

**Table of Contents**

**1.0 Introduction………………………………………………………………………2**

**2.0 System Design………………………………...…………………………………..2**

2.1 Key Components and Considerations ...………...………………………………3

**3.0 System Modeling ………………………………………………………………...4**

**3**.1 Context Diagrams ……………………………………………………………....4

3.1.1 Deriving a Context Diagram ………………………………………................4

3.1.2 Components of the Context Diagram ...............................................................4

**4.2 Sequence Diagram ...……………………………………………………………..5**

4.2.1 Deriving a Sequence Diagram ………………………………………..............5

4.2.2 Example Scenarios ……………………………………………........................5

4.2.3 Sequence Diagrams for Viewing Road Signs ………………………………...5

**4.3 Class Diagrams ...………………………..……………………………………….7**

4.3.1 Derivation Process ………………………………………………….................7

4.3.2 Example Scenario ..............................................................................................9

**4.4 Use Case Diagram …………….…………………………………………………9**

4.4.1 Derivation Process ………………………………………………....................9

4.4.2 Example Scenario………………………………………………......................11

**4.5 Deployment Diagram ...…………………………………………………………12**

4.5.1 Nodes and Components.....................................................................................12

**5.0 Conclusion ………………………………………………………………………...13**

**6.0 References ………………………………………………………………………...14**

**-**

# 1. Introduction

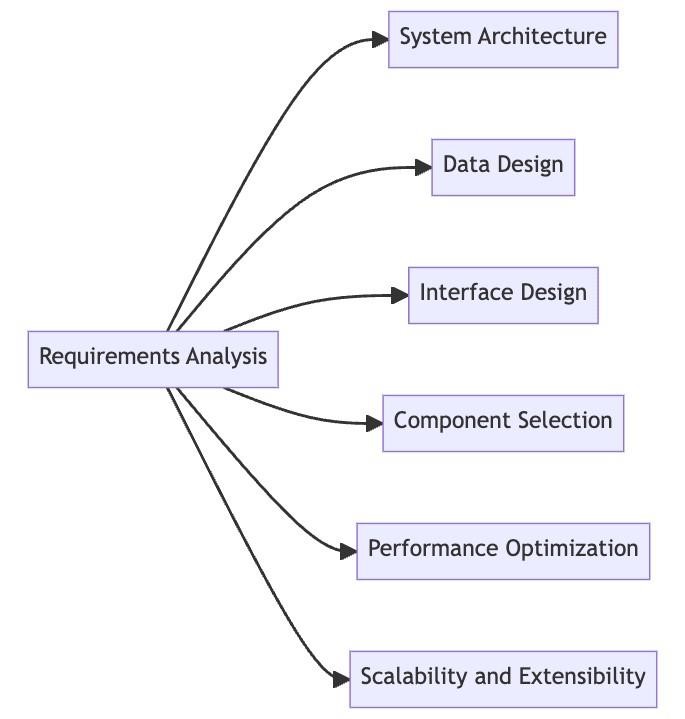
The Road State and Road Sign Mobile Application Notification system is designed to provide real-time updates to drivers about the state of the road and road signs. The aim is to enhance road safety and improve the driving experience by leveraging mobile technology to keep drivers informed about traffic conditions, road hazards, and important road signs.

# 2. System Design

Systems Design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements.

The goal is to create a well-organized and efficient structure that meets the intended purpose while considering factors like scalability, maintainability, and performance.

## 2.1 Key components and Considerations



System design encompasses various components and considerations, including:

1. **Requirements analysis:** Understanding the needs and expectations of the system's stakeholders
2. **System architecture:** A client server architecture was used.

=> Client (Front End): It includes ReactNative, Express

=> Server: Firebase server, and MongoDB server, Google Map Server.

1. **Data design:** MongoDB was chosen because it is faster and more scalable compared to SQL. The data will be sent using express.
2. **Interface design:** The touchscreen of the android app provides the user interface. Interface design: Firebase authentication, GoogleMap API, MongoDB provides us with the software interface.
3. **Component selection**: React Native implemented using JavaScript. Express was chosen to access our backend and Firebase
4. **Performance optimization**: Ensuring the system meets performance and efficiency requirements.
5. **Scalability and extensibility**: We choose JavaScript because it is lightweight and fast.

