

flyaway.com

Airline Booking Portal

Prototype of the Application

Name : ISHAN LAHOTI

GitHub : <https://github.com/inu1103/JavaPhase2.git>

The prototype of the application starts from the frontend, and it can also directly start from the project folder. This portal allows us to do flight management across administrator and provide flight booking facilities across client side which will at the end page goes to the payment portal(dummy). This prototype is built through various webpages (mainly .jsp file) which are interconnected with backend (servlets, database, models).

The implementation is done with the help of Hibernate, maven, Servlet, Java EE, Apache tomcat v8.5 and Eclipse.

Sprint Planning

The Implementation is done in two sprints which are mentioned below:

Sprint 1:

- Clarify the specification and requirements.
- Implement a framework of certain pages such as admin login, homepage.
- Implement a blueprint of Controller, Models part at backend and its core functionality.
- Identifying the various association for mapping of Passenger with Flight database along with its attributes containing primary key in both the tables.

Sprint 2:

- Building a platform for the prototype with hibernate, maven (webapp archetype) integration along with MySQL as a database which will run on local server (Tomcat v8.5) and required dependencies.
- Creating JSP Page as a starting point containing a hyperlink that will take us to the Admin Login page and contains a html table containing booking details such as source, destination, date of boarding and number of persons.
- Afterward, as this part is broadly categorized into two parts: admin and passenger section.
- Introducing a single controller (Servlet) as the data will be share not just within admin or passenger section but also with each other, packages consisting of models, hibernate configuration, data transfer objects for database connectivity.

Sprint 3:

- Implement functionality in controller for validation in admin login page consisting of email and password which is stated as email (admin@test.com) and password (admin).

- Implement functionality for changing password in admin section which consist of new password and confirmation password (same as new password for recheck).
- Implementing a web page for admin login page and after successful login will jump to the admin main page.
- Implementing another JSP page for changing password which is not connected currently.

Sprint 4:

- Developing the main page displaying flight details along with add, change password and logout button.
- Adding functionality for adding flights acting as hyperlink that will take us to next page asking for Flight Details such as flight number, airline, origin, destination, flight date.
- Once the admin submits the required details, it will store all the valid data into the database through servlet along with auto increment primary key not null values. And it will display in the main page along with a “edit” and “delete” button on status column.

Sprint 5:

- From the passenger side, once a user registered the details for their travel, a filter operation is performed in backend by retrieving the flight details from database and filter it according to source, destination, and boarding date.
- Along each flight details, there will be a hyperlink button name “Book Now”.
- From the “Book Now” button, it will jump to register page where user have to put his/her details according to the number of persons.

Sprint 6:

- In the registration page, there will be a option to select add passenger and a view table where user can view all the passenger details and modifications can also be done through status column option.
- Once the registration is done, it will show all the flight summary such as source, destination, date of boarding and total flight price.
- If a user registered passenger details more than number of persons, it will go back to the homepage again.

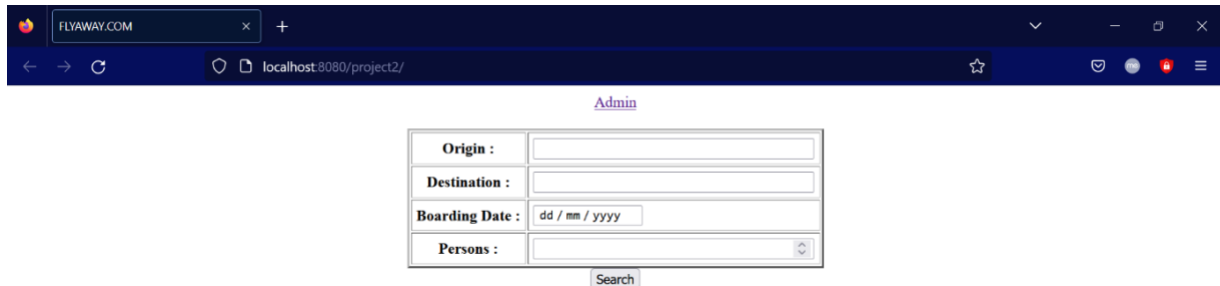
Sprint 7:

- From the passenger side, once a user registered the details for their travel, a filter operation is performed in backend by retrieving the flight details from database and filter it according to source, destination, and boarding date.
- Afterward, a dummy payment page will be displayed having an option to go back to home page as a hyperlink.
- Ensuring all the operations are tight and working well.
- Documentation.

Documentation of the functionality:

Here is the various static Java Servlet Pages that user will come across along the way.

1: Home Page

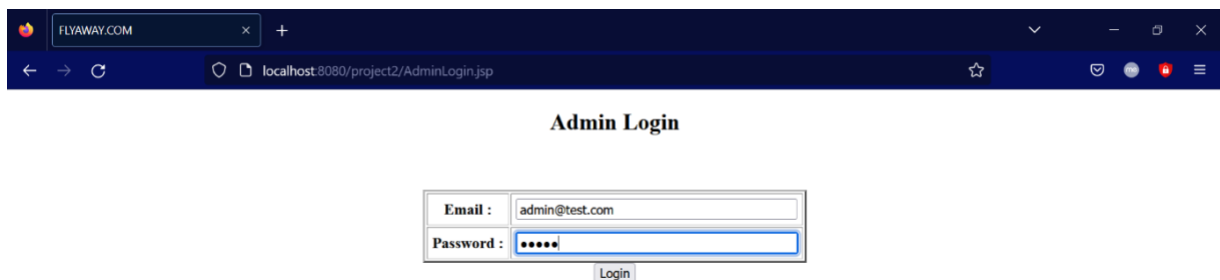


The screenshot shows a web browser window with the URL `localhost:8080/project2/`. The page has a title `Admin` and a form with the following fields:

Origin :	<input type="text"/>
Destination :	<input type="text"/>
Boarding Date :	<input type="text" value="dd / mm / yyyy"/>
Persons :	<input type="text"/>

Below the form is a `Search` button.

2: Admin Login Page (From admin side)



The screenshot shows a web browser window with the URL `localhost:8080/project2/AdminLogin.jsp`. The page has a title `Admin Login` and a form with the following fields:

Email :	<input type="text" value="admin@test.com"/>
Password :	<input type="password" value="*****"/>

Below the form is a `Login` button.

3: Admin main page (From admin side)



The screenshot shows a web browser window with the URL `localhost:8080/project2/login`. The page has a title `Flight Management` and a navigation bar with links: `Add Flight`, `Change Password`, and `Logout`. Below the navigation bar is a table with the following columns:

ID	Number	Air Name	Origin	Destination	Date	Booking Price	Status
----	--------	----------	--------	-------------	------	---------------	--------

4: Add Flight Page (From admin side)



The screenshot shows a web browser window with the URL `localhost:8080/project2/add`. The page has a form with the following fields:

Flight Number :	<input type="text" value="426"/>
Airline :	<input type="text" value="Indigo"/>
Origin :	<input type="text" value="Bengalore"/>
Destination :	<input type="text" value="Bhubaneswar"/>
Flight Date :	<input type="text" value="30 / 01 / 2023"/>
Ticket Price :	<input type="text" value="5500"/>

Below the form is a `Save` button.

*Note: Once the user save it after inserting details, it will be inserted into the database.



Flight Management

[Add Flight](#) [Change Password](#) [Logout](#)

Flight Details

ID	Number	Air Name	Origin	Destination	Date	Booking Price	Status
1	426	Indigo	Bengalore	Bhubaneswar	2023-01-30	5500.0	Edit Delete
2	673	Vistara	Hyderabad	Bangalore	2023-01-25	3500.0	Edit Delete
3	367	Air India	Bengalore	Chennai	2023-01-27	3000.0	Edit Delete

After, Admin can logout back to the homepage.

5: Insert the booking details from client side.



Admin

Origin :	<input type="text" value="Hyderabad"/>
Destination :	<input type="text" value="Bangalore"/>
Boarding Date :	<input type="text" value="25 / 01 / 2023"/>
Persons :	<input type="text" value="2"/>
<input type="button" value="Search"/>	

6: After inserting booking details, when user click the “search” button.



Selected Flight Details

ID	Number	Air Name	Origin	Destination	Date	Booking Price	Status
2	673	Vistara	Hyderabad	Bangalore	2023-01-25	3500.0	Book Now

7: After filtering flight details and user pressed the “Book Now” button. It will surf to Passenger Details.



