**Low-Level Architecture and Data Models**

**P07:Hitcherr**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **23100337** | **Irtza Tariq** |
| **22100021** | **M Muneeb Arshad** |
| **23100020** | **Ebad Ur Rehman** |
| **23100333** | **Nashrah Shaukat** |
| **23100040** | **Mustafa Afzal** |

|  |  |  |
| --- | --- | --- |
| **Content** | **Totals** | **Obtained** |
| Architecture diagram | 30 | 10 |
| Architecture justification | 20 | 0 |
| E/R diagram | 30 | 10 |
| E/R diagram description | 20 | 7 |
| Late submission |  |  |
| **Total** | **100** | **27** |
| **Individual Evaluation** |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

1. Introduction

2. System Architecture

2.1 Architecture Diagram—As it is in the prototype code

2.2 Architecture Diagram—As it should-be

3. Data Models

4. Tools and Technologies

5. Who Did What?

6. Review checklist

# **1.** **Introduction**

<Give an overview of the project here. The overview must highlight the overall objectives of the project and its potential users and customers.>

The project hitcher has been created keeping in mind the global fuel hike. Social media and general surveys suggest that people prefer carpooling over long distances in order to save travel costs. A cab hailed from an application may cost a customer over PKR 1000 for a distance of more than 25 kilometers. This application aims to reduce that cost by accommodating multiple customers looking to commute across the same route.

The niche customer base for this project comprises people who commute over long distances every day and wish to save on cost. According to financial classes, this application aims to accommodate lower- and middle-class customers. A survey at LUMS showed that people who commute from Bahria Town and Johar Town prefer a carpooling service due to the high cost of commute.

Apart from the major objective, the application serves to provide relief to the drivers and customers in a plethora of other forms. The application aims to provide drivers an incentive to drive more by providing them with 100% of the receipts from each ride after a certain threshold of rides per day has been attained. This was brought up by drivers who disliked the fact that they had to pay a cut to the company for every ride that they completed. This also brought a culture of drivers canceling rides and completing rides unofficially.

This culture brought a problem of security for customers as they could not be tracked by the ride hailing companies and in turn could not complain if something went wrong. To bring end to end security, we introduced a feature of QR codes. The rider must scan the QR code on the driver’s phone to start a ride that would act as an acknowledgement to start a ride and would provide a timestamp for accurate fare calculations.

In conclusion, our application aims to overcome the problems faced by drivers and customers who use other ride hailing services and we believe that this will provide a more satisfactory service.

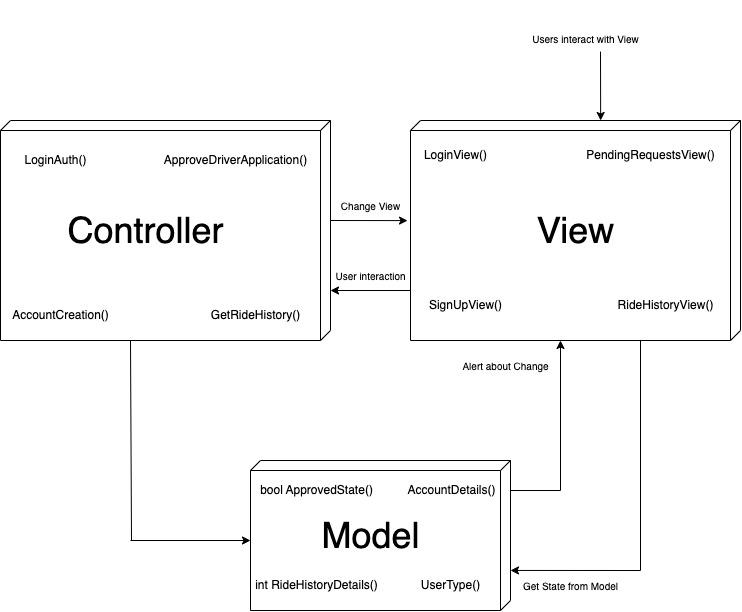
# **2.** **System Architecture**

## **2.1** **Architecture Diagram—As it is in the prototype code**

<Draw a diagram of the system architecture. The diagram must reflect the architecture of the code that you have written for prototype. For instance, if you are following layered architecture, your diagram must distinguish all layers along with classes/modules in each layer. You must mention exact names of classes/modules inside the layers as they are in your prototype code. If there are too many classes, your diagram must show classes/modules involved in at least two use cases. >

