AAGAH

Software Design (SD)

<https://github.com/maazstering/AAGAH>

***BY***

***Team Lead: Danial Zahid 22771***

***Member(s):***

***Laiba Shah 23003***

***Muhammad Maaz Siddiqui 22997***

***Danial Merchant 22775***

***Shehzad Khuwaja 22852***

# ***PROJECTS COMMITTEE (PC)***

***SUPERVISOR: Dr Muhammad Saeed***

***CO-SUPERVISOR: Waleed Arshad***

*SUBMITTED TO*

**PROJECTS Manager – FYP**

*ON*

DATE (22/01/2024)

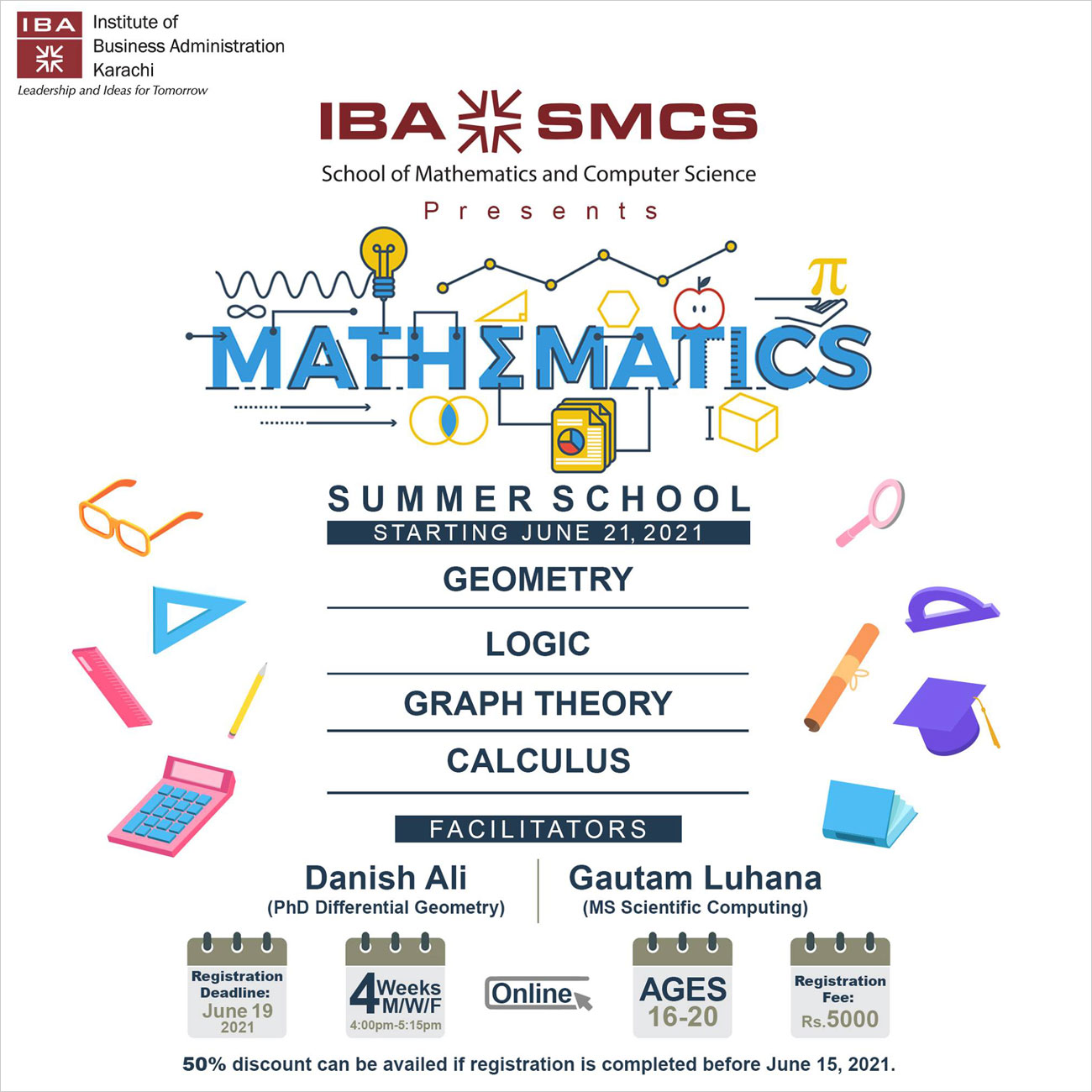


Table of Contents

[***PROJECTS COMMITTEE (PC)*** 1](#_Toc95986055)

