Software Requirements Specification

For

Citizen Squad

Table of Contents

1. Introduction

1.1 Purpose

The Citizen Squad project is an open-source Flutter application designed to facilitate seamless reporting of public issues like abandoned vehicles, potholes, and garbage to the relevant authorities. It provides OTP-based authentication and integrates Google Maps and Parivahan APIs to enhance user experience and reporting accuracy.

1.2 Intended Audience

- Developers: To understand the system architecture, required modules, and technical requirements for building the app.
- Project Managers: To track progress, milestones, and ensure that all requirements align with the project goals.
- Testers and QA Teams: To design test cases based on the functional and non-functional requirements specified.
- End Users: Citizens who will use the app to report public issues.

1.3 Project Scope

This application helps citizens report issues quickly and keeps them informed about the status of their reports through real-time notifications. The app provides multiple functionalities, including a navigation bar, authentication with OTP delivery, and support for both admin and user roles. The backend uses MongoDB and Firebase, making the app scalable and accessible on Android and iOS platforms.

1.4 References

- Google Maps API Documentation:https://developers.google.com/maps
- Parivahan API Documentation: https://parivahan.gov.in
- Flutter Documentation: https://flutter.dev/docs

2. Overall Description

2.1 Product Perspective

Citizen Squad is a new, self-contained product aimed at empowering citizens to report public issues like potholes, garbage, and abandoned vehicles. It is designed to bridge the gap between citizens and authorities (e.g., municipal corporations and RTO) by automating the reporting process through APIs and real-time tracking.

The product consists of the following major components:

- Frontend: Flutter-based mobile interface to ensure a consistent user experience across Android and iOS.
- Backend: Firebase for authentication, notifications, and real-time updates; MongoDB for storing reports and user data.
- APIs: Integration with Google Maps for location-based reporting and Parivahan API to fetch vehicle information.

2.2 Product Features

Citizen Squad offers the following high-level features:

- OTP-based Authentication: Ensures secure login using Firebase authentication.
- Issue Reporting: Allows users to report potholes, garbage, or abandoned vehicles with geolocation.
- Dashcam Integration: Uses the camera to detect and report potholes automatically while driving.
- Google Maps Integration: Enables users to mark the exact location of issues.
- Real-time Notifications: Keeps users updated on the status of their reports.
- Multi-Platform Support: Works seamlessly on both Android and iOS.
- Role-Based Interfaces: Different screens for admin, users, and authorities for role-specific actions.

2.3 User Classes and Characteristics

Citizen Squad serves the following types of users:

- General Users (Citizens):
  - Frequency: Frequent users who report issues in their locality.
  - Expertise: Limited technical expertise; should find the app easy to use.
  - Functionality Access: Can log in, report issues, and track status updates.
- Authorities (RTO/Municipal Workers):
  - Frequency: Moderate use to monitor reports and resolve issues.
  - Expertise: Basic understanding of admin functions.
  - Functionality Access: Access to location-based reports and vehicle details through the admin portal.
- Admin Users:
  - Frequency: Rare, only when management-level oversight is needed.
  - Expertise: Higher technical expertise for system management.
  - Functionality Access: Manage users, view statistics, and modify settings.

2.4 Operating Environment

Citizen Squad will operate under the following conditions:

- Hardware Requirements:
  Android 5.0+ / iOS 12.0+ smartphones.
  Minimum 2GB RAM for optimal performance.
- Software Requirements:
  Frontend: Flutter 3.0+ with Provider for state management.
  Backend: Firebase and MongoDB for data storage and authentication.
  APIs: Google Maps and Parivahan APIs for geolocation and vehicle information.
  Development Tools: Visual Studio Code / Android Studio.
  Network: Requires stable internet for API calls and notifications.

