UNIVERSITY OF BUEA

Knowledge with Wisdom

Université de Buéa

REPUBLIC OF CAMEROON

Peace_Work_Fatherland

République du Cameroun

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT: COMPUTER ENGINEERING

CEF 440: INTERNET & MOBILE PROGRAMMING

Course Instructor: Dr. Valery Nkemeni

Project Title:

Design and Implementation of a Road Sign and Road State Mobile Notification Application (Smart Roads Ahead).

Task 4:

System Modelling and Design

Group 1	**Matricule**
ARREY-TABOT PASCALINE	FE22A151
TANUI NORBERT TANGIE	FE22A306
OROCKTAKANG MANYI	FE22A293
NGATTA GEORGE TABOT	FE22A259
FOMECHE ABONGACHU SIDNEY	FE22A218

2024/2025 Academic year

Contents

Table of figures

Phase Overview

This project aims to improve road safety, awareness and driving experience by providing real-time notifications to road users about road signs, hazards, and conditions based on their current GPS location.

We focus on how the system is structured, how it interacts with its environment, and how different components work together to deliver its core functionality.

This task involved creating diagrams that visually describe how the system will work.

1. The Context Diagram

Shows the system as a single process and how it interacts with external entities. The system interacts with 3 key external entities:

- The admin, who is responsible for inputting and updating road information such as road signs and conditions.
- The Driver or Road user, who receives timely notifications from the system while driving.
- The GPS System, which continuously provides the user's current location to the system.

These interactions are driven by data flows;

The admin sends road updates to the system. The system receives location data from GPS. Based on these, it generates real-time notifications for the user.

This diagram helps define the system boundary and clarifies what is inside the system versus what comes from external actors.

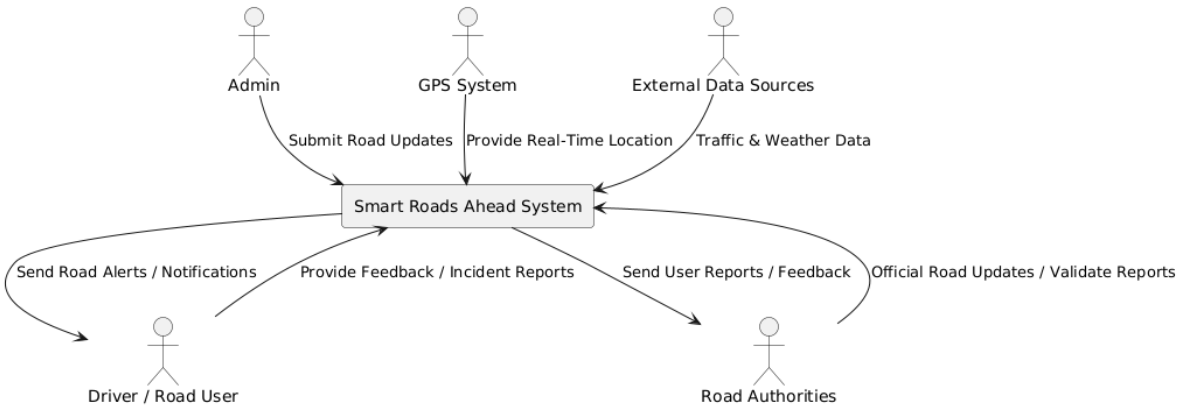

Figure 1: Context diagram

2. The Data Flow Diagram

Helps us understand how data moves through the system from external sources into processing units, and then out as useful information (road notifications).

There are three central processes:

Process 1: Receive Road Updates

This handles input from the admin.

Road signs and conditions are validated and stored in the Road Data Store.

Process 2: Track location

This process receives coordinates from the GPS system and stores it temporarily in the User Location Store.

Process 3: Generate Notification

This process compares the user's location with the road data.

If there's relevant information nearby, like a speed limit, hazard, or construction, it generates a road alert, which is then sent to the road user.

Level 0 breaks down the system into its main functional components;

- Receive road data from the admin.
- Process GPS location from the user's device.
- Send notifications based on road and location data.

It also shows two data stores:

- Road data store, where road signs and conditions are saved.
- User location store: temporarily stores current user locations for processing.

The system receives inputs from the GPS and Admin, processes this data, and outputs alerts to the User.

Data Flows

Data flows represent the movement of information between external entities and the main system process. They are depicted as arrows indicating the direction of data.

- **User to System: "Provides Input/Requests Info"**
    - **Description:** This flow represents all data originating from the Driver/User and entering the system. This includes explicit requests (e.g., searching for a road sign, querying for nearby petrol stations) and implicit inputs (e.g., the user's current location data for location-based alerts, user preferences).
    - **Purpose:** To enable the system to understand user needs and current context.

