

flyaway.com

Airline Booking Portal

Prototype of the Application

Name : Anurag Sharma

GitHub : <https://github.com/Instantgaming2356/JAVAFSD-Project02>

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

A screenshot of a web browser window titled "FLYAWAY.COM". The URL bar shows "localhost:8080/project2/". The page has a header "Admin". Below it is a form with four input fields: "Origin", "Destination", "Boarding Date" (with a placeholder "dd / mm / yyyy"), and "Persons". A "Search" button is at the bottom right of the form.

### 2: Admin Login Page (From admin side)

A screenshot of a web browser window titled "FLYAWAY.COM". The URL bar shows "localhost:8080/project2/AdminLogin.jsp". The page title is "Admin Login". It contains two input fields: "Email" with value "admin@test.com" and "Password" with value "\*\*\*\*\*". A "Login" button is below the fields.

### 3: Admin main page (From admin side)

A screenshot of a web browser window titled "FLYAWAY.COM". The URL bar shows "localhost:8080/project2/login". The page title is "Flight Management". It features three navigation links: "Add Flight", "Change Password", and "Logout". Below these is a table header "Flight Details" with columns: "ID", "Number", "Air Name", "Origin", "Destination", "Date", "Booking Price", and "Status".

### 4: Add Flight Page (From admin side)

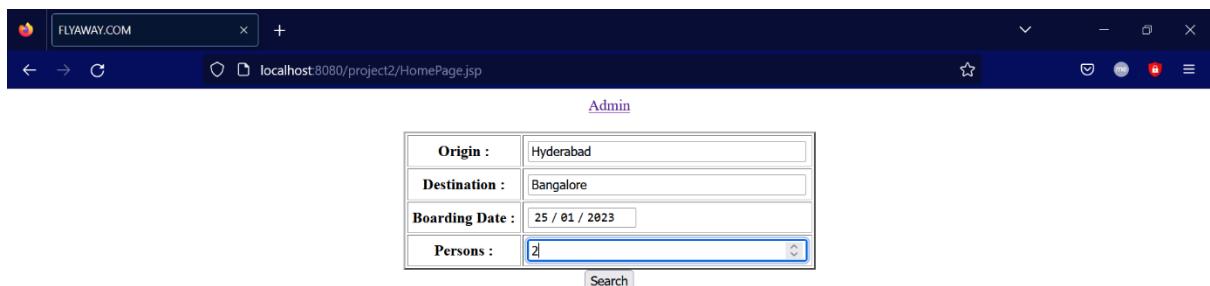
A screenshot of a web browser window titled "FLYAWAY.COM". The URL bar shows "localhost:8080/project2/add". The page title is "Flight Management". It contains a form with six input fields: "Flight Number" (426), "Airline" (Indigo), "Origin" (Bengalore), "Destination" (Bhubaneswar), "Flight Date" (30 / 01 / 2023), and "Ticket Price" (5500). A "Save" button is at the bottom right of the form.

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



After, Admin can logout back to the homepage.

5: Insert the booking details from client side.



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



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



### 8: Add Passenger Page, Entering Passenger Details

A screenshot of a web browser window titled "FLYAWAY.COM". The URL is "localhost:8080/project2/AddPassenger.jsp". The page contains a form with the following fields:

- First Name :
- Last Name :
- Contact :
- Age :
- Email :

A "Save" button is located at the bottom right of the form.

A screenshot of a web browser window titled "FLYAWAY.COM". The URL is "localhost:8080/project2/booking". The page displays the following content:

### Passenger Details

Add Passenger

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

[Delete](#) [Confirm](#)

### 9: Summary Page

A screenshot of a web browser window titled "FLYAWAY.COM". The URL is "localhost:8080/project2/register". The page displays the following content:

### 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

A screenshot of a web browser window titled "FLYAWAY.COM". The URL is "localhost:8080/project2/PaymentPage.jsp". The page displays the following content:

Payment Method :

[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 demo;
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 (0.00 sec)

mysql> select * from flight;
+-----+-----+-----+-----+-----+-----+
| flight_id | flight_name | Boarding_Date | flight_number | Source | Ticket_price | Destination |
+-----+-----+-----+-----+-----+-----+
| 1 | Indigo | 2023-01-30 | 426 | Bengalore | 5500 | Bhubaneswar |
| 2 | Vistara | 2023-01-25 | 673 | Hyderabad | 3500 | Bangalore |
| 3 | Air India | 2023-01-27 | 367 | Bengalore | 3000 | Chennai |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

5 rows in set (0.00 sec)

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 | 23 | 12345678 | test@gmail.com | Anurag | Sharma |
| 2 | 25 | 23456781 | test@test.com | Rahul | Singh |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

