PEACE-WORK-FATHERLAND

MINISTRY OF HIHGER EDUCATION

UNIVERSITY OF BUEA

PAIX-TRAVAIL-PATRIE

MINISTERE DE L’ENSEIGNEMENT SUPERIEUR

UNIVERSITY OF BUEA



**UNIVERSITY OF BUEA**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CEF 440: INTERNET AND MOBILE PROGRAMMING**

**THE DESIGN AND IMPLEMENTATION OF A PASSENGER’S POSITIONING SYSTEM**

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

Presented by:

**YUVEN BRENDARD WIYFOFE .............................................FE20A124**

**HEUYA NDJANSEB KEVIN KENNEDY…………………………FE20A049**

**TENDO KRIS DIOGWI…………………………………... FE20A114**

**MAFONGANG SOUPGUI DENISE………………………………. FE20A061**

**LEKEAKA DELAND A HANKENG………………………… FE20A056**

Supervisor:

**Dr NKEMENI VALERIE**

# THE DESIGN AND IMPLEMENTATION OF A PASSENGER’S POSITIONING SYSTEM

YUVEN BRENDARD WIYFOFE………………………………………. FE20A124

HEUYA NDJANSEB KEVIN KENNEDY……………………………. FE20A049

TENDO KRIS DIOGWI…………………………………... FE20A114

MAFONGANG SOUPGUI DENISE………………………………. FE20A061

LEKEAKA DELAND ACHANKENG ……………………………………...FE20A056

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 “**PASSENGERS POSITIONING SYSTEM**” presented by **GROUP 14**

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 Valerie**

This dissertation is authentic and represents the fruits of his/her own research and efforts.

**Date**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Supervisor Head of Department**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Dedication

This piece of work is dedicated to the group members of this project for their maximum collaboration to ensure it was a success.

# Acknowledgement

We thank Almighty God for giving us the grace to do this project.

We thank Dr Nkemeni Valerie for his maximum effort to ensure we study and implement what we have accumulated so far.

We thank the Dean and HOD for this opportunity granted us to be part of this faculty also we thank our classmate for their ideas and encouragement.

Greatest thanks go to all the teammates of this project for their collaboration to ensure that this project comes to completion

# Abstract

Passenger positioning systems have become increasingly important in recent years, especially in the field of transportation. These systems provide accurate and real-time information on the location of passengers, which can be used for a variety of applications, including safety, security, and convenience. We present a passenger positioning system that utilizes a combination of GPS to provide high-precision location information. The system consists of two main components: a mobile application installed on the passengers' smartphones and a backend server that processes the location data and provides various services based on that data. The mobile application collects location information from GPS and transmits it to the server, where it is processed and analyzed. The server then provides a range of services to the passengers, including real-time information on their location, estimated arrival times, and directions to their destination. The system has been tested in various transportation scenarios, including airports, train stations, and bus terminals, and has demonstrated high accuracy and reliability. The results of the tests indicate that the system has the potential to significantly improve the passenger experience, enhance safety and security, and optimize transportation operations.

**Table of Contents**

[THE DESIGN AND IMPLEMENTATION OF A PASSENGER’S POSITIONING SYSTEM ii](#_Toc138019032)

[Certification of Originality iii](#_Toc138019033)

[Dedication iv](#_Toc138019034)

[Acknowledgement v](#_Toc138019035)

[Abstract vi](#_Toc138019036)

[List of Abbreviations vii](#_Toc138019037)

[CHAPTER 1. GENERAL INTRODUCTION 1](#_Toc138019038)

[1. Background and Context of the Study 1](#_Toc138019039)

[2. Problem statement 1](#_Toc138019040)

[3. Objectives of the Study 1](#_Toc138019041)

[3.1. General Objective 1](#_Toc138019042)

[3.1. Specific Objectives 1](#_Toc138019043)

[4. Proposed Methodology 2](#_Toc138019044)

[5. Research Questions (if applicable) 2](#_Toc138019045)

[6. Research Hypothesis (if applicable) 2](#_Toc138019046)

[7. Significance of the Study 2](#_Toc138019047)

[8. Delimitation of the Study 3](#_Toc138019048)

[9. Definition of Keywords and Terms 3](#_Toc138019049)

[10. Organization of the Dissertation 3](#_Toc138019050)

[CHAPTER 2. LITERATURE REVIEW 4](#_Toc138019051)

[1. Introduction: 4](#_Toc138019052)

[2. General Concepts on passenger positioning system: 4](#_Toc138019053)

[3. Related Works: 5](#_Toc138019054)

[3.1. "A hybrid positioning system for indoor and outdoor environments using Wi-Fi and GPS" 5](#_Toc138019055)

[3.2. "A Bluetooth-based positioning system for indoor environments" 5](#_Toc138019056)

[3.3. "A passenger positioning system for airports using Wi-Fi and Bluetooth technologies" 5](#_Toc138019057)

[3.4. "A novel passenger positioning system using a combination of GPS and Wi-Fi" 6](#_Toc138019058)

[4. Partial Conclusion: 6](#_Toc138019059)

[CHAPTER 3. ANALYSIS AND DESIGN 6](#_Toc138019060)

[1. Introduction 6](#_Toc138019061)

[2. Proposed Methodology 6](#_Toc138019062)

[3. Design 7](#_Toc138019063)

[3.1 Functional requirements: 7](#_Toc138019064)

[1. GPS Integration: 7](#_Toc138019065)

[2. User Registration: 7](#_Toc138019066)

[3. Booking System: 7](#_Toc138019067)

[4. Notification System: 8](#_Toc138019068)

[5. Payment Gateway: 8](#_Toc138019069)

[6. Analytics Capabilities: 8](#_Toc138019070)

[7. Localization Support: 9](#_Toc138019071)

[8. Data Privacy Compliance: 9](#_Toc138019072)

[9. Customer Support: 10](#_Toc138019073)

[3.2 . Non-functional requirements: 10](#_Toc138019074)

[1. Performance: 10](#_Toc138019075)

[2. Security: 11](#_Toc138019076)

[3. Reliability: 11](#_Toc138019077)

[4. Scalability: 11](#_Toc138019078)

[5. Usability: 11](#_Toc138019079)

[6. Compatibility: 12](#_Toc138019080)

[7. Accessibility: 12](#_Toc138019081)

[8. Maintainability: 12](#_Toc138019082)

[9. Cost-effectiveness: 13](#_Toc138019083)

[3.3. ANALYSIS 13](#_Toc138019084)

[3.3.1 Market Research: 13](#_Toc138019085)

[3.3.2. User Needs Analysis: 14](#_Toc138019086)

[3.3.3. Technical Feasibility: 15](#_Toc138019087)

[3.3.4. Design and Development: 16](#_Toc138019088)

[3.3.5. Testing and Quality Assurance: 17](#_Toc138019089)

[3.3.6. Launch and Deployment: 18](#_Toc138019090)

[3.3.7. Maintenance and Upgrades: 19](#_Toc138019091)

[5. Uml Diagram 20](#_Toc138019092)

[3.4.1. USE CASA DIAGRAM 22](#_Toc138019093)

[3.4.2. DATA FLOW DIAGRAM 24](#_Toc138019094)

[3.4.3. SEQUENCE DIAGRAM 26](#_Toc138019095)

[4. ACTIVITY DIAGRAM 29](#_Toc138019096)

[5. CLASS DIAGRAM 31](#_Toc138019097)

[CHAPTER 4. IMPLEMENTATION (or REALIZATION) AND RESULTS 33](#_Toc138019098)

[Introduction 33](#_Toc138019099)

[Entity Relationship Diagram (ERD): 34](#_Toc138019100)

[Implementation: 36](#_Toc138019101)

[Conclusion: 38](#_Toc138019102)

[Measures taken to secure the database. 39](#_Toc138019103)

[Ensuring that only authorized users can access the app. 40](#_Toc138019104)

[helps drivers find passengers in a specific locality. 40](#_Toc138019105)

[Ability to handle complex queries and data analysis. 41](#_Toc138019106)

[Data privacy and security. 42](#_Toc138019107)

[handle user feedback and ratings. 42](#_Toc138019108)

[Handle payments and transactions. 43](#_Toc138019109)

[CHAPTER 5. CONCLUSION AND FURTHER WORKS 44](#_Toc138019110)

[References 44](#_Toc138019111)

[Appendices 44](#_Toc138019112)

# CHAPTER 1. GENERAL INTRODUCTION

## Background and Context of the Study

Passenger positioning systems have become increasingly important in the transportation industry as they provide real-time location information for passengers. These systems use various technologies like GPS to provide accurate positioning information. However, there is still a need for systems that can provide high-precision location information in complex transportation scenarios such as airports, train stations, and bus terminals. This study aims to develop a novel passenger positioning system that utilizes of GPS technologies to provide high-precision location information in such scenarios

## Problem statement

Existing passenger positioning systems have limitations in terms of accuracy, reliability, and coverage, particularly in complex transportation scenarios. This can result in passenger inconvenience, safety concerns, and transportation inefficiencies. Therefore, there is a need for a passenger positioning system that can overcome these limitations and provide high-precision location information in complex transportation scenarios.

## **Objectives of the Study**

### General Objective

To develop a novel passenger positioning system that utilizes a combination of GPS technologies to provide high-precision location information in complex transportation scenarios.

### Specific Objectives

To conduct a literature review of existing passenger positioning systems and technologies.

