ST. XAVIER'S COLLEGE AUTONOMOUS

5 , MAHAPALIKA MARG,FORT,MUMBAI-400001

2017-18

A PROJECT REPORT ON

RAILWAY RESERVATION SYSTEM !!

BY

RAHUL MUKESH SINGH -155051

&

SANJANA GOPALKRISHNA -155025

Under the Guidance of

PROF. SUBHASH KUMAR

PROJECT CERTIFICATE

This is to certify that the project entitled **"RAILWAY RESERVATION SYSTEM"** is Undertaken at St. Xavier's College Autonomous , Mumbai by **Mr. Rahul Mukesh Singh**(**UID-155051**) and **Ms. Sanjana Gopalkrishna**(**UID-155025**) in the year **2017-18**. In Partial Fulfilment of B.ScIT Degree(**Semester VI**) Examination has not been submitted for any other examination and does not form part of any other course undergone by the candidate. It is further certified that he/she has completed all the required phases of the project .

Signature of	Signature of	Signature of
Internal Guide	Internal Examiner	External Examiner

Signature of H.O.D. College Seal

DECLARATION

Rahul Mukesh Singh(UID-155051) and **Sanjana Gopalkrishna(UID-155025)**, hereby declare that this project report entitled: **"RAILWAY RESERVATION SYSTEM"** which is being submitted in fulfilment of the Bachelors of Science in Information Technology Examination conducted by St. Xavier's College – [Autonomous, under Mumbai University] is the result of the work carried out by us under the supervision of **Prof. Subhash Kumar** of St. Xavier's College-Autonomous Mumbai.

This work has not been previously submitted to any other University for any examination. Wherever references have been made to previous work of others, it has been clearly indicated as such and included in the bibliography.

Signature
[Rahul Mukesh Singh]

Signature
[Sanjana Gopalkrishna]

Date:

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to our teachers for their exemplary guidance, monitoring and constant encouragement throughout the course of this project.

The blessings, help and guidance given by them, from time to time, shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Head Of Department, Mr.Roy Thomas for his cordial support.

A large debt of gratitude is owned to my project guide Mr.Subhash Kumar who has not only endured, but also assisted and inspired us for taking up the project on "Railway Reservation System".

We want to acknowledge and thank him for giving us the opportunity to do this under his guidance and for sharing his knowledge. His continuous guidance, time, valuable suggestions, inputs and helpful criticisms have helped us to accomplish such a task.

Lastly, We thank almighty, our parents, family and all those persons 'Behind the Veil' for their constant encouragement and support, which enabled us to complete the project through thick and thin.

TABLE OF CONTENTS

Contents **Page No.**

PRELIMINARY INVESTIGATION

1.1 SYSTEM OVERVIEW

The Indian Railways (IR) which is the world's largest railway system under single management carries about 22.2 millions passengers in reserved accommodation every day. India's as well as Asia's first train streamed off from Mumbai to Thane on 16th April, 1853 covering a distance of 34 kms. Later, Railway network grew very rapidly and spread across all over India.

The Computerised Passenger Reservation System(PRS) facilitates the booking and cancellation of tickets from any of the 4000 terminals(i.e. PRS booking window all over the countries). These tickets can be booked or cancelled for journeys commencing in any part of India and ending in any other part, with travel time as long as 72hours and distance up to several thousand kilometres. The project of PRS was launched on 15th November 1985, over Northern Railway with the installation of Integrated Multiple Train Passenger Reservation System(IMPRESS), an online transaction processing system developed by Indian Railways in association with Computer Maintenance Corporation Ltd., at New Delhi. The objective was to provide reserved accommodations on any train from any counter, preparation of train charts and accounting of the money collected. The application was implemented in 1987 Mumbai, Chennai, Kolkata, Secunderabad subsequently.

Birth of Indian Railway Catering and Tourism Corporation [IRCTC] took place on 27th September, 1999. Corporate office of IRCTC is situated in New Delhi. IRCTC is a Public Sector Enterprise under Ministry of Railways.

Main functionality of IRCTC by Indian Railways is to upgrade the reservation system online, professionalize and manage the catering and hospitality services at stations, on trains and other locations and to promote domestic and international tourism through development of budget hotels, special tour packages, information & commercial publicity and global reservation systems.

IRCTC is a G2C [Government-to-Customer] business which has a very effective effect towards a positive development of India.

1.2 DESCRIPTION OF SYSTEM

RAILWAY RESERVATION SYSTEM is a website which provides the customer with ease of ticket booking from anywhere, anytime in less than a hour.

The main purpose of this project is Reserving train seats and cancellation online. At Present, where India is moving towards Digital India , this is a small effort taken by us.

The various advantages of using Online Reservation System are as follows:

- Convenient – You can book or cancel your tickets sitting in the comfort of your home or office.
- Saves Time and Effort – You can save the time needed to travel to the railway reservation office and waiting in the queue for your turn.
- Towards a greener Planet – Instead of printing your tickets you can choose to travel with the soft copy of your booked ticket in your laptop or mobile.

1.3 OBJECTIVE AND SCOPE OF PROJECT

Our project introduces railway reservation system with an objective to make the reservation system more efficient, easier and fast. This project explores how computer technology can be used to solve the problem of user.

The scope of the project as of now is limited to College Project purpose which may further be enhanced to bring into real application purpose.

The main objectives provided by this software are as follows:

- We can enquire about availability of trains.
- We can reserve and cancel the seats.
- We can see the history of the bookings and cancellations made by a account.
- We can download Ticket in PDF format.
- We have an Emergency Chat Board for user to report to railways.
- Management of all the details regarding trains, routes, Customers, Tickets, Payment, Booking and many more.
- Administration module where the Trains, Routes, Costs, Emergency Requests, Quota, Coaches details can be inserted, updated, modified in the database.
- Report Generations for analytic purpose.

1.4 LIMITATIONS OF PRESENT SYSTEM

The Present System- IRCTC Railway Reservation Website is one of the most used and trusted website of India. Lakhs of people book and cancel their tickets through this system on daily basis.

Some of the limitations which we found in the IRCTC Website are as follows:

- A user can book only upto a maximum of 6 tickets per month and 2 tickets per day.
- Individuals are allowed only two tickets per User-ID in a day for Tatkal booking from 10 am to 12 pm.
- Maximum of only 4 members can be included in Single Ticket or Single PNR.

1.5 PROPOSED SYSTEM AND ITS ADVANTAGES

RAILWAY RESERVATION Website helps in booking train tickets and also helps the User to check the Seat Availability in Trains.

A user can also Cancel the tickets, check the history of the Bookings and Cancellations made through corresponding account.

Emergency Chatting Board is also built for the customers to get instant help. If any emergency permits during the travelling, he/she can message through Emergency Chat Board and corresponding action could be taken by the Railways.

Passenger can reserve any number of tickets from one account. Various Seat Quotas have been provided for reserving seats. Also, according to the quotas, Concession has been provided per seat.

An Individual can book their ticket according to the Coaches they are comfortable. The Fare Prices of the individual varies according to the Coaches and Quotas selected by the user.

Individual can also see the number of Confirm, RAC, Waiting Seats available for the train and coach selected. Through this, the individual can get an idea of whether he/she should book the ticket for the selected train or he/she need to Check for different trains.

The payment of the Reservation is Online which can be done using Credit-card /Debit-card, etc.

The ticket is generated automatically which the user can download and save for travelling purpose.

Administration module which keeps the track of all the records of trains, quotas, coaches, costs, routes and many more.

Any future Updation can be made by the administration module.

Emergency chat where user requests for emergency request, the record of these are kept by administration module and corresponding actions would be taken by them.

REQUIREMENT SPECIFICATION

2.1 FUNCTIONAL REQUIREMENTS

Requirement Specification for a software system is a complete description of the behaviour of a system to be developed. It can be defined as what a system is supposed to accomplish. A function can be set of inputs, behaviour, and also outputs.

Following are the factors included in our functional requirements:

1. **Performance Requirements:**

   a) *Satisfaction of User:*

      The system is developed such that it meets user's needs.

   b) *Response Time:*

      By careful Programming, the response time of all the operation is good.

   c) *Error Handling:*

      Response to user errors and undesired situations has been taken care of to ensure that the system operates without halting.

   d) *Safety and Robustness:*

The system is able to tackle disastrous actions without the intervention of human.

e) *Portable:*

The system is not architecture specific. It can be easily transformed to other platforms is needed.

f) *User friendliness:*

The system is easy to use and understand. The native user can also access the system without any difficulty.

2. Design Constrains :

There are a number of factors in the client's environment that may restrict the choices of a designer. These factors could be standards to be followed, resource limits, operating environment, reliability, etc. Requirement Specification should identify and specify all such constraints.

a) *Standard Compliance:*

Requirements for the standards the system must follow. Standards may include report format.

b) *Hardware Limitations:*

The software may have to operate on some existing or predetermined hardware, thus imposing restriction on the design. Hardware limitations may be type of machines used, operating system available on the system, languages supported and limits on primary and secondary storage.

c) *Security:*

Security requirements are significant in database systems. They play restriction on the us of certain commands, access to some data, provide different kind of access requirements for different people.

3. Hardware Requirements:

This specifies the hardware requirements like memory restrictions, cache size, processor, RAM size, etc.

4. Software Requirements:

Any window based operating system with DOS support is the primary requirement. Others softwares required are as follows:

a) Eclipse EE
b) Xampp-MySQL

2.2 NON-FUNCTIONAL REQUIREMENTS

1. **Security:**

   The system should not leave any cookies on the customer's computer containing the user's password. The system's backend servers should be only be accessible by administration side and authenticated management.

2. **Reliability:**

   The important foundation of the reliability is the backup of the database which is continuously maintained and updated.

3. **Availability:**

   The system should be available all the time i.e. the user can access it using a web browser.

4. **Supportability:**

   The codes and supporting modules of the system will be well documented and in understandable format.

FEASIBILITY STUDY

Feasibility study is a high level version of the entire system analysis and designing a process. The purpose of the feasibility is not to solve the problem but to determine if the problem is worth solving. Performance is defined by the identification of specific system objects and descriptive of output. There are following types of interrelated feasibility. They are as follows:

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

3.1 TECHNICAL FEASIBILITY

In examining technical feasibility configuration ,the system is given more importance than the actual make of hardware. According to the definition of technical feasibility the compatibility between frontend and backend is very important. In our project the compatibility of both is very good. The degree of compatibility of JSP and MYSQL is very good. The speed of output is very good when we enter the data and click button. The response time is very fast and give result very quickly.

We chose JAVA because it provides following features and we thought it would make our project more efficient.

- It gives optimized performance.
- Portable
- User-friendly
- Sturdy Garbage Collection
- Native threads

3.2 ECONOMIC FEASIBILITY

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. Economic analysis is the procedure to determine the benefits and saving that are effected from a proposed system and compare them with cost. If benefits outweighs cost, decision is taken to design the system. Otherwise, further justification or alternative in the proposed system will have to be made. Total cost of the proposed system is very cheap and therefore the organization will not find any difficulty at the installation.

Economic feasibility is the determination of the resources that are:

- Time Management
- Cost of doing full system study
- Estimated cost of hardware
- Estimated cost of software
- Estimated cost of software development

3.3 OPERATIONAL FEASIBILITY

Operational feasibility is the problem or acceptability of the solution. There are two aspects of the operational feasibility.

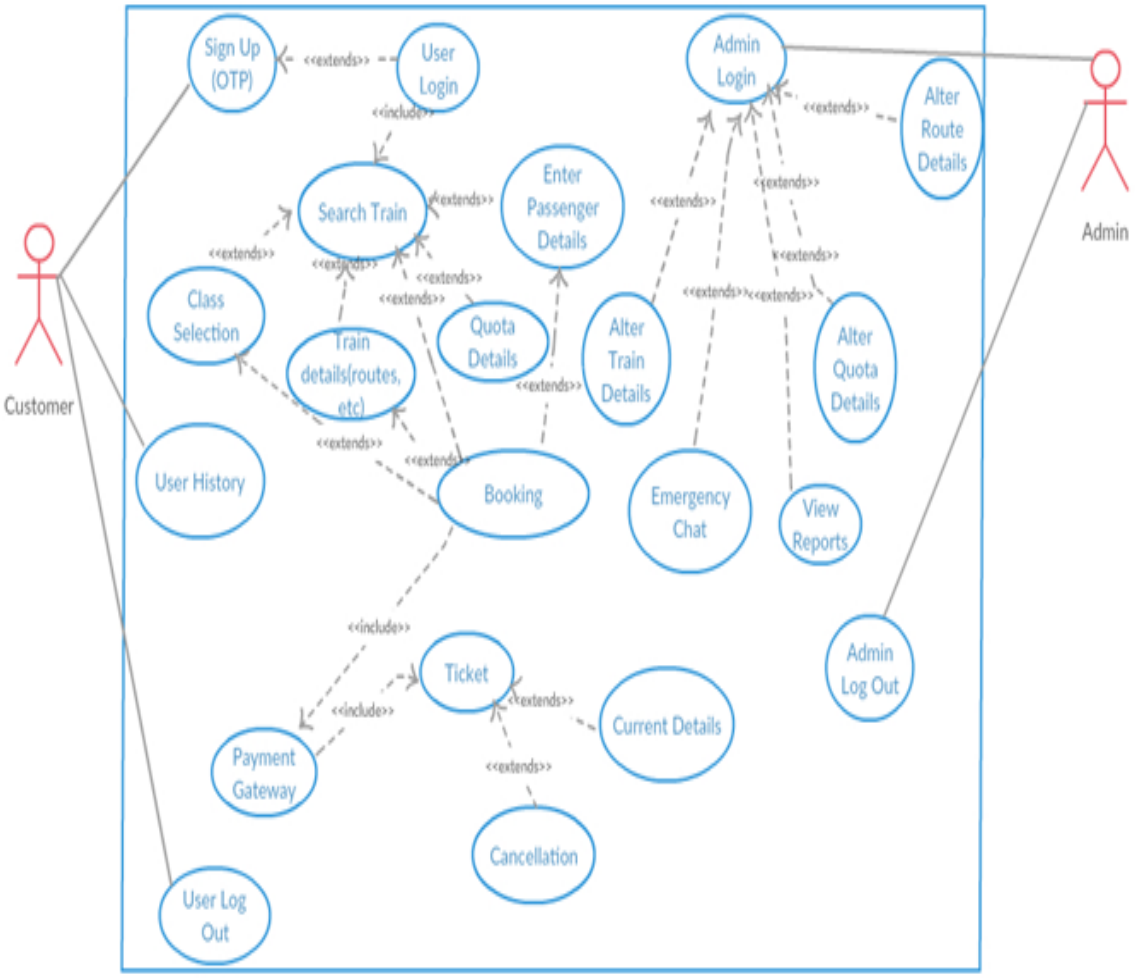
- If the problem is worth solving
- What to the end users and the management feel about the problem or solution.

The following points are considered for the above problems:

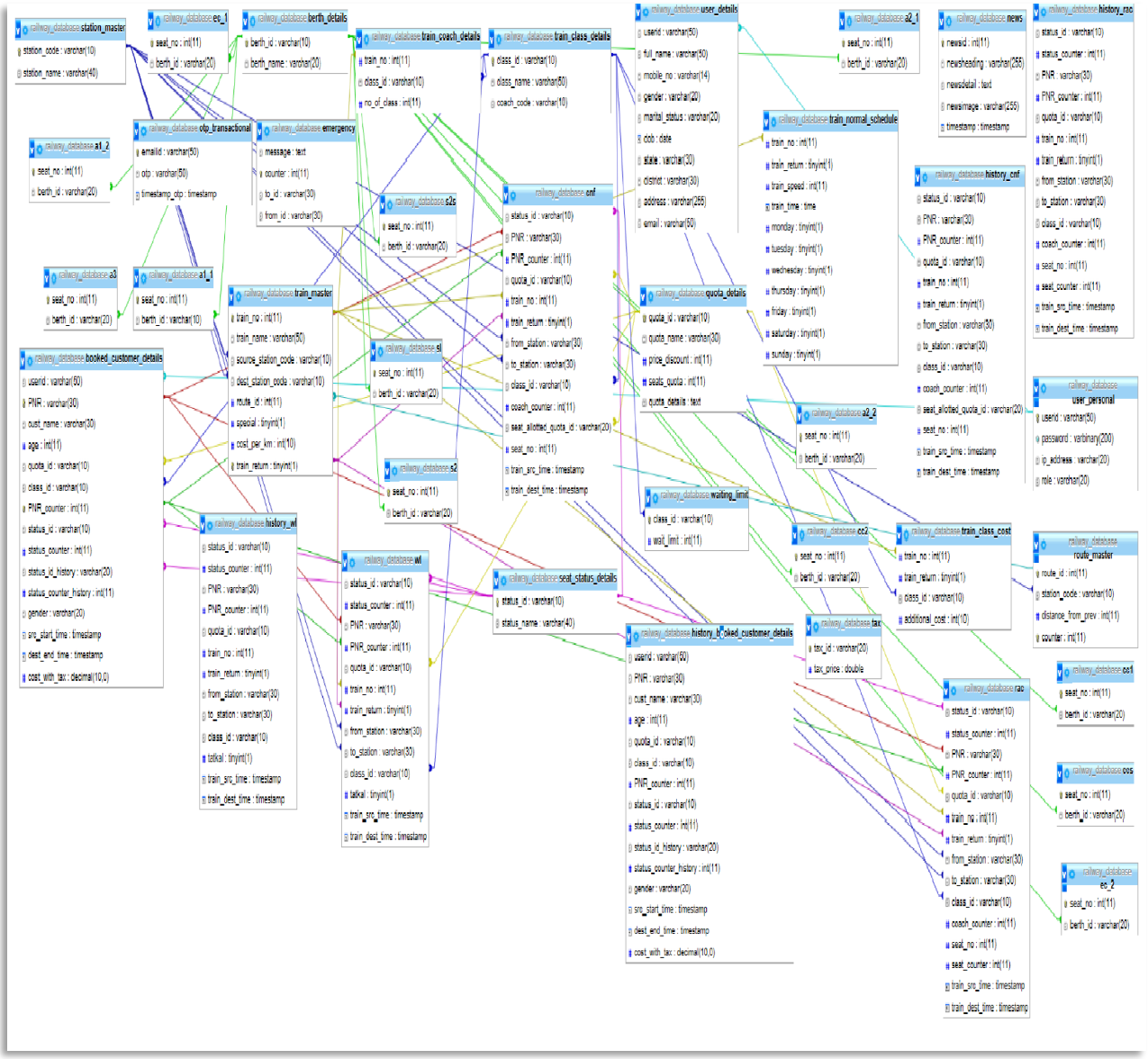
- Performance
- Information
- Economy
- Control
- Efficiency
- Services

SYSTEM ANALYSIS

4.1 USE CASE DIAGRAM

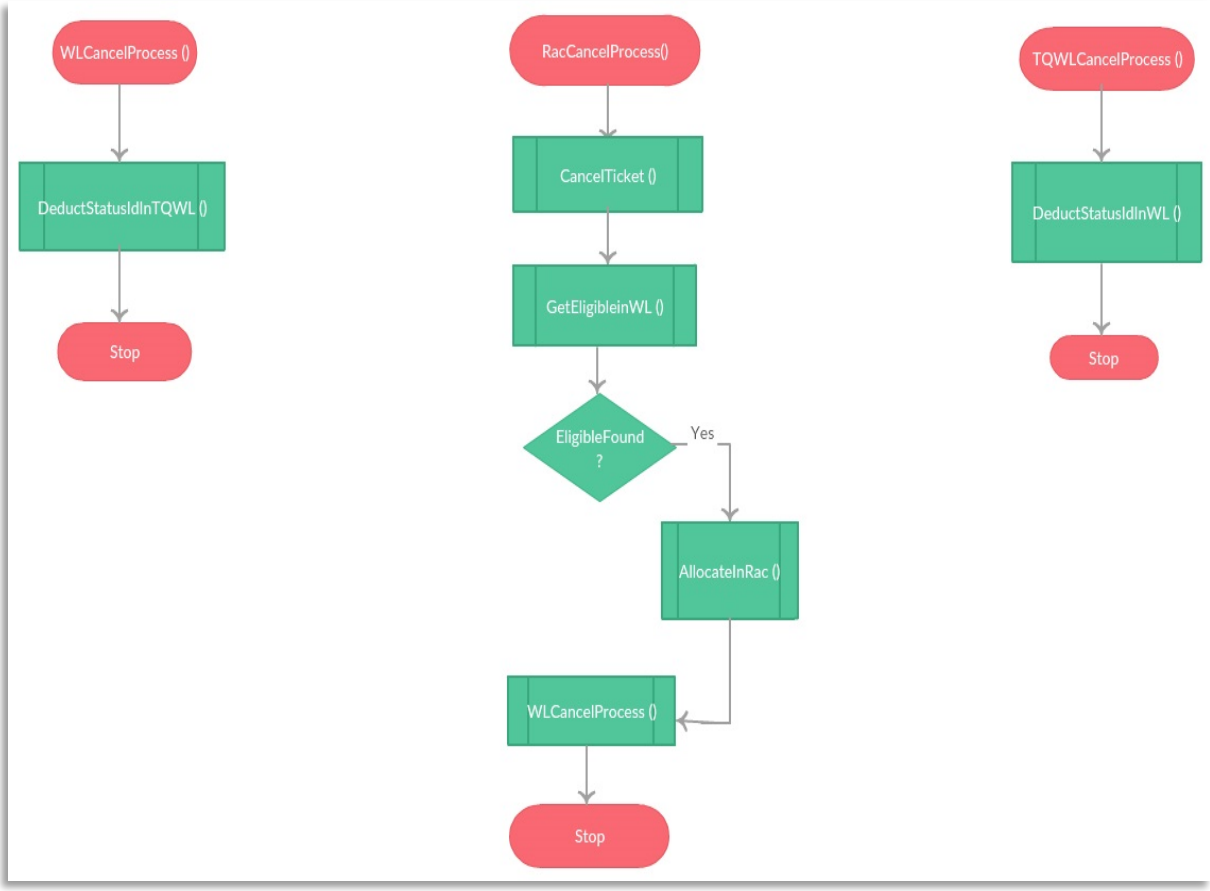


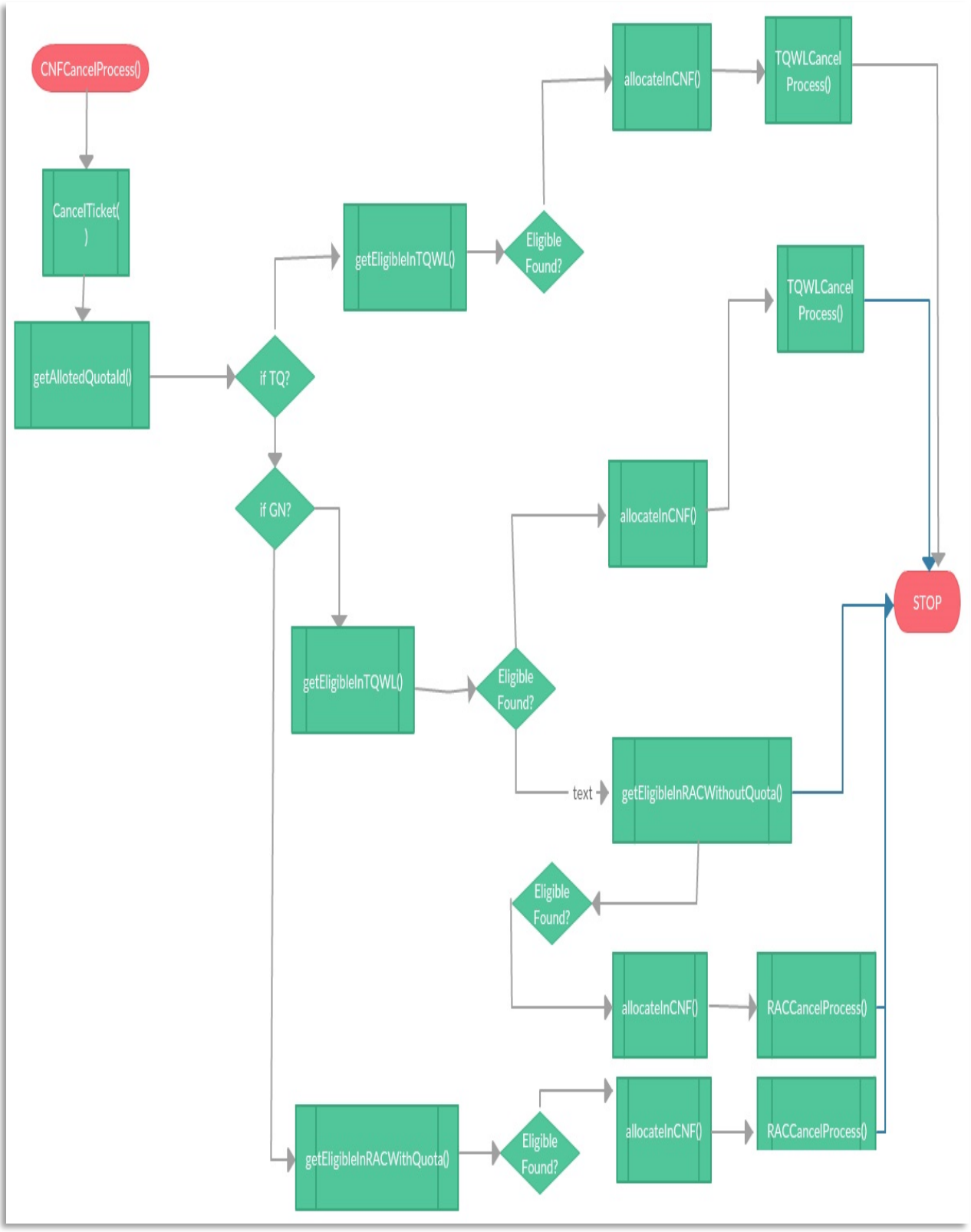
4.2 ER DIAGRAM [EXCLUDING 11 TABLES OF ANALYSIS]



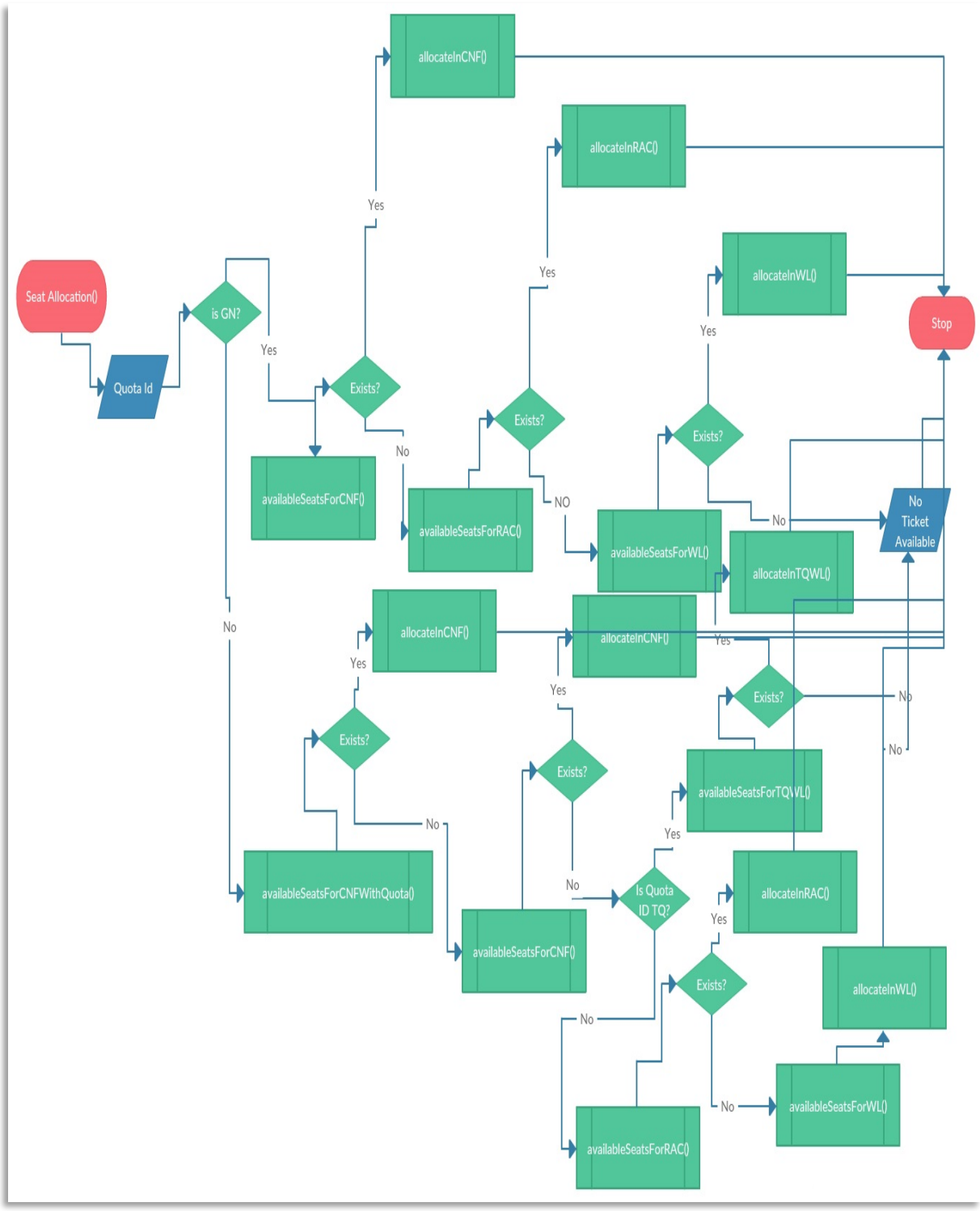
4.3 FLOWCHART OF ALGORITHMS

Cancellation flowchart :

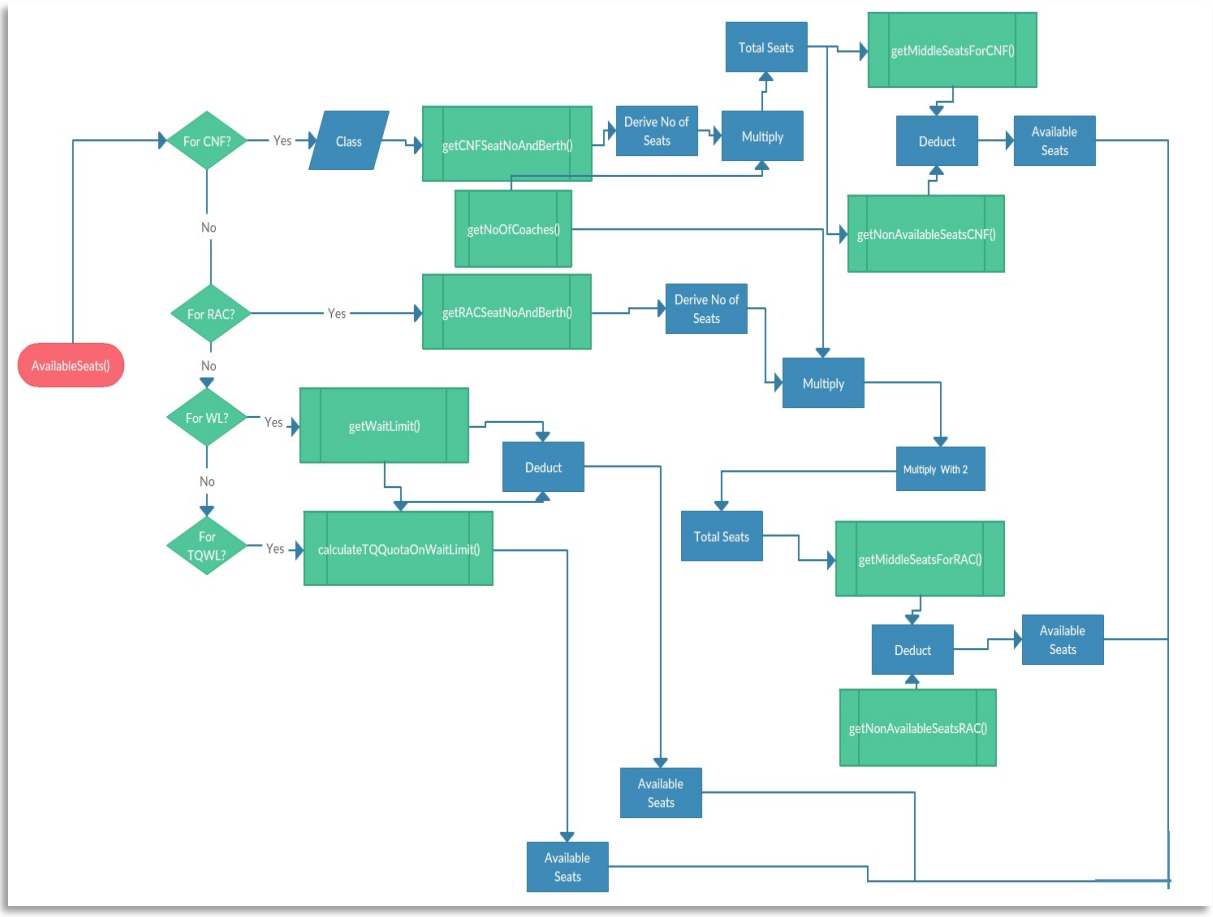


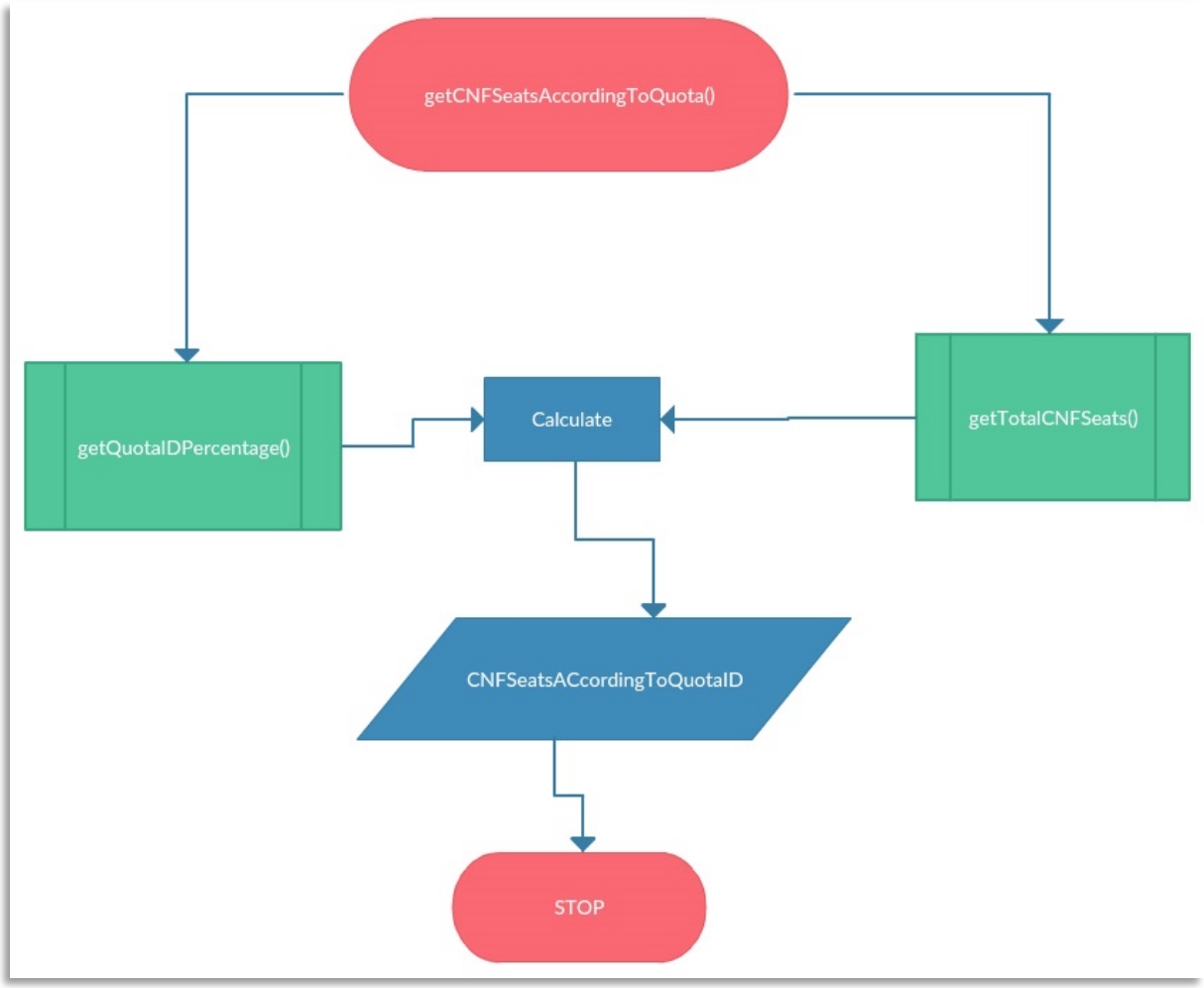

CNFCancelProcess()
CancelTicket( )
getAllotedQuotaId()
if TQ?
getEligibleInTQWL()
Eligible Found?
allocateInCNF()
TQWLCancel Process()
if GN?
getEligibleInTQWL()
Eligible Found?
allocateInCNF()
TQWLCancel Process()
STOP
text
getEligibleInRACWithoutQuota()
Eligible Found?
allocateInCNF()
RACCancelProcess()
getEligibleInRACWithQuota()
Eligible Found?
allocateInCNF()
RACCancelProcess()

SeatAllocation Flowchart :



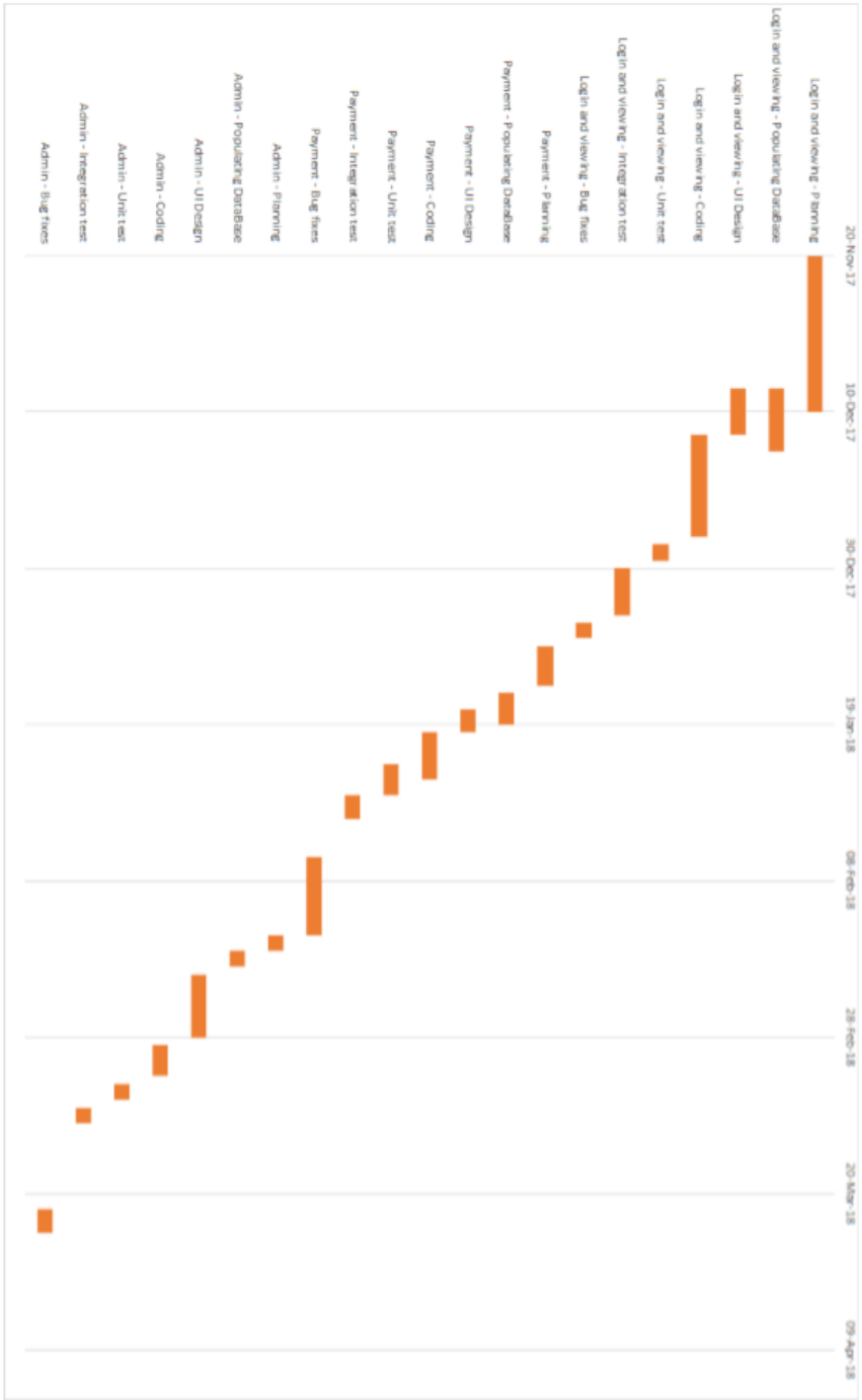
Seat Availability flowchart :




getCNFSeatsAccordingToQuota()
getQuotaIDPercentage()
Calculate
getTotalCNFSeats()
CNFSeatsACcordingToQuotaID
STOP

4.4 GANTT CHART

1.6 Gantt chart:



SYSTEM CODING

5.1 LIST OF TABLES

Tables are the collection of rows and fields with appropriate data inside the database.

There are about 50 tables used for this project. Every table has some valuable attributes and corresponding constrains.