- **System to User: "Provides Notifications/Information"**
    - **Description:** This flow represents all data generated by the system and delivered to the Driver/User. This encompasses real-time traffic alerts, weather hazard warnings, road closure notifications, display of road sign explanations, and information about nearby petrol stations.
    - **Purpose:** To inform and alert the user about relevant road conditions and provide educational content.
- **External Data Sources to System: "Provides Traffic/Weather Data"**
    - **Description:** This flow illustrates the continuous or on-demand provision of raw data from various external APIs to the "Smart Roads Ahead System." This includes live traffic conditions, weather forecasts and alerts, and geographical data for points of interest.
    - **Purpose:** To enrich the system's knowledge base with real-time and comprehensive external information.
- **Road Authorities to System: "Manages Road Data/Validates Reports"**
    - **Description:** This flow signifies the input from official Road Authorities into the system. This could involve direct updates to the road sign database, official notifications of road closures, or the process of validating user-submitted incident reports.
    - **Purpose:** To ensure the accuracy and official backing of critical road-related information within the application.
- **System to Road Authorities: "Sends User Reports"**
    - **Description:** This flow represents the transmission of user-submitted feedback
    - accident reports (e.g., reporting a pothole, an incorrect sign) from the system to the Road Authorities for review and moderation.
    - **Purpose:** To facilitate the validation process and leverage crowdsourced data for improving overall data quality and responsiveness to road issues.

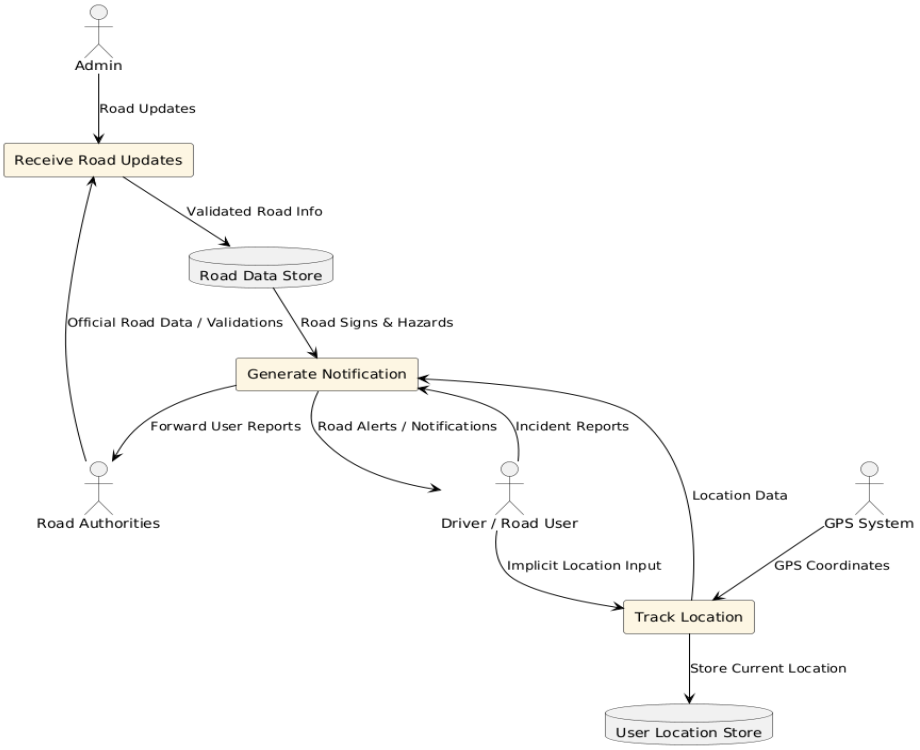

Figure 2: Data Flow Diagram (level 0).