[Add specific components in each subsystem, e.g., in model, view and controller. It is not a norm to write function names. Are there no business or data components in your system? In description of the architecture, don’t write use case steps. Instead, you should specify components and their interactions. See slides for details.]

[No architecture justification given.]



Use Case: Approve Driver

The admin will go to the PendingRequestsView() and from there click the approve button on the selected driver. This will cause the controller ApproveDriverApplication() to interact with the model and change this driver's state to Approved. The driver may log into the system and carry out other functions.

Use Case: Ride History

The logged in driver/user will see the RideHistoryView() which will fetch the ride history of that account. This is fetched from the RideHistoryDetails() which is updated after every ride carried out by the driver/user.

Use Case: Log In

User/Driver will navigate to the LogInView(), there they will enter their account details. The user type will be checked by the controller LogInAuth() which fetches the account type from the model UserType()

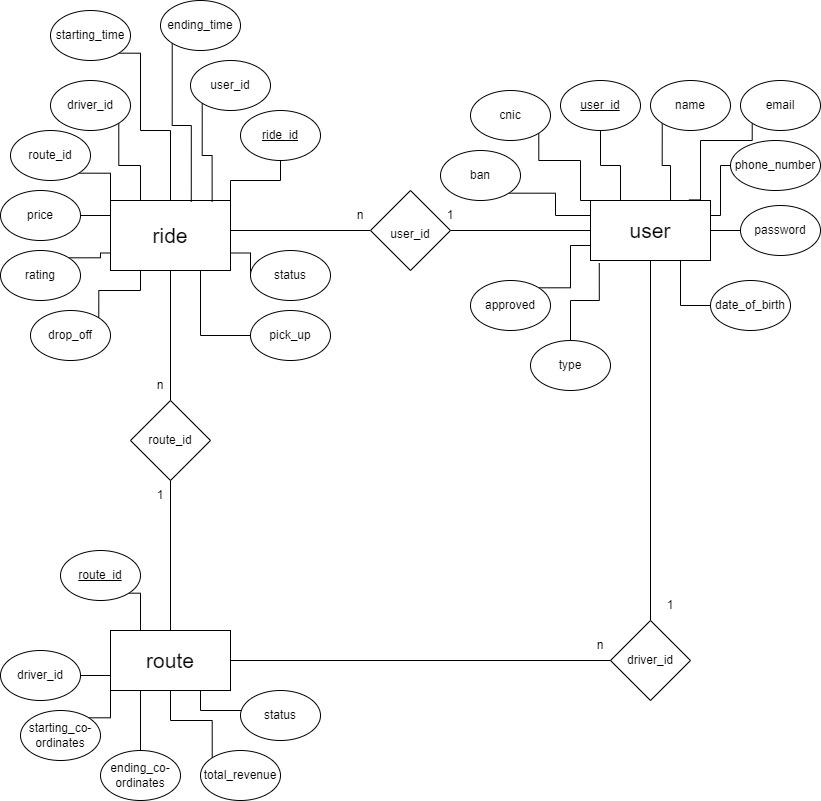
Use Case: Sign Up

User/Driver will navigate to the SignUpView() where they will enter their wanted account details. This will then be saved by calling the AccountCreation() controller which saves the account as an object in the model AccountDetails()

# **3.** **Data Models**

<Create data model of your system. For example, E/R diagram. Give brief description of entities and data fields.>

[Are there just three entities in your whole system? Payments? Authentication and authorization? Ratings?]