Below are the Screenshots of some of the tables used for User's module:

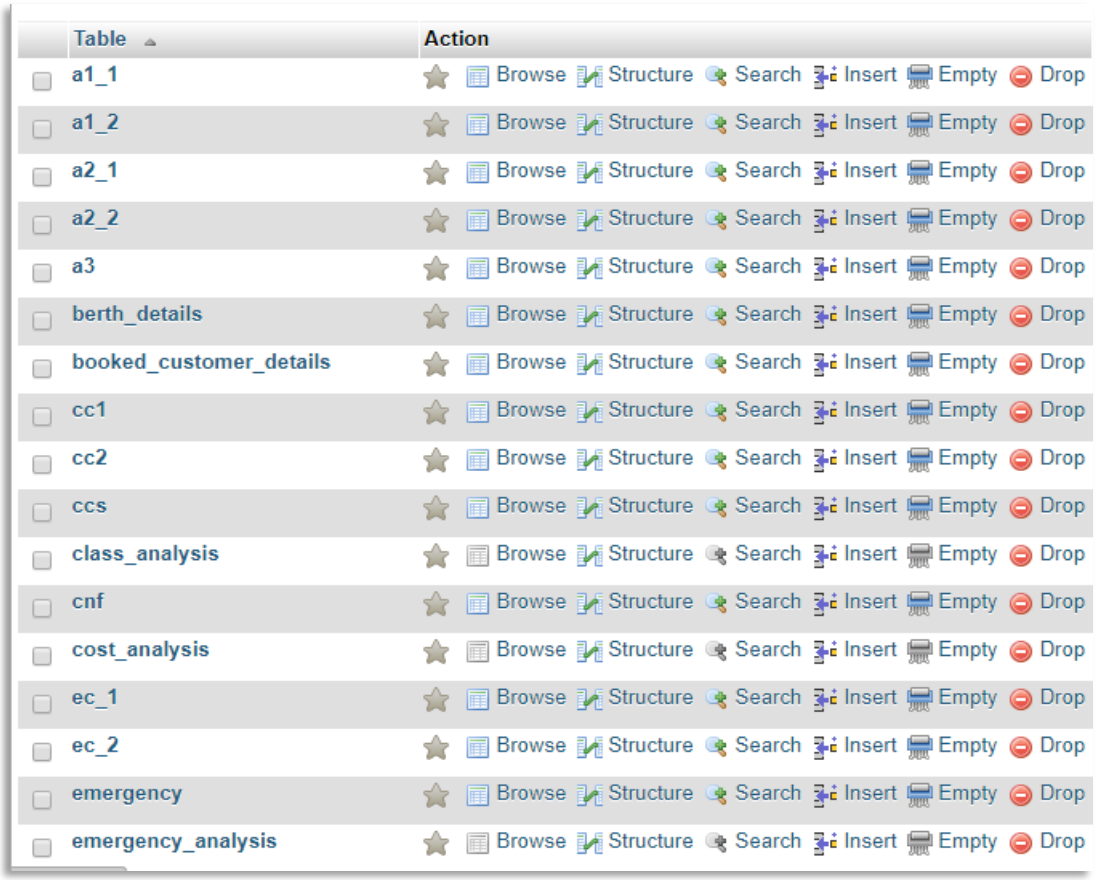
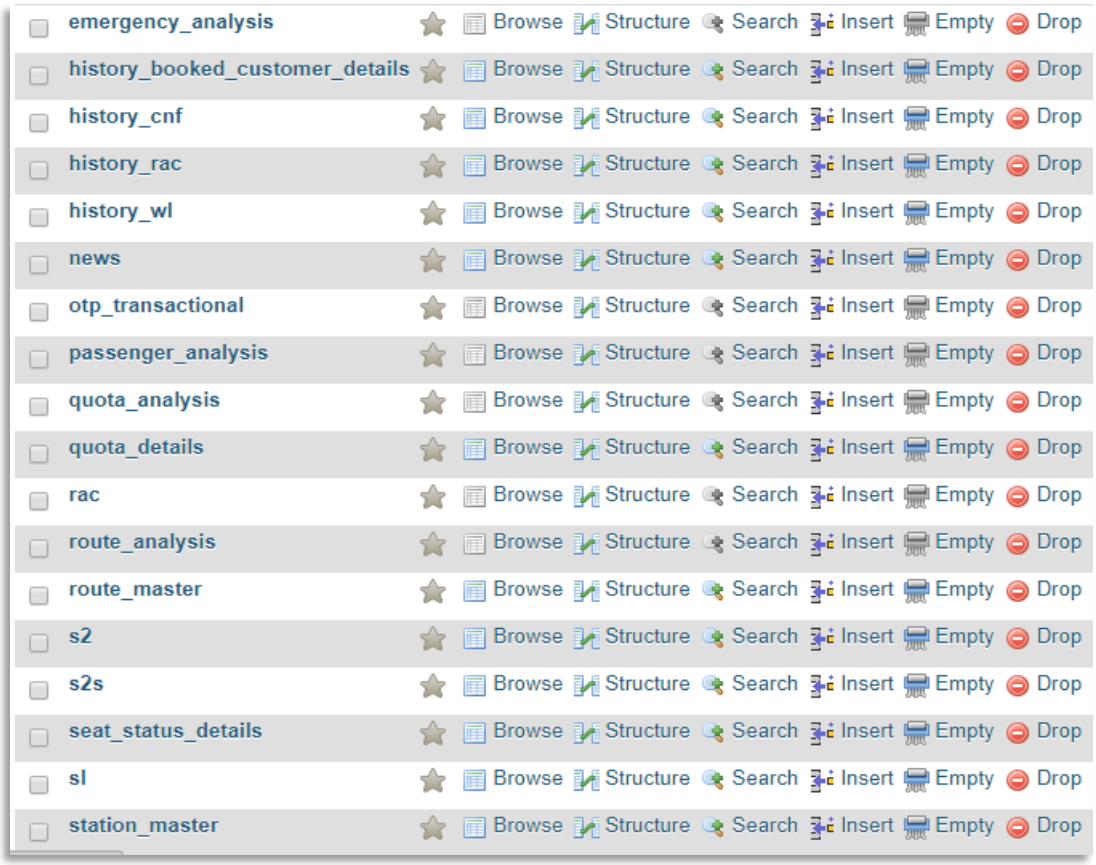


Table	Action
a1_1	Browse Structure Search Insert Empty Drop
a1_2	Browse Structure Search Insert Empty Drop
a2_1	Browse Structure Search Insert Empty Drop
a2_2	Browse Structure Search Insert Empty Drop
a3	Browse Structure Search Insert Empty Drop
berth_details	Browse Structure Search Insert Empty Drop
booked_customer_details	Browse Structure Search Insert Empty Drop
cc1	Browse Structure Search Insert Empty Drop
cc2	Browse Structure Search Insert Empty Drop
ccs	Browse Structure Search Insert Empty Drop
class_analysis	Browse Structure Search Insert Empty Drop
cnf	Browse Structure Search Insert Empty Drop
cost_analysis	Browse Structure Search Insert Empty Drop
ec_1	Browse Structure Search Insert Empty Drop
ec_2	Browse Structure Search Insert Empty Drop
emergency	Browse Structure Search Insert Empty Drop
emergency_analysis	Browse Structure Search Insert Empty Drop

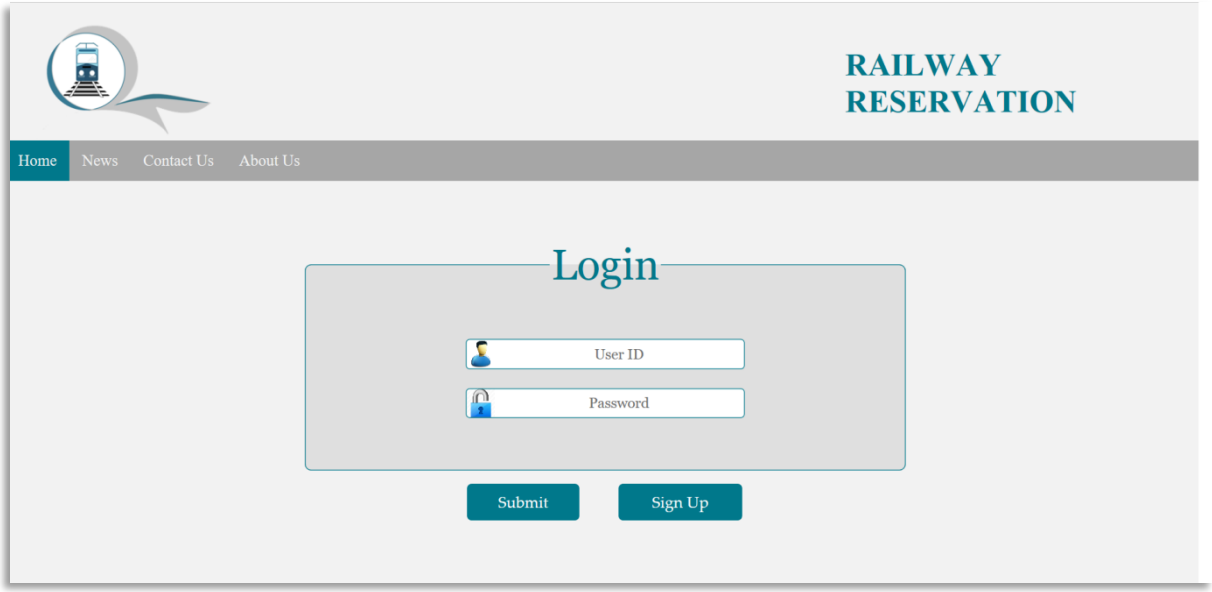


emergency_analysis	Browse Structure Search Insert Empty Drop
history_booked_customer_details	Browse Structure Search Insert Empty Drop
history_cnf	Browse Structure Search Insert Empty Drop
history_rac	Browse Structure Search Insert Empty Drop
history_wl	Browse Structure Search Insert Empty Drop
news	Browse Structure Search Insert Empty Drop
otp_transactional	Browse Structure Search Insert Empty Drop
passenger_analysis	Browse Structure Search Insert Empty Drop
quota_analysis	Browse Structure Search Insert Empty Drop
quota_details	Browse Structure Search Insert Empty Drop
rac	Browse Structure Search Insert Empty Drop
route_analysis	Browse Structure Search Insert Empty Drop
route_master	Browse Structure Search Insert Empty Drop
s2	Browse Structure Search Insert Empty Drop
s2s	Browse Structure Search Insert Empty Drop
seat_status_details	Browse Structure Search Insert Empty Drop
sl	Browse Structure Search Insert Empty Drop
station_master	Browse Structure Search Insert Empty Drop

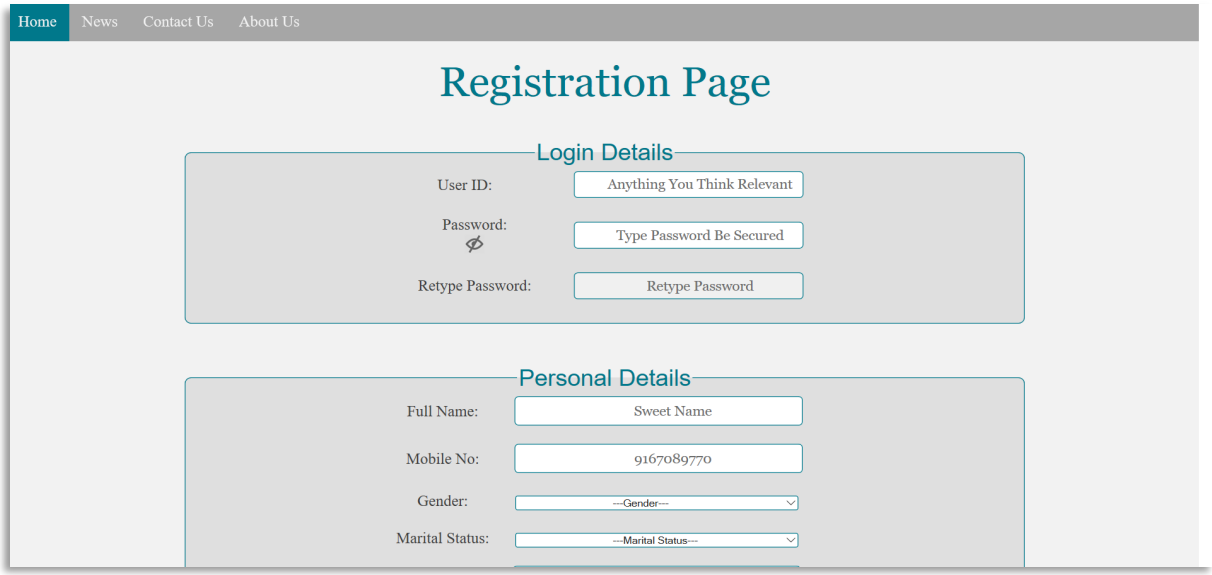

tax Browse Structure Search Insert Empty Drop
train_analysis Browse Structure Search Insert Empty Drop
train_class_cost Browse Structure Search Insert Empty Drop
train_class_details Browse Structure Search Insert Empty Drop
train_coach_details Browse Structure Search Insert Empty Drop
train_master Browse Structure Search Insert Empty Drop
train_normal_schedule Browse Structure Search Insert Empty Drop
user_analysis Browse Structure Search Insert Empty Drop
user_details Browse Structure Search Insert Empty Drop
user_personal Browse Structure Search Insert Empty Drop
waiting_limit Browse Structure Search Insert Empty Drop
wl Browse Structure Search Insert Empty Drop
zone Browse Structure Search Insert Empty Drop
zone_analysis Browse Structure Search Insert Empty Drop
zone_train_details Browse Structure Search Insert Empty Drop
49 tables Sum

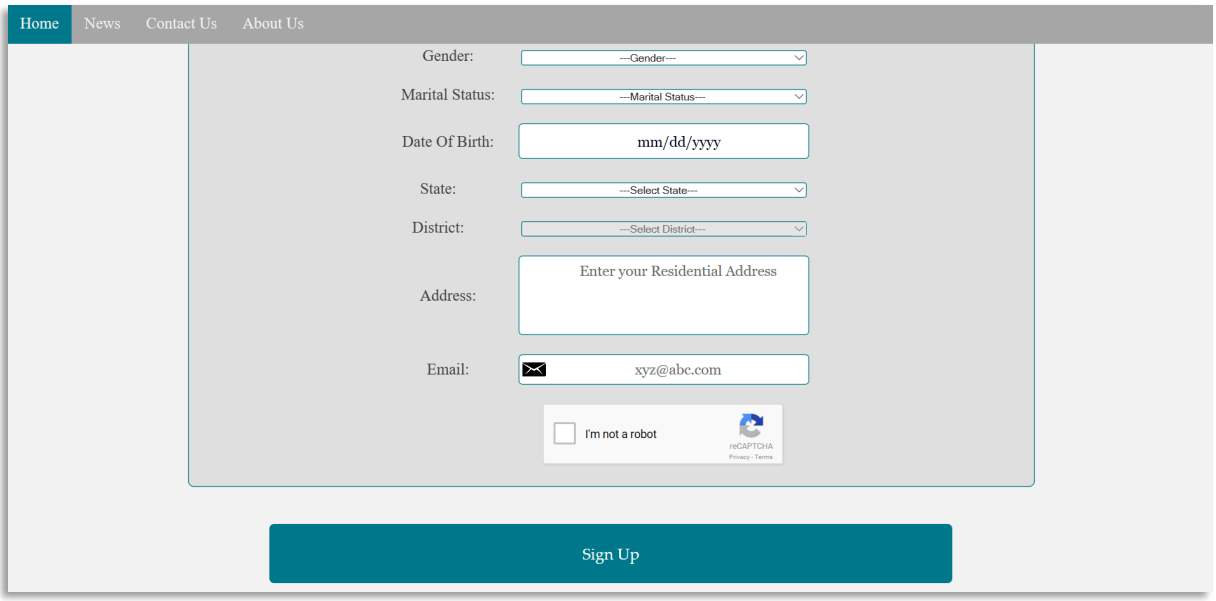
5.2 SCREEN SHOTS

1. INDEX PAGE

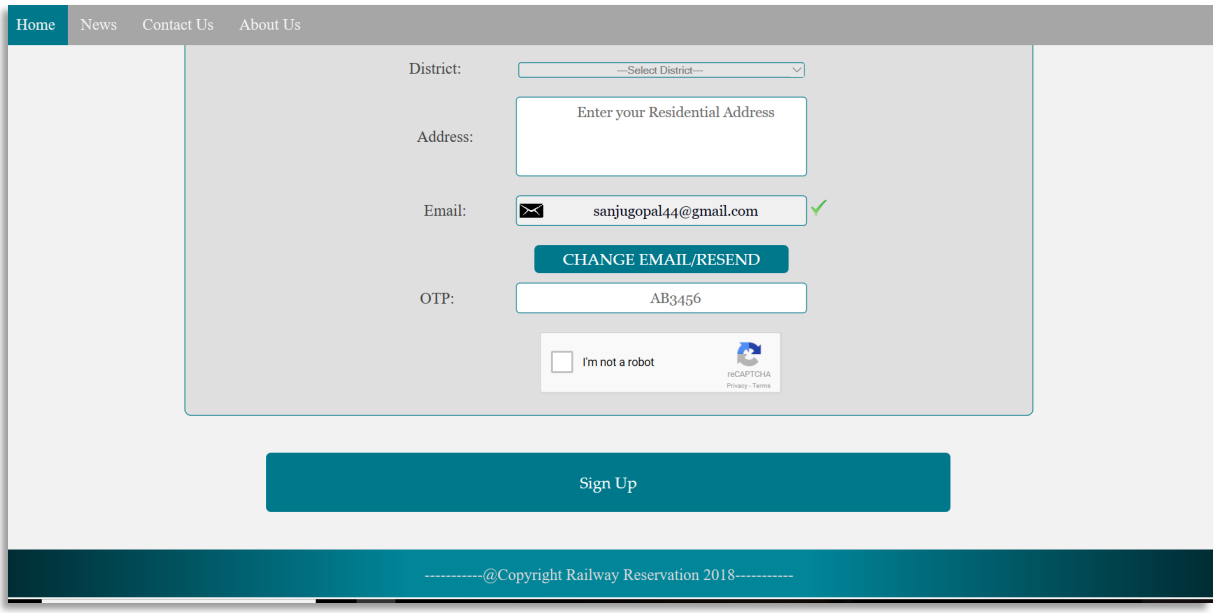


2. SIGNUP PAGE

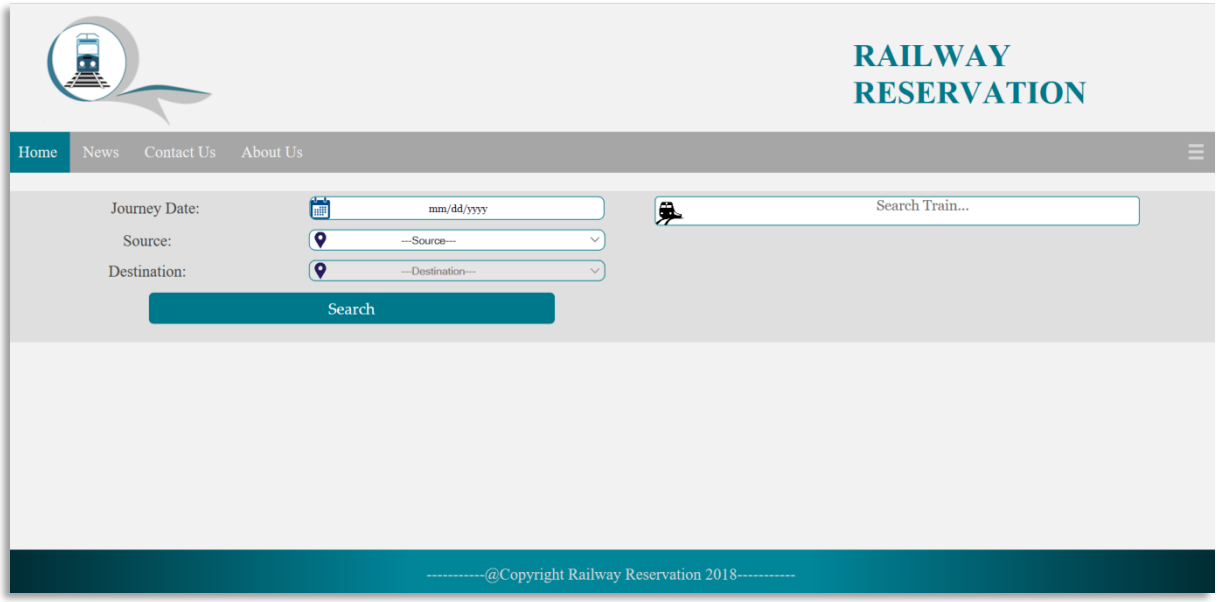


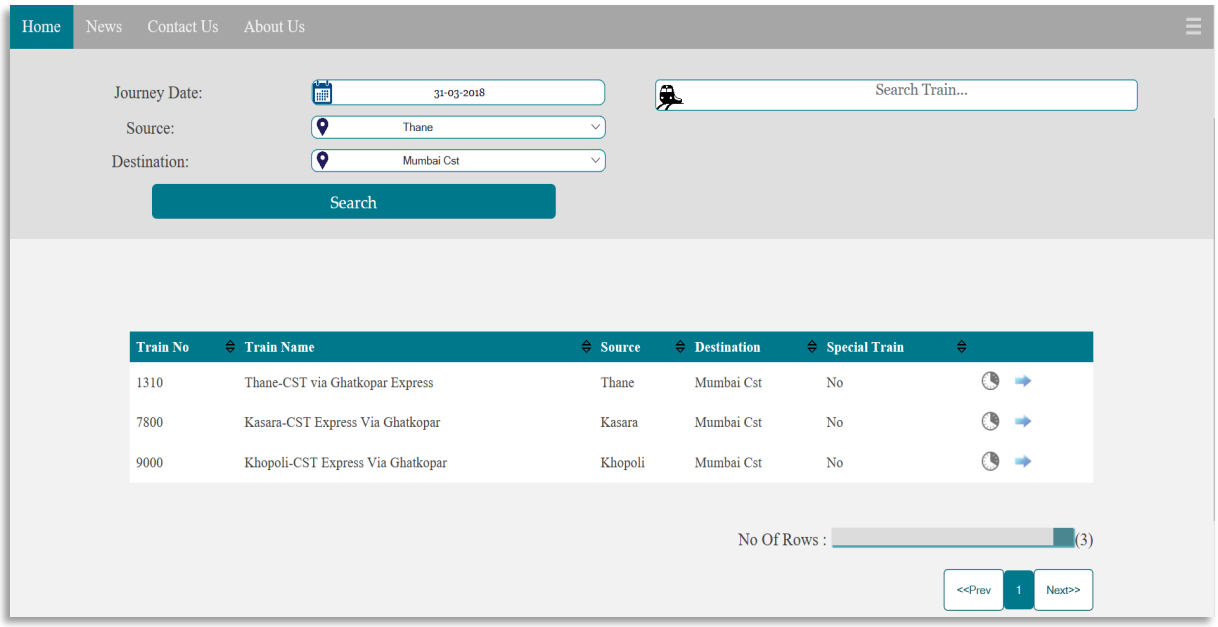


OTP in Signup Page:

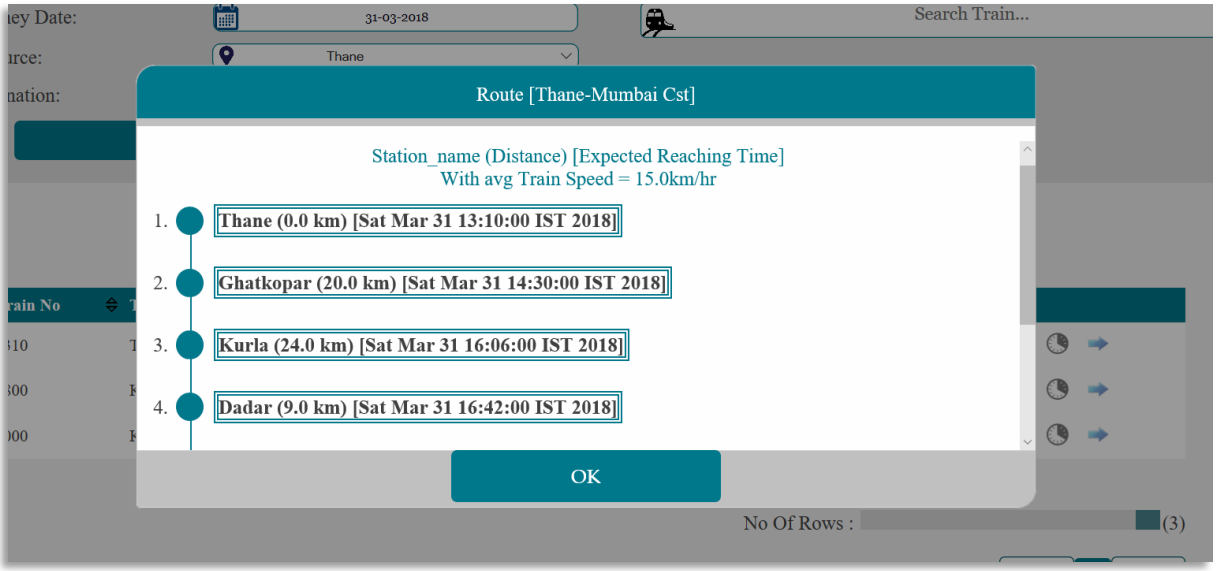


3. VIEWDETAILS :



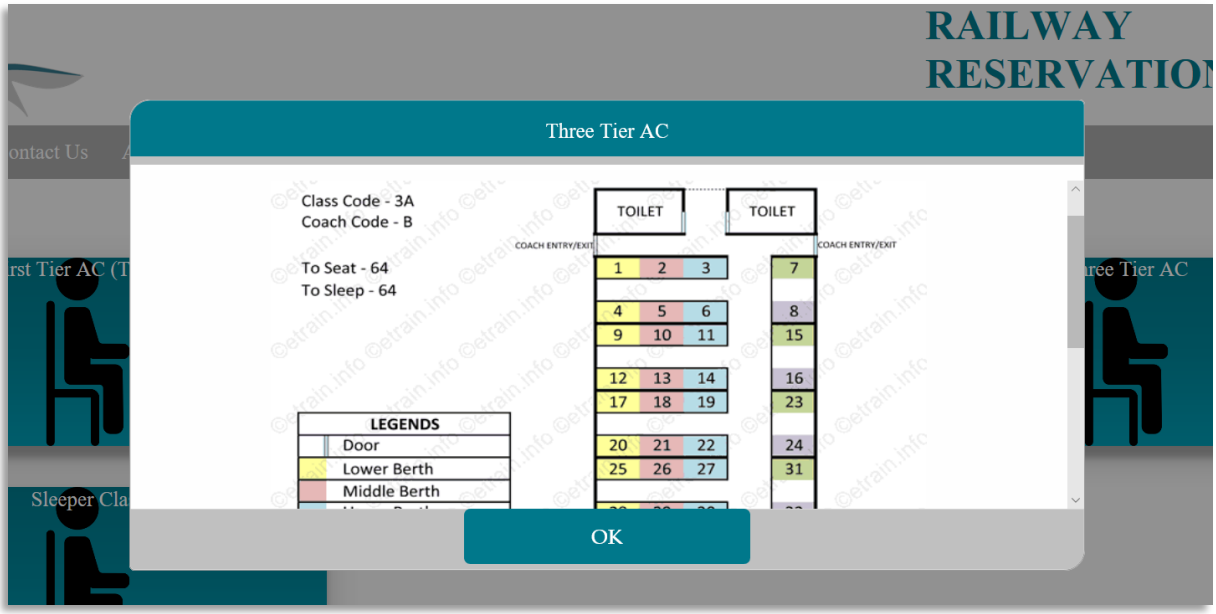


Display of timing expected to reach destination from source provided.



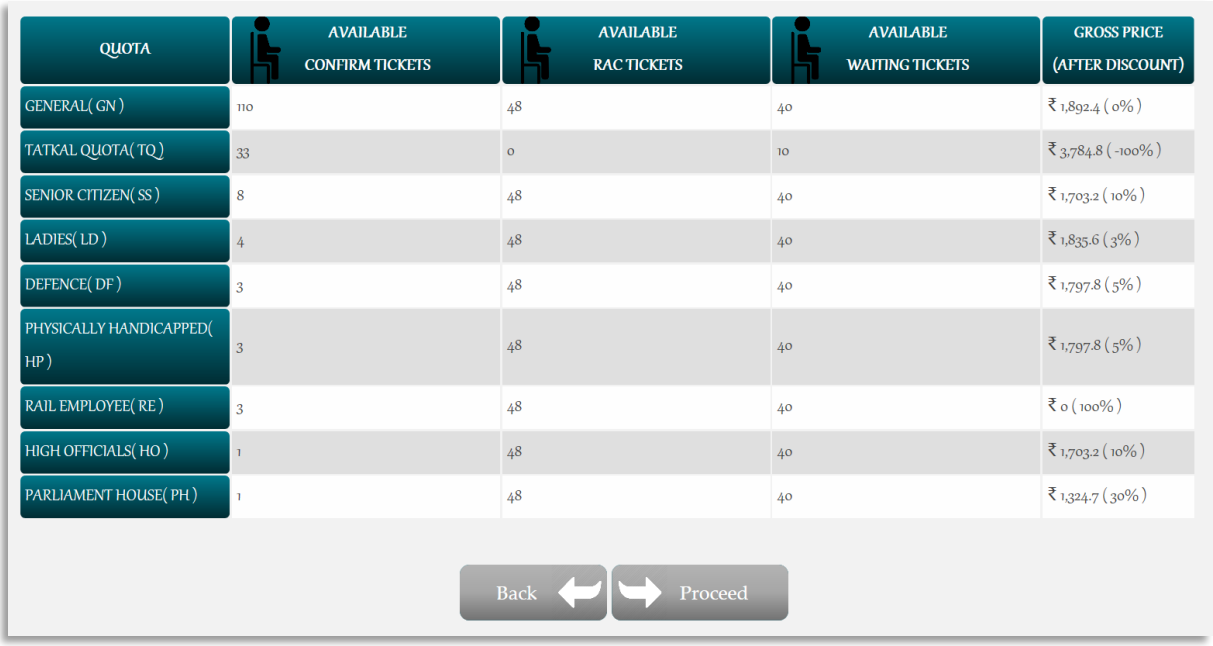
4. CLASS SELECTION:

On click of Seat Map:



5. QUOTA_DETAILS:

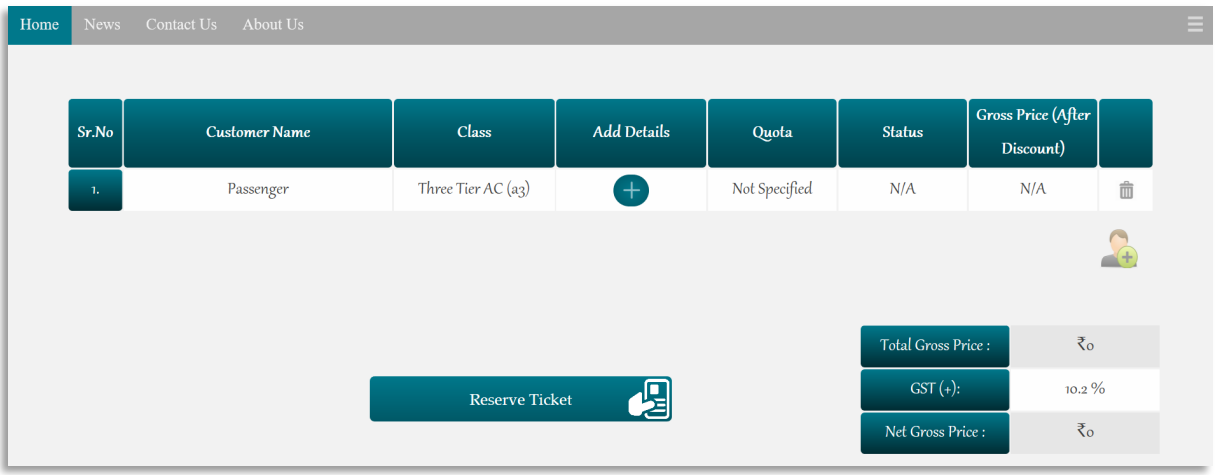
6. SEAT_AVAILABILITY WITH CORRESOPONDING COST AND QUOTA :



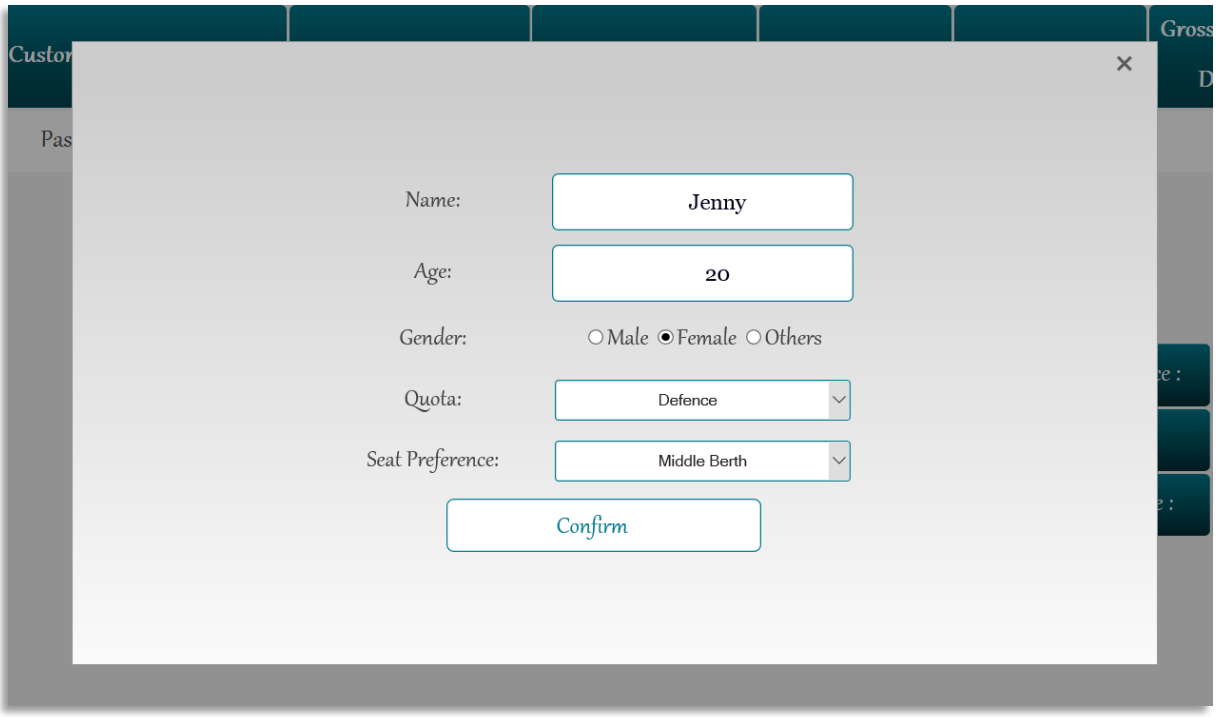
QUOTA	AVAILABLE CONFIRM TICKETS	AVAILABLE RAC TICKETS	AVAILABLE WAITING TICKETS	GROSS PRICE (AFTER DISCOUNT)
GENERAL(GN)	110	48	40	₹ 1,892.4 (0%)
TATKAL QUOTA(TQ)	33	0	10	₹ 3,784.8 (-100%)
SENIOR CITIZEN(SS)	8	48	40	₹ 1,703.2 (10%)
LADIES(LD)	4	48	40	₹ 1,835.6 (3%)
DEFENCE(DF)	3	48	40	₹ 1,797.8 (5%)
PHYSICALLY HANDICAPPED(HP)	3	48	40	₹ 1,797.8 (5%)
RAIL EMPLOYEE(RE)	3	48	40	₹ 0 (100%)
HIGH OFFICIALS(HO)	1	48	40	₹ 1,703.2 (10%)
PARLIAMENT HOUSE(PH)	1	48	40	₹ 1,324.7 (30%)

Back Proceed

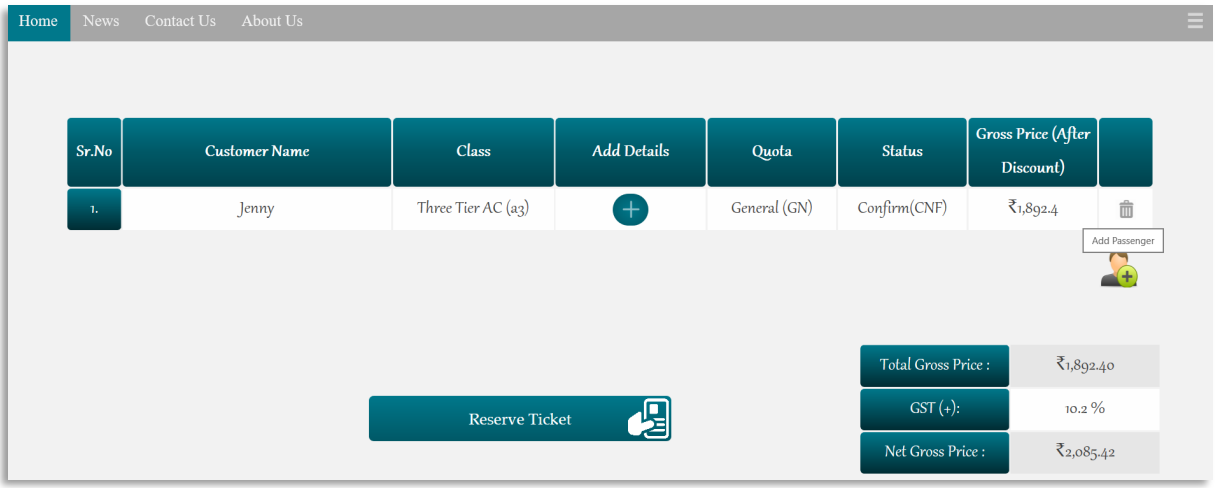
7. BOOKING :



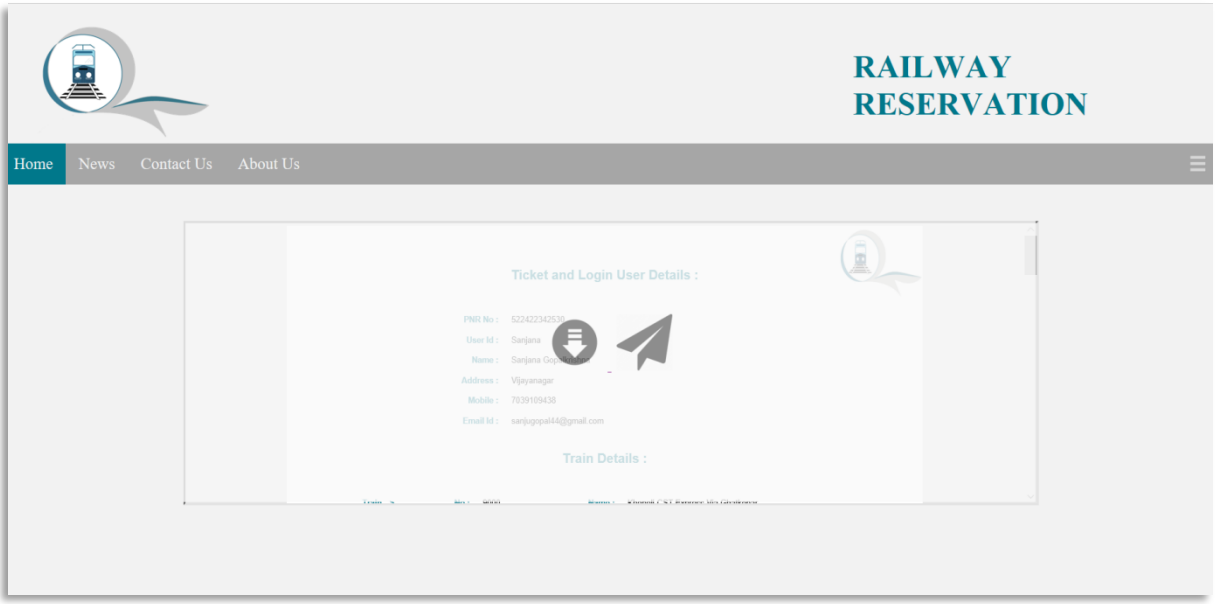
ADDING NUMBER OF PASSENGERS:



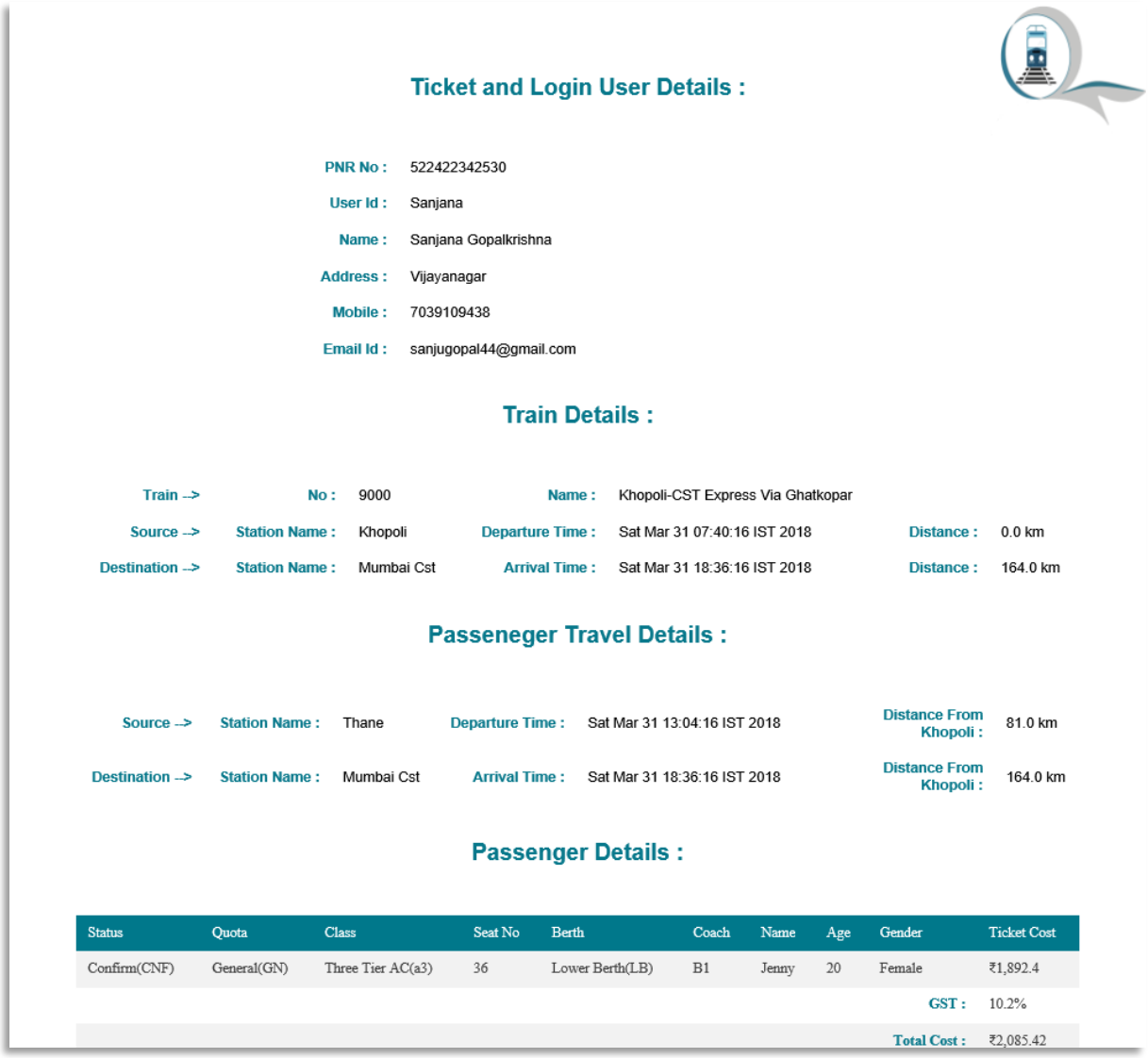
Display of number of passengers and cost :



8. **BOOKED :** Ticket display which can be downloaded or send to your mail account.



9. TICKET :



Ticket and Login User Details :

PNR No : 522422342530
User Id : Sanjana
Name : Sanjana Gopalkrishna
Address : Vijayanagar
Mobile : 7039109438
Email Id : sanjugopal44@gmail.com

Train Details :

Train -->	**No :**	9000	**Name :**	Khopoli-CST Express Via Ghatkopar		
Source -->	**Station Name :**	Khopoli	**Departure Time :**	Sat Mar 31 07:40:16 IST 2018	**Distance :**	0.0 km
Destination -->	**Station Name :**	Mumbai Cst	**Arrival Time :**	Sat Mar 31 18:36:16 IST 2018	**Distance :**	164.0 km

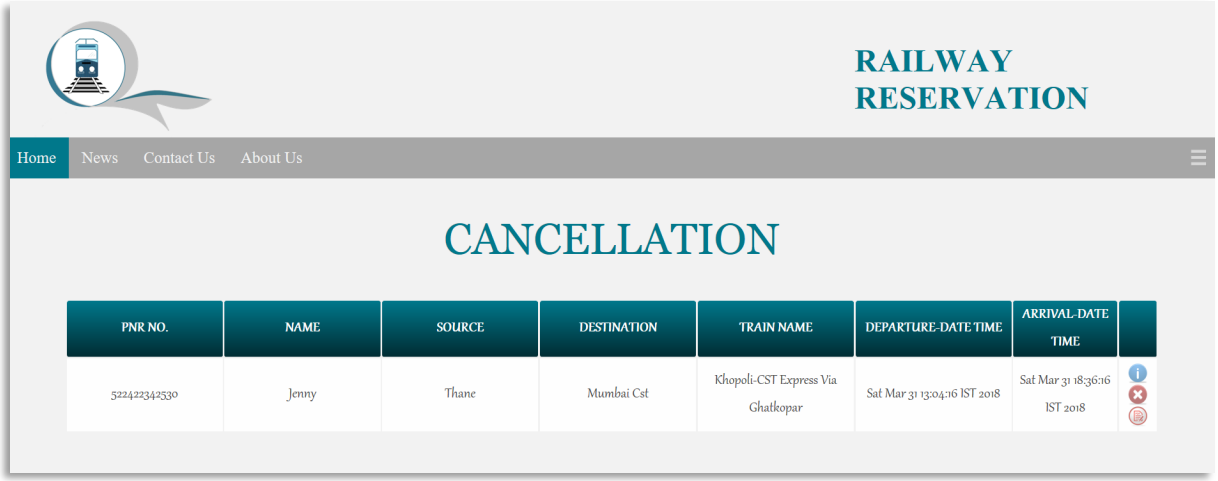
Passeneger Travel Details :

Source -->	**Station Name :**	Thane	**Departure Time :**	Sat Mar 31 13:04:16 IST 2018	**Distance From Khopoli :**	81.0 km
Destination -->	**Station Name :**	Mumbai Cst	**Arrival Time :**	Sat Mar 31 18:36:16 IST 2018	**Distance From Khopoli :**	164.0 km

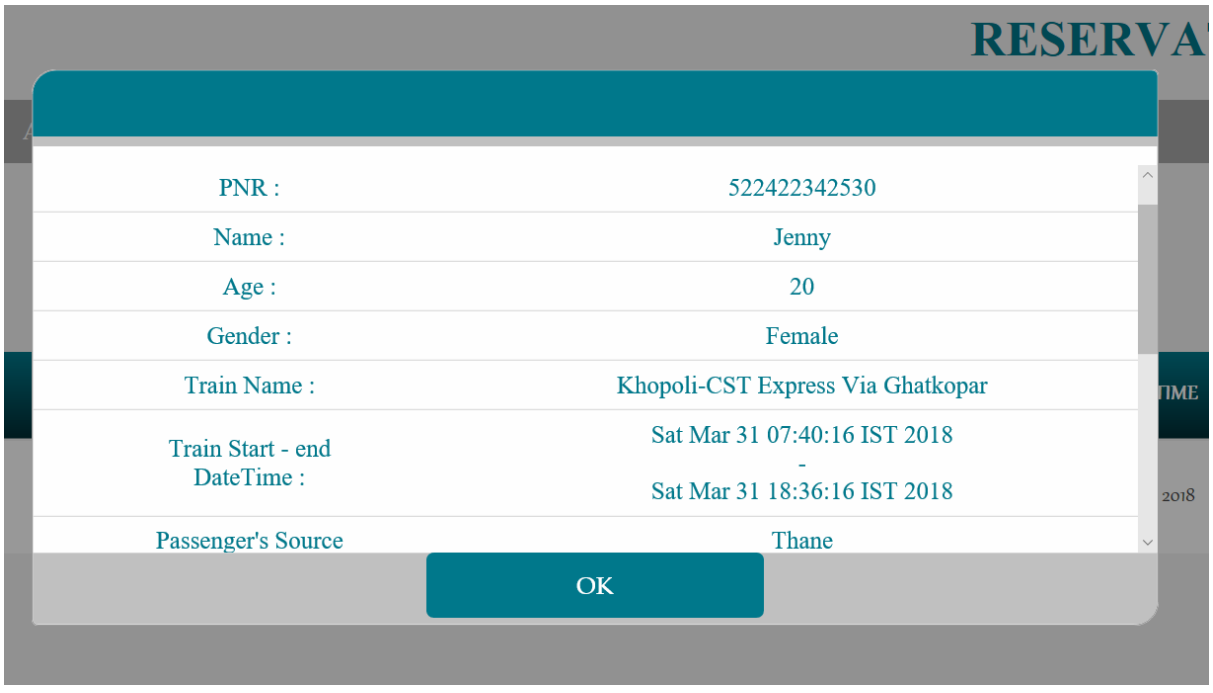
Passenger Details :

Status	Quota	Class	Seat No	Berth	Coach	Name	Age	Gender	Ticket Cost
Confirm(CNF)	General(GN)	Three Tier AC(a3)	36	Lower Berth(LB)	B1	Jenny	20	Female	₹1,892.4
								GST :	10.2%
								Total Cost :	₹2,085.42

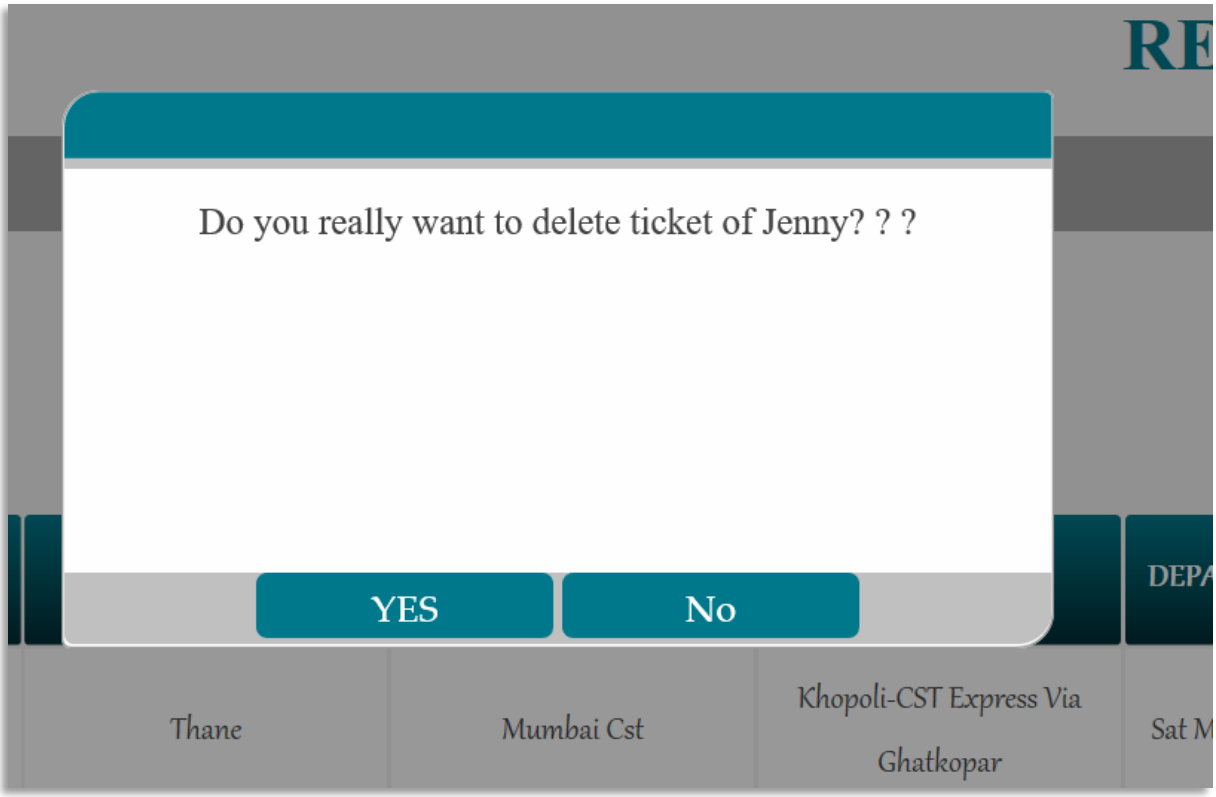
10. CANCELLATION OF TICKET :



On Clicking Show-details Icon: All the information is displayed regarding passengers travelling.

