Task 4 Report: System Design and Modeling

System Design Diagrams for Road Sign and Road State Mobile Notification Application

This section provides a detailed explanation of the system design diagrams for the Road Sign and Road State Mobile Notification Application, offering a visual representation of the system's architecture, data flow, and interactions.

---

1. Context Diagram

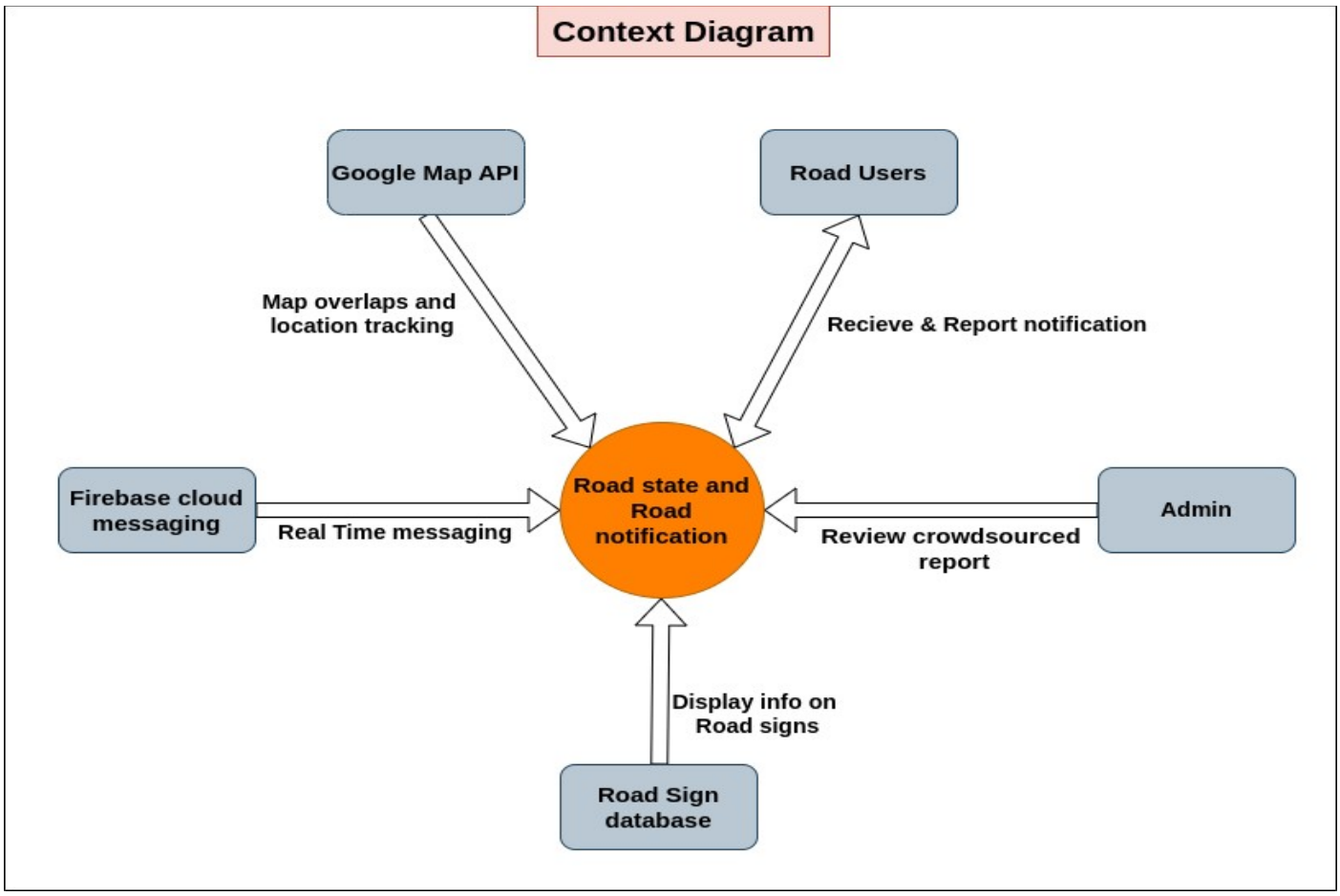

Diagram Name: Context Diagram
Description:

The Context Diagram provides a high-level overview of the Road Sign and Road State Notification System, illustrating its boundaries and interactions with external entities. It acts as a Level 0 Data Flow Diagram (DFD) by showing the system as a single process and detailing its interfaces with external actors and systems.

Explanation:

- **Road State and Road Notification (Central System):** This orange circle represents the entire mobile application system. It is the core component that processes information and interacts with all external entities.

- **External Entities:**
  - **Google Map API:** This external system provides "Map overlaps and location tracking" data to the Road State and Road Notification system. This indicates that the application will utilize Google Maps for displaying road information and tracking the user's location.

  - **Road Users:** These are the primary users of the application. They "Receive & Report notification," meaning they get alerts from the system and can also contribute by reporting hazards.

  - **Firebase Cloud Messaging:** This external service provides "Real Time messaging" to the system, suggesting its use for push notifications to users.

  - **Admin:** This user role interacts with the system to "Review crowdsourced report," indicating that user-submitted reports require moderation.

  - **Road Sign database:** This external data store "Display info on Road signs" to the system. This implies a preloaded or accessible database of road signs that the application uses for identification and notification.

Overall Purpose:

The Context Diagram clearly defines what is inside and outside the system, highlighting the essential interfaces required for its operation. It establishes that the application relies on external mapping services, a dedicated road sign database, and a messaging platform, while serving road users and being managed by administrators.

---

2. Data Flow Diagrams (DFDs)

Data Flow Diagrams (DFDs) illustrate how data is processed within the system. They show the flow of data from input to output, through processes, data stores, and external entities.

2.1. DFD Level 0

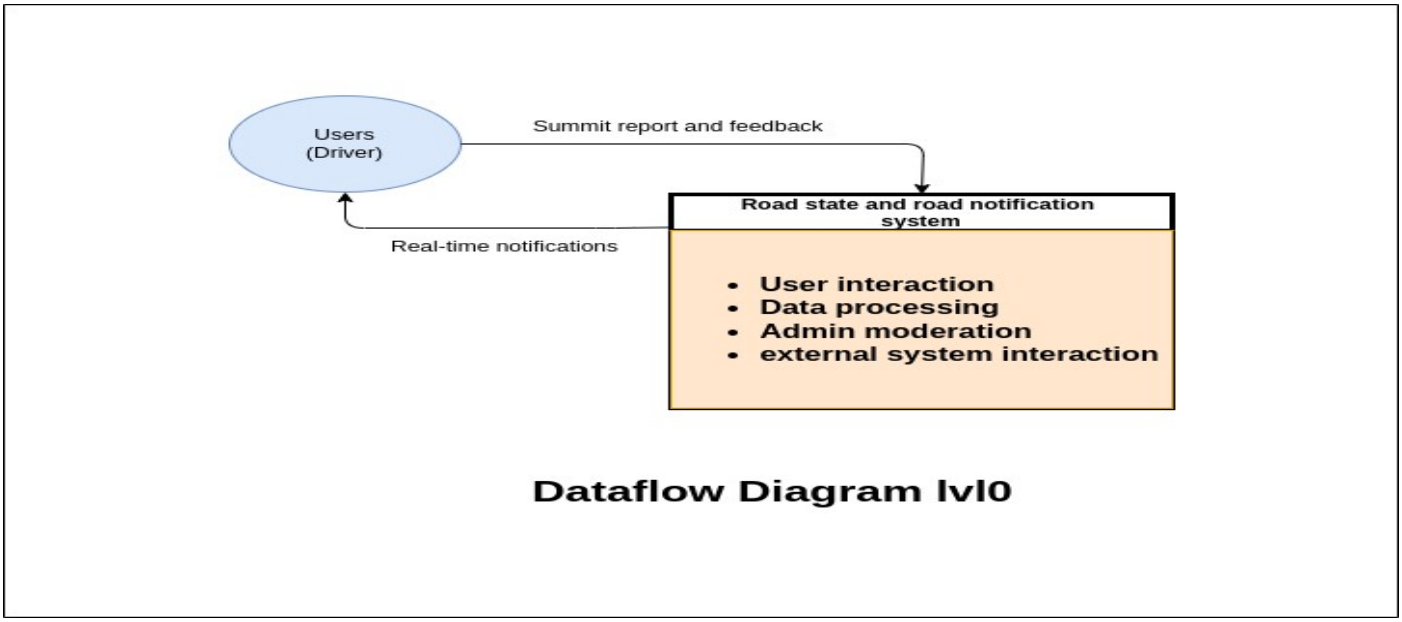

Diagram Name: Dataflow Diagram lvl0
Description:

DFD Level 0 presents the system as a single process, similar to the Context Diagram, but specifically focuses on the major data flows between the system and its external entities. It gives a broad overview of the system's primary inputs and outputs.

