirement in scope must map to one or more test cases. Coverage is reviewed during test readiness and updated as defects or scope changes occur.

5.7 Entry and Exit Criteria

Entry Criteria:

- Milestone 2 build deployed to test/staging environment with required feature flags enabled.

- Test accounts and roles provisioned (including admin for telemetry RBAC testing).
- AWS test account access validated for CloudFormation deployments and CloudWatch inspection.
- RTM reviewed and test cases baselined.

Exit Criteria:

- All planned test suites executed (or formally waived with justification).
- No open Critical or High defects; Medium/Low defects triaged with disposition.
- Pass/fail criteria in Section 7 met; test summary report completed.
- Evidence artifacts stored (logs, screenshots, pipeline outputs, CloudFormation events).

5.8 Requirements Traceability Matrix

Table 5.8.1

BNS 5 – Offline Data Storage & Synchronization

REQ ID	Requirement Description	Child REQ(s)	Use Case	Acceptance Criteria	Test Case ID(s)	Test Type
REQ-SB-101	Offline local storage	101-1 → 101-5	UC-001	AC-SB-OFF-001	TC-OFF-01	Functional
REQ-SB-102	Offline module restriction	102-1 → 102-4	UC-001	AC-SB-OFF-002	TC-OFF-05	Functional
REQ-SB-103	Local encryption at rest	101-2	UC-001	AC-SB-OFF-003	TC-SEC-02	Security
REQ-SB-104	Connectivity detection	—	UC-001	AC-SB-OFF-004	TC-SYNC-01	Integration
REQ-SB-105	Automatic synchronization	105-1	UC-001	AC-SB-OFF-005	TC-SYNC-01	Integration
REQ-SB-106	Duplicate prevention	106-1	UC-001	AC-SB-OFF-006	TC-SYNC-02	Integration
REQ-SB-107	Conflict resolution strategy	107-1	UC-001	AC-SB-OFF-007	TC-SYNC-02	Integration

REQ-SB-108	Limited local dataset scope	108-1	UC-001	AC-SB-OFF-008	TC-SEC-03	Security
REQ-SB-109	Sync event logging	109-1	UC-001	AC-SB-OFF-009	TC-OBS-01	Non-Functional
REQ-SB-110	Sync failure notification	—	UC-001	AC-SB-OFF-010	TC-SYNC-03	Integration
REQ-SB-111	Connectivity-based offline behavior override	111-1	UC-001	AC-SB-OFF-011	TC-OFF-06	Integration
REQ-SB-112	User-controlled sync pause while online	112-1	UC-001	AC-SB-OFF-012	TC-SYNC-06	Integration
REQ-SB-113	Durable queue replay across reconnect cycles	113-1	UC-001	AC-SB-OFF-013	TC-SYNC-07	Integration
REQ-SB-114	Deterministic ordering and idempotent sync processing	114-1	UC-001	AC-SB-OFF-014	TC-SYNC-08	Integration

Table 5.8.2

BNS 6 – Infrastructure Migration (CloudFormation)

REQ ID	Requirement Description	Child REQ(s)	Use Case	Acceptance Criteria	Test Case ID(s)	Test Type
REQ-SB-201	Provision via CloudFormation	201-1	UC-002	AC-SB-INF-001	TC-INF-01	Non-Functional
REQ-SB-202	Version-controlled templates	202-1	UC-002	AC-SB-INF-002	TC-INF-02	Non-Functional
REQ-SB-203	CI/CD stack deployment	—	UC-002	AC-SB-INF-003	TC-INF-03	Non-Functional
REQ-SB-204	Remove Terraform configs	204-1	UC-002	AC-SB-INF-004	TC-INF-04	Non-Functional
REQ-SB-205	Stack rollback support	205-1	UC-002	AC-SB-INF-006	TC-INF-05	Non-Functional
REQ-SB-206	Reproducible environments	206-1	UC-002	AC-SB-INF-006, AC-SB-	TC-INF-06	Non-Functional

				INF-007		
REQ-SB-207	Template validation	207-1	UC-002	AC-SB-INF-005	TC-INF-07	Non-Functional

Table 5.8.3

BNS 7 – Observability (Logs, Metrics, Traces)

REQ ID	Requirement Description	Child REQ(s)	Use Case	Acceptance Criteria	Test Case ID(s)	Test Type
REQ-SB-301	Centralized logging	301-1	UC-003	AC-SB-OBS-001	TC-OBS-01	Non-Functional
REQ-SB-302	Performance metrics collection	302-1	UC-003	AC-SB-OBS-002	TC-OBS-02	Non-Functional
REQ-SB-303	Distributed tracing	303-1	UC-003	AC-SB-OBS-003	TC-OBS-03	Non-Functional
REQ-SB-304	Log retention policy	—	UC-003	AC-SB-OBS-004	TC-OBS-04	Non-Functional
REQ-SB-305	RBAC for telemetry	—	UC-003	AC-SB-OBS-005	TC-SEC-04	Security
REQ-SB-306	Alert threshold monitoring	—	UC-003	AC-SB-OBS-006	TC-OBS-05	Non-Functional
REQ-SB-307	PHI exclusion enforcement	307-1	UC-003	AC-SB-OBS-007, AC-SB-OBS-008	TC-SEC-01	Security

Table 5.8.4

BNS 7 – Anonymous Feature Analytics

REQ ID	Requirement Description	Child REQ(s)	Use Case	Acceptance Criteria	Test Case ID(s)	Test Type
REQ-SB-401	Anonymous usage collection	401-1	UC-004	AC-SB-ANA-001	TC-ANA-01	Non-Functional
REQ-SB-402	No PII/PHI in analytics	402-1	UC-004	AC-SB-ANA-002	TC-SEC-03	Security
REQ-SB-	Usage	403-1	UC-004	AC-SB-	TC-ANA-	Non-

403	frequency tracking			ANA-003	02	Functional
REQ-SB-404	Feature adoption metrics	—	UC-004	AC-SB-ANA-004	TC-ANA-03	Non-Functional
REQ-SB-405	Aggregated admin dashboards	405-1	UC-004	AC-SB-ANA-005	TC-ANA-04	Non-Functional
REQ-SB-406	Separate analytics storage	—	UC-004	AC-SB-ANA-006, AC-SB-ANA-008	TC-SEC-05	Security
REQ-SB-407	Configurable retention policies	—	UC-004	AC-SB-ANA-007	TC-ANA-05	Non-Functional
REQ-SB-408	Telemetry database persistence enforcement	408-1	UC-004	AC-SB-ANA-009	TC-ANA-06	Security
REQ-SB-409	User opt-out disables analytics collection	409-1	UC-004	AC-SB-ANA-010	TC-ANA-07	Functional

5.9 Test Suites

The following suites define execution groupings for Milestone 2. Each suite is composed of detailed test cases in Section 9 artifacts.

Table 5.9.1

Test Suite Summary and Requirement Mapping

Suite ID	Suite Name	Type	Primary Requirement Coverage	Test Case Range	Objective / Expected Outcome
TS-01	Offline Data Capture	Integration	REQ-SB-101, REQ-SB-104	TC-OFF-01..03,	Verify supported features store data locally offline; entries

				TC-OFF-06	persist across restarts; queued for sync.
TS-02	Synchronization on Reconnect	Integration	REQ-SB-105, REQ-SB-106	TC-SYNC-01..02	Verify auto-sync triggers on reconnect; queue clears on success; partial failures remain queued.
TS-03	Synchronization Failure Handling	Resilience	REQ-SB-105..114	TC-SYNC-03..08	Verify safe retry behavior; no duplicates; failures logged without sensitive data; user notified.
TS-04	Infrastructure Deployment Validation	Non-Functional	REQ-SB-201..207	TC-INF-01..07	Validate CloudFormation stacks deploy/update/rollback via CI/CD; repeatability and validation checks.
TS-05	Observability & Analytics Validation	System	REQ-SB-301..307, REQ-SB-401..409	TC-OBS-01..05, TC-ANA-01..07	Validate logs/metrics/traces and anonymous analytics; verify PHI/PII exclusion and pipeline separation.
TS-06	User Registration (Regression)	Regression	Inherited MVP	TC-REG-REGN-01..02	Validate registration succeeds with valid input; rejects invalid input.
TS-07	Login/Logout (Regression)	Regression	Inherited MVP	TC-REG-AUTH-01..03	Validate login success/failure and session termination on logout.
TS-08	Calendar Assistant (End-to-End)	System/Regression	Inherited MVP	TC-REG-CAL-01..03	Validate calendar view/create/update; ensure no regression with offline-related

					enhancements.
TS-09	Patient Dashboard (Regression)	Regression	Inherited MVP	TC-REG-PD-01..02	Validate dashboard loads; widgets render correctly.
TS-10	Caregiver Dashboard (Regression)	Regression	Inherited MVP	TC-REG-CD-01..03	Validate dashboard navigation and assigned patient info display.
TS-11	Invoice Assistant (Regression)	Regression	Inherited MVP	TC-REG-INV-01..02	Validate invoice generation and details view.
TS-12	USPS Mail Assistant (Regression)	Regression	Inherited MVP	TC-REG-USPS-01..02	Validate access and mail request submission workflows.

5.9.1 Detailed Test Suite Procedures

TS-01: Offline Data Capture

- **Objective:** To ensure that data entered during offline states (Mood Tracking, Medication Logging) is saved locally and remains persistent.
- Procedure:
    1. Enter the application and disable all network connectivity (Airplane Mode).
    2. Perform data entry for a Mood log and a Medication log.
    3. Close the application and perform a device restart.
    4. Re-open the application while still offline and verify the local dashboard.
- **Expected Outcome:** Data persists locally; no data loss occurs after restart; a "Pending Sync" status is visible to the user.