[1. Interaction Diagrams 3](#_Toc95986056)

[2. Class Diagram and Interface Specification 3](#_Toc95986057)

[3. System Architecture and System Design 3](#_Toc95986058)

[3.1. Architectural Style 3](#_Toc95986059)

[3.2. Identifying Subsystems 3](#_Toc95986060)

[3.3. Mapping Subsystems to Hardware 3](#_Toc95986061)

[3.4. Persistent Data Storage 3](#_Toc95986062)

[3.5. Network Protocol 3](#_Toc95986063)

[3.6. Global Control Flow 3](#_Toc95986064)

[3.7. Hardware/Sofware Requirements 3](#_Toc95986065)

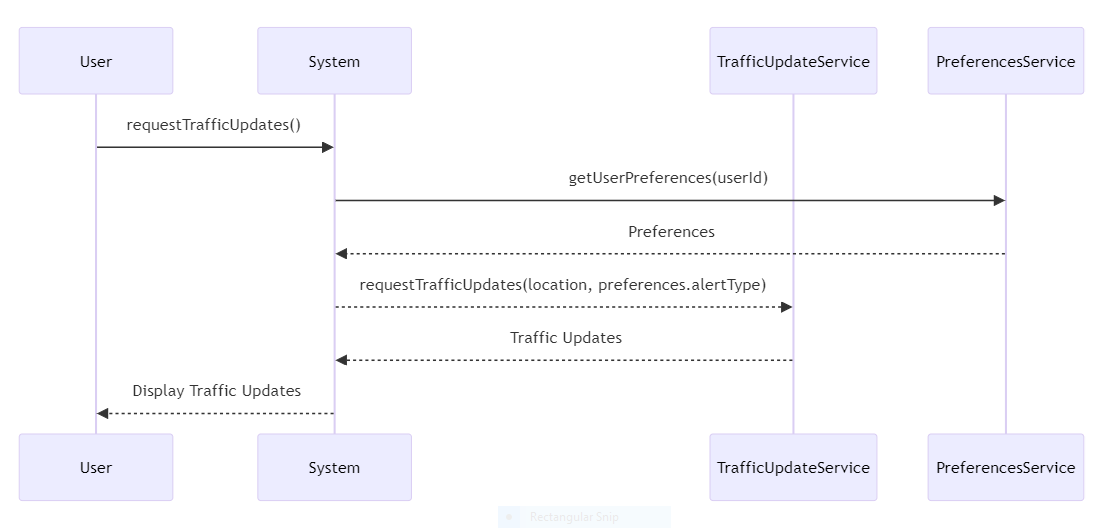
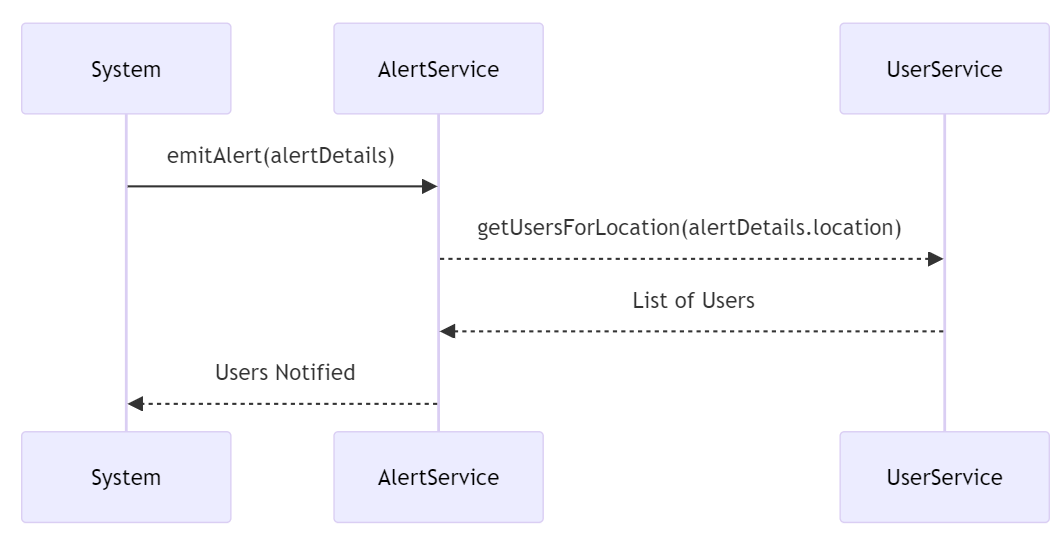
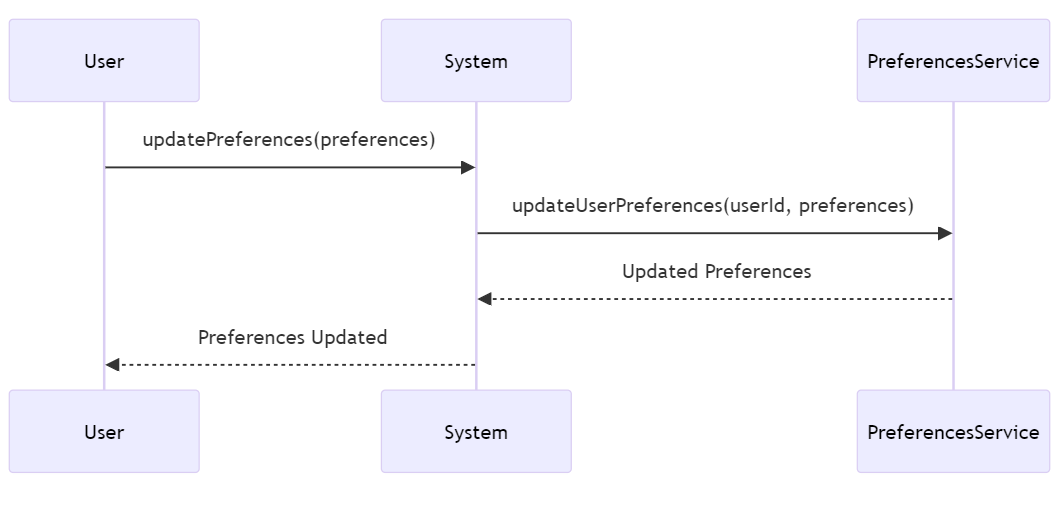
[4. Algorithms and Data Structures 4](#_Toc95986066)

