



Dharmsinh Desai University, Nadiad

**Faculty of Technology, Department of Computer
Engineering**

B. Tech. CE Semester – V
Subject: (CE – 515) Advanced
Technologies Project Title:

Cab Booking System

By:
Ashish Kotecha
Samyak Mehta

Guided By:
Prof. Prashant M.Jadav
Prof. Parth R.Dave



Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

CERTIFICATE

This is to certify that Advanced Technologies Project entitled “**Cab Booking System**” is the bonafied report of work carried out by

1) Ashish Kotecha

2) Samyak Mehta

Of Department of Computer Engineering, Semester V, academic year 2019-2020, under our supervision and guidance.

Guide

Prof. Prashant M. Jadav
Associate Professor of Department
of Computer Engineering,
Dharmsinh
Desai University, Nadiad

Guide

Prof. Parth R Dave
Assistant Professor of Department of
Computer Engineering, Dharmsinh
Desai University, Nadiad

HOD

Dr. C. K. Bhensdadia
Head of Department of Computer
Engineering, Dharmsinh Desai
University, Nadiad

Table of Contents

1. Abstract	4
2. Introduction	5
3. Software Requirements Specification	6
4. Design	13
5. Implementation	23
6. Testing	27
7. Screen-shots	29
8. Limitation and Future Extension	35
9. Conclusion	36
10. Bibliography	37

Abstract

The term “TAXI” strikes in as a public transport thought oriented inclination. Nonetheless, in transportation, a taxi is a type of vehicle for hire with a driver, used by a single passenger or small group of passengers often for a non-shared ride. A taxi cab conveys passengers between locations of their choice. In modes of public transport, the pickup and drop-off locations are determined by the service provider and by the passenger, although demand responsive transport and share responsive transport and share taxis provide hybrid/taxi mode.

Introduction

1 BRIEF INTRODUCTION



[Current System in Market]

Cab booking a web application. This application allows you to book a ride from pre-defined source and destination and it shows the fare that will have to be paid by the user . You can view your booked ride and also view the date and time of the rides for reference in case of any fault. The admin would also be able to add the routes and view all the bookings of the users in the system and users will be able to travel to different destinations any time.

1.1 PURPOSE

The main objective of cab booking system is to provide user friendly website so that normal Travellers can book the cabs and confirm it. The user can also see the rides that have been completed by the user till now by checking its own history with the application.

1.2 ABBREVIATIONS

- **BOOKING:**
-The request of a client to serve with a vehicle on a specified date and time.
- **REGISTRATION:**
-The users are registered to the system while request a booking. The employees are registered by the admin.
- **MAINTENANCE:**
-Users' information's are maintained in a separate log for maintenance.
- **DELETION:**
-The admin can delete any employee or client.
- **VIEW HISTORY:**
-Admin can view all the users booking and and user can see its own booking.

OVERALL DESCRIPTION:

1.3 PRODUCT PERSPECTIVE:

Book the rides, viewing the history and the cab book pricing.

1.4 PRODUCT FUNCTIONALITY:

Book the cab
Fare estimator
View booked ride
View maps

1.5 CONSTRAINTS

1. The system won't be able to reach to the remote places where there users are less and not specified in database.
2. Clients or employees do not have any rights to edit any data in the system.

1.6 ASSUMPTION AND DEPENDENCY

1. Vehicles are already purchased and available for use.
2. Roles and responsibilities are already established.
3. Administrator is already created.
4. Username is valid email addresses of respective user
5. Admin has the authority to add/delete user accounts.
6. Admin has the authority to delete drivers.

Tools/Technologies Used

Technology:

- HTML
- CSS3
- BOOTSTRAP4
- ANGULAR cli
- TypeScript
- NodeJS
- Express JS
- MongoDB

TOOLS

- Visual Studio Code

PLATFORMS

- localhost:4200 for Angular
- localhost:8000 for NodeJS
- mongodb://127.0.0.1:27017

3 Software Requirement Specification

The SRS is organised with the functional requirements such that admin, traveller's major functionalities are described from signing up, log in, booking cab.

3.1 Types of Users:-

1. users
2. admin

3.2 System Functional Requirements

R.1 - {User Authentication-Sign Up}

- Input : User Details
- Output : Data Stored Successfully
- Description : User Enters Details Like User Id, Name, Password, Phone Number, E-Mail As Per Type Of User.

R.2- {User Log-In}

- Input : User Credentials
- Output : User Logged In Account/Error Message
- Description : User Enters The Username/Email And Password And Checks Into The Web Application By Validating In Database.

R.3 - {Log Out}

- Description : User Logs Out Of The Website.

[USERS]

R.1 {Booking Cab Ride}

- Pre-Requisites: User Must Be Logged In.
- Input : Travelling Path
- Output : Booking Confirmation
- Description: Traveller Enters Source And Destination And Sees The Best Possible Route And The Available Cab Riding On The Route. He/She Can Book The Ride And Travel.

R.1.1 {User Source-Destination Details}

- Input : Enter Path Details
- Output : Maps Display

R.2 {View Ride History}

- Description : The User Can View Rides He/She Has One Till Now If He/She Is Logged In.

R.3 {View Fare Charge}

- Input : Enter Travelling Route
- Output : Display Fare

[ADMIN]

R.1 {Manage Routes}

- description : admin decides the the route to be added as per his choice.

R.2 {Manage User Deletion}

Input : User Details

Output : User Deleted

Description : The Admin Can Delete The Traveller Or The Drivers On The Basis Of The Complaints.

R.2 {Manage User Updation}

Input : User Details

Output : User Details Updated

Description : The Admin Can Update The Traveller Or The Drivers On The Basis Of The Complaints.

R.3 { View User Rides}

Input: View command

Output: All users rides details.

3.3 EXTERNAL INTERFACES

- CAB DELAY ALERT SERVICE.
- BOOKING TERMINALS
- INTERACTIVE RESPONSE SYSTEM

3.4 Other Nonfunctional Requirements

1. Performance

The system must be interactive and the delays involved must be less. So in every action-response of the system, there are no immediate delays.

2. Safety

User details should be securely stored to the server. The main security concern is for user account hence proper login mechanism should be used to avoid hacking.