TS-02: Synchronization on Reconnect

- **Objective:** To verify that the application automatically pushes local data to the server once a connection is re-established.
- Procedure:
    1. Start with a populated offline queue (from TS-01).
    2. Enable Wi-Fi or Cellular data.

3. Wait for the background sync interval (default 30 seconds).

- **Expected Outcome:** The local queue clears automatically. A "Sync Complete" notification appears. Database records on the backend match the local timestamps of the offline entries.

TS-03: Synchronization Failure Handling

- **Objective:** To validate system resilience during network instability or server-side errors.
- Procedure:
    1. Trigger a sync while the network is unstable (high packet loss).
    2. Simulate a server 500 error during a push.
- **Expected Outcome:** System attempts a retry with exponential backoff. No duplicate entries are created. Errors are logged anonymously (no PHI/PII).

TS-04: Infrastructure Deployment Validation

- **Objective:** To confirm that the CloudFormation stacks and CI/CD pipelines deploy the environment accurately.
- Procedure:
    1. Execute the deployment pipeline.
    2. Inspect the AWS Console for stack status and resource tags.
    3. Trigger a manual rollback and verify resource cleanup.
- **Expected Outcome:** Successful deployment with 100% repeatability; no orphaned resources after rollback.

TS-05: Observability & Analytics Validation

Objective: To verify that system logs, performance metrics, and application traces are accurately captured and stored in the PostgreSQL observability tables without exposing sensitive PII or PHI.

Procedure:

1. **Traffic Generation:** Perform a series of standard actions in the application to generate telemetry.
2. **Error Simulation:** Intentionally trigger an application error to generate error logs.
3. **Database Inspection:** Access the PostgreSQL instance and query the telemetry_events table, including the session_id correlation field introduced in schema migration V38, to verify that telemetry events are properly grouped per application session.
4. **Data Scrubbing Audit:** Search the query results for specific sensitive strings used during testing (e.g., Patient Name, specific Medication names).

Expected Outcome: The PostgreSQL table successfully record timestamps, error codes, and performance latency for each action.

- All logs contain performance metrics and trace IDs for debugging.
- **Security Validation:** All PII/PHI is confirmed to be masked, hashed, or entirely excluded from the database records; no sensitive data is stored in the telemetry stream.

Regression Testing (TS-06 to TS-10, TS-11, TS-12)

These suites utilize existing test scripts from the MVP to ensure that the new Offline Mode UI and backend sync logic have not broken core functionalities like User Registration, Login, Calendar Assistant, or the Dashboards [1, 2].

5.10 Test Cases and Results

The following section encompasses all the test suites and their test cases performed to ensure requirements and their pass or fail status.

Figure 5.10.1

Team B Unit Tests Coverage Report

Coverage Report

Input: coverage\telemetry_localdb.info
Files: 12
Excluded files: lib\config\env_constant.dart, lib\services\auth_token_manager.dart
Overall line coverage: 426 / 530 (80.4%)
Overall branch coverage: N/A

Per-file coverage

File	Covered Lines	Total Lines	Line %
lib\features\telemetry\telemetry.dart	48	58	82.8%
lib\features\telemetry\telemetry_guardrails.dart	10	10	100%
lib\features\telemetry\telemetry_settings.dart	12	12	100%
lib\services\local_db\app_database.dart	78	80	97.5%
lib\services\local_db\app_database_stub.dart	53	83	63.9%
lib\services\local_db\db_encryption_service.dart	15	15	100%
lib\services\local_db\local_db_startup.dart	2	2	100%
lib\services\local_db\local_db_startup_io.dart	3	3	100%
lib\services\local_db\local_db_startup_stub.dart	1	1	100%
lib\services\local_db\local_db_startup_web.dart	1	1	100%
lib\services\local_db\offline_sync_row.dart	1	1	100%
lib\services\local_db\offline_sync_service.dart	202	264	76.5%

5.10.1 Offline Data Capture

Offline Data Capture: Verify supported features store data locally offline; entries persist across restarts; queued for sync.

Total Test Cases: TC-OFF-01- TC-OFF-03, TC-OFF-05, TC-OFF-06

Requirements Coverage: REQ-SB-101, REQ-SB-102, REQ-SB-104, REQ-SB-111

Prerequisites for this test: User authenticated Device offline Local storage enabled	Software Versions: Application: CareConnect Software Version: Operating System: Linux / macOS / Windows
	Priority: High
Requirements validated: REQ-SB-101	
TEST EXECUTOR: Yismaw Tilaye	

STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	PASS/FAIL	Defects Found
TC-OFF-01: Create offline entry persists locally					
1.	Log Mood while Internet is not available	Entry saved in local drift database; UI marks ‘Mood saved in offline queue’; no network errors shown.	Entry saved in local drift database; UI marks ‘Mood saved in offline queue’; no network errors shown.	Pass	
TC-OFF-02: Restart app while offline retains queued items					

1.	Restart app while offline	Previously entered offline data remains stored locally; data is still present in offline queue	Previously entered offline data remains stored locally; data is still present in offline queue	Pass	None
2.	Log mood entry	Entry saved in local drift database; UI marks ‘Mood saved in offline queue’; no network errors shown.	Entry saved in local drift database; UI marks ‘Mood saved in offline queue’; no network errors shown.	Pass	None
3.	Verify entry in queue	Entry remains visible and queued for synchronization	Entry remained visible and queued for synchronization	Pass	None
TC-OFF-03: Multiple offline entries queued for sync					
1.	Log multiple mood entries while offline	All entries are stored locally and added to offline queue	All entries are stored locally and added to offline queue	Pass	None
2.	Navigate between screens and return to mood section	All entries remain available and visible in the application	All entries remained available and visible in the application	Pass	None
3.	Verify offline queue	All entries are present and added to offline queue	All entries are present and added to offline queue	Pass	None
TC-OFF-05: Restricted modules blocked offline					
1.	Attempt to access restricted module while Internet is not available	Restricted modules are disabled or unavailable	Restricted modules were disabled or unavailable	Pass	None
2.	Attempt to perform restricted action (e.g., video	Action is blocked and cannot be	Action is blocked and cannot be	Pass	None

	call or EVV)	executed	executed		
3.	Return to dashboard and access offline-supported features	Offline-supported features remain accessible and functional	Offline-supported features remained accessible and functional	Pass	None
TC-OFF-06: Connectivity-based offline behavior enforced					
1.	Log mood entry while Internet is not available	Entry saved in local drift database; UI marks ‘Mood saved in offline queue’; no network errors shown	Entry saved in local drift database; UI marks ‘Mood saved in offline queue’; no network errors shown	Pass	None
2.	Restore Internet connection	Application detects connectivity and prepares for synchronization	Application detected connectivity and prepared for synchronization	Pass	None
3.	Wait for synchronization to complete	Offline entries are synced to backend and removed from queue	Offline entries were synced to backend and removed from queue	Pass	None

5.10.2 Synchronization on Reconnect

Establishes the critical link between CareConnect’s offline-first local storage and the centralized cloud backend

Total Test Cases: 2 (TC-SYNC-01, TC-SYNC-02)

Requirements Coverage: REQ-SB-104, REQ-SB-105, REQ-SB-106, REQ-SB-107, REQ-SB-110, REQ-SB-112, REQ-SB-113, REQ-SB-114

Prerequisites for this test:	Software Versions:

User authenticated Device starting offline with populated queue Local storage enabled and encrypted	Application: CareConnect Software Version: Operating System: Linux/Mac/Windows Priority: High
Requirements validated: REQ-SB-104, REQ-SB-105, REQ-SB-106, REQ-SB-107, REQ-SB-110, REQ-SB-112, REQ-SB-113, REQ-SB-114	
TEST EXECUTOR: Eduardo Estrada	

STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	PASS/FAIL	Defects Found
TC-SYNC-01: Auto-sync triggers on reconnect					
1	Restore network (Wi-Fi/Cellular).	Application detects connection; background sync starts (30s interval).	Sync started; progress UI visible; queue cleared.	Pass	
TC-SYNC-02: Duplicate prevention & conflict strategy					
1.	Sync an offline edit where a server-side record already exists	No duplicate records; conflict resolved per strategy; final state logged.	Idempotency enforced; records matched backend.	Pass	None

5.10.3 Synchronization Failure Handling

Synchronization Failure Handling: Verify failed or interrupted synchronization attempts preserve queued data, prevent duplicates, retry safely, and keep users informed without exposing sensitive data.

Total Test Cases: 5 (TC-SYNC-03, TC-SYNC-04, TC-SYNC-05, TC-SYNC-06, TC-SYNC-07)

Requirements Coverage: REQ-SB-105, REQ-SB-106, REQ-SB-110, REQ-SB-112, REQ-SB-113

Prerequisites for this test: User is authenticated and logged into the CareConnect application Telemetry is enabled in application settings (user has not opted out) Device has no internet connection or backend is unreachable (offline state simulated) Offline queue functionality is enabled and initialized Application has permission to store local data (for queue persistence if applicable)	Software Versions: Application: CareConnect Software Version: Operating System: Linux/Mac/Windows Priority: High
Requirements validated: REQ-SB-105, REQ-SB-106, REQ-SB-110, REQ-SB-112, REQ-SB-113	
TEST EXECUTOR: Joseph Wojcik	

STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	PASS/FAIL	Defects Found
TC-SYNC-03: Sync failure notification and safe retry					
1.	Reconnect device while backend is unavailable	Synchronization attempt starts, fails safely, and preserves queued items	Synchronization attempted, failed safely, and preserved queued items	Pass	None
TC-SYNC-04: Network interruption during sync preserves queue					
1.	Start sync, then disable network before	Successfully synced items are removed; remaining items	Successfully synced items were removed; remaining	Pass	None

	completion	stay queued	items stayed queued		
2.	Restore Internet connection	Application detects reconnect and resumes synchronization	Application detected reconnect and resumed synchronization	Pass	None
3.	Verify queue after sync resumes	No data loss or duplicate records are present	No data loss or duplicate records were present	Pass	None
TC-SYNC-05: Partial sync failure leaves remaining items queued					
1.	Queue 3 or more offline entries, then force one backend failure	Successful items sync; failed item remains queued	Successful items synced; failed item remained queued	Pass	None
2.	Inspect server records after first sync attempt	No duplicate records are created on the backend	No duplicate records were created on the backend	Pass	None
3.	Retry sync after backend is restored	Remaining queued item syncs successfully and queue clears	Remaining queued item synced successfully and queue cleared	Pass	None
TC-SYNC-06: Pause sync while online queues locally					
1.	Enable sync pause while Internet is available	Synchronization is paused without disrupting normal app use	Synchronization was paused without disrupting normal app use	Pass	None
2.	Create a supported entry while sync pause is enabled	Entry is stored locally and marked Pending Sync	Entry was stored locally and marked Pending Sync	Pass	None
3.	Disable sync pause	Synchronization begins	Synchronization began automatically	Pass	None

		automatically			
TC-SYNC-07: Durable queue replay across reconnect cycles					
1.	Create multiple offline entries, then restore connectivity	Queue begins processing from the first pending item	Queue began processing from the first pending item	Pass	None
2.	Interrupt network mid-sync, then reconnect again	Queue resumes from the last confirmed successful item	Queue resumed from the last confirmed successful item	Pass	None
3.	Verify final queue and backend state	All queued items replay successfully with no data loss	All queued items replayed successfully with no data loss	Pass	None

5.10.4 Infrastructure Deployment Validation

Infrastructure Deployment Validation: Verify the automated provisioning of the CareConnect stack via cdeploy_cloudformation.sh. This includes validating the sequential creation of the Networking, Data, Platform, and Service stacks, as well as ECR image residency and ALB reachability. Tests were performed manually while verifying resources in the CLI and physically in AWS.

Total Test Cases: 6 (TC-INF-01 through TC-INF-06)

Requirements Coverage: REQ-SB-201, REQ-SB-202, REQ-SB-203, REQ-SB-204, REQ-SB-205, REQ-SB-206, REQ-SB-207

Prerequisites for this test: User is logged into the AWS CLI GitHub secrets configured with AWS Role needed for CI/CD	Software Versions: Application: CareConnect Software Version: Operating System: Linux/Mac/Windows
	Priority: High

Requirements validated: REQ-SB-201, REQ-SB-202, REQ-SB-203, REQ-SB-204, REQ-SB-205, REQ-SB-206, REQ-SB-207	
TEST EXECUTOR: Gary Jurado	

STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	PASS/FAIL	Defects Found
TC-INF-01 Provision infrastructure via CloudFormation					
1.	Run stack create via pipeline.	Stack succeeds; resources match template.	Stack succeeds; resources match template	Pass	None
2.	Validate resources	tags and outputs correct.	tags and outputs correct.	Pass	None
TC-INF-02 Templates version-controlled and reproducible					
1.	Deploy from previous SHA.	Previous SHA environment operation.	Both envs identical; drift minimal/none; outputs consistent.	Pass	Manual Test performed
2.	Redeploy same SHA to new env	outputs consistent.	outputs consistent.	Pass	Manual Test
TC-INF-03 CI/CD stack deployment update path works					
1.	Change parameter	Update succeeds	Update succeeds	Pass	None

2.	Run pipeline update.	no manual steps; change recorded.	no manual steps; change recorded.	Pass	None
TC-INF-04 Terraform configs removed/unused					
1.	Verify no Terraform executed in pipeline.	Pipeline uses only CloudFormation	Pipeline uses only CloudFormation	Pass	None
2.	Run pipeline.	no Terraform dependencies.	no Terraform dependencies.	Pass	None
TC-INF-05 Rollback on deployment failure					
1.	Trigger failure.	Rollback completes;			Test Not performed/implementation incomplete
2.	Observe rollback.	environment returns to last good state; incident logged.			
TC-INF-06 Reproducible environments across regions/accounts					
1.	Deploy stack in env A	Resource parity achieved;	Resource parity achieved; environment-specific values	Pass	None

			parameterized.		
2.	Deploy in env B.	Resource parity achieved;	Resource parity achieved; environment-specific values parameterized.	Pass	None
TC-INF-07 CI/CD Template validation gates deployments					
1.	Submit invalid template	Validation fails fast; deployment prevented	Validation fails fast; deployment prevented	Pass	None
2.	Observe pipeline	error captured	error captured	Pass	None

5.10.5 Observability & Analytics Validation

Observability & Analytics Validation: Verify logs, metrics, traces, and anonymous analytics events are captured correctly, privacy controls are enforced, and telemetry pipelines remain observable without PHI/PII exposure.

Total Test Cases: 5 (TC-OBS-01, TC-OBS-02, TC-OBS-03, TC-SEC-01, TC-ANA-01)

Requirements Coverage: REQ-SB-301, REQ-SB-302, REQ-SB-303, REQ-SB-307, REQ-SB-401

Prerequisites for this test: User is authenticated and logged into the CareConnect application Telemetry is enabled in application settings (user has not opted out) Device was previously offline with queued telemetry events stored locally Device has re-established internet connectivity Backend telemetry service is reachable and capable of processing queued events	Software Versions: Application: CareConnect Software Version: Operating System: Priority: High

Requirements validated: REQ-SB-301, REQ-SB-302, REQ-SB-303, REQ-SB-307, REQ-SB-401	
TEST EXECUTOR: Joseph Wojcik	

STEP	TEST STEP/INPUT	EXPECTED RESULTS	ACTUAL RESULTS	PASS/FAIL	Defects Found
TC-OBS-01: Centralized logging includes sync events					
1.	Trigger offline-to-online sync and inspect centralized logs	Sync events appear with timestamps and correlation identifiers	Sync events appeared with timestamps and correlation identifiers	Pass	None
TC-OBS-02: Performance metrics are visible					
1.	Generate normal application traffic	Latency, throughput, and error metrics are captured	Latency, throughput, and error metrics were captured	Pass	None
2.	Review observability dashboard or metrics store	Metrics are visible and aligned with generated activity	Metrics were visible and aligned with generated activity	Pass	None
3.	Verify no sensitive values appear in metric labels	Metrics contain operational values only, not PHI/PII	Metrics contained operational values only, not PHI/PII	Pass	None

TC-OBS-03: Distributed tracing spans end-to-end					
1.	Execute a sync workflow with tracing enabled	Client, API, and database spans share a consistent trace ID	Client, API, and database spans shared a consistent trace ID	Pass	None
2.	Inspect trace details in the observability tool	Trace shows ordered span relationships across components	Trace showed ordered span relationships across components	Pass	None
3.	Review trace payload for privacy compliance	Trace metadata excludes PHI/PII	Trace metadata excluded PHI/PII	Pass	None
TC-SEC-01: PHI/PII excluded from logs and traces					
1.	Generate sync events and controlled application errors	Logs and traces capture operational details without PHI/PII	Logs and traces captured operational details without PHI/PII	Pass	None
2.	Search observability records for known sensitive test strings	Sensitive values are not present in stored telemetry	Sensitive values were not present in stored telemetry	Pass	None
3.	Review structured log fields	Only approved operational fields are recorded	Only approved operational fields were recorded	Pass	None
TC-ANA-01: Anonymous usage events are collected					
1.	Use key application	Anonymous usage events are emitted	Anonymous usage events were emitted	Pass	None

	features with analytics enabled	successfully	successfully		
2.	Inspect analytics stream or storage destination	Events use anonymous identifiers and approved schema fields	Events used anonymous identifiers and approved schema fields	Pass	None
3.	Verify analytics payload for privacy compliance	No PHI/PII appears in analytics events	No PHI/PII appeared in analytics events	Pass	None

5.11 Test Design Specifications

The Test Design Specifications define how system requirements are transformed into logical groupings of test conditions and scenarios. For CareConnect Milestone 2, test design is driven by Business Need Statements (BNS), functional and non-functional requirements defined in the SRS, and architectural decisions documented in the TDD.

Test designs are organized into test suites (TS-01 through TS-13) as defined in Section 5.9. Each test suite represents a cohesive validation objective, such as offline persistence, synchronization reliability, infrastructure deployment, observability, or targeted regression of inherited MVP workflows.

For each test design, the following elements are specified:

- Covered requirement identifiers (REQ-SB-###)
- Associated acceptance criteria (AC-###)
- Test type (functional, integration, system, regression, security, or non-functional)
- Expected system behavior under normal and failure conditions

This design-level structure ensures traceability from requirements to execution while avoiding duplication across individual test cases.

5.12 Test Case Specifications

Test Case Specifications define the detailed conditions, inputs, actions, and expected results used to verify each test design. High-priority and representative test cases are documented in Section

5.10, with complete test case sets maintained in the project repository as part of the Test Case Specification (TCS) artifact.

Each test case includes, at minimum:

- Unique Test Case Identifier (TC-###)
- Associated Test Suite (TS-##)
- Mapped Requirement(s) and Acceptance Criteria
- Preconditions and test data assumptions
- High-level execution steps
- Expected results and pass/fail conditions

Test cases are designed to validate both positive and negative scenarios, including offline operation, synchronization failures, retry behavior, security and privacy enforcement, and regression stability for inherited MVP features.

5.13 Test Procedure Specifications

Test Procedure Specifications define the ordered steps and execution guidance required to run test cases in a consistent and repeatable manner. For CareConnect Milestone 2, test procedures are documented at a high level within this STP and executed using detailed checklists and scripts maintained as Test Procedure Specifications (TPS) artifacts. Test procedures are designed to ensure consistent execution order, controlled environment conditions, and repeatable results across multiple test runs and test environments.

