UNIVERSITY OF BUEA

P.O Boc 63,
Buea, South West Region
CAMEROON
Tel: (237) 3332 21 34/ 3332 26 90
Fax: (237) 3332 22 72

REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

ROAD STATE AND ROAD SIGN NOTIFICATION MOBILE APP DESIGN AND IMPLEMENTATION

A dissertation submitted to the Department of Computer Engineering, Faculty of Engineering and Technology, University of Buea, in fulfilment of the requirement for the end of second semester exam for the academic year 2023/2024

By:

GROUP 8

Option: Software Engineering

Supervisor:

Dr/Mr NKEMENI Valery

University of Buea

2023/2024 Academic Year

CERTIFICATION OF ORIGINALITY

We the undersigned, hereby certify that this dissertation entitled **ROAD STATE AND ROAD SIGN NOTIFICATION MOBILE APP** presented by the students;

- FOWEDLUNG ATSAFAC AGAFINA, **Matriculation number FE21A196**,
- KAMDEM KAMGAING GILLES CHRISTIAN, **Matriculation number FE21A209**,
- NEGUE KWAHAM MAEL GRACE, **Matriculation number FE21A252**,
- NGUEPI GNETEDEM PATERSON, **Matriculation number FE21A264**, and
- NOUPOUWO DONGMO STEPHANE MERCI, **Matriculation number FE21A283**

has been carried out by them in the Department of Computer Engineering, Faculty of Engineering and Technology, University of Buea under the supervision of **Dr/Mr NKEMENI Valery**.

This dissertation is authentic and represents the fruits of their own research and efforts.

Date_____

Students **Supervisor**

_____ _____

Head of Department

_____

DEDICATION

We dedicate this project to our parents and everyone who supported us.

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to everyone who supported us in the development of our project, " Road State and Road Sign Notification Mobile App."

First and foremost, we extend our deepest appreciation to our project supervisor, Dr/Mr Nkemeni Valery, for his invaluable guidance, encouragement, and support throughout this journey. Your insights, feedback, and expertise have greatly enhanced the quality and impact of our project.

We are also profoundly grateful to each member of our team for their dedication, hard work, and collaboration. This project was a true team effort, and it would not have been possible without the contributions and commitment of every individual. Thank you for your creativity, problem-solving skills, and perseverance in overcoming the challenges we faced.

Lastly, we give thanks to God for providing us with the strength, wisdom, and perseverance to see this project through to completion. Your guidance and blessings have been a cornerstone of our efforts.

ABSTRACT

This project report presents the development of a Road State and Road Sign Notification Mobile App designed to enhance road safety and improve the driving experience by providing real-time updates about road conditions and road signs. The application utilizes GPS, crowdsourced data, and image recognition technologies to deliver accurate and timely information to drivers.

The primary objective of this project is to develop an innovative, user-friendly mobile app that addresses the critical need for real-time road information. The app aims to notify drivers of current road conditions, including traffic congestion, accidents, roadworks, and weather conditions, as well as provide alerts about upcoming road signs and signals. By doing so, the app seeks to enhance situational awareness, reduce accidents, and improve overall traffic flow.

A comprehensive requirements gathering process was undertaken to ensure the app meets the needs of its diverse stakeholders, including end-users, project sponsors, and regulatory authorities. Stakeholders were classified into primary, secondary, and tertiary groups to identify their specific needs and contributions to the project.

Key features of the app include customizable alerts, voice-guided navigation, route optimization, and community engagement through crowdsourced data reporting. The app also incorporates accessibility features to accommodate all users, including those with visual impairments.

This project highlights the importance of integrating advanced technologies and user-centered design principles to develop a mobile app that significantly improves road safety and driver experience. By leveraging real-time data and user contributions, the app provides a valuable tool for drivers to navigate roads more safely and efficiently

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

UML: Unified Modelling Language

App: Application

ER: Entity Relationship Diagram

CHAPTER 1. GENERAL INTRODUCTION

1. Background and Context of Study

1.1. Background

Road safety is a critical concern worldwide, with millions of accidents occurring annually due to various factors, including poor road conditions, inadequate signage, and lack of real-time information for drivers. In many regions, including urban areas like Buea in Cameroon, the challenges of maintaining road infrastructure and providing timely information to drivers are particularly acute. This project aims to address these challenges by developing a Road State and Road Sign Notification Mobile App, leveraging modern technology to enhance road safety and improve driving experiences.

Road Safety Challenges:

- **Road Conditions:** Poorly maintained roads, potholes, and unexpected obstacles can significantly increase the risk of accidents. Drivers often lack real-time information about such hazards, leading to sudden maneuvers that can cause collisions.
- **Traffic Congestion:** Urban areas frequently experience heavy traffic congestion, which not only causes delays but also increases the likelihood of accidents. Drivers need timely updates about traffic conditions to plan their routes effectively.
- **Inadequate Signage:** Missing or unclear road signs contribute to driver confusion and non-compliance with traffic rules. Effective communication of road signs and their meanings is essential for safe driving.
- **Weather Conditions:** Weather-related hazards, such as heavy rain or fog, can impair visibility and road traction. Drivers need real-time weather updates to adjust their driving accordingly.

Technological Advancements:

Advancements in mobile technology, GPS, and data analytics offer new opportunities to address these challenges. By harnessing these technologies, it is possible to create a comprehensive solution that provides real-time road state and road sign information to drivers.

- **GPS and Real-Time Data:** GPS technology enables the accurate tracking of a vehicle's location, allowing for real-time updates about road conditions and traffic congestion. Crowdsourced data from other drivers can enhance the accuracy and timeliness of these updates.
- **Image Recognition:** Advanced image recognition technology can automatically identify road signs and provide drivers with relevant information and alerts. This can be particularly useful in areas where signage is inadequate or unclear.
- **Mobile Applications:** Mobile apps offer a convenient platform for delivering real-time information to drivers. With widespread smartphone adoption, mobile apps can reach a large user base and provide an accessible solution for road safety.

1.2. Context of Study

The study is conducted with a focus on Buea, Cameroon, an urban center facing significant road safety challenges. The city's unique geographical and climatic conditions, combined with its infrastructural limitations, make it an ideal case study for the development and implementation of a Road State and Road Sign Notification Mobile App. The project's findings and outcomes are expected to be applicable to other urban areas facing similar challenges, providing a scalable solution for improving road safety globally.

2. Problem Statement

Navigating roads efficiently and safely is a universal concern for drivers, yet current methods of accessing critical information about road conditions, weather hazards, and real-time traffic updates often fall short. Users face challenges in receiving timely notifications tailored to their preferences. In response to these challenges, there is a clear need for a robust road state notification application that provides users with accurate and customizable information in real-time. By addressing the gaps in existing solutions and incorporating features such as road sign notifications, weather hazard alerts, customizable notification preferences, and offline functionality, the project aims to empower drivers with the tools they need to navigate roads safely and efficiently.

3. Objectives of Study

3.1. General Objectives

To develop a comprehensive Road State and Road Sign Notification Mobile App that provides real-time updates on road conditions, traffic congestion, and road signage to enhance road safety and improve the driving experience in urban areas, with a focus on Buea, Cameroon.

3.2. Specific Objectives

- **Enhance Road Safety:** To reduce the number of accidents by providing drivers with timely notifications about hazardous road conditions, including potholes, roadworks, and accidents, thereby enabling safer driving decisions.
- **Improve Traffic Flow:** To alleviate traffic congestion by offering real-time updates and alternative route suggestions, helping drivers avoid traffic jams and reducing overall travel time.
- **Increase Signage Visibility and Compliance:** To improve driver compliance with traffic rules by using image recognition to detect road signs and provide clear, real-time alerts and explanations, particularly in areas where signage is inadequate or unclear.
- **Foster Community Engagement:** To engage the community in improving road safety by allowing users to report road conditions and hazards, thereby enhancing the accuracy and relevance of the information provided.
- **Enhance Situational Awareness:** To provide drivers with comprehensive situational awareness through real-time updates on road conditions, traffic, weather, and road signs, enabling better-informed driving decisions.

4. Proposed Methodology:

The core of this Road State and Road Sign Notification Mobile App revolves around real-time data acquisition, processing, and user notification. Utilizing GPS tracking, crowdsourced reports, the app gathers and aggregates information on road conditions, traffic congestion, and weather conditions. Algorithms analyze data provided by users on road state updates prior to publishing. Users receive timely notifications and alerts about current road conditions, hazards, and traffic situations, enhancing their ability to make informed driving decisions. The app also provides voice-guided navigation and customizable alerts, aiming to improve road safety and optimize the driving experience in urban areas like Buea, Cameroon.

5. Significance of the Study

The significance of this study lies in its potential to significantly improve road safety and driving efficiency in urban environments like Buea, Cameroon. By developing a Road State and Road Sign Notification Mobile App, the study addresses key issues such as hazardous road conditions, traffic congestion, and inadequate signage. The app provides real-time updates and alerts, which enhance driver awareness, reduce accidents, and optimize traffic flow. Additionally, it uses verification algorithms to engage the community in road safety efforts and ensure accurate information. The outcomes of this study offer a scalable solution that can be adapted for use in other cities, making a meaningful impact on road safety and urban mobility on a global scale.

6. Scope of the Study

The scope of this study focuses on the development, implementation, and evaluation of a Road State and Road Sign Notification Mobile App specifically designed for urban environments, with Buea, Cameroon, as the primary case study. The study encompasses several key aspects:

- **Geographic Scope:**
    - **Primary Focus:** Buea, Cameroon, as the urban setting for testing and deploying the app.
    - **Potential Expansion:** The app's design and findings are intended to be adaptable for similar urban areas facing road safety and traffic management challenges.

- **Functional Scope:**
    - **Core Features:** Real-time updates on road conditions, traffic congestion notifications, road sign recognition, and weather condition alerts.
    - **Additional Features:** User reporting of road hazards, voice-guided navigation, and customizable alerts for different road conditions and traffic situations.

- **Technological Scope:**
    - **Technologies Utilized:** GPS for location tracking, crowdsourced data for real-time updates, image recognition for detecting road signs, and integration with weather services for weather updates.
    - **Development Tools:** Mobile app development frameworks and platforms for building, testing, and deploying the application.

7. Delimitation of Study

The delimitation of this study defines the specific boundaries and limitations set for the development and implementation of the Road State and Road Sign Notification Mobile App, focusing on certain aspects while excluding others to maintain a clear and manageable scope. The key delimitations of this study are:

- **Geographic Limitation:**
    - **Focused Area:** The study is confined to the urban area of Buea, Cameroon, for the initial development and testing of the app. While the app's design is scalable for other cities, the scope of this study does not include deployment or testing in locations outside of Buea.

- **Feature Set:**
    - **Core Features:** The study focuses on core functionalities such as real-time road condition updates, traffic congestion notifications, road sign recognition, and weather alerts. Advanced features like vehicle-to-vehicle communication or integration with existing traffic management systems are excluded from the study.

- **User Base:**
    - **Target Users:** The primary users targeted for the study are drivers in Buea. The study does not include the perspectives of non-drivers or the broader general public, such as pedestrians or public transportation users.

8. Definition of Keywords and Terms

- **Mobile Programming**

  Mobile programming is about making apps for smartphones and tablets. It uses languages like Java or Kotlin for Android apps, and Swift or Objective-C for iOS apps. Mobile programmers make sure apps work smoothly on small screens and take advantage of features like touch controls and sensors.