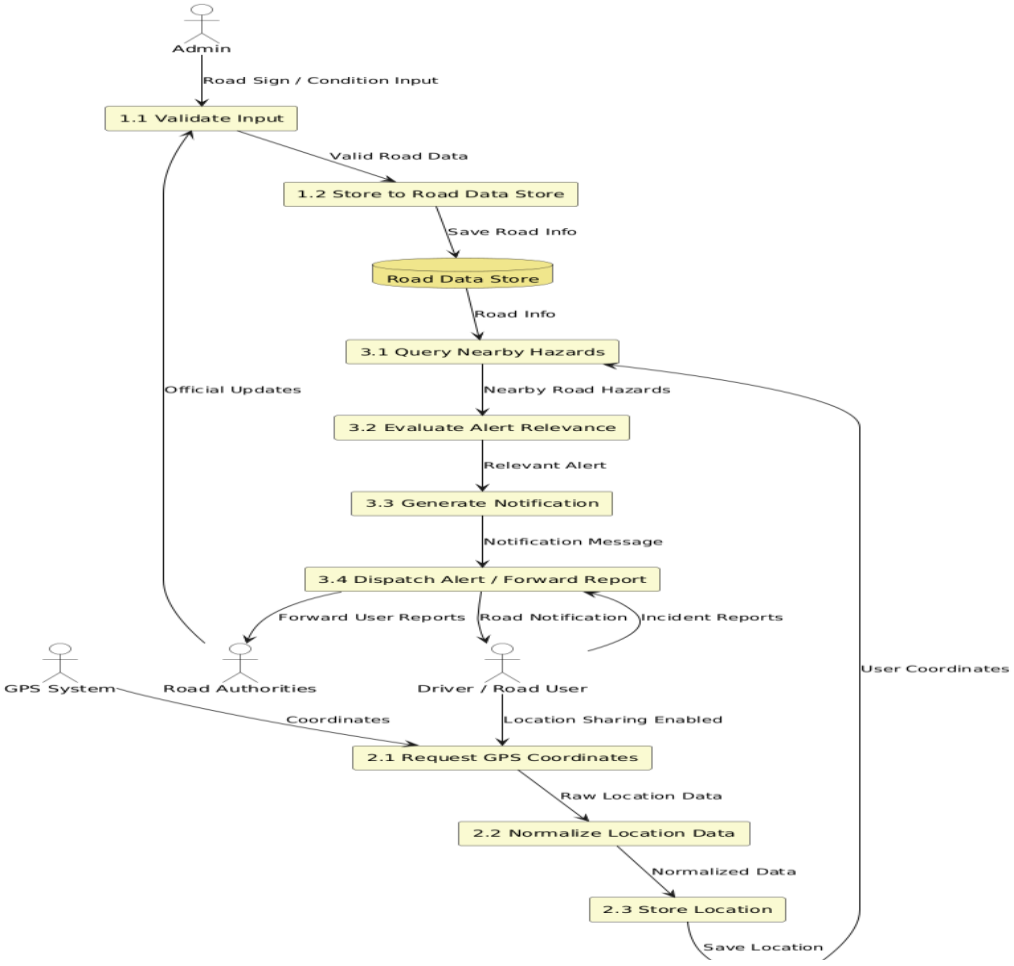

Figure 3:Data Flow Diagram (level 1).

This high-level view provides a foundational understanding of the system's boundaries and its essential data exchanges, serving as a solid basis for further, more detailed analysis in subsequent DFD levels.

3. The Use Case Diagram

Helps us visualize how users interact with the system. It highlights what the system does from the perspectives of external factors such as users, admins and GPS system.

There are three main actors in our system:

- The Admin can add road signs like speed limits or stop signs and update road conditions by reporting temporary conditions like potholes, construction zones. The admin generally manages road related data.
- The Driver/Road User can view notifications and receive real-time alerts, updates about nearby road conditions and timely warnings based on their location.
- The System also tracks users real-time location using the GPS, enabling personalized and relevant alerts.

  There are relationships among these use cases: 'Get Real-time alerts' depends on both 'Track user location' and 'Update about road condition'.

