Kavach
Crime Reporting and Hotspot Mapping System

Software Requirements Specification

1. Problem Statement

Crime reporting in many regions is slow, inconsistent, and lacks accurate geolocation data, making it difficult for authorities to respond quickly. Citizens do not have an easy and transparent way to report incidents or track the status of their reports. Law enforcement also struggles to identify crime-prone areas due to the absence of real-time geospatial analytics. Kavach aims to bridge this gap by providing a digital platform for instant crime reporting, verification, and automated hotspot detection.

2. Purpose

The purpose of this Software Requirements Specification (SRS) document is to define and describe the complete requirements for the **Kavach Crime Reporting & Hotspot Mapping System**.

The document is intended for:

- Software developers
- UI/UX designers
- QA & Test engineers
- Project managers
- Stakeholders (Admin, Super Admin)

3. Scope

Kavach is a web-based crime reporting and visualization platform designed to allow citizens to:

- Report crimes with geolocation and evidence
- Track their report status
- View verified crimes on an interactive map
- Identify high-risk areas via heatmap clustering

Administrators can:

- Verify or reject reported crimes
- Manage crime hotspots
- Analyze crime data patterns

Super Admin can:

- Add/Remove Admins
- Activate/Deactivate accounts
- View audit logs (actions performed by all users)

The system includes:

- A Next.js frontend
- A Node.js + Express backend
- A MongoDB database
- Geospatial clustering logic for hotspot detection

4. Product Description

a. Product Perspective

I.Kavach is a standalone web application with three primary modules:

- **Citizen portal**
- **Admin dashboard**
- **Super Admin panel**

II.It interacts with:

- MongoDB for persistence
- Leaflet.js for map rendering
- Node.js backend for data processing

III. System architecture:

- Frontend: Next.js (TypeScript)
- Backend: Express.js REST APIs
- Database: MongoDB with geospatial indexing

5. Functional and Non Functional Requirements

a. Functional Requirements

I.User Authentication & Authorization

The system must provide secure registration and login for:

- **Citizens**
- **Administrators**
- **Super Administrators**

II.Enforce Role-Based Access Control (RBAC):

- **Citizen:** Can report and view crimes.
- **Admin:** Can verify/reject reported crimes and view analytics.
- **Super Admin:** Can manage admins (create, update, remove), view all audit logs, and access platform-wide settings.

III.Crime Reporting

Citizens must be able to report a crime with the following details:

- **Crime Type**
- **Description**
- **Location** (via GPS or manual input)
- **Optional Media** (images/videos)

All reports will initially have a status of **“Pending.”**

IV.Crime Verification & Management

Admins can:

- View all pending reports.
- Verify or reject reports.
- Add remarks or reasons during verification.

Super Admins can:

- Monitor admin performance and activity logs.

V.Hotspot Detection & Visualization

The system must:

- Automatically detect hotspots — defined as **≥10 verified crimes within a 1 km radius.**
- Display hotspots on an interactive map (**Leaflet.js**) using **heatmap visualization.**

Filters:

- By crime type
- By date range
- By severity/frequency

VI.Admin Management (Super Admin Feature)

Super Admin can:

- Create new Admin accounts.
- Update or deactivate existing Admins.
- Access and filter audit logs for transparency and accountability.

VII.Audit Logging

Every critical system event must be logged, including:

- Report Creation (Citizen)
- Verification or Rejection (Admin)
- Admin Management Actions (Super Admin)

Log Fields include:

- **Actor:** User ID
- **Action:** Performed activity
- **Target Entity**
- **Timestamp**
- **Status/Remarks**

b. Non-Functional Requirements

I.Performance & Scalability

The system must:

- Support **1000+ concurrent users** without performance degradation.
- Ensure **geospatial clustering queries** complete within **3 seconds.**

- Enable the **backend and database** to be **horizontally scalable** on cloud infrastructure (**AWS / GCP**).