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

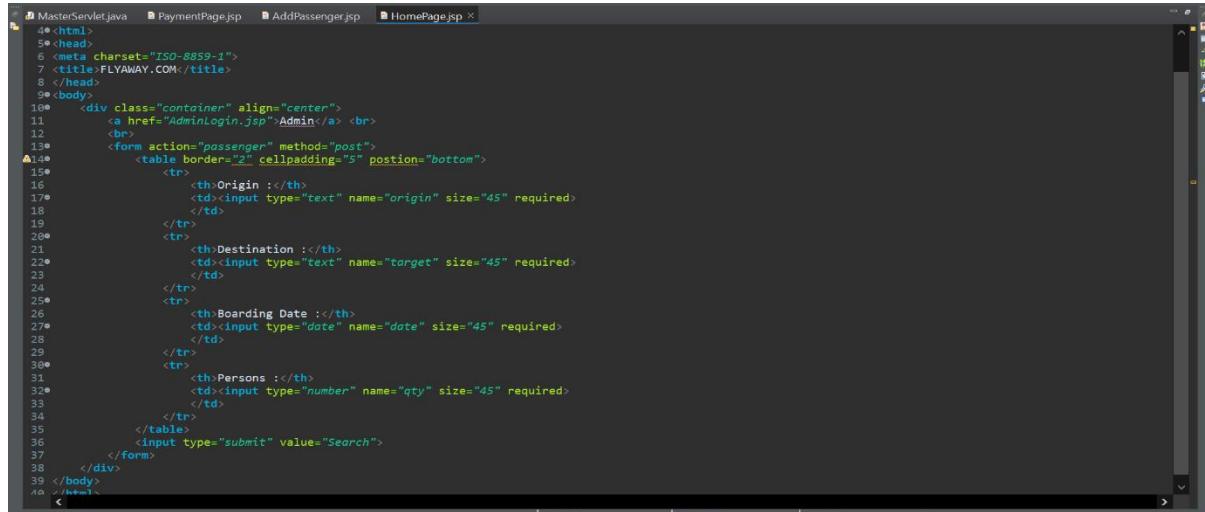
```
mysql> select * from flight_passenger;
+-----+-----+
| flight_id | passenger_id |
+-----+-----+
| 4 | 1 |
| 4 | 2 |
+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from flight;
+-----+-----+-----+-----+-----+-----+
| flight_id | flight_name | Boarding_Date | flight_number | Source | Ticket_price | Destination |
+-----+-----+-----+-----+-----+-----+
| 1 | Indigo | 2023-01-30 | 426 | Bengalore | 5500 | Bhubaneswar |
| 2 | Vistara | 2023-01-25 | 673 | Hyderabad | 3500 | Bangalore |
| 3 | Air India | 2023-01-27 | 367 | Bengalore | 3000 | Chennai |
| 4 | Vistara | 2023-01-25 | 673 | Hyderabad | 3500 | Bangalore |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

## Source Code:

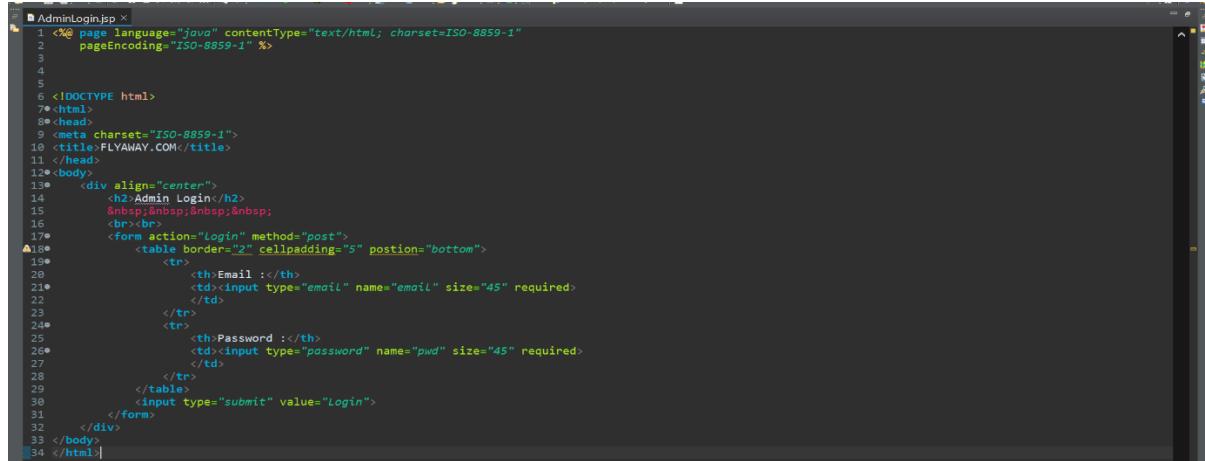
### 1: JSP Pages

#### a> Home Page (HomePage.jsp)



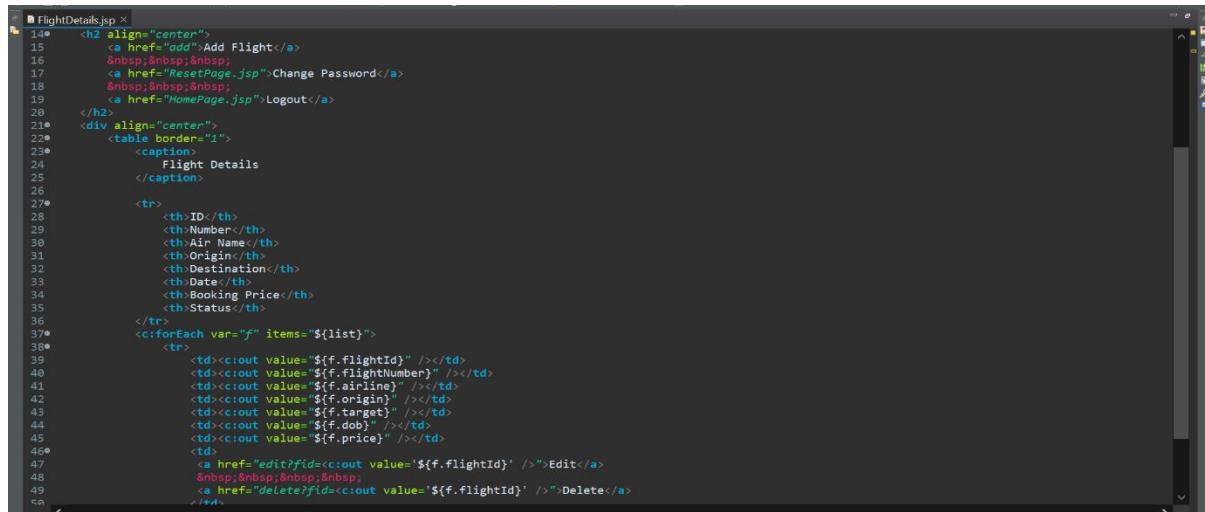
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>FLYAWAY .COM</title>
</head>
<body>
<div class="container" align="center">
<a href="AdminLogin.jsp">Admin</a> <br>
<br>
<form action="passenger" 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)



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>FLYAWAY .COM</title>
</head>
<body>
<div align="center">
<h2>Admin Login</h2>
&ampnbsp&ampnbsp&ampnbsp&ampnbsp
<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)



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
<head>
<title>Flight Details</title>
</head>
<body>
<h2 align="center">
<a href="add">Add Flight</a>
&ampnbsp&ampnbsp&ampnbsp
<a href="ResetPage.jsp">Change Password</a>
&ampnbsp&ampnbsp&ampnbsp
<a href="HomePage.jsp">Logout</a>
</h2>
<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>
<cc:forEach var="f" items="${list}">
<tr>
<td><cc:out value="${f.flightId}" /></td>
<td><cc:out value="${f.flightNumber}" /></td>
<td><cc:out value="${f.airline}" /></td>
<td><cc:out value="${f.origin}" /></td>
<td><cc:out value="${f.target}" /></td>
<td><cc:out value="${f.dob}" /></td>
<td><cc:out value="${f.price}" /></td>
<td>
<a href="edit?fid=<cc:out value="${f.flightId}" />">Edit</a>
&ampnbsp&ampnbsp&ampnbsp
<a href="Delete?fid=<cc:out value="${f.flightId}" />">Delete</a>
</td>
</tr>
</cc:forEach>
</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" %>
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="${f != null}">
15        <form action="update" method="post">
16      </c:if>
17      <c:if test="${f == null}">
18        <form action="insert" method="post">
19      </c:if>
20      <table border="1" cellpadding="5">
21        <c:if test="${f != null}">
22          <input type="hidden" name="fid"
23            value=<c:out value="${f.flightId}" />" />
24        </c:if>
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    </form>
61  </div>
62 </body>
63 </html>
```

```
1 <tr><td><input type="number" name="fnumber" size="45"
2   value=<c:out value="${f.flightNumber}" />" required /></td>
3 </tr>
4 <tr>
5   <th>Airline :</th>
6   <td><input type="text" name="fname" size="45"
7     value=<c:out value="${f.airline}" />" required /></td>
8 </tr>
9 <tr>
10   <th>Origin :</th>
11   <td><input type="text" name="forigin" size="45"
12     value=<c:out value="${f.origin}" />" required /></td>
13 </tr>
14 <tr>
15   <th>Destination :</th>
16   <td><input type="text" name="ftarget" size="45"
17     value=<c:out value="${f.target}" />" required /></td>
18 </tr>
19 <tr>
20   <th>Flight Date :</th>
21   <td><input type="date" name="fdate" size="45"
22     value=<c:out value="${f.dob}" />" required /></td>
23 </tr>
24 <tr>
25   <th>Ticket Price :</th>
26   <td><input type="number" name="fprice" size="45"
27     value=<c:out value="${f.price}" />" required /></td>
28 </tr>
29 <tr>
30   <td colspan="2" align="center"><input type="submit"
31     value="Save" /></td>
32 </tr>
33 </table>
34 </form>
35 </div>
36 </body>
37 </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>FLYAWAY.COM</title>
8   </head>
9   <body>
10    <div align="center">
11      <h2>Reset Password</h2>
12      &nbsp;&nbsp;&nbsp;&nbsp;
13      <br><br>
14      <form action="reset" method="post">
15        <table border="2" cellpadding="5" position="bottom">
16          <tr>
17            <th>Enter new Password :</th>
18            <td><input type="password" name="newpwd" size="45" required>
19            </td>
20          </tr>
21          <tr>
22            <th>Confirm Password :</th>
23            <td><input type="password" name="confpwd" size="45" required>
24            </td>
25          </tr>
26        </table>
27        <input type="submit" value="Save">
28      </form>
29    </div>
30  </body>
31 </html>
```