[5. User Interface Design and Implementation 4](#_Toc95986067)

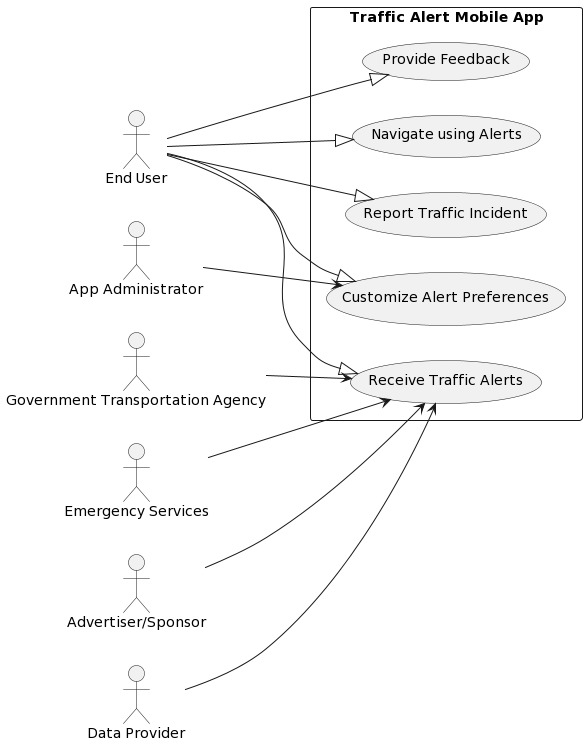
[6. Design of Tests 4](#_Toc95986068)