Passenger Details

[Add Passenger](#)

Passenger Details

ID	First Name	Last Name	Contact	Age	Email
Confirm					

8: Add Passenger Page, Entering Passenger Details

First Name : Rahul

Last Name : Singh

Contact : 23456781

Age : 25

Email : test@test.com

Save

FLYAWAY.COM

localhost:8080/project2/booking

Passenger Details

[Add Passenger](#)

Passenger Details						
ID	First Name	Last Name	Contact	Age	Email	
1	Anurag	Sharma	12345678	23	test@gmail.com	Delete
2	Rahul	Singh	23456781	25	test@test.com	Delete

[Confirm](#)

9: Summary Page

FLYAWAY.COM

localhost:8080/project2/register

Summary Details

Flight Number :	673
Flight Name :	Vistara
Flight From :	Hyderabad
Flight To :	Bangalore
Flight Boarding Date :	2023-01-25
Ticket Price :	7000.0

Payment

10: Payment Page

FLYAWAY.COM

localhost:8080/project2/PaymentPage.jsp

Payment Method : [PayPal](#) [PayTm](#) [Debit/Credit Cards](#)

[Back To Home](#)

11: In MySQL Database, we have implemented @ManyToMany associations, here is the two tables flight and passengers table with its attributes and primary key.

```
mysql> use rkg;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> desc flight;
```

Field	Type	Null	Key	Default	Extra
flight_id	int	NO	PRI	NULL	auto_increment
flight_name	varchar(255)	YES		NULL	
Boarding_Date	varchar(255)	YES		NULL	
flight_number	int	YES		NULL	
Source	varchar(255)	YES		NULL	
Ticket_price	float	YES		NULL	
Destination	varchar(255)	YES		NULL	

```
7 rows in set (0.01 sec)

mysql>
```

```
mysql> desc passenger;
```

Field	Type	Null	Key	Default	Extra
passenger_id	int	NO	PRI	NULL	auto_increment
passenger_age	int	YES		NULL	
passenger_mob	bigint	YES		NULL	
passenger_email	varchar(255)	YES		NULL	
passenger_fname	varchar(255)	YES		NULL	
passenger_lname	varchar(255)	YES		NULL	

```
6 rows in set (0.00 sec)

mysql> select * from passenger;
```

passenger_id	passenger_age	passenger_mob	passenger_email	passenger_fname	passenger_lname
1	37	9876543210	albert@test.com	Albert	Einstein

```
1 row in set (0.00 sec)
```

And a third table named (flight_passenger) containing the mapping of primary key of both the table is done.

```
mysql> select * from flight_passenger;
```

flight_id	passenger_id
6	1

```
1 row in set (0.00 sec)

mysql> select * from flight;
```

flight_id	flight_name	Boarding_Date	flight_number	Source	Ticket_price	Destination
1	JamesBond Airways	2023-01-21	7	IND	60000	UK
2	Indigo	2023-01-16	6332	DEL	4500	BLR
3	Indigo	2023-01-20	4332	DEL	3500	LKO
4	Vistara	2023-01-25	9883	BLR	4500	HYD
5	Vistara	2023-01-26	7654	Varanasi	3600	Delhi
6	Indigo	2023-01-16	6332	DEL	4500	BLR

```
6 rows in set (0.00 sec)
```

Source Code:

1: JSP Pages

a> Home Page (HomePage.jsp)

```
MasterServlet.java PaymentPage.jsp AddPassenger.jsp HomePage.jsp
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head>
<meta charset="ISO-8859-1">
<title>FLYAWAY.COM</title>
</head>
<body>
<div class="container" align="center">
<a href="AdminLogin.jsp" Admin /> <br>
<br>
<form action="passenger" method="post">
<table border="2" cellpadding="5" position="bottom">
<tr>
<th>Origin :</th>
<td><input type="text" name="origin" size="45" required>
</td>
</tr>
<tr>
<th>Destination :</th>
<td><input type="text" name="target" size="45" required>
</td>
</tr>
<tr>
<th>Boarding Date :</th>
<td><input type="date" name="date" size="45" required>
</td>
</tr>
<tr>
<th>Persons :</th>
<td><input type="number" name="qty" size="45" required>
</td>
</tr>
</table>
<input type="submit" value="Search">
</form>
</div>
</body>
</html>
```

b> Admin Login Page (AdminLogin.jsp)

```
AdminLogin.jsp
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head>
<meta charset="ISO-8859-1">
<title>FLYAWAY.COM</title>
</head>
<body>
<div align="center">
<h2>Admin Login</h2>
<br><br><br><br>
<form action="Login" method="post">
<table border="2" cellpadding="5" position="bottom">
<tr>
<th>Email :</th>
<td><input type="email" name="email" size="45" required>
</td>
</tr>
<tr>
<th>Password :</th>
<td><input type="password" name="pwd" size="45" required>
</td>
</tr>
</table>
<input type="submit" value="Login">
</form>
</div>
</body>
</html>
```

c> Admin Main Page (FlightDetails.jsp)

```
FlightDetails.jsp
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head>
<meta charset="ISO-8859-1">
<title>FLYAWAY.COM</title>
</head>
<body>
<div align="center">
<a href="add" Add Flight</a>
<br><br><br><br>
<a href="ResetPage.jsp" Change Password</a>
<br><br><br><br>
<a href="HomePage.jsp" Logout</a>
</div>
<div align="center">
<table border="1">
<caption>
Flight Details
</caption>
<tr>
<th>ID</th>
<th>Number</th>
<th>Air Name</th>
<th>Origin</th>
<th>Destination</th>
<th>Date</th>
<th>Booking Price</th>
<th>Status</th>
</tr>
<tr>
<td colspan="8">
<table border="1">
<tr>
<td><input type="text" value="${f.flightId}" /></td>
<td><input type="text" value="${f.flightNumber}" /></td>
<td><input type="text" value="${f.airline}" /></td>
<td><input type="text" value="${f.origin}" /></td>
<td><input type="text" value="${f.target}" /></td>
<td><input type="text" value="${f.dob}" /></td>
<td><input type="text" value="${f.price}" /></td>
<td></td>
</tr>
<tr>
<td colspan="8">
<a href="edit?fid=${f.flightId}" /> Edit</a>
<br><br>
<a href="delete?fid=${f.flightId}" /> Delete</a>
</td>
</tr>
</table>
</td>
</tr>
</table>
</div>
</body>
</html>
```

d> Add Flight Details (AddFlight.jsp)

```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1" isEligible="false"%>
3
4 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
5
6 <!DOCTYPE html>
7 <html>
8 <head>
9   <meta charset="ISO-8859-1">
10  <title>FLYAMAY.COM</title>
11 </head>
12 <body>
13   <div align="center">
14     <!-- Update Form -->
15     <form action="update" method="post">
16     </form>
17     <!-- Insert Form -->
18     <form action="insert" method="post">
19     </form>
20     <table border="1" cellpadding="5">
21       <tr>
22         <td><input type="hidden" name="fid"
23           value="<c:out value='${f.flightId}' />" /></td>
24       </tr>
25       <tr>
26         <th>Flight Number :</th>
27         <td><input type="number" name="fnumber" size="45"
28           value="<c:out value='${f.flightNumber}' />" required /></td>
29       </tr>
30       <tr>
31         <th>Airline :</th>
32         <td><input type="text" name="fname" size="45"
33           value="<c:out value='${f.airline}' />" required /></td>
34       </tr>
35       <tr>
36         <th>Origin :</th>
37         <td><input type="text" name="forigin" size="45"
38           value="<c:out value='${f.origin}' />" required /></td>
39       </tr>
40       <tr>
41         <th>Destination :</th>
42         <td><input type="text" name="ftarget" size="45"
43           value="<c:out value='${f.target}' />" required /></td>
44       </tr>
45       <tr>
46         <th>Flight Date :</th>
47         <td><input type="date" name="fdate" size="45"
48           value="<c:out value='${f.dob}' />" required /></td>
49       </tr>
50       <tr>
51         <th>Ticket Price :</th>
52         <td><input type="number" name="fprice" size="45"
53           value="<c:out value='${f.price}' />" required /></td>
54       </tr>
55       <tr>
56         <td colspan="2" align="center"><input type="submit"
57           value="Save" /></td>
58       </tr>
59     </table>
60   </div>
61 </body>
62 </html>
```

```
27 <td><input type="number" name="fnumber" size="45"
28   value="<c:out value='${f.flightNumber}' />" required /></td>
29 </tr>
30 <tr>
31   <th>Airline :</th>
32   <td><input type="text" name="fname" size="45"
33     value="<c:out value='${f.airline}' />" required /></td>
34 </tr>
35 <tr>
36   <th>Origin :</th>
37   <td><input type="text" name="forigin" size="45"
38     value="<c:out value='${f.origin}' />" required /></td>
39 </tr>
40 <tr>
41   <th>Destination :</th>
42   <td><input type="text" name="ftarget" size="45"
43     value="<c:out value='${f.target}' />" required /></td>
44 </tr>
45 <tr>
46   <th>Flight Date :</th>
47   <td><input type="date" name="fdate" size="45"
48     value="<c:out value='${f.dob}' />" required /></td>
49 </tr>
50 <tr>
51   <th>Ticket Price :</th>
52   <td><input type="number" name="fprice" size="45"
53     value="<c:out value='${f.price}' />" required /></td>
54 </tr>
55 <tr>
56   <td colspan="2" align="center"><input type="submit"
57     value="Save" /></td>
58 </tr>
59 </table>
60 </div>
61 </body>
62 </html>
```