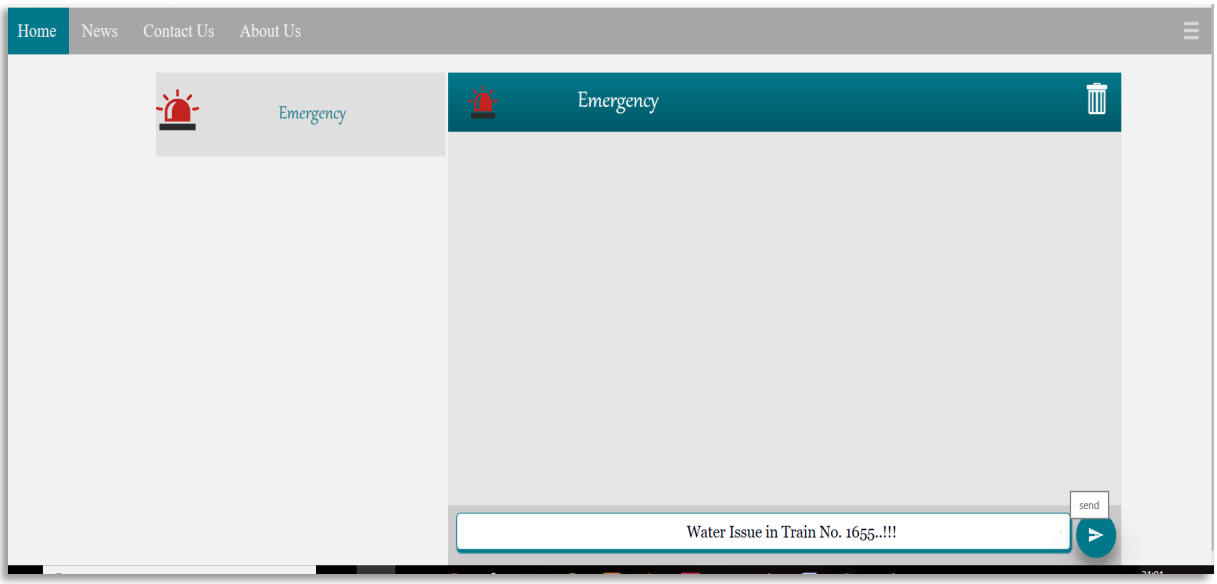


On Clicking Cancel button: By clicking Yes, the ticket gets cancelled.

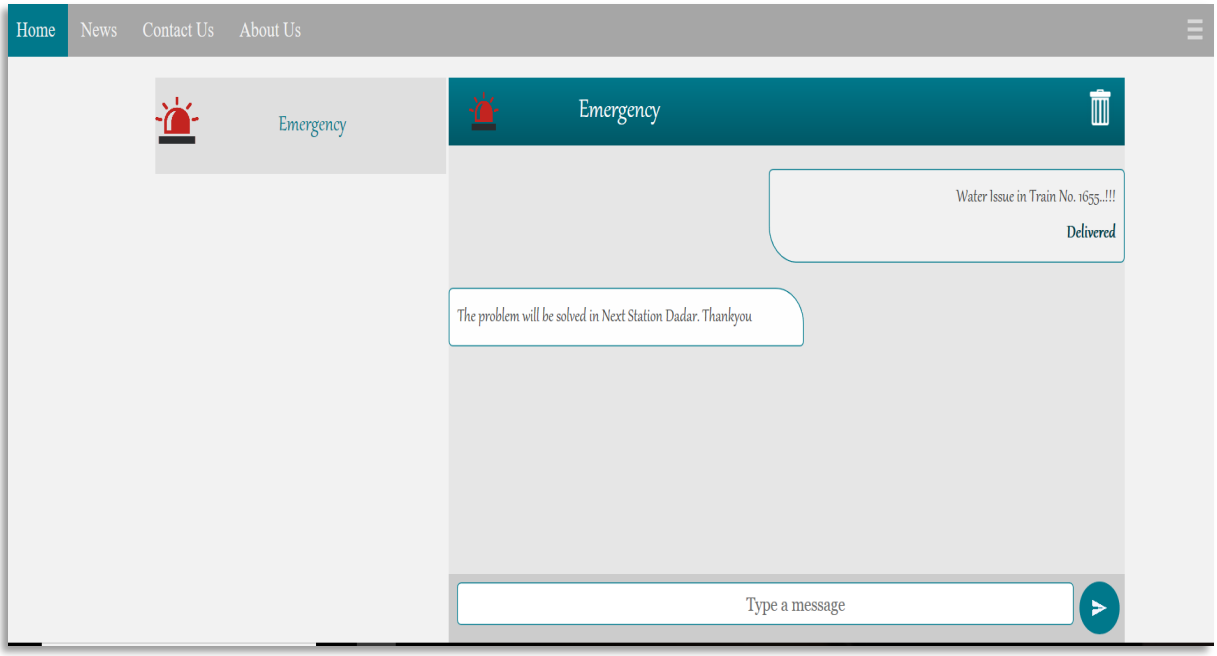


11. HISTORY OF USER :

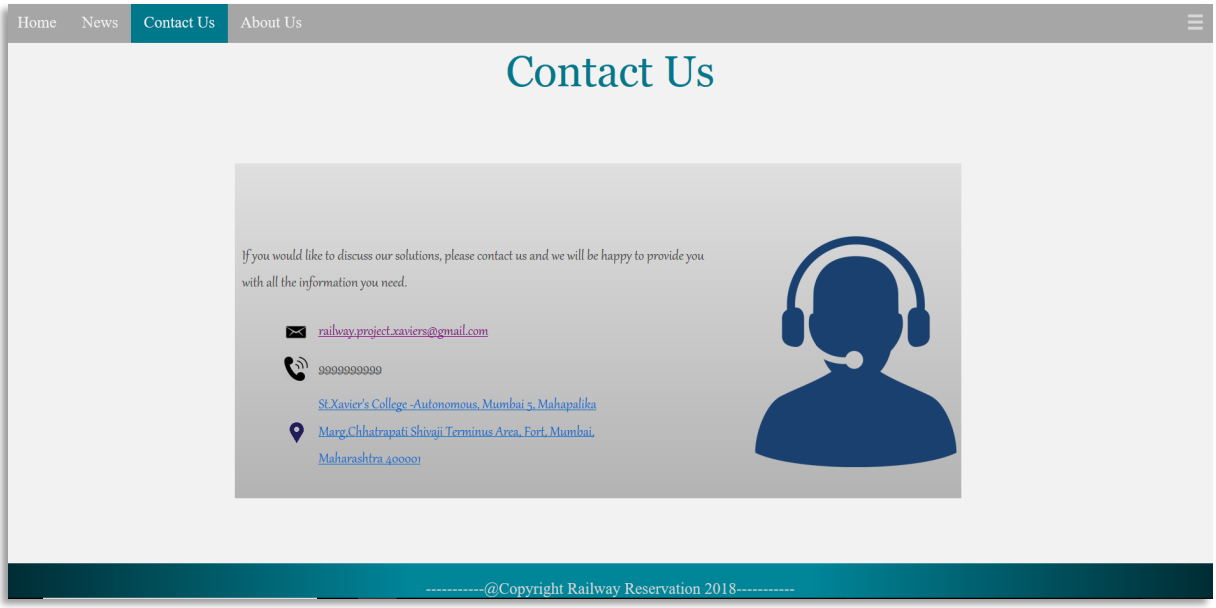
12. EMERGENCY CHAT BOARD :



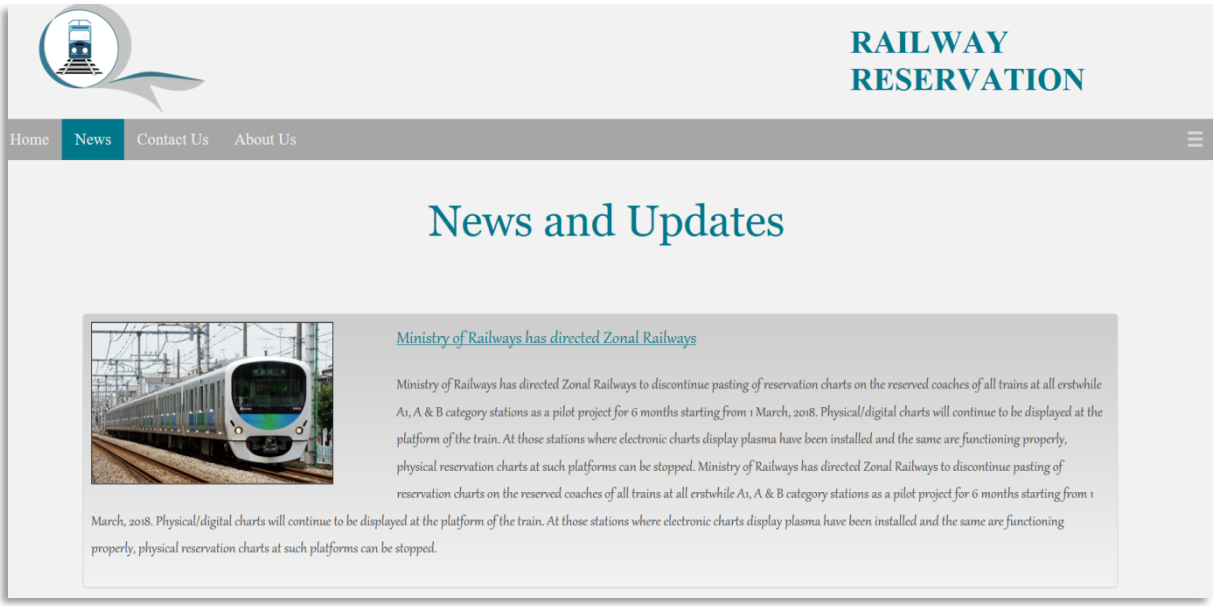
After Admin responding to the user :



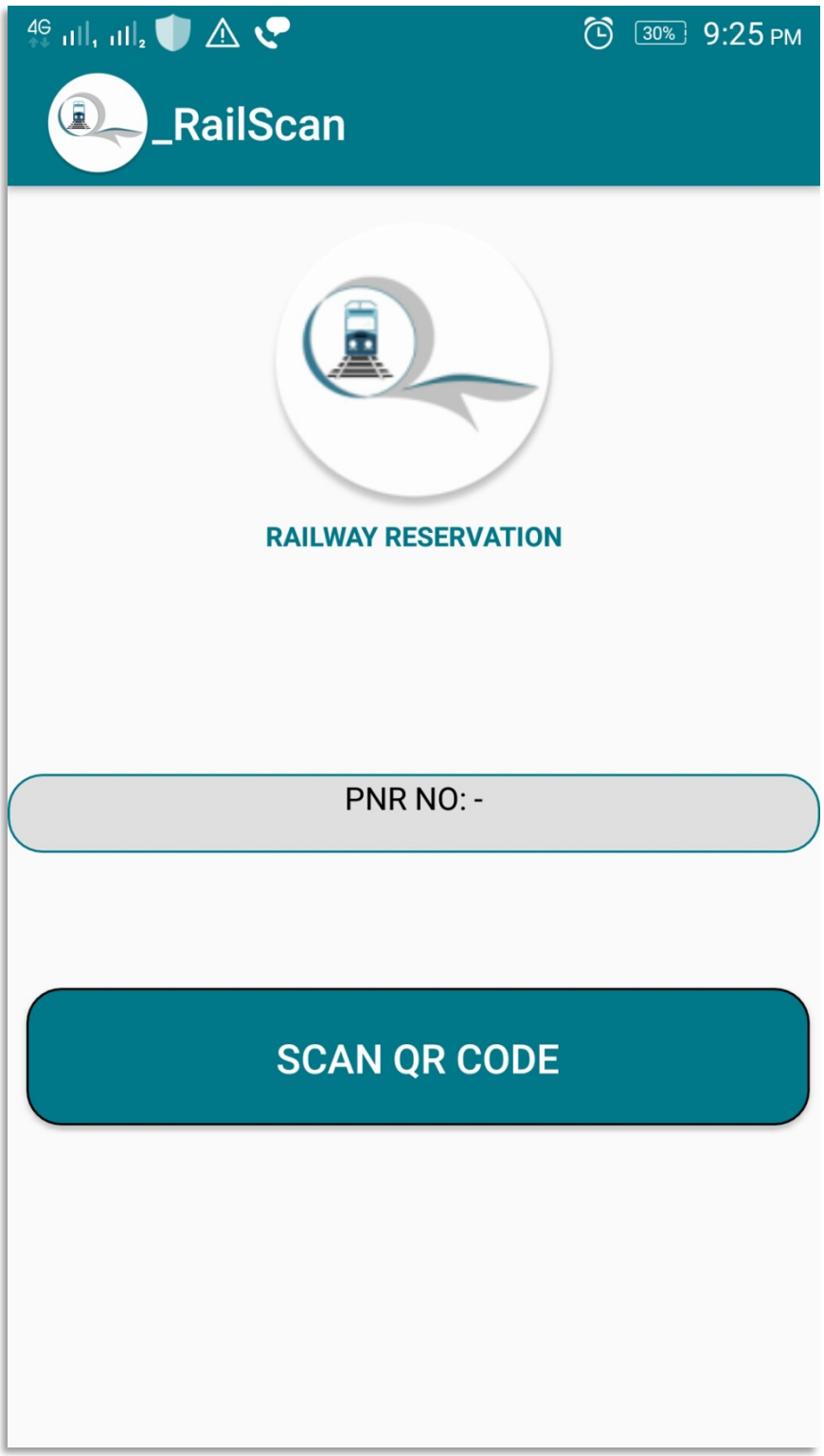
13. CONTACT US :



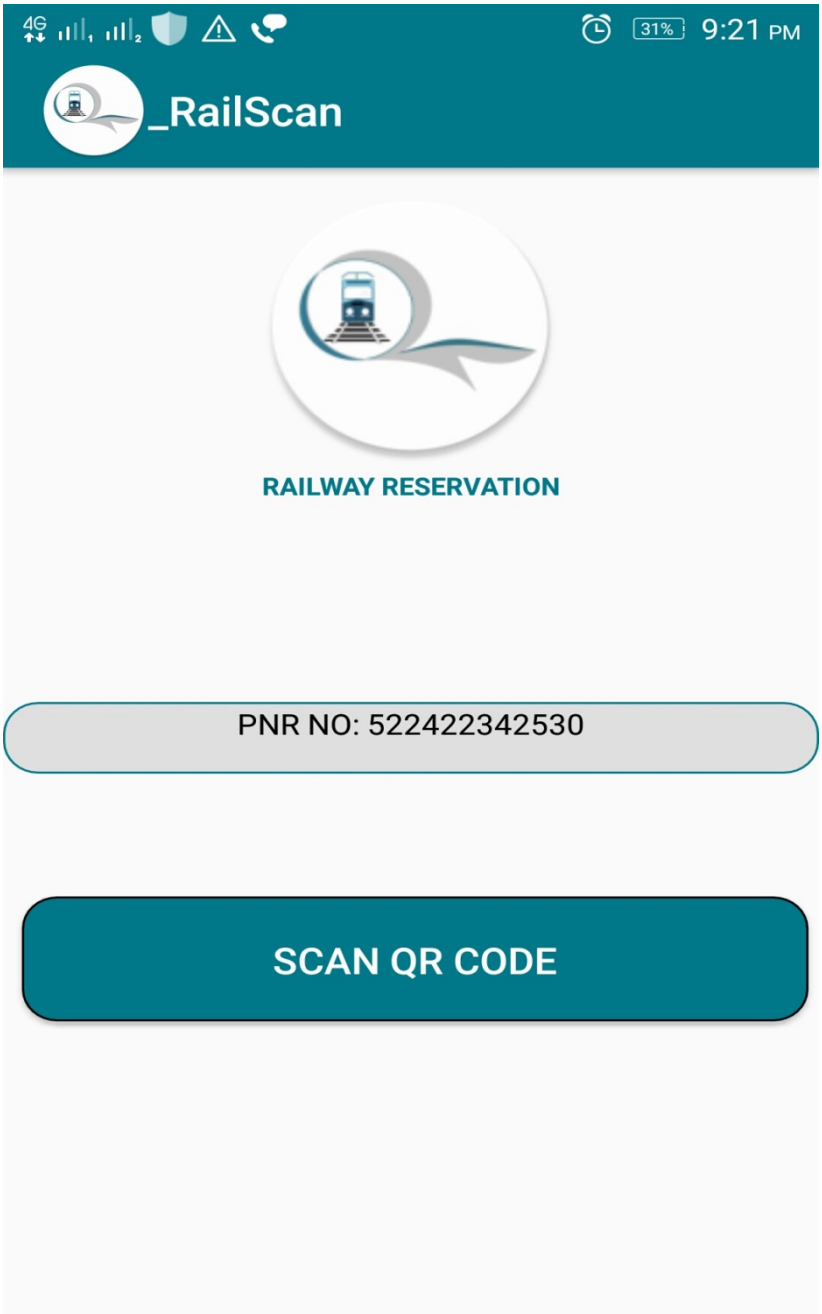
14. NEWS :



15. CAMERA_SCANNER TO SCAN QR CODE IN TICKET :



After Scanning QR Code: The PNR Displays.



5.3 CODES

1. JSP PAGES :

Header.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>RAILWAY</title>
<link href="../CSS/stylee.css" rel="stylesheet" type="text/css">
<link href="../CSS/imageForm.css" rel="stylesheet" type="text/css">
<script src="../js/jquery.js"></script>
<script src="../js/HtmlandCommon.js"></script>
</head>
<body>

<div class="pageloading"></div>
      <div class="alert">
            <div class="alertBox">
                  <div class="alertHeader">RAILWAY ALERT</div>
                  <div id="alertContent"></div>
                  <div class="alertFooter">
                        <button type="button" style="width:30%;
height:90%;" id="alertButton">OK</button>
                  </div>
            </div>
      </div>

<div class="modal">
            <div class="modal-box" >
                  <div class="modal-header">
                    <div style="background-
color:#c0c0c0;width:100%;height:15%;
                     top:100%;position:relative;">
                     </div>
                    <div style="height:10px;">
                     </div>
                  </div>
                  <br>
               <div id="modal-content">

                  </div>
                  <div class="modal-footer">
                        <button type="button" style="width:30%;
height:90%;" id="modalButton">OK</button>
                  </div>
           </div>
```

```
</div>

<div class="modalConfirm">
            <div class="modalConfirm-box" >
                  <div class="modalConfirm-header">
                    <div style="background-
color:#c0c0c0;width:100%;height:15%;
                     top:100%;position:relative;">
                     </div>
                   <div style="height:10px;">
                    </div>
                  </div>
                  <br>
              <div id="modalConfirm-content">

                  </div>
                  <div class="modalConfirm-footer">
                        <button type="button" style="width:30%;
height:90%;" id="modalConfirmButtonYes">YES</button>
                        <button type="button" style="width:30%;
height:90%;" id="modalConfirmButtonNo">No</button>
                  </div>
           </div>
</div>

<div id="header">
            <h1 id="title">RAILWAY RESERVATION</h1>
            <img src="../images/logo.png" alt="logo" id="logo" width=18%>
            <div id="nav">
                  <ul>
                        <li><a class="active"
href="../Phase2/viewdetails.jsp" id="viewdetailsss">Home</a></li>
                        <li><a href="../Phase1/News.jsp"
id="newsss">News</a></li>
                        <li><a href="../Phase1/ContactUs.jsp"
id="contactusss">Contact Us</a></li>
                        <li><a href="../Phase1/AboutUs.jsp"
id="aboutusss">About Us</a></li>
                        <% if(session.getAttribute("userid")!=null){

      if(((String)session.getAttribute("role")).equalsIgnoreCase("customer"
)){
                              %>
                        <script type="text/javascript">

      if((window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/index.
jsp"))

      ||(window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/signup.
jsp"))
                        ){

      window.location.href="../Phase2/viewdetails.jsp";
                        }
                        </script>

                        <li style="float:right">
                        <button title="Cancel Booking"
      class="imagebutton nobuttonhover nobuttondefaultstyle"
style="width:50px;height:50px;" id="menuClick">
```

```
      <img src="../images/menu.png"  class="imagebuttonImage"
id="cancelTicket" />
      </button>
                        </li>
                  </ul>
                   <div class="dropdown">
                  <a href="../Phase6/history.jsp"><img src="../images/user-
history.png" style="width:20%;height:40px;background:transparent;">User-
History</a>
                  <a href="../Phase6/cancellation.jsp"><img
src="../images/Ticket-Cancellation.png"
style="width:15%;height:40px;background:transparent;">Ticket
Cancellation</a>
                  <a href="../Phase7/chatting.jsp"><img
src="../images/emergency.png"
style="width:15%;height:40px;background:transparent;">Emergency Chat</a>
                  <a href="../Phase7/logout.jsp"><img
src="../images/logout.png"
style="width:15%;height:40px;background:transparent;">Logout</a>
                  </div>
                  <%}else{
                        %>
                        <script type="text/javascript">

      if(!((window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/inde
x.jsp"))
                              ||
(window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/signup.jsp"))
                              ||
(window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/News.jsp"))
                              ||
(window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/ContactUs.jsp")
)
                              ||
(window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/AboutUs.jsp"))
)){
                                    window.location.href="../Phase1/index.jsp";
                        }
                        </script>
                  <%
                  }
                  }else{%>
                        <script type="text/javascript">

      if(!((window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/inde
x.jsp"))
                                    ||
(window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/signup.jsp"))
                                    ||
(window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/News.jsp"))
                                    ||
(window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/ContactUs.jsp")
)
                                    ||
(window.location.pathname.includes("/RAILWAY_PROJECT/Phase1/AboutUs.jsp"))
)){window.location.href="../Phase1/index.jsp";
                              }
                        </script><%
                  }
                  %>
            </div></div>
```

Footer.jsp

```
<footer>
	<div id="grad">
		<p>-----------@Copyright Railway Reservation
			2018-----------</p>
	</div>
</footer>

</body>
</html>
```

Index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
	pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script src="../js/index.js"></script>
<!--  -->
<c:if test="${param.success != null}">
<script>
$(document).ready(function(){
	alertBox("Sign Up Successful!");
});
</script>
</c:if>
<!--  -->


<center>
	<form method="post" id="authenticate"
action="/RAILWAY_PROJECT/railway/loginAuth" >
		<fieldset style="width: 30%">
			<legend>
				<H2>Login</H2>
			</legend>
			<table style="width: 400px; height: 120px">
				<tr>
					<td colspan="2"><label
id="userhideeffect"><b>User ID</b></label></td>
				</tr>
				<tr>
					<td colspan="2"><input type="text"
placeholder="User ID"
					name="userid"	id="userhideeffectinput"
class="user-icon" autocomplete="off"/></td>
				</tr>
				<tr>
					<td colspan="2"><label
id="passhideeffect"><b>Password</b></label></td>
				</tr>

				<tr>
					<td colspan="2"><input type="password"
```

```
                                    name="password"   placeholder="Password"
id="passhideeffectinput" class="password-icon" />
                                    </td>
                              </tr>

                        </table>
                        <br>
                        <br>
                  </fieldset>
                  <table width="400" height="80">
                        <tr>
                              <td>
                                    <button type="button"
id="goToValidate">Submit</button>
                              </td>
                              <td>
                                    <button type="button" id="signup">Sign
Up</button>
                              </td>
                        </tr>
                  </table>
            </form>

</center>

<jsp:include page="../COMMON/footer.jsp" />
```

Signup.jsp

```
<%@ page import="railwayFrequentFunctions.DateFunctions"%>
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script src='https://www.google.com/recaptcha/api.js'></script>
<script src="../js/signup.js"></script>
<center>

      <H2>Registration Page</H2>
      <form name="form1" method="post"
action="/RAILWAY_PROJECT/railway/signUpSuccessful">

            <fieldset>
                  <legend>Login Details</legend>
                  <table style="width: 570px; height: 120px">

                        <tr>
                              <td>User ID:
                              <img src="../images/loading.gif"
class="loading" id="userid_loading">
                              </td>
                              <td><input type="text" name="txt_userid"
```

```
                                    placeholder="Anything You Think
Relevant">
                              <img src="" class="validity"
id="userid_validity">
                              </td>
                        </tr>
                        <tr>
                              <td></td>
                              <td class="warning" id="userid_warn"></td>
                        </tr>
                        <tr>
                              <td>Password:<br>
                              <button type="button" title="Show Password"
class="imagebutton nobuttonhover nobuttondefaultstyle">
                              <img src="../images/eyeclose.png"
class="eyepassword"></button></td>
                              <td><input type="password"
name="txt_password"
                                    placeholder="Type Password Be Secured">
                                    <img src="" class="validity"
id="password_validity"></td>
                        </tr>
                        <tr>
                              <td></td>
                              <td class="warning" id="password_warn"></td>
                        </tr>
                        <tr>
                              <td>Retype Password:</td>
                              <td><input type="password"
name="txt_retypepassword"
                                    placeholder="Retype Password" disabled>
                                    <img src="" class="validity"
id="retypepassword_validity"></td>
                        </tr>
                        <tr>
                              <td></td>
                              <td class="warning"
id="retypepassword_warn"></td>
                              </tr>
                  </table>
            </fieldset>

            <br>
            <fieldset>
                  <legend>Personal Details</legend>
                  <table style="width: 570px; height: 500px;">

                        <tr>
                              <td>Full Name:</td>
                              <td><input type="text" name="txt_fname"
                                    placeholder="Sweet Name"><img src=""
class="validity" id="fname_validity"></td>
                        </tr>
                        <tr>
                              <td></td>
                              <td class="warning" id="fname_warn"></td>
                        </tr>
                        <tr>
                              <td>Mobile No:</td>
                              <td><input type="text" name="mobile"
```

```
                                    placeholder="9999999999"><img src=""
class="validity" id="mobile_validity"></td>
                        </tr>
                        <tr>
                              <td></td>
                              <td class="warning" id="mobile_warn"></td>
                        </tr>
                        <tr>
                              <td>Gender:</td>
                              <td><select name="gender">
                              <option value="" disabled selected
hidden="true">---Gender---</option>
                              <option value="male">Male</option>
                              <option value="female">Female</option>
                              <option value="others">Others</option>
                              </select>
                              <img src="" class="validity"
id="gender_validity"></td>
                        </tr>
                        <tr>
                            <td></td>
                              <td class="warning" id="gender_warn"></td>
                        </tr>
                        <tr>
                              <td>Marital Status:</td>
                              <td><select name="marital">
                              <option value="" disabled selected
hidden="true">---Marital Status---</option>
                              <option value="married">Married</option>
                              <option value="unmarried">Unmarried</option>
                              </select>
                              <img src="" class="validity"
id="marital_validity"></td>
                        </tr>
                        <tr>
                              <td></td>
                              <td class="warning" id="marital_warn"></td>
                        </tr>
                        <tr>
                              <td>Date Of Birth:</td>
                              <td><input type="date" name="DOB" id="DOB"
                              max="<%= (new
DateFunctions()).dateforMinAge(18) %>"
                              min="<%= (new
DateFunctions()).dateforMinAge(60) %>">
                              <img src="" class="validity"
id="dob_validity"></td>
                        </tr>
                        <tr>
                              <td></td>
                              <td class="warning" id="dob_warn"></td>
                        </tr>
                        <tr>
                              <td>State:
                              <img src="../images/loading.gif"
class="loading" id="state_loading">
                              </td>
                              <td><select name="state">

                                    <option value="" disabled
selected hidden="true">---Select
```

```
                                                State---</option>
                                    </select>
                                    <img src="" class="validity"
id="state_validity">
                                    </td>
                              </tr>
                              <tr>
                                    <td></td>
                                    <td class="warning" id="state_warn"></td>
                              </tr>
                              <tr>
                                    <td>District:
                                    <img src="../images/loading.gif"
class="loading" id="district_loading">
                                    </td>
                                    <td><select name="district" disabled>
                                                <option value="" disabled
selected hidden="true">---Select
                                                      District---</option>

                                    </select>
                                    <img src="" class="validity"
id="district_validity">
                                    </td>
                              </tr>
                              <tr>
                                    <td></td>
                                    <td class="warning" id="district_warn"></td>
                              </tr>
                              <tr>
                                    <td>Address:</td>
                                    <td><textarea rows="4" cols="30"
                                          name="address"    placeholder="Enter
your Residential Address"></textarea>
                                    <img src="" class="validity"
id="address_validity"></td>
                              </tr>
                              <tr>
                                    <td></td>
                                    <td class="warning" id="address_warn"></td>
                              </tr>
                              <tr>
                                    <td>Email:</td>
                                    <td><input type="email" name="email"
                                          id="email" placeholder="xyz@abc.com"
class="email-icon">
                                    <img src="" class="validity"
id="email_validity"></td>
                              </tr>
                              <tr>
                                    <td></td>
                                    <td class="warning" id="email_warn"></td>
                              </tr>
                              <tr id="emailotptd">
                              <td><img src="../images/loading.gif"
class="loading" id="otp_loading"></td>
                              <td>
                              <button type="button" id="forotp"
class="nobuttonhover"
                              style="width:80%;height:100%;">GET OTP</button>
```

```
                        <button type="button" id="foremail"
class="nobuttonhover"
                        style="width:80%;height:100%;">CHANGE
EMAIL/RESEND</button>
                        <img src="" class="validity"></td>
                        </tr>
                        <tr id="otptd">
                              <td>OTP:<img src="../images/loading.gif"
class="loading" id="otpcheck_loading"></td>
                              <td><input type="text" name="OTP"
placeholder="AB3456">

                              <img src="" class="validity"
id="otp_validity"></td>
                        </tr>
                        <tr id="otptd1">
                              <td></td>
                              <td class="warning" id="otp_warn"></td>
                        </tr>
                        <tr>
                              <td></td>
                              <td>
                              <div style="text-align: center">
                                    <div class="g-recaptcha"
                                          data-
sitekey="6Lc48jwUAAAAAP3LTfSG8y7blgt1EgBhu-jCzora" style=" display: inline-
block;">
                                    </div>
                                    <img src="" class="validity">
                              </div>
                              </td>

                        </tr>
                        <tr>
                              <td colspan="2" class="warning"
id="recaptcha_warn"></td>
                        </tr>
                  </table>

            </fieldset>
            <br> <br>
            <button type="button" name="signup" id="button1">Sign
Up</button>


      </form>


</center>

<jsp:include page="../COMMON/footer.jsp" />
```

Aboutus.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script src="../js/AboutUs.js"></script>
<h2>About Us</h2>
<div id="general-info">
<div id="about-content"><b style="color:#00788b">Railway Reservation</b> is
a website
which helps consumers in online booking of Train Tickets very easily and
quickly with the comfort of the home or office.
Railways are the most important means of transport in India. Till 1985, All
tickets were issued manually and Stand-alone
Computerized ticketing and reservation was introduced in multiple
phases.<br>
The main functionality of the project is Reserving train seats and
cancellation of seats online. <br>
At Present, where India is moving towards Digital India , this is a small
effort taken by us.
</div>
</div>
   <div style="background:linear-gradient(#00788b,#005866);font-
family:fantasy;color:#fefefe;text-
align:left;width:80%;position:relative;height:30px;left:10%;top:30px">
   DEVELOPED BY:
   </div>
   <div
style="float:left;left:10%;position:relative;top:50px;width:30%;height:200p
x;">
   <img src="" id="photo1" style="border:1px solid
#00788b;height:200px;width:100%;">
   <h2>Rahul Mukesh Singh</h2>
   </div>

   <div
style="width:30%;height:200px;float:right;right:10%;position:relative;top:5
0px">
   <img src="" id="photo2" style="border:1px solid
#00788b;height:200px;width:100%">
   <h2>Sanjana Gopalkrishna</h2>
   </div>


<BR><BR><BR><br><br><br><br><br><BR><BR><BR><br><br><br><br><br>
<jsp:include page="../COMMON/footer.jsp" />
```

ContactUs.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script src="../js/ContactUs.js"></script>
<h2>Contact Us</h2>
   <div id="contact">
      <p style="left:1%;position:relative;top:100px;">If you would like to
discuss our solutions, please contact us and we will be happy to provide
you
      <br>with all the information you need.</p>
     <img src="../images/contact.png"
style="float:right;position:relative;height:300px;">
     <table style="top:100px;position:relative;width:50%;left:6%" >
     <tr>
           <td><img src="../images/email.png" style="height:30px"></td>
          <td><a
href="mailto:railway.project.xaviers@gmail.com">railway.project.xaviers@gma
il.com</a></td>
     </tr>
     <tr>
            <td><img src="../images/call.png" style="height:30px"></td>
            <td>9999999999</td>
     </tr>

     <tr>
          <td><img src="../images/source.png" style="height:30px"></td>
          <td><a href="">St.Xavier's College -Autonomous, Mumbai 5,
Mahapalika Marg,Chhatrapati Shivaji Terminus Area, Fort, Mumbai,
Maharashtra 400001</a></td>
     </table>
     </div>


<jsp:include page="../COMMON/footer.jsp" />
```

News.jsp

```
<%@page import="railwayFrequentFunctions.DatabaseConnection"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script src="../js/News.js"></script>
<sql:setDataSource var="con" driver="<%= (new
DatabaseConnection()).getDriver() %>" url="<%= (new
DatabaseConnection()).getURL() %>"
user="<%= (new DatabaseConnection()).getUsername() %>" password="<%= (new
DatabaseConnection()).getPassword() %>"/>
<sql:query dataSource="${con}" var="rs">
SELECT * FROM `news`;
</sql:query>
<h2>News and Updates</h2>
<c:forEach var="row" items="${rs.rows}">
<div id="news-container">

<div id="news-block">
       <div id="news-image">
          <img src="${row.newsimage}" class="news-image">
       </div>

      <div id="news-content">
      <div id="news-header">${row.newsheading}
      </div>
         <p>
   ${row.newsdetail}
        </p>
      </div>
   </div>

</div>
</c:forEach>
<br><br><br><br>

<jsp:include page="../COMMON/footer.jsp" />
```

Viewdetails.jsp

```
<%@page import="railwayFrequentFunctions.DatabaseConnection"%>
<%@page import="java.time.LocalDate"%>
<%@page import="java.util.Date"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script src="../js/jquery.tablesorter.js"></script>
<script src="../js/viewdetails.js"></script>

<br>
<c:set var="now" value="<%=new
Date(System.currentTimeMillis()+(1000*60*60*24)) %>"/>
<c:set var="maxMonth" value="<%=LocalDate.now().plusMonths(4)%>" />
<c:set var="exists" value="false" />
<div id="search-train">
<form action="viewdetails.jsp" method="post" name="details">

<table style="width:50%;float:left;">
     <tr>
       <td>Journey Date:</td>
       <td><input type="date" name="date" min='<fmt:formatDate
pattern="yyyy-MM-dd" value="${now}"/>' max="${maxMonth}"
       style="padding:1%;border-radius:8px;font-size:63%;width:97%;"
class="calendar-icon" value="${param.date}"></td>


     </tr>

    <tr>
       <td>Source:<img src="../images/loading.gif" class="loading"
                              id="source_station_loading"></td>
       <td>
         <select name="source" style="width:100%;padding:1%;border-
radius:8px;padding-left:30%" class="source-icon">
         <option value="" disabled selected hidden="true">---Source---
</option>
         </select>
       </td>

    </tr>

    <tr>
      <td>Destination:<img src="../images/loading.gif"
                              class="loading"
id="dest_station_loading"></td>
      <td>
      <select name="destination" style="width:100%;padding:1%;border-
radius:8px;padding-left:30%;" class="source-icon" disabled>
      <option value="" disabled selected hidden="true">---Destination---
</option>
      </select></td>

    </tr>
    <tr></tr>
```

```
    <tr>
      <td colspan="2" style="padding-left:15%"><button type="button"
id="search-train-button" style="width:80%;">Search</button></td>
    </tr>
  </table>

 <table style="width:50%;height:190px">
       <tr>
         <td colspan="3" style="vertical-align:top">

            <input type="text" style="width:75%;padding-top:0;"
class="train-icon searchtrainnow" placeholder="Search Train..."
autocomplete="off">

                 <div id="searchtrain-box">
                 </div>
         </td>

         <td style="vertical-align:top">
           </td>
       </tr>
 </table>


</form>
</div>


<c:if test="${param.date !=null}">
<form name="proceedToNextPage"
action="/RAILWAY_PROJECT/railway/trainSeltoClassSel" method="post">
<input type="hidden" name="selectedDatee" value="${param.date}"/>
<input type="hidden" name="selectedSrcStation1" id="selectedSrcStation"
value="${param.source}"/>
<input type="hidden" name="selectedDestStation1" id="selectedDestStation"
value="${param.destination}"/>
<input type="hidden" name="selectedTrainno" value=""/>
<input type="hidden" name="selectedTrainreturn" value=""/>
</form>
<fmt:parseDate var="journeydate" pattern="yyyy-MM-dd"
value="${param.date}"/>
<fmt:formatDate var="weekjourneydate" pattern="EEEE"
value="${journeydate}"/>

<sql:setDataSource var="con" driver="<%= (new
DatabaseConnection()).getDriver() %>" url="<%= (new
DatabaseConnection()).getURL() %>"
user="<%= (new DatabaseConnection()).getUsername() %>" password="<%= (new
DatabaseConnection()).getPassword() %>"/>
<sql:query dataSource="${con}" var="rs">

SELECT train_no,train_name,(SELECT station_name FROM station_master WHERE
station_code = source_station_code) AS source_station,
(SELECT station_name FROM station_master WHERE station_code =
dest_station_code) AS  dest_station,special,train_return
FROM train_master t where t.route_id IN
(SELECT DISTINCT route_id FROM route_master r WHERE
r.station_code=t.dest_station_code AND counter
>= (SELECT counter from route_master r1 where r1.route_id=r.route_id AND
r1.station_code=
```

```
(SELECT station_code FROM station_master WHERE station_name = ?) AND
counter
> (SELECT counter from route_master r2 where r2.route_id=r.route_id AND
r2.station_code=
(SELECT station_code FROM station_master WHERE station_name = ?) AND
counter
>= (SELECT counter from route_master r3 where r3.route_id=r.route_id AND
r3.station_code=t.source_station_code))))
AND (t.train_no,t.train_return) IN (SELECT train_no,train_return FROM
train_normal_schedule where ${weekjourneydate}=1);

<sql:param value="${param.destination}"/>
<sql:param value="${param.source}"/>

</sql:query>


<div class="pagination">
<table class="displayTableDesign">
<thead>
  <tr>
    <th>Train No</th>
    <th>Train Name</th>
    <th>Source</th>
      <th>Destination</th>
    <th>Special Train</th>
    <th></th>
  </tr>
</thead>
<tbody>
   <script>
      $(document).ready(function(){
            $('#searchtrain-box').empty();
      });
      </script>
      <c:forEach var="row" items="${rs.rows}">
      <c:set var="exists" value="true"/>
      <tr>
      <td><c:out value="${row.train_no}" />
            <input type="hidden" value="${row.train_return}" />
      </td>
      <td><c:out value="${row.train_name}" /></td>
      <td><c:out value="${row.source_station}" /></td>
      <td><c:out value="${row.dest_station}" /></td>
      <td>
      <c:choose>
      <c:when test="${row.special}" >
      Yes
      </c:when>
      <c:otherwise>
      No
      </c:otherwise>
      </c:choose>
      </td>
      <td><button type="button" title="Time" class="imagebutton
nobuttonhover nobuttondefaultstyle routetime">
                              <img src="../images/time.png"
class="imagebuttonImage"></button>
            <button type="button" title="Proceed" class="imagebutton
nobuttonhover nobuttondefaultstyle proceed" >
```

```
                                    <img src="../images/rightarrow.png"
class="imagebuttonImage"></button></td>
      </tr>

      <script>
      $(document).ready(function(){

            $('#searchtrain-box').append('<div><input type="radio"
name="Train-search"
style="float:left;">${fnc:toLowerCase(row.train_name)}</div>');

      });
      </script>
      </c:forEach>
</tbody>
</table>

<div class="divpaginationbutton">

</div>

</div>
<c:choose>
<c:when test="${exists eq true}">
<script>
$(document).ready(function(){
      $(".pagination").show();
});
</script>

</c:when>
<c:otherwise>
<script>
$(document).ready(function(){
      $(".pagination").hide();
      alertBox("No Trains Found!!!");
});
</script>
</c:otherwise>
</c:choose>
</c:if>
 <br><br><br><br><br><br>
<jsp:include page="../COMMON/footer.jsp" />
```

Class_selection.jsp

```
<%@page import="railwayFrequentFunctions.DatabaseConnection"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script src="../js/classselection.js"></script>
<c:set var="trainno" value="${sessionScope.trainno}" />
<sql:setDataSource var="con" driver="<%=(new
DatabaseConnection()).getDriver() %>"
 url="<%=(new DatabaseConnection()).getURL() %>" user="<%=(new
DatabaseConnection()).getUsername() %>"
 password="<%=(new DatabaseConnection()).getPassword() %>" />
<sql:query var="rs" dataSource="${con}">
SELECT class_id,class_name from train_class_details where class_id in
(SELECT class_id FROM train_coach_details where train_no=?);
<sql:param value="${trainno}" />
</sql:query>
<c:set var="counter" value="0"/>
<table class="train_class_container">
<tr>
<c:forEach var="row" items="${rs.rows}">
<td><div class="train_class"
style="background:url(../images/seat.png),linear-gradient(#00788b,
#008599,#009bb3);
background-repeat:no-repeat;background-size:contain;background-
position:center">
<c:out value="${row.class_name}" />
<input type="hidden" value="${row.class_id}">
<div class="train_class_know_more">
<button type="button" style="width:25%;height:60px;background-
color:#dfdfdf;color:#00788b;top:60px;
position:relative;border-radius:5px;" class="seatmap">Seat Map</button>
<button type="button" style="width:25%;height:60px;background-
color:#dfdfdf;color:#00788b;top:60px;
position:relative;border-radius:5px;"
class="proceedtoquotadetails">Proceed</button>
</div>
</div></td>
<c:set var="counter" value="${counter + 1}"/>
<c:if test="${counter%3==0}">
</tr><tr>
</c:if>
</c:forEach>
</tr>
</table>

<form name="proceedToQuotaDetails" method="post"
action="/RAILWAY_PROJECT/railway/classSelToQuotaDetails">
<input type="hidden" name="classid" value="">
</form>
<button style="background:url(../images/back.png),radial-gradient(#00788b,
#005866);background-repeat:no-repeat;
background-size:contain;background-position:left;padding-
left:60px;width:12%;top:70px;left:85%;position:relative"
id="goToViewDetails">Back</button>

<jsp:include page="../COMMON/footer.jsp" />
```

Quota_details.jsp

```
<%@page import="railwayFrequentFunctions.DatabaseConnection"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script type="text/javascript" src="../js/quotadetails.js"></script>
<sql:setDataSource var="con" driver="<%=(new
DatabaseConnection()).getDriver() %>"
 url="<%=(new DatabaseConnection()).getURL() %>" user="<%=(new
DatabaseConnection()).getUsername() %>"
 password="<%=(new DatabaseConnection()).getPassword() %>" />
<sql:query var="rs" dataSource="${con}">
SELECT quota_name,quota_id,quota_details FROM quota_details;
</sql:query>
<c:set var="counter" value="0"/>
<table class="quota_class_container">
<tr>
<c:forEach var="row" items="${rs.rows}">
<td>
<div class="quota_class" style="background-
image:url(../images/${row.quota_id}-icon.png),radial-gradient(#d9d9d9,
#00788b,#005866);background-repeat:no-repeat;background-
position:center;background-size:contain">
<c:out value="${row.quota_name}(${row.quota_id})"/>
<div class="quota_class_know_more">
<p><c:out value="${fnc:trim(row.quota_details)}"/></p>
</div>
</div></td>
<c:set var="counter" value="${counter + 1}"/>
<c:if test="${counter%3==0}">
</tr><tr>
</c:if>
</c:forEach>
</tr>
</table>
<button id="backToClassSelection" style="background-
image:url(../images/back-button.png),linear-
gradient(#b3b3b3,#a6a6a6,#737373);border-radius:10px;left:40%;padding-
right:40px;color:#fefefe;top:60px;position:relative;background-repeat:no-
repeat;width:10%;background-position:right;background-
size:contain">Back</button>
<button id="proceedToSeatAvailability" style="background-
image:url(../images/next-button.png),linear-
gradient(#b3b3b3,#a6a6a6,#737373);border-radius:10px;left:40%;;padding-
left:35px;color:#fefefe;top:60px;position:relative;background-repeat:no-
repeat;width:12%;background-position:left;background-
size:contain">Proceed</button>
<jsp:include page="../COMMON/footer.jsp" />
```

Seat_availability.jsp

```
<%@page import="railwayFrequentFunctions.DatabaseConnection"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script type="text/javascript" src="../js/seatavailability.js"></script>
<jsp:useBean id="price" class="railwayFrequentFunctions.DatabasePrice" />
<c:set var="pricewithoutquota"
value="${price.getTrainRoutePrice(sessionScope.source,sessionScope.destinat
ion,sessionScope.routeId,

      sessionScope.trainno,sessionScope.trainreturn,sessionScope.classid)}"
/>
<sql:setDataSource var="con" driver="<%= (new
DatabaseConnection()).getDriver() %>" url="<%= (new
DatabaseConnection()).getURL() %>"
user="<%= (new DatabaseConnection()).getUsername() %>" password="<%= (new
DatabaseConnection()).getPassword() %>"/>
<sql:query dataSource="${con}" var="rs">
SELECT quota_id,quota_name,price_discount FROM quota_details order by
seats_quota desc;
</sql:query>
<table class="available-seats-display" align="center">

<tr>
<th>Quota</th>
<th class="seat-icon">Available<br>Confirm Tickets</th>
<th class="seat-icon">Available<br>RAC Tickets</th>
<th class="seat-icon">Available<br>Waiting Tickets</th>
<th>Gross Price<br>(After Discount)</th>
</tr>
<c:forEach var="row" items="${rs.rows}">
<tr>
<td><c:out value="${row.quota_name}( ${row.quota_id} )"/></td>
<td></td>
<td></td>
<td></td>
<td>
&#8377;
<fmt:formatNumber var="pricewithquota" value='${pricewithoutquota -
(pricewithoutquota * row.price_discount / 100)}'
 type="NUMBER" maxFractionDigits="1"/>
 <c:out value="${pricewithquota} ( ${row.price_discount}% )"/></td>
</tr>
</c:forEach>
</table>
<br><br>
<button id="backToQuotaDetails" style="background-image:url(../images/back-
button.png),linear-gradient(#b3b3b3,#a6a6a6,#737373);border-
radius:10px;left:40%;padding-
right:40px;color:#fefefe;top:60px;position:relative;background-repeat:no-
repeat;width:10%;background-position:right;background-
size:contain">Back</button>
<button id="proceedToBooking" style="background-image:url(../images/next-
button.png),linear-gradient(#b3b3b3,#a6a6a6,#737373);border-
radius:10px;left:40%;;padding-
```

```
left:35px;color:#fefefe;top:60px;position:relative;background-repeat:no-
repeat;width:12%;background-position:left;background-
size:contain">Proceed</button>


<br>
<jsp:include page="../COMMON/footer.jsp" />
```