We will also choose modular design, leverage caching and load balancing.

# 3. System Modeling

Abstract models of the system were developed with each model representing a different view of the system. Doing so enabled us to understand the functionality of the system and to communicate with stakeholders.

There are different types of UML diagrams:

## 3.1. Context Diagram

A context diagram is a high-level, simplified view of a system that shows how it interacts with external entities. It provides a bird's-eye view of the system, its boundaries, and its interactions with the outside world. This helps in understanding the overall system scope and the primary data flows between the system and external entities.

**3.1.1 Deriving a Context Diagram**

The process of creating a context diagram involves several steps:

**Identify the System**: Define the system's scope and main functionality. For the Road State and Road Sign Mobile Notification Application, the system's purpose is to monitor road conditions and provide real-time notifications to users.

**Identify External Entities**: Determine the external actors that interact with the system. These are entities outside the system that either provide input to or receive output from the system. In our case, the main external entities are:

* Driver (User)
* GPS Service (Google Maps)
* Notification Service

**Define Interactions**: Establish the main interactions between the system and the external entities. This involves identifying the type of data that flows between the system and each external entity.

**Create the Diagram**: Draw the system at the center and connect it to the external entities using labeled arrows that indicate the data flow.

**3.1.2 Components of the Context Diagram**

* **System**: Represents the Road State and Road Sign Mobile Notification Application.
* **External Entities**: Represent the users and external services that interact with the system.
* **Data Flows**: Arrows that show the direction and type of information exchanged between the system and external entities.

The context diagram provides a high-level view of the system and its interactions with external entities.