e> Change Admin Password (ResetPage.jsp)

```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="ISO-8859-1">
7   <title>FLYAMAY.COM</title>
8 </head>
9 <body>
10  <div align="center">
11    <h2>Reset Password</h2>
12    <br><br><br><br>
13    <form action="reset" method="post">
14      <table border="2" cellpadding="5" position="bottom">
15        <tr>
16          <th>Enter new Password :</th>
17          <td><input type="password" name="newpwd" size="45" required>
18          </td>
19        </tr>
20        <tr>
21          <th>Confirm Password :</th>
22          <td><input type="password" name="confpwd" size="45" required>
23          </td>
24        </tr>
25      </table>
26      <input type="submit" value="Save">
27    </form>
28  </div>
29 </body>
30 </html>
```


f> Passenger Flights Page (PassengerFlights.jsp)

[illegible]

g> Passenger Register Page (RegisterPage.jsp)

```

1 <!-- Register.jsp -->
2 %<!DOCTYPE html>%
3 <html>
4 <head>
5   <meta charset="ISO-8859-1">
6   <title>FLYANAY.COM</title>
7 </head>
8 <body>
9   <h1 align="center">Passenger Details</h1>
10   <h2 align="center">
11     <a href="AddPassenger.jsp">Add Passenger</a>   &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
12   </h2>
13   <div align="center">
14     <table border="1">
15       <caption>Passenger Details</caption>
16       <tr>
17         <th>ID</th>
18         <th>First Name</th>
19         <th>Last Name</th>
20         <th>Contact</th>
21         <th>Age</th>
22         <th>Email</th>
23       </tr>
24       <c:forEach var="p" items="${list}"%>
25         <tr>
26           <td><c:out value='${p.pId}' /></td>
27           <td><c:out value='${p.fname}' /></td>
28           <td><c:out value='${p.lname}' /></td>
29           <td><c:out value='${p.contact}' /></td>
30           <td><c:out value='${p.age}' /></td>
31           <td><c:out value='${p.email}' /></td>
32           <td><a href='deletePassenger?id=<c:out value='${p.pId}'' />'>Delete</a></td>
33         </tr>
34       </c:forEach>
35     </table>
36     <a href="register">Confirm</a>
37   </div>
38 </body>
39 </html>
```

h> Add Passenger (AddPassenger.jsp)

```

1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2  pageEncoding="ISO-8859-1" isEligible="false" %>
3
4  <%taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
5
6  <!DOCTYPE html>
7  <html>
8  <head>
9  <meta charset="ISO-8859-1">
10 <title>FLYAWAY.COM</title>
11 </head>
12 <body>
13     <div align="center">
14         <c:if test='${p != null}'>
15             <form action="updatePassenger" method="post">
16                 </c:if>
17         <c:if test='${p == null}'>
18             <form action="insertPassenger" method="post">
19                 </c:if>
20         <table border="1" cellpadding="5">
21             <c:if test='${f != null}'>
22                 <input type="hidden" name="id"
23                     value='${c:out value='${p.pId}'}' /> />
24             </c:if>
25             <tr>
26                 <th>First Name :</th>
27                 <td><input type="text" name="fname" size="45"
28                     value='${c:out value='${p.fname}'}' /> " required /></td>
29             </tr>
30             <tr>
31                 <th>Last Name :</th>
32                 <td><input type="text" name="lname" size="45"
33                     value='${c:out value='${p.lname}'}' /> " required /></td>
34             </tr>
35             <tr>
36                 <th>Contact :</th>
37                 <td><input type="text" name="contact" size="45"

```

```

1  AddPassenger.jsp x
2  <input type="hidden" name="id"
3      value=<%=out value=${p.pId} %>"/>"/>
4  </c:if>
5  <tr>
6      <th>First Name :</th>
7      <td><input type="text" name="fname" size="45"
8          value=<%=out value=${p.fname} %>'"/> required /></td>
9      </tr>
10 <tr>
11     <th>Last Name :</th>
12     <td><input type="text" name="lname" size="45"
13         value=<%=out value=${p.lname} %>'"/> required /></td>
14     </tr>
15 <tr>
16     <th>Contact :</th>
17     <td><input type="number" name="contact" size="45"
18         value=<%=out value=${p.contact} %>'"/> required /></td>
19     </tr>
20 <tr>
21     <th>Age :</th>
22     <td><input type="text" name="age" size="45"
23         value=<%=out value=${p.age} %>'"/> required /></td>
24     </tr>
25 <tr>
26     <th>Email :</th>
27     <td><input type="email" name="email" size="45"
28         value=<%=out value=${p.email} %>'"/> required /></td>
29     </tr>
30 <tr>
31     <td colspan="2" align="center"><input type="submit"
32         value="Save" /></td>
33     </tr>
34 </table>
35 </form>
36 </div>
37 </body>
38 </html>

```

i> Flight Summary Page (SummaryPage.jsp)

[illegible]

j> Dummy Payment Gateway (PaymentPage.jsp)

```

1 <% page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1" isEligible="false"%>
3
4 <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
5
6 <!DOCTYPE html>
7 <html>
8 <head>
9 <meta charset="ISO-8859-1">
10 <title>FLYAWAY.COM</title>
11 </head>
12 <body>
13 <div align="center">
14 <table border="2" cellpadding="5" position="bottom">
15 <tr>
16 <th>Payment Method :</th>
17 <td><a href="#">PayPal</a></td>
18 <td><a href="#">PayTm</a></td>
19 <td><a href="#">Debit/Credit Cards</a></td>
20 </tr>
21 </table>
22 <br><br>
23 <a href="HomePage.jsp">Back To Home</a>
24 </div>
25 </body>
26 </html>

```

a> Passenger.java

```

79* public void setName(String fname) {
80     this.fname = fname;
81 }
82
83* public String getName() {
84     return lname;
85 }
86
87* public void setName(String lname) {
88     this.lname = lname;
89 }
90
91* public int getAge() {
92     return age;
93 }
94
95* public void setAge(int age) {
96     this.age = age;
97 }
98
99* public long getContact() {
100     return contact;
101 }
102
103* public void setContact(long contact) {
104     this.contact = contact;
105 }
106
107* public String getEmail() {
108     return email;
109 }
110
111* public void setEmail(String email) {
112     this.email = email;
113 }
114

```

```

88     this.airline = airline;
89 }
90
91 public int getAge() {
92     return age;
93 }
94
95 public void setAge(int age) {
96     this.age = age;
97 }
98
99 public long getContact() {
100     return contact;
101 }
102
103 public void setContact(long contact) {
104     this.contact = contact;
105 }
106
107 public String getEmail() {
108     return email;
109 }
110
111 public void setEmail(String email) {
112     this.email = email;
113 }
114
115 public List<Flight> getFlight() {
116     return flight;
117 }
118
119 public void setFlight(List<Flight> flight) {
120     this.flight = flight;
121 }
122
123 }

