A WEB APPLICATION

National Park Transportation System

By

Sagar Patel	**(Roll No.: CE-94)**	**(ID No.: 14CEUOS076)**
Manan Shah	**(Roll No.: CE-111)**	**(ID No.: 14CEUOS082)**
Smit Purohit	**(Roll No.: CE-120)**	**(ID No.: 14CEUON037)**

B.Tech Sem - VI

A project submitted

In

System Design Practice
(SDP)

Internal Guide
Prof. Jigar M. Pandya
Assistant Professor
Dept. of Comp. Engg.

Dr. C. K. Bhensdadia
Head,
Dept. of Comp. Engg.

Department of Computer Engineering

Faculty of Technology
Dharmsinh Desai University
April 2017

CERTIFICATE

This is to certify that the project work titled

National Park Transportation System

is the bonafide work of

Sagar Patel	**(Roll No.: CE-94)**	**(ID No.: 14CEUOS076)**
Manan Shah	**(Roll No.: CE-111)**	**(ID No.: 14CEUOS082)**
Smit Purohit	**(Roll No.: CE-120)**	**(ID No.: 14CEUON037)**

Carried out in System Design Practice(SDP) in Computer Engineering at Dharmsinh Desai University in the academic session
December 2016 to May 2017

(Prof. Jigar M. Pandya)
Asst. Prof
Dept. of Comp. Engg.

(Dr. C. K. Bhensdadia)
Head
Dept. of Comp. Engg.

Department of Computer Engineering

Faculty of Technology
Dharmsinh Desai University
April 2017

Acknowledgements

The developers of *National Park Transportation System* are thankful to Prof. Jigar M. Pandya, Asst. Prof., Dept. of Comp. Engg. and Prof. Ankit P. Vaishnav, Asst. Prof., Dept. of Comp. Engg. for their guidance, expertise, counseling and recommendations for the project.

Table of Contents

Chapter 1

Introduction

Project Details:

In the modern world digital technology is an asset and it is the future in itself. Traveling is one fun ride and National Park Transportation system fulfills the requirement of booking train seats in a national park. The reason behind development of this system was facilitating customers with the ability to book their ticket's in advance from anywhere around. The system is basically concerned with the reservation and cancellation of railways tickets of the passenger. Visitor's can use their PayTM to book their tickets.

Technology used :

Language: **Java Enterprise Edition**

Development Tool: Netbeans IDE

Netbeans IDE is an open source Integrated Development Enviornment for Web application development. It uses Java programming language for application development.

Database System: MySql

Database management tool: PhpMyAdmin

Servlet Container: Tomcat

WebServer: Apache

Diagram Tool: **Umlet**

Umlet is used for creating various purpose UML diagrams

Chapter 2

About the System

National Park Transportation system fulfills the requirement of booking seats for transportation in national park. The system was built such that it can successfully handle reservations for any small scale transportation system . The logic behind this system can be applied for making a reservation system for transportation in amusement parks, special trains like trams, etc. The customer provides his trip details and the details submitted by him will be saved in the databases of the system. This system consists of an administrator who keeps record of all customers. Also the admin can add new trains, routes, stations as per the requirements. The database maintains information for trains, routes, stations, user details, customer details and such others.

For availing all the services provided by National Park Transportation system, visitors (users) are required to first register themselves by providing valid details to the system. The system stores secure information like password in encrypted form. The administrator logs the user in the system and permits users to maintain and manage their profile. After booking a particular ticket the visitor will receive an email for the same as a notification. This email will contain all the details including PNR number, journey date, ticket price, source and destination station, the opted class of bogey and such that. Visitor can thus rest assured about his journey and show this email to the ticket checker while traveling.

Chapter 3

Software Requirement Specifications

Introduction

- ### Purpose

The purpose of this document is to present detailed description of requirements for “National Park Transportation System”. It will explain purpose and features of system and constraints under which it must operate. This document will be used by software developer, manager and end user. Using this document any of software developer, manager and end user would be easily able to delineate the functionalities and features of this software.

- ### Document Conventions

IEEE format for software requirement specification.

- ### Intended Audience and Reading Suggestions

  - This document is to be read by development team, the project managers, testers and documentation writer. Software engineers will use SRS to fully understand the expectations of Online National Park Transportation System to construct appropriate software.

- ### Product Scope

  - The software product to be produced is Online National Park Transportation System, which will automate National Park Train Reservations. Transportaion System allows user to book tickets online. Hence user will be able to book tickets online without the hassle of going to the location themselves. Two end users for this system are Administrator and Clients.

  - System also allows Administrator to add new stations, routes and trains. Automation will make manual system interactive and speedy. Automation will make administration more effective.

- ### References

IEEE recommended Software Requirement Specification.

Overall Description

- ### Product Perspective

Online National Park Transportation System is a new self-contained software product. It is an independent system. The software to be produced will not need any additional hardware or other component for it to be fully functional. However, it will use third party payment gateways for making secure payments.

- ### Product Functions

Major functions of this product are:

- Online Registration form for new Users
- Facility to add new trains, stations and routes
- Displaying list of trains between different stations
- Displaying seat availability for a train on a particular day
- Automatic generation of receipt upon secure transaction

- ### User Classes and Characteristics

1. Administrator:
Administrator can add new trains, stations and routes.
2. Clients:
Clients can view train list and seat availability for different trains between different stations.

- ### Operating Environment

This product is developed mainly using technologies like J2EE, oracle, MySQL etc. Any version of windows, linux or mac can be used to operate Online National Park Transportation System
<u>Software Requirements</u>
Operating System: Windows XP or higher, any flavor of linux, mac os.
Software:Web Browser

<u>Hardware Requirements</u>
Processor: Pentium or higher
Ram: 512 MB (minimum)

- ### Design and Implementation Constraints

- System will not work if appropriate network or hardware is not supplied.

- There are no other constraint at this point of time.

- **User Documentation**

  A well-documented user manual will be provided after the completion of the project. To delineate all the features of administration and other user account.

- **Assumptions and Dependencies**

  - Tickets will be booked after successful transaction between third party payment gateway and client.
  - Successful booking will be done only after proper authentication of Client.

External Interface Requirements

- **User Interfaces**

The user interface for the system will be a web page on the Internet. The user interface will be limited to the types of controls that can be generated using HTML, JavaScript, and Cascading Style Sheets. Application can be accessed through any browser interface. It is well suited for Microsoft Edge and Google Chrome.

This software starts with a login form for all end users.
//All the facilities for end users can be accessed through navigation bar after proper login.
There are also many pop-up menus placed for easy interaction.

- **Hardware Interfaces**

The Hardware Interfaces of the system are handled by the Windows Server 2012 Operating System. No hardware dependent code will be written by the team of Online National Park Transportation System.

- **Software Interfaces**

  - **Operating System**
    - The software is being designed to run on Windows Server 2012. Windows Server 2012 includes the latest version of Internet Information Services, version 8.5
  - **Web Server**
    - The software is being designed to run on Internet Information Server version 8.5.
  - **Database**
    - The software will access the SQL Server 2014 Enterprise Edition database for the following features.
  - **Libraries**
    - The software will be created using the Java Enterprise Edition version 4.5

- **Communications Interfaces**

    - **Web Interface:-**The application will be accessed over the Internet. All features will accessible through the web site. Communication would be through TCP/IP protocol.

System Requirements

- R1: System provides facility for the authentication of users

  Input : User-info

  Processing : Authentication

  Output : Authentication flag

    - R 1.1: System allows users to register.

      Input : User-info

      Output: User-credentials

    - R1.2: System provides facility to login for different users

      Input: User-credentials

      Process: Validation

      Output: Authentication flag , info

- R2: System allows admin to add new stations and routes between stations.
  Input: Station Details

  Process: Update-info

  Output: Update-info-flag

- R3: System allows Admin to add new trains.

  Input: Station-details, train-details

Output: train number

- R3.1 System provides admin facility to enter train details.

  Input: station-details, train details

  Process: fetch existing routes

  Output: list of routes

- R3.2 System allows admin to select a route

  Input: list of routes

  Process: fetch list of stations

  Output: list of stations

- R3.3 System allows admin to select stations on which train will stop

  Input: list of stations

  Output: train number

- R4 System allows admin to update schedule of a train.

  Input: train number, schedule-info

  Output: Update-schedule-flag

- R5: System allows Customer to book train tickets securely with the help of Paytm .

  Input: Station-details, train-details

  Process: secure transaction

  Output: train ticket

  - R5.1 System allows customer to enter station details.

    Input: station-details

    Process: fetch available trains

    Output: list of trains

  - R5.2 System provides customer facility to select train from train list

Input: list of trains

Output: train-details

- R5.3 System allows customer to perform secure transactions

  Input: ticket-details

  Process: secure transaction

  Output: train ticket

Other Nonfunctional Requirements

- ### Performance Requirements

  - System should be able to handle multiple users at a time.
  - Database must be large enough to store all the data.

- ### Safety Requirements

Database may get crashed at any time due to system failure or any other reasons. So, the backup of the database must be taken regularly.

- ### Security Requirements

  - To protect the software from accidental or malicious access, use, modification, destruction or disclosure some factors are required. They are:
    - Later versions of the software will incorporate encryption techniques in the user/license authentication process.
    - Communication needs to be restricted when the application is validating the user of license.

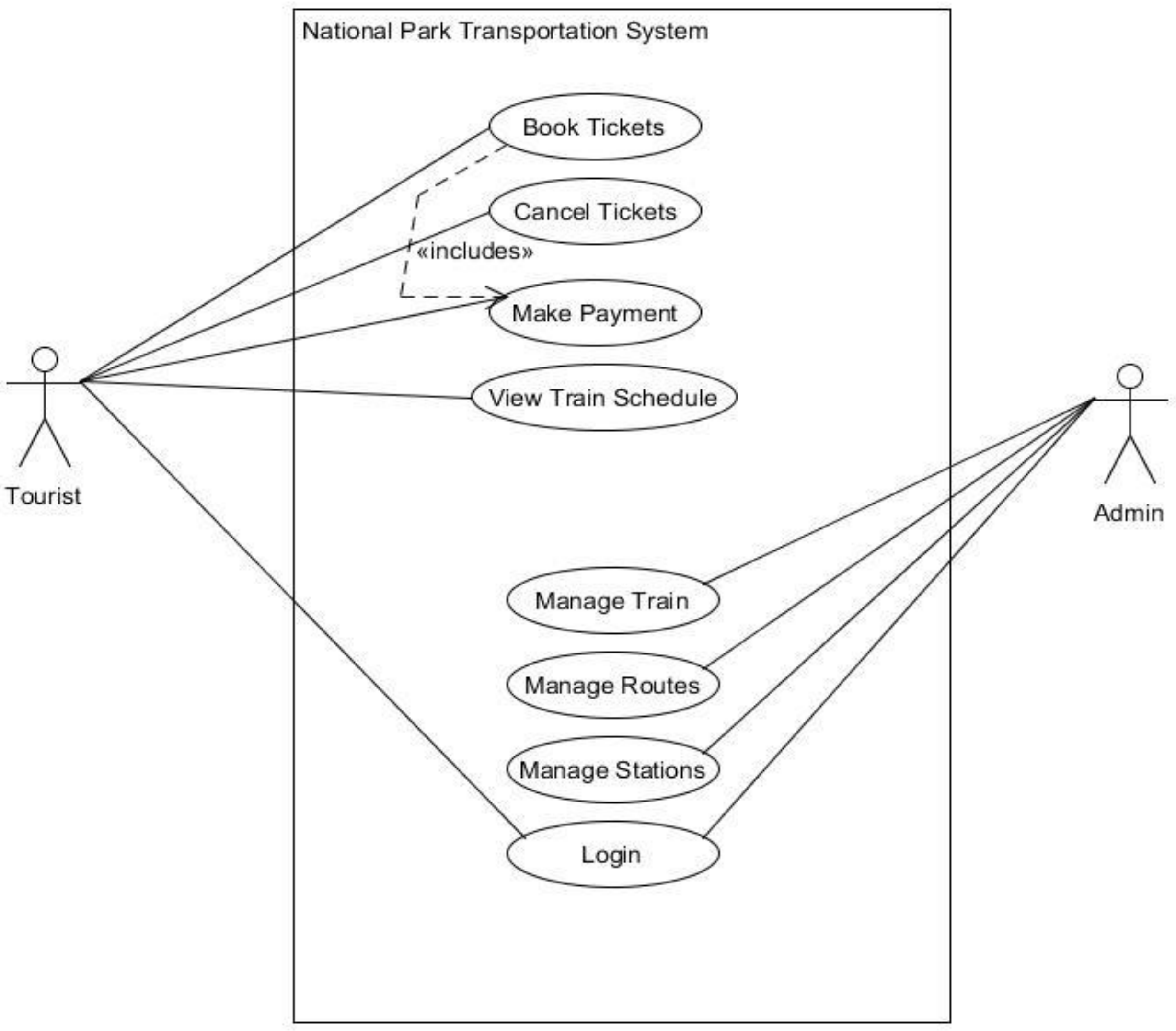
5.4 Software Quality Attributes

- **Reliability**: Data, as entered, must be correctly stored in the database. In addition, the database should use transactions so that partial entries are not stored in the database. Because the user of the system may be inexperienced, each feature must work correctly 99% of the time. The logic required is not complex enough to allow for a lower expectation.
- **Maintainability**: The system will likely be developed by a same team at the same time after 3 months. And so the code and design need to be documented well enough.

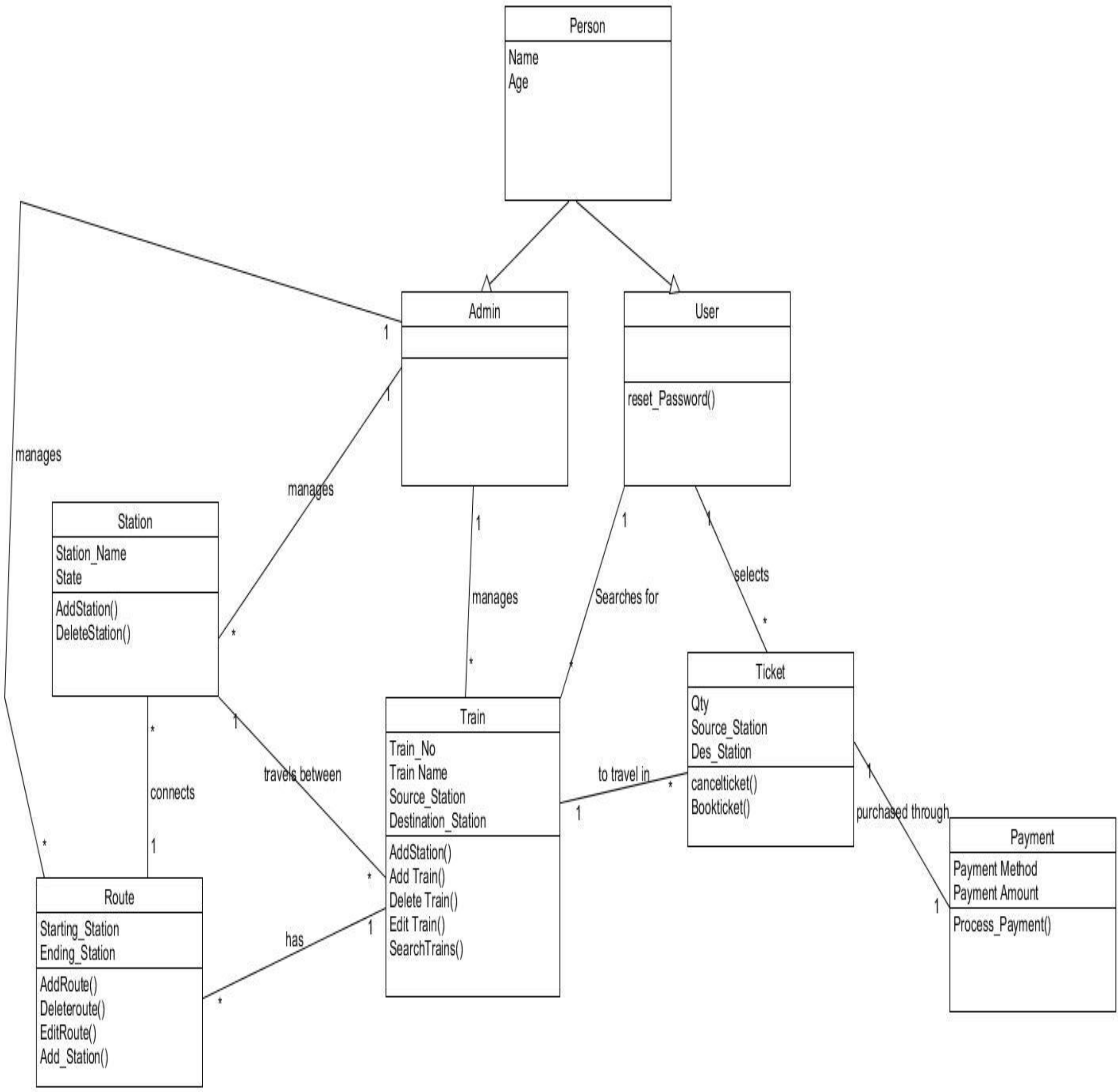
Chapter 4

Design

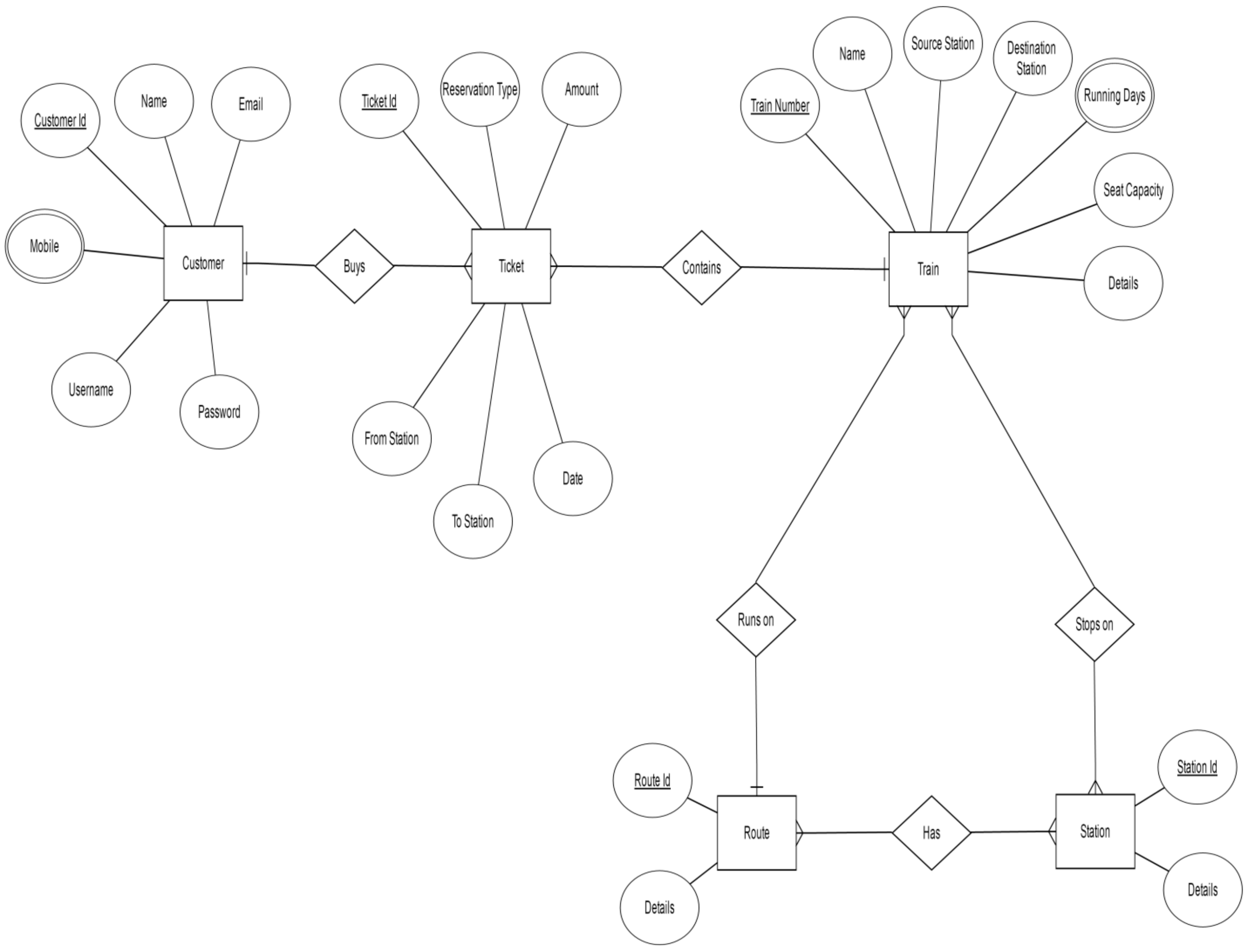
Use Case Diagram for National Park Transportation System



