**Ride Management Application for Bajajs in**

**Gondar City**

**by**



**University of Gondar**

**Faculty of Informatics**

**Department of Information System**

**Group Members**

Name Id No

1. Assefa Demses 00737/12
2. Dereje Aragaw 02236/12
3. Fikir Getu 01627/12
4. Firaol Teklu 01156/12
5. Henok Gashew 01181/12

A Group Project

Submitted to the Department of Information Systems, Faculty of Informatics, University of Gondar, in meeting the preliminary project requirement for partial fulfillment of the award of Bachelor of Science Degree in Information Systems.

Gondar, Ethiopia

Date/Year: April 2023

# **Approval Sheet**

This Group Project entitled “Ride Management Application for Bajajs in Gondar City” has been read and approved as meeting the preliminary project requirements of the Department of Information Systems in partial fulfillment for the award of a Bachelor of Science degree in Information Systems, University of Gondar, Gondar, Ethiopia.

**Approved by:**

1. Name of Advisor: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature:\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_
2. Name of Project Coordinator: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature: \_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_

# **Acknowledgments**

First and foremost, We would like to thank God. Then We would like to express gratitude to Mrs. Tigist Ayeragn of the Transport Bureau of Gondar City for providing the necessary information for the project. We would also like to give special thanks to our advisor Teshome Aychluhm for his tremendous support and guidance throughout the project.

Finally, we would like to thank everyone whose names were not mentioned but who helped us throughout the project.

Table of Contents

[**Approval Sheet** i](#_Toc133885127)

[**Acknowledgments** ii](#_Toc133885128)

[**List of Figures** vi](#_Toc133885129)

[**List of tables** vii](#_Toc133885130)

[**CHAPTER ONE** 1](#_Toc133885131)

[**Introduction** 1](#_Toc133885132)

[**1.1.** **Background of the project** 1](#_Toc133885133)

[**1.2.** **Statement of Problem and Justification** 2](#_Toc133885134)

[**1.2.1.** **Statement of problem** 2](#_Toc133885135)

[**1.2.2.** **Justification** 2](#_Toc133885136)

[**1.3.** **Project objectives** 3](#_Toc133885137)

[**1.3.1.** **General Objective** 3](#_Toc133885138)

[**1.3.2.** **Specific Objective** 3](#_Toc133885139)

[**1.4.** **Scope of the project** 4](#_Toc133885140)

[**1.5.** **Limitations of project** 4](#_Toc133885141)

[**1.6.** **System Development Methodology** 4](#_Toc133885142)

[**1.6.1.** **System development Approach** 4](#_Toc133885143)

[**1.6.2.** **System Development Tools** 4](#_Toc133885144)

[**Hardware tools** 5](#_Toc133885145)

[**Software tools** 5](#_Toc133885146)

[**Implementation languages** 5](#_Toc133885147)

[**1.7.** **Significance of Project** 5](#_Toc133885148)

[**1.8.** **Beneficiaries of the project** 6](#_Toc133885149)

[**1.9.** **Feasibility Study** 6](#_Toc133885150)

[**1.10.** **Project schedule** 7](#_Toc133885151)

[**1.11.** **Project Budget** 7](#_Toc133885152)

[**CHAPTER TWO** 9](#_Toc133885153)

[**2.** **Requirement Analysis** 9](#_Toc133885154)

[**2.1.** 9](#_Toc133885155)

[**2.1.** **Current System Description** 9](#_Toc133885156)

[**2.1.1.** **The major function of the current system** 9](#_Toc133885157)

[**2.1.2.** **The problem with the Existing System** 9](#_Toc133885158)

[**2.2.** **Requirement Gathering** 9](#_Toc133885159)

[**2.2.1.** **Requirement Gathering Methods** 9](#_Toc133885160)

[**2.2.2.** **Business Rules** 10](#_Toc133885161)

[**2.3.** **Proposed System Description** 10](#_Toc133885162)

[**2.3.1.** **Overview** 10](#_Toc133885163)

[**2.3.2.** **Functional Requirements** 11](#_Toc133885164)

[**2.3.3.** **Nonfunctional requirements** 12](#_Toc133885165)

[**2.3.3.1.** **Performance** 12](#_Toc133885166)

[**2.3.3.2.** **Scalability** 12](#_Toc133885167)

[**2.3.3.3.** **Availability** 12](#_Toc133885168)

[**2.3.3.4.** **Reliability** 12](#_Toc133885169)

[**2.3.3.5.** **Security** 12](#_Toc133885170)

[**2.3.3.6.** **Environmental** 13](#_Toc133885171)

[**2.3.3.7.** **Usability** 13](#_Toc133885172)

[**Chapter Three** 14](#_Toc133885173)

[**3.** **System Model** 14](#_Toc133885174)

[**3.1.** **Scenario** 14](#_Toc133885175)

[**3.1.1.** **Use Case Model** 14](#_Toc133885176)

[**3.1.2.** **Use Case Diagram** 2](#_Toc133885177)

[**3.1.3.** **Description of Use Case Model** 2](#_Toc133885178)

[**3.1.4.** **Activity Diagram** 15](#_Toc133885179)

[**3.1.5.** **Object Model** 19](#_Toc133885180)

[**3.1.6.** **Data Dictionary** 19](#_Toc133885181)

[**3.1.7.** **Class Model** 22](#_Toc133885182)

[**3.1.8.** **Dynamic Modeling** 23](#_Toc133885183)

[**3.1.8.1.** **Sequence diagram** 23](#_Toc133885184)

[**3.1.9.** **User Interface** 25](#_Toc133885185)

[**Chapter Four** 26](#_Toc133885186)

[**4.** **System Design** 26](#_Toc133885187)

[**4.1.** **Introduction** 26](#_Toc133885188)

[**4.2.** **Proposed software architecture** 26](#_Toc133885189)

[**4.3.** **System Decomposition** 27](#_Toc133885190)

[**4.4.** **Hardware/ software mapping** 30](#_Toc133885191)

[**4.5.** **Persistent data modeling** 31](#_Toc133885192)

[**4.6.** **Access control and security** 31](#_Toc133885193)

[**4.7.** **Detailed class diagram** 31](#_Toc133885194)

[**4.8.** **Package Diagram** 32](#_Toc133885195)

[**4.9.** **Deployment** 32](#_Toc133885196)

# **List of Figures**

[Figure 1.1: Schedule Gantt Chart 7](#_Toc133884150)

[Figure 3.1 UseCase Diagram 2](#_Toc133884151)

[Figure 3.1 Activity Diagram for Log in 15](#_Toc133884170)

[Figure 3.2 Activity Diagram for Register 16](#_Toc133884171)

[Figure 3.3 Activity Diagram for ride request 17](#_Toc133884172)

[Figure 3.4 Activity Diagram for track trip progress 18](#_Toc133884173)

[Figure 3.5 Activity Diagram for Cancel Ride 18](#_Toc133884174)

[Figure 3.6 Class Model DIagram 22](#_Toc133884175)

[Figure 3.7: sequence diagram for log in 23](#_Toc133884176)

[Figure 3.8 sequence diagram for ride request 23](#_Toc133884177)

[Figure 3.9 sequence diagram for tracking trip progress 24](#_Toc133884178)

[Figure 3.10 sequence diagram for canceling ride 24](#_Toc133884179)

[Figure 3.11 sequence diagram for managinging the database 25](#_Toc133884180)

[Figure 4.0‑1 Life cycle of a trip 26](#_Toc133884181)

[Figure 4.2 System decomposition diagram 29](#_Toc133884182)

[Figure 4.3 Deployment Diagram 30](#_Toc133884183)

[Figure 4.4 Persistent Data Modeling Diagram 31](#_Toc133884184)

[Figure 4.5 Package Diagram 32](#_Toc133884185)

# **List of tables**

[Table 3.1: Use case description for Login 2](#_Toc133884672)

[Table 3.2: Use case description for Request ride 3](#_Toc133884673)

[Table 3.3: Use case description for Track trip progress 4](#_Toc133884674)

[Table 3.4: Use case description for Rate ride 5](#_Toc133884675)

[Table 3.5: Use case description for View and manage available drivers 5](#_Toc133884676)

[Table 3.6: Use case description for Manage database 6](#_Toc133884677)

[Table 3.7: Use case description for View trip summary 7](#_Toc133884678)

[Table 3.8: Use case description for Search and view trip history 7](#_Toc133884679)

[Table 3.9: Use case description for Register as Driver 8](#_Toc133884680)

[Table 3.10: Use case description for View Request 8](#_Toc133884681)

[Table 3.11: Use case description for Accept Request 9](#_Toc133884682)

[Table 3.12: Use case description for Start Ride 10](#_Toc133884683)

[Table 3.13: Use case description for End Ride 10](#_Toc133884684)

[Table 3.14: Use case description for View Earnings 11](#_Toc133884685)

[Table 3.15: Use case description for Register 12](#_Toc133884686)

[Table 3.16 Use case description for View trip detail 12](#_Toc133884687)

[Table 3.17 Use case description for View list of customers 13](#_Toc133884688)

[Table 3.18 Use case description for View list of customer request 13](#_Toc133884689)

[Table 3.19 Use case description for Log out 14](#_Toc133884690)

# **Acronyms**

1. UI: User Interface
2. GPS: Global Positioning System
3. ETA: Estimated Time of Arrival
4. HTTP: Hypertext Transfer Protocol
5. SQL: Structured Query Language
6. JSX: JavaScript XML
7. CSS: Cascading Style Sheets
8. JSON: JavaScript Object Notation
9. CLI: Command Line Interface
10. SDK: Software Development Kit
11. RAD: Rapid Application Development
12. UML: Unified Modeling Language
13. BR: Business rule
14. UC: Use Case
15. API: Application Program Interface
16. iOS iPhone Operating System
17. IDE Integrated Development Environment
18. VCS Version Control System
19. 2FA Two Factor Authentication
20. OTP One Time Password

