Nibin Joseph

Software Requirements Specification (SRS)

Marian College Kuttikkanam

Document Details

Submission ID
trn:oid:::26696:104890412

Submission Date
Jul 18, 2025, 8:46 AM GMT+5:30

Download Date
Jul 18, 2025, 8:49 AM GMT+5:30

File Name
Software Requirements Specification (SRS) (1).pdf

File Size
259.5 KB

16 Pages

2,817 Words

17,977 Characters

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

0 AI-generated only 0%
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Software Requirements Specification (SRS)

Project Title: Xpose – AI-Enabled Smart Crime Reporting System with Blockchain Integration

Version: 1.0

Date: 17 July 2025

Prepared By.

Name : Nibin Joseph

Register Number: 24PMC137

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the requirements for Xpose – AI-Enabled Smart Crime Reporting System with Blockchain Integration. This document outlines the system's functionality, interfaces, performance, and design constraints to guide its development and evaluation. It aims to ensure secure, anonymous, and immutable reporting of crimes using AI for prioritization and blockchain for tamper-proof storage.

1.2. Document Conventions

- Headings are numbered based on section and subsection (e.g., 1.1, 1.2).
- Functional requirements are prefixed with "FR" (e.g., FR001).
- Non-functional requirements use "NFR" (e.g., NFR001).
- Keywords such as shall, should, and must indicate requirement importance.
- Technical terms and acronyms are defined in Section 1.5.

1.3. Intended Audience and Reading Suggestions

This document is intended for:

- Project Evaluators and Faculty: To assess the completeness, feasibility, and innovation of the proposed system.
- Development Team: To understand system requirements, components, and expected behavior.
- Future Contributors: To reference architectural and functional insights for maintenance or enhancement.

Suggested focus:

- Evaluators: Sections 1, 4, 5, and 6.
- Developers: Sections 3 and 5.
- Reviewers: Sections 2 and 3.

1.4. Project Scope

Xpose is a mobile-first, AI-integrated crime reporting platform that enables citizens to anonymously report crimes using a Flutter app. Reports are analyzed using AI/NLP to detect spam and assess urgency. Valid reports are stored immutably on a private blockchain developed using Go and LevelDB.

The system includes:

- A citizen-facing Flutter app for report submission.
- Firebase OTP-based authentication.
- Spring Boot backend for coordination.
- FastAPI microservice for AI classification.
- Blockchain for tamper-proof storage.
- Next.js-based admin and police dashboards.

Limitations:

- Admins cannot modify or reject reports.
- Reports are immutable once stored.
- Only the police can append status updates to reports on-chain.

1.5. Definitions, Acronyms, and Abbreviations

- AI: Artificial Intelligence
- NLP: Natural Language Processing
- OTP: One-Time Password (used for phone-based authentication)
- JWT: JSON Web Token (used for secure API access)
- API: Application Programming Interface
- FastAPI: Python web framework used for building AI/NLP microservices
- Flutter: Open-source UI toolkit used for building cross-platform mobile apps
- PostgreSQL: NoSQL database used for storing general application data
- LevelDB: Lightweight key-value store used for blockchain data
- Go (Golang): Programming language used to implement the private blockchain

- Spring Boot: Java-based framework used for building the core backend logic
- Gemini API: Google's multilingual language model used for translation

2. Overall Description

2.1. Product Perspective

Xpose is a distributed, AI-enhanced crime reporting system designed to function as a self-contained platform integrating multiple microservices. It is **not part of a larger existing system**, but it comprises modular components that interact through well-defined APIs. These include a mobile frontend (Flutter), backend (Spring Boot), AI/NLP microservice (FastAPI), and a private blockchain implemented in Go.

2.2. Product Functions

At a high level, Xpose performs the following core functions:

- Enables citizens to submit anonymous or identified crime reports with location and media.
- Uses AI to filter spam and classify reports by urgency
- Translates multilingual reports using Gemini API
- Stores valid reports immutably on a private blockchain (Go + LevelDB)
- Provides a dashboard for admin/police to monitor and escalate cases
- Sends push notifications and status updates to users

Details of each function are expanded in Section 3.1.

2.3. User Characteristics

Xpose will be used by the following user types:

- User Type Characteristics
- Citizen General public with mobile phones; no technical expertise required
- Admin Authorized monitoring personnel with basic web and data literacy
- Police Law enforcement officers accessing reports and updating statuses via dashboard

Most users will interact with mobile/web interfaces. The design prioritizes ease of use, multilingual support, and anonymity.