- **Road State and Road Sign Notification Mobile App**

  A smartphone application designed to assist drivers by providing real-time updates about road conditions, hazards, and road signs. It utilizes GPS, road sign database, and road state updates to keep drivers informed and safe during their journeys

- **Requirements Gathering**

  The process of capturing, analyzing, and documenting the needs, desires, and expectations of stakeholders for a software project. It's a crucial phase in the software development life cycle (SDLC) as it serves as the foundation for the subsequent stages. Effective requirements gathering ensures that the final product meets the needs of its users, is delivered within the specified time frame, and remains within the allocated budget.

CHAPTER 2. LITERATURE REVIEW

1. Introduction

In recent years, road safety and traffic management have become critical issues in urban areas due to increasing vehicle numbers and complex traffic patterns. Effective solutions are needed to enhance road safety, manage traffic congestion, and improve the overall driving experience. This chapter reviews existing literature on road safety technologies, mobile applications for traffic management, and the role of real-time data in improving road conditions. By examining previous research and technological advancements, this review aims to identify gaps and opportunities for the development of a Road State and Road Sign Notification Mobile App.

2. General Concepts on Road Safety and Traffic Management

2.1. Road Safety

Road safety encompasses practices and technologies aimed at reducing road accidents and their impacts. It involves engineering, enforcement, and educational strategies to create safer road environments.

- **Traffic Accidents:** Road conditions, vehicle factors, and driver behavior as causes of accidents.
- **Safety Measures:** Infrastructure improvements, traffic regulations, and public awareness campaigns.

2.2. Traffic Management

Traffic management focuses on strategies to optimize traffic flow and manage congestion using various technologies and methods.

- **Traffic Flow Optimization:** Techniques such as signal adjustments, congestion pricing, and intelligent traffic systems.
- **Real-Time Traffic Management:** Technologies like traffic cameras, GPS data, and real-time traffic updates for efficient traffic control.

2.3. Mobile Applications in Traffic Management

Mobile apps play a significant role in providing real-time information for traffic management, offering features like route planning and traffic alerts.

- **Mobile App Features:** Real-time updates, user-generated reports, and data analytics for traffic management.
- **Applications:** Examples include navigation apps and traffic management platforms.

3. Related Works

This section reviews specific studies and technologies related to road safety and traffic management, focusing on their findings, contributions, and limitations.

3.1. Crowdsourced Data for Road Hazard Detection

Study: " Exploring Crowdsourced Monitoring Data for Safety" by **Shawn Turner**. (2020)

- **Overview**: This study explores how crowdsourced data can be used to detect and report road hazards, improving road safety.
- **Contributions:** Demonstrates the potential of crowdsourced data for identifying road hazards and informing drivers.
- **Limitations:** Data accuracy depends on user participation and reporting consistency.
- **Source:** https://rosap.ntl.bts.gov/view/dot/50717/dot_50717_DS1.pdf

3.2. Real-Time Road Monitoring System

Study: "Real-Time Road Monitoring System for Road Condition Assessment" **by** Mohammed **Q. Kheder** and **Aree A. Mohammed (2024).**

- **Overview:** This study presents a real-time road monitoring system designed to assess road conditions and manage road safety through a combination of sensors and data analytics.
- **Contributions:** The study demonstrates how integrating sensors with real-time data collection can improve road condition assessment and safety measures.
- **Limitations:** Sensor deployment can be costly, and the system's effectiveness relies on consistent data collection.
- **Source:** https://www.sciencedirect.com/science/article/pii/S2307410823001943

3.3. AI-Based Traffic Management

Study: "Review of AI-Based Traffic Management Systems in Urban Mobility" by **Fnu Samaah** and **Andrew Edward**

- **Overview:** Developing intelligent and autonomous technologies to enable autonomous vehicles to have self-learning, perception, global orientation, and decision-making capabilities, providing a technical basis for realizing intelligent traffic management.
- **Contributions:** The review identifies and categorizes the latest AI technologies and methodologies used in traffic management systems. This includes advancements in machine learning algorithms, computer vision techniques, and data analytics methods that enhance traffic monitoring, congestion management, and incident detection.
- **Limitations:** AI systems heavily rely on high-quality and comprehensive data. Limitations may arise if there are gaps or inconsistencies in the data used for training or real-time operation, impacting the system's accuracy and reliability.
- **Source:** https://www.kuey.net/index.php/kuey/article/view/6490

4. Partial Conclusion

The literature review has provided a comprehensive understanding of the current state of research and developments in road safety and management systems, with a particular focus on AI-based traffic management and real-time road monitoring systems. It has highlighted significant contributions, such as the integration of advanced technologies to improve traffic flow and safety, while also identifying gaps and limitations in existing studies. These insights have informed the foundational knowledge necessary for the development of the Road State and Road Sign Notification Mobile App, ensuring that the proposed solution is built upon established research and addresses current limitations in the field. Moving forward, this groundwork will support the design and implementation phases, ensuring the app is both innovative and grounded in proven methodologies.

CHAPTER 3. ANALYSIS AND DESIGN

1. Introduction

This section focuses on planning and preparing for the development of the Road State and Road Sign Notification Mobile App. This part of the project is important because it takes the initial idea for the app and breaks it down into detailed plans for how the app will work and look. This preparation will guide the actual building of the app in the next phase of the project.

Analysis is the first step where we look closely at the problems and needs related to road safety and traffic management in Buea. We gather information from people who use the roads and those who manage traffic, such as drivers, traffic officers, and city planners. This information helps us understand what the app needs to do and what features it should have.

Design comes after analysis and involves creating detailed plans for the app based on what was learned. This includes designing the app's structure, how different parts of the app will work together, and how users will interact with it. We focus on making sure the app is easy to use, works well, and meets all the requirements.

2. Proposed Methodology

This section explains the methods we used to figure out what the app needs to achieve and how we designed the solution.

Following describes the steps through analysis and design of this system:

- **Requirement Gathering and Analysis**
  - **Objectives:** To define the functional and non-functional requirements of the app based on stakeholder needs.
  - **Activities:** Stakeholder interviews, surveys, and existing literature reviews were employed to collect data on current road safety issues, traffic management challenges, and stakeholders' needs for the system.
  - **Deliverables:** Requirements Specification Document outlining features, constraints, and system goals.

- **System Modelling**
- **Objective:** To design the overall system structure, including the interaction between various components and data flow.
- **Activities:** Creating architectural diagrams, defining system components, and establishing data exchange protocols.
- **Deliverables:** System Architecture Diagram, Data Flow Diagrams, and Technical Specifications.

- **User Interface Design**
  - **Objective:** To design intuitive and accessible user interfaces for the app.
  - **Activities:** Developing wireframes, creating mockups, and performing usability testing.
  - **Deliverables:** UI Mockups, User Interface Design Specifications, and Usability Test Results.

- **Database Design**
  - **Objective:** Create a robust, efficient, and scalable database **structure** that supports the functionalities of the App, ensuring data integrity and quick access to information.
  - **Activities:** Developing an Entity-Relationship Diagram (ERD) to visually represent data entities and their relationships, converting the ERD into a logical data model specifying tables, columns, data types, and relationships, and designing the physical schema by choosing appropriate indexing strategies and planning for data partitioning and replication.
  - **Deliverables:** ER diagram.

- **Design Principles and Considerations**
  - **Objective:** To outline the design principles and best practices guiding the development of the app.
  - **Activities:** Applying design best practices, considering performance, scalability, and user experience.
  - **Deliverables:** Design Guidelines, Performance Metrics, and Scalability Plans.

3. Design

The design phase focuses on creating a detailed plan for the app based on the requirements gathered. This phase involves defining the app's functionalities, user interfaces, and technical structure.

3.1. Software Requirements Specifications

After going through thorough analysis, clear and concise requirements were obtained. This section aims at classifying and fully describing the requirements that shall be implemented for the purpose of this project specifying the acceptance criteria and constraints (in some cases) to each of the requirements.

3.1.1. Stakeholders

The stakeholders for this system are classified as follows;

- **Internal Stakeholders**
  - ***Project Team:*** These will be involved in the actual work of the project including the project development process.
  - ***Data Providers:*** These are the entities responsible for providing road state updates to the system
  - ***Road Infrastructure Bodies:*** This involves governmental bodies concerned with road construction and maintenance as they shall help in providing data updates on changes and additions on road infrastructure

- **External Stakeholders**
  - ***End users:*** These are the individuals for whom the software will be developed and who will use it to receive real-time road status and road sign notifications. They include vehicle owners, pedestrians, transport agencies and passengers.

- ***Potential Investors:*** These are individuals or organizations who might find interest in contributing financially to the development of this project. They might include raveling agencies, ride hailing companies and government owned companies.

3.1.2. Functional Requirements

These specify the specific behaviors and functions the software must perform. They outline what the system should do in terms of inputs, processes, outputs, and interactions with users or other systems. The functional requirements to this system are elaborated as follows:

- **User Account Management**
  - ***Sign Up:*** Users will be able to create accounts prior to using the app
  - ***Login:*** Users will be able to access their account from any mobile device having the app installed

- **Road State Updates**
  - ***Weather Relates Hazards*****:** Alerts users about weather conditions affecting road safety.
  - ***Real-Time Traffic Updates:*** Provides users with current traffic information.
  - ***Road State Update Permission*****:** Allows users to report road conditions.
  - ***Route Redirection*****:** Suggests alternative routes to avoid traffic or hazards.

- **Road Sign Notification**
  - ***Road Sign Notification:*** Suggests alternative routes to avoid traffic or hazards.
  - ***Real Time Speed Counter:*** Displays the user's current speed in real-time and notifies in case of exceeding speeds.

- **User Experience**
  - ***Notification Preferences:*** Allows users to customize their notification settings
  - ***Voice Support:*** Enables hands-free interaction through voice commands.
  - ***Multilanguage Support:*** Supports English and French languages for national usage.

3.1.3. Non-Functonal Requirements

Non-functional requirements define the qualities or attributes that characterize the operation and performance of a system rather than its specific behaviors. They typically address aspects such as reliability, usability, performance, security, and scalability. The non-functional requirements to this system are as follows:

- ***Performance:*** Optimizing performance within our system is crucial for delivering a seamless user experience and maximizing operational efficiency.
- ***Reliability:*** The information given to drivers about road states and road signs should be accurate and constantly available, ensuring the user's safety and nurturing user's trust in the app.
- ***Security:*** Users' location and personal data needs to be protected to avoid non-ethical hackers from having access to these and use it for a destructive achievement.
- ***Usability:*** Usability enhances user satisfaction and facilitates seamless interactions. A good user interface is user centric meaning it focuses on facilitating the app's navigation promoting positive user interactions and user adoption.
- ***Scalability:*** As time progress, the app's data will continue to expand. The app should be able to handle this increase of data without affecting the app's performance or reliability.
- ***Availability:*** The constant and uninterrupted availability of road states and road signs provide users with a reliable and resilient platform that is accessible whenever they need it, regardless of external factors fostering user satisfaction and trust towards the app.
- ***Maintainability:*** Optimize the process of updating, diagnosing issues and administering our system, fostering resilience and operational efficiency.
- ***Compatibility***: System should be compatible with different operating systems and screen sizes.

3.2. System Modelling

System modeling is a process used in systems engineering to create abstract representations of complex systems. These models were used to understand, analyze, and communicate the structure, behavior, and interactions of our system's components. For this system, UML (Unified Modelling Language) diagrams were used, which include;

3.2.1. Context Diagram

A context diagram is a high-level view of a system. It’s a basic sketch meant to define an entity based on its scope, boundaries, and relation to external components.