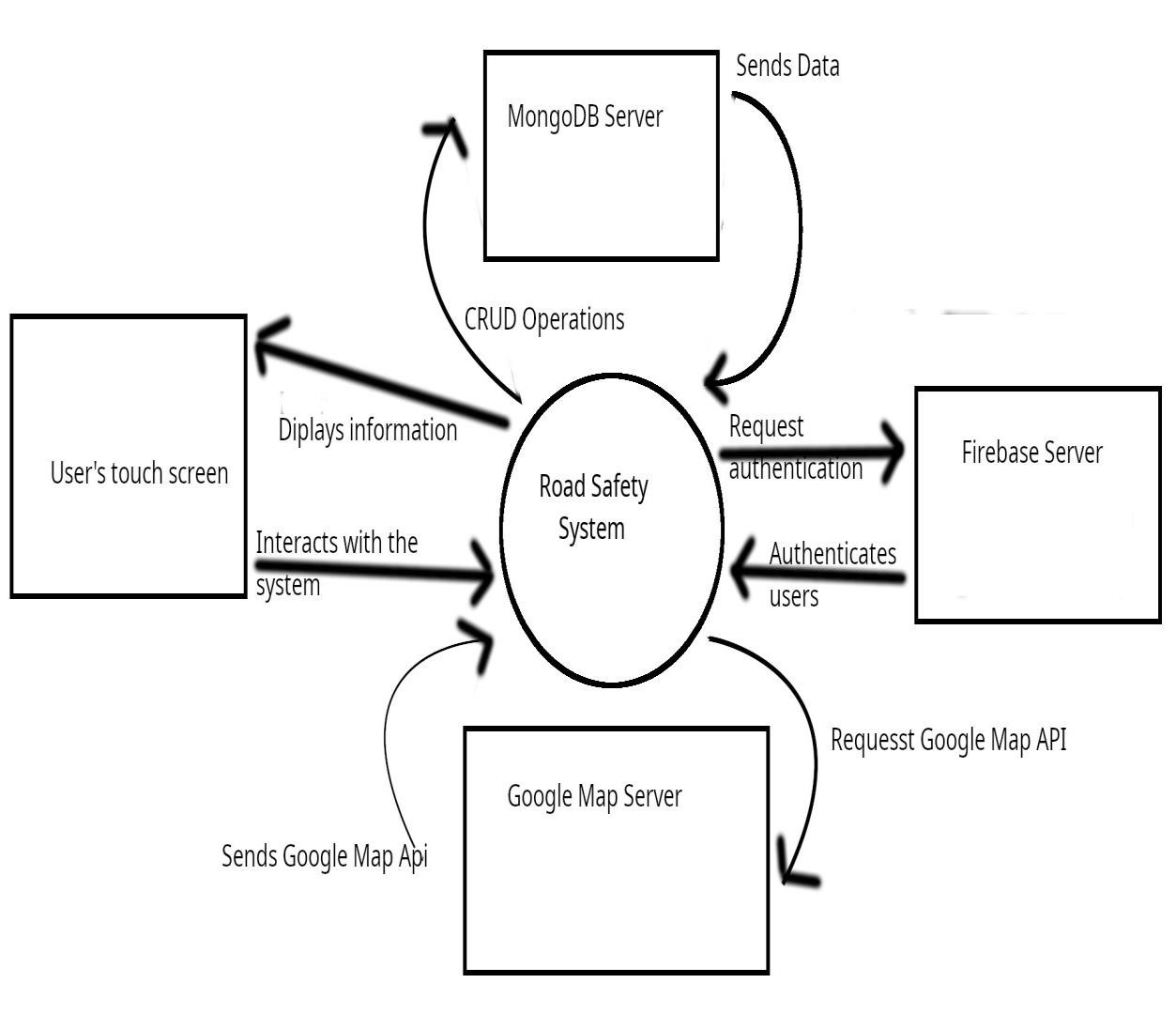
**Components:**

*User's Touch Screen*: Interface for users to interact with the Road Safety System. *Road Safety System*: Central system that handles user interactions, data processing, and communication with other servers.

*MongoDB Server*: Database for storing and retrieving road safety data.

*Firebase Server*: Provides user authentication services.

*Google Map Server*: Supplies mapping and location data through the Google Map API.



**Interactions**:

*User ↔ Road Safety System*: Users interact with the system to submit or receive information.

*Road Safety System ↔ MongoDB*: Performs data operations (CRUD).

*Road Safety System ↔ Firebase*: Requests and receives user authentication.

*Road Safety System ↔ Google Maps*: Requests and receives mapping data.

**Overall Flow:**

Users interact with the system via touch screens.

The system processes inputs, interacts with MongoDB for data, Firebase for authentication, and Google Maps for location services.

## 4.2. Sequence Diagram

## Sequence diagrams are a type of interaction diagram that show how objects interact in a particular scenario of a use case. They capture the sequence of messages exchanged between objects in order to carry out the functionality of the system.

**4.2.1 Derivation of Sequence Diagrams**

To derive sequence diagrams, follow these steps:

* **Identify Use Cases**: Start by identifying the primary use cases from the system requirements. Each use case represents a functionality that needs to be implemented.
* **Define Actors and Objects**: Identify the actors (users or external systems) and objects (system components) involved in each use case.
* **Determine Interactions**: Determine the sequence of interactions (messages) between the actors and objects to fulfill the use case.
* **Sequence of Messages**: Arrange the interactions in the sequence they occur, capturing the flow of control and data.
* **Visual Representation:** Draw the sequence diagram, representing actors and objects as vertical lines and messages as horizontal arrows.