Explanation:

- **Users (Driver) (External Entity):** This represents the primary users of the application.
- **"Submit report and feedback" (Data Flow):** Data flows from the Users to the "Road state and road notification system," indicating that users can provide input in the form of incident reports and general feedback.
- **"Real-time notifications" (Data Flow):** Data flows from the "Road state and road notification system" to the Users, showing that the system delivers alerts and notifications to its users.
- **Road state and road notification system (Single Process):** This rectangle represents the entire application, encapsulating its core functionalities. The bullet points within it (User interaction, Data processing, Admin moderation, external system interaction) indicate the major high-level processes contained within the system.

Overall Purpose:

DFD Level 0 provides a fundamental understanding of the system's main interactions with its users, emphasizing the core inputs (reports, feedback) and outputs (notifications). It sets the stage for breaking down the system into more detailed sub-processes.

2.2. DFD Level 1

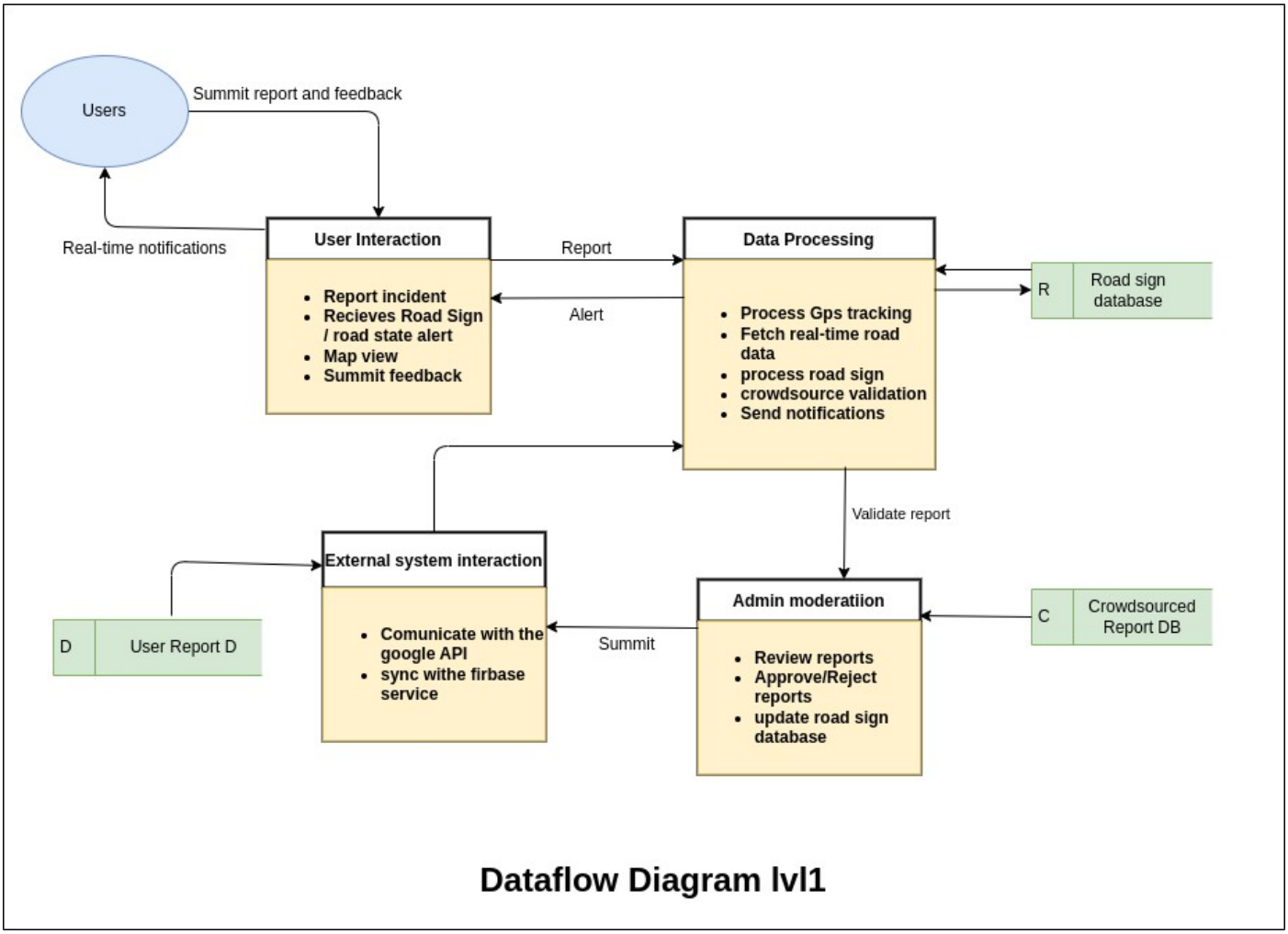

Diagram Name: Dataflow Diagram lvl1
Description:

DFD Level 1 decomposes the single process from DFD Level 0 into its major sub-processes. It shows how data flows between these internal processes, as well as between processes and external entities/data stores.

Explanation:
Users (External Entity): Remains the source of reports/feedback and receiver of notifications.

- **User Interaction (Process):**
  - Receives "Submit report and feedback" from Users.
  - Contains sub-functions like "Report incident," "Receives Road Sign / road state alert," "Map view," and "Summit feedback."
  - Sends "Report" data to "Data Processing."
  - Receives "Real-time notifications" from "Data Processing."
- **Data Processing (Process):**
  - Receives "Report" from "User Interaction."
  - Contains sub-functions like "Process Gps tracking," "Fetch real-time road data," "process road sign," "crowdsource validation," and "Send notifications."

    - Interacts with **Road sign database (Data Store 'R')** to fetch road sign information.
    - Sends "Alert" data back to "User Interaction."
    - Sends "Validate report" data to "Admin moderation."
- **Admin moderation (Process):**
    - Receives "Validate report" from "Data Processing."
    - Contains sub-functions like "Review reports," "Approve/Reject reports," and "update road sign database."
    - Interacts with **Crowdsourced Report DB (Data Store 'C')** for storing and managing user reports.
    - Sends "Summit" data to "External system interaction" (presumably for pushing validated reports).
- **External system interaction (Process):**
    - Receives "Summit" from "Admin moderation."
    - Contains sub-functions like "Comunicate with the google API" and "sync withe firebase service."
    - Interacts with **User Report D (Data Store 'D')**, likely representing the updated user reports for distribution.

Overall Purpose:

DFD Level 1 provides a more detailed view of the system's internal workings, breaking down the main functionalities into logical processes and showing how data flows between them and associated data stores. It clarifies the roles of user interaction, data processing, administration, and external system communication.