## f> Passenger Flights Page (PassengerFlights.jsp)

```
PassengerFlights.jsp
8<html>
9<head>
10<meta charset="ISO-8859-1">
11<title>FLYAWAY.COM</title>
12</head>
13<body>
14<div align="center">
15    <h2>Selected Flight Details:</h2>
16    &nbsp;&nbsp;&nbsp;&nbsp;
17    <table border="1">
18        <tr>
19            <th>ID</th>
20            <th>Number</th>
21            <th>Air Name</th>
22            <th>Origin</th>
23            <th>Destination</th>
24            <th>Date</th>
25            <th>Booking Price</th>
26            <th>Status</th>
27        </tr>
28        <c:forEach var="f" items="${Selectedlist}">
29            <tr>
30                <td><c:out value="${f.flightId}" /></td>
31                <td><c:out value="${f.flightNumber}" /></td>
32                <td><c:out value="${f.airline}" /></td>
33                <td><c:out value="${f.origin}" /></td>
34                <td><c:out value="${f.target}" /></td>
35                <td><c:out value="${f.dob}" /></td>
36                <td><c:out value="${f.price}" /></td>
37                <td><a href="find?flds=<c:out value='${f.flightId}'%>">Book Now</a></td>
38            </tr>
39        </c:forEach>
40    </table>
41</div>
42</body>
43</html>
```