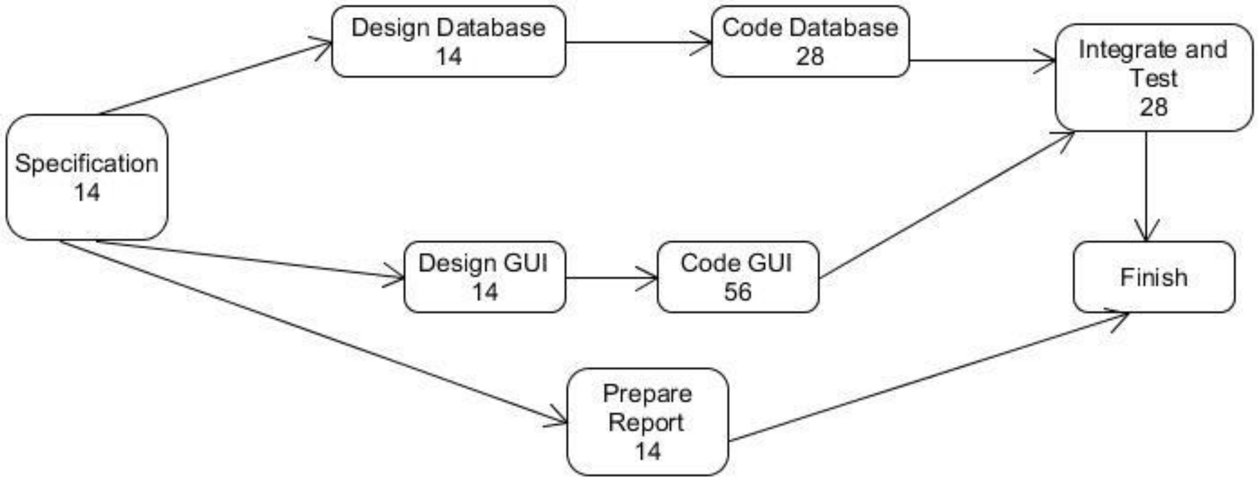
Class Diagram for National Park Transportation System



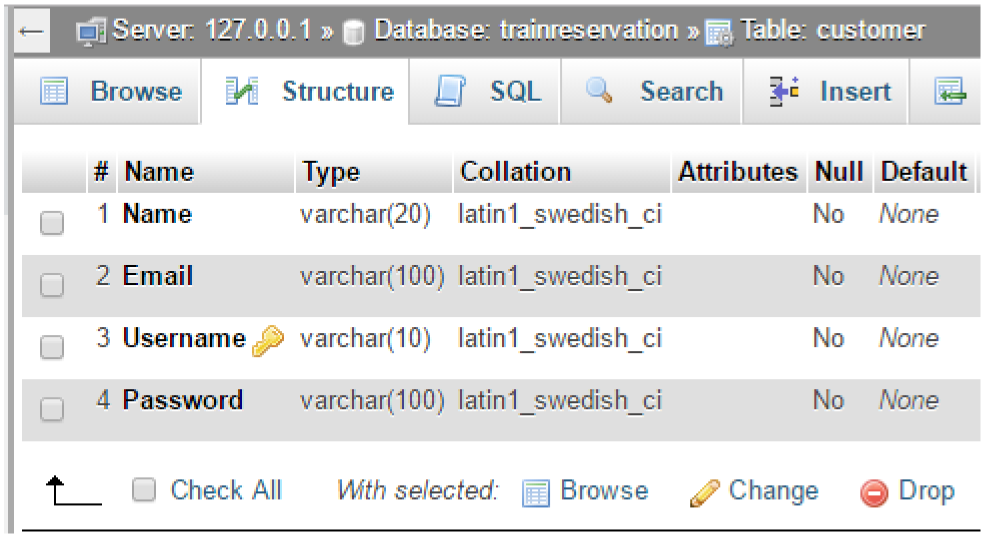
ER Diagram for National Park Transportation System



ACTIVITY NETWORK DIAGRAM:



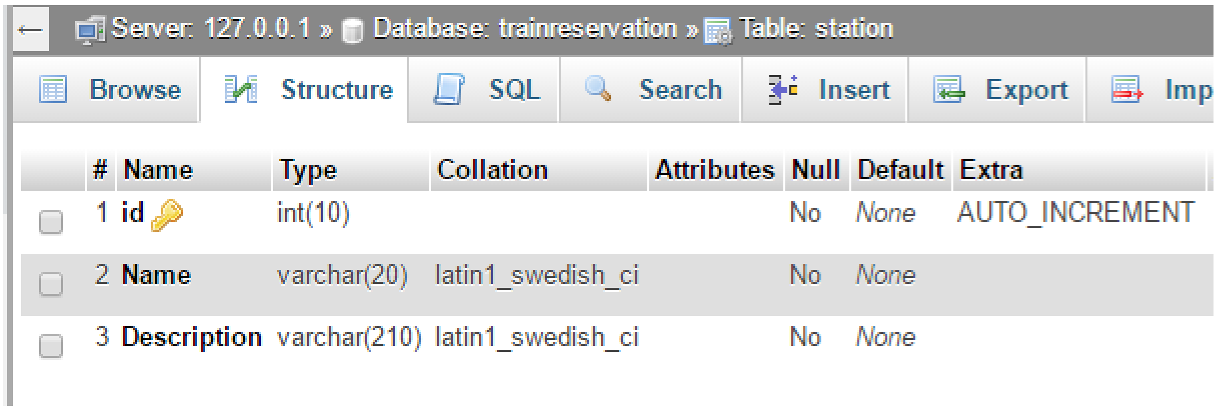
Data Dictionary for National Park Transportation System



← Server: 127.0.0.1 » Database: trainreservation » Table: customer
Browse Structure SQL Search Insert

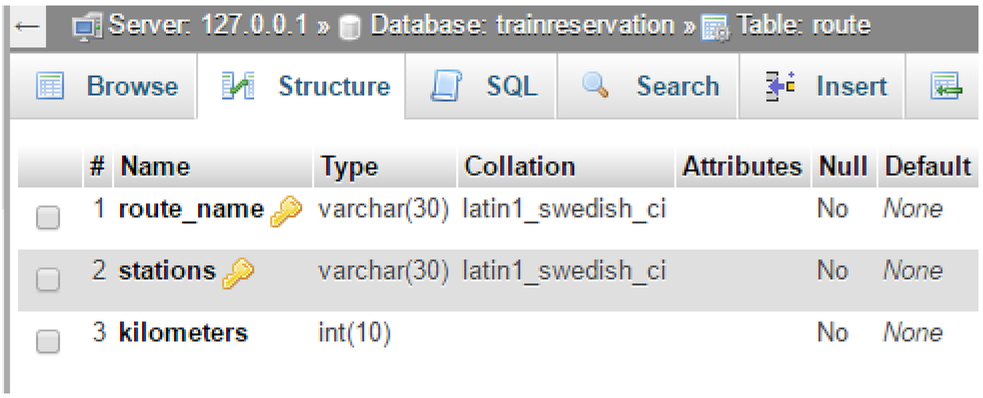
#	Name	Type	Collation	Attributes	Null	Default
1	**Name**	varchar(20)	latin1_swedish_ci		No	*None*
2	**Email**	varchar(100)	latin1_swedish_ci		No	*None*
3	**Username**	varchar(10)	latin1_swedish_ci		No	*None*
4	**Password**	varchar(100)	latin1_swedish_ci		No	*None*

Check All *With selected:* Browse Change Drop



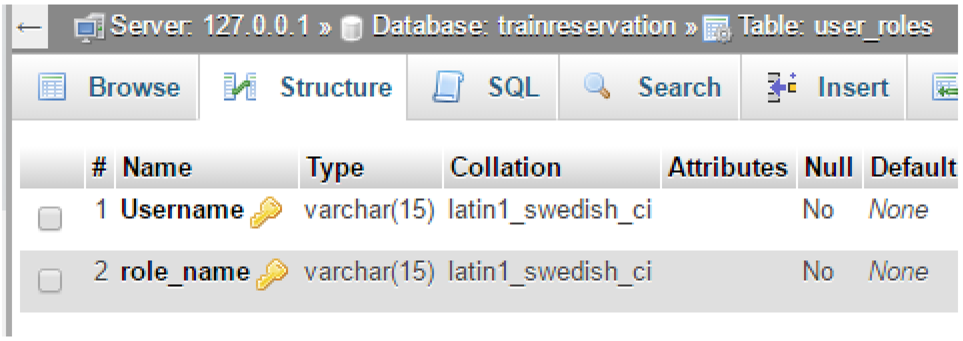
← Server: 127.0.0.1 » Database: trainreservation » Table: station
Browse Structure SQL Search Insert Export Imp

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**id**	int(10)			No	*None*	AUTO_INCREMENT
2	**Name**	varchar(20)	latin1_swedish_ci		No	*None*	
3	**Description**	varchar(210)	latin1_swedish_ci		No	*None*	



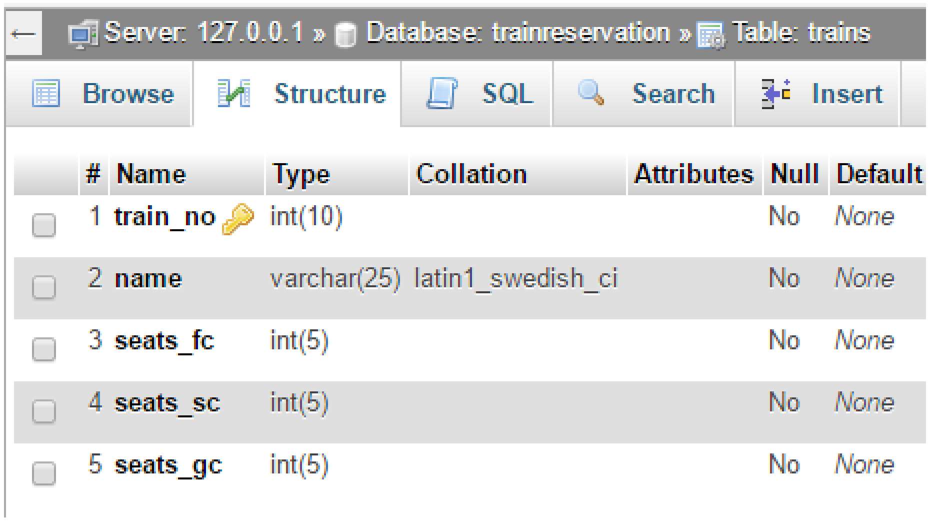
Server: 127.0.0.1 » Database: trainreservation » Table: route

	#	Name	Type	Collation	Attributes	Null	Default
☐	1	**route_name**	varchar(30)	latin1_swedish_ci		No	*None*
☐	2	**stations**	varchar(30)	latin1_swedish_ci		No	*None*
☐	3	**kilometers**	int(10)			No	*None*



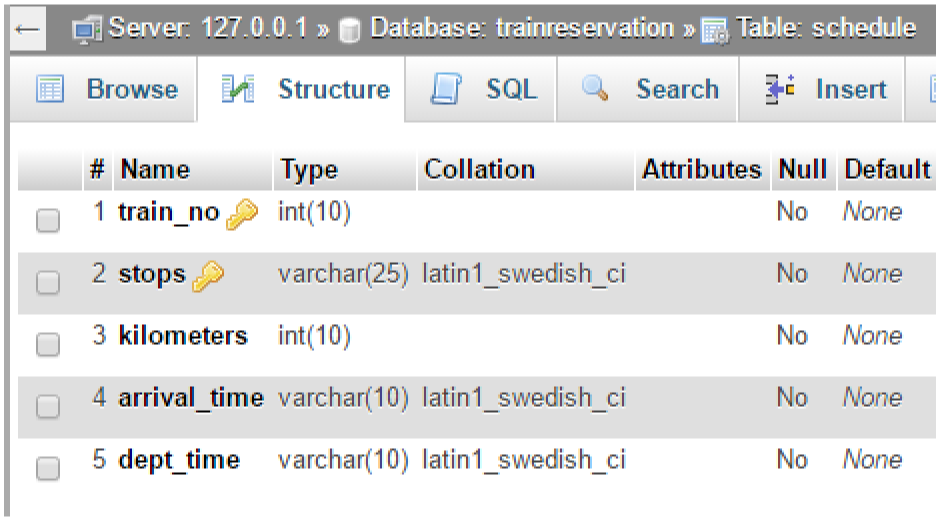
Server: 127.0.0.1 » Database: trainreservation » Table: user_roles

	#	Name	Type	Collation	Attributes	Null	Default
☐	1	**Username**	varchar(15)	latin1_swedish_ci		No	*None*
☐	2	**role_name**	varchar(15)	latin1_swedish_ci		No	*None*

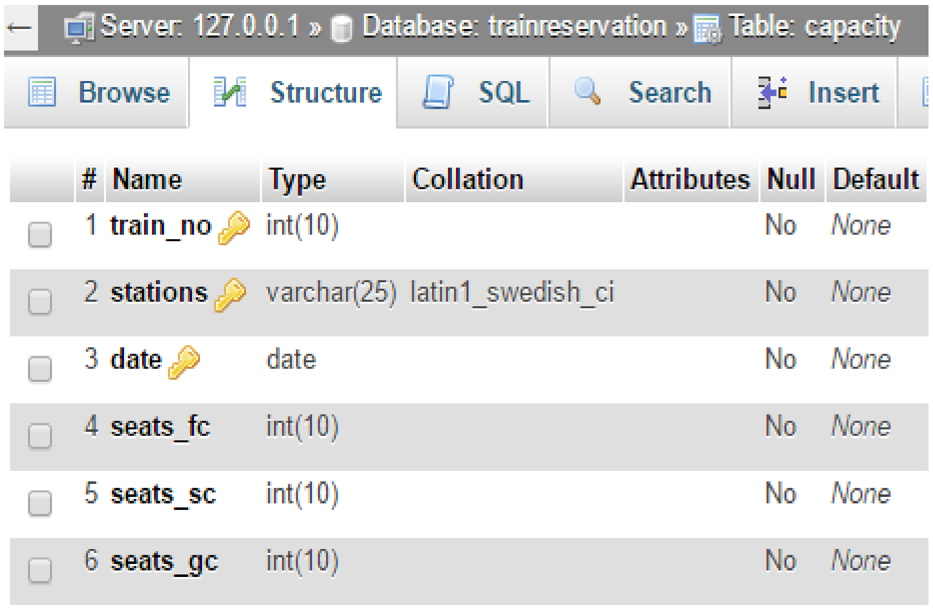


Server: 127.0.0.1 » Database: trainreservation » Table: trains

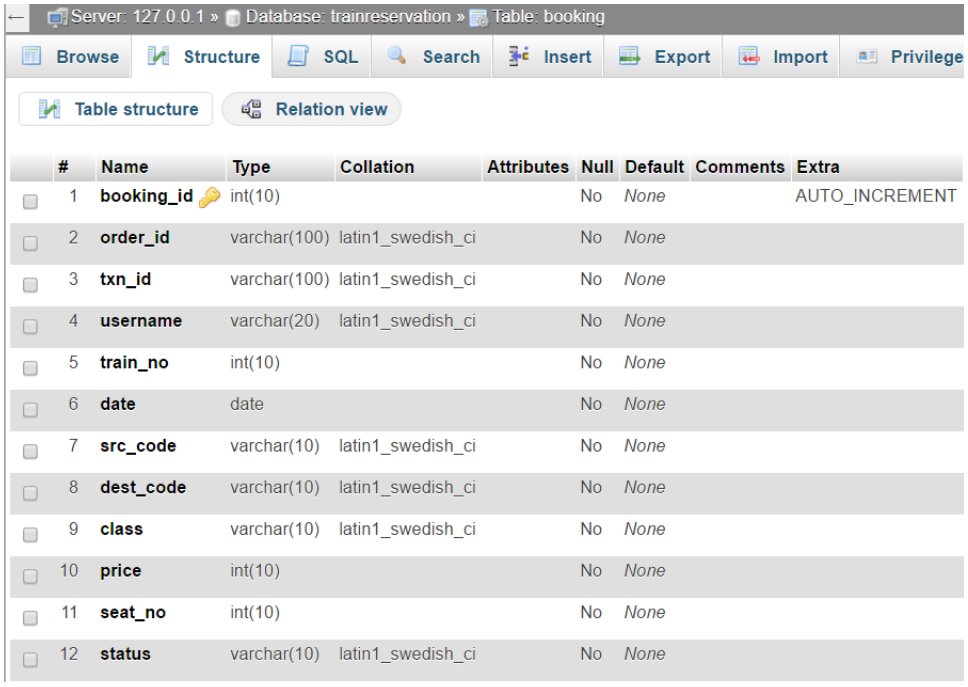
	#	Name	Type	Collation	Attributes	Null	Default
☐	1	**train_no**	int(10)			No	*None*
☐	2	**name**	varchar(25)	latin1_swedish_ci		No	*None*
☐	3	**seats_fc**	int(5)			No	*None*
☐	4	**seats_sc**	int(5)			No	*None*
☐	5	**seats_gc**	int(5)			No	*None*