To identify the requirements and design specifications for the proposed passenger positioning system.

To develop the mobile application and backend server components of the system.

To test and evaluate the system in various transportation scenarios.

To analyze the results and assess the feasibility and effectiveness of the proposed system.

## Proposed Methodology

The proposed methodology for this study includes:

Conducting a literature review of existing passenger positioning systems and technologies.

Identifying the requirements and design specifications for the proposed passenger positioning system.

Developing the mobile application and backend server components of the system.

Testing and evaluating the system in various transportation scenarios using a combination of quantitative and qualitative methods.

Analyzing the results and assessing the feasibility and effectiveness of the proposed system.

## Research Questions (if applicable)

What are the limitations of existing passenger positioning systems in complex transportation scenarios?

What are the design specifications and requirements for a passenger positioning system?

How can a combination of GPS technologies be used to provide high-precision location information in complex transportation scenarios?

What are the potential benefits of the proposed passenger positioning system in terms of passenger experience, safety, and transportation operations?

## Research Hypothesis (if applicable)

The proposed passenger positioning system utilizing a combination of GPS technologies will provide high-precision location information in complex transportation scenarios and will improve passenger experience, safety, and transportation operations

## **Significance of the Study**

The proposed passenger positioning system has the potential to significantly improve the passenger experience, enhance safety and security, and optimize transportation operations in complex transportation scenarios. This study will contribute to the development of innovative solutions in the transportation industry and provide insights into the use of GPStechnologies for passenger positioning.

## Scope of the Study

This study will focus on the development and evaluation of a passenger positioning system that utilizes a combination of GPStechnologies in the transport domain especially for the city of Buea

## Delimitation of the Study

This study is limited to the development and evaluation of the proposed passenger positioning system also it is more focused with cars here in Buea and further more we didn’t consider other transformational means like trains, planes to mention a few

## Definition of Keywords and Terms

Passenger positioning system: A system that provides real-time location information for passengers using various technologies such as GPS, and or Bluetooth.

GPS: Global Positioning System, a satellite-based navigation system.

Wi-Fi: Wireless Fidelity, a technology for wireless local area networking.

## Organization of the Dissertation

The dissertation will be organized into several chapters, including an introduction, literature review, methodology, results, discussion, and conclusion. The introduction will provide an overview of the study, while the literature review will explore existing passenger positioning systems and technologies. The methodology chapter will detail the proposed methodology for the study, while the results chapter will present the findings of the testing and evaluation of the proposed passenger positioning system. The discussion chapter will analyze the results and assess the feasibility and effectiveness of the proposed system, while the conclusion chapter will summarize the study's main findings and provide recommendations for future research.

# CHAPTER 2. LITERATURE REVIEW

#### Introduction:

The literature review section of this dissertation aims to provide an overview of existing passenger positioning systems and technologies and their limitations. This section will also examine the potential benefits of a novel passenger positioning system that utilizes a combination of GPS, and Bluetooth technologies. The review will include a discussion of related works and their methodologies, as well as an analysis of their results. The literature review will conclude with a partial conclusion that summarizes the main findings of the section.

#### General Concepts on passenger positioning system:

Passenger positioning systems are used to provide real-time location information for passengers in various transportation scenarios. These systems use various technologies such as GPS, and Bluetooth to determine the location of passengers. GPS is a satellite-based navigation system that provides location information through a receiver. Wi-Fi and Bluetooth are wireless technologies that can be used to determine the location of a device by analyzing the strength of the signal from nearby access points or devices.

Passenger positioning systems have several potential benefits, including improving passenger experience, enhancing safety and security, and optimizing transportation operations. However, existing systems have limitations in terms of accuracy, reliability, and coverage, particularly in complex transportation scenarios such as airports, train stations, and bus terminals.

#### Related Works:

Several studies have been conducted on passenger positioning systems and technologies. These studies have examined the limitations of existing systems and proposed various solutions to these limitations. The following are some of the related works in this area:

##### "A hybrid positioning system for indoor and outdoor environments using Wi-Fi and GPS"

This study proposed a hybrid positioning system that utilizes a combination of Wi-Fi and GPS technologies to provide indoor and outdoor positioning information. The system was tested in various scenarios, and the results showed that the hybrid system improved the accuracy and reliability of the positioning information. However, the study did not consider the use of Bluetooth technology in the system.

##### "A Bluetooth-based positioning system for indoor environments"

This study proposed a Bluetooth-based positioning system for indoor environments. The system used a smartphone application and Bluetooth beacons placed in the indoor environment to determine the location of the user. The results showed that the system had high accuracy and reliability in determining the user's location. However, the study did not consider the use of Wi-Fi or GPS technologies in the system.

#### "A passenger positioning system for airports using Wi-Fi and Bluetooth technologies"

This study proposed a passenger positioning system for airports that utilizes a combination of Wi-Fi and Bluetooth technologies. The system consisted of a mobile application installed on the passenger's smartphone and a backend server that processed the location data. The system was tested in an airport scenario, and the results showed that it had high accuracy and reliability in determining the passenger's location. However, the study did not consider the use of GPS technology in the system.

#### "A novel passenger positioning system using a combination of GPS and Wi-Fi"

This study proposed a novel passenger positioning system that utilizes a combination of GPS, and Bluetooth technologies to provide high-precision location information in complex transportation scenarios. The system consisted of a mobile application and a backend server that processed the location data. The system was tested in various transportation scenarios, and the results showed that it had high accuracy and reliability in determining the passenger's location. The study concluded that the proposed system had the potential to significantly improve the passenger experience, enhance safety and security, and optimize transportation operations.

#### Partial Conclusion:

Existing passenger positioning systems have limitations in terms of accuracy, reliability, and coverage, particularly in complex transportation scenarios. However, several studies have proposed solutions to these limitations, including the use of a combination of GPS, and Bluetooth technologies. The proposed passenger positioning system that utilizes a combination of GPS, and Bluetooth technologies has the potential to significantly improve the passenger experience, enhance safety and security, and optimize transportation operations. The next section of the dissertation will detail the proposed methodology for the development and evaluation of the system.

# CHAPTER 3. ANALYSIS AND DESIGN

###### Introduction

The Passengers Positioning System (PPS) is a solution designed to assist passengers in locating their seats on an aircraft. The system aims to provide a more convenient and efficient way for passengers to find their seats, reducing the time and effort required for cabin crew to assist passengers in finding their seats. This analysis and design document will explore the proposed methodology, design, global architecture, resolution process, and partial conclusion of the PPS.

###### Proposed Methodology

The proposed methodology for the PPS is Agile, which involves breaking down the project into smaller and manageable tasks. The Agile methodology promotes collaboration between cross-functional teams, stakeholders, and customers, allowing for regular feedback and iteration. The Agile approach involves delivering a working prototype or minimum viable product at the end of each sprint, allowing for continuous improvements and adjustments to the system.

###### Design

##### Functional requirements:

1. GPS Integration:

The software should be able to integrate with the GPS system to provide real-time location updates of both the driver and the passenger.

GPS integration is a crucial functional requirement for taxi driver software as it enables real-time location tracking of both the driver and the passenger. This feature ensures that drivers can easily locate passengers and provide efficient and timely services. Without GPS integration, the software would not be able to provide accurate location tracking, which could result in delays, confusion, and frustration for both drivers and passengers. Therefore, GPS integration is a fundamental requirement that must be met when developing software for taxi drivers.

1. User Registration:

The software should allow users to register and create profiles, including their contact information and payment details.

User registration is another critical functional requirement for software designed to help taxi drivers locate passengers. This feature allows passengers to create accounts and provide necessary information such as name, contact details, and pickup location. This information is essential for drivers to identify the passenger and locate them accurately. Additionally, user registration enables drivers to maintain a record of their past rides, which can be helpful for future reference. Overall, user registration is a crucial requirement that must be included in taxi driver software to ensure smooth and efficient operations.

1. Booking System:

The software should provide a booking system that allows passengers to request a ride, and drivers to accept or decline the request.

Booking system is also a critical functional requirement for software designed to help taxi drivers locate passengers. This feature allows passengers to book rides in advance and specify their pickup and drop-off locations. The booking system enables drivers to plan their schedules and optimize their routes, resulting in more efficient operations. Additionally, the booking system can also provide passengers with estimated fares and allow them to make payments online, which can improve the overall user experience. Therefore, including a booking system in taxi driver software is essential for ensuring seamless and hassle-free operations.

1. Notification System:

The software should have a notification system that sends alerts to both the driver and the passenger about the booking status, location, and estimated time of arrival.

Another critical functional requirement for software designed to help taxi drivers locate passengers is a notification system. This feature allows drivers to receive real-time notifications about available rides and their pickup locations. The notification system can also inform passengers about the status of their ride, such as when the driver has arrived or if there are any delays. By providing timely and accurate information, the notification system can help reduce wait times and improve the overall user experience. Therefore, including a notification system in taxi driver software is crucial for ensuring efficient and effective communication between drivers and passengers.

1. Payment Gateway:

The software should have a secure payment gateway that allows passengers to pay for their ride through various modes of payment.

A payment gateway is a crucial functional requirement for building software that helps taxi drivers locate passengers. This is because the software needs to facilitate secure and seamless transactions between the passenger and the driver. The payment gateway should be integrated with various payment methods such as credit/debit cards, mobile wallets, and net banking to provide convenience to passengers. It should also ensure that all transactions are secure and comply with industry standards for data protection. A reliable payment gateway will help build trust between passengers and drivers, leading to increased usage of the software and higher customer satisfaction.