**4.2.2 Example Scenario: Viewing Road Signs**

**Use Case**: Viewing Road Signs

**Actors**: User (Driver), Mobile Application, Server, Database, Third Party API

**Steps:**

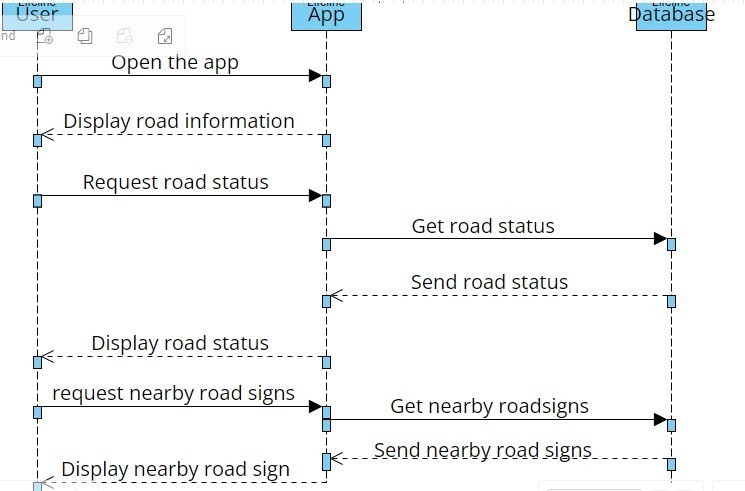
* The user requests to view nearby road signs.
* The mobile application requests the user's location from the API.
* The API service provides the user's current location.
* The mobile application queries the Road Sign Database for road signs near the user's location.
* The Road Sign Database returns the relevant road signs.
* The mobile application displays the road signs to the user.

**4.2.3 Sequence Diagram for Viewing Road Signs**

**Participants:**

* User
* Mobile Application
* Server
* Road Sign Database
* API Service

The sequence diagram illustrates the sequence of interactions between objects in a particular scenario. Below is the diagram:



*User*: Initiates actions such as opening the app or reporting a hazard.

*Mobile Application*: Sends requests to the backend server (e.g., user data, hazard reports). *Backend Server*: Processes the requests, stores/retrieves data from the database, and fetches additional data from third-party APIs.

*Database:* Stores and retrieves data as requested by the backend server.

*Third-Party APIs*: Provide additional data to the backend server when requested.

The interactions show the sequence of operations, starting from the user's action to the final response displayed in the mobile application.

## 4.3. Class Diagram

A class diagram is a type of static structure diagram that describes the structure of a system by showing its classes, attributes, methods, and the relationships among objects. It helps in visualizing the system's architecture and design. Let's break down the class diagram for the Road State and Road Sign Mobile Notification Application, explaining how we derived it from the requirements and use cases.

**4.3.1 Derivation Process**

The derivation of these classes stems from the system's requirements and the identified interactions between the user and the system, as outlined in the use cases and sequence diagrams:

**Identifying Key Entities**

From the use cases, it was clear that users need to view road signs, report road states, and receive notifications. These interactions pointed to the need for entities representing users, road signs, road states, and notifications.

**Defining Attributes**

Each entity was analyzed to determine the essential data that must be captured. For example, a user needs a unique ID, a name, and a location for personalized notifications and reports. Road signs need an ID, type, and location to be displayed accurately on the map.