←
Server: 127.0.0.1 » Database: trainreservation » Table: schedule
Browse
Structure
SQL
Search
Insert

	#	Name	Type	Collation	Attributes	Null	Default
☐	1	**train_no**	int(10)			No	*None*
☐	2	**stops**	varchar(25)	latin1_swedish_ci		No	*None*
☐	3	**kilometers**	int(10)			No	*None*
☐	4	**arrival_time**	varchar(10)	latin1_swedish_ci		No	*None*
☐	5	**dept_time**	varchar(10)	latin1_swedish_ci		No	*None*


←
Server: 127.0.0.1 » Database: trainreservation » Table: capacity
Browse
Structure
SQL
Search
Insert

	#	Name	Type	Collation	Attributes	Null	Default
☐	1	**train_no**	int(10)			No	*None*
☐	2	**stations**	varchar(25)	latin1_swedish_ci		No	*None*
☐	3	**date**	date			No	*None*
☐	4	**seats_fc**	int(10)			No	*None*
☐	5	**seats_sc**	int(10)			No	*None*
☐	6	**seats_gc**	int(10)			No	*None*



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	booking_id	int(10)			No	*None*		AUTO_INCREMENT
2	order_id	varchar(100)	latin1_swedish_ci		No	*None*		
3	txn_id	varchar(100)	latin1_swedish_ci		No	*None*		
4	username	varchar(20)	latin1_swedish_ci		No	*None*		
5	train_no	int(10)			No	*None*		
6	date	date			No	*None*		
7	src_code	varchar(10)	latin1_swedish_ci		No	*None*		
8	dest_code	varchar(10)	latin1_swedish_ci		No	*None*		
9	class	varchar(10)	latin1_swedish_ci		No	*None*		
10	price	int(10)			No	*None*		
11	seat_no	int(10)			No	*None*		
12	status	varchar(10)	latin1_swedish_ci		No	*None*		

Data Definition Language

1. CUSTOMER

```
CREATE TABLE IF NOT EXISTS `customer` (
  `Name` varchar(20) NOT NULL,
  `Email` varchar(100) NOT NULL,
  `Username` varchar(10) NOT NULL,
  `Password` varchar(100) NOT NULL
)
```

2. USER ROLES

```
CREATE TABLE IF NOT EXISTS `user_roles` (
  `Username` varchar(15) NOT NULL,
  `role_name` varchar(15) NOT NULL
)
```

3. STATION

```
CREATE TABLE IF NOT EXISTS `station` (
  `id` int(10) NOT NULL,
  `Name` varchar(20) NOT NULL,
  `Description` varchar(210) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=latin1;
```

4. ROUTE

```
CREATE TABLE IF NOT EXISTS `route` (
  `route_name` varchar(30) NOT NULL,
  `stations` varchar(30) NOT NULL,
  `kilometers` int(10) NOT NULL
)
```

5. TRAIN

```
CREATE TABLE IF NOT EXISTS `trains` (
  `train_no` int(10) NOT NULL,
  `name` varchar(25) NOT NULL,
  `seats_fc` int(5) NOT NULL,
  `seats_sc` int(5) NOT NULL,
  `seats_gc` int(5) NOT NULL
)
```

6. SCHEDULE

```
CREATE TABLE IF NOT EXISTS `schedule` (
  `train_no` int(10) NOT NULL,
  `stops` varchar(25) NOT NULL,
  `kilometers` int(10) NOT NULL,
  `arrival_time` varchar(10) NOT NULL,
  `dept_time` varchar(10) NOT NULL
)
```

7. RUNNING DAYS

```
CREATE TABLE IF NOT EXISTS `running_days` (
  `train_no` int(10) NOT NULL,
  `running_days` varchar(25) NOT NULL
)
```

8. CAPACITY

```
CREATE TABLE IF NOT EXISTS `capacity` (
  `train_no` int(10) NOT NULL,
  `stations` varchar(25) NOT NULL,
  `date` date NOT NULL,
  `seats_fc` int(10) NOT NULL,
  `seats_sc` int(10) NOT NULL,
  `seats_gc` int(10) NOT NULL
)
```

9. BOOKING

```
CREATE TABLE IF NOT EXISTS `booking` (
  `booking_id` int(10) NOT NULL,
  `username` varchar(20) NOT NULL,
  `train_no` int(10) NOT NULL,
  `date` date NOT NULL,
  `src_code` varchar(10) NOT NULL,
  `dest_code` varchar(10) NOT NULL,
  `class` varchar(10) NOT NULL,
  `price` int(10) NOT NULL,
  `seat_no` int(10) NOT NULL,
  `status` varchar(10) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=87 DEFAULT CHARSET=latin1;
```

Chapter 5

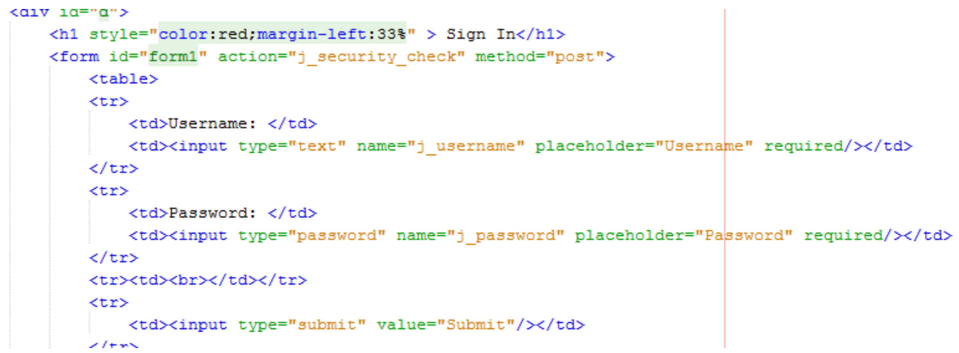
Implementation Details

The main modules present in the system are:

- User Management Module
- Admin module
- Customer module

User Management Module: Manages login and signup activity with proper authentication ,authorization and user role management.

Authentication: User's are authenticated using Form Based Authentication. With form based authentication password encryption using md5 algorithm is provided to enhance the security of passwords.

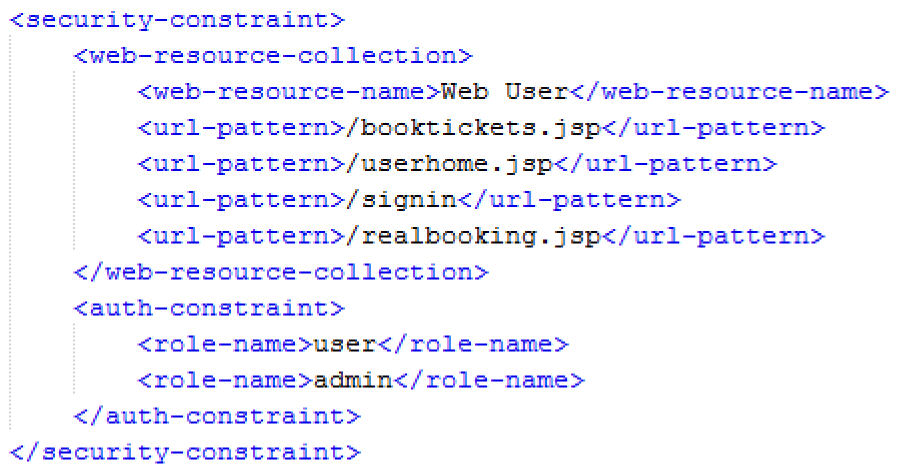


```
<div id="d">
    <h1 style="color:red;margin-left:33%" > Sign In</h1>
    <form id="form1" action="j_security_check" method="post">
        <table>
        <tr>
            <td>Username: </td>
            <td><input type="text" name="j_username" placeholder="Username" required/></td>
        </tr>
        <tr>
            <td>Password: </td>
            <td><input type="password" name="j_password" placeholder="Password" required/></td>
        </tr>
        <tr><td><br></td></tr>
        <tr>
            <td><input type="submit" value="Submit"/></td>
        </tr>
```

>The above snippet shows use of J_security_check for peforming form bsed authentication while logging in.

Authorization: Authorization is provided through security constraints in web.xml which states which pages can be accessed by which kind of user.

After sign in based on user role they are redirected to either admin home page or customer home page.



```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Web User</web-resource-name>
        <url-pattern>/booktickets.jsp</url-pattern>
        <url-pattern>/userhome.jsp</url-pattern>
        <url-pattern>/signin</url-pattern>
        <url-pattern>/realbooking.jsp</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>user</role-name>
        <role-name>admin</role-name>
    </auth-constraint>
</security-constraint>
```

>The above snippet shows use of security constraint for providing role based access to users for accessing system resources.

Sign-up:System provides facility to user's to register themselves online.

Note:Only admin can add new admin user.

Customer Module: Provide facility to customers to book and cancel tickets.

Book Ticket – Here the customer will select his source and destination stations along with the date of his journey. The results of available trains will be shown on the basis of the date selected. From that date given, system will find out which day it is and then further all the trains running on that day and running between both those stations will be shown. In the result generated number of vacant seats available for different classes of a train will be displayed along with its price. Customer selects the train of his choice and confirms his choice using the book ticket button.

Paytm Service – Customer will be redirected to Paytm website for the payment of his ticket. After the payment process he will again be redirected to National Park Transportation System.

>Paytm refund api url:https://pguat.paytm.com/oltp/HANDLER_INTERNAL/REFUND

>Paytm staging server url for transaction:https://pguat.paytm.com/oltp-web/processTransaction.

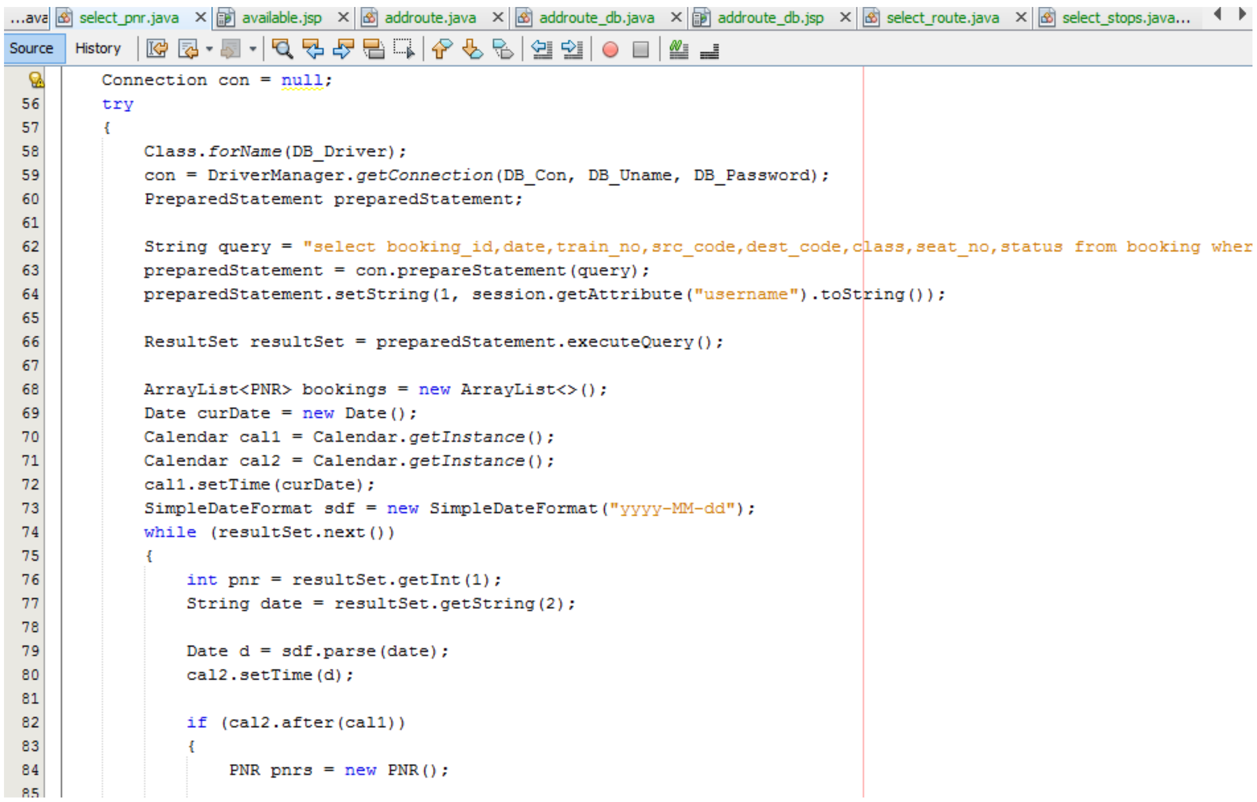
>Paytm merchant id:Nation20229563076978

>Paytm staging mobile number:7777777777

>Paytm staging otp:489871

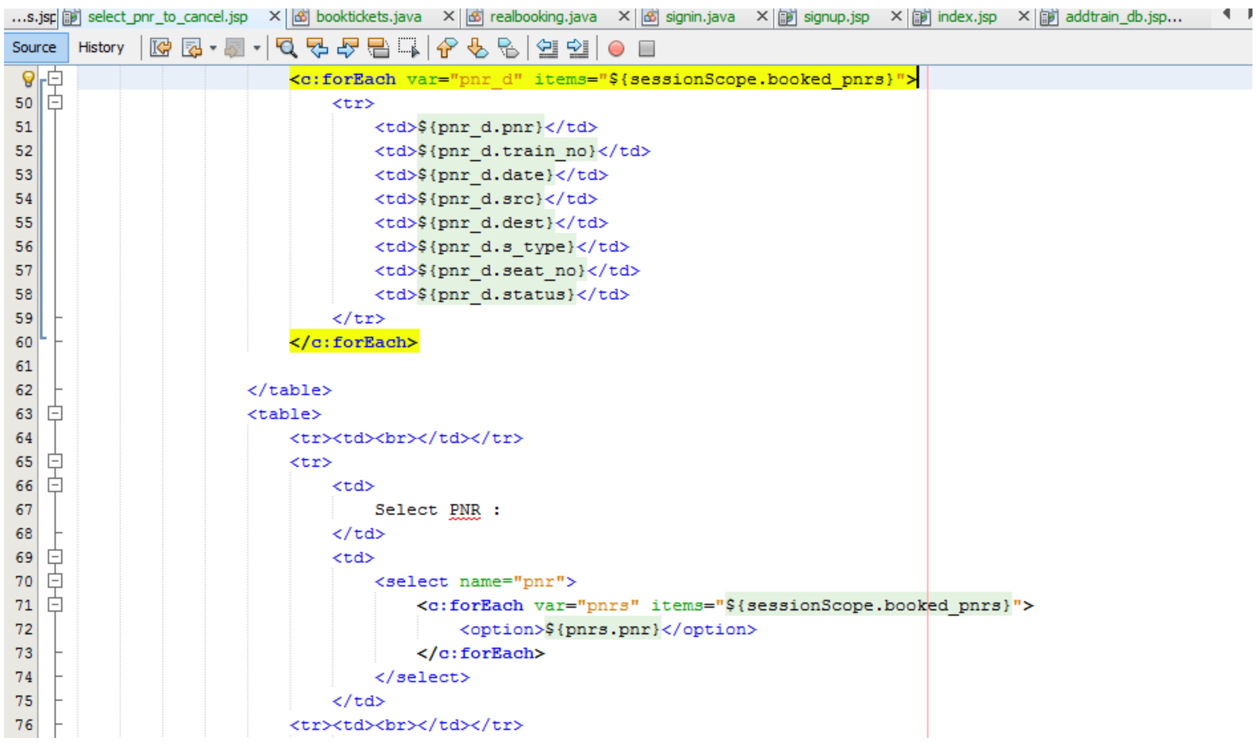
Email – If the payment is successful the user will be alloted seat in respective class as per his selection and then user will be notified with an email. Every details of the plan will be included in this email.

Cancel Ticket - A Passenger can cancel the ticket by Selecting a PNR of the ticket which he wants to cancel. Upon successful cancellation, the seat vacated is allocated to the first customer in waiting list of that train for that date.



```
    Connection con = null;
    try
    {
        Class.forName(DB_Driver);
        con = DriverManager.getConnection(DB_Con, DB_Uname, DB_Password);
        PreparedStatement preparedStatement;

        String query = "select booking_id,date,train_no,src_code,dest_code,class,seat_no,status from booking wher
        preparedStatement = con.prepareStatement(query);
        preparedStatement.setString(1, session.getAttribute("username").toString());

        ResultSet resultSet = preparedStatement.executeQuery();

        ArrayList<PNR> bookings = new ArrayList<>();
        Date curDate = new Date();
        Calendar cal1 = Calendar.getInstance();
        Calendar cal2 = Calendar.getInstance();
        cal1.setTime(curDate);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        while (resultSet.next())
        {
            int pnr = resultSet.getInt(1);
            String date = resultSet.getString(2);

            Date d = sdf.parse(date);
            cal2.setTime(d);

            if (cal2.after(cal1))
            {
                PNR pnrs = new PNR();
```

Upon request for cancellation all the booked tickets after current date is fetched from the database and passenger is asked to select the PNR which he wants to cancel.



```
    <c:forEach var="pnr_d" items="${sessionScope.booked_pnrs}">
        <tr>
            <td>${pnr_d.pnr}</td>
            <td>${pnr_d.train_no}</td>
            <td>${pnr_d.date}</td>
            <td>${pnr_d.src}</td>
            <td>${pnr_d.dest}</td>
            <td>${pnr_d.s_type}</td>
            <td>${pnr_d.seat_no}</td>
            <td>${pnr_d.status}</td>
        </tr>
    </c:forEach>

</table>
<table>
    <tr><td><br></td></tr>
    <tr>
        <td>
            Select PNR :
        </td>
        <td>
            <select name="pnr">
                <c:forEach var="pnrs" items="${sessionScope.booked_pnrs}">
                    <option>${pnrs.pnr}</option>
                </c:forEach>
            </select>
        </td>
    <tr><td><br></td></tr>
```

In the above code, All the fetched PNR details are displayed in a jsp and passenger is asked to select PNR to be cancelled from the dropdownlist.

