**APPLICATION**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**INTERNET AND MOBILE PROGRAMMING**

**CEF 440**

Presented by:

**GROUP 4**

Course Instructor

**Dr. Nkemeni Valery**

Academic Year

**2023/2024**



Republique du Cameroun

\*\*\*\*\*\*

Paix

-

Travail

-

Patrie

\*\*\*\*\*

Min

istere de l’enseignement

Superieurieure

\*\*\*\*\*

Universite de Buea

Republic of Cameroon

\*\*\*\*\*\*

Peace

-

Work

-

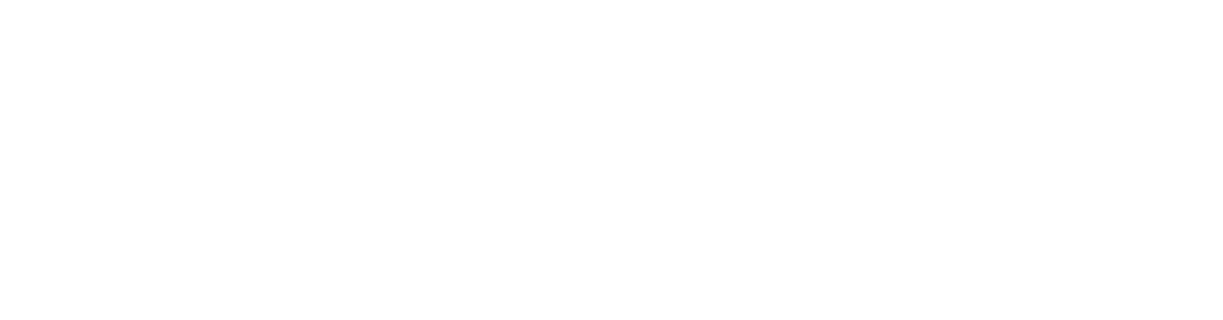
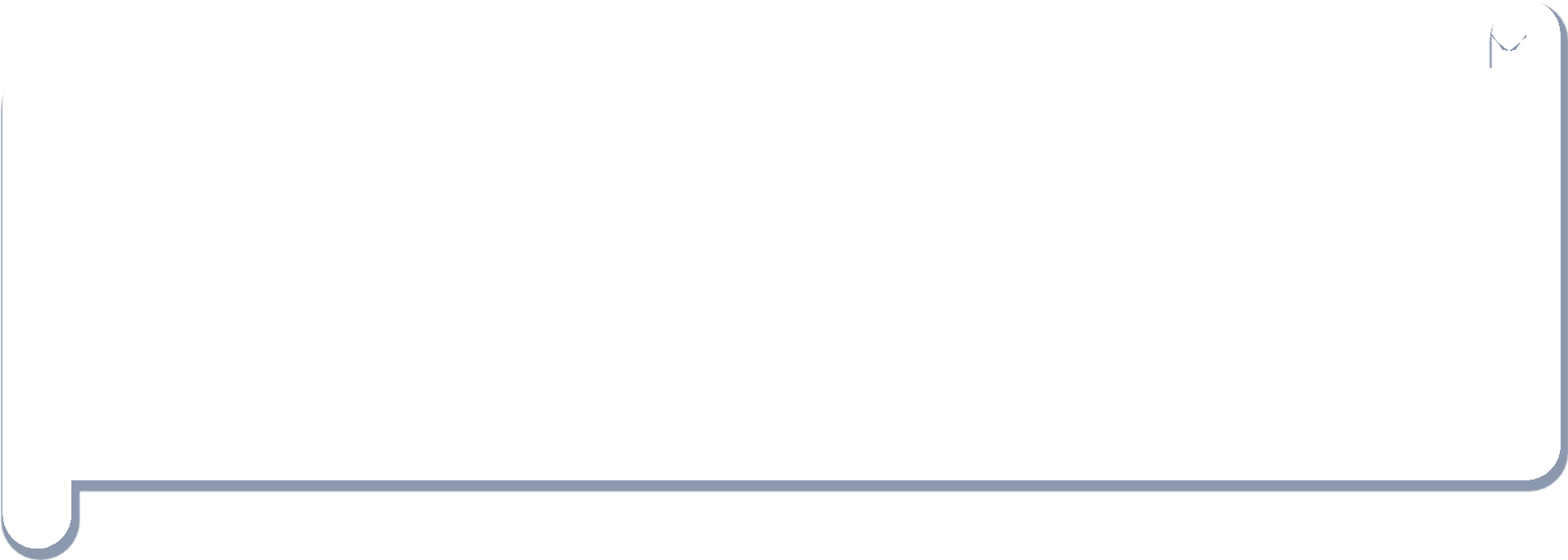
Fatherland

\*\*\*\*\*

Minister of Higher Education

\*\*\*\*\*

University of Buea



**DESIGN AND IMPLEMENTATION OF A ROAD**

**STATE AND ROAD SIGN NOTIFICATION**

**Group Members**

|  |  |  |
| --- | --- | --- |
| SN | NAME | MATRICULE NUMBER |
| 1 | DJOUMESSI DONGMO RONSARD CARNEGIE | FE21A174 |
| 2 | DJOUMESSI NGUEFACK IVAN | FE21A172 |
| 3 | GATCHUINNE MOMO MURIELLE CYNTHIA | FE21A200 |
| 4 | NJOYONG GODWILL | FE21A273 |
| 5 | TETUH WIBINAG ENGONWEI | FE21A322 |

# ABSTRACT

This project proposes the development of a Road Sign and Road State Mobile Notification Application. The goal is to leverage mobile technology and real-time data sources to create a dynamic and interactive platform for delivering road sign information and updates on road conditions to drivers. The mobile application will utilize GPS location tracking and crowdsourced data to provide users with relevant details about road signs, hazards, construction zones, and other critical factors affecting road safety.

The project will result in a user-friendly mobile application featuring a comprehensive database of road signs, the ability to receive timely updates on real-time road conditions, and customizable notification preferences based on user travel needs. The application will integrate seamlessly with popular navigation systems, allowing drivers to access road sign details and road state updates within their preferred mapping and routing apps.

By addressing the limitations of traditional static road signage and limited communication channels, this project aims to empower drivers with up-to-date, location-aware information to improve overall road safety and driving experiences. The development of this mobile application represents an innovative approach to enhancing driver awareness and decision-making through the strategic use of modern technology and crowdsourced data.

# TABLE OF CONTENTS

[1. ABSTRACT 3](#_Toc171206916)

[2. TABLE OF CONTENTS 4](#_Toc171206917)

[1. CHAPTER 1: GENERAL INTRODUCTION 8](#_Toc171206918)

[1.1. Overview 8](#_Toc171206919)

[1.2. Background and Context of the Study 9](#_Toc171206920)

[1.3. Problem Statement 10](#_Toc171206921)

[1.4. Problem Statement 11](#_Toc171206922)

[1.4.1. General Objectives 11](#_Toc171206923)

[1.4.2. Specific Objectives 12](#_Toc171206924)

[1.5. Proposed Methodology 13](#_Toc171206925)

[1.6. Significance of the study 14](#_Toc171206926)

[1.6.1. Improved Road Safety 14](#_Toc171206927)

[1.6.3. Enhanced Driver Experience 14](#_Toc171206928)

[1.6.4. Support for Diverse Driver Needs 14](#_Toc171206929)

[1.6.5. Contribution to Smart City Initiatives 14](#_Toc171206930)

[1.6.6. Business Opportunities and Scalability: 14](#_Toc171206931)

[1.7. Delimitation of the study 15](#_Toc171206932)

[1.7.1. Geographical Scope 15](#_Toc171206933)

[1.7.2. Data Sources 15](#_Toc171206934)

[1.7.3. Road Sign Database 15](#_Toc171206935)

[1.7.4. Platform Integration: 15](#_Toc171206936)

[1.7.5. User Demographic 16](#_Toc171206937)

[1.7.6. Monetization Strategies 16](#_Toc171206938)

[2. CHAPTER 2: MOBILE APP DEVELOPMENT PROCESS 16](#_Toc171206939)

[2.1. Introduction 16](#_Toc171206940)

[2.1.1. Types of mobile Apps 17](#_Toc171206941)

[2.2. Summary of Differences: Progressive Web Apps, Hybrid Apps, and Native Apps 20](#_Toc171206942)

[2.3. Review and Compare Mobile App Programming Languages 22](#_Toc171206943)

[2.5. Mobile Applications Architecture and design patterns 27](#_Toc171206944)

[2.6. HOW TO COLLECT AND ANALYSE USER REQUIREMENTS FOR MOBILE APP DEVELOPMENT 30](#_Toc171206945)

[2.7. Study How To Estimate Mobile App Development Cost 33](#_Toc171206946)

[3. CHAPTER 3 : REQUIREMENT GATHERING 35](#_Toc171206947)

[3.1. Introduction 35](#_Toc171206948)

[3.2. App Requirements 35](#_Toc171206949)

[3.2.1. Functional Requirements 35](#_Toc171206950)

[3.2.1.1. Higher priority Requirements 36](#_Toc171206951)

[3.2.1.2. Medium priority Requirements 37](#_Toc171206952)

[3.2.1.3. lower priority Requirements 38](#_Toc171206953)

[2.1.2. Non-Functional Requirements 39](#_Toc171206954)

[2.1.2.1. High Priority Requirements 39](#_Toc171206955)

[2.1.2.2. Meduim Priority Reaquirements 39](#_Toc171206956)

[2.1.2.3. Low priority requirement 40](#_Toc171206957)

[3. CHAPTER 4 : REQUIREMENT ANALYSIS 42](#_Toc171206958)

[4.1. Introduction 42](#_Toc171206959)

[3.1. App Requirements 42](#_Toc171206960)

[4.2.1. Functional Requirements 42](#_Toc171206961)

[4. Higher priority Requirements 43](#_Toc171206962)

[5. Medium priority Requirements 44](#_Toc171206963)

[6. lower priority Requirements 45](#_Toc171206964)

[3.1.2. Non-Functional Requirements 46](#_Toc171206965)

[High Priority Requirements 46](#_Toc171206966)

[Performance 46](#_Toc171206967)

[7. 46](#_Toc171206968)

[8. Meduim Priority Reaquirement 46](#_Toc171206969)

[Usability: 46](#_Toc171206970)

[Data Management: 47](#_Toc171206971)

[9. 47](#_Toc171206972)

[10. Low priority requirement 47](#_Toc171206973)

[Security: 47](#_Toc171206974)

[4.2.3 48](#_Toc171206975)

[4. CHAPTER 5: SYSTEM MODELLING AND DESIGN 50](#_Toc171206976)

[5.1. Introduction 50](#_Toc171206977)

[5.2. System Design 51](#_Toc171206978)

[5.2.3. System Architecture 51](#_Toc171206979)

[5.2.3. Behavioral Diagram 60](#_Toc171206980)

[5.2.4. Interaction Diagrams 61](#_Toc171206981)

[5. CHAPTER 6: UI DESIGN AND IMPLEMENTATION 65](#_Toc171206982)

[6.1. Introduction 65](#_Toc171206983)

[5.4. Wireframes and Mockups 68](#_Toc171206984)

[5.5. Design Specifications 69](#_Toc171206985)

[5.5.2. Typography 69](#_Toc171206986)

[5.5.3. Color Palette 69](#_Toc171206987)

[5.5.4. Iconography and Imagery 70](#_Toc171206988)

[5.5.5. UI Components and Patterns 70](#_Toc171206989)

[6. CHAPTER 7: DESIGN AND IMPLEMENTATION OF DATABASE 75](#_Toc171206990)

[7.1. Introduction 75](#_Toc171206991)

[7.2. Database Schema 75](#_Toc171206992)

[7.3. Implementation 78](#_Toc171206993)

[Backend Data Flow 80](#_Toc171206994)

[Figure 7.3:Backend Data Flow Diagram 80](#_Toc171206995)

[7.2. Implementation of database on Appwrite 80](#_Toc171206996)

[7.2.1. File structure of the of the frontend 80](#_Toc171206997)

[7.2.3. 82](#_Toc171206998)

[Database Security and Reliability 83](#_Toc171206999)

[7. CONCLUSION 84](#_Toc171207000)

[7.1. Challenges Faced 84](#_Toc171207001)

[7.2. Future Recommendations 84](#_Toc171207002)

[8. REFERENCES 85](#_Toc171207003)

# 

**Aucune entrée de table d'illustration n'a été trouvée.**

# CHAPTER 1: GENERAL INTRODUCTION

## Overview

This project will result in the development of a robust, user-friendly mobile application that leverages real-time data and crowdsourcing to enhance road safety and driving experiences for users. The app will provide drivers with instant access to critical information about road signs, traffic conditions, weather hazards, and other factors impacting travel.

At the core of the application will be an expansive database of road signs found on major highways and roads. Users will be able to quickly look up the meaning, purpose, and regulations associated with a wide variety of traffic signs, allowing them to better understand and respond appropriately to the signage they encounter. The database will include detailed information and visual representations for common road signs, as well as less prevalent or regionally-specific signage.

To deliver timely updates on current road conditions, the application will integrate data from numerous real-time sources. This will include live video feeds from traffic cameras, readings from weather and environmental sensors, and crowdsourced reports submitted by other app users. The app will monitor this data in the background and proactively push relevant alerts and notifications to drivers based on their location and travel route.

Alerts may include warnings about traffic congestion, accidents, construction zones, flooded roadways, icy conditions, and other potential hazards. Users will have the ability to customize their notification preferences, allowing them to receive tailored updates on the specific types of information most important to their driving needs and habits.

Beyond providing real-time road condition updates, the application will also seamlessly integrate with popular navigation platforms. This will enable drivers to access the app's road sign database and dynamic alerts directly within their preferred mapping and routing apps, ensuring the information is available when and where they need it most during their travels [[3].](#_Implementation)

The overall goal of this project is to empower drivers with a comprehensive, user-friendly tool that enhances their awareness of road signs and current driving conditions. By addressing the limitations of traditional static signage and limited communication channels, the mobile application represents an innovative approach to improving road safety and the driving experience.

## Background and Context of the Study

Ensuring road safety and providing timely information to drivers are critical priorities in modern transportation systems. However, the traditional methods used to convey road sign information and update drivers on current road conditions often fall short in meeting the needs of today's mobile, connected drivers [1].

Historically, road sign information has been primarily delivered through static signage placed along the roadways. While this approach can be effective for communicating basic regulatory, warning, and guidance information to drivers, it has several key limitations. Static signs are unable to provide dynamic, real-time updates on evolving road conditions. They also have a limited geographical reach, only informing drivers about the specific signs they physically encounter during their travels.

Similarly, conventional channels for communicating road state updates, such as radio traffic reports and dynamic message signs, offer a limited scope of coverage and lack the immediacy required to keep pace with rapidly changing driving environments. These communication methods often rely on centralized reporting systems that can be slow to identify and disseminate information about unexpected incidents, hazards, or changes in road conditions [2].

The emergence of ubiquitous mobile technology, abundant sensor data, and crowdsourcing platforms presents new opportunities to address these shortcomings. By leveraging smartphones, GPS tracking, and real-time data sources, it is now possible to develop intelligent, location-aware applications that can deliver comprehensive, up-to-the-minute information about road signs and driving conditions directly to drivers.

This project seeks to capitalize on these technological advancements by creating a mobile application that transforms the way road sign information and road state updates are accessed and consumed by drivers. By putting this critical data at the fingertips of users through a user-friendly, dynamic platform, the application has the potential to significantly improve road safety, driver awareness, and overall transportation network efficiency [7].

The development of this Road Sign and Road State Mobile Notification Application represents a timely and innovative response to the evolving needs and expectations of modern drivers. As transportation systems continue to grow in complexity, this project offers a compelling solution to enhance the driving experience and foster safer, more informed decision-making on the roads [5].

## Problem Statement

The current systems and methods for communicating road sign information and real-time road condition updates to drivers are inadequate and fail to meet the needs of today's mobile, connected driving population.

Static road signs, while effective for providing basic regulatory and guidance information, are limited in their ability to convey dynamic, location-specific details about evolving driving environments. These fixed signs cannot quickly respond to unexpected incidents, changing weather conditions, or other real-time factors impacting road safety and traffic flow.

Existing channels for disseminating road state updates, such as radio traffic reports and dynamic message signs, also suffer from limitations. These communication methods often rely on centralized reporting systems that can be slow to identify and share information about rapidly changing road conditions. Additionally, the geographical reach of these updates is restricted, leaving many drivers without access to critical, location-relevant driving information.

As a result, drivers frequently lack the timely, comprehensive knowledge required to make informed decisions and navigate the roadways safely. The lack of up-to-date, location-aware information about road signs and current road conditions contributes to increased risk of accidents, traffic congestion, and driver frustration.

This problem is exacerbated by the proliferation of mobile technology and the growing expectations of drivers to have instant access to real-time information through their smartphones and other connected devices. The current systems in place fail to fully leverage the capabilities of modern mobile platforms and data sources to deliver a seamless, user-centric driving experience [5].

To address these shortcomings, there is a pressing need to develop an innovative, technology-driven solution that can provide drivers with immediate access to critical road sign information and dynamic updates on current road conditions. This solution must leverage the ubiquity of mobile devices, the abundance of real-time data sources, and the power of crowdsourcing to transform the way drivers receive and consume vital transportation-related information.

By developing a comprehensive, user-friendly mobile application that addresses the limitations of existing road sign and road state communication channels, this project aims to enhance road safety, improve driver awareness, and promote more efficient transportation network operations.[6]

# Problem Statement

### General Objectives

The primary objective of this project is to develop a comprehensive, user-friendly mobile application that enhances road safety and driving experiences by providing drivers with immediate access to critical information about road signs and real-time updates on current road conditions.

The application will address the limitations of traditional methods for communicating road sign information and disseminating road state updates. It will deliver a comprehensive database of road signs, integrate real-time data from multiple sources to provide timely alerts and notifications about evolving road conditions, leverage the capabilities of modern mobile devices to proactively push relevant, location-based information to drivers, and seamlessly integrate with popular navigation platforms.