Test procedures include:

- Environment preparation steps (account setup, feature flags, network state)
- Execution sequencing for related test cases within a suite
- Required tools and configurations (e.g., network toggling, CloudFormation pipelines, CloudWatch inspection)
- Evidence collection requirements (logs, screenshots, pipeline outputs)
- Cleanup and reset steps to ensure repeatability

Procedures are executed manually for complex offline and UX-dependent scenarios and via automation where appropriate for infrastructure validation and telemetry inspection. Execution results are recorded in the Test Execution Log (TEL) and summarized in the Test Summary Report (TSR) as defined in Section 9.

6. Pass/Fail Criteria

- A test item is considered **PASS** when all the following criteria are satisfied:
    - **Defect Resolution:** 100% of Critical (Severity 1) and High (Severity 2) defects associated with the test item are resolved and verified.
    - **Execution Rate:** Test case execution success rate is at least 95%, with all remaining failures analyzed and formally waived by the team.
    - **Data Integrity:** Offline/Sync mechanics successfully preserve data state without loss or unauthorized duplication during network transition simulations.
    - **Privacy Compliance:** Payload inspection confirms zero (0) instances of PHI/PII leakage in captured logs, metrics, traces, or analytics strings.
    - **Infrastructure Stability:** CloudFormation templates deploy and update successfully, with all validation gates passing and automated rollback confirmed in a controlled failure scenario.

A test item is considered **FAIL** when any of the following occur:

- **Unresolved Risks:** A Critical or High defect remains open or unmitigated against the test item.
- **Data Corruption:** Data loss, sync corruption, or duplicate record creation is observed during offline-to-online transition scenarios.
- **Security Breach:** Any PHI/PII is detected within the operational telemetry or analytics capture packets.
- **Deployment Failure:** BNS 6 CloudFormation scripts fail to reach a "CREATE_COMPLETE" or "UPDATE_COMPLETE" state, or fail to roll back to a stable known-good configuration.

7. Suspension Criteria and Resumption Requirements

7.1 Suspension Criteria

- Environment instability prevents consistent test execution (e.g., repeated outages).
- Critical defect blocks core flows (authentication, offline persistence, or synchronization).
- Data corruption or unsafe privacy exposure detected (PHI/PII leakage).
- Infrastructure deployments repeatedly fail due to external constraints (permissions, quotas).

7.2 Resumption Requirements

- Blocking defects mitigated and verified in a new build.

- Environment restored and validated (smoke suite passes).
- If privacy leakage detected: logging/analytics fixed, and evidence confirms remediation.
- Infrastructure permission/quota issues resolved; CloudFormation validation step passes.

8. Test Deliverables

- Software Test Plan (this document) – STP-02072026-B001.
- Requirements Traceability Matrix.
- Test Case Specification (included in Section 5.10).
- Test Procedures / Scripts (manual checklists and automated scripts).
- Test Execution Log **(**run records, build numbers, evidence links).
- Defect/Incident Reports (GitHub Issues).
- Infrastructure Deployment Evidence (pipeline logs, CloudFormation events).
- Telemetry/Analytics Validation Evidence (payload inspection outputs)
- Test Summary Report (exit assessment and sign-off recommendation)

Note: Specific document identification numbers for the deliverables listed above have not been determined yet and will be updated in future revisions.

9. Testing Tasks

Table 9.1

Testing Tasks

Task ID	Task	Owner	Inputs	Outputs
TASK-01	Finalize STP and baseline RTM	Test Lead	SRS/TDD, prior STP	Approved STP, RTM baseline
TASK-02	Environment readiness and smoke validation	QA Analysts + DevOps	Deployments, configs	Smoke results, readiness checklist
TASK-03	Design/maintain test cases & procedures	QA Analysts	RTM, requirements	TCS/TPS updates
TASK-04	Execute TS-01..TS-05 (Team B	QA Analysts	Test build, test data	Execution logs, evidence artifacts

	enhancements)			
TASK-05	Execute TS-06..TS-13 (targeted regression)	QA Analysts	Seed data, accounts	Regression results, defect entries
TASK-06	Defect triage & verification	QA + Developers	Defect reports	Resolved defects, retest records
TASK-07	Infrastructure deployment validation in CI/CD	DevOps + QA	Templates, pipeline	IDE artifacts, pass/fail gates
TASK-08	Telemetry and analytics privacy validation	QA + Security reviewer	Payload inspection scripts	TAE artifacts, compliance notes
TASK-09	Reporting and final sign-off recommendation	Test Lead	All results and metrics	TSR and approvals section completion

10. Environmental Needs

10.1 Hardware

- Developer/QA workstations capable of running local services and test tools (8+ GB RAM recommended).
- Test devices or emulators as applicable to CareConnect client platforms (mobile and/or web).
- Reliable network toggling capability for offline/online transition testing (airplane mode / firewall / proxy).

10.2 Software

- CareConnect client application build for Milestone 2 (test/staging)
- Backend services deployed locally or in cloud test environment (APIs required by suites)
- AWS account(s) for CloudFormation deployments and CloudWatch inspection
- Supported OS and browser baseline for web testing (latest stable versions in test lab)

10.3 Test Tools

- Issue tracking: GitHub Issues
- **API testing:** Postman / curl scripts
- **CI/CD:** GitHub Actions or equivalent pipeline runner
- **AWS:** CloudFormation console/CLI, CloudWatch dashboards/log insights, IAM role management
- **Optional automation frameworks (as applicable):** unit/integration test runners used by project (e.g., Jest/JUnit)

11. Responsibilities

Table 11.1

Responsibilities

Role	Assigned To	Responsibilities
Test Lead / Test Manager	Team B (rotating lead)	Owns STP, RTM integrity, readiness reviews, reporting, and sign-off recommendation.
QA Analysts / Testers	Eduardo Estrada, Gary Jurado, Yismaw Tilaye, Joseph Wojcik, Chastity Sapp	Execute test suites, collect evidence, document results, and log defects.
Developers	CareConnect Dev Team	Fix defects, builds, support test instrumentation and reproductions.
DevOps / Cloud Engineer	Team B DevOps owner	Maintain CloudFormation templates/pipelines; support deployment validation and rollback drills.
Security/Privacy Reviewer	Designated reviewer	Validate PHI/PII exclusion and RBAC controls; review telemetry/analytics evidence.
Stakeholders / Instructors	Course stakeholders	Review deliverables and approve final submission.

12. Staffing and Training Needs

Staffing assumes Team B members fulfill QA, DevOps, and reporting roles. Training needs include:

- AWS CloudFormation and CloudWatch fundamentals for validation and evidence collection.
- Offline testing techniques (network toggling, state persistence verification, conflict simulation).
- Privacy/security validation practices (redaction, payload inspection, evidence handling).
- Defect reporting standards (repro steps, severity, traceability to requirement and test case IDs).

13. Schedule

Table 13.1

Schedule

Milestone	Date Range (2026)	Exit Criteria
STP/RTM Baseline & Approval	2026-02-08 to 2026-02-14	STP approved; RTM baselined; environments requested
Environment Setup & Smoke	2026-02-15 to 2026-02-21	Smoke suite pass; accounts/test data ready
Team B Enhancement Execution (TS-01..TS-05)	2026-02-22 to 2026-03-07	All enhancement suites executed; defects logged
Targeted Regression Execution (TS-06..TS-13)	2026-03-08 to 2026-03-14	Regression suite executed; critical paths validated
Defect Fix/Retest & Final Validation	2026-03-15 to 2026-03-21	No open Sev1/Sev2; pass rate ≥95%
Test Summary Report & Sign-off	2026-03-22 to 2026-03-25	TSR complete; approvals captured

14. Risks and Contingencies

Risks are assessed for likelihood/impact and include mitigations and contingencies.

Table 14.1

Risks and Contingencies

Risk ID	Risk	Like lihood	Impact	Mitigation	Contingency
RISK-01	Offline-to-online conflict resolution produces incorrect final state	Medium	High	Execute controlled conflict simulations (TC-SYNC-02) and expand coverage for edge cases.	Pause release; implement/adjust conflict strategy; retest affected suites.
RISK-02	Duplicate records created during retries or partial failures	Medium	High	Validate idempotency/duplicate prevention (TC-SYNC-02/04).	Implement server-side dedupe; run data cleanup in test env; revalidate.
RISK-03	CloudFormation rollback fails leaving environment unstable	Low-Medium	High	Perform rollback drills (TC-INF-05) and parameterize safely; validation gates (TC-INF-07).	Restore from last known good stack or redeploy fresh environment.
RISK-04	PHI/PII leakage in logs/traces/analytics	Low	Very High	Payload inspection and redaction checks (TC-SEC-01/03; TC-ANA-01).	Suspend testing; remediate logging/analytics; rotate credentials; re-run privacy validation.
RISK-05	Backend services partially simulated causing false positives/negatives	Medium	Medium	Document stubs; prefer integrated staging when available; add environment notes to TEL.	Re-run affected cases in staging; adjust expected results with stakeholder approval.
RISK-06	Unstable connectivity in test environment reduces repeatability	Medium	Medium	Use controlled network toggling tools and defined procedures (TS-03).	Reschedule runs; switch to local harness for determinism.

15. Approvals

Approval indicates review and acceptance of this test plan for Milestone 2 execution.

Name	Role	Signature	Date
Chastity Sapp	Test Lead	Chastity Sapp	2/13/2026
Joseph Wojcik	DevOps/Cloud Owner	Joseph Wojcik	2/13/2026

Gary Jurado	Security/Privacy Reviewer	Gary Jurado	2/13/2026
Yismaw Tilaye	Tester/Developer	Yismaw Tilaye	2/13/2026
Eduardo Estrada	Tester/Developer	Eduardo Estrada	2/12/2026
Dr. Mir Assadullah	Stakeholder/Instructor		

Appendix A Testing Results and Coverage Report Supplement

Testing Objectives

The objectives of the testing effort were:

1. Verify correctness of application logic across frontend and backend layers
2. Validate system behavior under expected operational conditions
3. Confirm proper enforcement of security rules and access restrictions
4. Ensure telemetry monitoring infrastructure operates correctly
5. Validate local database behavior and offline synchronization mechanisms
6. Measure code coverage for critical components
7. Establish a repeatable automated testing framework for continued development

Test Environment

Testing was conducted using the following development and testing environments.