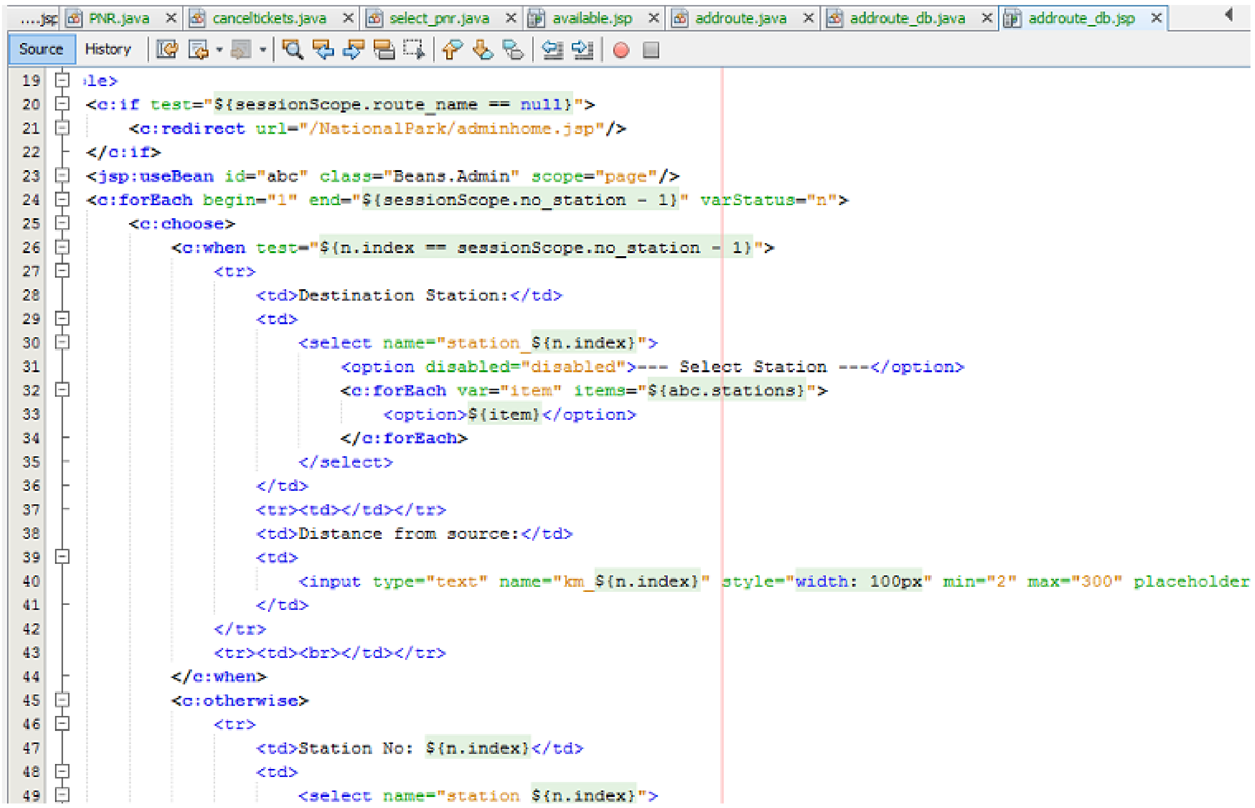
Upon confirmation, ticket is cancelled and seat allocated to the passenger is given to another passenger in waiting list and capacity of train is incremented accordingly.

Admin Module - Provide facility to admin to add new Stations, Routes and Trains.

Add New Station - Admin can add new station to the system by entering station name and brief details about the station. As a result new station will be added to the system.

Add New Route - Admin will enter the route name as well as the starting station for that route. Also he will select the total number of stations present in that route including the source and destination station. Upon entering all the details, he will be prompted to select stations present in that route and enter the distance of those stations from starting station.

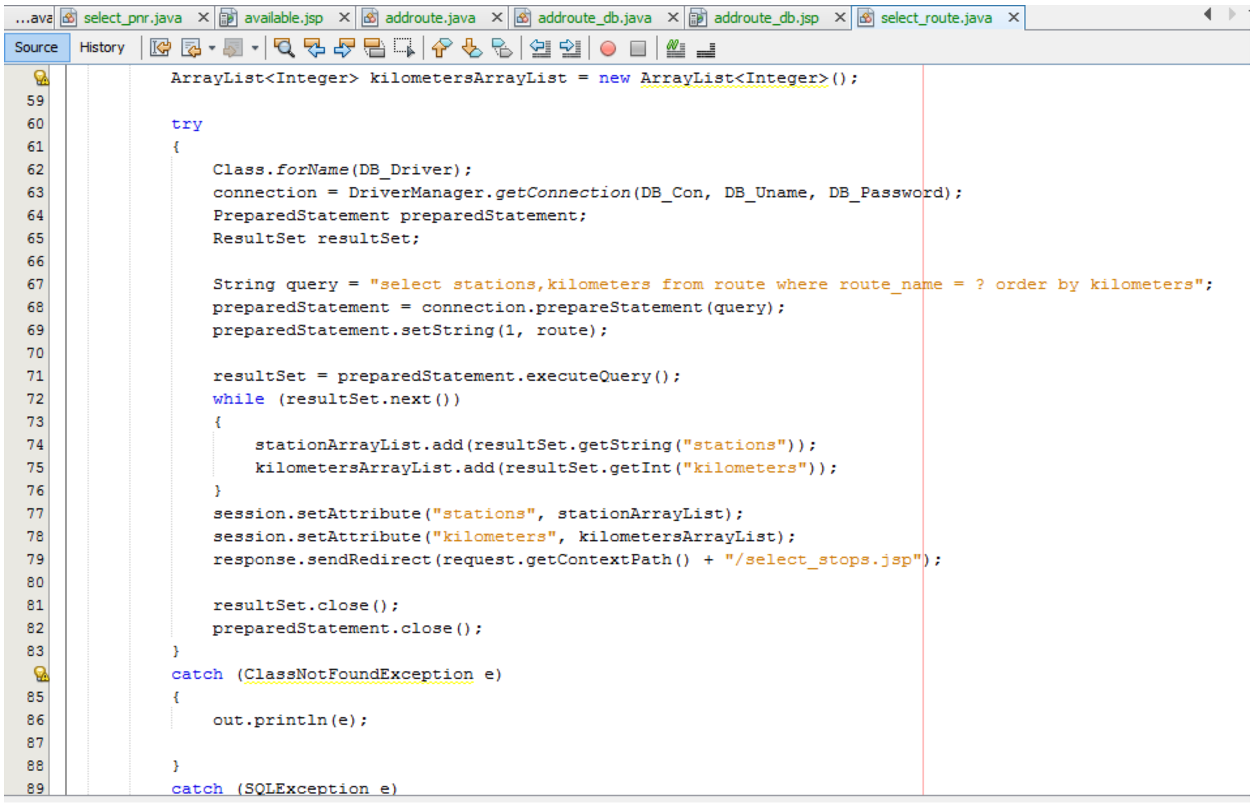
The Distance measured between stations is not relative but is measured from starting station, hence addition of new station in future will be easier.



```
le>
<c:if test="${sessionScope.route_name == null}">
    <c:redirect url="/NationalPark/adminhome.jsp"/>
</c:if>
<jsp:useBean id="abc" class="Beans.Admin" scope="page"/>
<c:forEach begin="1" end="${sessionScope.no_station - 1}" varStatus="n">
    <c:choose>
        <c:when test="${n.index == sessionScope.no_station - 1}">
            <tr>
                <td>Destination Station:</td>
                <td>
                    <select name="station_${n.index}">
                        <option disabled="disabled">--- Select Station ---</option>
                        <c:forEach var="item" items="${abc.stations}">
                            <option>${item}</option>
                        </c:forEach>
                    </select>
                </td>
                <tr><td></td></tr>
                <td>Distance from source:</td>
                <td>
                    <input type="text" name="km_${n.index}" style="width: 100px" min="2" max="300" placeholder
                </td>
            </tr>
            <tr><td><br></td></tr>
        </c:when>
        <c:otherwise>
            <tr>
                <td>Station No: ${n.index}</td>
                <td>
                    <select name="station ${n.index}">
```

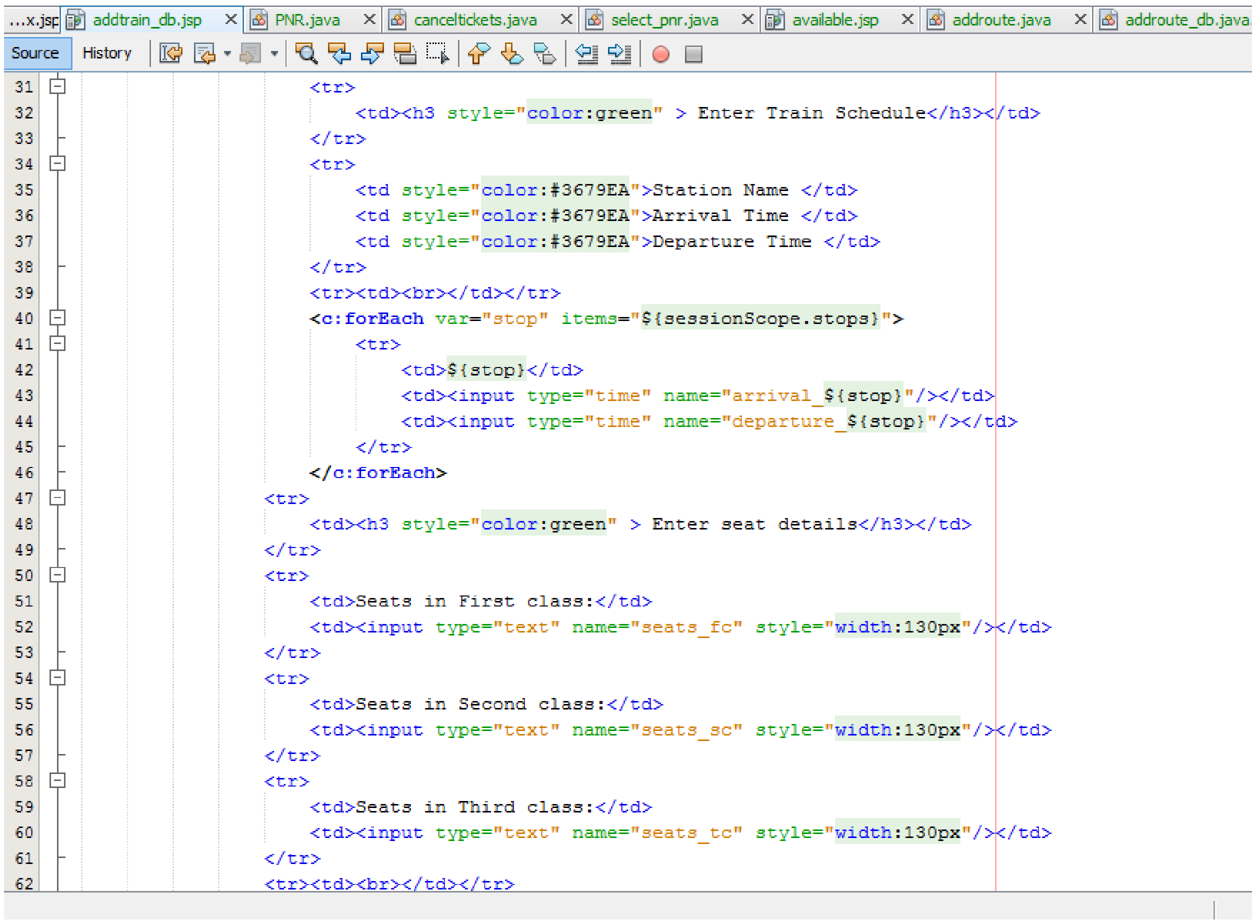
In the above code, admin is asked to enter the stations to be added in a new route and also the absolute value of distance from source(in kilometers).

Add New Trains - Admin enters the train number, train name and selects the route on which the train will run. Then he will select the stations on which the train will stop on a given route. Also he will select the Days of Week on which the train will run. Upon entering above details, he will need to enter arrival and departure timings for each stop and will also need to enter total number of seats in each class



```
        ArrayList<Integer> kilometersArrayList = new ArrayList<Integer>();

        try
        {
            Class.forName(DB_Driver);
            connection = DriverManager.getConnection(DB_Con, DB_Uname, DB_Password);
            PreparedStatement preparedStatement;
            ResultSet resultSet;

            String query = "select stations,kilometers from route where route_name = ? order by kilometers";
            preparedStatement = connection.prepareStatement(query);
            preparedStatement.setString(1, route);

            resultSet = preparedStatement.executeQuery();
            while (resultSet.next())
            {
                stationArrayList.add(resultSet.getString("stations"));
                kilometersArrayList.add(resultSet.getInt("kilometers"));
            }
            session.setAttribute("stations", stationArrayList);
            session.setAttribute("kilometers", kilometersArrayList);
            response.sendRedirect(request.getContextPath() + "/select_stops.jsp");

            resultSet.close();
            preparedStatement.close();
        }
        catch (ClassNotFoundException e)
        {
            out.println(e);

        }
        catch (SQLException e)
```

In the above code, the list of available stations in the selected route are fetched from the database and stored in station Arraylist which is stored further in a session attribute named stations.



```
                    <tr>
                        <td><h3 style="color:green" > Enter Train Schedule</h3></td>
                    </tr>
                    <tr>
                        <td style="color:#3679EA">Station Name </td>
                        <td style="color:#3679EA">Arrival Time </td>
                        <td style="color:#3679EA">Departure Time </td>
                    </tr>
                    <tr><td><br></td></tr>
                    <c:forEach var="stop" items="${sessionScope.stops}">
                        <tr>
                            <td>${stop}</td>
                            <td><input type="time" name="arrival_${stop}"/></td>
                            <td><input type="time" name="departure_${stop}"/></td>
                        </tr>
                    </c:forEach>
                <tr>
                    <td><h3 style="color:green" > Enter seat details</h3></td>
                </tr>
                <tr>
                    <td>Seats in First class:</td>
                    <td><input type="text" name="seats_fc" style="width:130px"/></td>
                </tr>
                <tr>
                    <td>Seats in Second class:</td>
                    <td><input type="text" name="seats_sc" style="width:130px"/></td>
                </tr>
                <tr>
                    <td>Seats in Third class:</td>
                    <td><input type="text" name="seats_tc" style="width:130px"/></td>
                </tr>
                <tr><td><br></td></tr>
```

After selecting the list of stops for a new train, admin has to enter the arrival and departure time for each stop as well as the total number of seats in each class.

For this we have used JSTL, in which <c:foreach> iterates through the entire arraylist and for each stop, corresponding arrival and departure time are taken from the administrator.

Chapter 6

Testing

Testing:

Testing Plan:

Testing technique that is going to be used in this project is black box testing. Appropriate inputs are applied to the system and the output is validated.

Testing Strategy:

The development process repeats this testing process a number of times for the following phases.

- Unit Testing.
- Integration Testing

Unit Testing tests a unit of code after coding of that unit is completed. Integration Testing tests the integration amongst the units or modules. System testing ensures that the system meets its stated design specifications. Acceptance testing is testing by users to ascertain whether the system developed is a correct implementation of the software requirements specification.

Testing is carried out in a hierarchical manner so that each component is correct and the assembly/combination of component is correct. Merely testing a whole system in the end would most likely throw many errors that would be very costly to trace and fix. We have performed both Unit Testing and System Testing to detect and fix errors.

Testing Methods:

We have performed Black-box testing for the testing purpose. A brief description is given below:

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well.