II.Security

The system must implement the following security measures:

- Use **bcrypt** for password hashing.
- Use **JWT tokens** for session management.
- Enforce **role-based authorization** for API protection.
- Require **HTTPS** for all endpoints.
- Ensure **audit logs are immutable (append-only).**

6. User Stories

Citizen

- As a **Citizen**, I want to **register/login securely**, so that I can access my account safely.
- As a **Citizen**, I want to **report a crime** with a description, type, and location.
- As a **Citizen**, I want to **upload images or videos** to strengthen my report.
- As a **Citizen**, I want to **track my report statuses** (Pending, Verified, Rejected).
- As a **Citizen**, I want to **view verified crimes and hotspots** near my area.

Admin

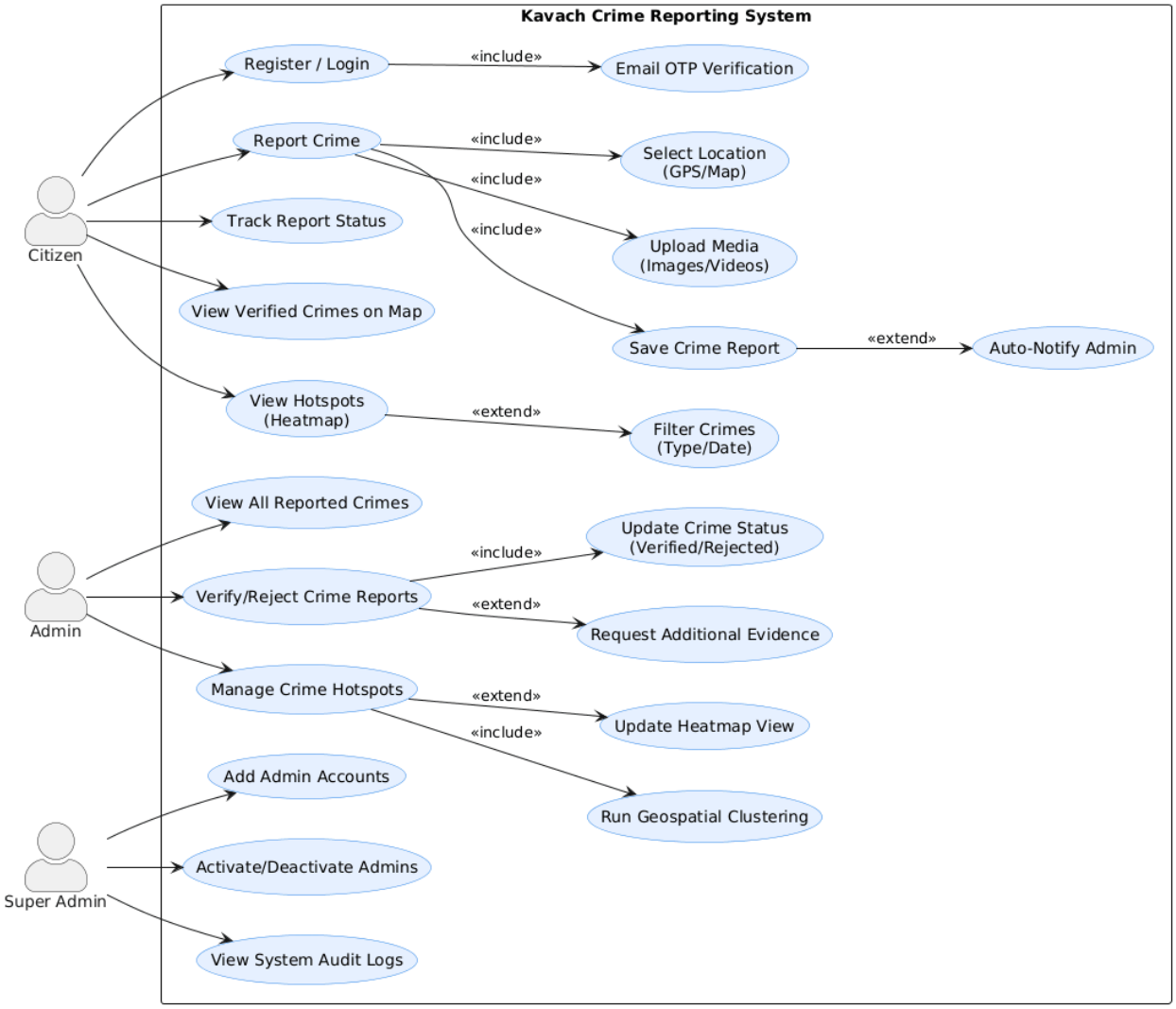
- As an **Admin**, I want to **view pending reports**, so that I can verify them quickly.
- As an **Admin**, I want to **approve/reject reports** with proper remarks.