3. Reliability

As the system provides the right tools for discussion, problem solving it must be made sure that the system is reliable in its operations and for securing the sensitive details.

4. Database

System requires to access users data fast to maintain the performance.

4.Design

4.1 XML

```
<?xml version="1.0" ?>
<!DOCTYPE users SYSTEM "user.dtd">
<users xmlns="https://www.w3schools.com"
xmlns:xsi=http://www.w3.org/2001/xmlschema-instance
xsi:schemaLocation="https://www.w3schools.com/xml users.xsd">
  <user id="1">
    <name>Ashish</name>
    <username>AshishKotecha</username>
    <password>ashish</password>
    <email>ashishkotecha80@gmail.com</email>
    <contactno>9512450525</contactno>
  </user>
  <user id="2">
    <name>Samyak</name>
    <username>SamyakMehta</username>
    <password>samyak</password>
    <email>samyakjm@gmail.com</email>
    <contactno>79459950098</contactno>
  </user>
</users>
```

4.2 DTD

```
<!ELEMENT Users(User+)>
  <!ELEMENT User(name,username,password,email,contactno)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT username (#PCDATA)>
    <!ELEMENT password (#PCDATA)>
    <!ELEMENT EmailId (#PCDATA)>
    <!ELEMENT contactno (#PCDATA)>
  <!ATTLIST user id CDATA #REQUIRED>
```

4.3 XSD

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  attributFormDefault="qualified" elementFormDefault="qualified">
  <xs:element name="users">
    <xs:complexType>
      <xs:all>
        <xs:element name="user" minOccurs="0" maxOccurs="unbounded"
type="user"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
  <xs:all>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="username">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:minLength value="3"/>
          <xs:maxLength value="10" />
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="password">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:minLength value="6"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="contactno">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:minLength value="10"/>
          <xs:maxLength value="12"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="email">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="^[^@]+@[^\.]+\.\.+"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:all>
</xs:schema>
```

```

        </xs:simpleType>
        </xs:element>
    </xs:all>
</xs:element>
</xs:element>
</xs:schema>

```

4.4 XSLT

```

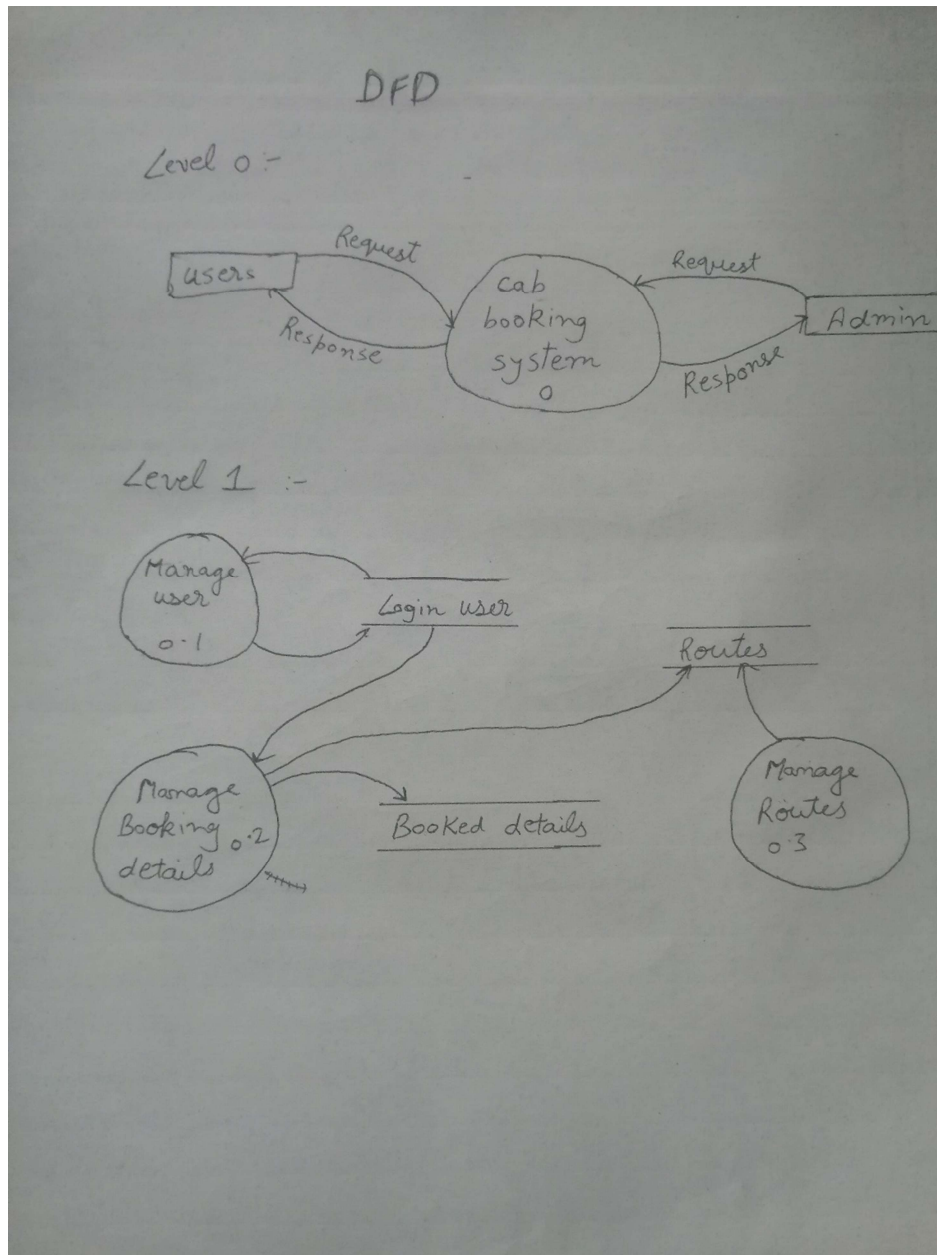
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html>
    <body>
    <h1>users</h1>
    <table>
        <tr>
            <th>id</th>
            <th>name</th>
            <th>username</th>
            <th>password</th>
            <th>email</th>
            <th>contactno</th>
        </tr>
        <xsl:for-each select="users/user">
            <tr>
                <td><xsl:value-of select="@id"/></td>
                <td><xsl:value-of select="name"/></td>
                <td><xsl:value-of select="username"/></td>
                <td><xsl:value-of select="password"/></td>
                <td><xsl:value-of select="email"/></td>
                <td><xsl:value-of select="contactno"/></td>
            </tr>
        </xsl:for-each>
    </table>
    </body>
    </html>
</xsl:template>
</xsl:stylesheet>

```

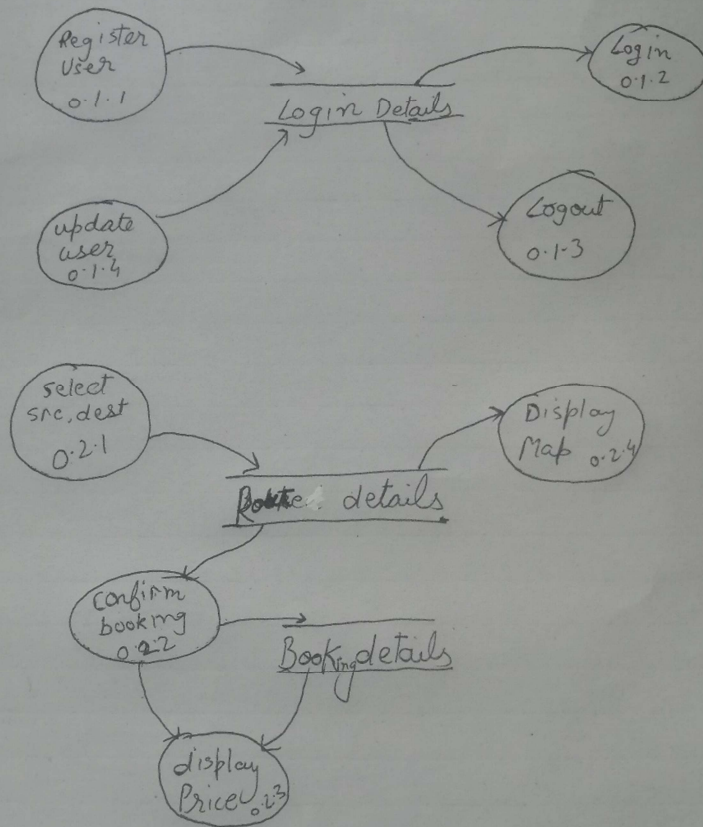


