SafeKarachi

Kaavish Report
presented to the academic faculty
by

Jibran Sheikh	js08312
Wania Hussain	sh08450
Daniyal Farooqui	mf08320
Ghulam Mustafa	gm08291
Ikhlas Ahmad	ia08308

$\in -sicludegraphics[width = .4]Images/logo.png$

In partial fulfillment of the requirements for
Bachelor of Science
Computer Science

Dhanani School of Science and Engineering

Habib University
Fall 2025

SafeKarachi

This Kaavish project was supervised by:

My Internal Supervisor
Faculty of Computer Science
Habib University

Approved by the Faculty of Computer Science on ______________.

1. Introduction

1.1 Problem Statement

Public safety and urban mobility have become significant challenges in Karachi due to increasing incidents such as theft, harassment, and accidents. Commuters often lack reliable information about which routes or areas are safer at different times of the day. Existing navigation systems focus mainly on the shortest or fastest route but do not consider safety factors. This lack of safety-aware routing and localized reporting leads to anxiety, delays, and sometimes harm for citizens traveling within the city. Our project aims to address this gap by providing a community-based data-driven platform that promotes safer travel decisions and community awareness

1.2 Proposed Solution

To address the above problem, we propose the development of a mobile application called SafeKarachi. The system will allow users to report safety-related incidents, view community updates, and find optimal travel routes that consider both distance and safety. Using aggregated data from reports and public sources, the system will generate a safety score using predictive analysis for each area and recommend the safest route accordingly. The application will also include features such as emergency contacts and community groups. This solution aims to empower citizens with real time, localized safety information and promote safer urban mobility

1.3 Project Gantt Chart and Deliverables

This section outlines the project timeline divided into two phases — Kaavish I and Kaavish II — along with the major deliverables and individual responsibilities.

1.3.1 Kaavish I Deliverables (Weeks 1–8)

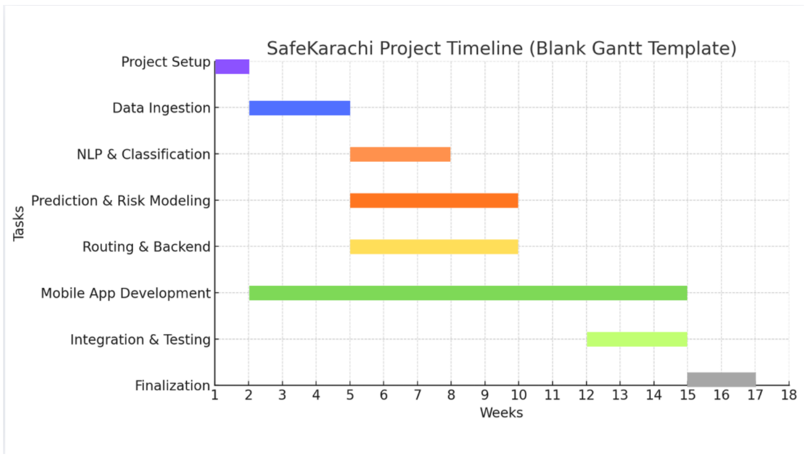


Figure 1.1: Gant chart

This phase focuses on building the system foundation, data pipelines, and initial models.

- **D1.1: Repository Setup** – GitHub setup with version control and CI/CD.
- **D1.2: Data Collection Pipeline** – Social media and news data ingestion for analysis.
- **D1.3: NLP Module** – Multilingual text processing for event and location extraction.
- **D1.4: Credibility Engine** – Basic verification of reports and duplicate detection.
- **D1.5: Prediction Model (v1)** – Initial hotspot prediction using past data.
- **D1.6: Routing Prototype** – Basic map routing integrated with backend.
- **D1.7: Admin Dashboard (Alpha)** – Early version of the web dashboard for data review.

1.3.2 Kaavish II Deliverables (Weeks 9–18)

This phase focuses on integration, optimization, and the final deployment.