2.4. Constraints

- Reports, once stored on the block-chain, are **immutable**
- Admins **cannot modify, reject, or delete** reports
- Limited by capabilities of Gemini API and custom AI models
- AI model accuracy depends on training data quality
- Works with internet access (especially for syncing and blockchain interactions)
- Technology choices: Flutter (mobile), Spring Boot (backend), Go (blockchain), FastAPI (AI)
- Budget and resource constraints may limit infrastructure scale

2.5. Assumptions and Dependencies

- Users have access to Android/iOS smartphones with internet
- Firebase services (Auth) are available and reliable
- Google Gemini API supports target regional languages
- Hosting on GCP/Render is stable
- AI/NLP microservice will run on a containerized deployment (Docker)
- PostgreSQL and LevelDB instances are persistent and performant
- Police will have access to the system through a secured admin panel

3. Specific Requirements

This section details the functional and non-functional requirements. Each requirement is uniquely identified using a standardized label.

3.1. Functional Requirements

Each functional requirement outlines a specific feature or action the system must perform.

FR001 – Report Submission

- **Description**: Allows users to submit crime reports with text, media, and location.
- **Input**: Report description, optional images/videos, and location data.
- **Processing**: Data is validated and prepared for AI analysis.
- **Output**: Report data is forwarded to the AI module and saved if valid.
- **Pre-conditions**: User is authenticated via Firebase OTP.
- **Post-conditions**: Validated report is passed to the AI/NLP service.
- **Error Handling**: User is notified if mandatory fields are missing or if network fails.

FR002 – Anonymity Feature

- **Description**: Allows users to choose anonymous mode while submitting reports.
- **Input**: Boolean flag for anonymity.
- **Processing**: The system omits personal details if anonymity is enabled.
- **Output**: Report is tagged as anonymous and saved without user ID.
- **Pre-conditions**: Anonymity toggle must be enabled before submission.
- **Post-conditions**: Report is stored with anonymity metadata.
- **Error Handling**: Reverts to default if toggle fails; user is notified.

FR003 – AI/NLP-Based Report Processing

- **Description**: Filters spam and classifies urgency using FastAPI microservice.

- **Input**: Text description of the report.
- **Processing**: AI model processes input for spam probability and urgency level.
- **Output**: Returns classification labels (e.g., High, Medium, Low urgency).
- **Pre-conditions**: Valid report content is available
- **Post-conditions**: Classification results are attached to the report.
- **Error Handling**: Fallback to manual flag if AI service is down.

FR004 – Blockchain Storage

- **Description**: Stores validated reports immutably on a blockchain.
- **Input**: Report metadata, AI classifications, media references.
- **Processing**: A new block is created and added to the chain using Go and LevelDB.
- **Output**: Hash and timestamp of the stored report.
- **Pre-conditions**: Report is classified as valid.
- **Post-conditions**: Block is created and chained.
- **Error Handling**: If blockchain service fails, report is queued for retry.

FR005 – Admin Dashboard: Report Viewing and Analytics

- **Description**: Admins view report summaries, analytics, and filter reports.
- **Input**: Filters like date, urgency, crime type.
- **Processing**: Dashboard fetches blockchain-logged data and visualizes it.
- **Output**: Charts, tables, and case list.
- **Pre-conditions**: Admin is logged in with JWT token.
- **Post-conditions**: Reports and insights are displayed.
- **Error Handling**: Shows error message if data cannot be fetched.

FR006 – Police Portal: Case Escalation and Status Updates

- **Description**: Enables police to update investigation status.
- **Input**: Selected report ID, status text (e.g., Under Investigation).
- **Processing**: Status is appended to blockchain.

- **Output**: Updated chain entry.
- **Pre-conditions**: Police is authenticated.
- **Post-conditions**: Chain reflects new status.
- **Error Handling**: Logs failed update attempt.

FR007 – Notification and Feedback Loop

- **Description**: Sends report status updates to users via push notifications.
- **Input**: Report status change, user token.
- **Processing**: Triggers notification service.
- **Output**: Push message to user's device.
- **Pre-conditions**: Report has an update.
- **Post-conditions**: Notification delivered.
- **Error Handling**: Retry if push delivery fails.

FR008 – User Authentication (Firebase OTP Login)

- **Description**: Authenticates users using phone number and OTP via Firebase Authentication.
- **Input**: User phone number and one-time password (OTP).
- **Processing**: Firebase validates the OTP and generates a session token.
- **Output**: Authenticated session allowing access to report submission and notifications.
- **Pre-conditions**: User provides a valid phone number.
- **Post-conditions**: User is logged in and their UID is accessible for report association.
- **Error Handling**: Displays error if OTP is invalid or expired. Retry mechanism for resending OTP.

3.2. Non-Functional Requirements

Non-functional requirements describe the overall system attributes and quality goals that the system must meet.

3.2.1. Performance Requirements

- The system shall respond to user interactions within 2 seconds under normal load.
- The AI/NLP classification microservice shall process incoming reports within 1 second on average.
- The blockchain layer shall confirm and store valid reports within 5 seconds.
- The mobile app shall support concurrent use by at least 1000 active users.