```

b> Flight.java

```

1 package models;
2
3 import java.util.ArrayList;
4
5 @Entity
6 @Table(name = "Flight")
7 public class Flight {
8
9     @Id
10     @GeneratedValue(strategy = GenerationType.IDENTITY)
11     @Column(name = "flight_id")
12     private int flightId;
13
14     @Column(name = "flight_number")
15     private int flightNumber;
16
17     @Column(name = "flight_name")
18     private String airline;
19
20     @Column(name = "Source")
21     private String origin;
22
23     @Column(name = "Destination")
24     private String target;
25
26     @Column(name = "Boarding_Date")
27     private String dob;
28
29     @Column(name = "Ticket_price")
30     private float price;
31
32     public Flight() {
33     }
34
35     @ManyToMany(cascade = CascadeType.ALL)
36     @JoinTable(name = "flight_passenger",

```

```

37
38     @ManyToMany(cascade = CascadeType.ALL)
39     @JoinTable(name = "flight_passenger",
40         joinColumns = {
41             @JoinColumn(name = "flight_id")
42         },
43         inverseJoinColumns = {
44             @JoinColumn(name = "passenger_id")
45         }
46     )
47     List<Passenger> passenger = new ArrayList<Passenger>();
48
49     public Flight(int flightId, int flightNumber, String airline, String origin, String target, String dob, float price) {
50         super();
51         this.flightId = flightId;
52         this.flightNumber = flightNumber;
53         this.airline = airline;
54         this.origin = origin;
55         this.target = target;
56         this.dob = dob;
57         this.price = price;
58     }
59
60     public Flight(int flightNumber, String airline, String origin, String target, String dob, float price) {
61         super();
62         this.flightNumber = flightNumber;
63         this.airline = airline;
64         this.origin = origin;
65         this.target = target;
66         this.dob = dob;
67         this.price = price;
68     }
69
70     public int getFlightId() {
71         return flightId;
72     }

```

```
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

```
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

c> Password.java

```
1 package models;  
2  
3 public class Password {  
4  
5     private static String pwd;  
6  
7     public Password() {  
8         pwd = "admin";  
9     }  
10  
11     public static String getPwd() {  
12         return pwd;  
13     }  
14  
15     public static void setPwd(String pwd) {  
16         Password.pwd = pwd;  
17     }  
18  
19 }  
20
```

3: Data Access Objects

a> PassengerDAO.java

```
1 package transferobjectaccess;  
2  
3 import java.util.List;  
4  
5 public class PassengerDAO {  
6  
7     @SuppressWarnings("unchecked")  
8     public List<Flight> getAllDetailsByOriginDate(String origin, String date, String target) {  
9  
10         Transaction transaction = null;  
11         Session dbSession = null;  
12         List<Flight> list = null;  
13  
14         try {  
15             dbSession = HibernateConfig.getSessionFactory().openSession();  
16             transaction = dbSession.beginTransaction();  
17             @SuppressWarnings("rawtypes")  
18             Query query = dbSession  
19                 .createQuery("from Flight f where f.origin = :origin and f.target = :target and f.dob = :date");  
20             query.setParameter("origin", origin);  
21             query.setParameter("target", target);  
22             query.setParameter("date", date);  
23             list = query.list();  
24         } catch (Exception e) {  
25             if (transaction != null) {  
26                 transaction.rollback();  
27             }  
28             e.printStackTrace();  
29         } finally {  
30             dbSession.close();  
31         }  
32  
33         return list;  
34     }  
35  
36     public void insertPassengerInDB(Passenger p) {  
37  
38         Transaction transaction = null;  
39  
40     }  
41  
42 }  
43
```



```

42
43* public void insertPassengerInDB(Passenger p) {
44
45     Transaction transaction = null;
46     Session dbSession = null;
47     try {
48         dbSession = HibernateConfig.getSessionFactory().openSession();
49         transaction = dbSession.beginTransaction();
50         dbSession.save(p);
51         transaction.commit();
52     } catch (Exception e) {
53         if (transaction != null) {
54             transaction.rollback();
55         }
56         e.printStackTrace();
57     } finally {
58         dbSession.close();
59     }
60 }
61
62
63* @SuppressWarnings("unchecked")
64 public List<Passenger> getAllDetails() {
65
66     Session dbSession = null;
67     List<Passenger> list = null;
68     try {
69         dbSession = HibernateConfig.getSessionFactory().openSession();
70         list = dbSession.createQuery("from Passenger").list();
71     } catch (Exception e) {
72         e.printStackTrace();
73     } finally {
74         dbSession.close();
75     }
76     return list;
77 }
78

```

```

78
79 }
80
81* public Passenger getPassengerById(int id) {
82
83     Transaction transaction = null;
84     Session dbSession = null;
85     Passenger p = null;
86     try {
87         dbSession = HibernateConfig.getSessionFactory().openSession();
88         transaction = dbSession.beginTransaction();
89         p = dbSession.get(Passenger.class, id);
90     } catch (Exception e) {
91         if (transaction != null) {
92             transaction.rollback();
93         }
94         e.printStackTrace();
95     } finally {
96         dbSession.close();
97     }
98     return p;
99 }
100
101
102* public Passenger deletePassenger(Passenger p) {
103
104     Transaction transaction = null;
105     Session dbSession = null;
106     try {
107         dbSession = HibernateConfig.getSessionFactory().openSession();
108         transaction = dbSession.beginTransaction();
109         dbSession.delete(p);
110         transaction.commit();
111     } catch (Exception e) {
112         if (transaction != null) {
113             transaction.rollback();
114         }
115         e.printStackTrace();
116     } finally {
117         dbSession.close();
118     }
119     return p;
120 }
121
122
123 }

```

```

88     transaction = dbSession.beginTransaction();
89     p = dbSession.get(Passenger.class, id);
90 } catch (Exception e) {
91     if (transaction != null) {
92         transaction.rollback();
93     }
94     e.printStackTrace();
95 } finally {
96     dbSession.close();
97 }
98     return p;
99 }
100
101
102* public Passenger deletePassenger(Passenger p) {
103
104     Transaction transaction = null;
105     Session dbSession = null;
106     try {
107         dbSession = HibernateConfig.getSessionFactory().openSession();
108         transaction = dbSession.beginTransaction();
109         dbSession.delete(p);
110         transaction.commit();
111     } catch (Exception e) {
112         if (transaction != null) {
113             transaction.rollback();
114         }
115         e.printStackTrace();
116     } finally {
117         dbSession.close();
118     }
119     return p;
120 }
121
122
123 }