System Design

# Interaction Diagrams

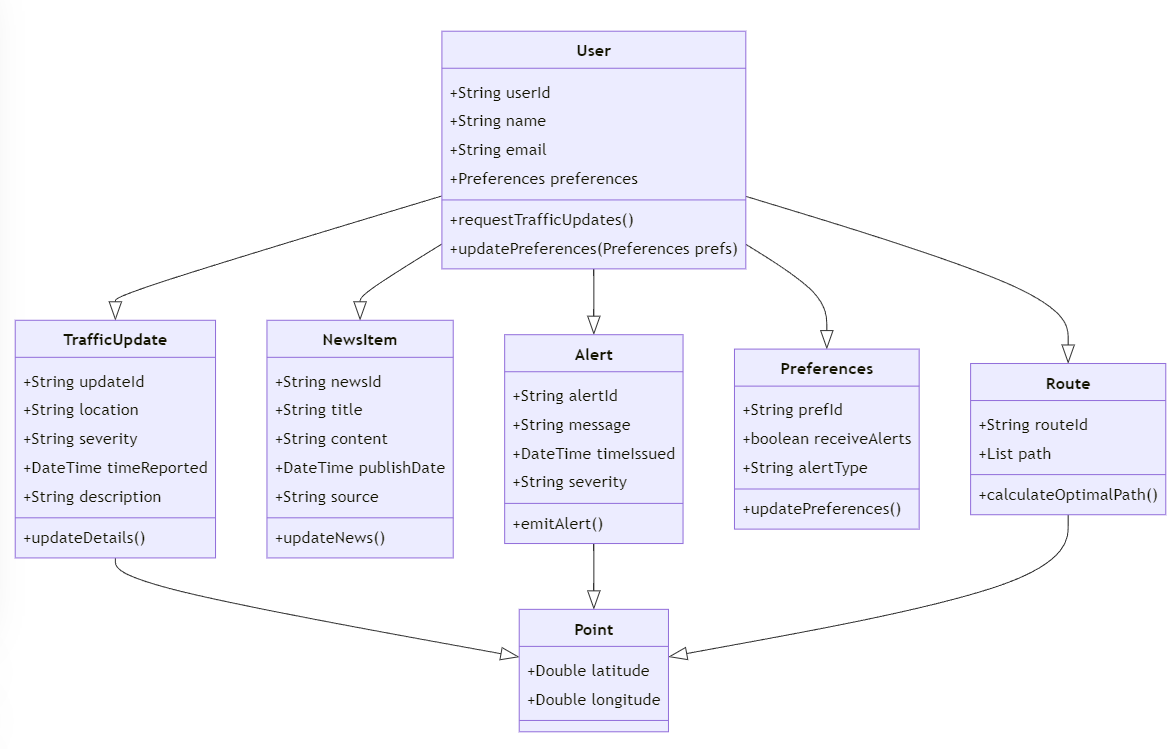
**Sequence diagram 1****Sequence diagram 2****Sequence diagram 3**