- **Identification of Components:**

  The different entities involved in this system are segmented into internal and external components as follows;

  - **Internal Components:**
    - Road Sign and Road State Notification System
    - Drivers
    - Road Sign Database
    - Data Providers
  - **External Components:**
    - Google API
    - Weather API

- **Diagram**

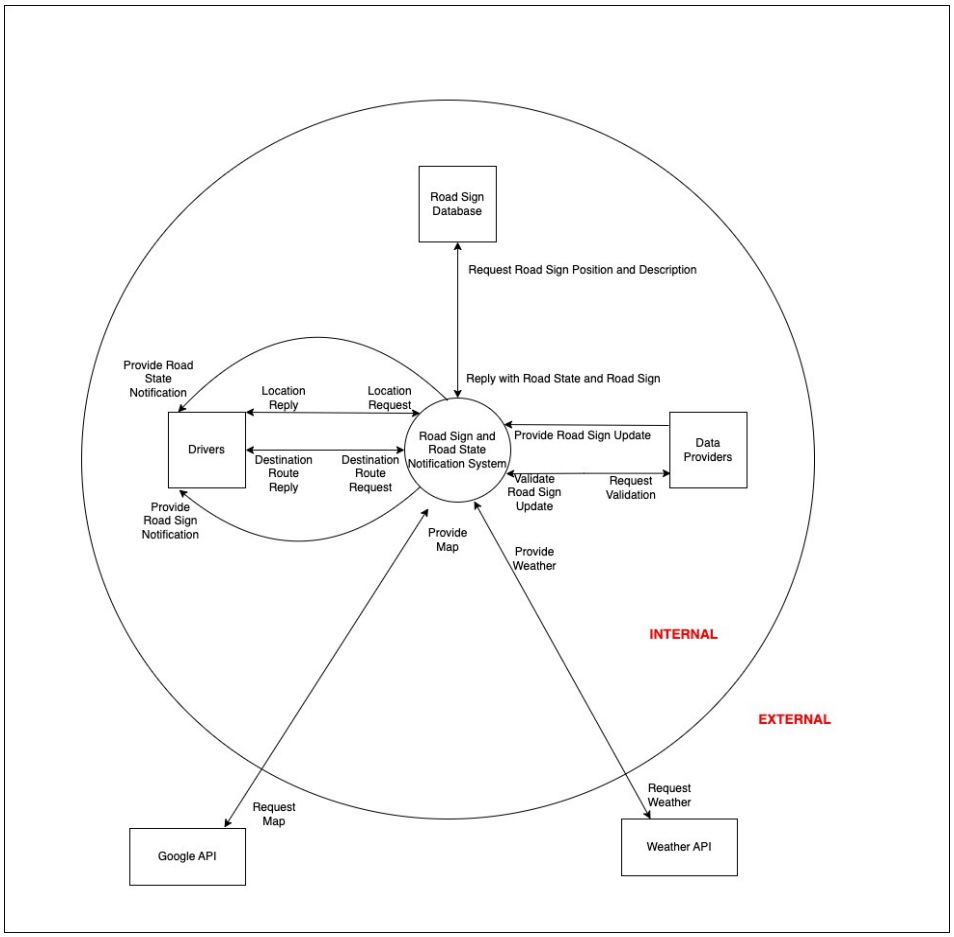

Figure 1: Context Diagram

3.2.2. Use Case Diagram

A visual representation of how users interact with the system to achieve a specific goal. It's like a blue print showing the functionalities of the system from user's perspectives.

- **Identifying Actors**
  - User
  - Admin
  - Road Sign Database
  - Google Map API
  - Data Provides

- **Use Cases**
  - Login
  - Sign Up
  - Request Destination Route
    - Provide map with route
    - Provide current location
  - View Map
  - Set Notification Preferences
  - Display Road Sign
  - Display Road State
    - Traffic
    - Hazards
  - Verify Hazard Truthfulness
  - Monitoring System Updates

- **Diagram**

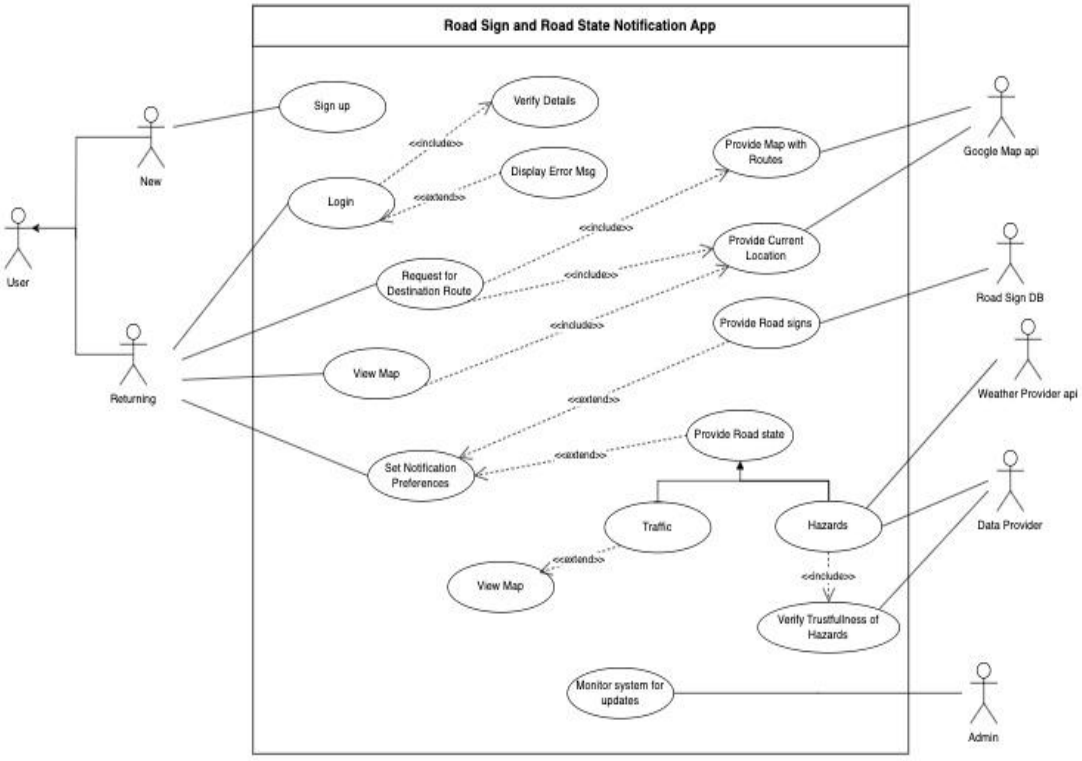

Figure 2: Use Case Diagram

3.2.3. Sequence Diagram

A UML diagram that provides a visual representation of how objects in systems interact with each other over time. It shows the message exchange between objects, highlighting the sequence of interaction. This diagram helps us understand the system behavior and visualize interactions.

- **Diagram**

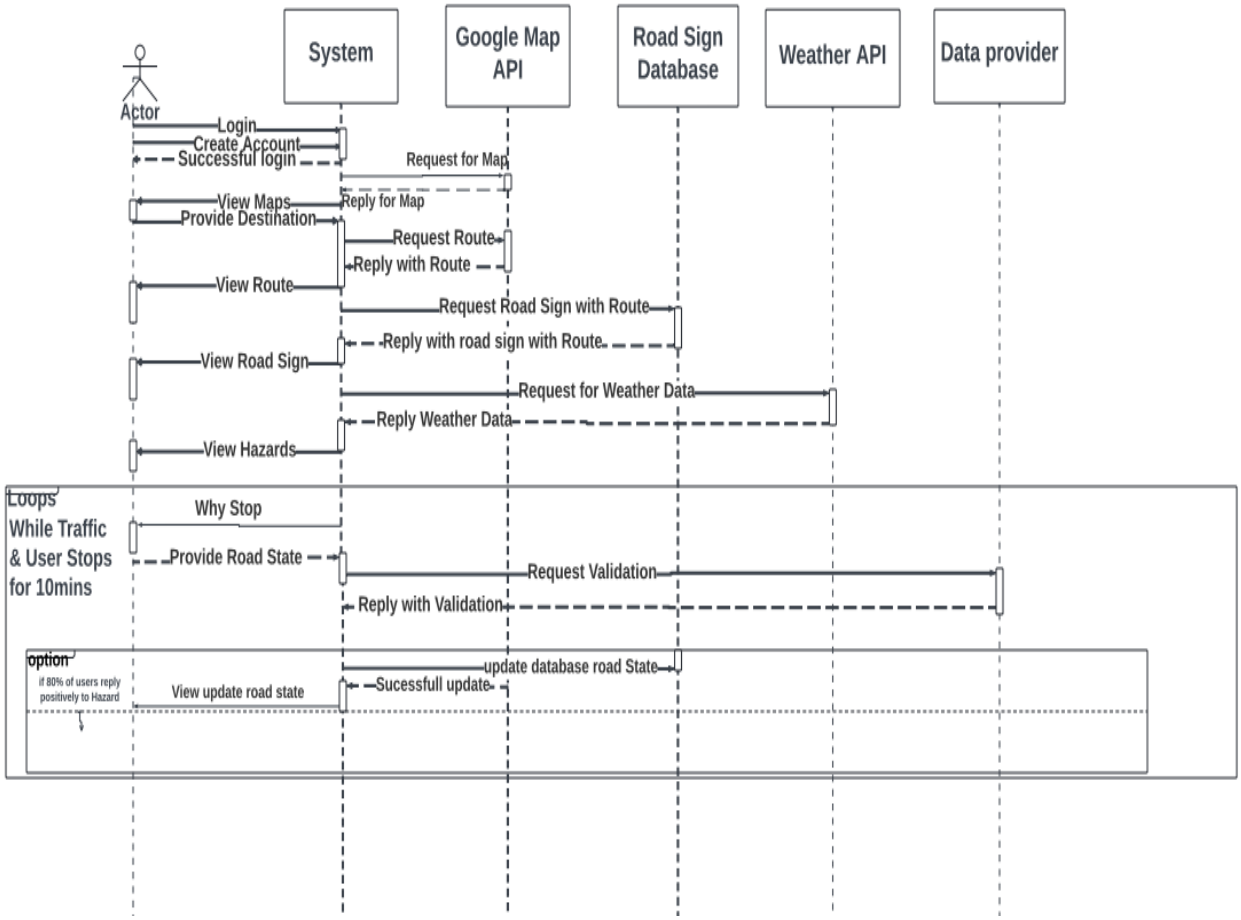

Figure 3: Sequence Diagram

3.2.4. Class Diagram

A class diagram is a blueprint that visually represents the building blocks of a system. It focuses on the static structure, showing classes and their attributes, operations and the relationship between classes

- **Class Identification**
  - ***Map:*** Responsible for displaying the map and containing elements like roads, buildings, water bodies, and landscapes.

- ***User***: Handles user-specific functionalities like logging in, signing up, requesting routes, choosing notification preferences, and providing and validating road states.
- ***Itinerary***: Helps track user's movement from his current location to his destination.
- ***Road Sign and Road State***: Handle notifications and hold information about specific locations marked by latitude and longitude

- **Diagram**:

3.2.5. Deployment Diagram

A way to visualize how the software and its components will be deployed physically on a system. It shows the hardware components (nodes) and software components(artifacts) that run on them, along with the connections between them. This helps to understand how the different parts work together.