Booking.jsp

```
<%@page import="railwayFrequentFunctions.DatabaseConnection"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp"%>
<jsp:include page="../COMMON/header.jsp" />
<script type="text/javascript" src="../js/booking.js"></script>
<sql:setDataSource var="con" driver="<%= (new
DatabaseConnection()).getDriver() %>" url="<%= (new
DatabaseConnection()).getURL() %>"
user="<%= (new DatabaseConnection()).getUsername() %>" password="<%= (new
DatabaseConnection()).getPassword() %>"/>
<sql:query dataSource="${con}" var="rs">
SELECT class_name FROM train_class_details where class_id=?;
<sql:param value="${sessionScope.classid}"/>
</sql:query>
<c:forEach var="row" items="${rs.rows}">
<input type="hidden" id="classNameWithID" value="${row.class_name}
(${sessionScope.classid})"/>
</c:forEach>
<div class="addcustomer-modal">
      <div class="addcustomer-modal-box">
            <div class="addcustomer-modal-header">
                  <button id="closeAddCustomerModal" style="left: 47%;
position: relative; margin-top: 0px"
                        class="imagebutton nobuttonhover
nobuttondefaultstyle" title="Close">
                        <span class="imagebuttonImage" style="font-size:
30px">&times;</span>
                  </button>

            </div>

            <div id="addcustomer-modal-content">

                  <form>
                        <input type="hidden" id="tableRowNoInModal"/>
                        <table style="width: 50%; left: 24%; position:
relative; top: 70px;">
                              <tr>
                                    <td>Name:</td>
                                    <td><input type="text" name="nameModal"
style="height: 35px" placeholder="Name"></td>
                              </tr>
```

```
                                    <tr>
                                          <td>Age:</td>
                                          <td><input type="text" name="ageModal"
style="height: 35px" placeholder="Age"></td>
                                    </tr>
                                    <tr>
                                          <td>Gender:</td>
                                          <td><input type="radio" name="gender"
value="Male">Male
                                                <input type="radio" name="gender"
value="Female">Female
                                                <input type="radio" name="gender"
value="Others">Others</td>
                                    </tr>
                                    <tr>
                                          <td>Quota:</td>
                                          <td><select name="quota" style="height:
35px">
                                                </select>
                                          </td>
                                    </tr>
                                    <tr>
                                          <td>Seat Preference:</td>
                                          <td><select name="seatpref"
style="height: 35px">
                                                </select>
                                          </td>
                                    </tr>
                                    <tr>
                                          <td colspan="2"><button type="button"
id="confirmDetails"
                                                style="width: 60%; height:
45px; font: 23px fantasy, serif, sans-serif;
                                                background-color:
white;border: 1px solid #00788b;color: #00788b;">
                                                Confirm<img
src="../images/loading.gif"
                                                class="loading"
id="confirmDetailsLoading"></button></td>
                                    </tr>
                              </table>
                        </form>
                  </div>
            </div>
      </div>

<table class="addcustomer-display">
      <tr>
            <th style="width: 5%">Sr.No</th>
            <th style="width: 25%">Customer Name</th>
            <th style="width: 15%">Class</th>
            <th style="width: 14%">Add Details</th>
            <th>Quota</th>
            <th>Status</th>
            <th>Gross Price (After Discount)</th>
            <th style="width: 5%"></th>
      </tr>

      <tr>
```

```
            <td>1.</td>
            <td>Passenger</td>
            <td></td>
            <td>
                  <button title="Add Details" id="adddetails"
                        class="imagebutton nobuttonhover
nobuttondefaultstyle"
                        style="width: 25%; border-radius: 160px; height:
40px; background: radial-gradient(#00788b, #005866);">
                        <img src="../images/add.png"
class="imagebuttonImage" />
                  </button>

            </td>
            <td>
            Not Specified
            </td>
            <td>N/A</td>
            <td class="calculate">N/A</td>
            <td>
                  <button type="button" title="Delete Passenger"
id="delPassenger"
                        class="imagebutton nobuttonhover
nobuttondefaultstyle">
                        <img src="../images/del.png"
class="imagebuttonImage" />
                  </button>
            <input type="hidden" value="">
            </td>
      </tr>


</table>

<button title="Add Passenger" id="addCustomer"
class="imagebutton nobuttonhover nobuttondefaultstyle"
style="width:4%;top:80px;left:90%;position:relative;height:50px;">
<img src="../images/add-customer.png" class="imagebuttonImage"
style="height:50px;width:100%" />
</button>

<sql:setDataSource var="con" driver="<%= (new
DatabaseConnection()).getDriver() %>" url="<%= (new
DatabaseConnection()).getURL() %>"
user="<%= (new DatabaseConnection()).getUsername() %>" password="<%= (new
DatabaseConnection()).getPassword() %>"/>
<sql:query dataSource="${con}" var="rs">
SELECT tax_id,tax_price from tax;
</sql:query>


<table class="addcustomerprice-display">
      <tr>
            <td>Total Gross Price :</td>
            <td>&#8377;0</td>
      </tr>
<c:forEach var="row" items="${rs.rows}">
      <tr>
            <td><c:out value="${row.tax_id}"/>  (+):</td>
            <td class="taxcalculate"><c:out value="${row.tax_price}"/>
%</td>
```

```
		</tr>
</c:forEach>
		<tr>
			<td>Net Gross Price :</td>
			<td>&#8377;0</td>
		</tr>
</table>



<button style="width:25%;top:50px;position:relative;left:30%;
background:url(../images/book.png),linear-gradient(#00788b,
#005866);background-position:right;
background-repeat:no-repeat;background-size:contain"
id="res_ticket">Reserve Ticket</button>

<form name="moveToPaymentGateway" method="post"
action="../railway/bookTopaymentgateway">
<input type="hidden" name="ticketData" value=""/>
</form>
<br><br><br><br><br><br><br><br><br><br>
<jsp:include page="../COMMON/footer.jsp" />
```

Booked.jsp

```
<%@page import="railwayFrequentFunctions.DatabaseConnection"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
	pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp"%>
<jsp:include page="../COMMON/header.jsp" />
<script src="../js/booked.js"></script>
<c:set var="userid" value="${sessionScope.userid}" />
<sql:setDataSource var="con"
	driver="<%=(new DatabaseConnection()).getDriver()%>"
	url="<%=(new DatabaseConnection()).getURL()%>"
	user="<%=(new DatabaseConnection()).getUsername()%>"
	password="<%=(new DatabaseConnection()).getPassword()%>" />
<sql:query var="rs" dataSource="${con}">
SELECT email FROM user_details WHERE userid=?;
<sql:param value="${userid}" />
</sql:query>
<c:forEach var="row" items="${rs.rows}">
	<c:set var="email" value="${row.email}" />
</c:forEach>
<br />
<br />
<center>
<div class="sendMailNowLoading"></div>
	<div id="pdfContainer">
```

```
            <div class="sendMailNowLoading"></div>
            <iframe src="../railway/ticket" id="iframePDF"> </iframe>
            <div id="buttonsContainer">
                  <center>
                        <br/><br/><br/><br/><br/>
                        <div id="showButtons">
                              <a href="../railway/ticket" download>
                                    <button type="button" title="Download"
                                          class="imagebutton nobuttonhover
nobuttondefaultstyle">
                                          <img src="../images/download.png"
                                                style="width: 70px; height:
70px;" class="imagebuttonImage">
                                    </button>
                              </a>
                              <button type="button" title="Send Mail (
${email} )"
                                    class="imagebutton nobuttonhover
nobuttondefaultstyle" id="sendMailNow">
                                    <img src="../images/sendmail.png"
                                          style="width: 70px; height:
70px;" class="imagebuttonImage">
                              </button>
                        </div>
                  </center>
            </div>
      </div>
</center>
<jsp:include page="../COMMON/footer.jsp" />
```

Cancellation.jsp

```
<%@page import="java.util.Date"%>
<%@page import="java.text.SimpleDateFormat"%>
<%@page import="java.util.Calendar"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="java.sql.Connection"%>
<%@page import="railwayFrequentFunctions.DatabaseConnection"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script src="../js/cancellation.js"></script>
<h2>CANCELLATION</h2>
<table align="center" class="cancellation">
<tr>
 <th>PNR no.</th>
 <th>Name</th>
 <th>Source</th>
 <th>Destination</th>
 <th>Train name</th>
```

```
 <th>Departure-date time</th>
 <th>Arrival-date time</th>
 <th></th>
</tr>

<%
try{
Connection con=(new DatabaseConnection()).dbConnect();
String query="SELECT `PNR`, `cust_name`, `age`,`quota_id`, "+
"`class_id`, `PNR_counter`, `status_id`, "+
"`status_counter`, `gender`, `src_start_time`, `dest_end_time`, "+
"`cost_with_tax`,(SELECT `quota_name` FROM `quota_details` WHERE
`quota_id`=b.quota_id), "+
"(SELECT `class_name` FROM `train_class_details` WHERE
`class_id`=b.`class_id`), "+
"(SELECT `coach_code` FROM `train_class_details` WHERE
`class_id`=b.`class_id`), "+
"(SELECT `status_name` FROM `seat_status_details` WHERE
`status_id`=b.`status_id`) "+
"FROM `booked_customer_details` b where `userid`=? ORDER BY
`src_start_time`,`PNR`,`PNR_counter`";
PreparedStatement ps=con.prepareStatement(query);
ps.setString(1, (String)session.getAttribute("userid"));
ResultSet rs=ps.executeQuery();
while(rs.next()){
      String statusTableQuery="";
      if(!rs.getString(7).equalsIgnoreCase("wl")){
      statusTableQuery="SELECT (SELECT train_name from train_master t where
t.train_no=s.train_no AND "
+"t.train_return=s.train_return),(SELECT station_name from station_master s
where s.station_code=from_station),"+
"(SELECT station_name from station_master s where
s.station_code=to_station), "+
"`coach_counter`, `seat_no`, `train_src_time`, `train_dest_time` ,(SELECT
`berth_id` FROM `berth_details` "+
"WHERE `berth_id` IN (SELECT `berth_id` FROM "+rs.getString(5)+" WHERE
`seat_no`=s.`seat_no`)), "+
"(SELECT `berth_name` FROM `berth_details` "+
            "WHERE `berth_id` IN (SELECT `berth_id` FROM
"+rs.getString(5)+" WHERE `seat_no`=s.`seat_no`)) "+
"FROM "+rs.getString(7)+" s WHERE PNR=? AND PNR_counter=?";}
      else{
            statusTableQuery="SELECT (SELECT train_name from train_master t
where t.train_no=s.train_no AND "
                        +"t.train_return=s.train_return),(SELECT
station_name from station_master s where s.station_code=from_station),"+
                        "(SELECT station_name from station_master s where
s.station_code=to_station), "+
                        "0, 0, `train_src_time`, `train_dest_time` ,'none',
"+
                        "'none'"+
                        "FROM "+rs.getString(7)+" s WHERE PNR=? AND
PNR_counter=?";
      }
PreparedStatement ps1=con.prepareStatement(statusTableQuery);
      ps1.setString(1, rs.getString(1));
      ps1.setInt(2, rs.getInt(6));
      ResultSet rs1=ps1.executeQuery();
      while(rs1.next()){
%>
<tr>
```

```
<td><%= rs.getString(1) %></td>
<td><%= rs.getString(2) %></td>
<td><%= rs1.getString(2) %></td>
<td><%= rs1.getString(3) %></td>
<td><%= rs1.getString(1) %></td>
<td><% Calendar c=Calendar.getInstance();
            SimpleDateFormat sd=new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
            c.setTime(sd.parse(rs.getString(10)));
            Date departure=c.getTime();
            out.print(c.getTime());
%></td>
<td><% c.setTime(sd.parse(rs.getString(11)));
            Date arrival=c.getTime();
            out.print(c.getTime());
            %></td>
<td>
<center>
<button title="Show Details"
class="imagebutton nobuttonhover nobuttondefaultstyle">
<img src="../images/show-details.png"  class="imagebuttonImage"
id="showDetailsForPassenger" />
</button>

<button title="Cancel Booking"
class="imagebutton nobuttonhover nobuttondefaultstyle">
<img src="../images/cancel.png"  class="imagebuttonImage" id="cancelTicket"
/>
</button>
<a href="../railway/ticketdownloadlatest?pnr=<%= rs.getString(1).trim() %>"
download>
<button title="Download Pdf"
class="imagebutton nobuttonhover nobuttondefaultstyle">
<img src="../images/pdf.png"  class="imagebuttonImage" />
</button>
</a>
</center>
<input type="hidden" class="pnr" value="<%= rs.getString(1) %>" />
<input type="hidden" class="cust_name" value="<%= rs.getString(2) %>" />
<input type="hidden" class="age" value="<%= rs.getInt(3) %>" />
<input type="hidden" class="quota_id" value="<%= rs.getString(4) %>" />
<input type="hidden" class="class_id" value="<%= rs.getString(5) %>" />
<input type="hidden" class="PNR_counter" value="<%= rs.getInt(6) %>" />
<input type="hidden" class="status_id" value="<%= rs.getString(7) %>" />
<input type="hidden" class="status_counter" value="<%= rs.getInt(8) %>" />
<input type="hidden" class="gender" value="<%= rs.getString(9) %>" />
<input type="hidden" class="src_start_time" value="<%=departure %>" />
<input type="hidden" class="dest_end_time" value="<%=arrival %>" />
<input type="hidden" class="cost_with_tax" value="<%= rs.getDouble(12) %>"
/>
<input type="hidden" class="quota_name" value="<%= rs.getString(13) %>" />
<input type="hidden" class="class_name" value="<%= rs.getString(14) %>" />
<input type="hidden" class="coach_code" value="<%= rs.getString(15) %>" />
<input type="hidden" class="status_name" value="<%= rs.getString(16) %>" />
<input type="hidden" class="train_name" value="<%= rs1.getString(1) %>" />
<input type="hidden" class="from_station_name" value="<%= rs1.getString(2)
%>" />
<input type="hidden" class="to_station_name" value="<%= rs1.getString(3)
%>" />
<input type="hidden" class="coach_counter" value="<%= rs1.getInt(4) %>" />
<input type="hidden" class="seat_no" value="<%= rs1.getInt(5) %>" />
```

```
<input type="hidden" class="train_src_time" value="<%
c.setTime(sd.parse(rs1.getString(6))); out.print(c.getTime()); %>" />
<input type="hidden" class="train_dest_time" value="<%
c.setTime(sd.parse(rs1.getString(7))); out.print(c.getTime()); %>" />
<input type="hidden" class="berth_id" value="<%= rs1.getString(8) %>" />
<input type="hidden" class="berth_name" value="<%= rs1.getString(9) %>" />
</td>
</tr>
<%
      }}
con.close();
}catch(Exception e){
      e.printStackTrace();
}

%>
</table>



<BR><BR><BR>
<jsp:include page="../COMMON/footer.jsp" />
```

History.jsp

```
<%@page import="java.util.Date"%>
<%@page import="java.text.SimpleDateFormat"%>
<%@page import="java.util.Calendar"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="java.sql.Connection"%>
<%@page import="railwayFrequentFunctions.DatabaseConnection"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script src="../js/jquery.tablesorter.js"></script>
<script src="../js/history.js"></script>
<h2>HISTORY</h2>
<div class="pagination">
<table align="center" class="history displayTableDesign">
<thead>
<tr>
 <th>PNR no.</th>
 <th>Name</th>
 <th>Source</th>
 <th>Destination</th>
 <th>Train name</th>
 <th>Departure-date time</th>
 <th>Arrival-date time</th>
```

```
 <th></th>
</tr>
</thead>
<tbody>
<%
try{
Connection con=(new DatabaseConnection()).dbConnect();
String query="SELECT `PNR`, `cust_name`, `age`,`quota_id`, "+
"`class_id`, `PNR_counter`, `status_id`, "+
"`status_counter`, `gender`, `src_start_time`, `dest_end_time`, "+
"`cost_with_tax`,(SELECT `quota_name` FROM `quota_details` WHERE
`quota_id`=b.quota_id), "+
"(SELECT `class_name` FROM `train_class_details` WHERE
`class_id`=b.`class_id`), "+
"(SELECT `coach_code` FROM `train_class_details` WHERE
`class_id`=b.`class_id`), "+
"(SELECT `status_name` FROM `seat_status_details` WHERE
`status_id`=b.`status_id`) "+
"FROM `history_booked_customer_details` b where `userid`=? ORDER BY
`src_start_time`,`PNR`,`PNR_counter`";
PreparedStatement ps=con.prepareStatement(query);
ps.setString(1, (String)session.getAttribute("userid"));
ResultSet rs=ps.executeQuery();
while(rs.next()){
      String statusTableQuery="";
      if(!rs.getString(7).equalsIgnoreCase("wl")){
      statusTableQuery="SELECT (SELECT train_name from train_master t where
t.train_no=s.train_no AND "
+"t.train_return=s.train_return),(SELECT station_name from station_master s
where s.station_code=from_station),"+
"(SELECT station_name from station_master s where
s.station_code=to_station), "+
"`coach_counter`, `seat_no`, `train_src_time`, `train_dest_time` ,(SELECT
`berth_id` FROM `berth_details` "+
"WHERE `berth_id` IN (SELECT `berth_id` FROM "+rs.getString(5)+" WHERE
`seat_no`=s.`seat_no`)), "+
"(SELECT `berth_name` FROM `berth_details` "+
            "WHERE `berth_id` IN (SELECT `berth_id` FROM
"+rs.getString(5)+" WHERE `seat_no`=s.`seat_no`)) "+
"FROM history_"+rs.getString(7)+" s WHERE PNR=? AND PNR_counter=?";}
      else{
            statusTableQuery="SELECT (SELECT train_name from train_master t
where t.train_no=s.train_no AND "
                        +"t.train_return=s.train_return),(SELECT
station_name from station_master s where s.station_code=from_station),"+
                        "(SELECT station_name from station_master s where
s.station_code=to_station), "+
                        "0, 0, `train_src_time`, `train_dest_time` ,'none',
"+
                        "'none'"+
                        "FROM history_"+rs.getString(7)+" s WHERE PNR=? AND
PNR_counter=?";
      }
PreparedStatement ps1=con.prepareStatement(statusTableQuery);
      ps1.setString(1, rs.getString(1));
      ps1.setInt(2, rs.getInt(6));
      ResultSet rs1=ps1.executeQuery();
      while(rs1.next()){
%>
<tr>
<td><%= rs.getString(1) %></td>
```

```
<td><%= rs.getString(2) %></td>
<td><%= rs1.getString(2) %></td>
<td><%= rs1.getString(3) %></td>
<td><%= rs1.getString(1) %></td>
<td><% Calendar c=Calendar.getInstance();
            SimpleDateFormat sd=new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
            c.setTime(sd.parse(rs.getString(10)));
            Date departure=c.getTime();
            out.print(c.getTime());
%></td>
<td><% c.setTime(sd.parse(rs.getString(11)));
            Date arrival=c.getTime();
            out.print(c.getTime());
            %></td>
<td>
<center>
<button title="Show Details"
class="imagebutton nobuttonhover nobuttondefaultstyle">
<img src="../images/show-details.png"  class="imagebuttonImage"
id="showDetailsForPassenger" />
</button>
</center>
<input type="hidden" class="pnr" value="<%= rs.getString(1) %>" />
<input type="hidden" class="cust_name" value="<%= rs.getString(2) %>" />
<input type="hidden" class="age" value="<%= rs.getInt(3) %>" />
<input type="hidden" class="quota_id" value="<%= rs.getString(4) %>" />
<input type="hidden" class="class_id" value="<%= rs.getString(5) %>" />
<input type="hidden" class="PNR_counter" value="<%= rs.getInt(6) %>" />
<input type="hidden" class="status_id" value="<%= rs.getString(7) %>" />
<input type="hidden" class="status_counter" value="<%= rs.getInt(8) %>" />
<input type="hidden" class="gender" value="<%= rs.getString(9) %>" />
<input type="hidden" class="src_start_time" value="<%=departure %>" />
<input type="hidden" class="dest_end_time" value="<%=arrival %>" />
<input type="hidden" class="cost_with_tax" value="<%= rs.getDouble(12) %>"
/>
<input type="hidden" class="quota_name" value="<%= rs.getString(13) %>" />
<input type="hidden" class="class_name" value="<%= rs.getString(14) %>" />
<input type="hidden" class="coach_code" value="<%= rs.getString(15) %>" />
<input type="hidden" class="status_name" value="<%= rs.getString(16) %>" />
<input type="hidden" class="train_name" value="<%= rs1.getString(1) %>" />
<input type="hidden" class="from_station_name" value="<%= rs1.getString(2)
%>" />
<input type="hidden" class="to_station_name" value="<%= rs1.getString(3)
%>" />
<input type="hidden" class="coach_counter" value="<%= rs1.getInt(4) %>" />
<input type="hidden" class="seat_no" value="<%= rs1.getInt(5) %>" />
<input type="hidden" class="train_src_time" value="<%
c.setTime(sd.parse(rs1.getString(6))); out.print(c.getTime()); %>" />
<input type="hidden" class="train_dest_time" value="<%
c.setTime(sd.parse(rs1.getString(7))); out.print(c.getTime()); %>" />
<input type="hidden" class="berth_id" value="<%= rs1.getString(8) %>" />
<input type="hidden" class="berth_name" value="<%= rs1.getString(9) %>" />
</td>
</tr>
<%
      }}
con.close();
}catch(Exception e){
      e.printStackTrace();
}
```

```
%>
</tbody>
</table>

<div class="divpaginationbutton">

</div>

</div>



<BR><BR><BR>
<jsp:include page="../COMMON/footer.jsp" />
```

Chatting.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />
<script src="../js/chatting.js"></script>
<center>
<div class="chat-screen">

   <div id="vertical-bar">
            <img src="../images/emergency.png"
style="width:15%;top:16px;position:relative;height:50px;float:left"><a
href="#Emergency">Emergency</a>
   </div>

   <div id="chat-content">
         <table id="message-header">
         <td style="width:10%"><img src="../images/emergency.png"
style="width:50%;height:40px"></td>
         <td style="width:30%"><div class="chat-name">Emergency</div></td>
         <td style="float:right"><button title="Delete" class="imagebutton
nobuttonhover nobuttondefaultstyle"
          style="background:transparent;top:3px;position:relative"
id="deleteChats">
                                    <img src="../images/delete.png"
style="height:35px;">
                                </button>

          </td>
         </table>
         <div id="message-content">
         </div>


           <div id="message-footer">
```

```
            <input type="text" placeholder="Type a message"
style="height:5px;" id="chatChat-box">
            <button title="send" style="width:50px;border-
radius:40px;left:0.5%;top:7px;position:relative;height:50px"
id="sendChatNow"><img src="../images/send.png" style="width:50%"></button>
         </div>


   </div>



</div>
<input type="hidden" id="lastReceivedID" value="0"/>
<input type="hidden" id="userid" value="${sessionScope.userid}"/>
</center>
```

Logout.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%
session.invalidate();
response.sendRedirect("../Phase1/index.jsp");
                                        %>
```

RailwayRenameExtra.jsp

```
<%@ page import="railwayFrequentFunctions.pdfGeneration"%>
<%@ page import ="railwayFrequentFunctions.emailattachmentrailway"%>
<%@ page import ="railwayFrequentFunctions.TableGenerator"%>
<%@ page import ="railwayFrequentFunctions.mailnow"%>

<html>
   <head>
      <title>Send Attachment Email using JSP</title>
   </head>
   <body>
         <h1>Send Attachment Email using JSP</h1>


      <p align = "center">
     <%
            emailattachmentrailway ea=new emailattachmentrailway();
            TableGenerator fisg=new TableGenerator();
            mailnow mn=new mailnow();
            String[] headers={"FirstName","Lastname","Previous"};
          String[][]
data={{"Rahul","Singh","59"},{"Sanjana","GopalKrishna","68"},{"Railway","Pr
oject","NewNow"}};
               String address="rahulsinghmukesh@gmail.com";
            String subject="Railway Ticket";
```

```
            String attachedname="RAILWAY TICKET";
            String msg=fisg.MsgGeneratortable(headers,
data,"table1",false);
          out.println("Normal Mail Done :
"+mn.MailNowMsg(address,msg,subject)+"<br>");
                  msg=fisg.MsgGeneratortable(headers, data,"table2",true);
          String PNR="PNR";
          String csss="";
          out.println("Attached Mail Done :
"+ea.createattachandemail(address, subject, attachedname));
     %>
      </p>
   </body>
</html>
```

CommonInAllFile.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="../JSTLTagLib/jstlTaglib.jsp" %>
<jsp:include page="../COMMON/header.jsp" />

<jsp:include page="../COMMON/footer.jsp" />
${sessionScope.userid}<br>
${sessionScope.fname}<br>
${sessionScope.role}<br>
${sessionScope.date}<br>
${sessionScope.source}<br>
${sessionScope.destination}<br>
${sessionScope.trainno}<br>
${sessionScope.trainreturn}<br>
```

Style.css

```
.pageloading{
    position: fixed;
    left: 0px;
    top: 0px;
    width: 100%;
    height: 100%;
    z-index: 999;
    background: url('../images/loading.gif') 50% 50% no-repeat
rgb(249,249,249);
    opacity:0.8;

}

@font-face {
```

```
      font-family: headings;
      src: url(Quicksand-Light.otf);
}

/*--------------------------------------------------------------------------
-----*/
.alert {
      display:none;
    z-index: 100;
    position: fixed;
    left: 0;
    top: 0;
      bottom: 0;
    right:0;
    overflow: auto;
    background-color: rgba(0,0,0,0.4);
      width: 100%;
    height: 100%;
    }

.alertBox {

      position:absolute;
    left: 0;
    top:20%;
      right:0;
      border-radius:20px 5px 20px 5px;
    background-color: #fefefe;
    margin: auto;
    height:30%;
    border: 1px solid #999999;
    width: 30%;
    display: table;
    overflow: hidden;

}

.alertHeader{

border-radius:19px 5px 0px 0px;
text-align:center;
height:17%;
width:100%;
position:absolute;
left:0;
top:0;
background-color:#00788b;
font: 20px Palatino Linotype sans-serif Lucida Fax;
color:white;
width:100%;
}
.alertFooter {
border-radius:0px 0px 19px 5px;
text-align:center;
height:19%;
width:100%;
position:absolute;
left:0;
bottom:0;
background-color:#c0c0c0;
font: 20px Palatino Linotype sans-serif Lucida Fax;
```

```
}
#alertContent{
    display: table-cell;
    vertical-align: middle;
    width: 100%;
    margin: 0 auto;
    text-align: center;
      word-break: break-all;

}
/*-------------------------------------------------------------------------
-----*/

.modal {

    display:none;
    z-index: 50;
    position: fixed;
    left: 0;
    top: 0;
      bottom: 0;
    right:0;

    background-color: rgba(0,0,0,0.4);
      width: 100%;
    height: 100%;


    }

.modal-box {

      position:relative;
    left: 0;
    top:20%;
      right:0;
      border-radius:20px 5px 20px 5px;
    background-color: #fefefe;
    margin: 0 auto;
    height:60%;
    border: 1px solid #999999;
    width: 60%;
    overflow:hidden;
}

.modal-header{
border-radius:19px 5px 0px 0px;
text-align:center;
height:12%;
width:100%;
position:relative;
left:0;
top:0;
background-color:#00788b;
font: 20px Palatino Linotype sans-serif Lucida Fax;
color:white;

}
.modal-footer {
border-radius:0px 0px 19px 5px;
text-align:center;
```

```
height:13%;
width:100%;
position:relative;
left:0;
bottom:0;
background-color:#c0c0c0;
font: 20px Palatino Linotype sans-serif Lucida Fax;

}

#modal-content{
      position:relative;
      height:70%;
    font: 20px Palatino Linotype sans-serif Lucida Fax;
    border-color:#c0c0c0;
    vertical-align: middle;
    width: 100%;


      overflow:auto;
      }
/*-------------------------------------------------------------------------
-----*/


.displayTableDesign {
      margin:0 auto;
      margin-top:7%;
    border-collapse: collapse;
    width: 80%;
      font: 17px Palatino Linotype sans-serif Lucida Fax;

}
.displayTableDesign th,.displayTableDesign td {
    text-align: left;
    padding: 8px;
}

.displayTableDesign tbody tr:nth-child(even) td{background-color: white}
.displayTableDesign tbody tr:nth-child(odd) td{background-color: #d4d4d4}
.displayTableDesign th {
    background-color: #00788b;
    color: white;

}
.displayTableDesign th.header {
          background-image: url(../images/bothsort.png);
          background-size: 20px 20px;
          background-repeat: no-repeat;
          background-position: right;
          cursor: pointer;
      }

.displayTableDesign th.headerSortUp {
          background-image: url(../images/sortup.png);
          background-size: 20px 20px;
          background-repeat: no-repeat;
          background-position: right;
          background-color: #005866;
      }
.displayTableDesign th.headerSortDown {
```

```
            background-image: url(../images/sortdown.png);
            background-size: 20px 20px;
            background-repeat: no-repeat;
            background-position: right;
            background-color: #005866;

      }

.paginationbutton:not(#activepaginationbutton){

      background-color: white;
      padding:1%;
      border: 1px solid #005866;
      color: #005866;
      cursor : pointer;
      text-decoration : none;
      border-radius:5px;

}
.paginationbutton:hover:not(#activepaginationbutton){
      padding:1.3%;
}
.divpaginationbutton{
      position:relative;
      top:50px;
      text-align: right;
      right:10%;
      margin-bottom:0;

}
#activepaginationbutton{
      background-color: #00788b;
      padding:1%;
      border: none;
      color: white;
      cursor : pointer;
      text-decoration : none;
      border-radius:5px;

}

.noofrowpagination {
    -webkit-appearance: none;
    width: 20%;
    height: 20px;
    background: #d3d3d3;
    outline: none;
    opacity: 0.7;
    -webkit-transition: .2s;
}

.noofrowpagination:hover {
    opacity: 1;
}

.noofrowpagination::-webkit-slider-thumb {
    -webkit-appearance: none;
    appearance: none;
    width: 25px;
    height: 25px;
    background: #005866;
```

```
    cursor: pointer;
}

/*----------------------------------------------------------------------
------*/

*:focus {
    outline: none;
}
#logo {
      display: inline-block;
      float: left;
      position: absolute;
      top: 0px;
      left: 0px;
}

#title {
      padding-left: 70%;
      color: #00788b;
      float: right;
      font-family: Britannic Palatino Linotype sans-serif Lucida Fax;
      margin-top: 10px;
      margin-right: 20px;
      position: relative;
      display: inline-block;
}

body {
      color: #404040;
      background-color: #f2f2f2;
      font: 20px Palatino Linotype sans-serif Lucida Fax;
      margin: 0%;
}

input:not(.noofrowpagination), textarea {
      border-radius: 5px;
      border: solid 1px #00788b;
      padding: 1%;
      padding-bottom: 2%;
      padding-top: 2%;


}

#nav ul {
      list-style-type: none;
      margin: 0;
      padding: 0;
      overflow: hidden;
      background-color: #a6a6a6;
      position: relative;
      top: 0;
      width: 100%;
}

#nav li {
      float: left;
      display: inline-block;
}
```

```
#nav li a {
	display: block;
	color: #f2f2f2;
	text-align: center;
	padding: 14px 16px;
	text-decoration: none;
}

#nav ul li a:hover:not(.active) {
	background-color: #dfdfdf;
	color: #00788b;
}

#nav .active {
	background-color: #00788b;
}

#nav {
	position: relative;
	margin-top: 3%;
}

/*--------------------------------------------------------------------*/
#grad {
	position:absolute;
	height: 60px;
	background: -webkit-linear-gradient(left,
#002c33,#008599,#008599,#002c33 );
	/* For Safari 5.1 to 6.0 */
	background: -o-linear-gradient(left,
#002c33,#008599,#008599,#002c33);
	/* For Opera 11.1 to 12.0 */
	background: -moz-linear-gradient(left,
#002c33,#008599,#008599,#002c33);
	/* For Fx 3.6 to 15 */
	background: linear-gradient(to right,
#002c33,#008599,#008599,#002c33);
	/* Standard syntax (must be last) */
	width:100%;
	bottom: 0;
}


H2 {
	text-align:center;
	color: #00788b;
	font: normal 54px Georgia, Times, serif;
}

button:hover:not(.nobuttonhover) {
	box-shadow: 0 12px 16px 0 rgba(0, 0, 0, 0.24), 0 17px 50px 0
		rgba(0, 0, 0, 0.19);
}

button:not(.nobuttondefaultstyle) {
	background-color: #00788b;
	border: none;
	color: white;
	cursor : pointer;
	text-decoration : none;
	border-radius : 6px;
```

```
        font-size :100%;
        font-family: Palatino Linotype;
        -webkit-transition-duration : 0.4s; /* Safari */
        transition-duration: 0.4s;
        padding-bottom: 1%;
        padding-top: 1%;
        width: 80%;
        height: 70%;

        }

footer {
        position: relative;
        bottom:0px;
        width: 100%;
        text-align: center;
        margin-top: 8%;
        height:60px;
        color:#d9d9d9;
}

fieldset
{
        margin-bottom: 4%;
        border:1px solid #00788b;
        border-radius:8px;
        background-color:#dfdfdf;
        margin-top: 2%;
        margin-bottom:0%;
        width:50%;
        padding-left:10%;
        padding-top: 0%;
        padding-right: 10%;
        pading-bottom:0%;
}

select
{   padding-left:30%;
        border: solid 1px #00788b;
        width:90%;
        border-radius:3px;
        height: 100%;

}

input[type=text],input[type=password],input[type=date],input[type=email],te
xtarea
{
        width: 80%;
        text-align:center;
        color: #00001a;
        font: normal 18px Georgia, Times, serif;
        padding-left:10%;

}
#button1
{
        position:absolute;
        left:21.7%;
        width:56.5%;
        height: 10%
```

```
}
H3 {
	padding-right: 50%;
	color: #00788b;
	font: normal 54px Georgia, Times, serif;
}
legend{
	color:#00788b;
	font-size:30px;
	font-family:sans-serif;
	text-decoration:bold;
}
td
{
text-align: center;
padding: 5px;
}
#userhideeffect,#passhideeffect
{
	display:none;
}
.warning
{
	color:red;
	padding : 0px;
	font-size: 15px;
}
.validity
{
	height: 20px;
	width: 20px;
	visibility: hidden;
}
.imagebutton
{
	background-color: Transparent;
    background-repeat:no-repeat;
    border: none;
    cursor:pointer;
    overflow: hidden;
    outline:none;

}
.imagebutton:hover .eyepassword
{
   filter:opacity(100%);
}
.imagebutton:hover .imagebuttonImage
{
   filter:opacity(100%);
}
.eyepassword,.imagebuttonImage
{
	filter:opacity(55%);
	height: 25px;
	width: 25px;

}
#otptd,#emailotptd,#foremail,#otptd1
{
	display: none;
```

```
}
.loading
{
      height: 20px;
      width: 20px;
      visibility:hidden;
}

#searchtrain-box{
      display:none;
      width:86%;
    height:120px;
      background-color:#fefefe;
      border:solid 1px #00788b;
      border-radius:3px;
      overflow-y:auto;
      margin-left:7%;

}


#search-train{
      top:0;
      position:relative;
      background-color:#dfdfdf;

}
input:focus,textarea:focus{

      box-shadow:1px 2px 1px 1px #00788b;

}
/*-------------------------------------------------------------------------
-----------------------------------*/


.routestation li {
  padding-bottom: 40px;
  position:relative;
}

.routestation li span{
  padding: 3px 14px;
  border-radius: 50%;
  margin-right: 10px;
  background:#00788b;
  color:black;

}
.routestation li span:before{
  content:'';
  position:absolute;
  border:1px solid #00788b ;
  left:14px;
  height:100%;
  color:black;
}

.routestation li:last-child span:before{
 content:none;
}
```

```
/*-------------------------------------------------------------------------
-----------------------------------*/
.train_class_container{
width:100%;
height:100%;
top:40px;
position:relative;
border-spacing:30px;
table-layout:fixed;
}


.train_class{
height:180px;
width:100%;
background: linear-gradient(to bottom right, #dfdfdf 50px, #00788b 160px);
box-shadow:0 12px 10px 0 rgba(0, 0, 0, 0.24), 0 10px 50px 0
            rgba(0, 0, 0, 0.19);
color:#fefefe;
text-align:center;
}
.train_class_know_more{
    z-index: 1;
      background-color: rgba(0,0,0,0.4);
      width:100%;
      height:160px;
      display:none;
}
/*-------------------------------------------------------------------------
----------------*/
.available-seats-display{
      width:80%;
      top:60px;
      position:relative;
      font-family:fantasy,serif,sans-serif;
      font-size:20px;
}
.available-seats-display td{
      text-align:left;
      width:18%;

      }

.available-seats-display th{
      color:#fefefe;
      border-radius:5px;
      background:linear-gradient(#00788b,#002c33);
      text-transform: uppercase;


}


.available-seats-display .seat-icon{
  background-image:url(../images/seat.png),linear-
gradient(#00788b,#002c33);
  background-position:left;
      background-repeat:no-repeat;
      background-size:contain;
  text-transform: uppercase;
```

```
  padding-left:4px;
  width:23%;

}

.available-seats-display tr:nth-child(even){background-color:#fefefe}
.available-seats-display tr:nth-child(odd){background-color: #dfdfdf}
.available-seats-display tr td:nth-child(1) {background:linear-
gradient(#00788b,#002c33);color:#fefefe;
                        text-transform: uppercase;border-radius:5px;}

.back-icon{
      background-image:url(../images/back.png);
      background-position:left;
      background-repeat:no-repeat;
      background-size:contain;

}
.quota_class_container{
width:100%;
height:100%;
top:40px;
position:relative;
border-spacing:40px;
table-layout: fixed;

}

.quota_class{
height:180px;
width:100%;
box-shadow:0 12px 10px 0 rgba(0, 0, 0, 0.24), 0 15px 50px 0
            rgba(0, 0, 0, 0.19);
color:#fefefe;
font-size:28px;
text-align:center;
overflow:hidden;
position:relative;

}
.quota_class_know_more{
    z-index: 1;
      background-color:rgba(0,0,0,0.4);
       position:absolute;
         left: 0;
      bottom: 0;
      width:100%;
      height:140px;
      display:none;
      font-size:20px;

}
/*-------------------------------------------------------------------------
------------------------------*/

.addcustomer-modal {

    display:none;
    z-index: 90;
    position: fixed;
    left: 0;
```

```
    top: 0;
      bottom: 0;
    right:0;
    font:25px fantasy,serif,sans-serif;
    background-color: rgba(0,0,0,0.4);
      width: 100%;
    height: 100%;


    }

.addcustomer-modal-box {
      position:fixed;
    left: 0;
    top:20%;
      right:0;
    background-color: #fefefe;
    margin: 0 auto;
    height:70%;
    border: 1px solid #999999;
    width: 60%;
    overflow:hidden;
}

.addcustomer-modal-header{
text-align:center;
height:6.4%;
width:100%;
left:0;
top:0;
background:#cccccc;
}

#addcustomer-modal-content{
      position:relative;
      height:100%;
    font: 23px fantasy,serif,sans-serif;
    vertical-align: middle;
    width: 100%;
    background:linear-gradient(#cccccc,#fefefe);
      overflow:auto;
      }


.addcustomer-display{
      top:60px;
      left:75px;
      width:90%;
      position:relative;
      font:25px fantasy,serif,sans-serif;
      table-layout: fixed;
      }


      .addcustomer-display th{
      color:#fefefe;
    border-radius:5px;
    background:linear-gradient(#00788b,#005866,#00424d);

}
```

```
.addcustomer-display tr:nth-child(even){background-color:#fefefe}
.addcustomer-display tr:nth-child(odd){background-color:#e6e6e6}
.addcustomer-display tr td:nth-child(1) {
      background:linear-gradient(#00788b,#002c33);
      color:#fefefe;
    border-radius:5px;}

.addcustomerprice-display{
      top:150px;
      left:70.5%;
      width:25%;
      position:relative;
      font:25px fantasy,serif,sans-serif;
      table-layout: fixed;
      }

.addcustomerprice-display tr:nth-child(even){background-color:#fefefe}
.addcustomerprice-display tr:nth-child(odd){background-color:#e6e6e6}
.addcustomerprice-display tr td:nth-child(1) {
      background:linear-gradient(#00788b,#002c33);
      color:#fefefe;
    border-radius:5px;}
/*-------------------------------------------------------------------------
----------------------------*/
#pdfContainer{
      width: 70%;
      height: 350px;
      position:relative;
}
#buttonsContainer{
      display:none;
      position:absolute;
      background-color: rgb(249,249,249);
    left: 0%;
    top: 0%;
    width: 100%;
    height: 100%;
    z-index: 2;
    opacity:0.8;
}
#iframePDF{
      width: 100%;
      height: 350px;

}
.sendMailNowLoading{
    display:none;
    position:fixed;
    left: 0px;
    top: 0px;
    width: 100%;
    height: 100%;
    z-index: 80;
    background: url('../images/sendmailloading.gif') 50% 50% no-repeat;
    background-size: 300px 200px;
    opacity:0.8;

}
```

```
/*-------------------------------------------------------------------------
---------------------------*/
 .cancellation{
      width:90%;
      font-family: fantasy,serif,sans-serif;
      }
 .cancellation th{
      color:#fefefe;
      border-radius:3px;
      background:linear-gradient(#00788b,#002c33);
      text-transform: uppercase;
            }

.cancellation tr:nth-child(even){background-color:#fefefe}
.cancellation tr:nth-child(odd){background-color: #dfdfdf}
.cancellation td{
      text-align:center;
      width:15%;

      }

/*-------------------------------------------------------------------------
-------------------------------*/
#modal-content #displayPassengerDetailsModel {
    border-collapse: collapse;
    width: 100%;
}

#modal-content #displayPassengerDetailsModel th, #modal-content
#displayPassengerDetailsModel td {
    padding: 8px;
    text-align: center;
    border-bottom: 1px solid #ddd;
      color:#00788b;
}

#modal-content #displayPassengerDetailsModel tr:hover {background-
color:#f5f5f5;}
/*-------------------------------------------------------------------------
-------------------------------*/

.modalConfirm {

    display:none;
    z-index: 50;
    position: fixed;
    left: 0;
    top: 0;
      bottom: 0;
    right:0;

    background-color: rgba(0,0,0,0.4);
      width: 100%;
    height: 100%;


    }

.modalConfirm-box {

      position:relative;
```

```
    left: 0;
    top:20%;
      right:0;
      border-radius:20px 5px 20px 5px;
    background-color: #fefefe;
    margin: 0 auto;
    height:40%;
    border: 1px solid #999999;
    width: 35%;
    overflow:hidden;
}

.modalConfirm-header{
border-radius:19px 5px 0px 0px;
text-align:center;
height:12%;
width:100%;
position:relative;
left:0;
top:0;
background-color:#00788b;
font: 20px Palatino Linotype sans-serif Lucida Fax;
color:white;

}
.modalConfirm-footer {
border-radius:0px 0px 19px 5px;
text-align:center;
height:13%;
width:100%;
position:relative;
left:0;
bottom:0;
background-color:#c0c0c0;
font: 20px Palatino Linotype sans-serif Lucida Fax;

}

#modalConfirm-content{
      position:relative;
      height:67%;
    font: 20px Palatino Linotype sans-serif Lucida Fax;
    border-color:#c0c0c0;
    vertical-align: middle;
    width: 100%;
    overflow:auto;
      }
/*-------------------------------------------------------------------------
-----*/
 .chat-screen{
  width:80%;
  background-color:#fefefe;
  position:relative;
  font-family: fantasy;
  left:2.5%;
  }

  #chat-header{
  height:150px;
  background:linear-gradient(#a6a6a6,#dfdfdf);
```

```
}

#chat-content{
height:540px;
font-family:fantasy;
width:69.7%;
  top:20px;
  position:relative;
background-color:#e6e6e6;
float:right;

}

#Container-dark{
border: 1px solid #00788b;
  background-color: #f1f1f1;
  border-radius:5px 5px 5px 35px;
  padding: 10px;
  width:50%;
  margin: 12px 0;
  float:right;
  top:10px;
  position:relative;
  text-align:right;
}
#Container{
border: 1px solid #00788b;
  background-color:#fefefe;
  border-radius:  5px 35px 5px 5px;
  padding: 10px;
  width:50%;
  float:left;
  top:10px;
  position:relative;
  margin: 12px 0;
  text-align:left;
}



#vertical-bar{
width:30%;
top:20px;
position:relative;
float:left;
background-color:#dfdfdf;
}

#vertical-bar a{
  color:#00788b;
  display: block;
  padding: 25px;
  text-decoration:none;
  font-size:25px;

}


#vertical-bar a:hover{
background-color:#00788b;
```

```
  color:#fefefe;
  }
  #message-content{
  width:100%;
  overflow-y:auto;
  height:410px;
  }
  #message-header{
  background:linear-gradient(#00788b,#005866);
  width:100%;
  }
  #message-footer{
   height:65px;
   background-color:#cccccc;
   top:0px;
   text-align:center;
   position:relative;
  }
  .chat-name{
  color:#fefefe;
  font-size:30px;
  }
  #ack{
  color:#00424d;
 position:relative;

  font-weight: bold;
  }
/*--------------------------------------------------------------------------
-----*/
.dropdown {
      position:absolute;
    display:none;
    right:0px;
    background-color: #f9f9f9;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    width:17%;
    z-index:35;
    border-radius:6px;

}

.dropdown a {
    padding: 12px 16px;
    display:block;
    text-align:left;
    color:#404040;
    text-decoration:none;
}

.dropdown a:hover {background-color: #dfdfdf;color:#00788b}
/*--------------------------------------------------------------------------
-----*/

 #general-info{
  width:80%;
  background-color:#dfdfdf;
  left:10%;
  position:relative;
  font-family:fantasy
  }
```

```
  #about-content{
  left:10%;
  right:10%;
  position:relative;
  width:80%
  }
  #contact{
  height:420px;
  width:60%;
  left:19%;
  position:relative;
  top:40px;
  background:linear-gradient(#dfdfdf,#b3b3b3);
  font-family: fantasy;
  }
  #contact td{
 text-align:left}
/*--------------------------------------------------------------------------
------------------------------*/
#news-container{
 width:85%;
 left:7%;
 position:relative;
 top:40px;
 font-family: fantasy;
 }