- **D2.1: API Gateway** – RESTful APIs for mobile and web access.
- **D2.2: Mobile App (MVP)** – Map view, incident reporting, and basic alerts.
- **D2.3: Safety Routing API** – Safer route suggestion using model predictions.
- **D2.4: Notification System** – Real-time alerts based on nearby incidents.
- **D2.5: Community Module** – User posts and area-based discussions.
- **D2.6: Testing & Deployment** – System testing and deployment on cloud.
- **D2.7: Final Report & Demo** – Final documentation and live demonstration.

1.3.3 Resource Allocation

The following outlines each member's primary responsibilities within the project:

- **Ghulam Mustafa** : Backend and API development using Spring Boot, REST design.
- **Jibran Sheikh** : Mobile app and UX development in React Native. Implementation of maps, alerts, and push notifications.
- **Daniyal Farooqui** : NLP pipeline design for classification and entity extraction and Routing Algorithm.
- **Ikhlas Ahmed** : Machine learning for hotspot prediction.
- **Syeda Wania** : Data quality and NLP pipeline.

2. Software Requirement Specification (SRS)

This chapter provides detailed specifications of the system under development.

2.1 Functional Requirements

This section describes each function/feature provided by our system. These functions are logically grouped into modules based on their purpose and mode of operation.

- **Module 1: Community Engagement**
  - **FR 1.1 (Community Membership):** The system shall allow users to browse and join specific, area-based communities (e.g., Gulshan, Saddar).
  - **FR 1.2 (Community Feed):** The system shall display a real-time feed of posts and updates from the communities the user has joined.
- **Module 2: Safe Routing & Navigation**
  - **FR 2.1 (Route Input):** The system shall provide an interface for users to select a starting location (defaulting to current GPS location) and a destination for route planning.
  - **FR 2.2 (Destination Search):** The system shall display predictive search suggestions as the user types a destination address or landmark.
  - **FR 2.3 (Route Options):** The system shall calculate and display multiple route options to the destination.
  - **FR 2.4 (Safest Route Highlight):** The system shall analyze all route options based on aggregated safety data and clearly highlight the recommended "Safest Route".

  - **FR 2.5 (Predictive Hotspot Integration):** The system shall incorporate predictive hotspot analysis in the backend to evaluate safety scores for each route segment and use these in determining the safest route.

- **Module 3: Incident Reporting**
  - **FR 3.1 (Report Initiation):** The system shall provide a persistent, high-visibility option to allow users to quickly initiate an incident report.
  - **FR 3.2 (Location Selection):** Upon report initiation, the system shall present a map interface allowing the user to precisely place a pin (e.g., via drag-and-drop) at the incident's location.
  - **FR 3.3 (Report Details Form):** After location confirmation, the system shall display a form prompting the user to submit key details, including:
    - Incident Category (e.g., Snatching, Harassment, Roadblock)
    - Time of Occurrence
    - Optional text field for additional information.
  - **FR 3.4 (Report Verification):** The system shall process submitted reports through an automated verification pipeline before publishing them to the community feed.

- **Module 4: Settings & Emergency Access**
  - **FR 4.1 (Notification Control):** The system shall provide a settings menu where users can globally enable or disable all push notifications.
  - **FR 4.2 (Emergency Access):** The system shall include a dedicated feature within the settings or main menu to provide quick-dial access to official emergency numbers (e.g., Police, Ambulance).

- **Module 5: User Profile & Customization**
  - **FR 5.1 (Frequent Destinations):** The system shall allow users to save, label, and manage frequent destinations (e.g., "Home," "University") within their user profile for faster route planning.
  - **FR 5.2 (Saved Daily Routes):** The system shall enable users to save specific daily routes (e.g., "Home to Work") and associate them with typical travel times, enabling proactive alerts.