```
        </tr>
      </xsl:for-each>
    </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

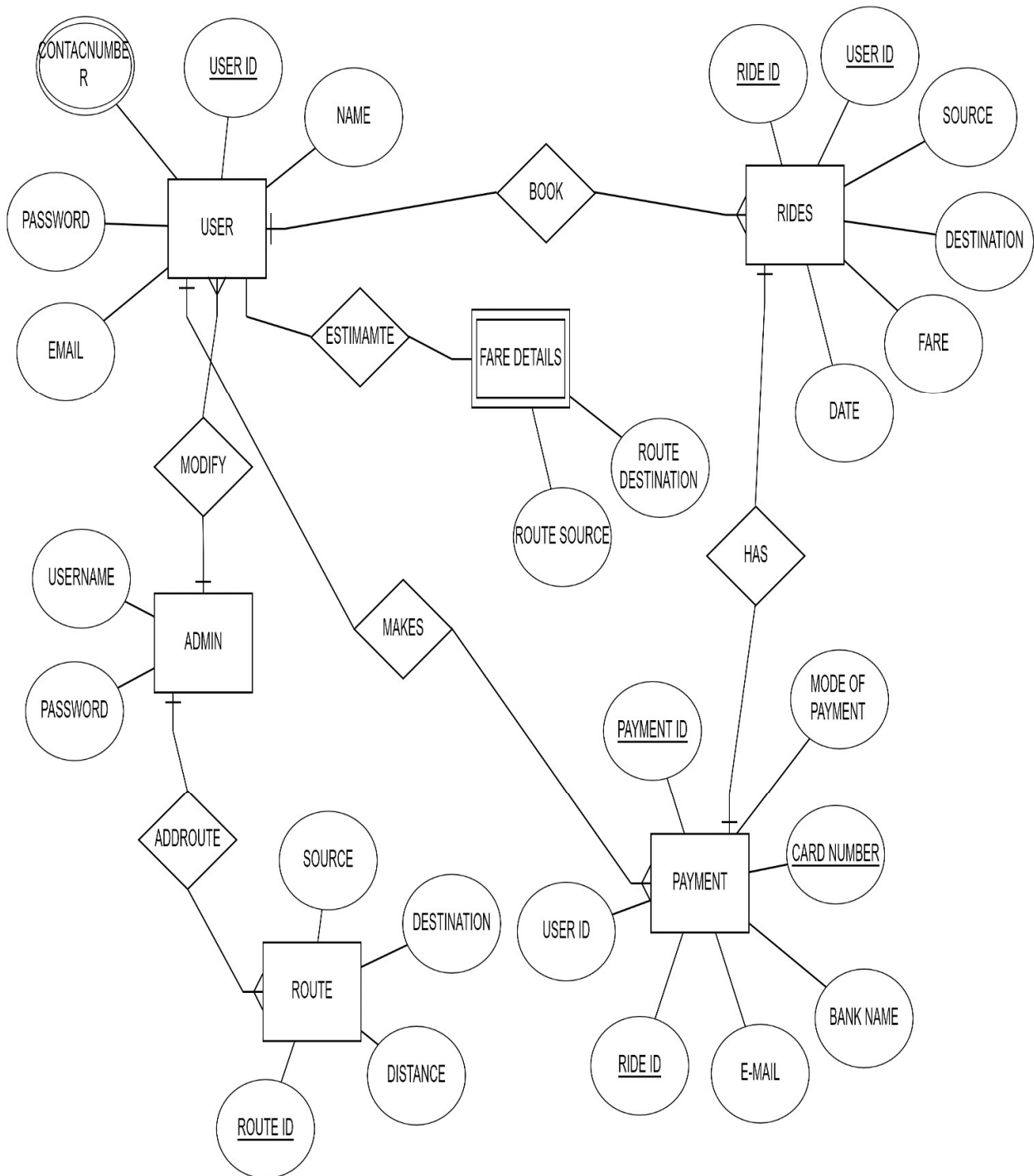
DFD



Level-2



4.5 E-R Diagram



4.6 Data Dictionary

<u>User</u>								
Sr. No.	Field Name	Data Type	Width	Required	Unique	Pk/Fk	Reference Table	Description
1	Username	Varchar	10	Yes	Yes	pk		
2	Name	Varchar	15	Yes	No			
3	Email	Varchar	50	Yes	Yes			
4	Password	Varchar	10	Yes	No			
5	Contact No	Number	15	Yes	No			

Note:Admin login is included along with User login.[Specific username and password is provided to admin].