By achieving these objectives, the mobile application will empower drivers with a comprehensive, dynamic tool that enhances their awareness of road signs and current driving conditions, ultimately contributing to improved road safety, reduced traffic congestion, and a more efficient transportation network.[1]

### Specific Objectives

* Develop a comprehensive road sign database for easy lookup of sign meanings, purposes, and regulations.
* Integrate real-time data from multiple sources to provide timely alerts and notifications about evolving road conditions like traffic, accidents, construction, and weather hazards.
* Leverage mobile device capabilities to proactively push relevant, location-based information to drivers.
* Seamlessly integrate the application with popular navigation platforms.
* Design a clean, intuitive user interface for distraction-free access to application features.
* Empower drivers with a comprehensive tool to enhance their awareness of road signs and driving conditions, improving safety and efficiency.
* Develop a scalable and flexible architecture to accommodate new data sources and features.
* Ensure robust data privacy and security measures.
* Implement a user-centric design approach to meet the needs of diverse driver demographics.
* Establish a sustainable business model and monetization strategy for long-term viability.

## Proposed Methodology

1. **Mobile App Development Process**
2. **Requirement Gathering**

* Conduct user research to understand pain points and needs
* Collaborate with experts to identify critical data sources

1. **Requirement Analysis**

* Analyze existing solutions to identify gaps and opportunities

1. **System modelling and Design**

* Define the high-level structure and components of the system
* Identify the key subsystems, their interactions, and interfaces
* Select appropriate architectural patterns and styles

1. **User Interface Design and Development**

* Follow a user-centric design approach based on user research
* Create an intuitive, visually appealing, and distraction-free interface
* Implement accessibility features for diverse user needs

1. **Database Design and Development**

* Design a scalable database architecture for road sign information
* Implement efficient data storage and retrieval mechanisms
* Establish processes for regularly updating and expanding the database

1. **Final implementation**

* Data Integration and Aggregation
* Platform Integration and Deployment
* Ongoing Maintenance and Enhancements

## Significance of the study

### Improved Road Safety

* Providing drivers with comprehensive road sign information and real-time alerts to enhance awareness and safer driving behaviors.
  + 1. **Reduced Traffic Congestion**
* Equipping drivers with up-to-date information about traffic flow and disruptions to help them make informed route choices and improve transportation network efficiency.

### Enhanced Driver Experience

* Offering a more intuitive and convenient way for drivers to access critical driving information through user-centric design and seamless integration with popular navigation platforms.

### Support for Diverse Driver Needs

* Catering to the unique requirements and preferences of various driver demographics, including accessibility features for inclusivity.

### Contribution to Smart City Initiatives

* Providing valuable insights to transportation authorities and urban planners to support the development of more intelligent and responsive transportation systems.

### Business Opportunities and Scalability:

* Potential for revenue generation through monetization strategies and the ability to easily incorporate new data sources and expand features based on user feedback and evolving transportation needs.

# Delimitation of the study

### Geographical Scope

The initial focus of the study and development of the application will be on a specific geographical region or country, where the transportation infrastructure and regulatory environment have been thoroughly analyzed.

The intention is to establish a successful model in the initial target market before considering expansion to other regions.

### Data Sources

The study will primarily rely on integrating data from established, reputable sources, such as government transportation agencies, road authorities, and reputable weather and traffic monitoring services.

While the application may eventually incorporate crowdsourced data from users, the initial development phase will focus on ensuring the accuracy and reliability of the core data inputs.

### Road Sign Database

The initial road sign database will be focused on the most common and critical road signs within the target geographical region, as identified through extensive research and consultation with transportation experts.

The database will be designed to be scalable and expandable, allowing for the addition of new road sign types and updates as the application evolves [3].

### Platform Integration:

The study will prioritize the integration of the application with the most widely used navigation platforms within the target market, ensuring seamless user experience and maximum reach.

While the long-term vision may include integration with a broader range of platforms, the initial development will concentrate on the dominant players in the navigation app ecosystem.

### User Demographic

The primary target user demographic for the initial launch of the application will be private vehicle drivers and commuters within the chosen geographical region.

While the application may eventually expand to cater to other road user groups, such as commercial drivers or public transportation users, the initial focus will be on the needs of individual private vehicle drivers.

### Monetization Strategies

The study will explore various monetization strategies, such as advertising, premium subscriptions, and partnerships with transportation service providers, to ensure the long-term sustainability of the application.

However, the initial development phase will prioritize providing a valuable user experience over maximizing revenue, with a focus on establishing a strong user base and brand reputation[7].

# CHAPTER 2: MOBILE APP DEVELOPMENT PROCESS

## 2.1. Introduction

What's An APP?

An app is software that allows you to exchange information with customers and help them complete specific tasks. The different types of applications differ in their development method and internal functionality. Web applications are delivered via a web browser. Users don't need to install them on their devices. Native apps, on the other hand, are designed for a specific platform or device type. The user must install the correct version of the software on the device of their choice. Hybrid apps are native, but have a web browser built in.

### Types of mobile Apps

**Native applications**

When you think of an app, you likely picture a branded little icon on your smartphone’s screen.

In reality, mobile apps from an app store are just one type of app. They’re called native mobile apps.

Native apps developed for Android are [written in Java](https://www.upwork.com/resources/java-tips-app-development-project), while apps developed for iOS are written in Swift (you may find older iOS apps written in Objective-C).

Native mobile apps are typically faster and more reliable than hybrid or web apps, which lets them deliver a better user experience (UX).

They also let you interact with a device’s application programming interface (API) and internal hardware, granting your company’s app access to features like:

**Advantages of Native Apps:**

* Superior performance, even for heavy graphics
* Access to platform-specific features on iOS or Android
* Superior user interface that meets platform-specific standards
* Better visibility and discoverability on app stores compared to web or hybrid apps[1]

**Disadvantages of Native Apps:**

* Greater upfront costs, especially if developing for multiple platforms which requires hiring multiple development teams
* Requires experienced app developers proficient in platform-specific languages like Swift or Java
* Need to start over from design and development if expanding to a new platform (e.g. from iOS to Android)[1]

**Web applications:**

A Progressive Web App (PWA) is a type of web application that uses modern web technologies to deliver an app-like experience to users. It aims to combine the best features of both web and native mobile applications.

PWAs are designed to be reliable, fast, and engaging, providing an immersive user experience similar to that of native apps. They can be accessed through a web browser, eliminating the need for users to download and install them from an app store.

**Key features of Progressive Web Apps include:**

1. . Responsive Design: PWAs are built to be responsive and adaptive, offering a consistent experience across different devices and screen sizes.
2. App Shell: times and offline functionality. PWAs make use of an app shell model, which separates the core application infrastructure from the dynamic content. This allows for faster loading
3. Service Workers: Service workers are JavaScript files that run in the backgroun d and enable PWAs to work offline or with a poor network connection. They can cache data, handle push notifications, and perform background synchronization.
4. App Manifest: PWAs have a web app manifest, which is a JSON file that provides metadata about the application, such as its name, icons, and color scheme. This allows PWAs to be installed on a user's device and appear like a native app.
5. Push Notifications: PWAs can send push notifications to users, even when the application is not actively running. This helps in engaging users and providing timely updates.
6. Secure Connections: PWAs are required to be served over HTTPS to ensure data security and prevent unauthorized access[5]

**Advantages of Web Apps:**

* Lower entry barrier and cheaper/easier to develop compared to native apps
* Can be accessed from any device with a web browser, improving accessibility
* Easy to maintain a common codebase across platforms
* No need for app store approval or user updates

**Disadvantages of Web Apps:**

* Tend to be slower and have less intuitive user interfaces compared to native apps
* Cannot access many native device features like location tracking or push notifications
* More complex user experience since they are browser-based
* Users may be more easily distracted and leave the app compared to being in a native app

**Progressive web applications**

Progressive Web Apps are a type of web application that aims to combine the best features of web and native mobile apps. Key characteristics of PWAs include:

* Responsive design that adapts to different devices and screen sizes
* An "app shell" model for faster loading times and offline functionality
* Use of Service Workers to enable offline access, caching, and background tasks
* A web app manifest that allows PWAs to be installed on a user's device
* Push notification support to engage users

**Advantages of PWAs :**

1. Cross-platform compatibility - they work across desktop, mobile, and tablet.
2. Offline functionality through caching and Service Workers.
3. Discoverability through web search engines.
4. No dependence on app stores for distribution.
5. Cost-effectiveness compared to developing separate native apps.

**Disadvantages of PWA’s**

1. Access to fewer device-specific hardware and software features.
2. Inconsistent support across different browsers and platforms.
3. Potentially lower user awareness compared to native apps.

**[Hybrid applications](https://www.upwork.com/nx/signup/?dest=job-posting)**

Hybrid mobile apps are a combination of web apps and native mobile apps. They are built with two main components:

1. Backend code - typically web technologies like HTML, CSS, and JavaScript.
2. A native "shell" that allows the app to be installed and run on a device.

Hybrid apps share some similarities with progressive web apps (PWAs), as they both aim to provide an app-like experience through web technologies. However, the key differences are:

* Hybrid apps can be distributed through app stores, while PWAs are accessed directly through the web browser.[2]
* Hybrid apps leverage native device features to a greater degree than PWAs.

The main advantages of hybrid apps include:

1. Developer productivity - the ability to use common web technologies.
2. Lower development costs compared to building separate native apps.
3. Cross-platform availability on both Android and iOS.
4. Access to some native device features like location and push notifications.

However, hybrid apps also have some drawbacks:

1. Potentially inconsistent user experience, as performance depends on internet speed.
2. A learning curve for integrating with hybrid app frameworks like Cordova, Ionic, or React Native.

## Summary of Differences: Progressive Web Apps, Hybrid Apps, and Native Apps

|  |  |  |  |
| --- | --- | --- | --- |
| **Characteristics** | **Web application** | **Hybrid Application** | **Native app** |
| Usage | Users can access directly from a browser | Users need to install the app on the device of their choice | Users need to install the app on the device of their choice |
| Inner workings | Client code in the browser communicates with remote server-side code and databases | Client code and browser code encapsulated in a native shell or container | Client code written in technology and language specific to the device or platform on which it will be installed |
| Native device features | Not accessible | Accessible | Accessible |
| User Experience | Inconsistent and browser-dependent | Consistent and engaging | Consistent and engaging |
| Access | Limited by browser and network connectivity | One-step access with offline features | One-step access with offline features |
| Performance | Slower and less responsive | Faster, but can consume more battery | Performance can be optimized for the device |
| Development | Cost-effective and faster time-to-market | Cost-effective and faster time-to-market | Costly and slower time-to-market |
|  |  |  |  |

## Review and Compare Mobile App Programming Languages

In the modern era, mobile app development is rapidly evolving. Different mobile programming languages have unique strengths, weaknesses, and criteria that can impact a developer's choice. The programming language selected can affect the development process and the app's performance. This review will cover the best mobile programming languages.[3]

**Best Native iOS Mobile Programming Languages**

1. **Swift**
   * A modern, open-source language developed by Apple for iOS app development
   * Known for speed, safety features, and expressive syntax
   * Advantages: Improved safety and performance, modern syntax, active community
   * Disadvantages: Limited to iOS platform
2. **Objective-C and C#**
   * Objective-C was the primary iOS development language before Swift
   * C# provides a robust and familiar environment for powerful mobile apps
   * Advantages: Mature, stable ecosystems; interoperability with C; useful for legacy codebases
   * Disadvantages: Complicated, tedious syntax compared to Swift

**Best Native Android Mobile Programming Languages**

1. **Java**
   * A widely used language known for "write once, run anywhere" philosophy
   * Advantages: Platform independence, robustness, scalability, broad community support
   * Disadvantages: Complex syntax resulting in lengthy code
2. **Kotlin**
   * A modern, statically-typed language with seamless Java interoperability
   * Advantages: Concise, expressive syntax; null safety features
   * Disadvantages: Smaller community compared to Java

**Best Cross-Platform Mobile Programming Languages**

1. **JavaScript (React Native)**
   * Allows building native mobile apps for iOS and Android using a single codebase
   * Advantages: Cross-platform development, code reusability, large developer community
   * Disadvantages: Platform-specific functionality may require additional native coding
2. **Flutter (Dart)**
   * An open-source UI framework for building natively compiled applications
   * Advantages: Cross-platform development, strong community, fast and customizable UI
   * Disadvantages: Limited libraries and integrations compared to more established languages

**Comparison of JavaScript and Flutter**

* Speed: JavaScript is interpreted and lighter, but Dart is faster
* Ease of Use: JavaScript is more mature and stable, while Dart can be more challenging for developers
* Type Support: JavaScript supports dynamic and duck typing, while Dart has stronger, more robust type support
  1. **Compare Mobile App Developoment Frameworks By Comparing In Their Key Features**

**Frontend Frameworks for Mobile App Development:**

**Native App Development Frameworks:**

* **Android Development Frameworks** 
  + Kotlin - A modern, concise language that is fully interoperable with Java. Offers features like Kotlin Multiplatform for cross-platform development.
  + Costs range from $30,000-$300,000 based on complexity, with development times of 2-15 months.
  + Provides a modern UI toolkit and strong community support[4].

**Hybrid App Development Frameworks:**

* Ionic - An open-source framework for building cross-platform mobile apps using web technologies like HTML, CSS, and JavaScript.
  + Costs and development times similar to native frameworks.
  + Provides a rich library of customizable UI components and navigation patterns.
  + Performance optimization is crucial for delivering a smooth user experience.
  + Limitations include declining popularity and the phasing out of the underlying Cordova technology.
* React Native - Developed by Facebook, allows creating near-native experiences for iOS and Android with a single codebase.
  + Costs and development times similar to native frameworks.
  + Component-based UI development and access to native APIs enable a native-like user experience.
  + Has a large and active developer community providing support.
  + Advantages include fast development through features like Fast Refresh and integration with the Flipper debugger.
  + Limitations include bottlenecks in long-term testing and a smaller ecosystem of third-party libraries compared to some other frameworks[6].

Ionic:

* Ionic was built on top of Apache Cordova, but has not been Cordova-based since May 2021. The App Center service no longer accepts Cordova SDK calls, so Cordova is not recommended for new projects.
* Ionic can still be used for cross-platform app development, but its popularity has declined significantly, making it harder to find engineers to develop or support Ionic apps.

React Native:

* Developed by Facebook, React Native allows developers to build near-native mobile apps for both iOS and Android using a single codebase in JavaScript.
* It has a large and active community that provides good support.
* React Native focuses on the UI, with native platform components and the ability to customize the user experience.
* It has features like Fast Refresh and integration with the Flipper debugger that improve development and performance.
* Example apps built with React Native include Microsoft Office, Skype, and Facebook.

Flutter:

* Developed by Google, Flutter is an open-source UI toolkit for building natively compiled apps for mobile, web, and desktop from a single codebase.
* It uses the Dart programming language and offers a wide range of customizable widgets and layout options for creating visually rich and complex user interfaces.
* Flutter has an active community and strong performance with features like hot reload and native rendering.[1]
* However, it may not be sufficient for certain native-dependent, augmented reality, or 3D game development use cases.

Xamarin (.NET MAUI):

* Owned by Microsoft, Xamarin allows developers to build cross-platform apps using C# and .NET.
* It provides a streamlined approach to creating cross-platform user interfaces using XAML or C#, with the ability to customize UI elements.
* Xamarin focuses on delivering a smooth and responsive user experience across platforms, with access to native device features.
* The .NET MAUI framework, which is the latest iteration of Xamarin, has an active and growing community.

NativeScript:

* NativeScript allows developers to build native mobile apps using JavaScript or TypeScript.
* It provides direct access to native Android and iOS APIs, rendering platform-native UIs without relying on WebViews.
* NativeScript offers various plugins and pre-built app templates, but has seen a decline in community activity and third-party support.

UI Complexity:

* NativeScript provides access to native UI components for building mobile app interfaces, allowing developers to use XML markup or JavaScript/TypeScript.
* NativeScript allows creating custom UI components using JavaScript/TypeScript and CSS, giving flexibility in UI design but potentially increasing development complexity.

UX Complexity:

* Designing intuitive navigation structures and user flows is important for a good user experience. NativeScript offers navigation patterns like stack, tab, and side drawer to facilitate seamless in-app navigation.
* Performance optimization is crucial - developers need to optimize UI rendering, app startup time, and network requests to ensure a smooth, responsive user experience across platforms.

Community Support:

* NativeScript leverages well-known web technologies like JavaScript and Angular, which appeals to many developers. However, it is typically used by smaller companies and startups.

Limitations of .NET MAUI:

* .NET MAUI is a new technology that lacks some essential UI controls.
* User experience is not a strong point of MAUI, so it is more suitable for projects focused on functionality rather than appearance and UX.

Backend Development Frameworks:

* Django (Python) - Supports the DRY principle, has excellent REST API support, and provides robust security and content management features.
* Express.js (JavaScript) - Offers flexibility for client and server-side routing, middleware functions, and HTTP methods to reduce programming complexities.
* Laravel (PHP) - Provides a modern and secure CLI, database migration tools, extensive libraries/APIs, and innovative security features.[4]
  1. Mobile Applications Architecture and design patterns**.**

Design patterns are reusable solutions to common software development problems. They have had a significant impact on software development, including mobile app development. The implementation of mobile apps has established some proven models and standards to overcome the challenges and limitations of mobile app development.

Most mobile applications were built with low code and were not based on architecture. Mobile app development with the right design patterns can effectively integrate user interfaces with data models and business logic. This will affect the quality of your source code.

There are very few architectural design patterns available for mobile development.

**a. Monolithic Architecture**:

* In a monolithic architecture, all components of the app are tightly coupled within a single codebase.
* This simplicity can make development and testing easier, but it can lead to challenges in scaling and maintenance.