# **CHAPTER ONE**

# **Introduction**

## **Background of the project**

The transportation industry in Ethiopia has undergone significant changes in recent years, with the rise of meter-taxis and ride-hailing services like Ride. Prior to the introduction of these services, public transportation in Ethiopia was largely informal and unregulated, with taxis and buses operating on an ad-hoc basis. However, the introduction of meter-taxis and ride-hailing services has brought a new level of professionalism and reliability to the industry. Meter-taxis were first introduced in Ethiopia in the early 2000s, and quickly gained popularity due to their reliability and affordability. These taxis are equipped with meters that calculate the fare based on the distance traveled, and are regulated by the government to ensure that passengers are not overcharged. Meter-taxis are still a popular mode of transportation in Ethiopia, particularly in urban areas where traffic congestion is a major issue. In recent years, ride-hailing services like ZayRide have also gained popularity in Ethiopia. These services offer a convenient and affordable alternative to traditional taxis, and have been embraced by both locals and tourists. However, the introduction of ride-hailing services has also been met with resistance from traditional taxi drivers, who fear that they will lose business to these new competitors. Despite these challenges, the ride-hailing industry in Ethiopia continues to grow, with new players entering the market and existing companies expanding their operations.

The government has also taken steps to regulate the industry, with the introduction of new laws and regulations aimed at ensuring the safety and security of passengers. Overall, the rise of meter-taxis and ride-hailing services in Ethiopia has brought significant benefits to the transportation industry, including increased professionalism, reliability, and affordability.[1]

Gondar city is a densely populated urban center in Ethiopia with a high demand for transportation services. However, the current transportation system in the city is characterized by a limited availability of taxis and a large number of bajaj, a three-wheeled vehicle widely used for transportation.

However in the city of Gondar the growth of ride services is very slow. Currently in the city transportation is one of the hot issues. the Cost and security of transport are major sources of concern. Meter-taxi applications like Taxiye, ZayRide, and others present a solution but haven’t been active in Gondar . Further such systems are inapplicable in the city of Gondar as a result of their costliness. And as they are relatively new to the transport market of the city of Gondar, they are not the default choice of the community.

In some way a city that is a destination for tourists and tourism is slow on current trends. This project aims to develop a similar application to those in trend and involve the local transport community, making both the technology available and the people open to alternatives of meter-taxi. This will make the transportation of the city grow on par with current trend and help for better tourism as well.

## **Statement of Problem and Justification**

## **Statement of problem**

Existing ride services like ZayRide and Zoble Ride are not suitable for the city due to their high initial cost and the focus on car-based transportation. Additionally, the security and availability of bajaj transportation are major concerns. Some problems of the existing structure are listed as follows:

* The inapplicableness of already existing systems like zayride or Zoble Ride for a city like Gondar
* Low availability and security of bajaj transport
* Inconsistent and variable cost of a private contract driver bajaj

## **Justification**

* + - 1. We propose the development of a new ride-hailing platform that specifically caters to the transportation needs of Gondar city. The platform will focus on providing safe, affordable, and reliable transportation services, with a particular emphasis on bajaj vehicles.

Our platform will be designed to address the limitations of existing ride-hailing services by offering a low-cost, on-demand, and reliable transportation option for the residents of Gondar city. Our proposed system will use features like enabling riders to easily book a bajaj ride through their mobile phones, providing real-time tracking of the vehicle's location and estimated time of arrival.

* + - 1. To ensure the safety and security of our riders, all drivers will undergo background checks, and our platform will allow riders to rate their drivers after each trip.

We will also work closely with local law enforcement agencies to address any security concerns and ensure a safe and secure transportation experience for our riders.

* + - 1. We aim to partner with bajaj vehicle owners and drivers to provide them with a steady stream of customers and help them improve their earnings. Our platform will enable drivers to earn more by offering them a transparent and fair pricing model, which will be based on distance traveled and time spent on the ride.

## **Project objectives**

## **General Objective**

The objective of this project is to develop a low-cost, secure and accessible transportation service for Gondar city using bajaj.

## **Specific Objective**

The specific objectives the project aims to achieve and meet are the following

* Collect data, analyze, design system requirements and identify problems in the existing systems in detail
* Design an easy-to-use and responsive user interface that allows users to track details of their orders and travel information
* To develop a user-friendly and intuitive ride service app that allows users to easily request and pay for rides.
* To ensure the safety and security of both riders and drivers by implementing appropriate background checks, safety features, and insurance policies.
* To provide a reliable and efficient service by optimizing the app's algorithms for matching riders with drivers and minimizing wait times.
* To offer competitive pricing and incentives to attract and retain both riders and drivers.
* To continuously improve the app's features and functionality based on user feedback and market trends.
* Develop a ride service platform that focuses on bajaj transportation and offers affordable and secure transportation services to customers.
* Implement a pricing model that is compatible with the transportation patterns in the city and encourages customer retention.
* Implement safety measures, such as GPS tracking, real-time monitoring, and driver identification.
* Evaluate the system

## **Scope of the project**

The aim of this project is to create a ride-hailing app that offers a convenient and efficient mode of transportation for users. The app will enable users to book rides, view ride details, track driver location. However, The scope of the project will be limited to the development of the app for iOS and Android platforms.

The scope of the project mainly focuses on the following features:

* User Registration: The app will allow users to register and create accounts to access the app's features. Users will be required to provide basic personal information such as name, phone number, email address. This will require more information for driver registration.
* Ride Booking: Users will be able to book rides using the app. They will be able to select their pickup and drop-off locations and choose the type of ride they want.
* Driver Matching: The app will match users with available drivers based on their location and ride request. Users will be able to view driver details such as name, photo, and rating.
* Ride Tracking: Users will be able to track the driver's location and estimated time of arrival in real-time
* Driver Management: The app will allow drivers to register and create accounts. Drivers will be able to view ride requests, accept or decline them, and receive payment.

## **Limitations of project**

Some of the main limitations of the system are as follows:

* The system does not provide an electronic payment processing system
* The system doesn’t have an Amharic language interface
* The system doesn’t have accessibility options for visually challenged users

## **System Development Methodology**

## **System development Approach**

The chosen development methodology for its adaptiveness and with the limitation of time in mind is Rapid Application Development(RAD).

* Allows to move back and forward between design phases, for updating and changing
* Easy to make improvements to the design and implementation
* Important issues addressed before final system delivery
* Allows for faster code generation

The drawback of this methodology is it needs huge collaborations and joint efforts from more than the development team. The other main problem is flexibility, since it's easy to change and update t requirements coming from users the changes made may end up dragging the project beyond the scheduled scope.

## **System Development Tools**

The tools that were used for the execution of the project both hardware and software are as follows

### **Hardware tools**

* **laptop Computer**: - To store files, prepare documentation, install and use required software for development
* **Internet Connector (broadband cable)**: - To get relevant information from the internet.
* **Printer:** - helps to print documentation
* **Mobile phone:** - to check the design implementation of the system
* **USB type-c cable: -** to connect the development PC with the mobile phone
* **Mouse**

### **Software tools**

* **Window 10:** - This Operating system is to be used to install and run the development software needed for this project.
* **Browser:** - like chrome, used for retrieving information for designing and developing and also used for running designing tools like.
* **Microsoft Word:** -For writing documentation.
* **Microsoft PowerPoint:** -will be used for preparing the presentation of this project documentation.
* **Draw.io:** - used to design the UML diagram for the project.
* **Visual Studio code:** -IDE, a code editor used for implementation.
* **Figma:** - for UI designing
* **GitHub:** - VCS and team collaboration tool, also used to contact the advisor.
* **Android Studio**
* **Expo Go**

### **Implementation Languages**

* **Node.**js
* **Django**
* **Python 3.11**
* **Redux toolkit**
* **React native**

## **Significance of Project**

After the completion of the project, the system developed will provide the listed important services to the following parties:

For Bajaj drivers:

* Solve the availability problem of passengers
* Provide places where their services are required
* Ensure their security
* Provide means to save fuel consumption
* Provide means for work as a side job

For passengers:

* Make bajaj readily available when the need arises
* Ensure their security
* Removes the need to stand by the side of the road and wait for bajaj
* Pick their start and finish location and know how much it cost beforehand

## **Beneficiaries of the project**

We can see our beneficiaries as direct beneficiaries and indirect beneficiaries.

Our direct beneficiaries are:

* Bajaj drivers
* Passengers (riders)
* The owning company that gets profit from the users

Our indirect beneficiaries are:

* The government further divided to
  + The financial sector (Tax and revenue minister)
  + Regional police force
  + The community of Gondar

## **Feasibility Study**

**Economic Feasibility**

The proposed system is economically feasible because the opportunity cost of the proposed system on the market is very compensating in the long run. Since the market is open for the service, the production and deployment cost is high but the financial gain to be extracted from it is higher.

**Technical Feasibility**

In terms of technical feasibility, the project uses already existing technologies and concepts we have already learned. Thus, we have both the technology and the required personnel for it.

**Operational Feasibility**

The proposed project is operationally feasible because the earlier identified problems are addressed to the best extent. Further, the users will be presented with an easy-to-use UI hence promoting use.

## **Project schedule**

The project schedule is more extended than initial estimations. none the less the schedule Gantt chart is developed with the initial given start date. The chart was developed using an online tool called `teamgantt.com`.