3.2.2. Security Requirements

- The system shall use Firebase OTP for user authentication.
- Admin and police dashboards shall implement JWT-based access control.
- Report data shall be anonymized if the user enables anonymity.
- All communication between services shall use HTTPS.
- The blockchain layer shall guarantee immutability of stored data.

3.2.3. Usability Requirements

- The mobile app shall be usable by non-technical users with minimal training.
- The system shall support multilingual input and UI, including translation via Gemini API.
- User interface shall be intuitive and responsive across common screen sizes.
- Icons, labels, and feedback messages shall be clear and accessible.
- The dashboard shall support filters and visual analytics for efficient report review.

3.3. External Interface Requirements

3.3.1. User Interfaces

- The citizen-facing Flutter app shall offer a clean UI with navigation for submitting reports, viewing report status, and receiving updates.

- The admin and police dashboards (built with Next.js) shall include responsive design, sidebar navigation, data tables, status filters, and charts for analytics.

3.3.2. Hardware Interfaces

- The system does not interact with any hardware sensors directly.
- Smartphones are assumed to have GPS, camera, and internet connectivity for complete functionality.

3.3.3. Software Interfaces

- The system interfaces with Firebase Authentication for OTP-based user login.
- The AI/NLP classification microservice communicates with the main backend via REST APIs.
- PostgreSQL is used for general data storage.
- LevelDB is used in the private blockchain module for immutable data storage.
- Gemini API is used for multilingual report translation.

3.3.4. Communications Interfaces

- All inter-service communication will use HTTPS REST APIs.
- The mobile app and dashboards communicate with the backend over HTTPs.
- No Bluetooth or other local communication protocols are used.

3.4. Data Model/Database Requirements

- PostgresSQL shall store user profiles, feedback data, and metadata for crime reports.
- Each report document includes fields such as: report ID, user ID (or anonymous), location, media URLs, timestamp, urgency level, and spam flag.
- LevelDB in the blockchain layer stores each confirmed crime report with a hash, timestamp, and immutable reference.

- The database shall be optimized for read/write concurrency and fault tolerance.
- All databases shall be hosted in containers with persistent volumes for durability.

4. Problem Identification & Analysis

4.1. Problem Statement

In many regions, citizens face significant barriers when reporting crimes due to fear of retaliation, lack of anonymity, or distrust in traditional reporting systems. Additionally, existing systems often suffer from inefficiencies, lack of transparency, and susceptibility to data tampering. There is a critical need for a secure, anonymous, and trustworthy platform that enables streamlined and tamper-proof crime reporting and follow-up.

4.2. Analysis of Existing Solutions/Literature Review

Several platforms exist for reporting crimes (e.g., India's Crime and Criminal Tracking Network & Systems (CCTNS), ReportIt, and regional police apps), but they typically lack:

- True anonymity
- Spam filtering and urgency prioritization
- Tamper-proof record keeping
- Multilingual accessibility

Blockchain-based public grievance systems and AI chatbots have been explored in research, yet there remains a gap in integrating these technologies for secure crime reporting.

Xpose aims to bridge this gap by:

- Using AI for classifying urgency and filtering spam
- Storing validated reports on an immutable blockchain
- Allowing users to report anonymously
- Supporting multilingual input with translation

4.3. Proposed Approach and Innovation

Xpose integrates four major components:

- Flutter Mobile App: User-friendly UI for report submission and tracking
- FastAPI Microservice: Spam filtering and urgency detection using NLP models
- Spring Boot Backend: Core logic and API gateway to coordinate services
- Private Blockchain (Go + LevelDB): Immutable storage of validated reports

Innovations include:

- Citizen anonymity powered by OTP-auth and metadata stripping
- Multilingual input processed using Gemini API
- Report traceability and tamper-proof logs via custom blockchain implementation

4.4. Research Questions

- How effective are AI models in classifying urgency and filtering false reports in regional languages?
- Can a private blockchain implementation with LevelDB ensure both immutability and performance at scale?
- What are the usability trade-offs between anonymity and accountability in citizen crime reporting?

5. System Architecture (High-Level Design)

The system architecture of Xpose follows a modular, microservices-based design pattern. It integrates several distributed components to ensure secure, scalable, and efficient crime reporting, classification, and immutable storage.

The architecture supports:

- Multi-platform access via a mobile app and web dashboard.
- AI-powered report classification through a FastAPI-based microservice.
- Tamper-proof data storage using a custom blockchain implementation.
- Seamless interaction between services using REST APIs over HTTPS.

✧ Architecture Diagram Type: UML Component Diagram – This diagram illustrates the major system components and their interactions.