**b. Microservices Architecture:**

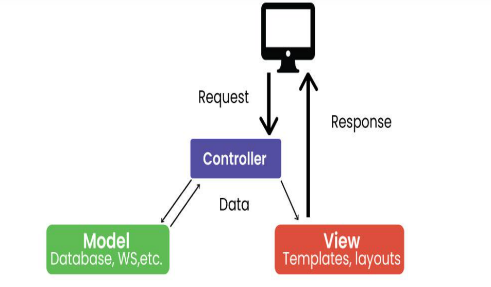
* Microservices architecture involves breaking the app into smaller, independent services that communicate with each other via APIs.
* It promotes flexibility, scalability, and easier maintenance but can introduce complexity.

**c. Layered Architecture:**

* A layered architecture divides the app into distinct layers, such as presentation, business logic, and data.
* This approach simplifies management and separation of concerns.

**d. Model-View-Controller (MVC):**

* MVC is a design pattern that separates the app into three interconnected components: the model (data), the view (UI), and the controller (business logic).
* It's widely used for building mobile apps, especially in iOS and Android development.



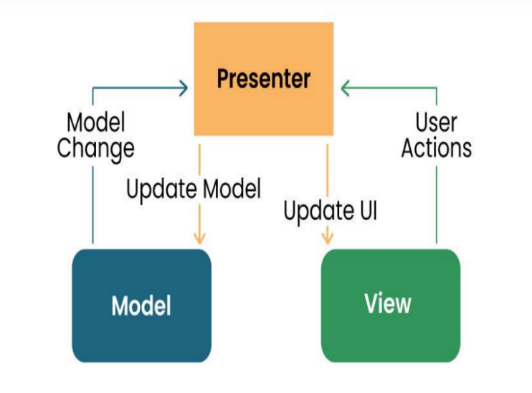
**Model**: Represent Represents application data and business logic.

**View:** Displays data to the user.

**Controller:** Processes user input and controls data flow between Model and View.

Figure 1 MVC pattern

**e. Model-View-Presenter (MVP):**

* MVP is a variation of MVC where the presenter acts as an intermediary between the view and the model.
* It enhances testability and separation of concerns.

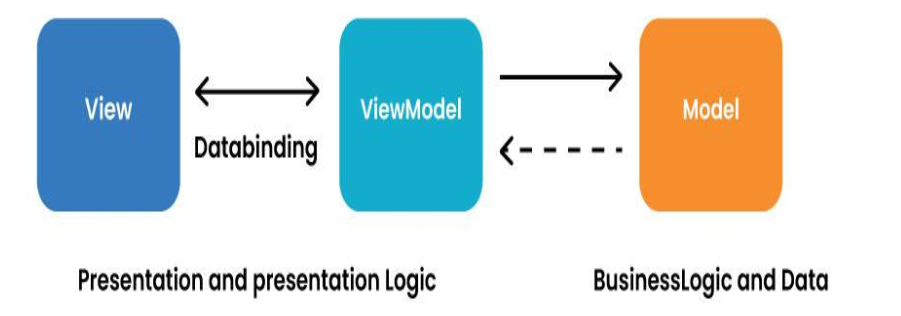
**Model:** Manages data and business logic.

**View**: Represents the user interface.

**Designer:** Acts as an intermediary processing user input and updating the View and Model.

**f. Model-View-ViewModel (MVVM):**

* MVVM is a design pattern popular in Android development.
* It separates the UI (View), business logic (ViewModel), and data (Model) into distinct components.



**Model:** Represents data and business logic.

**View:** Represents the user interface.

**ViewModel:** Acts as an interface between the Model and the View, which contains the reference logic.

## HOW TO COLLECT AND ANALYSE USER REQUIREMENTS FOR MOBILE APP DEVELOPMENT

**Introduction**

According to the software development life cycle the first phase of this process is the requirement gathering phase followed by requirement analysis phase. These phases are very essential for the overall efficiency of a software and acts as a blue print for quality assurance.

Requirement gathering can be done using different methods such as

* Focus group meetings
* Surveys
* By examining pre-requisite data to bring about statistical facts
* Questionnaires

All these requirements should be gotten from the targeted market, the target market could be a group of people such as a company or a research institute, a particular geographical location such as Africa or maybe Cameroon, maybe a specific gender, or a specific group age.

**User requirements focuses on who you are targeting with your application.**

* What are your target audience’s age group, lifestyle, and social background?
* Which geographical regions and languages are you targeting?
* What issues are your potential users facing that your app can solve?
* With which requests are they turning to your app?
* Which device types (smartphone, tablet, etc.) and OS (Android, iOS) do your potential users prefer?
* What are the parameters they value a lot in a digital product (e.g., simplicity, unusual design, wearables integration)

Based on your input, we usually create a user persona – these are imaginary users of your future app. We describe the smallest details about them, from work life to hobbies. Such methodology helps us have a clear understanding of who we are designing an app for and what they would be happy to see.[5]

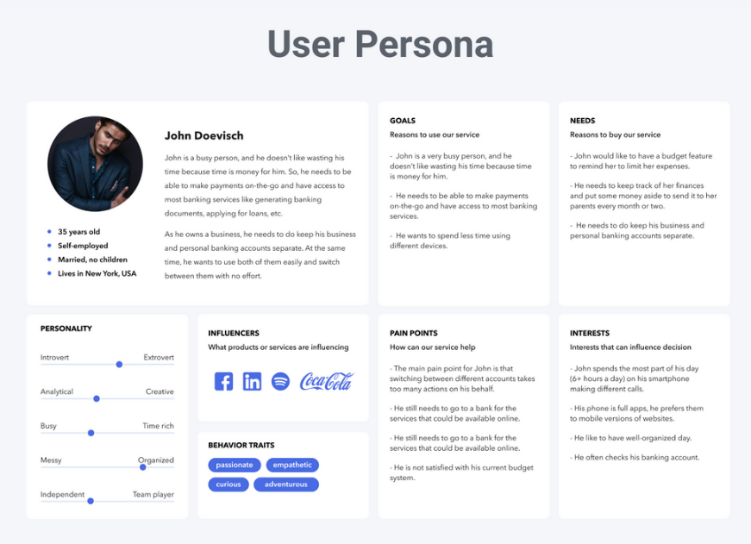


Figure 1.1: user persona

After the requirement gathering phase these requirements have to be analyzed. This analysis is done by the business analyst although some institutes have a requirement analyst. Analysis is very important as it helps:

* Capturing user stories or use cases
* Defining system constrains and interfaces
* Prioritized requirements based on their importance and feasibility
* Indicate a measurement of the total budget
* Outline functional and non-functional requirements

1. **Functional requirements:**

Functional requirements describe the specific functionalities, features and behaviors that a software system must exhibit in other to fulfil the needs of the user.

It typically answers questions like what should the system do?

Examples of this requirements include

* The system should allow users to create accounts and login
* The system should allow the prices of each item in a shopping cart

1. **Non-functional requirements:**

Also known as quality attributes or constraints, define the characteristics and qualities that the software system should possess. They are not directly related to specific functionalities but rather focus on aspects of the system’s performance, usability, security, scalability and other qualities that impacts its overall effectiveness. Examples include:

* Security
* Reliability
* Usability

## 2.7. Study How To Estimate Mobile App Development Cost

**Methods for Estimating Mobile App Development Costs**

**Fixed Price Model**: Under this model, [itCraft](https://itcraftapps.com/about-us/) estimates the total cost of the project based on the defined requirements and specifications before the development starts. This method is most suitable for small projects with clear, unchanging requirements. The main advantage is that the client knows the exact cost upfront, helping with budget planning. However, it lacks flexibility for changes or additions once the project is underway.

**Time and Material Model**: This model is based on the actual time and resources utilized on the project. Clients are billed according to the hours worked by the development team. This approach is suitable for projects where requirements are expected to evolve or are not fully defined at the beginning. It offers high flexibility as changes can be made during the development process, but the final cost might exceed the initial estimates if the project scope expands or unexpected challenges occur.

**Dedicated Team Model**: In this model, itCraft provides a dedicated team of professionals who work exclusively on your project. The cost is based on the team size and composition, and the time they spend on the project. This model is ideal for long-term projects where requirements might change over time. It gives you greater control over the development process and ensures that the team is fully invested in your project.[7]

At itCraft, we understand that each project is unique. Therefore, we offer these different [pricing](https://itcraftapps.com/mobile-app-development-cost/) models to cater to the specific needs and requirements of each client. Our main goal is to deliver high-quality mobile app solutions in a cost-effective and transparent manner. Before deciding on a pricing model, we thoroughly discuss the project with the client to ensure that the chosen model aligns with their project goals and budget.

**Tips for Accurate Cost Estimation**

**Ensuring Clear Communication with the Development Team**

Open and continuous communication with your **development team is critical for accurate cost estimation.** As a client, you should be clear and precise about your requirements, expectations, and budget. At itCraft, we value transparency and encourage clients to discuss their project thoroughly with us. This helps in understanding the project scope better, eliminating misunderstandings, and providing an accurate estimate.

**Taking into Account Unexpected Costs**

No matter how well you plan, there will always be some unforeseen expenses during the app development process. These could be due to sudden changes in requirements, extra time spent on bug fixing, or additional features requested by users after launch. Therefore, it’s wise to set aside a contingency fund as part of your budget. This prepares you for unexpected costs and ensures that the development process isn’t hampered due to budget constraints.

**Regularly Revisiting and Revising the Budget**

Mobile app development is a dynamic process, and costs can fluctuate based on a variety of factors. For example, additional features may be needed, or development could take longer than expected. It’s important to revisit and revise your budget periodically, keeping track of the actual spend against the estimated cost[2]. This will help identify if the project is on track financially, and if not, corrective actions can be taken early.

# CHAPTER 3 : REQUIREMENT GATHERING

## Introduction

Requirements analysis, also known as requirements engineering, is a crucial phase in the development of a new product or the modification of an existing one. It involves understanding and defining the needs and expectations of users or stakeholders to ensure that the final product meets their requirements. This process typically requires the collaboration of a team with diverse skills and expertise.

One essential aspect of requirements analysis is the identification and documentation of user expectations[7]. This involves gathering information about the desired functionalities, features, and qualities of the product. The requirements may encompass both functional aspects (what the product should do) and non-functional aspects (qualities such as performance, reliability, and usability).

## App Requirements

### Functional Requirements

Functional requirements are specific features and capabilities that the application should possess to fulfill its purpose effectively. During the brainstorming session, the internal stakeholders likely discussed and documented functional requirements.

We evaluated these features based on the following standards.

* Clarity
* Completeness
* Consistency
* Feasibility

Functional requirements are specific features and capabilities that the application should possess to fulfill its purpose effectively. During the brainstorming session, the internal stakeholders likely discussed and documented functional requirements such as

#### Higher priority Requirements

They are those gotten from surveys with high number of votes and interviews since they correspond to the ones directly gotten from the users additionally the ones gotten from retro-engineering.

1. **Real-time Traffic Updates**:

Integrate a feature that provides

* Real-time updates on construction zones
* Weather conditions
* Road closures, hazards, and checkpoints.
* Providing timely and accurate updates on the current state of the road
* Including information on road conditions and traffic congestion.

This could be achieved by tapping into various data sources such as government agencies, traffic cameras, and user reports.

1. **Location-Based Services**

* Utilize GPS data to provide location-based services
* allowing users to receive updates relevant to their current location

1. **Automatic Start and Stop**

Implement a feature that automatically starts and stops updating based on user preferences or when the app detects the user is driving.

1. **Display Road Signs**

Incorporate technology that recognizes and displays road signs in the app interface or overlay them on the video feed if applicable.

1. **User Feedback and Collaboration**

Allow users to provide feedback on the accuracy of updates and collaborate by reporting road conditions or incidents they encounter.

1. **Audio Notifications**

Implement audio notifications for important updates to minimize distractions. Users can customize the types of notifications they receive audibly.

1. **Current Speed, GPS Data, Date, and Time**

Embed relevant information such as current speed, GPS data, date, and time into the video file to provide additional context and ensure accuracy.

#### Medium priority Requirements

Addressing the medium priority requirements involves enhancing navigation features, optimizing routes, and improving data visualization. Here's how these requirements can be implemented.

1. **Navigation Assistance**

Provide turn-by-turn navigation assistance to users, guiding them along their chosen route. This can include voice prompts, visual cues, and lane guidance to help drivers navigate complex intersections.

1. **Data Visualization**

Implement data visualization techniques to present information such as traffic congestion, road closures, and construction zones in an intuitive and easy-to-understand manner. This could include color-coded maps, charts, and graphs to highlight relevant data points.

1. **Route Optimization**

Develop algorithms that analyze real-time traffic data to suggest the most efficient routes to users. This can help drivers avoid congestion, reduce travel time, and save fuel.

1. **Navigation Assistance with Rerouting Options**

Offer users the ability to reroute their journey in case of unexpected obstacles or changes in traffic conditions. This could be triggered automatically based on real-time updates or initiated by the user manually.

1. **Speed Camera Warnings**

Integrate a feature that alerts users to the presence of speed cameras along their route. This can help drivers adhere to speed limits and avoid potential fines.

1. **Exposure Compensation Feature**

Implement a feature that automatically adjusts the brightness levels of the app interface or camera feed to compensate for changes in lighting conditions. This ensures that the display remains visible and readable at all times, even in varying light conditions.

#### lower priority Requirements

Addressing the lower priority requirements involves adding supplementary features that enhance user convenience and interaction with the application. Here's how these requirements can be implemented:

1. **Hands-Free Reporting Capabilities**:

Develop a hands-free reporting feature that allows users to report incidents, such as accidents or road hazards, using voice commands or gesture controls. This ensures that users can report incidents safely while driving without being distracted.

1. **Integration of Existing Traffic Monitoring Systems**

Collaborate with existing traffic monitoring systems, such as government agencies or private companies, to integrate their data into the application. This can provide users with access to additional real-time traffic information and improve the accuracy of updates.

1. **User Incident Reporting**

Enable users to report incidents they encounter on the road, such as accidents, potholes, or debris. This crowdsourced data can supplement existing traffic information and help other users navigate more safely.

1. **Easy Sharing of Captured Video Footage on YouTube**

Implement a feature that allows users to easily share captured video footage of their journey on YouTube or other social media platforms. This can be done through integration with the respective platforms' APIs and providing a streamlined sharing process within the application.

### Non-Functional Requirements

#### High Priority Requirements

**Responsiveness:**

* + The app should load quickly and respond to user actions instantly to avoid frustration while driving. This is critical for safety reasons.

**Low Latency Notifications:**

* + Real-time updates on road conditions need to be delivered with minimal delay to ensure users have the latest information for safe navigation.

**Battery Efficiency:**

* + Since the app might run in the background for extended periods, it should be optimized to minimize battery consumption on mobile devices.

#### Meduim Priority Reaquirements

* **Intuitive Interface:**
  + The user interface (UI) should be clean, uncluttered, and easy to navigate, even for users with limited experience with smartphones. Icons and buttons should be clear and easily recognizable.
* **Minimal Distraction:**

Visual information and notification alerts should be designed to be clear and informative at a glance, minimizing the time users need to look away from the road.

* **Voice Control Options:**
  + Consider incorporating voice commands for hands-free interaction with the app, allowing users to access information or report incidents without taking their eyes off the road.

**Data Management**

* **Road Sign Information Source:**
  + Decide whether road sign data will be pre-loaded on the app or updated dynamically. Pre-loaded data might be suitable for static signs, while real-time updates are crucial for temporary signs or variable message boards.
* **Real-time Data Acquisition:**
  + Explore options for obtaining real-time road state data. This could involve integrating with existing traffic management systems, leveraging data from connected vehicles, or incorporating user reports.
* **Data Update Mechanism:**
  + Establish a system for updating the app's database with new signs, changes in road conditions, or any other relevant information. This might involve automatic updates or user-triggered refresh options.

#### Low priority requirement

* **User Privacy:**
  + The app should collect minimal user data, ideally just location for notification purposes. This data should be anonymized and used only for the stated purpose of improving road safety.
* **Data Security:**
  + The app should implement robust security measures to protect user data from unauthorized access, hacking attempts, or malware. Secure data storage and transmission protocols are essential.

**Technical requirements Gathered**

The following are the technical requirements analyzed for the proper execution of this project:

**Frontend Framework for Mobile App:**

* **React Native**

A popular framework for building cross-platform mobile apps using JavaScript and React.

**Backend Frameworks for Mobile App:**

* **Express.js**

A fast and minimalist web application framework for Node.js.

* **Laravel**

A PHP framework for building scalable and feature-rich web applications.

**Database Frameworks for Mobile App**

* **Firebase**: A comprehensive backend platform by Google that offers real-time database, authentication, storage, and more.
* **MongoDB Stitch**: A serverless platform for MongoDB that provides data synchronization and backend services.
* **SQLite**: A lightweight and embedded database engine that is widely used in mobile app development.

**Modelling and System Design**

* StarUML

This is an open source software used for building usecase diagrams and modelling of the system

# CHAPTER 4 : REQUIREMENT ANALYSIS

# 4.1. Introduction

Requirements analysis, also known as requirements engineering, is a crucial phase in the development of a new product or the modification of an existing one. It involves understanding and defining the needs and expectations of users or stakeholders to ensure that the final product meets their requirements. This process typically requires the collaboration of a team with diverse skills and expertise.

One essential aspect of requirements analysis is the identification and documentation of user expectations. This involves gathering information about the desired functionalities, features, and qualities of the product. The requirements may encompass both functional aspects (what the product should do) and non-functional aspects (qualities such as performance, reliability, and usability).

## App Requirements

### Functional Requirements

Functional requirements are specific features and capabilities that the application should possess to fulfill its purpose effectively. During the brainstorming session, the internal stakeholders likely discussed and documented functional requirements.

We evaluated these features based on the following standards.

* Clarity
* Completeness
* Consistency
* Feasibility

Functional requirements are specific features and capabilities that the application should possess to fulfill its purpose effectively. During the brainstorming session, the internal stakeholders likely discussed and documented functional requirements such as

1. Higher priority Requirements

They are those gotten from surveys with high number of votes and interviews since they correspond to the ones directly gotten from the users additionally the ones gotten from retro-engineering.