2.5 Design and Implementation Constraints

- Technology Constraints:
  - The app is built using Flutter; only libraries and tools compatible with it can be used.
  - Firebase and MongoDB are mandatory for backend operations, limiting alternatives.
- Security Constraints:
  - Must comply with GDPR for data privacy, especially for user data and location tracking.
  - Phone authentication requires adherence to telecom regulatory policies for OTP delivery.
- Performance Constraints:
  - Continuous use of the camera (Dashcam feature) may affect battery life.
  - Location tracking requires GPS to be enabled, which may drain the battery.
- External Dependencies:
  - Reliance on Parivahan API and Google Maps API could affect performance if their services are interrupted.
  - Requires Firebase Cloud Messaging (FCM) for notifications.

2.6 Assumptions and Dependencies

- Assumptions:
  - The municipal authorities and RTO will actively monitor reports submitted via the app.
  - Users will have access to a stable internet connection for reporting issues.
  - APIs (Google Maps and Parivahan) will remain operational and responsive.
  - OTP delivery will work across all telecom providers without delays.
- Dependencies:
  - Third-Party APIs: The app relies on Google Maps API for location tracking and Parivahan API for vehicle information.
  - Backend Services: Firebase and MongoDB are essential for authentication and data storage.

- Platform Updates: The app may require updates to remain compatible with the latest Android/iOS versions.
- Notifications: Dependent on Firebase Cloud Messaging (FCM) for delivering notifications to users.

3. System Features

This section organizes the major functionalities provided by the Citizen Squad application, aligned with user needs and operational flow. Each feature is described with its priority, stimulus/response sequences, and specific functional requirements.

3.1 Issue Reporting

This feature allows citizens to report public issues by providing relevant details, including the location and optional images

3.1.1 Description and Priority

- Priority: High
  This feature forms the core of the application by facilitating real-time issue reporting to the authorities.
- Benefit: 9/9 (Improves public service efficiency)
- Penalty: 8/9 (Without this, the app loses relevance)
- Cost: 4/9 (Moderate effort to implement API calls and UI)
- Risk: 3/9 (API failures can delay reports)

3.1.2 Stimulus/Response Sequences

1. User Action: The user logs in with OTP and clicks on "Report Issue."
   System Response: Displays the issue reporting form with fields like issue type, description, and location.
2. User Action: The user selects the issue type (e.g., pothole) and captures an image.
   System Response: Stores the image and enables location selection on the map.
3. User Action: The user submits the report.

System Response: Sends the report to the backend and confirms submission with a notification.

4. User Action: The user views the report status in their profile.
   System Response: Displays the issue status (e.g., Pending, In Progress, Resolved).

3.1.3 Functional Requirements

- REQ-1: The system shall allow users to select the issue type from predefined categories (e.g., pothole, garbage).
- REQ-2: The system shall use GPS to capture the issue's location automatically or allow manual location selection.
- REQ-3: The system shall allow users to upload a photo while reporting.
- REQ-4: The system shall store the report details in the backend (Firebase/MongoDB).
- REQ-5: The system shall notify users upon successful submission of the report.
- REQ-6: The system shall display the issue status in real time using data from the backend.

3.2 Dashcam-based pothole detection

This feature enables the automatic detection of potholes while the user is driving, utilizing the smartphone camera.

3.2.1 Description and Priority

- Priority: Medium
  This feature enhances user experience by offering automated reporting, reducing manual intervention.
- Benefit: 7/9 (Increases accuracy and convenience)
- Penalty: 5/9 (Manual reporting would still be available as a fallback)
- Cost: 6/9 (Requires image processing and AI integration)
- Risk: 6/9 (Battery consumption and performance issues)

3.2.2 Stimulus/Response Sequences

1. User Action: The user enables the Dashcam feature in the settings.
   System Response: Activates the camera and starts analyzing the road.

2. System Action: The system detects a pothole from camera input using AI-based detection algorithms.
   System Response: Automatically creates a report with the pothole location and stores it in the backend.
3. User Action: The user reviews the report in their profile.
   System Response: Displays the automated report with the captured location and timestamp.

3.2.3 Functional Requirements

- REQ-7: The system shall allow users to enable or disable the Dashcam feature from the settings.
- REQ-8: The system shall use the smartphone camera to analyze road conditions.
- REQ-9: The system shall generate an automatic report when a pothole is detected.
- REQ-10: The system shall notify users when an automated report is created.
- REQ-11: The system shall minimize battery usage while the Dashcam is running.