<u>Admin</u>								
Sr. No.	Field Name	Data Type	Width	Required	Unique	Pk/Fk	Reference Table	Description
1	Email	Varchar	50	Yes	Yes	Pk		
2	Password	Varchar	50	Yes	No			

Rides

Sr. No.	Field Name	Data Type	Width	Required	Unique	Pk/Fk	Reference Table	Description
1	Ride id	Number	10	Yes	Yes	Pk		
2	Username	varchar	10	Yes	No	Fk	Travellers	
4	Source	Varchar	15	Yes	No			
5	Destination	Varchar	15	Yes	No			
6	Fare	Number	15	Yes	No			
7	Date	Date		Yes	No			

Routes

Sr. No.	Field Name	Data Type	Width	Required	Unique	Pk/Fk	Reference Table	Description
1	RouteId	Varchar	10	Yes	Yes	Pk		
2	Place1	Varchar	10	Yes	No			
3	Place2	Varchar	10	Yes	No			
4	Distance	Varchar	10	Yes	No			

Payment

Sr. No.	Field Name	Data Type	Width	Required	Unique	Pk/Fk	Reference Table	Description
1	Payment Id	Number	10	Yes	Yes	Pk		
1	Mode Of Payment	Varchar	10	Yes	No			
2	Card Number	Varchar	20	No	Yes			
3	Bank Name	Varchar	10	No	No			
4	E-Mail	Varhcar	10	Yes	Yes	Fk	Travelllers	
5	Fare	Number	10	Yes	No	Fk	Ride	
7	Ride No	Number	10	Yes	No	Fk	Ride	

5.Implementation Details

5.1 Modules Description

Main User module:

This module gives the main home page of the system which provides basic information about the website. It contains inpage navigation to go to different parts pf the pages. Tis module dosent needs authentication. It contains buttons for signin and login.

Input: User Selection

Output:Corresponding response

Registration module:

This module is used to store user's data to the database and enables the user to login to the system.All the fields in this module contains required validations.User can also navigate to login page if he/she has already registered

Input: User's Informations

Output: User Registered and redirect to login page

Processing: validating user's data and then storing them to database

Login module:

This module takes users credentions and then verifies it with registered users , if user is not registered the invalid credentials is shown else if they match with database then login user.

Input: User Credentials

Output: Logging user.

Processing: Verifying user credentials with the database.

User Booking module:

This module is accessed by authenticated user.

User is provided the booking page where the user gives source and destination and user books the ride by submitting it .

The user can only book those rides which are provided in the database . The user can view the rides and its details that on which dates the rides are covered.

Input: Booking Details

Output: Booking Confirmation if details are correct.

User Pricing module:

This module occurs when the user gives correct information and is already present in the database, then the user is redirected to see the fare that should be paid by the user to the driver at the end of the ride. Also on confirm booking Email is sent to user for convenience. The user can also view the google maps page as he/she confirms the booking.

Output: Fare details and Google Maps from source to destination if user wants to see.

Admin module:

This module is accessed only by Admin. It provides Admin to view User details with all the details. It also provides Admin to

Update the details of different users and delete the particular user if he/she wants to. The admin can also add routes in the application as per the availability of the drivers.

Input: User Selection

Output:Corresponding response

5.2 Functional Prototype

Login:

```
this.user.login(this.user1).subscribe( data => {
```

```
private _url1 :string = "http://localhost:8000/login"

login(user : User) {
  console.log(user)
  var user1=JSON.stringify(user)
  console.log(user1)
  return this.http.post(this._url1,user1, httpOptions)
}
```

Registration:

```
db.collection('user').insertOne(myData1, function (err, collection) {
  if (err)
  {
    throw err;
  }
})
```

```
private _url :string = "http://localhost:8000/user"
public e;
constructor(private http: HttpClient) { }
postUser(user : User) {
  console.log(user)
  var user1=JSON.stringify(user)
  console.log(user1)
  return this.http.post(this._url,user1, httpOptions)
}
```