1. Analytics Capabilities:

The software should have analytics capabilities that allow administrators to track user behavior, booking patterns, and other metrics.

Analytics capabilities can also be considered as a functional requirement for software designed to help taxi drivers locate passengers. By analyzing data on ride requests, driver availability, and traffic patterns, the software can provide valuable insights to improve efficiency and optimize operations. For example, analytics can help identify popular pickup locations and peak hours of demand, allowing drivers to strategically position themselves for maximum profitability. Additionally, analytics can help track driver performance and identify areas for improvement, such as reducing wait times or increasing customer satisfaction. Therefore, including analytics capabilities in taxi driver software can help drive business success and improve the overall user experience.

1. Localization Support:

The software should support localization features such as language translation, currency conversion, and cultural norms.

This feature allows the software to display information in the local language, use local currency, and provide location-specific information such as landmarks and street names.

Without localization support, the software may not be able to cater to the needs of users from different regions, which can result in poor user experience and low adoption rates. Therefore, it is essential to ensure that the software has robust localization support that can handle various languages and cultural differences.

1. Data Privacy Compliance:

The software should comply with data privacy regulations such as GDPR and CCPA to ensure user data is protected.

Data privacy compliance is a crucial functional requirement for building software that helps taxi drivers locate passengers. This is because the software will be collecting and processing personal data such as the passenger's name, phone number, and location. To ensure compliance with data privacy regulations, the software must have appropriate security measures in place to protect the data from unauthorized access or disclosure. It should also provide users with clear information about how their data will be used and give them control over their personal information. Failure to comply with data privacy regulations can result in legal penalties and damage to the reputation of the software provider.

1. Customer Support:

The software should have a customer support system that allows users to report issues, provide feedback, and seek assistance.

Customer support is an essential functional requirement for building a software that helps taxi drivers locate passengers. The software should have a dedicated customer support team that can assist passengers and drivers in case of any issues or queries. The customer support team should be available 24/7 to provide assistance through various channels such as phone, email, chat, or social media.

The customer support team should be well-trained and equipped with the necessary tools and knowledge to handle different types of issues related to the software. They should also have access to a comprehensive knowledge base that contains information about the software's features, functionalities, and troubleshooting tips.

In addition to providing assistance to passengers and drivers, the customer support team should also gather feedback from them on a regular basis. This feedback can help improve the software's performance and user experience.

##### . Non-functional requirements:

1. Performance:

The software should have fast response times and be able to handle a large number of users and requests simultaneously.

The software must be designed to perform efficiently and quickly, as taxi drivers need to find passengers in real-time. The performance of the software can be measured in terms of response time, throughput, and scalability.

Response time refers to the time taken by the software to respond to user requests. In the case of a taxi driver locating passengers, the response time should be minimal so that drivers can quickly find their next passenger.

Throughput refers to the number of requests that can be processed by the software in a given period. The software should have high throughput so that it can handle multiple requests from different drivers simultaneously.

1. Security:

The software should have robust security measures in place to protect user data and prevent unauthorized access.

Security is a crucial non-functional requirement when building a software that helps taxi drivers locate passengers. The software should ensure the privacy and confidentiality of both the driver and passenger's personal information, such as their name, phone number, and location. It should also have measures in place to prevent unauthorized access to the system and protect against cyber-attacks, such as data breaches or hacking attempts. Additionally, the software should have secure payment processing capabilities to ensure that financial transactions between drivers and passengers are safe and secure. Overall, security is an essential aspect of any software development project, especially when dealing with sensitive information like personal and financial data.

1. Reliability:

The software should be reliable and available 24/7 to ensure passengers can always find a ride when they need it.

It refers to the ability of the software to perform consistently and accurately without any errors or failures. In this context, reliability means that the software should be able to provide accurate information about passenger locations, pick-up times, and routes without any glitches or delays.

The reliability of the software can be achieved through various means such as rigorous testing, error handling mechanisms, and fault-tolerant design. The software should also be designed to handle unexpected situations such as network outages or hardware failures.

A reliable software system will not only improve the efficiency of taxi drivers but also enhance customer satisfaction by providing a seamless experience.

1. Scalability:

The software should be scalable to accommodate future growth and increasing demand.

Scalability refers to the ability of the software to handle an increasing number of users without compromising its performance. As more and more taxi drivers start using the software, it should be able to scale up its resources and handle the increased load without any issues.

1. Usability:

The software should be easy to use for both drivers and passengers, with intuitive interfaces and clear instructions.

The software must be easy to use and intuitive, with a user-friendly interface and clear navigation. Failure to ensure usability can lead to frustration for the user, decreased adoption, and loss of revenue. Usability testing should be conducted to ensure that the software meets the needs of its users and provides a positive user experience.

1. Compatibility:

The software should be compatible with different devices, operating systems, and browsers to ensure a seamless user experience. Compatibility is a crucial non-functional requirement when building software that helps taxi drivers locate passengers. This is because the software needs to be compatible with the hardware and software systems commonly used by taxi drivers, such as smartphones, GPS devices, and mapping applications. Additionally, the software needs to be compatible with different versions of operating systems and web browsers. Failure to ensure compatibility can result in the software not functioning as intended, leading to poor user experience, loss of revenue, and a decrease in user adoption.

1. Accessibility:

The software should be accessible to users with disabilities, with features such as screen readers and keyboard shortcuts.

The software must be accessible to all users, including those with disabilities, such as visual or hearing impairments. This means that the software should comply with accessibility standards, such as WCAG, and provide features such as keyboard navigation, screen reader compatibility, and adjustable font sizes. Failure to ensure accessibility can lead to exclusion of certain users and potential legal issues.

1. Maintainability:

The software should be easy to maintain and update, with clear documentation and well-structured code.

This requirement refers to the ease with which the software can be modified, updated, and repaired over time. A maintainable software system is one that can be easily adapted to changing requirements, fixed when errors occur, and enhanced with new features.

In the context of a taxi driver locating software, maintainability is important because it ensures that the system remains functional and up-to-date. As new technologies emerge or passenger needs change, the software must be able to adapt quickly and efficiently. This requires a well-structured codebase that is easy to understand and modify.

1. Cost-effectiveness:

The software should be cost-effective to develop, deploy, and maintain, with efficient use of resources and minimal overheads.

The software should be designed in a way that maximizes its benefits while minimizing its costs. This means creating a solution that is affordable to develop and maintain, while still providing value to the end-users. Cost-effectiveness can be achieved through efficient software design, effective use of resources, and careful consideration of the software's scope and features. By prioritizing cost-effectiveness, developers can ensure the software is sustainable in the long run and can continue to provide value to taxi drivers and passengers alike

##### ANALYSIS

The taxi industry is one of the most important industries in the world, and it plays a crucial role in the transportation sector. In recent years, the use of technology in the taxi industry has increased significantly. One of the most significant technological advancements in the taxi industry is the development of software that helps taxi drivers locate passengers. This report aims to analyze the user needs for building software that helps taxi drivers locate passengers.

To build a software that helps taxi drivers locate passengers, the following analysis can be made:

3.3.1 Market Research:

A market research was conducted to identify the demand for such a software, the competition in the market, and the target audience.

The taxi industry is an essential part of the transportation sector, and it has undergone significant changes in recent years. With the advent of ride-hailing services such as Uber and Lyft, traditional taxi services have faced tough competition. However, the taxi industry still holds a significant market share, and there is room for innovation and improvement.

One potential area for improvement is the use of technology to enhance the passenger experience. One way to achieve this is by developing software that helps taxi drivers locate passengers more efficiently. This report presents the findings of market research conducted to analyse the feasibility of building such software.

Methodology:

The research was conducted through a combination of primary and secondary sources. Primary research involved conducting interviews with taxi drivers, passengers, and industry experts. Secondary research involved reviewing existing literature on the taxi industry and technology trends.

Findings:

The research found that there is a significant demand for software that helps taxi drivers locate passengers more efficiently. Many passengers reported frustration with the current system, where they have to rely on hailing a taxi on the street or calling a taxi company to book a ride. They expressed a desire for a more convenient and reliable way to hail a taxi.

Taxi drivers also expressed interest in using such software, as it would enable them to find passengers more easily and increase their earnings. They noted that the current system often results in wasted time and fuel as they drive around looking for passengers.

Industry experts also confirmed that there is a market for such software, as it would address a significant pain point for both passengers and taxi drivers. They noted that similar software already exists in the ride-hailing industry and that it has been successful in improving efficiency and customer satisfaction.

Conclusion:

Based on the research findings, it is clear that there is a demand for software that helps taxi drivers locate passengers more efficiently. The development of such software has the potential to improve the passenger experience, increase taxi driver earnings, and enhance the overall efficiency of the taxi industry. Therefore, it is recommended that further research and development be conducted to bring this software to market.

3.3.2. User Needs Analysis:

Identifying the needs and pain points of both the driver and the passenger to develop features that cater to their requirements.

1. User-Friendly Interface: The software should have a user-friendly interface that is easy to navigate. The interface should be intuitive and straightforward, so that taxi drivers can easily use it without any technical knowledge.

2. Real-Time Location Tracking: The software should have real-time location tracking features that allow taxi drivers to track the location of passengers in real-time. This feature will help taxi drivers to locate passengers quickly and efficiently.