Use Case diagram



# Class Diagram and Interface Specification

**Class diagram**

****

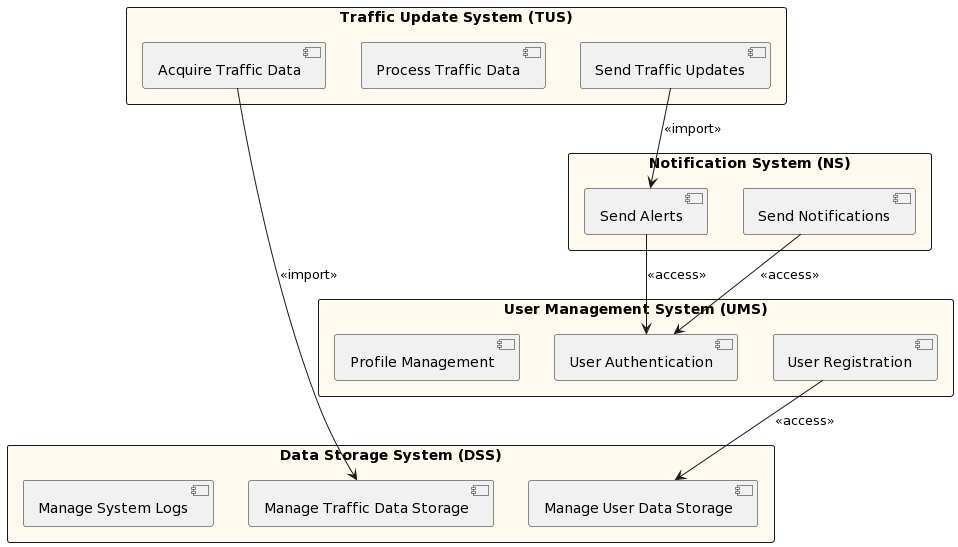
# System Architecture and System Design

## Architectural Style: Client-Server

* The client will handle user interface and user interactions.
* The server will handle traffic data processing, user management, and alert generation.

## Identifying Subsystems

* Traffic Update System (TUS): Responsible for acquiring, processing, and sending real-time traffic updates.
* User Management System (UMS): Handles user registration, authentication, and profile management.
* Notification System (NS): Manages sending alerts and notifications to users based on their preferences and current traffic conditions.
* Data Storage System (DSS): Manages persistent storage of traffic data, user data, and system logs.



## Mapping Subsystems to Hardware

* **Server Components:**

1. **Traffic Update System & Data Storage System:** Hosted on a high-performance server with robust storage solutions to handle large volumes of traffic data.
2. **User Management System & Notification System:** Hosted on secured servers, ensuring user data integrity and secure communication.

* **Client Components:**

1. The client application running on user's mobile device, handling user interface and interactions.

## Persistent Data Storage

* **Database Schema**

1. **Users Table:** Stores user information including IDs, preferences, and contact details.
2. **Traffic Table**: Holds real-time and historical traffic data.
3. **Alerts Table:** Logs of alerts sent to users, including timestamps and user responses.

* **Technology**

Utilizing a SQL-based database for structured data requirements, ensuring ACID properties and reliable transactions.

## Network Protocol

* **HTTP/HTTPS:** For secure communication between clients and server.
* **WebSockets:** For real-time communication, especially in sending instant traffic updates and alerts.

## Global Control Flow

* **Event-driven Architecture:** The system will be heavily event-driven, particularly in handling incoming traffic data and user interactions.
* **Concurrency Handling:** Implementing multi-threading or asynchronous processing to handle multiple user requests and data streams effectively.
* **Time Dependency and Scheduling:** Utilizing cron jobs or similar scheduling for routine tasks like database backups or traffic data updates.

## Hardware/Software Requirements

* **Client Requirements:**
* Minimum OS: Android 8.0 or iOS 12
* RAM: Minimum 2 GB
* Storage: 100 MB free space
* **Server Requirements:**
* OS: Linux-based OS for better security and performance.
* Processor: Intel Xeon or equivalent with at least 8 cores.
* RAM: Minimum 16 GB.
* Storage: At least 1 TB, with solid-state drives for faster data access.
* Network: High-speed internet connection with at least 1 Gbps speed for seamless data handling.

# Algorithms and Data Structures

**4.1** **Traffic Alert Generation Algorithm:**

* Objective: To analyze real-time traffic data, predict potential delays, and notify users with the best possible routes.
* Input: Real-time traffic data, historical traffic patterns, user's current location, destination, and preferences.
* Process:
* Data Collection: Gather real-time traffic information from various sources like sensors, cameras, or user reports.
* Data Analysis & Prediction: Employ machine learning models or statistical algorithms to predict traffic conditions, considering historical trends and current data.
* Route Optimization: Calculate the best possible routes from the current location to the destination, considering the predicted traffic conditions.
* Alert Generation: Determine if and when a user should be alerted based on their preferences and the severity of traffic conditions.
* Output: Traffic alerts with suggested routes and estimated times of arrival.
* Data Structures Used:
* Graphs: To represent the network of roads and intersections.
* Priority Queues: To manage the alerts based on their severity and timing.
* Trees: Particularly, balanced search trees to quickly update and retrieve traffic conditions for specific routes.

