

UNIVERSITY OF BUEA



REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

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

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGINEERING**

## **Passenger Positioning System (DigiTekisi)**

*A dissertation submitted to the Department of Computer Engineering, Faculty of Engineering and Technology, University of Buea, in Partial Fulfilment of the Requirements for the Award of Bachelor of Engineering (B.Eng.) Degree in Computer Engineering.*

**By:**

NAME	MATRICULE NO
YIMNAI NERUS ZAUMU	FE20A123
TABOT CHARLES BESSONG	FE20A106
BEBONGNCHU YANNICK NKWETTA	FE20A022
BALEMBA JUNIOR BALEMBA	FE20A021
TANDONGFOR SHALOM CHANGEH	FE20A111

**Option:** Network & Software Engineering

**Supervisor**

Dr. NKEMENI VALERY  
University of Buea

**2022/2023 Academic Year**

## **Passenger Positioning System (DigiTekisi)**

NAME	MATRICULE NO
YIMNAI NERUS ZAUMU	FE20A123
TABOT CHARLES BESSONG	FE20A106
BEBONGNCHU YANNICK NKWETTA	FE20A022
BALEMBA JUNIOR BALEMBA	FE20A021
TANDONGFOR SHALOM CHANGEH	FE20A111

**2022/2023 Academic Year**

***Dissertation submitted in partial fulfilment of the Requirements for the award of Bachelor of Engineering (B.Eng.) Degree in Computer Engineering.***

**Department of Computer Engineering**

**Faculty of Engineering and Technology**

**University of Buea**

### **Certification of Originality**

We the undersigned, hereby certify that this dissertation entitled **“Passenger Positioning System (DigiTekisi)”** presented by **Group 6** has been carried out by them in the Department of Computer Engineering, Faculty of Engineering and Technology, University of Buea under the supervision of **Dr. NKEMENI VALERY**.

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

Date\_\_\_\_\_

**Student**

**Supervisor**

\_\_\_\_\_

\_\_\_\_\_

**Head of Department**

\_\_\_\_\_

## **Dedication**

To all the members of Group six, TANDONGFOR SHALOM CHANGEH, BALEMBA JUNIOR BALEMBA, YIMNAI NERUS ZAUMU, TABOT CHARLES BESSONG and BEBONGNCHU YANNICK NKWETTA for the effort and time they put in to make this project a reality.

## **Acknowledgement**

We express our profound gratitude to all those who assisted us during the project, being with us and guiding us to the fulfillment of it all.

- Our immense gratitude to our supervisor, Dr. Nkemeni Valery for the guidance he gave us.
- To all our course mates, we say thank you to you all.
- Above all, we heartily thank the almighty God for providing us with the strength, courage and guidance we needed to carry out this project.

## **Abstract**

The Digitekisi passenger positioning system is a cutting-edge technological solution designed to enhance the efficiency and convenience of passenger transportation. The system leverages real-time tracking, communication, and advanced navigation algorithms to provide seamless and reliable passenger experiences. By addressing the challenges associated with passenger positioning and improving communication between passengers and drivers, the Digitekisi system aims to optimize the overall passenger journey.

Through the Digitekisi system, passengers can effortlessly track their assigned vehicles, receive real-time updates on their ride status, and communicate with drivers as needed. The system utilizes state-of-the-art technologies to ensure accurate and timely positioning information, allowing passengers to plan their journeys more effectively and eliminate uncertainties associated with transportation logistics.

The system's intelligent routing algorithms optimize passenger pickup and drop-off routes, reducing travel time and minimizing congestion. By integrating with existing transportation infrastructure, such as public transportation networks or ride-sharing platforms, the Digitekisi system creates a seamless and connected transportation ecosystem.

Security and privacy are paramount in the Digitekisi system. Robust measures, including secure login mechanisms and data encryption, are implemented to protect passenger information and provide a safe environment. The user interface is designed with a focus on usability and a seamless user experience, allowing passengers to interact with the system effortlessly.

The scalability and future growth potential of the Digitekisi system ensure its adaptability to changing passenger demands and evolving transportation trends. With its innovative features and functionalities, the Digitekisi passenger positioning system aims to revolutionize the transportation industry, improving passenger experiences, optimizing transportation efficiency, and paving the way for a more connected and convenient future.

**Keywords:** Passenger positioning system, real-time updates, communication, navigation algorithms, transportation efficiency, user experience, security, scalability.

## Table of Contents

Dedication .....	iv
Acknowledgement.....	v
Abstract .....	vi
List of Figures .....	ix
List Of Abbreviations.....	xi
CHAPTER 1. GENERAL INTRODUCTION.....	1
1.1. Background and Context of the Study:.....	1
1.2. Problem Statement:.....	1
1.3. Objectives of the Study: .....	2
1.3.1. General Objective:.....	2
1.3.2. Specific Objectives:.....	2
1.4. Proposed Methodology:.....	2
1.5. Research Questions: .....	3
1.6. Significance of the Study:.....	3
1.7. Scope of the Study:.....	4
1.8. Delimitation of the Study: .....	4
1.9. Definition of Keywords and Terms: .....	4
1.10. Organization of the Dissertation: .....	4
CHAPTER 2. LITERATURE REVIEW .....	6
2.1. Introduction: .....	6
2.2. General Concepts on the System: .....	6
2.3. Related Works: .....	9
2.4. Partial Conclusion: .....	10
CHAPTER 3. ANALYSIS AND DESIGN .....	13
3.1. INTRODUCTION .....	13
3.2. Requirements Gathering: .....	13

3.2.1.	Functional Requirements.....	13
3.2.2.	Non-Functional Requirements .....	14
3.2.3.	User Requirements .....	15
3.2.4.	Regulatory Requirements .....	16
3.2.6.	Technical Requirements .....	16
3.3.	System Architecture: .....	17
3.4.	System Design: .....	19
3.4.1.	Use case diagram:.....	20
3.4.2.	Class Diagram: .....	21
3.4.3.	Sequence Diagrams: .....	22
3.4.4.	Activity Diagram:.....	32
3.4.5.	Data Flow Diagram: .....	34
CHAPTER 4.	IMPLEMENTATION (or REALIZATION) AND RESULTS .....	35
4.1.	UI DESIGN .....	35
4.1.1.	Introduction .....	35
4.1.2.	Tools:.....	35
4.2.	Implementation:.....	38
4.3.	UI Section Description .....	39
4.4.	Implementation for the frontend.....	43
4.5.	DATABASE DESIGN AND IMPLEMENTATION.....	51
4.5.1.	Introduction .....	51
4.5.2.	Conceptual Database Design.....	53
4.6.	Physical Database Design.....	55
CHAPTER 5.	CONCLUSION .....	58



## List of Figures

Figure 1 Usecase Diagram .....	20
Figure 2 Class Diagram.....	21
Figure 3: Signup Sequence Diagram.....	22
Figure 4: Login Sequence Diagram .....	23
Figure 5: Forgot password Sequence Diagram .....	23
Figure 6: Reset Password Sequence Diagram.....	24
Figure 7: View Ride Sequence Diagram.....	24
Figure 8: Book ride sequence Diagram.....	25
Figure 9: Cancel Ride Sequence Diagram .....	26
Figure 10: Upload Money Sequence Diagram.....	26
Figure 11: Passenger History Sequence Diagram .....	27
Figure 12: Create Driver Account Sequence Diagram.....	27
Figure 13: Login Driver Sequence Diagram .....	28
Figure 14: Reset Driver Sequence Diagram.....	28
Figure 15: Subscribe to Bundle Sequence Diagram.....	29
Figure 16: Unsubscribe from bundle Sequence Diagram .....	30
Figure 17: View Routes Sequence Diagram .....	30
Figure 18: Set status Sequence Diagram.....	31
Figure 19: Withdraw money from account Sequence Diagram.....	32
Figure 20: Activity Diagram .....	33
Figure 21: Dataflow Diagram .....	34
Figure 22: Welcome Page .....	39
Figure 23: Login Page .....	40
Figure 24: Map section.....	41
Figure 25: Pay for Ride .....	42
Figure 26: React Dependencies.....	43
Figure 27: Partial Implementation.....	44
Figure 28: Login Implementation .....	45
Figure 29: State Management .....	46
Figure 30: Signup Implementation.....	46
Figure 31: Entry point for Application.....	47
Figure 32: Styling UI.....	48

Figure 33: Root of Frontend Implementation .....	49
Figure 34: Overview of Styling.....	49

## **List Of Abbreviations**

PPS

Passenger Positioning System



# CHAPTER 1. GENERAL INTRODUCTION

The present state of transportation within the community of Buea is in shambles. Though it often gets the job done, it is nonetheless with a lot of difficulty. I remember on April 12 having to stand for about thirty minutes, stopping without success more than 40 taxis with no one going in my direction. What makes it worse is that this had nothing to do with change or the lack of it thereof. When you factor this in, you cannot help but imagine how much worse it probably would have been. This problem is not unique to me. Most people will attest to facing these same difficulties albeit to varying degrees.

This is where the **Passenger Positioning System** comes into play.

## 1.1. Background and Context of the Study:

The transportation industry has witnessed significant advancements in recent years, with the rise of ride-sharing services and the increasing use of technology to enhance passenger experiences. However, one challenge that persists is the difficulty for passengers to accurately locate and track their ride, leading to delays, confusion, and inconvenience. In response to this problem, the project "Digitekisi" aims to develop a passenger positioning system that provides real-time information and improves the overall ride experience.

## 1.2. Problem Statement:

The existing transportation systems often lack an efficient and user-friendly mechanism for passengers to track and locate their rides. This results in a range of issues such as extended waiting times, miscommunication between passengers and drivers, and an overall subpar passenger experience. The lack of a reliable passenger positioning system poses a significant challenge that needs to be addressed to enhance the efficiency and convenience of transportation services.

### **1.3. Objectives of the Study:**

#### **1.3.1. General Objective:**

The general objective of the Digitekisi project is to design and develop a passenger positioning system that enables real-time tracking and location information for passengers using transportation services.

The main goal of this project is to solve this problem by greatly simplifying the process of getting a taxi for both the passengers and drivers. Passengers should be able to know which taxis are going in what direction and likewise the drivers should be able to know the same about the passengers.

Furthermore, passengers should be able to book taxis ahead of time. This will greatly improve the wait times of the taxis at each stop. This is going to cause a domino effect as it will in turn reduce traffic in some hotspots around the locality which will in turn just lead to better community life.

#### **1.3.2. Specific Objectives:**

- To analyze the current challenges and limitations faced by passengers in locating and tracking their rides.
- To design and develop a user-friendly mobile application that allows passengers to track their ride and view the driver's location in real-time.
- To implement a robust and accurate positioning algorithm that ensures reliable location updates for passengers.
- To evaluate the effectiveness and user satisfaction of the Digitekisi passenger positioning system through user testing and feedback.

### **1.4. Proposed Methodology:**

The project will follow a comprehensive methodology that encompasses the following steps:

- Conducting a literature review to gain insights into existing passenger positioning systems and related technologies.

- Analyzing the requirements and expectations of passengers and transportation service providers through surveys and interviews.
- Designing and prototyping the user interface and features of the Digitekisi mobile application.
- Implementing the positioning algorithm and integrating it with the mobile application.
- Conducting usability testing and collecting feedback from a group of selected participants.
- Iteratively refining the system based on user feedback and finalizing the Digitekisi passenger positioning system.

### **1.5. Research Questions:**

- How does the lack of a reliable passenger positioning system affect the overall ride experience for passengers?
- What are the key features and functionalities that passengers expect from a passenger positioning system?
- How can the Digitekisi passenger positioning system be designed to effectively address the challenges faced by passengers?
- What are the factors influencing user satisfaction and acceptance of the Digitekisi system?

### **1.6. Significance of the Study:**

The Digitekisi project holds several significant implications:

- Enhancing the overall passenger experience by providing real-time tracking and location information.
- Improving transportation efficiency and reducing waiting times for passengers.
- Facilitating better communication and coordination between passengers and drivers.
- Reducing the amount of time passengers have to wait for a taxi
- Aiding drivers to optimize fuel consumption by guiding them to move to locations where there are more potential customers.

- Contributing to the advancement of technology-driven solutions in the transportation industry.

### **1.7. Scope of the Study:**

The scope of the Digitekisi project will focus on developing a passenger positioning system for a specific geographic region or city (Buea, Cameroon). It will primarily target transportation services such as ride-sharing platforms and taxi services.

### **1.8. Delimitation of the Study:**

The study acknowledges certain limitations, including:

- The implementation will rely on existing smartphone technologies, such as GPS and mobile networks, and will not delve into the development of new positioning technologies.

### **1.9. Definition of Keywords and Terms:**

- **Passenger Positioning System:** A technology-enabled system that allows passengers to track and locate their rides in real-time.
- **Ride-sharing:** A transportation service where multiple passengers share a single vehicle, typically arranged through a mobile application.
- **User Interface:** The visual and interactive elements of a software application that enable users to interact with the system.