 #news-block{
 padding:10px;
 background:linear-gradient(#cccccc,#dfdfdf,#f2f2f2);
 margin: 35px 0;
 border:1px solid #d4d4d4;
 border-radius:5px;
 }

 #news-image{
 height:200px;
 float:left;
 width:30%
 }

 #news-content{
 text-align:left;
 min-height: 180px;
 }
 #news-header{
 color:#00788b;
 font-size:25px;
 text-decoration: underline;}

 .news-image{
 height:200px;
 border:1px solid #000000
 }
```

ImageForm.css

```
.calendar-icon{
      background-image:url(../images/calendar.png);
      background-position:left;
      background-repeat:no-repeat;
      padding-left:30px;
      background-size:contain;

      }

.train-icon{
      background-image:url(../images/Train.png);
      background-position:left;
      background-repeat:no-repeat;
      background-size:contain;
      padding-left:30px;
}


.source-icon{
      background-image:url(../images/source.png);
      background-position:left;
      background-repeat:no-repeat;
      background-size:contain;
      padding-left:30px;

}
.email-icon{
      background-image:url(../images/email.png);
      background-position:left;
      background-repeat:no-repeat;
      background-size:contain;
      padding-left:30px;

}

.password-icon{
      background-image:url(../images/password.png);
      background-position:left;
      background-repeat:no-repeat;
      background-size:contain;
      padding-left:30px;
}

.user-icon{
      background-image:url(../images/user.png);
      background-position:left;
      background-repeat:no-repeat;
      background-size:contain;
      padding-left:30px;

}
::-ms-clear {
  display: none;
}
```

2. **Servlet pages :**

DatabaseConnection.java

```
package railwayFrequentFunctions;

import java.sql.Connection;
import java.sql.DriverManager;

public class DatabaseConnection {
   private static Connection con=null;
   public final String getDriver(){
        return "com.mysql.jdbc.Driver";
   }
   public final String getURL(){
        return "jdbc:mysql://localhost:3306/railway_database";
   }
   public final String getUsername(){
        return "railway_project";
   }
   public final String getPassword(){
        return "railway";
   }
   public Connection dbConnect()
   {
        try {
                Class.forName(getDriver());
                con = DriverManager.getConnection(
                                getURL(), getUsername(),
                                getPassword());
                } catch (Exception e) {
                e.printStackTrace();
                }
   return con;
   }
}
```

DatabasePrice.java

```
package railwayFrequentFunctions;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

public class DatabasePrice {
   public double getTrainRoutePrice(String fromId, String toId,
                String routeId, String trainNo, String trainReturn, String classId)
{
        Double trainCostWithoutQuota = 0.0;
        try {
                Connection con = (new DatabaseConnection()).dbConnect();
                String trainCostQuery = "SELECT cost_per_km FROM
train_master WHERE train_no=? and train_return=?";
                PreparedStatement ps = con.prepareStatement(trainCostQuery);
                ps.setInt(1, Integer.parseInt(trainNo));
                ps.setInt(2, Integer.parseInt(trainReturn));
                ResultSet rs = ps.executeQuery();
                rs.next();
                Double traincost = rs.getDouble(1);

                String totalFromToRouteDistanceQuery = "select
sum(distance_from_prev) from route_master where route_id=? and "
                                + "counter>(select counter from route_master
where route_id=? and station_code=?) and "
                                + "counter<=(select counter from route_master
where route_id=? and station_code=?)";
                ps = con.prepareStatement(totalFromToRouteDistanceQuery);
                ps.setInt(1, Integer.parseInt(routeId));
                ps.setInt(2, Integer.parseInt(routeId));
                ps.setInt(4, Integer.parseInt(routeId));
                ps.setString(3, fromId);
                ps.setString(5, toId);
                rs = ps.executeQuery();
                rs.next();
                Double totalfromtoroutedistance = rs.getDouble(1);
                Double trainCostWithoutQuotaAndClass = traincost
                                * totalfromtoroutedistance;
```

```
                String classCostQuery = "SELECT additional_cost FROM
train_class_cost WHERE train_no=? and "
                                + "train_return=? and class_id=?";
                ps = con.prepareStatement(classCostQuery);
                ps.setInt(1, Integer.parseInt(trainNo));
                ps.setInt(2, Integer.parseInt(trainReturn));
                ps.setString(3, classId);
                rs = ps.executeQuery();
                rs.next();
                Double classCostPercentage = rs.getDouble(1);

                trainCostWithoutQuota = trainCostWithoutQuotaAndClass
                                +          (trainCostWithoutQuotaAndClass          *
(classCostPercentage / 100));

                con.close();
        } catch (Exception e) {
                e.printStackTrace();
        }

        return trainCostWithoutQuota;
    }

    public String formatCurrency(double price) {
        String priceWithDecimalString = String.valueOf(price);
        String[] priceArray = priceWithDecimalString.split("\\.");
        String priceString = priceArray[0];
        String formatPrice = "";
        int counter = 0;
        for (int i = priceString.length() - 1; i >= 0; i--) {
                if ((counter - 3) >= 0 && (counter - 3) % 2 == 0) {
                        formatPrice = "," + formatPrice;
                }
                char priceAtLocation = priceString.charAt(i);
                formatPrice = priceAtLocation + formatPrice;
                counter++;
        }
        String decimall = "";
        if (priceArray.length > 1) {
                decimall = "." + priceArray[1];
        }
```

```
        return "\u20B9" + formatPrice + decimall;
    }

    public double getPriceWithoutTax(String fromid, String toid,
            String routeid, String trainno, String trainreturn, String classid,
            String quotaid)  {
        double price=0.0;
        try {
            double priceWithoutQuota = getTrainRoutePrice(fromid, toid,
routeid,
                    trainno, trainreturn, classid);
            Connection con = (new DatabaseConnection()).dbConnect();
            String query = "SELECT price_discount FROM quota_details
where quota_id=?";
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, quotaid);
            ResultSet rs = ps.executeQuery();
            rs.next();
            double quotaDiscount = rs.getDouble(1);
            price = priceWithoutQuota
                    - (priceWithoutQuota * quotaDiscount / 100);
            price = Math.round(price * 10) / 10D;
            con.close();

        } catch (Exception e) {
            // TODO: handle exception
        }
                return price;
    }

    public JSONArray getPriceWithTax(double priceWithoutTax) {
        JSONArray jarr = new JSONArray();
        try {
            Connection con = (new DatabaseConnection()).dbConnect();
            String taxQuery = "SELECT tax_id,tax_price from tax;";
            PreparedStatement ps = con.prepareStatement(taxQuery);
            ResultSet rs = ps.executeQuery();
            double sumTax = 0.0;

            JSONObject jobj = new JSONObject();
            while (rs.next()) {
                jobj = new JSONObject();
```

```
                    jobj.put("taxid", rs.getString(1));
                    jobj.put("taxpercent", rs.getDouble(2));
                    jarr.add(jobj);
                    sumTax+=rs.getDouble(2);
                }
                double priceWithTax = priceWithoutTax
                        + (priceWithoutTax * sumTax / 100);
                jobj = new JSONObject();
                jobj.put("pricewithtax",  Math.round(priceWithTax  *  100D)  /
100D);
                jarr.add(jobj);
                con.close();

        } catch (Exception e) {
                // TODO: handle exception
        }
        return jarr;
    }
}
```

DateFunctions.java

```
package railwayFrequentFunctions;

/**
 * @author Rahul Mukesh Singh
 *
 */
import java.text.DateFormat;
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
public class DateFunctions {
public String dateforMinAge(int minAge)
{
   DateFormat dateFormat = new SimpleDateFormat("MM-dd");
   DateFormat adult=new SimpleDateFormat("yyyy");
   Date date = new Date();
   int adultMinYear=Integer.parseInt(adult.format(date))-minAge;
   return (adultMinYear+"-"+dateFormat.format(date));
}
public boolean isTatkalAvailable(String sdate){
   boolean tatkalAvailable=false;
   sdate+=" 10:00:00";
   try {
        SimpleDateFormat format=new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
        Date startTimeStamp=format.parse(sdate);
        long nowww=System.currentTimeMillis();
        long startTrainTime=startTimeStamp.getTime();
        long range=1000*60*60*24; // 1 day
        if((startTrainTime-nowww) <= range){
              tatkalAvailable=true;
        }
   } catch (ParseException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
   }
   return tatkalAvailable;
}}
```

DateTimeOfStation.java

```
package railwayFrequentFunctions;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
```

```
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

public class DateTimeofStation {
   public JSONArray getDateAndTimeOfStation(String src, String dest,
                String tno, String treturn, String sdate, String rid) {
        Connection con;
        Date d;
        Calendar calendar = Calendar.getInstance();
        JSONArray data = new JSONArray();

        int trainno = Integer.parseInt(tno);
        int trainreturn = Integer.parseInt(treturn);
        int routeid = Integer.parseInt(rid);
        String selectedDate = sdate;
        try {
                con = (new DatabaseConnection()).dbConnect();
                String routedetails = "SELECT (SELECT station_name FROM
station_master sr where "
                                + "sr.station_code = rm.station_code) AS
station_name,distance_from_prev,station_code from route_master rm where "
                                + "rm.station_code IN (SELECT station_code
FROM route_master WHERE route_id=? "
                                + "AND counter >= (SELECT counter from
route_master where route_id=? "
                                + "and station_code=(SELECT source_station_code
FROM `train_master` WHERE train_no=? AND train_return=?)) "
                                + "AND counter <= (SELECT counter from
route_master where route_id=? "
                                + "and station_code=(SELECT dest_station_code
FROM train_master WHERE train_no=? AND train_return=?)) "
                                + "ORDER BY route_id,counter) AND route_id =
?;";
                String traindetails = "SELECT train_speed,train_time FROM
train_normal_schedule"
                                + " WHERE train_no=? AND train_return=?";
                Double speed = 0.0, distance = 0.0;
                int timeinsecs = 0;
                String timestamp = "";
                String stationName = "";
```

```
PreparedStatement ps = con.prepareStatement(traindetails);
ps.setInt(1, trainno);
ps.setInt(2, trainreturn);
ResultSet rs = ps.executeQuery();
if (rs.next()) {
        speed = rs.getDouble("train_speed");
        timestamp = rs.getString("train_time");
}

timestamp = selectedDate + " " + timestamp;

ps = con.prepareStatement(routedetails);
ps.setInt(1, routeid);
ps.setInt(2, routeid);
ps.setInt(3, trainno);
ps.setInt(4, trainreturn);
ps.setInt(5, routeid);
ps.setInt(6, trainno);
ps.setInt(7, trainreturn);
ps.setInt(8, routeid);

rs = ps.executeQuery();
int loopCounter = 0;
double sumDistance = 0.0;
while (rs.next()) {
        stationName = rs.getString("station_name");
        if (loopCounter == 0) {
                distance = 0.0;
        } else {
                distance = rs.getDouble("distance_from_prev");
        }
        sumDistance += distance;
        timeinsecs = (int) (Math.round((distance / speed) *
3600));
        d = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
                        .parse(timestamp);
        calendar.setTime(d);
        calendar.add(Calendar.SECOND, timeinsecs);
        timestamp = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss")
                        .format(calendar.getTime());
        if (rs.getString("station_code").equals(src)
```

```
                                        || rs.getString("station_code").equals(dest)
                                        || loopCounter == 0) {
                                JSONObject jobj = new JSONObject();
                                jobj.put("stationname", stationName);
                                jobj.put("stationtime", calendar.getTime());
                                jobj.put("distance", sumDistance+" km");
                                data.add(jobj);
                        }
                        loopCounter++;
                }
                JSONObject jobj = new JSONObject();
                jobj.put("stationname", stationName);
                jobj.put("stationtime", calendar.getTime());
                jobj.put("distance", sumDistance+" km");
                data.add(jobj);
                con.close();

        } catch (Exception e) {
                e.printStackTrace();
        }
        return data;
    }
}
```

DeleteFiles.java

```
package railwayFrequentFunctions;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.attribute.BasicFileAttributes;
import java.util.Calendar;
import java.util.Timer;
import java.util.TimerTask;

public class DeleteFiles {

    public static void main(String[] args) {
```

```
        // TODO Auto-generated method stub
        Timer timer = new Timer();
        String                          pathLocation                    =
"E:\\JSP\\RAILWAY_PROJECT\\WebContent\\WEB-
INF\\GeneratedDocuments\\";
        TimerTask timertask = new TimerTask() {


                public void run() {
                        Calendar cal = Calendar.getInstance();
                        int hour = cal.get(Calendar.HOUR_OF_DAY);
                        if (hour == 12) {

                                Path paths;
                                BasicFileAttributes bfa;
                                try {
                                        File folder = new File(pathLocation);
                                        File[] listOfFiles = folder.listFiles();
                                        for (File file : listOfFiles) {
                                                if (file.isFile()) {
                                                        paths                   =
Paths.get(pathLocation + file.getName());
                                                        bfa                     =
Files.readAttributes(paths, BasicFileAttributes.class);
                                                        long
fileTimeInDays=bfa.lastModifiedTime().toMillis()/(60*60*24*1000);
                                                        long
currentTimeInDays=System.currentTimeMillis()/(60*60*24*1000);
                                                        long
periodToKeepFileInDays=31*4; //(31 days * 4 = 4 months)
                                                        if((currentTimeInDays-
fileTimeInDays)>periodToKeepFileInDays){
                                                                if(new
File(pathLocation + file.getName()).exists()){
                                                                new    File(pathLocation
+ file.getName()).delete();
                                                                }
                                                        }
                                                }
                                        }
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
```

```
                                        e.printStackTrace();
                                    }
                                }
                            }
                        };
                        timer.schedule(timertask, 0, 1000*60*60*24);// delay 0 and then run task every 1 day

                    }

                }
```

Emailattachmentrailway.java

```
/**
 *
 */
package railwayFrequentFunctions;

/**
 * @author Rahul Mukesh Singh
 *
 */

import java.util.HashMap;
import java.util.Map;
import java.util.Properties;

import javax.mail.Authenticator;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Message;
import javax.mail.Transport;
import javax.mail.MessagingException;
```

```
import javax.mail.internet.MimeMessage;
import javax.mail.internet.InternetAddress;
import javax.mail.BodyPart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.Multipart;
import javax.mail.internet.MimeMultipart;
import javax.activation.FileDataSource;
import javax.activation.DataSource;
import javax.activation.DataHandler;

import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.io.InputStream;
import java.io.ByteArrayInputStream;

import com.google.zxing.EncodeHintType;
import com.google.zxing.qrcode.decoder.ErrorCorrectionLevel;
import com.itextpdf.text.Document;
import com.itextpdf.text.Image;
import com.itextpdf.text.pdf.PdfWriter;
import com.itextpdf.tool.xml.XMLWorkerHelper;

public class emailattachmentrailway {
   public boolean createattachandemail(String to,  String subject,
                  String attachfilename) {
         boolean result;
         PersonalData ed = new PersonalData();
         String from = ed.user;
         String host = "smtp.gmail.com";
         String port = "587";
         String FromName = "RAILWAY";
         String                                    filename                                    =
"E:\\JSP\\RAILWAY_PROJECT\\WebContent\\WEB-
INF\\GeneratedDocuments\\" +
         to.replace(".", "")+ ".pdf";

         final String user = ed.user;
         final String pass = ed.pass;

         Properties properties = System.getProperties();
         // Setup mail server
```

```
properties.setProperty("mail.smtp.host", host);
properties.setProperty("mail.smtp.port", port);
properties.setProperty("mail.smtp.auth", "true");
properties.setProperty("mail.smtp.starttls.enable", "true");
// creates a new session with an authenticator
Authenticator authenticator = new Authenticator() {
        public PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(user, pass);
        }
};
// Get the default Session object.
Session mailSession = Session.getInstance(properties, authenticator);

try {

        MimeMessage message = new MimeMessage(mailSession);// Create-a-default-MimeMessage-object.
        message.setFrom(new InternetAddress(from, FromName));// Set-From:-header-field-of-the-header.
        message.addRecipient(Message.RecipientType.TO, new InternetAddress(
                        to));// Set-To:-header-field-of-the-header.

        message.setSubject(subject);// Set-Subject:-header-field
        BodyPart messageBodyPart = new MimeBodyPart();
        String bpart = "<center><h3 style='color:#00788b'>"
                        + "[Auto-Generated Mail with Attachment and PDF]</h3></center>";
        messageBodyPart.setContent(bpart, "text/html");
        Multipart multipart = new MimeMultipart();// Create-a-multipart-message
        multipart.addBodyPart(messageBodyPart);// Set-text-message-part
        messageBodyPart = new MimeBodyPart();// Part-two-is-attachment
        DataSource source = new FileDataSource(filename);
        messageBodyPart.setDataHandler(new DataHandler(source));
        messageBodyPart.setFileName(attachfilename + ".pdf");
        multipart.addBodyPart(messageBodyPart);
        message.setContent(multipart);// Send-the-complete-message-parts
```

```
                Transport.send(message);// Send-message
                result = true;
        } catch (MessagingException mex) {
                result = false;

        } catch (Exception e) {
                result = false;
                System.out.println("attach : "+e);

        }
        return result;
    }
}
```

Mailnow.java

```
/**
 *
 */
package railwayFrequentFunctions;

/**
 * @author Rahul Mukesh Singh
 *
 */
import java.util.Properties;
import javax.mail.Authenticator;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Message;
import javax.mail.Transport;
import javax.mail.MessagingException;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.InternetAddress;
import java.io.UnsupportedEncodingException;
public class mailnow {
public boolean MailNowMsg(String to,String msg,String subject) throws
UnsupportedEncodingException
    {
       boolean result;
```

```
PersonalData ed = new PersonalData();
String from = ed.user;
String host = "smtp.gmail.com";
String port="587";
String FromName="RAILWAY";
final String user=ed.user;
final String pass=ed.pass;
Properties properties = System.getProperties();
properties.setProperty("mail.smtp.host", host);
properties.setProperty("mail.smtp.port", port);
   properties.setProperty("mail.smtp.auth", "true");
   properties.setProperty("mail.smtp.starttls.enable", "true");
   // creates a new session with an authenticator
   Authenticator authenticator = new Authenticator() {
         public PasswordAuthentication getPasswordAuthentication() {
               return new PasswordAuthentication(user, pass);
         }
   };
// Get the default Session object.
Session mailSession = Session.getInstance(properties,authenticator);

try {
   // Create a default MimeMessage object.
   MimeMessage message = new MimeMessage(mailSession);

   // Set From: header field of the header.
   message.setFrom(new InternetAddress(from,FromName));

   // Set To: header field of the header.
   message.addRecipient(Message.RecipientType.TO,                    new
InternetAddress(to));
//          message.addRecipient(Message.RecipientType.TO,                    new
InternetAddress(to1));
   // Set Subject: header field
   message.setSubject(subject);

   // Send the actual HTML message, as big as you like
   message.setContent(msg, "text/html" );

   // Send message
   Transport.send(message);
   result = true;
```

```
        }
        catch (MessagingException mex) {
          result = false;
          System.out.println("Non Attach : "+mex);
        }
        return result;
      }}
```

Package-info.java

```
/**
 *
 */
/**
 * @author Rahul Mukesh Singh
 *
 */
                        package railwayFrequentFunctions;
```

pdfGeneration.java

```
/**
 *
 */
package railwayFrequentFunctions;

/**
 * @author Rahul Mukesh Singh
 *
 */

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Map;
```

```
import com.google.zxing.EncodeHintType;
import com.google.zxing.qrcode.decoder.ErrorCorrectionLevel;
import com.itextpdf.text.Document;
import com.itextpdf.text.Image;
import com.itextpdf.text.pdf.PdfWriter;
import com.itextpdf.tool.xml.XMLWorkerHelper;

public class pdfGeneration {
   public pdfGeneration(String filename, String msg,String seatMapLocation,
String qrCodeData,
                String qrLocation,String csss,int keyShift) throws IOException {
        File                                                    directory=new
File("E:\\JSP\\RAILWAY_PROJECT\\WebContent\\WEB-
INF\\GeneratedDocuments");
        if(!(directory.exists())){
                directory.mkdirs();
        }
        if (new File(filename).exists()) {
                new File(filename).delete();
        }
        if(new File(qrLocation).exists()){
                new File(qrLocation).delete();
        }
        OutputStream file = null;
        Document document = null;
        String
imgLocation="E:\\JSP\\RAILWAY_PROJECT\\WebContent\\images\\lo
go.png";
        try {
                File fileloc = null;

                fileloc = new File(filename);


                file = new FileOutputStream(fileloc);
                document = new Document();
                PdfWriter writer = PdfWriter.getInstance(document, file);
                document.open();

                // ---------------------------------ADD_LOGO_IMAGE-----------------
-----------------------
```

```
                Image img = Image.getInstance(imgLocation);

                img.scaleAbsolute(100, 90);
                img.setAbsolutePosition(document.right()-60,    document.top()-
45);
                document.add(img);
                // ---------------------------------ADD_DATA_TABLE------------------
----------------------
                InputStream                is                =                new
ByteArrayInputStream(msg.getBytes(StandardCharsets.UTF_8));
                InputStream cs = new ByteArrayInputStream(csss.getBytes());
                XMLWorkerHelper.getInstance().parseXHtml(writer, document,
is, cs,Charset.forName("UTF-8"));
                //  ---------------------------------ADD_SEAT_MAP----------------------
----------------------
                img = Image.getInstance(seatMapLocation);
                img.setAlignment(Image.ALIGN_CENTER);
                img.scaleAbsolute(300, 400);
                document.add(img);
                //  ---------------------------------ADD_QR_CODE-----------------------
--------------------


                String charset = "UTF-8"; // or "ISO-8859-1"
                Map hintMap = new HashMap();
                hintMap.put(EncodeHintType.ERROR_CORRECTION,
ErrorCorrectionLevel.L);
                (new    qrGeneration()).createQRCode(qrCodeData,    qrLocation,
charset,
                                hintMap, 100, 100,keyShift);
                img = Image.getInstance(qrLocation);
                img.setAlignment(Image.ALIGN_CENTER);
                document.add(img);
                //-------------------------------------------------------------------------
-------

        } catch (Exception e) {

        } finally {
                document.close();
                file.close();
                if(new File(qrLocation).exists()){
```

```
                    new File(qrLocation).delete();
                }
            }
        }
    }
```

PNR.java

```
package railwayFrequentFunctions;

public class PNR {
public String getPNR(){
      long nowww=System.currentTimeMillis() % (long)(1000000000000.0);
      String pnr=String.valueOf(nowww);
      return pnr;
}
}
```

properFormat.pdf

```
package railwayFrequentFunctions;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class properFormatForPDF {
public String[] getProperFormatForPDF(String quotaid,String classId,String
berthType,String coachCounter){
   String[] properFormatForPDFF=new String[4];

   try {
        Connection con=(new DatabaseConnection()).dbConnect();
        PreparedStatement ps;
        ResultSet rs;
        String quotaNameQuery="SELECT quota_name FROM quota_details
WHERE quota_id=?";
```

```
        String classCoachNameQuery="SELECT class_name,coach_code FROM train_class_details WHERE class_id=?";
        String berthNameQuery="SELECT berth_name FROM berth_details WHERE berth_id=?";
        ps=con.prepareStatement(quotaNameQuery);
        ps.setString(1, quotaid);
        rs=ps.executeQuery();
        rs.next();
        properFormatForPDFF[0]=rs.getString(1)+"("+quotaid+")";
        ps=con.prepareStatement(classCoachNameQuery);
        ps.setString(1, classId);
        rs=ps.executeQuery();
        rs.next();
        properFormatForPDFF[1]=rs.getString(1)+"("+classId+")";
        if(!berthType.equalsIgnoreCase("-")){
        properFormatForPDFF[3]=rs.getString(2)+coachCounter;
        ps=con.prepareStatement(berthNameQuery);
        ps.setString(1, berthType);
        rs=ps.executeQuery();
        rs.next();
        properFormatForPDFF[2]=rs.getString(1)+"("+berthType+")";
        }
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
        // TODO: handle exception
    }
    return properFormatForPDFF;
}
}
```

qrGeneration.java

```
/**
 *
 */
package railwayFrequentFunctions;

/**
 * @author Rahul Mukesh Singh
 *
 */

import java.io.File;
import java.io.IOException;
import java.util.Map;
import com.google.zxing.BarcodeFormat;
import com.google.zxing.MultiFormatWriter;
import com.google.zxing.WriterException;
import com.google.zxing.client.j2se.MatrixToImageWriter;
import com.google.zxing.common.BitMatrix;
public class qrGeneration {

	public static void createQRCode(String qrCodeData, String filePath,
			String charset, Map hintMap, int qrCodeheight, int
qrCodewidth,int keyShift)
			throws WriterException, IOException {
			qrCodeData=(new SecretKey()).getSecretKey(qrCodeData, 5);
	BitMatrix matrix = new MultiFormatWriter().encode(
				new String(qrCodeData.getBytes(charset), charset),
				BarcodeFormat.QR_CODE, qrCodewidth, qrCodeheight,
hintMap);

   MatrixToImageWriter.writeToFile(matrix, filePath.substring(filePath
				.lastIndexOf('.') + 1), new File(filePath));
   }

}
```

SeatAllocation.java

```
package railwayFrequentFunctions;
```

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

class quotaSeatStructure {
    private String quotaId;
    private int totalNoOfSeats;

    public quotaSeatStructure(String quotaId, int totalNoOfSeats) {
        this.setQuotaId(quotaId);
        this.setTotalNoOfSeats(totalNoOfSeats);
    }

    public String getQuotaId() {
        return quotaId;
    }

    public void setQuotaId(String quotaId) {
        this.quotaId = quotaId;
    }

    public int getTotalNoOfSeats() {
        return totalNoOfSeats;
    }

    public void setTotalNoOfSeats(int totalNoOfSeats) {
        this.totalNoOfSeats = totalNoOfSeats;
    }
}

class seatsForClass {
    private int coachSeatNo;
    private int coachNo;
    private String berthType;
    private String allotedQuotaId;
```

```
private int coachSeatCounter;

public seatsForClass(int coachNo, int coachSeatNo, String berthType) {
	this.setCoachSeatNo(coachSeatNo);
	this.setCoachNo(coachNo);
	this.setBerthType(berthType);
}

public seatsForClass(String allotedQuotaId, int coachNo, int coachSeatNo) {
	this.setCoachSeatNo(coachSeatNo);
	this.setCoachNo(coachNo);
	this.setAllotedQuotaId(allotedQuotaId);
}

public seatsForClass(int coachSeatNo, String berthType) {
	this.setCoachSeatNo(coachSeatNo);
	this.setBerthType(berthType);
}

public seatsForClass(int coachNo, int coachSeatNo, int coachSeatCounter) {
	this.setCoachSeatNo(coachSeatNo);
	this.setCoachNo(coachNo);
	this.setCoachSeatCounter(coachSeatCounter);
}

public int getCoachSeatNo() {
	return coachSeatNo;
}

public void setCoachSeatNo(int coachSeatNo) {
	this.coachSeatNo = coachSeatNo;
}

public int getCoachNo() {
	return coachNo;
}

public void setCoachNo(int coachNo) {
	this.coachNo = coachNo;
}

public String getBerthType() {
```

```
        return berthType;
    }

    public void setBerthType(String berthType) {
        this.berthType = berthType;
    }

    public String getAllotedQuotaId() {
        return allotedQuotaId;
    }

    public void setAllotedQuotaId(String allotedQuotaId) {
        this.allotedQuotaId = allotedQuotaId;
    }

    public int getCoachSeatCounter() {
        return coachSeatCounter;
    }

    public void setCoachSeatCounter(int coachSeatCounter) {
        this.coachSeatCounter = coachSeatCounter;
    }

}

public class SeatAllocation {
    private Connection con;
    private int noOfCoaches = 0, noOfConfirmSeatsInPerClass = 0,
                noOfRACSeatsInPerClass = 0;
    private int trainno, trainreturn, routeid;
    private String classid, fromid, toid;
    private DatabaseConnection db = new DatabaseConnection();
    private PreparedStatement ps;
    private ResultSet rs;
    private int totalNoConfirmSeats = 0, totalNoRACSeats = 0;
    private ArrayList<quotaSeatStructure> totalQuotaSeatsList = new
ArrayList<quotaSeatStructure>();
    private ArrayList<seatsForClass> totalConfirmSeatsFromMiddle = new
ArrayList<seatsForClass>();
    private ArrayList<seatsForClass> totalRACSeatsFromMiddle = new
ArrayList<seatsForClass>();
```

```
   private ArrayList<quotaSeatStructure> availableQuotaSeatsList = new
ArrayList<quotaSeatStructure>();
   private ArrayList<seatsForClass> availableConfirmSeatsFromMiddle =
new ArrayList<seatsForClass>();
   private ArrayList<seatsForClass> availableRACSeatsFromMiddle = new
ArrayList<seatsForClass>();
   private int availableNoOfRACSeats = 0, totalWL = 0, totalTQWL = 0,
                  availableWL = 0, availableTQWL = 0;
   private Timestamp startTimeStampSql;

   public SeatAllocation(String trainno, String trainreturn, String classid,
                  String routeid, String fromid, String toid,String date) {
         this.trainno = Integer.parseInt(trainno);
         this.trainreturn = Integer.parseInt(trainreturn);
         this.routeid = Integer.parseInt(routeid);
         this.classid = classid;
         this.fromid = fromid;
         this.toid = toid;
         try {
                  con = db.dbConnect();
                  String noOfCoachesQuery = "SELECT no_of_class FROM
train_coach_details WHERE class_id=? and train_no=?";
                  ps = con.prepareStatement(noOfCoachesQuery);
                  ps.setString(1, this.classid);
                  ps.setInt(2, this.trainno);
                  rs = ps.executeQuery();
                  rs.next();
                  noOfCoaches = rs.getInt(1);
                  String startTimeStampQuery="SELECT train_time FROM
train_normal_schedule WHERE train_no=? "
                                    + "AND train_return=?";
                  ps=con.prepareStatement(startTimeStampQuery);
                  ps.setInt(1,this.trainno);
                  ps.setInt(2,this.trainreturn);
                  rs=ps.executeQuery();
                  rs.next();
                  String startTimeStamp=date+" "+rs.getString(1);
                  //System.out.println(startTimeStamp);
                  SimpleDateFormat format = new SimpleDateFormat("yyyy-
MM-dd HH:mm:ss");
                  Date startDateTimeStamp=format.parse(startTimeStamp);
                  //System.out.println("sa "+startDateTimeStamp);
```

```
                startTimeStampSql=new
Timestamp(startDateTimeStamp.getTime());
                con.close();
                getCNFTotalSeats();
                getRACTotalSeats();
                getWLTotalSeats();
        } catch (Exception e) {
                e.printStackTrace();
        }
    }

    public ArrayList<seatsForClass> arrayListExclude(
                ArrayList<seatsForClass> totalList,
                ArrayList<seatsForClass> excludeThisList) {
        ArrayList<seatsForClass> excluded = new ArrayList<seatsForClass>();
        boolean found = false;
        for (seatsForClass tl : totalList) {
                found = false;
                for (seatsForClass etl : excludeThisList) {
                        if (tl.getCoachNo() == etl.getCoachNo()
                                        &&          tl.getCoachSeatNo()          ==
etl.getCoachSeatNo()) {
                                found = true;
                                break;
                        }
                }
                if (!found) {
                        excluded.add(new seatsForClass(tl.getCoachNo(), tl
                                        .getCoachSeatNo(), tl.getBerthType()));
                }
        }
        return excluded;
    }

    public ArrayList<quotaSeatStructure> arrayListDeduct(
                ArrayList<quotaSeatStructure> totalQuotaList,
                ArrayList<seatsForClass> deductQuotaList) {
        ArrayList<quotaSeatStructure>          deducted          =          new
ArrayList<quotaSeatStructure>();
        for (quotaSeatStructure tql : totalQuotaList) {
                deducted.add(new quotaSeatStructure(tql.getQuotaId(), tql
                                .getTotalNoOfSeats()));
```

```
        }
        for (seatsForClass dql : deductQuotaList) {
                for (quotaSeatStructure d : deducted) {
                        if (d.getQuotaId().equals(dql.getAllotedQuotaId())) {
                                d.setTotalNoOfSeats(d.getTotalNoOfSeats() - 1);
                                break;
                        }
                }
        }

        return deducted;
   }

   public void getAvailableQuotaAndConfirmList(
                ArrayList<quotaSeatStructure> totalQuotaSeats,
                ArrayList<seatsForClass> totalMiddleConfirmSeats) {
        ArrayList<seatsForClass> notAvailableSeats = new
ArrayList<seatsForClass>();
        try {
                String NotAvailableSeatsQuery = "SELECT
coach_counter,seat_no,seat_allotted_quota_id from cnf c "
                        + "where (SELECT count(*) from route_master
where route_id=? and "
                        + "(counter BETWEEN (select counter from
route_master where station_code=c.from_station and route_id=?) AND "
                        + "(SELECT counter from route_master where
station_code=c.to_station and route_id=?)) and "
                        + "counter IN (SELECT counter from route_master
where "
                        + "(counter BETWEEN (select counter from
route_master where station_code=? and route_id=?) AND "
                        + "(SELECT counter from route_master where
station_code=? and route_id=?)) AND route_id=?)) > 1 "
                        + "and train_no=? and train_return=? AND
class_id=? AND train_src_time = ?";
                con = db.dbConnect();
                ps = con.prepareStatement(NotAvailableSeatsQuery);
                ps.setInt(1, routeid);
                ps.setInt(2, routeid);
                ps.setInt(3, routeid);
                ps.setString(4, fromid);
                ps.setInt(5, routeid);
```

```
                ps.setString(6, toid);
                ps.setInt(7, routeid);
                ps.setInt(8, routeid);
                ps.setInt(9, trainno);
                ps.setInt(10, trainreturn);
                ps.setString(11, classid);
                ps.setTimestamp(12, startTimeStampSql);
                rs = ps.executeQuery();
                while (rs.next()) {
                        notAvailableSeats.add(new  seatsForClass(rs.getString(3),
rs
                                        .getInt(1), rs.getInt(2)));
                //      System.out.println(rs.getString(3)+"  "+  rs.getInt(1)+"  "+
rs.getInt(2));
                }
                availableConfirmSeatsFromMiddle = arrayListExclude(
                                totalMiddleConfirmSeats, notAvailableSeats);
                availableQuotaSeatsList = arrayListDeduct(totalQuotaSeats,
                                notAvailableSeats);
                con.close();
        } catch (Exception e) {
                e.printStackTrace();
        }

   }

   public ArrayList<seatsForClass> getMiddleSeatsPerClass(
                ArrayList<seatsForClass> seatsInAscendingOrder) {
        int length = seatsInAscendingOrder.size();
        int startIndex = length / 2;
        int minIndex = startIndex - 1;
        int maxIndex = startIndex + 1;
        ArrayList<seatsForClass>        middleSeatsPerClass        =        new
ArrayList<seatsForClass>();

        middleSeatsPerClass.add(seatsInAscendingOrder.get(startIndex));
        for (int i = 0; i < startIndex; i++) {
                if (minIndex >= 0) {

   middleSeatsPerClass.add(seatsInAscendingOrder.get(minIndex));
                        minIndex--;
                }
```

```
            if (maxIndex <= (length - 1)) {

middleSeatsPerClass.add(seatsInAscendingOrder.get(maxIndex));
                    maxIndex++;
            }
    }
    return middleSeatsPerClass;
}

public ArrayList<Integer> getMiddleSeatsPerClassRAC(
            ArrayList<Integer> seatsInAscendingOrder) {
    int length = seatsInAscendingOrder.size();
    ArrayList<Integer> middleSeatsPerClass = new ArrayList<Integer>();
    if (length > 0) {
            int startIndex = length / 2;
            int minIndex = startIndex - 1;
            int maxIndex = startIndex + 1;

middleSeatsPerClass.add(seatsInAscendingOrder.get(startIndex));
            for (int i = 0; i < startIndex; i++) {
                    if (minIndex >= 0) {
                            middleSeatsPerClass

.add(seatsInAscendingOrder.get(minIndex));
                            minIndex--;
                    }
                    if (maxIndex <= (length - 1)) {
                            middleSeatsPerClass

.add(seatsInAscendingOrder.get(maxIndex));
                            maxIndex++;
                    }
            }
    }
    return middleSeatsPerClass;
}

public void getCNFTotalSeats() {

    try {
            con = db.dbConnect();
```

```
String noOfConfirmSeatsInPerClassQuery = "SELECT
seat_no,berth_id FROM "
                + classid + " WHERE berth_id <> ? order by
seat_no";
ps = con.prepareStatement(noOfConfirmSeatsInPerClassQuery);
ps.setString(1, "SLB");
rs = ps.executeQuery();
ArrayList<seatsForClass> seatsForConfirmPerCoach = new
ArrayList<seatsForClass>();
while (rs.next()) {
    seatsForConfirmPerCoach.add(new
seatsForClass(rs.getInt(1), rs
                .getString(2)));
}
noOfConfirmSeatsInPerClass = seatsForConfirmPerCoach.size();
seatsForConfirmPerCoach =
getMiddleSeatsPerClass(seatsForConfirmPerCoach);
for (int i = 1; i <= noOfCoaches; i++) {
    for (seatsForClass seatsDetails :
seatsForConfirmPerCoach) {
        totalConfirmSeatsFromMiddle.add(new
seatsForClass(i,
                seatsDetails.getCoachSeatNo(),
seatsDetails
                        .getBerthType()));
    }
}

totalNoConfirmSeats = totalConfirmSeatsFromMiddle.size();
String getQuotaSeatsQuery = "SELECT quota_id,seats_quota
FROM quota_details order by seats_quota desc";
ps = con.prepareStatement(getQuotaSeatsQuery);
rs = ps.executeQuery();
int quotaSeats = 0;
int remQuotaSeats = totalNoConfirmSeats;
while (rs.next()) {
    String quotaid = rs.getString(1);
    quotaSeats = (int) (Math.floor(totalNoConfirmSeats
            * (rs.getDouble(2) / 100)));
    totalQuotaSeatsList.add(new quotaSeatStructure(quotaid,
            quotaSeats));
    remQuotaSeats -= quotaSeats;
```

```
                }
                for (quotaSeatStructure qs : totalQuotaSeatsList) {
                        if (qs.getQuotaId().equalsIgnoreCase("GN")) {
                                qs.setTotalNoOfSeats(qs.getTotalNoOfSeats()      +
remQuotaSeats);
                        }
                }
                con.close();
                getAvailableQuotaAndConfirmList(totalQuotaSeatsList,
                                totalConfirmSeatsFromMiddle);

        } catch (Exception e) {
                e.printStackTrace();
        }

   }

   public void getRACTotalSeats() {
        ArrayList<Integer>          RACSeatsAscendingOrder          =          new
ArrayList<Integer>();
        ArrayList<Integer>          RACSeatsMiddleOrder          =          new
ArrayList<Integer>();
        try {
                con = db.dbConnect();
                String noOfRACSeatsInPerClassQuery = "SELECT seat_no
FROM "
                                + classid + " WHERE berth_id = ?";
                ps = con.prepareStatement(noOfRACSeatsInPerClassQuery);
                ps.setString(1, "SLB");
                rs = ps.executeQuery();
                while (rs.next()) {
                        RACSeatsAscendingOrder.add(rs.getInt(1));
                }
                noOfRACSeatsInPerClass = RACSeatsAscendingOrder.size();
                RACSeatsMiddleOrder                                          =
getMiddleSeatsPerClassRAC(RACSeatsAscendingOrder);
                for (int i = 1; i <= noOfCoaches; i++) {
                        for (Integer rsmo : RACSeatsMiddleOrder) {
                                for (int j = 1; j <= 2; j++) {
                                        totalRACSeatsFromMiddle.add(new
seatsForClass(i, rsmo,
                                                        j));
```

```
                        }
                    }
                }
                totalNoRACSeats = totalRACSeatsFromMiddle.size();
                getAvailableRACList(totalRACSeatsFromMiddle);
                con.close();
                availableNoOfRACSeats =
availableRACSeatsFromMiddle.size();
        } catch (Exception e) {
                e.printStackTrace();
        }
    }

    public void getAvailableRACList(ArrayList<seatsForClass>
totalMiddleRACSeats) {
        ArrayList<seatsForClass> notAvailableSeats = new
ArrayList<seatsForClass>();
        try {
                String NotAvailableSeatsQuery = "SELECT
coach_counter,seat_no,seat_counter from rac r "
                        + "where (SELECT count(*) from route_master
where route_id=? and "
                        + "(counter BETWEEN (select counter from
route_master where station_code=r.from_station and route_id=?) AND "
                        + "(SELECT counter from route_master where
station_code=r.to_station and route_id=?)) and "
                        + "counter IN (SELECT counter from route_master
where "
                        + "(counter BETWEEN (select counter from
route_master where station_code=? and route_id=?) AND "
                        + "(SELECT counter from route_master where
station_code=? and route_id=?)) AND route_id=?)) > 1 "
                        + "and train_no=? and train_return=? AND
class_id=? AND train_src_time = ?";
                con = db.dbConnect();
                ps = con.prepareStatement(NotAvailableSeatsQuery);
                ps.setInt(1, routeid);
                ps.setInt(2, routeid);
                ps.setInt(3, routeid);
                ps.setString(4, fromid);
                ps.setInt(5, routeid);
                ps.setString(6, toid);
```

```
                ps.setInt(7, routeid);
                ps.setInt(8, routeid);
                ps.setInt(9, trainno);
                ps.setInt(10, trainreturn);
                ps.setString(11, classid);
                ps.setTimestamp(12, startTimeStampSql);
                rs = ps.executeQuery();
                while (rs.next()) {
                        notAvailableSeats.add(new seatsForClass(rs.getInt(1), rs
                                        .getInt(2), rs.getInt(3)));
                }
                con.close();
                availableRACSeatsFromMiddle = arrayListRACExclude(
                                totalMiddleRACSeats, notAvailableSeats);

        } catch (Exception e) {
                e.printStackTrace();
        }

    }

    public ArrayList<seatsForClass> arrayListRACExclude(
                ArrayList<seatsForClass> totalList,
                ArrayList<seatsForClass> excludeThisList) {
        ArrayList<seatsForClass> excluded = new ArrayList<seatsForClass>();
        boolean found = false;
        for (seatsForClass tl : totalList) {
                found = false;
                for (seatsForClass etl : excludeThisList) {
                        if (tl.getCoachNo() == etl.getCoachNo()
                                        && tl.getCoachSeatNo() ==
etl.getCoachSeatNo()
                                        && tl.getCoachSeatCounter() == etl
                                                        .getCoachSeatCounter()) {
                                found = true;
                                break;
                        }
                }
                if (!found) {
                        excluded.add(new seatsForClass(tl.getCoachNo(), tl
                                        .getCoachSeatNo(),
tl.getCoachSeatCounter()));
```

```
                }
        }
        return excluded;
    }

    public void getWLTotalSeats() {
        try {
                con = db.dbConnect();
                String totalSeatsQuery = "SELECT wait_limit,(SELECT
seats_quota FROM quota_details WHERE "
                        + "quota_id=?) from waiting_limit where
class_id=?";
                ps = con.prepareStatement(totalSeatsQuery);
                ps.setString(1, "TQ");
                ps.setString(2, classid);
                rs = ps.executeQuery();
                rs.next();
                int seatsQuota = rs.getInt(2);
                totalWL = rs.getInt(1) - (rs.getInt(1) * seatsQuota / 100);
                totalTQWL = rs.getInt(1) - totalWL;
                String notAvailableNormalSeatsQuery = "SELECT COUNT(*)
FROM wl WHERE "
                        + "train_no=? AND train_return=? AND
class_id=? AND tatkal=? AND train_src_time = ?";
                ps = con.prepareStatement(notAvailableNormalSeatsQuery);
                ps.setInt(1, trainno);
                ps.setInt(2, trainreturn);
                ps.setString(3, classid);
                ps.setInt(4, 0);
                ps.setTimestamp(5, startTimeStampSql);
                rs = ps.executeQuery();
                rs.next();
                availableWL = totalWL - rs.getInt(1);
                String notAvailableTatkalSeatsQuery = "SELECT COUNT(*)
FROM wl WHERE "
                        + "train_no=? AND train_return=? AND
class_id=? AND tatkal=? AND train_src_time = ?";
                ps = con.prepareStatement(notAvailableTatkalSeatsQuery);
                ps.setInt(1, trainno);
                ps.setInt(2, trainreturn);
                ps.setString(3, classid);
                ps.setInt(4, 1);
```

```
                ps.setTimestamp(5, startTimeStampSql);
                rs = ps.executeQuery();
                rs.next();
                availableTQWL = totalTQWL - rs.getInt(1);
                con.close();
        } catch (Exception e) {
                e.printStackTrace();
        }
    }

    public JSONArray getTotalNoOfSeats() {


        JSONArray seats = new JSONArray();

        for (quotaSeatStructure aqsl : availableQuotaSeatsList) {
                JSONObject seat = new JSONObject();
                seat.put("cnf", aqsl.getTotalNoOfSeats());
                if (!aqsl.getQuotaId().equals("TQ")) {
                        seat.put("rac", availableNoOfRACSeats);
                        seat.put("wl", availableWL);
                } else {
                        seat.put("rac", 0);
                        seat.put("wl", availableTQWL);
                }
                seats.add(seat);
        }
        return seats;
    }

    public String getAvailableSeats(String quotaId,int cnfGn,int rac,int wl,int
tqwl,int cnfQuot){
        String allotedSeat="";

        //System.out.println(cnfGn+" "+ rac+" "+ wl+" "+ tqwl+" "+ cnfQuot);
        if(quotaId.equals("GN")){
                allotedSeat=getAvailableSeatForGN(quotaId, rac, wl,cnfQuot);

        }else{
                allotedSeat=getAvailableSeatForOther(quotaId, cnfGn, rac, wl,
tqwl,cnfQuot);
        }
```

```
        return allotedSeat;
    }
    public String getAvailableSeatForGN(String quotaId,int rac,int wl,int
cnfQuot){
        String allotedSeat="";
        for (quotaSeatStructure aqsl : availableQuotaSeatsList) {
            if(aqsl.getQuotaId().equals(quotaId)){
                if((aqsl.getTotalNoOfSeats()-cnfQuot)>0){
                    allotedSeat="Confirm(CNF)/////"+quotaId;
                }
                break;
            }
        }
        if(allotedSeat.length()==0 && (availableNoOfRACSeats-rac)>0){
            allotedSeat="Reserve Against Cancellation(RAC
"+(getRACLatestListNo()+rac)+")/////"+quotaId;
        }else if(allotedSeat.length()==0 && (availableWL-wl)>0){
            allotedSeat="Waiting(WL
"+(getWLLatestListNo()+wl)+")/////"+quotaId;
        }else if(allotedSeat.length()==0){
            allotedSeat="No Ticket Available";
        }

        return allotedSeat;
    }
    public String getAvailableSeatForOther(String quotaId,int cnfGn,int rac,int
wl,int tqwl,int cnfQuot){
        String allotedSeat="";
        for (quotaSeatStructure aqsl : availableQuotaSeatsList) {
            if(aqsl.getQuotaId().equals(quotaId)){
                if((aqsl.getTotalNoOfSeats()-cnfQuot)>0){
                    allotedSeat="Confirm(CNF)/////"+quotaId;
                }
                break;
            }
        }
        if(allotedSeat.length()==0){
            for (quotaSeatStructure aqsl : availableQuotaSeatsList) {
                if(aqsl.getQuotaId().equals("GN")){
                    if((aqsl.getTotalNoOfSeats()-cnfGn)>0){
                        allotedSeat="Confirm(CNF)/////GN";
                    }
```

```
                        break;
                    }
                }
            }
            if(allotedSeat.length()==0 && (availableNoOfRACSeats - rac)>0 &&
!(quotaId.equals("TQ"))){
                allotedSeat="Reserve Against Cancellation(RAC
"+(getRACLatestListNo()+rac)+")/////"+quotaId;
            }else if(allotedSeat.length()==0){
                //System.out.println("Hello");
                if(quotaId.equals("TQ") && (availableTQWL - tqwl)>0){
                allotedSeat="Tatkal Waiting(TQWL
"+(getTQWLLatestListNo()+tqwl)+")/////"+quotaId;
                }
                if((!(quotaId.equals("TQ"))) && (availableWL -wl)>0){
                    allotedSeat="Waiting(WL
"+(getWLLatestListNo()+wl)+")/////"+quotaId;
                }
            }
            if(allotedSeat.length()==0){
                allotedSeat="No Ticket Available";
            }

            return allotedSeat;
    }

    public int getRACLatestListNo(){
        return totalNoRACSeats-availableNoOfRACSeats+1;
    }
    public int getWLLatestListNo(){
        return totalWL-availableWL+1;
    }
    public int getTQWLLatestListNo(){
        return totalTQWL-availableTQWL+1;
    }
    public JSONArray getAvailableBerth(){

        JSONArray availableBerth=new JSONArray();
        JSONArray availableBerthWithName=new JSONArray();

        int i=0;
        for (seatsForClass acsfm : availableConfirmSeatsFromMiddle) {
```

```
                if(i==0){
                        availableBerth.add(acsfm.getBerthType());
                }else{
                        boolean found=false;
                        for (Object ab : availableBerth) {
                                if(ab.toString().equals(acsfm.getBerthType())){
                                        found=true;
                                        break;
                                }
                        }
                        if(!found){
                                availableBerth.add(acsfm.getBerthType());
                        }
                }

                i++;
        }
        String ab=availableBerth.toString().replace("[", "").replace("]",
"").replace("\"", "\'");
        String queryBerthName="SELECT berth_name,berth_id FROM
berth_details WHERE berth_id in ("+ab+")";
        try{
        con=db.dbConnect();
        ps=con.prepareStatement(queryBerthName);
        rs=ps.executeQuery();
        i=0;
        JSONObject jobj=new JSONObject();
        jobj.put("name", "No Preference");
        jobj.put("id", "NP");
        availableBerthWithName.add(jobj);
        while(rs.next()){
                jobj=new JSONObject();
                jobj.put("name", rs.getString(1));
                jobj.put("id", rs.getString(2));
                availableBerthWithName.add(jobj);
                i++;
        }
        con.close();
        }catch(Exception e){
                e.printStackTrace();
        }
        return availableBerthWithName;
```

```
}

public String[] getFormattedStatus(String totalStatusWithId) {
	String[] totalStatusWithIdArray=totalStatusWithId.split("/////");
	String totalStatus=totalStatusWithIdArray[0];
	String[] formattedStatusArray=new String[4];
	int startBracket=totalStatus.indexOf('(');
	int endBracket=totalStatus.indexOf(')');
	formattedStatusArray[0]=totalStatus.substring(0, startBracket);
	String insideBracket=totalStatus.substring(startBracket+1, endBracket);
	String[] idAndNumber=insideBracket.split(" ");
	int count=1;
	for (String ian : idAndNumber) {
		formattedStatusArray[count]=ian;
		count++;
	}
	if(formattedStatusArray[1].equalsIgnoreCase("CNF")){
		formattedStatusArray[2]="0";
	}
	formattedStatusArray[3]=totalStatusWithIdArray[1];
	return formattedStatusArray;
}
public seatsForClass getCNFSeatsForPassenger(String pref){
	seatsForClass sfc=null;
	boolean found=false;
	for (seatsForClass acsfm : availableConfirmSeatsFromMiddle) {
		if(acsfm.getBerthType().equalsIgnoreCase(pref)){
			sfc=acsfm;
			found=true;
			break;
		}
	}
	if(!found){
		sfc=availableConfirmSeatsFromMiddle.get(0);
	}
	return sfc;
}

public seatsForClass getRACSeatsForPassenger(){
	seatsForClass sfc=availableRACSeatsFromMiddle.get(0);
	return sfc;
}
```

```
   public JSONObject insertIntoCNFTable(String[] formattedStatus,String
pnr,String quotaId,String pref,int pnrCounter,
                Connection c,PreparedStatement p,JSONObject
dataINJSONObject,Timestamp trainSrcTime,
                Timestamp trainDestTime){
        try {
                String cnfQuery="INSERT INTO cnf VALUES (?, ?, ?, ?, "
                                + "?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
                p=c.prepareStatement(cnfQuery);
                p.setString(1, formattedStatus[1]);
                p.setString(2, pnr);
                p.setInt(3, pnrCounter);
                p.setString(4, quotaId);
                p.setInt(5, trainno);
                p.setInt(6, trainreturn);
                p.setString(7, fromid);
                p.setString(8, toid);
                p.setString(9, classid);
                seatsForClass sfc=getCNFSeatsForPassenger(pref);
                p.setInt(10, sfc.getCoachNo());
                p.setString(11, formattedStatus[3]);
                p.setInt(12, sfc.getCoachSeatNo());
                p.setTimestamp(13, trainSrcTime);
                p.setTimestamp(14, trainDestTime);

                dataINJSONObject.put("coachNo", sfc.getCoachNo());
                dataINJSONObject.put("berthId", sfc.getBerthType());
                dataINJSONObject.put("seatNo", sfc.getCoachSeatNo());
                p.executeUpdate();
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }
        return dataINJSONObject;
   }
   public JSONObject insertIntoRACTable(String[] formattedStatus,String
pnr,String quotaId,int pnrCounter,
                Connection c,PreparedStatement p,JSONObject
dataINJSONObject,Timestamp trainSrcTime,
                Timestamp trainDestTime){
        try {
                String racQuery="INSERT INTO rac VALUES (?, ?, ?, ?,"
```

```
                        + " ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
                p=c.prepareStatement(racQuery);
                p.setString(1, formattedStatus[1]);
                p.setInt(2,Integer.parseInt(formattedStatus[2].trim()));
                p.setString(3, pnr);
                p.setInt(4, pnrCounter);
                p.setString(5, quotaId);
                p.setInt(6, trainno);
                p.setInt(7, trainreturn);
                p.setString(8, fromid);
                p.setString(9, toid);
                p.setString(10, classid);
                seatsForClass sfc=getRACSeatsForPassenger();
                p.setInt(11, sfc.getCoachNo());
                p.setInt(12, sfc.getCoachSeatNo());
                p.setInt(13, sfc.getCoachSeatCounter());
                p.setTimestamp(14, trainSrcTime);
                p.setTimestamp(15, trainDestTime);
                p.executeUpdate();
                dataINJSONObject.put("coachNo", sfc.getCoachNo());
                dataINJSONObject.put("berthId", "SLB");
                dataINJSONObject.put("seatNo", sfc.getCoachSeatNo());
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }
        return dataINJSONObject;
   }
   public JSONObject insertIntoWLTable(String[] formattedStatus,String
pnr,String quotaId,int pnrCounter,int tatkal,
                Connection c,PreparedStatement p,JSONObject
dataINJSONObject,Timestamp trainSrcTime,
                Timestamp trainDestTime){
        try {
                String racQuery="INSERT INTO wl VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?);";
                p=c.prepareStatement(racQuery);
                p.setString(1, formattedStatus[1]);
                p.setInt(2,Integer.parseInt(formattedStatus[2].trim()));
                p.setString(3, pnr);
                p.setInt(4, pnrCounter);
                p.setString(5, quotaId);
```

```
                p.setInt(6, trainno);
                p.setInt(7, trainreturn);
                p.setString(8, fromid);
                p.setString(9, toid);
                p.setString(10, classid);
                p.setInt(11, tatkal);
                p.setTimestamp(12, trainSrcTime);
                p.setTimestamp(13, trainDestTime);
                p.executeUpdate();
                dataINJSONObject.put("coachNo", "-");
                dataINJSONObject.put("berthId", "-");
                dataINJSONObject.put("seatNo", "-");
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }
        return dataINJSONObject;
    }

    public JSONObject insertIntoStatusTable(String[] formattedStatus,String
pnr,String quotaId,String pref,
                int pnrCounter,Connection c,PreparedStatement p,JSONObject
dataINJSONObject,Timestamp trainSrcTime,
                Timestamp trainDestTime){
        JSONObject newDataINJSONObject=new JSONObject();
        if(formattedStatus[1].equalsIgnoreCase("CNF")){

    newDataINJSONObject=insertIntoCNFTable(formattedStatus,pnr,quotaId,
pref,pnrCounter,c,p,dataINJSONObject,trainSrcTime,trainDestTime);
        }else if(formattedStatus[1].equalsIgnoreCase("RAC")){

    newDataINJSONObject=insertIntoRACTable(formattedStatus,pnr,quotaId,
pnrCounter,c,p,dataINJSONObject,trainSrcTime,trainDestTime);
        }else if(formattedStatus[1].equalsIgnoreCase("WL")){

    newDataINJSONObject=insertIntoWLTable(formattedStatus,pnr,quotaId,p
nrCounter,0,c,p,dataINJSONObject,trainSrcTime,trainDestTime);
        }else if(formattedStatus[1].equalsIgnoreCase("TQWL")){
                formattedStatus[1]="WL";

    newDataINJSONObject=insertIntoWLTable(formattedStatus,pnr,quotaId,p
nrCounter,1,c,p,dataINJSONObject,trainSrcTime,trainDestTime);
```

```
        }
        return newDataINJSONObject;
    }

}
```