**4.2** **User Preference Algorithm:**

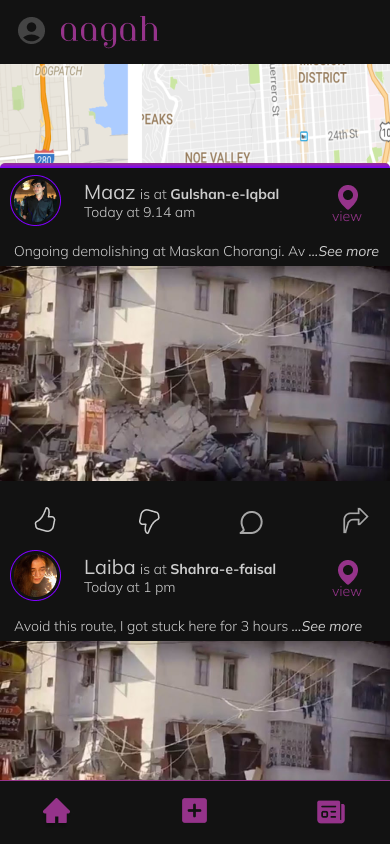
* Objective: To tailor traffic alerts and suggestions based on individual user settings and historical responses.
* Input: User's saved preferences, historical response data to alerts.
* Process:
* Preference Aggregation: Compile the user's settings regarding types of alerts, preferred travel times, and routes.
* Historical Analysis: Analyze the user's past responses to alerts to understand their preferences better.
* Customization: Adjust alert generation and routing suggestions based on the aggregated preferences and historical analysis.
* Output: Customized traffic alerts and routes for the user.
* Data Structures Used:
* Hash Tables: For quick access and storage of user preferences and historical data.
* Arrays: To hold temporary data about routes and preferences during processing.

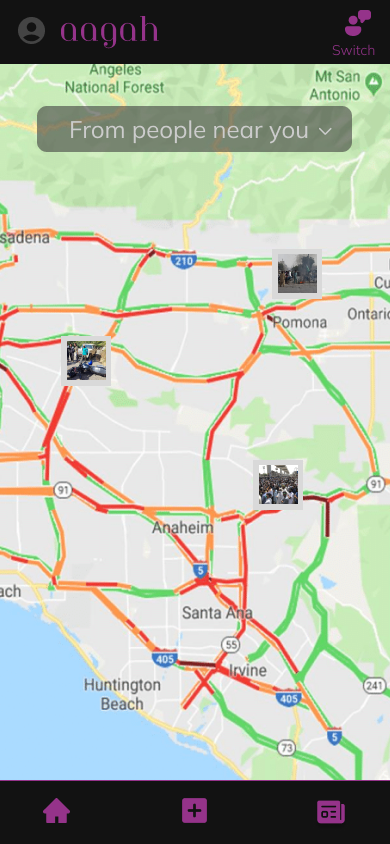
**4.3 Data Storage and Retrieval Algorithm:**