Component	Technology
Frontend framework	Flutter
Frontend language	Dart
Backend framework	Spring Boot
Backend language	Java
Frontend testing framework	Flutter Test
Backend testing framework	JUnit 5
HTTP mocking	http/testing MockClient
Backend web testing	Spring MockMvc
Mocking framework	Mockito
Frontend coverage tool	LCOV
Backend coverage tool	JaCoCo

Continuous integration GitHub Actions

Testing Methodology

Testing followed a layered validation approach to ensure coverage of multiple system components.

Unit Testing

Unit tests were written to verify individual functions, classes, and services in isolation. These tests validated core logic including:

- telemetry event generation
- telemetry validation guardrails
- database encryption key lifecycle
- offline synchronization behavior
- telemetry service persistence
- telemetry toggle logic

Unit tests allow fast execution and help detect logic errors during development.

Integration-Style Testing

Integration-style tests validated interaction between system components such as:

- telemetry controller endpoints and services
- database persistence layers
- security configuration rules
- offline database operations

These tests simulate real application flows while still executing in a controlled test environment.

Security Testing

Security configuration tests were implemented to verify that API endpoints enforce correct authentication and authorization behavior.

Testing validated:

- anonymous access restrictions
- admin-only endpoint access
- proper handling of unauthorized users
- enforcement of API security rules

Coverage Analysis

Coverage analysis tools were used to measure the proportion of code executed during testing.

Coverage analysis helps identify:

- untested logic paths
- insufficient test coverage
- potential reliability risks

The two primary tools used were:

Tool	Purpose
LCOV	Flutter code coverage
JaCoCo	Java backend coverage

Frontend Testing

Frontend testing focused on the Flutter application components responsible for telemetry monitoring, configuration management, and application state behavior.

Key files tested include:

File	Purpose
telemetry.dart	Telemetry orchestration and event submission
telemetry_guardrails.dart	Telemetry event validation
telemetry_settings.dart	User telemetry preference management

Telemetry System Testing

The telemetry subsystem collects runtime event information and transmits telemetry data to backend services for monitoring and analysis.

Telemetry functionality validated

Tests confirmed correct behavior for the following scenarios:

- telemetry events are not sent when users opt out
- telemetry events are sent when enabled
- backend toggle state is respected
- backend toggle updates function correctly
- invalid telemetry events are rejected
- network failures do not crash telemetry logic
- telemetry payloads contain required identifiers

The telemetry event payload includes:

- eventName
- sessionId
- traceId
- spanId
- device information
- event details

Telemetry coverage results

File	Coverage
telemetry.dart	91%
telemetry_guardrails.dart	100%
telemetry_settings.dart	100%

These results indicate that nearly all telemetry validation and preference logic paths are exercised during testing.

Backend Testing

Backend testing focused on the telemetry subsystem implemented in the Spring Boot service architecture.

Key backend test classes include:

Test Class	Purpose
DevTelemetryControllerTest	Telemetry controller endpoint validation

TelemetryServiceTest	Telemetry persistence logic
TelemetryToggleServiceTest	Telemetry enable/disable behavior
TelemetrySecurityConfigTest	API security rules
TelemetryEventTest	Model lifecycle validation

Telemetry Controller Testing

Controller tests verified the behavior of the development telemetry endpoints.

Validated endpoints include:

Endpoint	Function
POST /v1/api/dev/telemetry	Submit telemetry event
GET /v1/api/dev/telemetry/recent	Retrieve recent telemetry events
GET /v1/api/dev/telemetry/enabled	Check telemetry enabled state
PUT /v1/api/dev/telemetry/enabled	Enable or disable telemetry

Controller tests validated:

- correct response codes
- proper event handling
- correct toggle behavior
- null-safe payload handling

Telemetry Service Testing

TelemetryService tests verified core service logic.

Validated behaviors include:

- recording telemetry events when enabled
- ignoring events when telemetry is disabled
- retrieving recent telemetry records
- creating anonymous telemetry events

Edge cases such as null results and limit clamping were also validated.

Telemetry Toggle Testing

The TelemetryToggleService controls whether telemetry persistence is enabled.

Tests validated:

- default enable state
- toggle behavior from enabled to disabled
- toggle behavior from disabled to enabled
- consistent state persistence across repeated operations

Security Configuration Testing

Security configuration tests verified access control behavior for telemetry endpoints.

Validated rules include:

- POST telemetry endpoint accessible without authentication for development telemetry ingestion
- administrative endpoints require authentication
- non-admin users cannot access protected endpoints
- admin users can access telemetry configuration endpoints

This ensures telemetry administrative functionality remains secure.

7. Database Migration Testing

A new migration was added to extend telemetry capabilities.

Migration

V38__add_session_id_to_telemetry_events.sql

This migration adds a session_id column to the telemetry events table.

Purpose

The new column allows telemetry events generated within the same application session to be correlated more effectively.

This improves observability and allows event chains to be reconstructed for debugging and analytics.

Local Database Testing

The CareConnect application includes an encrypted local database to support offline data storage and synchronization.

The following database components were tested:

Component	Purpose
AppDatabase	Local SQLite database wrapper
DbEncryptionService	SQLCipher encryption key management
OfflineSyncService	Offline request queue
Local database startup initialization	Database initialization logic

Offline Synchronization Testing

Offline synchronization tests validated the behavior of queued network operations when connectivity is unavailable.

Test scenarios included:

- adding requests to the offline queue
- deduplicating identical queued requests
- deleting queued operations
- synchronizing queued operations when connectivity returns
- marking failed operations correctly
- handling invalid URLs
- handling HTTP error responses

Encryption Key Lifecycle Testing

Encryption key lifecycle tests verified:

- generation of new encryption keys
- reuse of existing encryption keys
- detection of existing encryption keys
- deletion of encryption keys
- safe escaping of encryption keys for SQLCipher usage

Local Database Coverage

Local database test coverage achieved the following results:

File	Coverage
app_database.dart	95%
db_encryption_service.dart	100%
offline_sync_service.dart	89.5%
local_db_startup.dart	100%
local_db_startup_io.dart	100%
offline_sync_row.dart	100%

Overall coverage for the local database subsystem exceeded 90%.

Continuous Integration Testing

Automated testing was also performed through the project's GitHub Actions pipeline.

The pipeline includes:

- static analysis tools
- security scanning tools
- dependency scanning
- code quality analysis

These tools identify potential security risks, code smells, and configuration issues.

Limitations

Although significant testing coverage was achieved for critical components, several limitations remain.

First, the repository contains many modules unrelated to the telemetry or database subsystems that were not included in this milestone's testing scope.

Second, full system integration tests involving a live database environment were not implemented because they would require additional infrastructure configuration.

Finally, UI-level end-to-end tests were outside the scope of this milestone.

Future Improvements

Future testing work could include:

- automated end-to-end testing of user workflows
- integration tests using embedded database environments
- expanded coverage for remaining backend modules
- performance testing for telemetry ingestion endpoints
- load testing for telemetry event throughput

These improvements would further strengthen the reliability and scalability of the CareConnect system.

The testing effort conducted during Milestone 4 successfully validated the major functional components of the CareConnect system. Automated tests were implemented across frontend and backend layers, security rules were verified, and coverage metrics demonstrate strong validation of critical subsystems.

The telemetry subsystem achieved particularly strong coverage and validation, while the local database and offline synchronization logic also reached high coverage levels.

The project now includes an automated testing foundation that will support continued development and future feature expansion.

Software Test Plan

CareConnect

University of Maryland Global Campus

SWEN 670 - Software Engineering Capstone

Dr. Mir Assadullah

February 7th, 2026

Contributors: Colyn Mahoney, Zechariah Hillman, Obeng Buo, Henry Stewart, Ja'Lisa Hawkins

Revision History

Date	Version	Description	Author
02/07/2026	1.0	Initial Release	Team C
03/27/2026	2.0	Includes Test Report	Team C

Sign-off Sheet

Role	Name	Signature	Date
Project Manager	Colyn Mahoney		
Team Lead	Colyn Mahoney		
Lead Tester			
Client	Dr. Mir Assadullah		

Table of Contents

1. Test Report Summary

CareConnect is a mobile-first, cross-platform software application designed to enable collaboration between caregivers and their patients. The system consists of a Flutter-based mobile and web interface and a backend composed of multiple microservices. CareConnect provides features such as real-time communication, symptom tracking, wearable device integration, and care-plan task management to enhance the caregiver and patient experience. The backend architecture is designed with AWS scale-to-zero capabilities, allowing compute resources to be allocated dynamically based on demand, helping reduce infrastructure costs. The system is developed in compliance with healthcare regulations including HIPAA and GDPR.

This test report summarizes the testing activities performed by Team C to validate that the system meets all functional and non-functional requirements associated with the current development effort. This phase introduced changes to the production hosting environment, a redesigned role-based access control (RBAC) system, and new testing standards through the implementation of additional automated unit tests and mock data frameworks. As part of this effort, testing included full-system validation to ensure that these changes did not negatively impact existing functionality.

Testing activities included a combination of manual and automated approaches. Manual testing consisted of “act-like-a-user” smoke testing in the deployed environment to verify end-to-end functionality. Automated testing included backend unit testing using JUnit and frontend testing using Flutter testing frameworks, covering both the existing MVP functionality and newly introduced changes. Additionally, User Acceptance Testing (UAT) was conducted to validate that system behavior and performance remain consistent from the perspective of end users.