2.3. DFD Level 2 - Admin Moderation (AM)

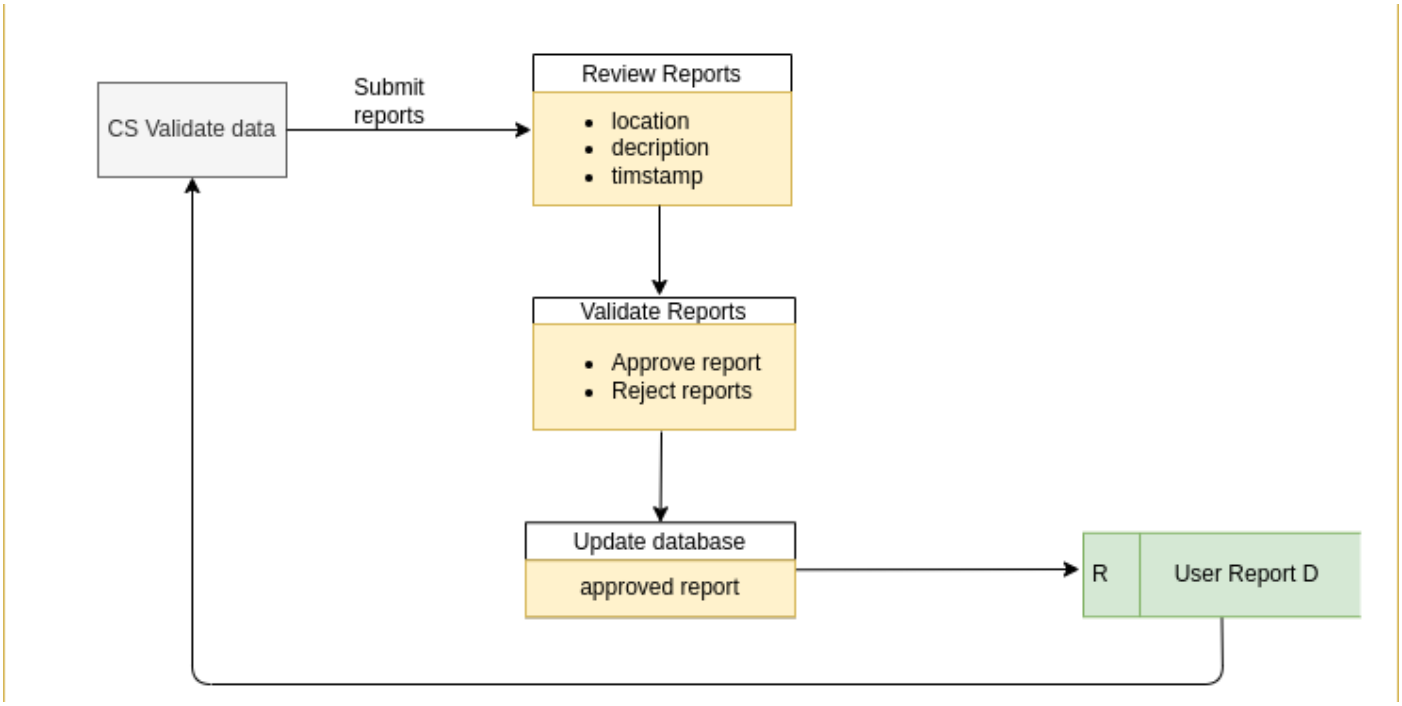

Diagram Name: DFD lvl2 Admin Moderation
Description:

This DFD Level 2 diagram focuses specifically on the "Admin moderation" process identified in DFD Level 1, detailing its sub-processes and data flows.

Explanation:

- **CS Validate data (Data Flow):** This appears to be an incoming data flow of crowdsourced validation data, likely leading into the review process.
- **Review Reports (Process):**
  - Receives "Submit reports" (which are effectively "CS Validate data" in this context).
  - Internal activities involve reviewing "location," "description," and "timestamp" of the reports.
  - Sends data to "Validate Reports."
- **Validate Reports (Process):**
  - Receives reports from "Review Reports."
  - Internal activities involve "Approve report" or "Reject reports."
  - Sends data to "Update database."
- **Update database (Process):**
  - Receives validated reports (specifically "approved report").
  - Sends approved report data to **User Report D (Data Store 'R')**.
  - A feedback loop from "User Report D" back to "CS Validate data" suggests that the updated database might feed into subsequent validation or reporting processes.

Overall Purpose:

This DFD provides a granular view of the administrative workflow for moderating crowdsourced reports, from initial review to final approval/rejection and database update.

2.4. DFD Level 2 - Data Processing (DP)

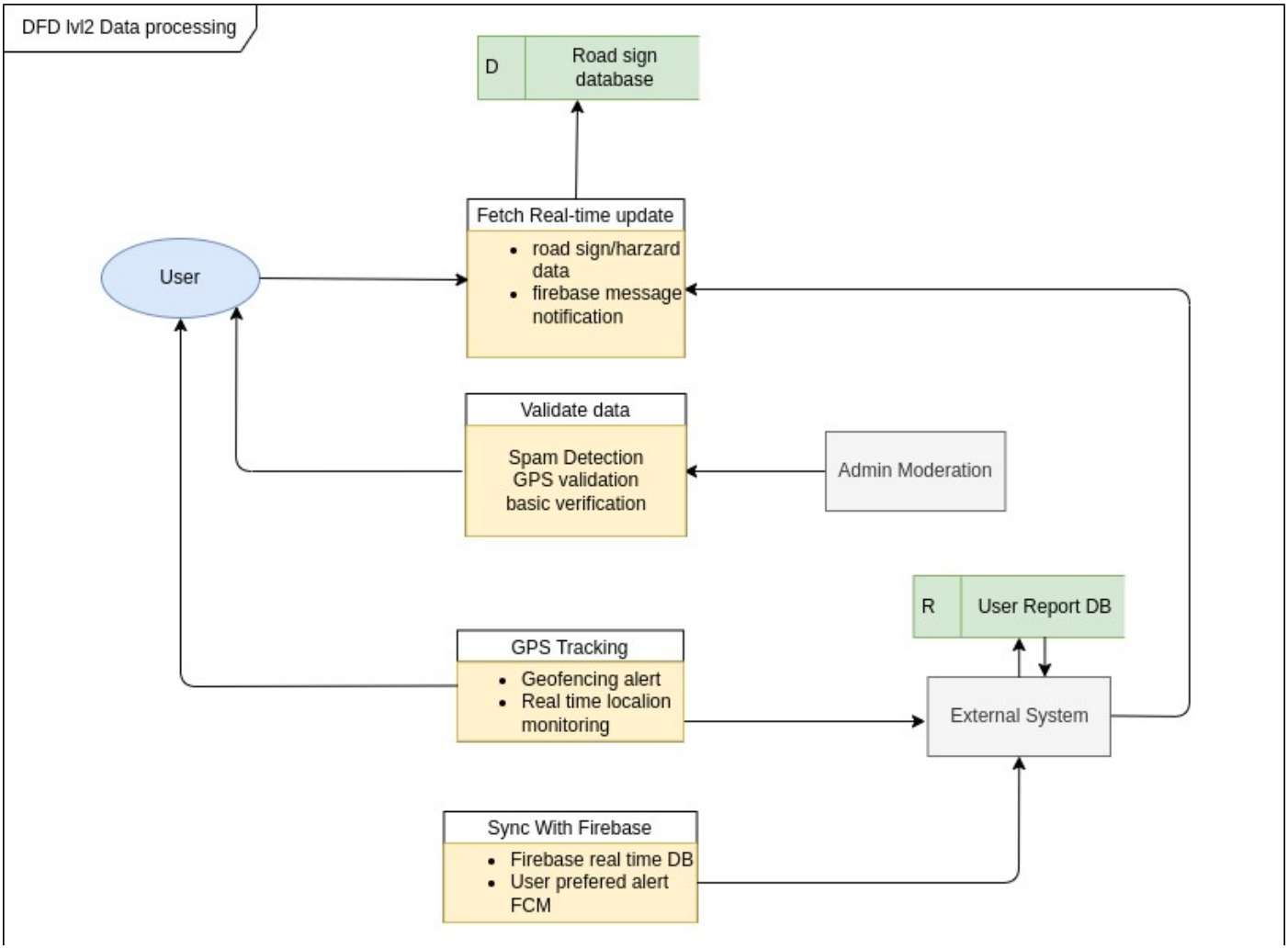

Diagram Name: DFD lvl2 Data processing
Description:

This DFD Level 2 diagram elaborates on the "Data Processing" process from DFD Level 1, breaking it down into its constituent sub-processes and their data interactions.