seatCancellation.java

```
package railwayFrequentFunctions;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;

import com.sun.org.apache.bcel.internal.classfile.PMGClass;

class EligibleSeatStructure {
    private String pnr = "";
    private int pnrCounter = 0;

    public EligibleSeatStructure() {

    }

    public EligibleSeatStructure(String pnr, int pnrCounter) {

        this.pnr = pnr;
        this.pnrCounter = pnrCounter;
    }

    public String getPnr() {
        return pnr;
    }

    public void setPnr(String pnr) {
        this.pnr = pnr;
    }
```

```
    public int getPnrCounter() {
        return pnrCounter;
    }

    public void setPnrCounter(int pnrCounter) {
        this.pnrCounter = pnrCounter;
    }
}

public class SeatCancellation {
    @Override
    public String toString() {
        return "SeatCancellation [pnrCounter=" + pnrCounter + ", seatNo="
                + seatNo + ", seatCounter=" + seatCounter + ", coachNo="
                + coachNo + ", statusCounter=" + statusCounter + ",
trainNo="
                + trainNo + ", trainReturn=" + trainReturn + ", istatkal="
                + istatkal + ", pnr=" + pnr + ", quotaId=" + quotaId
                + ", classId=" + classId + ", fromId=" + fromId + ", toId="
                + toId + ", statusId=" + statusId + ", allottedQuotaId="
                + allottedQuotaId + ", con=" + con + ", query=" + query
                + ", train_src_time=" + train_src_time + ",
train_dest_time="
                + train_dest_time + ", ps=" + ps + ", rs=" + rs
                + ", extremeStartRoute=" + extremeStartRoute
                + ", extremeEndRoute=" + extremeEndRoute + ",
routeId="
                + routeId + "]";
    }

    private int pnrCounter = 0, seatNo = 0, seatCounter = 0, coachNo = 0,
            statusCounter = 0, trainNo = 0, trainReturn = 0, istatkal = 0;
    private String pnr = "";
    private String quotaId = "", classId = "", fromId = "", toId = "",
            statusId = "", allottedQuotaId = "";
    private Connection con;
    private String query;
    private Timestamp train_src_time, train_dest_time;
    private PreparedStatement ps;
    private ResultSet rs;
    private int extremeStartRoute = 0, extremeEndRoute = 0, routeId = 0;
```

```
    private int beforeChangeSeatNo = 0, beforeChangeSeatCounter = 0,
                beforeChangeCoachCounter = 0;

    public SeatCancellation(String pnr, int pnrCounter) {
        this.pnr = pnr;
        this.pnrCounter = pnrCounter;
        con = (new DatabaseConnection()).dbConnect();
        initializeVariable(pnr, pnrCounter);
    }

    public void initializeVariable(String pnr, int pnrCounter) {
        try {
                query = "SELECT `status_id` FROM `booked_customer_details`
WHERE `PNR`= ? AND `PNR_counter`= ?;";
                ps = con.prepareStatement(query);
                ps.setString(1, pnr);
                ps.setInt(2, pnrCounter);
                rs = ps.executeQuery();
                rs.next();
                if (rs.getString(1).equalsIgnoreCase("CNF")) {
                        query = "SELECT * FROM `cnf` WHERE `PNR`= ? AND
`PNR_counter`= ?;";
                        ps = con.prepareStatement(query);
                        ps.setString(1, pnr);
                        ps.setInt(2, pnrCounter);
                        rs = ps.executeQuery();
                        rs.next();
                        statusId = rs.getString(1);
                        pnr = rs.getString(2);
                        pnrCounter = rs.getInt(3);
                        quotaId = rs.getString(4);
                        trainNo = rs.getInt(5);
                        trainReturn = rs.getInt(6);
                        fromId = rs.getString(7);
                        toId = rs.getString(8);
                        classId = rs.getString(9);
                        coachNo = rs.getInt(10);
                        allottedQuotaId = rs.getString(11);
                        seatNo = rs.getInt(12);
                        train_src_time = rs.getTimestamp(13);
                        train_dest_time = rs.getTimestamp(14);
                } else if (rs.getString(1).equalsIgnoreCase("RAC")) {
```

```
                        query = "SELECT * FROM `rac` WHERE `PNR`= ? AND
`PNR_counter`= ?;";
                        ps = con.prepareStatement(query);
                        ps.setString(1, pnr);
                        ps.setInt(2, pnrCounter);
                        rs = ps.executeQuery();
                        rs.next();
                        statusId = rs.getString(1);
                        statusCounter = rs.getInt(2);
                        pnr = rs.getString(3);
                        pnrCounter = rs.getInt(4);
                        quotaId = rs.getString(5);
                        trainNo = rs.getInt(6);
                        trainReturn = rs.getInt(7);
                        fromId = rs.getString(8);
                        toId = rs.getString(9);
                        classId = rs.getString(10);
                        coachNo = rs.getInt(11);
                        seatNo = rs.getInt(12);
                        seatCounter = rs.getInt(13);
                        train_src_time = rs.getTimestamp(14);
                        train_dest_time = rs.getTimestamp(15);

                    } else if (rs.getString(1).equalsIgnoreCase("WL")) {
                        query = "SELECT * FROM `wl` WHERE `PNR`= ? AND
`PNR_counter`= ?;";
                        ps = con.prepareStatement(query);
                        ps.setString(1, pnr);
                        ps.setInt(2, pnrCounter);
                        rs = ps.executeQuery();
                        rs.next();
                        statusId = rs.getString(1);
                        statusCounter = rs.getInt(2);
                        pnr = rs.getString(3);
                        pnrCounter = rs.getInt(4);
                        quotaId = rs.getString(5);
                        trainNo = rs.getInt(6);
                        trainReturn = rs.getInt(7);
                        fromId = rs.getString(8);
                        toId = rs.getString(9);
                        classId = rs.getString(10);
                        istatkal = rs.getInt(11);
```

```
                    train_src_time = rs.getTimestamp(12);
                    train_dest_time = rs.getTimestamp(13);

                }
                query = "SELECT `route_id` FROM `train_master` WHERE
`train_no`= ? AND `train_return`=? ;";
                ps = con.prepareStatement(query);
                ps.setInt(1, trainNo);
                ps.setInt(2, trainReturn);
                rs = ps.executeQuery();
                rs.next();
                routeId = rs.getInt(1);
                query = "SELECT counter FROM `route_master` WHERE
`station_code`= ? AND `route_id`=?";
                ps = con.prepareStatement(query);
                ps.setString(1, fromId);
                ps.setInt(2, routeId);
                rs = ps.executeQuery();
                rs.next();
                extremeStartRoute = rs.getInt(1);
                query = "SELECT counter FROM `route_master` WHERE
`station_code`= ? AND `route_id`=?";
                ps = con.prepareStatement(query);
                ps.setString(1, toId);
                ps.setInt(2, routeId);
                rs = ps.executeQuery();
                rs.next();
                extremeEndRoute = rs.getInt(1);
                beforeChangeSeatNo = seatNo;
                beforeChangeSeatCounter = seatCounter;
                beforeChangeCoachCounter = coachNo;
        } catch (Exception e) {
                e.printStackTrace();
        }
    }

    public boolean cancelTicket(boolean passengerCancelled, String
pastStatusId) {
        try {
                if (passengerCancelled) {
                        query = "UPDATE booked_customer_details SET PNR=?
WHERE PNR=? and PNR_counter=?";
```

```
                    ps = con.prepareStatement(query);
                    ps.setString(1, pnr + "<br>(CANCELLED)");
                    ps.setString(2, pnr);
                    ps.setInt(3, pnrCounter);
                    ps.executeUpdate();
                    pnr = pnr + "<br>(CANCELLED)";
                    query = "DELETE FROM booked_customer_details
WHERE PNR=? and PNR_counter=?";
                    ps = con.prepareStatement(query);
                    ps.setString(1, pnr);
                    ps.setInt(2, pnrCounter);
                    ps.executeUpdate();
                }
        } catch (Exception e) {
                e.printStackTrace();
                return false;
        }
        if (pastStatusId.equalsIgnoreCase("CNF")) {
                System.out.println(pnr + " " + pnrCounter + " Cancelled From
CNF");
                return true;
        } else if (pastStatusId.equalsIgnoreCase("RAC")) {
                cancelRACTicket();
                return true;
        } else if (pastStatusId.equalsIgnoreCase("WL")) {
                cancelWLTicket();
                return true;
        } else if (pastStatusId.equalsIgnoreCase("TQWL")) {

                cancelTQWLTicket();
                return true;
        }
        return false;
   }

   public boolean cancelRACTicket() {
        try {
                System.out.println(pnr + " " + pnrCounter + " Cancelled From
RAC");
                query = "DELETE FROM rac WHERE PNR=? and
PNR_counter=?";
                ps = con.prepareStatement(query);
```

```
                ps.setString(1, pnr);
                ps.setInt(2, pnrCounter);
                ps.executeUpdate();

                query = "UPDATE `rac` SET `status_counter`=`status_counter`-
1 WHERE `status_counter` > ?";
                ps = con.prepareStatement(query);
                ps.setInt(1, statusCounter);
                ps.executeUpdate();
                return true;
        } catch (Exception e) {
                return false;
        }

    }

    public boolean cancelTQWLTicket() {
        try {
                System.out.println(pnr + " " + pnrCounter + " Cancelled From
TQWL");
                query = "DELETE FROM wl WHERE PNR=? and
PNR_counter=?";
                ps = con.prepareStatement(query);
                ps.setString(1, pnr);
                ps.setInt(2, pnrCounter);
                ps.executeUpdate();
                query = "UPDATE `wl` SET `status_counter`=`status_counter`- 1
WHERE `status_counter` > ? AND `tatkal`=?";
                ps = con.prepareStatement(query);
                ps.setInt(1, statusCounter);
                ps.setInt(2, 1);
                ps.executeUpdate();
                return true;
        } catch (Exception e) {
                return false;
        }
    }

    public boolean cancelWLTicket() {
        try {
                System.out.println(pnr + " " + pnrCounter
                                + " Cancelled From Just WL");
```

```
                query = "DELETE FROM wl WHERE PNR=? and
PNR_counter=?";
                ps = con.prepareStatement(query);
                ps.setString(1, pnr);
                ps.setInt(2, pnrCounter);
                ps.executeUpdate();

                query = "UPDATE `wl` SET `status_counter`=`status_counter`- 1
WHERE `status_counter` > ? AND `tatkal`=?";
                ps = con.prepareStatement(query);
                ps.setInt(1, statusCounter);
                ps.setInt(2, 0);
                ps.executeUpdate();
                return true;
        } catch (Exception e) {
                e.printStackTrace();
                return false;
        }
   }

   public EligibleSeatStructure getTQWLEligiblePassenger() {
        try {
                query = "SELECT `PNR`, `PNR_counter` FROM `wl` WHERE
from_station IN "
                        + "(SELECT `station_code` FROM `route_master`
WHERE `route_id`=? AND `counter`>=?) "
                        + "AND to_station IN (SELECT `station_code`
FROM `route_master` WHERE `route_id`=? AND "
                        + "`counter`<=?) AND `train_src_time`=? AND
`tatkal`=? ORDER BY status_counter;";
                ps = con.prepareStatement(query);
                ps.setInt(1, routeId);
                ps.setInt(2, extremeStartRoute);
                ps.setInt(3, routeId);
                ps.setInt(4, extremeEndRoute);
                ps.setTimestamp(5, train_src_time);
                ps.setInt(6, 1);
                rs = ps.executeQuery();
                rs.next();
                EligibleSeatStructure eligible = new EligibleSeatStructure(
                        rs.getString(1), rs.getInt(2));
```

```
                return eligible;
        } catch (Exception e) {
                return null;
        }
    }

    public EligibleSeatStructure getWLEligiblePassenger() {
        try {
                query = "SELECT `PNR`, `PNR_counter` FROM `wl` WHERE
from_station IN "
                        + "(SELECT `station_code` FROM `route_master`
WHERE `route_id`=? AND `counter`>=?) "
                        + "AND to_station IN (SELECT `station_code`
FROM `route_master` WHERE `route_id`=? AND "
                        + "`counter`<=?) AND `train_src_time`=? AND
`tatkal`=? ORDER BY status_counter;";
                ps = con.prepareStatement(query);
                ps.setInt(1, routeId);
                ps.setInt(2, extremeStartRoute);
                ps.setInt(3, routeId);
                ps.setInt(4, extremeEndRoute);
                ps.setTimestamp(5, train_src_time);
                ps.setInt(6, 0);
                rs = ps.executeQuery();
                rs.next();
                EligibleSeatStructure eligible = new EligibleSeatStructure(
                        rs.getString(1), rs.getInt(2));
                return eligible;
        } catch (Exception e) {
                return null;
        }
    }

    public EligibleSeatStructure getRACEligiblePassengerWithQuota(
                String cancelledquotaid) {
        try {
                query = "SELECT `PNR`, `PNR_counter` FROM `rac` WHERE
from_station IN "
                        + "(SELECT `station_code` FROM `route_master`
WHERE `route_id`=? AND `counter`>=?) "
                        + "AND to_station IN (SELECT `station_code`
FROM `route_master` WHERE `route_id`=? AND "
```

```
                        + "`counter`<=?) AND `train_src_time`=? AND
`quota_id`=? ORDER BY status_counter;";
                ps = con.prepareStatement(query);
                ps.setInt(1, routeId);
                ps.setInt(2, extremeStartRoute);
                ps.setInt(3, routeId);
                ps.setInt(4, extremeEndRoute);
                ps.setTimestamp(5, train_src_time);
                ps.setString(6, cancelledquotaid);
                rs = ps.executeQuery();
                rs.next();
                EligibleSeatStructure eligible = new EligibleSeatStructure(
                        rs.getString(1), rs.getInt(2));
                return eligible;
        } catch (Exception e) {
                return null;
        }

   }

   public EligibleSeatStructure getRACEligiblePassengerWithOutQuota() {
        try {
                query = "SELECT `PNR`, `PNR_counter` FROM `rac` WHERE
from_station IN "
                        + "(SELECT `station_code` FROM `route_master`
WHERE `route_id`=? AND `counter`>=?) "
                        + "AND to_station IN (SELECT `station_code`
FROM `route_master` WHERE `route_id`=? AND "
                        + "`counter`<=?) AND `train_src_time`=? ORDER
BY status_counter;";
                ps = con.prepareStatement(query);
                ps.setInt(1, routeId);
                ps.setInt(2, extremeStartRoute);
                ps.setInt(3, routeId);
                ps.setInt(4, extremeEndRoute);
                ps.setTimestamp(5, train_src_time);
                rs = ps.executeQuery();
                rs.next();
                EligibleSeatStructure eligible = new EligibleSeatStructure(
                        rs.getString(1), rs.getInt(2));
                return eligible;
        } catch (Exception e) {
```

```
                return null;
        }

    }

    public boolean allocateEligibleInCNF(int cancelledSeatNo,
                int cancelledCoachNo, String cancelledAllottedQuotaId) {
        try {
                query = "UPDATE `booked_customer_details` SET
`status_id`=?,`status_counter`=? "
                                + "WHERE PNR=? and PNR_counter=?";
                ps = con.prepareStatement(query);
                statusId = "CNF";
                ps.setString(1, statusId);
                statusCounter = 0;
                ps.setInt(2, statusCounter);
                ps.setString(3, pnr);
                ps.setInt(4, pnrCounter);
                ps.executeUpdate();
                query = "INSERT INTO `cnf` VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?); ";
                ps = con.prepareStatement(query);
                ps.setString(1, statusId);
                ps.setString(2, pnr);
                ps.setInt(3, pnrCounter);
                ps.setString(4, quotaId);
                ps.setInt(5, trainNo);
                ps.setInt(6, trainReturn);
                ps.setString(7, fromId);
                ps.setString(8, toId);
                ps.setString(9, classId);
                coachNo = cancelledCoachNo;
                ps.setInt(10, coachNo);
                allottedQuotaId = cancelledAllottedQuotaId;
                ps.setString(11, allottedQuotaId);
                seatNo = cancelledSeatNo;
                ps.setInt(12, seatNo);
                ps.setTimestamp(13, train_src_time);
                ps.setTimestamp(14, train_dest_time);
                ps.executeUpdate();
                return true;
        } catch (Exception e) {
```

```
                return false;
        }
    }

    public boolean allocateEligibleInRAC(int cancelledSeatNo,
                int cancelledSeatCounter, int cancelledCoachNo) {
        try {
                query = "SELECT count(*) from rac where `train_no`=? and
`train_return`=? and `class_id`=?";
                ps = con.prepareStatement(query);
                ps.setInt(1, trainNo);
                ps.setInt(2, trainReturn);
                ps.setString(3, classId);
                rs = ps.executeQuery();
                rs.next();
                statusCounter = rs.getInt(1) + 1;
                query = "UPDATE `booked_customer_details` SET
`status_id`=?,`status_counter`=? "
                                + "WHERE PNR=? and PNR_counter=?";
                ps = con.prepareStatement(query);
                statusId = "RAC";
                ps.setString(1, statusId);
                ps.setInt(2, statusCounter);
                ps.setString(3, pnr);
                ps.setInt(4, pnrCounter);
                ps.executeUpdate();
                query = "INSERT INTO `rac` VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?); ";
                ps = con.prepareStatement(query);
                ps.setString(1, statusId);
                ps.setInt(2, statusCounter);
                ps.setString(3, pnr);
                ps.setInt(4, pnrCounter);
                ps.setString(5, quotaId);
                ps.setInt(6, trainNo);
                ps.setInt(7, trainReturn);
                ps.setString(8, fromId);
                ps.setString(9, toId);
                ps.setString(10, classId);
                coachNo = cancelledCoachNo;
                ps.setInt(11, coachNo);
                seatNo = cancelledSeatNo;
```

```
                ps.setInt(12, seatNo);
                seatCounter = cancelledSeatCounter;
                ps.setInt(13, seatCounter);
                ps.setTimestamp(14, train_src_time);
                ps.setTimestamp(15, train_dest_time);
                ps.executeUpdate();
                return true;
        } catch (Exception e) {
                return false;
        }
    }

    public boolean WLCancelProcess(boolean isPassengerCancelled) {
        if (cancelTicket(isPassengerCancelled, "WL")) {
                return true;
        }
        return false;
    }

    public boolean TQWLCancelProcess(boolean isPassengerCancelled) {
        if (cancelTicket(isPassengerCancelled, "TQWL")) {

                return true;
        }
        return false;
    }

    public boolean RACCancelProcess(boolean isPassengerCancelled) {
        EligibleSeatStructure eligible;
        if (cancelTicket(isPassengerCancelled, "RAC")) {
                eligible = getWLEligiblePassenger();
                if (eligible != null) {
                        SeatCancellation eligibleSeatCancel = new
SeatCancellation(
                                        eligible.getPnr(), eligible.getPnrCounter());
                        System.out.println(beforeChangeSeatNo + " "
                                        + beforeChangeCoachCounter + " "
                                        + beforeChangeSeatCounter);
                        if (eligibleSeatCancel.allocateEligibleInRAC(
                                        beforeChangeSeatNo,
beforeChangeSeatCounter,
                                        beforeChangeCoachCounter)) {
```

```
                        eligibleSeatCancel.WLCancelProcess(false);
                        return true;
                    }
                }
            }
            return false;
        }

        public boolean CNFCancelProcess(boolean isPassengerCancelled) {
            EligibleSeatStructure eligible;
            if (cancelTicket(isPassengerCancelled, "CNF")) {
                if (allottedQuotaId.equalsIgnoreCase("TQ")) {
                    eligible = getTQWLEligiblePassenger();
                    if (eligible != null) {
                        SeatCancellation eligibleSeatCancel = new
SeatCancellation(
                                eligible.getPnr(),
eligible.getPnrCounter());

                        if
(eligibleSeatCancel.allocateEligibleInCNF(seatNo,
                                coachNo, allottedQuotaId)) {

        eligibleSeatCancel.TQWLCancelProcess(false);
                            return true;
                        }
                    }
                } else if (allottedQuotaId.equalsIgnoreCase("GN")) {
                    eligible = getTQWLEligiblePassenger();
                    if (eligible != null) {
                        SeatCancellation eligibleSeatCancel = new
SeatCancellation(
                                eligible.getPnr(),
eligible.getPnrCounter());
                        if
(eligibleSeatCancel.allocateEligibleInCNF(seatNo,
                                coachNo, allottedQuotaId)) {

        eligibleSeatCancel.TQWLCancelProcess(false);
                            return true;
                        }
                    } else {
```

```
                    eligible =
getRACEligiblePassengerWithOutQuota();
                    if (eligible != null) {
                        SeatCancellation eligibleSeatCancel = new
SeatCancellation(
                                eligible.getPnr(),
eligible.getPnrCounter());

   System.out.println(eligibleSeatCancel.toString());
                        if
(eligibleSeatCancel.allocateEligibleInCNF(seatNo,
                                coachNo, allottedQuotaId)) {


   eligibleSeatCancel.RACCancelProcess(false);
                            return true;
                        }
                    }
                }
            } else {
                eligible =
getRACEligiblePassengerWithQuota(allottedQuotaId);
                if (eligible != null) {
                    SeatCancellation eligibleSeatCancel = new
SeatCancellation(
                            eligible.getPnr(),
eligible.getPnrCounter());
                    if
(eligibleSeatCancel.allocateEligibleInCNF(seatNo,
                            coachNo, allottedQuotaId)) {

   eligibleSeatCancel.RACCancelProcess(false);
                        return true;
                    }
                }
            }
        }
        return false;
   }

   public void totalProcess(String receivedStatusID, String receivedQuotaID) {
        try {
```

```
                    if (receivedStatusID.equalsIgnoreCase("CNF")) {
                            CNFCancelProcess(true);
                    } else if (receivedStatusID.equalsIgnoreCase("RAC")) {
                            RACCancelProcess(true);
                    } else if (receivedStatusID.equalsIgnoreCase("WL")
                                    && receivedQuotaID.equalsIgnoreCase("TQ")) {
                            WLCancelProcess(true);
                    } else if (receivedStatusID.equalsIgnoreCase("WL")) {
                            TQWLCancelProcess(true);
                    }
            } catch (Exception e) {
                    e.printStackTrace();
            } finally {
                    try {
                            con.close();
                    } catch (SQLException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                    }
            }

      }
    }
```

secretKey.java

```
package railwayFrequentFunctions;

public class SecretKey {
public final String getSecretKey(String plain,int shift){
      String key="";
      for (int i = 0; i < plain.length(); i++) {
      key+=(char)(plain.charAt(i)+shift);
      }
      return key;
}
}
```

SourceandDestination.java

```
package railwayFrequentFunctions;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import org.json.simple.JSONArray;



public class sourceAndDestination {
 public JSONArray src_dest(String src){
		JSONArray result = new JSONArray();
		try {
				DatabaseConnection db=new DatabaseConnection();

				Connection con = db.dbConnect();
				String query="";

				if (src.trim().equals("")) {
					query="SELECT Distinct station_name FROM station_master ORDER BY station_name ASC";
				} else {

					query="SELECT DISTINCT station_name FROM station_master WHERE station_code IN "
```

```
                                        + "(select DISTINCT station_code from
route_master s2 where counter > "
                                        + "(select counter from route_master where
station_code IN "
                                        + "(SELECT station_code from
station_master where station_name=?) "
                                        + "and route_id=s2.route_id)) ORDER BY
station_name ASC ";



                        }
                        PreparedStatement ps=con.prepareStatement(query);


                        if(!src.trim().equals(""))
                        {
                                ps.setString(1, src);
                        }


                        ResultSet rs=ps.executeQuery();
                        while (rs.next()) {
                                result.add(rs.getString(1));


                        }
                        con.close();


                }catch(Exception e){
                        e.printStackTrace();
                }
```

```
        return result;
}
}
```

State_and_District.java

```
/**
*
*/
package railwayFrequentFunctions;

/**
* @author Rahul Mukesh Singh
*
*/

import org.geonames.WebService;
import org.geonames.ToponymSearchCriteria;
import org.geonames.ToponymSearchResult;
import java.util.Collections;
import org.geonames.Toponym;
import org.json.simple.JSONArray;


public class State_And_District {
```

```
        public JSONArray getData(String q, String countrycode, String
featurecode) {
                JSONArray data = new JSONArray();
                try {
                        PersonalData pd = new PersonalData();
                        WebService.setUserName(pd.geouser); // add your username

                        // here
                        ToponymSearchCriteria searchCriteria = new
ToponymSearchCriteria();
                        searchCriteria.setQ(q);
                        searchCriteria.setCountryCode(countrycode);
                        searchCriteria.setFeatureCode(featurecode);
                        ToponymSearchResult searchResult =
WebService.search(searchCriteria);

                        for (Toponym toponym : searchResult.getToponyms()) {
                                data.add(toponym.getName().replace((char)257,
'a').replace((char)363, 'u'));
                        }
                        Collections.sort(data);
                } catch (Exception e) {
                        data = null;
                }
                return data;
        }
}
```

TableGenerator.java

```
/**
 *
 */
package railwayFrequentFunctions;


/**
 * @author Rahul Mukesh Singh
 *
 */


public class TableGenerator {
	public String MsgGeneratortable(String[] headers, String[][] data,
			String tableClass, boolean isPDFMail) {
		String msg = "";
		if (isPDFMail) {
			msg += "<table class='" + tableClass + "'><tr>";
			for (String head : headers) {
				msg += "<th>" + head + "</th>";


			}
			msg += "</tr>";
			int count = 0;
			for (String[] row : data) {
				if ((count % 2) == 0) {
```

```
                msg += "<tr class='even'>";
            } else {
                msg += "<tr class='odd'>";
            }

            for (String col : row) {
                msg += "<td>" + col + "</td>";
            }
            msg += "</tr>";
            count++;
        }
        msg += "</table>";
    } else {
        msg += "<table style='border-collapse: collapse;width: 100%;'><tr>";
        for (String head : headers) {
            msg += "<th style='text-align: left;padding:8px;background-color: #00788b;color: white;'>"
                    + head + "</th>";

        }
        msg += "</tr>";
        int count = 0;
        for (String[] row : data) {
            if ((count % 2) == 0) {
                msg += "<tr style='background-color:#f2f2f2;color:#404040'>";
```

```
                } else {
                        msg += "<tr style='color:#404040'>";
                }
                for (String col : row) {

                        msg += "<td style='text-align: left;padding: 8px;'>"
+ col
                                        + "</td>";
                }
                msg += "</tr>";
                count++;
            }
            msg += "</table>";

        }

        return msg;
    }
}
```

ajaxCancellationProcess.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
```

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import railwayFrequentFunctions.SeatCancellation;

/**
 * Servlet implementation class ajaxCancellationProcess
 */
@WebServlet("/ajaxCancellationProcess")
public class ajaxCancellationProcess extends HttpServlet {
		private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ajaxCancellationProcess() {
        super();
        // TODO Auto-generated constructor stub
    }

		/**
		 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
		 */
```

```
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		response.setContentType("text/html");
		PrintWriter out=response.getWriter();
		try{
			String pnr=(String)request.getParameter("pnr");
			int pnrCounter=Integer.parseInt(request.getParameter("pnrCounter"));
			String statusId=(String)request.getParameter("statusId");
			String quotaId=(String)request.getParameter("quotaId");

			SeatCancellation sc=new SeatCancellation(pnr, pnrCounter);
			sc.totalProcess(statusId, quotaId);

			out.print("Cancellation Successful !!!");

			}catch(Exception e){
				e.printStackTrace();
```

```
                    out.print("Cancellation Unsuccessful !!!");
                    }
            }

}
```

AvailableSeats.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;

import railwayFrequentFunctions.SeatAllocation;
```

```
/**
 * Servlet implementation class AvailableSeats
 */
@WebServlet("/AvailableSeats")
public class AvailableSeats extends HttpServlet {
	private static final long serialVersionUID = 1L;

	/**
	 * @see HttpServlet#HttpServlet()
	 */
	public AvailableSeats() {
		super();
		// TODO Auto-generated constructor stub
	}

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
	 *      response)
	 */
	protected void doGet(HttpServletRequest request,
			HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		doPost(request, response);
	}
```

```
	/**
	* @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse
	*  response)
	*/
	protected void doPost(HttpServletRequest request,
			HttpServletResponse response) throws ServletException,
IOException {
		// TODO Auto-generated method stub
		try {
			HttpSession session = request.getSession();
			String trainno = (String) session.getAttribute("trainno");
			String trainreturn = (String) session.getAttribute("trainreturn");
			String classid = (String) session.getAttribute("classid");
			String routeid = (String) session.getAttribute("routeId");
			String fromid = (String) session.getAttribute("source");
			String toid = (String) session.getAttribute("destination");
			String sdate = (String) session.getAttribute("date");
			SeatAllocation sa = new SeatAllocation(trainno, trainreturn,
					classid, routeid, fromid, toid,sdate);
			PrintWriter out = response.getWriter();
			response.setContentType("application/json");
			JSONArray seats = new JSONArray();
			seats = sa.getTotalNoOfSeats();
			out.print(seats);
		} catch (Exception e) {
```

```
                e.printStackTrace();
            }
        }

}
```

BookingToPaymentGateway.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
```

```
import railwayFrequentFunctions.DatabasePrice;
import railwayFrequentFunctions.DateTimeofStation;
import railwayFrequentFunctions.SeatAllocation;

/**
 * Servlet implementation class BookingToPaymentGateway
 */
@WebServlet("/BookingToPaymentGateway")
public class BookingToPaymentGateway extends HttpServlet {
	private static final long serialVersionUID = 1L;

	/**
	 * @see HttpServlet#HttpServlet()
	 */
	public BookingToPaymentGateway() {
		super();
		// TODO Auto-generated constructor stub
	}

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse
	 *      response)
	 */
	protected void doGet(HttpServletRequest request,
			HttpServletResponse response) throws ServletException,
IOException {
```

```
        // TODO Auto-generated method stub
        doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse
     *     response)
     */
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        // TODO Auto-generated method stub

        PrintWriter out = response.getWriter();
        out.print("<center><h2 style='color:#00788b'>20%</h2></center>");
        HttpSession session = request.getSession();

        response.setContentType("application/json");

        String tno = (String) session.getAttribute("trainno");
        String treturn = (String) session.getAttribute("trainreturn");
        String sdate = (String) session.getAttribute("date");
        String routeid = (String) session.getAttribute("routeId");
        String src = (String) session.getAttribute("source");
        String dest = (String) session.getAttribute("destination");
        String classid = (String) session.getAttribute("classid");
```

```
//out.print(src);
session.setAttribute("dateTimeInJSON", (new DateTimeofStation())
                .getDateAndTimeOfStation(src, dest, tno, treturn, sdate,
                                routeid));
//out.print( (new DateTimeofStation())
//              .getDateAndTimeOfStation(src, dest, tno, treturn, sdate,
//                              routeid));
DatabasePrice dp = new DatabasePrice();
SeatAllocation sa = new SeatAllocation(tno, treturn, classid, routeid,
                src, dest,sdate);
String quotaid = "";
double totalPriceWithoutTax = 0.0;
//out.print(request.getParameter("ticketData"));
String[] datas = request.getParameter("ticketData").split("///");
JSONArray jarr = new JSONArray();

int count=0;
for (String data : datas) {
        JSONObject jobj = new JSONObject();
        String[] d = data.split(";");
        quotaid=d[3];
        int cnfGn=0, rac=0, wl=0, tqwl=0, cnfQuot=0;
        for (int i = (count-1); i >=0; i--) {
                String
quotaColumnValue=((JSONObject)jarr.get(i)).get("quota").toString();
                String
statusColumnValue=((JSONObject)jarr.get(i)).get("status").toString();
```

```
                        if(statusColumnValue.contains("(CNF") &&
quotaColumnValue.equalsIgnoreCase(quotaid)){
                                cnfQuot=cnfQuot+1;
                        }
                        if(statusColumnValue.contains("(CNF") &&
statusColumnValue.contains("GN")){
                                cnfGn=cnfGn+1;
                        }
                        if(statusColumnValue.contains("(RAC")){
                                rac=rac+1;
                        }
                        if(statusColumnValue.contains("(WL")){
                                wl=wl+1;
                        }
                        if(statusColumnValue.contains("(TQWL")){
                                tqwl=tqwl+1;
                        }
                }
                //System.out.println(quotaid+"--> "+cnfGn+" "+ rac+" "+ wl+"
"+ tqwl+" "+ cnfQuot);
                String status = sa.getAvailableSeats(quotaid,cnfGn, rac, wl,
tqwl, cnfQuot);
                if (!status.equalsIgnoreCase("No Ticket Available")) {

                        double priceWithoutTax = dp.getPriceWithoutTax(src,
dest,
                                        routeid, tno, treturn, classid, quotaid);
                        totalPriceWithoutTax += priceWithoutTax;
```

```
                    jobj.put("name", d[0]);
                    jobj.put("age", d[1]);
                    jobj.put("gender", d[2]);
                    jobj.put("quota", quotaid);
                    jobj.put("pref", d[4]);
                    jobj.put("status", status);
                    jobj.put("priceWithoutTax", priceWithoutTax);
                    jarr.add(jobj);
                    count++;
                }

            }

            session.setAttribute("dataInJSON", jarr);
            JSONArray taxNamePlusPercentPlusTaxedPrice = dp
                    .getPriceWithTax(totalPriceWithoutTax);
            session.setAttribute("taxNamePlusPercentPlusTaxedPriceInJSON",
                    taxNamePlusPercentPlusTaxedPrice);
            JSONObject totalPriceWithTax = (JSONObject)
taxNamePlusPercentPlusTaxedPrice
                    .get(taxNamePlusPercentPlusTaxedPrice.size() - 1);
            double totalTaxedPrice = (double)
totalPriceWithTax.get("pricewithtax");
            //out.print(jarr);
            //out.print(taxNamePlusPercentPlusTaxedPrice);
            //out.print((JSONArray) session.getAttribute("dateTimeInJSON"));
```

```
                RequestDispatcher
rd=request.getRequestDispatcher("../railway/insertPassengers");
                rd.forward(request, response);


            }


}
```

checkOTP.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;

import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import railwayFrequentFunctions.DatabaseConnection;

/**
 * Servlet implementation class CheckOTP
 */
@WebServlet("/CheckOTP")
public class CheckOTP extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public CheckOTP() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```

```
        // TODO Auto-generated method stub
        doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        String email=request.getParameter("emailid");
        String otpByUser=request.getParameter("otp");
        try {
            DatabaseConnection db=new DatabaseConnection();

            Connection con = db.dbConnect();
            String checkExists="select otp from otp_transactional where emailid=?";
            PreparedStatement ps=con.prepareStatement(checkExists);
            ps.setString(1, email);
            ResultSet rs=ps.executeQuery();
            String otp="";
            if(rs.next())
            {
```

```
                    otp=rs.getString(1);
                    if(otp.equals(otpByUser))
                    {
                        out.print("true");
                    }else{
                        out.print("false");
                    }
                    }else{
                        out.print("Time Out! Request OTP Again");
                    }

                    con.close();
                } catch (Exception e) {
                    // TODO: handle exception
                }
            }

}
```

classSelectiontoQuotaDetails.java

```
package servletss;

import java.io.IOException;
```

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class ClassSelectionToQuotaDetails
 */
@WebServlet("/ClassSelectionToQuotaDetails")
public class ClassSelectionToQuotaDetails extends HttpServlet {
		private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ClassSelectionToQuotaDetails() {
        super();
        // TODO Auto-generated constructor stub
    }

		/**
		 * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
		 */
```

```
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		doPost(request, response);
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		HttpSession session=request.getSession();
		String classid=request.getParameter("classid");
		session.setAttribute("classid",classid);

		response.sendRedirect("../Phase3/quotadetails.jsp");
	}

}
```

deletechatsforUserId.java

```
package servletss;
```

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import railwayFrequentFunctions.DatabaseConnection;

/**
 * Servlet implementation class deleteChatsForUserId
 */
@WebServlet("/deleteChatsForUserId")
public class deleteChatsForUserId extends HttpServlet {
		private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public deleteChatsForUserId() {
```

```
    super();
    // TODO Auto-generated constructor stub
  }

        /**
        * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
        */
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            doPost(request, response);
        }

        /**
        * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
        */
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            response.setContentType("text/html");
            HttpSession session=request.getSession();
            PrintWriter out=response.getWriter();
            System.out.println("Deleted");
            Connection con=null;
            try{
            con=(new DatabaseConnection()).dbConnect();
```

```
		String query="DELETE FROM `emergency` WHERE to_id=? OR from_id=? ;";
		PreparedStatement ps=con.prepareStatement(query);
		ps.setString(1, (String)session.getAttribute("userid"));
		ps.setString(2, (String)session.getAttribute("userid"));
		ps.executeUpdate();
		con.close();
		out.print("");
		}catch(Exception e){
			e.printStackTrace();

		}
	}

}
```