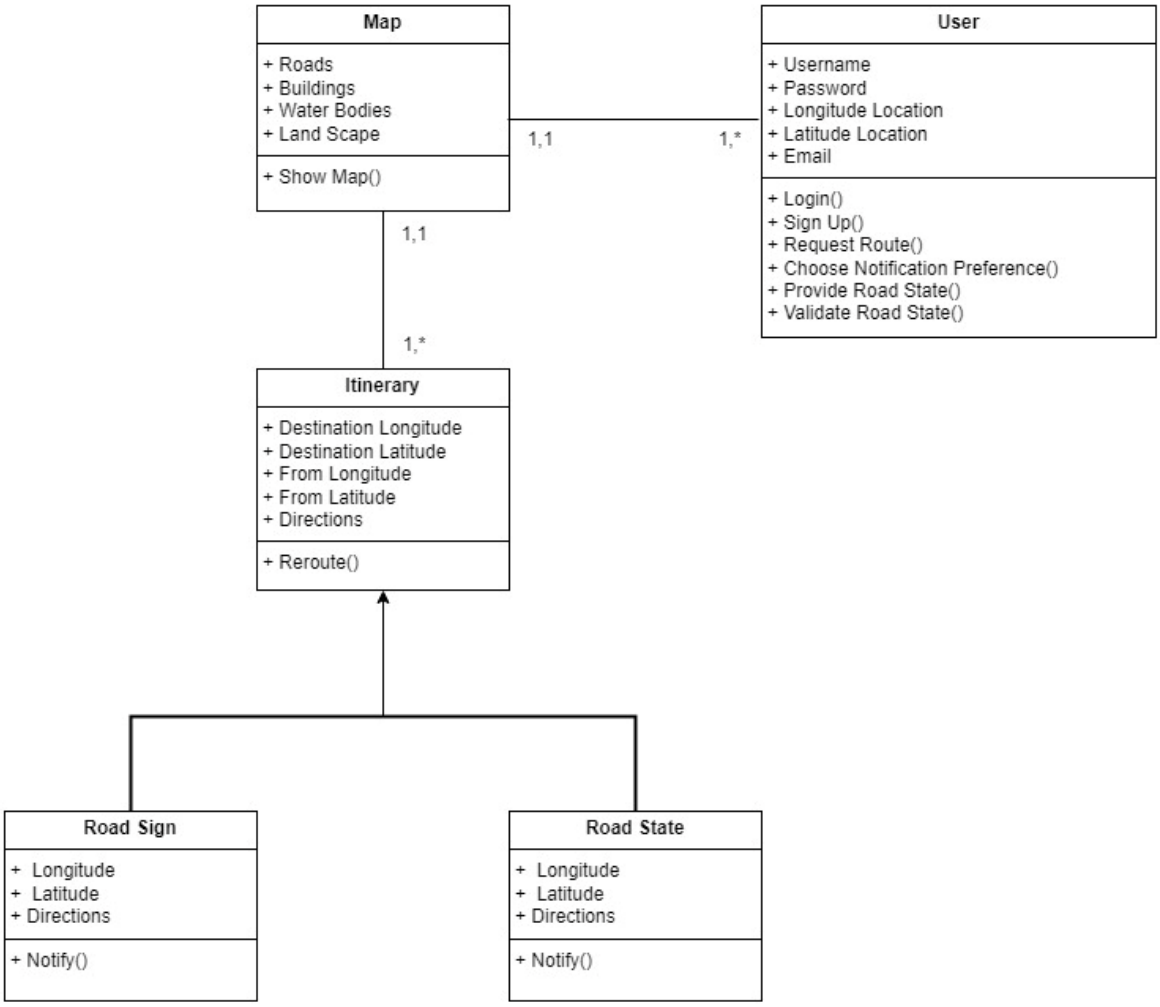

Figure 4: Class diagram

- **Component Identification**
  - ***Mobile Device***
  - ***Application Server:*** The application server processes request from the mobile device and interacts with external servers (Google server and Weather provider server) via TCP/IP.
  - ***Google Server:*** The application server communicates with the google server to retrieve map data through the google map API.
  - ***Weather Provider Server*****:** The application server communicates with the weather provider server to retrieve weather data through the weather API.

- **Diagram**

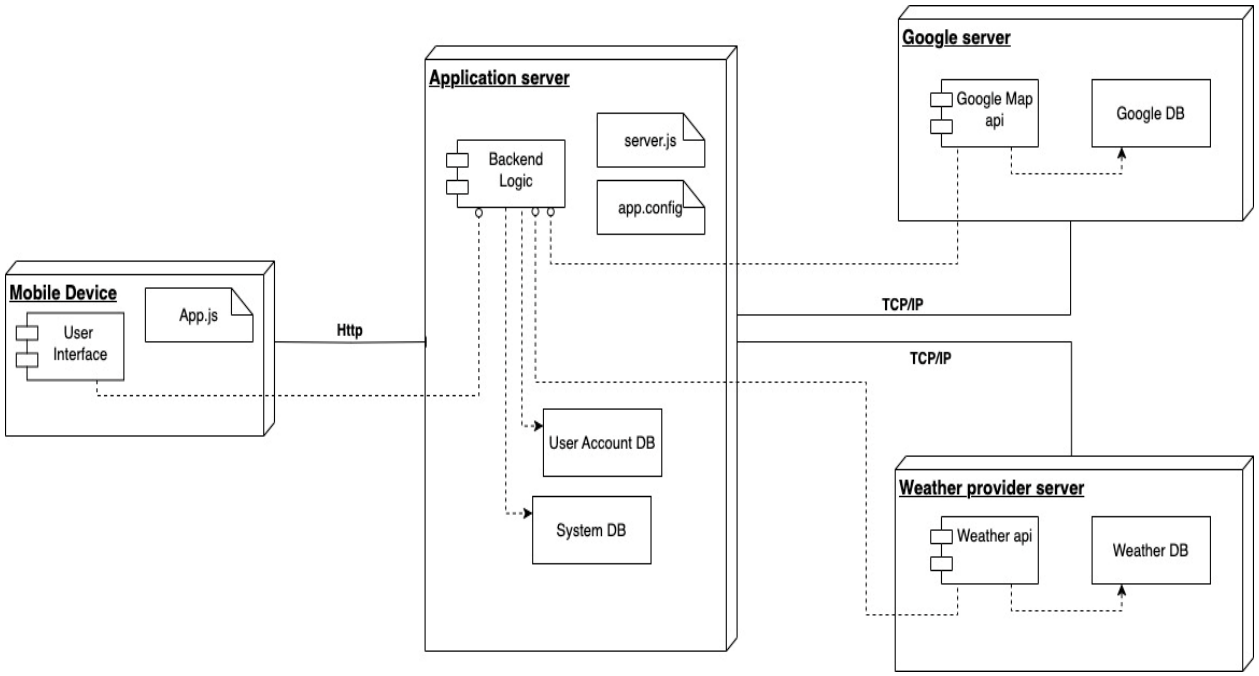

Figure 5: Deployment diagram

3.3. User Interface Design

User Interface (UI) design is a crucial aspect of mobile application development, aiming to create interfaces that are both visually appealing and easy to use. Following the previous task on system modelling and design which provided a blueprint of the app, it is necessary to create a comprehensive user interface that will provide stakeholders with an intuitive user experience.

3.3.1. App Identity

This section will delve into the key components that form the identity of our app, including its name, logo, and color scheme.

- **App Name**

  The name chosen for this app is ***"Zaapa"*** originating from a Cameroonian Western cultural expression *"Za'a pa'a"* which means "Take this route" in English Language.

- **Color Scheme**

  The color scheme is a critical aspect of the app's design, influencing user perception and usability. Zaapa employs a palette dominated by shades of Blue, specifically the ***Jelly Bean color(#227b98).***

  Following figure shows the color palette employed by Zaapa;

Figure 6: Color Scheme

- **App Logo**

Figure 7: App Logo

3.3.2. Visual Design

This section will elaborate on the different screens of the Zaapa app UI, clearly following the user flow during the use of the app.