Test Cases:

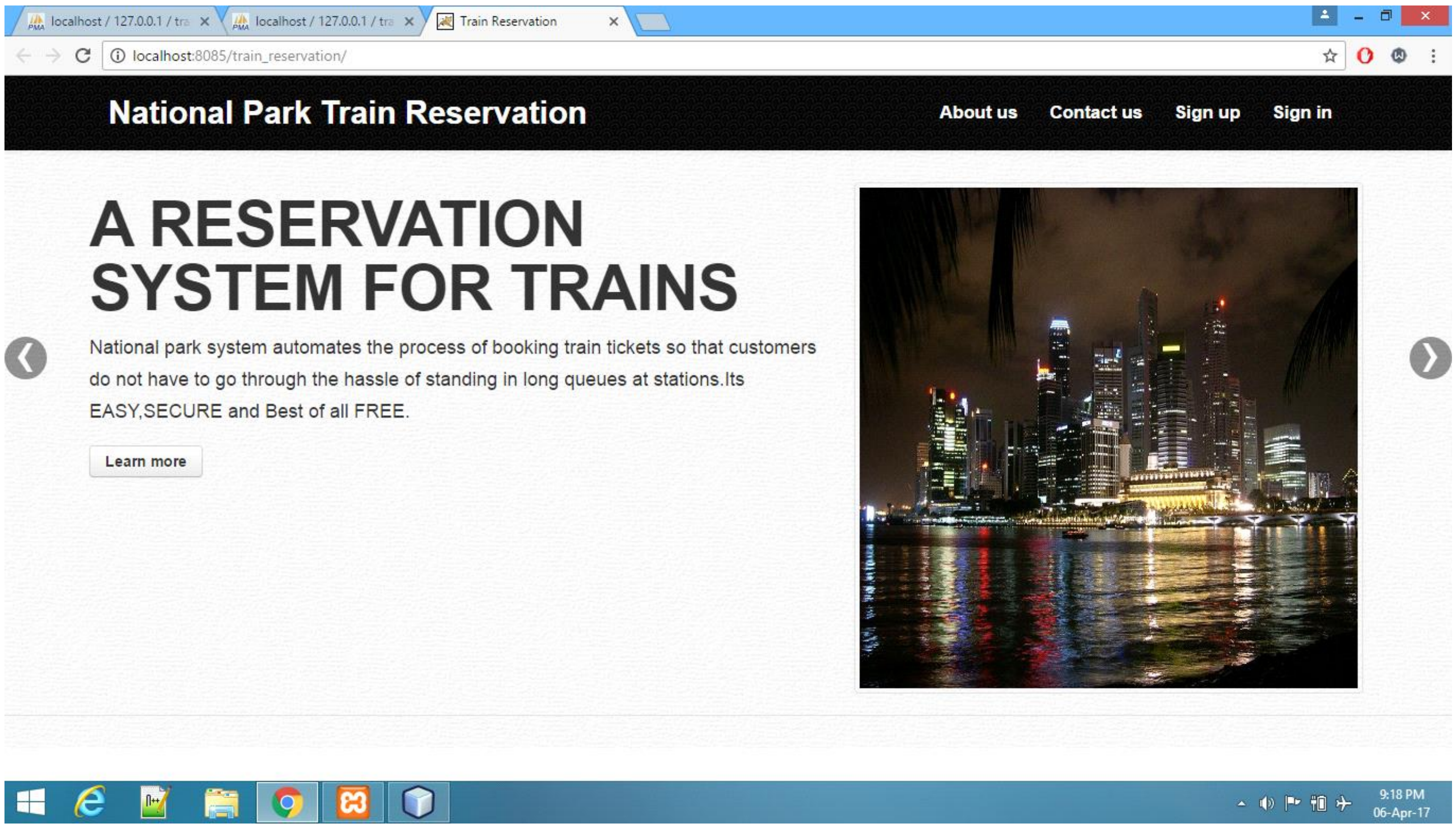
Test Case ID	**Software Requirement No.**	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Results**	**Actual Results**
T01	R2	Add new Station	1. Enter Station Details 2. Click Submit	Station Details	New Station Added	PASS
T02	R2	Add new Route	1. Select Starting Station and number of stations 2. Enter distance from starting station for each stops 3. Click Submit	Stations and their distances From starting station	New Route Added	PASS
T03	R3	Add new Train	1. Enter Train Details and Select Route 2. Click Submit 3. Select stops and running	Train Details and schedule	New Train Added	PASS

			days 4.Enter schedule for each station and seat details			
T04	R5	View Available Trains	1. Select Source, Destination stations and date of journey. 2. Click Submit	Travel info	List of available Trains and Seats Availability for each	PASS
T05	R5	Book Tickets	1.Select Train Number and Class 2. Click Submit	Train details	Ticket Successfully Booked And Email verification	PASS
T06	R6	Cancel Tickets	1. Select PNR to cancel 2. Click Submit	PNR number	Successful cancellation	PASS

Chapter 7

Screenshots of the System

- **Default Index Page**

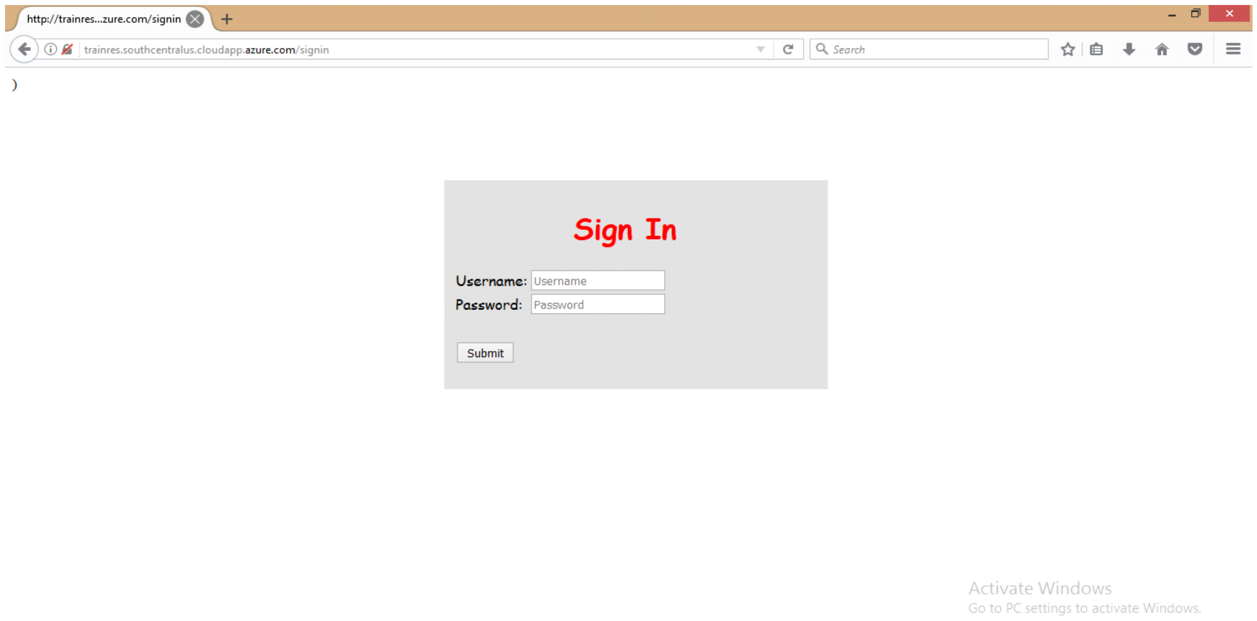

localhost / 127.0.0.1 / tra
localhost / 127.0.0.1 / tra
Train Reservation
localhost:8085/train_reservation/
National Park Train Reservation
About us
Contact us
Sign up
Sign in

A RESERVATION SYSTEM FOR TRAINS

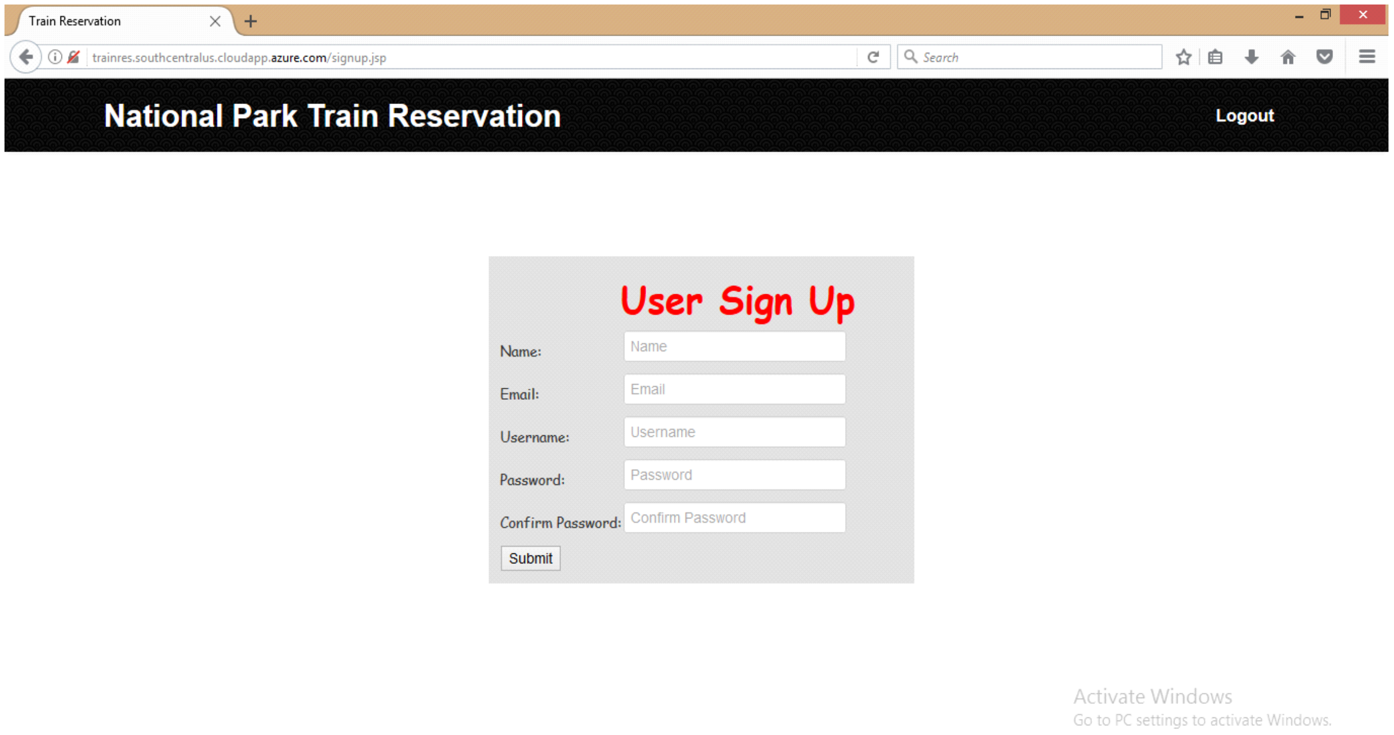
National park system automates the process of booking train tickets so that customers do not have to go through the hassle of standing in long queues at stations.Its EASY,SECURE and Best of all FREE.

Learn more
9:18 PM
06-Apr-17

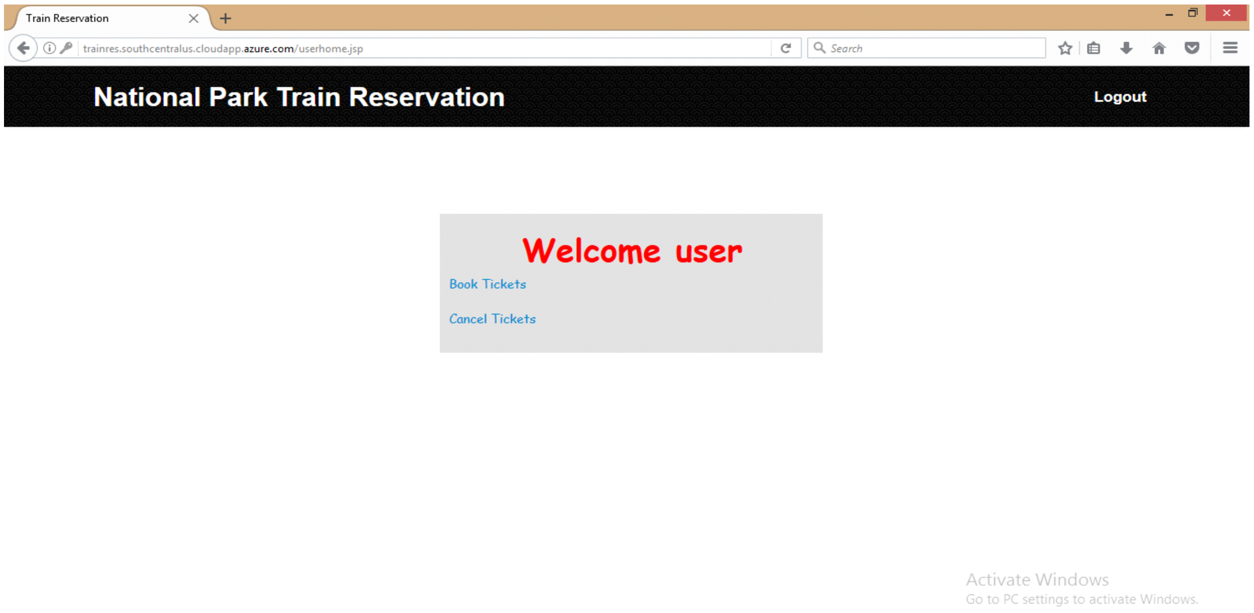
- **Sign In Page**


http://trainres...zure.com/signin
trainres.southcentralus.cloudapp.azure.com/signin
Search
Sign In
Username:
Username
Password:
Password
Submit
Activate Windows
Go to PC settings to activate Windows.

- **Sign Up Page**



- **User Home Page**



Train Reservation

trainres.southcentralus.cloudapp.azure.com/userhome.jsp

National Park Train Reservation

Logout

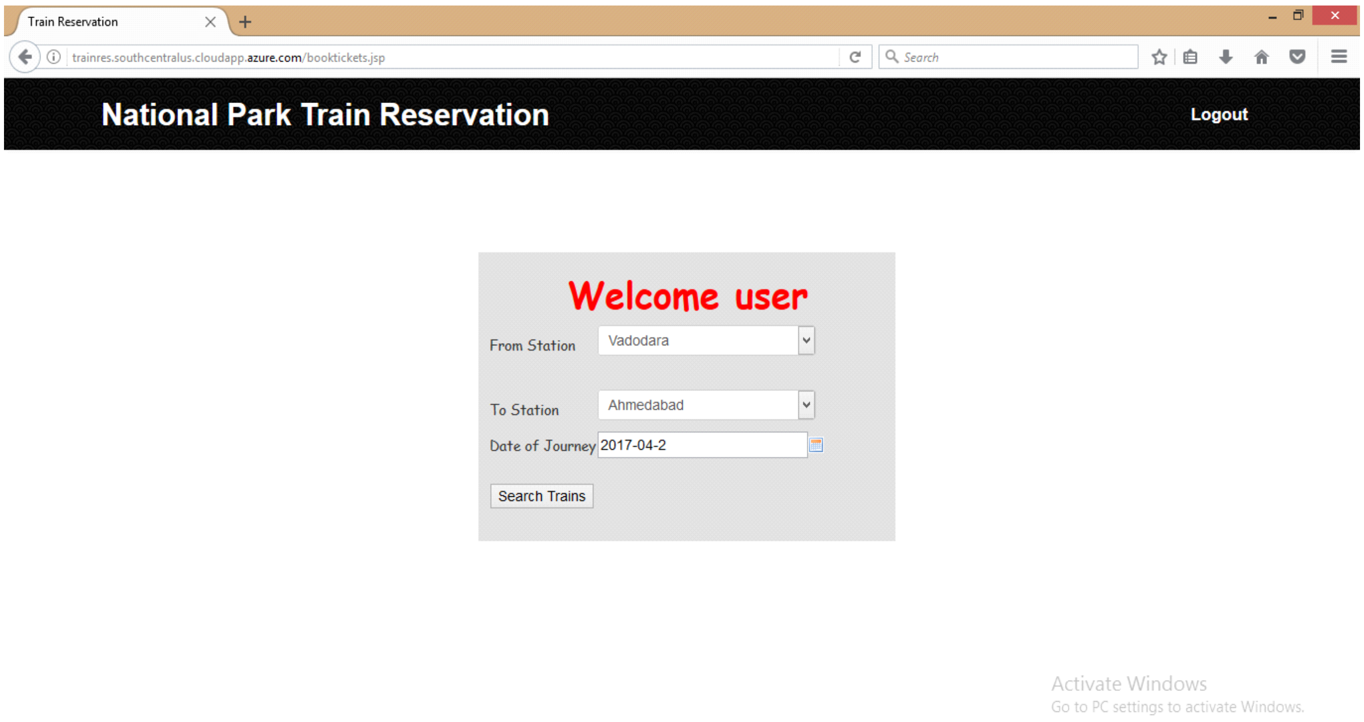
Welcome user

Book Tickets

Cancel Tickets

Activate Windows
Go to PC settings to activate Windows.

- **Select Source and Destination Stations along with date**



Train Reservation

trainres.southcentralus.cloudapp.azure.com/booktickets.jsp

National Park Train Reservation

Logout

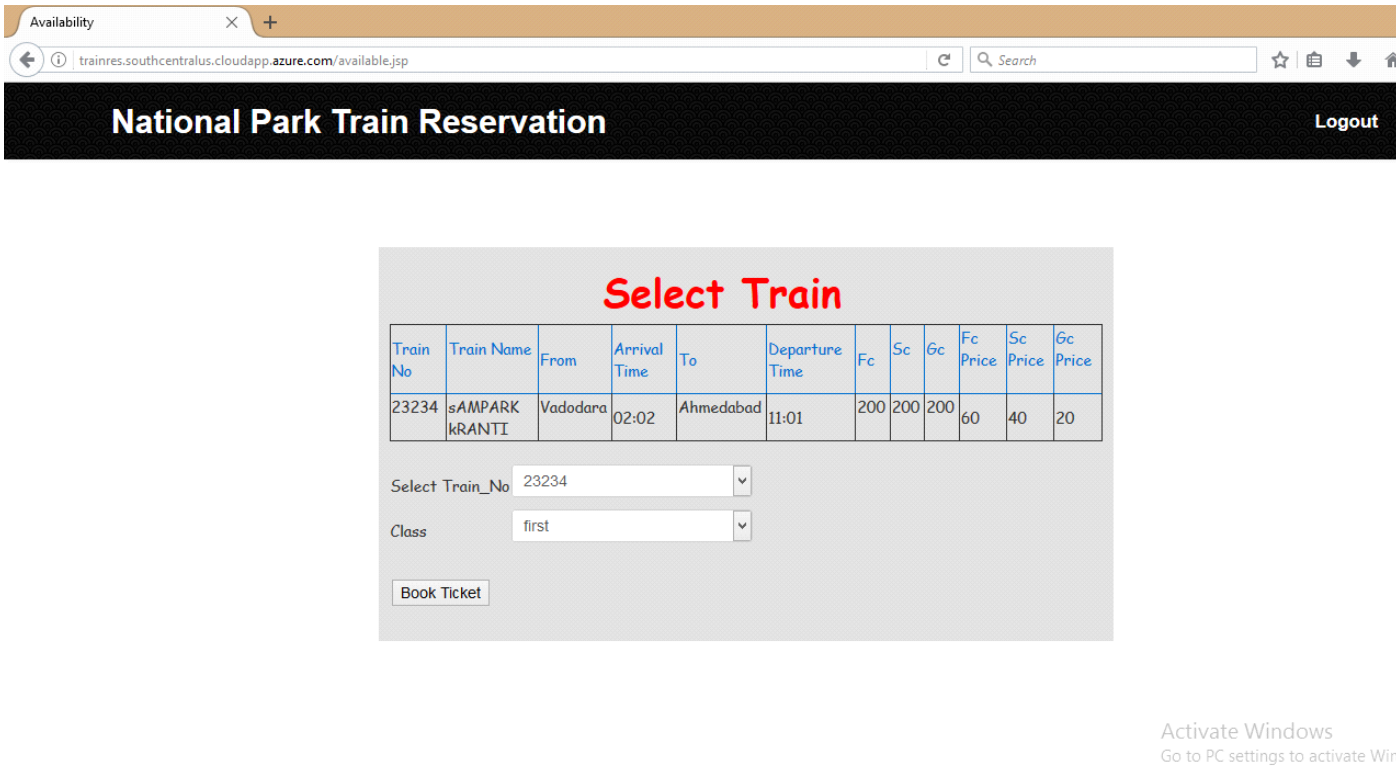
Welcome user

From Station	Vadodara
To Station	Ahmedabad
Date of Journey	2017-04-2

Search Trains

Activate Windows
Go to PC settings to activate Windows.