```

b> FlightDAO.java

```
1 package transferobjectaccess;
2
3 import java.util.List;
4
5 public class FlightDAO {
6
7     public void insertFlightInDB(Flight f) {
8
9         Transaction transaction = null;
10        Session dbSession = null;
11        try {
12            dbSession = HibernateConfig.getSessionFactory().openSession();
13            transaction = dbSession.beginTransaction();
14            dbSession.save(f);
15            transaction.commit();
16        } catch (Exception e) {
17            if (transaction != null) {
18                transaction.rollback();
19            }
20            e.printStackTrace();
21        } finally {
22            dbSession.close();
23        }
24    }
25
26    @SuppressWarnings("unchecked")
27    public List<Flight> getAllDetails() {
28
29        Session dbSession = null;
30        List<Flight> list = null;
31        try {
32            dbSession = HibernateConfig.getSessionFactory().openSession();
33            list = dbSession.createQuery("from Flight").list();
34        } catch (Exception e) {
35            e.printStackTrace();
36        }
37    }
38}
```

```
39        list = dbSession.createQuery("from Flight").list();
40    } catch (Exception e) {
41        e.printStackTrace();
42    } finally {
43        dbSession.close();
44    }
45    return list;
46}
47
48 public Flight getFlightById(int flightId) {
49
50    Transaction transaction = null;
51    Session dbSession = null;
52    Flight f = null;
53    try {
54        dbSession = HibernateConfig.getSessionFactory().openSession();
55        transaction = dbSession.beginTransaction();
56        f = dbSession.get(Flight.class, flightId);
57    } catch (Exception e) {
58        if (transaction != null) {
59            transaction.rollback();
60        }
61        e.printStackTrace();
62    } finally {
63        dbSession.close();
64    }
65    return f;
66}
67
68 public Flight updateFlight(Flight f) {
69
70    Transaction transaction = null;
71    Session dbSession = null;
72    try {
73        dbSession = HibernateConfig.getSessionFactory().openSession();
74        transaction = dbSession.beginTransaction();
75        dbSession.update(f);
76        transaction.commit();
77    } catch (Exception e) {
78        if (transaction != null) {
79            transaction.rollback();
80        }
81        e.printStackTrace();
82    } finally {
83        dbSession.close();
84    }
85    return f;
86}
87}
```

```
88        e.printStackTrace();
89    } finally {
90        dbSession.close();
91    }
92    return f;
93}
94
95 public Flight updateFlight(Flight f) {
96
97    Transaction transaction = null;
98    Session dbSession = null;
99    try {
100        dbSession = HibernateConfig.getSessionFactory().openSession();
101        transaction = dbSession.beginTransaction();
102        dbSession.update(f);
103        transaction.commit();
104    } catch (Exception e) {
105        if (transaction != null) {
106            transaction.rollback();
107        }
108        e.printStackTrace();
109    } finally {
110        dbSession.close();
111    }
112    return f;
113}
114
115 public Flight deleteFlight(Flight f) {
116
117    Transaction transaction = null;
118    Session dbSession = null;
119    try {
120        dbSession = HibernateConfig.getSessionFactory().openSession();
121        transaction = dbSession.beginTransaction();
122        dbSession.delete(f);
123        transaction.commit();
124    } catch (Exception e) {
125        if (transaction != null) {
126            transaction.rollback();
127        }
128        e.printStackTrace();
129    } finally {
130        dbSession.close();
131    }
132    return f;
133}
134}
```

```

198 Session dbSession = null;
199 try {
200     dbSession = HibernateConfig.getSessionFactory().openSession();
201     transaction = dbSession.beginTransaction();
202     dbSession.delete(f);
203     transaction.commit();
204 } catch (Exception e) {
205     if (transaction != null) {
206         transaction.rollback();
207     }
208     e.printStackTrace();
209 } finally {
210     dbSession.close();
211 }
212 return f;
213 }
214 }
215 }
216 public void relation(Flight f, List<Passenger> list) {
217     Transaction transaction = null;
218     Session dbSession = null;
219     try {
220         dbSession = HibernateConfig.getSessionFactory().openSession();
221         transaction = dbSession.beginTransaction();
222         f.setPassenger(list);
223         dbSession.save(f);
224         transaction.commit();
225     } catch (Exception e) {
226         if (transaction != null) {
227             transaction.rollback();
228         }
229         e.printStackTrace();
230     } finally {
231         dbSession.close();
232     }
233 }

```

4: Hibernate Integration (HibernateConfig.java)

```

1 package Configuration;
2
3 import java.util.Properties;
4
5 public class HibernateConfig {
6     private static SessionFactory sessionFactory;
7
8     public static SessionFactory getSessionFactory() {
9         if (sessionFactory == null) {
10             try {
11                 Configuration cfg = new Configuration();
12
13                 Properties settings = new Properties();
14                 settings.put(Environment.DRIVER, "com.mysql.cj.jdbc.Driver");
15                 settings.put(Environment.URL, "jdbc:mysql://localhost:3306/demo?useSSL=false");
16                 settings.put(Environment.USER, "root");
17                 settings.put(Environment.PASS, "cisco");
18                 settings.put(Environment.DIALECT, "org.hibernate.dialect.MySQL5Dialect");
19                 settings.put(Environment.SHOW_SQL, "true");
20                 settings.put(Environment.CURRENT_SESSION_CONTEXT_CLASS, "thread");
21                 // settings.put(Environment.HBM2DDL_AUTO, "update");
22                 settings.put(Environment.HBM2DDL_AUTO, "create");
23
24                 cfg.setProperties(settings);
25                 cfg.addAnnotatedClass(Flight.class);
26                 cfg.addAnnotatedClass(Passenger.class);
27                 ServiceRegistry servReg = new StandardServiceRegistryBuilder()
28                     .applySettings(cfg.getProperties()).build();
29                 sessionFactory = cfg.buildSessionFactory(servReg);
30             } catch (Exception e) {
31                 e.printStackTrace();
32             }
33         }
34         return sessionFactory;
35     }
36 }

```

5: Controller (MasterServlet.java)

```

1 package Controller;
2
3 import java.io.IOException;
4
5 @WebServlet("/")
6 public class MasterServlet extends HttpServlet {
7
8     private FlightDAO ob;
9     private Flight f;
10    private PassengerDAO ob1;
11    private Passenger p;
12    private int num;
13    private String pd = null;
14    private int count, flag;
15    private List<Passenger> list1;
16
17    private static final long serialVersionUID = 1L;
18
19    public void init() {
20        ob = new FlightDAO();
21        ob1 = new PassengerDAO();
22        pd = "admin";
23        count = 0;
24        flag = 0;
25        list1 = null;
26    }
27
28    public MasterServlet() {
29        count = 0;
30        list1 = null;
31    }
32
33    @Override
34    public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

```



```

MasterServlet.java
49
50 @Override
51 public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
52
53     String action = request.getServletPath();
54     try {
55         switch (action) {
56
57             case "/add":
58                 showNewForm(request, response);
59                 break;
60             case "/delete":
61                 deleteDetails(request, response);
62                 break;
63             case "/edit":
64                 editDetails(request, response);
65                 break;
66             case "/insert":
67                 insertDetails(request, response);
68                 break;
69             case "/update":
70                 updateDetails(request, response);
71                 break;
72             case "/reset":
73                 changePassword(request, response);
74                 break;
75             case "/register":
76                 register(request, response);
77                 break;
78             case "/storage":
79                 storage(request, response);
80                 break;
81             case "/find":
82                 getFlightDetailsById(request, response);
83                 break;
84             case "/login":
85                 login(request, response);
86                 break;
87             case "/passenger":
88                 passengerFlightDetails(request, response);
89                 break;
90             case "/insertPassenger":
91                 count++;
92                 passengerInsertDetails(request, response);
93                 break;
94             case "/deletePassenger":
95                 count--;
96                 passengerDeleteDetails(request, response);
97                 break;
98             case "/booking":
99                 bookingDetails(request, response);
100                 break;
101             default:
102                 showAllDetails(response, request);
103                 break;
104         }
105     } catch (Exception e) {
106         e.printStackTrace();
107     }
108 }

```

```

MasterServlet.java
73     changePassword(request, response);
74     break;
75     case "/register":
76         register(request, response);
77         break;
78     case "/storage":
79         storage(request, response);
80         break;
81     case "/find":
82         getFlightDetailsById(request, response);
83         break;
84     case "/login":
85         login(request, response);
86         break;
87     case "/passenger":
88         passengerFlightDetails(request, response);
89         break;
90     case "/insertPassenger":
91         count++;
92         passengerInsertDetails(request, response);
93         break;
94     case "/deletePassenger":
95         count--;
96         passengerDeleteDetails(request, response);
97         break;
98     case "/booking":
99         bookingDetails(request, response);
100         break;
101     default:
102         showAllDetails(response, request);
103         break;
104 }
105 } catch (Exception e) {
106     e.printStackTrace();
107 }
108 }

```

```

MasterServlet.java
106     } catch (Exception e) {
107         e.printStackTrace();
108     }
109 }
110 }
111
112 private void bookingDetails(HttpServletRequest request, HttpServletResponse response)
113     throws ServletException, IOException {
114
115     List<Passenger> list = obl.getAllDetails();
116     list1 = list;
117     request.setAttribute("list", list);
118     RequestDispatcher rd = request.getRequestDispatcher("RegisterPage.jsp");
119     rd.forward(request, response);
120 }
121
122
123 private void passengerInsertDetails(HttpServletRequest request, HttpServletResponse response) throws IOException {
124
125     if (count > num)
126         response.sendRedirect("HomePage.jsp");
127     else {
128         String fname = request.getParameter("fname");
129         String flname = request.getParameter("lname");
130         long contact = Long.parseLong(request.getParameter("contact"));
131         int age = Integer.parseInt(request.getParameter("age"));
132         String email = request.getParameter("email");
133         Passenger p = new Passenger(fname, flname, age, contact, email);
134         obl.insertPassengerInDB(p);
135         response.sendRedirect("booking");
136     }
137 }
138
139
140 private void passengerDeleteDetails(HttpServletRequest request, HttpServletResponse response) throws IOException {
141
142     int id = Integer.parseInt(request.getParameter("id"));