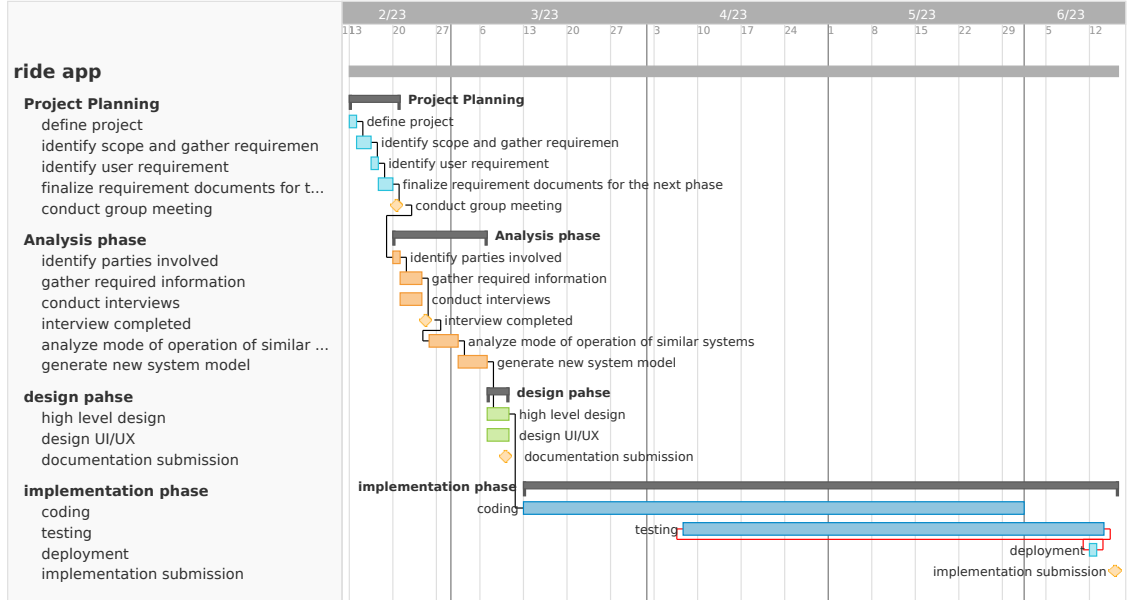


Figure 1.1: Schedule Gantt Chart

## **Project Budget**

**The budget for the project for tangible expenses is collected from a team member, but most technology expenses written are used on a trial basis thus their costs are not incurred expenses but rather estimated expenses.**

|  |  |  |  |
| --- | --- | --- | --- |
| *Item* | Quantity | Unit Price | Total cost in birr |
| *Paper* |  | ETB 2 | ETB 50 |
| *Google API* |  |  |  |
| *Transport cost* | 5 | ETB 20 | ETB 100 |
| *Total* |  |  | 480 |

# **CHAPTER TWO**

1. **Requirement Analysis**

## 

## **Current System Description**

The existing transport system is a manual one. From the passenger’s point of view whoever needs to use a transport goes directly to the road and waits for a transport to come whether a taxi or bajaj. After that, if they have any luggage, they go back to their houses for loading transportation or go out with their luggage to the road, sometimes if they have the phone numbers of bajaj drivers they call from home. As for the driver’s point of view, they drive around looking for passengers or stay in place waiting for a client to call or come talk to them.

## **The major function of the current system**

* Negotiable pricing
* Found everywhere
* Doesn’t require literacy

## **The problem with the Existing System**

* Time wasting
* No record of travels
* Prone to criminal activities like theft, kidnapping, rape, and murder
* Costs more than the set tariff
* not accessible to everyone
* the feeling of insecurity by both driver and riders

## **Requirement Gathering**

## **Requirement Gathering Methods**

The methodologies that were used to gather information for the development of the project are interview and document analysis.

**Interview**

As an interview, we went to the transport bureau of Gondar city and asked for the required information and we got referred to Mrs. Tigist Ayeragn. And from the information we got, we found out about the existence of ZayRide in Gondar and went to their office too.

**Document analysis**

Since there are similar ride applications, we used their documents to learn and understand more about the project requirements. we found documents on the mode of operations of systems from abroad.

We reviewed various diagrams and documentation(developers’ and users’) from Uber.

## **Business Rules**

Business rules are a set of guidelines and constraints that define how a business operates. They are used to ensure that business processes are consistent, efficient, and effective. Business rules can be used to enforce policies, regulations, and best practices, and they can be applied to any aspect of a business, from customer service to product development. [2] We have listed the business rule statements below.

1. Drivers must follow traffic laws and drive safely at all times.
2. Riders must treat drivers and other riders with respect and refrain from any behavior that makes others feel unsafe or uncomfortable.
3. Transparent pricing and billing.
4. Drivers and riders are prohibited from using drugs or alcohol while using the platform.
5. Both drivers and riders must adhere to community standards and policies, including those related to discrimination, hate speech, and harassment.
6. Drivers and riders are encouraged to report any safety concerns or issues
7. A driver currently a client must identify himself as occupied
8. A driver cannot serve more than one order at a time without the rider’s permission
9. A rider cannot refuse payment after a service
10. A rider refusing to pay is enforceable by law
11. A rider cannot cancel an order after the order has started
12. Compliance with local regulations and laws, including licensing requirements and safety standards

## **Proposed System Description**

## **Overview**

We are in a technology era. This is signified by large access to smartphones, the internet, and other technologies. In this technology era, by exploiting this access a lot of countries and cities have changed the way they operate a lot of things from manual to computer-based. The transportation system is one of them. Similarly, many cities in Ethiopia have adopted systems like rides as a possible alternative for transport.

That being said such systems have not been implemented in Gondar city administration. The current high demand for transportation and high number of thefts related to bajaj transports, as more kidnappings related to bajaj make such systems appear in demand.

The system we propose is similar to that of the others currently operational in Addis Ababa and others. The extra capability we propose is for it to support bajaj drivers as service givers. The proposed system is signified by live GPS tracking of the trip, easy UI/UX both for both the rider and driver, storing the history of travel details, and they can know how much distance they are traveling and how much is left.

The proposed system will have big impact on the tourism of Gondar by both international and local tourists because it will provide convenient transportation, especially for those who have never visited the historical city. Thus, our system will play great role on the secure and sustainable development of Gondar.

The other support our system may provide will be in the work of Gondar city administration’s police. In Gondar crimes involving bajaj are high. Although it is declining in the central part of the city it is the same as ever in the outer skirts of the city. The help our system will provide here is it will put location tags and details on both registered drivers and riders. As a result, if any crime or accident were to occur involving the users the system will help the police identify each individual.

## **Functional Requirements**

Functional requirements are a set of specifications that define what a system or product should do. They describe the features, functions, and capabilities that are required to meet the needs of the users and stakeholders.[3]

The functional requirements of the proposed system are classes of functions that have further subfunctions and are listed as follows.

* User registration and login
* Driver and passenger verification
* Ride booking and scheduling
* Real-time tracking of drivers and passengers
* Driver and passenger ratings and reviews
* Integration with mapping and navigation services
* Customizable settings for users, such as payment view and ride preferences.
* Database management for the administrator
* Calculating trip cost:calculate fare based on factors more than travel distance

## **Nonfunctional requirements**

## **Performance**

In terms of performance, the system proposed is aimed to address various aspects to the maximum possible. The first is number of user requests. Others are fast response time and efficiency, which are designed to work by demanding low CPU, memory, and the already existing operating system on the device.

## **Scalability**

As for the scalability aspect, the system proposed will have an elastically scalable character, this will be good for current small user starts and still give the intended service regardless of increasing user traffic on the system.

## **Availability**

The availability of the system is a server-side concern, as long as the server that’s leased is up the system is designed to be up 24/7 and the availability of night shift bajaj drivers becomes the key controller too.

## **Reliability**

The system proposed is not designed to have failures raising from prolonged use because of intended periodic updates. But worst case, if it was to happen it can be reported via user feedback and resolved promptly.

## **Security**

Since the account creation process is designed to work with OTP and not create individual accounts with usernames and passwords the security issue is not a big concern. All that is because accounts are created with phone numbers and OTP(one-time password) confirmation numbers sent directly to the registering users' phone number.

## **Environmental**

According to environmental nonfunctional requirements, the system we propose doesn’t even need any actions. It uses already existing technologies like smartphones. Thus, it doesn’t affect the environment significantly making the requirement of addressing green computing issues and reducing carbon footprint motions useless.

## **Usability**

The design of the UI is aimed to be easy for all users whether proficient with smartphones or not. It's designed to be easily understandable with clear instructions and steps of application.

# 

# **Chapter Three**

1. **System Model**

## **Scenario**

## **Use Case Model**

Actors**:** In the use cases an actor interacts with the system to perform a piece of meaningful work that helps them to achieve a goal and has access to define their overall role in the system and the scope of their action. Depending on the above explanation, the actors in this system are the following:

* Driver: These are individuals who use the Bajaj ride app to accept ride requests, pick up riders, and provide transportation services.
* Riders: These are individuals who use the Bajaj ride app to request rides and make payments for the service.
* Admins: These are individuals who are responsible for managing the Bajaj ride platform, including overseeing driver and rider accounts, monitoring safety and security, and maintaining the technical infrastructure.