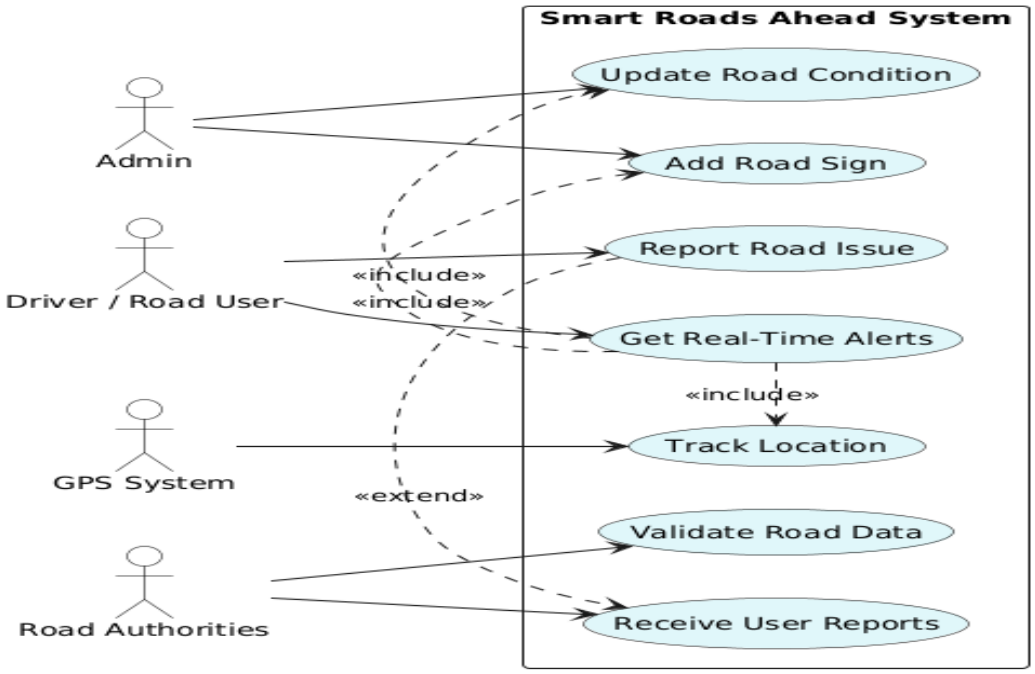

Figure 4:Use Case Diagram

These use cases capture the core functional requirements of our app.

4. The Sequence Diagram

Outlines the workflow of the app when generating real-time alerts for users. It focuses on interactions between system components, data flow, and error handling. It represents interaction over time when a user receives a road alert.

The key participants involved are:

- The user who initiates the interaction by opening the app.
- The mobile app acts as the user interface and handles communication
- GPS system provides the user's current location.
- The backend system processes location data and manages logic.
- A relevant alert is generated and displayed to the user by the notification manager.

The main flow of this sequence:

- The user launches the Smart Roads App on their mobile device.
- The App requests the current GPS location from the GPs system.
- Once the location is received, the app sends It to the backend server.
- The backend calls the notification manager, which;
- Queries the Road Data Store for any road signs or conditions near that location.
- Generates a custom notification message based on the data retrieved.
- The notification is then sent back to the mobile app and
- Finally, the alert is displayed to the user.

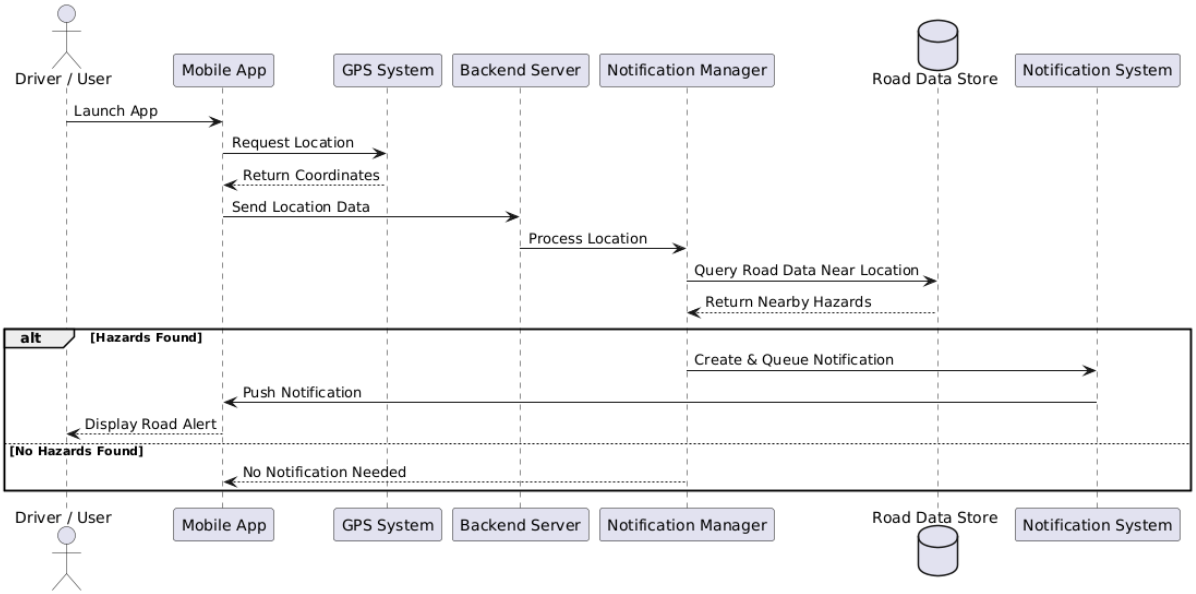

Figure 5: Sequence Diagram.

This sequence diagram demonstrates a scalable, real-time alert system prioritizing user safety. By integrating external APIs and a feedback mechanism, the app ensures proactive hazard notifications while continuously improving data accuracy.