## g> Passenger Register Page (RegisterPage.jsp)

## Add Passenger (AddPassenger.jsp)

```
■ AddPassenger.jsp
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1" isELIgnored="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     <cif test="${p != null}">
15       <form action="updatePassenger" method="post">
16     </cif>
17     <cif test="${p == null}">
18       <form action="insertPassenger" method="post">
19     </cif>
20     <table border="1" cellpadding="5">
21       <cif test="${f != null}">
22         <input type="hidden" name="id"
23           value=<cout value='${p.pid}' /> />
24       </cif>
25     <tr>
26       <th>First Name :</th>
27       <td><input type="text" name="fname" size="45"
28           value=<cout value='${p.fname}' /> required /></td>
29     </tr>
30     <tr>
31       <th>Last Name :</th>
32       <td><input type="text" name="lname" size="45"
33           value=<cout value='${p.lname}' /> required /></td>
34     </tr>
35     <tr>
36       <th>Contact :</th>
37       <td><input type="number" name="contact" size="45"
38           value=<cout value='${p.contact}' /> /></td>
39     </tr>
40   </table>
41   <input type="submit" value="Submit" />
42 </cif>
43 </div>
```

```

AddPassenger.jsp ×
1<form action="AddPassenger" method="post">
2    <table border="1">
3        <tr>
4            <td><input type="hidden" name="pid" value=">" />
5        </td>
6        <tr>
7            <th>First Name :</th>
8            <td><input type="text" name="fname" size="45" value="" required /></td>
9        </tr>
10       <tr>
11           <th>Last Name :</th>
12           <td><input type="text" name="lname" size="45" value="" required /></td>
13       </tr>
14       <tr>
15           <th>Contact :</th>
16           <td><input type="number" name="contact" size="45" value="" required /></td>
17       </tr>
18       <tr>
19           <th>Age :</th>
20           <td><input type="text" name="age" size="45" value="" required /></td>
21       </tr>
22       <tr>
23           <th>Email :</th>
24           <td><input type="email" name="email" size="45" value="" required /></td>
25       </tr>
26       <tr>
27           <td colspan="2" align="center"><input type="submit" value="Save" /></td>
28       </tr>
29   </table>
30 </form>
31 </div>
32 </body>
33 </html>

```

### i> Flight Summary Page (SummaryPage.jsp)

```

SummaryPage.jsp ×
1<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2    pageEncoding="ISO-8859-1" isErrorPage="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         &nbsp;&nbsp;&nbsp;
15         <table border="1">
16             <tr>
17                 <th>Flight Number :</th>
18                 <td><c:out value="${f.flightNumber}" /></td>
19             </tr>
20             <tr>
21                 <th>Flight Name :</th>
22                 <td><c:out value="${f.airline}" /></td>
23             </tr>
24             <tr>
25                 <th>Flight From :</th>
26                 <td><c:out value="${f.origin}" /></td>
27             </tr>
28             <tr>
29                 <th>Flight To :</th>
30                 <td><c:out value="${f.target}" /></td>
31             </tr>
32             <tr>
33                 <th>Flight Boarding Date :</th>
34                 <td><c:out value="${f.dob}" /></td>
35             </tr>
36             <tr>
37                 <th>Ticket Price :</th>
38                 <td><c:out value="${f.price * n}" /></td>
39             </tr>
40         </table>
41         <input type="submit" value="Payment">
42     </div>
43 </body>
44 </html>

```

### j> Dummy Payment Gateway (PaymentPage.jsp)

```

PaymentPage.jsp ×
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2    pageEncoding="ISO-8859-1" isErrorPage="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         &nbsp;&nbsp;&nbsp;
15         <table border="1" cellpadding="5" position="bottom">
16             <tr>
17                 <th>Payment Method :</th>
18                 <td><a href="#">PayPal</a></td>
19                 <td><a href="#">PayTm</a></td>
20                 <td><a href="#">Debit/Credit Cards</a></td>
21             </tr>
22         </table>
23         <br/>
24         <a href="HomePage.jsp">Back To Home</a>
25     </div>
26 </body>
27 </html>