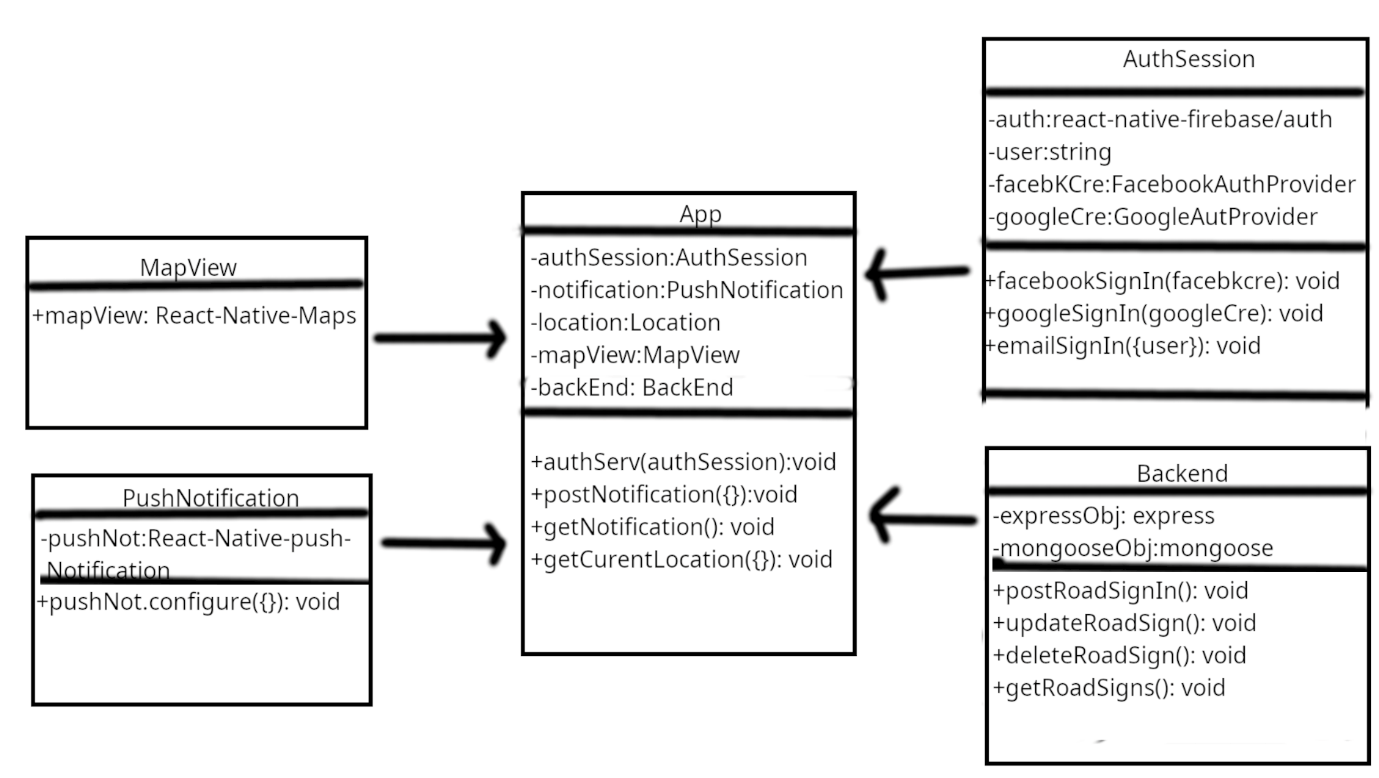
**Determining Methods**

Methods were defined based on the required actions each class needs to perform. For example, the User class requires methods to get user details and update location because these are fundamental actions users will perform within the app.

**Relationships**

Relationships between classes were established to reflect how these entities interact. For example, notifications are linked to users (each notification is intended for a specific user), and road states are linked to locations to ensure reports are geographically relevant.

It shows the object classes in the system and the associations between these classes.



**Explanation**

*MapView*: This component represents a map view using the React-Native-Maps library. *PushNotification*: This component handles push notifications using the React-Native-PushNotification library.

*App*: This is the main application component that integrates the MapView, PushNotification, AuthSession, and Backend components.

*AuthSession:* This component handles user authentication, including Facebook, Google, and email sign-in.

*Backend:* This component represents the backend server, which uses Express.js and Mongoose to provide various functionalities, such as posting, updating, deleting, and retrieving road signs.

The arrows in the diagram indicate the connections and interactions between the different components. For example, the App component interacts with the AuthSession, PushNotification, MapView, and Backend components to provide the overall functionality of the application.

**4.3.2 Example Scenario**

To illustrate how these classes interact, consider the scenario where a user reports a road state:

* User reports a road state via the app, invoking the getCurrentLocation() method to ensure their location is current.
* The RoadState object is created with the reported condition and location.
* The road state is saved to the database, and a **Notification** object is generated.
* The PushNotification() method of the Notification class is invoked, which sends alerts to other users near the reported location.

By following this structured approach to derive and define the classes, we ensure that the system is well-organized and capable of fulfilling its functional requirements efficiently.

## 4.4. Use Case Diagram

Use Case diagrams are a type of behavioral diagram defined by UML that captures the functional requirements of a system. They depict the interactions between actors (users or other systems) and the system itself, describing what the system should do without specifying how it does it. Use Case diagrams help to visualize the different ways that users might interact with the system.