3.3 Real Time Status Tracking

This feature allows users to monitor the progress of their submitted reports, from submission to resolution.

3.3.1 Description and Priority

- Priority: High
  Status tracking ensures users are informed about the handling of reported issues.

- Benefit: 8/9 (Improves transparency)
- Penalty: 6/9 (User engagement might decrease without this feature)
- Cost: 3/9 (Requires real-time syncing with the backend)
- Risk: 2/9 (Low risk, as this involves standard API calls)

3.3.2 Stimulus/Response Sequences

1. User Action: The user logs into their profile and clicks on "My Reports."
   System Response: Displays a list of all submitted reports with their current status.
2. User Action: The user clicks on a specific report to view more details.
   System Response: Shows the report status timeline (e.g., Submitted → In Progress → Resolved).
3. System Action: When the status of a report changes, the system sends a notification to the user.
   System Response: The user receives the notification and sees the updated status.

3.3.3 Functional Requirements

- REQ-12: The system shall display all submitted reports in the "My Reports" section.
- REQ-13: The system shall show the current status of each report (e.g., Pending, In Progress).
- REQ-14: The system shall allow users to view a detailed timeline of the report’s progress.
- REQ-15: The system shall notify users whenever the status of their report changes.

3.4 Multi User Role Management

This feature provides different access levels to users based on their roles (e.g., citizen, municipal worker, admin).

3.4.1 Description and Priority

- Priority: High

  Role management ensures that each user class has access only to the features relevant to their role.
- Benefit: 9/9 (Enhances security and usability)
- Penalty: 7/9 (Improper management may compromise data security)
- Cost: 4/9 (Requires backend logic and UI separation)
- Risk: 5/9 (Complexity increases with more roles)

3.4.2 Stimulus/Response Sequences

1. System Action: The system assigns roles (citizen, municipal worker, admin) at user registration.
2. User Action: A municipal worker logs in to the admin portal.

   System Response: Displays relevant reports for their jurisdiction only.
3. User Action: The admin accesses the app settings to modify roles.

   System Response: Allows adding, editing, or removing user roles.

3.4.3 Functional Requirements

- REQ-16: The system shall assign roles based on user type (e.g., citizen, worker, admin).
- REQ-17: The system shall provide different interfaces based on the assigned role.
- REQ-18: The system shall restrict access to sensitive data based on roles.
- REQ-19: The system shall allow admins to modify user roles.
- REQ-20: The system shall notify users if their roles or permissions change.

4. External Interface Requirements

4.1 User Interfaces

The user interface (UI) for the Citizen Squad app will be designed for ease of use and accessibility, following modern UI/UX design principles.

Logical Characteristics

- Screen Layout: The main screens include the Home Screen, Report Issue Screen, My Reports Screen, and User Profile Screen.
- Standards: The UI will adhere to Material Design guidelines to ensure consistency across Android devices.

Standard Elements:

- Navigation Bar: Includes icons for Home, Report, and Profile, appearing on all screens.
- Buttons: Standard buttons will include "Submit," "Cancel," "Edit," and "Delete."
- Help Icon: A question mark icon will provide access to help documentation from any screen.
- Error Messages: Error messages will be displayed in a consistent format at the top of the screen, using red text to indicate errors and yellow for warnings.

Sample Screen Images

- Home Screen: Displays available features with buttons for reporting issues and viewing statuses.
- Report Issue Screen: Includes input fields for issue type, description, and location, along with an image upload option.
- My Reports Screen: Lists all reported issues with statuses and allows users to filter by type.

4.2 Hardware Interfaces

The Citizen Squad application will interact with various hardware components to provide its functionalities.

Supported Device Types

- Mobile Devices: Smartphones and tablets running Android OS (version 8.0 and above).
- Camera: Utilized for capturing images of reported issues.

Data and Control Interactions

- Camera Interaction: The app will access the device’s camera through the Android Camera API to capture images during issue reporting.
- GPS Interaction: The app will use the device's GPS module to automatically fetch the user’s location when submitting a report.