### **1.10. Organization of the Dissertation:**

The remaining chapters of the dissertation will be structured as follows:

- Chapter Two: Literature Review



- Chapter Three: System Analysis and Design
- Chapter Four: Implementation and Integration
- Chapter Five: Evaluation and Results
- Chapter Six: Conclusion and Future Work

## **CHAPTER 2. LITERATURE REVIEW**

### **2.1. Introduction:**

Passenger positioning systems (PPS) are used to track the location of passengers in real time. This information can be used for a variety of purposes, such as passenger safety, crowd management, and marketing. Digitekisi is a PPS that uses GPS to track the location of passengers

### **2.2. General Concepts on the System:**

Accurate passenger tracking, real-time updates, and effective communication between passengers and drivers play a crucial role in revolutionizing the transportation industry. They bring numerous benefits and address key challenges faced by both passengers and service providers. Here is the importance of each aspect:

#### **1. Accurate Passenger Tracking:**

Accurate passenger tracking enables transportation service providers to precisely locate and monitor passengers throughout their journey. This information is invaluable for efficient resource allocation, including assigning vehicles, optimizing routes, and predicting demand patterns. It allows service providers to streamline operations, reduce waiting times, and enhance overall service quality. Additionally, accurate passenger tracking enhances passenger safety by ensuring that their whereabouts are known in real-time.

#### **2. Real-time Updates:**

Real-time updates provide passengers with timely and relevant information about their rides, such as estimated arrival times, delays, or changes in routes. By keeping passengers informed, real-time updates minimize uncertainties, alleviate anxiety, and enhance the overall passenger experience. Passengers can plan their time more effectively, make informed decisions, and

adjust their schedules accordingly. Real-time updates also help service providers maintain transparency and build trust with their passengers, leading to improved customer satisfaction and loyalty.

### 3. Effective Communication between Passengers and Drivers:

Effective communication is vital in the transportation industry as it enables seamless interaction between passengers and drivers. It allows passengers to communicate their specific needs, provide additional instructions, or seek clarifications. Similarly, drivers can communicate important updates, such as unexpected delays or alternative routes, to passengers. This two-way communication fosters a sense of collaboration and ensures that both parties are on the same page, resulting in smoother operations and enhanced customer service. Additionally, effective communication can help address any issues or concerns promptly, leading to better problem resolution and customer satisfaction.

Overall, accurate passenger tracking, real-time updates, and effective communication significantly improve the passenger experience and enhance operational efficiency in the transportation industry. They contribute to reduced wait times, increased reliability, and improved customer satisfaction. By leveraging advanced technologies and seamless integration, transportation service providers can optimize their services, increase passenger engagement, and stay competitive in an ever-evolving industry.

Implementing a passenger positioning system like Digitekisi can bring a wide range of benefits and implications for the transportation industry. Here are some potential benefits and implications of implementing such a system:

#### 1. Enhanced Passenger Experience:

The implementation of a passenger positioning system improves the overall passenger experience by providing real-time updates, accurate tracking, and effective communication. Passengers can easily track their assigned vehicles, receive estimated arrival times, and stay informed about any changes or delays. This enhances convenience, reduces uncertainty, and increases passenger satisfaction.

## 2. Improved Efficiency and Resource Allocation:

With accurate passenger tracking and intelligent routing algorithms, the system optimizes resource allocation. Transportation service providers can efficiently assign vehicles, plan routes, and optimize pick-up and drop-off points. This leads to reduced waiting times, improved route efficiency, and better utilization of resources, resulting in cost savings and increased operational efficiency.

## 3. Increased Safety and Security:

Accurate passenger tracking enhances safety and security measures within the transportation system. In the event of an emergency or incident, the system enables quick identification of passenger locations, facilitating prompt assistance and response. Additionally, the implementation of secure login mechanisms, data encryption, and privacy measures ensures the protection of passenger information, instilling trust and confidence in the system.

## 4. Enhanced Communication and Customer Service:

Effective communication channels between passengers and drivers enable seamless interaction, allowing passengers to relay specific instructions, ask questions, or address concerns. This improves customer service, enables quick problem resolution, and fosters a positive customer-provider relationship. Drivers can also communicate important updates or alternative routes to passengers, ensuring transparency and reducing frustration.

## 5. Data-driven Insights and Decision-making:

The implementation of a passenger positioning system generates valuable data on passenger behavior, travel patterns, and service demand. Transportation service providers can leverage this data to gain insights into passenger preferences, optimize service offerings, and make data-driven decisions. This data can also be utilized for future planning, service improvements, and strategic decision-making.

## 6. Integration and Interoperability:

Implementing a passenger positioning system facilitates integration and interoperability with existing transportation infrastructure and platforms. This allows for seamless connectivity between different modes of transportation, such as public transit systems or ride-sharing platforms. It promotes multi-modal transportation options, simplifies interconnections, and offers passengers a more integrated and convenient travel experience.

## 7. Innovation and Future Growth:

The implementation of such a system showcases technological innovation and positions transportation service providers at the forefront of industry advancements. It opens up opportunities for future growth, including potential partnerships, collaborations, and expansion into new markets. The system's scalability allows for accommodating increasing passenger demands, expanding service areas, and adapting to evolving transportation trends.

While there are numerous benefits, implementing a passenger positioning system also brings implications that need to be considered. These may include initial costs, technical implementation challenges, data privacy and security considerations, and the need for training and adoption by both passengers and drivers. However, with careful planning, effective implementation strategies, and continuous evaluation, the benefits can outweigh the challenges, resulting in a transformative and efficient transportation ecosystem.

## **2.3. Related Works:**

The first step of the literature review was to conduct an online Transportation Research Information Services search. This search yielded some very important documents, the most relevant of which were reviewed and used as input to this report. The second step was to obtain and review articles, press releases, and website information directly from others you have made use of, or are still making use of the system. The third step was to review research report papers and articles that were obtained from different sources. These researches proved that there is a growing body of literature on PPSs. Some of the most relevant studies include:

- "A Survey on Passenger Positioning Systems" (2017) by Wang et al. This study provides a comprehensive overview of PPSs, including their different technologies, applications, and challenges.
- "A Review of Passenger Positioning Systems for Public Transportation" (2018) by Zhang et al. This study focuses on PPSs for public transportation, and discusses the different technologies that are used in these systems.
- "A Performance Evaluation of Passenger Positioning Systems" (2019) by Chen et al. This study evaluates the performance of different PPSs, and discusses the factors that affect their accuracy.

## **2.4. Partial Conclusion:**

The literature review shows that PPSs are a promising technology with a variety of potential applications. The Digitekisi passenger positioning system holds significant relevance within the broader research landscape of transportation systems and technologies. Its innovative features and functionalities contribute to the ongoing efforts to enhance the efficiency, convenience, and overall passenger experience in the transportation industry. Here are the key aspects highlighting the relevance and significance of the Digitekisi system:

### **1. Technological Advancements:**

The Digitekisi system incorporates advanced technologies such as real-time tracking, communication, and navigation algorithms. By leveraging these technological advancements, the system addresses the challenges associated with passenger positioning and communication, paving the way for more efficient and seamless transportation experiences. It aligns with the research focus on developing and implementing cutting-edge technologies to optimize transportation operations and improve passenger satisfaction.

### **2. User-Centric Approach:**

The Digitekisi system places a strong emphasis on the needs and preferences of passengers. It aims to provide a user-centric approach by offering accurate tracking, real-time updates, and effective communication channels. This aligns with the research objective of creating

passenger-centric transportation systems that prioritize user experience and satisfaction. The system's focus on enhancing passenger convenience, reducing uncertainties, and providing transparent and personalized services contributes to the broader research landscape focused on passenger-centric transportation solutions.

### 3. Integration and Interoperability:

The Digitekisi system highlights the importance of integration and interoperability within the transportation ecosystem. By seamlessly integrating with existing transportation infrastructure, such as public transit systems or ride-sharing platforms, the system enables a more connected and convenient travel experience for passengers. This aligns with the research focus on developing integrated transportation networks, fostering intermodal connectivity, and promoting seamless travel across different modes of transportation.

### 4. Data-driven Decision-making:

The implementation of the Digitekisi system generates valuable data on passenger behavior, travel patterns, and service demand. This data can be utilized for data-driven decision-making, optimizing resource allocation, and planning future transportation strategies. The system's contribution to data-driven research aligns with the broader research landscape focused on leveraging data analytics and insights to improve transportation systems' efficiency, sustainability, and overall performance.

### 5. Industry Transformation:

The Digitekisi system represents a significant step towards transforming the transportation industry. By addressing key challenges related to passenger positioning, communication, and convenience, the system offers a comprehensive solution that enhances operational efficiency, passenger experiences, and service provider profitability. Its potential for scalability and future growth contributes to the ongoing research on disruptive technologies and business models that drive industry transformation.

Overall, the Digitekisi passenger positioning system stands as a relevant and significant contribution within the broader research landscape of transportation systems. Its technological advancements, user-centric approach, integration capabilities, data-driven insights, and potential for industry transformation align with the ongoing research endeavors aimed at improving transportation efficiency, enhancing passenger experiences, and creating sustainable and connected transportation ecosystems.



## **CHAPTER 3. ANALYSIS AND DESIGN**

### **3.1. INTRODUCTION**

After carrying out various tasks on mobile applications like finding out the major types of mobile applications and their differences, we went further to look at the programming languages used for mobile application as well as the very large number of frameworks available for mobile application development. More so, we looked at how we can collect and analyze the requirements for mobile applications. Considering we are building a mobile application for municipal commuting; we analyzed our system based on both the perspectives of the drivers and passengers.

### **3.2. Requirements Gathering:**

We identified six different kinds of requirements for this project which are

- Functional requirements
- Non-Functional requirements
- Business requirements
- Regulatory requirements
- User requirements and
- Technical requirements

#### **3.2.1. Functional Requirements**

Functional requirements define the features the application must possess and functions it must perform. In short, they define the application in terms of its inputs, outputs and behaviours. Below are the functional requirements we have come up with for this application.

- Users(passengers) and Drivers should have the ability to Create accounts and log in to those accounts.
- Passengers should have the ability to view different drivers and their various locations
- Passengers should have the ability to book/cancel rides from the application
- Passengers should be able to pay for rides via Mobile money
- Passengers should have the ability to view their ride history from within the application
- Passengers should have the ability to track the drivers and the estimated time of arrival at their locations
- Drivers should have the ability to view active users and their locations
- Drivers should be able to view routes and their degrees of traffic
- Drivers should be able to know passenger concentration in different locations
- Drivers should be able to view statistics about their rides
- Drivers should be able to accept payments into their accounts
- Drivers should be able to withdraw payments into their local accounts
- Passengers and Drivers should be able to set their state to active or inactive in order not to confuse each other

### **3.2.2. Non-Functional Requirements**

Unlike functional requirements, non-functional requirements describe how the system should perform instead of what it does.

These are the key nonfunctional requirements that we pinpointed which we believe are vital for this application's success.

- **Performance**

The system should be able to manage with relative ease a huge number of interactions between drivers and passengers and concurrent transactions. This is representative of how often passengers interact with drivers physically

- **Usability**

The application should be easy and intuitive to use to both the passengers and the drivers. It should have clear and concise navigation with explicit error messages in cases where things don't go as planned. Users shouldn't have to imagine why things are behaving the way that they are.

- **Reliability**

The system should be available and responsive at all times with minimal downtime and errors. When drivers are booked by passengers, they should be confident that those are real people they are going to carry and not just some bots spamming the system.

- **Security**

User data should be well secured. Appropriate measures should be put in place to ensure this such as data encryption, authorization and authentication.

- **Scalability**

The system should be able handle an increasing number of users(drivers and passengers alike) without significant change or degradation in its performance.

- **Compatibility**

The application should be readily available on both android and IOS devices with similar functionality.

### **3.2.3. User Requirements**

These requirements have to do with the expectations of the users of the system. In this case both the drivers and passengers.

- Accurate and real-time positioning information

- Easy to use interface for passengers
- Privacy protection for user data
- Compatibility with different devices and operating systems
- Reliable performance in different situations

#### **3.2.4. Regulatory Requirements**

- Compliance with data protection and privacy laws
- Compliance with transportation safety regulations
- Accurate and reliable positioning information for emergency response and rescue operations
- Integration with emergency response systems and protocols
- Compliance with environmental regulations and standards

#### **3.2.5. Business Requirements**

From a business perspective, the goals of this project is;

- Cost effective implementation and maintenance
- Scalability to accommodate a large number of passengers and vehicles
- Integration with existing transportation systems and infrastructure
- Ability to generate reports and analytics
- Compliance with industry standards and regulations

#### **3.2.6. Technical Requirements**

Technical requirements here refer to the hardware, software and infrastructure requirements of the project. In an ideal situation, the system will contain both a mobile application for passengers and drivers and a web dashboard for administrative control and even for account management by drivers. But in this case, we'll be focusing more on the interface for the driver and passenger. As a result, we'll be building just the

mobile application that meets the needs of both the drivers and passengers. Of course, the system will be built in such a way that it can be expanded to allow many other interfaces to be integrated into it.

Also, another very important part of the technical requirements has to do with the tools and the reasons why we chose them.

### **3.3. System Architecture:**

The Digitekisi passenger positioning system follows a high-level system architecture that enables the seamless integration of its various components and modules. The architecture is designed to ensure reliability, scalability, and efficient data flow within the system. Here is a discussion of the high-level system architecture of the Digitekisi passenger positioning system:

#### **1. Client Applications:**

The system architecture begins with the client applications, which include the passenger application installed on passengers' smartphones and the driver application installed on drivers' smartphones. These applications serve as the primary interfaces for passengers and drivers to interact with the system. They provide functionalities such as ride booking, real-time tracking, communication, and feedback submission. The client applications are designed to be user-friendly and intuitive, ensuring a seamless experience for users.

