Modified April 4, 2025

CovaCare Quality Assurance

1. Quality Assurance Plan

1.1 Purpose and Scope

The purpose of CovaCare's quality assurance is to ensure that our system meets both its functional and non-functional requirements. We will focus on verifying critical features—such as slip/trip/fall detection, prolonged inactivity detection, and alerting—alongside the system's performance, reliability, privacy, usability, security, compatibility, and maintainability. This plan outlines how we will verify that the system behaves as expected under real-world conditions.

1.2 Specific Requirements to Validate

Below is a summary of the non-functional requirements taken from our project documentation, and the focus of this QA Plan:

1. Performance
    - Real-time or near real-time video processing (ideally detect accidents and send alerts within one minute).
    - Manage system frame rate/resolution so the system remains responsive and does not miss incidents.
2. Reliability
    - 99% system uptime target.
    - <5% false-negative rate for fall detection (i.e., missed falls).

- Minimize false positives, but especially false negatives.

3. Privacy
    - All video processing is done locally, with no external storage of video frames.
    - Only minimal pose data is retained for short periods (30–60 frames).
    - Alerts contain no sensitive images or video unless explicitly authorized by the user.
4. Usability
    - Simple, user-friendly interface for configuring emergency contacts, camera settings, and detection thresholds.
    - Clear feedback when the system is online or offline.
    - Mobile application accessible on the same network
5. Security
    - All network traffic is restricted to the local Wi-Fi network or a VPN-protected environment.
    - Encrypted API communication for external alerts (HTTPS, if feasible).
    - Strict access control to camera data and system settings.
6. Compatibility
    - Support for iOS and Android mobile apps.
    - Integration with multiple camera types (IP cameras, RTSP-compatible, etc.).
7. Maintainability
    - Modular, well-documented code to simplify future changes.

- Agile, iterative development with regular testing so new features and bug fixes can be added efficiently.

Along with validating these non-functional requirements, we will also ensure our system works the way we intend it to, functionally.

2. QA Activities

This section describes the desired outcomes, the tasks we will conduct to achieve these outcomes, and the records we will keep to monitor quality and identify any mitigation strategies.

2.1 QA Outcomes

1. Validate Non-Functional Requirements
    - Confirm the system can process video in near real-time and dispatch alerts quickly.
    - Maintain high reliability (99% uptime, <5% false negatives).
    - Adhere to privacy standards: no long-term video storage
    - Ensure a secure local environment, restricting access to trusted devices/users.
    - Provide a user-friendly interface that meets user needs with minimal confusion.
2. Mitigate Defects Early
    - Identify bugs quickly
    - Detect and resolve security vulnerabilities (API misconfigurations, unencrypted data transfers).

- Pinpoint usability problems (form confusion, hidden fields, unclear error messages).

2.2. QA Tasks

(A) Required QA Tasks – These tasks are critical and must be performed:

1. Unit Testing & Integration Testing
    - Test each module (mobile UI, API, real-time processing) individually, then validate they work as a unit.
    - Required to meet performance, reliability, and maintainability goals.
    - Tools used: Jest/React Testing Library for UI, pytest for backend services.
2. System Testing (End-to-End)
    - Deploy the entire system (camera, real-time processing, alerts, mobile app) on a local network.
    - Verify performance constraints (fall detection < 1 minute).
    - Ensure accuracy thresholds (<5% false negatives, minimal false positives).
    - Required to confirm the entire system works seamlessly together.
3. Security Testing
    - Confirm local network isolation is effective.
    - Check that only authorized devices can access the system’s API.
    - Verify that no sensitive data is stored or transmitted outside the local network (or if transmitted, it uses secure protocols).
    - Required to meet privacy and security standards.

4. Usability Testing
    - Conduct user tests (both non-technical and technical individuals) to measure ease of use, clarity of forms, and overall comfort.
    - Required to validate the usability requirement and identify user confusion or form design issues.
5. Performance/Load Testing
    - Simulate extended operation (e.g., 24-hour test) to check system uptime.
    - Measure response time for alert generation under varied FPS or multiple cameras.
    - Required to ensure the system consistently meets near real-time performance.

(B) Additional Required QA Task

1. Model Accuracy & Long-Form Video Testing
    - Evaluate the fall detection model on a known test set (e.g., 1,100 samples, as outlined in the Code Testing document).
    - Test the LSTM-based approach with extended, real-world videos to verify accuracy (target <5% false negatives).
    - Confirm inactivity detection logic scales to longer durations without false alerts or performance issues.

(C) Recommended QA Tasks – Not strictly required but strongly suggested:

1. Code Reviews
    - Manual inspection of code for adherence to best practices.

   - Identify potential maintainability and performance issues early.

2. Additional Usability Feedback Rounds
   - After initial form redesign, recommended to run another usability study with an older demographic, who are the primary end-users.

2.3 QA Records

The following section details historic QA records, aligning with our plan in action. Each item shows what was done and when it was completed, along with references to relevant documentation.

1. Unit & Integration Testing Execution
   - Dates: Conducted iteratively from March 1 to March 10, 2025.
   - Reference: Code Testing Document (includes test cases for fall detection, inactivity detection, API endpoints, and the mobile UI).
   - Outcome: Discovered minor API issues (fixed by March 10). One major bug in inactivity detection logic (fixed by March 10).
2. User Acceptance Testing (UAT)
   - Date: March 30, 2025
   - Scope: Two users (both ~55 years old) set up CovaCare as if for a family member. Confirmed camera feeds, added themselves as emergency contacts, tested fall detection, received SMS alerts.

- Outcome: Successfully met functional requirements. Minor feedback about clarifying “system is fully set up” status. Logged in bug tracker for future UI improvement.

3. Usability Test & Peer Testing Reports
    - Dates: February 5 & March 13, 2025
    - Reference: Usability & Peer Testing Reports (details tasks, user confusion points, and suggestions like tooltips and camera status indicators).
    - Outcome: Led to form redesign (fields for IP address, username/password made clearer).
4. Model Accuracy Testing
    - Dates: March 10, 2025
    - Reference: Code Testing Document for fall detection LSTM results (93% accuracy on the 1,100-sample test set).
    - Outcome: Confirmed the system meets performance goals but flagged potential improvements for lower FPS streams. Updated code to handle <30 FPS gracefully.