1. **Real-time Traffic Updates**:

Integrate a feature that provides

* Real-time updates on construction zones
* Weather conditions
* Road closures, hazards, and checkpoints.
* Providing timely and accurate updates on the current state of the road
* Including information on road conditions and traffic congestion.

This could be achieved by tapping into various data sources such as government agencies, traffic cameras, and user reports.

1. **Location-Based Services**

* Utilize GPS data to provide location-based services
* allowing users to receive updates relevant to their current location

1. **Automatic Start and Stop**

Implement a feature that automatically starts and stops updating based on user preferences or when the app detects the user is driving.

1. **Display Road Signs**

Incorporate technology that recognizes and displays road signs in the app interface or overlay them on the video feed if applicable.

1. **User Feedback and Collaboration**

Allow users to provide feedback on the accuracy of updates and collaborate by reporting road conditions or incidents they encounter.

1. **Audio Notifications**

Implement audio notifications for important updates to minimize distractions. Users can customize the types of notifications they receive audibly.

1. **Current Speed, GPS Data, Date, and Time**

Embed relevant information such as current speed, GPS data, date, and time into the video file to provide additional context and ensure accuracy.

1. Medium priority Requirements

Addressing the medium priority requirements involves enhancing navigation features, optimizing routes, and improving data visualization. Here's how these requirements can be implemented.

1. **Navigation Assistance**

Provide turn-by-turn navigation assistance to users, guiding them along their chosen route. This can include voice prompts, visual cues, and lane guidance to help drivers navigate complex intersections.

1. **Data Visualization**

Implement data visualization techniques to present information such as traffic congestion, road closures, and construction zones in an intuitive and easy-to-understand manner. This could include color-coded maps, charts, and graphs to highlight relevant data points.

1. **Route Optimization**

Develop algorithms that analyze real-time traffic data to suggest the most efficient routes to users. This can help drivers avoid congestion, reduce travel time, and save fuel.

1. **Navigation Assistance with Rerouting Options**

Offer users the ability to reroute their journey in case of unexpected obstacles or changes in traffic conditions. This could be triggered automatically based on real-time updates or initiated by the user manually.

1. **Speed Camera Warnings**

Integrate a feature that alerts users to the presence of speed cameras along their route. This can help drivers adhere to speed limits and avoid potential fines.

1. **Exposure Compensation Feature**

Implement a feature that automatically adjusts the brightness levels of the app interface or camera feed to compensate for changes in lighting conditions. This ensures that the display remains visible and readable at all times, even in varying light conditions.

1. lower priority Requirements

Addressing the lower priority requirements involves adding supplementary features that enhance user convenience and interaction with the application. Here's how these requirements can be implemented:

1. **Hands-Free Reporting Capabilities**:

Develop a hands-free reporting feature that allows users to report incidents, such as accidents or road hazards, using voice commands or gesture controls. This ensures that users can report incidents safely while driving without being distracted.

1. **Integration of Existing Traffic Monitoring Systems**

Collaborate with existing traffic monitoring systems, such as government agencies or private companies, to integrate their data into the application. This can provide users with access to additional real-time traffic information and improve the accuracy of updates.

1. **User Incident Reporting**

Enable users to report incidents they encounter on the road, such as accidents, potholes, or debris. This crowdsourced data can supplement existing traffic information and help other users navigate more safely.

1. **Easy Sharing of Captured Video Footage on YouTube**

Implement a feature that allows users to easily share captured video footage of their journey on YouTube or other social media platforms. This can be done through integration with the respective platforms' APIs and providing a streamlined sharing process within the application.

### Non-Functional Requirements

High Priority Requirements

Performance**:**

* **Responsiveness:**
  + The app should load quickly and respond to user actions instantly to avoid frustration while driving. This is critical for safety reasons.
* **Low Latency Notifications:**
  + Real-time updates on road conditions need to be delivered with minimal delay to ensure users have the latest information for safe navigation.
* **Battery Efficiency:**
  + Since the app might run in the background for extended periods, it should be optimized to minimize battery consumption on mobile devices.

2. Meduim Priority Reaquirement

## Usability:

* **Intuitive Interface:**
  + The user interface (UI) should be clean, uncluttered, and easy to navigate, even for users with limited experience with smartphones. Icons and buttons should be clear and easily recognizable.
* **Minimal Distraction:**

Visual information and notification alerts should be designed to be clear and informative at a glance, minimizing the time users need to look away from the road.

* **Voice Control Options:**
  + Consider incorporating voice commands for hands-free interaction with the app, allowing users to access information or report incidents without taking their eyes off the road.

## Data Management:

* **Road Sign Information Source:**
  + Decide whether road sign data will be pre-loaded on the app or updated dynamically. Pre-loaded data might be suitable for static signs, while real-time updates are crucial for temporary signs or variable message boards.
* **Real-time Data Acquisition:**
  + Explore options for obtaining real-time road state data. This could involve integrating with existing traffic management systems, leveraging data from connected vehicles, or incorporating user reports.
* **Data Update Mechanism:**
  + Establish a system for updating the app's database with new signs, changes in road conditions, or any other relevant information. This might involve automatic updates or user-triggered refresh options.

2. Low priority requirement

## Security:

* **User Privacy:**
  + The app should collect minimal user data, ideally just location for notification purposes. This data should be anonymized and used only for the stated purpose of improving road safety.
* **Data Security:**

The app should implement robust security measures to protect user data from unauthorized access, hacking attempts, or malware. Secure data storage and transmission protocols are essential.

### 4.2.3

The following are the technical requirements analyzed for the proper execution of this project:

**Frontend Framework for Mobile App:**

* **React Native**

A popular framework for building cross-platform mobile apps using JavaScript and React.

**Backend Frameworks for Mobile App:**

* **Express.js**

A fast and minimalist web application framework for Node.js.

* **Laravel**

A PHP framework for building scalable and feature-rich web applications.

**Database Frameworks for Mobile App**

* **Firebase**: A comprehensive backend platform by Google that offers real-time database, authentication, storage, and more.
* **MongoDB Stitch**: A serverless platform for MongoDB that provides data synchronization and backend services.
* **SQLite**: A lightweight and embedded database engine that is widely used in mobile app development
* .
  1. **Modelling and System Design**
* StarUML

This is an open source software used for building usecase diagrams and modelling of the system

# CHAPTER 5: SYSTEM MODELLING AND DESIGN

## 5.1. Introduction

Effective system modeling and design is a critical step in developing complex software applications. It involves creating visual representations and abstractions of the system's structure, behavior, and interactions to better understand and communicate the system requirements and architecture. The key objectives are to capture functional requirements through use case diagrams and user stories, model the system's static structure using class diagrams and ER diagrams, capture the dynamic behavior with activity diagrams and sequence diagrams, design the user interface, and define the high-level system architecture. By creating these visual models and designs, the development team can gain a shared understanding of the system, identify and resolve potential issues early, and communicate the system's structure and behavior to stakeholders more effectively. The subsequent sections of this document will cover the specific design artifacts for the road sign, road state, and notification application, including use case diagrams, class diagrams, and user interface designs [5].

* + 1. **Purpose**

The purpose of this document is to design and model the requirements for a road sign, road state, and notification app that provides users with real-time traffic updates, location-based services, and other advanced features.

* + 1. **Scope**

The scope of this project includes the development of a mobile application that offers the following key functionalities:

* Real-time traffic updates
* Location-based services
* Automatic start and stop
* Road sign recognition and display
* User feedback and collaboration
* Audio notifications
* Integration of current speed, GPS data, date, and time
* Navigation assistance
* Data visualization
* Route optimization
* Rerouting options
* Speed camera warnings
* Exposure compensation feature

## System Design

### 5.2.3. System Architecture

The system architecture for the road sign, road state, and notification app will be based on a client-server model, with the mobile app as the client and a backend server handling the data processing and management [6].

The key components of the system architecture include:

**Mobile App:** The user-facing application that provides the core functionalities and user interface.

**Backend Server:** The server-side component responsible for data collection, processing, and storage.

**Data Sources:** External data sources, such as traffic data providers, map APIs, and road sign databases, that supply the necessary information for the app.

**Communication Protocols:** The protocols used for data exchange between the mobile app, backend server, and external data sources, such as HTTP, WebSocket, or API calls.

The component of the Model-View Controller include:

**Model**

**Road State**: Represents the current state of the road, including construction zones, weather, closures, hazards, and checkpoints.

**Road Sign:** Represents a road sign with its type, location, and other attributes.

**User Location:** Represents the user's current location from the device's GPS.

**User Preferences**: Represents the user's preferences for notifications.

**Feedback Report:** Represents user feedback on the accuracy of updates and reports of road conditions.

**View**

**Road State View:** Displays the current road state using data visualization.

**Road Sign View:** Displays recognized road signs in the app or overlays them on the video.

**User Location View:** Displays the user's location on a map and provides navigation assistance.

**Notification View:** Handles the display of audio and visual notifications based on user preferences.

**Feedback View:** Allows users to provide feedback and report road conditions.

**Controller**

**Road State Controller:** Manages the fetching and updating of real-time road state data.

**Road Sign Controller:** Manages the detection and recognition of road signs.

**User Location Controller:** Manages the user's location data and provides location-based services.

**Notification Controller:** Manages the delivery of notifications based on user preferences.

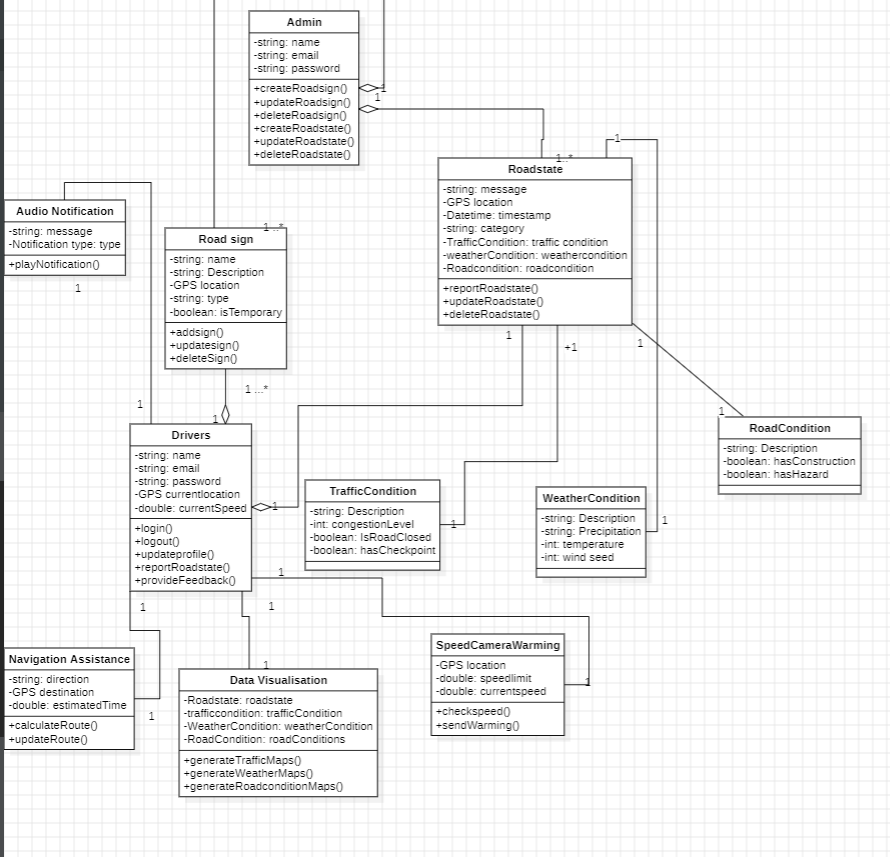
**Feedback Controller:** Manages the collection and processing of user feedback.

* + 1. **UML Diagrams**

UML are categorized into the following:

1. **Structural Diagrams**:
   * Class Diagrams
   * Context diagram
   * Deployment diagram
2. **Behavioral Diagrams**:
   * Activity Diagrams
   * Use Case Diagrams
   * Sequence Diagrams
     1. **Structural Diagram**

A structural diagram is a type of diagram that focuses on the static structure of a system, including its components, their properties, and the relationships between them.

1. **Class Diagram**

**Figure 5.1 : Class Diagram**

A class diagram is a type of structural diagram in software engineering that depicts the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

The following diagram works as follows:

**Primary Actors**

**Pedestrians**: Users of the app who are walking or using other modes of transportation besides driving.

**Drivers**: Users of the app who are driving vehicles.

**Travel Agencies**: Travel agencies that provide data on road conditions, traffic, and other relevant information.

**Secondary Actors**

**GPS System**: Provides location data to the app.

**Data Providers**: Third-party sources that supply real-time data on road conditions, weather, and other relevant information.

**App Administrator**: Responsible for maintaining the app, updating the database, and managing user feedback.

Use Case Relationships:

**Association**

Pedestrians and Drivers use the Road Sign, Road State, and Notification app.

State Agencies provide data to the app.

GPS System and Data Providers supply data to the app.

**Include**

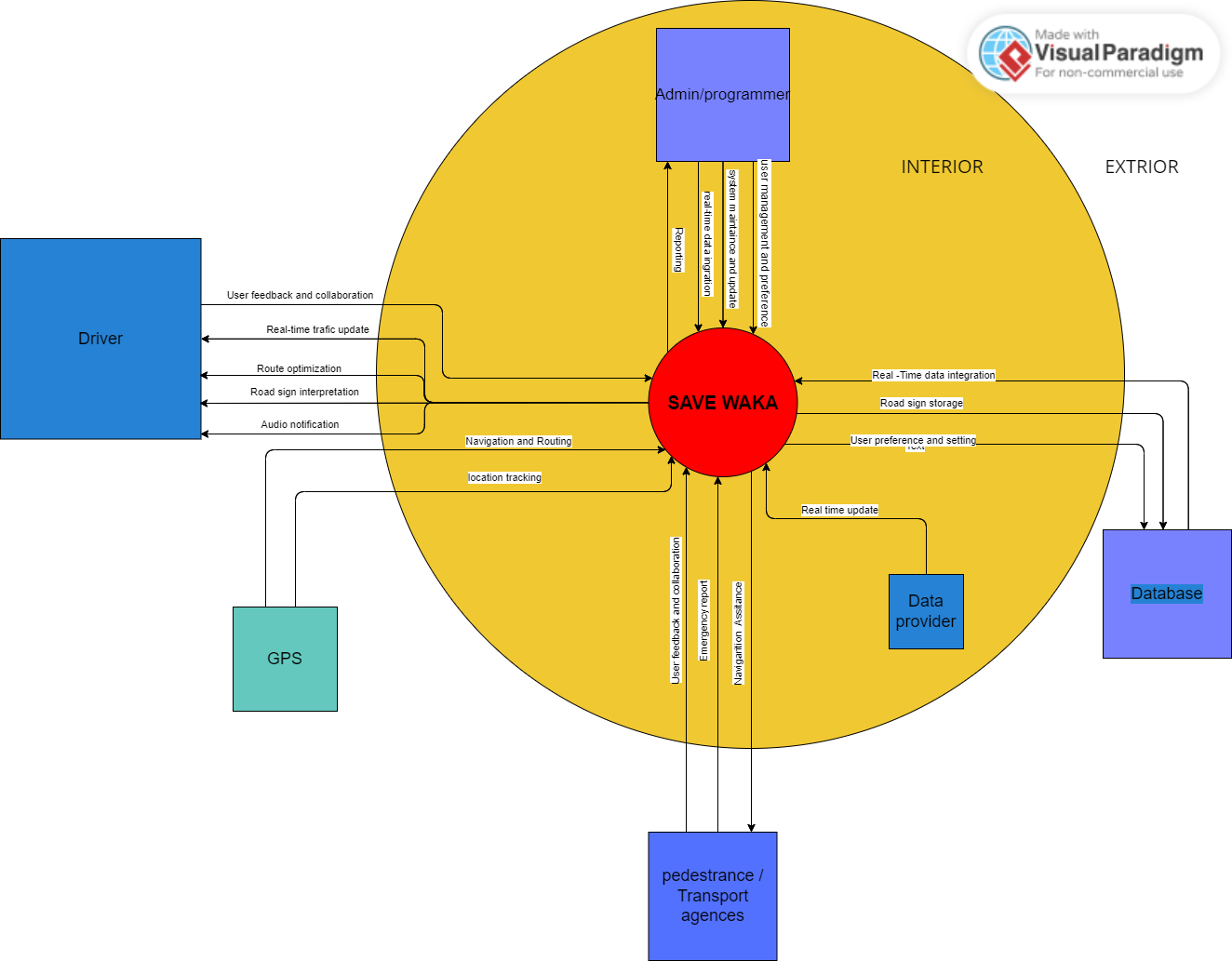
The app includes the use cases for Real-time Traffic Updates, Location-Based Services, and Audio Notifications.

The app includes the use cases for Navigation Assistance, Data Visualization, and Route Optimization.

**Generalization**

Pedestrians and Drivers are both specialized types of Users of the app.

Multiplicities between different classes.

1. **Context Diagram**

**Figure 5.2: Context Diagram**

**Interpretation:**

The context diagram for the "System - Safewaka" road state and road sign notification system provide an overview of the system's interactions with internal and external entities, showcasing the data flows and system boundaries.

**Internal Entities**:

Admin/Programmers: This entity is responsible for managing the system configuration, maintenance, and updates. They input configurations, updates, and system maintenance commands into the application server.

Data Provider: The data provider supplies real-time data on road conditions, traffic, and weather to the application server. They respond to requests for real-time data updates.

**External Entities:**

Drivers: Drivers interact with the mobile application, providing input preferences and receiving real-time road updates and notifications. The mobile application delivers road conditions, road sign information, and notifications to the drivers.

GPS: The GPS entity provides real-time location data to the mobile application, allowing it to deliver context-aware updates to drivers based on their current location.

Pedestrians/Transport Agencies: These entities access road sign and condition information through the mobile application, receiving updates and notifications. The application server delivers road sign and condition information to pedestrians and transport agencies.

Database: The database stores road sign information, user data, and system logs. The application server queries and updates road sign information in the database.

