REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

UNIVERSITY OF BUEA

REPUBLIQUE DU CAMEROUN

PAIX-TRAVAIL-PATRIE

UNIVERSITE DE BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

COURSE CODE/TITLE: CEF 440 INTERNET PROGRAMMING AND MOBILE PROGRAMMING

COURSE INSTRUCTOR: DR. NKEMENI VALERY

PROJECT TITLE:

SYSTEM MODELLING AND DESIGN REPORT FOR THE ROAD SIGN AND ROAD STATE MOBILE NOTIFICATION APPLICATION

GROUP 9

PRESENTED BY:

AMINATU MOHAMMED AWAL	FE22A147
DEFANG MARGARET AKUMAYUK	FE22A185
EKWOGE JUNIOR	FE22A200
ESIMO GODWILL EYABI	FE22A207
SUH AKUMAH TILUI-NTONG	FE22A299

TABLE OF CONTENTS

1. **INTRODUCTION:**

The **Road Sign and Road State Mobile Notification Application** is designed to enhance road safety and traffic efficiency by providing drivers with real-time updates on road conditions and comprehensive information about road signs. The application leverages smartphone technology to deliver location-based alerts, ensuring drivers are well-informed about hazards, traffic congestion, and the meanings of the various road signs. This report details the system modeling and design, outlining the architectural and functional blueprints necessary for its development.

2. **BACKGROUND:**

Driving conditions in many developing countries, including Cameroon, present unique challenges that impact road safety and traffic flow.

- **Challenges in Developing Countries**: Poor or inconsistent road signage, a notable lack of real-time traffic updates, and widespread driver unfamiliarity with road sign meanings significantly contribute to a high incidence of accidents and overall traffic inefficiencies. These issues underscore a critical need for accessible and reliable road information.

- **Smartphone Penetration**: The steadily increasing adoption of mobile phones in these regions offers a robust platform for deploying innovative solutions. A dedicated mobile application can effectively bridge the existing information gap, substantially enhancing road safety awareness among drivers.

- **Existing Gaps**: While current navigation applications (e.g., Google Maps, Waze) excel at providing directions, they often lack localized road sign education modules and the capability for highly customizable, real-time alerts tailored to specific road conditions beyond basic traffic. This application aims to fill these precise gaps.

3. **PROJECT OBJECTIVES:**

The primary objectives of the Road Sign and Road State Mobile Notification Application are focused on its core mission of immediate safety and information delivery. Secondary objectives support and enhance these primary goals, ensuring a robust and comprehensive solution.

3.1 **PRIMARY OBJECTIVES:**

The primary objectives of the Road Sign and Road State Mobile Notification Application are:

a. **Road Sign Education:** To provide a comprehensive digital directory that explains road signs, thereby improving driver awareness and adherence to road regulations.

b. **Real-Time Road Condition Alerts:** To deliver customizable notifications to drivers regarding traffic congestion, accidents, adverse weather hazards, and road closures relevant to their location or planned routes.

3.2 **SECONDARY OBJECTIVES:**

c. **User-Driven Reporting:** To enable crowd-sourced updates from drivers regarding current road conditions and incidents, enhancing the accuracy and immediacy of available data.

d. **Integration with Third-Party APIs:** To seamlessly integrate with external APIs (specifically weather, traffic, and mapping services) to leverage real-time data for comprehensive road state analysis and accurate information delivery.

4. **STAKEHOLDERS:**

The success of the application relies on the effective collaboration and interaction of several key stakeholders:

4.1 **END USERS (DRIVERS):**

A. **Role:** The primary consumers of the application's services, relying on real-time information for safe and efficient navigation.

B. **Responsibilities:**

- ❖ Use the app for receiving road condition updates and notifications.
- ❖ Contribute crowd-sourced reports on observed road conditions and incidents.
- ❖ Customize notification settings based on personal preferences and driving needs.

4.2 **EXTERNAL APIS:**

A. **Role:** Essential external systems that supply real-time data crucial for enhancing the application's functionality, user information, and overall user experience.

B. **Importance and Examples:**

1. **Traffic Data APIs:** Provide real-time traffic information, including congestion levels, road conditions, and incident reports (e.g., Google Maps API, Waze API).
2. **Weather APIs:** Deliver current and forecast weather updates that can significantly affect road conditions and driving safety (e.g., OpenWeather, WeatherAPI).
3. **Mapping APIs:** Enable seamless map integration for navigation, location services, and the visual overlay of road signs and hazards (e.g., Google Maps API, Mapbox).

4.3 **ADMIN:**

A. **Role:** The system's administrator, responsible for the oversight and management of application data and operational integrity.

B. **Responsibilities:**

- Manage static content, including road sign information.
- Validate and manage crowd-sourced incident reports for accuracy and reliability.
- Monitor system performance and logs.

4.4 **DEVELOPMENT TEAM:**

A. **Role**: Responsible for the comprehensive design, development, testing and implementation of the application. Their expertise is paramount in transforming conceptual designs into a functional and robust product.

5. **SYSTEM MODELLING AND DESIGN:**

This section presents the various UML diagrams that illustrate the architecture, functionality, and deployment of the Road Sign and Road State Mobile Notification Application. These diagrams serve as blueprints for the development process, ensuring a clear understanding of the system's structure and behavior.

5.1 **CONTEXT DIAGRAM:**

A. **PURPOSE:** A Context Diagram (or Level 0 Data Flow Diagram) provides a high-level overview of the system, illustrating its interaction with external entities (actors and external systems). It defines the system's boundary and its interfaces with the outside world.

B. **COMPONENTS:**

i. **Primary Actors**:

- Drivers: End-users who submit reports and receive notifications.

- Admins: Oversee and manage system data, including report verification and dispatching responses.

ii. **External APIs**:

- Weather API Provides real-time weather hazards.

- **Traffic API**: Supplies real-time congestion and accident data.

- **Mapping Service**: Renders geographical data and supports road sign/hazard overlays on maps.

C. **DATA FLOW:**

The following data flows represent the information exchange between the system and its external entities:

1. **Driver → System**:

- <u>User location</u>: Real-time GPS data from the driver's device.

- Incident Report: Crowd-sourced data including incident type, location and description.

- Notification Preferences: User's personalized settings for receiving alerts.

2. **System → Driver**:

- Notifications: Alerts about accidents, weather conditions, road closures, and general road state.

- Map Display: Visual representation of geographical data.

- Displayed Road Signs: Visual information about road signs based on location.

3. **Admin → System**:

- Road sign/state data: Initial input and updates of road sign information (images, meanings) and road state adjustments.

- Incident management: Manual review, validation or deletion of crowd-sourced reports.

4. **System → Admin:**

- System data, logs: Operational data, error logs, and a view of reported incidents for administrative management and monitoring.

5. **System → APIs**:

- Request Weather,Traffic and Map data: Queries sent to external services for real-time information.

6. **APIs → System:**

- Real-time Map, Weather and Traffic Responses: Data received from external APIs in response to system requests.

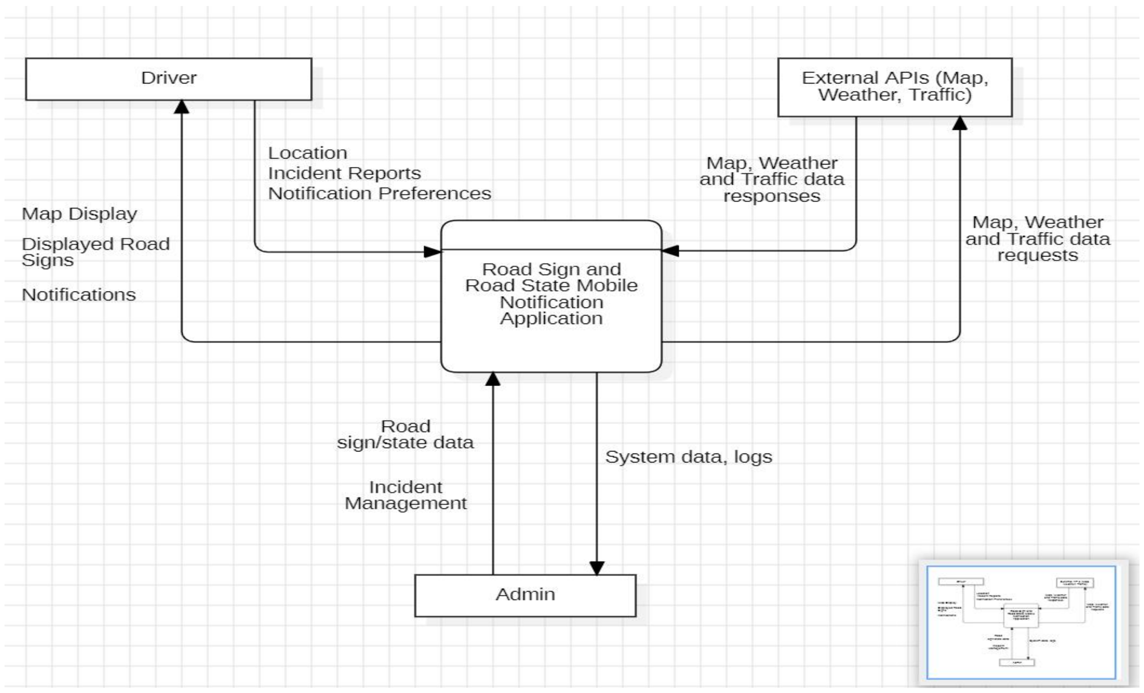
Driver
External APIs (Map, Weather, Traffic)
Location
Incident Reports
Notification Preferences
Map, Weather and Traffic data responses
Map Display
Displayed Road Signs
Notifications
Map, Weather and Traffic data requests
Road Sign and Road State Mobile Notification Application
Road sign/state data
Incident Management
System data, logs
Admin