> viewrid

Find Rides of user:

```
Ride.find({ username: uname},function (err,ride)
```

6. Testing

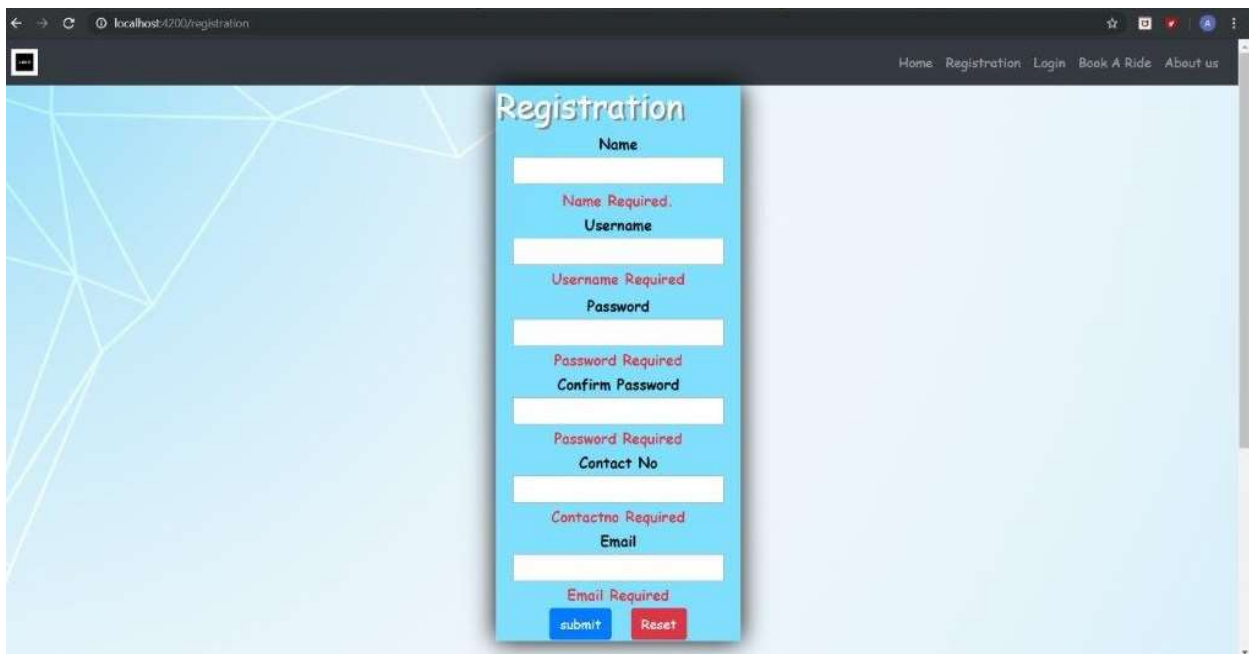
6.1 Test Methods

We have performed Black-box testing for the testing purpose.

6.2 Test Cases

For Registration:

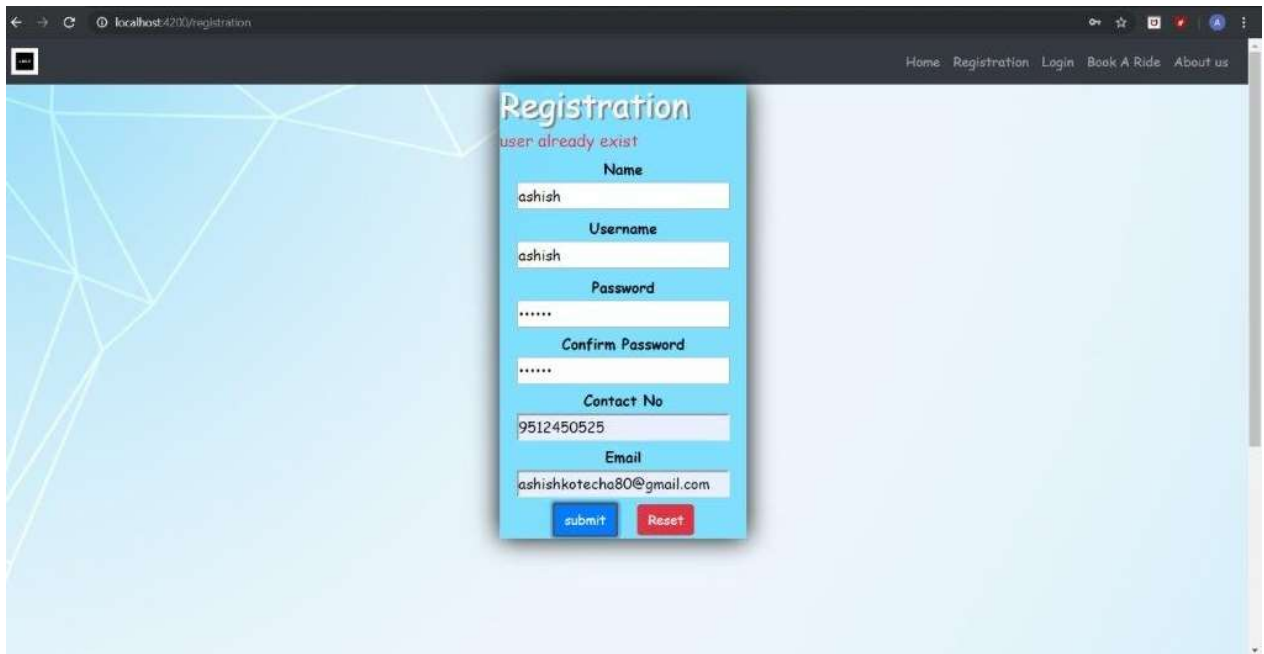
None of the fields should be empty.



The screenshot shows a web browser window with the address bar displaying 'localhost:4200/registration'. The page has a dark header with navigation links: Home, Registration, Login, Book A Ride, and About us. The main content area features a light blue background with a geometric pattern. A central white registration form is displayed, titled 'Registration'. The form contains the following fields and validation messages:

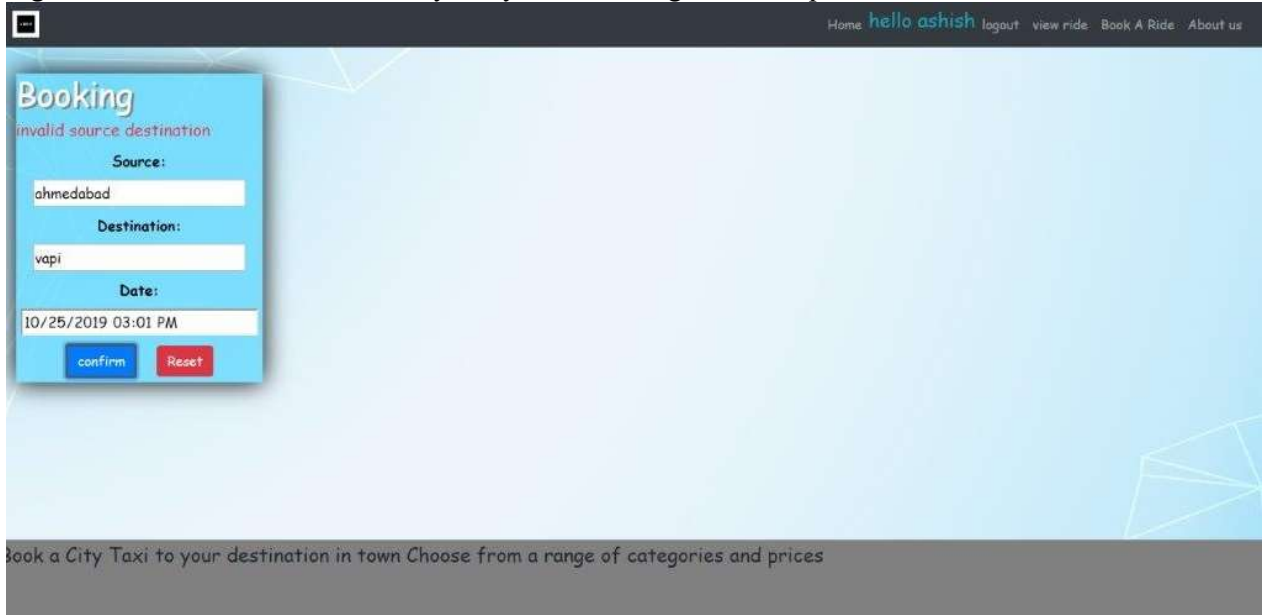
- Name**: A text input field with the message 'Name Required' in red text below it.
- Username**: A text input field with the message 'Username Required' in red text below it.
- Password**: A text input field with the message 'Password Required' in red text below it.
- Confirm Password**: A text input field with the message 'Password Required' in red text below it.
- Contact No**: A text input field with the message 'Contactno Required' in red text below it.
- Email**: A text input field with the message 'Email Required' in red text below it.