**4.4.1 Derivation of Use Case Diagrams**

The derivation of use case diagrams involves the following steps:

1. **Identify Actors:** Determine who will interact with the system. Actors can be human users or other systems.
2. **Identify Use Cases:** Define the Various ways actors will interact with the system. Each use case represents a functional requirement of the system.
3. **Define Relationships:** Establish relationships between actors and use cases, and between different use cases if necessary.
4. **Define Relationships:** Establish relationships between actors and use cases, and between different use cases if necessary.
5. **Create the Diagram:** Draw the use case diagram to visually represent the actors, use cases, and their relationships.

For the Road State and Road Sign Mobile Notification Application, we have identified the following actors and use cases based on the system’s requirements and interactions.

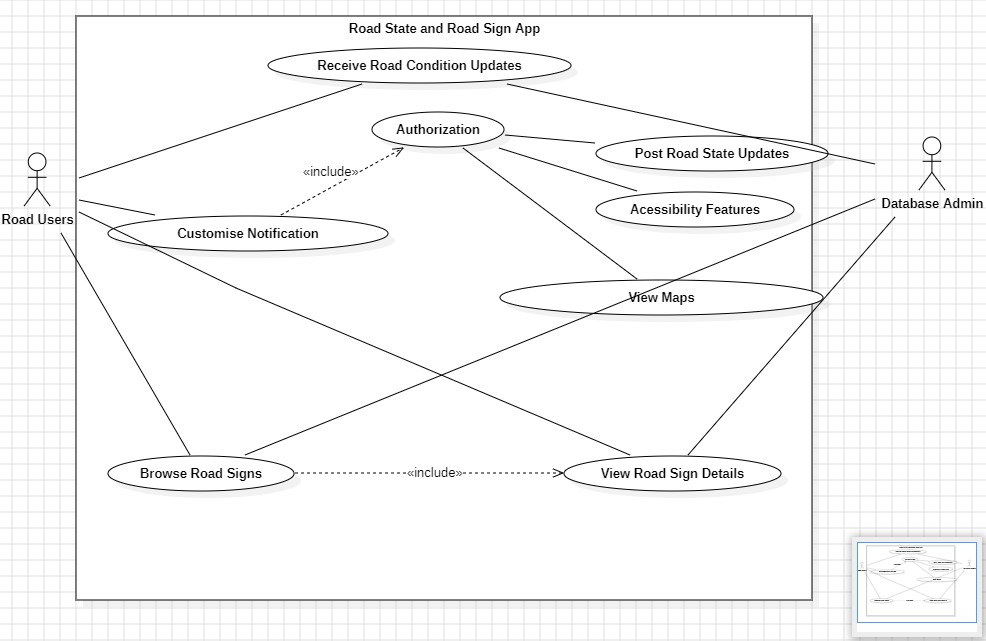
**Identified Actors**

* **Road Users (Drivers,Padestrians):** The primary user of the application who interacts with the system to receive notifications, view road signs and details, and report road conditions,etc
* **Admin:** A user responsible for managing data within the system, such as maintaining the database of road signs and road states.

**Identified Use Cases**

* View Road signs
* Report Road State
* Receive Notification
* Manage Data

**Diagram**



**Road Users:**

This is the primary actor interacting with the application.

The Road Users can perform various actions, as shown by the connections.

**Receive Road Condition Updates:**

Road Users can receive updates on the current road conditions.

**Customise Notification:**

Road Users can customize the notifications they receive**.**

**Authorization:**

Road Users need to be authorized to perform certain actions, such as posting road state updates.

**Post Road State Updates:**

Authorized Road Users can post updates on the current road conditions.

**Accessibility Features:**

The application provides accessibility features for Road Users.

**View Maps:**

Road Users can view maps within the application.

**Browse Road Signs:**

Road Users can browse the available road signs.

**View Road Sign Details:**

Road Users can view the details of specific road signs.

**Database Admin:** The Database Admin is responsible for managing the application's database.

**4.4.2 Example Scenario**

Let's consider an example scenario where a driver uses the application to report the current state of the road and receive notifications about nearby road signs.