- **Module 6: Administrator Dashboard**
  - **FR 6.1 (Report Moderation):** The system shall provide an administrative interface to review, approve, edit, or delete user-submitted incident reports that are flagged by the verification pipeline.
  - **FR 6.2 (Analytics Dashboard):** The system shall provide an administrative dashboard to visualize key metrics and trends, such as incident hotspots, report volume, and user activity.
- **Module 7: Data Aggregation and Processing Pipeline**
  - **FR 7.1 (External Data Ingestion):** The system shall automatically ingest incident-related data from pre-configured external sources (e.g., social media feeds, news APIs).
  - **FR 7.2 (NLP Text Processing):** The system shall use an NLP pipeline to automatically process ingested text data to classify the incident type and extract key entities (e.g., location, time).
  - **FR 7.3 (Credibility Scoring):** The system shall automatically assign a credibility score to each aggregated report based on source reliability, user verifications, and cross-referencing.
- **Module 8: User Account Management**
  - **FR 8.1 (User Registration):** The system shall allow a new user to register for an account.
  - **FR 8.2 (User Login):** The system shall provide an interface for a registered user to securely log in.

2.2 Non-functional Requirements

This section outlines the non-functional requirements that define the quality attributes of the SafeKarachi system. Each requirement has been categorized according to the system's operational and design constraints.

2.2.1 Performance

- **Response Time:** The system shall be optimized for responsive query times for core functions like routing and data retrieval.

- **Availability:** The system shall be designed for high availability.

2.2.2 Security

- **Authentication:** The system shall implement a secure user authentication mechanism to help prevent unauthorized access.
- **Data Encryption:** The system shall employ standard encryption practices for sensitive user data, both in transit and at rest.

2.2.3 Reliability

- **Fault Tolerance:** The system shall be designed with fault tolerance in mind to minimize the impact of individual component failures.

2.2.4 Maintainability

- **Modular Design:** The system shall be developed using a modular architecture to facilitate independent updates and maintenance.
- **Documentation:** Internal APIs and models shall be documented to aid future development.
- **Code Reusability:** The system design shall promote code reusability where practical.

2.2.5 Usability

- **Mobile-First Design:** The application interface shall be designed with a mobile-first approach, targeting Android and iOS devices.
- **Accessibility:** The interface shall be designed for ease of use, using clear and simple visuals.

2.2.6 Interoperability

- **Integration:** The system shall be capable of interfacing with specified third-party map and data APIs.

- **Data Formats:** The system shall utilize standard data formats (e.g., JSON/-GeoJSON) for internal and external communication.
- **Cross-Platform:** The mobile application shall aim for a consistent user experience across supported platforms.

2.2.7 Scalability

- **System Scalability:** The system's architecture shall be designed to support future growth in user load and data volume.
- **Database Scalability:** The chosen database technologies shall support standard scaling mechanisms.

2.3 External Interfaces

2.3.1 User Interfaces

Below are sample mock up screens for both the User and the Admin:

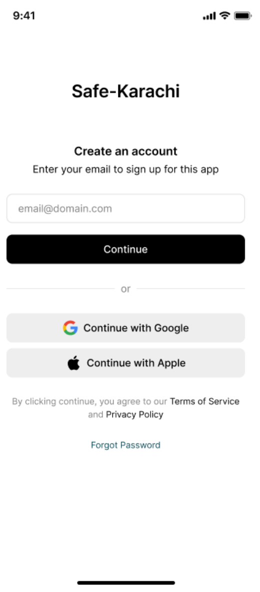

9:41
Safe-Karachi
Create an account
Enter your email to sign up for this app
email@domain.com
Continue
or
Continue with Google
Continue with Apple
By clicking continue, you agree to our Terms of Service and Privacy Policy
Forgot Password