Explanation:

- **User (External Entity):** The source of real-time update requests and recipient of notifications.
- **Fetch Real-time update (Process):**
  - Receives requests from "User."
  - Activities include fetching "road sign/hazard data" and "firebase message notification."
  - Interacts with **Road sign database (Data Store 'D')** to retrieve road sign information.
  - Sends data to "Validate data."
- **Validate data (Process):**
  - Receives data from "Fetch Real-time update."
  - Activities include "Spam Detection" and "GPS validation basic verification."

    - Interacts with "Admin Moderation" (likely for rules or feedback on validation).
    - Sends validated data to "GPS Tracking."
- **GPS Tracking (Process):**
    - Receives validated data.
    - Activities include "Geofencing alert" and "Real time location monitoring."
    - Interacts with **User Report DB (Data Store 'R')** and "External System" (presumably to get/send location-based data related to user reports).
- **Sync With Firebase (Process):**
    - Receives data from "GPS Tracking" and "External System."
    - Activities include "Firebase real time DB," "User preferred alert," and "FCM."
    - Sends notifications to "User."

Overall Purpose:

This DFD provides a detailed understanding of how real-time data is fetched, validated, processed with GPS information, and ultimately synchronized with Firebase to deliver notifications to users. It highlights the critical steps involved in delivering timely alerts.

2.5. DFD Level 2 - User Interaction (UI)

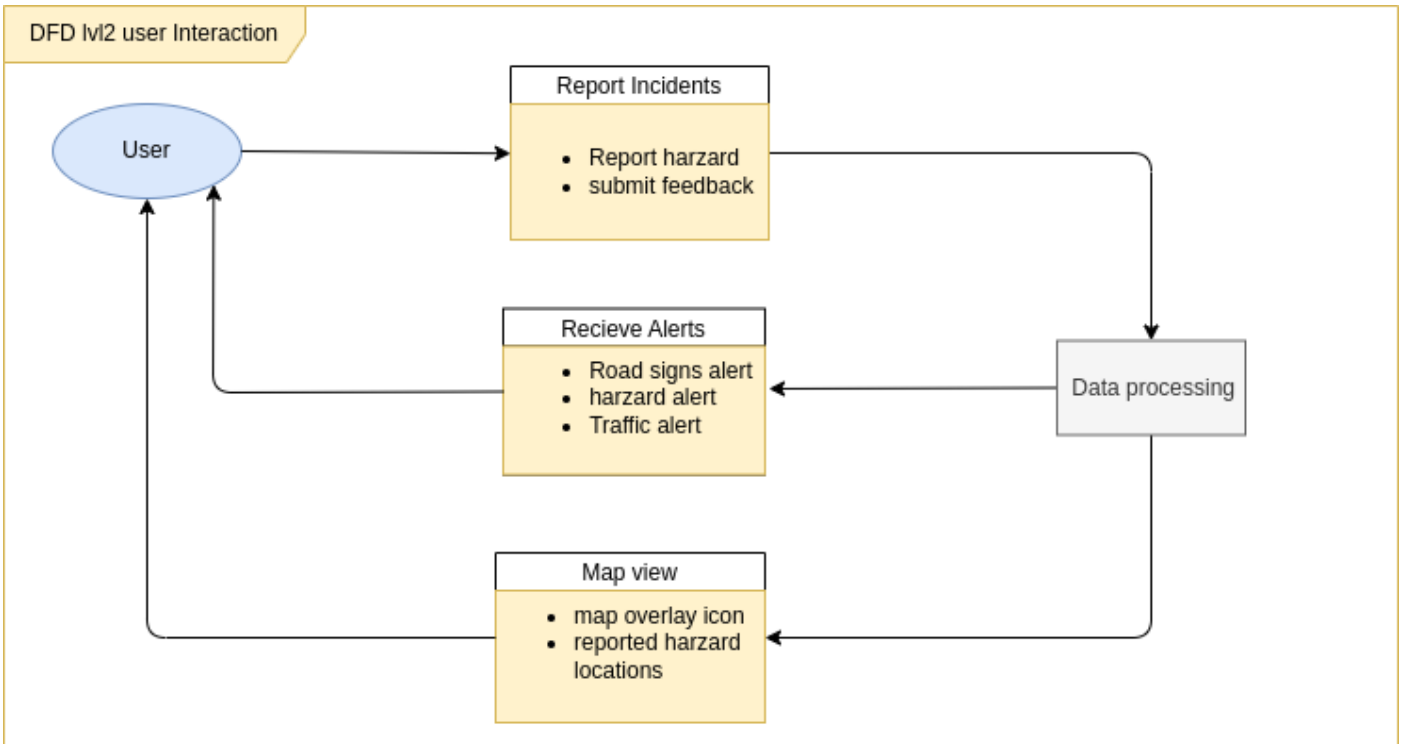

Diagram Name: DFD lvl2 user Interaction
Description:

This DFD Level 2 diagram focuses on the "User Interaction" process from DFD Level 1, detailing the ways users interact with the application.

Explanation:

- **User (External Entity):** The primary actor initiating actions and receiving information.

- **Report Incidents (Process):**
    - Receives input from "User" (to "Report hazard" and "submit feedback").
    - Sends reported data to "Data processing."

    - **Recieve Alerts (Process):**
    - Receives processed data/alerts from "Data processing."
    - Delivers "Road signs alert," "hazard alert," and "Traffic alert" to the "User."
- **Map view (Process):**
    - Receives map-related data from "Data processing."
    - Displays "map overlay icon" and "reported hazard locations" to the "User."

Overall Purpose:

This DFD clearly outlines the three main user-facing functionalities: reporting incidents, receiving various types of alerts, and visualizing information on a map. It shows the direct data flow between these user interactions and the "Data processing" core.

---

3. Sequence Diagrams

Sequence Diagrams illustrate the order of interactions between objects or participants in a particular scenario. They are time-ordered and show which messages are sent between objects and when.

3.1. Scenario 1: Real-Time Road Sign/State Notification

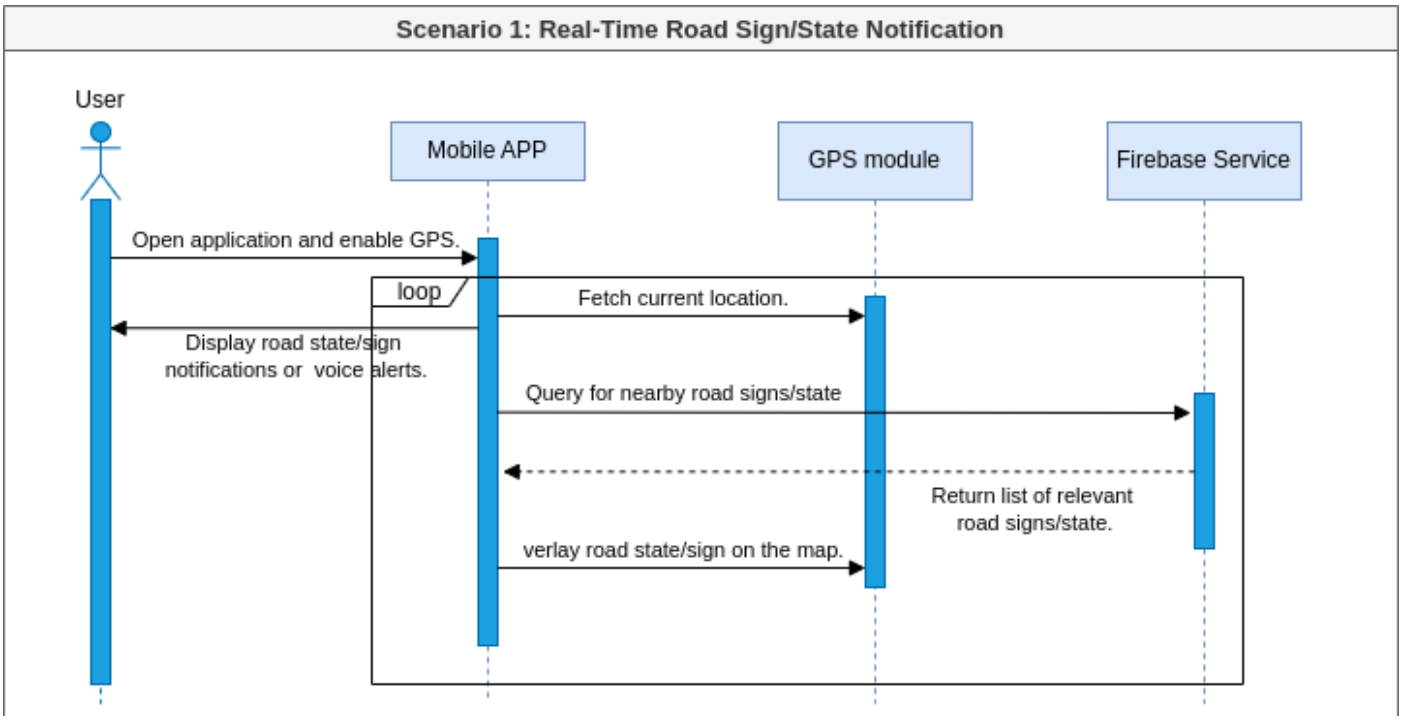

Diagram Name: SQD-scenario1.drawio.png
Description:

This sequence diagram depicts the flow of interactions when a user receives real-time notifications for road signs or road states as they drive.