- **Available Trains**



Availability

trainres.southcentralus.cloudapp.azure.com/available.jsp

National Park Train Reservation

Logout

Select Train

Train No	Train Name	From	Arrival Time	To	Departure Time	Fc	Sc	Gc	Fc Price	Sc Price	Gc Price
23234	sAMPARK kRANTI	Vadodara	02:02	Ahmedabad	11:01	200	200	200	60	40	20

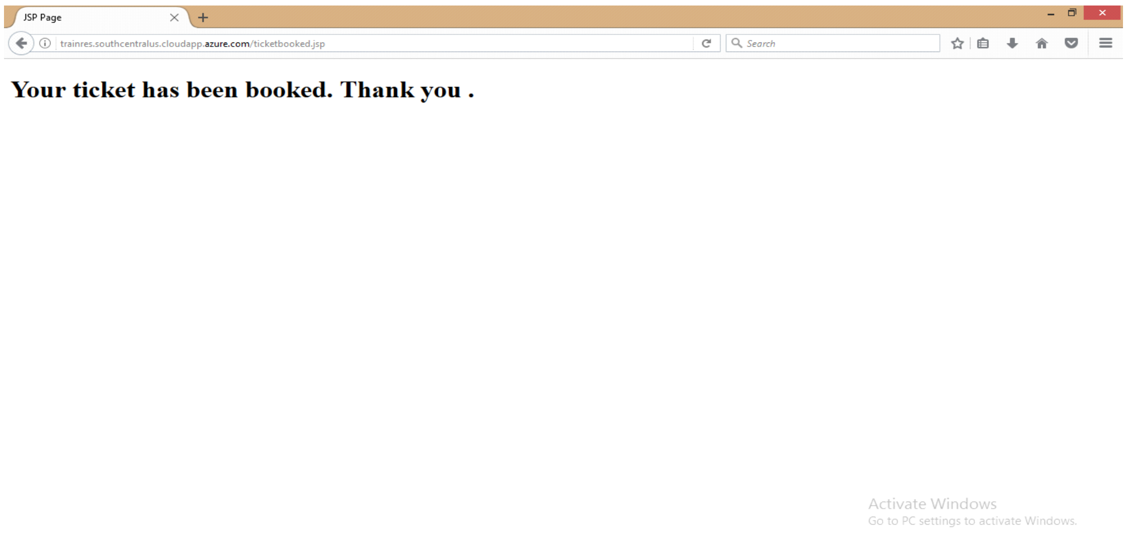
Select Train_No: 23234

Class: first

Book Ticket

Activate Windows
Go to PC settings to activate Wir

- Booking Confirmation



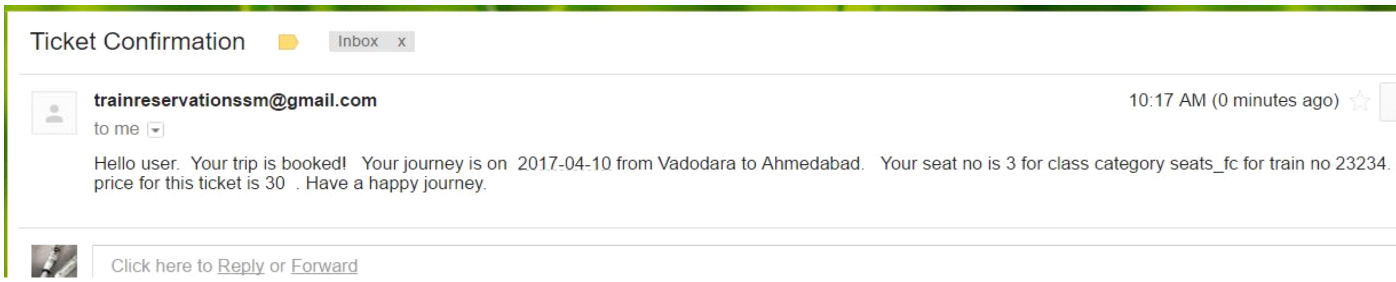
JSP Page

trainres.southcentralus.cloudapp.azure.com/ticketbooked.jsp

Your ticket has been booked. Thank you .

Activate Windows
Go to PC settings to activate Windows.

- **Email Confirmation**



Ticket Confirmation Inbox x

trainreservationssm@gmail.com

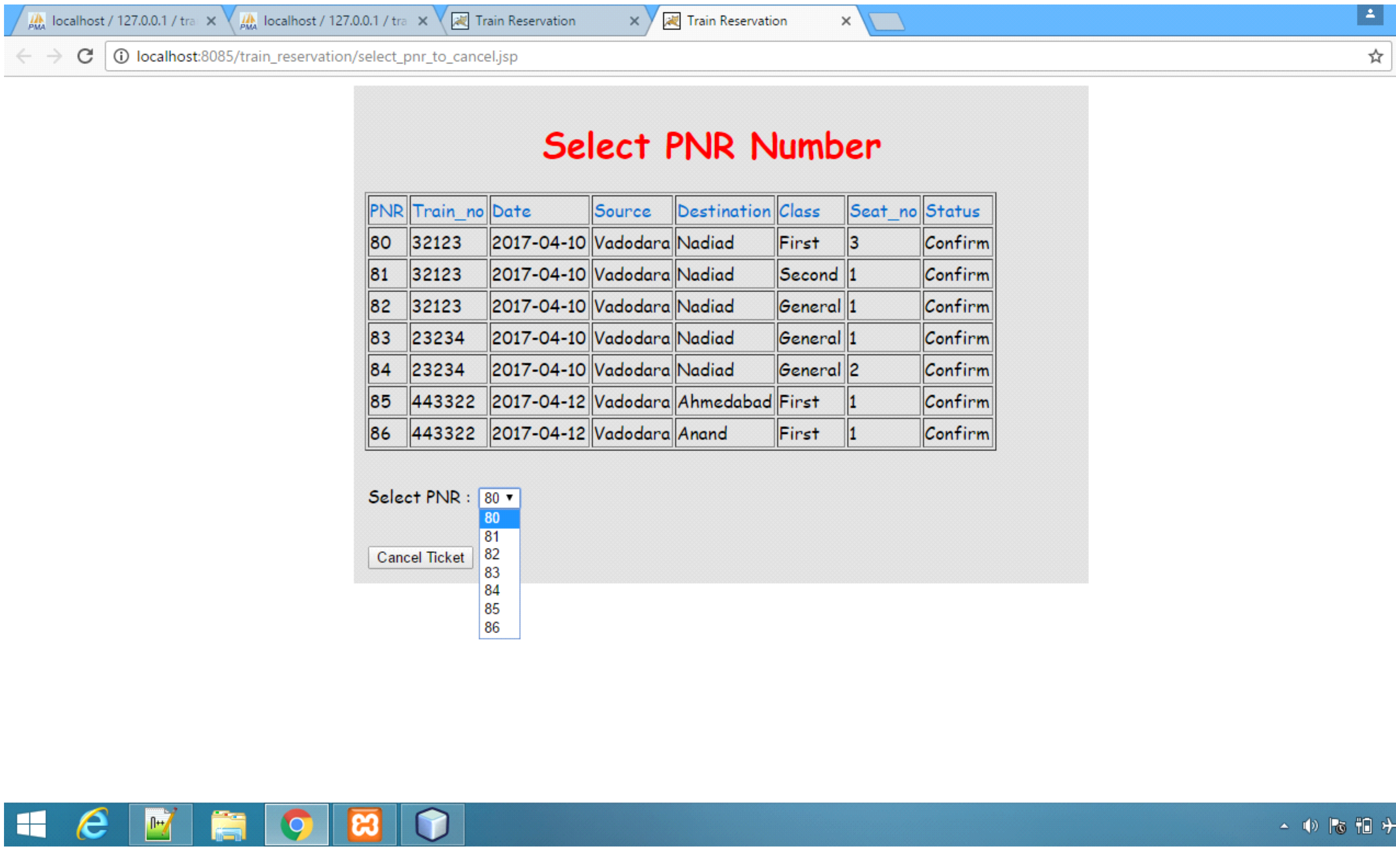
to me

10:17 AM (0 minutes ago)

Hello user. Your trip is booked! Your journey is on 2017-04-10 from Vadodara to Ahmedabad. Your seat no is 3 for class category seats_fc for train no 23234.
price for this ticket is 30 . Have a happy journey.

Click here to Reply or Forward

- **Cancel Tickets**

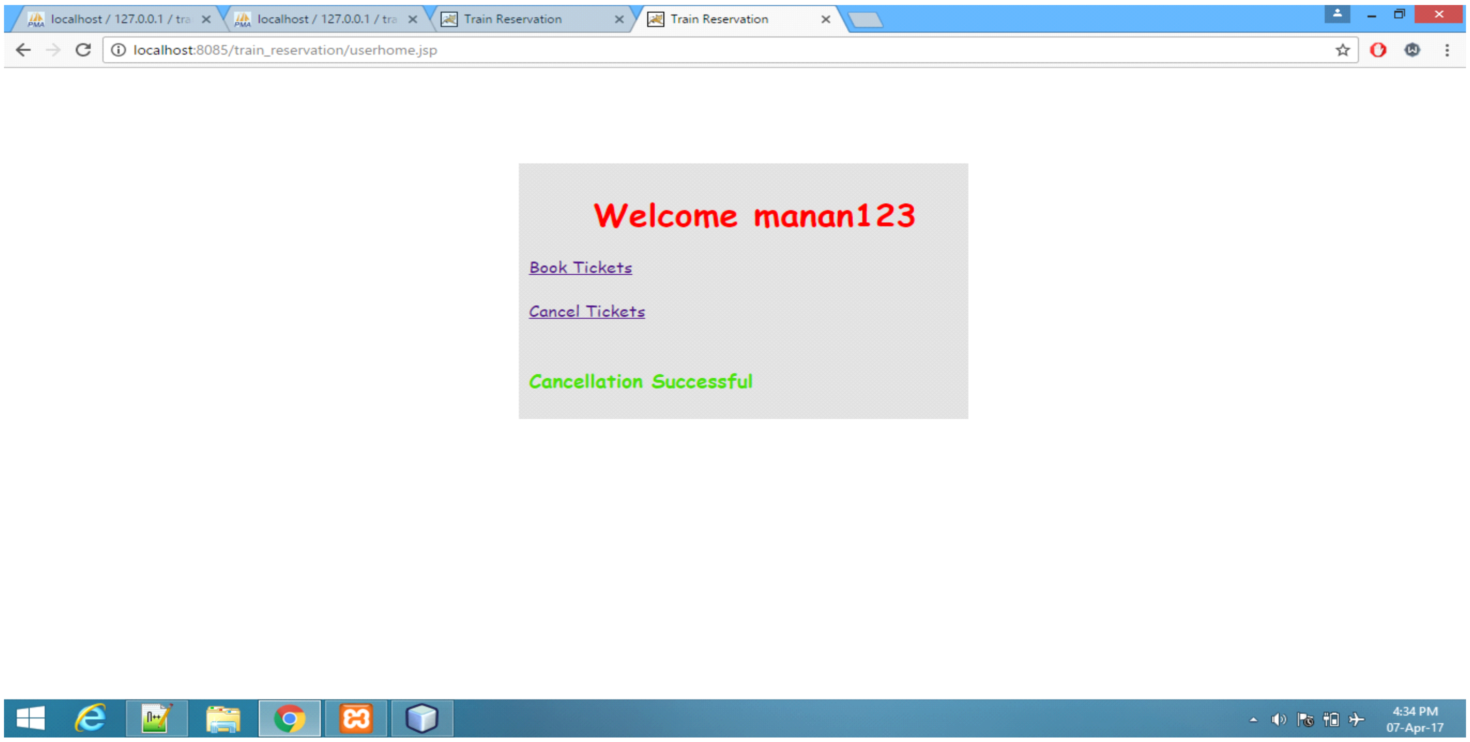


localhost:8085/train_reservation/select_pnr_to_cancel.jsp

Select PNR Number

PNR	Train_no	Date	Source	Destination	Class	Seat_no	Status
80	32123	2017-04-10	Vadodara	Nadiad	First	3	Confirm
81	32123	2017-04-10	Vadodara	Nadiad	Second	1	Confirm
82	32123	2017-04-10	Vadodara	Nadiad	General	1	Confirm
83	23234	2017-04-10	Vadodara	Nadiad	General	1	Confirm
84	23234	2017-04-10	Vadodara	Nadiad	General	2	Confirm
85	443322	2017-04-12	Vadodara	Ahmedabad	First	1	Confirm
86	443322	2017-04-12	Vadodara	Anand	First	1	Confirm

Select PNR : 80



localhost:8085/train_reservation/userhome.jsp

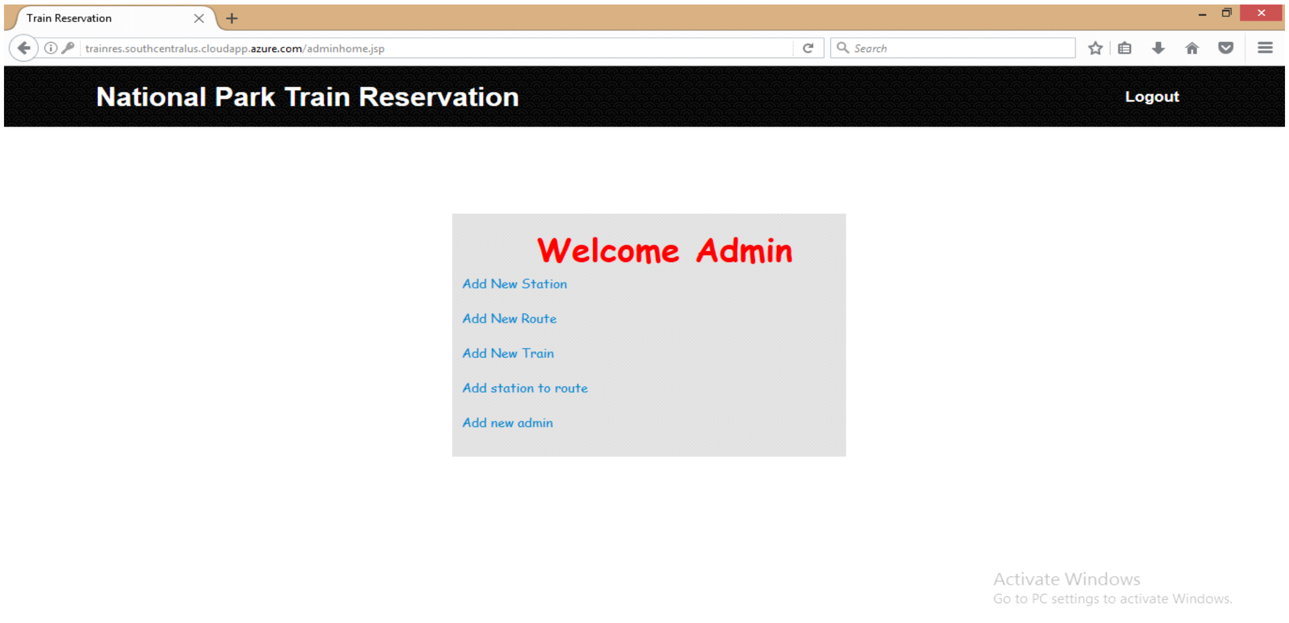
Welcome manan123

Book Tickets

Cancel Tickets

Cancellation Successful

- **Admin Home Page**



trainres.southcentralus.cloudapp.azure.com/adminhome.jsp

National Park Train Reservation

Logout

Welcome Admin

Add New Station

Add New Route

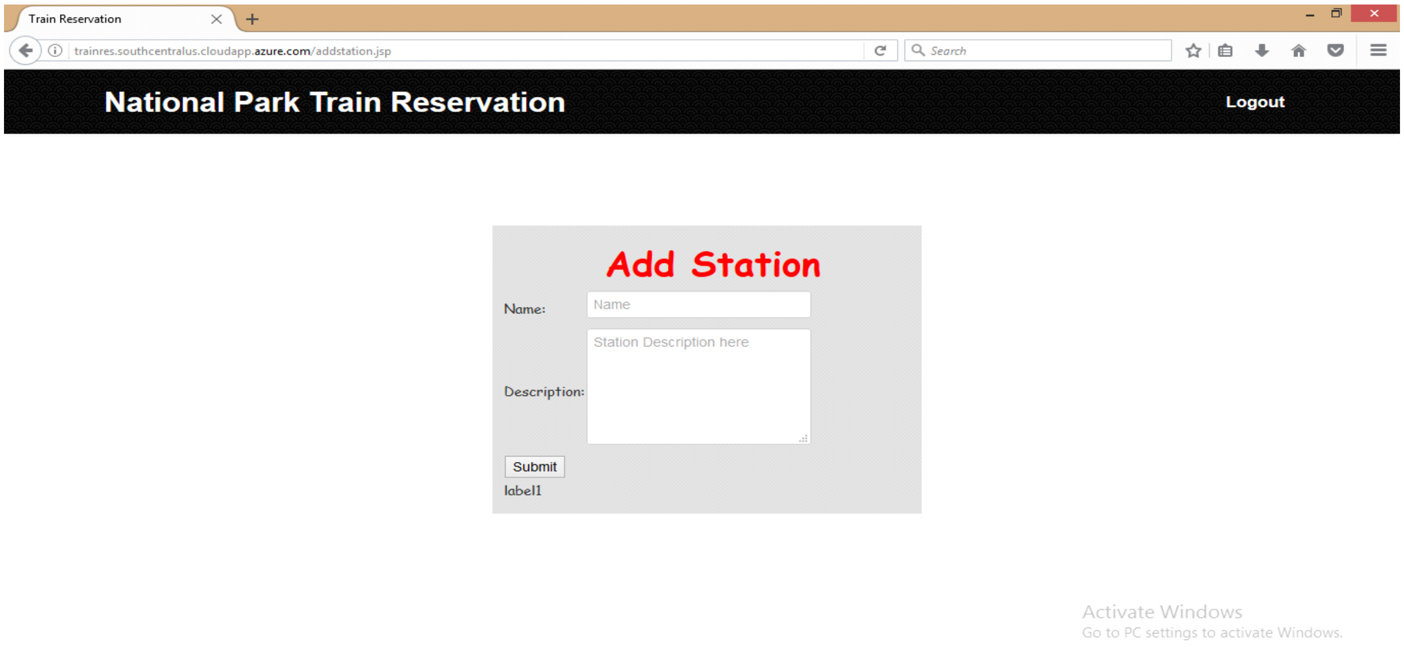
Add New Train

Add station to route

Add new admin

Activate Windows
Go to PC settings to activate Windows.