Fig 1: Context Diagram for the Mobile Application

5.2 **DATA FLOW DIAGRAM:**

A. **PURPOSE:** A Data Flow Diagram (DFD) illustrates how data flows through a system, depicting its interactions with external entities, internal processes, and data stores. It provides a more detailed logical view of the system's functions compared to the Context Diagram.

B. **PROCESSES:**

i. **P1: Manage User Interface & Location:** Handles all user interactions, displays information and manages the device's GPS location.

ii. **P2: Handle Road Sign Information:** Manages the storage and retrieval of road sign data.

iii. **P3: Process Road State Data:** Gathers, consolidates and processes real-time road state information from various internal and external sources.

iv. **P4: Manage Notifications:** Determines when and how to send notifications to users based on processed data and user preferences.

C. **DATA STORES:**

i. **DS1: Road Sign Database:** Stores static information about road signs (images, meanings, categories).

ii. **DS2: Incident Reports Database:** Stores crowd-sourced incident reports (type, location, description, timestamp, status).

iii. **DS3: User Preferences:** Stores individual user notification settings and preferences.

D. **DATA FLOW:** These are arrows showing how data moves between external entities, processes, and data stores, illustrating the dynamic interactions within the system.

E. **DIAGRAM ELEMENTS:** Diagram Elements include rectangles for external entities, circles (or rounded rectangles) for processes, open-ended rectangles for data stores, and labeled arrows for data flow.

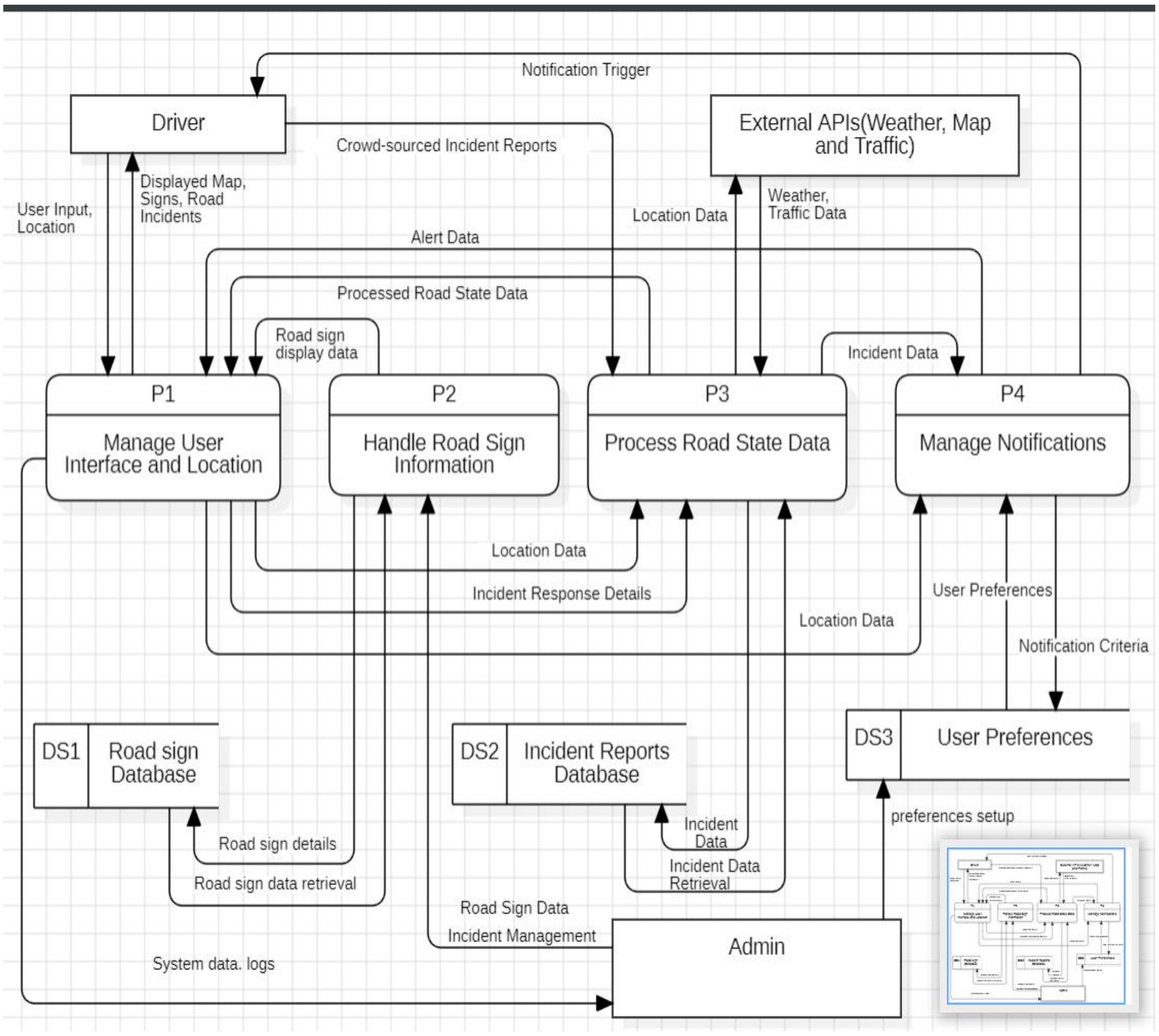

Fig 2: Data Flow Diagram for the Mobile Application

Explanation of Processes and Data Flows:

1. **P1: Manage User Interface & Location:** Handles all user interactions, displays information, and manages device's GPS location.

   - **<u>Inputs</u>:** User Input (taps, gestures), Raw Location Data from device, Road Sign Display Data from P2, Processed Road State Data from P3, Alert Data from P4, Map Data from External APIs.
   - **<u>Outputs</u>**: Display to User (Displayed Map, Signs, Road Incidents), Processed Location Data to P3 & P4, Incident Report Details to P3, Requests Map Data to External APIs.

2. **P2: Handle Road Sign Information:** Manages the storage and retrieval of road sign data.

   - **<u>Inputs:</u>** Road Sign Data from Admin.
   - **<u>Outputs</u>**: Road Sign Display Data to P1.

3. **P3: Process Road State Data:** Gathers and processes real-time road state information (weather, crowd-sourced incidents).

   - **<u>Inputs:</u>** Location Data from P1, Weather Data from External APIs, Traffic Data from External APIs, Crowd-sourced Incident Reports from Driver.
   - **<u>Outputs:</u>** Processed Road State Data to P1, Consolidated Incident Data to P4.

4. **P4: Manage Notifications:** Determines when to send notifications based on user location, preferences, and road state data.

   - **<u>Inputs</u>**: Location Data from P1, Consolidated Incident Data from P3, User Preferences from DS3.
   - **<u>Outputs:</u>** Notification Trigger to Driver, Alert Data to P1.

5.3 USE CASE DIAGRAM:

A. **PURPOSE:** A Use Case Diagram illustrates the functional requirements of the system by depicting its various users (actors) and the goals (use cases) they achieve by interacting with the system. It defines the boundaries of the system from a functional perspective, showcasing "what" the system does.