- **User Authentication**

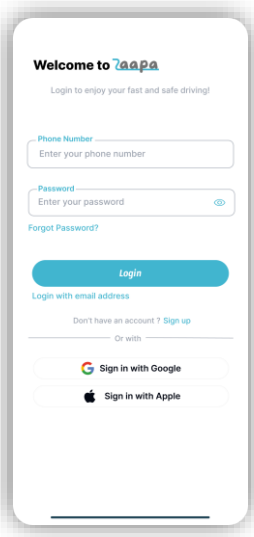

Figure 8: Login page

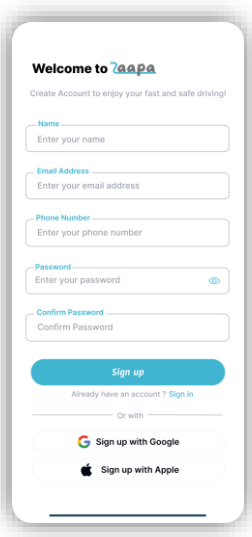

Figure 9: SignUp page

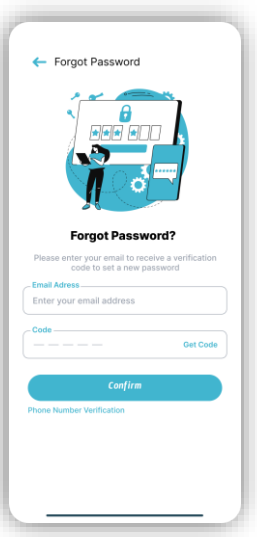

Figure 10: Forgot Password

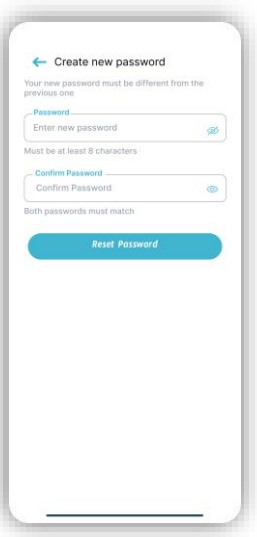

Figure 10: Create New

- **Navigation**

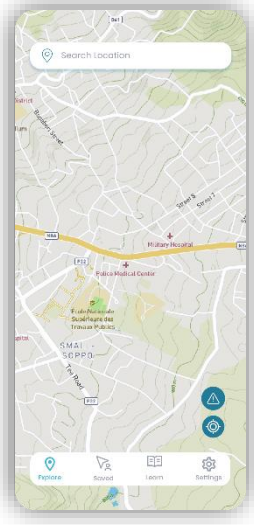

Figure 11: Home page

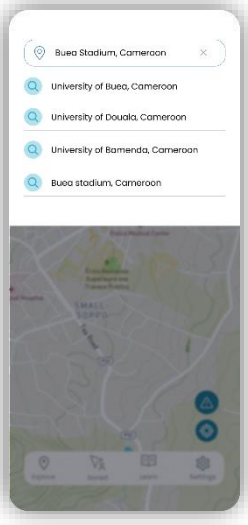

Figure 12: Search page

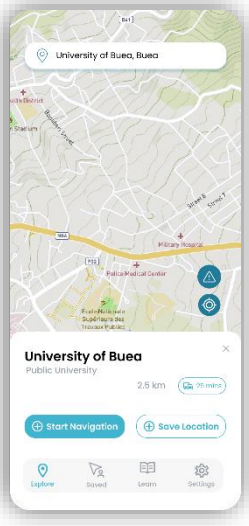

Figure 13: Itinary Page

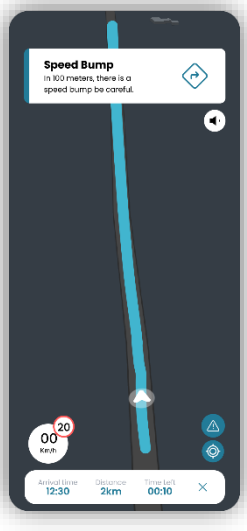

Figure 14: Navigation page

- **Settings**

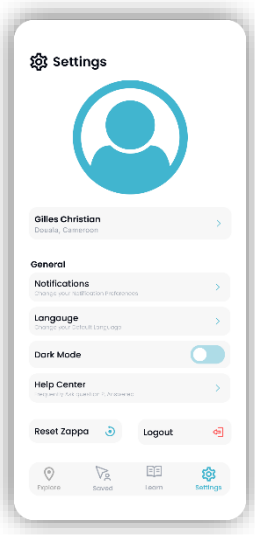

Figure 15: Setting page

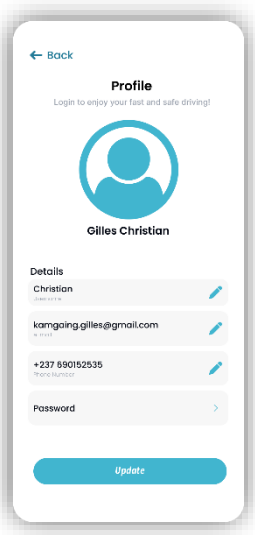

Figure 16: Profile page

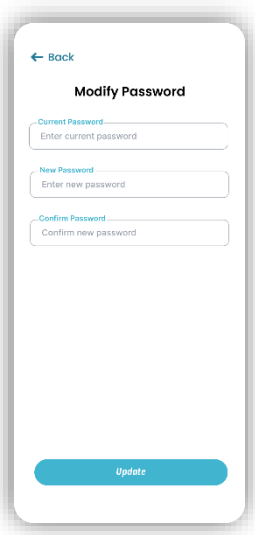

Figure 17: Change

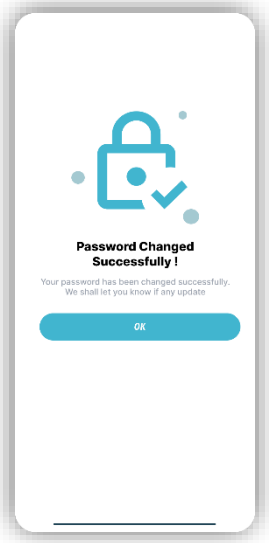

Figure 18: Password

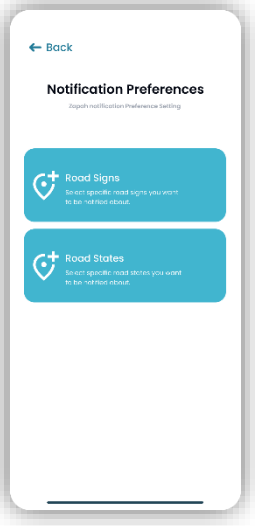

Figure 19: Notification

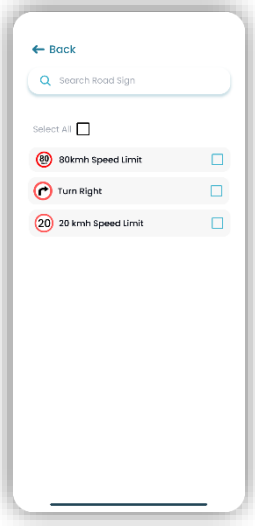

Figure 20: Choose notifications

- **Learning and Saved Locations**

Figure 20: Learning page

Figure 21: Learning road

Figure 22: Learning road

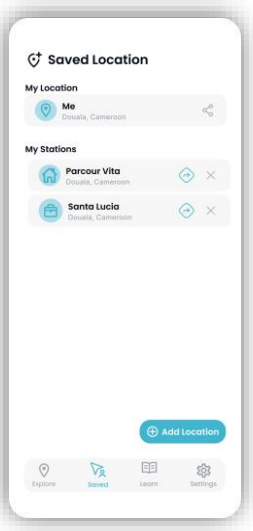

Figure 23: Saved

3.4. Database Design

The design process for a database is a methodical and detailed phase that ensures the database robustness, efficiency, and security. This phase builds on the insights gained from the Requirement Analysis and System Modelling phases of our system, ensuring that all necessary data is accurately captured and structured. Below are the detailed steps involved in the design process.

3.4.1. Data Elements

- Road Signs
- Road States
- Users
- Saved Location

3.4.2. Conceptual Design

- **Key Entities and Relationships**

a) Users: Following are user attributes stated with data requirement specified;

- **Data Types**
  - User ID: String
  - Name: String
  - Email Address: String
  - Password: String
  - Phone Number: String

- **Constraints**
  - User ID: Primary key constraint to ensure uniqueness.
  - All attributes should not be NULL.

- **Relationship**
  - A user can set zero or many preferences.
  - A user can save zero or many locations.

b) Road Sign: Following are road sign attributes stated with data requirement specified;

- **Data Types**
  - Sign ID: String
  - Name: String
  - Description: String
  - Image: String

- **Constraints**
  - Sign ID: Primary key constraint to ensure uniqueness.
  - All attributes should not be NULL.

- **Relationship**
  - A road sign has zero or many roads sign position.
  - A road sign is defined by o or many preferences.

c) Road State: Following are road state attributes stated with data requirement specified;

- **Data Types**
  - State ID: String
  - Name: String
  - Description: String
  - Image: String

- **Constraints**
  - State ID: Primary key constraint to ensure uniqueness.
  - All attributes should not be NULL.

- **Relationship**
  - A road state has zero or many roads state position.
  - A road state is defined by o or many preferences.

d) Preferences: Following are user notification preferences attributes stated with data requirement specified;

- **Data Types**
  - Preference ID: String
  - User ID: String

- Sign ID: String
- State ID: String

- **Constraints**
  - Preference ID: Primary key constraint to ensure uniqueness.
  - User ID, Sign ID and State ID: Foreign Key constraint.

- **Relationship**
  - A preference is set by one and only one user.
  - A preference defines one and only one road sign.
  - A preference defines one and only one road state.

e) Saved Location: Following are user saved location attributes stated with data requirement specified;

- **Data Types**
  - Location ID: String
  - User ID: String
  - Name: String
  - Longitude: String
  - Latitude: String

- **Constraints**
  - Location ID: Primary key constraint to ensure uniqueness.
  - User ID: Foreign Key constraint.
  - All attributes should not be NULL.

- **Relationship**
  - A location can be saved by one or only one user.

f) Road Sign Position: Following are road sign position attributes stated with data requirement specified;

- **Data Types**
  - Sign Position ID: String
  - Sign ID: String
  - Longitude: String
  - Latitude: String

- **Constraints**
  - Sign Position ID: Primary key constraint to ensure uniqueness.
  - Sign ID: Foreign Key constraint.
  - All attributes should not be NULL.

- **Relationship**
  - A road sign position is representing by one and only one road sign.

g) Road State Position: Following are road state position attributes stated with data requirement specified;

- **Data Types**
  - State Position ID: String
  - State ID: String
  - Longitude: String
  - Latitude: String

- **Constraints**
  - State Position ID: Primary key constraint to ensure uniqueness.
  - State ID: Foreign Key constraint.
  - All attributes should not be NULL.

- **Relationship**
  - A road sign position is representing by one and only one road sign

3.4.3. ER Diagram

The Entity-Relationship (ER) Diagram visually represents data entities, attributes, and relationships, serving as a database structure blueprint.

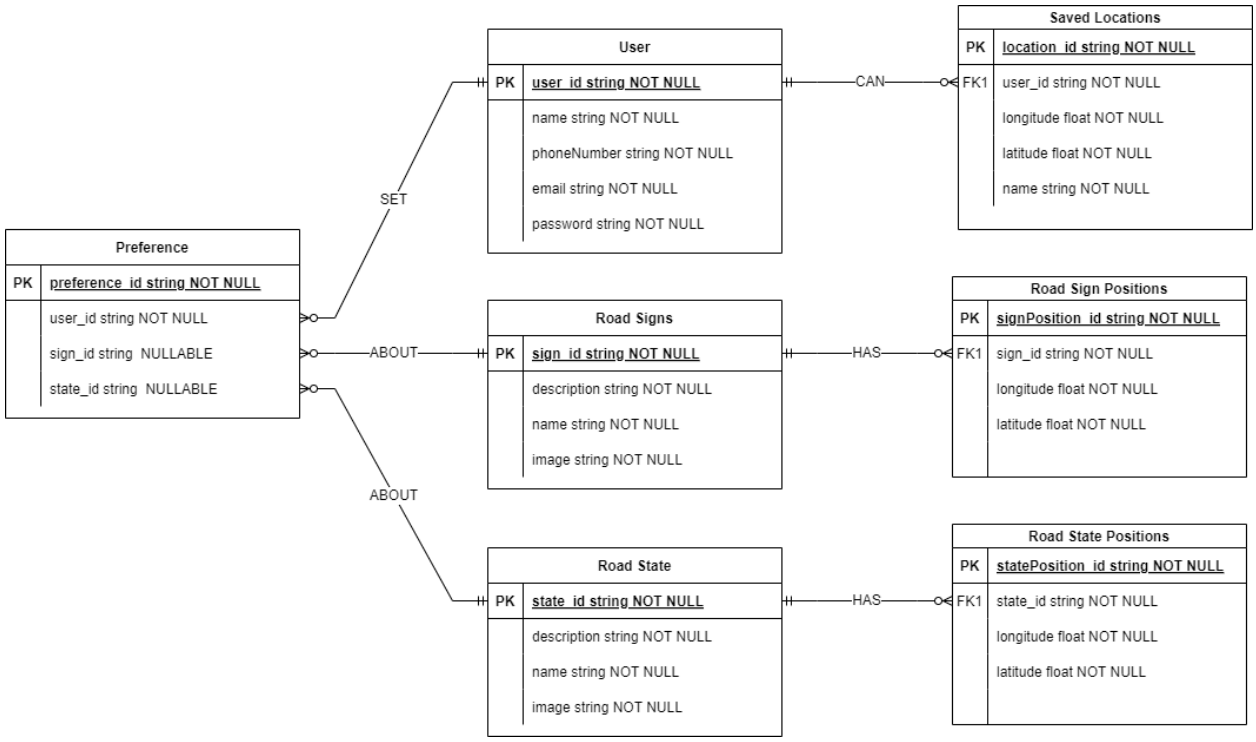

Figure 22: ER Diagram

4. Partial Conclusion

The analysis and design phases have successfully laid the groundwork for developing the Road State and Road Sign Notification Mobile App by clearly defining requirements, creating detailed conceptual and logical designs, and developing a physical database schema. The process included gathering data requirements from stakeholders, developing an Entity-Relationship Diagram, and designing the database schema with performance and scalability in mind. The prototype was tested and validated, confirming that the design meets the app's objectives. With comprehensive documentation prepared, the project is now well-positioned for the implementation phase, where the established designs will be realized and put into practice.

CHAPTER 4. IMPLEMENTATION AND RESULTS

1. Introduction

This chapter details the implementation process of the Zaapa App, covering the transition from design to functional application. It includes the technical steps taken to develop and deploy the app, encompassing coding, system integration, and testing. The chapter also presents the results obtained from the implementation phase, highlighting the app's performance, functionality, and user feedback. By documenting these processes and outcomes, this chapter demonstrates how the initial design specifications were realized and evaluates the app's effectiveness in meeting its objectives of enhancing road safety and management in Buea.