The purpose of a use case is to provide a clear and detailed description of the interactions that occur between a user and a system or application. This helps to ensure that everyone involved in the development process has a shared understanding of the system or application's functionality and requirements.[4] The most important and basic use cases of this system are the following: -

1. View list of customer request
2. Manage database
3. View list of customers
4. View and manage available drivers
5. View available requests
6. View trip detail
7. Search and view trip history
8. View trip summary
9. Track trip progress
10. Rate ride
11. Log out
12. Log in
13. Register
14. Request ride
15. Cancel ride
16. Accept ride
17. Start ride
18. End ride

## 

## **Use Case Diagram**

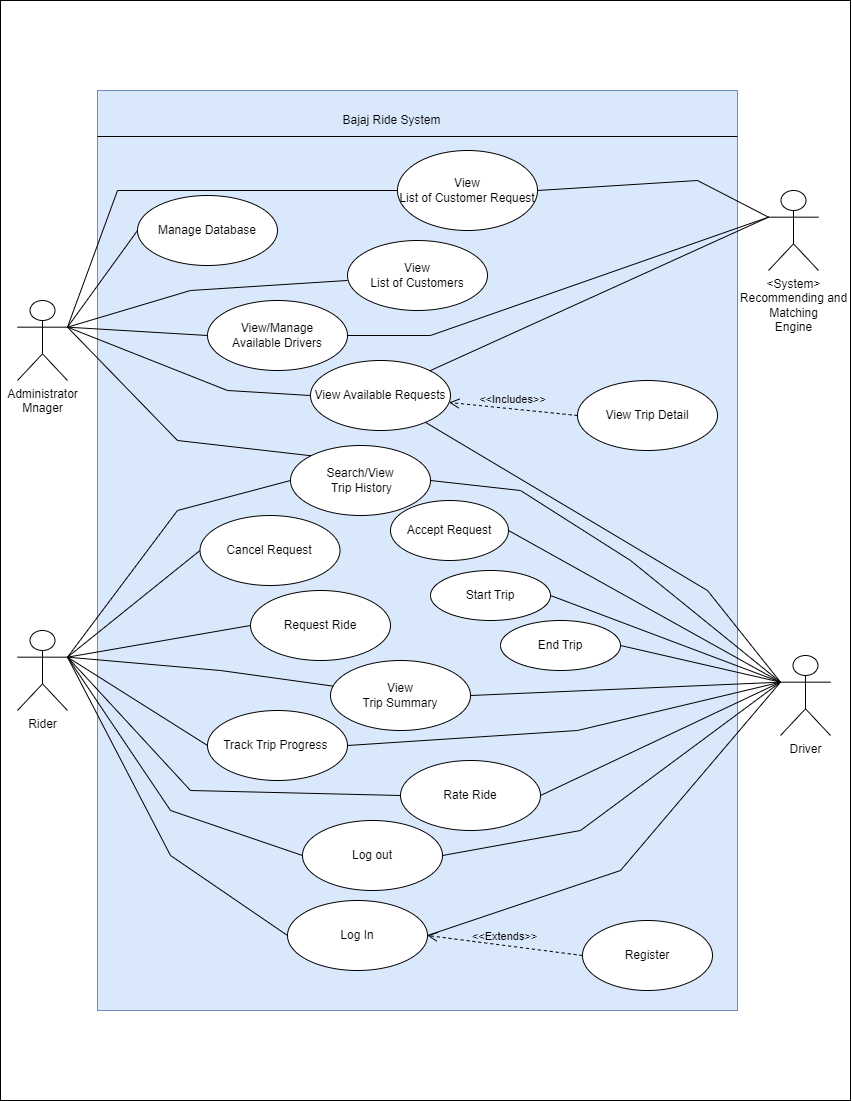


Figure 3.1 Use Case Diagram

## **Description of Use Case Model**

Table 3.1: Use case description for Login

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_001 | |
| **Use Case Name** | Login | |
| **Actor** | Rider, Driver | |
| **Description:** | Users can log in to their account with their registered phone number and OTP password to access the Bajaj ride app. | |
| **precondition** | User must have a registered account with Bajaj ride. | |
| **Post condition** | User is logged in and has access to the app's features. | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪User opens the Bajaj ride app. 2. 🡪 User enters their registered phone number 3. 🡨 user enters given OTP as password | 1. 🡨 System sends OTP to the entered phone number. 2. 🡨 System verifies password and logs the user into their account and displays the app's home screen. |
| **Alternative Flows** | 1. If the user enters an incorrect password, the system will display an error message and prompt the user to try again. 2. If the user enters an unregistered phone number, the system will take the user to account setup interface. | |

Table 3.2: Use case description for Request ride

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_002 | |
| **Use Case Name** | Request ride | |
| **Actor** | Rider | |
| **Description:** | Riders can send a request for a ride. | |
| **precondition** | Rider must be logged in to their Bajaj ride account. | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 Rider opens the Bajaj ride app. 2. 🡪 Rider selects the "Where to?" option. 3. 🡪Rider enters their destination. 4. 🡪Rider confirms the ride request 5. 🡪Driver accepts the ride request. | 1. 🡨System displays the estimated cost and wait time for the ride. 2. 🡨System searches for a nearby available driver and assigns the ride to them. 3. 🡨System provides the rider with the driver's name, photo, and vehicle information. 4. 🡨System notifies the rider that the driver is on their way. 5. 🡨 engine updates list of active requests |
| **Post condition** | Rider's ride request has been sent and assigned to a driver. | |
| **Alternative Flows** | None | |

Table 3.3: Use case description for Track trip progress

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_003 | |
| **Use Case Name** | Track trip progress | |
| **Actor** | Rider, Driver | |
| **Description:** | Riders and drivers can track the ride's progress and estimated time of arrival to the destination through the Bajaj ride app. | |
| **precondition** | Actors must have an active ride request. | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 Rider opens the Bajaj ride app. 2. 🡪 Rider selects the "Track Ride" option. | 1. 🡨 System displays the ride's progress and ETA to the user. |
| **Post condition** | Rider is able to track the ride's progress and ETA. | |
| **Alternative Flows** | None | |

Table 3.4: Use case description for Rate ride

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_004 | |
| **Use Case Name** | Rate ride | |
| **Actor** | Rider, Driver | |
| **Description:** | Riders can rate their driver and ride experience after their ride has ended. | |
| Driver rates the ride experience exchanged with riders. | |
| **precondition** | Rider and driver must complete ride the ride. | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 user opens the Bajaj ride app. 2. 🡪user selects the "Rate Ride" option. 3. 🡪 user selects the rating for the driver. 4. 🡪user may provide optional comments to provide feedback | 1. 🡨 System displays the ride details and prompts the user to rate their trip and the other user(rider or driver) on a scale of 1-5. 2. 🡨 System records the rating and feedback and updates the user’s profile with the rating. |
| **Post condition** | Rider's rating and feedback has been recorded and the driver's profile has been updated. | |
| **Alternative Flows** | None | |

Table 3.5: Use case description for View and manage available drivers

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_005 | |
| **Use Case Name** | View and manage available drivers | |
| **Actor** | Administrator Manager, Engine | |
| **Description:** | System automatically updates available drivers list. | |
| administrator can modify or delete drivers on the available drivers list. | |
| **precondition** | None | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 administrator views result and approves | 1. 🡪 system check drivers list and active trips list. 2. 🡪 system updates available drivers list from the above lists |
| **Post condition** | Available drivers list is updated for request assignment. | |
| **Alternative Flows** | 1. 🡪 administrator modifies the resulting available drivers list | |

Table 3.6: Use case description for Manage database

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_006 | |
| **Use Case Name** | Manage database | |
| **Actor** | Administrator | |
| **Description:** | The administration manager manages database of entire system | |
| **precondition** | none | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 manager access, modify and maintain system database |  |
| **Post condition** | Database maintained | |
| **Alternative Flows** | none | |

Table 3.7: Use case description for View trip summary

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_007 | |
| **Use Case Name** | View trip summary | |
| **Actor** | Rider, driver | |
| **Description:** | Rider and driver view entire summary of their trip. | |
| **precondition** | The trip must be finished. | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 user opens the Bajaj rider app. 2. 🡪rider and driver view finished trip information | 1. 🡪System displays summary page |
| **Post condition** | Riders continue to rating and payment | |
| **Alternative Flows** | None | |

Table 3.8: Use case description for Search and view trip history

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_008 | |
| **Use Case Name** | Search and view trip history | |
| **Actor** | Administrator, rider, driver | |
| **Description:** | Users view their past trip history and detailed information. | |
| Administrator views trip history of all customers. | |
| **precondition** | Trips must have been conducted in the past | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 user opens bajaj ride application. 2. 🡪 user goes to history page | 1. 🡪 System shows individual trips list |
| **Post condition** | None | |
| **Alternative Flows** | 1. 🡪 administrator opens all conducted trips from database | |

Table 3.9: Use case description for Register as Driver

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_009 | |
| **Use Case Name** | Register as driver | |
| **Actor** | Driver | |
| **Description:** | Users can apply to become a Bajaj ride driver. | |
| **precondition** |  | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 User opens the Majaj ride app. 2. 🡪 User selects the "Become a Driver" option. 3. 🡪 User provides their personal and driver information. 4. 🡪 System Administration verifies the user's information and approves their driver application. |  |
| **Post condition** | User's driver application is approved. | |
| **Alternative Flows** | 3a. If the user's driver information is incomplete or inaccurate, the system prompts the user to update their information.  4a. If the user's driver application is rejected, the system notifies the user and provides a reason for the rejection. | |

Table 3.10: Use case description for View Request

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC\_010 | | |
| **Use Case Name** | View available requests | | |
| **Actor** | Driver, Engine, administrator | | |
| **Description:** | Driver can view available or assigned ride requests. | | |
| Administrator and the system view list of all active ride requests that haven’t been completed. | | |
| Engine updates the available request list when passengers make requests. | | |
| **precondition** | Driver must be logged into their Bajaj ride account. | | |
| User must make ride requests. | | |
| **Main Flow** | **User action** | | **System response** |
|  | 1. 🡪 Driver opens the Bajaj ride Driver app. 2. 🡪 Driver selects the "View Requests" option. 3. 🡪 Driver can view ride requests and their details, such as pick-up location, drop-off location, and fare amount. | |  |
| **Post condition** | Driver has viewed their ride requests. | | |
| **Alternative Flows** | 1. 🡪Rider makes request 2. 🡪 |  | |