B. **ACTORS:**

i. **Drivers:**

- ❖ **Description:** The primary end-user of the mobile application. This actor represents any individual operating a vehicle who uses the app to get road information, report incidents, and receive notifications for safer and more efficient travel.
- ❖ **Interactions:** Initiates actions like logging in, managing preferences, Browse data, reporting, and receiving information.

ii. **Admin:**

- ❖ **Description:** Represents the administrative user(s) responsible for managing the application's core data and overseeing operations.
- ❖ **Interactions:** Manages system data (road signs, user reports) and monitors system activity.

iii. **External APIs(Map, Weather and Traffic Services):**

- ❖ **Description:** Represents external software systems that provide critical real-time data and services to the application, enabling its core functionalities. These are essential external systems.
- ❖ **Interactions:** Provides map data, real-time weather conditions, and traffic incident data to the application upon request.

C. **USE CASES:**

The following use cases define the functional goals achievable within the Road Sign and Road State Mobile Notification Application:

i. **Use Cases for the Driver:**

1. **Login:**

- **Description:** The driver authenticates themselves to gain access to personalized features of the application.
- **Actor:** Driver

2. **Manage Profile:**

- **Description:** The driver updates and manages their personal information and general account settings within the application.
- **Actor:** Driver

3. **Browse Road Sign Directory:**

- **Description:** The driver views and searches a comprehensive directory of road signs and their detailed meanings, enhancing their understanding of road regulations.
- **Actor:** Driver

4. **Report Road Incident:**

- **Description:** The driver submits information about current road conditions or hazards (e.g., potholes, traffic jams, accidents) at their location, contributing to crowd-sourced data.
- **Actor:** Driver

5. **Receive Road Condition Notifications:**

- **Description:** The system automatically alerts the driver about real-time road conditions, traffic congestion, accidents, weather hazards, or road sign information relevant to their location or monitored routes. This includes all types of alerts and notifications.
- **Actor:** Driver

6. **Customize Notification Preferences:**

- **Description:** The driver personalizes the types of notifications they receive, their alert radius, and specific routes they wish to monitor for updates.

- **Actor:** Driver

7. **Logout:**

- **Description:** The driver securely ends their session in the application.
- **Actor:** Driver

ii. **Use Cases for the Admins:**

1. **Update Road Sign Database:**

- **Description:** The administrator adds new road signs, modifies existing sign details (images, meanings), or removes outdated road sign information from the system's database.
- **Actor:** Admin

2. **Monitor System Logs:**

- **Description:** The administrator views and analyzes system operational data and logs to track performance, identify issues, and ensure the smooth functioning of the application.
- **Actor:** Admin

3. **Manage Crowd-sourced Reports:**

- **Description:** The administrator reviews, verifies, approves, or deletes user-submitted road incident reports to maintain the accuracy and reliability of the crowd-sourced data.
- **Actor:** Admin

iii. **Use Cases for External APIs:**

1. **Integrate Real-time Traffic Data:**
- **Description:** The system retrieves current traffic conditions, congestion levels, and accident reports from an external Traffic API to provide real-time road state information.

- **Actor:** External APIs

2. **Integrate Map Data:**

- **Description:** The system retrieves map tiles and geographical data from an external Mapping Service to display road networks, locations, and overlays within the application.
- **Actor:** External APIs

3. **Integrate Real-time Weather Data:**

- **Description:** The system fetches current weather conditions and weather-related hazards from an external Weather API to provide environmental road state information.
- **Actor:** External APIs

D. **RELATIONSHIPS:**

- **Associations:** Represented by a line connecting an actor to a use case, indicating that the actor participates in or initiates that use case.

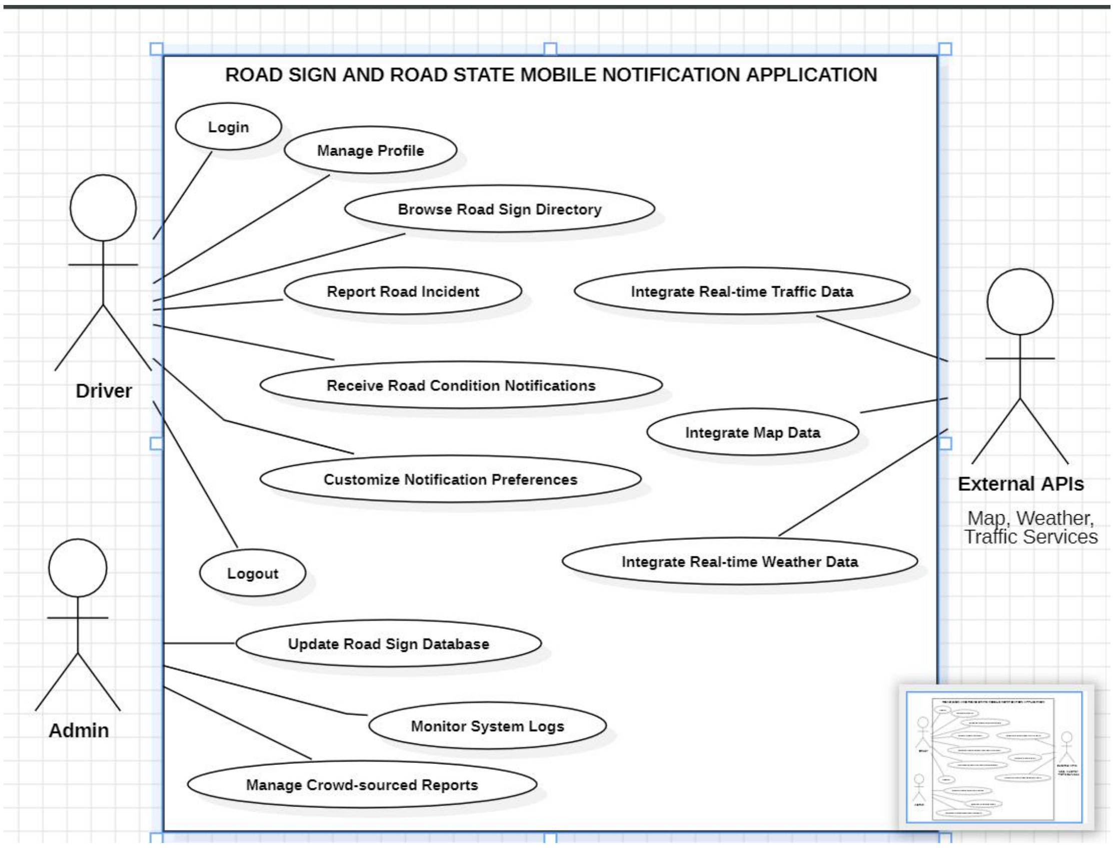
ROAD SIGN AND ROAD STATE MOBILE NOTIFICATION APPLICATION
Login
Manage Profile
Browse Road Sign Directory
Report Road Incident
Integrate Real-time Traffic Data
Receive Road Condition Notifications
Integrate Map Data
Customize Notification Preferences
Integrate Real-time Weather Data
Logout
Update Road Sign Database
Monitor System Logs
Manage Crowd-sourced Reports
Driver
Admin
External APIs
Map, Weather, Traffic Services

Fig 3: Use Case Diagram for the Mobile Application

5.4 SEQUENCE DIAGRAM:

A. INTRODUCTION:

Sequence diagrams are a type of interaction diagram within the Unified Modeling Language (UML) that visually represent the dynamic behavior of a system. They depict the order in which messages are exchanged between participating objects or actors over time to achieve a specific functionality or use case.

B. PURPOSE:

The primary purpose of a sequence diagram includes:

- ❖**Visualizing Dynamic Behavior**: Sequence diagrams depict how objects or systems interact with each other in a sequential manner, making it easier to understand dynamic processes and workflows.
- ❖**Clear Communication**: They provide an intuitive way to convey system behavior, helping teams understand complex interactions without diving into code.
- ❖**Use Case Analysis:** Sequence diagrams are useful for analyzing and representing use cases, making it clear how specific processes are executed within a system.
- ❖**Designing System Architecture**: They assist in defining how various components or services in a system communicate, which is essential for designing complex, distributed systems or service-oriented architectures.
- ❖**Documenting System Behavior:** Sequence diagrams provide an effective way to document how different parts of a system work together, which can be useful for both developers and maintenance teams.
- ❖**Debugging and Troubleshooting:** By modeling the sequence of interactions, they help identify potential bottlenecks, inefficiencies, or errors in system processes.