- **Add new Station**


Train Reservation

trainres.southcentralus.cloudapp.azure.com/addstation.jsp

National Park Train Reservation

Logout

Add Station

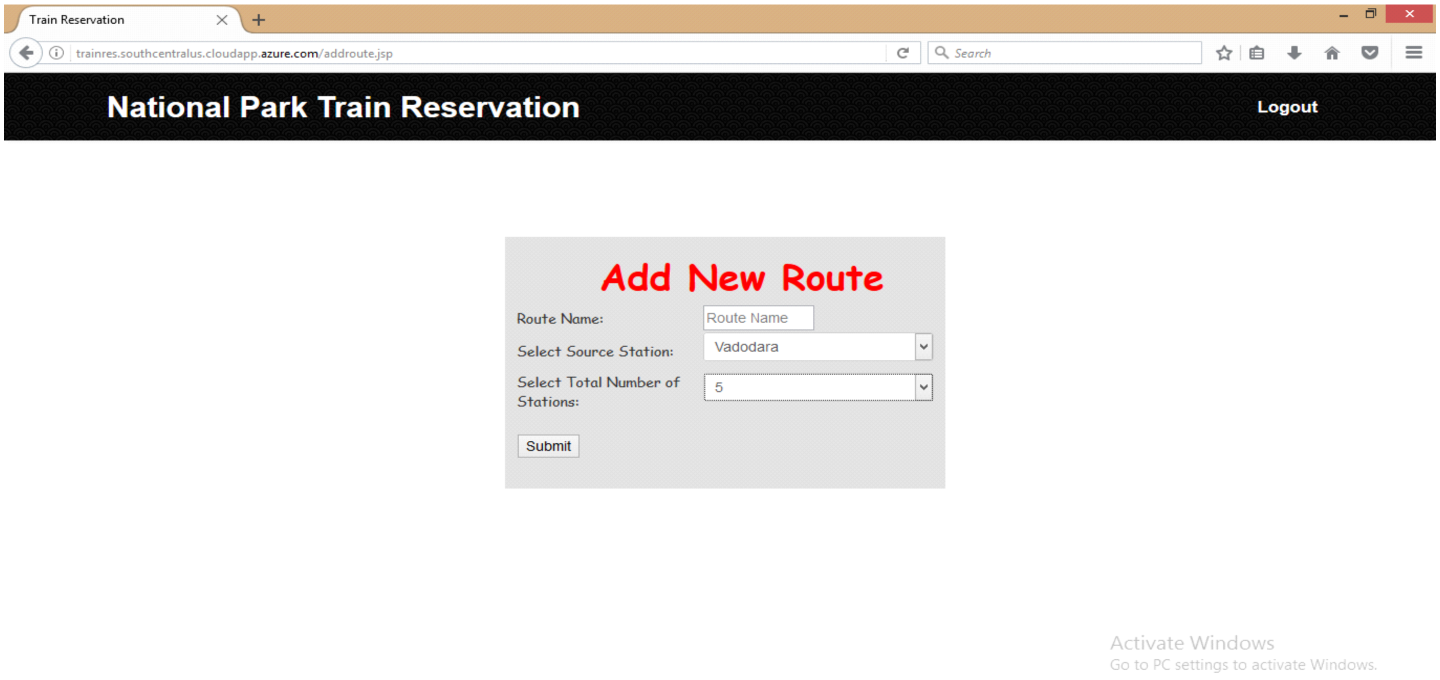
Name: Name

Description: Station Description here

Submit

label1

- **Add new Route**


Train Reservation

trainres.southcentralus.cloudapp.azure.com/addroute.jsp

National Park Train Reservation

Logout

Add New Route

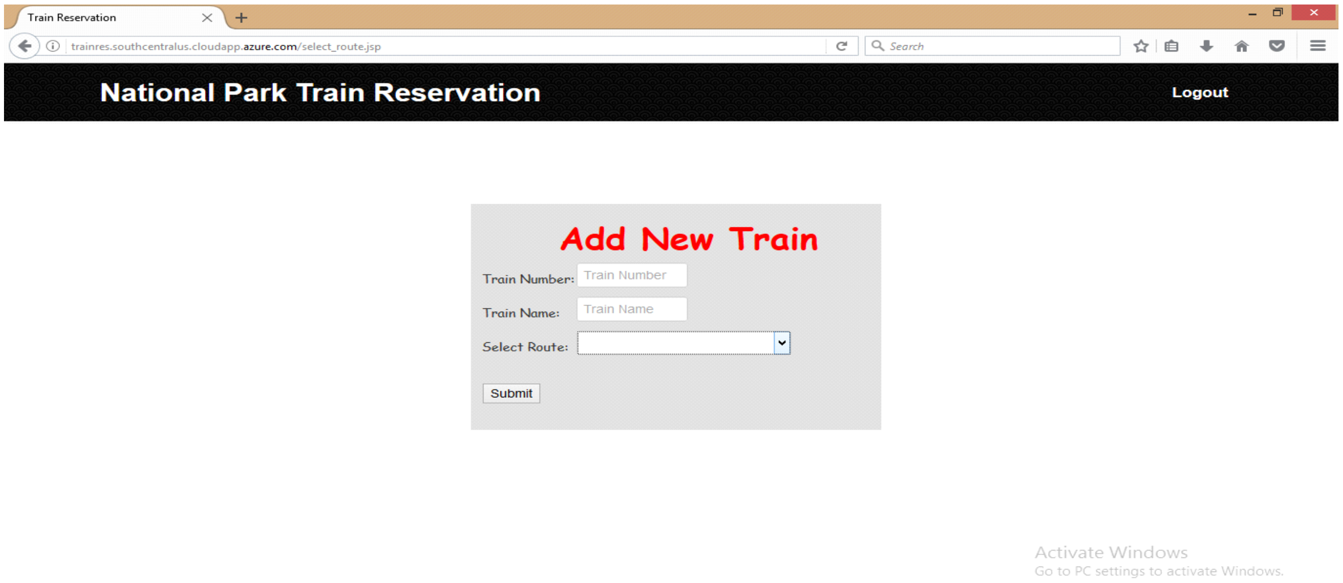
Route Name: Route Name

Select Source Station: Vadodara

Select Total Number of Stations: 5

Submit

- **Add new Train**



Train Reservation

trainres.southcentralus.cloudapp.azure.com/select_route.jsp

National Park Train Reservation

Logout

Add New Train

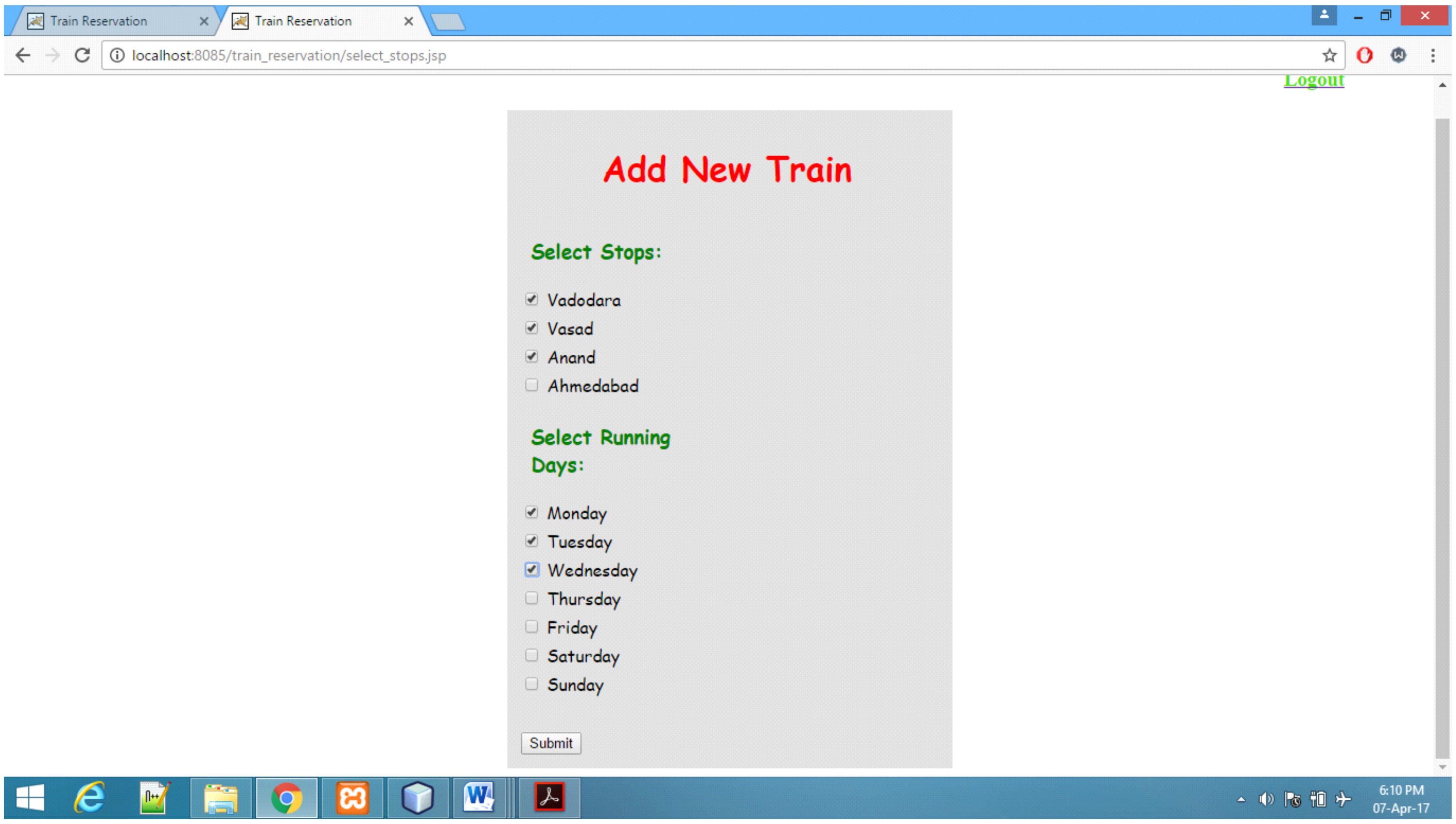
Train Number: Train Number

Train Name: Train Name

Select Route:

Submit

Activate Windows
Go to PC settings to activate Windows.



Train Reservation

localhost:8085/train_reservation/select_stops.jsp

Logout

Add New Train

Select Stops:

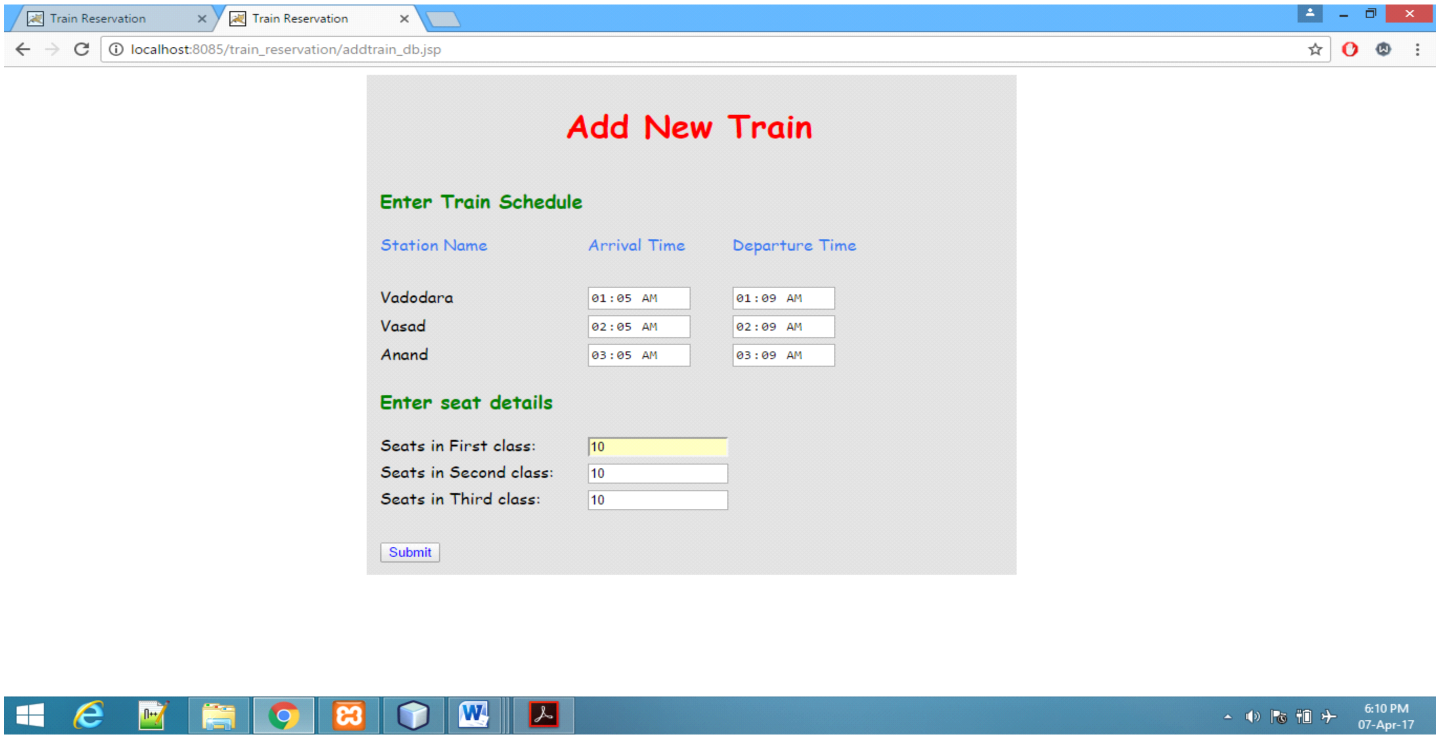
- [✓] Vadodara
- [✓] Vasad
- [✓] Anand
- [ ] Ahmedabad

Select Running Days:

- [✓] Monday
- [✓] Tuesday
- [✓] Wednesday
- [ ] Thursday
- [ ] Friday
- [ ] Saturday
- [ ] Sunday

Submit

6:10 PM
07-Apr-17


Train Reservation
Train Reservation
localhost:8085/train_reservation/addtrain_db.jsp

Add New Train

Enter Train Schedule

Station Name	Arrival Time	Departure Time
Vadodara	01:05 AM	01:09 AM
Vasad	02:05 AM	02:09 AM
Anand	03:05 AM	03:09 AM

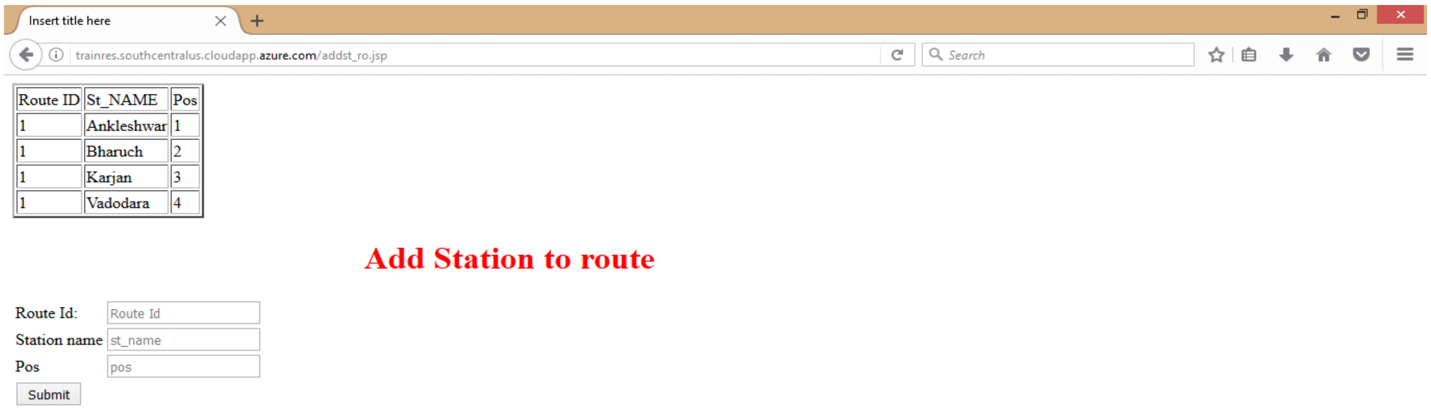
Enter seat details

Seats in First class:	10
Seats in Second class:	10
Seats in Third class:	10

Submit

6:10 PM
07-Apr-17

- **Add Station to Route**


Insert title here
trainres.southcentralus.cloudapp.azure.com/addst_ro.jsp

Route ID	St_NAME	Pos
1	Ankleshwar	1
1	Bharuch	2
1	Karjan	3
1	Vadodara	4

Add Station to route

Route Id:	Route Id
Station name	st_name
Pos	pos

Submit

Chapter 8

Conclusion

Conclusion:

Hence we developed the project by understanding all the modules of this project. We checked the feasibility and requirements of this system. Then we defined the overall look and processing logic for different modules on paper. After this, we started actual design of our modules in Java. All the modules of our system were developed separately. Then we integrated all modules by means of control flow among all modules.

After Coding and integration of all modules, we tested all modules separately using Unit Testing. After completion of Unit Testing, whole system was then tested once again using Integration Testing. Black box testing was carried out by designing suitable test cases and by validating output for each test case.

Chapter 9

Limitations and further Extensions

Limitations:

- Various factors such as allocation of seats according to weight distribution algorithms have not been taken into consideration while booking tickets.
- At a time booking for only one passenger can be done for a particular user id.
- Only one payment method is provided that is paytm.