- As an **Admin**, I want to **see real-time heatmaps**, so that I can identify critical areas.
- As an **Admin**, I want to **access analytics dashboards** for decision-making.

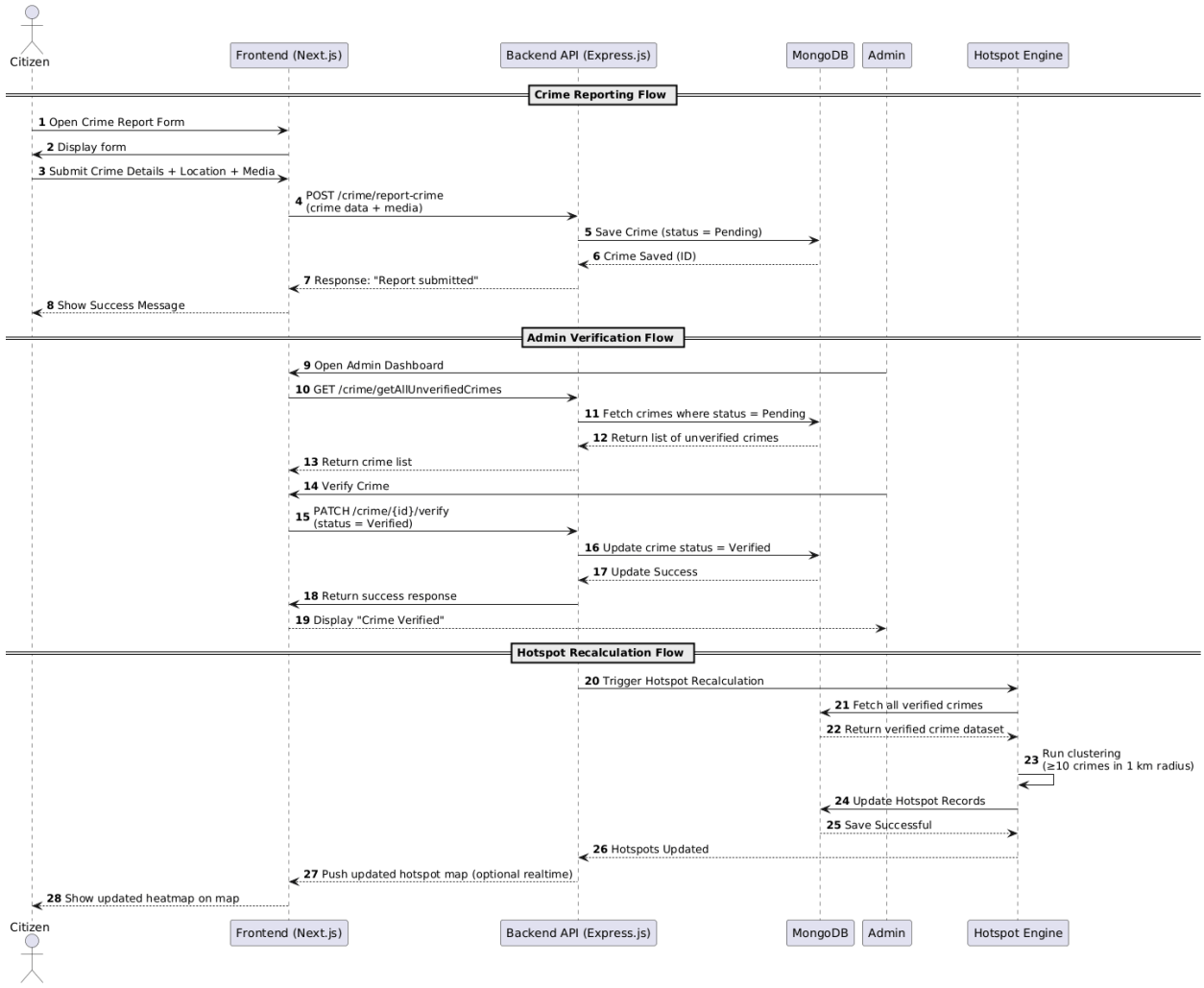
Super Admin

- As a **Super Admin**, I want to **create/manage admins** for different zones.
- As a **Super Admin**, I want to **view all audit logs** to ensure accountability.
- As a **Super Admin**, I want to **deactivate inactive or malicious admins.**
- As a **Super Admin**, I want to **oversee all system operations** from one dashboard.