```

## 2: Models (Passenger.java, Flight.java, Password.java)

a> Passenger.java

```
1 package models;
2
3 import java.util.ArrayList;
4
5 @Entity
6 @Table(name = "Passenger")
7 public class Passenger {
8
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    @Column(name="passenger_id")
12    private int pid;
13
14    @Column(name="passenger_fname")
15    private String fname;
16
17    @Column(name="passenger_lname")
18    private String lname;
19
20    @Column(name="passenger_age")
21    private int age;
22
23    @Column(name="passenger_mob")
24    private long contact;
25
26    @Column(name="passenger_email")
27    private String email;
28
29    @ManyToMany(mappedBy = "passenger")//, cascade = CascadeType.MERGE)
30    private List<Flight> flight = new ArrayList<Flight>();
31
32    public Passenger() {
33    }
34
35    public Passenger(int pId, String fname, String lname, int age, long contact, String email) {
36        super();
37        this.pid = pId;
38        this.fname = fname;
39        this.lname = lname;
40        this.age = age;
41        this.contact = contact;
42        this.email = email;
43    }
44}
```

```
45
46    public Passenger(int pId, String fname, String lname, int age, long contact, String email) {
47        super();
48        this.pid = pId;
49        this.fname = fname;
50        this.lname = lname;
51        this.age = age;
52        this.contact = contact;
53        this.email = email;
54    }
55
56
57    public Passenger(String fname, String lname, int age, long contact, String email) {
58        super();
59        this.fname = fname;
60        this.lname = lname;
61        this.age = age;
62        this.contact = contact;
63        this.email = email;
64    }
65
66    public int getId() {
67        return pid;
68    }
69
70    public void setId(int pId) {
71        this.pid = pId;
72    }
73
74    public String getFname() {
75        return fname;
76    }
77
78    public void setFname(String fname) {
79        this.fname = fname;
80    }
81}
```

```
82
83    public String getLname() {
84        return lname;
85    }
86
87    public void setLname(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}
```

```
B Passenger.java ×
1 package models;
2
3     lname = lname;
4
5     }
6
7     public int getAge() {
8         return age;
9     }
10
11    public void setAge(int age) {
12        this.age = age;
13    }
14
15    public long getContact() {
16        return contact;
17    }
18
19    public void setContact(long contact) {
20        this.contact = contact;
21    }
22
23    public String getEmail() {
24        return email;
25    }
26
27    public void setEmail(String email) {
28        this.email = email;
29    }
30
31    public List<Flight> getFlight() {
32        return flight;
33    }
34
35    public void setFlight(List<Flight> flight) {
36        this.flight = flight;
37    }
38
39 }
```

## b> Flight.java

```
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        joinColumns = {
38            @JoinColumn(name = "flight_id")
39        },
40        inverseJoinColumns = {
41            @JoinColumn(name = "passenger_id")
42        }
43    )
44    List<Passenger> passenger = new ArrayList<Passenger>();
45
46
47 }
```

```
B Flight.java ×
48
49    @ManyToMany(cascade = CascadeType.ALL)
50    @JoinTable(name = "flight_passenger",
51        joinColumns = {
52            @JoinColumn(name = "flight_id")
53        },
54        inverseJoinColumns = {
55            @JoinColumn(name = "passenger_id")
56        }
57    )
58    List<Passenger> passenger = new ArrayList<Passenger>();
59
60
61    public Flight(int flightId, int flightNumber, String airline, String origin, String target, String dob, float price) {
62        super();
63        this.flightId = flightId;
64        this.flightNumber = flightNumber;
65        this.airline = airline;
66        this.origin = origin;
67        this.target = target;
68        this.dob = dob;
69        this.price = price;
70    }
71
72    public Flight(int flightNumber, String airline, String origin, String target, String dob, float price) {
73        super();
74        this.flightNumber = flightNumber;
75        this.airline = airline;
76        this.origin = origin;
77        this.target = target;
78        this.dob = dob;
79        this.price = price;
80    }
81
82    public int getFlightId() {
83        return flightId;
84    }
85
86 }
```

```

1 Flight.java X
2
3     public int getFlightId() {
4         return flightId;
5     }
6
7     public void setFlightId(int flightId) {
8         this.flightId = flightId;
9     }
10
11     public int getFlightNumber() {
12         return flightNumber;
13     }
14
15     public void setFlightNumber(int flightNumber) {
16         this.flightNumber = flightNumber;
17     }
18
19     public String getAirline() {
20         return airline;
21     }
22
23     public void setAirline(String airline) {
24         this.airline = airline;
25     }
26
27     public String getOrigin() {
28         return origin;
29     }
30
31     public void setOrigin(String origin) {
32         this.origin = origin;
33     }
34
35     public String getTarget() {
36         return target;
37     }
38
39
40 Flight.java X
41     this.origin = origin;
42
43     public String getTarget() {
44         return target;
45     }
46
47     public void setTarget(String target) {
48         this.target = target;
49     }
50
51     public String getDob() {
52         return dob;
53     }
54
55     public void setDob(String dob) {
56         this.dob = dob;
57     }
58
59     public float getPrice() {
60         return price;
61     }
62
63     public void setPrice(float price) {
64         this.price = price;
65     }
66
67     public List<Passenger> getPassenger() {
68         return passenger;
69     }
70
71     public void setPassenger(List<Passenger> passenger) {
72         this.passenger = passenger;
73     }
74
75
76