Data Flows:

Drivers to Mobile Application: Drivers input preferences and interact with the system through the mobile application, enabling them to receive personalized updates.

Mobile Application to Drivers: The mobile application provides real-time road conditions, road sign information, and notifications to the drivers.

Mobile Application to GPS: The mobile application receives real-time location data from the GPS entity, allowing it to deliver location-based updates to drivers.

GPS to Mobile Application: The GPS entity supplies real-time location data to the mobile application for context-aware updates.

Mobile Application to Application Server: The mobile application sends requests for data and updates to the application server based on the user's location and preferences.

Application Server to Mobile Application: The application server processes the requests and sends back the relevant data and notifications to the mobile application for display to the drivers.

Application Server to Data Provider: The application server requests real-time data on road conditions, traffic, and weather from the data provider.

Data Provider to Application Server: The data provider supplies real-time data updates in response to the application server's requests.

Application Server to Road Sign Database: The application server queries and updates road sign information stored in the road sign database.

Road Sign Database to Application Server: The road sign database provides road sign information to the application server for user notifications.

Application Server to Admin/Programmers: The application server sends logs, system status, and maintenance data to the admin/programmers for monitoring and maintenance purposes.

Admin/Programmers to Application Server: The admin/programmers input configurations, updates, and system maintenance commands into the application server for system management.

Application Server to Pedestrians/Transport Agencies: The application server delivers road sign and condition information to pedestrians and transport agencies.

Pedestrians/Transport Agencies to Mobile Application: Pedestrians and transport agencies access road sign and condition information through the mobile application's interface.

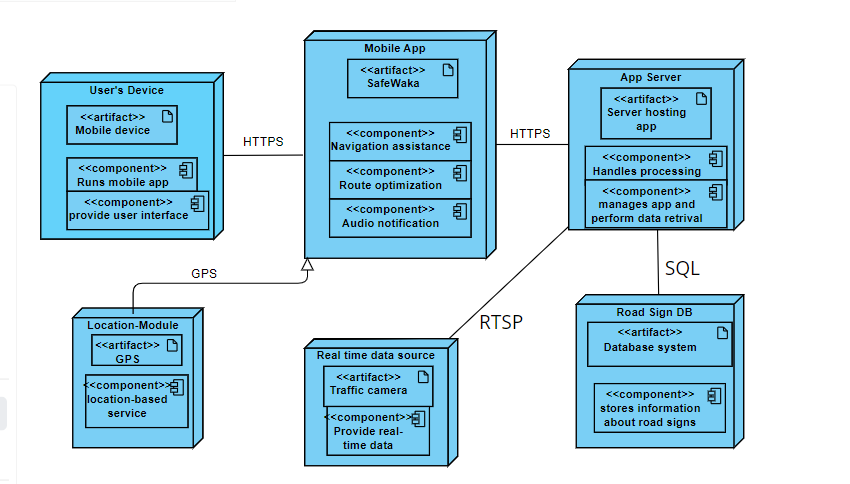
1. **Deployment Diagram**

**Deployment Diagram**

The deployment diagram is a type of UML (Unified Modeling Language) diagram that represents the physical deployment of software components on hardware nodes. It illustrates how software artifacts (such as modules, classes, files, etc.) are allocated to different nodes (e.g., servers, computers, devices) in a system's architecture.

Key elements in a deployment diagram include nodes (representing hardware devices), artifacts (representing software components or files), and deployment relationships (showing how artifacts are deployed on nodes). Additional notations can be used to indicate communication paths, deployment constraints, and other relevant details

For a Safewaka, a road state, road sign notification app,the deployment model will show how the user's device interacts with the mobile application, which communicates with the application server. The application server retrieves data from various sources, including traffic cameras, weather sensors, and a road sign database. Each component uses specific protocols to facilitate communication and provide the intended functionalities.



**Figure 5.3: Deployment Diagram**

**Interpretation**

* User's Device: Smartphones or tablets run the mobile application, providing an interface for users to interact with road sign information and receive real-time updates on road conditions.
* Mobile Application: Installed on the user's device, it allows access to road sign information, real-time updates, customizable notifications, and integration with navigation systems. It communicates with the application server for data retrieval and updates.
* Application Server: Hosts the mobile application, processing and managing logic. It retrieves data from traffic cameras, weather sensors, and a road sign database, processes user preferences, and responds to the mobile application.
* Location-based services: The location-based module leverages GPS and geofencing to provide real-time, context-aware notifications about road conditions and relevant road signs based on the user's current location. It integrates with navigation systems to enhance route guidance and safety.
* Real-time data sources: Provide real-time data on road conditions, traffic, accidents, and weather hazards. These devices transmit data via protocols like HTTP(S) or RTSP to the application server.
* Road Sign Database: Stores information about road signs on major highways, enabling instant access to their meanings and significance. It uses SQL or NoSQL protocols for data retrieval and storage.

**Importance of the deployment diagram**

The deployment diagram for a road state and road sign notification app offers several key benefits:

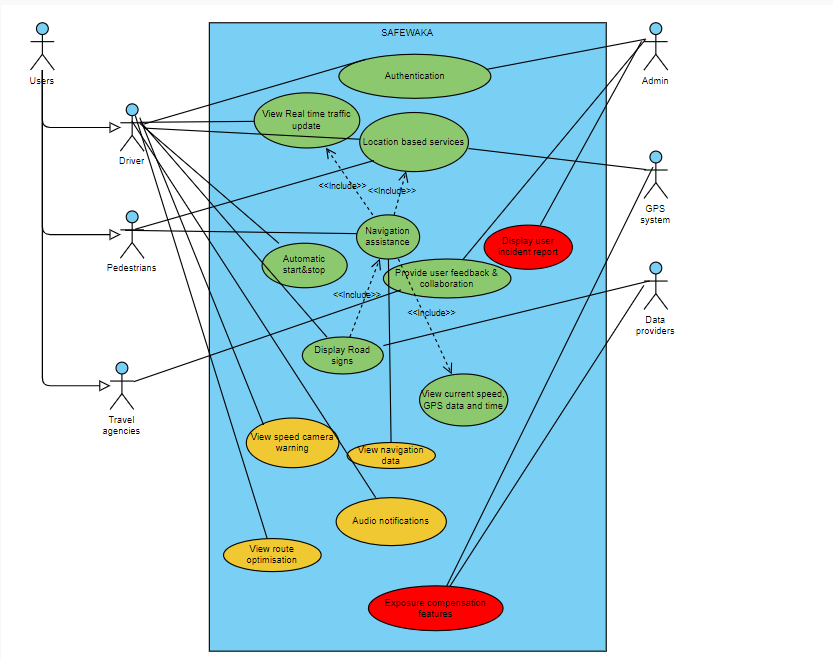
* System Visualization: It provides a visual representation of the system's architecture, helping stakeholders understand the deployment and interaction of components.
* Component Interaction: It illustrates information flow and communication between system components, crucial for accurate design and implementation.
* Scalability and Performance: It helps identify potential scalability and performance issues, allowing effective resource planning and optimization based on load and usage patterns.
* Integration Points: It highlights integration points with external systems, aiding in the identification of necessary protocols and interfaces for seamless data exchange.
* Fault Tolerance and Redundancy: It assists in designing fault-tolerant systems by visualizing redundancy and backup mechanisms, helping ensure system availability and reliability.
* Collaboration and Communication: It serves as a communication tool among project stakeholders, facilitating discussions and collaborative decision-making.

### Behavioral Diagram

The behavioral diagrams are used to model the dynamic aspects of a system, focusing on the behavior of the system and its components over time.

1. **Use Case**

A use case diagram for a road state and road sign notification app illustrates the interactions between actors (users or external systems) and the system itself, showcasing the different use cases or functionalities provided by the app

* Pedestrians and car drivers are the primary users of the app, interacting with various use cases such as viewing road signs, receiving updates, receive road conditions, customizing preferences, and getting directions.
* Admin has additional use cases related to managing the app, such as updating road data and integrating GPS data.
* Travel agencies can use the app to access road state and road sign information for their clients.
* GPS system provides location-based services and data integration to the app.
* ****Data providers supply real-time road data, which is utilized by the app to provide accurate updates and information.

**Figure 5.4: Use Case Diagram**

Green use cases represents functionalities with low priority.

Yellow use cases represents functionalities with medium priority.

Red use cases represents functionalities with low priority.

### Interaction Diagrams

These are diagrams that focus on modeling the interactions between objects or components within a system.

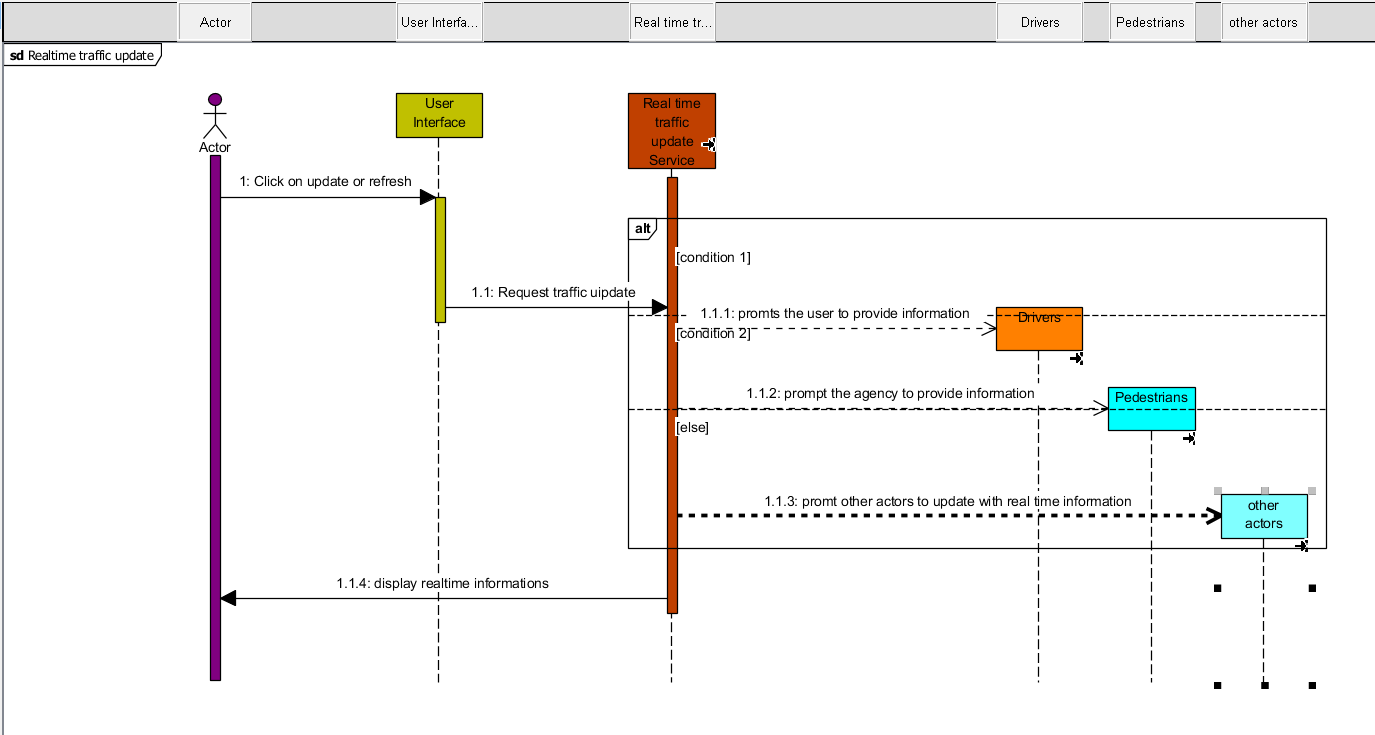
1. **Sequence Diagram**

**Defintion**

**Why use a sequence diagram**

**Various Sequence Diagrams in the system**

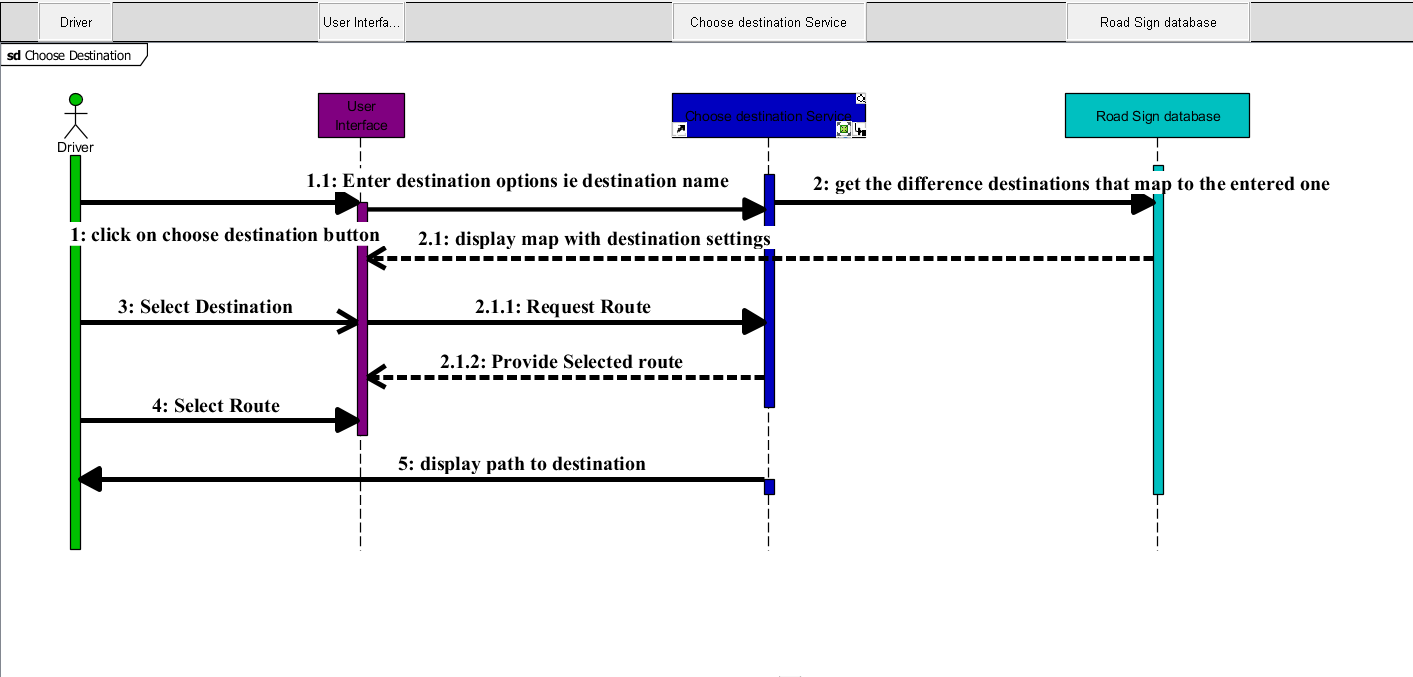
**Sequence Diagram 1: Get Real Time Traffic updates**

****

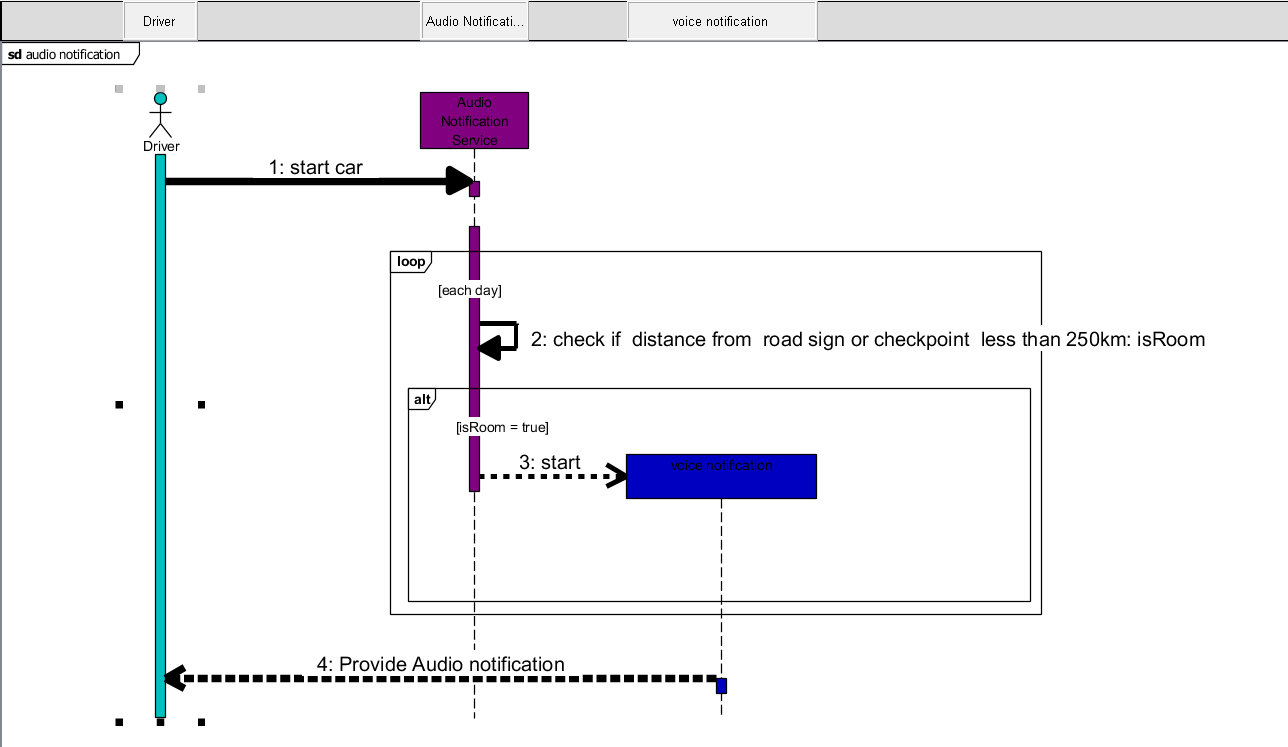
**Figure 5.5.1 : Sequence Diagram 1**

**Sequence Diagram 2: Choose Destination**

The sequence diagram below shows the interaction between the user and Safewaka in choosing the destination to reach his given destination