At the bottom of the form, there are two buttons: a blue 'submit' button and a red 'Reset' button.



Username should not be redundant else error will be produced. Similarly if wrong credentials are added for login username and password error will be produced.

Figure: if database doesn't have any entry then booking invalid is produced.



7 Screenshots

Figure7.1: WEB APPLICATION HOME PAGE...

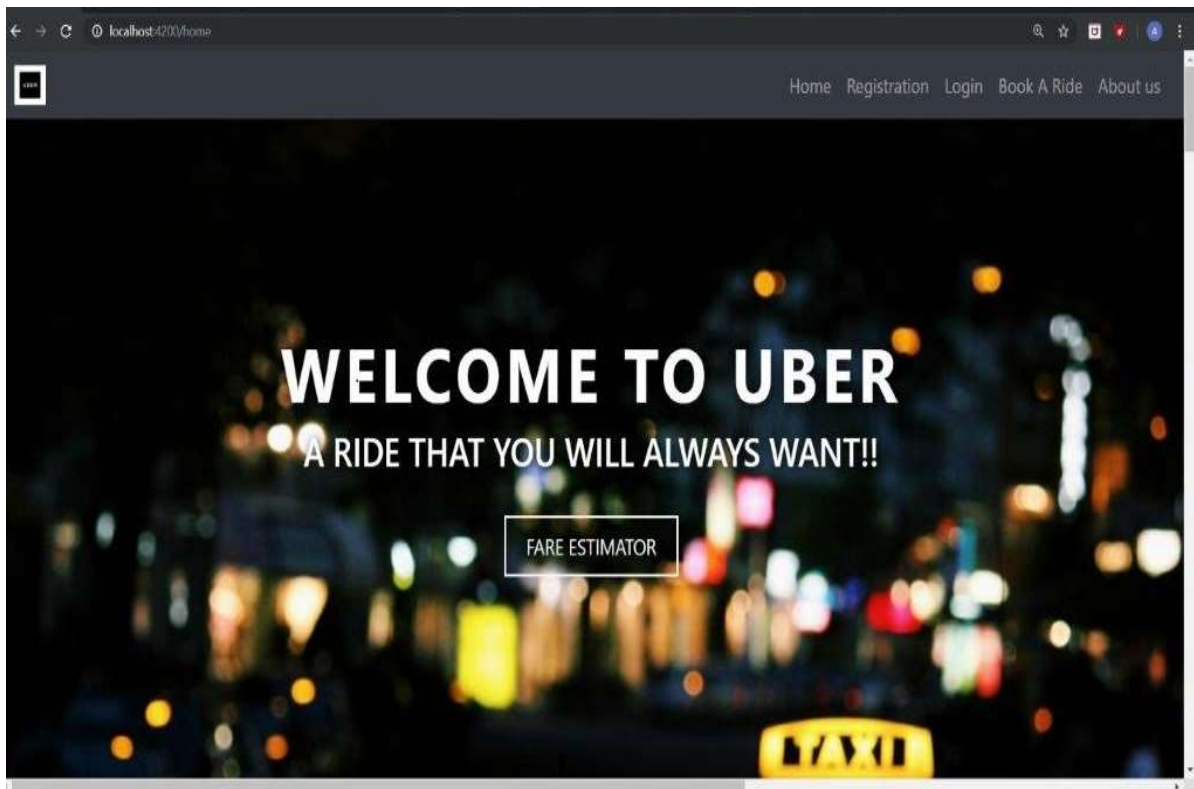


Figure7.2: User Registration

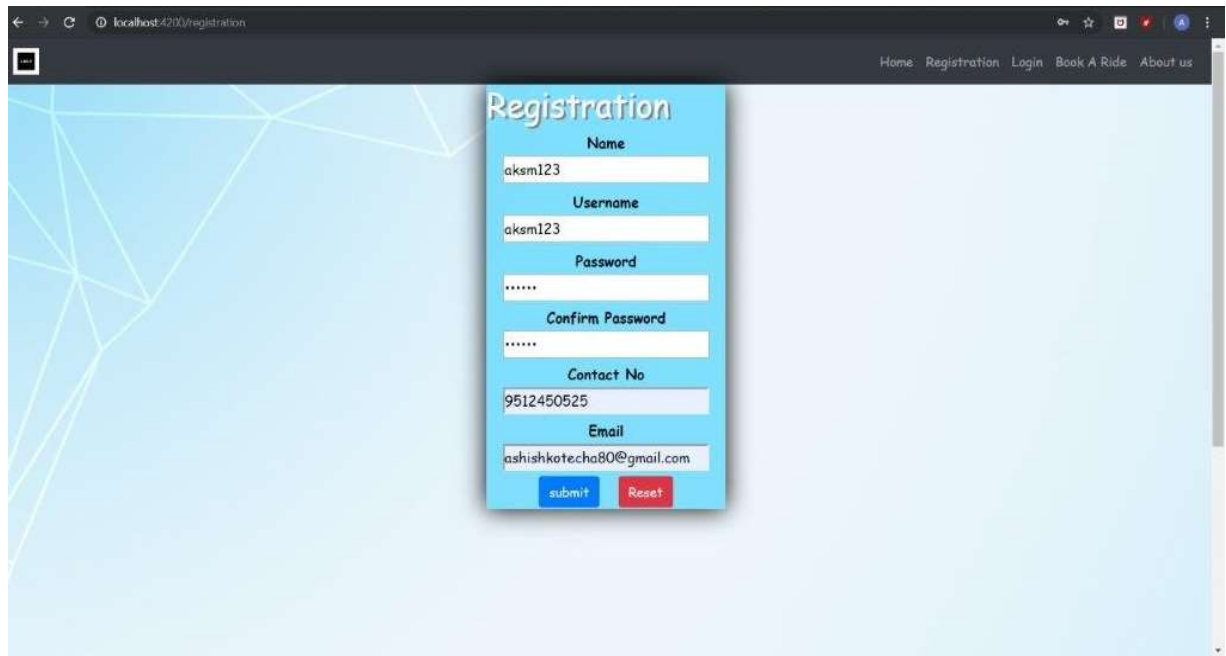


Figure7.3: Login successfully user correct credentials.