For complex systems, it is common practice to create multiple sequence diagrams, each focusing on a distinct, critical use case. This approach helps to manage complexity, keeping individual diagrams clear and readable while collectively illustrating the system's comprehensive dynamic behavior. For the Road Sign and Road State Mobile Notification Application, several key operational flows are presented to demonstrate its core functionalities.

C. **SEQUENCE DIAGRAM BASIC NOTATIONS:**

1. **Actors (Users):**

   An actor in a UML diagram represents a type of role where it interacts with the system and its objects. We use actors to depict various roles including human users and other external subjects. We can have multiple actors in a sequence diagram.

2. **Object symbol:**

   Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system.

3. **Activation box:**

   Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.

4. **Package symbol:**

   Used to contain interactive elements of the diagram. Also known as a frame, this rectangular shape has a small inner rectangle for labeling the diagram.

5. **Lifeline symbol:**

   Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbol.

6. **Option loop symbol:**

   Used to model if/then scenarios, i.e., a circumstance that will only occur under certain conditions.

7. **Alternative symbol:**

   Symbolizes a choice (that is usually mutually exclusive) between two or more message sequences. To represent alternatives, use the labeled rectangle shape with a dashed line inside.

5.4.1 **SEQUENCE DIAGRAM: RECEIVE ROAD CONDITION NOTIFICATIONS**

A. **PURPOSE:** To illustrate the sequential flow of interactions between the Driver (User), Mobile Application, Backend Server, various external APIs (Mapping, Weather, Traffic), and internal databases (Incident Reports Database, User Preferences) when a driver receives a notification about road conditions, including traffic data.

B. **ACTORS AND PARTICIPANTS:**

The diagram for involves the following actors and system components (lifelines):

- **Driver (User):** The human user interacting with the mobile application.
- **MobileApp:** The "Road Sign and Road State Mobile Notification Application" running on the user's smartphone or tablet.
- **BackendServer:** The server-side component of the application responsible for data processing, API integrations, and notification logic.
- **IncidentReportsDB (DS2):** The database storing crowd-sourced road incident reports.
- **UserPreferencesDB (DS3):** The database storing individual user notification preferences.
- **WeatherAPI:** An external service providing real-time weather data.
- **TrafficAPI:** An external service providing real-time traffic incident data (e.g., congestion, accidents).
- **MappingAPI:** An external service providing map tiles and geographical information for display.
- **Admin:** The administrative user or team responsible for monitoring system logs.

C. **SEQUENCE OF INTERACTIONS:**

The diagram outlines the following sequence of messages and actions:

I. **MAIN FLOW: INITIATING ROAD STATE UPDATES:**

1. **User Opens App / Continues Driving:** The scenario begins with the Driver either launching the mobile application or having it running in the background while they are driving.
2. **Gets Current GPS Location:** The MobileApp continuously utilizes the device's GPS sensor to determine and update the Driver's current geographical location (latitude and longitude).
3. **Periodically Poll for Road State Updates:** The MobileApp sends periodic requests to the BackendServer, providing the Driver's current location and a defined notification area/radius. This initiates the process of checking for relevant road conditions.
4. **Query Incident Reports:** Upon receiving the update request, the BackendServer queries the IncidentReportsDB (DS2) to retrieve all crowd-sourced incident data (e.g., potholes, local hazards) that fall within the specified notification area.
5. **Crowdsourced Incident Data:** The IncidentReportsDB responds to the BackendServer with the relevant crowd-sourced incident information.
6. **Request Weather Data:** Simultaneously, the BackendServer sends a request to the WeatherAPI for current weather conditions pertinent to the Driver's location.
7. **Current Weather Data:** The WeatherAPI responds with the requested weather information (e.g., temperature, precipitation, weather alerts).
8. **Request Traffic Incident Data:** The BackendServer also sends a request to the TrafficAPI to obtain real-time traffic incident data (e.g., congestion, accidents, road closures) within the Driver's vicinity.
9. **Real-time Traffic Data:** The TrafficAPI responds with the relevant traffic incident information.
10. **Consolidate Road State Data:** The BackendServer then aggregates all the collected road state information: crowd-sourced incidents, weather data, and traffic incident data, creating a comprehensive picture of the current road conditions.
11. **Retrieve User Notification Preferences:** To personalize the alerts, the BackendServer retrieves the Driver's saved notification preferences from the UserPreferencesDB (DS3). These preferences dictate what types of alerts the

Driver wishes to receive (e.g., only major accidents, all hazards, specific routes).

12. **Apply Notification Logic:** The BackendServer applies its internal notification logic, comparing the consolidated road state data against the Driver's location and their retrieved notification preferences to determine if any relevant alert should be triggered.

II. **ALTERNATIVE FLOW: IF RELEVANT NOTIFICATION TRIGGERED**

1. **Send Notification Payload:** If the notification logic determines that a relevant alert should be sent, the BackendServer sends a "Notification Payload" (containing the alert content and type) to the MobileApp.
2. **Process Notification Payload:** The MobileApp receives and processes this payload.
3. **Display Notification:** The MobileApp then displays a local notification to the Driver (e.g., a pop-up alert, a sound, or a vibration), immediately informing them of the road condition.
4. **Update Map with New Incident/Traffic Info:** Concurrently, the MobileApp updates its internal map display to reflect the new incident or traffic information visually.
5. **(Optional) Request Updated Map Tiles:** If the new information necessitates a map refresh or more detailed tiles, the MobileApp may send a request to the MappingAPI for updated map data.
6. **Map Tiles:** The MappingAPI responds with the requested map tiles.
7. **Display Updated Map with Consolidated Incidents:** Finally, the MobileApp presents the updated map to the Driver, showing their current location along with the newly reported or detected incidents and traffic conditions.

III. **ALTERNATIVE FLOW: NO RELEVANT NOTIFICATION:**

1. **No New Updates:** If the notification logic determines that no relevant alerts are triggered for the current period, the BackendServer simply sends a "No new updates" message back to the MobileApp.

IV. **OPTIONAL FLOW: ADMIN MONITORING:**

1. **Send System Data, Logs:** Periodically, the MobileApp sends operational data and logs to the BackendServer.
2. **Forward System Data, Logs:** The BackendServer then forwards these "System data, logs" to the Admin, allowing them to monitor the application's performance, identify potential issues, and gain insights into user activity and reported data.