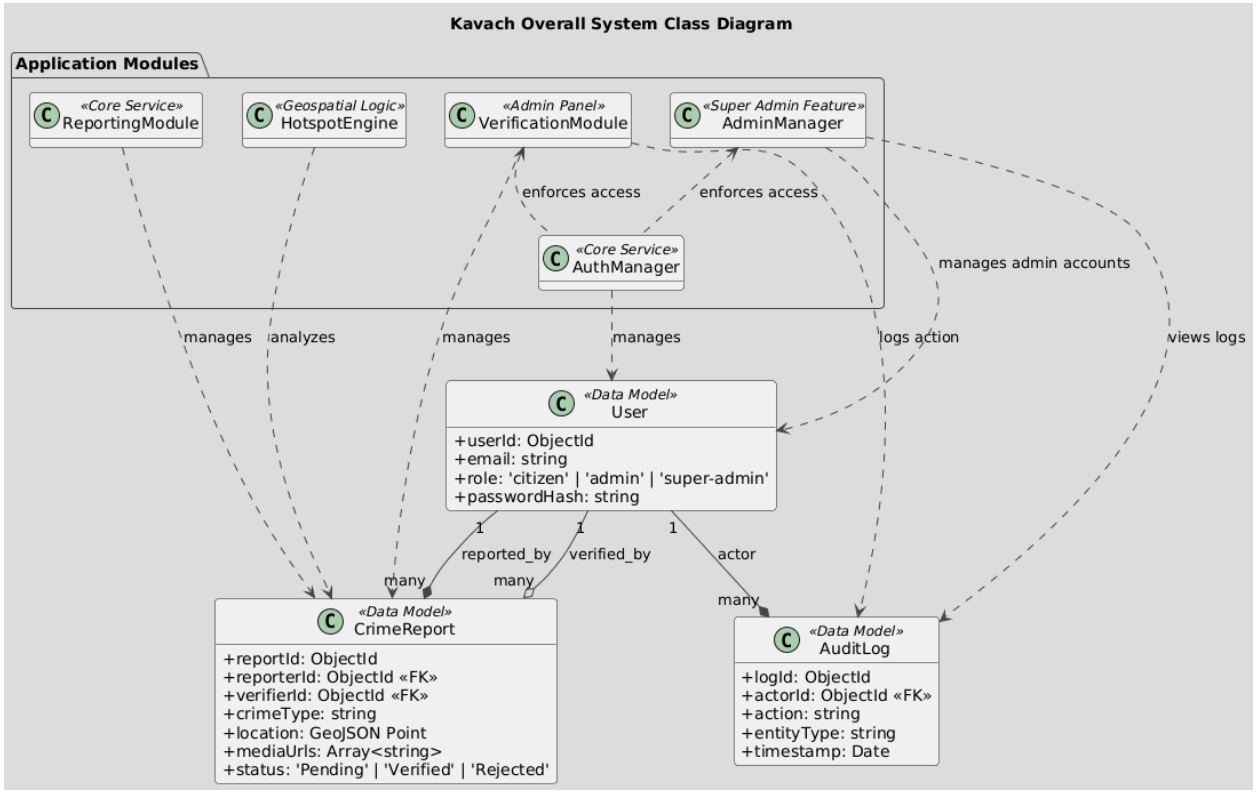
7. Use Case Diagram



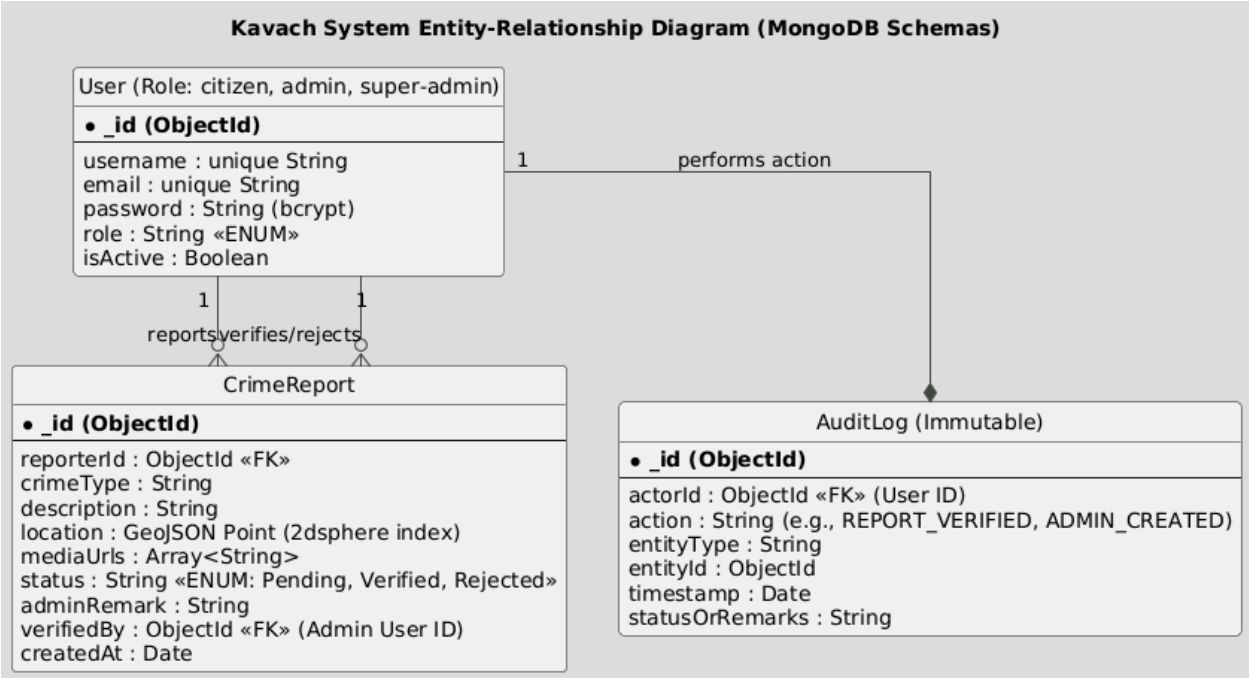
8. Sequence Diagram



9. Class Diagram



10. ER Diagram



11. Acceptance Criteria

Authentication & Authorization

- Citizens, Admins, and Super Admins can register/login securely.
- Passwords hashed using bcrypt.
- JWT tokens generated and verified on every protected route.
- Role-based access enforced for every endpoint.
- Unauthorized access denied.

Crime Reporting

- Citizens can submit crime reports with media and location.
- Crime reports saved in MongoDB with geospatial data.
- Status updates reflected in user dashboard.

Verification Workflow

- Admins can verify/reject reports.
- Every verification/rejection triggers an **audit log**.
- Verified reports appear publicly on the map.

Hotspot Detection

- Hotspots identified for >10 crimes within 1 km.
- Clusters auto-refresh when new crimes are verified.
- Map interaction time < 3 seconds.

Super Admin Features

- Super Admin can manage Admin accounts.
- Audit logs record all create/update/delete actions.
- Logs cannot be edited or deleted manually.