Communication Protocols

- The app will employ standard Android APIs for camera and GPS functionality, ensuring compatibility across devices.

4.3 Software Interfaces

The Citizen Squad application will integrate with various software components to operate effectively.

Connections with Other Software Components

- Database: Firebase Realtime Database (version TBD) for storing user reports and managing user data.
- Operating System: Android OS (version 8.0 and above).

Data Items

- Incoming Data: User inputs from report forms, including issue type, description, and location coordinates.
- Outgoing Data: Confirmation messages upon successful report submission, status updates, and user profile information.

Services Needed

- The application will require services for user authentication, report submission, and status tracking, with communications handled via RESTful API calls

4.4 Communications Interfaces

The Citizen Squad app will implement various communication functions to enhance user experience and functionality.

Requirements

Protocols: The app will use HTTP/HTTPS for all API communications, ensuring secure data transfer.

Message Formatting

All API requests and responses will follow JSON formatting to ensure compatibility and ease of parsing.

Communication Security

Data transferred between the app and backend services will be encrypted using HTTPS to protect user information.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The Citizen Squad app is expected to perform efficiently under various conditions to ensure a seamless user experience.

- Response Time: The app must have a maximum response time of 2 seconds for loading screens and fetching data from the database under normal network conditions.
- Scalability: The application should be able to handle up to 10,000 concurrent users without degradation of performance.
- Data Sync Frequency: The app must sync data with the Firebase backend every 5 seconds while the user is active, ensuring real-time updates for all users.
- Image Upload Time: The maximum upload time for images should not exceed 5 seconds on a standard 4G network to provide timely feedback to users.
- These requirements aim to enhance user satisfaction and maintain engagement, allowing users to interact with the application effectively.

5.2 Safety Requirements

Safety is a crucial aspect of the Citizen Squad application, particularly considering the nature of user-reported incidents.

- Data Privacy: Ensure that user-reported data does not include personally identifiable information (PII) without explicit consent, to mitigate risks related to data exposure.
- Incident Reporting Accuracy: The app must validate user inputs for reports to prevent false reporting, which could lead to misallocation of resources.
- User Safety Notifications: If a user reports an incident classified as a safety concern (e.g., crime, medical emergency), the app must trigger an immediate notification to local authorities.
- Compliance with relevant safety regulations, such as GDPR for data privacy and local laws regarding reporting crimes, is mandatory. No safety certifications are currently required for the software.

5.3 Software Quality Attributes

The following software quality attributes are crucial for the Citizen Squad application to ensure it meets user and developer expectations:

- Usability: The application should have a user satisfaction rating of at least 85% in user surveys, ensuring it is intuitive and easy to navigate.
- Reliability: The application must maintain 99.9% uptime to ensure continuous access for users.
- Maintainability: Code should follow standard coding conventions and practices, allowing developers to implement changes with a maximum average time of 4 hours per feature update.
- Portability: The application should be compatible with Android OS versions 8.0 and above, facilitating broad access across devices.
- Testability: The application must be designed to allow automated testing for at least 90% of its functions, ensuring high-quality releases.
- These quality attributes will guide the development process and ensure that the application meets user needs and expectations effectively.

Appendix A: Glossary

This glossary defines the key terms, acronyms, and abbreviations used in this Software Requirements Specification (SRS) for the Citizen Squad application.

- API: Application Programming Interface; a set of functions and protocols for building software applications.
- CRUD: Create, Read, Update, Delete; the four basic operations for managing data in a database.
- DBMS: Database Management System; software for creating and managing databases.
- Firebase: A cloud-based platform by Google for building mobile and web applications, which provides real-time database capabilities.
- GPS: Global Positioning System; a satellite-based navigation system used for determining the precise location of a user.

- UI: User Interface; the space where user interactions occur, including the layout and design of the application.
- UX: User Experience; the overall experience and satisfaction a user has when interacting with an application.
- 2FA: Two-Factor Authentication; a security process in which the user provides two different authentication factors to verify themselves.