5. The Class Diagram

Shows the system's main components (as classes), their attributes, operations, and their relationships. This diagram is particularly useful for developers as it guides both backend logic and database design.

The class diagram includes the following core classes:

1. User
   Attributes: userID, name, location
   Methods: viewNotification()
2. Admin (inherits from user)
   Additional methods:
   addRoadSign(), updateCondition()
3. RoadSign
   Attributes: signID, location, type, description
   Methods: displayConditionInfo()
4. RoadCondition
   Attributes: conditionID, location, severity, description
   Methods: displayConditionInfo()
5. GPsLocation
   Attributes: latitude, longitude, timestamp
   Methods: getCoordinates()
6. Notification
   Attributes: notificationID, messages, timestamp, userID
   Methods: SendToUser()
7. NotificationManager
   Methods: generateAlert(), matchLoactiontoHazard()

The relationship between these classes:

The user has a one-to-many relationship with notification. Meaning each user can receive multiple notifications.

The admin inherits from the user class. This simplifies reusability since admins are a special kind of user with additional privileges.

The notification manager interacts with both RoadSign and RoadCondition classes to determine which notifications to generate based on the user's GPSLocation.

The notification manager also calls the sendToUser() methods of the Notification class to deliver alerts.

The class diagram gives a clear blueprint for out system's architechture

- **User** and **Admin** interact with **RoadSign**, **RoadCondition**, and **Notification**.
- **GPSLocation** is a shared class for all entities needing coordinates.
- **NotificationManager** controls logic like querying conditions and generating alerts.

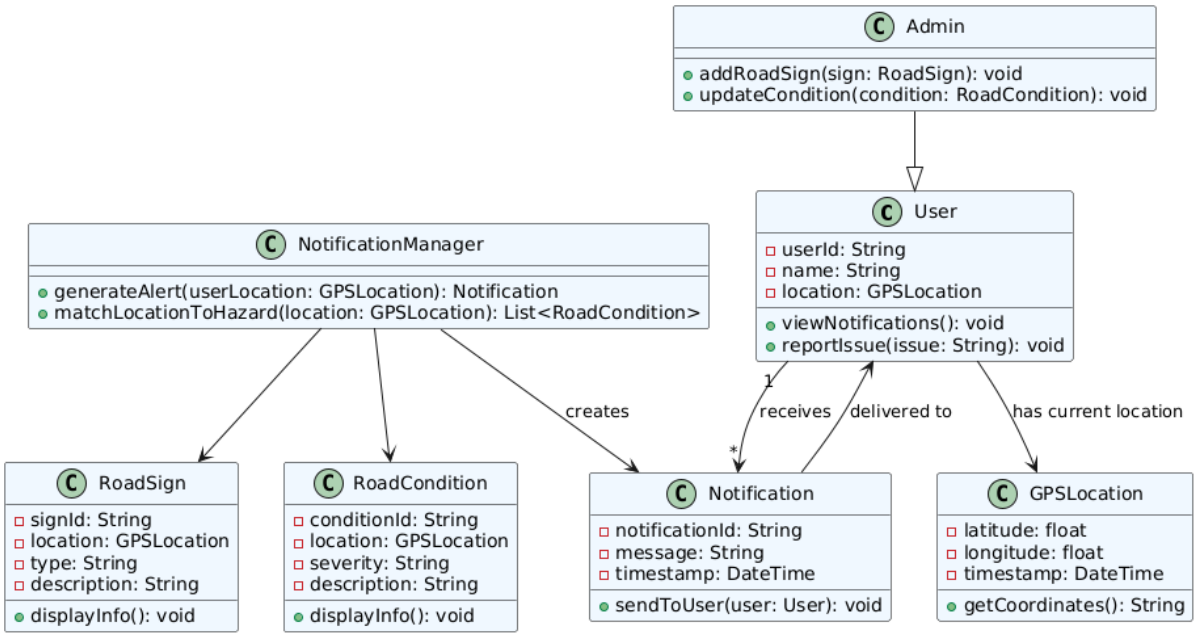

Figure 6:Class Diagram.

6. The Deployment Diagram

Shows how the system is physically set up. It models the physical architecture of the system, showing how software components are deployed across hardware nodes like servers, mobile devices, and third part systems and how they communicate over the network.

- A mobile device runs the app.
- An Admin PC connects to a cloud backend.
- The backend server manages logic and data via APIs and the Notification Manager.
- The GPS system supplies real-time location data.