```

## c> Password.java

```

1 Password.java X
2
3 package models;
4
5 public class Password {
6
7     private static String pwd;
8
9     public Password() {
10        pwd = "admin";
11    }
12
13     public static String getPwd() {
14         return pwd;
15     }
16
17     public static void setPwd(String pwd) {
18         Password.pwd = pwd;
19     }
20
21
22

```

## 3: Data Access Objects

### a> PassengerDAO.java

```

1 PassengerDAO.java X
2
3 package transferobjectaccess;
4
5 import java.util.List;
6
7 public class PassengerDAO {
8
9     @SuppressWarnings("unchecked")
10    public List<Flight> getAllDetailsByOriginDate(String origin, String date, String target) {
11
12        Transaction transaction = null;
13        Session dbSession = null;
14        List<Flight> list = null;
15
16        try {
17            dbSession = HibernateConfig.getSessionFactory().openSession();
18            transaction = dbSession.beginTransaction();
19            @SuppressWarnings("rawtypes")
20            Query query = dbSession
21                .createQuery("from Flight f where f.origin = :origin and f.target = :target and f.dob = :date");
22            query.setParameter("origin", origin);
23            query.setParameter("target", target);
24            query.setParameter("date", date);
25            list = query.list();
26        } catch (Exception e) {
27            if (transaction != null) {
28                transaction.rollback();
29            }
30            e.printStackTrace();
31        } finally {
32            dbSession.close();
33        }
34        return list;
35    }
36
37    public void insertPassengerInDB(Passenger p) {
38
39        Transaction transaction = null;
40
41    }
42
43
44

```

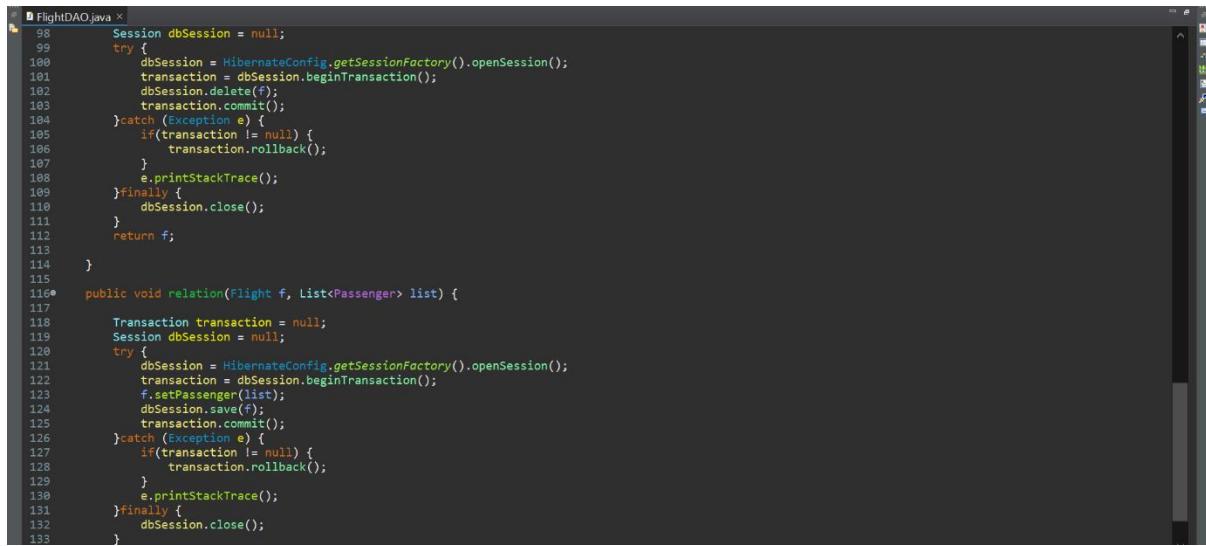
```
① PassengerDAO.java ×
42
43*     public void insertPassengerInDB(Passenger p) {
44
45         Transaction transaction = null;
46         Session dbSession = null;
47         try {
48             dbSession = HibernateConfig.getSessionFactory().openSession();
49             transaction = dbSession.beginTransaction();
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
72         } catch (Exception e) {
73             e.printStackTrace();
74         } finally {
75             dbSession.close();
76         }
77     }
78 }
```

```
① PassengerDAO.java ×
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 }
```

```
① PassengerDAO.java ×
68         transaction = dbSession.beginTransaction();
69         p = dbSession.get(Passenger.class, id);
70     } catch (Exception e) {
71         if (transaction != null) {
72             transaction.rollback();
73         }
74         e.printStackTrace();
75     } finally {
76         dbSession.close();
77     }
78     return p;
79
80 }
81
82     public Passenger deletePassenger(Passenger p) {
83
84         Transaction transaction = null;
85         Session dbSession = null;
86         try {
87             dbSession = HibernateConfig.getSessionFactory().openSession();
88             transaction = dbSession.beginTransaction();
89             dbSession.delete(p);
90             transaction.commit();
91         } catch (Exception e) {
92             if (transaction != null) {
93                 transaction.rollback();
94             }
95             e.printStackTrace();
96         } finally {
97             dbSession.close();
98         }
99     return p;
100
101 }
```