Table 3.11: Use case description for Accept Request

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_011 | |
| **Use Case Name** | Accept request | |
| **Actor** | Driver | |
| **Description:** | Driver must be logged into their Bajaj ride account. | |
| **precondition** | Driver must have viewed a ride requests. | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 Driver selects the assigned request. 2. 🡪 Driver confirms that they want to accept the ride request. 3. 🡪 Driver is directed to the pick-up location of the rider. |  |
| **Post condition** | Driver has accepted the ride request. | |
| **Alternative Flows** | 2a. 🡪driver declines the assigned request .  2b🡪 driver select request manually | |

Table 3.12: Use case description for Start Ride

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_012 | |
| **Use Case Name** | Start ride | |
| **Actor** | Driver | |
| **Description:** | Driver starts the ride on meeting the rider to. | |
| **precondition** | Driver must have accepted a ride request. | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 Driver arrives at the pick-up location. 2. 🡪 Driver confirms that the rider is present. 3. 🡪 Driver selects the "Start Ride" option. 4. The ride officially begins and the system starts tracking the ride. |  |
| **Post condition** | Driver has started the ride and both users can track progress. | |
| **Alternative Flows** | None | |

Table 3.13: Use case description for End Ride

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_013 | |
| **Use Case Name** | End ride | |
| **Actor** | Driver | |
| **Description:** | Driver can end the ride once rider reaches the requested final destination. | |
| **precondition** | Driver must have started a ride. | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 Driver arrives at the drop-off location. 2. 🡪 Driver confirms that the rider has been dropped off. 3. 🡪 Driver selects the "End Ride" option. 4. 🡪 The rider pays the fare with cash. | 1. 🡨 The system calculates the fare amount and displays it to the driver and rider. |
| **Post condition** | Driver has ended the ride and rider has paid the fare. | |
| **Alternative Flows** | 2a. If the rider requests to change the drop-off location, the driver can select the "Change Destination" option and follow the updated directions.  4a. If there is a dispute over the fare amount, the driver and rider can contact Bajaj ride customer service for assistance. | |

Table 3.14: Use case description for View Earnings

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_014 | |
| **Use Case Name** | Cancel ride | |
| **Actor** | Rider | |
| **Description:** | The rider cancels the ride request before the driver reaches or before the trip starts. | |
| **precondition** | Trip request must be made. | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 The rider opens bajaj ride application. 2. 🡪 rider choose “cancel” option | 1. 🡪 . show rider current order. 2. 🡪 The system updates available request list |
| **Post condition** | The rider has canceled request.  The system notifies the driver of cancelation. | |
| **Alternative Flows** | none | |

Table 3.15: Use case description for Register

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_015 | |
| **Use Case Name** | Register | |
| **Actor** | Rider | |
| **Description:** | New users that setup their account after verifying their phone numbers with OTP. | |
| **precondition** | User login with new phone numbers. | |
| **Post condition** | User is logged in and is taken to account setup page. | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪User enters phone number and OTP . 2. 🡪 User enters name, email, image(optional) | 1. 🡨 application takes user to account creation page. 2. 🡨 name and email are saved in database |
| **Alternative Flows** | 1. If the user enters an invalid email, the system will display an error message and prompt the user to try again. 2. If the user enters an image of >2MB, the system will ask user to try another picture. | |

Table 3.16 Use case description for View trip detail

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_016 | |
| **Use Case Name** | View trip detail | |
| **Actor** | Driver, administrator, engine | |
| **Description:** | Driver sees the detail of a request before accepting ride. | |
| **precondition** | Request must be selected | |
| **Post condition** | Driver accepts or decline ride request | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪driver selects the request. . 2. 🡪 driver sees details of the request | 1. 🡨 application shows details page |
| **Alternative Flows** | None | |

Table 3.17 Use case description for View list of customers

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_017 | |
| **Use Case Name** | View list of customers | |
| **Actor** | Administrator | |
| **Description:** | Administrator views list of all users registered on the system. | |
| **precondition** | None | |
| **Post condition** |  | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪administrator enters credentials 2. 🡪 accesses database and sees detailed information of all customers. | 1. 🡪 system verifies credentials and allows access |
| **Alternative Flows** | None | |

Table 3.18 Use case description for View list of customer request

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_018 | |
| **Use Case Name** | View list of customer request | |
| **Actor** | Administrator, engine | |
| **Description:** | View list of all customer requests(finished, ongoing, active, canceled)  The engine views this list to sort or update other related lists | |
| **precondition** |  | |
| **Post condition** |  | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 administrator requests access all list of requests 2. 🡪 administrator/ engine reads table | 1. 🡪 system grants access to administrator |
| **Alternative Flows** |  | |

Table 3.19 Use case description for Log out

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_019 | |
| **Use Case Name** | Log out | |
| **Actor** | Rider, Driver | |
| **Description:** | The users log out of the system to stop the service. | |
| **precondition** | User must be logged in | |
| **Post condition** |  | |
| **Main Flow** | **User action** | **System response** |
|  | 1. 🡪 user chooses the log out option 2. 🡪 user confirms to log out | 1. 🡪 system asks for confirmation. 2. 🡪 user services are terminated and logged out 3. 🡪 log in page viewed |
| **Alternative Flows** | 3a. 🡪 user denies log out  4a. 🡪 system cancels log out processes and goes back | |

## **Data Dictionary**

* **User Account table**: This contains data about users of the Uber Ride app, such as their names, contact information, account information.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Key | Field Size | Description | Required? | Accepts null value? |
| UserId | Integer | PK | 8 | Unique ID for the user | Y | N |
| First\_Name | Varchar |  | 50 | First name of the user | Y | N |
| Last\_Name | Varchar |  | 50 | Last name of the user | Y | N |
| Email | Varchar |  | 50 | Email of the driver |  | N |
| Phone\_No | Integer |  | 10 | Phone number of the user | Y | N |
| Password | Varchar |  | 8 | Password of the user |  |  |
| Rating | float |  | 5 | Average rating of user |  |  |

Table 3.20 User Account Table

* **Driver Table**: This contains data on the drivers who offer rides through the Uber Ride application, including their names, contact information, rating and information about their vehicles (license number, car plate number).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Key | Field Size | Description | Required? | Accepts null value? |
| D\_Id | Integer | PK | 8 | Unique identifier each driver | Y | N |
| User\_id | Varchar | FK | 50 | Foreign key to the corresponding user | Y | N |
| Rating | Varchar |  | 50 | Average rating of the driver | Y | N |
| License\_number | Varchar |  | 50 | Driver's license number | Y | N |
| plate\_number | Integer |  | 10 | License plate number of the driver's car | Y | N |
| Insurance\_info | Varchar |  | 20 | Name of insurance company | Y | N |
| Insurance\_id | varchar |  | 15 | Renewed insurance identification number | Y | N |
| Rating | float |  | 5 | Average rating of driver | N |  |

Table 3.21 Driver Table

* **Ride Table**: Information on the pickup location, destination, length of the ride, cost, and other pertinent factors are all kept in this table for the rides.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Key | Field Size | Description | Required? | Accepts null value? |
| Ride\_Id | Integer | PK | 8 | Unique identifier for each ride | Y | N |
| RiderId | Varchar |  | 8 | Foreign key to the corresponding rider | Y | N |
| DriverId | Varchar |  | 8 | Foreign key to the corresponding driver | Y | N |
| Pickup\_location | Varchar |  | 50 | Starting location of the ride | Y | N |
| destination | Varchar |  | 50 | Ending location of the ride | Y | N |
| Fare | Integer |  | 5 | Total fare of the ride | Y | N |
| Started\_at | Datetime |  | 10 | Timestamp of when the ride was started |  |  |
| Ended\_at | Datetime |  | 10 | Timestamp of when the ride was ended |  |  |

Table 3.22 Ride Table

* **Admin Table**: This contains data who manages the rides through the Uber Ride application, including their names, contact information and information.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Key | Field Size | Description | Required? | Accepts null value? |
| Id | Integer | PK | 8 | Unique ID for the user | Y | N |
| Name | Varchar |  | 50 | First name of the user | Y | N |
| Email | Varchar |  | 50 | Email of the driver |  | N |
| Phone\_No | Integer |  | 10 | Phone number of the user | Y | N |
| Password | Varchar |  | 8 | A password of the admin | Y | N |

Table 3.23 Admin Table

* **Location Table**- This table contains data about the areas where Uber operates, including geographic information, traffic statistics, and other pertinent data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Key | Field Size | Description | Required? | Accepts null value? |
| LocationId | Integer | PK | 8 | Unique identifier each rating |  |  |
| Longitude | Varchar |  | 12 | The score of the rating |  |  |
| Latitude | Varchar |  | 12 | The id of the driver who provided the ride |  |  |

Table 3.24 Locations Table

## **Class Model**

A class model is a kind of diagram that shows the static organization of a system or application. It is made up of classes, interfaces, and connections between them such association, dependency, and inheritance. A class model’s purpose is to provide a high-level view of the structure of a system or application that can help stakeholders to understand the components of the system and how they are related to each other. Class models are commonly used in software development to design the structure of software systems.[5]