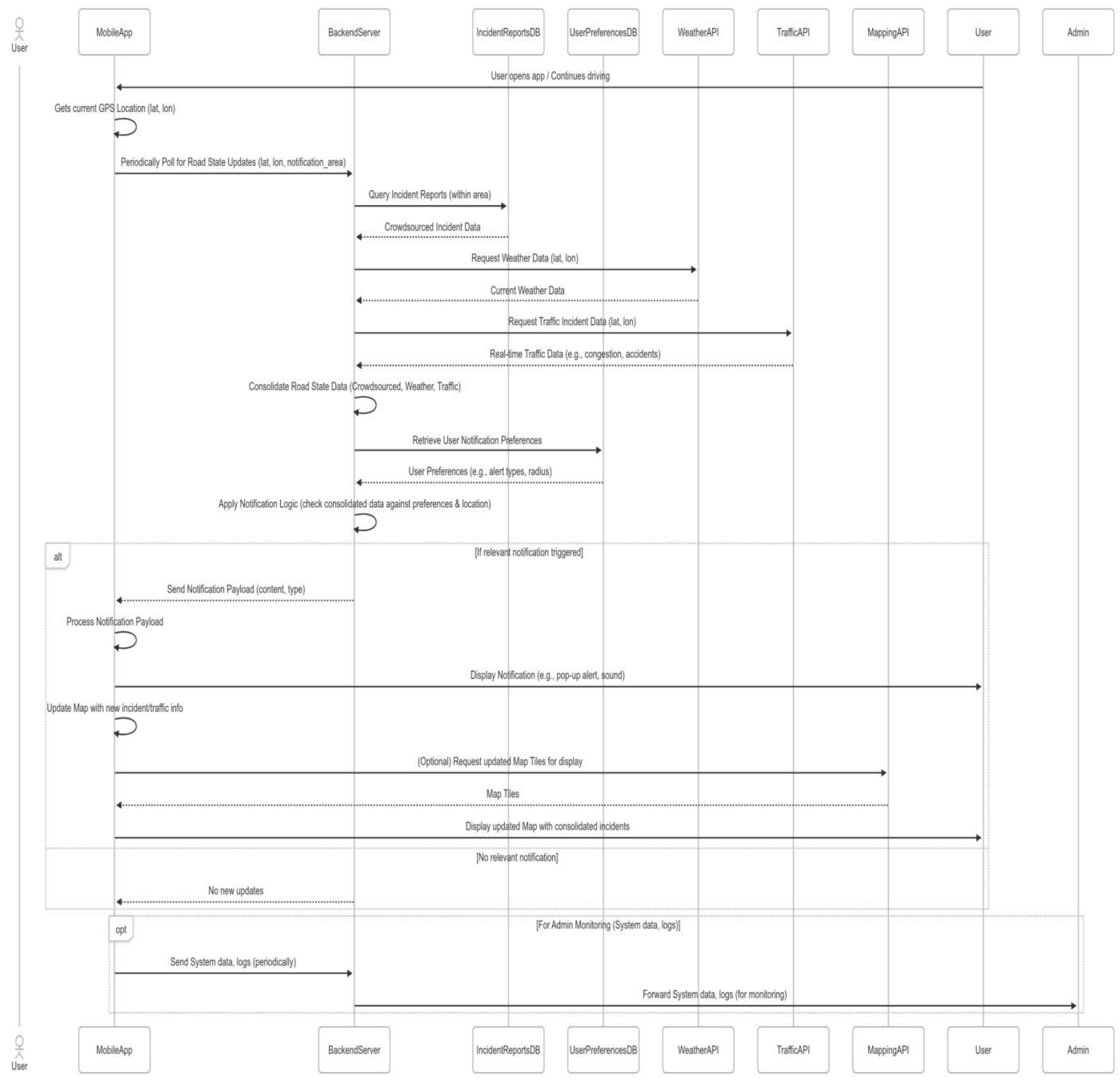
User
MobileApp
BackendServer
IncidentReportsDB
UserPreferencesDB
WeatherAPI
TrafficAPI
MappingAPI
User
Admin
User opens app / Continues driving
Gets current GPS Location (lat, lon)
Periodically Poll for Road State Updates (lat, lon, notification_area)
Query Incident Reports (within area)
Crowdsourced Incident Data
Request Weather Data (lat, lon)
Current Weather Data
Request Traffic Incident Data (lat, lon)
Real-time Traffic Data (e.g., congestion, accidents)
Consolidate Road State Data (Crowdsourced, Weather, Traffic)
Retrieve User Notification Preferences
User Preferences (e.g., alert types, radius)
Apply Notification Logic (check consolidated data against preferences & location)
alt
[If relevant notification triggered]
Send Notification Payload (content, type)
Process Notification Payload
Display Notification (e.g., pop-up alert, sound)
Update Map with new incident/traffic info
(Optional) Request updated Map Tiles for display
Map Tiles
Display updated Map with consolidated incidents
[No relevant notification]
No new updates
opt
[For Admin Monitoring (System data, logs)]
Send System data, logs (periodically)
Forward System data, logs (for monitoring)
User
MobileApp
BackendServer
IncidentReportsDB
UserPreferencesDB
WeatherAPI
TrafficAPI
MappingAPI
User
Admin

Fig 4: Sequence Diagram for Receiving Road Condition Notifications

5.4.2 **SEQUENCE DIAGRAM: REPORT ROAD ACCIDENT**

A. **PURPOSE:** To illustrate the sequential flow of interactions when a Driver reports a new road incident through the mobile application, detailing the steps from user input to database storage.

B. **ACTORS AND PARTICIPANTS:**

The diagram for "Report Road Incident" involves the following actors and system components (lifelines):

- **Driver:** The primary end-user initiating the report of a new road incident.
- **MobileApp:** The client-side application responsible for collecting incident details from the user and sending them to the backend.
- **BackendServer:** The server-side component responsible for validating and processing the incident report.
- **IncidentReportsDB (DS2):** The internal database where all crowd-sourced incident reports are stored after validation.
- **Admin:** The administrative user or team, optionally alerted when a new incident report is submitted for review.

C. **SEQUENCE OF INTERACTIONS:**

The diagram outlines the following sequence of messages and actions:

1. **Driver Selects "Report Incident":** The sequence begins with the Driver interacting with the MobileApp to initiate the incident reporting process.
2. **Capture GPS Location:** The MobileApp automatically captures the Driver's current GPS location (latitude and longitude).
3. **Prompt for Incident Details:** The MobileApp prompts the Driver to enter or select details about the incident, such as its type (e.g., pothole, accident, construction), a brief description, and optionally, to attach a photo.
4. **Driver Enters Incident Details:** The Driver provides the requested information via the MobileApp's interface.
5. **Submit Incident Report:** The MobileApp bundles all the incident details, including the location and any attached media, and sends it to the BackendServer for processing.

6. **Validate Incident Report Data:** The BackendServer performs validation checks on the submitted report to ensure data integrity and completeness.
7. **If Validation Successful (Alternative Flow):**
    1. **Save New Incident Report:** If validation passes, the BackendServer proceeds to save the new incident report into the IncidentReportsDB (DS2).
    2. **Confirmation of Save:** The IncidentReportsDB responds with a confirmation that the data has been successfully saved.
    3. **Incident Report Submission Confirmation:** The BackendServer then sends a confirmation message back to the MobileApp.
    4. **Display "Report Submitted Successfully":** The MobileApp displays a success message to the Driver, confirming that their report has been submitted.
    5. **Notify Admin for Review (Optional):** In some systems, a new incident report might trigger an alert to the Admin for immediate review or validation.
8. **If Validation Fails (Alternative Flow):** If the validation fails, the BackendServer sends an error message back to the MobileApp. The MobileApp then displays a "Submission Failed" message along with the reason for the failure to the Driver.

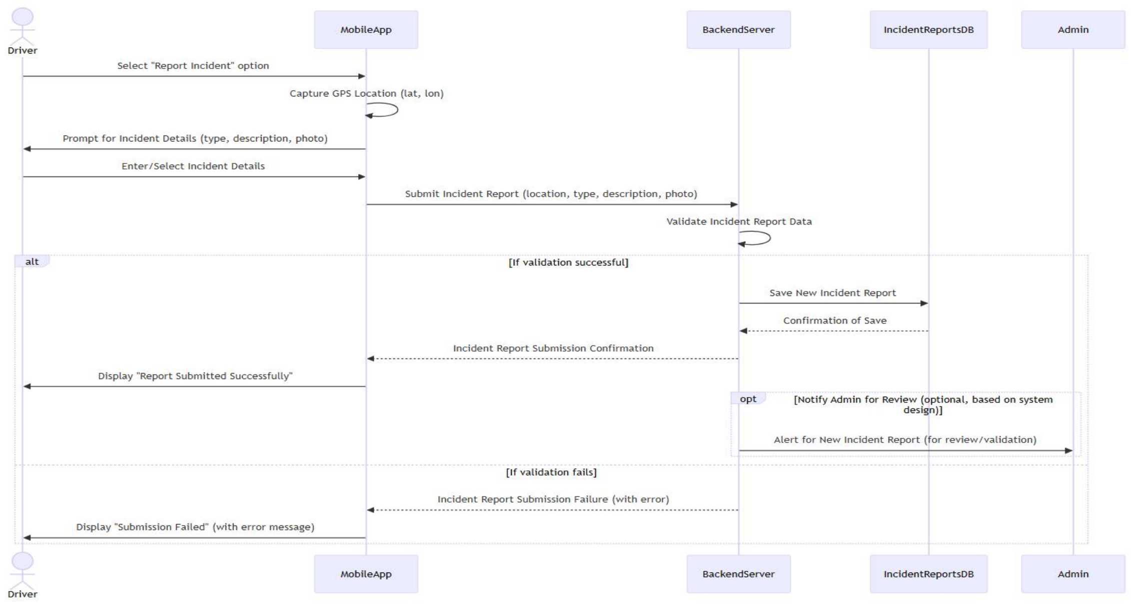
Driver
MobileApp
BackendServer
IncidentReportsDB
Admin
Select "Report Incident" option
Capture GPS Location (lat, lon)
Prompt for Incident Details (type, description, photo)
Enter/Select Incident Details
Submit Incident Report (location, type, description, photo)
Validate Incident Report Data
alt
[If validation successful]
Save New Incident Report
Confirmation of Save
Incident Report Submission Confirmation
Display "Report Submitted Successfully"
opt
[Notify Admin for Review (optional, based on system design)]
Alert for New Incident Report (for review/validation)
[If validation fails]
Incident Report Submission Failure (with error)
Display "Submission Failed" (with error message)
Driver
MobileApp
BackendServer
IncidentReportsDB
Admin

Fig 5: Sequence Diagram for Reporting a Road Incident

5.4.3 **SEQUENCE DIAGRAM: ADMIN UPDATES ROAD SIGN DATA**

A. **Purpose:** To illustrate the sequential flow of interactions when an Administrator updates or adds new road sign information to the system's database, covering the process from input to data storage.