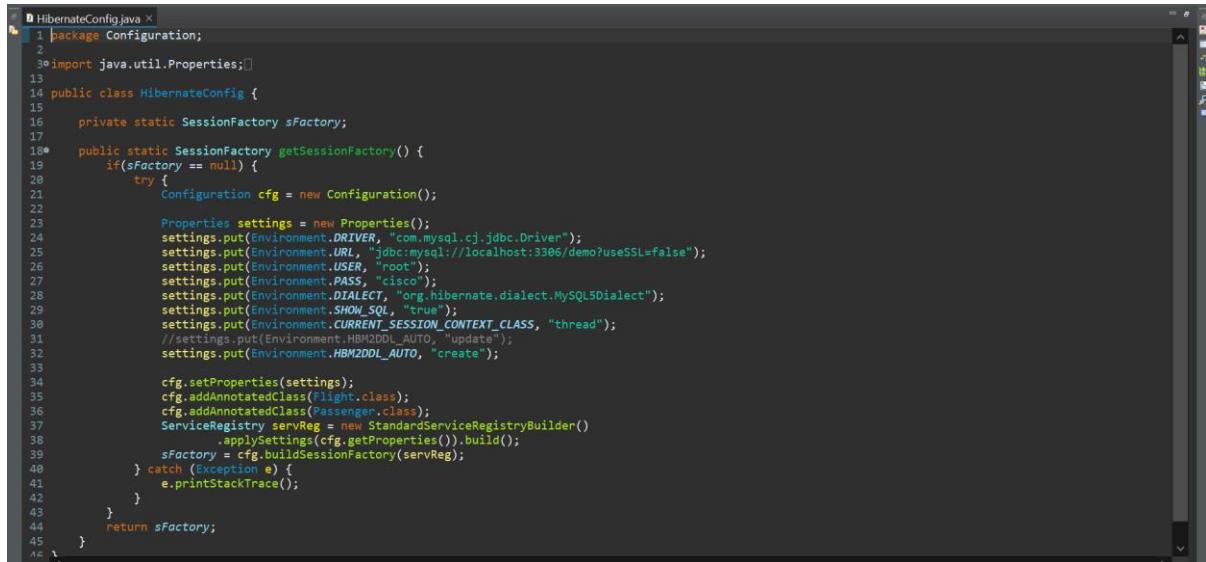
## b> FlightDAO.java

```
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
12        try {
13            dbSession = HibernateConfig.getSessionFactory().openSession();
14            transaction = dbSession.beginTransaction();
15            dbSession.save(f);
16            transaction.commit();
17        }catch (Exception e) {
18            if(transaction != null) {
19                transaction.rollback();
20            }
21            e.printStackTrace();
22        }finally {
23            dbSession.close();
24        }
25    }
26
27
28    @SuppressWarnings("unchecked")
29    public List<Flight> getAllDetails() {
30
31        Session dbSession = null;
32        List<Flight> lists=null;
33
34        try {
35            dbSession = HibernateConfig.getSessionFactory().openSession();
36            list = dbSession.createQuery("from Flight").list();
37
38        }catch (Exception e) {
39            e.printStackTrace();
40        }finally {
41            dbSession.close();
42        }
43    }
44
45
46    public Flight getFlightById(int flightId) {
47
48        Transaction transaction = null;
49        Session dbSession = null;
50        Flight f = null;
51
52        try {
53            dbSession = HibernateConfig.getSessionFactory().openSession();
54            transaction = dbSession.beginTransaction();
55            f = dbSession.get(Flight.class, flightId);
56
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
73        try {
74            dbSession = HibernateConfig.getSessionFactory().openSession();
75            transaction = dbSession.beginTransaction();
76            dbSession.update(f);
77            transaction.commit();
78        }catch (Exception e) {
79            if(transaction != null) {
80                transaction.rollback();
81            }
82            e.printStackTrace();
83        }finally {
84            dbSession.close();
85        }
86        return f;
87    }
88
89    public Flight deleteFlight(Flight f) {
90
91        Transaction transaction = null;
92        Session dbSession = null;
93
94        try {
95            dbSession = HibernateConfig.getSessionFactory().openSession();
96            transaction = dbSession.beginTransaction();
97            dbSession.delete(f);
98            transaction.commit();
99        }catch (Exception e) {
100            if(transaction != null) {
101                transaction.rollback();
102            }
103            e.printStackTrace();
104        }finally {
105            dbSession.close();
106        }
107    }
108}
```