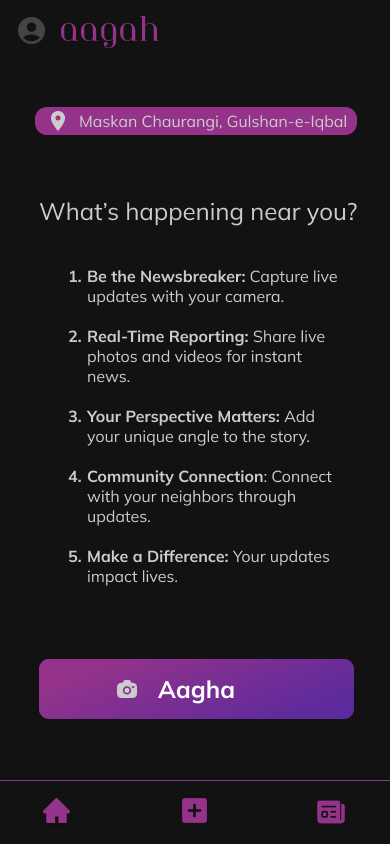
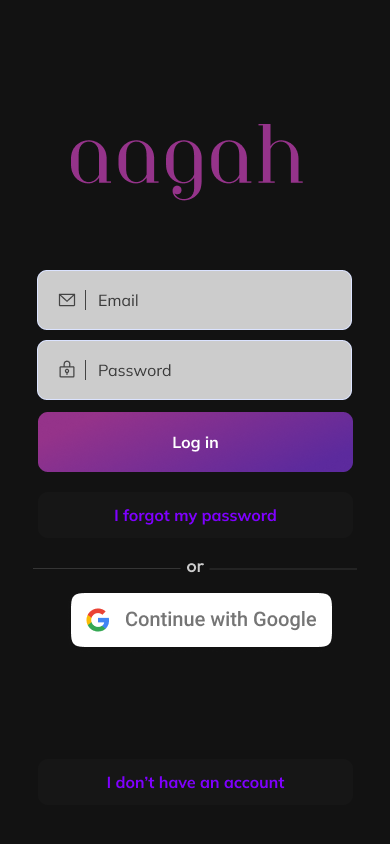
* Objective: To efficiently store and retrieve traffic data, user information, and system logs.
* Input: Traffic data, user data, system logs.
* Process:
* Indexing: Implement indexing on databases to speed up the retrieval of traffic data and user information.
* Caching: Employ caching strategies for frequently accessed data to reduce latency.
* Data Compression: Use data compression techniques for efficient storage of large volumes of traffic data and logs.
* Output: Quick and efficient data storage and retrieval system.
* Data Structures Used:
* B-Trees: Used in database indexing to allow quick data retrieval.
* Cache: Implementing LRU (Least Recently Used) or other caching strategies for frequently accessed data.

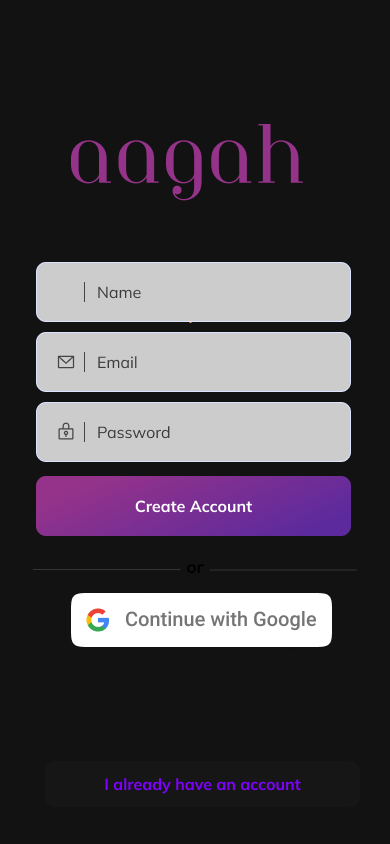
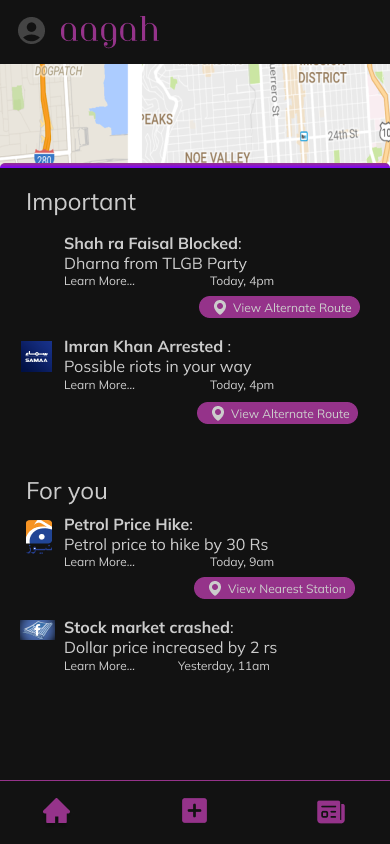
# User Interface Design and Implementation