Figure 2.1: User sign in screen

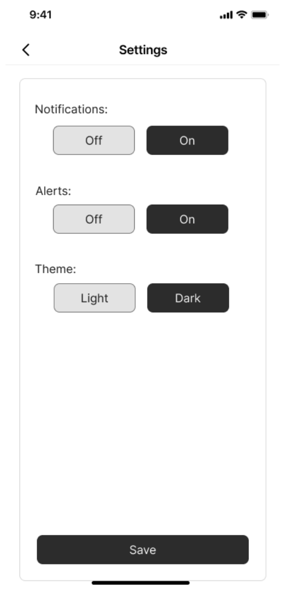


Figure 2.2: User settings screen

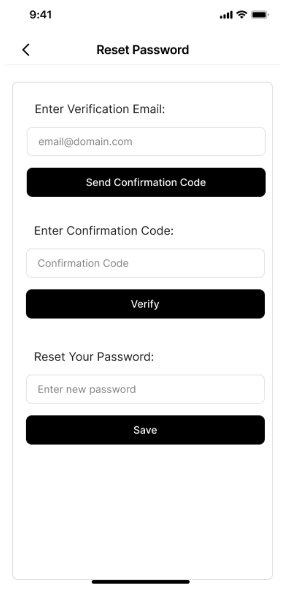
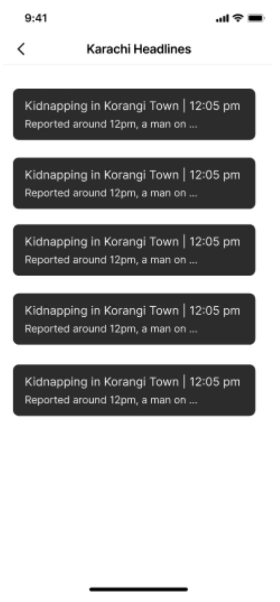


Figure 2.3: User Reset password screen



Kidnapping in Korangi Town | 12:05 pm
Reported around 12pm, a man on ...

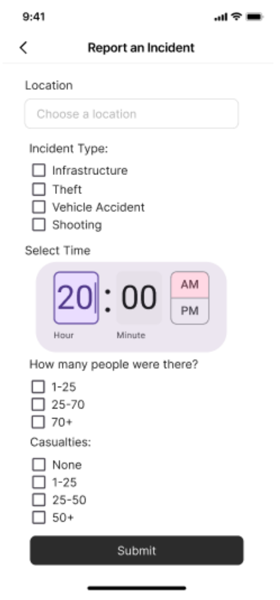
Kidnapping in Korangi Town | 12:05 pm
Reported around 12pm, a man on ...

Kidnapping in Korangi Town | 12:05 pm
Reported around 12pm, a man on ...

Kidnapping in Korangi Town | 12:05 pm
Reported around 12pm, a man on ...

Kidnapping in Korangi Town | 12:05 pm
Reported around 12pm, a man on ...

Figure 2.4: User Main headlines screen



Location

Choose a location

Incident Type:

- [ ] Infrastructure
- [ ] Theft
- [ ] Vehicle Accident
- [ ] Shooting

Select Time

20 : 00 AM PM

Hour Minute

How many people were there?

- [ ] 1-25
- [ ] 25-70
- [ ] 70+

Casualties:

- [ ] None
- [ ] 1-25
- [ ] 25-50
- [ ] 50+

Figure 2.5: Incident reporting screen

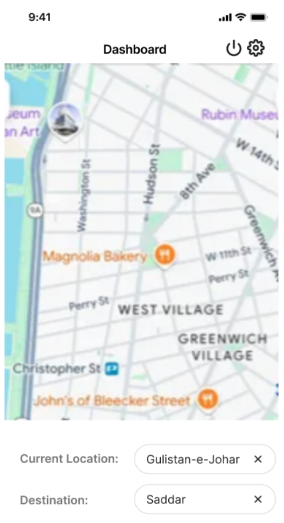


Figure 2.6: User dashboard screen

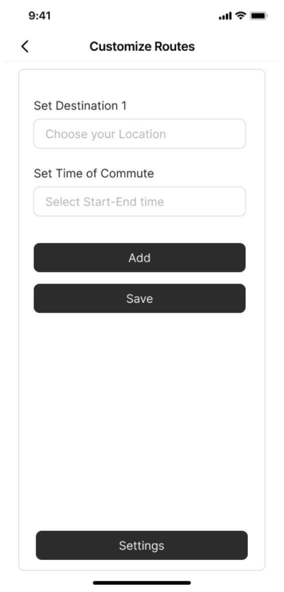


Figure 2.7: User Customize routes screen

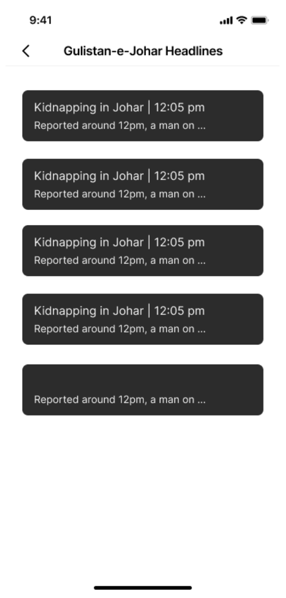

9:41
Gulistan-e-Johar Headlines
Kidnapping in Johar | 12:05 pm
Reported around 12pm, a man on ...
Kidnapping in Johar | 12:05 pm
Reported around 12pm, a man on ...
Kidnapping in Johar | 12:05 pm
Reported around 12pm, a man on ...
Kidnapping in Johar | 12:05 pm
Reported around 12pm, a man on ...
Reported around 12pm, a man on ...

Figure 2.8: community headlines screen

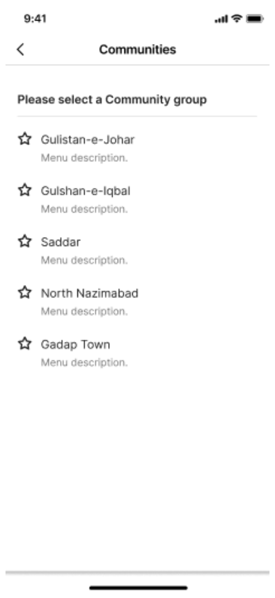


Figure 2.9: User communities screen

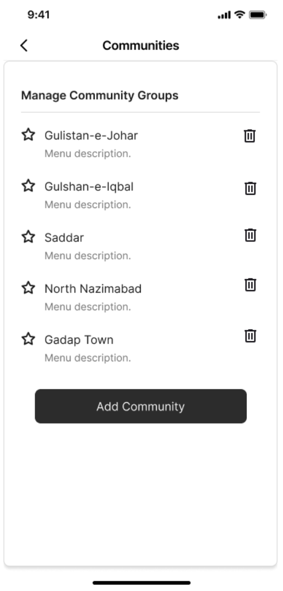


Figure 2.10: Admin manage communities screen

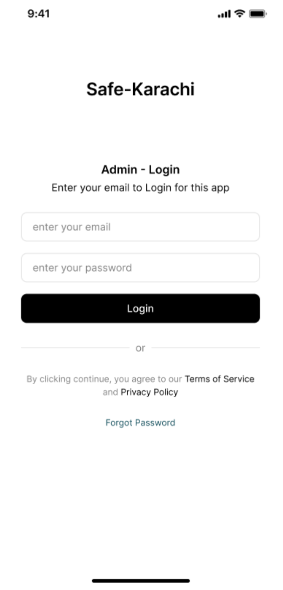

9:41
Safe-Karachi
Admin - Login
Enter your email to Login for this app
enter your email
enter your password
Login
or
By clicking continue, you agree to our Terms of Service and Privacy Policy
Forgot Password