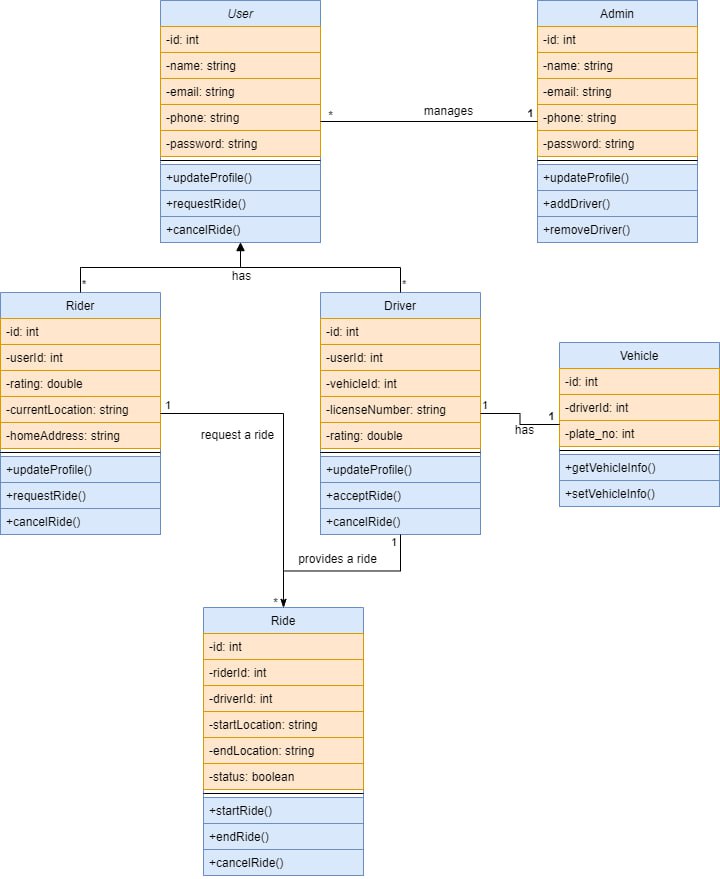


Figure 3.2 Class Model Diagram

## **Dynamic Modeling**

A dynamic model is a type of UML model that represents the dynamic behavior of a system or application. It consists of state machines, activity diagrams, sequence diagrams, and other types of diagrams that show how the system or application responds to external stimuli and how it changes over time.[5] We will have two of the aforementioned dynamic diagrams: an activity diagram and a sequence diagram.

## **Activity Diagram**

An activity diagram is a type of diagram in UML that represents a workflow or process. It consists of nodes and edges that represent activities and the flow between them.

They are made for the purpose of providing a visual representation of processes which can help stakeholders to understand the flow of activities, the dependencies between them, and any decision points or alternate flows within the process. They are commonly used to model business processes.[6]

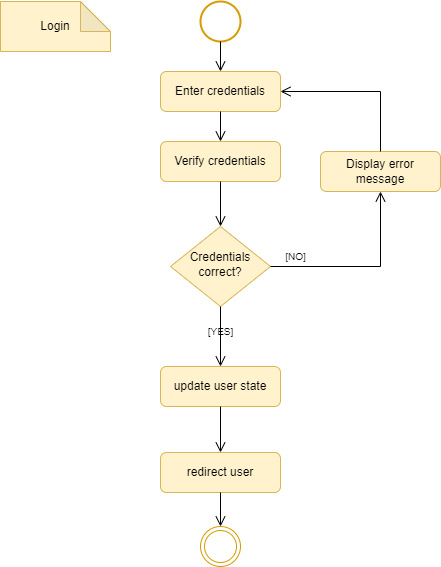


Figure 3.3 Activity Diagram for Log in

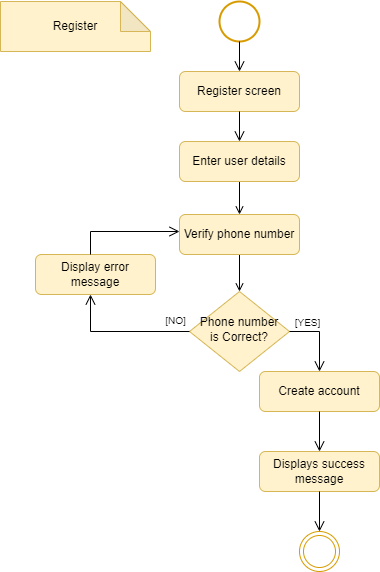


Figure 3.4 Activity Diagram for Register

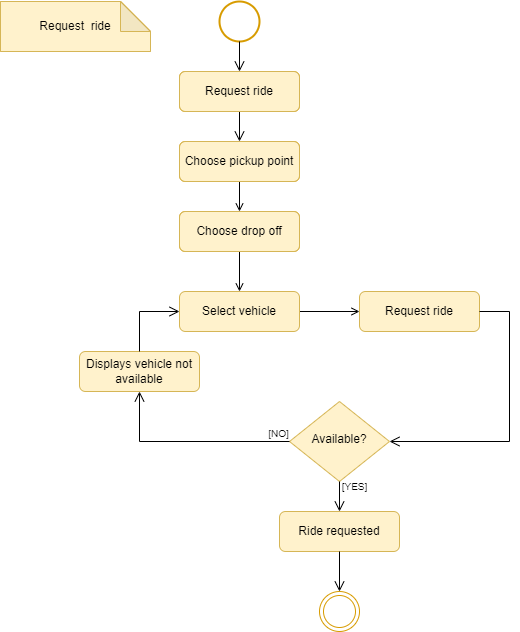


Figure 3.5 Activity Diagram for a ride request

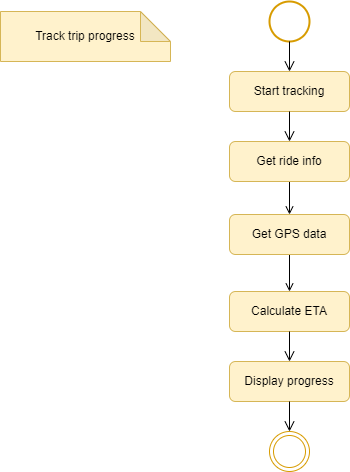


Figure 3.6 Activity Diagram for tracking trip progress

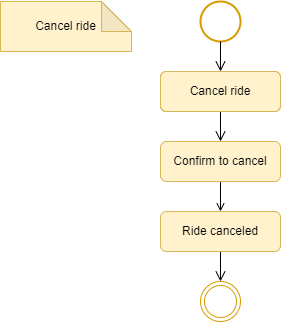


Figure 3.7 Activity Diagram for Cancel Ride

## **Sequence diagram**

A sequence diagram is a type of diagram that shows dynamic changes in the system upon interaction. It shows the sequence of interactions between objects or components in a system or application.[7] It consists of lifelines, which represent objects or components, and messages, which represent the interactions between them.