#### **2. System Backend:**

At the core of the architecture is the system backend, which serves as the central processing unit. It handles the logic and functionality of the Digitekisi system. The backend manages user authentication, ride requests, driver assignments, location tracking, communication, and data storage. It interacts with the client applications, mapping and navigation services, and the database to facilitate data exchange and system operations. The backend is designed to be robust, scalable, and capable of handling a large number of concurrent users and ride requests.

### **3. Mapping and Navigation Services:**

The Digitekisi system relies on mapping and navigation services to provide accurate location tracking and route optimization. These services integrate with mapping platforms and GPS technologies to enable real-time vehicle tracking, route planning, and navigation. They ensure that drivers receive optimal routes to reach passengers' locations efficiently. The mapping and navigation services interact with the system backend to provide location data and route information to the client applications and facilitate efficient communication between passengers and drivers.

### **4. Database:**

The system architecture includes a database component responsible for storing and managing various types of data. This includes passenger profiles, driver profiles, ride details, vehicle information, and historical data. The database enables efficient data retrieval and storage for system functionality and supports data analytics for generating insights. It interacts with the system backend to store and retrieve data as required, ensuring data consistency and integrity.

The high-level system architecture of the Digitekisi passenger positioning system ensures the smooth integration and coordination of its components. It enables real-time tracking, and seamless interaction between passengers, drivers, and the system backend. The architecture is designed to support scalability, reliability, and data integrity, ensuring a robust and dependable system for passenger positioning and transportation management.

Overall, the Digitekisi passenger positioning system functions by integrating the passenger and driver applications, a web-based dashboard, system backend, database, mapping and navigation services, and communication interfaces. These components work together to enable accurate passenger tracking, real-time updates, and effective communication, ensuring a seamless and efficient transportation experience for both passengers and drivers.

### **3.4. System Design:**

Design here is a drawing which is produced to show the function or workings of a system. On designing our system, we took into consideration all the analysis we gathered which we will elaborate on based on various UML diagrams. They include:

- Use case diagram
- Class diagram
- Sequence diagrams
- Activity diagrams
- Data flow diagram.

In our design, we made use of the waterfall model which has to do with breaking down of project activities into linear sequential phases where each phase depends on the previous one in order to function.

### 3.4.1. Use case diagram:

Our use case diagram here shows a graphical representation on how the various users of the system interact with the system. Below is the use case diagram of our system.

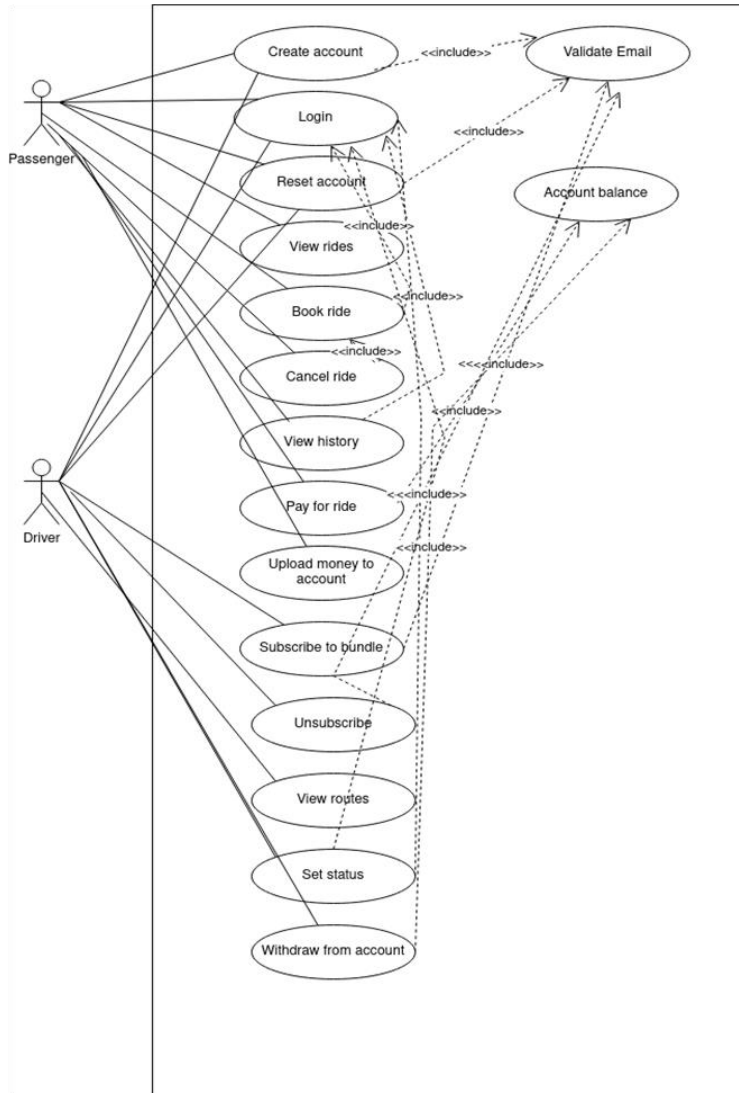


Figure 1 Usecase Diagram

### Explanation:

Our use case diagram shows the two main actors of the system which are the drivers and the passengers. Both the drivers and passengers have some common actions in which they both perform for example, create account, login, reset account. The drivers and passengers also have actions in which only them perform, for example the driver can subscribe to bundles, unsubscribe, view route etc. as well as the passenger can book ride, cancel ride, view history etc. From the diagram above, the dotted arrows with include shows operations which can not



take place unless a different operation has already been done. For example a passenger cannot validate email without creating an account, cannot view history without login etc. also the driver cannot unsubscribe from a bundle without first subscribing, cannot view his profile without login etc.

### 3.4.2. Class Diagram:

A class diagram is the main building block of object oriented modeling. We model our system with a class diagram to show the relationship between our various classes and the services that they provide to our system. Below is the class diagram of our system.

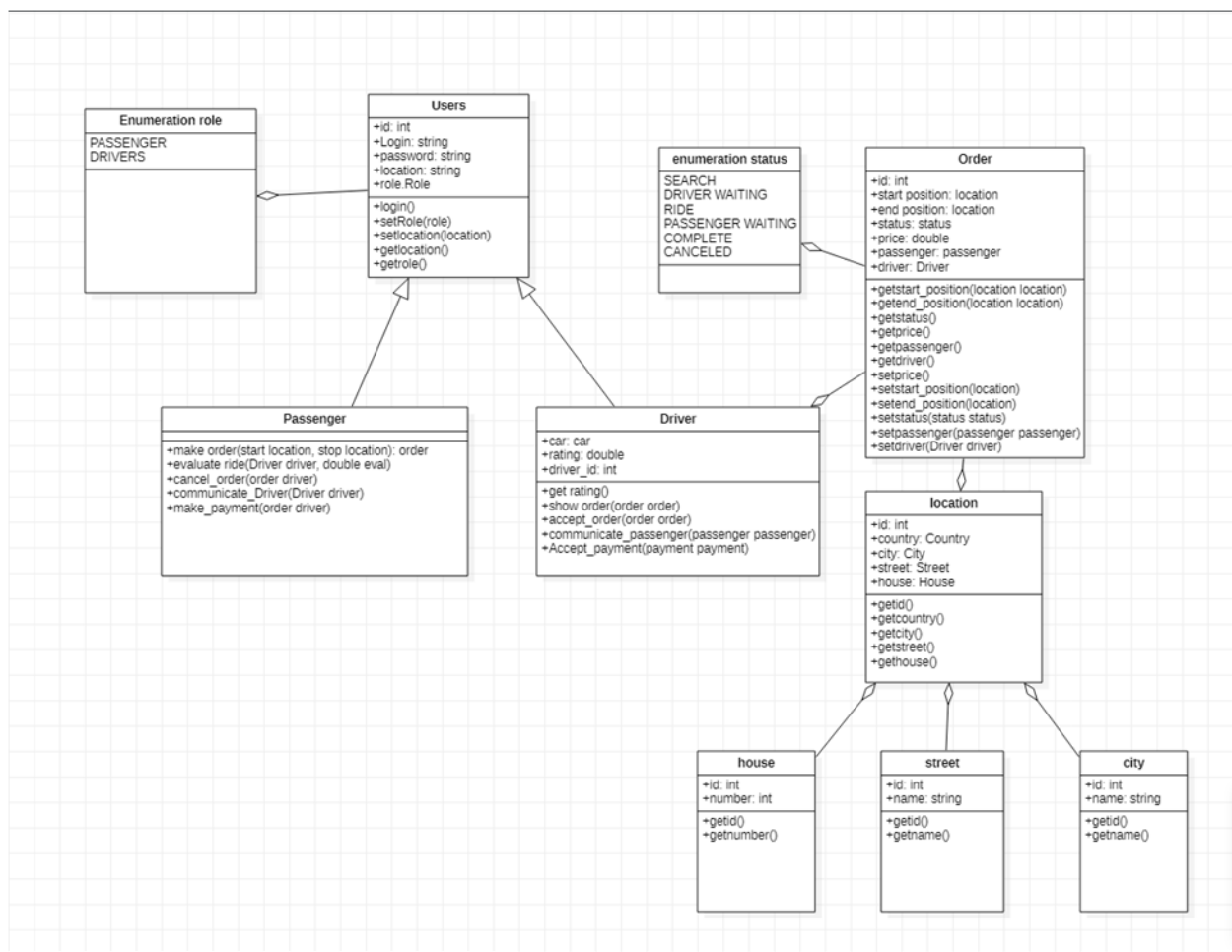


Figure 2 Class Diagram

From the above diagram, we designed our system to have 8 classes with some of them having relationships with others. Based on our design, we have our class users. The user class has certain attributes like id, login password, location and role. This user class can perform functions which can be seen on the above diagram. Our system is basically designed to

support two types of user, which therefore leads us to a generalization which exists between the user class and its sub two classes which are the passenger class and the driver class.

The driver and the passenger automatically inherits all the attributes and functions which the user class has and also has other additional attributes and functions.

We also have a class order with its attributes as well as its operations. It forms an aggregation relationship with the class location which means the class order is subordinate to the class location as well as house, street and city and subordinates to the class location.

### 3.4.3. Sequence Diagrams:

A sequence diagram is a diagram that shows how objects interact with each other and what messages they exchange over time. For our system, we made sure that each use case has its own sequence diagram to show the interactions between its objects. This can be seen in both the passenger and the drivers perspectives.

For the passengers use cases, we have the following:

#### Signup Sequence:

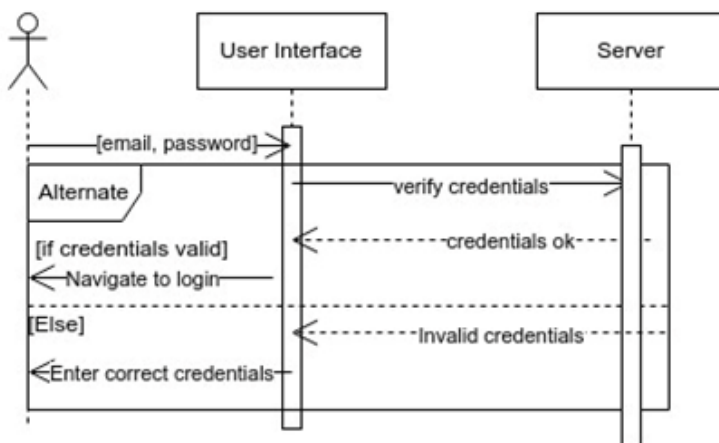


Figure 3: Signup Sequence Diagram

Here, the system ask the user to enter certain credentials which are his email and password, now the system checks if the information he entered was valid that is in the right way, if valid, the user navigates to login and if not valid, the systems pops up a message for the user to enter correct credentials.

## Login sequence:

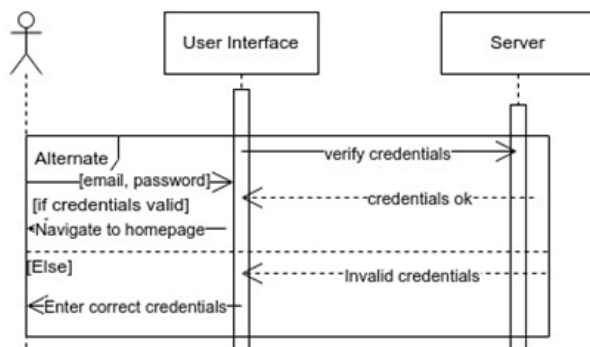


Figure 4: Login Sequence Diagram

From the diagram above, we can see the interaction between the user and the server. It's seen that the user enters their credentials and if credentials are correct, moves to the homepage of the app while if not correct, an error message of invalid credentials pops up. The user then tries again.

## Forgot Password sequence:

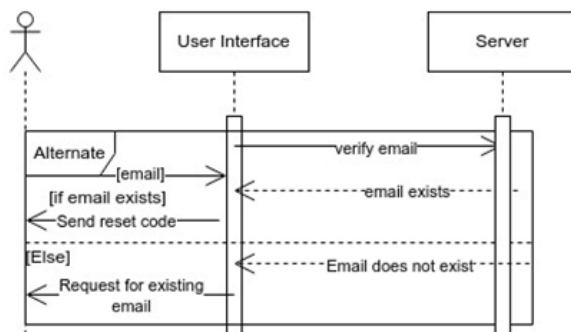


Figure 5: Forgot password Sequence Diagram

If for example a user forgets his/her credentials such as password, this user will be sent a reset code if the email is valid and if not valid, the system will request for a valid email.