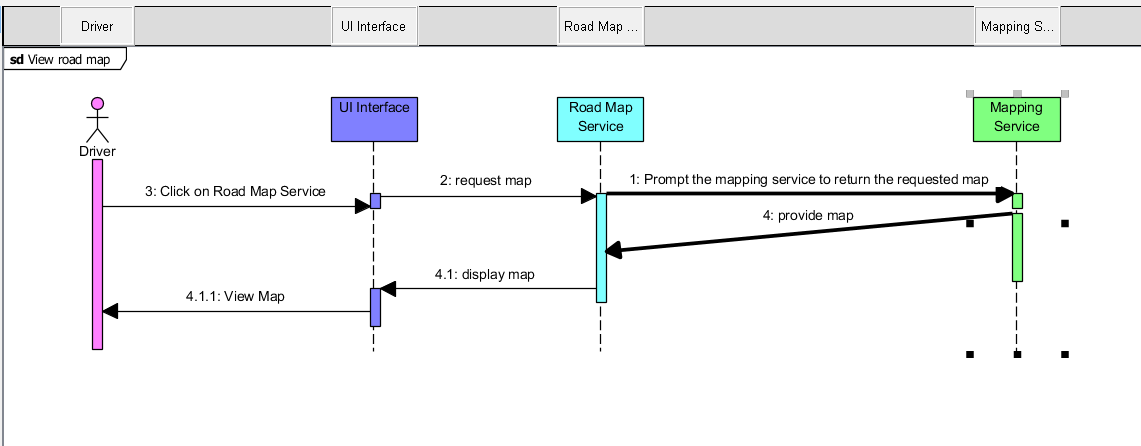


**Figure 5.5.2 : Sequence Diagram 2**

**Sequence Diagram 3: Audio Notifications**

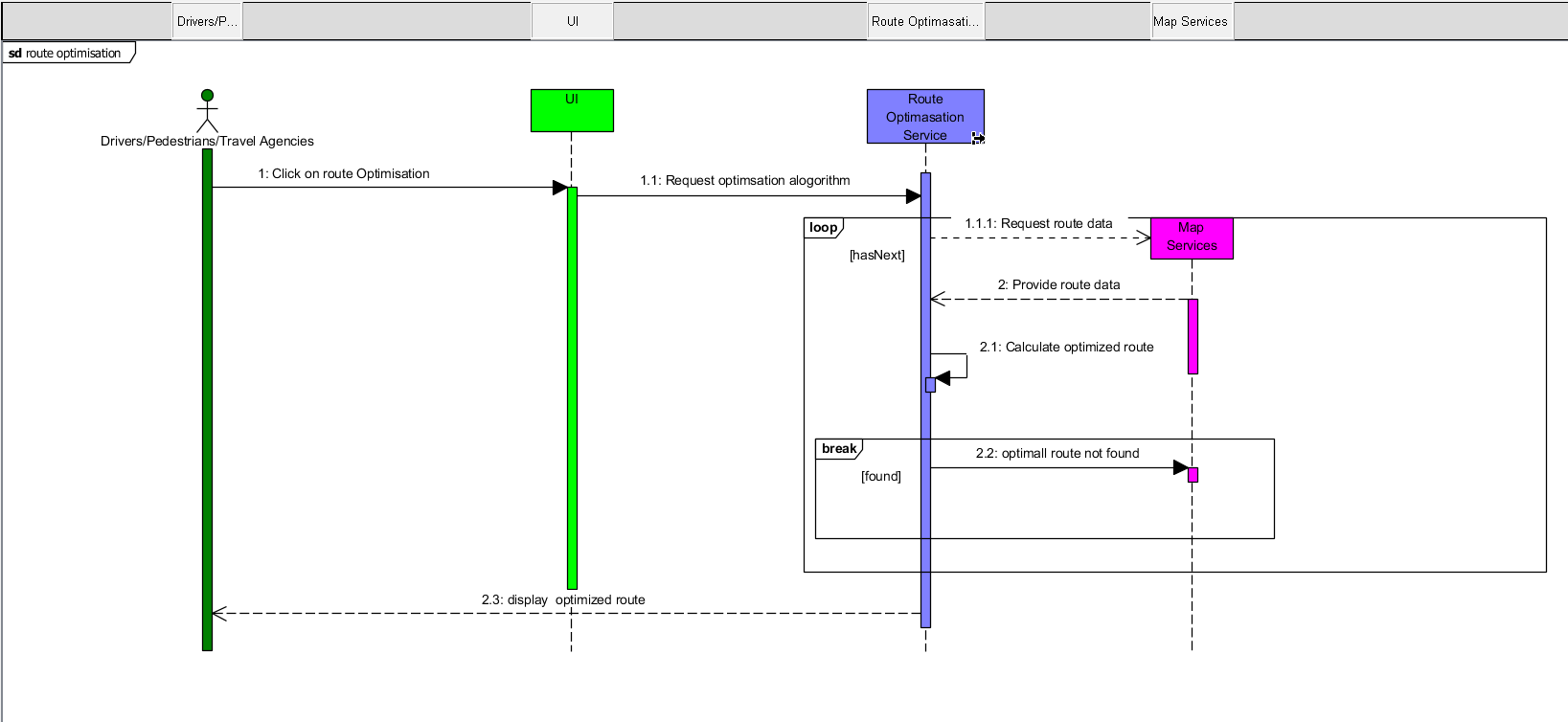
**Figure 5.5.3 : Sequence Diagram 3**

**Sequence Diagram 4: View Road map**

****

**Figure 5.5.4 : Sequence Diagram 4**

**Sequence Diagram5: Route Optimisation**

** Figure 5.5.5 : Sequence Diagram**

# CHAPTER 6: UI DESIGN AND IMPLEMENTATION

## 6.1. Introduction

The purpose of this UI Design and Implementation Report is to document the design process, technical implementation, and deployment of the new user interface for the SAFE WAKA web application. This report serves as a comprehensive record of the project's progress, design decisions, and the final delivered solution. It aims to provide a detailed account of how the UI was conceived, developed, and ultimately deployed to meet the evolving needs of the application's target users.

* 1. **Key Use Cases:**

1. **Identification and Interpretation of Road Signs**:

* Users can use the application to quickly identify and understand the meaning of various road signs encountered during their journeys.
* This can help drivers stay informed about traffic regulations, road hazards, and other important information, improving their overall driving experience and compliance with traffic laws.

1. **Real-Time Road Condition Updates**:

* Users can access real-time information about the current state of the roads, including traffic congestion, accidents, weather-related hazards, and road closures.
* This information can assist drivers in making informed decisions about their routes, allowing them to avoid delays, plan alternative paths, and reach their destinations more efficiently.

1. **Personalized Notification and Alerts**:

* Users can customize the application to receive personalized notifications and alerts based on their preferences, such as specific road signs, routes, or geographic areas of interest.
* These alerts can help drivers stay informed about important developments and make timely adjustments to their driving plans, enhancing overall safety and reducing the risk of incidents.

**Integration with Navigation Systems**:

The application's integration with popular navigation systems can provide users with a seamless experience, allowing them to access road sign information and road condition updates directly within their preferred navigation app.

This integration can enhance the overall driving experience, providing users with a comprehensive set of tools to navigate the road network effectively.

* 1. **Design Goals and Principles**
     1. **Design Goals**

1. **Ease of Use**: We designed the user interface to be intuitive and user-friendly by employing a minimalist approach, focusing on essential features and information, and minimizing the cognitive load on drivers and commuters.
2. **Accessibility**: We designed the application to be accessible to a wide range of users by adhering to accessibility guidelines, ensuring the information and functionality are easily understandable and usable by all users.
3. **Brand Consistency**: We designed the application's visual style and branding to align with the identity of the organization responsible for its development, to build trust and recognition among the target audience.
4. **Information Accuracy and Reliability**: We designed robust data validation processes and integrated reliable data sources to ensure the information about road signs and real-time road conditions is accurate and up-to-date, allowing users to trust the data presented.
5. **Contextual Relevance**: We designed the application to offer personalized information and alerts based on the user's current location, driving route, and preferences, ensuring they receive the most valuable and actionable insights.
   * 1. **Design Principles**
6. **Simplicity**: We designed the user interface with a minimalist approach, focusing on essential features and information, and maintaining a clean, organized, and uncluttered layout.
7. **Consistency**: We designed the visual and interactive design to be consistent across all screens and functionalities, to help users navigate the application more efficiently and reduce the learning curve.
8. **Intuitive Navigation**: We designed the navigation structure to be logical and intuitive, using clear and descriptive labels, as well as intuitive gestures and interactions, to enhance the user's ability to find and access the information they need.
9. **Responsive Design**: We designed the application to be responsive, adapting to different screen sizes and device characteristics, to ensure a seamless user experience across a range of mobile devices.
10. **Contextual Awareness**: We designed the application to leverage the user's location, driving context, and preferences to provide personalized information and alerts, enhancing the relevance and usefulness of the application.
11. **Prioritization of Critical Information**: We designed the interface to prioritize the display of critical information, such as urgent road condition updates or safety-related road sign details, to ensure users can quickly identify and act on the most important data.

By taking these design approaches, we aimed to create a user-centric and effective solution that enhances the driving experience and promotes road safety for the users of the Road Sign and Road State Mobile Notification Application.

* 1. **User Research and Analysis**

1. **Summary of User Research Findings**: Through in-depth user interviews and usability testing, we gathered valuable insights about the target users. We learned that they prioritize real-time and accurate road condition information, personalized alerts and notifications based on their commute routes, and an intuitive and distraction-free interface.
2. **Insights Gathered About User Needs, Pain Points, and Behaviors**: Our research revealed that users often struggle to find relevant road information, face frustration with outdated or inaccurate data, and find it challenging to stay informed while driving safely. These insights helped us identify the key user needs and pain points that the application should address.

Design Ideation and Concepts:

1. **Brainstorming and Ideation Process**: We conducted a collaborative brainstorming session with the project team, including designers, developers, and subject matter experts. This process allowed us to generate a wide range of initial design concepts and explore various approaches to meeting the identified user needs and addressing their pain points.
2. **Exploration of Initial Design Concepts and Alternatives**: Based on the insights from user research, we explored several initial design concepts, each with a unique focus or approach. This included exploring different information hierarchies, notification systems, and interaction models to identify the most promising solutions that would resonate with the target users and align with the project's objectives.

The findings from the user research and the exploration of initial design concepts provided a solid foundation for the design and development of the Road Sign and Road State Mobile Notification Application, ensuring it would meet the needs and expectations of the end-users.

## Wireframes and Mockups

**Key Wireframe Sketches and Interactive Prototypes**: As part of the design process, we created detailed wireframe sketches and interactive prototypes to explore the user interface design. These wireframes and prototypes helped us visualize the information hierarchy, navigation flows, and key functionalities, ensuring they aligned with the user needs and project requirements.

**Visual Mockups Showcasing the Final UI Design**: Building upon the wireframes and prototypes, we developed high-fidelity visual mockups that showcased the final user interface design. These mockups incorporated the branding, typography, color palettes, and visual elements that would be used in the actual application, enabling stakeholders and users to get a clear understanding of the application's look and feel



Figure 6.1: low fidelity prototype

## Design Specifications

### Typography

* The primary font family used is **poppins**, a clean and modern sans-serif font selected for its excellent readability on mobile devices.
* Headings are set in **Roboto**, with sizes ranging from 24-32px and a medium weight.
* Body text is displayed in **Roboto**, with sizes ranging from 16-30px and a regular weight.
* Captions and labels utilize Roboto, set at 14px with a regular weight.

### Color Palette

* + The primary color is **#64C178**, a **bright green** chosen to create a sense of trust and reliability for the road condition information.
  + The secondary color is #**64C178**, a used as **Gray** the main background color to provide a clean and uncluttered interface.
  + An accent color of #**FFFFF**, a used **white**, is employed sparingly to draw attention to critical alerts and warnings.
  + The color palette was selected to balance visibility, accessibility, and brand alignment.

### Iconography and Imagery

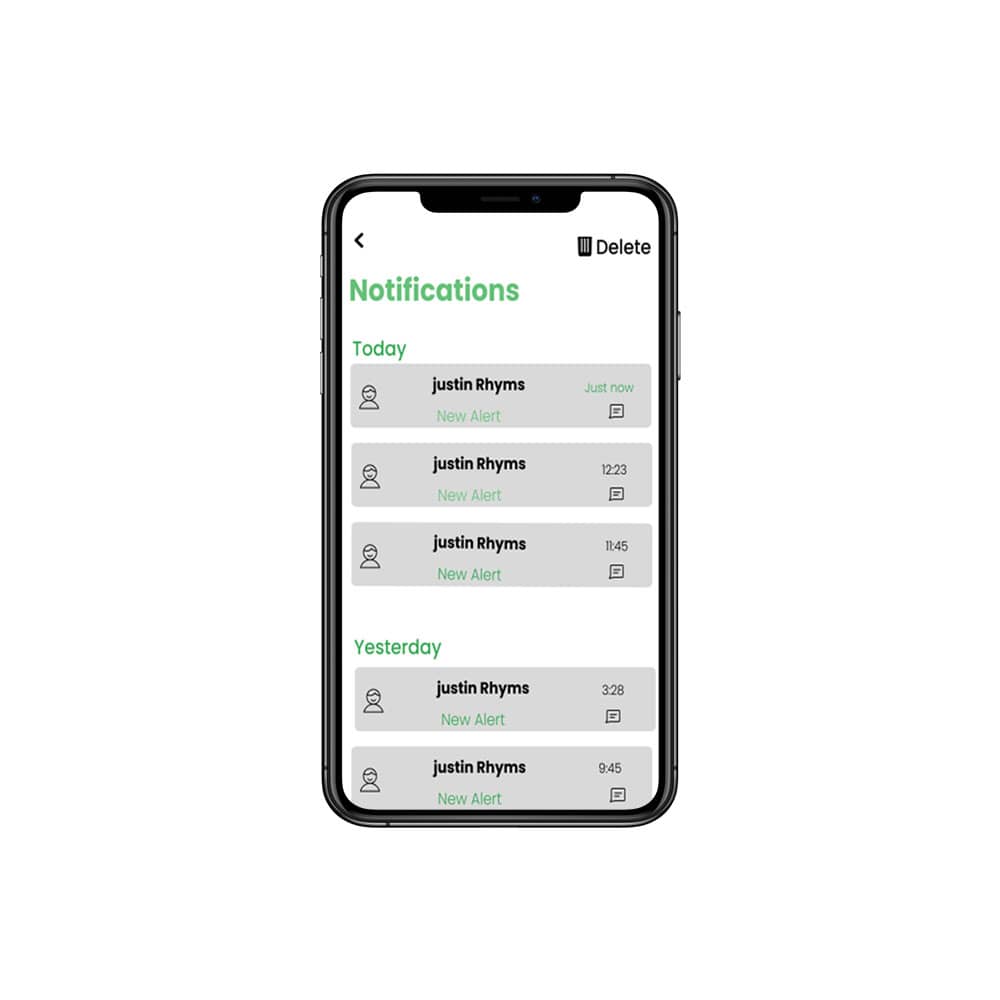
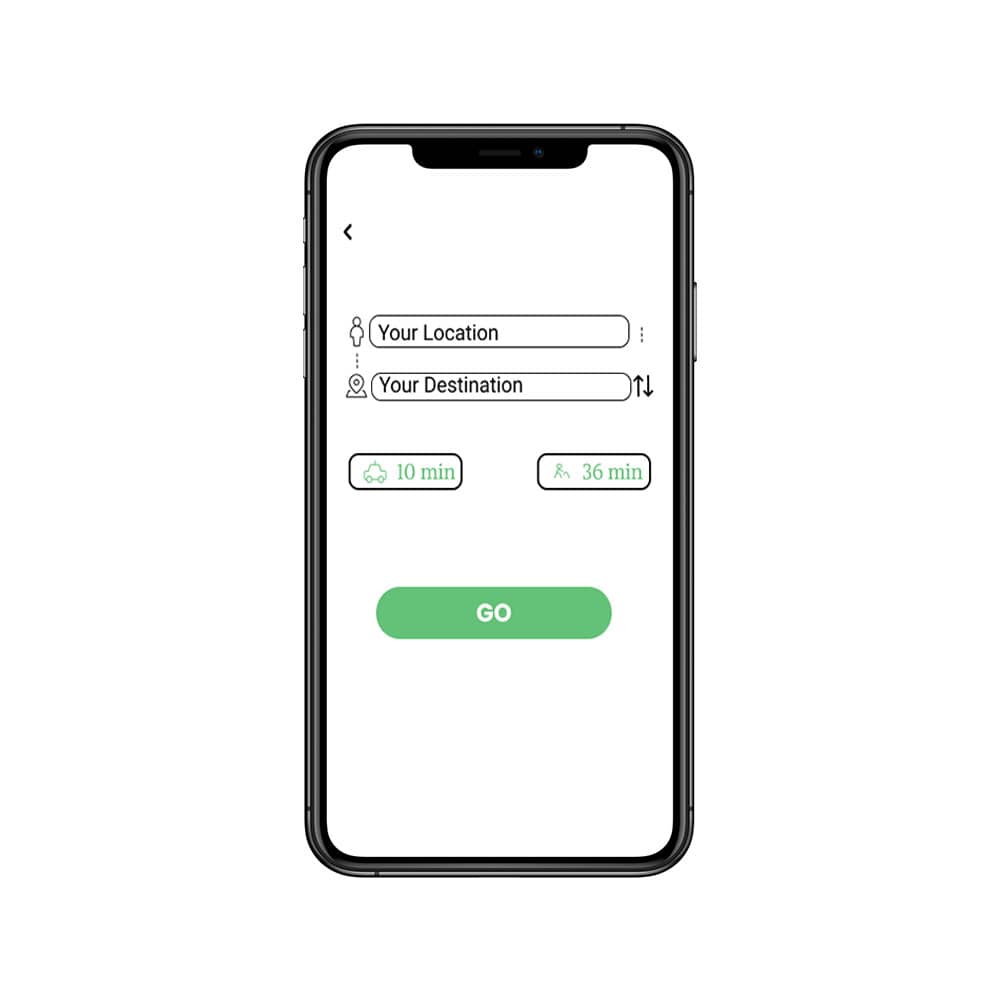
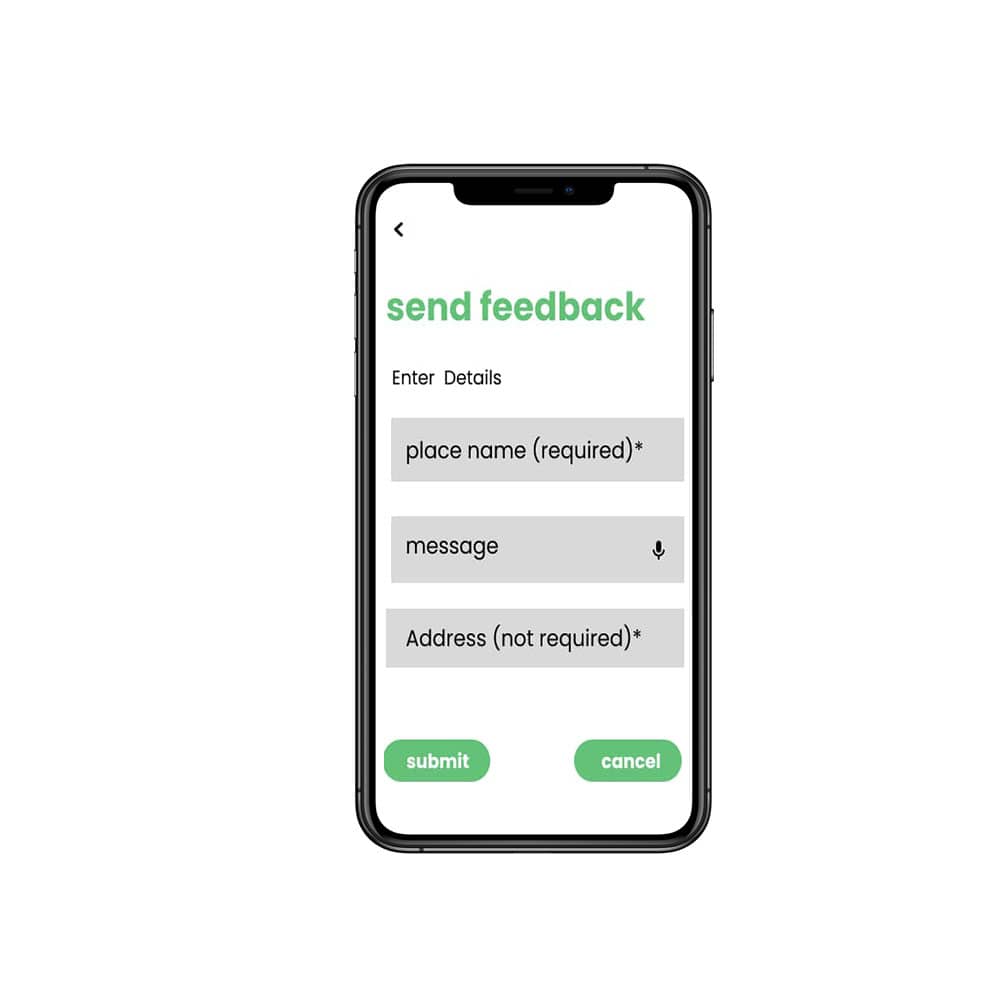
* + The icons used are a set of custom-designed elements, following a consistent style and weight, and adhering to Material Design guidelines for mobile applications.
  + Minimal photography is used, with a focus on simple illustrations and data visualizations to convey road conditions and traffic information.
  + The iconography and imagery were designed to be intuitive, easily recognizable, and supportive of the overall user experience.