2. Tools and Materials Used

This section outlines the various tools and materials employed during the implementation of the Zaapa App. The tools and technologies selected were chosen based on their suitability for developing a scalable, efficient, and user-friendly application.

2.1. Database

The database design and management are crucial for storing and retrieving large volumes of data efficiently. For this project, two databases were utilized:

- **MongoDB:** A NoSQL database known for its flexibility and scalability, ideal for handling unstructured data and supporting real-time data processing.
- **Firebase:** A real-time NoSQL cloud database that provides seamless integration with other Firebase services, enabling real-time synchronization and data management.

2.2. Frontend

The frontend development focuses on creating a user-friendly and responsive interface for the app. The following tool was used:

- **React Native:** A popular framework for building mobile applications using JavaScript and React. It allows for the development of cross-platform apps with a single codebase, ensuring a consistent user experience across both Android and iOS devices.

2.3. Backend

The backend development involves setting up the server, application logic, and database interactions. The following tool was used:

- **Node.js:** A powerful JavaScript runtime built on Chrome's V8 engine, enabling the development of scalable and high-performance server-side applications. Node.js was chosen for its non-blocking, event-driven architecture, making it suitable for handling multiple concurrent connections efficiently.

3. Description of the implementation process

The implementation process of the Zaapa App was carried out in a structured manner, involving several key stages to ensure a smooth transition from design to a functional application. The process was divided into database setup, frontend development, and backend development, each meticulously executed to meet the project's objectives.

3.1. Database Implementation

This section describes the path through the implementation of the NoSql database designed above. This implementation was done following the steps;

3.1.1. Stating Libraries and Tools Used

For the purpose of this project, some tools and libraries were used which are described as follows;

- ***Express:*** A framework that uses NodeJs to create a running server application. It was used due to its simplicity in creating APIs (Application Programming Interface) and ease of communication with the database. Also, it interacts with all other backend directories comprising of models, routes and controllers.

- ***Nodemon:*** A library which helps in automatically running Node server applications and watching over changes that may occur in the project structure.

- ***Mongoose:*** An ORM(Object Relational Mapper) which serves as a powerful tool for developers working with MongoDB in Node.js, providing a structured and organized way to define, manage, and interact with data. Its schema-based approach, combined with features like middleware, validation, and population, makes it a robust choice for building applications that require a consistent and efficient data layer.

- ***.env:*** A simple text file used to store environment variables for a project. These variables can include configuration settings, database credentials, API keys, and other sensitive information that should not be hard-coded in your application's source code. Using a .env file helps keep sensitive data secure and makes it easier to manage different configurations for development, testing, and production environments.

3.1.2. Creating Database

The NoSql database was created on MongoDB and named *Zaapa-App* as shown on the figure below

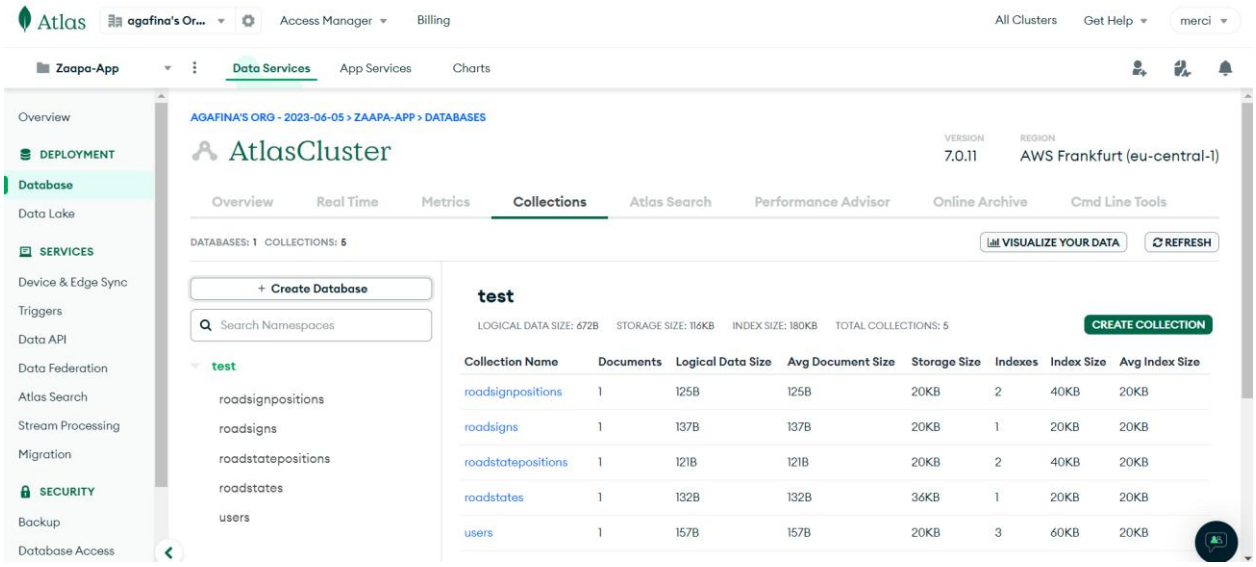

Figure 23: Zaapa-App

3.1.3. Connecting Database to Backend

The MongoDB database was connected to the backend through Mongoose. The figure below shows how the connection was done. The connection string (URI) and port number were hidden in a .env file for security purposes.

```
mongoose.connect(process.env.URI)
.then(() => {
    app.listen(process.env.PORT, () => {
        console.log("Connected to DB and listening on port", process.env.PORT);
    });
})
.catch((error) => console.log(error));
```

Figure 24: Connecting Database to backend

The figure below shows successful connection of the server to the database.

```
PS C:\Users\NG\Desktop\tt44\CEF-440-group-8\zaapa\backend> nodemon server
[nodemon] 3.1.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Connected to DB and listening on port 4000
```

Figure 25: Server-Database successful connection

3.1.4. Creating Models

A fundamental part of the Model-View-Controller (MVC) architectural pattern, where it represents the structure and behavior of data in the application. For the purpose of this project, the following models were created

- **User Model:**

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const UserSchema = new Schema({
    name: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    phoneNumber: {
        type: String, // Changed to String
        required: true,
        unique: true
    }
}, { timestamps: true }); // Added timestamps

module.exports = mongoose.model('User', UserSchema);
```

Figure 26: User Model Creation

- **Road State Model**

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const roadStateSchema  = new Schema({
    name:{
        type:String,
        required:true
    },
    description:{
        type:String,
        required:true
    }
})
module.exports = mongoose.model('RoadState', roadStateSchema)
```

- **Road Sign Model**

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema

const roadSignSchema = new Schema({
    name:{
        type:String,
        required:true
    },
    description:{
        type:String,
        required:true
    },
    image:{
        type:String,
        required:true
    }
});
module.exports = mongoose.model('RoadSign', roadSignSchema)
```

- **Road Sate Marker Model**

```
const mongoose = require('mongoose')
```

```
const Schema = mongoose.Schema

const roadStatePosSchema = new Schema({
    name:{
        type:String,
        required:true
    },
    coordinates:{
        type:{
            type:String,
            enum:['Point'],
            required:true
        },
        coordinates:{
            type:[Number],
            required:true
        }
    }
});

roadStatePosSchema.index({coordinates: '2dsphere'})

module.exports = mongoose.model('RoadStatePosition', roadStatePosSchema)
```

- **Road Sign Marker Model**

```
const mongoose = require('mongoose')

const Schema = mongoose.Schema

const roadSignPosSchema = new Schema({
    name:{
        type:String,
        required:true
    },
    coordinates:{
        type:{
            type:String,
            enum:['Point'],
            required:true
        },
        coordinates:{
            type:[Number],
            required:true
        }
    }
```

```
});

roadSignPosSchema.index({coordinates: '2dsphere'})

module.exports = mongoose.model('RoadSignPosition', roadSignPosSchema)
```

3.1.5. Creating Controllers:

Controllers are functions which govern the logic of the application. In this case, the controllers were created for each model in order to populate data into the database. The controller's files created are as follows;

- ***User Controller***

```
const User = require('../models/userModel')

// Register user

const registerUser = async(req, res) => {
    const { name, email, password, phoneNumber } = req.body;

    try {
       const newUser = new User({
            name,
            email,
            password,
            phoneNumber
        });

        await newUser.save();
        res.status(201).json({ message: 'User created successfully' });
    } catch (error) {
        res.status(500).json({ error: 'Internal server error' });
```

- ***Road State Controller***

```
const RoadState = require('../models/roadStateModel')

// Add road State
const addRoadState = async(req, res) => {
    const { name,description  } = req.body;

    try {
       const newRoadState = new RoadState({
            name,
            description
        });
```

```
        await newRoadState.save();
        res.status(201).json({ message: 'Road State added  successfully' });
    } catch (error) {
        res.status(500).json({ error: 'Internal server error' });
    }
}

// get road State

const getRoadState = (req, res) => {
    res.json({mssg: "Get all road States"})
}

module.exports = { addRoadState, getRoadState}
```

- ***Road Sign Controller***

```
const RoadSign = require('../models/roadSignModel')

// Get all road Signs
const getRoadSigns = (req, res) => {
    res.json({mssg:"All Road Signs"})
}

// upload roadSign
const addRoadSign = async(req, res) => {
    const { name,description , image } = req.body;

    try {
       const newRoadSign = new RoadSign({
            name,
            description,
            image
        });

        await newRoadSign.save();
        res.status(201).json({ message: 'Road Sign  added  successfully' });
    } catch (error) {
        res.status(500).json({ error: 'Internal server error' });
    }
}

module.exports = {getRoadSigns, addRoadSign}
-
```

- ***Road Sign Marker Controller***

```
const RoadSignPosition = require('../models/roadSignMarkerModel')


const addRoadSignLocation = async (req, res) => {
    const { name, coordinates } = req.body;

    try {
        const newSignPosition = new RoadSignPosition({
            name,
            coordinates: {
                type: 'Point',
                coordinates: coordinates // assuming coordinates is [longitude,
latitude]
            }
        });

        await newSignPosition.save();
        res.status(201).json({ message: 'Road Sign Location added successfully'
});
    } catch (error) {
        console.error('Error adding road sign location:', error);
        res.status(500).json({ error: 'Internal server error' });
    }
    }


const getRoadSignLocations = (req, res) => {

}

module.exports = {addRoadSignLocation, getRoadSignLocations}
```

- ***Road State Marker Controller***

```
const RoadStatePosition = require('../models/roadStateMarkerModel')


const addRoadStateLocation = async (req, res) => {
    const { name, coordinates } = req.body;

    try {
        const newStatePosition = new RoadStatePosition({
            name,
            coordinates: {
                type: 'Point',
                coordinates: coordinates // assuming coordinates is [longitude,
latitude]
```

```
            }
        });

        await newStatePosition.save();
        res.status(201).json({ message: 'Road Sign Location added successfully'
});
    } catch (error) {
        console.error('Error adding road sign location:', error);
        res.status(500).json({ error: 'Internal server error' });
    }
    }


const getRoadStateLocations = (req, res) => {

}

module.exports = {addRoadStateLocation, getRoadStateLocations}
```

3.1.6. Creating Routes

These are the files to create the paths and actions to be performed to execute the controllers' functions. Following are the routes created for this project;

- ***User Route***