The objective of this testing effort was to ensure that CareConnect continues to operate at or above previously established levels following the implementation of these changes. Based on the results of testing, the system maintains expected functionality, performance, and user experience, with no significant regressions identified within the scope of this development phase.

2. Introduction

2.1 Objectives

A system test plan for the CareConnect application should support the following objectives:

1. To detail the activities required to prepare for and conduct the system test.
2. To communicate to all responsible parties the tasks that they are to perform and the schedule to be followed in performing the tasks.
3. To define the sources of the information used to prepare the plan.

4. To define the test tools and environment needed to conduct the system test.

The purpose of this test plan is to detail a plan for the testing of the new requirements for development team C. This document will also double as the testing report. As test cases are executed on, the results will be reported in this test plan. At the conclusion of this phase of the CareConnect project. The goal is to demonstrate CareConnect meets all its new functional and non-functional requirements in addition to its old requirements with no loss of functionality as detailed in the old project documentation and new project documentation respectively.

2.2 Background

This project is a continuation of work done on the CareConnect application by several cohorts of the SWEN 670 course. Team C has inherited an MVP and has three distinct requirements; implement a mock data framework for unit testing and implement new unit tests for the prior cohorts work, redesign and refactor roll-based access control in the existing codebase, and migrate the backend to a new production host.

2.3 Scope

This report covers both functional and non-functional testing performed for the current CareConnect release.

Functional testing focused on validating that:

- Role-based access control (RBAC) changes did not alter the user experience
- Migration to the new production environment did not impact system functionality

Non-functional testing focused on validating:

- System performance remained consistent with previous benchmarks
- Security requirements, including HIPAA and GDPR compliance, were maintained

Additionally, regression testing was conducted to ensure that previously implemented features continue to function as expected without degradation.

2.4 Project Documents

The following project documents were referenced during testing to validate system requirements and expected behavior:

Table 1 Documents

Document	Version	Date
Project Plan	1.0	01/24/2026
Software Requirements Document	1.0	01/24/2026
Technical Design Document	1.0	02/07/2026
Software Test Plan	1.0	02/07/2026
Project Plan (Last Cohort)	2.0	09/01/2025
Software Requirements Specification (Last Cohort)	2.0	09/01/2025
Technical Design Document (Last Cohort)	2.0	09/01/2025
Test Report (Last Cohort)	2.0	09/01/2025

2.5 References

- Requirements document provided by stakeholder mentors.
- Software Requirements Specification document from Summer/Fall 2025 cohort https://umgc-cappms.azurewebsites.net/download/d342ee6d-bbe6-49ee-aa03-75a359b26bf8----SWEN670_CareConnect_Fall2025_SRS.pdf
- AWS Faregate https://aws.amazon.com/fargate/
- AWS App Runner https://aws.amazon.com/apprunner/
- An introduction to unit testing https://docs.flutter.dev/cookbook/testing/unit/introduction
- JUnit User Guide https://docs.junit.org/6.0.2/overview.html
- Technical Design Document from Summer/Fall 2025 cohort https://umgc-cappms.azurewebsites.net/download/1c45811b-6d19-415a-8b17-02205cf5c942----SWEN670_CareConnect_TechnicalDesignDocument_v3.0.pdf
- IEEE Standard for Software Test Documentation rev. 829-1998

3. Test Items

The following components of the CareConnect system were included in testing activities:

- Backend application services, including API endpoints and business logic
- Frontend application components and user interface functionality
- Role-Based Access Control (RBAC) implementation across backend and frontend
- Deployment and hosting infrastructure in the new production environment (AWS)
- Unit testing frameworks and mock data implementations

Testing was conducted at the unit, integration, and system levels to ensure that all components function correctly both independently and as part of the complete system.

For acceptance criteria please see Team C's SRS document in the requirements section.

3.1 Requirements Traceability Matrices

A requirements traceability matrix is used to clearly communicate which requirements are being tested by which test case. Due to the nature of the requirements in this development effort, this software test plan will include two requirements traceability matrices. The first will include functional requirements specific to this development effort. The second will be the requirements traceability matrix from the previous cohort. Team C is including this because the requirements of this development effort require changes to the underlying infrastructure of the application creating the need for a full careful retest of the entire system.

3.1.1 Team C Requirements Traceability Matrix

Requirement ID	Requirement Description	Test Case ID(s)	Status
REQ-1	Migrate production environment to Faregate	TC-1,TC-1.1	Pass
REQ-1.1	Build out docker file for spring boot backend	TC-1.2	Pass
REQ-1.2	Deploy spring boot backend to Faregate	TC-1.3	Pass
REQ-2	Finish implementing unit testing on existing code base (goal 85% code coverage)	TC-2	Pass
REQ-2.1	Set up framework for mock data for java junit tests	TC-2.1	Pass
REQ-2.2	Setup framework for mock data for Flutter tests	TC-2.2	Pass
REQ-2.3	Implement unit tests to achieve maximum code coverage on existing code base	TC-2.3	Pass
REQ-3	Redesign and implement a new system for RBAC	TC-3	Pass
REQ-3.1	Set up Java class with any methods needed for roll and access processing to better organize RBAC logic on the backend	TC-3.1	Pass
REQ-3.2	Implement RBAC on each backend endpoint using these class and methods created for REQ-3.1	TC-3.2	Pass
REQ-3.3	Setup Dart class and methods needed for roll and access processing to organize RBAC for the front end	TC-3.3	Pass
REQ-3.4	Implement RBAC on each frontend screen using the class and methods created for REQ-3.3	TC-3.4	Pass

4 Requirements to be Tested

All requirements listed in Sections 3.1.1 and 3.1.2 were included in testing activities.

- Requirements defined by Team C were tested according to the test cases developed for this phase
- Requirements inherited from the previous cohort were tested using previously defined test cases to ensure system behavior remained unchanged

This approach ensures full coverage of both new and existing functionality and supports regression validation across the system.

5 Requirements Not To Be Tested

Testing of requirements that are being developed concurrently by other teams, or planned for future development phases, is not in the scope of this testing effort. Responsibility for testing those requirements will be assigned to the respective development teams.

6. Approach

The testing approach will employ a set of test types common in software development efforts. This set of tests will ensure that after Team C has implemented their new requirements, there is no new bugs or degradation of performance or feature set. Team C will consult all relevant technical documentation during test case design and during the testing phase of the effort to ensure focused and impactful testing as part of this phase.
(Summer/Fall 2025 software test plan was referenced while creating this list to ensure consistency)
Note: The list of test types is subject to change due to discovery. This initial set of test types is intended to demonstrate that the testing approach is comprehensive and all encompassing.

6.1 Unit Testing

Unit testing will be conducted to ensure individual units of code are returning output consistent with specified input. This is the most granular type of testing as it is checking all the individual functions within the code.

6.2 Integration Testing

Integration testing will be executed on the app once deployed in a live environment to ensure all the new features have fit into the system as expected. An example of this will be testing that roll-based access control is still functioning as expected after the new framework is implemented.

6.3 System Testing

System testing like integration testing will be executed on CareConnect in a live environment. The key difference is instead of checking that individual features fit into the application, the team will be focused on the entire app making sure everythingis still functional.

6.4 User Acceptance Testing

Utilizing individuals who know the application requirements but don't know the inner workings of the system and can perform black box tests and verify the user interface and experience are acceptable with the requirements.

6.5 Performance Testing

To measuring execution and response time of different functions to ensure system is performing under various loads. A successful effort includes no degradation in performance. Please reference the summer/Fall 2025 documentation for goal performance metrics.

6.6 Security Testing

To validate compliance with HIPAA/GDPR and other privacy expectations from clients. This will include testing the frontend, backend and infrastructure for policy compliance.

6.7 Usability Testing

To validate the user experience is consistent with client expectation (e.g. intuitiveness or minimal learning curve).

6.8 Regression Testing

Involves re-executing various other test types to ensure code modifications don't have any unintended consequences in the existing parts of the application. This will take place on a live environment and will have a focus on everything Team C inherited from the previous cohort.

7. Pass/Fail Criteria

Each test case was evaluated using the following outcome classifications:

- Pass: Actual results match expected results
- Fail: Actual results do not match expected results
- Blocked: Test execution could not be completed due to unmet prerequisites or external dependencies
  Not Executed: Test case was not executed during the testing phase

8. Suspension and Resumption Criteria

8.1 Suspension Criteria

Testing could be suspended if one or more of the following events occur:

- Defects:
    - At least one critical severity defect that prevents execution of a major piece of the workflow for example login/logout, patient or caregiver dashboard and there is no available workaround.
    - At least three high severity defects in the same functional flow that cause repeated test failure or prevent at least twenty percent of planned test cases.
- Environment or Infrastructure Issues:
    - The testing environment is unavailable.
    - A resource needed for testing (frontend, backend, or third-party service) is unavailable for use in testing.
- Build quality
    - If more than 30% of initial smoke tests fail, integration testing will be halted.

8.2 Resumption Criteria

- Defects:
    - All critical defects are resolved or reasonable workaround is found.
    - High severity defects are reduced enough to not block testing and target fix dates are agreed upon.
- Environment or Infrastructure Issues
    - Test environments become available and stable.
    - Less than five percent of planned cases remain blocked due to environmental or integration issues.
- Build quality
    - A new build is deployed and passes initial smoke tests.
    - New build does not introduce the required number of defects to suspend testing.

9. Deliverables

The following deliverables were produced as part of the testing effort:

- - This test report, including test results and analysis
- - Test execution reports generated from JUnit (backend testing)

- - Test execution reports generated from Flutter testing frameworks (frontend testing)

10. Testing Tasks

Detailed testing tasks and their execution timelines are provided in Appendix Section 17.3.

11. Environment Needs

11.1 Hardware

Testing was conducted using developer workstations and mobile device emulators to simulate iOS and Android environments. Web application testing was performed using standard desktop computing systems.