B. **ACTORS AND PARTICIPANTS:**

The diagram for "Admin Updates Road Sign Data" involves the following actors and system components (lifelines):

- **Admin:** The administrative user responsible for managing the road sign data within the system.
- **AdminInterface (Admin Web/Mobile UI):** The interface (e.g., web portal or dedicated admin module) through which the Administrator interacts with the system to manage road signs.
- **BackendServer:** The server-side component responsible for handling administrative requests, validating data, and interacting with the database.
- **RoadSignDB (DS1):** The internal database specifically storing static information about road signs (images, meanings, categories).

C. **SEQUENCE OF INTERACTIONS:**

The diagram outlines the following sequence of messages and actions:

1. **Admin Accesses "Manage Road Signs" Module:** The sequence starts with the Admin logging into an administrative interface (e.g., a web portal or a specific admin module within the application).
2. **Request Existing Road Sign Data:** The AdminInterface sends a request to the BackendServer to retrieve existing road sign data, which the Admin intends to review or modify.
3. **Retrieve Road Sign Data:** The BackendServer queries the RoadSignDB (DS1) to fetch the requested road sign information.
4. **Road Sign Data:** The RoadSignDB responds with the relevant road sign data.
5. **Display Road Sign Data:** The BackendServer sends the retrieved data to the AdminInterface for display to the Admin.
6. **Input New/Updated Road Sign Details:** The Admin makes necessary modifications or inputs details for a new road sign via the AdminInterface.

7. **Submit Road Sign Update/Add Request:** The AdminInterface sends the new or updated road sign data to the BackendServer.
8. **Validate Road Sign Data:** The BackendServer validates the incoming road sign data for correctness and integrity.
9. **If Validation Successful (Alternative Flow):**
   - **Save/Update Road Sign Details:** If the data is valid, the BackendServer proceeds to save or update the RoadSignDB (DS1) with the new information.
   - **Confirmation of Save:** The RoadSignDB confirms the successful operation.
   - **Road Sign Update Confirmation:** The BackendServer sends a confirmation back to the AdminInterface.
   - **Display "Road Sign Updated Successfully":** The AdminInterface notifies the Admin of the successful update.
10. **If Validation Fails (Alternative Flow):** If validation fails, the BackendServer sends an error message to the AdminInterface, which then displays it to the Admin, indicating that the update could not be completed.

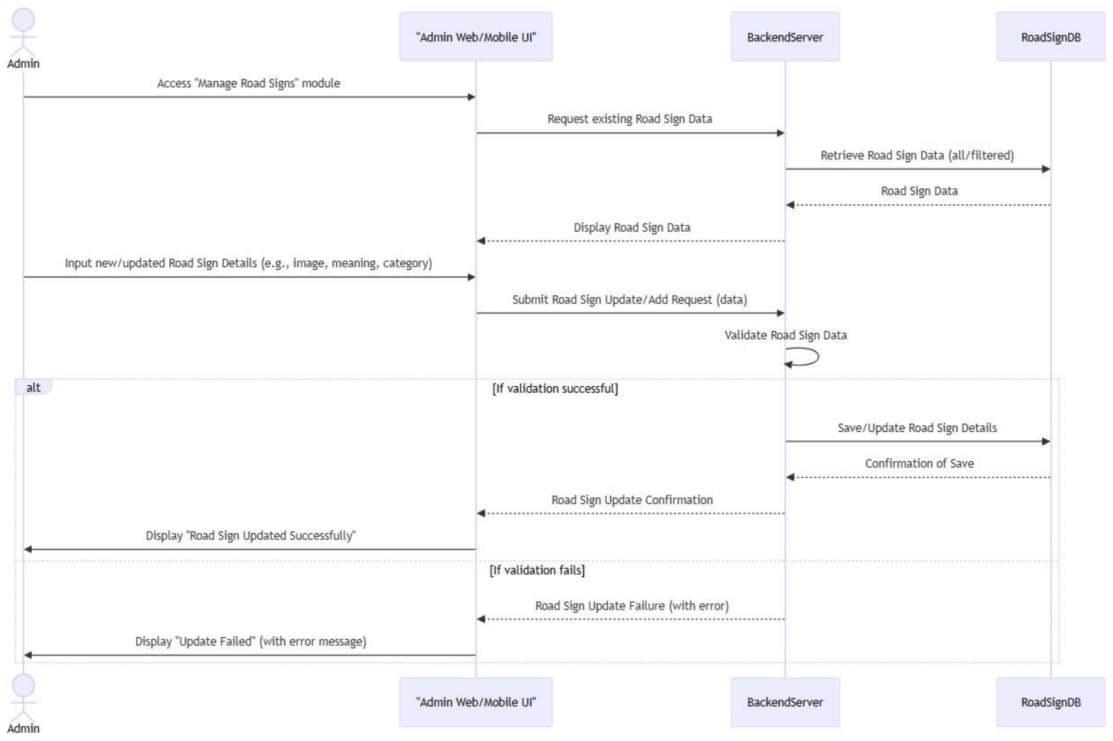
Admin
"Admin Web/Mobile UI"
BackendServer
RoadSignDB
Access "Manage Road Signs" module
Request existing Road Sign Data
Retrieve Road Sign Data (all/filtered)
Road Sign Data
Display Road Sign Data
Input new/updated Road Sign Details (e.g., image, meaning, category)
Submit Road Sign Update/Add Request (data)
Validate Road Sign Data
alt
[If validation successful]
Save/Update Road Sign Details
Confirmation of Save
Road Sign Update Confirmation
Display "Road Sign Updated Successfully"
[If validation fails]
Road Sign Update Failure (with error)
Display "Update Failed" (with error message)
Admin
"Admin Web/Mobile UI"
BackendServer
RoadSignDB

Figure 6: Sequence Diagram for Admin Updating Road Sign Data

5.5 **CLASS DIAGRAM:**

A. **PURPOSE:** The class diagram represents the static structure of the system, showing its fundamental building blocks: classes, their attributes, operations and the various relationships that define how these classes interact.

B. **CLASSES AND THEIR DESCRIPTIONS:**

The following classes represent the core entities within the Road Sign and Road State Mobile Notification Application:

1. **DRIVER:**

   - **Description:** Represents the primary end-users of the mobile application. This class encapsulates all information pertinent to a registered driver and their interactions with the system.
   - **Attributes:**
     - userID: Unique identifier for each driver.
     - name: The driver's full name.
     - role: The role of the user within the system (e.g., standard driver).
     - password: The driver's secure password for authentication.
   - **Methods:**
     - +Login(): Authenticates the driver's credentials.
     - +setNotificationPreference(): Allows the driver to modify their notification settings.
     - +getNotifications(): Enables the driver to retrieve their received notifications.
     - +reportRoadConditions(): Facilitates the driver's ability to submit new road incident reports.

2. **ROADSIGN:**

   - **Description:** Represents a static entity storing information about various road signs. This class forms the basis of the road sign directory feature.
   - **Attributes:**

- signID: Unique identifier for each road sign.
- image: Digital representation of the road sign's image.
- description: Detailed textual explanation of the road sign's meaning.
- type: Categorization of the road sign (e.g., warning, regulatory, informative).
- location: The geographical coordinates where the physical road sign is typically found.
- name: A descriptive name for the road sign.

- **Methods:**

  - +viewDetails(): Retrieves and displays comprehensive information about a specific road sign.

3. **ADMIN:**

- **Description:** Represents administrative users responsible for managing the application's data and overseeing its operations.
- **Attributes:**

  - adminID: Unique identifier for each administrator.
  - name: The administrator's full name.
  - password: The administrator's secure password for authentication.

- **Methods:**

  - +updateRoadSignDatabase(): Allows the admin to modify or add road sign information.
  - +manageCrowdsourcedReports(): Enables the admin to validate or reject submitted incident reports.
  - +monitorSystemLogs(): Enables the admin to check the performance of the system.

4. **INCIDENTREPORT:**

- **Description:** Represents a report submitted by a driver detailing a specific road incident or hazard.
- **Attributes:**
  - reportID: Unique identifier for each incident report.
  - type: Classification of the incident (e.g., pothole, accident, traffic jam).
  - location: Geographical coordinates of the incident.
  - timestamp: Date and time when the report was submitted.
  - status: Current state of the report (e.g., pending, verified, rejected).
- **Methods:**
  - +submitReport(): Handles the submission process of a new incident report.
  - +attachPhoto(): Allows for attaching images to the incident report.