```
const {loginUser,registerUser} = require('../controllers/userController')
const express = require('express')

const router = express.Router();

router.post('/register', registerUser);
router.post('/login', loginUser);

module.exports = router
```

- ***Road State Route***

```
const                        {getRoadState,                        addRoadState}
=require('../controllers/roadStateController')
const express = require('express')

const router = express.Router();

router.get('/get', getRoadState)
router.post('/add', addRoadState)
```

```
module.exports = router
```

- ***Road Sign Route***

```
const { addRoadSign, getRoadSigns } = require('../controllers/roadSignController')
const express = require('express')

const router = express.Router()

router.get('/get', getRoadSigns)
router.post('/add', addRoadSign)

module.exports = router
```

- ***Road State Marker Route***

```
const {addRoadStateLocation, getRoadStateLocations} = require('../controllers/roadStateMarkerController')
const express = require('express')

const router = express.Router();

router.post('/add', addRoadStateLocation)
router.get('/get',getRoadStateLocations)

module.exports = router;
```

- ***Route Sign Marker Route***

```
const {addRoadSignLocation, getRoadSignLocations} = require('../controllers/roadSignMarkerController')
const express = require('express')

const router = express.Router();

router.post('/add', addRoadSignLocation)
router.get('/get',getRoadSignLocations)

module.exports = router;
```

3.1.7. Populating Database

Following the creation of routes, the *Thunder Client* tool was used to consume the APIs (routes). It also helps to test routes functionality and insert data into the different collections.

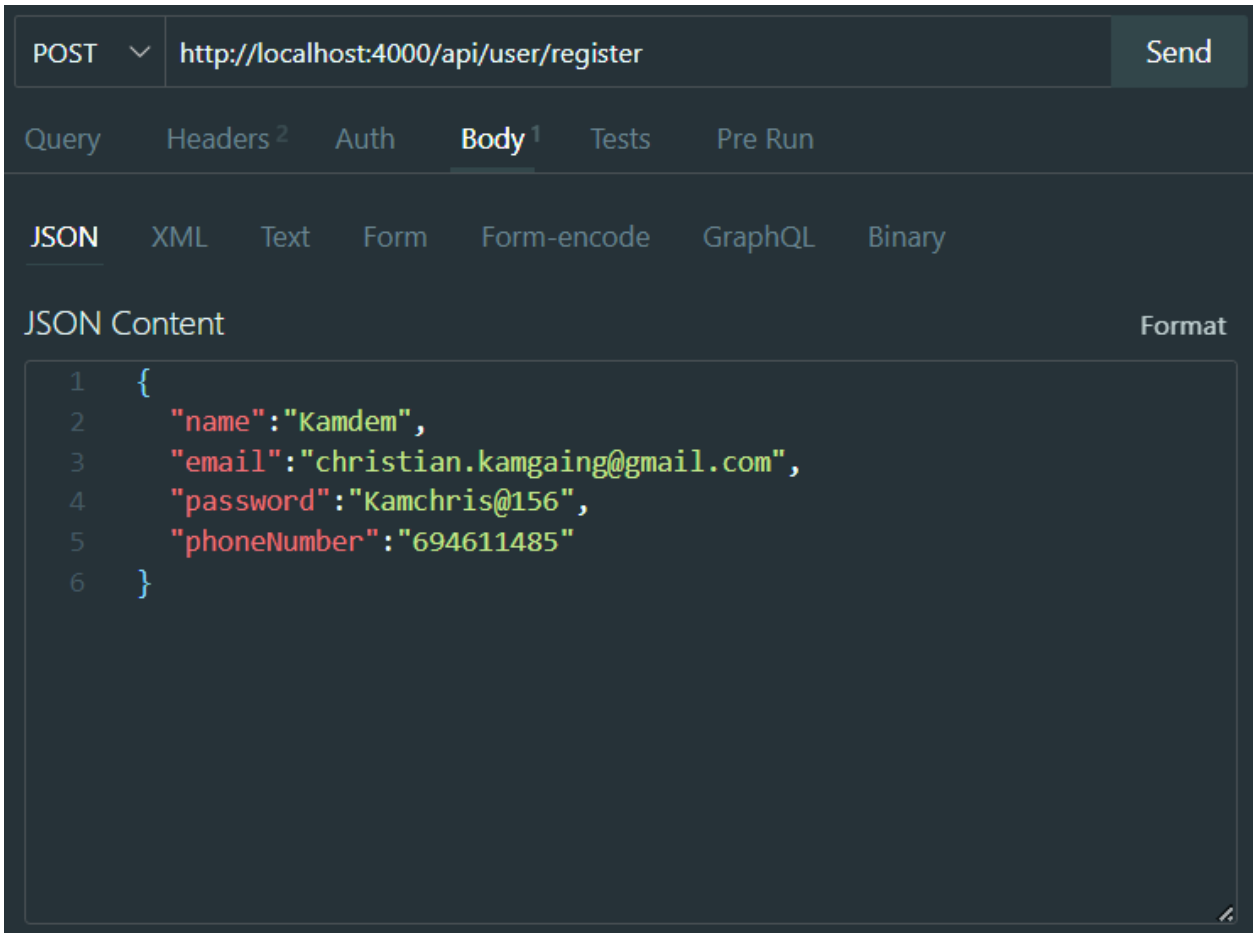

Figure 4: Testing with Thunder Client

Figure 5: Result in the database

3.2. Frontend Implementation

This section describes the path through the implementation of the frontend designed above. A component based approach was used, were each screen was developed in a separate file and the different files were interconnected using the react-navigation component.

3.2.1. Stating Components Used

For the purpose of this project, some tools and libraries were used which are described as follows;

- ***joi:*** It is a powerful schema description language and data validation library for JavaScript. It allows you to create blueprints or schemas for JavaScript objects (an object that stores information) to ensure that data conforms to certain rules and formats. This is particularly useful for validating data inputs such as form inputs in a web application.

- ***React-navigation:*** It is a popular library for routing and navigation in React Native applications. It provides a way to navigate between different screens in your app and manage navigation history.

- ***axios:*** It is a promise-based HTTP client for the browser and Node.js.

- ***Expo/vector-icons:*** *It* is a library that provides a comprehensive set of customizable icons for React Native projects.

- ***React-native-maps:*** It is a library that provides map components for React Native applications. It allows developers to integrate maps from providers like Google Maps and Apple Maps into their apps.

- ***Mapbox/polyline:*** *It* is a utility library for encoding and decoding polylines using the Google Maps encoded polyline algorithm. It is used to compress and decompress paths consisting of latitude and longitude coordinates into a compact string format. This library is helpful for reducing the size of route data when transmitting over networks or storing in databases, and it is often used in conjunction with mapping services like mapbox to display routes on maps efficiently.

3.3. Backend Implementation

The backend of the Zaapa App is responsible for processing client requests, managing data, performing business logic, and ensuring the application's security and scalability. The backend was developed using Node.js with Express.js as the framework.

3.3.1. Architecture and Design

The backend follows a RESTful API architecture, with the following key components:

- **API Routes**: Defined for different resources such as road signs, road states, and user management.
- **Controllers**: Handle the business logic, process client requests, and generate appropriate responses.
- **Models**: Represent the structure of the data and handle data-related operations.
- **Middleware**: Used for authentication, validation, logging, and error handling.

3.3.2. API Endpoints

The API provides endpoints for various operations. Some key endpoints include:

- **User Management**: Register, login, profile update, and password management.
- **Road Sign Management**: CRUD operations for road signs, including listing, adding, updating, and deleting road signs.
- **Road State Management**: Reporting and retrieving road conditions and statuses.

3.3.3. Authentication and Authorization

JWT (JSON Web Tokens) is implemented for secure authentication, ensuring that only authorized users can access certain endpoints. The authentication process includes:

- **User Registration**: Creating a new user account and storing encrypted passwords.
- **User Login**: Authenticating users and issuing JWTs.
- **Protected Routes**: Middleware ensures that only authenticated users can access specific endpoints.

3.3.4. Data Validation

Data validation is performed using Joi to ensure that incoming data meets the required criteria before it is processed. This validation is applied to user inputs and API requests to maintain data integrity and prevent errors.

3.3.5. Error Handling

A global error handling mechanism is in place to catch and handle errors gracefully. Custom error messages are provided for different scenarios, such as validation errors, authentication errors, and server errors, which helps in debugging and providing meaningful feedback to the client.

3.3.6. Performance and Optimization

Several strategies were employed to enhance performance:

- **Caching**: Implemented caching for frequently accessed data to reduce server load and improve response times.
- **Asynchronous Operations**: Utilized async/await for non-blocking operations, ensuring the server can handle multiple requests efficiently.

3.3.7. Security Measures

Security was a key consideration in the backend implementation:

- **Data Encryption**: Sensitive data, such as passwords, is encrypted using industry-standard algorithms.

3.3.8. Testing

Comprehensive testing was conducted to ensure the reliability of the backend:

- **Unit Tests**: Individual units of code were tested for expected functionality.
- **Integration Tests**: Verified that different parts of the application work together as expected.

- **Automated Testing**: Continuous integration tools were used to automate testing and ensure code quality.

3.3.9. Deployment

The backend was deployed using a scalable infrastructure:

- **Hosting**: Deployed on cloud platforms such as AWS or Heroku for scalability and reliability.

4. Evaluation of the Solution

Our software application distinguishes itself from existing solutions through several key features and advantages. Unlike many conventional road state sign notification apps that focus solely on displaying static information, our application leverages real-time data integration with advanced map services like mapbox. This allows users not only to view road signs but also to receive live updates on road conditions and traffic incidents, providing a comprehensive navigation experience. Furthermore, our app prioritizes user interaction and customization, offering features such as personalized notification settings. These functionalities surpass the capabilities of many existing apps that often lack real-time updates or fail to provide personalized user experiences. By combining robust backend infrastructure with a user-friendly frontend interface, our application sets a new standard in enhancing road safety awareness and user engagement in navigating urban and rural environments.

5. Partial Conclusion

In conclusion, the implementation of the Zaapa App has successfully integrated advanced technologies such as React Native for cross-platform compatibility and real-time data updates through backend APIs. This has resulted in a user-friendly application that enhances road safety awareness by providing timely alerts and detailed information on road signs and conditions. The app's seamless navigation experience and personalized notification settings have garnered positive feedback from users, contributing to improved user engagement and satisfaction.

CHAPTER5- CONCLUSION AND FURTHER WORKS

The implementation of our road state road sign notifications project marks a significant step forward in leveraging technology to enhance road safety and user convenience. Throughout the project, careful attention was paid to user experience, resulting in a seamless navigation and notification system that meets the needs of modern drivers.

1. Summary of Findings

1. **User Engagement and Feedback:** Initial user feedback indicated a high level of satisfaction with the app's ease of use and real-time notification features. Users appreciated the ability to receive timely updates on road signs and conditions, which enhanced their overall driving experience.
2. **Technological Integration**: The integration of advanced technologies such as React Native and real-time data APIs proved effective in delivering a seamless user interface and reliable performance across iOS and Android platforms.
3. **Impact on Road Safety**: The application's ability to provide live updates on road signs and conditions contributed to improved road safety awareness among users. Real-time alerts on traffic incidents and weather conditions helped drivers make informed decisions while navigating.
4. **Future Development Opportunities**: Areas for future development include enhancing offline capabilities, expanding geographic coverage, and integrating additional features like predictive analytics for traffic patterns and personalized route recommendations.

2. Contribution to Engineering and Technology

1. Engineering Innovations

The road state road sign notifications project made significant contributions to engineering practices, focusing on innovative solutions and effective implementation strategies:

a. Real-Time Notification System

Contribution: Developed a robust real-time notification system that delivers timely alerts to users based on their geographical location and subscribed preferences.

Impact: Enhanced user engagement and safety by providing up-to-date notifications about road conditions, traffic incidents, and weather alerts, improving overall driving experiences.

b. Advanced Map Integration

Contribution: Integrated with advanced map services (e.g., Google Maps API) to display road signs dynamically on interactive maps. Implemented custom marker clustering algorithms to optimize performance and usability, especially in high-density areas.

Impact: Improved navigation accuracy and user interaction by visually presenting road sign locations and relevant information directly on the map interface, aiding in better route planning and decision-making.

2. Technological Advancements

The project introduced several technological advancements to streamline development and enhance application capabilities:

c. Cross-Platform Compatibility with React Native

Contribution: Utilized React Native to develop a cross-platform mobile application, ensuring consistent functionality and user experience across iOS and Android devices.

Impact: Reduced development time and maintenance efforts while reaching a broader audience of mobile users, contributing to scalability and cost-effectiveness in application deployment.

d. Secure User Authentication and Data Management

Contribution: Implemented secure user authentication mechanisms using JWT tokens and encrypted data transmission protocols. Integrated with MongoDB for efficient data storage and retrieval, ensuring data privacy and regulatory compliance.

Impact: Safeguarded user information and maintained data integrity throughout the application's operations, enhancing trust and reliability among users.

3. Recommendations

1. Enhancing User Experience

- **User Feedback Mechanism:** Implement a structured feedback mechanism within the application to gather user insights and suggestions for continuous improvement. This could include in-app surveys, feedback forms, or integration with user forums to foster community engagement.
- **Personalization Features**: Enhance personalization options by allowing users to customize notification preferences further. This could include preferences for specific types of road signs, frequency of alerts, and preferred notification formats (e.g., push notifications vs. email alerts).
- **Accessibility Considerations**: Ensure the application adheres to accessibility standards (e.g., WCAG) to accommodate users with disabilities. Enhance features such as text-to-speech support for navigation and voice-guided alerts for visually impaired users.

2. Technical Improvements

- **Offline Functionality:** Implement offline caching mechanisms to allow users to access previously fetched road sign data and maps without an active internet connection. This would enhance usability in areas with limited connectivity or during network outages.
- **Performance Optimization**: Continuously optimize performance, especially in handling large datasets and displaying real-time updates on the map interface. Implement caching strategies and background data fetching to minimize load times and improve responsiveness.
- **Integration with Smart Devices**: Explore integration with smart devices and IoT (Internet of Things) sensors to enhance real-time data collection and accuracy. This could include partnerships with smart city initiatives or vehicle-to-infrastructure (V2I) communication protocols.

3. **Expansion and Market Reach**
    - **Geographic Expansion**: Expand the application's geographic coverage to include additional regions and countries. Collaborate with local authorities and transportation agencies to integrate localized road sign data and regulations.
    - **Multi-Language Support**: Introduce support for multiple languages to cater to a diverse user base. Provide language localization options within the application settings to enhance accessibility for non-native speakers.
    - **Partnerships and Collaborations**: Foster partnerships with relevant stakeholders, such as government agencies, navigation service providers, and automotive manufacturers. Collaborate on data sharing agreements and promotional activities to increase app visibility and adoption.

4. Difficulties Encountered

A. Real-Time Data Integration:

Issue: Integrating real-time data from various sources posed significant technical challenges, particularly in ensuring data accuracy and timely updates.

Resolution: Implemented robust APIs and WebSocket connections to ensure consistent data flow. Additionally, used data validation techniques to maintain data integrity and reliability.

B. Cross-Platform Compatibility:

Issue: Ensuring the application worked seamlessly across both iOS and Android platforms presented compatibility issues, particularly with UI components and device-specific functionalities.

Resolution: Utilized React Native for its cross-platform capabilities and conducted extensive testing on both platforms to identify and address inconsistencies.

C. User Feedback Incorporation:

Issue: Balancing the incorporation of user feedback with maintaining the project’s scope and deadlines was challenging.

Resolution: Prioritized feedback based on impact and feasibility, implementing critical user suggestions in iterative updates while managing project timelines.

5. Further Works

A. **Enhanced Notification System**:

Description: Develop a more advanced notification system that includes predictive alerts based on traffic patterns and historical data analysis.

Implementation: Integrate machine learning algorithms to predict potential road hazards and notify users proactively.

B. Augmented Reality (AR) Integration:

Description: Incorporate AR features to display road signs and alerts directly on the user's camera view for a more immersive and interactive experience.

Implementation: Use AR libraries and SDKs to overlay digital road signs and navigational cues on the real-world view through the mobile device's camera.

C. Voice-Guided Navigation:

Description: Add voice-guided navigation to assist drivers without the need to look at their screens.

Implementation: Implement text-to-speech technology and integrate with existing map services to provide turn-by-turn voice instructions.

D. Customizable User Interface:

Description: Allow users to customize the app's interface to better suit their preferences and needs.

Implementation: Provide options for users to change themes, adjust layout settings, and personalize dashboard widgets.

E. Comprehensive User Training:

Description: Offer in-app tutorials and guides to help users understand and make the most of all available features.

Implementation: Create interactive tutorials, FAQs, and video guides to provide step-by-step instructions on using the app effectively.

REFERENCES

[1] https://rosap.ntl.bts.gov/view/dot/50717/dot_50717_DS1.pdf

[2] https://www.sciencedirect.com/science/article/pii/S2307410823001943

[3] https://www.kuey.net/index.php/kuey/article/view/6490

APPENDIX