11.2 Software

The following software environments were required for testing:

11.2.1 Operating System.

Team C will need access to Android OS and iOS through emulators.

11.2.2 Local Environment

- - Docker Desktop
- - Java Development Kit (JDK)
- - Flutter SDK and Dart

- Integrated Development Environment (IDE) for application execution and testing

11.3 Tools

Testing activities utilized tools and environments that closely replicate production conditions. The following tools were used:

- - Mobile emulators for Android and iOS testing
- - Web browsers for web application testing
- - JUnit for backend unit and integration testing
- - Flutter/Dart testing frameworks for frontend testing
- - API testing tools (e.g., Postman)
- - Automated UI testing tools (e.g., Selenium)
- - GitHub Actions for continuous integration and automated test execution
- - Logging and monitoring tools for observing system behavior during testing

(List as cited in Summer/Fall 2025 software Test Plan)

Note: this list is subject to change if during the testing phase a better tool or technique is discovered.

12. Responsibilities

There are two groups that will split the testing responsibilities.

12.1 Team C

Team C will coordinate all of the testing effort as different testing needs come up. Team C will also execute all of the testing where a deep technical understanding of how requirements were implemented. These tests include:

- Unit testing
- Integration Testing
- Regression Testing
- System Testing
- Performance Testing
- Security Testing

12.2 Customer/Mentors

The stakeholders will be responsible for usability testing and user acceptance testing as these two test types success or failure depends on the opinion of the product owners/stakeholders. This group will also be responsible for reviewing testing documentation.

13. Staffing and Training Needs

13.1 Staffing

- Colyn Mahoney – Team Lead
- Ja'Lisa Hawkins – Business Analyst
- Obeng Buo - Technical Lead/Architect
- Zechariah Hillman – Testing Lead
- Henry Stewart – Software Developer

13.2 Training

Due to the requirements being focused on refactoring existing features and not adding anything new, there is no training needed at this time.

14. Schedule

Please see task list in appendix section 17.3.

15. Risks

The following events could impact the testing schedule,

- AWS outage.
- If product owner/stakeholders are unavailable for testing.
- If CareConnect itself becomes unavailable for whatever reason.

If any of these events occur there will be a shift in the testing schedule. The schedule will shift as much as possible until the end of the cohort.

16. Approvals

Approvals are required from the individuals listed in the approvals table at the top of this document.

17 Appendix

17.1 New Requirements

1 Migration to a New Production Host:
   - Investigate and migrate CareConnect to a new cloud-based hosting service on AWS.
   - Choose between AWS Fargate or AWS App Runner to improve flexibility and scalability.

2 Redesign and Implementation of New Testing Norms:
   - Design and implement a new unit testing paradigm for the existing CareConnect application.
   - Streamline the testing process to make it more efficient, reliable, and maintainable for current and future development.

3 New Role-Based Access Control (RBAC) System:
   - Redesign and refactor the current role-based access control structure.
   - Enforce authorization at both the UI and backend API levels so users can only access resources appropriate to their assigned roles.

17.2 Acronym List(Inherited From Summer/Fall 2025 STP and Added To)

- API: Application Programming Interface. A set of rules that allows one software component to interact with another.
- AI: Artificial Intelligence. Techniques enabling software to perform tasks typically requiring human intelligence.
- HIPAA: Health Insurance Portability and Accountability Act. U.S. legislation that sets standards for protecting sensitive patient health information.
- PHI: Protected Health Information. Any individually identifiable health data regulated under HIPAA.

- RBAC: Role-Based Access Control. A method of restricting system access based on a user's role or permissions.
- RDS: Relational Database Service. A managed database service used to store structured application data.
- DynamoDB: A managed NoSQL database service (Amazon DynamoDB) used for scalable, high-throughput data storage.
- TDD: Technical Design Document. A document that describes the architecture, design and major components of the system.
- STP: Software Test Plan. A document outlining the testing strategy, scope, resources and schedule for a project.
- SRS: Software Requirements Specification. A document detailing functional and non-functional requirements for the system.
- RTM: Requirements Traceability Matrix. A table mapping each requirement to its corresponding test case(s).
- UAT: User Acceptance Testing. Validation performed by end users to ensure the software meets their needs.
- CI/CD: Continuous Integration / Continuous Deployment. A set of practices that enable frequent code integration, automated testing and automated deployment.
- AWS: Amazon Web Services. A cloud platform offering various on-demand computing services (used to host CareConnect's backend).
- DBMS: Database Management System. Software used to store, retrieve and manage data
- REST API: Representational State Transfer API. A web service that uses HTTP methods to perform actions on resources.
- SOS: A distress signal; in CareConnect, triggers an emergency alert to a

caregiver.

- OTP: One-Time Password. A short-lived code used to verify a user's identity during authentication.

17.3 Task List

Task	Predecessor Task	Special Skills	Completion Date
Prepare Test Plan	CareConnect Design Documentation	N/A	2/14/2026
Unit Testing	Implement mock data framework	Software Developer	3/24/2026
Integration Testing	Application Deployed	Software Tester	3/24/2026
Regression Testing	Application Deployed	Software Tester	3/24/2026
System Testing	Application Deployed	Software Tester	3/24/2026
Performance Testing	Application Deployed	Software Tester	3/24/2026
Security Testing	Application Deployed	Software Tester	3/24/2026
Usability Testing	Application Deployed	No knowledge of inner workings of code	3/31/2026
User Acceptance Testing	Application Deployed	No knowledge of inner workings of code	3/31/2026
Submit Documentaion	All other testing tasks	N/A	3/31/2026

17.4 Test Cases

Field	Description
Test Case Specification Identifier	TC-1
Test Items	CareConnect backend deployment and production hosting environment verification test.
Related Requirements	REQ-1
Input Specifications	Dockerized Spring Boot backend application's account with ECS/Fargate configured. Valid container image stored in container registry. Environment variables and secrets configured
Output Specifications (Expected Results)	Backend services deploy successfully to AWS Fargate. Application containers start without errors Backend APIs are reachable from the client application. No regression in existing backend functionality

Environmental Needs	AWS ECS with Fargate enabled Internet connectivity Access to AWS Console or CI/CD pipeline Production-equivalent configuration
Special Procedural Requirements	None
Intercase Dependencies	REQ-1.1, REQ-1.2
Result	Pass

Step	Action / Input	Expected Result
1	Launch the application. On the welcome screen, select **Register**.	Registration screen appears with email and password fields.
2	Enter a valid email address and strong password. Select **Submit**.	Account is created; confirmation email/SMS is sent; user is redirected to dashboard or confirmation screen.
3	Close and relaunch the application. Attempt to log in using the new credentials.	Login succeeds and user is authenticated successfully.

Field	Description
Test Case Specification Identifier	TC-1.1
Test Items	CareConnect backend deployment and production hosting environment
Related Requirements	REQ-1
Input Specifications	Dockerized Spring Boot backend application.AWS account with ECS/Fargate configured. Valid container image stored in container registry.Environment variables and secrets configured
Output Specifications (Expected Results)	Backend services deploy successfully to AWS Fargate. Application containers start without errors Backend APIs are reachable from the client application. No regression in existing backend functionality

Environmental Needs	AWS ECS with Fargate enabled Internet connectivity Access to AWS Console or CI/CD pipeline Production-equivalent configuration
Special Procedural Requirements	None
Intercase Dependencies	REQ-1.1, REQ-1.2
Result	Pass

Step	Action / Input	Expected Result
1	Build Docker image for the Spring Boot backend.	Docker image builds successfully with no errors.
2	Push Docker image to configured container registry.	Image is available and versioned in registry.
3	Deploy backend service to AWS Fargate using the container image.	Service deploys successfully and tasks enter running state.
4	Access backend health endpoint or API endpoint.	Backend responds with HTTP 200 status.
5	Perform basic API call from frontend or API client.	API call succeeds and returns expected data.

Field	Description
Test Case Specification Identifier	TC-REQ-1.1
Test Items	Spring Boot backend Docker configuration and container build process
Related Requirements	REQ-1.1
Input Specifications	Spring Boot backend source code Dockerfile definition for backend service Required environment variables defined Local Docker runtime or CI build agent
Output Specifications (Expected Results)	Docker image builds successfully Application starts correctly inside the container

	Required ports are exposed No runtime or dependency errors occur during startup
Environmental Needs	Local development machine or CI environment Docker installed and running Internet access to retrieve dependencies
Special Procedural Requirements	None
Intercase Dependencies	None
Result	Pass

Step	Action / Input	Expected Result
1	Create or update Dockerfile for Spring Boot backend.	Dockerfile exists and is syntactically valid.
2	Run docker build using the Dockerfile.	Docker image builds successfully without errors.
3	Run container locally using the built image.	Container starts and application initializes.
4	Access application health or API endpoint.	Endpoint responds successfully.
5	Review container logs.	No critical errors or crashes are present.

Field	Description
Test Case Specification Identifier	TC-REQ-1.2
Test Items	CareConnect Spring Boot backend deployment on AWS Fargate
Related Requirements	REQ-1.2
Input Specifications	Docker image of Spring Boot backend available in container registry AWS ECS cluster configured for Fargate Task definition with correct CPU, memory, ports, and environment variables Networking configuration (VPC, subnets, security groups)

Output Specifications (Expected Results)	Backend service deploys successfully to AWS Fargate ECS tasks enter a running and healthy state Backend APIs are accessible externally or internally as configured No service crashes or restart loops occur
Environmental Needs	AWS ECS with Fargate enabled Configured VPC and networking resources Internet connectivity Access to AWS Console or CI/CD pipeline
Special Procedural Requirements	None
Intercase Dependencies	TC-REQ-1.1
Result	Pass

Step	Action / Input	Expected Result
1	Create or update ECS task definition using backend Docker image.	Task definition registers successfully.
2	Configure service to use AWS Fargate launch type.	Service configuration is saved without errors.
3	Deploy service to ECS cluster.	ECS service starts and launches tasks.
4	Monitor task status in AWS console.	Tasks transition to RUNNING and remain stable.
5	Send request to backend API endpoint.	API responds with expected HTTP status and data.