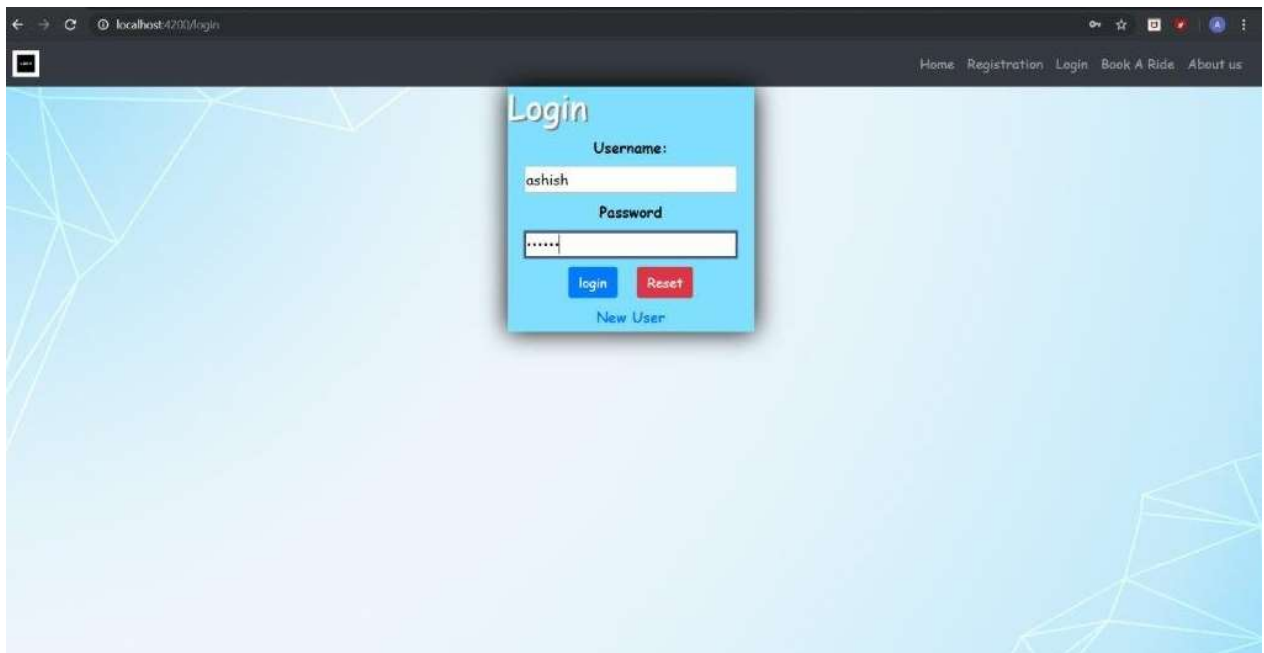


Figure 7.4 User logged in and sees user home page

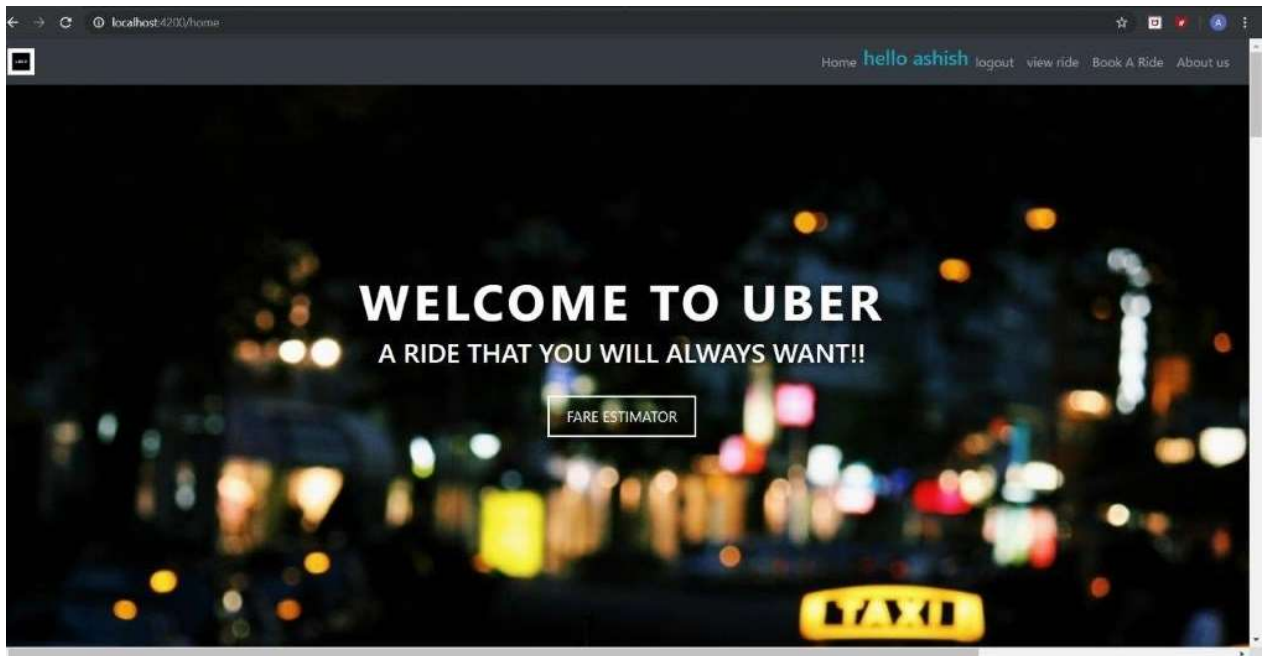


Figure 7.3: user gives source and destination to book a ride

Booking

Source:
ahmedabad

Destination:
baroda

Date:
10/25/2019 01:01 PM

confirm Reset

Book a City Taxi to your destination in town Choose from a range of categories and prices

Figure7.4:Confirm Booking to view Fare Price

Source:ahmedabad
Destination:baroda
Your Fare Is 1000
Journey Date 2019-10-25T13:01

confirm booking
View In Map

Figure 7.5 view in map

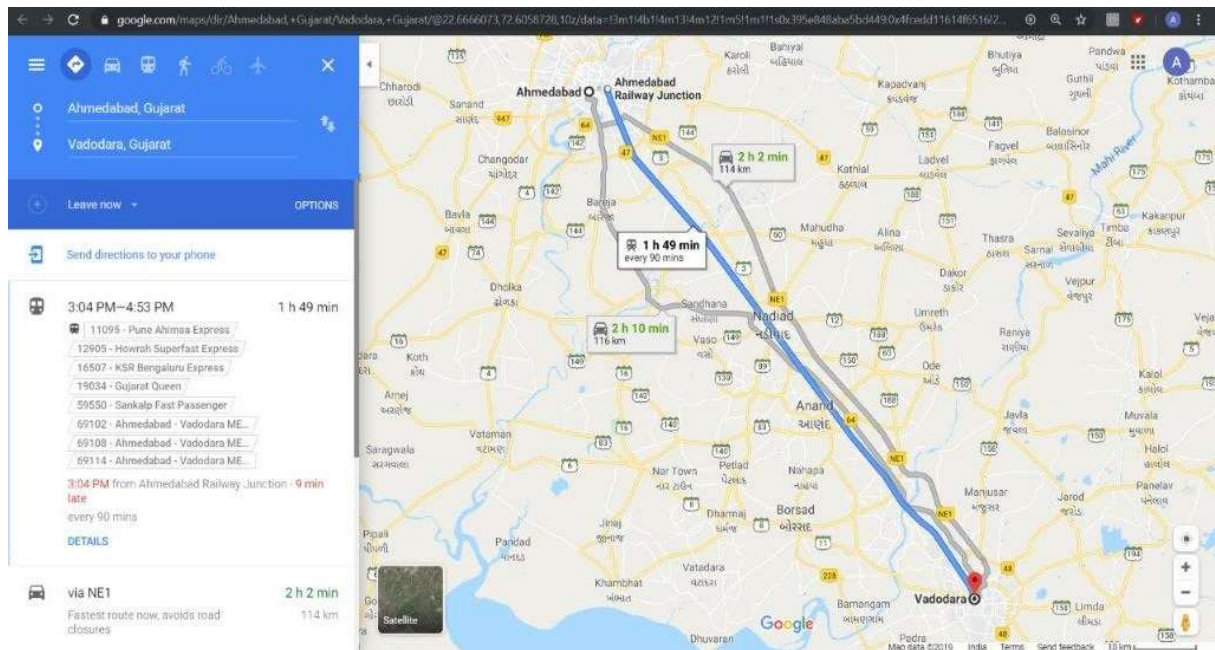


Figure 7.6

ADMIN SIDE

Admin can create update delete and view rides as well as add routes.



localhost:4200/user/create

Name
ashish123

Username
ashish123

password
123456

contact
7043237722

email
sam@gmail.com

Create User

Admin adds the user if any complaints comes.

localhost:4200/users

Name	Username	Password	email	Actions
sam1	sam1	12345	samyakjm@gmail.com	Edit Delete

Admin can update and delete the users.

localhost:4200/admin/addroute

Source
nadiad

Destination
baroda

distance
70

Create Route

Admin can Add the routes in the application.

Limitations and Future Enhancements

Limitations

- The Web application only allows booking of pre-defined Source Destination which are added by admin.
- The current position of the user not shown in the system without use of Api.

Functionalities not Implemented

- In this system online payment functionality is not implemented.