|  |  |  |
| --- | --- | --- |
| Entity Name: user | | |
| Attribute Name | Type | Description |
| user\_id | reference | Primary key. |
| name | string | Name identifier for the user |
| phone\_number | number | User’s phone number |
| email | string | User’s email |
| date\_of\_birth | timestamp | User’s date of birth for birthday discounts etc |
| type | number | Type of the user i.e. Ride(1), Driver(2), Admin(3) |
| approved | boolean | Approval status |
| ban | boolean | Ban flag |
| cnic | string | Government issued id |
| password | string | Secured hash of passwords for authorization and changing password |

|  |  |  |
| --- | --- | --- |
| Entity Name: route | | |
| Attribute Name | Type | Description |
| route\_id | reference | Primary key. |
| driver\_id | reference | User\_id of the driver offering |
| status | boolean | Ongoing status |
| total\_revenue | number | Total revenue of the driver |
| starting\_co-ordinates | geopoint | Starting coordinates of the route |
| ending\_co-ordinates | geopoint | Ending coordinates of the route |
| starting \_time | timestamp | Starting time of the route |
| ending\_time | timestamp | Ending time of the route |

|  |  |  |
| --- | --- | --- |
| Entity Name: ride | | |
| Attribute Name | Type | Description |
| Ride\_id | reference | Primary key. |
| user\_id | reference | Reference of the rider taking the ride |
| route\_id | reference | Reference of the route which the driver offered |
| price | number | Amount paid by the rider |
| rating | number | Rider’s rating of the experience out of 5. |
| starting \_time | timestamp | Starting time of the ride |
| ending\_time | timestamp | Ending time of the ride |
| pick\_up | geopoint | Pick up coordinates |
| drop\_off | geopoint | Drop off coordinates |
| status | boolean | Ongoing status |

# **4.** **Tools and Technologies**

<List down tools and technologies that you are using for development and deployment. Make sure that you mention name and version of the tools.>

The project will consist of two applications. One will be a mobile application for the

customers and the drivers and the other would be a web application suitable for the

administrators. The following tools will be used for development purposes.

For the mobile application we would be using flutter for front-end and firebase for

backend.

Flutter:

- Version 3.3.4

- Flutter is a free and open-source mobile UI framework created by google for

mobile app development. Since it is easy to learn and use, we will be using it for

the front end.

Firebase:

- Latest release

- Fire base is a google backed application development software which will be

used as the back-end for the application. It boasts a real-time database along

with multi-platform authentication.

For the web application we would be using Firebase for the back and React JS,

Express JS and Node JS for the front end.

Express JS:

- Version 4.18.2

- Express is a node js web application framework that provides broad features for

building web and mobile applications

React JS:

- Version 18.2.0

- The React. js framework is an open-source JavaScript framework. It's used for

building interactive user interfaces and web applications quickly and efficiently

with significantly less code than you would with vanilla JavaScript

Node JS:

- Version 16.9.0

- Node. js lets developers use JavaScript to write command line tools and for

server-side scripting

For both web and mobile applications we would be using Selenium for testing purposes.

Selenium:

- Version 4.5.0

- Selenium is a free (open-source) automated testing framework used to validate

web applications across different browsers and platforms.

For deployment purposes we will be using AWS.

AWS:

- Latest Version

- AWS CodeDeploy is a fully managed deployment service that automates

software deployments and will be used for the deployment of both web and

mobile applications

-

Throughout the project a project management tool, Trello will be used.

Trello:

- Latest version

- Helps manage the project, workflow and with task management among the team.

# **5.** **Who Did What?**

# 

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| M Muneeb | Introduction |
| Mustafa | Architecture Diagram - As it is in the Prototype |
| Ebad | Data Models |
| Irtza | Architecture Diagram |
| Nashrah | Tools and Technologies |

# **6.** **Review checklist**

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

# 

|  |  |
| --- | --- |
| **Section** **Title** | **Reviewer Name(s)** |
| Tools and Technologies | M Muneeb Arshad |
| Introduction | Mustafa Afzal |
| Tools and Technology | Irtza |
| Data Models | Ebad |
| Architecture Diagram | Nashrah |