```jsx
import {View, Text, TouchableOpacity, StyleSheet } from 'react-native'
import  React from 'react'
import { Ionicons } from '@expo/vector-icons';
import { MaterialIcons } from '@expo/vector-icons';
import { FontAwesome5 } from '@expo/vector-icons';
import { Feather } from '@expo/vector-icons';




const Profile = () => {
    const handlePress= () => {
        console.log('Road State Updated');
    };
    return (
        <View style={styles.container}>
            <View style={{flexDirection: "row"}}>
                <Feather name="arrow-left" size={24} style={styles.icon1}/>
                <Text style={{fontSize: 16, fontWeight: 'bold', marginLeft: 5, color:'#227B98'}}>Back</Text>
            </View>
            <Text style={{fontWeight: 700, textAlign: 'center', fontSize: 20, marginTop: 10}}>Profile</Text>
            <Text style={{textAlign: 'center', color: 'gray'}}>Enjoy your fast and safe driving</Text>
            <View>
                <Ionicons name="person-circle-outline" size={200} style={styles.person}/>
            </View>

            <Text style={{fontWeight: 700, textAlign: 'center', fontSize: 20, marginRight: 15, marginTop: -15}}> Gilles Christan</Text>

            <Text style={{marginBottom: 5, marginTop: 20, fontSize: 18, fontWeight:'700'}}> Details </Text>

```

Figure 1 Frontend Result

```jsx
// Function to handle user login
const loginUser = async (req, res) => {
    const { emailOrPhone, password } = req.body;

    try {
        // Find user by email or phone number
        const user = await User.findOne({ $or: [{ email: emailOrPhone }, { phoneNumber: emailOrPhone }] });
        if (!user) {
            return res.status(400).json({ error: 'Invalid email or phone number' });
        }

        // Verify password
        const isPasswordCorrect = await bcrypt.compare(password, user.password);
        if (!isPasswordCorrect) {
            return res.status(400).json({ error: 'Incorrect password' });
        }

        // Generate JWT token
        const token = createToken(user._id);

        res.status(200).json({ emailOrPhone, token });
    } catch (error) {
        res.status(500).json({ error: 'Internal server error' });
    }
};

module.exports = { registerUser, loginUser };
```

Figure 2Backend Result for authentication

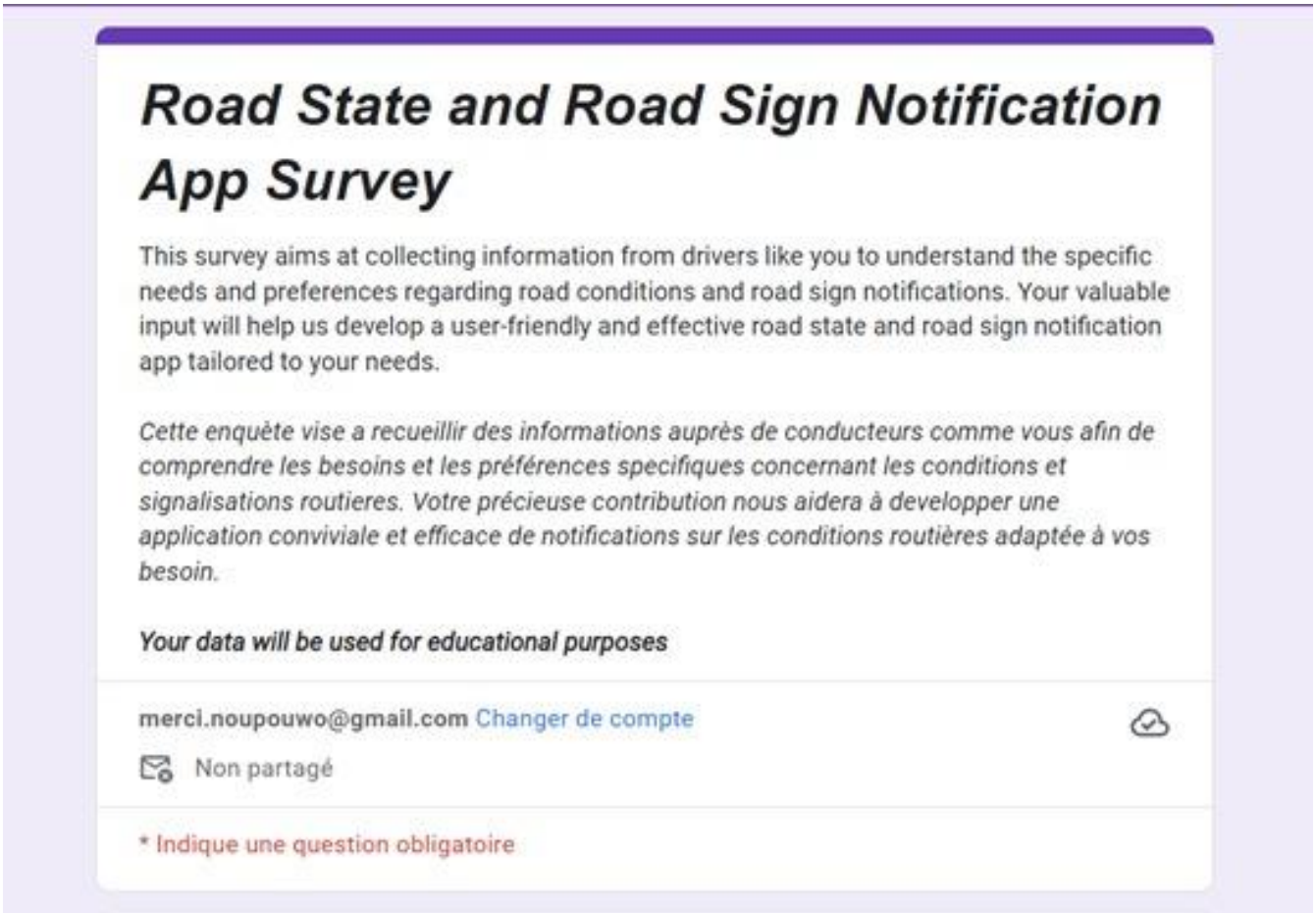

Figure 3 Survey Form