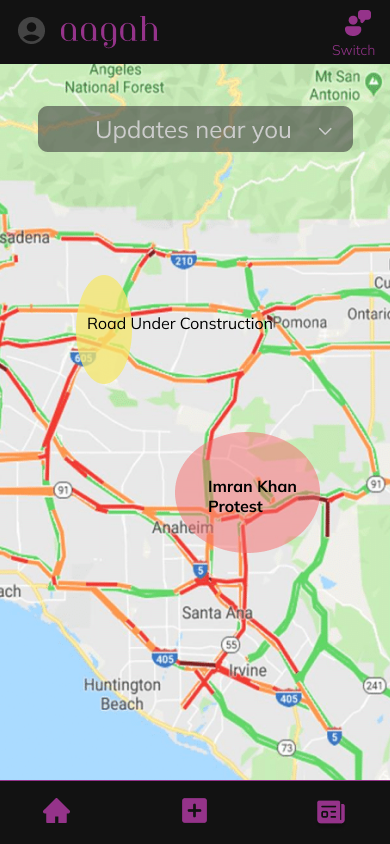
The UI overview of the designs is showcased below.











# Design of Tests

**6.1 Traffic Alert Accuracy Test:**

* Objective: To ensure that traffic alerts are accurate and timely.
* Method: Simulate various traffic conditions and compare the system-generated alerts against actual data.
* Expectation: The system should generate accurate traffic conditions and timely alerts.

**6.2 User Preference Compliance Test:**

* Objective: To verify that the system adheres to user preferences when generating alerts.
* Method: Set various user preferences and verify the generated alerts match the set preferences.
* Expectation: Alerts generated should comply with the user's preferences regarding types, severity, and timing.

**6.3 System Performance under Load Test:**

* Objective: To ensure the system performs optimally under high load conditions.
* Method: Simulate a high number of simultaneous users and a large influx of traffic data.
* Expectation: The system should maintain functional integrity and performance standards under heavy load.

**6.4 Persistent Data Storage Integrity Test:**

* Objective: To ensure that data storage and retrieval mechanisms are robust and error-free.
* Method: Perform a series of data insertion, update, deletion, and retrieval operations.
* Expectation: Data integrity must be maintained with no loss or corruption.

**6.5 Network Protocol Efficiency Test:**

* Objective: To ensure efficient and reliable communication between the client and server.
* Method: Measure data transfer rates, latency, and error rates under different network conditions.
* Expectation: Communication should be efficient, with minimal latency and errors.

**6.6 Global Control Flow Management Test:**

* Objective: To ensure that the system handles execution order, time dependency, and concurrency as expected.
* Method: Simulate various operational scenarios, including high-frequency events and multi-threaded operations.
* Expectation: The system should manage global control flow effectively without deadlocks or performance bottlenecks.

**6.7 User Interface Responsiveness Test:**

* Objective: To ensure that the user interface is responsive and intuitive under various conditions.
* Method: Perform user interface testing on different devices and operating systems, simulating real user interactions.
* Expectation: The user interface should be consistently responsive and provide a positive user experience.

**6.8 Route Optimization Algorithm Test:**

* Objective: To verify the effectiveness and efficiency of the route optimization algorithm.
* Method: Compare the suggested routes under various traffic conditions with known optimal routes.
* Expectation: The system should provide routes that are close to optimal in terms of travel time and convenience.

**6.9 Scalability Test:**

* Objective: To ensure the system can scale effectively as the number of users and data volume grows.
* Method: Gradually increase load on the system, adding users and data, and monitor system performance and resource utilization.
* Expectation: The system should handle increased load without significant degradation in performance or functionality.

**6.10 Security and Privacy Test:**

* Objective: To ensure that user data is secure and privacy is maintained.
* Method: Conduct vulnerability assessments and penetration tests, simulate various attack scenarios.
* Expectation: The system should resist common security threats and protect user data effectively.