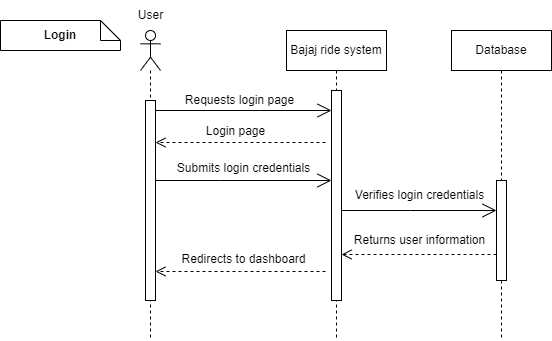


Figure 3.8: sequence diagram for login

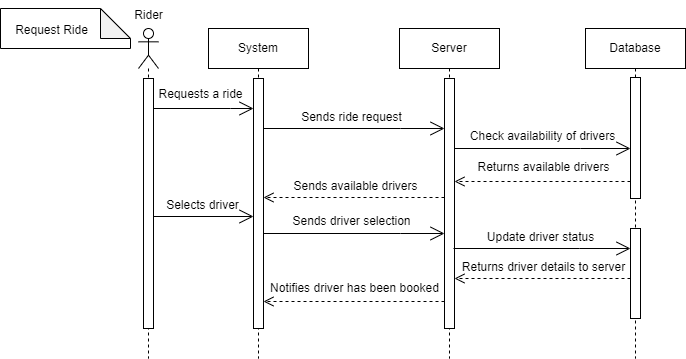


Figure 3.9 sequence diagram for a ride request

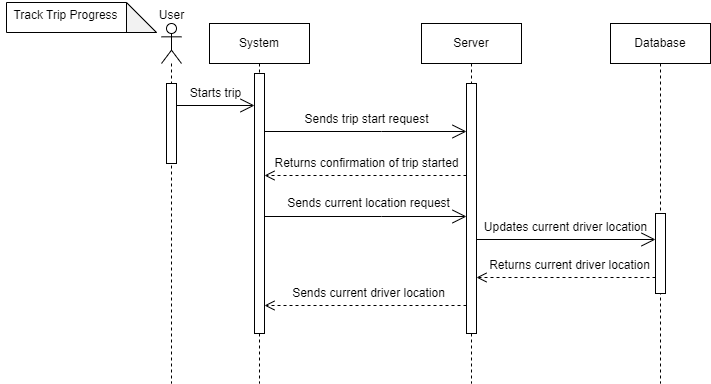


Figure 3.10 Sequence diagram for tracking trip progress

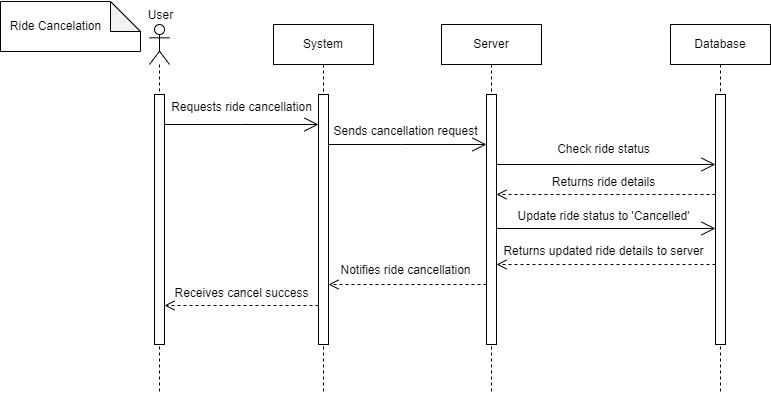


Figure 3.11 sequence diagram for canceling a ride

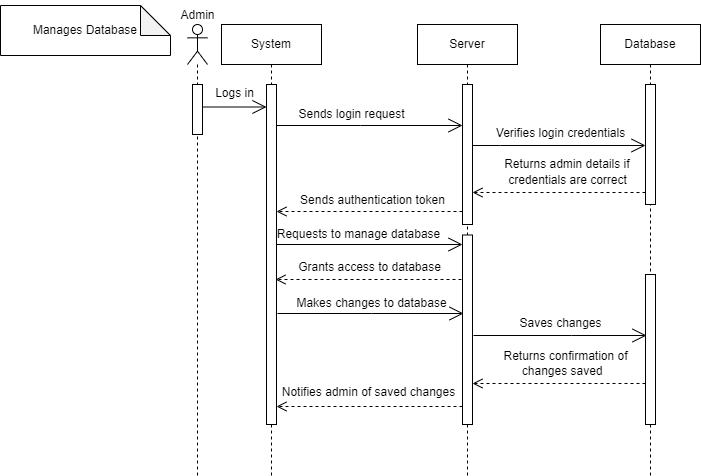


Figure 3.12 Sequence diagram for managing the database

## **User Interface**

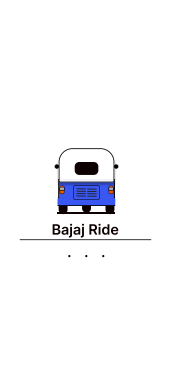
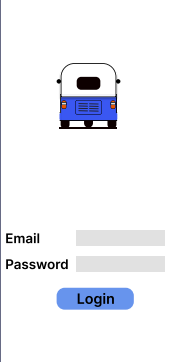


Figure 3.14 UI for Log in screen

Figure 3.13 UI for Loading screen

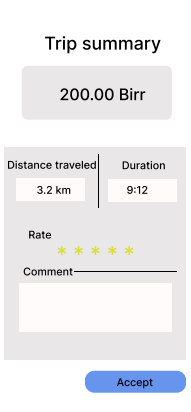
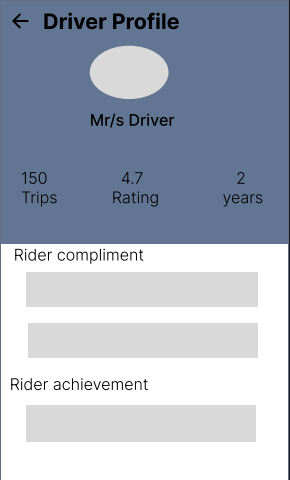


Figure 3.16 UI for trip summary screen

Figure 3.15 UI for Account view screen

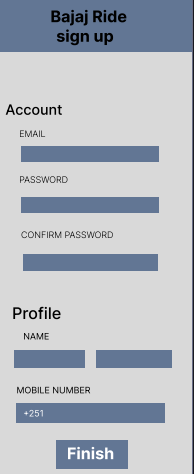


Figure 3.17 UI for Registration screen

# 

# **Chapter Four**

1. **System Design**

## **Introduction**

The purpose of designing the Bajaj ride system is to provide a clear direction on how the mobile application should be built, and to obtain the necessary information required to drive the actual implementation of the system. This process is based on a comprehensive understanding of the model the Bajaj ride system is built on, and it focuses on breaking down the system into manageable parts.

System design for the Bajaj ride system involves transforming the analysis model into a system design model. This is the first step towards getting into the solution domain in the development of the software. The design process involves creating a blueprint for the system, which outlines how different parts of the Bajaj ride system will work together to provide a seamless user experience.

Through the system design process, the Bajaj ride system is broken down into manageable components, including the front-end interface, server-side logic, and database management. Each of these components is designed to work together seamlessly to provide a reliable and efficient ride service to users.

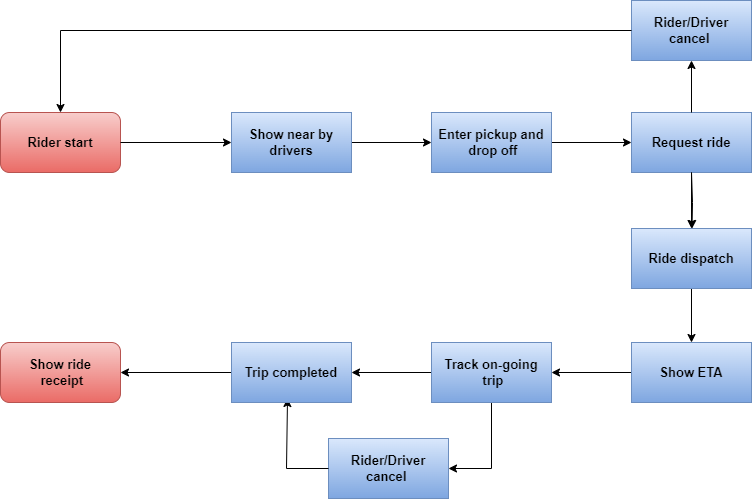


Figure 4.0‑1 Life cycle of a trip

1. **Ride start**: This component initializes the ride-hailing process and allows the user to begin requesting a ride.
2. **Show nearby drivers**: This component displays the available drivers near the user's location and enables the user to select a driver.
3. **Enter pickup and drop off**: This component allows the user to input the pickup and drop off locations for the ride.
4. **Request ride**: This component enables the user to request a ride from the selected driver.
5. **Ride dispatch**: This component dispatches the selected driver to the user's location.
6. **Show ETA**: This component shows the estimated time of arrival for the driver to the user's location.
7. **Track on-going trips**: This component enables the user to track the ride's progress and monitor the driver's location.
8. **Trip completed**: This component indicates the end of the ride and allows the user to rate the driver and provide feedback.
9. **Rider/Driver cancel**: This component allows either the rider or driver to cancel the ride in case of unforeseen circumstances.
10. **Show ride receipt**: This component displays the ride details, including fare and payment information, for the user's reference.

## **Proposed software architecture**

## **System Decomposition**

System decomposition for the Bajaj ride system involves breaking down the overall ride-hailing system into smaller, more manageable subsystems or components. This enables the system to be designed, developed, and maintained more effectively.

1. **User Interface Subsystem**: The user interface subsystem is responsible for providing a simple interface for users to interact with the ride system. It includes screens that allow users to request rides and track ride progress. The subsystem includes the following components:

* Ride Request Screen: Allows users to enter their pickup and drop-off locations and request a ride.
* Driver Selection Screen: Displays the available drivers near the user's location and allows the user to select a driver.
* Ride Tracking Screen: Allows users to track the progress of their ride and monitor the driver's location.

1. **Ride Request Subsystem**: The ride request subsystem is responsible for processing ride requests and dispatching drivers to pick up passengers. It includes various components that work together to ensure a seamless ride-hailing experience for users. The subsystem includes the following components:

* Ride Request Processing: Receives ride requests from users and processes them to match the user with the nearest available driver.
* Driver Matching: Matches the user with the nearest available driver and provides the user with driver information.
* Ride Confirmation and Dispatch: Dispatches the selected driver to the user's location and confirms the ride.

1. **Driver Management Subsystem**: The driver management subsystem is responsible for managing drivers and ensuring they provide quality service to riders. It includes various components that work together to manage driver registration, approval, location tracking, rating, and feedback. The subsystem includes the following components:

* Driver Registration and Approval: Registers new drivers and approves them for service.
* Driver Location Tracking: Tracks the location of drivers
* Driver Rating and Feedback Management: This enables users to rate drivers and provide feedback to help improve the quality of service.

1. **Rider Management Subsystem**: The rider management subsystem is responsible for managing riders and ensuring they have a smooth and convenient ride-hailing experience. It includes various components that work together to manage rider registration, authentication, profile management, and support. The subsystem includes the following components:

* Rider Registration and Authentication: Registers new riders and authenticates them for service.
* Rider Profile Management: Manages rider profiles, including personal information and ride history.
* Rider Support: Provides rider support, including help with ride issues and account-related issues.

1. **Data Management Subsystem**: The data management subsystem is responsible for managing rider and driver data, system logs, database management, and backups. It includes various components that work together to ensure the system operates efficiently and effectively. The subsystem includes the following components:

* User and Driver Data Management: Manages user and driver data, including personal information and contact details.
* System Logs and Error Handling: Manages system logs and error handling to ensure the system operates smoothly.
* Database Management and Backups: Manages system databases and backups to ensure data is secure and accessible.