```

```

142     int id = Integer.parseInt(request.getParameter("id"));
143     Passenger p = obl.getPassengerById(id);
144     obl.deletePassenger(p);
145     response.sendRedirect("booking");
146 }
147
148 private void EditDetails(HttpServletRequest request, HttpServletResponse response)
149     throws SQLException, ServletException, IOException {
150
151     int id = Integer.parseInt(request.getParameter("fid"));
152     Flight f = ob.getFlightById(id);
153     RequestDispatcher dispatcher = request.getRequestDispatcher("AddFlight.jsp");
154     request.setAttribute("f", f);
155     dispatcher.forward(request, response);
156 }
157
158 private void updateDetails(HttpServletRequest request, HttpServletResponse response)
159     throws SQLException, IOException {
160
161     int fid = Integer.parseInt(request.getParameter("fid"));
162     int fnumber = Integer.parseInt(request.getParameter("fnumber"));
163     String fname = request.getParameter("fname");
164     String forigin = request.getParameter("forigin");
165     String ftarget = request.getParameter("ftarget");
166     String fdate = request.getParameter("fdate");
167     float fprice = Float.parseFloat(request.getParameter("fprice"));
168     Flight fl = new Flight(fid, fnumber, fname, forigin, ftarget, fdate, fprice);
169     ob.updateFlight(fl);
170     response.sendRedirect("view");
171 }
172
173 private void deleteDetails(HttpServletRequest req, HttpServletResponse resp) throws IOException {
174
175     int id = Integer.parseInt(req.getParameter("fid"));

```

```

172     response.sendRedirect("view");
173 }
174
175 private void deleteDetails(HttpServletRequest req, HttpServletResponse resp) throws IOException {
176
177     int id = Integer.parseInt(req.getParameter("fid"));
178     Flight f = ob.getFlightById(id);
179     ob.deleteFlight(f);
180     resp.sendRedirect("view");
181 }
182
183 private void showAllDetails(HttpServletResponse response, HttpServletRequest request)
184     throws ServletException, IOException {
185
186     List<Flight> list = ob.getAllDetails();
187     System.out.println(list);
188     request.setAttribute("list", list);
189     RequestDispatcher rd = request.getRequestDispatcher("FlightDetails.jsp");
190     rd.forward(request, response);
191 }
192
193 private void showNewForm(HttpServletRequest request, HttpServletResponse response)
194     throws ServletException, IOException {
195
196     RequestDispatcher rd = request.getRequestDispatcher("AddFlight.jsp");
197     rd.forward(request, response);
198 }
199
200 private void insertDetails(HttpServletRequest request, HttpServletResponse response) throws IOException {
201
202     int fnumber = Integer.parseInt(request.getParameter("fnumber"));
203     String fname = request.getParameter("fname");
204     String forigin = request.getParameter("forigin");
205     String ftarget = request.getParameter("ftarget");
206     String fdate = request.getParameter("fdate");
207     float fprice = Float.parseFloat(request.getParameter("fprice"));
208     Flight fl = new Flight(fnumber, fname, forigin, ftarget, fdate, fprice);
209     ob.insertFlightInDB(fl);
210     response.sendRedirect("view");
211 }
212
213 private void passengerFlightDetails(HttpServletRequest request, HttpServletResponse response)
214     throws ServletException, IOException {
215
216     String origin = request.getParameter("origin");
217     String target = request.getParameter("target");
218     String date = request.getParameter("date");
219     num = Integer.parseInt(request.getParameter("qty"));
220
221     List<Flight> list = obl.getAllDetailsByOriginDate(origin, date, target);
222     // System.out.println(list);
223     // request.setAttribute("num", num);
224     request.setAttribute("Selectedlist", list);
225     RequestDispatcher rd = request.getRequestDispatcher("PassengerFlights.jsp");
226     rd.forward(request, response);
227 }
228
229 private void getflightDetailsById(HttpServletRequest request, HttpServletResponse response)
230     throws ServletException, IOException {
231
232     int id = Integer.parseInt(request.getParameter("fid"));

```

```

202 }
203
204 private void insertDetails(HttpServletRequest request, HttpServletResponse response) throws IOException {
205
206     int fnumber = Integer.parseInt(request.getParameter("fnumber"));
207     String fname = request.getParameter("fname");
208     String forigin = request.getParameter("forigin");
209     String ftarget = request.getParameter("ftarget");
210     String fdate = request.getParameter("fdate");
211     float fprice = Float.parseFloat(request.getParameter("fprice"));
212     Flight fl = new Flight(fnumber, fname, forigin, ftarget, fdate, fprice);
213     ob.insertFlightInDB(fl);
214     response.sendRedirect("view");
215 }
216
217 private void passengerFlightDetails(HttpServletRequest request, HttpServletResponse response)
218     throws ServletException, IOException {
219
220     String origin = request.getParameter("origin");
221     String target = request.getParameter("target");
222     String date = request.getParameter("date");
223     num = Integer.parseInt(request.getParameter("qty"));
224
225     List<Flight> list = obl.getAllDetailsByOriginDate(origin, date, target);
226     // System.out.println(list);
227     // request.setAttribute("num", num);
228     request.setAttribute("Selectedlist", list);
229     RequestDispatcher rd = request.getRequestDispatcher("PassengerFlights.jsp");
230     rd.forward(request, response);
231 }
232
233 private void getflightDetailsById(HttpServletRequest request, HttpServletResponse response)
234     throws ServletException, IOException {
235
236     int id = Integer.parseInt(request.getParameter("fid"));

```

```

232
233
234
235
236 private void getFlightDetailsById(HttpServletRequest request, HttpServletResponse response)
237     throws ServletException, IOException {
238
239     int id = Integer.parseInt(request.getParameter("fid"));
240     FlightDAO x = new FlightDAO();
241     f = x.getFlightById(id);
242     // request.setAttribute("num", num);
243     RequestDispatcher rd = request.getRequestDispatcher("booking");
244     rd.forward(request, response);
245 }
246
247 private void register(HttpServletRequest request, HttpServletResponse response)
248     throws ServletException, IOException {
249
250     if (count != 0) {
251         ob.relation(f, list1);
252         request.setAttribute("n", num);
253         request.setAttribute("f", f);
254         // request.setAttribute("p", p);
255         RequestDispatcher rd = request.getRequestDispatcher("SummaryPage.jsp");
256         rd.forward(request, response);
257     } else
258         response.sendRedirect("HomePage.jsp");
259 }
260
261
262 private void storage(HttpServletRequest request, HttpServletResponse response) throws IOException {
263
264     ob1.insertPassengerInDB(p);
265     response.sendRedirect("HomePage.jsp");
266 }
267
268