5. **NOTIFICATION:**

- **Description:** Represents a message or alert sent by the system to a driver, informing them about road conditions, incidents, or other relevant updates.
- **Attributes:**
  - notificationID: Unique identifier for each notification.
  - message: The content of the notification.
  - timestamp: Date and time when the notification was generated/sent.
  - status: The delivery status of the notification (e.g., sent, delivered, read).
- **Methods:**
  - +sendNotification(): Initiates the process of dispatching the notification to the intended recipient.

6. **NOTIFICATIONPREFERENCE:**

- **Description:** Stores the personalized settings of a driver regarding the types and frequency of notifications they wish to receive.
- **Attributes:**
  - preferenceID: Unique identifier for a set of notification preferences.
  - preferenceType: Specifies categories of notifications (e.g., traffic alerts, weather warnings, road sign updates).
  - frequency: How often or under what conditions notifications are sent (e.g., real-time, daily digest).
- **Methods:**
  - +setPreference(): Allows the driver to modify their notification preferences.

7. **ROADSTATE:**

- **Description:** Represents aggregated real-time and near real-time data about road conditions, compiled from various sources. This class provides the basis for generating relevant notifications.
- **Attributes:**
  - stateID: Unique identifier for a specific road state record.
  - type: Classification of the road state (e.g., congestion, clear, hazardous weather).
  - description: Detailed information about the road state.
  - startLocation: Geographical start point of the affected road segment.
  - endLocation: Geographical end point of the affected road segment.
  - startTime: The time when the road state condition began.
  - endTime: The time when the road state condition ended or is expected to end.
  - source: The origin of the road state data (e.g., traffic API, weather API, crowd-sourced).
- **Methods:**

- +viewDetails(): Retrieves and displays detailed information about a specific road state condition.

8. **EXTERNAL APIs:**

- **Description:** Represents the external third-party Application Programming Interfaces (APIs) that the system integrates with to acquire real-time data, such as mapping, weather, and traffic information.
- **Attributes:**
  - +weatherdata: Represents raw or processed weather data obtained from a weather API.
  - +trafficdata: Represents raw or processed traffic data obtained from a traffic API.
  - +mapdata: Represents map-related data obtained from a mapping API.
- **Methods:**
  - +getWeatherData(): Method to call and retrieve data from the external weather API.
  - +getTrafficData(): Method to call and retrieve data from the external traffic API.
  - +getMapData(): Method to call and retrieve data from the external mapping API.

C. **RELATIONSHIPS BETWEEN CLASSES:**

The classes within the Road Sign and Road State Mobile Notification Application interact through various associations, defining the structural links and dependencies between them. Multiplicity notation (e.g., 1, *, 1..*) indicates the number of instances involved in a relationship.

1. **DRIVER and NOTIFICATIONPREFERENCE:** A DRIVER has exactly one NOTIFICATIONPREFERENCE, and conversely, a

NOTIFICATIONPREFERENCE belongs to exactly one DRIVER. This ensures personalized notification settings for each user.

2. **DRIVER and NOTIFICATION:** A DRIVER receives zero to many NOTIFICATION instances. Each NOTIFICATION instance is sent to exactly one DRIVERS instance, reflecting the delivery of alerts to users.

3. **DRIVER and INCIDENTREPORT:** A DRIVER submits zero to many INCIDENTREPORT instances. Conversely, each INCIDENTREPORT instance is submitted by exactly one DRIVER, supporting the crowd-sourcing feature.

4. **ADMIN and Management Classes (ROADSIGN, ROADSTATE, INCIDENTREPORT):** An ADMIN instance manages or verifies zero to many instances of ROADSIGN, ROADSTATE, and INCIDENTREPORT respectively. Each of these managed/verified instances is associated with exactly one ADMIN instance, illustrating the administrative control over system data.

5. **INCIDENTREPORT and NOTIFICATION:** An INCIDENTREPORT instance triggers zero to many NOTIFICATION instances. This allows a single reported incident to generate multiple alerts, potentially to different drivers.

6. **ROADSTATE and EXTERNAL APIs:** A ROADSTATE instance retrieves data from one to many EXTERNAL APIs instances. This signifies that ROADSTATE data is compiled from various external sources like weather, traffic, and mapping APIs.

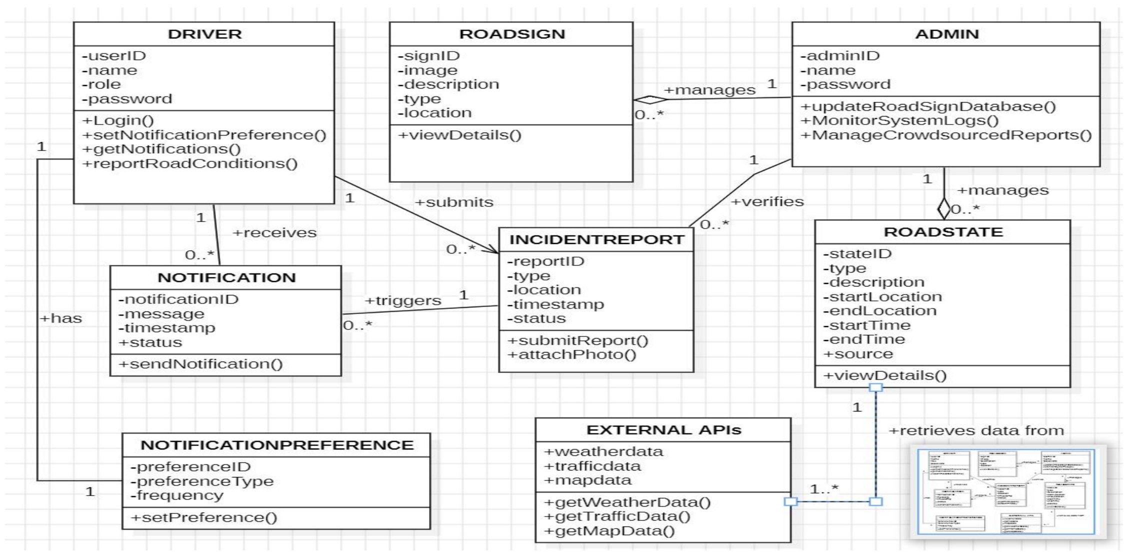

Fig 7: Class Diagram for the Mobile Application

5.6 DEPLOYMENT DIAGRAM:

A. **PURPOSE:** A Deployment Diagram in UML visualizes the physical architecture of a system. It shows the configuration of run-time processing nodes and the components that live on them. For this application, it will illustrate how the various software components (like the mobile app, backend server, databases, and external APIs) are deployed across different hardware environments.

B. **COMPONENTS:**

1. **NODES:**

Nodes represent the physical or logical computational resources where the system's software components are executed.

i. **MOBILE DEVICE:**

- **Description:** Represents a typical smartphone or tablet belonging to an end-user. This is the client-side environment where the primary user interaction with the application takes place.
- **Deployed Software:** Mobile Application (Android/iOS), GPS Module (often a hardware/software interface), Device Storage.

ii. **CLOUD SERVER INFRASTRUCTURE:**

- **Description:** A generic representation of a scalable, high-availability cloud-based server environment (e.g., utilizing services like AWS, Azure, or Google Cloud). This node hosts the core backend services and logic of the application.
- **Deployed Software:** Backend Server Application, API Gateway, Notification Service, Data Processing Module, Admin Portal / Web UI.

iii. **EXTERNAL API SERVICES:**

- **Description:** Represents the servers or infrastructure belonging to third-party providers of essential real-time data. These services are external to the application's core infrastructure but are vital for its functionality.
- **Deployed Software:** Weather API, Traffic API, Mapping API.

iv. **ADMINISTRATOR WORKSTATION:**

- **Description:** Represents a standard desktop or laptop computer used by an administrator for system management and oversight.
- **Deployed Software:** Web Browser.