## Reset password:

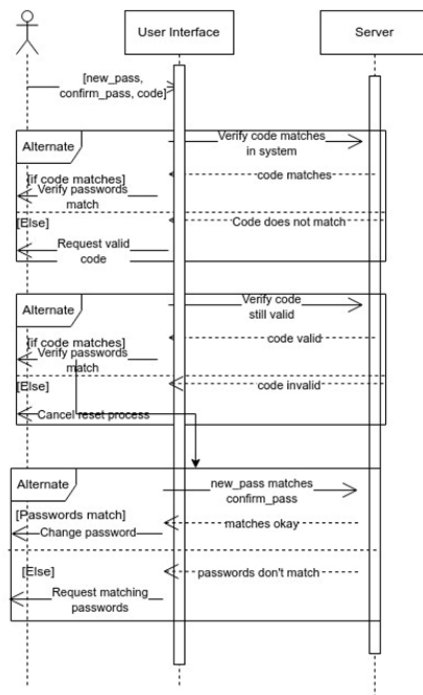


Figure 6: Reset Password Sequence Diagram

After the system sends a confirmation code to the user, he then enters the code to verify, if the code does not match, the system tells the user that the code he entered isn't valid, and asks him to request for a valid code and if the code doesn't still match, the system cancels the reset process. On the other hand, if the code matches, then the system provides a password to the user which he then uses.

## View ride sequence:

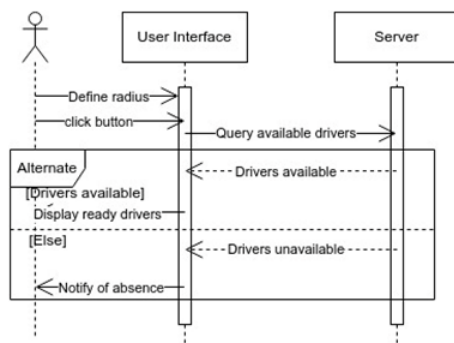


Figure 7: View Ride Sequence Diagram

The user has to select the estimated distance he/she wants to go then click the view ride button to see if there are any drivers for that direction. The system then tells the user that there are available drivers but if not, it still sends a message saying no available drivers.

### Book ride sequence:

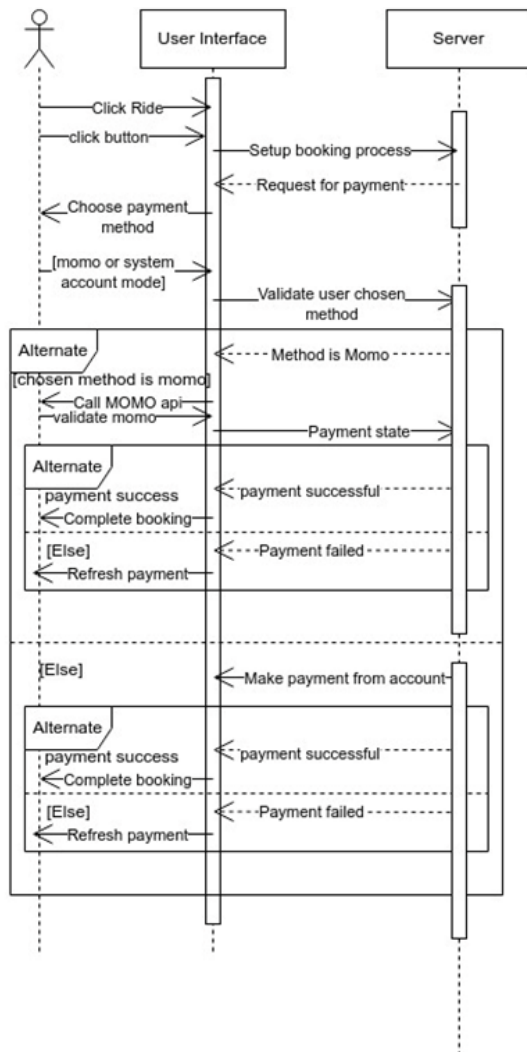


Figure 8: Book ride sequence Diagram

The system is designed in a way that it has a button on which the user can click to book a ride, after which he makes payment either through the system or MoMo. If payment was successful, the system pops a payment validated message and if on failure, a payment failed message which the user can refresh and try to pay again.

## Cancel Ride:

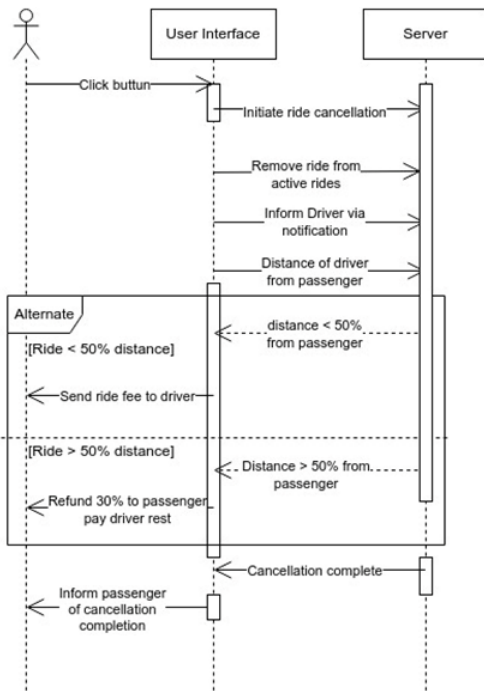


Figure 9: Cancel Ride Sequence Diagram

Our system has a cancel ride button which when clicked, the passengers' ride is canceled based on a few conditions. If the driver has already covered more than 50% to where the user is, the ride fee is sent to the driver but if the driver has covered less than 50%, 30% of the fee is sent to the user and the rest to the driver. The system then informs the user that his cancellation was successful.

## Upload money sequence:

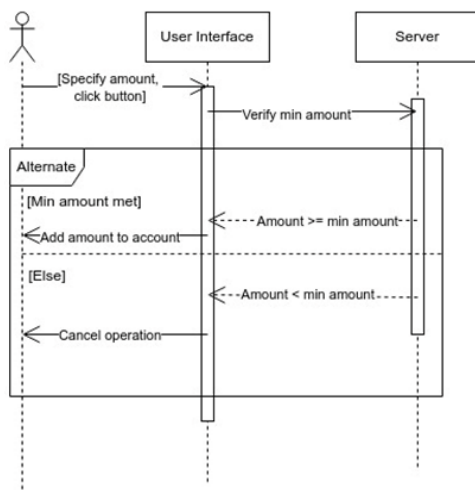


Figure 10: Upload Money Sequence Diagram

When the user wants to upload money to his/her account, the user specifies the amount to upload and if the amount is greater or equal to minimum amount, the user adds the amount to account but if the money isn't greater than minimum amount, the operation is canceled.

### Passenger history sequence:

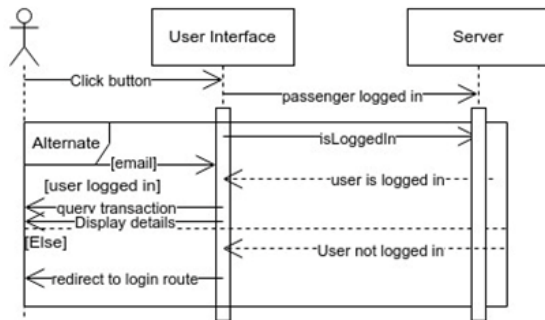


Figure 11: Passenger History Sequence Diagram

For the passenger or user to view their history, the user clicks the view history button and if the user has already logged in, he can view his transaction details and if the user not logged in, the system redirects him to the logging page.

### For the drivers use cases, we have the following sequence diagrams:

#### Creates Account sequence:

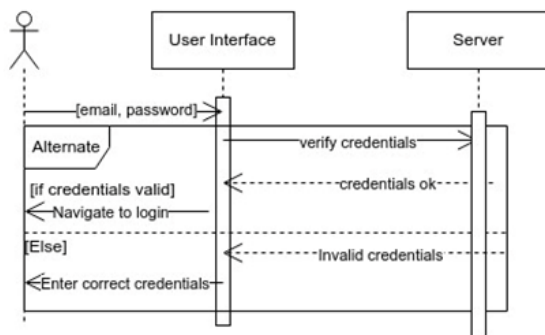


Figure 12: Create Driver Account Sequence Diagram

The system asks the user to enter certain credentials which are his email and password, now the system checks if the information he entered was valid that is in the right way, if valid, the user navigates to login and if not valid, the system pops up a message for the user to enter correct credentials.

## Login sequence:

The user enters their credentials and if credentials are correct, moves to the homepage of the app while if not correct, an error message of invalid credentials pops up. The user then tries again.

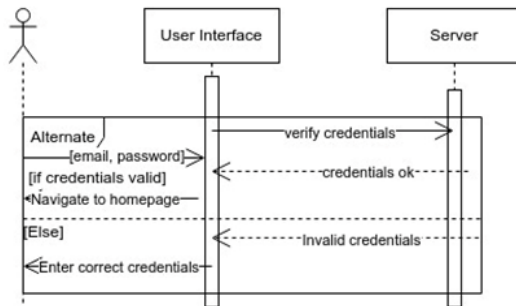


Figure 13: Login Driver Sequence Diagram

## Reset password sequence:

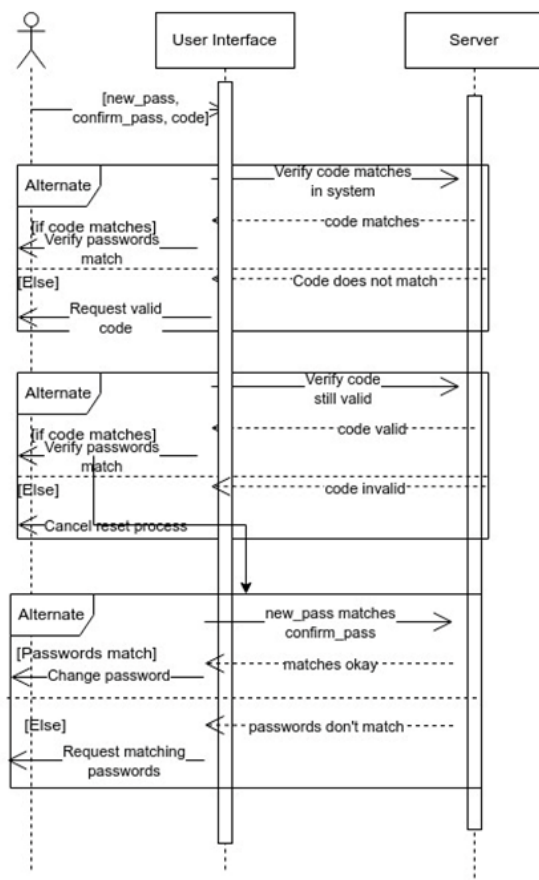


Figure 14: Reset Driver Sequence Diagram



After the system sends a confirmation code to the driver, he then enters the code to verify, if the code does not match, the system tells the user that the code he entered isn't valid, and ask him to request for a valid code and if the code doesn't still match, the system cancels the reset process. On the other hand if the code matches, then the system provides a password to the user which he then uses.

### Subscribe to bundles sequence:

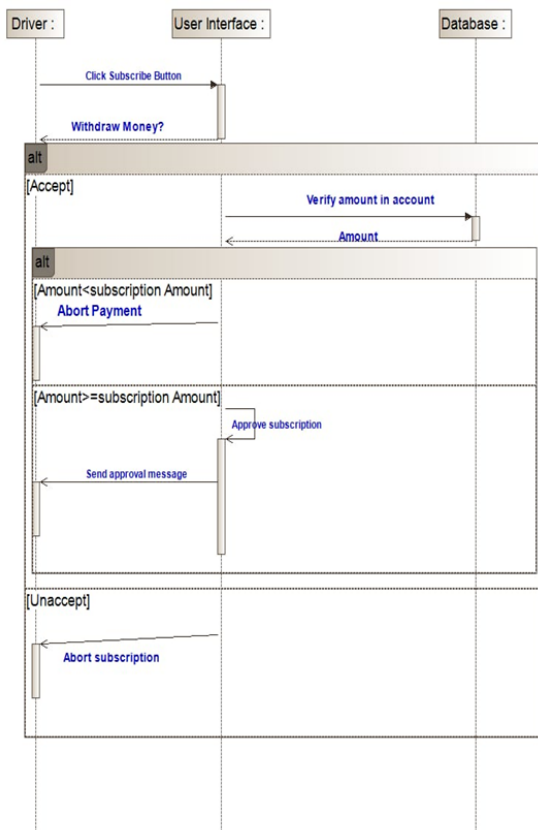


Figure 15: Subscribe to Bundle Sequence Diagram

The driver clicks on a subscribe button if he's interested in a particular bundle. After clicking the system take system ask for permission to withdraw his subscription fee from his account. If he accepts then the system withdraws and displays a success message and if the transaction is not successful due to insufficient funds, the system displays a failure message.

### Unsubscribe sequence:

When the driver clicks the unsubscribe button, the system checks if he's subscribed to that bundle and if true, it grants his request and displays unsubscribed.