Figure 2.11: Admin sign in screen

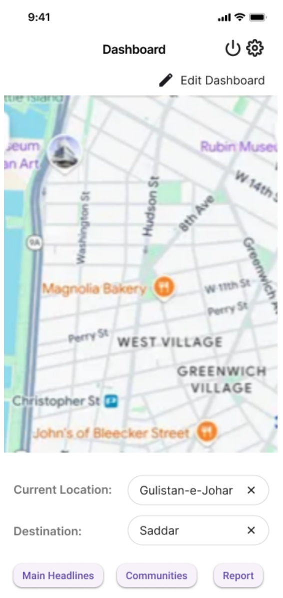


Figure 2.12: Admin dashboard screen

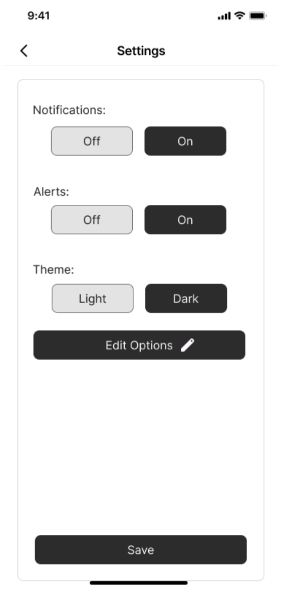


Figure 2.13: Admin settings screen

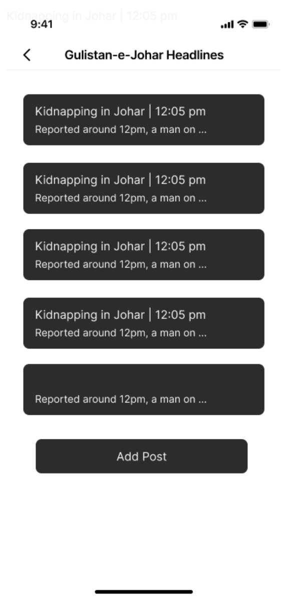

9:41
Gulistan-e-Johar Headlines
Kidnapping in Johar | 12:05 pm
Reported around 12pm, a man on ...
Kidnapping in Johar | 12:05 pm
Reported around 12pm, a man on ...
Kidnapping in Johar | 12:05 pm
Reported around 12pm, a man on ...
Kidnapping in Johar | 12:05 pm
Reported around 12pm, a man on ...
Reported around 12pm, a man on ...
Add Post

Figure 2.14: Admin manage communities headlines screen

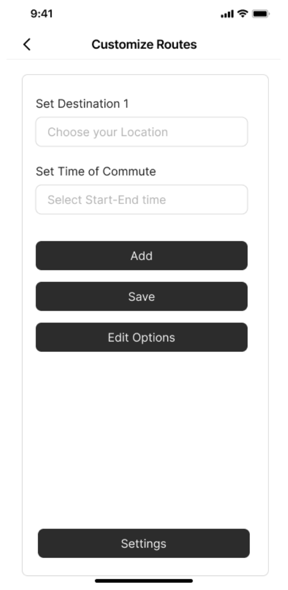


Figure 2.15: Admin customize routes screen

2.3.2 Application Program Interface (API)

This section outlines the internal and external APIs that enable communication between SafeKarachi system components. All APIs use standard web protocols for data exchange.

Internal APIs

The backend provides a set of RESTful APIs for core functionality.

User Management Handles registration, authentication, and profile operations. Supports both registered accounts and limited anonymous access for reporting.

Incident Reporting Allows submission of crime reports with location details, media attachments, and metadata. Includes verification mechanisms for data quality.

Safe Routing Provides route calculation with safety-weighted algorithms, returning multiple path options and predicted risk levels for each segment.

Community Features Manages area-based community groups and real-time feed updates.

Administration Enables content moderation, report review, and analytics dashboard access for authorized personnel.

External APIs

Mapping Services Integrates with OSM/OSRM for primary routing and Mapbox for mobile map rendering. Google Maps serves as a fallback option.

Data Sources Pulls incident information from news aggregators and social media platforms through official APIs.

ML Infrastructure Uses external services for multilingual text processing and model versioning.

2.3.3 Hardware/Communication Interfaces

This section outlines the planned hardware and network infrastructure for SafeKarachi, covering client devices, server components, and external communication channels.

Client-Side Hardware

The mobile application will run on standard consumer smartphones with GPS capability and internet connectivity.

For administrators, any modern computer with a web browser will be able to access the management portal.

Server Infrastructure

Compute Resources The backend will deploy on cloud virtual machines with auto-scaling capabilities. Standard instances will handle API services, while slightly larger instances will support ML processing workloads. Database servers will use memory-optimized configurations for spatial query performance.