Extensions:

- The services that this project provides can be useful for making reservations in similar transportation systems like trams, special trains and buses (holiday specials).
- The project focuses on reservation of a ticket online after user logs in using his username and password. But we can extend ticket booking to offline mode. Where there can be a different user role for staff member who logs in, accepts passenger information and trips detail from customer ,feeds it to the system and then the system generates their tickets.

Chapter 10

Miscellaneous

Form Based Authentication:

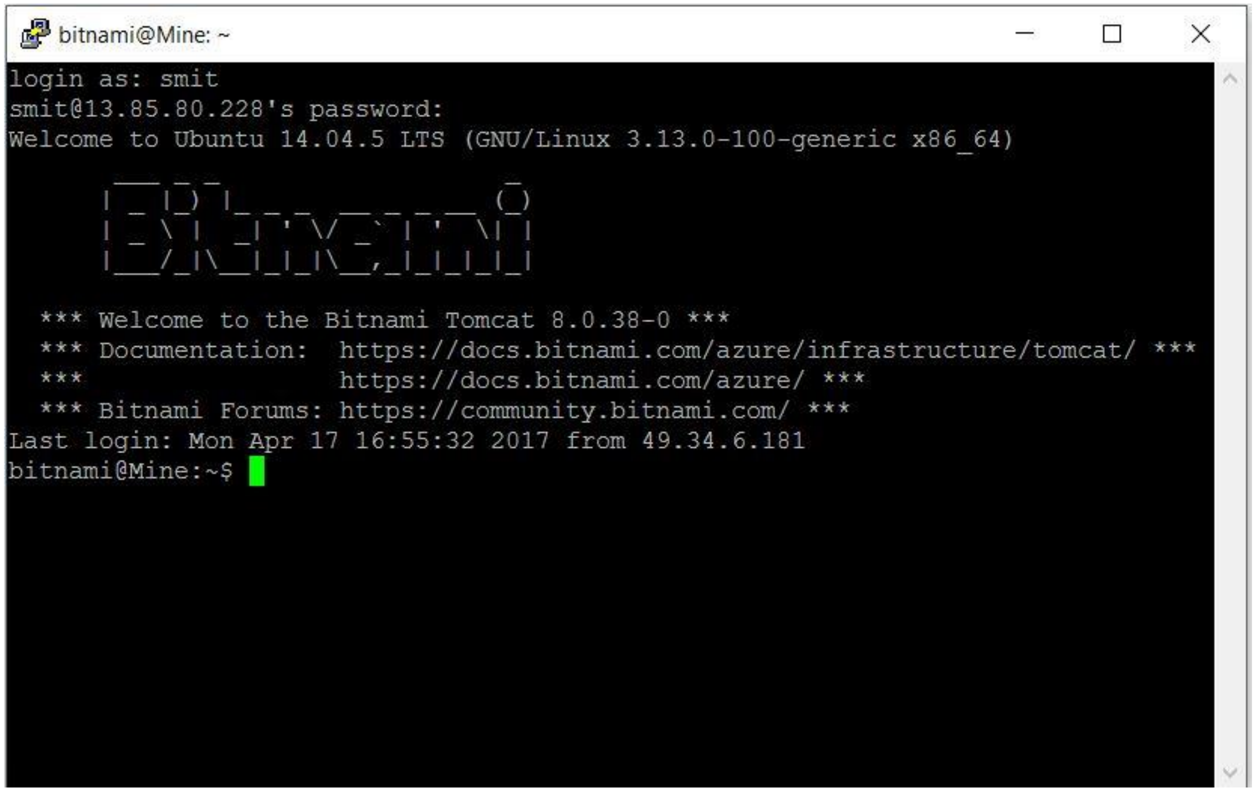
1. For implementing form based authentication using values from database we have used jdbc realm and credential handeler for validating md5 digested password. Above fields are defined in our tomcat's server.xml file.

```
<Realm className="org.apache.catalina.realm.JDBCRealm"
      driverName="org.gjt.mm.mysql.Driver"
   connectionURL="jdbc:mysql://localhost/trainreservation?user=smit1&amp;password=smit1"
       userTable="customer" userNameCol="Username" userCredCol="Password"
   userRoleTable="user_roles" roleNameCol="role_name">
    <CredentialHandler
          className="org.apache.catalina.realm.MessageDigestCredentialHandler"
algorithm="md5" /></Realm></Realm>
```

>The above snippet shows that for using database for retrieving user credentials rather than tomcat-users.xml we have created a jdbc realm using <realm> tag and as the password are encrypted before they are stored in md5 format we have used <credentialHandler> so the hashed values are respectively compared with the hashed value of password entry made by user.

Software Version Deployment Document:

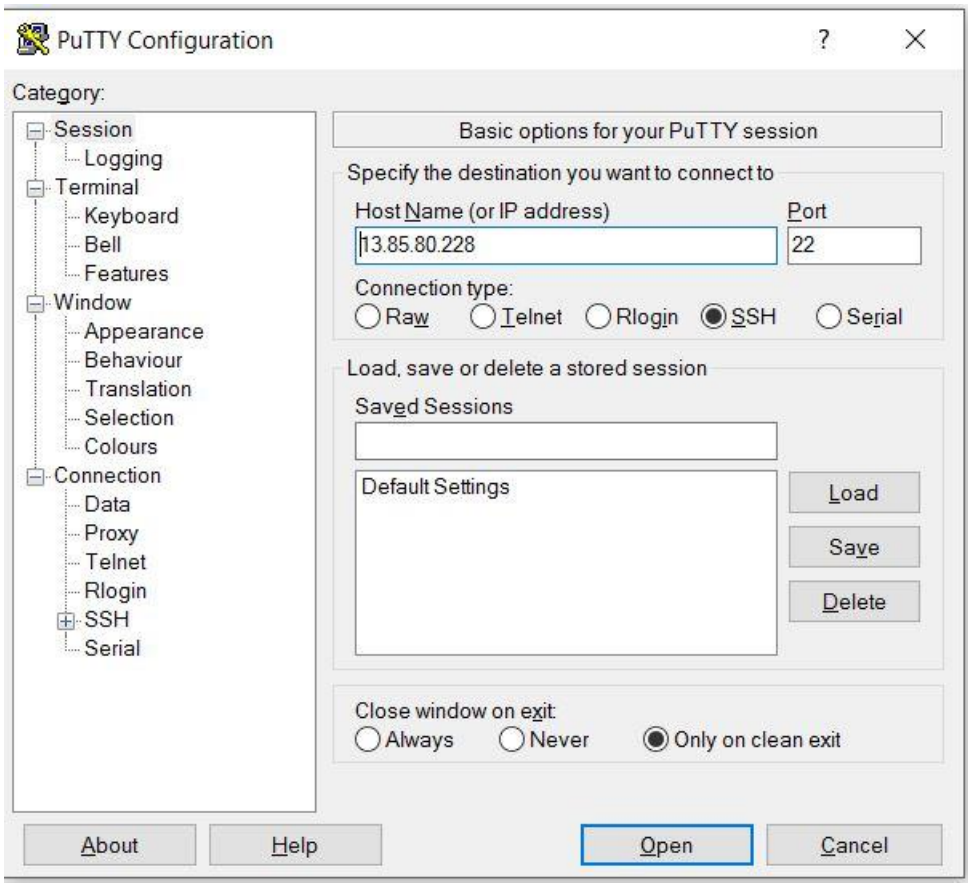
1. For deployment we have used bitnami virtual machine with public ip and dns name.



bitnami@Mine: ~

```
login as: smit
smit@13.85.80.228's password:
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-100-generic x86_64)
       ___ _ _                   _
      | _ |_) |_ _ _  __ _ _ __ (_)
      | _ \ |  _| ' \/ _` | '  \| |
      |___/_|\__|_|_|\__,_|_|_|_|_|

  *** Welcome to the Bitnami Tomcat 8.0.38-0 ***
  *** Documentation:  https://docs.bitnami.com/azure/infrastructure/tomcat/ ***
  ***                 https://docs.bitnami.com/azure/ ***
  *** Bitnami Forums: https://community.bitnami.com/ ***
Last login: Mon Apr 17 16:55:32 2017 from 49.34.6.181
bitnami@Mine:~$
```

2. We used ssh tunneling using "putty" to access phpmyadmin on bitnami virtual machine on which we have hosted our database.


PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 13.85.80.228

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

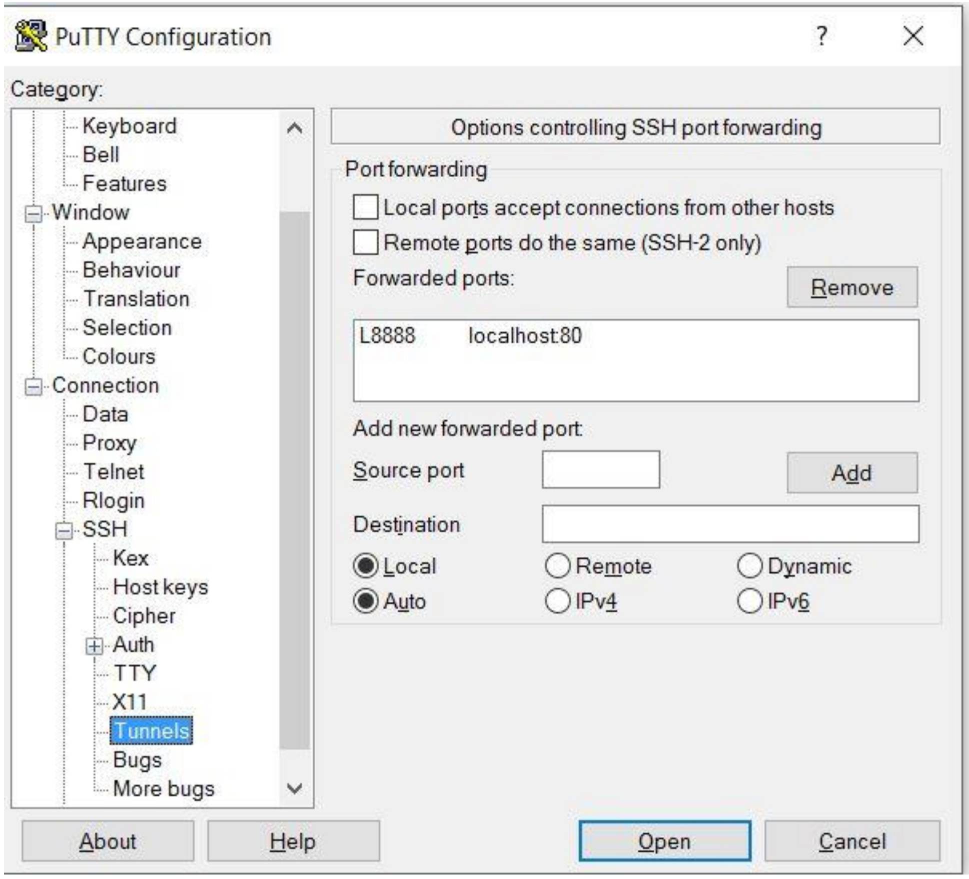
Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

3. To transfer war file to bitnami virtual machine we used sftp via filezilla and changed it's name to ROOT.war so that tomcat identifies it as the default webapp.



PuTTY Configuration

Category: Tunnels

Options controlling SSH port forwarding

Port forwarding

- [ ] Local ports accept connections from other hosts
- [ ] Remote ports do the same (SSH-2 only)

Forwarded ports:

```
L8888      localhost:80
```

Add new forwarded port:

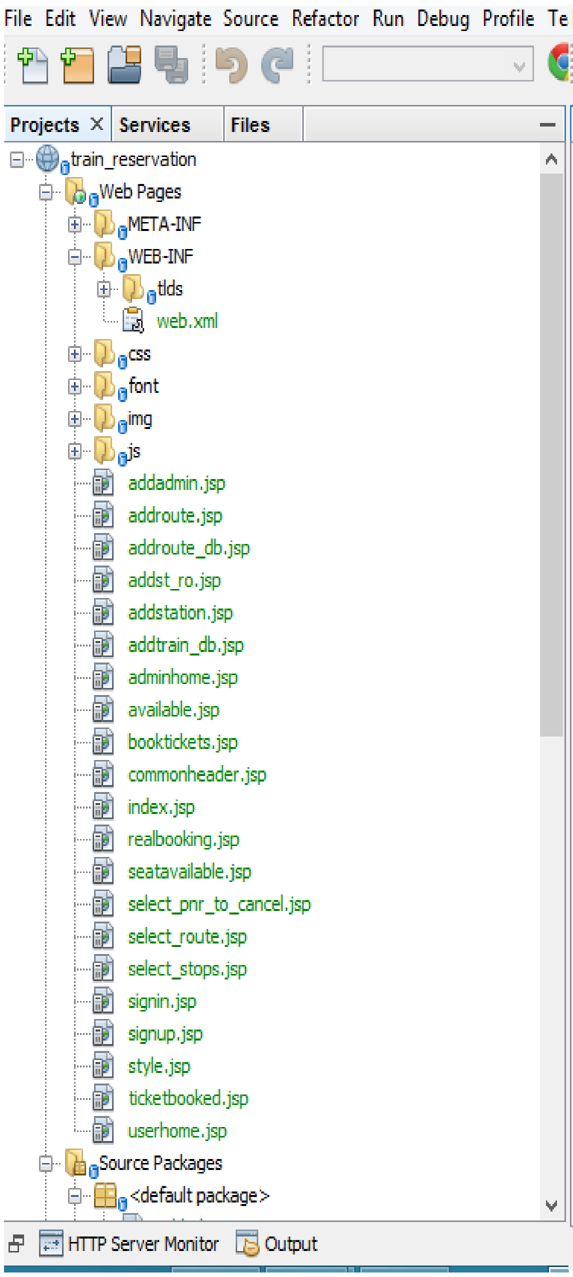
Source port:

Destination:

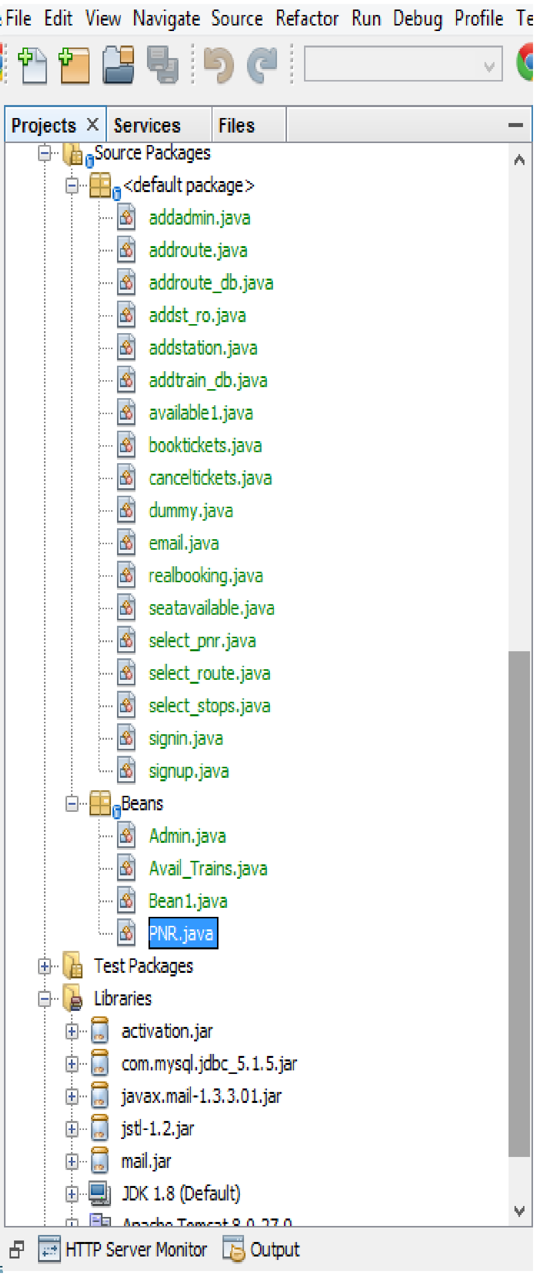
[✓ Local] Remote Dynamic

[✓ Auto] IPv4 IPv6

Directory Structure:


File Edit View Navigate Source Refactor Run Debug Profile Te
Projects
Services
Files
HTTP Server Monitor
Output

```
train_reservation
  Web Pages
    META-INF
    WEB-INF
      tlds
      web.xml
    css
    font
    img
    js
    addadmin.jsp
    addroute.jsp
    addroute_db.jsp
    addst_ro.jsp
    addstation.jsp
    addtrain_db.jsp
    adminhome.jsp
    available.jsp
    booktickets.jsp
    commonheader.jsp
    index.jsp
    realbooking.jsp
    seatavailable.jsp
    select_pnr_to_cancel.jsp
    select_route.jsp
    select_stops.jsp
    signin.jsp
    signup.jsp
    style.jsp
    ticketbooked.jsp
    userhome.jsp
  Source Packages
    <default package>
```


File Edit View Navigate Source Refactor Run Debug Profile Te
Projects
Services
Files
HTTP Server Monitor
Output

```
Source Packages
  <default package>
    addadmin.java
    addroute.java
    addroute_db.java
    addst_ro.java
    addstation.java
    addtrain_db.java
    available1.java
    booktickets.java
    canceltickets.java
    dummy.java
    email.java
    realbooking.java
    seatavailable.java
    select_pnr.java
    select_route.java
    select_stops.java
    signin.java
    signup.java
  Beans
    Admin.java
    Avail_Trains.java
    Bean1.java
    PNR.java
Test Packages
Libraries
  activation.jar
  com.mysql.jdbc_5.1.5.jar
  javax.mail-1.3.3.01.jar
  jstl-1.2.jar
  mail.jar
  JDK 1.8 (Default)
```

Work Distribution:

Module Number	Module Name	Developed By
1	User	
1.1	Login	Sagar Patel
1.1.1	Form Based Authentication and Hashing	Smit Purohit
1.2	Sign Up	Manan Shah
2	Admin	
2.1	Add new Station	Manan Shah
2.2	Add new Route	Manan Shah
2.3	Add new Train	Manan Shah
3	Passenger	
3.1	Book Tickets	
3.1.1	Display Available Trains	Sagar Patel
3.1.2	Update Capacity Details	Smit Purohit
3.1.3	Make Payment using PayTM	Smit Purohit
3.1.4	Send Email Notification	Sagar Patel
3.2	Cancel Tickets	Manan Shah