```

```

259
260
261
262 private void storage(HttpServletRequest request, HttpServletResponse response) throws IOException {
263
264     ob1.insertPassengerInDB(p);
265     response.sendRedirect("HomePage.jsp");
266 }
267
268
269 private void changePassword(HttpServletRequest request, HttpServletResponse response)
270     throws IOException, ServletException {
271
272     String newpwd = request.getParameter("newpwd");
273     String confpwd = request.getParameter("confpwd");
274
275     if (newpwd.compareTo(confpwd) == 0) {
276         pd = newpwd;
277         response.sendRedirect("AdminLogin.jsp");
278     } else {
279         RequestDispatcher rd = request.getRequestDispatcher("ResetPage.jsp");
280         PrintWriter out = response.getWriter();
281         rd.include(request, response);
282         out.println("<center> <span style='color:red'> Invalid Credentials!!! </span></center>");
283     }
284 }
285
286
287 private void login(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
288
289     String email = request.getParameter("email");
290     String pwd = request.getParameter("pwd");
291
292     RequestDispatcher rd = null;
293
294     if (email.equalsIgnoreCase("admin@test.com") && pwd.equals(pd)) {
295         rd = request.getRequestDispatcher("view");

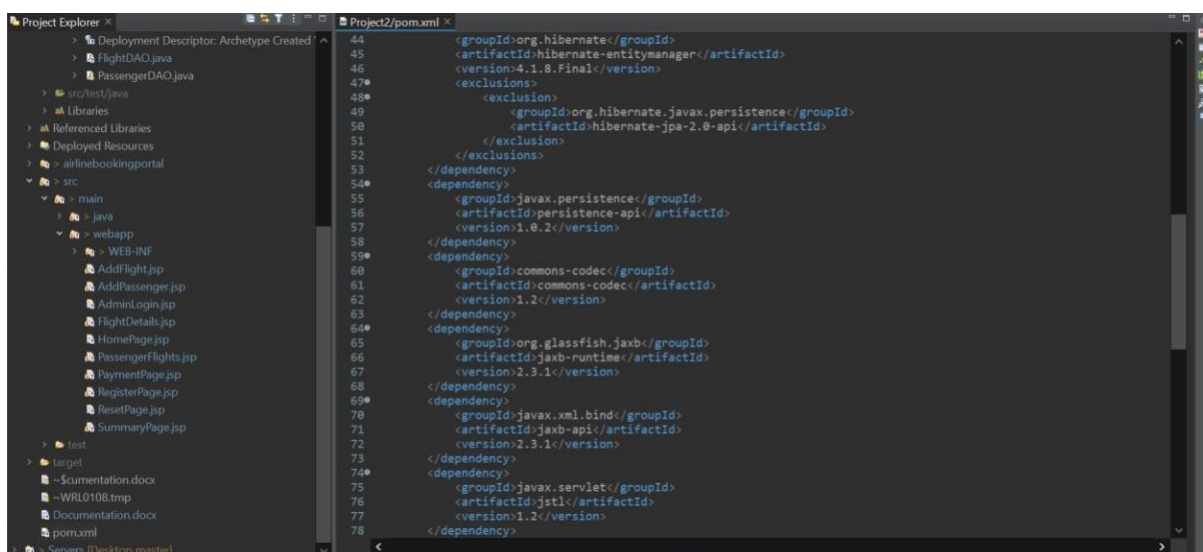
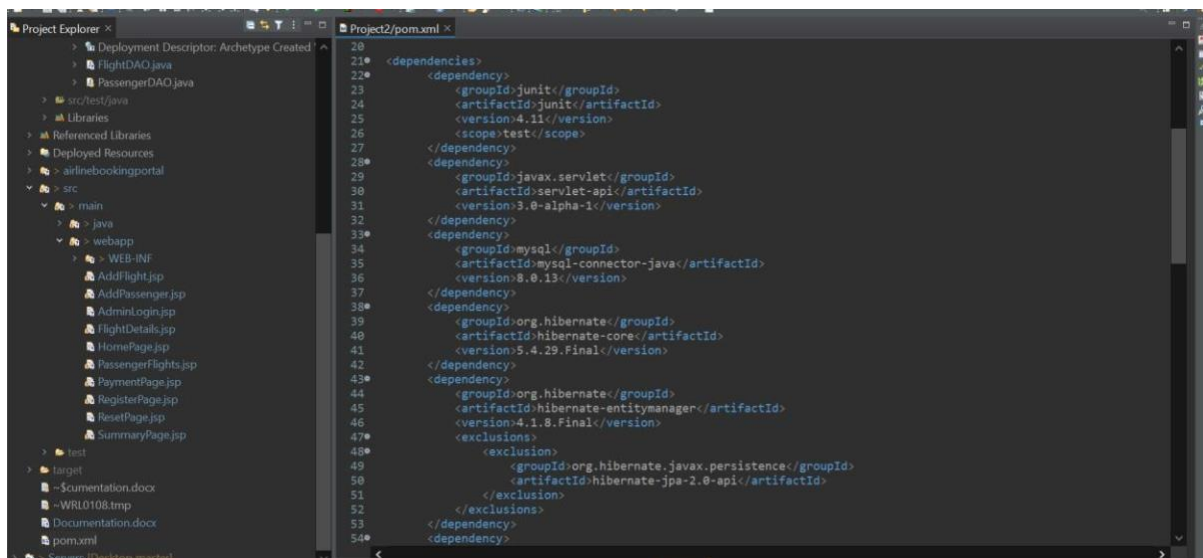
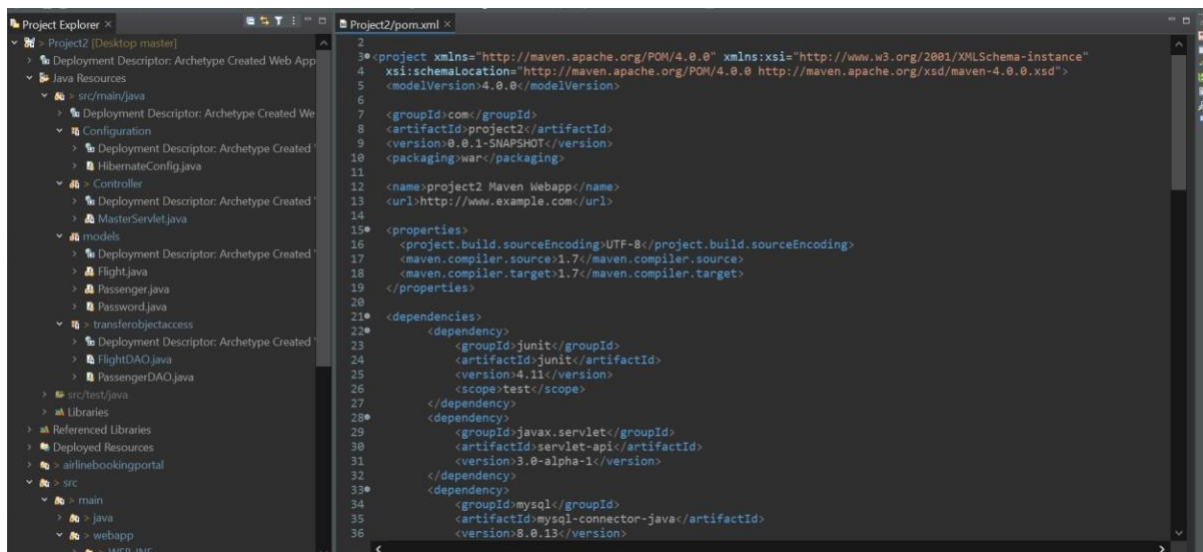
```