Field	Description
Test Case Specification Identifier	TC-REQ-2
Test Items	CareConnect backend and frontend unit testing coverage for existing code base
Related Requirements	REQ-2
Input Specifications	Existing CareConnect backend (Spring Boot) code base Existing CareConnect frontend (Flutter) code base

	Unit testing frameworks configured (JUnit, Flutter test) Mock data frameworks available
Output Specifications (Expected Results)	Unit tests execute successfully for existing code No failing tests in CI or local execution Test results are generated and accessible Existing functionality remains unchanged
Environmental Needs	Local development environment or CI environment JDK for backend testing Flutter SDK for frontend testing GitHub Actions or equivalent CI pipeline
Special Procedural Requirements	None
Intercase Dependencies	REQ-2.1, REQ-2.2, REQ-2.3
Result	Pass

Step	Action / Input	Expected Result
1	Configure unit testing frameworks for backend and frontend.	Test frameworks are correctly initialized.
2	Execute existing unit tests.	All existing tests pass successfully.
3	Add new unit tests for untested existing code.	New tests compile and execute successfully.
4	Run full test suite locally or in CI.	All unit tests pass with no regressions.
5	Review test reports.	Test coverage and results are documented.

Field	Description
Test Case Specification Identifier	TC-REQ-2.1
Test Items	Mock data framework for CareConnect backend Java unit tests
Related Requirements	REQ-2.1
Input Specifications	Spring Boot backend source code

	JUnit testing framework Mocking libraries (e.g., Mockito) Defined test data requirements
Output Specifications (Expected Results)	Mock data framework is successfully configured Backend unit tests can execute without reliance on live services Tests produce consistent and repeatable results No database or external API dependencies are required during test execution
Environmental Needs	Local development or CI environment Java JDK Build tool (Maven or Gradle) JUnit and mocking libraries available
Special Procedural Requirements	None
Intercase Dependencies	None
Result	Pass

Step	Action / Input	Expected Result
1	Configure mocking libraries in backend test dependencies.	Mocking libraries are available to test suite.
2	Create mock objects for database and external services.	Mock objects initialize without errors.
3	Update existing unit tests to use mock data.	Tests execute without accessing live systems.
4	Run backend unit tests.	All tests pass using mock data.
5	Review test output and logs.	Test results are consistent and repeatable.

Field	Description
Test Case Specification Identifier	TC-REQ-2.2
Test Items	Mock data framework for CareConnect Flutter frontend unit tests

Related Requirements	REQ-2.2
Input Specifications	Flutter frontend source code Flutter test framework Mock data definitions for UI and service layers Test configuration files
Output Specifications (Expected Results)	Mock data framework is configured for Flutter tests UI and service layer tests run without live backend dependency Test results are deterministic and repeatable No network calls are made during unit tests
Environmental Needs	Local development machine or CI environment Flutter SDK installed Dart test packages available
Special Procedural Requirements	None
Intercase Dependencies	None
Result	Pass

Step	Action / Input	Expected Result
1	Configure mock data utilities for Flutter tests.	Mock utilities load successfully.
2	Define mock responses for backend services.	Mock responses match expected data formats.
3	Update existing Flutter tests to use mock data.	Tests compile and execute successfully.
4	Execute Flutter unit tests.	All tests pass without backend connectivity.
5	Review test results.	Test output is consistent across runs.

Field	Description
Test Case Specification Identifier	TC-REQ-2.3
Test Items	Unit test implementation for existing CareConnect backend and frontend code

Related Requirements	REQ-2.3
Input Specifications	Existing backend and frontend source code Configured unit testing frameworks Mock data frameworks for backend and frontend Defined coverage goals
Output Specifications (Expected Results)	Unit tests cover the majority of executable code paths All unit tests pass successfully Test coverage reports are generated No regression or functional behavior changes occur
Environmental Needs	Local development or CI environment Java JDK Flutter SDK Coverage reporting tools
Special Procedural Requirements	None
Intercase Dependencies	TC-REQ-2.1, TC-REQ-2.2
Result	Pass

Step	Action / Input	Expected Result
1	Identify untested or low-coverage code areas.	Coverage gaps are documented.
2	Write unit tests for identified code paths.	Tests compile and execute successfully.
3	Execute full unit test suite.	All tests pass without failures.
4	Generate coverage reports.	Coverage metrics meet defined goals.
5	Review test results and reports.	No regressions or unexpected behavior observed.

Field	Description
Test Case Specification Identifier	TC-REQ-3
Test Items	CareConnect role-based access control system across backend and frontend

Related Requirements	REQ-3
Input Specifications	Updated RBAC design and implementation Backend authorization logic Frontend role and permission handling Defined user roles and permissions
Output Specifications (Expected Results)	RBAC rules are enforced consistently across the system Users can only access features allowed by their role Unauthorized access attempts are blocked No existing functionality is unintentionally restricted
Environmental Needs	Local or deployed CareConnect environment Test user accounts with different roles Backend and frontend services running
Special Procedural Requirements	None
Intercase Dependencies	REQ-3.1, REQ-3.2, REQ-3.3, REQ-3.4
Result	Pass

Step	Action / Input	Expected Result
1	Log in as a user with a defined role.	User is authenticated successfully.
2	Attempt to access features permitted for that role.	Access is granted and functionality works as expected.
3	Attempt to access features not permitted for that role.	Access is denied or feature is hidden.
4	Repeat tests with different user roles.	RBAC rules apply correctly for each role.
5	Monitor logs or audit output.	Unauthorized access attempts are recorded.

Field	Description
Test Case Specification Identifier	TC-REQ-3.1
Test Items	Backend Java RBAC utility classes and access-

	processing methods
Related Requirements	REQ-3.1
Input Specifications	Java RBAC utility class source code Defined roles and permissions Backend services using RBAC logic
Output Specifications (Expected Results)	RBAC utility classes compile successfully Role and permission checks return correct results Backend endpoints can consume RBAC logic No authorization logic is duplicated across services
Environmental Needs	Local development or CI environment Java JDK Spring Boot backend project
Special Procedural Requirements	None
Intercase Dependencies	None
Result	Pass

Step	Action / Input	Expected Result
1	Implement Java class for role and access processing.	Class compiles without errors.
2	Write unit tests for RBAC methods.	Tests execute and validate logic.
3	Invoke RBAC methods with allowed roles.	Access is granted as expected.
4	Invoke RBAC methods with disallowed roles.	Access is denied as expected.
5	Review backend logs during authorization checks.	RBAC decisions are logged correctly.

Field	Description
Test Case Specification Identifier	TC-REQ-3.2
Test Items	Backend API endpoints secured using RBAC logic
Related Requirements	REQ-3.2
Input Specifications	Backend API endpoints RBAC utility classes and methods

	Authenticated user requests with assigned roles
Output Specifications (Expected Results)	Backend endpoints enforce role-based access rules Authorized requests succeed Unauthorized requests are rejected with appropriate error responses No unsecured endpoints remain
Environmental Needs	Running backend service (local or deployed) Test user accounts with various roles API testing tool (e.g., Postman)
Special Procedural Requirements	None
Intercase Dependencies	TC-REQ-3.1
Result	Pass

Step	Action / Input	Expected Result
1	Authenticate as a user with a valid role.	User receives valid authentication token.
2	Send request to an endpoint allowed for the role.	Request succeeds with expected response.
3	Send request to an endpoint not allowed for the role.	Request is denied with authorization error.
4	Repeat requests using different user roles.	RBAC rules are enforced consistently.
5	Review API logs or responses.	Unauthorized access attempts are logged.

Field	Description
Test Case Specification Identifier	TC-REQ-3.3
Test Items	Frontend Dart RBAC utility classes and role-processing logic
Related Requirements	REQ-3.3
Input Specifications	Flutter frontend source code Dart RBAC utility classes

	Defined user roles and permissions
Output Specifications (Expected Results)	RBAC utility classes compile without errors Frontend correctly evaluates user roles UI components respond appropriately to role permissions No role-based logic is duplicated across UI components
Environmental Needs	Local development or CI environment Flutter SDK CareConnect frontend project
Special Procedural Requirements	None
Intercase Dependencies	None
Result	Pass

Step	Action / Input	Expected Result
1	Implement Dart class for role and access handling.	Class compiles successfully.
2	Write unit tests for frontend RBAC logic.	Tests validate role checks correctly.
3	Evaluate RBAC logic with permitted roles.	UI access checks return true.
4	Evaluate RBAC logic with restricted roles.	UI access checks return false.
5	Review application behavior.	Restricted UI elements are hidden or disabled.

Field	Description
Test Case Specification Identifier	TC-REQ-3.4
Test Items	CareConnect frontend screens secured using role-based access control
Related Requirements	REQ-3.4
Input Specifications	Flutter frontend screens Frontend RBAC utility classes Authenticated user sessions with assigned roles

Output Specifications (Expected Results)	Frontend screens enforce role-based access rules Authorized users can view and interact with permitted screens Unauthorized users cannot access restricted screens No unintended UI regressions occur
Environmental Needs	Running CareConnect frontend application Test user accounts with different roles Backend services available for authentication
Special Procedural Requirements	None
Intercase Dependencies	TC-REQ-3.3
Result	Pass

Step	Action / Input	Expected Result
1	Log in as a user with a specific role.	User session is established successfully.
2	Navigate to screens permitted for the role.	Screens load and function correctly.
3	Attempt to navigate to restricted screens.	Access is blocked or user is redirected.
4	Repeat navigation with different user roles.	Role restrictions apply consistently.
5	Observe UI behavior and logs.	No unauthorized screen access occurs.