DeleteNonLoginSession.java

```
package servletss;

import java.io.IOException;
import java.util.Enumeration;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class DeleteNonLoginSession
 */
@WebServlet("/DeleteNonLoginSession")
public class DeleteNonLoginSession extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public DeleteNonLoginSession() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		doPost(request, response);
	}
```

```
	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		HttpSession session=request.getSession();
		Enumeration<String> sessionNames=session.getAttributeNames();
		String sname="";
		while(sessionNames.hasMoreElements()){
			sname=sessionNames.nextElement();
			if(!(sname.equalsIgnoreCase("userid") || sname.equalsIgnoreCase("role") || sname.equalsIgnoreCase("fname"))){
				session.removeAttribute(sname);
			}
		}
	}

}
```

DownloadTicket.java

```
package servletss;

import java.io.FileInputStream;
```

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import railwayFrequentFunctions.DatabaseConnection;

/**
 * Servlet implementation class DownloadTicket
 */
@WebServlet("/DownloadTicket")
public class DownloadTicket extends HttpServlet {
		private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public DownloadTicket() {
```

```
        super();
        // TODO Auto-generated constructor stub
    }


        /**
        * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
        */
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            doPost(request, response);
        }


        /**
        * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
        */
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            HttpSession session=request.getSession();
            String userid = (String) session.getAttribute("userid");
            String to ="";
            try{
            Connection con = (new DatabaseConnection()).dbConnect();
            String emailQuery = "SELECT email FROM user_details WHERE userid=?";
```

```
PreparedStatement ps = con.prepareStatement(emailQuery);
ps.setString(1, userid);
ResultSet rs = ps.executeQuery();
rs.next();
to = rs.getString(1);
con.close();
}catch(Exception e){
	e.printStackTrace();
}
response.setContentType("application/pdf");
PrintWriter out = response.getWriter();
String filepath =
"E:\\JSP\\RAILWAY_PROJECT\\WebContent\\WEB-
INF\\GeneratedDocuments\\" +
		to.replace(".", "")+ ".pdf";
response.setContentType("APPLICATION/OCTET-STREAM");
response.setHeader("Content-Disposition","inline;
filename=\"RAILWAY TICKET.pdf\"");

FileInputStream fileInputStream = new FileInputStream(filepath);

int i;
while ((i=fileInputStream.read()) != -1) {
out.write(i);
}
fileInputStream.close();
out.close();
```

```
        }

}
```

Downloadticketforcancellation.java

```
package servletss;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import railwayFrequentFunctions.DatabaseConnection;
```

```
/**
 * Servlet implementation class DownloadTicketForCancellation
 */
@WebServlet("/DownloadTicketForCancellation")
public class DownloadTicketForCancellation extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public DownloadTicketForCancellation() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		doPost(request, response);
	}

	/**
```

```
	* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	*/
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		HttpSession session=request.getSession();
		pdfGenerationForCancellation pgfc=new pdfGenerationForCancellation();
		pgfc.doPost(request, response);
		String userid = (String) session.getAttribute("userid");
		String to ="";
		try{
		Connection con = (new DatabaseConnection()).dbConnect();
		String emailQuery = "SELECT email FROM user_details WHERE userid=?";
		PreparedStatement ps = con.prepareStatement(emailQuery);
		ps.setString(1, userid);
		ResultSet rs = ps.executeQuery();
		rs.next();
		to = rs.getString(1);
		con.close();
		}catch(Exception e){
			e.printStackTrace();
		}
		response.setContentType("application/pdf");
		PrintWriter out = response.getWriter();
```

```
String filepath = "E:\\JSP\\RAILWAY_PROJECT\\WebContent\\WEB-INF\\GeneratedDocuments\\" +
                    to.replace(".", "")+ ".pdf";
response.setContentType("APPLICATION/OCTET-STREAM");
response.setHeader("Content-Disposition","inline; filename=\"RAILWAY TICKET.pdf\"");

FileInputStream fileInputStream = new FileInputStream(filepath);

int i;
while ((i=fileInputStream.read()) != -1) {
out.write(i);
}
fileInputStream.close();
out.close();
}

}
```

GetAllChatsAtStart.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import railwayFrequentFunctions.DatabaseConnection;

/**
 * Servlet implementation class GetAllChatsAtStart
 */
@WebServlet("/GetAllChatsAtStart")
public class GetAllChatsAtStart extends HttpServlet {
		private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public GetAllChatsAtStart() {
```

```
        super();
        // TODO Auto-generated constructor stub
    }


        /**
        * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
        */
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            doGet(request, response);
        }


        /**
        * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
        */
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            response.setContentType("application/json");
            HttpSession session=request.getSession();
            PrintWriter out=response.getWriter();
            JSONArray jarr=new JSONArray();
            try{
            Connection con=(new DatabaseConnection()).dbConnect();
```

```
		String query="SELECT * FROM `emergency` WHERE `to_id`=? OR `from_id`=? ;";
		PreparedStatement ps=con.prepareStatement(query);
		ps.setString(1, (String)session.getAttribute("userid"));
		ps.setString(2, (String)session.getAttribute("userid"));
		ResultSet rs=ps.executeQuery();

		while(rs.next()){
			JSONObject jobj=new JSONObject();
			jobj.put("message",rs.getString(1));
			jobj.put("counter",rs.getInt(2));
			jobj.put("to",rs.getString(3));
			jobj.put("from",rs.getString(4));
			jarr.add(jobj);
		}
		con.close();
		out.print(jarr);
		}catch(Exception e){
			e.printStackTrace();
			out.print(jarr);
		}

	}

}
```

GetOTP.java

```
package servletss;

import railwayFrequentFunctions.DatabaseConnection;
import railwayFrequentFunctions.mailnow;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Random;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/**
```

```
 * Servlet implementation class GetOTP
 */
@WebServlet("/GetOTP")
public class GetOTP extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public GetOTP() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		doPost(request, response);
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
```

```
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        String email=request.getParameter("emailid");
        String otpDigits="qw1ertyu72iopas0dfghj3klzxcv8bnm4QWERT96YUI5OPASDFGHJKLZXCVBNM";
        Random random=new Random();
        String otp="";
        for (int i = 0; i < 6; i++) {

    otp+=otpDigits.charAt(random.nextInt(200)%otpDigits.length());
        }
        try {
            DatabaseConnection db=new DatabaseConnection();

            Connection con = db.dbConnect();
            String checkExists="select count(*) from otp_transactional where emailid=?";
            PreparedStatement ps=con.prepareStatement(checkExists);
            ps.setString(1, email);
            ResultSet rs=ps.executeQuery();
            rs.next();
            int exists=rs.getInt(1);
            if(exists > 0)
```

```
                {
                String deleteQuery="delete from otp_transactional where emailid=?";
                ps=con.prepareStatement(deleteQuery);
                ps.setString(1, email);
                ps.executeUpdate();
                }
                String insertQuery="insert into otp_transactional values(?,?,now())";
                ps=con.prepareStatement(insertQuery);
                ps.setString(1, email);
                ps.setString(2, otp);
                ps.executeUpdate();
                con.close();
                mailnow mn=new mailnow();
                String msg="Your OTP is "+otp;
                String sub="Railway Sign Up OTP";
                if(mn.MailNowMsg(email, msg, sub))
                {
                        out.print("true");
                }else{
                        out.print("false");
                }

        } catch (Exception e) {
                // TODO: handle exception
        }
```

}}

GetQuotaListWithID.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import railwayFrequentFunctions.DatabaseConnection;
import railwayFrequentFunctions.DateFunctions;

/**
 * Servlet implementation class GetQuotaListWithId
 */
@WebServlet("/GetQuotaListWithId")
public class GetQuotaListWithId extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public GetQuotaListWithId() {
        super();
        // TODO Auto-generated constructor stub
    }
```

```
/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
	// TODO Auto-generated method stub
doPost(request, response);
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
	// TODO Auto-generated method stub
	Connection con;
	response.setContentType("application/json");
	PrintWriter out=response.getWriter();
	try {
		HttpSession session = request.getSession();
		String sdate = (String) session.getAttribute("date");

		con=(new DatabaseConnection()).dbConnect();
		String query="select quota_id,quota_name from quota_details order by seats_quota desc";
		PreparedStatement ps=con.prepareStatement(query);
		ResultSet rs=ps.executeQuery();
		JSONArray jarr=new JSONArray();
		DateFunctions df=new DateFunctions();
		boolean tatkalAvailable=df.isTatkalAvailable(sdate);
		while(rs.next()){
			if(!tatkalAvailable && rs.getString(1).equalsIgnoreCase("TQ")){
				continue;
			}
			JSONObject jobj=new JSONObject();
			jobj.put("id",rs.getString(1));
			jobj.put("name",rs.getString(2));
			jarr.add(jobj);
```

```
                }
                con.close();
                out.print(jarr);
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }


    }
}
```

GetSeatsandPrice.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.text.NumberFormat;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONObject;

import railwayFrequentFunctions.DatabasePrice;
import railwayFrequentFunctions.SeatAllocation;

/**
 * Servlet implementation class GetSeatsAndPrice
 */
@WebServlet("/GetSeatsAndPrice")
public class GetSeatsAndPrice extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
```

```
    */
    public GetSeatsAndPrice() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            try {
                    HttpSession session = request.getSession();
                    String trainno = (String) session.getAttribute("trainno");
                    String trainreturn = (String) session.getAttribute("trainreturn");
                    String classid = (String) session.getAttribute("classid");
                    String routeid = (String) session.getAttribute("routeId");
                    String fromid = (String) session.getAttribute("source");
                    String toid = (String) session.getAttribute("destination");
                    String quotaid = request.getParameter("quotaid");
                    String sdate = (String) session.getAttribute("date");
                    int cnfGn = Integer.parseInt(request.getParameter("cnfGn"));
                    int rac = Integer.parseInt(request.getParameter("rac"));
                    int wl = Integer.parseInt(request.getParameter("wl"));
                    int tqwl = Integer.parseInt(request.getParameter("tqwl"));
                    int cnfQuot = Integer.parseInt(request.getParameter("cnfQuot"));


                    SeatAllocation sa = new SeatAllocation(trainno, trainreturn,
                                    classid, routeid, fromid, toid,sdate);
```

```
                DatabasePrice dp=new DatabasePrice();
                double price=dp.getPriceWithoutTax(fromid, toid, routeid,
trainno, trainreturn, classid, quotaid);

                NumberFormat formatt = NumberFormat.getInstance();
                String priceFormatted = formatt.format(price);

                PrintWriter out = response.getWriter();
                response.setContentType("application/json");
                JSONObject jobj=new JSONObject();
                jobj.put("seatavailable",
(sa.getAvailableSeats(quotaid,cnfGn,rac,wl,tqwl,cnfQuot).split("/////"))[0]);
                jobj.put("price", priceFormatted);

                out.print(jobj);
        } catch (Exception e) {
                e.printStackTrace();
        }

    }

}
```

InsertPassengers.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;


import java.util.Date;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import railwayFrequentFunctions.DatabaseConnection;
import railwayFrequentFunctions.DatabasePrice;
import railwayFrequentFunctions.PNR;
import railwayFrequentFunctions.SeatAllocation;

/**
 * Servlet implementation class InsertPassengers
 */
@WebServlet("/InsertPassengers")
public class InsertPassengers extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public InsertPassengers() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     *       @see       HttpServlet#doGet(HttpServletRequest       request,
HttpServletResponse response)
     */
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        try {
                PrintWriter out = response.getWriter();
                out.print("<center><h2
style='color:#00788b'>40%</h2></center>");
                DatabasePrice dp=new DatabasePrice();
                Connection con=(new DatabaseConnection()).dbConnect();
                HttpSession session = request.getSession();
                String tno = (String) session.getAttribute("trainno");
                String treturn = (String) session.getAttribute("trainreturn");
                String src = (String) session.getAttribute("source");
                String dest = (String) session.getAttribute("destination");
                String userid = (String) session.getAttribute("userid");
                String classid = (String) session.getAttribute("classid");
                String routeid = (String) session.getAttribute("routeId");
                String sdate = (String) session.getAttribute("date");
                String pnr = (new PNR()).getPNR();
                session.setAttribute("pnr", pnr);

                String insertBookedCustomerQuery = "INSERT INTO
booked_customer_details VALUES (?, ?, ?, ?,"
                                + " ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
                PreparedStatement ps;
                ResultSet rs;
                JSONArray dataInJSON = (JSONArray)
session.getAttribute("dataInJSON");
                JSONArray dateTimeInJSON = (JSONArray)
session.getAttribute("dateTimeInJSON");
                JSONObject srcOfUser = (JSONObject) dateTimeInJSON.get(1);
                JSONObject destOfUser = (JSONObject) dateTimeInJSON.get(2);
```

```
                JSONObject srcOfTrain = (JSONObject) dateTimeInJSON.get(0);
                JSONObject destOfTrain = (JSONObject) dateTimeInJSON.get(3);

                Date srcTime=(Date)srcOfUser.get("stationtime");
                Date destTime=(Date)destOfUser.get("stationtime");
                Date srcTrainTime=(Date)srcOfTrain.get("stationtime");
                Date destTrainTime=(Date)destOfTrain.get("stationtime");
                java.sql.Timestamp srcSqlTime=new java.sql.Timestamp(srcTime.getTime());
                java.sql.Timestamp destSqlTime=new java.sql.Timestamp(destTime.getTime());
                java.sql.Timestamp srcTrainSqlTime=new java.sql.Timestamp(srcTrainTime.getTime());
                java.sql.Timestamp destTrainSqlTime=new java.sql.Timestamp(destTrainTime.getTime());
                JSONObject dataINJSONObject;
                int counter=0;
                JSONArray newDataInJSON=new JSONArray();
                for (Object dataInObject : dataInJSON) {
                    SeatAllocation sa=new SeatAllocation(tno, treturn, classid, routeid, src, dest,sdate);
                    dataINJSONObject=(JSONObject)dataInObject;
                    String quotaId=dataINJSONObject.get("quota").toString();
                    String totalStatus=dataINJSONObject.get("status").toString();
                    String pref=dataINJSONObject.get("pref").toString();
                    Double priceWithoutTax=(Double)dataINJSONObject.get("priceWithoutTax");
                    JSONArray taxNamePlusPercentPlusTaxedPrice = dp.getPriceWithTax(priceWithoutTax);
                    JSONObject priceWithTax = (JSONObject) taxNamePlusPercentPlusTaxedPrice

                .get(taxNamePlusPercentPlusTaxedPrice.size() - 1);
                    double taxedPrice = (double) priceWithTax.get("pricewithtax");
                    String[] formattedStatus=sa.getFormattedStatus(totalStatus);
                    String statusId=formattedStatus[1];
                    if(formattedStatus[1].equalsIgnoreCase("TQWL")){
```

```
                        statusId="WL";
                    }
                    ps=con.prepareStatement(insertBookedCustomerQuery);
                    ps.setString(1, userid);
                    ps.setString(2, pnr);
                    ps.setString(3, dataINJSONObject.get("name").toString());

   ps.setInt(4,Integer.parseInt(dataINJSONObject.get("age").toString()));
                    ps.setString(5,quotaId);
                    ps.setString(6,classid);
                    ps.setInt(7, (counter+1));
                    ps.setString(8,statusId);
                    ps.setInt(9,Integer.parseInt(formattedStatus[2].trim()));
                    ps.setString(10,statusId);
                    ps.setInt(11,Integer.parseInt(formattedStatus[2].trim()));
                    ps.setString(12,
dataINJSONObject.get("gender").toString());
                    ps.setTimestamp(13, srcSqlTime);
                    ps.setTimestamp(14, destSqlTime);
                    ps.setDouble(15, taxedPrice);
                    ps.executeUpdate();
                    JSONObject
newDataINJSONObject=sa.insertIntoStatusTable(formattedStatus,          pnr,
quotaId, pref,
                                        (counter+1),                   con,
ps,dataINJSONObject,srcTrainSqlTime,destTrainSqlTime);
                    newDataInJSON.add(newDataINJSONObject);
                    counter++;
                }
                con.close();
                session.setAttribute("dataInJSON",newDataInJSON);

                RequestDispatcher
rd=request.getRequestDispatcher("../railway/ticketGeneration");
                rd.forward(request, response);

        } catch (Exception e) {
                e.printStackTrace();
                // TODO: handle exception
        }
   }
```

}

LoginAuthentication.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;

import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import railwayFrequentFunctions.DatabaseConnection;
import railwayFrequentFunctions.SecretKey;

/**
 * Servlet implementation class LoginAuthentication
 */
@WebServlet("/LoginAuthentication")
public class LoginAuthentication extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public LoginAuthentication() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
```

```
    * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
    */
   protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
   }

   /**
    * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
    */
   protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        String userid=request.getParameter("userid");
        String password=request.getParameter("password");
        try {
                DatabaseConnection db=new DatabaseConnection();

                Connection con = db.dbConnect();
                String getFname="select full_name from user_details where
userid=?";
                PreparedStatement ps=con.prepareStatement(getFname);
                ps.setString(1, userid);
                ResultSet rs=ps.executeQuery();
                if(rs.next()){
                String selectQuery="select AES_DECRYPT(password,?) as
password,role from user_personal where userid=?";
                String fname=rs.getString(1);
                String key=(new SecretKey()).getSecretKey(userid,
fname.length());
                ps=con.prepareStatement(selectQuery);
                ps.setString(1, key);
                ps.setString(2, userid);
                rs=ps.executeQuery();
                rs.next();
                String pass="";

                pass=rs.getString(1);
```

```
                String role=rs.getString(2);

                if(pass.equals(password) && role.equals("customer"))
                {
                        HttpSession session=request.getSession();
                        session.setAttribute("userid", userid);
                        session.setAttribute("fname", fname);
                        session.setAttribute("role", role);
                        response.sendRedirect("../Phase2/viewdetails.jsp");

                }else{
                        out.println("<script src='../js/jquery.js'></script>");
                        out.println("<script
src='../js/HtmlandCommon.js'></script>");

   out.println("<script>$(document).ready(function(){alertBox('Wrong
Password!');});</script>");
                        RequestDispatcher
rd=request.getRequestDispatcher("../Phase1/index.jsp");
                        rd.include(request, response);
                        }
                }else{
                        out.println("<script src='../js/jquery.js'></script>");
                        out.println("<script
src='../js/HtmlandCommon.js'></script>");

   out.println("<script>$(document).ready(function(){alertBox('Wrong
Username!');});</script>");
                        RequestDispatcher
rd=request.getRequestDispatcher("../Phase1/index.jsp");
                        rd.include(request, response);
                }

                con.close();
        }catch(Exception e)
        {
                e.printStackTrace();
        }

   }

}
```

pdfGenerationforCancellation.java

```
package servletss;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import railwayFrequentFunctions.DatabaseConnection;
import railwayFrequentFunctions.DatabasePrice;
import railwayFrequentFunctions.DateTimeofStation;
import railwayFrequentFunctions.TableGenerator;
import railwayFrequentFunctions.pdfGeneration;
import railwayFrequentFunctions.properFormatForPDF;

/**
 * Servlet implementation class pdfGenerationForCancellation
 */
@WebServlet("/pdfGenerationForCancellation")
public class pdfGenerationForCancellation extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public pdfGenerationForCancellation() {
```

```
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse
     *  response)
     */
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        // TODO Auto-generated method stub
        doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse
     *  response)
     */
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        // TODO Auto-generated method stub
        try {

            PrintWriter out = response.getWriter();
            DatabasePrice dp=new DatabasePrice();
            HttpSession session = request.getSession();
            String userid = (String) session.getAttribute("userid");
            String fname = (String) session.getAttribute("fname");
            String pnr = (String) request.getParameter("pnr");

            Connection con = (new DatabaseConnection()).dbConnect();
            String userDetailQuery = "SELECT email,address,mobile_no
FROM user_details WHERE userid=?";
            PreparedStatement ps =
con.prepareStatement(userDetailQuery);
            ps.setString(1, userid);
            ResultSet rs = ps.executeQuery();
            rs.next();
```

```
                String to = rs.getString(1);
                String address = rs.getString(2);
                String mobile = rs.getString(3);
                String taxSumQuery="SELECT sum(`tax_price`) FROM `tax`";
                ps = con.prepareStatement(taxSumQuery);
                rs = ps.executeQuery();
                rs.next();
                double taxSum=rs.getDouble(1);
                String src="", dest="", sdate="";
                String filename = "", qrLocation = "", seatMapLocation = "",
classid = "";
                String[] headers;
                String[][] data;
                String pdfData = "";
                Double totalPrice=0.0;
                TableGenerator tg = new TableGenerator();
                String css = ".tabularDesign{border-collapse: collapse;width:
100%;font-size: 10px;font-family:Times New Roman;}"
                                + ".tabularDesign th,.tabularDesign td {text-align:
left;padding: 8px;}"
                                + ".tabularDesign .even{background-color:
#f2f2f2}"
                                + ".tabularDesign th{background-color:
#00788b;color: white;}"
                                + ".tabularDesign tr{color:#404040;}";

                css += ".nonTabularDesign{border-collapse: collapse;width:
100%;font-size: 10px;float:right;}"
                                + ".nonTabularDesign td {text-align: left;padding:
8px;}"
                                + "table td p {text-align: right;color: #07788b;font-
weight: bold;}";
                css += "p {color: #07788b;font-weight: bold;text-align: center;}";

                pdfData = "<p>Ticket and Login User Details : </p><br/>";
                headers = new String[] {};
                data = new String[][] { { "<p>PNR No : </p>", pnr },
                                { "<p>User Id : </p>", userid },
                                { "<p>Name : </p>", fname },
                                { "<p>Address : </p>", address },
                                { "<p>Mobile : </p>", mobile },
                                { "<p>Email Id : </p>", to } };
```

```
String query = "SELECT `PNR`,`cust_name`, `age`,`quota_id`, "
+ "`class_id`, `PNR_counter`, `status_id`, "
+ "`status_counter`, `gender`, `src_start_time`,
`dest_end_time`, "
+ "`cost_with_tax`,(SELECT `quota_name` FROM
`quota_details` WHERE `quota_id`=b.quota_id), "
+ "(SELECT `class_name` FROM
`train_class_details` WHERE `class_id`=b.`class_id`), "
+ "(SELECT `coach_code` FROM
`train_class_details` WHERE `class_id`=b.`class_id`), "
+ "(SELECT `status_name` FROM
`seat_status_details` WHERE `status_id`=b.`status_id`) "
+ "FROM `booked_customer_details` b where
`userid`=? AND `PNR`=? ORDER BY `PNR_counter`";
ps = con.prepareStatement(query);
ps.setString(1, userid);
ps.setString(2, pnr);
rs = ps.executeQuery();
int initialCounter = 0,tno=0, rid=0,treturn=0;
String tname="";
ArrayList<String[]> passengerDatas=new ArrayList<String[]>();
while (rs.next()) {
String statusTableQuery="";
if(!rs.getString(7).equalsIgnoreCase("wl")){
statusTableQuery="SELECT (SELECT train_name from
train_master t where t.train_no=s.train_no AND "
+ "t.train_return=s.train_return),(SELECT
station_name from station_master s where s.station_code=from_station),"
+ "(SELECT station_name from
station_master s where s.station_code=to_station), "
+ "`coach_counter`, `seat_no`,
`train_src_time`, `train_dest_time` ,(SELECT `berth_id` FROM `berth_details`
"
+ "WHERE `berth_id` IN (SELECT
`berth_id` FROM "
+ rs.getString(5)
+ " WHERE `seat_no`=s.`seat_no`)), "
+ "(SELECT `berth_name` FROM
`berth_details` "
+ "WHERE `berth_id` IN (SELECT
`berth_id` FROM "
```

```
                                        + rs.getString(5)
                                        + " WHERE `seat_no`=s.`seat_no`)),
`train_no`, `train_return`, (SELECT route_id from train_master t where
t.train_no=s.train_no AND "
                                        + "t.train_return=s.train_return),
`from_station`, `to_station` "
                                        + "FROM "
                                        + rs.getString(7)
                                        + " s WHERE PNR=? AND
PNR_counter=?";}
                        else{
                                statusTableQuery="SELECT (SELECT train_name
from train_master t where t.train_no=s.train_no AND "
                                                +
"t.train_return=s.train_return),(SELECT station_name from station_master s
where s.station_code=from_station),"
                                                + "(SELECT station_name from
station_master s where s.station_code=to_station), "
                                                + "0, 0, `train_src_time`,
`train_dest_time` ,'null', "
                                                + "'null', `train_no`, `train_return`,
(SELECT route_id from train_master t where t.train_no=s.train_no AND "
                                                + "t.train_return=s.train_return),
`from_station`, `to_station` "
                                                + "FROM "
                                                + rs.getString(7)
                                                + " s WHERE PNR=? AND
PNR_counter=?";
                        }
                        PreparedStatement ps1 =
con.prepareStatement(statusTableQuery);
                        ps1.setString(1, rs.getString(1));
                        ps1.setInt(2, rs.getInt(6));
                        ResultSet rs1 = ps1.executeQuery();

                        while (rs1.next()) {

                                if (initialCounter == 0) {

                                        classid = rs.getString(5);
```

```
filename =
"E:\\JSP\\RAILWAY_PROJECT\\WebContent\\WEB-
INF\\GeneratedDocuments\\"
+ to.replace(".", "") + ".pdf";
qrLocation =
"E:\\JSP\\RAILWAY_PROJECT\\WebContent\\WEB-
INF\\GeneratedDocuments\\"
+ to.replace(".", "") + ".png";
seatMapLocation =
"E:\\JSP\\RAILWAY_PROJECT\\WebContent\\images\\"
+ classid + ".gif";
if (new File(filename).exists()) {
new File(filename).delete();
}
pdfData += tg.MsgGeneratortable(headers,
data,
"nonTabularDesign", true);
tno=rs1.getInt(10);
treturn=rs1.getInt(11);
tname=rs1.getString(1);
src=rs1.getString(13);
dest=rs1.getString(14);
rid=rs1.getInt(12);
sdate=rs1.getString(6).split(" ")[0];
JSONArray dateTimeInJSON = new
DateTimeofStation().getDateAndTimeOfStation(src, dest, String.valueOf(tno),
String.valueOf(treturn), sdate, String.valueOf(rid));

JSONObject srcOfTrain = (JSONObject)
dateTimeInJSON.get(0);
JSONObject srcOfUser = (JSONObject)
dateTimeInJSON.get(1);
JSONObject destOfUser = (JSONObject)
dateTimeInJSON.get(2);
JSONObject destOfTrain = (JSONObject)
dateTimeInJSON.get(3);
pdfData += "<p><br/>Train Details
:</p><br/>";
headers = new String[] {};
data = new String[][] {
{ "<p>Train --> </p>",
"<p>No : </p>", String.valueOf(tno),
```

```
                                                    "<p>Name        :
</p>", tname, "", "" },
                                                {
                                                    "<p>Source    -->
</p>",
                                                    "<p>Station
Name : </p>",
    srcOfTrain.get("stationname")
    .toString(),
                                                    "<p>Departure
Time : </p>",
    srcOfTrain.get("stationtime")
    .toString(),
                                                    "<p>Distance    :
</p>",
    srcOfTrain.get("distance").toString() },
                                                {
                                                    "<p>Destination
--> </p>",
                                                    "<p>Station
Name : </p>",
    destOfTrain.get("stationname")
    .toString(),
                                                    "<p>Arrival
Time : </p>",
    destOfTrain.get("stationtime")
    .toString(),
                                                    "<p>Distance    :
</p>",
    destOfTrain.get("distance").toString() } };
                                    pdfData += tg.MsgGeneratortable(headers,
data, "nonTabularDesign",
```

```
                                        true);

                                pdfData += "<p><br/>Passeneger Travel
Details :</p><br/>";
                                headers = new String[] {};
                                data = new String[][] {
                                        {
                                                "<p>Source -->
</p>",
                                                "<p>Station
Name : </p>",
   srcOfUser.get("stationname").toString(),
                                                "<p>Departure
Time : </p>",
   srcOfUser.get("stationtime").toString(),
                                                "<p>Distance
From <br/>"
                                                        +
srcOfTrain.get("stationname").toString()
                                                        + "
: </p>",
   srcOfUser.get("distance").toString() },
                                        {
                                                "<p>Destination
--> </p>",
                                                "<p>Station
Name : </p>",
   destOfUser.get("stationname").toString(),
                                                "<p>Arrival
Time : </p>",
   destOfUser.get("stationtime").toString(),
                                                "<p>Distance
From <br/>"
                                                        +
srcOfTrain.get("stationname").toString()
                                                        + "
: </p>",
```

```
   destOfUser.get("distance").toString() } };

                                    pdfData += tg.MsgGeneratortable(headers,
data, "nonTabularDesign",
                                                    true);

                                    pdfData += "<br/><p>Passenger Details
:</p><br/>";
                                    headers = new String[] { "Status", "Quota",
"Class", "Seat No",
                                "Berth", "Coach", "Name", "Age", "Gender", "Ticket
Cost" };


                                }
                                passengerDatas.add(new
String[]{rs.getString(16)+" ("+rs.getString(7)+" "+rs.getInt(8)+")",

   rs.getString(13)+"("+rs.getString(4)+")",rs.getString(14)+"("+rs.getString(5)+
")",

   ""+rs1.getInt(5),rs1.getString(9)+"("+rs1.getString(8)+")",rs.getString(15)+rs1
.getInt(4),

   rs.getString(2),""+rs.getInt(3),""+rs.getString(9),

   dp.formatCurrency(Math.round((rs.getDouble(12)/(1+(taxSum/100)))*10)/
10D)
                                });

   totalPrice+=Math.round((rs.getDouble(12)/(1+(taxSum/100)))*10)/10D;

                        }
                        initialCounter++;
                }

                query="SELECT `tax_id`,`tax_price` FROM `tax`";
                ps = con.prepareStatement(query);

                rs = ps.executeQuery();
                while(rs.next()){
```

```
                passengerDatas.add(new String[] {
                                "",
                                "",
                                "",
                                "",
                                "",
                                "",
                                "",
                                "",
                                "<p>" + rs.getString(1)
                                                + " : </p>",
                                rs.getDouble(2) + "%" });
                }
                double
totalTaxedAmount=(totalPrice+(totalPrice*taxSum/100));
                passengerDatas.add(new String[] {
                                "",
                                "",
                                "",
                                "",
                                "",
                                "",
                                "",
                                "",
                                "<p>Total Cost : </p>",

   dp.formatCurrency(Math.round(totalTaxedAmount*100)/100D) });
                data = new String[passengerDatas.size()][];
                int k=0;
                for (String[] datass : passengerDatas) {
                        data[k]=datass;
                        k++;
                }
                                pdfData  +=  tg.MsgGeneratortable(headers,
data, "tabularDesign",
                                true);

                // out.print(pdfData);
                pdfGeneration pg = new pdfGeneration(filename, pdfData,
                                seatMapLocation, pnr, qrLocation, css, to.length());

                con.close();
```

```
		} catch (Exception e) {
			e.printStackTrace();
		}

	}

}
```

Receivechatfromemergency.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import railwayFrequentFunctions.DatabaseConnection;

/**
 * Servlet implementation class receiveChatFromEmergency
 */
@WebServlet("/receiveChatFromEmergency")
public class receiveChatFromEmergency extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public receiveChatFromEmergency() {
```

```
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            response.setContentType("application/json");
            HttpSession session=request.getSession();
            PrintWriter out=response.getWriter();
            JSONArray jarr=new JSONArray();
            try{
            Connection con=(new DatabaseConnection()).dbConnect();
            String query="SELECT `message`,`counter` FROM `emergency` WHERE `counter`> ? AND `to_id`=? ;";
            PreparedStatement ps=con.prepareStatement(query);

    ps.setInt(1,Integer.parseInt((String)request.getParameter("lastReceivedID")));
            ps.setString(2, (String)session.getAttribute("userid"));
            ResultSet rs=ps.executeQuery();

            while(rs.next()){
                    JSONObject jobj=new JSONObject();
                    jobj.put("message",rs.getString(1));
                    jobj.put("counter",rs.getInt(2));
                    jarr.add(jobj);
            }
```

```
            con.close();
            out.print(jarr);
            }catch(Exception e){
                    e.printStackTrace();
                    out.print(jarr);
            }

    }

}
```

RouteDisplay.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.json.simple.JSONArray;

import railwayFrequentFunctions.DatabaseConnection;

/**
 * Servlet implementation class RouteDisplay
 */
@WebServlet("/RouteDisplay")
public class RouteDisplay extends HttpServlet {
    private static final long serialVersionUID = 1L;
```

```
/**
 * @see HttpServlet#HttpServlet()
 */
public RouteDisplay() {
    super();
    // TODO Auto-generated constructor stub
}

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.setContentType("application/json");
        PrintWriter out=response.getWriter();
        Connection con;
        Date d;
        Calendar calendar = Calendar.getInstance();
        JSONArray data=new JSONArray();
        int trainno=Integer.parseInt(request.getParameter("trainNo").trim());
        int trainreturn=Integer.parseInt(request.getParameter("returnRoute"));
        String selectedDate=request.getParameter("selectedDate");
        try {
con = (new DatabaseConnection()).dbConnect();
String routedetails = "SELECT (SELECT station_name FROM station_master sr where "
                + "sr.station_code = rm.station_code) AS station_name,distance_from_prev from route_master rm where "
                + "rm.station_code IN (SELECT station_code FROM route_master WHERE route_id=(SELECT route_id from "
```

```
                + "train_master where train_no=? AND train_return=?) AND
counter >= (SELECT counter from route_master "
                + "where route_id=(SELECT route_id from train_master where
train_no=? AND train_return=?) and "
                + "station_code=(SELECT source_station_code FROM
`train_master` WHERE train_no=? AND train_return=?)) "
                + "AND counter <= (SELECT counter from route_master where
route_id=(SELECT route_id from train_master "
                + "where train_no=? AND train_return=?) and
station_code=(SELECT dest_station_code FROM train_master "
                + "WHERE train_no=? AND train_return=?)) ORDER BY
route_id,counter) AND route_id = (SELECT route_id "
                + "from train_master where train_no=? AND train_return=?)";
   String traindetails="SELECT train_speed,train_time FROM
train_normal_schedule"
                + " WHERE train_no=? AND train_return=?";
   Double speed=0.0,distance=0.0;
   int timeinsecs=0;
   String timestamp="";
   String stationName="";
        PreparedStatement ps=con.prepareStatement(traindetails);
        ps.setInt(1, trainno);
        ps.setInt(2, trainreturn);
        ResultSet rs=ps.executeQuery();
        if(rs.next()){
                speed=rs.getDouble("train_speed");
                timestamp=rs.getString("train_time");
        }
        data.add("Station_name (Distance) [Expected Reaching Time]<br>With
avg Train Speed = "+speed+"km/hr");
        timestamp=selectedDate+" "+timestamp;

        ps=con.prepareStatement(routedetails);
        ps.setInt(1, trainno);
        ps.setInt(2, trainreturn);
        ps.setInt(3, trainno);
        ps.setInt(4, trainreturn);
        ps.setInt(5, trainno);
        ps.setInt(6, trainreturn);
        ps.setInt(7, trainno);
        ps.setInt(8, trainreturn);
        ps.setInt(9, trainno);
```

```
        ps.setInt(10, trainreturn);
        ps.setInt(11, trainno);
        ps.setInt(12, trainreturn);
        rs=ps.executeQuery();
        int loopCounter=0;
        while(rs.next()){
                stationName=rs.getString("station_name");
                if(loopCounter==0){
                distance=0.0;
                }
                else{
                distance=rs.getDouble("distance_from_prev");
                }
                timeinsecs=(int)(Math.round((distance/speed)*3600));
                d=new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").parse(timestamp);
                calendar.setTime(d);
                calendar.add(Calendar.SECOND, timeinsecs);
                timestamp=new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").format(calendar.getTime());
                data.add(stationName+" ("+distance+" km)
["+calendar.getTime()+"]");
                loopCounter++;
        }
        con.close();
        out.print(data);
        } catch (Exception e) {
                e.printStackTrace();
        }

   }}
```

SeatPreferences.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;

import railwayFrequentFunctions.SeatAllocation;

/**
 * Servlet implementation class SeatPreferences
 */
@WebServlet("/SeatPreferences")
public class SeatPreferences extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public SeatPreferences() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.setContentType("application/json");
        PrintWriter out= response.getWriter();
        HttpSession session = request.getSession();
```

```
        String trainno = (String) session.getAttribute("trainno");
        String trainreturn = (String) session.getAttribute("trainreturn");
        String classid = (String) session.getAttribute("classid");
        String routeid = (String) session.getAttribute("routeId");
        String fromid = (String) session.getAttribute("source");
        String toid = (String) session.getAttribute("destination");
        String sdate = (String) session.getAttribute("date");
        SeatAllocation sa = new SeatAllocation(trainno, trainreturn,
                        classid, routeid, fromid, toid,sdate);

        JSONArray jarr=sa.getAvailableBerth();
        out.print(jarr);

    }

}
```

SendchatToemergency.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```
import railwayFrequentFunctions.DatabaseConnection;

/**
 * Servlet implementation class sendChatToEmergency
 */
@WebServlet("/sendChatToEmergency")
public class sendChatToEmergency extends HttpServlet {
   private static final long serialVersionUID = 1L;

   /**
    * @see HttpServlet#HttpServlet()
    */
   public sendChatToEmergency() {
      super();
      // TODO Auto-generated constructor stub
   }

   /**
    * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
    */
   protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
         // TODO Auto-generated method stub
         doPost(request, response);
   }

   /**
    * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
    */
   protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
         // TODO Auto-generated method stub
         response.setContentType("text/html");
         HttpSession session=request.getSession();
         PrintWriter out=response.getWriter();
         try{
         Connection con=(new DatabaseConnection()).dbConnect();
         String query="INSERT INTO `emergency` VALUES (?,NULL,?,?)";
         PreparedStatement ps=con.prepareStatement(query);
```

```
            ps.setString(1,(String)request.getParameter("message"));
            ps.setString(2, "emergency");
            ps.setString(3, (String)session.getAttribute("userid"));
            ps.executeUpdate();
            con.close();
            out.print("Delivered");
            }catch(Exception e){
                    e.printStackTrace();
                    out.print("Not Delivered");
            }
    }

}
```

sendPDFmail.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import railwayFrequentFunctions.DatabaseConnection;
```

```
import railwayFrequentFunctions.emailattachmentrailway;

/**
 * Servlet implementation class SendPDFMail
 */
@WebServlet("/SendPDFMail")
public class SendPDFMail extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public SendPDFMail() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        HttpSession session=request.getSession();
        String userid = (String) session.getAttribute("userid");
        String to ="";
        String subject="AUTO GENERATED RAILWAY TICKET";
        String attachfilename="RAILWAY TICKET";
        boolean flag=false;
        try{
        Connection con = (new DatabaseConnection()).dbConnect();
```

```
        String emailQuery = "SELECT email FROM user_details WHERE
userid=?";
        PreparedStatement ps = con.prepareStatement(emailQuery);
        ps.setString(1, userid);
        ResultSet rs = ps.executeQuery();
        rs.next();
        to = rs.getString(1);
        con.close();
        flag=(new emailattachmentrailway()).createattachandemail(to, subject,
attachfilename);
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.print(String.valueOf(flag));
        }catch(Exception e){
                e.printStackTrace();
        }

   }

}
```

SignUpToDatabase.java

```
package servletss;

import java.io.IOException;
import java.sql.Connection;

import java.sql.PreparedStatement;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

import railwayFrequentFunctions.DatabaseConnection;
import railwayFrequentFunctions.SecretKey;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
/**
 * Servlet implementation class SignUpToDatabase
 */
@WebServlet("/SignUpToDatabase")
public class SignUpToDatabase extends HttpServlet {
   private static final long serialVersionUID = 1L;

   /**
    * @see HttpServlet#HttpServlet()
    */
   public SignUpToDatabase() {
      super();
      // TODO Auto-generated constructor stub
   }

   /**
    * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
    */
   protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
         // TODO Auto-generated method stub
         doPost(request, response);
   }

   /**
    * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
    */
   protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
         // TODO Auto-generated method stub
         try {
                  DatabaseConnection db=new DatabaseConnection();

                  Connection con = db.dbConnect();
                  String role="";
                  String userid=request.getParameter("txt_userid");
                  String pass=request.getParameter("txt_password");

                  String ipaddress=request.getRemoteAddr();
```

```
String fname=request.getParameter("txt_fname");
String mobile=request.getParameter("mobile");
String key=(new SecretKey()).getSecretKey(userid,
fname.length());
DateFormat formatter = new SimpleDateFormat("yyyy-MM-
dd");
Date dobParse=formatter.parse(request.getParameter("DOB"));

String address=request.getParameter("address");
String email=request.getParameter("email");
String district=request.getParameter("district");
String state=request.getParameter("state");
String gender=request.getParameter("gender");
String marital=request.getParameter("marital");
String insertUserTables="insert into user_personal
values(?,AES_ENCRYPT(?,?),?,?)";

PreparedStatement ps=con.prepareStatement(insertUserTables);
ps.setString(1, userid);
ps.setString(2, pass);
ps.setString(3, key);
ps.setString(4, ipaddress);
if(request.getParameter("role")!=null)
{
role=request.getParameter("role");
ps.setString(5, role);
}else{
ps.setString(5, "customer");
}
ps.executeUpdate();

String insertDetailsTables="insert into user_details
values(?,?,?,?,?,?,?,?,?,?)";
ps=con.prepareStatement(insertDetailsTables);

ps.setString(1, userid);
ps.setString(2, fname);
ps.setString(3, mobile);
ps.setString(4, gender);
ps.setString(5, marital);
ps.setDate(6,new java.sql.Date(dobParse.getTime()) );
ps.setString(7, state);
```

```
                ps.setString(8, district);
                ps.setString(9, address);
                ps.setString(10, email);

                ps.executeUpdate();

                con.close();
                RequestDispatcher
rd=request.getRequestDispatcher("../Phase1/index.jsp?success=true");
                rd.forward(request, response);
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }
   }}
```

SourceandDestination.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.json.simple.JSONArray;

import railwayFrequentFunctions.sourceAndDestination;

/**
 * Servlet implementation class SourceAndDestination
 */
@WebServlet("/SourceAndDestination")
public class SourceAndDestination extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public SourceAndDestination() {
```

```
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub

        String src=request.getParameter("src");
        response.setContentType("application/json");
        JSONArray jar=(new sourceAndDestination()).src_dest(src);
        PrintWriter out=response.getWriter();
        out.print(jar);
    }

}
```

StateDistrict.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import org.json.simple.JSONArray;
import railwayFrequentFunctions.State_And_District;
/**
 * Servlet implementation class StateDistrict
 */
@WebServlet("/StateDistrict")
public class StateDistrict extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public StateDistrict() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
    doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        PrintWriter out=response.getWriter();
        response.setContentType("application/json");
        String q=request.getParameter("q");
        String countrycode=request.getParameter("countrycode");
        String featurecode=request.getParameter("featurecode");
        State_And_District sad=new State_And_District();
        JSONArray ja=sad.getData(q, countrycode, featurecode);
```

```
        out.print(ja);