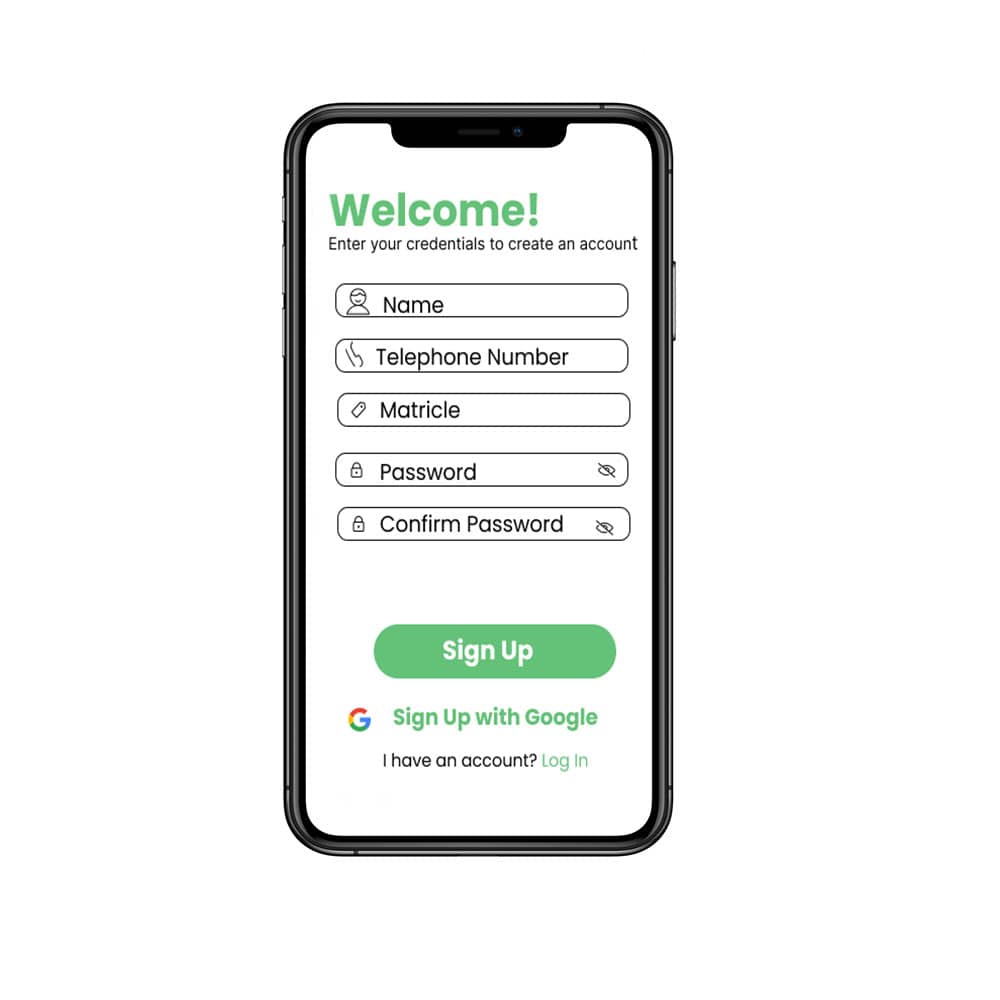
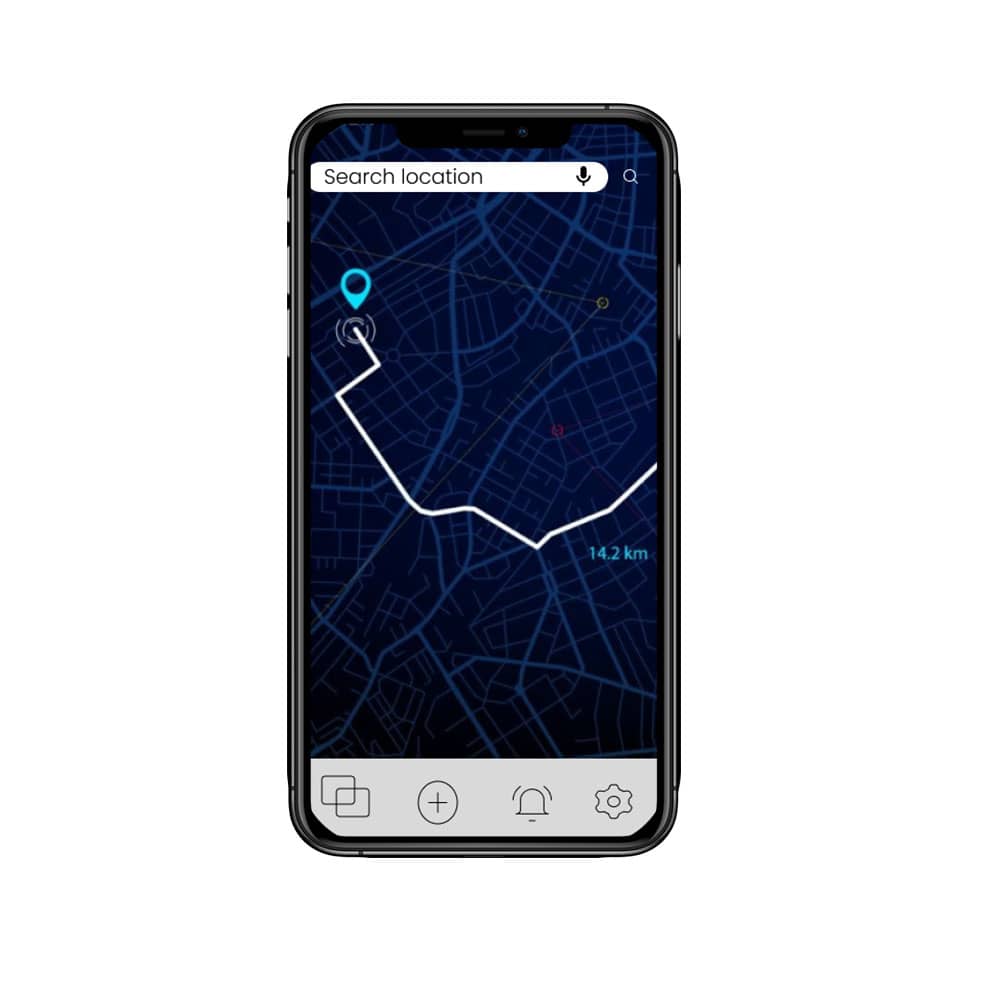
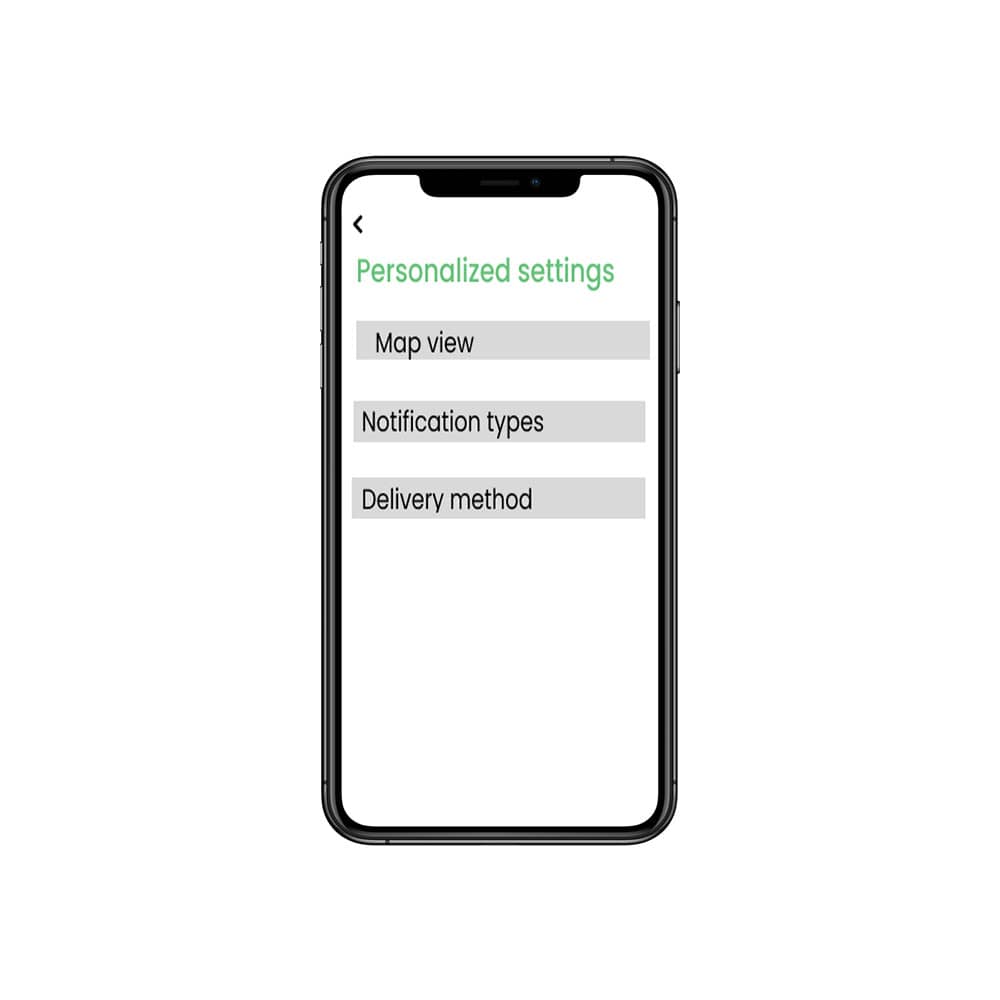
### UI Components and Patterns

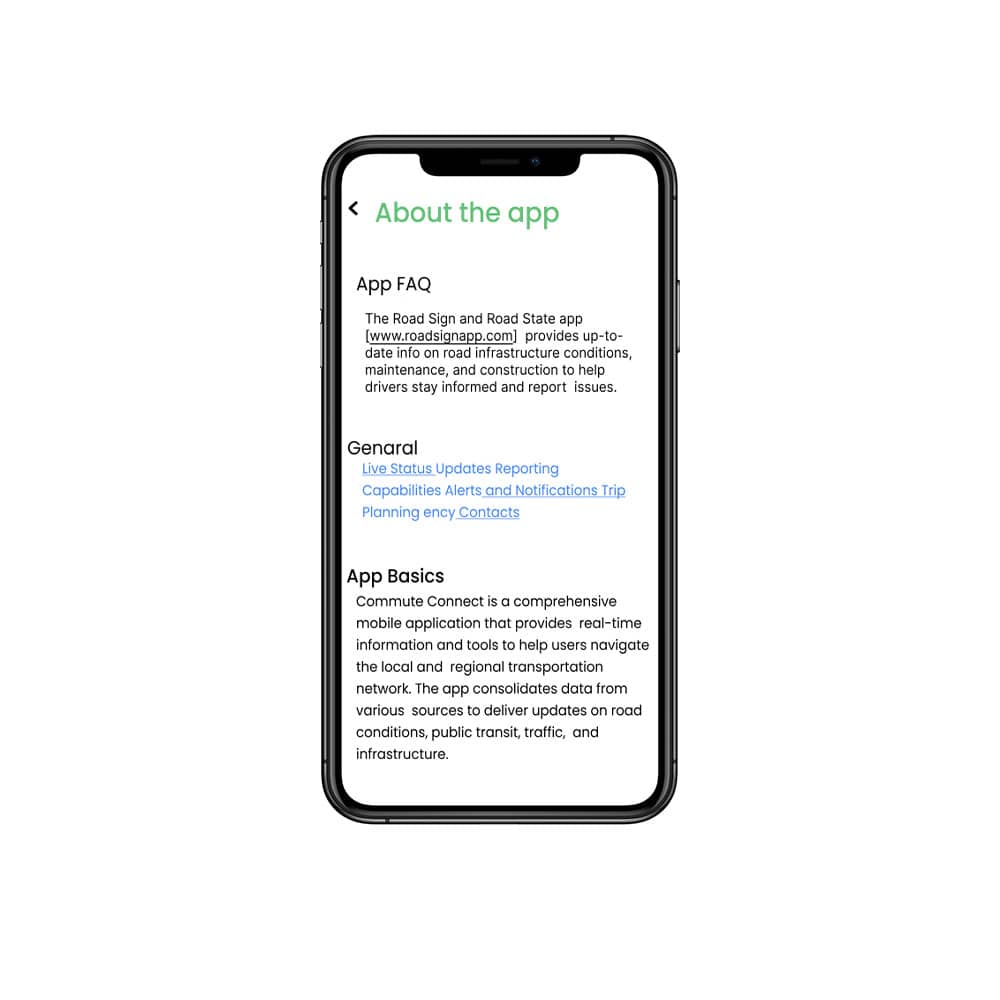
* + Navigation is implemented as a bottom navigation bar with large, tap-friendly icons for the main sections (Home, Alerts, Settings).
  + The layout utilizes a modular card-based design to display road condition updates, incident reports, and personalized notifications.
  + Prominent, interruptive alerts are used for critical information, with a collapsible notification center for less urgent updates.
  + The UI components and patterns were selected to align with established mobile design best practices and provide a familiar, intuitive experience for the users.
    1. **Accessibility and Usability**

1. **Adherence to Accessibility Guidelines**
   * The design and implementation of the application adheres to the Web Content Accessibility Guidelines (WCAG) 2.1 at the AA level, ensuring that the interface is accessible to users with various disabilities.
   * Key accessibility features include:
     + Alternative text descriptions for all images and icons to support screen readers.
     + Keyboard navigation and focus management to enable users to navigate the interface without a mouse.
     + High color contrast ratios between text and background colors to accommodate users with visual impairments.
     + Adjustable text sizes and the ability to increase or decrease font sizes to improve readability.
     + Optimized touch targets and spacing between interactive elements to support users with motor impairments.
2. **Usability Testing Findings and Improvements**
   * Extensive usability testing was conducted with a diverse group of participants, including users with different levels of technical proficiency and accessibility needs.
   * The key findings from the usability testing include:
     + Users were able to quickly and easily locate the most critical information, such as real-time road conditions and incident alerts.
     + The navigation flow and information architecture were intuitive, allowing users to find the desired content with minimal effort.
     + Participants with visual impairments provided positive feedback on the readability and contrast of the interface elements.
     + Some users requested additional customization options, such as the ability to personalize the app's color scheme and notification preferences.
   * Based on the usability testing findings, the following improvements were implemented:
     + Refined the information hierarchy and layout to surface the most important data more prominently.
     + Introduced user-configurable settings for adjusting text size, color themes, and notification preferences.
     + Conducted an additional round of accessibility testing to ensure the interface met the highest standards for users with diverse needs.
   1. **Implementation of user interface**

**6.8.1. Technical Requirements and Constraints**

* + The application was designed to be cross-platform, supporting both iOS and Android operating systems to reach the widest possible user base.
  + Due to the real-time nature of the road condition data, the application was required to have low latency and high responsiveness to provide users with up-to-date information.
  + To ensure scalability and handle potential spikes in user traffic, the backend infrastructure was designed to be highly available and capable of handling large volumes of data.
  + Security and data privacy were critical concerns, as the application would be processing sensitive information about users' locations and driving patterns.
    1. **Development Approach**
  + The project utilized an Agile development methodology, with two-week sprints and regular stakeholder feedback sessions.
  + This approach allowed the team to quickly iterate on the design and implementation, respond to changing requirements, and deliver incremental updates to the users.
  + Cross-functional teams, consisting of designers, developers, and subject matter experts, worked collaboratively throughout the development process to ensure seamless integration and alignment with the overall project goals.
    1. **Coding Standards and Best Practices**
  + The codebase adhered to industry-standard coding conventions and best practices, such as the use of consistent naming conventions, modular architecture, and thorough documentation.
  + Code reviews were conducted regularly to maintain code quality, identify potential issues, and ensure adherence to established coding standards.
  + Automated testing suites, including unit tests, integration tests, and end-to-end tests, were implemented to ensure the reliability and maintainability of the codebase.
  1. **High Fidelity prototype**





­­­­­­­figure 6.2:high fidelity prototype

* 1. **Implementation of user Logo**

**Align with Brand Identity**

The logo should reflect the application's purpose, target audience, and core values.