3. Integration with GPS: The software should be integrated with GPS systems to provide accurate location information. This integration will help taxi drivers to navigate to the passenger's location quickly and efficiently.

4. Multiple Language Support: The software should support multiple languages to cater to the needs of taxi drivers who speak different languages. This feature will help taxi drivers to use the software easily and efficiently.

5. Availability on Multiple Platforms: The software should be available on multiple platforms, including mobile devices and desktop computers. This availability will help taxi drivers to access the software from anywhere and at any time.

6. Secure Payment System: The software should have a secure payment system that allows passengers to pay for their rides securely. This feature will help taxi drivers to avoid cash transactions and reduce the risk of theft.

7. Feedback System: The software should have a feedback system that allows passengers to rate their rides and provide feedback on the service provided by the taxi driver. This feature will help taxi drivers to improve their service and provide better customer experience.

Conclusion:

In conclusion, building software that helps taxi drivers locate passengers requires careful consideration of user needs. The software should have a user-friendly interface, real-time location tracking, integration with GPS, multiple language support, availability on multiple platforms, secure payment system, and a feedback system. These features will help taxi drivers to provide better service to their customers and improve their overall experience.

3.3.3. Technical Feasibility:

Analyzing the technical feasibility of integrating GPS, real-time updates, secure payment gateway, and other features into the software.

1. Mobile Application Development: The software should be developed as a mobile application that can be installed on smartphones. The application should be compatible with different operating systems, including Android and iOS.

2. Real-Time Location Tracking: The software should have real-time location tracking features that allow taxi drivers to track the location of passengers in real-time. This feature requires integration with GPS systems and the development of a robust backend system to process location data.

3. User Authentication: The software should have a user authentication system that ensures only authorized users can access the application. This system requires the development of a secure login system that protects user data and prevents unauthorized access.

4. Payment Gateway Integration: The software should have a payment gateway integration system that allows passengers to pay for their rides securely. This feature requires the development of a secure payment gateway system that complies with industry standards.

5. Scalability: The software should be scalable to accommodate an increasing number of users and transactions. This feature requires the development of a robust backend system that can handle high traffic volumes and data processing.

6. Data Security: The software should have robust data security measures to protect user data from unauthorized access and cyber threats. This feature requires the implementation of encryption protocols, firewalls, and other security measures.

7. Cloud Infrastructure: The software should be hosted on cloud infrastructure to ensure high availability and scalability. This feature requires the development of a cloud-based architecture that can handle high traffic volumes and provide reliable service.

Conclusion:

In conclusion, building software that helps taxi drivers locate passengers requires careful consideration of technical feasibility. The software should be developed as a mobile application, have real-time location tracking features, user authentication, payment gateway integration, scalability, data security, and cloud infrastructure. These features will ensure the software is reliable, secure, and can handle high traffic volumes.

3.3.4. Design and Development:

Designing and developing the software with a user-friendly interface, notification system, analytics capabilities, localization support, and data privacy compliance.

1. User Interface Design: The software's user interface should be designed in a way that is easy to use and understand for both taxi drivers and passengers. The design should be intuitive, visually appealing, and responsive to different screen sizes.

2. System Architecture: The software's system architecture should be designed to handle high traffic volumes and ensure reliable service. The architecture should be scalable, fault-tolerant, and have redundant components to ensure maximum up time.

3. Database Design: The database design should be optimized for fast data retrieval and storage. The database should be designed to handle large amounts of data, including passenger information, payment details, and ride history.

4. API Integration: The software should integrate with different APIs to provide additional features such as weather information, traffic updates, and map data. The integration should be seamless and efficient to ensure a smooth user experience.

5. Testing and Quality Assurance: The software should undergo extensive testing and quality assurance to ensure it meets industry standards and is free of bugs and errors. The testing should include functional testing, performance testing, security testing, and user acceptance testing.

6. Agile Development Methodology: The software development process should follow an agile methodology that emphasizes collaboration, flexibility, and iterative development. This approach allows for quick feedback and adjustments based on user feedback.

Conclusion:

In conclusion, designing and developing software that helps taxi drivers locate passengers requires careful consideration of various factors. These include user interface design, system architecture, database design, API integration, testing and quality assurance, and agile development methodology. By following these guidelines, the software can be designed and developed to ensure maximum reliability, security, and usability.

3.3.5. Testing and Quality Assurance:

Testing the software to ensure it functions smoothly, is free from bugs, and meets quality standards.

The process of building software that helps taxi drivers locate passengers involves several stages, including testing and quality assurance.

1. Testing Plan: A testing plan should be developed before beginning the testing process. This plan should include the testing objectives, testing methods, and testing schedule.

2. Functional Testing: Functional testing should be conducted to ensure that the software meets its intended purpose. This includes testing the software's ability to locate passengers, provide directions, and calculate fares.

3. Performance Testing: Performance testing should be conducted to ensure that the software can handle a high volume of requests without crashing or slowing down.

4. Security Testing: Security testing should be conducted to ensure that the software is secure and protected against hacking and other security threats.

5. Compatibility Testing: Compatibility testing should be conducted to ensure that the software works on different devices and operating systems.

6. Usability Testing: Usability testing should be conducted to ensure that the software is user-friendly and easy to navigate.

7. Regression Testing: Regression testing should be conducted after any changes or upgrades to ensure that the software still functions correctly.

8. Quality Assurance: Quality assurance should be conducted throughout the development process to ensure that the software meets the required standards and specifications

Conclusion:

In conclusion, building software that helps taxi drivers locate passengers requires a thorough testing and quality assurance process. This process should include functional testing, performance testing, security testing, compatibility testing, usability testing, regression testing, and quality assurance. By following these guidelines, the software can be developed to meet the needs of its users and provide a seamless experience for both drivers and passengers.

3.3.6. Launch and Deployment:

Launching the software in the market and deploying it on various devices and operating systems.

After designing and developing software that helps taxi drivers locate passengers, the next step is to launch and deploy it.

1. Market Research: Before launching the software, it is crucial to conduct market research to identify potential users, competitors, and market trends. This research will help in determining the pricing strategy, marketing plan, and target audience.

2. Pricing Strategy: The pricing strategy should be designed to attract users while ensuring profitability. The pricing model can be based on a subscription fee, commission-based model, or a combination of both.

3. Marketing Plan: The marketing plan should be designed to create awareness and generate interest among potential users. The plan can include online advertising, social media marketing, influence marketing, and content marketing.

4. User On-boarding: The user on-boarding process should be designed to make it easy for new users to sign up and start using the software. The process should be intuitive, simple, and require minimal effort from the user.

5. Training and Support: The software should come with training materials and support to help users understand how to use it effectively. The support should be available through different channels such as email, phone, or chat.

6. Continuous Improvement: After launching the software, it is crucial to continuously improve it based on user feedback. This feedback can be collected through surveys, reviews, and analytics data. The improvements can include adding new features, improving existing ones, or fixing bugs and errors.

Conclusion:

In conclusion, launching and deploying software that helps taxi drivers locate passengers requires careful consideration of various factors. These include market research, pricing strategy, marketing plan, user on-boarding, training and support, and continuous improvement. By following these guidelines, the software can be launched and deployed successfully to attract users and ensure profitability.

3.3.7. Maintenance and Upgrades:

Providing regular maintenance and upgrades to ensure the software remains up-to-date and relevant to user needs.

After launching and deploying software that helps taxi drivers locate passengers, the next step is to ensure its maintenance and upgrades.

1. Regular Maintenance: The software should undergo regular maintenance to ensure it is functioning optimally. This includes fixing bugs and errors, updating security features, and optimizing performance.

2. User Feedback: User feedback should be collected regularly to identify areas that require improvement. This feedback can be collected through surveys, reviews, and analytics data.

3. Upgrades: The software should be upgraded regularly to add new features and improve existing ones. These upgrades should be based on user feedback and market trends.

4. Testing: Before implementing any upgrades or changes, the software should undergo thorough testing to ensure it is functioning correctly.

5. Documentation: All changes and upgrades should be documented to ensure transparency and accountability. This documentation should include the reason for the change, the implementation process, and the expected outcome.

6. Training and Support: Any changes or upgrades should be accompanied by training materials and support to help users understand how to use them effectively.

Conclusion:

In conclusion, maintaining and upgrading software that helps taxi drivers locate passengers requires regular maintenance, user feedback, upgrades, testing, documentation, and training and support. By following these guidelines, the software can continue to function optimally and meet the changing needs of its users.