Figure 16: Unsubscribe from bundle Sequence Diagram

### View routes sequence:

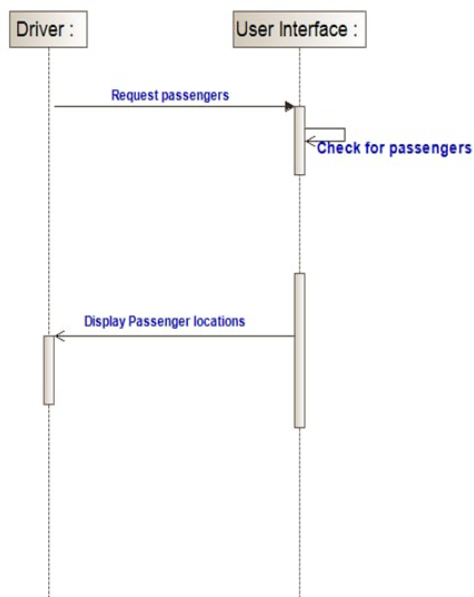


Figure 17: View Routes Sequence Diagram

The driver checks the position in which there is a high concentration of passengers then the system simply displays the location.

### Set status sequence:

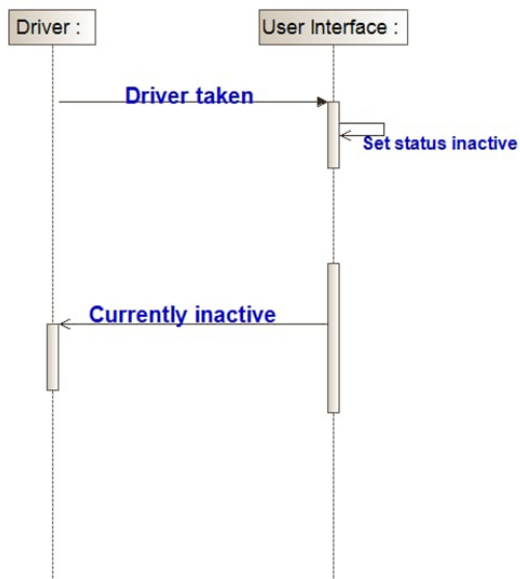


Figure 18: Set status Sequence Diagram

If a has already been booked, he sets his status and inactive until after he's done with that ride, then he can change his status to active.

### Withdraw from account sequence:

The driver requests for withdrawal and the system sends him a password and asks him to input the password, if the password is incorrect, the system aborts the process but if correct, the driver inputs the amount to be withdrawn. If the amount he entered is not up to what is in his account, the system aborts the process but if it's enough, the system approves withdrawal and after it's done, sends a transaction complete message.

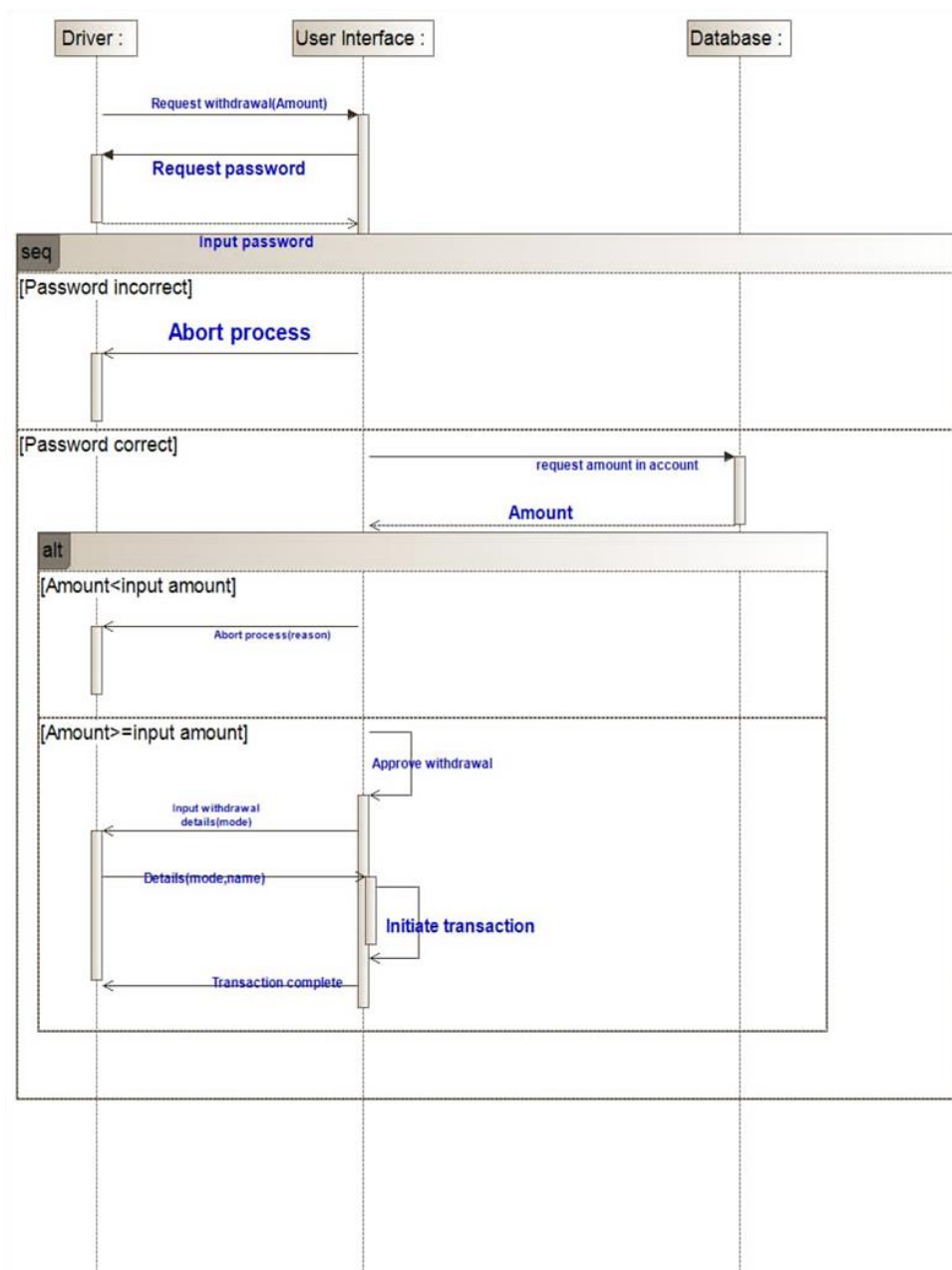


Figure 19: Withdraw money from account Sequence Diagram

### 3.4.4. Activity Diagram:

An activity diagram provides a view of the behavior of the system by describing the sequence of actions in a system.

Based on our municipal commuting system, there are varieties of actions taking place performed by the use cases in the system. Below is the activity diagram for our system.

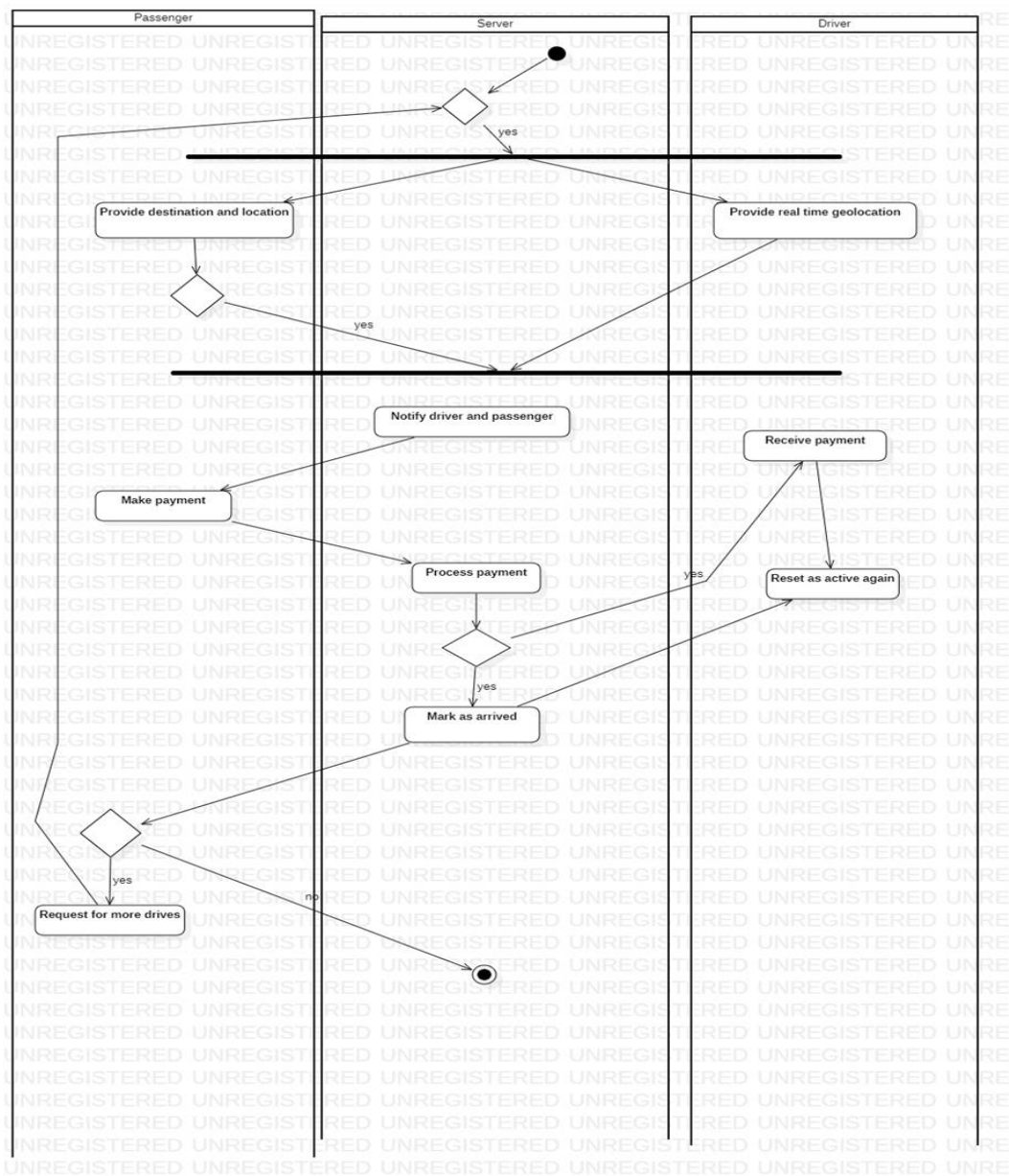


Figure 20: Activity Diagram

Our system has mainly three actors which are passenger, server and the driver.

No action is done until when a user comes and performs an action. When a passenger gets into the system and books a ride, the server gets the location of the passenger and also gets the drivers that are available at that particular location. If there are passengers available at a particular area, it sends a signal to the driver that there are passengers at this location. When the passenger initiates the payment, the server checks if the amount you've entered is greater than the

minimum amount. If its greater, the system withdraws from the users account and sends a success message and if not, terminates the process.

Now if the user reaches his destination and wishes to book another ride to somewhere else, the system restarts the entire process again but if not, the process terminates.

### 3.4.5. Data Flow Diagram:

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. For our system, we have

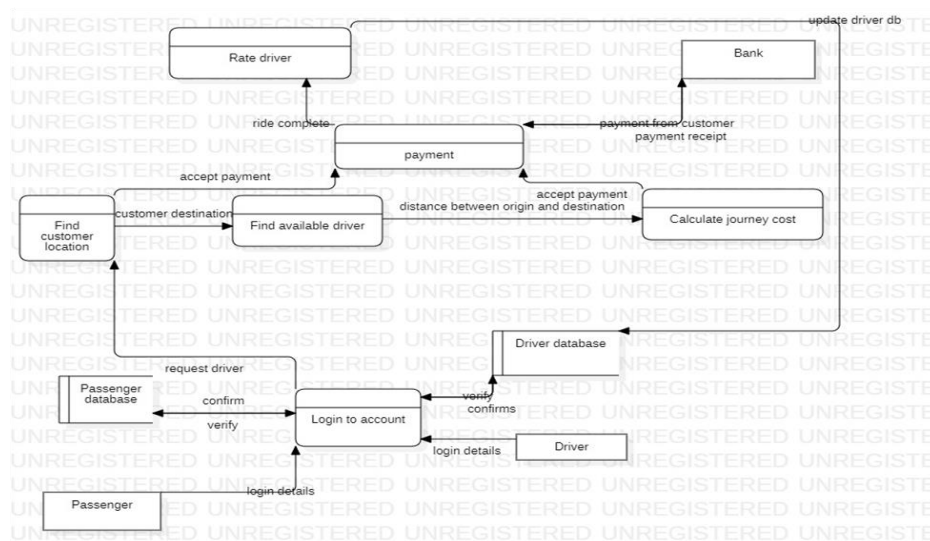


Figure 21: Dataflow Diagram

When the passenger logs in, his information goes into the database and it's checked if this user already exists in our system, if true, then login but if the user doesn't exist, then an error message. When the passenger request for a driver, the system checks for available drivers for that direction and if available, the system calculates the cost to that destination and sends to the passenger to make payments and after the passenger makes the payment, a receipt is sent to both the passenger and the driver that this person has made payments.

For the driver, he logs in too and his credentials are checked and if valid, access is granted and if, error signal. Receives his money on any payment and then services the ride.