Explanation:
Participants:

- **User:** The person operating the mobile application.
- **Mobile APP:** The front-end application running on the user's device.
- **GPS module:** The device's GPS hardware/software component responsible for location tracking.
- **Firebase Service:** The backend service (likely Firebase Realtime Database or Firestore) storing and managing road sign/state data.

Flow of Events:

- **User -> Mobile APP: Open application and enable GPS.** The user starts the app and ensures GPS is active.
- **Mobile APP -> GPS module: Fetch current location.** The app requests the current geographical coordinates from the device's GPS module.
- **GPS module --> Mobile APP: Return current location.** The GPS module provides the location data back to the app.
- **Loop (continually):** This indicates that steps 5 and 6 occur repeatedly as the user moves.
- **Mobile APP -> Firebase Service: Query for nearby road signs/state.** The app sends the current location to Firebase to retrieve relevant road signs or state information within a certain proximity.
- **Firebase Service --> Mobile APP: Return list of relevant road signs/state.** Firebase responds with a list of applicable road signs or hazard data.
- **Mobile APP -> User: Display road state/sign notifications or voice alerts.** The app presents the information to the user, either visually or through voice.
- **Mobile APP -> User: Overlay road state/sign on the map.** The app also visually overlays the relevant information onto the map interface.

Overall Purpose:

This sequence diagram clearly outlines the continuous, real-time process of location tracking, data querying from the backend, and notification delivery to the user, which is central to the application's core functionality.

3.2. Scenario 2: Crowdsourced Hazard Reporting

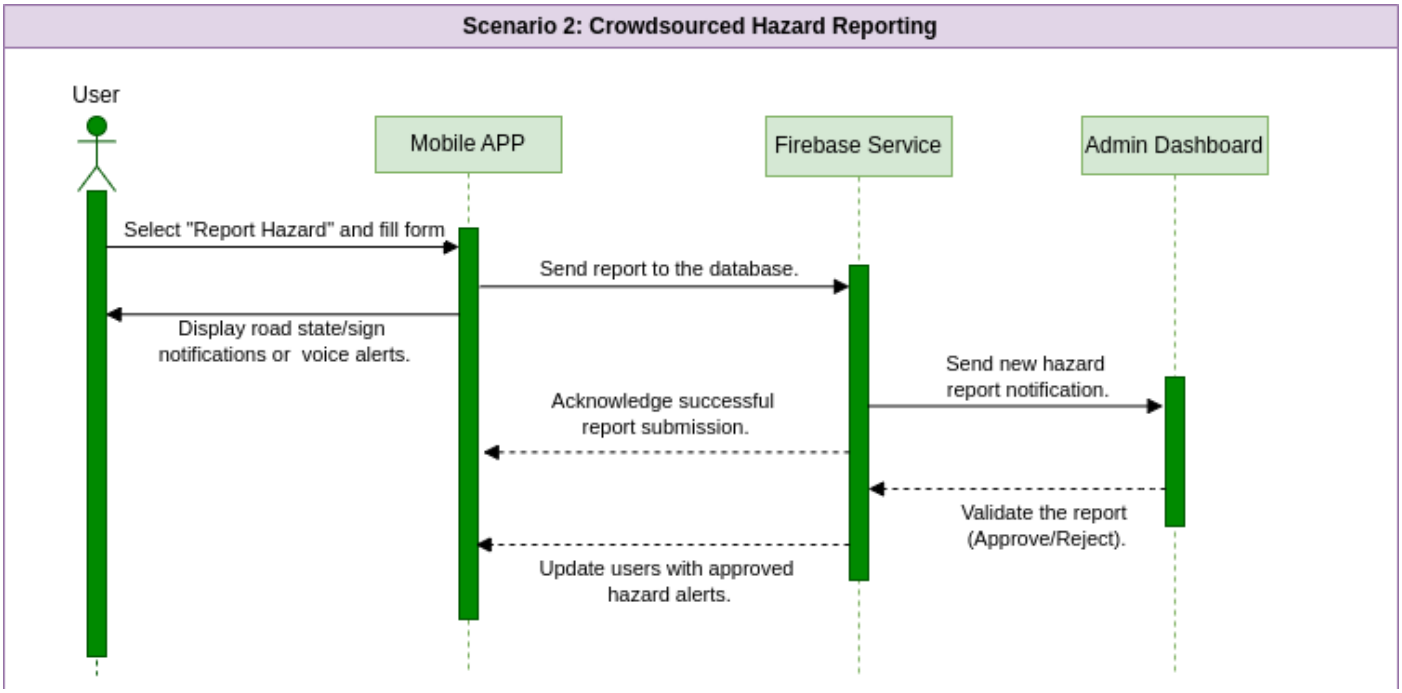

Diagram Name: SQD-scenario2.drawio.png
Description:

This sequence diagram illustrates the process a user follows to report a hazard and how that report is processed through the system.

Explanation:
Participants:

- **User:** The person reporting the hazard.
- **Mobile APP:** The client-side application.
- **Firebase Service:** The backend service for data storage and notifications.
- **Admin Dashboard:** The administrative interface used for moderation.

Flow of Events:

- **User -> Mobile APP: Select "Report Hazard" and fill form.** The user initiates the reporting process by selecting the option and providing necessary details (e.g., photo, description, location).
- **Mobile APP -> Firebase Service: Send report to the database.** The app transmits the submitted report data to the Firebase backend.
- **Firebase Service --> Mobile APP: Acknowledge successful report submission.** Firebase confirms that the report has been received.
- **Firebase Service -> Admin Dashboard: Send new hazard report notification.** Firebase triggers a notification or update to the Admin Dashboard, indicating a new report needs review.
- **Admin Dashboard --> Firebase Service: Validate the report (Approve/Reject).** An administrator reviews the report via the Admin Dashboard and decides to approve or reject it.

- **Firebase Service --> Mobile APP: Update users with approved hazard alerts.** If approved, Firebase pushes updates to other users' mobile apps, notifying them of the new validated hazard.

Overall Purpose:

This diagram effectively shows the end-to-end flow of crowdsourced reporting, from user submission to administrative moderation and subsequent dissemination of approved alerts to other users.