    }

}
```

TicketGeneration.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import railwayFrequentFunctions.DatabaseConnection;
import railwayFrequentFunctions.DatabasePrice;
import railwayFrequentFunctions.PNR;
import railwayFrequentFunctions.emailattachmentrailway;
import railwayFrequentFunctions.pdfGeneration;
import railwayFrequentFunctions.properFormatForPDF;

import java.io.File;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import railwayFrequentFunctions.TableGenerator;

/**
 * Servlet implementation class TicketGeneration
```

```
 */
@WebServlet("/TicketGeneration")
public class TicketGeneration extends HttpServlet {
   private static final long serialVersionUID = 1L;

   /**
    * @see HttpServlet#HttpServlet()
    */
   public TicketGeneration() {
        super();
        // TODO Auto-generated constructor stub
   }

   /**
    *        @see        HttpServlet#doGet(HttpServletRequest        request,
HttpServletResponse
    *     response)
    */
   protected void doGet(HttpServletRequest request,
                HttpServletResponse    response)    throws    ServletException,
IOException {
        // TODO Auto-generated method stub
        doPost(request, response);
   }

   /**
    *        @see        HttpServlet#doPost(HttpServletRequest        request,
HttpServletResponse
    *     response)
    */
   protected void doPost(HttpServletRequest request,
                HttpServletResponse    response)    throws    ServletException,
IOException {
        // TODO Auto-generated method stub
        try {
                PrintWriter out = response.getWriter();
                out.print("<center><h2
style='color:#00788b'>80%</h2></center>");
                HttpSession session = request.getSession();
                String tno = (String) session.getAttribute("trainno");
                String treturn = (String) session.getAttribute("trainreturn");
                String src = (String) session.getAttribute("source");
```

```
String dest = (String) session.getAttribute("destination");
String userid = (String) session.getAttribute("userid");
String fname = (String) session.getAttribute("fname");
String pnr = (String) session.getAttribute("pnr");
String classid = (String) session.getAttribute("classid");
Connection con = (new DatabaseConnection()).dbConnect();
String userDetailQuery = "SELECT email,address,mobile_no FROM user_details WHERE userid=?";
PreparedStatement ps = con.prepareStatement(userDetailQuery);
ps.setString(1, userid);
ResultSet rs = ps.executeQuery();
rs.next();
String to = rs.getString(1);
String address = rs.getString(2);
String mobile = rs.getString(3);



String trainDetailQuery = "SELECT train_name FROM train_master WHERE train_no=? and train_return=?";
ps = con.prepareStatement(trainDetailQuery);
ps.setInt(1, Integer.parseInt(tno));
ps.setInt(2, Integer.parseInt(treturn));
rs = ps.executeQuery();
rs.next();
String tname = rs.getString(1);

String filename = "E:\\JSP\\RAILWAY_PROJECT\\WebContent\\WEB-INF\\GeneratedDocuments\\"
        + to.replace(".", "") + ".pdf";
String qrLocation = "E:\\JSP\\RAILWAY_PROJECT\\WebContent\\WEB-INF\\GeneratedDocuments\\"
        + to.replace(".", "") + ".png";
String seatMapLocation = "E:\\JSP\\RAILWAY_PROJECT\\WebContent\\images\\"
        + classid + ".gif";
if (new File(filename).exists()) {
    new File(filename).delete();
}
```

```
String[] headers;
String[][] data;
String pdfData = "";
TableGenerator tg = new TableGenerator();
DatabasePrice dp = new DatabasePrice();
String css = ".tabularDesign{border-collapse: collapse;width: 100%;font-size: 10px;font-family:Times New Roman;}"
        + ".tabularDesign th,.tabularDesign td {text-align: left;padding: 8px;}"
        + ".tabularDesign .even{background-color: #f2f2f2}"
        + ".tabularDesign th{background-color: #00788b;color: white;}"
        + ".tabularDesign tr{color:#404040;}";

css += ".nonTabularDesign{border-collapse: collapse;width: 100%;font-size: 10px;float:right;}"
        + ".nonTabularDesign td {text-align: left;padding: 8px;}"
        + "table td p {text-align: right;color: #07788b;font-weight: bold;}";
css += "p {color: #07788b;font-weight: bold;text-align: center;}";

pdfData = "<p>Ticket and Login User Details : </p><br/>";
headers = new String[] {};
data = new String[][] { { "<p>PNR No : </p>", pnr },
        { "<p>User Id : </p>", userid },
        { "<p>Name : </p>", fname },
        { "<p>Address : </p>", address },
        { "<p>Mobile : </p>", mobile },
        { "<p>Email Id : </p>", to } };
pdfData += tg.MsgGeneratortable(headers, data, "nonTabularDesign",
        true);

JSONArray dateTimeInJSON = (JSONArray) session
        .getAttribute("dateTimeInJSON");
JSONObject srcOfTrain = (JSONObject) dateTimeInJSON.get(0);
JSONObject srcOfUser = (JSONObject) dateTimeInJSON.get(1);
JSONObject destOfUser = (JSONObject) dateTimeInJSON.get(2);
```

```
                JSONObject destOfTrain = (JSONObject)
dateTimeInJSON.get(3);
                pdfData += "<p><br/>Train Details :</p><br/>";
                headers = new String[] {};
                data = new String[][] {
                                { "<p>Train --> </p>", "<p>No : </p>", tno,
                                                "<p>Name : </p>", tname, "", "" },
                                { "<p>Source --> </p>", "<p>Station Name :
</p>",

   srcOfTrain.get("stationname").toString(),
                                                "<p>Departure Time : </p>",

   srcOfTrain.get("stationtime").toString(),
                                                "<p>Distance : </p>",
                                                srcOfTrain.get("distance").toString()
},
                                { "<p>Destination --> </p>", "<p>Station Name :
</p>",

   destOfTrain.get("stationname").toString(),
                                                "<p>Arrival Time : </p>",

   destOfTrain.get("stationtime").toString(),
                                                "<p>Distance : </p>",
                                                destOfTrain.get("distance").toString()
} };
                pdfData += tg.MsgGeneratortable(headers, data,
"nonTabularDesign",
                                true);

                pdfData += "<p><br/>Passeneger Travel Details :</p><br/>";
                headers = new String[] {};
                data = new String[][] {
                                { "<p>Source --> </p>", "<p>Station Name :
</p>",
                                                srcOfUser.get("stationname").toString(),
                                                "<p>Departure Time : </p>",
                                                srcOfUser.get("stationtime").toString(),
                                                "<p>Distance From
<br/>"+srcOfTrain.get("stationname").toString()+" : </p>",
                                                srcOfUser.get("distance").toString() },
```

```
                    { "<p>Destination --> </p>", "<p>Station Name : </p>",
                                destOfUser.get("stationname").toString(),
                                "<p>Arrival Time : </p>",
                                destOfUser.get("stationtime").toString(),
                                "<p>Distance From
<br/>"+srcOfTrain.get("stationname").toString()+" : </p>",
                                destOfUser.get("distance").toString() } };

            pdfData += tg.MsgGeneratortable(headers, data,
"nonTabularDesign",
                        true);

            pdfData += "<br/><p>Passenger Details :</p><br/>";

            headers = new String[] { "Status", "Quota","Class", "Seat No",
"Berth",
                        "Coach", "Name", "Age", "Gender", "Ticket Cost" };
            JSONArray dataInJSON = (JSONArray) session
                        .getAttribute("dataInJSON");
            JSONArray taxNamePlusPercentPlusTaxedPriceInJSON =
(JSONArray) session

   .getAttribute("taxNamePlusPercentPlusTaxedPriceInJSON");
            data = new String[dataInJSON.size()
                        +
taxNamePlusPercentPlusTaxedPriceInJSON.size()][];


            int count = 0;
            for (Object userData : dataInJSON) {
                  JSONObject userDataObj = (JSONObject) userData;
                  String[] formattedPDFData=(new
properFormatForPDF())

   .getProperFormatForPDF(userDataObj.get("quota").toString().trim(),
classid.trim(),

   userDataObj.get("berthId").toString().trim(),

   userDataObj.get("coachNo").toString().trim());
                  data[count] = new String[] {
```

```
(userDataObj.get("status").toString().split("/////"))[0],
                                        formattedPDFData[0],
                                        formattedPDFData[1],
                                        userDataObj.get("seatNo").toString(),
                                        formattedPDFData[2],
                                        formattedPDFData[3],
                                        userDataObj.get("name").toString(),
                                        userDataObj.get("age").toString(),
                                        userDataObj.get("gender").toString(),

    dp.formatCurrency(Double.parseDouble(userDataObj.get(

    "priceWithoutTax").toString())) };
                            count++;
                    }

                    for (Object userPriceData :
taxNamePlusPercentPlusTaxedPriceInJSON) {
                            JSONObject userPriceDataObj = (JSONObject)
userPriceData;
                            if ((data.length - 1) == count) {
                                    data[count] = new String[] {
                                                    "",
                                                    "",
                                                    "",
                                                    "",
                                                    "",
                                                    "",
                                                    "",
                                                    "",
                                                    "<p>Total Cost : </p>",
                                                    dp.formatCurrency(Double

    .parseDouble(userPriceDataObj.get(

    "pricewithtax").toString())) };
                            } else {
                                    data[count] = new String[] {
                                                    "",
                                                    "",
                                                    "",
```

```
                                                ",
                                                ",
                                                ",
                                                ",
                                                ",
                                                "<p>"                                   +
userPriceDataObj.get("taxid").toString()
                                                                        + " : </p>",

   userPriceDataObj.get("taxpercent").toString() + "%" };

                        }
                        count++;
                }
                pdfData        +=        tg.MsgGeneratortable(headers,        data,
"tabularDesign",
                                true);
                //out.print(pdfData);
                pdfGeneration pg = new pdfGeneration(filename, pdfData,
seatMapLocation ,pnr,
                                qrLocation, css,to.length());

                con.close();
        } catch (Exception e) {
                e.printStackTrace();
        }
        response.sendRedirect("../Phase5/booked.jsp");
        /*RequestDispatcher
rd=request.getRequestDispatcher("../Phase5/booked.jsp");
        rd.forward(request, response);*/
   }

}
```

TrainSelectiontoClassSelection.java

```
package servletss;

import java.io.IOException;
import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import railwayFrequentFunctions.DatabaseConnection;

/**
 * Servlet implementation class TrainSelectionToClassSelection
 */
@WebServlet("/TrainSelectionToClassSelection")
public class TrainSelectionToClassSelection extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public TrainSelectionToClassSelection() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        HttpSession session=request.getSession();
        String date=request.getParameter("selectedDatee");
        String srcStation=request.getParameter("selectedSrcStation1");
        String destStation=request.getParameter("selectedDestStation1");
        String trainNo=request.getParameter("selectedTrainno");
        String trainReturn=request.getParameter("selectedTrainreturn");
        session.setAttribute("date",date);
        session.setAttribute("trainno",trainNo);
        session.setAttribute("trainreturn",trainReturn);
        try{
                Connection con=(new DatabaseConnection()).dbConnect();
                String routeidquery="SELECT route_id FROM train_master WHERE train_no=? and train_return=?";
                PreparedStatement ps=con.prepareStatement(routeidquery);
                ps.setInt(1,Integer.parseInt(trainNo));
                ps.setInt(2,Integer.parseInt(trainReturn));
                ResultSet rs=ps.executeQuery();
                rs.next();
                String routeId=""+rs.getInt(1);

                String fromIdQuery="SELECT station_code FROM station_master WHERE station_name=?";
                ps=con.prepareStatement(fromIdQuery);
                ps.setString(1, srcStation);
                rs=ps.executeQuery();
                rs.next();
                String fromId=rs.getString(1);

                String toIdQuery="SELECT station_code FROM station_master WHERE station_name=?";
                ps=con.prepareStatement(toIdQuery);
                ps.setString(1, destStation);
                rs=ps.executeQuery();
                rs.next();
                String toId=rs.getString(1);

                session.setAttribute("routeId",routeId);
                session.setAttribute("source",fromId);
                session.setAttribute("destination",toId);
```

```
                con.close();
        }catch(Exception e){
                e.printStackTrace();
        }
        response.sendRedirect("../Phase2/classselection.jsp");
    }

}
```

ValidationCheck.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import railwayFrequentFunctions.DatabaseConnection;

/**
 * Servlet implementation class ValidationCheck
 */
@WebServlet("/ValidationCheck")
public class ValidationCheck extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ValidationCheck() {
        super();
        // TODO Auto-generated constructor stub
```

```
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse
     * response)
     */
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        // TODO Auto-generated method stub
        doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse
     * response)
     */
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        // TODO Auto-generated method stub
        Connection con;
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        try {
            if (request.getParameterNames().nextElement().toString()
                    .equals("userid")) {
                String value = request.getParameter("userid").trim();
                DatabaseConnection db=new DatabaseConnection();

                con = db.dbConnect();
                Statement stmt = con.createStatement();
                String query = "select count(*) from user_personal where
userid='" + value+"'";
                ResultSet rs = stmt.executeQuery(query);
                rs.next();
                int rowno=rs.getInt(1);
                if(rowno==0)
                {
                out.print("true");
```

```
                    }else{
                    out.print("false");
                    }
                    con.close();
                }
            } catch (Exception e) {

            }
        }

    }
```

3. Android Codes for Camera Scanner of QR Code :

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:textAlignment="center"
    tools:context="com.example.hp.railwaycamscanner.MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
```

```
tools:layout_editor_absoluteX="0dp"
tools:layout_editor_absoluteY="46dp">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="210dp"
    android:gravity="center"
    android:orientation="vertical"
    tools:layout_editor_absoluteY="-1dp">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="193dp"
        android:layout_height="137dp"
        android:layout_margin="1dp"
        android:padding="2dp"
        android:scaleType="fitCenter"
        app:srcCompat="@mipmap/ic_launcher" />

    <TextView
        android:id="@+id/Title"
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:text="RAILWAY RESERVATION"
        android:textAlignment="center"
        android:textColor="@color/colorPrimary"
        android:textSize="12sp"
        android:textStyle="bold" />

</LinearLayout>

<LinearLayout
    android:id="@+id/ll-v"
    android:layout_width="match_parent"
    android:layout_height="256dp"
    android:layout_marginBottom="2dp"
    android:layout_marginTop="1dp"
    android:gravity="center"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <TextView
        android:id="@+id/result"
        android:layout_width="match_parent"
        android:layout_height="35dp"
        android:background="@drawable/rounded_corner"
        android:text="PNR NO: -"
        android:textAlignment="center"
        android:textColor="@android:color/black"
        android:textSize="14sp"
        tools:layout_editor_absoluteX="37dp"
        tools:layout_editor_absoluteY="343dp" />

    <Button
        android:id="@+id/scanner"
        android:layout_width="347dp"
        android:layout_height="61dp"
        android:layout_marginLeft="2dp"
        android:layout_marginTop="60dp"
        android:background="@drawable/rounded_button"
        android:cursorVisible="true"
        android:text="Scan QR Code"
        android:textAlignment="center"
        android:textColor="@android:color/background_light"
        android:textSize="18sp" />
</LinearLayout>
```

```
    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

Activity_splash_screen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

tools:context="com.example.rahulmukeshsingh.scannerforrailwayproject.SplashScreen">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:gravity="center"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageView
            android:id="@+id/imageView2"
            android:layout_width="218dp"
            android:layout_height="194dp"
            app:srcCompat="@mipmap/ic_launcher_round" />
    </LinearLayout>
</android.support.constraint.ConstraintLayout>
```

Android_manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.rahulmukeshsingh.scannerforrailwayproject">

    <uses-permission android:name="android.permission.CAMERA" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">

        </activity>
        <activity android:name=".ScanActivity" />
        <activity android:name=".SplashScreen">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
```

```
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Strings.xml

```
<resources>
    <string name="app_name">_RailScan</string>
</resources>
```

Rounded_corner.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" >
    <stroke
        android:width="1dp"
        android:color="#00788b" />

    <solid android:color="#dfdfdf" />

    <padding
        android:left="1dp"
        android:right="1dp"
        android:top="1dp" />

    <corners android:radius="15dp" />
</shape>
```

Rounded_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" >
    <stroke
        android:width="1dp" />

    <solid android:color="#00788b" />

    <padding
        android:left="1dp"
        android:right="1dp"
        android:top="1dp" />

    <corners android:radius="15dp" />
</shape>
```

Main_Activity.java

```
package com.example.rahulmukeshsingh.scannerforrailwayproject;

import android.Manifest;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Build;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    public static TextView result;
    private Button scann;
    private Context mainContext;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        getSupportActionBar().setLogo(R.mipmap.ic_launcher);
        getSupportActionBar().setDisplayUseLogoEnabled(true);
        getSupportActionBar().setBackgroundDrawable(new
ColorDrawable(Color.parseColor("#00788B")));
        Window window=getWindow();
        window.clearFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS);

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS);
        window.setStatusBarColor(Color.parseColor("#00788B"));
        setContentView(R.layout.activity_main);
        mainContext=MainActivity.this;

        result = (TextView) findViewById(R.id.result);

        scann = (Button) findViewById(R.id.scanner);

        scann.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                goForScan();

            }
        });

    }
    public void goForScan(){
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M &&
                checkSelfPermission(Manifest.permission.CAMERA) !=
                        PackageManager.PERMISSION_GRANTED) {

            requestPermissions(new String[]{Manifest.permission.CAMERA},
                    100);
        }else{
            Intent i = new Intent(mainContext, ScanActivity.class);
            startActivity(i);
        }
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions,
```

```
                                           @NonNull int[] grantResults) {
        if (requestCode == 100) {
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    goForScan();
            } else {
                Toast.makeText(this, "Until you grant the permission," +
                        " we cannot scan", Toast.LENGTH_SHORT).show();

            }
        }
    }

}
```

splashScreen.java

```
package com.example.rahulmukeshsingh.scannerforrailwayproject;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.view.Window;
import android.view.WindowManager;

public class SplashScreen extends AppCompatActivity {
Context splashContext;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        getSupportActionBar().setLogo(R.mipmap.ic_launcher);
        getSupportActionBar().setDisplayUseLogoEnabled(true);
        getSupportActionBar().setBackgroundDrawable(new
ColorDrawable(Color.parseColor("#00788B")));
        Window window=getWindow();
        window.clearFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS);

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS);
        window.setStatusBarColor(Color.parseColor("#00788B"));
        setContentView(R.layout.activity_splash_screen);
        splashContext=SplashScreen.this;

    }

    @Override
    protected void onResume() {
        super.onResume();
        Thread timer= new Thread(){

            @Override
            public void run() {
                try{
                    sleep(500);
                    Intent i = new Intent(splashContext, MainActivity.class);
                    startActivity(i);
```

```
                }catch (Exception e){
                    e.printStackTrace();
                }

            }
        };
        timer.start();
    }

}
```

DecryptandFormatString.java

```
package com.example.rahulmukeshsingh.scannerforrailwayproject;

/**
 * Created by Rahul Mukesh Singh on 3/30/2018.
 */

public class DecryptAndFormatString {
    public final String getDecryptedAndFormattedString(String plain,int shift){
        String plainText="";
        for (int i = 0; i < plain.length(); i++) {
            plainText+=(char)(plain.charAt(i)-shift);
        }
        return "PNR NO: "+plainText;
    }

}
```

scanActivity.java

```
package com.example.rahulmukeshsingh.scannerforrailwayproject;

import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Window;
import android.view.WindowManager;

import com.google.zxing.Result;

import me.dm7.barcodescanner.zxing.ZXingScannerView;

public class ScanActivity extends AppCompatActivity implements
ZXingScannerView.ResultHandler{
    private ZXingScannerView scanView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        getSupportActionBar().setLogo(R.mipmap.ic_launcher);
        getSupportActionBar().setDisplayUseLogoEnabled(true);
        getSupportActionBar().setBackgroundDrawable(new
ColorDrawable(Color.parseColor("#00788B")));
        Window window=getWindow();
        window.clearFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS);

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS);
        window.setStatusBarColor(Color.parseColor("#00788B"));
        scanView=new ZXingScannerView(this);
        setContentView(scanView);

    }

    @Override
    protected void onPause() {
        super.onPause();
       scanView.stopCamera();
    }



    @Override
    protected void onResume() {
        super.onResume();
        scanView.setResultHandler(this);
        scanView.startCamera();
    }

    @Override
    public void handleResult(Result result) {
        MainActivity.result.setText((new
DecryptAndFormatString()).getDecryptedAndFormattedString(result.getText(),5));
        onBackPressed();
    }
}
```

SYSTEM TESTING

6.1 TEST TECHNIQUES

Testing is the process of executing a program in order to find the software errors.

It can be said as a process of validating and verifying that a software meets all the requirements, with no errors.

There are various techniques used for testing :

1. Black-Box Testing
2. White-Box Testing
   Our Project has been tested with both Black-box testing and White Box testing methods.

   Black-Box Testing:

   The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

   White-Box Testing:

   White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open-box testing**. In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code.
   The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

6.2 TEST DATA

Dynamictesting.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import railwayFrequentFunctions.DatabasePrice;
import railwayFrequentFunctions.DateTimeofStation;
import railwayFrequentFunctions.SeatAllocation;

/**
 * Servlet implementation class dynamicTesting
```

```
 */
@WebServlet("/dynamicTesting")
public class dynamicTesting extends HttpServlet {
	private static final long serialVersionUID = 1L;


   /**
    * @see HttpServlet#HttpServlet()
    */
   public dynamicTesting() {
      super();
      // TODO Auto-generated constructor stub
   }
	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		doPost(request, response);
	}


	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```

```
// TODO Auto-generated method stub
PrintWriter out = response.getWriter();
out.print("<center><h2 style='color:#00788b'>20%</h2></center>");
HttpSession session = request.getSession();

response.setContentType("application/json");
out.print("Hello");
String tno = (String) session.getAttribute("trainno");
String treturn = (String) session.getAttribute("trainreturn");
String sdate = (String) session.getAttribute("date");
String routeid = (String) session.getAttribute("routeId");
String src = (String) session.getAttribute("source");
String dest = (String) session.getAttribute("destination");
String classid = (String) session.getAttribute("classid");
session.setAttribute("dateTimeInJSON", (new DateTimeofStation())
                    .getDateAndTimeOfStation(src, dest, tno, treturn, sdate,
                                        routeid));
DatabasePrice dp = new DatabasePrice();
SeatAllocation sa = new SeatAllocation(tno, treturn, classid, routeid,
                    src, dest,sdate);
String quotaid = "";
double totalPriceWithoutTax = 0.0;
//out.print(request.getParameter("ticketData"));
String[] datas = new String[165];
for (int i = 0; i < datas.length; i++) {
```

```
                datas[i]="PQR"+i+";145;Female;GN;LB";
        }
        JSONArray jarr = new JSONArray();

        int count=0;
        for (String data : datas) {
                JSONObject jobj = new JSONObject();
                String[] d =null;
                try{
                d = data.split(";");
                out.print(data);
                }catch(Exception e){
                        out.print("error");
                }
                quotaid=d[3];
                int cnfGn=0, rac=0, wl=0, tqwl=0, cnfQuot=0;
                for (int i = (count-1); i >=0; i--) {
                        String
quotaColumnValue=((JSONObject)jarr.get(i)).get("quota").toString();
                        String
statusColumnValue=((JSONObject)jarr.get(i)).get("status").toString();
                        if(statusColumnValue.contains("(CNF")                   &&
quotaColumnValue.equalsIgnoreCase(quotaid)){
                                cnfQuot=cnfQuot+1;
                        }
                        if(statusColumnValue.contains("(CNF")                   &&
statusColumnValue.contains("GN")){
```

```
                    cnfGn=cnfGn+1;
                }
                if(statusColumnValue.contains("(RAC")){
                    rac=rac+1;
                }
                if(statusColumnValue.contains("(WL")){
                    wl=wl+1;
                }
                if(statusColumnValue.contains("(TQWL")){
                    tqwl=tqwl+1;
                }
            }
            //System.out.println(quotaid+"--> "+cnfGn+" "+ rac+" "+ wl+" "+ tqwl+" "+ cnfQuot);
            String status = sa.getAvailableSeats(quotaid,cnfGn, rac, wl, tqwl, cnfQuot);
            //out.print("status : "+status+" "+count+"\n");

            if (!status.equalsIgnoreCase("No Ticket Available")) {
                double priceWithoutTax = dp.getPriceWithoutTax(src, dest,
                        routeid, tno, treturn, classid, quotaid);
                totalPriceWithoutTax += priceWithoutTax;
                jobj.put("name", d[0]);
                jobj.put("age", d[1]);
                jobj.put("gender", d[2]);
                jobj.put("quota", quotaid);
```

```
                    jobj.put("pref", d[4]);
                    jobj.put("status", status);
                    jobj.put("priceWithoutTax", priceWithoutTax);
                    jarr.add(jobj);
                    count++;
                }

            }

            session.setAttribute("dataInJSON", jarr);
            JSONArray taxNamePlusPercentPlusTaxedPrice = dp
                    .getPriceWithTax(totalPriceWithoutTax);
            session.setAttribute("taxNamePlusPercentPlusTaxedPriceInJSON",
                    taxNamePlusPercentPlusTaxedPrice);
            JSONObject totalPriceWithTax = (JSONObject) taxNamePlusPercentPlusTaxedPrice
                    .get(taxNamePlusPercentPlusTaxedPrice.size() - 1);
            double totalTaxedPrice = (double) totalPriceWithTax.get("pricewithtax");
            //out.print(jarr);
            //out.print(taxNamePlusPercentPlusTaxedPrice);
            //out.print((JSONArray) session.getAttribute("dateTimeInJSON"));
            RequestDispatcher rd=request.getRequestDispatcher("../railway/dynamicinsert");
            rd.forward(request, response);

        }
```

}

DynamicInsert.java

```
package servletss;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Date;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import railwayFrequentFunctions.DatabaseConnection;
```

```
import railwayFrequentFunctions.DatabasePrice;
import railwayFrequentFunctions.PNR;
import railwayFrequentFunctions.SeatAllocation;

/**
 * Servlet implementation class dynamicInsert
 */
@WebServlet("/dynamicInsert")
public class dynamicInsert extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public dynamicInsert() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		doPost(request, response);
```

```
	}


	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		try {
			PrintWriter out = response.getWriter();
			out.print("<center><h2 style='color:#00788b'>40%</h2></center>");
			DatabasePrice dp=new DatabasePrice();
			Connection con=(new DatabaseConnection()).dbConnect();
			HttpSession session = request.getSession();
			String tno = (String) session.getAttribute("trainno");
			String treturn = (String) session.getAttribute("trainreturn");
			String src = (String) session.getAttribute("source");
			String dest = (String) session.getAttribute("destination");
			String userid = (String) session.getAttribute("userid");
			String classid = (String) session.getAttribute("classid");
			String routeid = (String) session.getAttribute("routeId");
			String sdate = (String) session.getAttribute("date");
			String pnr = (new PNR()).getPNR();
			session.setAttribute("pnr", pnr);
```

```
String insertBookedCustomerQuery = "INSERT INTO booked_customer_details VALUES (?, ?, ?, ?,"
                + " ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
PreparedStatement ps;
ResultSet rs;
JSONArray dataInJSON = (JSONArray) session.getAttribute("dataInJSON");

JSONArray dateTimeInJSON = (JSONArray) session.getAttribute("dateTimeInJSON");
JSONObject srcOfUser = (JSONObject) dateTimeInJSON.get(1);
JSONObject destOfUser = (JSONObject) dateTimeInJSON.get(2);
JSONObject srcOfTrain = (JSONObject) dateTimeInJSON.get(0);
JSONObject destOfTrain = (JSONObject) dateTimeInJSON.get(3);

Date srcTime=(Date)srcOfUser.get("stationtime");
Date destTime=(Date)destOfUser.get("stationtime");
Date srcTrainTime=(Date)srcOfTrain.get("stationtime");
Date destTrainTime=(Date)destOfTrain.get("stationtime");
java.sql.Timestamp srcSqlTime=new java.sql.Timestamp(srcTime.getTime());
java.sql.Timestamp destSqlTime=new java.sql.Timestamp(destTime.getTime());
java.sql.Timestamp srcTrainSqlTime=new java.sql.Timestamp(srcTrainTime.getTime());
java.sql.Timestamp destTrainSqlTime=new java.sql.Timestamp(destTrainTime.getTime());
JSONObject dataINJSONObject;
```

```
int counter=0;
JSONArray newDataInJSON=new JSONArray();
for (Object dataInObject : dataInJSON) {
SeatAllocation sa=new SeatAllocation(tno, treturn, classid, routeid, src, dest,sdate);
dataINJSONObject=(JSONObject)dataInObject;
String quotaId=dataINJSONObject.get("quota").toString();
String totalStatus=dataINJSONObject.get("status").toString();
String pref=dataINJSONObject.get("pref").toString();
Double priceWithoutTax=(Double)dataINJSONObject.get("priceWithoutTax");
JSONArray taxNamePlusPercentPlusTaxedPrice = dp.getPriceWithTax(priceWithoutTax);
JSONObject priceWithTax = (JSONObject) taxNamePlusPercentPlusTaxedPrice

.get(taxNamePlusPercentPlusTaxedPrice.size() - 1);
double taxedPrice = (double) priceWithTax.get("pricewithtax");
String[] formattedStatus=sa.getFormattedStatus(totalStatus);

String statusId=formattedStatus[1];
if(formattedStatus[1].equalsIgnoreCase("TQWL")){
statusId="WL";
}
ps=con.prepareStatement(insertBookedCustomerQuery);
ps.setString(1, userid);
```

```
                    ps.setString(2, pnr);
                    ps.setString(3, dataINJSONObject.get("name").toString());

        ps.setInt(4,Integer.parseInt(dataINJSONObject.get("age").toString()));
                    ps.setString(5,quotaId);
                    ps.setString(6,classid);
                    ps.setInt(7, (counter+1));
                    ps.setString(8,statusId);
                    ps.setInt(9,Integer.parseInt(formattedStatus[2].trim()));
                    ps.setString(10,statusId);
                    ps.setInt(11,Integer.parseInt(formattedStatus[2].trim()));
                    ps.setString(12,
dataINJSONObject.get("gender").toString());
                    ps.setTimestamp(13, srcSqlTime);
                    ps.setTimestamp(14, destSqlTime);
                    ps.setDouble(15, taxedPrice);
                    ps.executeUpdate();
                    JSONObject
newDataINJSONObject=sa.insertIntoStatusTable(formattedStatus, pnr, quotaId,
pref,
                                        (counter+1), con,
ps,dataINJSONObject,srcTrainSqlTime,destTrainSqlTime);
                    newDataInJSON.add(newDataINJSONObject);
                    counter++;
                }
                con.close();
                session.setAttribute("dataInJSON",newDataInJSON);
```

```
		} catch (Exception e) {
			e.printStackTrace();
			// TODO: handle exception
		}
	}

}
```

DataBaseRandomInsert.java

```
package topsub;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class DataBaseRandomInsert {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		try {
			Class.forName("com.mysql.jdbc.Driver");
			Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/railway_database",
```

```
                    "railway_project", "railway");
//----------
/*      String insert_a1_1="insert into a1_1 values(?,?)";
        PreparedStatement ps=con.prepareStatement(insert_a1_1);
        for (int i = 1; i <= 22; i++) {
                if(i%2==1){
                ps.setInt(1, i);
                ps.setString(2, "LB");
                ps.executeUpdate();
                }else{
                        ps.setInt(1, i);
                        ps.setString(2, "UB");
                        ps.executeUpdate();
                }

        } */
//-----------
/*      String insert_a1_2="insert into a1_2 values(?,?)";
        PreparedStatement ps=con.prepareStatement(insert_a1_2);
        for (int i = 1; i <= 24; i++) {
                if(i%2==1){
                ps.setInt(1, i);
                ps.setString(2, "LB");
                ps.executeUpdate();
                }else{
                        ps.setInt(1, i);
```

```
                ps.setString(2, "UB");
                ps.executeUpdate();
            }

        } */
//-----------
/*  String insert_ec_1="insert into ec_1 values(?,?)";
        PreparedStatement ps=con.prepareStatement(insert_ec_1);
        for (int i = 1; i <= 45; i+=4) {
            ps.setInt(1, i);
            ps.setString(2, "WS");
            ps.executeUpdate();

            ps.setInt(1, (i+1));
            ps.setString(2, "AS");
            ps.executeUpdate();


        }

        for (int i = 3; i <= 43; i+=4) {
            ps.setInt(1, i);
            ps.setString(2, "AS");
            ps.executeUpdate();

            ps.setInt(1, (i+1));
```

```
                ps.setString(2, "WS");
                ps.executeUpdate();



        } */
//-----------
/*      String insert_ec_2="insert into ec_2 values(?,?)";
        PreparedStatement ps=con.prepareStatement(insert_ec_2);
        for (int i = 1; i <= 53; i+=4) {
                ps.setInt(1, i);
                ps.setString(2, "WS");
                ps.executeUpdate();

                ps.setInt(1, (i+1));
                ps.setString(2, "AS");
                ps.executeUpdate();



        }

        for (int i = 3; i <= 55; i+=4) {
                ps.setInt(1, i);
                ps.setString(2, "AS");
                ps.executeUpdate();

                ps.setInt(1, (i+1));
```

```
                ps.setString(2, "WS");
                ps.executeUpdate();



        } */

//-----------
/*      String insert_a2_1="insert into a2_1 values(?,?)";
        PreparedStatement ps=con.prepareStatement(insert_a2_1);
        for (int i = 1; i <= 37; i+=6) {
                ps.setInt(1, i);
                ps.setString(2, "LB");
                ps.executeUpdate();

                ps.setInt(1, (i+1));
                ps.setString(2, "UB");
                ps.executeUpdate();

                ps.setInt(1, (i+2));
                ps.setString(2, "LB");
                ps.executeUpdate();

                ps.setInt(1, (i+3));
                ps.setString(2, "UB");
                ps.executeUpdate();
```

```
            ps.setInt(1, (i+4));
            ps.setString(2, "SLB");
            ps.executeUpdate();

            ps.setInt(1, (i+5));
            ps.setString(2, "SUB");
            ps.executeUpdate();

}

ps.setInt(1, 43);
ps.setString(2, "LB");
ps.executeUpdate();

ps.setInt(1, 44);
ps.setString(2, "UB");
ps.executeUpdate();

ps.setInt(1, 45);
ps.setString(2, "SLB");
ps.executeUpdate();

ps.setInt(1, 46);
ps.setString(2, "SUB");
ps.executeUpdate(); */
```

```
//-----------
/*      String insert_a2_2="insert into a2_2 values(?,?)";
        PreparedStatement ps=con.prepareStatement(insert_a2_2);
        for (int i = 1; i <= 49; i+=6) {
                ps.setInt(1, i);
                ps.setString(2, "LB");
                ps.executeUpdate();

                ps.setInt(1, (i+1));
                ps.setString(2, "UB");
                ps.executeUpdate();

                ps.setInt(1, (i+2));
                ps.setString(2, "LB");
                ps.executeUpdate();

                ps.setInt(1, (i+3));
                ps.setString(2, "UB");
                ps.executeUpdate();

                ps.setInt(1, (i+4));
                ps.setString(2, "SLB");
                ps.executeUpdate();

                ps.setInt(1, (i+5));
                ps.setString(2, "SUB");
```

```
            ps.executeUpdate();

} */

//-----------
/* String insert_a3="insert into a3 values(?,?)";
PreparedStatement ps=con.prepareStatement(insert_a3);
for (int i = 1; i <= 57; i+=8) {
            ps.setInt(1, i);
            ps.setString(2, "LB");
            ps.executeUpdate();

            ps.setInt(1, (i+1));
            ps.setString(2, "MB");
            ps.executeUpdate();

            ps.setInt(1, (i+2));
            ps.setString(2, "UB");
            ps.executeUpdate();

            ps.setInt(1, (i+3));
            ps.setString(2, "LB");
            ps.executeUpdate();

            ps.setInt(1, (i+4));
            ps.setString(2, "MB");
```

```
                ps.executeUpdate();

                ps.setInt(1, (i+5));
                ps.setString(2, "UB");
                ps.executeUpdate();

                ps.setInt(1, (i+6));
                ps.setString(2, "SLB");
                ps.executeUpdate();

                ps.setInt(1, (i+7));
                ps.setString(2, "SUB");
                ps.executeUpdate();

        } */
        //-----------
/*      String insert_cc1="insert into cc1 values(?,?)";
        PreparedStatement ps=con.prepareStatement(insert_cc1);
        for (int i = 1; i <= 70;) {
                if(i==1 || i==70){
                        ps.setInt(1, i);
                        ps.setString(2, "WS");
                        ps.executeUpdate();

                        ps.setInt(1, (i+1));
                        ps.setString(2, "MS");
```

```
            ps.executeUpdate();



            ps.setInt(1, (i+2));
            ps.setString(2, "AS");
            ps.executeUpdate();

            ps.setInt(1, (i+3));
            ps.setString(2, "WS");
            ps.executeUpdate();
            i+=4;
        }else{
        ps.setInt(1, i);
        ps.setString(2, "WS");
        ps.executeUpdate();

        ps.setInt(1, (i+1));
        ps.setString(2, "MS");
        ps.executeUpdate();

        ps.setInt(1, (i+2));
        ps.setString(2, "AS");
        ps.executeUpdate();

        ps.setInt(1, (i+3));
        ps.setString(2, "AS");
```

```
                ps.executeUpdate();

                ps.setInt(1, (i+4));
                ps.setString(2, "WS");
                ps.executeUpdate();

                i+=5;
                }

        }       */


        //----------
/*      String insert_cc2="insert into cc2 values(?,?)";
        PreparedStatement ps=con.prepareStatement(insert_cc2);
        for (int i = 1; i <= 70;) {
                if(i==1 || i==70){
                        ps.setInt(1, i);
                        ps.setString(2, "WS");
                        ps.executeUpdate();

                        ps.setInt(1, (i+1));
                        ps.setString(2, "AS");
                        ps.executeUpdate();
```

```
            ps.setInt(1, (i+2));
            ps.setString(2, "MS");
            ps.executeUpdate();

            ps.setInt(1, (i+3));
            ps.setString(2, "WS");
            ps.executeUpdate();
            i+=4;
        }else{
        ps.setInt(1, i);
        ps.setString(2, "WS");
        ps.executeUpdate();

        ps.setInt(1, (i+1));
        ps.setString(2, "AS");
        ps.executeUpdate();

        ps.setInt(1, (i+2));
        ps.setString(2, "AS");
        ps.executeUpdate();

        ps.setInt(1, (i+3));
        ps.setString(2, "MS");
        ps.executeUpdate();

        ps.setInt(1, (i+4));
```

```
                ps.setString(2, "WS");
                ps.executeUpdate();

                i+=5;
                }

        } */


        //-----------

/*      String insert_ccs="insert into ccs values(?,?)";
        PreparedStatement ps=con.prepareStatement(insert_ccs);
        for (int i = 1; i <= 75;) {
                if(i==1 || i==75){
                        ps.setInt(1, i);
                        ps.setString(2, "WS");
                        ps.executeUpdate();

                        ps.setInt(1, (i+1));
                        ps.setString(2, "MS");
                        ps.executeUpdate();


                        ps.setInt(1, (i+2));
                        ps.setString(2, "AS");
```

```
            ps.executeUpdate();

            ps.setInt(1, (i+3));
            ps.setString(2, "WS");
            ps.executeUpdate();
            i+=4;
        }else{
        ps.setInt(1, i);
        ps.setString(2, "WS");
        ps.executeUpdate();

        ps.setInt(1, (i+1));
        ps.setString(2, "MS");
        ps.executeUpdate();

        ps.setInt(1, (i+2));
        ps.setString(2, "AS");
        ps.executeUpdate();

        ps.setInt(1, (i+3));
        ps.setString(2, "AS");
        ps.executeUpdate();

        ps.setInt(1, (i+4));
        ps.setString(2, "WS");
        ps.executeUpdate();
```

```
            i+=5;
            }

        } */


        //-----------
/*      String insert_sl="insert into sl values(?,?)";
        PreparedStatement ps=con.prepareStatement(insert_sl);
        for (int i = 1; i <= 65; i+=8) {
            ps.setInt(1, i);
            ps.setString(2, "LB");
            ps.executeUpdate();

            ps.setInt(1, (i+1));
            ps.setString(2, "MB");
            ps.executeUpdate();

            ps.setInt(1, (i+2));
            ps.setString(2, "UB");
            ps.executeUpdate();

            ps.setInt(1, (i+3));
            ps.setString(2, "LB");
            ps.executeUpdate();
```

```
                ps.setInt(1, (i+4));
                ps.setString(2, "MB");
                ps.executeUpdate();

                ps.setInt(1, (i+5));
                ps.setString(2, "UB");
                ps.executeUpdate();

                ps.setInt(1, (i+6));
                ps.setString(2, "SLB");
                ps.executeUpdate();

                ps.setInt(1, (i+7));
                ps.setString(2, "SUB");
                ps.executeUpdate();

        } */

        //-----------
/*      String insert_s2="insert into s2 values(?,?)";
        PreparedStatement ps=con.prepareStatement(insert_s2);
        for (int i = 1; i <= 103; i+=6) {
                ps.setInt(1, i);
                ps.setString(2, "WS");
                ps.executeUpdate();
```

```
            ps.setInt(1, (i+1));
            ps.setString(2, "MS");
            ps.executeUpdate();

            ps.setInt(1, (i+2));
            ps.setString(2, "AS");
            ps.executeUpdate();

            ps.setInt(1, (i+3));
            ps.setString(2, "AS");
            ps.executeUpdate();

            ps.setInt(1, (i+4));
            ps.setString(2, "MS");
            ps.executeUpdate();

            ps.setInt(1, (i+5));
            ps.setString(2, "WS");
            ps.executeUpdate();

} */

//-----------

String insert_s2s="insert into s2s values(?,?)";
```

```
PreparedStatement ps=con.prepareStatement(insert_s2s);
for (int i = 6; i <= 96; i+=6) {
        ps.setInt(1, i);
        ps.setString(2, "WS");
        ps.executeUpdate();

        ps.setInt(1, (i+1));
        ps.setString(2, "MS");
        ps.executeUpdate();

        ps.setInt(1, (i+2));
        ps.setString(2, "AS");
        ps.executeUpdate();

        ps.setInt(1, (i+3));
        ps.setString(2, "AS");
        ps.executeUpdate();

        ps.setInt(1, (i+4));
        ps.setString(2, "MS");
        ps.executeUpdate();

        ps.setInt(1, (i+5));
        ps.setString(2, "WS");
        ps.executeUpdate();
```

```
}
ps.setInt(1, 1);
ps.setString(2, "WS");
ps.executeUpdate();

ps.setInt(1, 2);
ps.setString(2, "MS");
ps.executeUpdate();

ps.setInt(1, 3);
ps.setString(2, "AS");
ps.executeUpdate();

ps.setInt(1, 4);
ps.setString(2, "MS");
ps.executeUpdate();

ps.setInt(1, 5);
ps.setString(2, "WS");
ps.executeUpdate();

ps.setInt(1, 102);
ps.setString(2, "WS");
ps.executeUpdate();

ps.setInt(1, 103);
```

```
            ps.setString(2, "MS");
            ps.executeUpdate();

            ps.setInt(1, 104);
            ps.setString(2, "AS");
            ps.executeUpdate();

            ps.setInt(1, 105);
            ps.setString(2, "MS");
            ps.executeUpdate();

            ps.setInt(1, 106);
            ps.setString(2, "WS");
            ps.executeUpdate();

            //-----------
            System.out.println("Done");
        } catch (Exception e) {
                    e.printStackTrace();  }}}
```

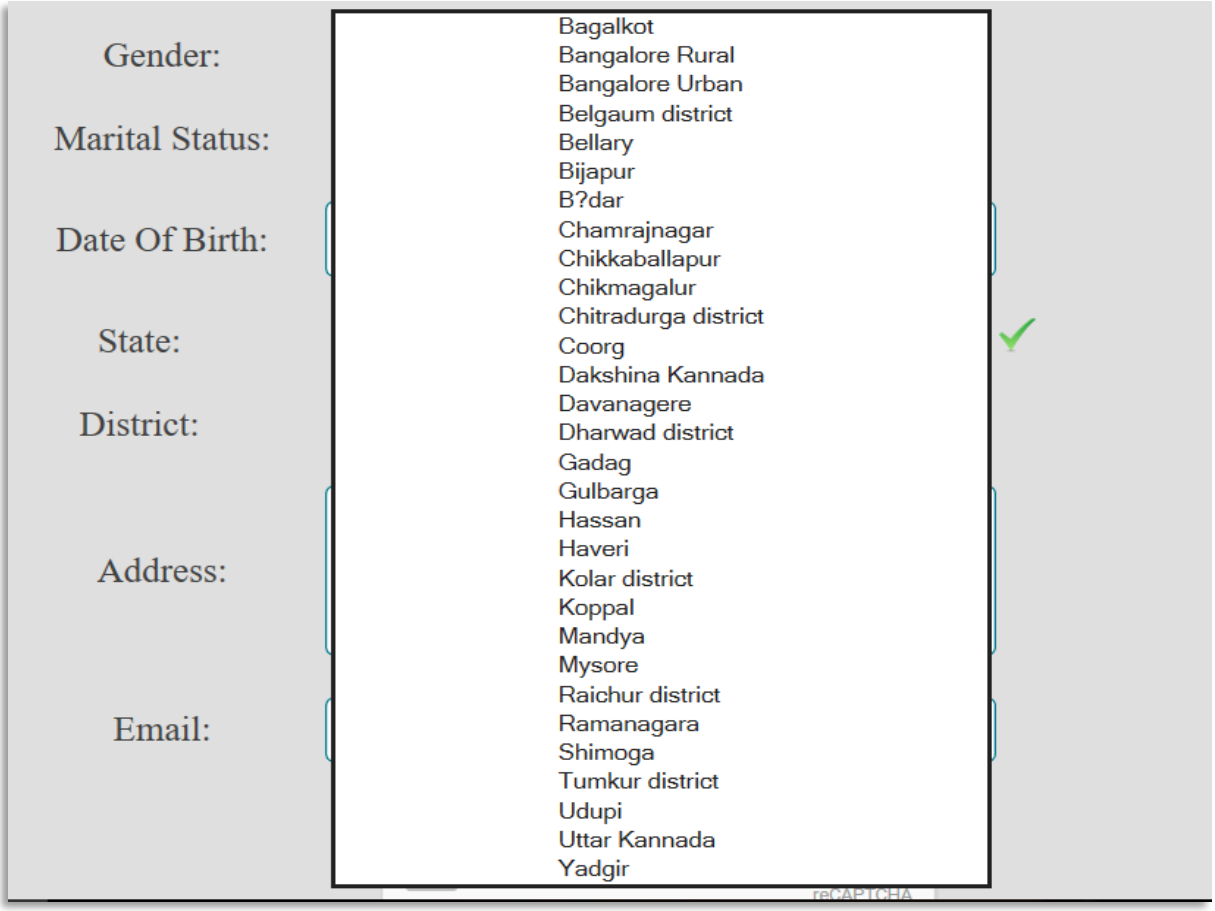
6.3 VALIDATION

Validation is determining if the system complies with the requirements and performs functions for which it is intended and meets the organization's goals and user needs.

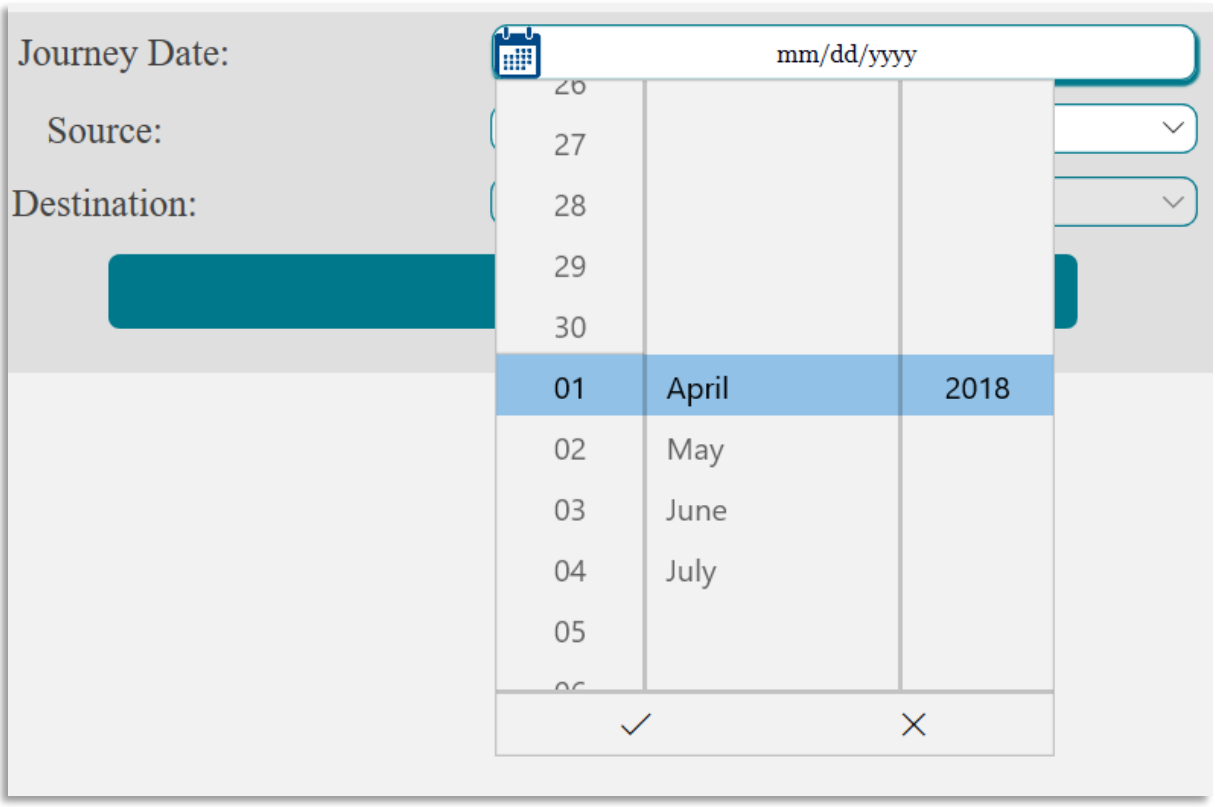
- Password validation check for User's login

- UI based Validations included in the project.Some of them are as follows:

1) In Signup page, only the districts of the selected State will be displayed.



2) Password should match while creating new password with retype password.
3) The date for the Journey in the Viewdetails page is enabled only from present date.



4) Tatkal option in quota is available only within 24 hours of the departure of the train from 10 a.m.
5) The Links in the navigation bar redirects to proper pages.
6) User is not allowed to access the train booking page until he/she is logged in.
7) Corresponding appropriate alert boxes for any failure.

FUTURE ENHANCEMENTS

This project could be further enhanced adding the running status of the train with real time update.

Maintaining the record of all the railway Locomotives driving the Train per day. This will give an analysis based on Payment of the employes according to the miles travelled.

It is a research-oriented project as well the task of performance evaluation of different database designs for efficiency, is in this spirit.

REFERENCES AND BIBLIOGRAPHY

The references which provided the required information regarding understanding the whole Railway Reservation System are as follows.

Online Websites such as

- www.irctc.com
- www.trainenquiry.com
- www.indianrail.com
- www.etrain.info

In addition to these above mentioned resources, our Parents, Professors, Guardians also provided us with all the necessary information.