Hardware Nodes and Devices:

The main nodes in our deployment diagram include:

1. Mobile Device (Client Node)
   Runs the Smart Roads mobile app
   Hosts the user interface and handles GPS access
   Communicates with backend APIs over the internet
2. Web Server
   Hosts the API endpoints and handles request routing
   Deploys backend logic and interacts with the database and other services
3. Database Server
   Stores all persistent data including road signs, conditions, user profiles, and notification logs
   Communicates securely with the web server
4. GPS System (External Node)
   Third-party service or internal module used to fetch the user's current location.
5. Notification Server (optional separate node)
   May be scaled separately to handle large volumes of real-time notifications
   Communicates with both the web server and mobile app

Connections and Communication

Now, how do these components interact?

The mobile device connects to the web server using HTTPS REST API calls.

The web server processes the request, fetches GPS and road data, and updates the database server accordingly.

When necessary, the notification server sends alerts back to the mobile device.

GPS data is fetched either directly from the device or via a GPS service provider.

Significance of the Diagram

This diagram is crucial for the deployment and scaling of the system:

It helps system administrators understand what infrastructure is needed.

It allows engineers to plan network configurations and security (e.g., securing mobile-to-server communication).

It highlights areas where load balancing or failover might be needed such as separating notification services from the core backend.

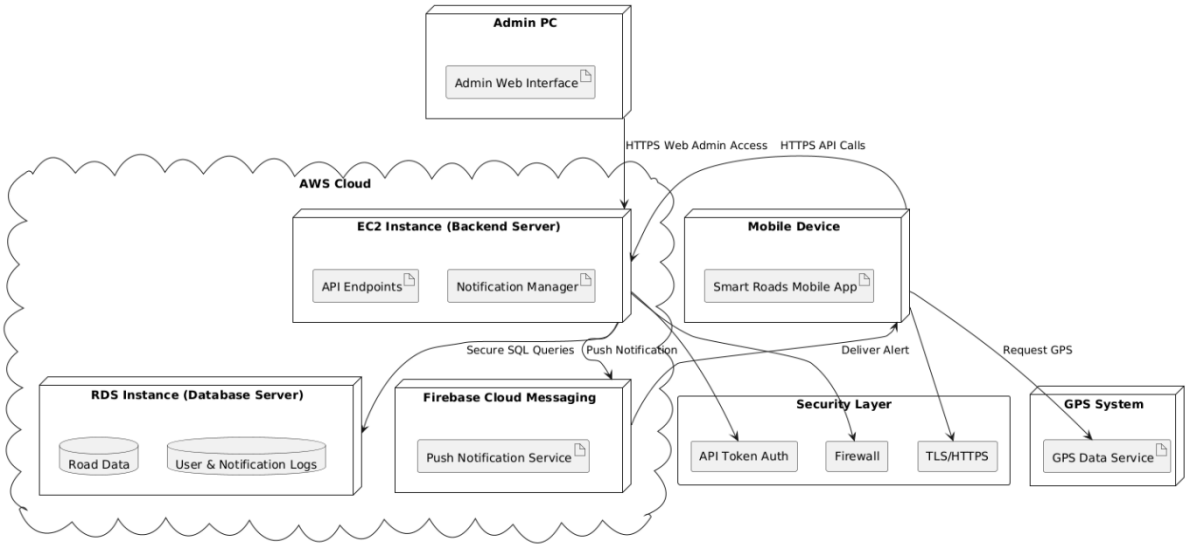

Figure 7:Deployment Diagram.

7. The State Diagram

A state diagram is used to model the different states an object can be in during its lifecycle, as well as the events or conditions that trigger transitions between those states. It's especially useful for modeling dynamic behavior like how a road alert is processed and displayed to the user.

Maps the app lifecycle:

From launching the app,

Requesting GPS location,

To generating and displaying the alert.

Primary Object: Notification Process

In our diagram, we focus on the notification lifecycle — from the moment a user's location is detected to the moment they receive an alert.

States and Transitions

1. Idle State

The system is waiting for user interaction or GPS data.

2. Location Detected

The user’s current location has been successfully fetched via GPS.

3. Hazard Check in Progress

The system queries the database to check for road signs or dangerous conditions near the user.

4. Notification Generated

If a hazard or road sign is nearby, the system constructs a relevant notification message.

5. Notification Delivered

The notification is pushed to the user's device and displayed on the screen.

6. Acknowledged/Viewed

The user sees the notification or interacts with it (e.g., taps to view more).

7. Returned to Idle

The cycle resets, and the system waits for a new GPS update or movement.