## CHAPTER 4. IMPLEMENTATION (or REALIZATION) AND RESULTS

### 4.1. UI DESIGN

#### 4.1.1. Introduction

The UI design phase is the process of designing a software system's user interface (UI). The UI is the part of the system that users interact with, and it includes things like the layout, colors, fonts, and icons. The goal of UI design is to create a UI that is easy to use and understand and helps users achieve their goals.

#### 4.1.2. Tools:

##### 1) UI Design: Figma

**Reason:** Besides the fact that it is really good and gets the job done, it is very easy to use, free and all team members have some minimum amount of experience with it.

##### 2) Mobile Application UI: React JS(RJ)

**Reason:** We initially planned to go with React Native for our application but we faced some issues so we had to switch to React JS. React JS is a Javascript library used to build one page applications with reusable components. Multiple pages can easily be added with react-router-dom. React also gives you the possibility to deploy your app as a progressive web app (PWA) which can be used on a mobile phone then still maintaining our mobile ability

##### 3) Styling UI: Tailwind CSS

**Reason:** Tailwind CSS is a popular CSS framework that can be used to style React JS apps. One of the main benefits of using Tailwind CSS is that it provides a set of pre-designed utility classes that can be used to quickly style HTML elements without having to write custom CSS code. This can save development

time and streamline your CSS workflow. Therefore making it easy to produce responsive design. Responsive design gives a very good user experience

#### 4) **Database:** PostgresQL/Redis Cache

**Reason:** At the beginning, we started by using planet scale but during our work with planet scale, we realized it does not support the foreign key constraint. So due to that, we had to switch to postgresQL.

The entire team has been working with SQL right from the start of our program at FET so we are fairly good with it. With MYSQL being an sql database, it was going to be easy to make use of PostgresQL and Prisma for our ORM. The combination of the two provides us with the following advantages.

PostgreSQL is a widely used open-source relational database management system that provides excellent reliability, data integrity, and performance. It has a proven track record of being used in production environments for many years, and it supports a wide range of data types and features.

Prisma is a modern ORM (Object-Relational Mapping) tool that provides a type-safe and intuitive way to interact with your database. It generates efficient SQL queries and provides an easy-to-use API for database operations, reducing the amount of boilerplate code you need to write.

Prisma also integrates well with GraphQL, a popular API design language that makes it easy to build flexible and efficient APIs. By using Prisma with GraphQL, you can easily define your data schema and query your database in a type-safe and efficient way.

Prisma also provides a powerful data modeling tool that allows you to define your database schema using a simple and intuitive syntax. This makes it easy to create and modify your database schema, reducing the risk of errors and making it easier to maintain your database over time.

Finally, both PostgreSQL and Prisma are open-source tools with active communities and excellent documentation. This



## 5) **Backend Language/Framework:** Typescript/NestJs

**Reason:** NestJs is a NodeJs framework for building performant and scalable backend applications(APIs). The fact that it uses typescript also means it provides a high level of type safety for us thus catching a lot of errors at compile time instead of runtime. This makes it very suitable for our use case. More important is the fact that part of the team has some experience working with it so it won't require the entire team to learn an entire framework hence making us move faster.

## 6) **Deployment**

**Backend:** Azure Container Apps: We intend to host the final backend as an Azure Container application. This means we'll have to build the app as a docker container in the end before deployment. The reason for this choice is that one of our team members already has a functioning account on Azure and so it makes sense to use it instead of looking for something brand new. Also, it just works and is not too difficult to pick up by other team members.

**Mobile App:** Unfortunately, we will only be able to deploy on Android at this point via APKs as the google play store will require a \$25 fee for deployment and even worse than that, the Apple App store requires a \$99 charge and doesn't even support APKs. Depending on the direction the other team members chose to go, we might consider these other options even after this course.

**Documentation:** This application will be documented with the help of Swagger documentation with the direction of the OpenApi specification. This is going to be very vital for the developers of the UI as they'll need guidance when consuming the API services that power the application.

## **4.2. Implementation:**

For our UI Design, we used User-centered design (UCD) which is a methodology that focuses on the needs of the user. It involves understanding the user's goals, tasks, and abilities, and then designing a user interface that meets those needs.

We used Figma for the creation of our User Interfaces.

Figma is a popular design tool that is used by designers and developers to create user interfaces (UIs). It is a web-based tool, which means that it can be accessed from anywhere with an internet connection. This makes it a convenient option for teams that are working remotely or that have members in different locations.

### 4.3. UI Section Description

#### Welcome Section

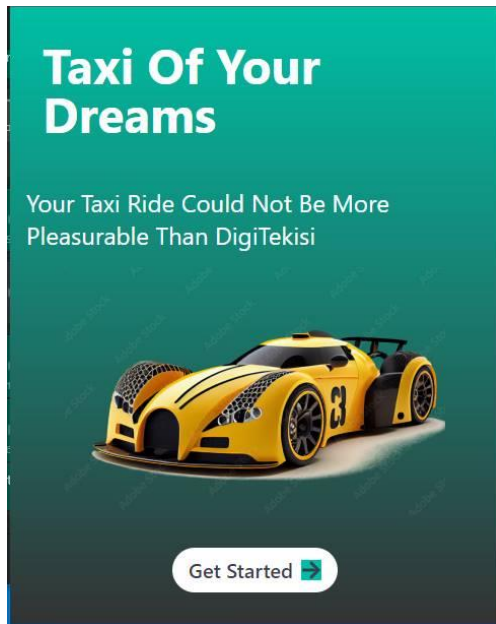


Figure 22: Welcome Page

After successful login into the system, the user (passenger or driver) is welcome into the application with a Get started page. The page is designed in such a way that it gives the user the feel of what services the system will provide for them. There is a get started button that will take the user to other pages where they will see all the advantages the application will offer to them. Advantages such as

Not waiting in traffic

No change issues

Waiting long hours

In those pages to help the user navigate from one advantage to another, we added navigation for the user to move from one page to another page. Also keeping in Mind that not every User will love to see the boring advantages especially if they already know what the system offers, we added a skip button that will navigate the user directly to the homepage

## Authentication section

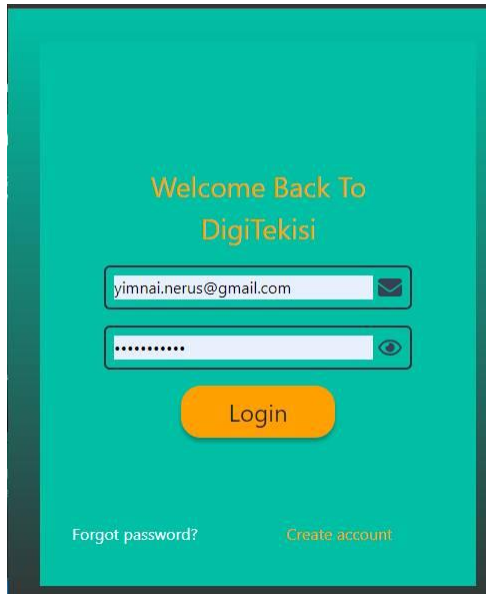


Figure 23: Login Page

First time the user opens the app, they will not be prompted to create an account. They will have the chance of viewing the homepage. But when they want to view available rides or become a driver or book ride, then they will need to create an account. Creating an account will be very important as we will need to keep track of each user's activities and be able to bill them or pay them for every ride. For the account creation, we need just the user email and password.

The email will be used to uniquely identify the user. When the next user opens the app after they already have an account, logging in will be the only thing they will have to do. For the login, only the email and password will also be required. When logging in or signing up, we have added errors to be displayed on the screen to show the user where they made a mistake or did not input the correct value.

For example when signing up, if the email already existed the user will be shown an error stating that that email has already been taken by another user. As for the password to make the system secure, we have chosen a strong password pattern which if the user does not follow when creating their account, they will be shown an error telling them what's left to make the password complete.

To avoid bugs entering our system, we have added a verification of account which will just require a user to enter the code sent to them in the verification mail via the mail they used to create their account.

As humans we tend to forget, but we did not want to forget that important aspect about our users. So we implemented a UI section that will enable them to change their password if they have forgotten the old one. But to avoid people from stealing other people's accounts, the user will have to enter their email so that we send them a verification code with a unique code they will use to change their password.

## Maps section

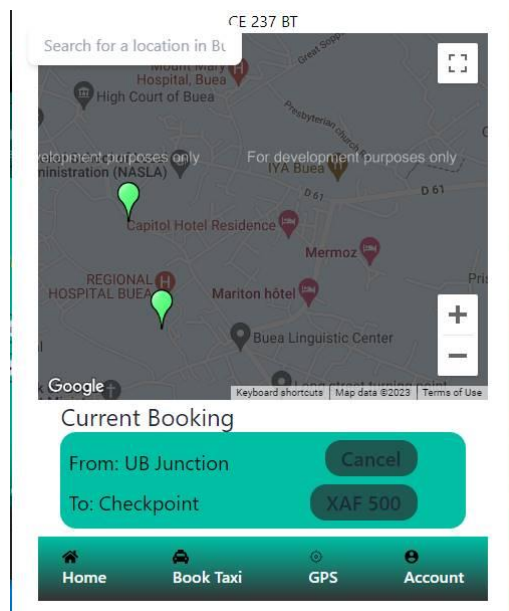
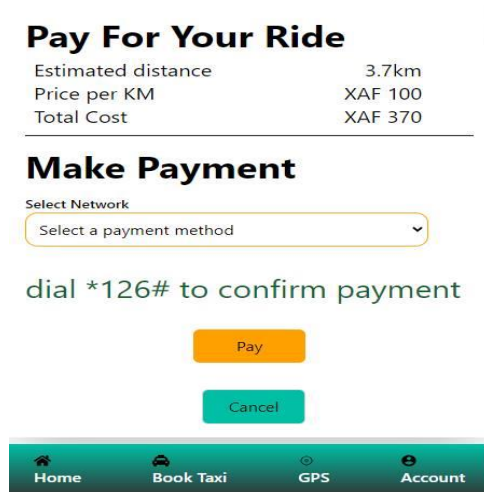


Figure 24: Map section

We have thought of making the experience easy for our users as much as possible. Whenever the passenger wants to book a ride, they have a map interface which helps them select their location and destination. For the location and destination they will just have to either enter it as text input or select it as an option from our different clusters. Once the passenger has chosen their destination and present location, the total distance of the journey is being calculated and displayed to the user.

## Payment section

The passenger chooses a method of payment and pays the driver. The driver receives a notification when the payment is complete.



The screenshot shows a mobile application interface for paying for a ride. At the top, under the heading "Pay For Your Ride", there is a summary of the ride: "Estimated distance 3.7km", "Price per KM XAF 100", and "Total Cost XAF 370". Below this, the section "Make Payment" contains a "Select Network" label and a dropdown menu with the text "Select a payment method". A green instruction "dial \*126# to confirm payment" is displayed. At the bottom of the payment section are two buttons: an orange "Pay" button and a green "Cancel" button. The app's bottom navigation bar includes icons and labels for "Home", "Book Taxi", "GPS", and "Account".

Figure 25: Pay for Ride

## Apply as driver section

Every User who creates an account for the first time is considered as a passenger, to become a driver, you will have to fill in additional information.

## 4.4. Implementation for the frontend

We ended up implementing our frontend with react JS due to some challenges with the react native environment. We used react JS to produce all the different pages in our application and implemented the navigation from one page to another using react-router-dom

React Packages that enabled our application development to be easy.

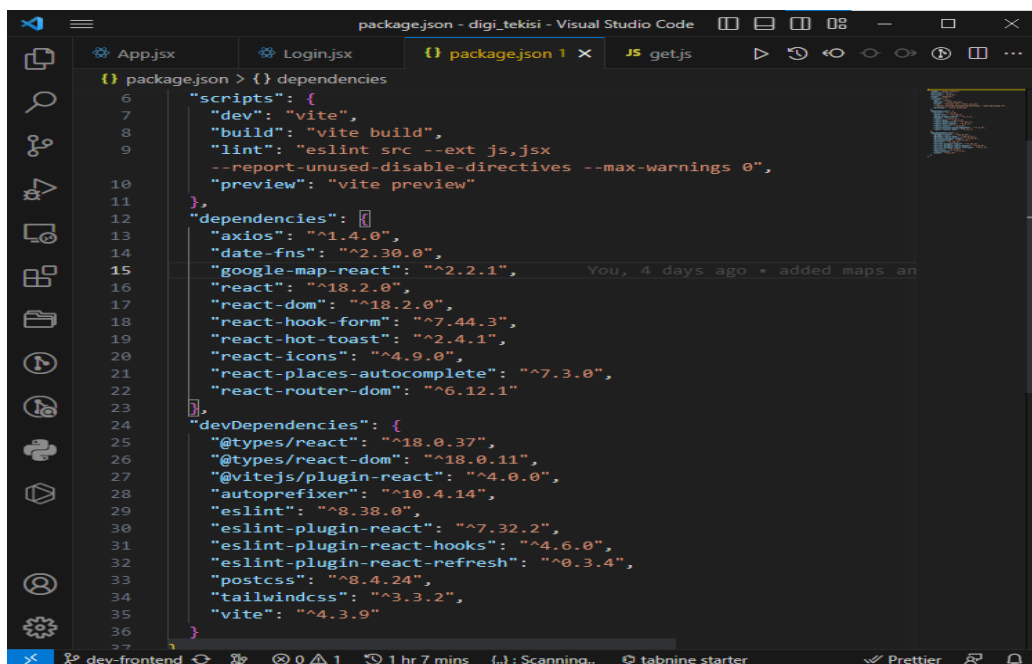


Figure 26: React Dependencies

### 1) React router dom:

Installation: `npm i react-router-dom` or `pnpm add react-router-dom`

React Router DOM is a library for React that simplifies the process of creating navigable pages in a single-page application. It provides a declarative way to define routes and render components based on the current URL. With React Router DOM, you can create a multi-page application experience while still keeping the benefits of a single-page application.