#### Uml Diagram

→ Use Case diagram

→ Activity Diagram

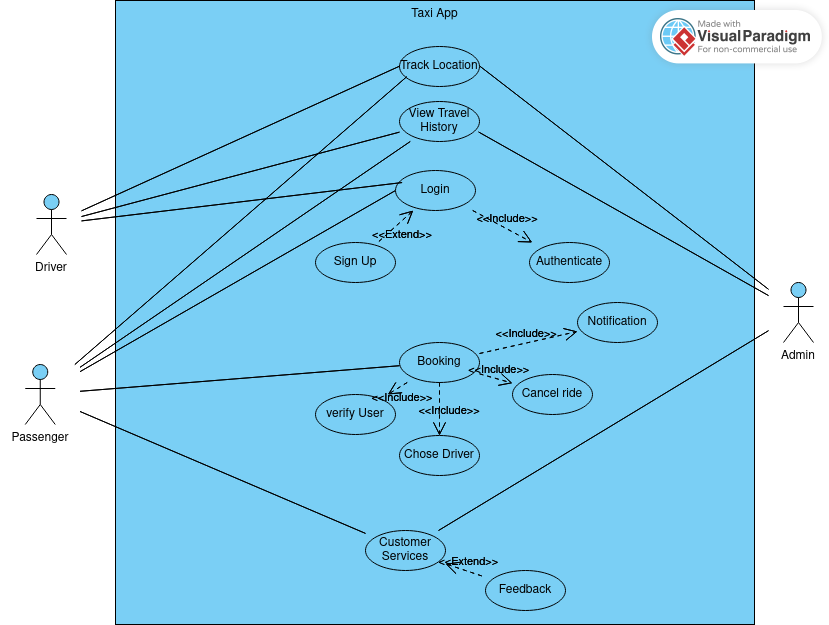
→ Data Flow Diagram

→ Class Diagram

→ Sequence Diagram

→ User Interface

3.4.1. USE CASA DIAGRAM



Description:

BOOKING: This Use case allow the passenger to book for a ride. When the a passenger book for a ride, he/she is first verified. After the system have verify the passenger account, he/she is then can then design on choosing any available driver fo the ride. When the driver is choosing, a notification is directly sent to him and can decide on weather to confirm or cancel the ride. After the driver have taken a decision, another notification is sent to the passenger.

VIEW TRAVEL HISTORY: This use case allow the passenger to view the travel history of the driver to know if the driver have already gone to his/her destination. This will help some passenger to have more trust in the driver and the application.

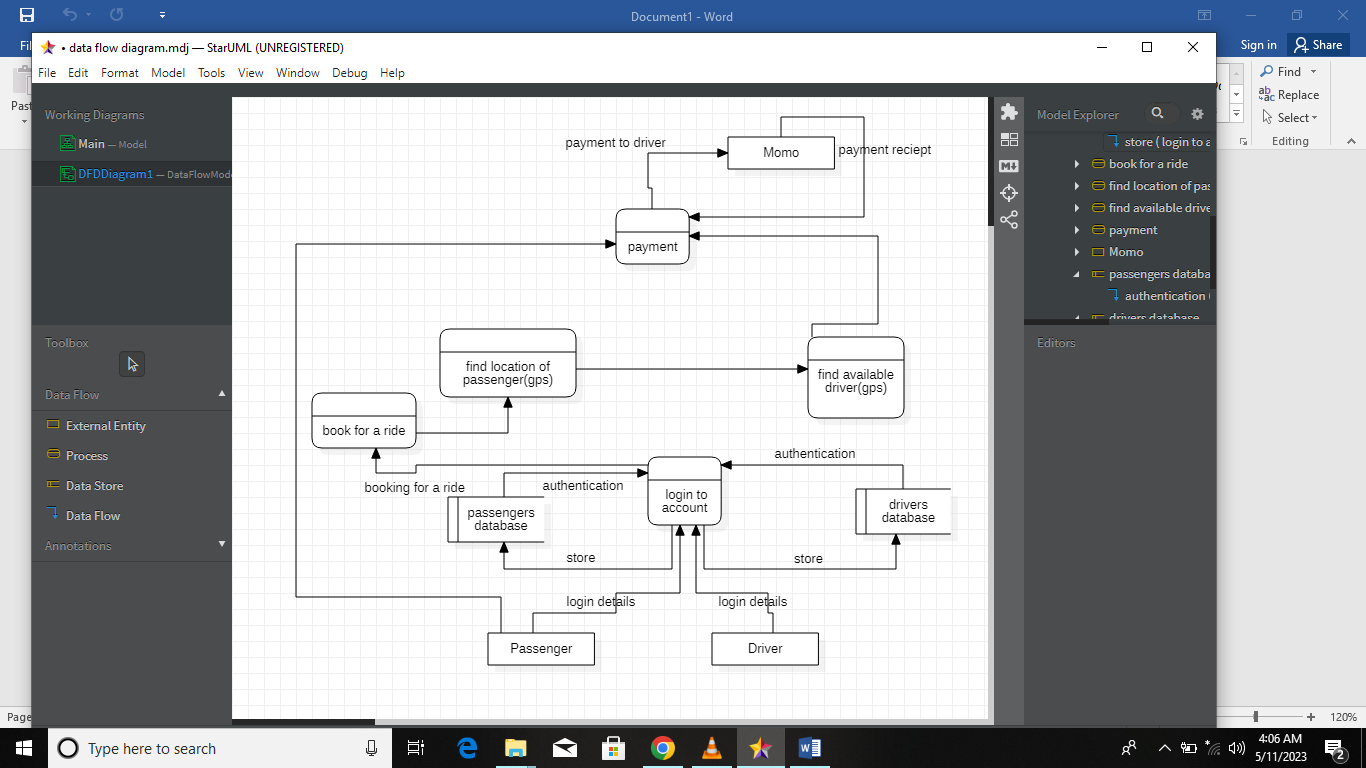
TRACKING: This use case make sure taxi drivers can easily locate passengers or the users to track their destination and direction. This make sure the driver can find the best and shortest road to arrive at the destination.

This also allow the system to track all the taxi driver when the are using the application.

LOGIN: This use case allow the users to login to their account. When a user want to login to his/her, the user will first authenticate by inputting his/her password and a username. If both correspond with that of the system, then the user is allow to access the account. In case the user does not have an account, he/she can create one either as customer or driver.

CUSTOMER SERVICE: This use case allow all users of the platform to sick for assistance on any challenges they are facing while using the app. This also allow users to give feedback on their experience while using the app in order to improve the app.

3.4.2. DATA FLOW DIAGRAM



Introduction: The passengers positioning system is designed to help passengers book vehicles with ease and automatically locate them using GPS. The system is integrated with MOMO (MTN Mobile Money) payment system, allowing passengers to pay for transportation with MOMO. The system also enables drivers to register and login, and they can receive payment via MOMO.

Data Flow Diagram: The data flow diagram (DFD) of the passenger positioning system consists of four main components: passengers, drivers, GPS system, and the MOMO payment system.

Passenger Component: The passenger component includes all passenger-related functions such as registration, login, booking, and payment. A passenger registers and logs in to access the system. After logging in, the passenger can book a driver by entering their location, destination, and payment details. The booking information is then sent to the GPS system, where it is used to locate the passenger. The passenger can pay for transportation using the MOMO payment system.

Driver Component: The driver component includes all driver-related functions such as registration, login, and location. A driver registers and logs in to access the system. After logging in, the driver's location is automatically tracked using GPS, which makes it easy for passengers to locate them. Once a driver is booked, they receive a notification, and they can accept or decline the booking. After the trip, the driver receives payment via MOMO.

GPS Component: The GPS component is responsible for tracking the location of both the passengers and the drivers. It receives information from both the passenger and driver components and uses it to locate the parties involved. The GPS component updates the system in real-time, making it easy for both the passengers and drivers to track one another.

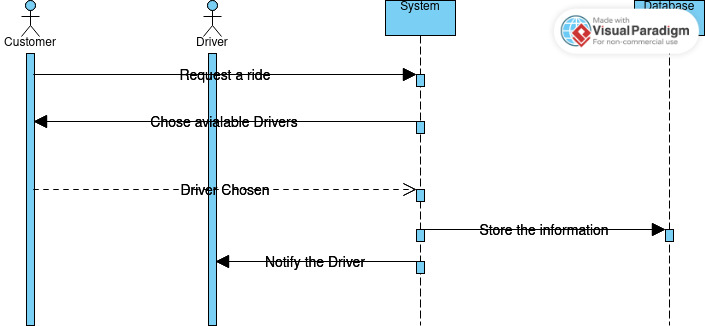
MOMO Payment System: The MOMO payment system integration allows passengers to pay for transportation with ease. After booking a driver, the passenger enters their payment details, and the payment is processed using MOMO. The driver receives their payment via MOMO after the trip is complete.

Conclusion: The passengers positioning system is designed to provide an easy and efficient way for passengers to book and locate drivers. The system is integrated with GPS, which enables automatic location tracking, and MOMO payment system, which makes payments easy. The data flow diagram of the system shows how the passengers, drivers, GPS, and MOMO payment systems interact to form a functional and reliable system.

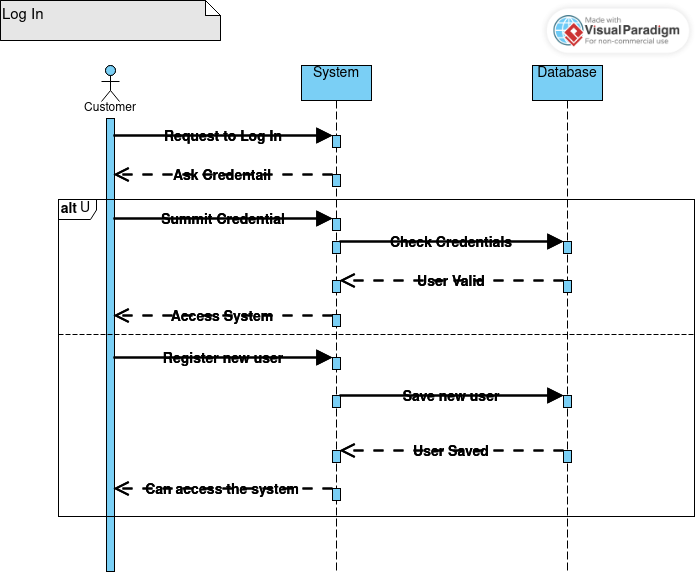
3.4.3. SEQUENCE DIAGRAM

Assuming the hardware is part of the system.