2. **ARTIFACTS:**

Artifacts are the concrete, deployable pieces of software, while components are modular, replaceable parts of the system.

i. **WITHIN MOBILE DEVICE:**

- **Mobile Application (Android/iOS):** The executable application installed by the end-user, handling user interface, local processing, and communication with the backend. This artifact supports use cases like "Receive Notifications" and "Report Road Incident".
- **GPS Module:** Represents the software interface and access to the device's Global Positioning System hardware, crucial for location-based services.
- **Device Storage:** Represents the local storage on the mobile device used for application data, caching, and user-specific information.

ii. **Within Cloud Server Infrastructure:**

- **Backend Server Application:** The central application logic that processes user requests, manages data, and orchestrates interactions with other services and databases. It embodies processes like "Manage User Interface and Location" (P1) and "Manage Notifications" (P4) from the DFD.
- **API Gateway:** Acts as the single entry point for all client requests, distributing traffic to various backend components for scalability and handling initial authentication/security.
- **Notification Service:** A dedicated component responsible for sending push notifications to Mobile Applications, triggered by the Backend Server Application or Data Processing Module.

- **Data Processing Module:** Responsible for aggregating and processing data from various sources (crowd-sourced reports, external APIs) to determine relevant ROADSTATE information and trigger notifications. This aligns with "Process Road State Data" (P3) in the DFD.
- **Admin Portal:** The web-based interface that administrators use to manage the system, verify incident reports, and update road data.

iii. **Within External API Services:**

- **Weather API:** Provides real-time weather data to the system.
- **Traffic API:** Provides real-time traffic information (e.g., congestion, incidents).
- **Mapping API:** Provides geographical and map-related data for display and location services.

iv. **Within Administrator Workstation:**

- **Web Browser:** The standard application used by administrators to access the web-based Admin Portal.

3. **DATABASES:**

Databases represent the persistent data stores accessed by the application's components.

- **Road Sign Database (DS1):** Stores all static road sign information, managed by the Backend Server Application.
- **Incident Reports Database (DS2):** Stores crowd-sourced incident reports submitted by DRIVERS, also managed by the Backend Server Application.
- **User Preferences Database (DS3):** Stores individual DRIVERS' notification preferences, managed by the Backend Server Application.

4. **ASSOCIATIONS (COMMUNICATION PATHWAYS):**

Associations depict the communication links between nodes and the dependencies between components/artifacts.

i. **External Network Communication (Internet: HTTPS/TCP/IP):**

- **Mobile Device to Cloud Server Infrastructure**: The primary channel for Mobile Application to send requests (e.g., submit reports) and receive data (e.g., map updates, notifications) from the backend.
- **Cloud Server Infrastructure to External API Services**: The backend system's way of querying and retrieving data from external Weather API, Traffic API, and Mapping API services.
- **Administrator Workstation to Cloud Server Infrastructure**: How administrators access the Admin Portal hosted on the cloud.

ii. **Database Connectivity:**

- **Cloud Server Infrastructure to Road Sign Database (DS1), Incident Reports Database (DS2), User Preferences Database (DS3):** The Backend Server Application performs CRUD Operations (Create, Read, Update, Delete) on these databases to manage application data.

iii. **Internal Component/Artifact Interactions (Dependencies):**

- Mobile Application Uses GPS Module and Stores data in Device Storage.
- Backend Server Application Processes data from Data Processing Module, and Data Processing Module Provides Data to Backend Server Application.
- Backend Server Application Triggers the Notification Service.
- API Gateway Routes Requests to both Backend Server Application and Admin Portal.
- Data Processing Module Requests Data from Weather API, Traffic API, and Mapping API.
- Notification Service performs Push operations to the Mobile Application.
- Admin Portal performs Admin Operations by interacting with the Backend Server Application.
- Web Browser Accesses UI provided by the Admin Portal.

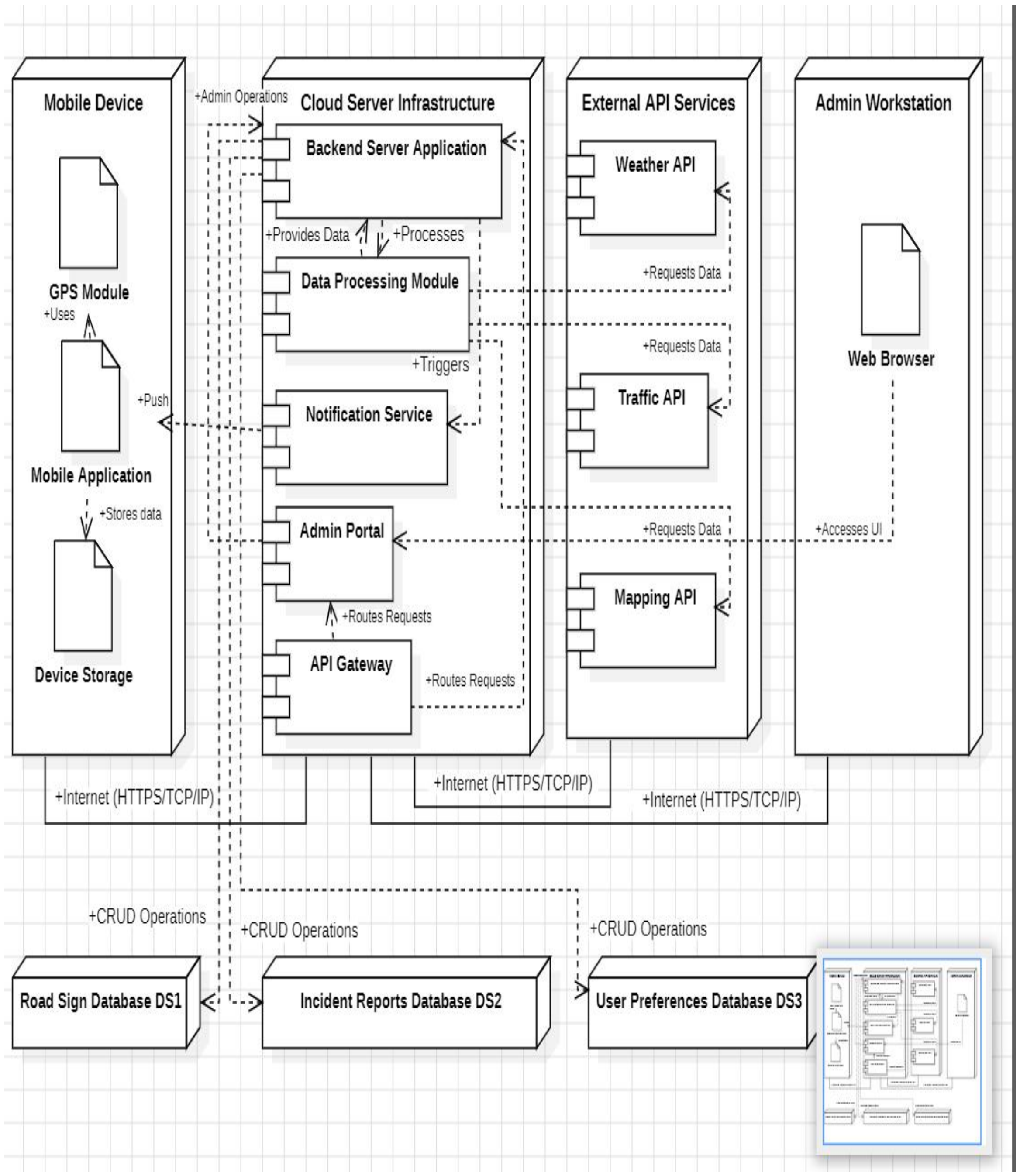

Fig 8: Deployment Diagram for the Mobile Application

6. **CONCLUSION:**

The **Road Sign and Road State Mobile Notification Application** is designed as a comprehensive solution to enhance road safety and traffic efficiency by leveraging mobile technology and real-time data. Through a detailed system modeling approach, this report has provided a clear blueprint of the application's functionality, structure, and deployment.

The **Context Diagram** sets the system boundaries, illustrating its interaction with key external stakeholders and services. The **Data Flow Diagram (Level 1)** delves deeper into the system's internal processes and data management. The **Use Case Diagram** defines the functional scope from the perspective of its users, while the **Sequence Diagrams** (for "Receive Road Condition Notifications," "Report Road Incident," and "Admin Updates Road Sign Data") meticulously detail the dynamic interactions between system components for critical functionalities. The **Class Diagram** provides a static, object-oriented view of the system's data and behavior. Finally, the **Deployment Diagram** outlines the physical architecture, showing how software components are deployed across hardware nodes and how they communicate.

Together, these diagrams provide a robust and detailed foundation for the development team, ensuring a shared understanding of the system's design and facilitating a structured implementation process. The application's ability to integrate real-time data, support user contributions, and provide personalized alerts positions it as a valuable tool for improving road safety in regions facing significant road infrastructure challenges.