To use React Router DOM, you define routes that correspond to specific components in your application. You can then use the Link component to create links to those routes,

and the Route component to render the appropriate component based on the current URL. React Router DOM also provides other useful components, such as Switch to render the first matching route, and Redirect to redirect to another URL.

Overall, React Router DOM simplifies the process of creating navigation in a React application, providing a flexible and powerful way to define routes and navigate between different pages or views within the application.



```
40
41 <Router>
42   <Routes>
43     <Route path="/" element={<WelcomePage/>} />
44     <Route path="/signup" element={<Signup/>} />
45     <Route path="/login" element={<Login/>} />
46     <Route path="/intro1" element={<WalkTroughPageOne/>} />
47     <Route path="/intro2" element={<WalkTroughPageTwo/>} />
48     <Route path="/intro3" element={<WalkTroughPageThree/>} />
49     <Route path="/validate" element={<ValidationPage/>} />
50     <Route path="/forgot" element={<ForgotPassword/>} />
51     <Route path="/reset" element={<ResetPassword/>} />
52     <Route path="/rides" element={<Rides/>} />
53     <Route path="/register" element={<RegisterDriver/>} />
54     <Route path="/passenger/pay" element={<MakePayment/>} />
55     <Route path="/driver" element={<DriverProfile/>} />
56   </Routes>
57   <Footer/>
58 </Router>
59
60 )
```

Figure 27: Partial Implementation

useNavigate is a newer hook introduced in version 6 of React Router DOM. It provides a way to navigate programmatically by returning a navigate function that you can call with a path or location object. This is useful when you need to trigger a navigation event in response to a certain action or event in your application.

In the image below for example, we used useNavigate to navigate our user to the welcome page after successful login



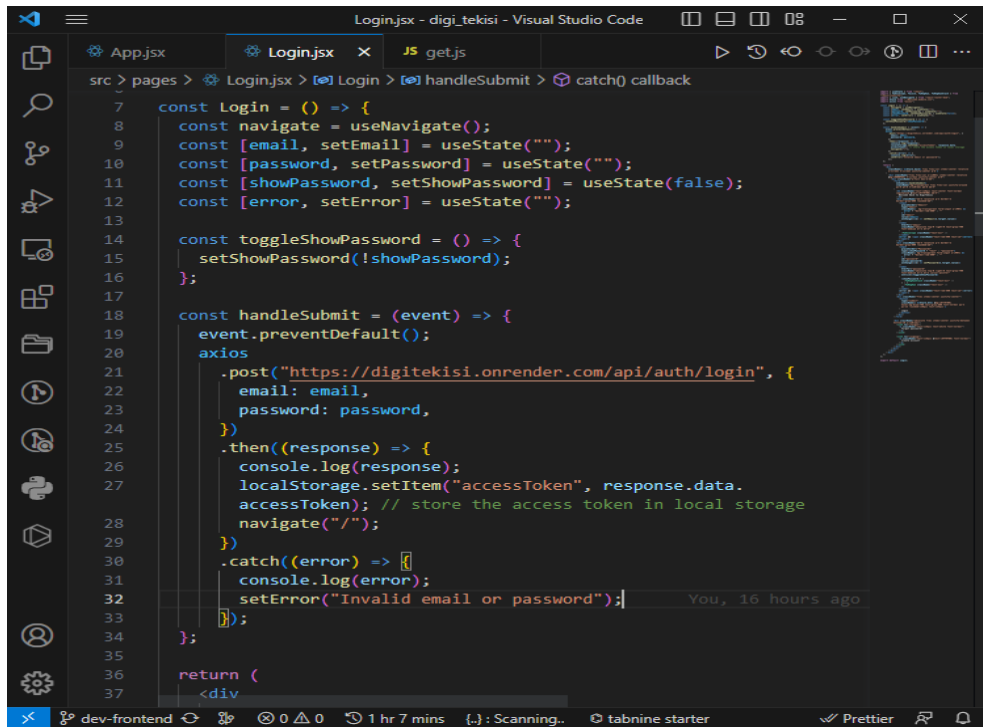


Figure 28: Login Implementation

## 2) Axios:

Installation: `npm i axios` or `pnpm add axios`

Our users can't make much use of our system if they do not create an account. And the passenger can book a ride if there is no driver or no rides. In order to give the users all these facilities and chance of making much use of our system, we added login and signup functionalities which is one of our functional requirements mentioned above. As for the other functional requirements like making a subscription and becoming a driver we had to make use of axios in order to make get , post and put requests to our backend.

Axios is a popular JavaScript library that allows you to make HTTP requests from a web browser or from Node.js. It provides a simple and intuitive API to perform common HTTP methods like GET, POST, PUT, DELETE, etc

The Image below shows a post request made to the backend to login the user into the system. The information it post to the server is the email and the password of the user

```

src > pages > Login.jsx > Login > handleSubmit
4 import style from './welcome.module.css';
5 import axios from 'axios';
6
7 const Login = () => {
8   const navigate = useNavigate();
9   const [email, setEmail] = useState("");
10  const [password, setPassword] = useState("");
11  const [showPassword, setShowPassword] = useState(false);
12  const [error, setError] = useState("");
13
14  const toggleShowPassword = () => {
15    setShowPassword(!showPassword);
16  };
17
18  const handleSubmit = (event) => {
19    event.preventDefault();
20    axios
21      .post("https://digitekisi.onrender.com/api/auth/login", {
22        email: email,
23        password: password,
24      })
25      .then((response) => {
26        console.log(response);
27        localStorage.setItem("accessToken", response.data.accessToken); // store the access token in local storage
28        navigate("/");
29      })
30      .catch((error) => {
31        console.log(error);
32        setError("Invalid email or password");
33      });
34  };
35

```

Figure 29: State Management

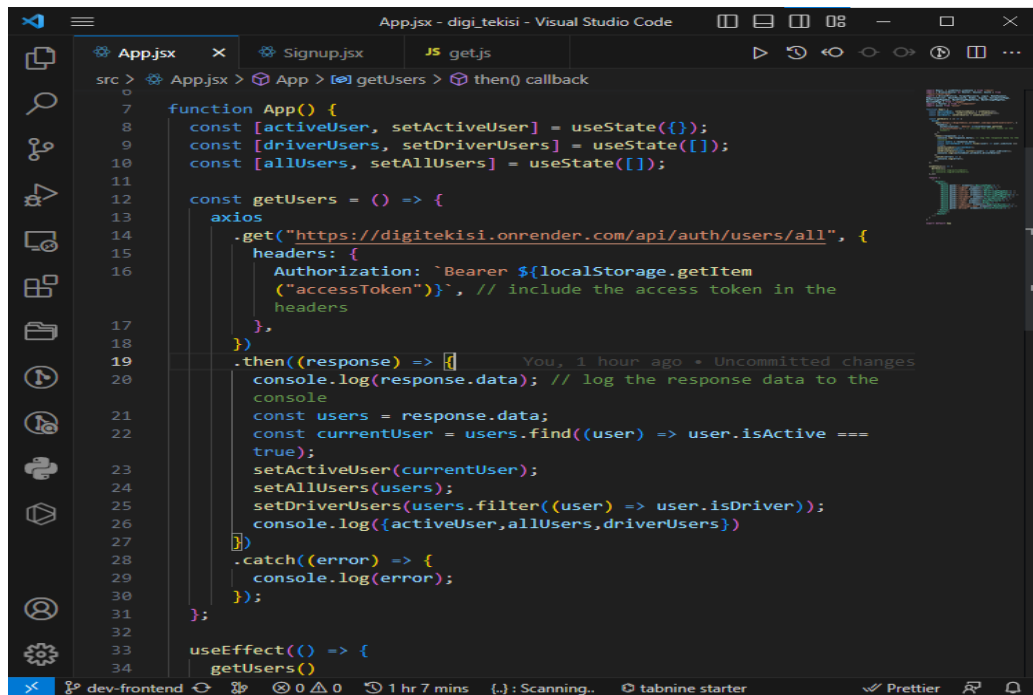
```

src > pages > Signup.jsx > Signup > handleSubmit > catch() callback
5 import axios from "axios";
6
7 const Signup = () => {
8   const navigate = useNavigate();
9
10  const [email, setEmail] = useState("");
11  const [password, setPassword] = useState("");
12  const [showPassword, setShowPassword] = useState(false);
13
14  const toggleShowPassword = () => {
15    setShowPassword(!showPassword);
16  };
17
18  const handleSubmit = (e) => {
19    e.preventDefault();
20    const data = { email, password };
21    axios
22      .post("https://digitekisi.onrender.com/api/auth/signup-start",
23        data)
24      .then((response) => {
25        console.log(response);
26        navigate("/validate"); // navigate to login page on
27        // successful signup
28      })
29      .catch((error) => {
30        console.log(error);
31      });
32  };
33
34  return (
35    <div
36      className={ `${Style.back} w-screen flex flex-col items-center

```

Figure 30: Signup Implementation

The Image below shows a post request made to the backend to create a user account into the system. The information it post to the server is the email and the password of the user. A new record is created in the user table as the user signs up successfully. When there is an error, the user will be prompted by an error.



```
App.jsx - digi_tekisi - Visual Studio Code
src > App.jsx > App > getUsers > then() callback
0
1 function App() {
2   const [activeUser, setActiveUser] = useState({});
3   const [driverUsers, setDriverUsers] = useState([]);
4   const [allUsers, setAllUsers] = useState([]);
5
6   const getUsers = () => {
7     axios
8       .get("https://digitekisi.onrender.com/api/auth/users/all", {
9         headers: {
10           Authorization: `Bearer ${localStorage.getItem("accessToken")}`, // include the access token in the headers
11         },
12       })
13       .then((response) => {
14         console.log(response.data); // log the response data to the console
15         const users = response.data;
16         const currentUser = users.find((user) => user.isActive === true);
17         setActiveUser(currentUser);
18         setAllUsers(users);
19         setDriverUsers(users.filter((user) => user.isDriver));
20         console.log({activeUser, allUsers, driverUsers})
21       })
22       .catch((error) => {
23         console.log(error);
24       });
25   };
26
27   useEffect(() => {
28     getUsers()
29   }, []);
30
31 }
```

Figure 31: Entry point for Application

### 3) Tailwind css

**Installation:** `pnpm add -D autoprefixer postcss tailwind css`

**Setting up:** `npx init tailwind`

## Configuration:

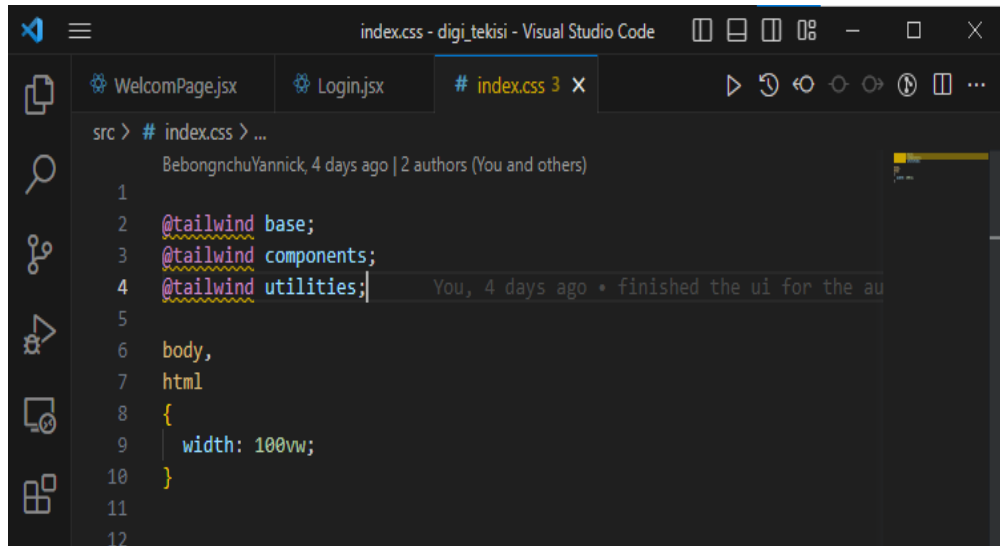


Figure 32: Styling UI

4)

Import this index.css file in the main.jsx file so that the css can be recognized by every file in our application.

Tailwind CSS is a popular utility-first CSS framework that provides a set of predefined styles and classes to help you quickly build custom user interfaces. It allows you to create a consistent and responsive design system for your application without the need for custom CSS.

One of the main advantages of using Tailwind CSS in a React JS application is that it provides a very efficient way to style your components. With Tailwind, you can quickly apply styles to your components by using predefined classes, rather than writing custom CSS. This makes it easier to maintain your code and avoid the need for repetitive styling.