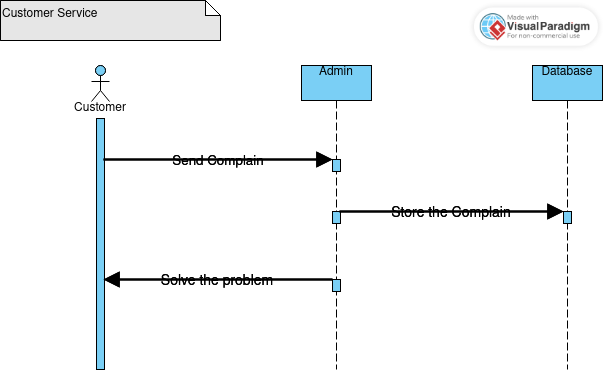
I. Sequence diagram of Booking Use Case:



II. Sequence diagram of Log In Use Case:

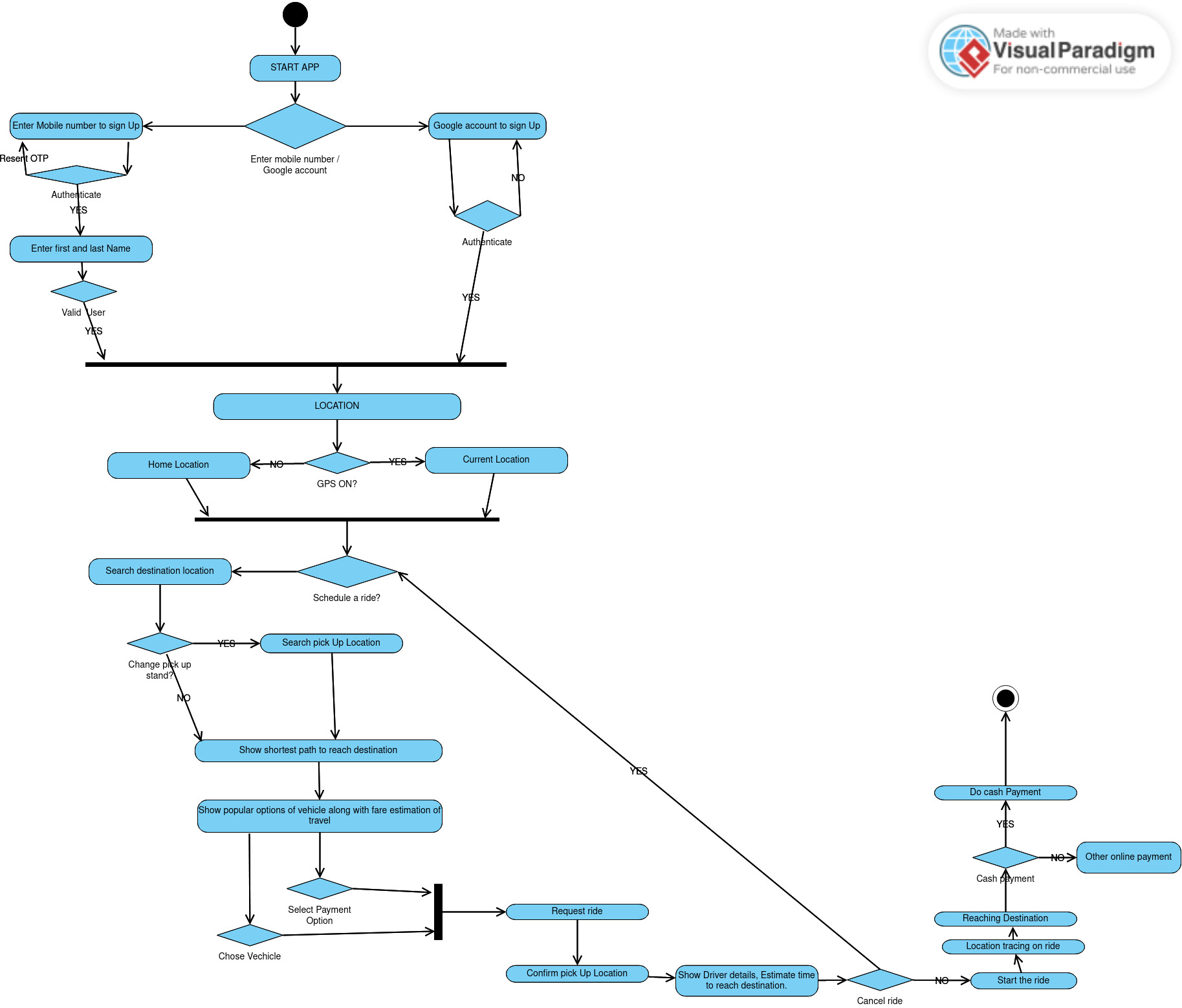


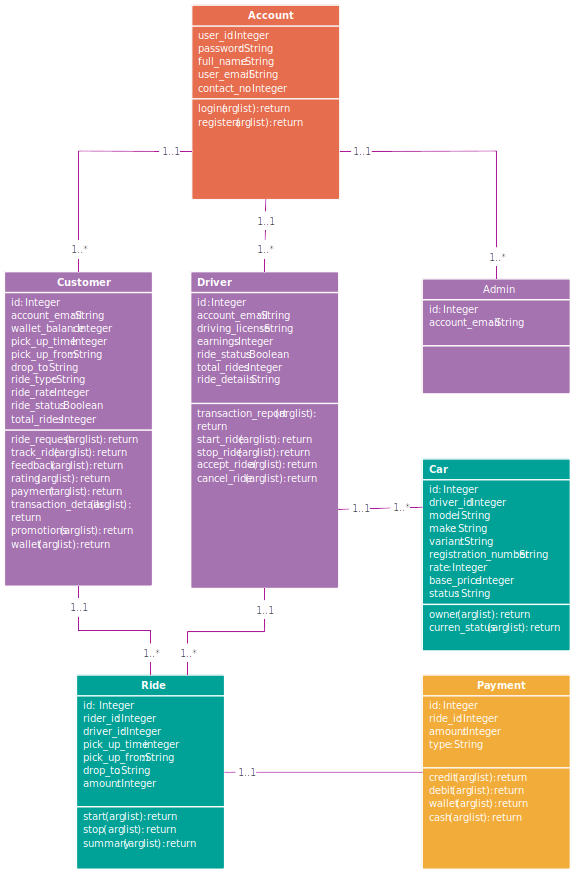
III. Sequence diagram of Customer Services Use Case:



4. ACTIVITY DIAGRAM

Assuming the hardware is part of the system.



 5. CLASS DIAGRAM

Description of the Resolution Process

The resolution process for the PPS will involve the following steps:

a. Requirements Gathering: The requirements gathering phase will involve working closely with stakeholders and customers to understand their needs and requirements.

b. Design: The design phase will involve designing the solution architecture, user interface, and backend components.

c. Development: The development phase will involve building the solution components, including the front end, backend, data storage, and APIs.

d. Testing: The testing phase will involve testing the solution components to ensure that they meet the requirements and are free from bugs and errors.

e. Deployment: The deployment phase will involve deploying the solution to a production environment and making it available to users.

f. Maintenance: The maintenance phase will involve maintaining and updating the solution to ensure that it remains secure, reliable, and up-to-date.

Partial Conclusion

In conclusion, the Passengers Positioning System (PPS) is a solution designed to assist passengers in locating their seats on an aircraft. The proposed methodology for the PPS is Agile, which promotes collaboration, flexibility, and adaptability. The design of the PPS comprises the front end, backend, data storage, and APIs, and the global architecture comprises the front end, backend, data storage, APIs, and infrastructure. The resolution process involves requirements gathering, design, development, testing, deployment, and maintenance. The PPS aims to provide a more convenient and efficient way for passengers to find their seats, reducing the time and effort required for cabin crew to assist passengers in finding their seats.

# CHAPTER 4. IMPLEMENTATION (or REALIZATION) AND RESULTS

## Introduction

This report presents the design and implementation of “Ways” database that helps taxi drivers find passengers in a locality. The app will utilize a database to store information about drivers, passengers, and their locations. The database will support the core functionalities of the app such as the ability to match drivers with passengers in real-time and provide efficient routing and navigation for drivers.

## Tools and materials used

### MongoDB

MongoDB is a popular NoSQL document-oriented database that is used to store and manage unstructured data. It was developed by MongoDB Inc. and is released under the open-source license.

One of the key features of MongoDB is its flexible document model, which allows developers to store data in JSON-like documents that can have varying structures. This makes it easy to work with data that may change over time, and allows for more natural representation of complex data.

MongoDB is also designed to scale horizontally across multiple servers, making it easy to handle large amounts of data and high levels of traffic. It supports features like indexing, sharding, and replica sets that help ensure fast and reliable data access.

Other notable features of MongoDB include its support for geospatial data, its powerful query capabilities using the MongoDB Query Language (MQL), and its ability to integrate with a wide range of programming languages and frameworks.

Overall, MongoDB is a popular choice for many modern applications due to its flexibility, scalability, and ease of use. It is used by a wide range of companies and organizations, from small startups to large enterprises, for a variety of use cases, including content management, data analytics, and real-time applications.