3.3. Scenario 3: Admin Report Moderation

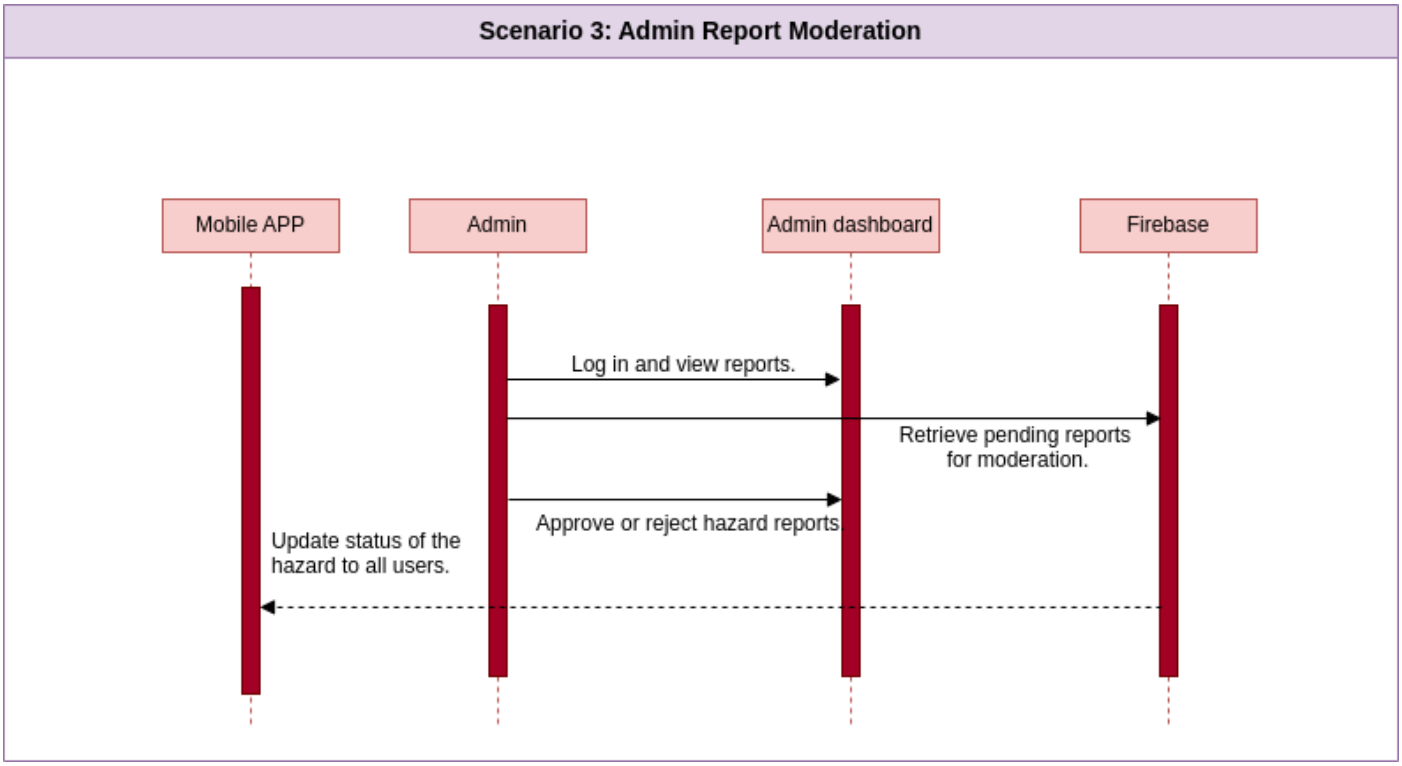

Diagram Name: SQD-scenario3.drawio.png
Description:

This sequence diagram details the process an administrator undertakes to review and moderate user-submitted reports.

Explanation:
Participants:

- **Mobile APP:** The client-side application (potentially receiving status updates).
- **Admin:** The administrator user.
- **Admin Dashboard:** The web or dedicated interface for administrative tasks.
- **Firebase:** The backend service where reports are stored and managed.

Flow of Events:

- **Admin -> Admin dashboard: Log in and view reports.** The administrator accesses the Admin Dashboard and requests to see pending reports.
- **Admin dashboard -> Firebase: Retrieve pending reports for moderation.** The Admin Dashboard queries Firebase for reports awaiting review.

- **Firebase --> Admin dashboard: Return pending reports.** Firebase sends the list of reports back to the Admin Dashboard.
- **Admin -> Admin dashboard: Approve or reject hazard reports.** The administrator reviews each report and performs the approval or rejection action.
- **Admin dashboard -> Firebase: Update report status.** The Admin Dashboard sends the updated status (approved/rejected) of the reports to Firebase.
- **Firebase --> Mobile APP: Update status of the hazard to all users.** Firebase then broadcasts this updated status to all relevant mobile application instances, affecting what users see regarding that specific hazard.

Overall Purpose:

This diagram provides a clear view of the administrative side of the system, specifically the workflow for managing user-generated content and ensuring the accuracy and relevance of the information disseminated to all users.

---

4. Class Diagram

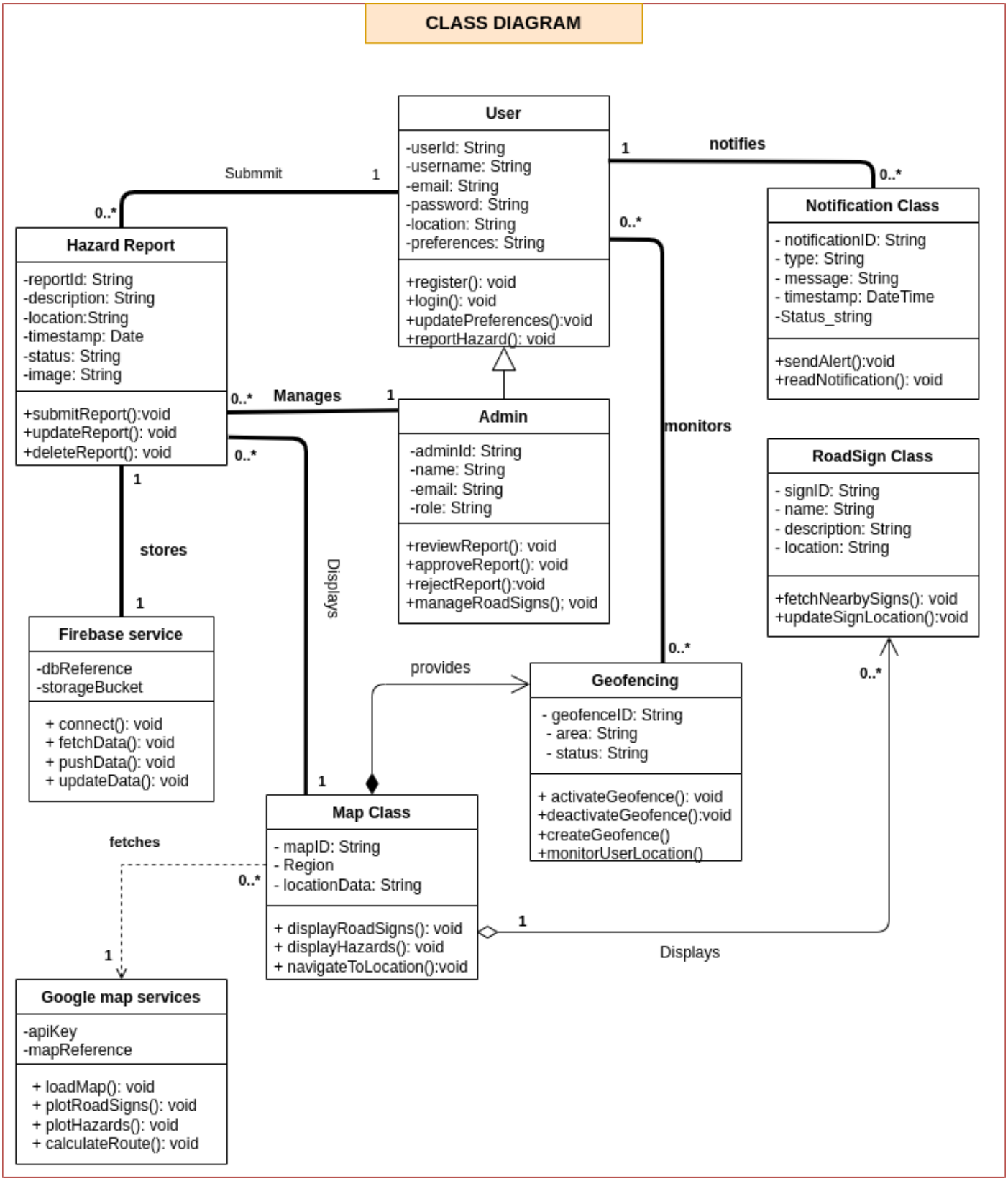

Diagram Name: Class Diagram
Description:

The Class Diagram models the static structure of the system, showing the classes, their attributes, operations (methods), and the relationships between them. It provides a blueprint for the application's object-oriented design.