Events Triggering Transitions

GPS Update Received → moves from Idle to Location Detected

Query Sent to Backend → triggers Hazard Check

Result Found → triggers Notification Generation

Push Notification Sent → moves to Delivered

User Views Notification → moves to Acknowledged

Timeout or Update → returns to Idle

Why the State Diagram is Important

This diagram is useful for:

Identifying decision points in system behavior

Designing proper event handling (like timeouts or location refreshes)

Ensuring that notifications are not duplicated or missed

It also supports testing by clarifying expected behavior at every step of the process.

In summary, the state diagram of the Smart Roads Notification System clearly outlines how the notification feature behaves in response to user movement and GPS data. It helps ensure that the user experience is smooth and contextually relevant.

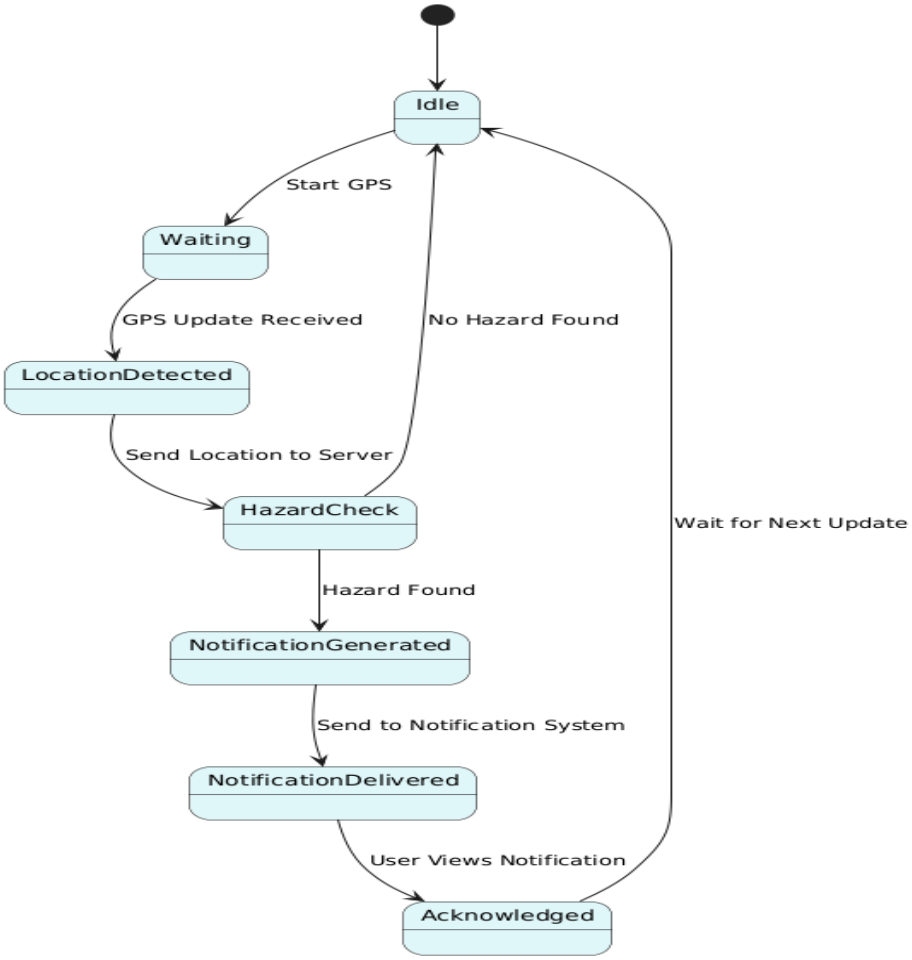

Figure 8:State Diagram.

8. The Activity Diagram

Outlines the steps in receiving a road alert:

- Launch the app
- Get location,
- Query road data,
- Generate and display notification.

The activity diagram illustrates the workflow of the system — the sequence of actions, decision points, and parallel processes. It's useful for understanding how the system handles a specific feature from start to finish.

We modeled the activity diagram around the real-time road alert process.

1. Start (Initial Node)
   The user launches the mobile app.
2. Request GPS Location
   The app automatically requests the user's current location.
3. Receive GPS Coordinates
   Once permission is granted, the app obtains the latitude and longitude.
4. Send Location to Server
   The app sends the location data to the backend server for processing.
5. Check Nearby Road Hazards or Signs
   The server queries the road data store to check for any relevant road signs, alerts, or conditions near the user’s current location.
6. Decision: Are There Nearby Alerts?
   If yes: proceed to generate a notification.
   If no: end activity without notification.
7. Generate Notification Message
   The system prepares a meaningful alert (e.g., "Pothole ahead" or "Speed Limit: 50 km/h")
8. Send Notification to User
   The message is pushed to the mobile device.
9. Display Notification to User
   The alert pops up in the app interface or as a mobile notification.
10. End (Final Node)
    The activity completes and waits for the next GPS update or trigger.