**Strive for Simplicity and Memorability**

Keep the design clean, easy to recognize, and visually impactful.

**Ensure Versatility and Scalability**

The logo should work well across various contexts and sizes.

**Incorporate Symbolic Representation**

If possible, include visual elements that represent the application's functionality or values.

**Thoughtfully Select the Color Palette**

Choose a color scheme that is visually appealing and aligned with the brand.

**Prioritize Originality and Uniqueness**

Aim to create a logo that sets your application apart from competitors.

**Evoke Positive Emotional Connections**

Design the logo to foster trust, excitement, or a sense of belonging in users.



Figure

# CHAPTER 7: DESIGN AND IMPLEMENTATION OF DATABASE

* 1. Introduction

In the modern era of smart transportation, the need for comprehensive road information and real-time updates has become increasingly vital for drivers. The road state and road sign notification app aim to address this need by providing a user-friendly mobile application that offers intuitive access to road sign information and up-to-date insights on road conditions. This report delves into the design and implementation of the database that powers the core functionalities of the app.

* 1. Database Schema  
     The database schema for the road state and road sign notification app consists of the various tables in our database. Below are key tables in our database:

1. **Road State**

This table stores information about the current state of the road.

**Columns**

*stateID*: Unique identifier for the road state. Acts as the PK(primary key) of this table

*eventype*:Describes the current event of road at a given time.

*severity*: Describes the urgency of the road state event.

*location*: Gives the exact location of the road state event.

*timestamp*: Gives the time at which the driver encounters the road state.

1. **Navigation**

This table provides adequate navigation for the car drivers.

**Columns**

*navID*: Unique identifier for the navigation template. Acts as the PK(primary key) of this table

*appName*: Gives the name of the navigation provider.

*version*: Gives the version of the navigation provider.

*userID:* Describes the ID of the user.

*original\_location*: Gives the initial location of the driver before embarking on a journey.

*destination\_location*: Gives the destination location of the driver.

1. **RoadSign**

This table contains detailed information about the road signs located along the road network, including the sign type, text or symbols, GPS coordinates, and any additional metadata.

**Columns**

*signID*: Unique identifier for the sign template. Acts as the PK(primary key) of this table.

*sign\_type*: Describes the road sign the driver encounters on the road.

*interpretation*: Gives the meaning of the road signs.

*location*: Gives the location of the road sign.

*imageURl*: Provides the link to the images of the various road signs.

1. **User**

This table stores the individual preferences and notification settings for each user of the app(drivers) allowing them to customize the types of alerts and information they receive.

***Columns***

*userID*: Unique identifier for the user template. Acts as the PK(primary key) of this table.

*username*: Gives the name of the driver.

*setting*: Describes the personalized settings of the driver.

*location*: Gives the current location of the driver.

1. **Notifications**

This table logs all the notifications sent to users, including the type of notification, the recipient, the timestamp, and any relevant details.

**Columns**

*notifyID*: Unique identifier for the notification template. Acts as the PK(primary key) of this table.

*userID*: Gives the ID of the driver.

*stateID*:Gives the ID of the road state.

*priority*: Describes the priority of the notification.

*signID*: Gives the ID of the road sign.

1. **Developer**This table stores the feedbacks from the users regarding the app.

**Columns**

*developerID*: Unique identifier for the developer template. Acts as the PK(primary key) of this table.

*name*: Gives the name of the admin

*email*: Specifies the email of the admin that receives feedback about the app.

1. **Route**

This table is used to give the drivers the best route for a journey using the initial and destination locations of the driver.

**Columns**

*routeID*: Unique identifier for the route template. Acts as the PK(primary key) of this table.

*original\_location*: Gives the initial location of the driver before embarking on a journey.

*destination\_location*: Gives the destination location of the driver.

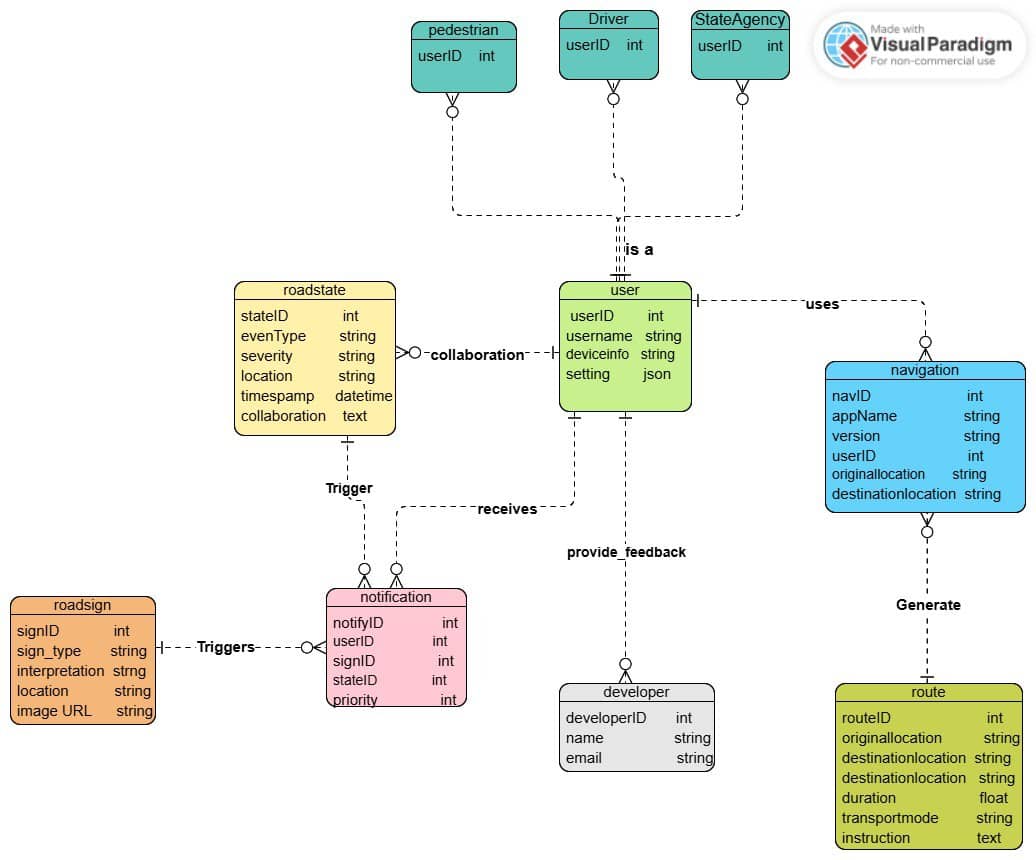
*duration*: Gives the duration of the journey using the optimized route.

*instruction*: Describes the directives given to the driver using the best route.

## Implementation

**ER Diagram**

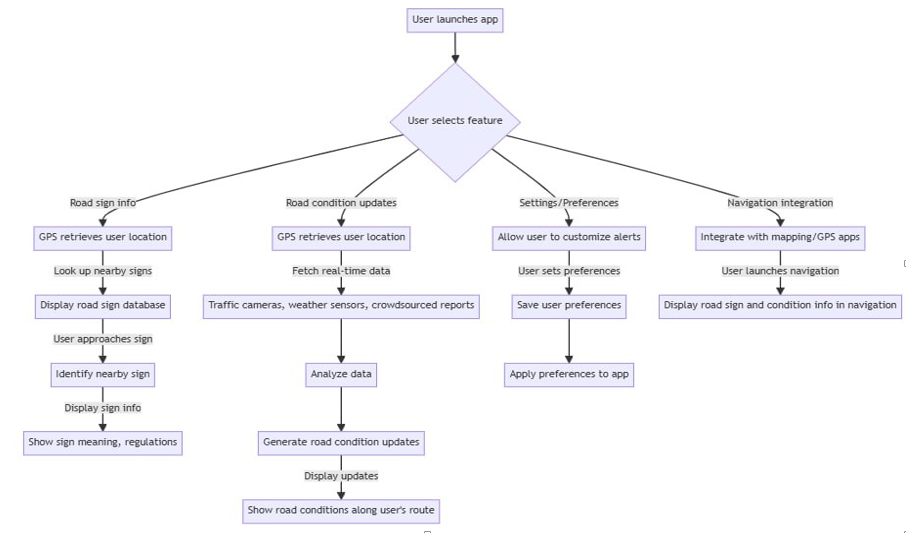
This diagram depicts the relationships between entities (tables and objects) in our database.



*Figure 7.1: ER diagram of the database of SafeWaka*

## 

## 

**DATA FLOW DIAGRAM**

**Figure 7.2 :Data Flow Diagram**

Backend Data Flow

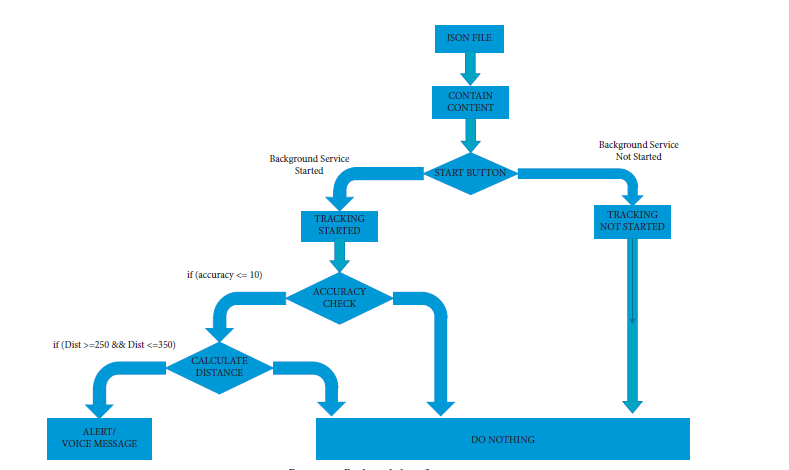
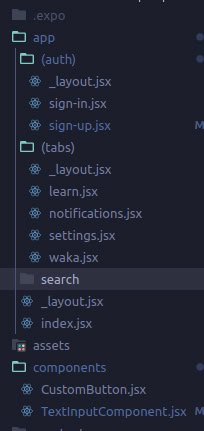
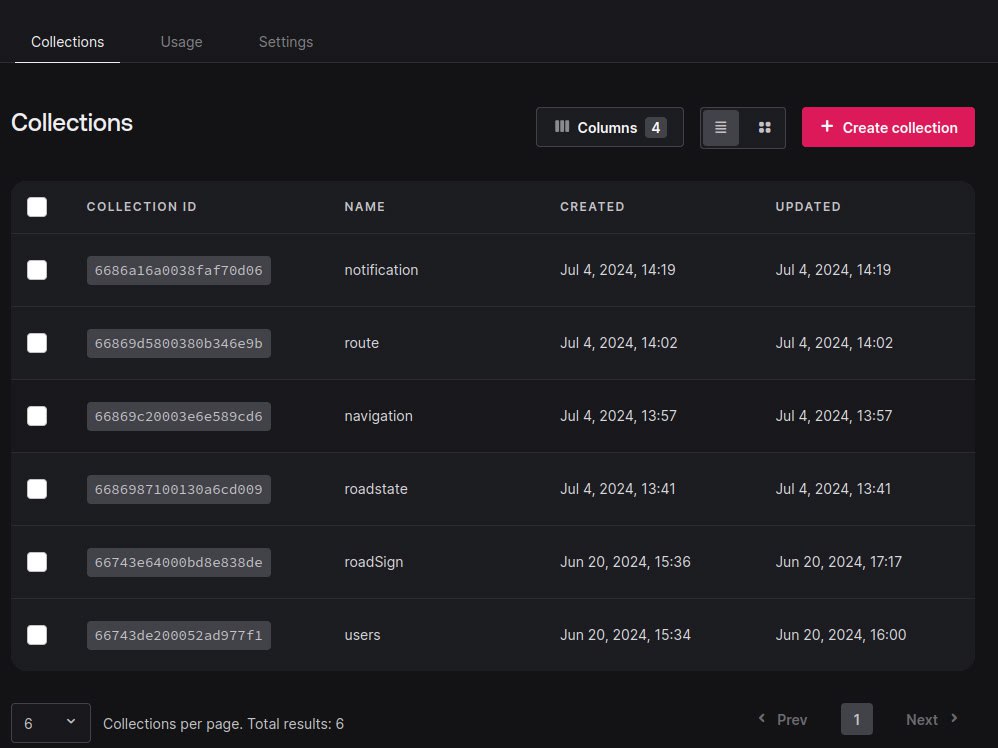
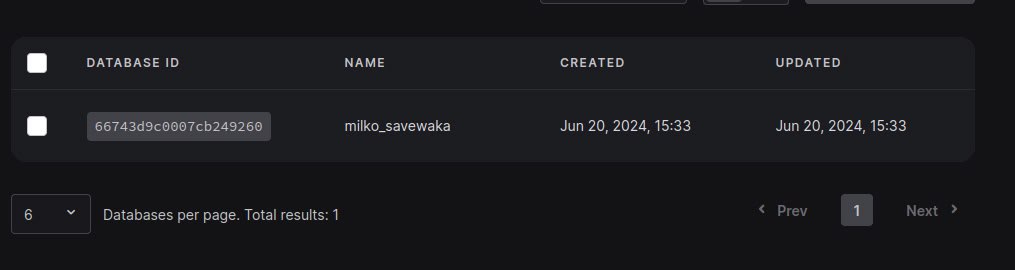


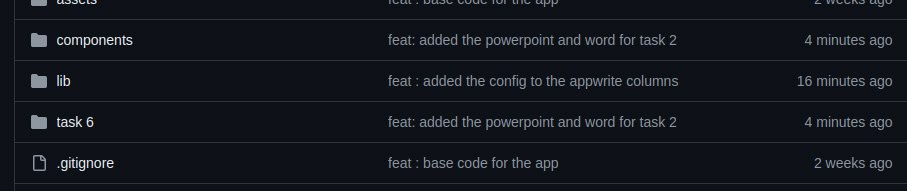
Figure 7.3:Backend Data Flow Diagram

* 1. Implementation of database on Appwrite
     1. File structure of the of the frontend





* + 1. 



Database Security and Reliability  
The database powering the road state and road sign notification app is designed with robust security and reliability measures, including:

* Secure authentication and authorization mechanisms to control access to the database
* Encryption of data at rest and in transit to protect sensitive information
* Automated backups and disaster recovery capabilities to ensure data integrity and availability
* Scalable infrastructure and load-balancing to handle increased user traffic and data growth
* Comprehensive logging and auditing to monitor database activities and detect any suspicious events

These security and reliability measures are carefully implemented to safeguard the app's data and provide a reliable service to users.

# CONCLUSION

## Challenges Faced

q

## Future Recommendations

# REFERENCES

1. [1] How To Design a MySQL Database for your Basic Notification System? - DEV Community. (n.d.). Retrieved from <https://dev.to/codehakase/how-to-design-a-mysql-database-for-your-basic-notification-system-4aic>
2. [2] Designing a notification system | Notification database design | Medium. (n.d.). Retrieved from <https://medium.com/@amrit.kumar/designing-a-notification-system-notification-database-design-8b33b8169027>
3. [3] Designing a Notification System in a Microservices Architecture. (2021, June 23). Retrieved from <https://www.freecodecamp.org/news/designing-a-notification-system-in-a-microservices-architecture/>
4. [4] Notification System Design: Building a Scalable Notification Infrastructure. (2019, August 20). Retrieved from <https://www.callstack.com/blog/notification-system-design-building-a-scalable-notification-infrastructure>
5. [5] Database Design for a Notification System. (n.d.). Retrieved from <https://www.codeproject.com/Articles/5291376/Database-Design-for-a-Notification-System>
6. [6] Building a Scalable Notification System. (2018, April 23). Retrieved from <https://blog.bitsrc.io/building-a-scalable-notification-system-7e42c6b65e4d>
7. [7] Designing a Robust Notification System. (2020, December 8). Retrieved from <https://engineering.shopify.com/blogs/engineering/designing-a-robust-notification-system>