Explanation:

User Class:

- **Attributes:** userID, username, email, password, location, preferences. These represent the data associated with each user.
  Operations: register(), login(), updatePreferences(), disableHazard(). These are the actions a user can perform.

Relationships:

- 0..* **to Hazard Report (Submits):** A user can submit zero or many Hazard Reports.
- 0..* **to Notification Class (Notifies):** A user can receive zero or many Notifications.
- 0..* **to Geofencing (Monitors):** A user's location is monitored by zero or more Geofencing instances.
- 0..* **to Map Class (Displays):** A user views information displayed by zero or many Map instances.

Hazard Report Class:

- **Attributes:** reportID, description, location, timestamp, date, status, image. These define a hazard report.
- **Operations:** submitReport(), updateReport(), deleteReport(). Actions related to managing reports.

Relationships:

- 0..* **to Firebase service (Stores):** Zero or many Hazard Reports are stored by one Firebase service.

Admin Class:

- **Attributes:** adminID, name, email, role. Information about an administrator.
- **Operations:** reviewReport(), approveReport(), rejectReport(), manageRoadSigns(). Actions an admin can perform.

Relationships:

- 1 **to Hazard Report (Manages):** One Admin manages zero or many Hazard Reports.
- 1 **to RoadSign Class (Manages):** One Admin manages zero or many RoadSigns.

Firebase service Class:

- **Attributes:** dbReference, storageBucket. Pointers to Firebase resources.
- **Operations:** connect(), pushData(), WorkspaceData(), updateData(). Core interactions with Firebase.

Relationships:

- 1 **to Hazard Report (Stores):** One Firebase service stores zero or many Hazard Reports.
- 1 **to Google Map services (Fetches):** One Firebase service fetches data from one Google Map services (or at least interacts with it).
- 1 **to Geofencing (Provides):** One Firebase service provides data/services to zero or many Geofencing instances.

Notification Class:

- **Attributes:** notificationID, type, message, timestamp, date, status_String. Details of a notification.

- **Operations:** sendAlert(), readNotification(). Actions related to notifications.

Relationships:

- 0..* **to User (Notifies):** Zero or many Notifications are sent to a User.

RoadSign Class:

- **Attributes:** signID, name, description, location. Details about a road sign.
- **Operations:** WorkspaceNearbySigns(), updateSignLocation(). Actions related to road signs.

Relationships:

- 0..* **to Map Class (Displays):** Zero or many RoadSigns are displayed by one Map Class.

Geofencing Class:

- **Attributes:** geofenceID, area, string, status. Defines a geofence zone.
- **Operations:** activateGeofence(), deactivateGeofence(), createGeofence(), monitorUserLocation(). Actions related to geofencing.

Relationships:

- 1 **to Map Class (Displays):** One Map Class displays the results/areas from Geofencing.

Map Class:

- **Attributes:** mapID, mapString, locationData. Data related to the map view.
- **Operations:** displayRoadSigns(), displayHazards(), MapsToLocation(). Actions related to map display.

Relationships:

- 1 **to Google Map services (Fetches):** One Map Class fetches data from one Google Map services.

Google Map services Class:

- **Attributes:** apiKey, mapReference. Credentials and references for the Google Maps API.
- **Operations:** loadMap(), getRoadSigns(), plotHazards(), calculateRoute(). Core Google Maps API interactions.

Overall Purpose:

The Class Diagram provides a detailed structural view of the system. It defines the core data entities, their properties, behaviors, and how they relate to each other, forming the foundation for the application's object-oriented implementation. It clearly shows how users, reports, notifications, road signs, and administrative functions are organized and interact within the system.

---

5. Deployment Diagram

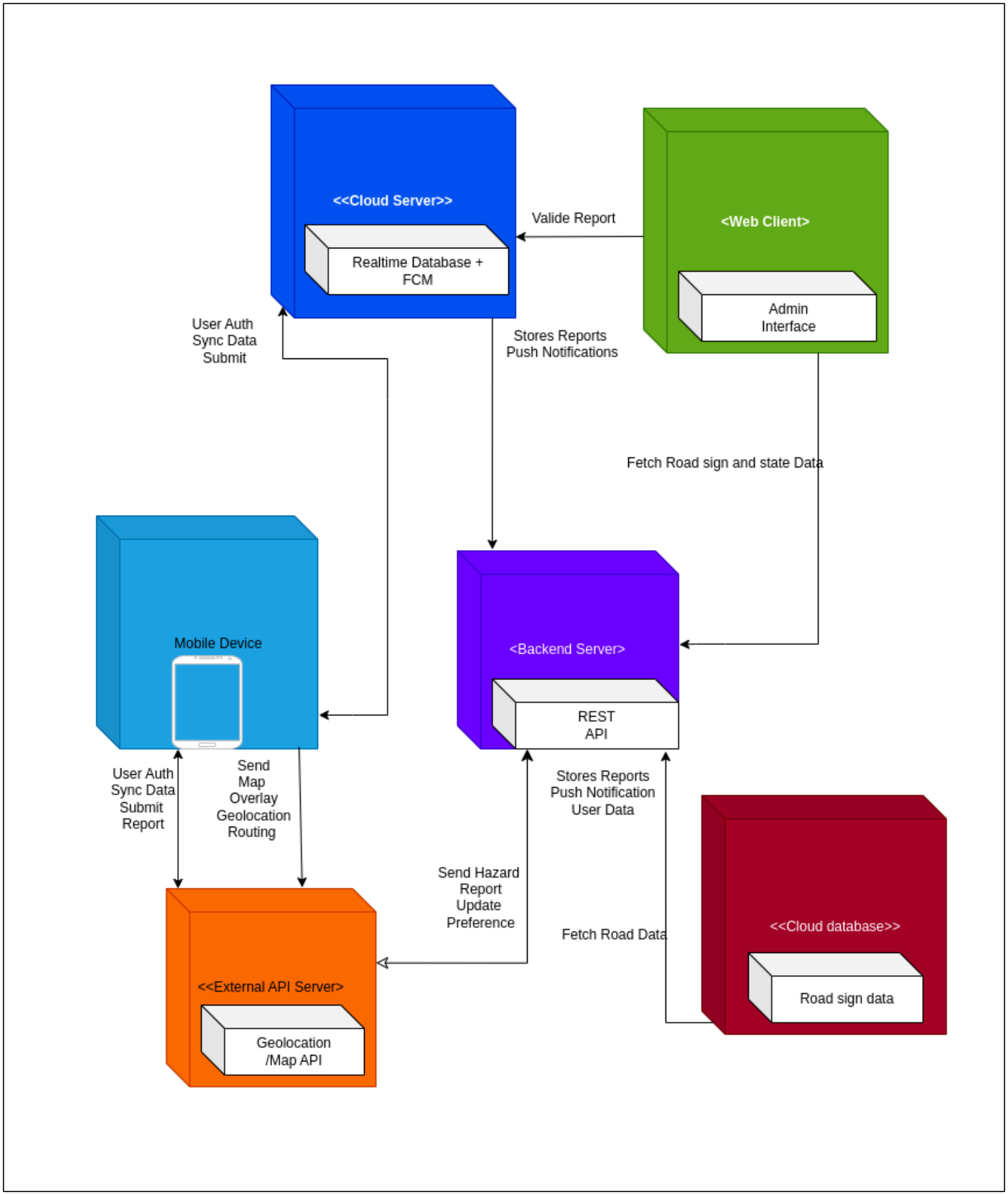

Diagram Name: Deployment Diagram
Description:

A Deployment Diagram illustrates the physical deployment of software components on hardware nodes. It shows the architecture of the system at runtime, including where software artifacts are located and how they connect.

Explanation:

Nodes (Physical Devices/Servers):

Mobile Device (Node):

Represents the user's smartphone (Android/iOS).