External Communication

Location Services Mobile devices will obtain location data through built-in GPS chips, assisted by cellular networks when available. Wi-Fi positioning will serve as a backup in areas with poor satellite reception.

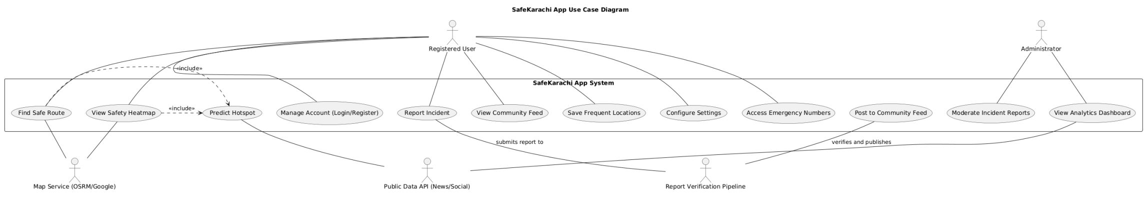
Emergency Services The system will interface with local emergency response systems through secure webhooks for critical incident escalation.

Data Sources External news and social media platforms will be accessed through their public APIs, using standard authentication methods and respecting rate limits.

Scalability

The system will use scalable infrastructure to handle growth in users and data volume, ensuring consistent performance.

2.4 Use Cases



Use Case diagram for SafeKarachi

2.5 System Diagram

This diagram gives a high-level view of the different components of our system and their interactions. Each component and the particular tools, technologies, or libraries used to build it are described here.

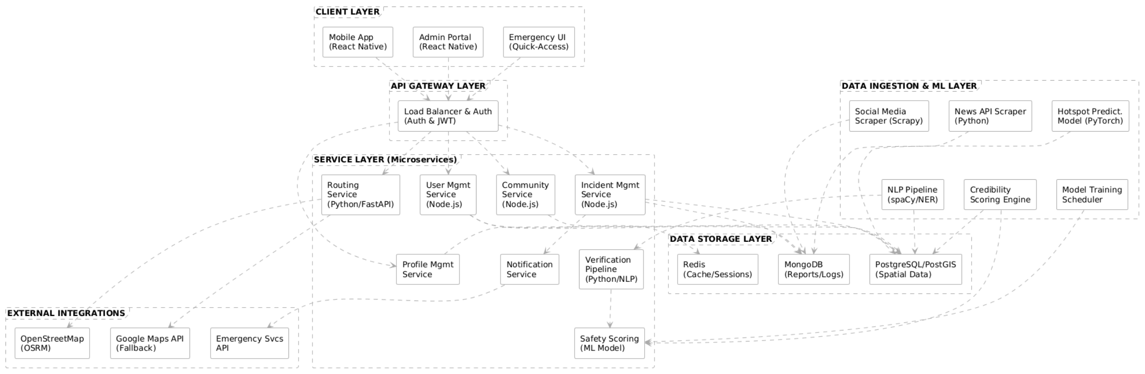

CLIENT LAYER
Mobile App (React Native)
Admin Portal (React Native)
Emergency UI (Quick-Access)
API GATEWAY LAYER
Load Balancer & Auth (Auth & JWT)
DATA INGESTION & ML LAYER
Social Media Scraper (Scrapy)
News API Scraper (Python)
Hotspot Predict. Model (PyTorch)
NLP Pipeline (spaCy/NER)
Credibility Scoring Engine
Model Training Scheduler
SERVICE LAYER (Microservices)
Routing Service (Python/FastAPI)
User Mgmt Service (Node.js)
Community Service (Node.js)
Incident Mgmt Service (Node.js)
Profile Mgmt Service
Notification Service
Verification Pipeline (Python/NLP)
Safety Scoring (ML Model)
DATA STORAGE LAYER
Redis (Cache/Sessions)
MongoDB (Reports/Logs)
PostgreSQL/PostGIS (Spatial Data)
EXTERNAL INTEGRATIONS
OpenStreetMap (OSRM)
Google Maps API (Fallback)
Emergency Svcs API

Figure 2.16: High level system diagram