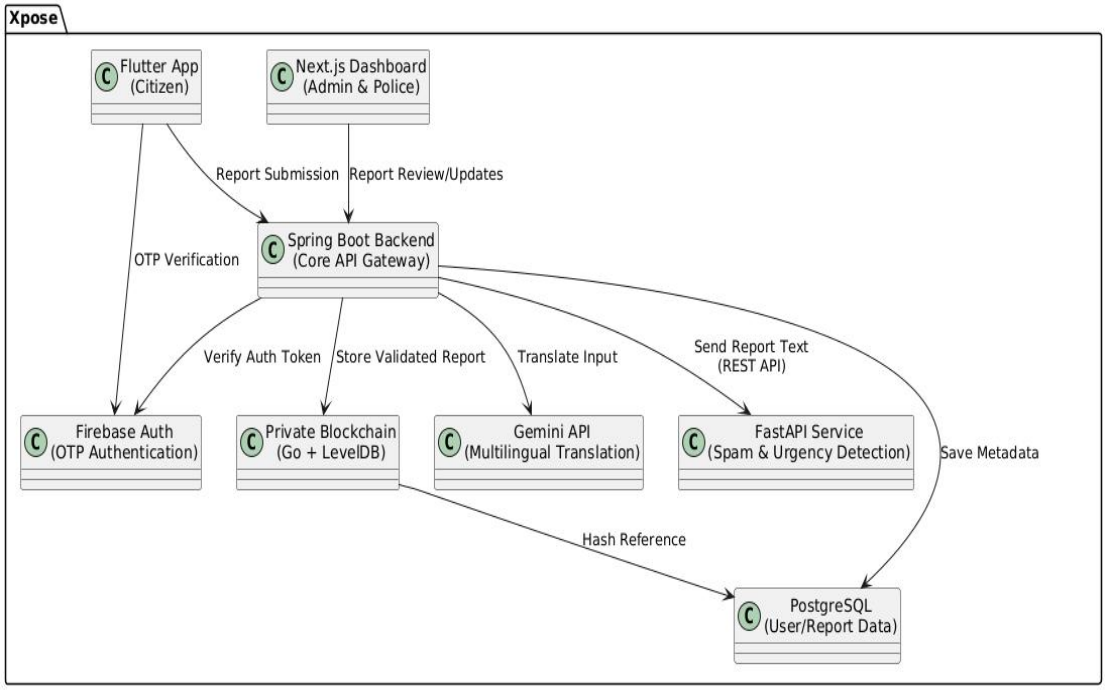

Figure 1: High-Level Architecture of Xpose

6. Future Enhancements/Research Directions

- ❖ Integration of Image/Video Analysis: Enhance the AI/NLP module to process and classify multimedia content attached to reports.
- ❖ Decentralized Identity Management: Introduce DID (Decentralized Identifiers) for secure and privacy-preserving user authentication.
- ❖ Advanced Anomaly Detection: Incorporate graph-based or unsupervised learning methods to detect fake or coordinated reports.
- ❖ Blockchain Interoperability: Explore compatibility with public blockchains for auditability without compromising privacy.
- ❖ AI Explainability: Implement explainable AI techniques to improve transparency in urgency classification.
- ❖ Offline Reporting Capability: Enable reports to be stored locally and synced when internet becomes available.

- Broader Language Support: Extend translation capabilities to cover more regional dialects.

7. References

- Firebase Authentication Documentation – https://firebase.google.com/docs/auth
- Spring Boot Reference Guide – https://docs.spring.io/spring-boot/docs/current/reference/html/
- FastAPI Documentation – https://fastapi.tiangolo.com/
- Gemini API (Google AI) – https://ai.google.dev/gemini-api
- Flutter Framework – https://flutter.dev/
- PostgreSQL Documentation – https://www.postgresql.org/docs/
- LevelDB Key-Value Store – https://github.com/google/leveldb
- Government of India, CCTNS (Crime and Criminal Tracking Network & Systems) – https://digitalpolice.gov.in
- Research articles on Blockchain-based Grievance Redressal Systems and AI in Public Safety (Citations to be appended from literature review if required)
- IEEE & ACM Digital Libraries – For supporting research on secure and anonymous reporting platforms.

8. Disclosure Statement

This Software Requirements Specification (SRS) document was primarily developed by the student, **Nibin Joseph (Register Number: 24PMC137)**, as part of the MCA program at **Marian College Kuttikkanam (Autonomous)**.

AI tools such as **ChatGPT-4** and **Grammarly** were utilized in a supportive role — specifically for:

- Brainstorming initial ideas for functional and non-functional requirements
- Refining sentence clarity and improving grammar
- Structuring technical content more effectively

All **core analysis**, **requirement identification**, **architecture design**, and **critical decision-making** reflect the original work and technical understanding of the author.