Another advantage of using Tailwind CSS in a React application is that it provides a responsive design system out of the box. Tailwind provides a set of classes that allow you to define styles for different screen sizes, making it easy to create a responsive design that works well on different devices.

Overall, Tailwind CSS can help you create custom and responsive user interfaces for your React JS application quickly and efficiently, while also making the code easier to maintain and update over time.

```

1  import React from 'react'
2  import ReactDOM from 'react-dom/client'
3  import App from './App.jsx'
4  import './index.css'
5
6  ReactDOM.createRoot(document.getElementById('root')).render(
7    <React.StrictMode>
8      <App />
9    </React.StrictMode>,
10 )
11

```

Figure 33: Root of Frontend Implementation

```

1
2
3  import React from "react";
4  import images from "../constant/images";
5  import { FaArrowRight } from "react-icons/fa";
6  import { Link } from "react-router-dom";
7  import Style from "../Welcome.module.css";
8
9  const WelcomPage = () => {
10    return (
11      <div className={`${Style.back} flex flex-col items-center`}>
12        <div className="bg-transparent p-8 text-center">
13          <h1 className="text-5xl text-left font-bold text-white capitalize">
14            Taxi of your dreams
15          </h1>
16        </div>
17        <div className="bg-transparent p-4 text-center">
18          <p className="text-2xl text-left text-white capitalize mt-[-1rem]">
19            Your taxi ride could not be more pleasurable than DigiTekisi
20          </p>
21        </div>
22        <div className="bg-transparent flex justify-center items-center">
23          <img src={images.lambo} alt="car" className="" />
24        </div>
25        <div className="bg-transparent p-8 text-center">
26          <Link to="/intro1">
27            <button
28              className="bg-white hover:bg-[#00BFA5] text-gray-700
29

```

Figure 34: Overview of Styling

## **Google map react**

**Instalation:** pnpm add google-map-react Inorder to use google map and display map in your application correctly, you will need an API\_key which you will create from your google cloud in the console.

## **4.5. DATABASE DESIGN AND IMPLEMENTATION**

### **4.5.1. Introduction**

The database design phase of a software system is the process of creating a database that will store the data for the system. This phase includes three main steps: conceptual design, logical design, and physical design.

Conceptual design is the process of creating a high-level overview of the database. This includes identifying the entities (tables) that will be stored in the database, the relationships between those entities, and the attributes (columns) that will be stored in each entity.

Logical design is the process of translating the conceptual design into a more detailed model. This includes specifying the data types for each attribute, the constraints on the data, and the indexes that will be used to improve performance.

Physical design is the process of creating the physical database. This includes creating the tables, columns, indexes, and relationships in the database management system (DBMS).

The database design phase is an important part of the software development process. A well-designed database can improve the performance, scalability, and security of the software system.

Here are some of the key benefits of good database design:

- Improved performance:

A well-designed database can be accessed and queried more efficiently, which can improve the performance of the software system.

- Increased scalability:

A well-designed database can be scaled to handle more data and users, which can help the software system to grow and meet the needs of its users.

- Enhanced security:

A well-designed database can be protected from unauthorized access, which can help to protect the data and privacy of the software system's users.

Here are some of the key challenges of database design:

**Complexity:**

Database design can be a complex and challenging process. There are many factors to consider, such as the data requirements of the software system, the performance requirements, and the security requirements.

**Change:**

The data requirements of a software system can change over time, which can require changes to the database design.

**Cost:**

The cost of database design can be significant, depending on the complexity of the software system and the database.

Despite the challenges, database design is an important part of the software development process. A well-designed database can improve the performance, scalability, and security of the software system.

Relationship between the different tables

**Passenger and payment relationship**

2. Objective of our system

This is an app that is used by both passengers and drivers, where the passengers use it to specify their

positions at a given time, whereas the drivers use it to locate where the passengers are found in a given

time. This can help both drivers and passengers in the following ways:



- It reduces the amount of time passengers have to wait for a taxi.
- Drivers can optimize fuel consumption as the app will guide them to move to locations where there are more potential custom

## 4.5.2. Conceptual Database Design

### Entity Descriptions

Our system has six main entities which are; **“Passenger”**, **“Driver”**, **“Trip”**, **“Subscription”**, **“Payment”** and **“Zone”**.

The **“Passenger”** entity includes attributes such as “passenger\_id”, “passenger\_name”, “passenger\_email”, “passenger\_contact”, “passenger\_location”.

The **“Driver”** entity includes the attributes; “driver\_id”, “driver\_name”, “driver\_contact”, “license\_number”, “driver\_carNo”, “direction”, “zone\_id”.

The **“Trip”** entity includes the following attributes; “trip\_id”, “passenger\_id”, “driver\_id”, “start\_location”, “end\_location”, “datetime”, “distance”.

The **“Subscription”** entity contains the attributes; “subscription\_id”, “driver\_id”, “subscription\_type”, “amount”, “subscription\_date”, “expiration\_date”.

The **“Payment”** entity has the attributes; “payment\_id”, “trip\_id”, “method”, “amount”.

The **“Zone”** entity has the following attributes; “zone\_id”, “zone\_name”.

### Relationships Description between different Entities

#### Passenger and Payment:

We have a one to many relationship between the passenger and the payment. As a site that will love to generate some revenue , we will love to have our users make use of the system as many times as possible. Our passengers can book as many rides as they want and even on the same day. So in the process of booking a ride many times, our passengers tend to pay every ride and

that makes a one to many relationship between the passenger and the payment tables. So we have a passengerId in every record of the payment table and this shows which passenger made which payment at what time.

### **Payment and Trip:**

Every trip will have just one payment. As a result we will have just a one to one relationship between the trip and the payment. Since each trip takes place at a different time, and can have different passengers. So it makes sense to link a single trip to one payment cause each trip will be different so can not be linked to another payment

### **Passenger and Trip:**

Here we have many to many relationships. Taking into consideration our country Cameroon and how transportation has already been handled here. In a single trip, as Tekisi's have 5 passenger seats, so as a Tekisi company to make profit, they tend to have more than one passenger and less than 5 passengers. So in a single trip we have many trip and we know as human beings are very mobile, they can travel many times to different locations even in a single day. So reason for a many to many relationship

### **Trip and Driver:**

Now a trip can be done by only one car and our system is designed in such a way that when our passengers register, they add a license plate number for their Tekisi. And that tells us that one driver can have one car, as a consequent, We have a one to many relationship between the driver and the trip. Cause a trip is something is a journey or a movement from one cluster to another. So we have one driver having many trips

### **Driver and Zone:**

We have a one to many relationship between the driver and their respective Zone. As one of the objectives of our system was to help drivers save fuel, we decided to make the system in such a way that our drivers can move only within a certain cluster or Zone. The zone will be of a certain geometric surface area. So as a consequent, every zone will have its own group of drivers making the relationship between driver and zone a many to one

### **Driver and Subscription:**

4. In order to generate some revenue for our system to keep running, we decided to add a subscription system so that our drivers will have to subscribe to different subscriptions where they will be charged based on how much of the system they use. Now we have

different subscription plan, like monthly plans, weekly plans, monthly and quarterly plans. A driver can not select 2 different types of users and as a system with many users, we will have different users selecting their particular subscription. So we will have a one to many relationship between the driver and subscription table.

## **4.6. Physical Database Design**

### **Database Management System Selection**

After several researches and brainstorming sessions, we decided to go with the postgresql DBMS for our system.

PostgreSQL is a suitable choice for our passenger positioning system for several reasons: Relational Database Management System (RDBMS): PostgreSQL is a powerful and feature-rich open-source RDBMS that adheres to the relational database model. It provides robust support for managing structured data, relationships, and enforcing data integrity.

Scalability and Performance: PostgreSQL is known for its scalability, allowing you to handle a large number of drivers, passengers, trips, and payments efficiently. It offers various performance optimization techniques, including indexing, query optimization, and parallel processing.

ACID Compliance: PostgreSQL follows the principles of ACID (Atomicity, Consistency, Isolation, Durability), ensuring that your data remains consistent and reliable even in the presence of concurrent transactions or system failures. This is important for maintaining data integrity and preserving the accuracy of the passenger and driver information.

Data Types and Constraints: PostgreSQL provides a wide range of data types and constraints that can suit the needs of your system. It supports standard data types such as integer, string, date/time, and also allows you to define custom data types if necessary. Additionally, it offers various constraints, including primary key, foreign key, not null, unique, and check constraints, which can help enforce data consistency and integrity.

Advanced Features: PostgreSQL offers advanced features like stored procedures, triggers, views, and full-text search capabilities. These features allow you to implement complex business logic, automate tasks, and enhance the functionality of your system.

**Extensibility and Community Support:** PostgreSQL has a vibrant and active community of developers and users who contribute to its ongoing development and provide support. It also offers extensive documentation and resources to help you effectively work with the database.

**Cross-Platform Compatibility:** PostgreSQL is cross-platform and can be deployed on various operating systems, including Windows, macOS, and different Linux distributions. This ensures flexibility and compatibility with your preferred deployment environment.

We utilize supabase for the implementation of our database.

Supabase is a popular open-source alternative to traditional database management systems, offering a range of features that make it suitable for our passenger positioning system. Here are some reasons why Supabase is a good fit:

1. **PostgreSQL Compatibility:** Supabase is built on top of PostgreSQL, which means it provides full compatibility with PostgreSQL features, including advanced querying, data types, and ACID compliance. This allows us to leverage the rich functionality of PostgreSQL while benefiting from the additional features and ease of use provided by Supabase.
2. **Real-time Capabilities:** Supabase includes real-time functionality through its support for WebSockets. This is particularly useful for our passenger positioning system, as it enables instant updates and notifications to be sent to drivers and passengers when relevant events occur, such as new trip requests or driver location updates.
3. **Scalability and Performance:** Supabase is designed to scale horizontally by automatically distributing data across multiple nodes. This ensures that our system can handle increasing workloads and deliver consistent performance even as the user base grows.
4. **Authentication and Authorization:** Supabase offers built-in authentication and authorization mechanisms. This allows you to easily manage user access and permissions, ensuring that only authorized individuals can perform actions such as creating trips, making payments, or accessing sensitive data.
5. **Real-time Database Triggers:** Supabase allows you to define database triggers that automatically execute serverless functions in response to specified events. This feature can be utilized to implement custom business logic, perform data validations, or trigger actions based on specific database events.

6.     **Serverless Functions:** Supabase supports serverless functions, allowing you to run custom code in response to API requests or database events. This feature enables you to extend the functionality of your system by implementing custom APIs, integrating with third-party services, or performing complex calculations or validations.

7.     **Developer-Friendly Tools:** Supabase provides a user-friendly web interface and a rich set of developer tools, including a SQL editor, authentication management, and data management features. This helps streamline the development process, allowing you to focus on building your passenger positioning system rather than managing infrastructure.

8.     **Open Source and Community Support:** Supabase is an open-source project with an active and growing community. This means you can benefit from community-contributed features, bug fixes, and ongoing development. The community also provides support and resources to help you troubleshoot issues and explore best practices.

Considering these reasons, Supabase's PostgreSQL compatibility, real-time capabilities, scalability, authentication features, serverless functions, developer-friendly tools, and active community support make it a suitable choice for implementing our passenger positioning system.

## CHAPTER 5. CONCLUSION

In conclusion, the Digitekisi passenger positioning system presents an innovative solution to enhance the efficiency and effectiveness of the transportation industry. The system leverages advanced technologies such as mobile applications, mapping and navigation services, real-time updates, and effective communication interfaces to improve the passenger experience and streamline the operations of transportation providers.

The importance of accurate passenger tracking, real-time updates, and effective communication between passengers and drivers cannot be overstated in the transportation industry. The implementation of the Digitekisi system addresses these crucial aspects, providing passengers with the ability to track their assigned vehicles, receive real-time updates on arrival times, and communicate with drivers for seamless coordination. This not only improves passenger satisfaction but also enhances safety, reduces waiting times, and increases overall operational efficiency.

The potential benefits of implementing such a system are extensive. The Digitekisi passenger positioning system offers improved visibility and control for transportation providers, allowing them to efficiently manage their fleets, optimize routes, and make data-driven decisions. The system enables accurate monitoring of ride activities, leading to improved resource allocation, reduced fuel consumption, and minimized operational costs. Additionally, the system's communication capabilities foster better communication and trust between passengers and drivers, creating a positive user experience and promoting customer loyalty.

The implications of implementing the Digitekisi system extend beyond operational efficiency. The system contributes to environmental sustainability by optimizing routes and reducing unnecessary mileage, resulting in reduced carbon emissions. It also promotes the adoption of digital technology within the transportation industry, paving the way for future advancements and innovations in the field.

The relevance and significance of the Digitekisi passenger positioning system within the broader research landscape are evident. The system combines elements of transportation management, location-based services, communication technology, and data analytics to create a comprehensive solution that addresses the challenges faced by the transportation industry. Its potential impact reaches beyond individual transportation providers, extending to urban planning, traffic management, and sustainable mobility initiatives.

In summary, the Digitekisi passenger positioning system offers a transformative approach to passenger tracking, real-time updates, and effective communication in the transportation industry. By harnessing the power of technology, the system improves the passenger experience, enhances operational efficiency, and contributes to a more sustainable and connected transportation ecosystem. The successful implementation of the Digitekisi system holds great promise for revolutionizing the way transportation services are delivered, benefiting passengers, drivers, and the entire transportation industry as a whole.