- Service provided by system is of only defined cities. It means system is restricted to fixed cities.

Future Extension

- Using online payment methods.
- Expanding our reach by providing service to more precise locations and including more powerful tools to compete with currently Existing system like Ola and UBER with using Google maps API.

Conclusion

The functionality implemented in the system was done after understanding all the system modules according to the requirements.

Functionalities that are successfully implemented in the system are:

- User Registration containing all the necessary validations on fields.
- Mailing System
- Login
- User Authentication
- Session Management
- Cab Booking
- Show Fare
- View Total Rides
- Admin Add, update, Delete user
- Admin Add Routes

After Implementation the whole application was tested with different validations and then have tried the best to remove the loopholes and tried to complete most of the functionalities.

Bibliography

□ Websites:

1. <https://www.w3schools.com> –For Html, CSS, Bootstrap, XML, JavaScript.
2. <https://www.tutorialspoint.com> –For NodeJS, Typescript, Angular.
3. <https://angular.io> –For Understanding Angular.
4. <https://stackoverflow.com> –For solving problems.

□ Useful Links:

1. <https://nodejs.org/en/> -For download and setup of NodeJS.
2. <https://fonts.google.com> –For getting fonts families.
3. <https://www.youtube.com/watch?v=1tRLveSyNz8&t=4413s> – For understanding.
4. <https://www.youtube.com/watch?v=MPuX11DS1vU&t=235s>
5. <https://www.awfis.com>