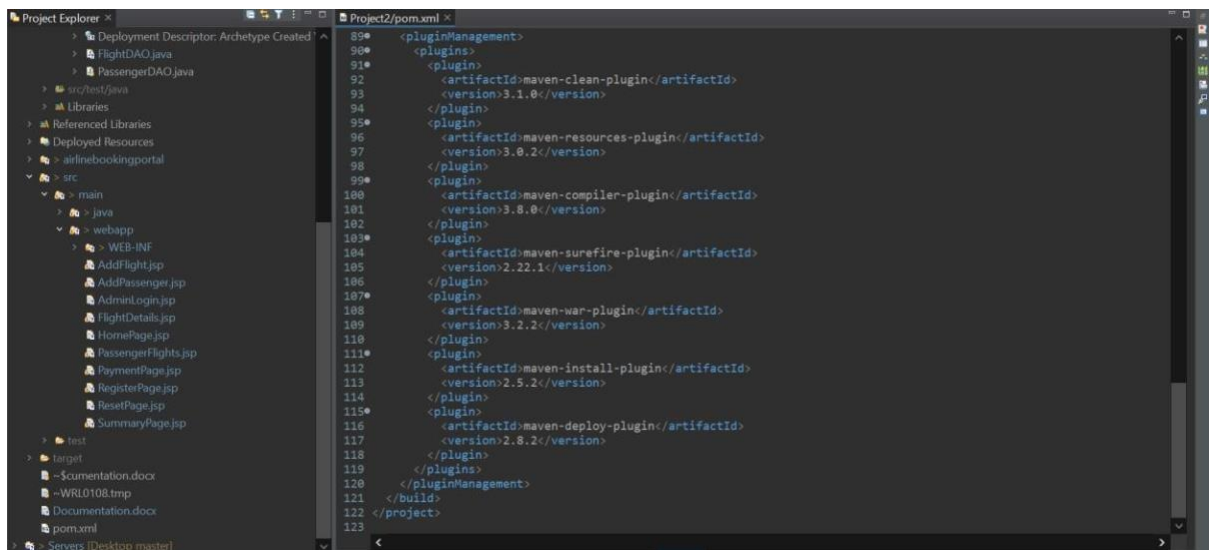
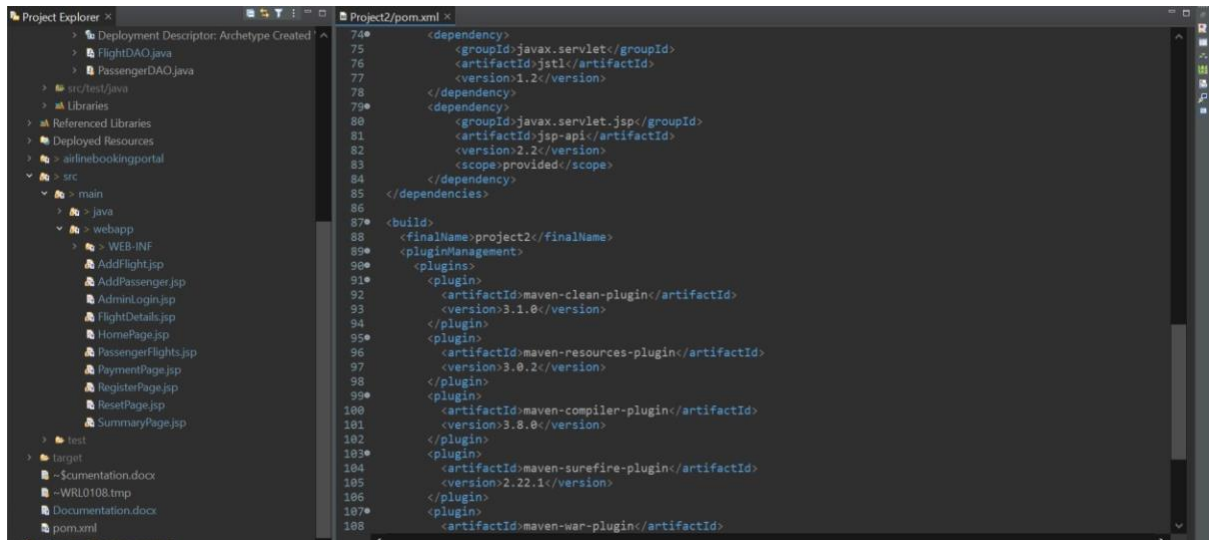
```

272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287 private void login(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
288
289     String email = request.getParameter("email");
290     String pwd = request.getParameter("pwd");
291
292     RequestDispatcher rd = null;
293
294     if (email.equalsIgnoreCase("admin@test.com") && pwd.equals(pd)) {
295         rd = request.getRequestDispatcher("view");
296         rd.forward(request, response);
297     } else {
298         rd = request.getRequestDispatcher("AdminLogin.jsp");
299         PrintWriter out = response.getWriter();
300         rd.include(request, response);
301         out.println("<center> <span style='color:red'> Invalid Credentials!!! </span></center>");
302     }
303 }
304
305
306

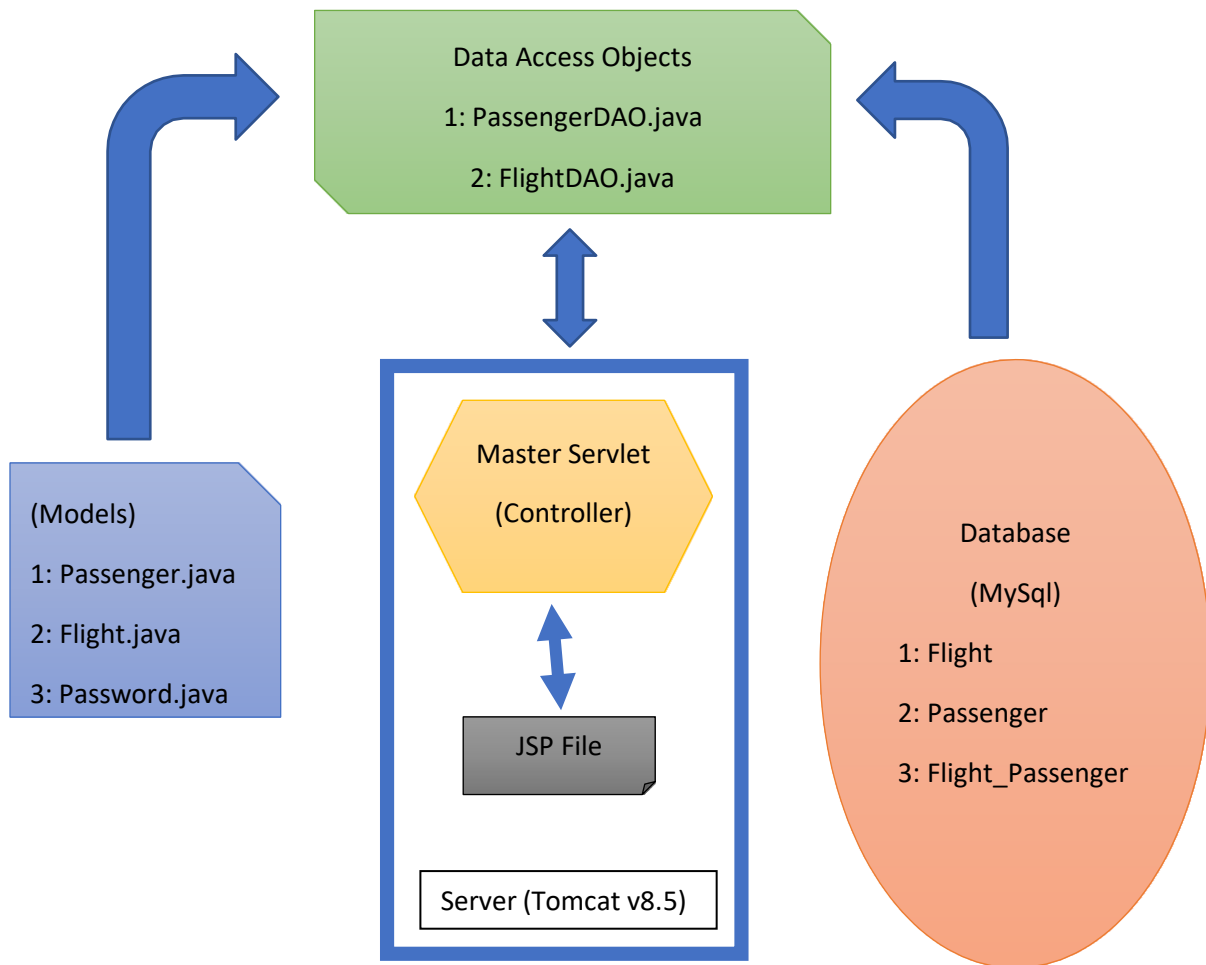
```

6: Program Structure and pom.xml





Flow Diagram



- ❖ Core Concepts used in this project are mostly maven, hibernate, MySQL, jdbc connector, associations, java, CRUD operations in a database, web development.

Algorithm

Step 1> Start.

Step 2> Two options in the home page:

Case 1: If user select "admin" section then go to step 3.

Case 2: If user goes to passenger side through registering all booking details, then go step 7.

Step 3> Once a user select admin, it will prompt for admin email and admin password.

Step 4> An admin main page will be displayed containing three options and showing list of flights that admin has entered:

Case 1: Add Flights -> step 5.

Case 2: Change Password -> step 6.

Case 3: Logout and go back to step 2.

Step 5> A new window will be shown where admin can enter flight details and go back step 4.

Step 6> For changing password, it will ask for new and confirmation password from admin. Once it is validated correctly it will go back to home page.

Step 7> This will show a window having flights details that are filtered out through booking details. And the user has to select the flight for further action.

Step 8> Once, the user has selected the flight, user has to register his/her details and should be less than or equal to number of persons that user has specified.

Step 9> After continuation, it will show the summary of user's flight and will prompt the user for payment (Dummy Payment Gateway).

Step 10> Once payment is successful, it will take the user back to step 2.

Step 11> Stop

Conclusion

1: The prototype is robust and platform independent.

2: User can easily use the prototype and safely exit out of it.

3: As a developer, we can enhance it by introducing several new features such as guards along each web pages as currently its statically connected with each along with backend as will not allow to go back once admin has been logout, routing, custom validators and can have more user-friendly by adding styling (CSS, Bootstrap), custom loaders.

4: Though this prototype is tightly connected, the data will only persist in database until server is running and gets reset with each restarting of sever because of manual configuration of hibernate.

5: This prototype can also be implemented with multithreading to enable better performance.

6: And lastly, this prototype can be upgraded by implementing with securities patches to make it more versatile and secure in both local environment and global and later can be configured dynamically with connection of database through hibernate.