- Mobile App **(Component):** The core application running on the mobile device, containing the User Interface, GPS Module, and Notification Handler functionalities. This is where users interact with the app.
- Local Road Sign Cache **(Artifact/Data Store):** Indicates that road sign data can be stored locally on the device for offline access, as specified in the SRS.

Cloud Server (Node):

Represents the backend infrastructure where core services and databases reside.

- Backend API **(Component):** The application's server-side logic, handling requests from mobile apps.
- Database **(Component/Data Store):** This likely encompasses the Road Sign Database and Crowdsourced Report DB mentioned in the SRS. This is where all persistent data is stored.
- Admin Dashboard **(Component):** The web-based or specific application for administrators to manage the system and moderate reports.

External Systems/Services:

Google Maps API (External Service): A third-party service providing mapping, location tracking, and navigation data. It communicates with the Mobile App and potentially the Backend API.

Firebase Cloud Messaging (FCM) (External Service): A third-party service used for sending real-time push notifications to mobile devices. It primarily interacts with the Backend API to trigger notifications and the Mobile App to receive them.

Connections (Communication Paths):

- HTTP/HTTPS **(Connection):** Standard protocol for secure communication between the Mobile Device and the Cloud Server (Backend API).
- API Calls **(Connection):** Communication between the Mobile App/Backend API and external services like Google Maps API and Firebase Cloud Messaging.
- Database Connectivity **(Connection):** Internal communication between the Backend API and the Database on the Cloud Server.

Overall Purpose:

The Deployment Diagram provides a clear architectural view of where the different software components will physically reside and how they will communicate. It shows that the application is built as a client-server architecture, leveraging cloud services for backend processing, data storage, and external APIs for core functionalities like mapping and push notifications.

---

6. Use Case Diagram

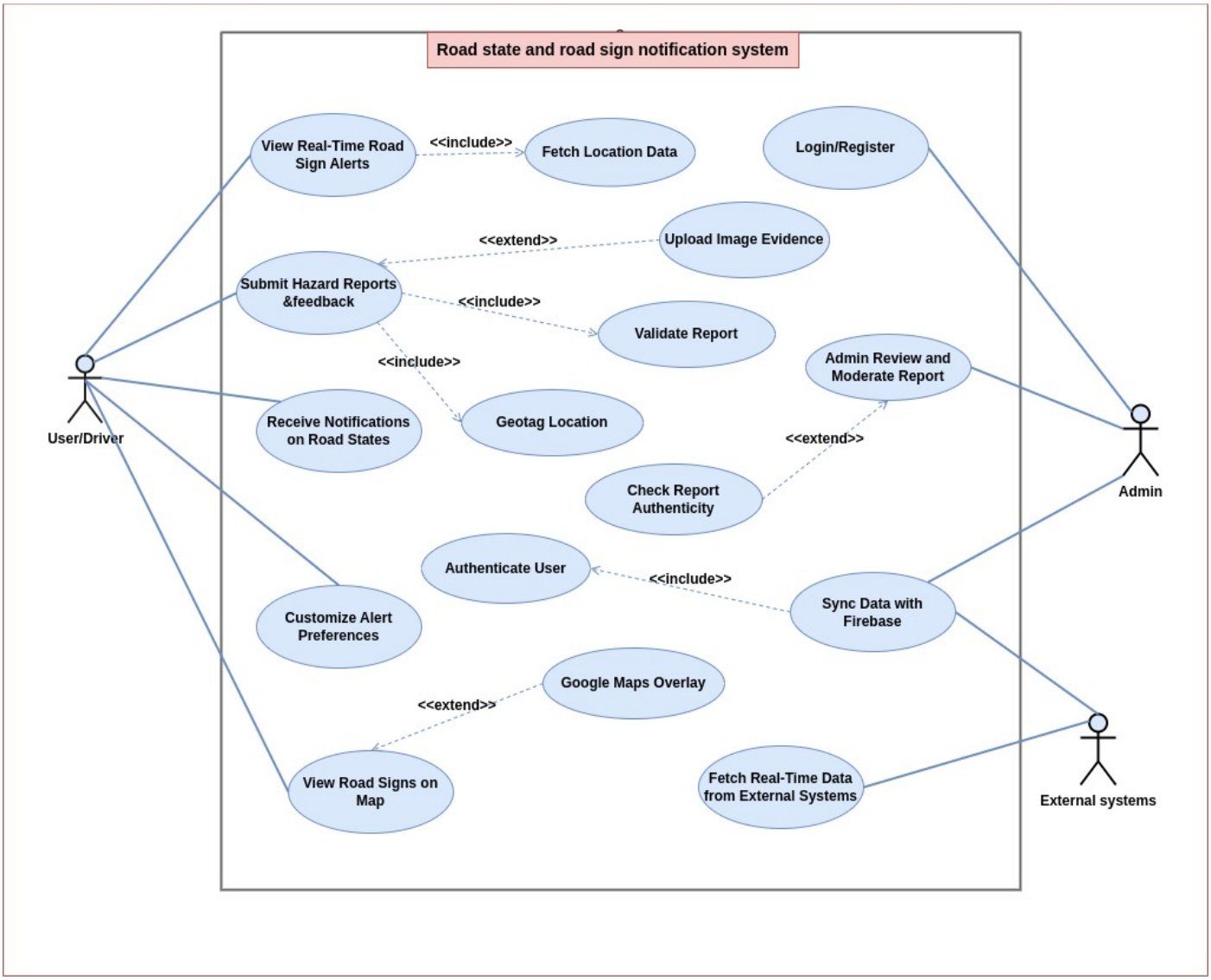

Diagram Name: Usecase Diagram
Description:

A Use Case Diagram depicts the functional requirements of a system in terms of use cases and actors. It shows what the system does from the perspective of its users, without specifying how these functions are implemented.

Explanation:
Actors:

- **User (Primary Actor):** Represents any individual interacting with the mobile application to receive information or report incidents.
- **Admin (Secondary Actor):** Represents a user with elevated privileges responsible for managing and moderating system data.
- **Google Map API (External System):** An external service that the system interacts with to provide map-related functionalities.
- **Firebase (External System):** An external service providing backend services like real-time database and messaging.

Use Cases (System Functions):

Associated with User:

- **Login/Register:** Allows users to access the application with an account.
- **View Road Signs:** Users can see information about various road signs.
- **Receive Road State Alerts:** Users get notifications about traffic, accidents, construction, etc.
- **Receive Road Sign Alerts:** Users get notifications when approaching road signs.
- **Report Hazard:** Users can submit information about new hazards (crowdsourcing).
- **Submit Feedback/Bug:** Users can provide input on the app's performance or suggest improvements.
- **Customize Alerts:** Users can set preferences for the types and frequency of alerts they receive.
- **Access Offline Road Sign Library:** Users can view road sign information even without internet connectivity.
- **View Map:** Users can see their location and relevant road information on a map.
- **Integrate with Navigation:** The app can work alongside or launch external navigation systems.

Associated with Admin:

- **Login/Register (Admin):** Admins need to authenticate to access their panel.
- **Manage User Accounts:** Admins can oversee user profiles.
- **Moderate Reports:** Admins review, approve, or reject user-submitted hazard reports.
- **Manage Road Sign Database:** Admins can add, update, or remove road sign information.

Interactions with External Systems:

- <<uses>> **GPS Tracking:** The system uses GPS for location tracking. This is a common functionality integrated across several use cases like Receive Road State Alerts and Receive Road Sign Alerts.
- <<uses>> **Send Push Notifications:** The system uses push notification services (like Firebase Cloud Messaging) to deliver alerts. This is used by Receive Road State Alerts and Receive Road Sign Alerts.
- <<uses>> **Google Map API:** The system uses Google Map API for View Map and Integrate with Navigation.
- <<uses>> **Firebase Service:** The system uses Firebase for Report Hazard, Manage Road Sign Database, and other backend operations.

Overall Purpose:

The Use Case Diagram clearly defines the functional scope of the application from the perspective of its various users and interacting external systems. It outlines the essential features and helps in understanding what the system is expected to do. It visually confirms that the application will allow users to receive alerts, report hazards, customize their experience, and access offline content, while admins manage the data and user-generated content, all supported by external mapping and backend services.