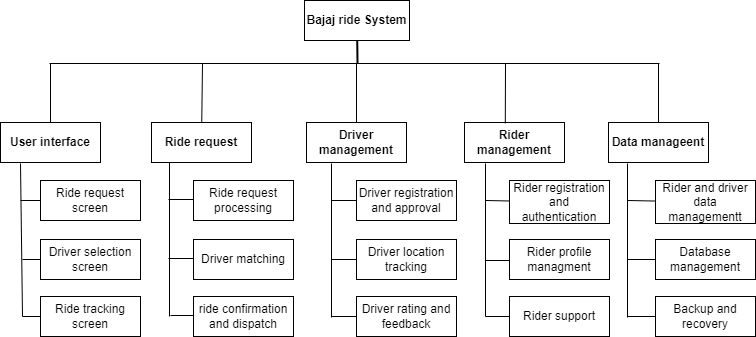


Figure 4.2 System decomposition diagram

## **Hardware/ software mapping**

Hardware/software mapping is a process in software engineering that involves mapping software functionality to the hardware components that are responsible for executing it. The purpose of this process is to ensure that software components are designed to take advantage of the capabilities of the underlying hardware and to optimize the overall performance and efficiency of the system.[8]

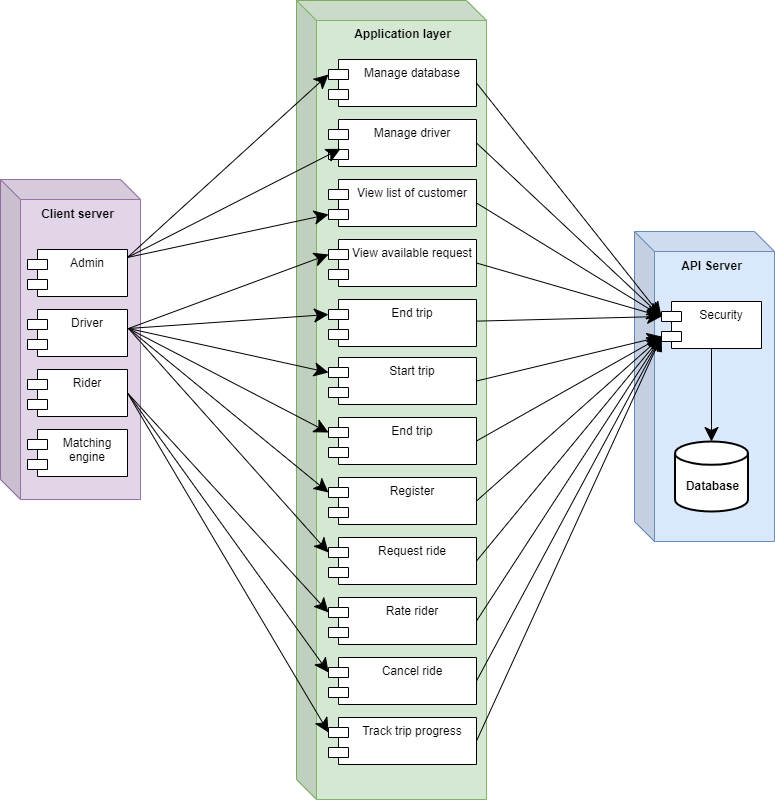


Figure 4.3 Deployment Diagram

## **Persistent data modeling**

A persistent data modeling diagram is a visual representation of the structure of a database that shows how data is organized and related to each other. Its purpose is to provide a clear and concise view of the data model, which can be used by developers, designers, and stakeholders to understand the data requirements of a system.[9]

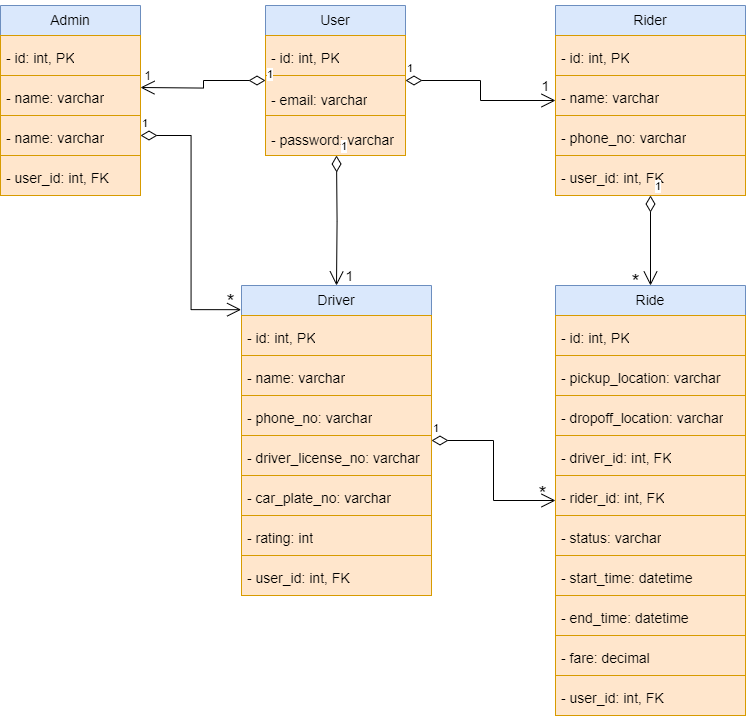


Figure 4.4 Persistent Data Modeling Diagram

## **Access control and security**

Access control and security are critical components of any software system, including a Bajaj ride system. To ensure the security and privacy of users and their data, it is important to implement appropriate access control and security measures.

Access control refers to the mechanisms used to restrict or grant access to resources in a system. In a Bajaj ride system, access control would be used to ensure that only authorized users can access certain parts of the system, such as driver or rider accounts, or ride requests.

Security refers to the measures taken to protect the system and its users from unauthorized access, data breaches, or other forms of malicious activity. In a Bajaj ride system, security measures would include things like encryption of sensitive data, secure communication protocols, and user authentication and verification.

Here are some specific access control and security measures that could be implemented in a Bajaj ride system:

1. User authentication: Require users to log in with a username and password, or implement other forms of authentication such as 2FA.
2. Role-based access control: Assign different roles to users (e.g. driver, rider, admin) and restrict access to certain parts of the system based on those roles.
3. Encryption: Use encryption to protect sensitive data such as user passwords, and ride histories.
4. Secure communication: Use secure communication protocols such as HTTPS to protect user data in transit.
5. Regular security audits: Conduct regular security audits to identify and address vulnerabilities in the system.

By implementing these and other access control and security measures, the Bajaj ride system can ensure the safety, privacy, and security of its users and their data.

## **Detailed class diagram**

## **Package Diagram**

A package diagram is a type of UML diagram that shows the dependencies between the packages that make up a software system. Its purpose is to provide a high-level view of the system's architecture and organization, and to help identify the relationships between different parts of the system.[8]

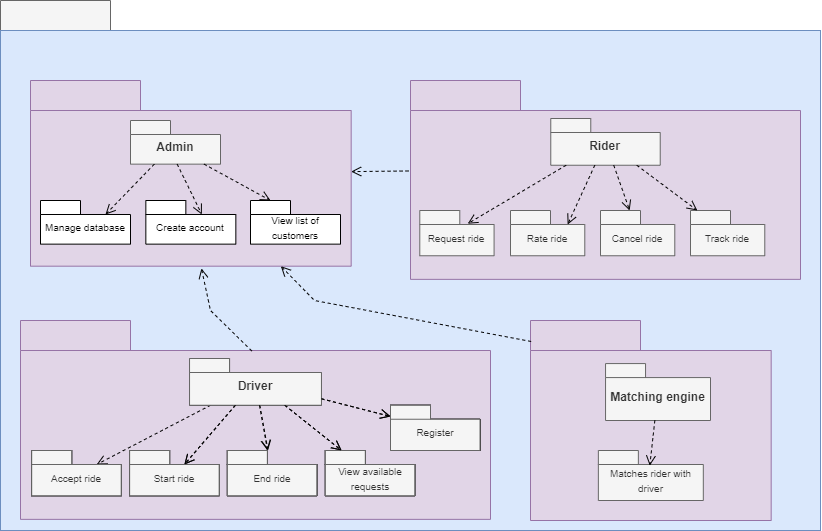


Figure 4.5 Package Diagram

## **Deployment**

Deployment is the act of preparing a software application for use by setting it up on the hardware and software environment where it will operate. This usually involves transferring the software application from a development or testing environment to a production environment, where it can be utilized by end-users.

For our project the steps for deploying the app are as follows:

* Setting up the infrastructure: set up a server to host the application and a database to store data. This can be done using a cloud provider like Amazon Web Services or by setting up our own server and database.
* Configuring the environment: You'll need to configure the environment on the server to support the Bajaj ride app. This involves installing the necessary software dependencies and setting up any required environment variables.
* Building and packaging the application: Once the environment is set up, building and package the application into a format that can be deployed on the server. This can be done using a build tool like Gradle.
* Deploying the application: deploying the application to the server using a deployment tool like Jenkins or Ansible. This may involve copying the packaged application to the server and starting it up using a CLI.
* Testing and monitoring: Once the app is deployed, we test it to make sure it's functioning correctly.

[1] IvyPanda, “Uber Company,” *https://ivypanda.com/essays/uber-company/*, Apr. 29, 2023.

[2] W. M. Temm, “New Kid on the Block: The ALWD Citation Manual,” 2007.

[3] G. S. Jackson and S. A. Haugerud, “Functional requirements,” *fib Bulletin 96. Guidelines for Submerged Floating Tube Bridges*, 2020.

[4] BABOK Guide committee, *A Guide to the Business Analysis Body of Knowledge*, 3rd ed. International Institute of Business Analysis (IIBA), 2015.

[5] International technology standards consortium, *UML 2.5 Specification*. Object Management Group, 2017.

[6] International technology standards consortium, *UML 2.5 Specification*. Object Management Group, 2017.

[7] International technology standards consortium, *UML 2.5 Specification*. Object Management Group, 2017.

[8] R. S. Pressman, “Software Engineering: A Practitioner’s Approach By Roger Pressman,” 2020.

[9] D. Shah and S. Slaughter, “Chapter III Data Modeling and UML,” 2019.