Parallel Actions and Exception Handling

In the real implementation:
Permission check and location retrieval can run in parallel.
If GPS is disabled or network is lost, the system may show a fallback message or prompt the user to enable services.

This diagram is essential for:

Visualizing system behavior for a specific task.

Understanding how control flows, especially in complex features.

Designing testing workflows for user interaction and system response.

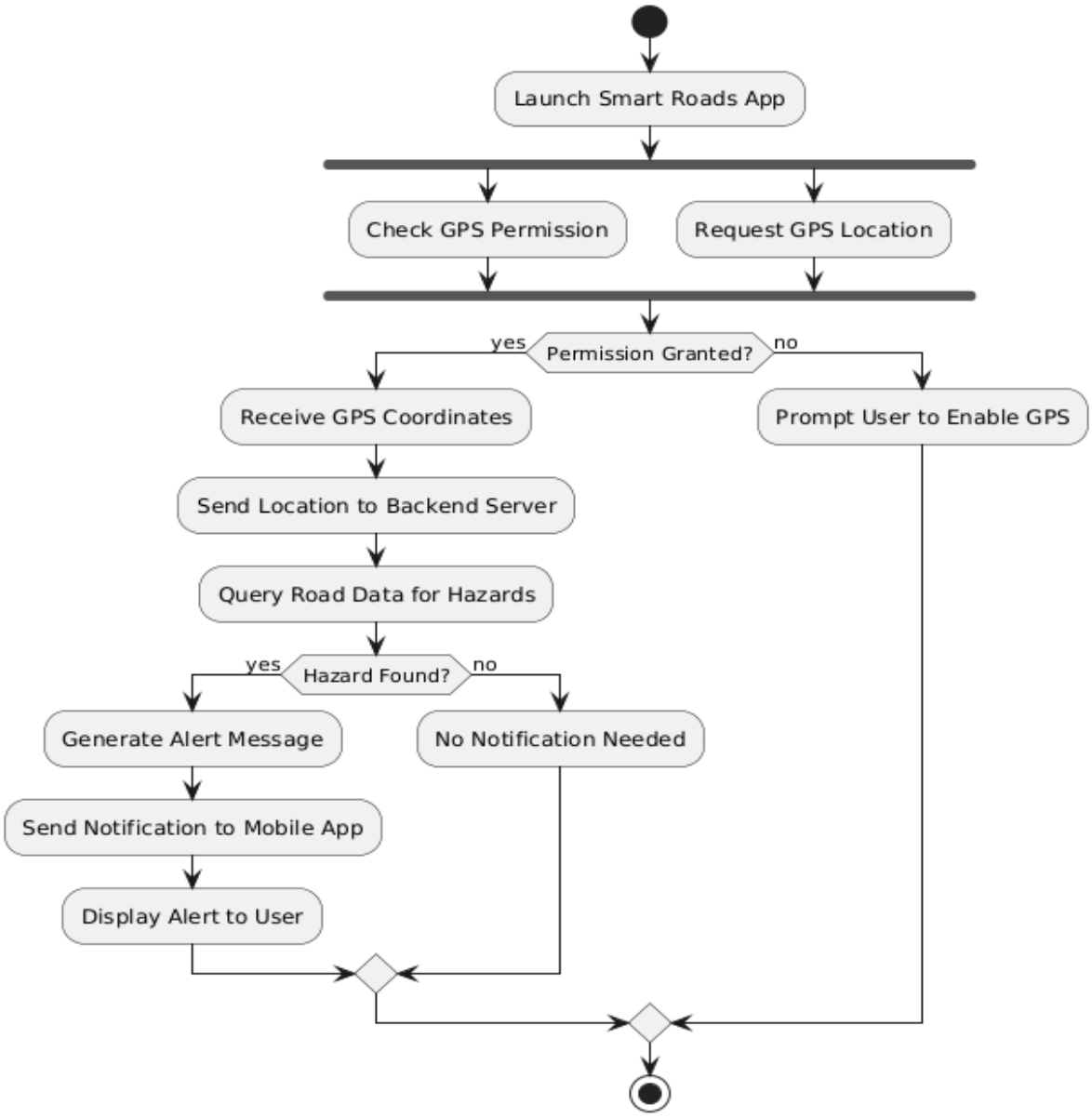

Figure 9:Activity Diagram.

To conclude, the activity diagram shows us how the Smart Roads app transforms user location into meaningful road alerts, ensuring timely and automated guidance for safer driving.