Scenario: Reporting Road State and Receiving Notifications

* Driver opens the mobile application to report an icy road condition.
* The Report Road State use case is initiated: The driver selects the option to report a road state.
* The application prompts the driver to describe the condition (e.g., "icy road"). The driver inputs the condition and submits the report.
* The application saves the reported road state to the database and triggers the Receive Notifications use case
* The application processes the new road state information.
* The notification service sends a real-time notification to other drivers in the vicinity about the icy road condition. Meanwhile, the driver also views the nearby road signs using the View Road Signs use case:
* The driver selects the option to view road signs.
* The application retrieves the current location of the driver using the GPS service.
* The application fetches the relevant road signs from the database and displays them to the driver.

This use case diagram demonstrates the key functionalities and interactions within the Road State and Road Sign application, highlighting the roles and actions of the Road Users and the Database Admin**.**

## 4.5. Deployment Diagram

A deployment diagram is a type of UML diagram that shows the physical deployment of artifacts on nodes. It depicts the hardware and software components, their configurations, and the interactions between them. This diagram is essential for understanding the system's infrastructure and ensuring that it can support the application's requirements.

**4.5.1 Nodes and Components**

In the context of the Road State and Road Sign Mobile Notification Application, the main nodes and components involved are:

**1. Mobile Device**

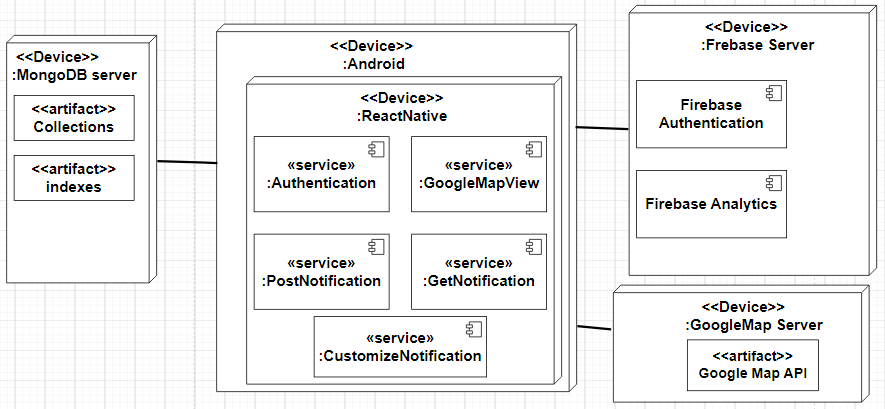
* Runs the mobile application.
* Components: User Interface (UI).

**2. Server**

* Hosts the business logic, database, and notification service.
* Components: Web Server, Application Server, Database Server, Notification Service.

**3. GPS Service**

* Provides location data to the application.
* An external service not hosted by the application but interacted with frequently.



This diagram provides an overview of the system architecture, highlighting the different components and their relationships within the mobile app and the backend server.

# 5. Conclusion

In conclusion, the system design and modeling of the Road State and Road Sign Mobile Application Notification presents a comprehensive framework for enhancing road safety and user experience. Through meticulous analysis and consideration of various factors such as realtime data integration, user interface design, and notification algorithms, the application aims to provide timely and relevant information to drivers, thereby reducing the risk of accidents and improving overall road management. By leveraging modern technologies and methodologies, this system strives to make significant contributions to the efficiency and effectiveness of road transportation systems. As technology continues to advance, further refinements and enhancements can be implemented to ensure that the application remains at the forefront of promoting safer and more informed driving practices.

# 6. References:

* [Chapter 5 – System Modeling (pace.edu)](http://csis.pace.edu/~marchese/CS389/L5/Chap5_summary.pdf) <https://www.tutorialspoint.com/system_analysis_and_design/index.htm>

[Systems modeling - Wikipedia](https://en.wikipedia.org/wiki/Systems_modeling)

* https://www.geeksforgeeks.org/data-modeling-in-system-design/