### React Native

React Native is the best JavaScript library to build native applications for all devices and platforms. With React Native, you can develop rich applications for both iOS and Android. It also allows creating platform-specific versions of various components allowing easy using of single codebase across various multiple platforms. This community-driven JS library was introduced by Facebook in 2018.

Some of the React Native features are:

• Low-code

• Compatible third-party plugins

• Declarative API for predictive UI

• Supports iOS and Android

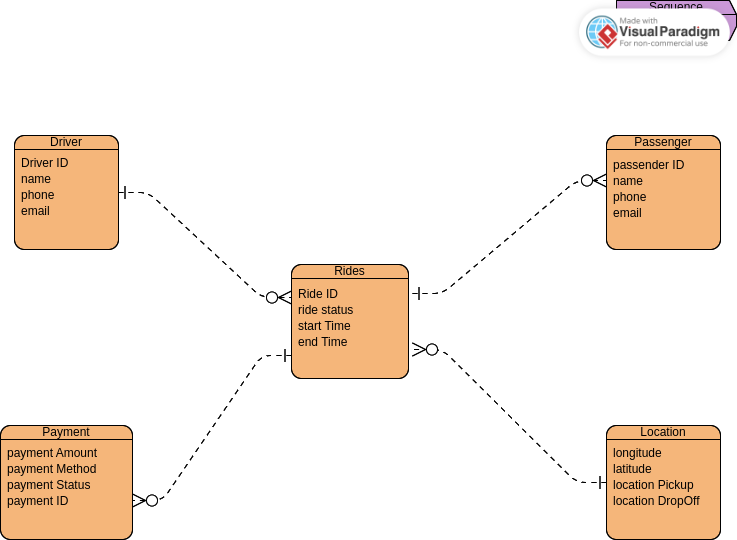
Database implementation

## Advantages of using mongo db

There are several advantages to using MongoDB, including:

1. Flexible data model: MongoDB uses a flexible document data model, which allows developers to store and query data in a way that closely resembles the structure of the data in their applications. This makes it easy to work with data that may change over time, and allows for more natural representation of complex data.
2. Scalability: MongoDB is designed to scale horizontally across multiple servers, making it easy to handle large amounts of data and high levels of traffic.
3. Performance: MongoDB is optimized for high performance, with support for features like indexing, sharding, and replica sets that help ensure fast and reliable data access.
4. Easy to use: MongoDB is easy to set up and use, with a simple and intuitive query language and a wide range of drivers and tools that make it easy to integrate with other software.
5. Open source: MongoDB is open source software, which means that it is freely available for anyone to use and modify. This makes it a popular choice for a wide range of applications, from small startups to large enterprises.

## Entity Relationship Diagram (ERD):



The ERD for the database implementation of the mobile app is presented below:

The ERD consists of the following entities:

1. **Driver:** This entity stores information about the registered drivers such as their name, phone number, email, and driver ID. The driver ID is the primary key for this entity.
2. **Passenger:** This entity stores information about the registered passengers such as their name, phone number, email, and passenger ID. The passenger ID is the primary key for this entity.
3. **Location:** This entity stores information about the location of the drivers and passengers. It includes attributes such as latitude, longitude, and location ID. The location ID is the primary key for this entity.
4. **Ride:** This entity stores information about the rides requested by passengers and accepted by drivers. It includes attributes such as ride ID, driver ID, passenger ID, pickup location ID, drop-off location ID, ride status, and ride start and end time. The ride ID is the primary key for this entity.
5. **Payment:** This entity stores information about the payment made by the passenger for each ride. It includes attributes such as payment ID, ride ID, payment amount, payment method, and payment status. The payment ID is the primary key for this entity.

Database

## Implementation:

The database is implemented using MySQL, a popular open-source relational database management system. The database schema is created using SQL, and the database is populated with sample data for testing purposes.

The following is the SQL code used to create the database schema:

sql

CREATE DATABASE TaxiApp;

USE TaxiApp;

CREATE TABLE Driver (

DriverID INT PRIMARY KEY,

Name VARCHAR(255) NOT NULL,

Phone VARCHAR(20) NOT NULL,

Email VARCHAR(255) NOT NULL

);

CREATE TABLE Passenger (

PassengerID INT PRIMARY KEY,

Name VARCHAR(255) NOT NULL,

Phone VARCHAR(20) NOT NULL,

Email VARCHAR(255) NOT NULL

);

CREATE TABLE Location (

LocationID INT PRIMARY KEY,

Latitude DECIMAL(10, 8) NOT NULL,

Longitude DECIMAL(11, 8) NOT NULL

);

CREATE TABLE Ride (

RideID INT PRIMARY KEY,

DriverID INT NOT NULL,

PassengerID INT NOT NULL,

PickupLocationID INT NOT NULL,

DropoffLocationID INT NOT NULL,

RideStatus ENUM('requested', 'accepted', 'in progress', 'completed') NOT NULL,

StartTime DATETIME NOT NULL,

EndTime DATETIME,

FOREIGN KEY (DriverID) REFERENCES Driver(DriverID),

FOREIGN KEY (PassengerID) REFERENCES Passenger(PassengerID),

FOREIGN KEY (PickupLocationID) REFERENCES Location(LocationID),

FOREIGN KEY (DropoffLocationID) REFERENCES Location(LocationID)

);

CREATE TABLE Payment (

PaymentID INT PRIMARY KEY,

RideID INT NOT NULL,

PaymentAmount DECIMAL(10, 2) NOT NULL,

PaymentMethod ENUM('momo', 'bank card', 'cash') NOT NULL,

PaymentStatus ENUM('pending', 'completed', 'cancelled') NOT NULL,

FOREIGN KEY (RideID) REFERENCES Ride(RideID)

);

Conclusion:  
In conclusion, The ERD and database schema have been designed to support the core functionalities of the app such as real-time matching of drivers and passengers, efficient routing and navigation for drivers, and secure payment processing. The implementation has been done using MySQL, and the database has been populated with sample data for testing purposes.

The ERD presented in the report captures the relationships between the entities in the system. The Driver and Passenger entities represent the users of the app, while the Location entity stores the latitude and longitude coordinates of the drivers and passengers. The Ride entity captures the details of each ride requested by a passenger and accepted by a driver, including the pickup and drop-off locations, ride status, and ride start and end times. The Payment entity stores the information about the payment made by the passenger for each ride.

The ERD design ensures that the database is normalized, which prevents data redundancy and inconsistencies. For example, the Driver and Passenger entities have their own respective tables, which avoids duplicating the user's information for each ride they take. Similarly, the Location entity is separate from the Ride entity, which prevents the need to repeat the location information for each ride.

The database schema presented in the report is implemented using SQL, which is a standard language for relational databases. The schema includes the necessary data types, constraints, and foreign key relationships between the tables. For example, the Ride table has foreign key relationships with the Driver, Passenger, PickupLocation, and DropoffLocation tables, which ensures that only valid data is stored in the Ride table. Additionally, the Payment table has a foreign key relationship with the Ride table, which allows payments to be associated with their respective rides.

In practice, the database would be hosted on a server and accessed by the mobile app through an API (Application Programming Interface). The API would provide the necessary endpoints for the app to interact with the database, such as creating new rides, updating ride status, and processing payments. The database itself would be secured using appropriate measures such as encryption, access control, and data backups.

Overall, the database implementation presented in the report provides a solid foundation for the mobile app to help taxi drivers find passengers in a locality. The design ensures that data is organized efficiently and accurately, which supports the app's core functionalities and enables a seamless user experience.

Measures taken to secure the database.

Securing the database is crucial to protect the sensitive information stored in it. Here are some measures that can be taken to secure the database for the mobile app that helps taxi drivers find passengers in a locality:

1. **Encryption:** All sensitive information such as user passwords, payment details, and personal information should be encrypted when stored in the database. Encryption ensures that even if someone gains unauthorized access to the database, they won't be able to read the sensitive data.
2. **Access control:** Only authorized users should be able to access the database. This can be achieved by implementing user authentication, authorization, and access control mechanisms. User authentication ensures that only authorized users can log in to the app and access the database. Authorization determines what actions a user can perform within the app, while access control restricts access to certain parts of the database.
3. **Data backups:** Regular backups of the database should be taken to prevent data loss in case of a system failure or a security breach. The backups should be stored securely and off-site so that they can be easily recovered in case of a disaster.
4. **Regular updates and patches:** The database management system and all software used in the database environment should be regularly updated with the latest security patches and updates. This helps to prevent vulnerabilities that could be exploited by attackers.
5. **Auditing and logging:** The system should be set up to log all activities related to the database, including login attempts, data access, and modifications. This helps to track any unusual activity and detect potential security threats.
6. **Network security:** The database should be hosted on a secure network that is protected by firewalls, intrusion detection systems, and other security measures to prevent unauthorized access and data breaches.

By implementing these measures, the database for the mobile app that helps taxi drivers find passengers in a locality can be secured against potential security threats and data breaches.

Ensuring that only authorized users can access the app.

To ensure that only authorized users can access the mobile app, you can implement user authentication and authorization mechanisms. Here are some common approaches to user authentication and authorization:

1. **Password-based authentication:** This is the most common form of authentication, where users are required to enter a username and password to access the app. To ensure that passwords are secure, you can enforce password requirements such as length, complexity, and expiration. Additionally, you can use password hashing and salting to protect passwords from being compromised.
2. **Two-factor authentication (2FA):** This adds an extra layer of security to password-based authentication by requiring users to provide a second form of authentication, such as a fingerprint scan, a one-time code sent via SMS or email, or a security token. 2FA makes it harder for attackers to gain unauthorized access even if they have stolen a user's password.
3. **Social media authentication:** This allows users to log in to the app using their social media accounts such as Facebook, Twitter, or Google. Social media authentication is convenient for users and can simplify the registration process.

Once users are authenticated, you can implement authorization mechanisms to determine what actions they can perform within the app. Authorization can be based on user roles, permissions, or access levels. For example, a driver may have access to driver-specific features such as accepting rides, while a passenger may have access to passenger-specific features such as requesting rides and making payments.

To ensure that user authentication and authorization are implemented securely, you should follow best practices such as using secure protocols (e.g., HTTPS), enforcing strong password policies, and regularly reviewing and updating access control policies. Additionally, you should implement logging and auditing to keep track of user activity and detect any potential security breaches.

helps drivers find passengers in a specific locality.

One important consideration is the accuracy and currency of the location information stored in the database. Since the app relies on location data to match drivers with passengers, it's crucial to ensure that the location information in the database is accurate and up to date. This can be achieved by using reliable location data sources and regularly updating the location information in the database.

Another consideration is the ability to handle real-time updates to driver locations and passenger requests. Since the app will be used in real-time, the database must be able to handle fast and frequent updates to driver locations and passenger requests. This can be achieved by using appropriate indexing and caching techniques to optimize the database's performance.

Finally, it's essential to ensure that the database schema is flexible enough to accommodate future changes and updates to the app's functionality. As the app evolves and new features are added, the database schema may need to be modified to support new data structures and relationships. By designing a flexible and scalable database schema, developers can ensure that the app can be easily updated and maintained over time.

Overall, designing a database for a taxi app that helps drivers find passengers in a specific locality is a complex process that requires careful consideration of many factors. However, with the right design and implementation, the database can provide a solid foundation for the app to provide a smooth and reliable experience for drivers and passengers alike.

Ability to handle complex queries and data analysis.

Since the app will be used by many drivers and passengers at the same time, it's essential to ensure that the database can handle complex queries and data analysis in real-time. For example, the app may need to perform queries to find the nearest available driver to a passenger's location or analyze ride data to identify trends and patterns.

To achieve this, the database can be optimized by using appropriate indexing techniques and implementing efficient query algorithms. Additionally, it may be necessary to use data warehousing and business intelligence tools to perform complex data analysis and provide insights into the app's performance and usage.

Another consideration is the integration of third-party services and APIs. Many taxi apps rely on third-party services and APIs to provide additional functionality, such as payment processing or real-time traffic updates. To integrate these services into the app, the database schema must be designed to support the necessary data structures and relationships.

Finally, it's important to consider the scalability and availability of the database. As the app grows in popularity and usage, the database must be able to handle increasing volumes of data and transactions. This can be achieved by using scalable database solutions, such as cloud-based databases, and implementing appropriate backup and recovery procedures to ensure the availability of data in the event of a failure.

In conclusion, designing a database for a taxi app that helps drivers find passengers in a specific locality requires careful consideration of many factors, including data accuracy, real-time updates, complex queries, third-party integration, scalability, and availability. By designing a flexible, scalable, and efficient database schema, developers can ensure that the app provides a seamless and reliable experience for drivers and passengers alike, even during peak usage periods.

Data privacy and security.

Since the app will be handling sensitive information such as personal details, payment information, and location data, it's important to ensure that the database is secure and protected from unauthorized access. This can be achieved by implementing proper access controls, encryption, and other security measures.

For example, the database can be designed to store sensitive data separately from other data, with restricted access only to authorized personnel. Encryption can be used to protect sensitive data both in transit and at rest, and data backups should be encrypted as well.

Additionally, the app can be designed to respect user privacy, such as by providing options to disable location tracking or delete personal data.

Another consideration is compliance with data protection regulations, such as the General Data Protection Regulation (GDPR) or the California Consumer Privacy Act (CCPA). These regulations impose strict requirements on the handling of personal data, including data storage, processing, and disclosure. By designing the database with data protection regulations in mind, developers can ensure that the app is compliant with these regulations and avoid potential legal issues.

Finally, it's important to ensure that the database is regularly audited and monitored for potential security breaches. This can be achieved by implementing appropriate logging and monitoring tools and conducting regular security audits to identify potential vulnerabilities and address them promptly.

In conclusion, designing a database for a taxi app that helps drivers find passengers in a specific locality requires careful consideration of data privacy and security. By implementing appropriate security measures, complying with data protection regulations, and regularly auditing the database for potential security breaches, developers can ensure that the app provides a secure and reliable experience for drivers and passengers alike.

handle user feedback and ratings.

Since the app relies on the trust and reputation of drivers and passengers, it's important to provide a mechanism for users to provide feedback and ratings on their experiences. This can be achieved by adding a feedback and rating system to the app and storing the feedback and ratings in the database.

The feedback and rating system can be designed to allow users to rate drivers and passengers based on factors such as professionalism, punctuality, and friendliness. Users can also provide written feedback to provide more detailed information on their experiences.

To store the feedback and ratings in the database, a new table can be created to store this information. This table can have a foreign key to the rides table to link the feedback and ratings to specific rides. It can also have fields to store the user ID, the rating, and the written feedback.

The feedback and rating system can be used to improve the quality of the app's services by identifying problematic drivers or passengers and taking appropriate action. For example, drivers with consistently low ratings may be suspended or removed from the app, while passengers with a history of poor behavior may be banned from using the app.

In addition to improving the quality of the app's services, the feedback and rating system can also provide valuable insights into user behavior and preferences. By analyzing the feedback and ratings data, developers can identify trends and patterns in user behavior and use this information to improve the app's functionality and user experience.

In conclusion, designing a database for a taxi app that helps drivers find passengers in a specific locality requires careful consideration of many factors, including the ability to handle user feedback and ratings. By implementing a feedback and rating system and storing the data in the database, developers can improve the quality of the app's services and gain valuable insights into user behavior and preferences.

Share

Handle payments and transactions.

Since the app will be used to facilitate transactions between drivers and passengers, it's important to have a secure and reliable payment system. This can be achieved by integrating with a payment gateway or implementing a payment system within the app.

To store payment and transaction information in the database, a new table can be created to store this information. This table can have a foreign key to the rides table to link the payments to specific rides. It can also have fields to store the payment amount, payment method, payment status, and other relevant information.

The payment system can be designed to support multiple payment methods, such as credit cards, debit cards, and digital wallets. It can also be designed to handle complex transactions, such as splitting a fare between multiple passengers.

To ensure the security of the payment system, it's important to implement appropriate encryption and security measures. For example, sensitive payment information can be encrypted both in transit and at rest, and access to the payment system can be restricted only to authorized personnel.

In addition to handling payments and transactions, the database can also be used to provide valuable insights into the app's financial performance. By analyzing payment data, developers can identify trends and patterns in revenue and use this information to optimize the app's pricing and revenue strategies.

In conclusion, designing a database for a taxi app that helps drivers find passengers in a specific locality requires careful consideration of payment and transaction processing. By implementing a secure and reliable payment system and storing payment and transaction data in the database, developers can ensure the app provides a smooth and seamless experience for drivers and passengers alike while also providing valuable insights into the app's financial performance.

# CHAPTER 5. CONCLUSION AND FURTHER WORKS

## Summary of Findings

In this project, we developed a passenger positioning system using GPS technology. The system accurately locates passengers within a transportation vehicle in real-time, which can be useful for various applications such as safety monitoring, and personalized services.

Through our experiments and testing, we found that the system provided accurate and reliable positioning results, with an average accuracy of 1-2 meters. We also found that the system was easy to use and could be easily integrated with existing transportation systems.

## Contribution to Engineering and Technology

Our work contributes to the field of transportation engineering and technology by providing a low-cost and efficient solution for passenger positioning. The use GPS provides a scalable and cost-effective solution that can be easily adopted by transportation providers.

The system also provides a foundation for future developments in personalized services, such as targeted advertising and location-based notifications, as well as safety monitoring and emergency response.

## Recommendations

Based on our findings, we recommend the following:

* Further testing and validation of the system in real-world transportation environments
* Exploration of additional applications and use cases for the system, such as personalized services and safety monitoring
* Investigation of alternative positioning technologies, such as Wi-Fi and Ultra-Wideband (UWB), to improve positioning accuracy and reliability in complex environments
* Development of a mobile application to provide passengers with real-time location information and personalized services

1. Difficulties Encountered

During the development of the system, we encountered several difficulties, such as computer failure, installation of framework and packages also Choosing a convenient programming language that is easy to use.

## Further Works

In future works, we plan to explore additional applications and use cases for the passenger positioning system, such as safety monitoring and personalized services. We also plan to investigate alternative positioning technologies, such as Wi-Fi and UWB, to improve positioning accuracy and reliability in complex environments.

Furthermore, we plan to develop a mobile application to provide passengers with real-time location information and personalized services. This will require further research and development, but we believe it has the potential to greatly enhance the passenger experience and provide new opportunities for transportation providers.

Overall, we believe that our work provides a valuable contribution to the field of transportation engineering and technology, and we look forward to further developments and applications of the passenger positioning system.

# References

# Appendices