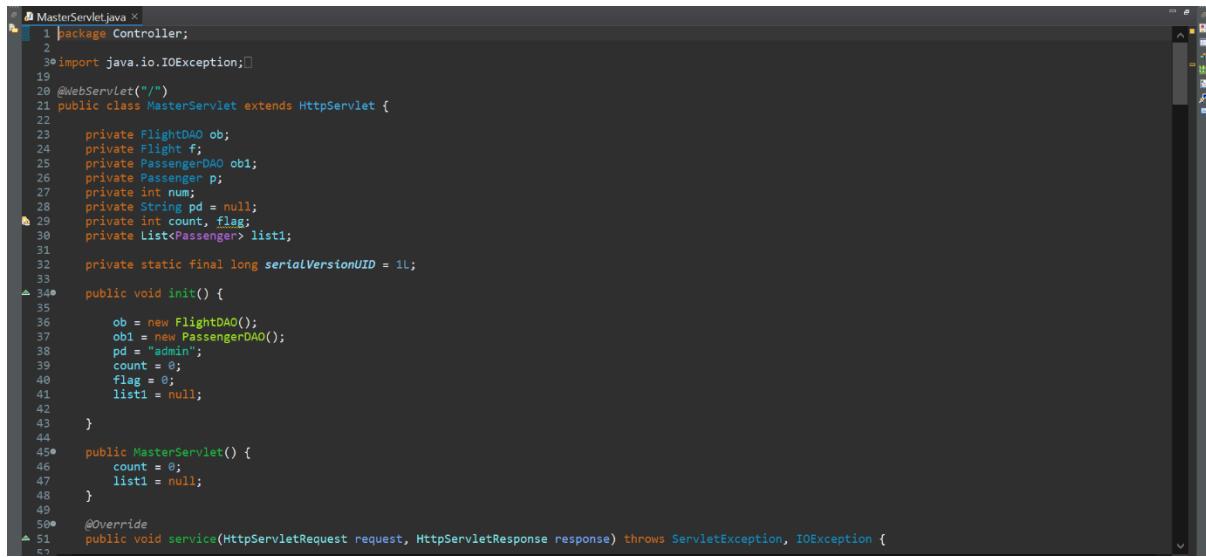
```
88     Session dbSession = null;
89     try {
90         dbSession = HibernateConfig.getSessionFactory().openSession();
91         transaction = dbSession.beginTransaction();
92         dbSession.delete(f);
93         transaction.commit();
94     }catch (Exception e) {
95         if(transaction != null) {
96             transaction.rollback();
97         }
98         e.printStackTrace();
99     }finally {
100        dbSession.close();
101    }
102    return f;
103}
104}
105
106 public void relation(Flight f, List<Passenger> list) {
107
108     Transaction transaction = null;
109     Session dbSession = null;
110     try {
111         dbSession = HibernateConfig.getSessionFactory().openSession();
112         transaction = dbSession.beginTransaction();
113         f.setPassenger(list);
114         dbSession.save(f);
115         transaction.commit();
116     }catch (Exception e) {
117         if(transaction != null) {
118             transaction.rollback();
119         }
120         e.printStackTrace();
121     }finally {
122        dbSession.close();
123    }
124}
```

#### 4: Hibernate Integration (HibernateConfig.java)



```
1 package Configuration;
2
3 import java.util.Properties;
4
5 public class HibernateConfig {
6
7     private static SessionFactory sFactory;
8
9     public static SessionFactory getSessionFactory() {
10        if(sFactory == null) {
11            try {
12                Configuration cfg = new Configuration();
13
14                Properties settings = new Properties();
15                settings.put(Environment.DRIVER, "com.mysql.cj.jdbc.Driver");
16                settings.put(Environment.URL, "jdbc:mysql://localhost:3306/demo?useSSL=false");
17                settings.put(Environment.USER, "root");
18                settings.put(Environment.PASS, "cisco");
19                settings.put(Environment.DIALECT, "org.hibernate.dialect.MySQLDialect");
20                settings.put(Environment.SHOW_SQL, "true");
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                sFactory = cfg.buildSessionFactory(servReg);
30            } catch (Exception e) {
31                e.printStackTrace();
32            }
33        }
34        return sFactory;
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
21        ob = new FlightDAO();
22        ob1 = new PassengerDAO();
23        pd = "admin";
24        count = 0;
25        flag = 0;
26        list1 = null;
27    }
28
29    public MasterServlet() {
30
31        count = 0;
32        list1 = null;
33    }
34
35    @Override
36    public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
37
```

```

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     }
109
110 }
111
112* private void BookingDetails(HttpServletRequest request, HttpServletResponse response)
113     throws ServletException, IOException {
114
115     List<Passenger> list = ob1.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 lname = 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, lname, age, contact, email);
134         ob1.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"));
143     ob1.deletePassenger(id);
144 }

```

```
# D MasterServlet.java ×
142     int id = Integer.parseInt(request.getParameter("id"));
143     Passenger p = ob.getPassengerById(id);
144     ob.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("name");
164     String forigin = request.getParameter("forigin");
165     String ftarget = request.getParameter("ftarget");
166     String date = request.getParameter("date");
167     float fprice = Float.parseFloat(request.getParameter("fprice"));
168     Flight f1 = new Flight(fid, fnumber, fname, forigin, ftarget, date, fprice);
169     ob.updateFlight(f1);
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"));
176 }
```

```
# D MasterServlet.java ×
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
184 private void showAllDetails(HttpServletRequest response, HttpServletResponse request)
185 throws ServletException, IOException {
186
187     List<Flight> list = ob.getAllDetails();
188     System.out.println(list);
189     request.setAttribute("list", list);
190     RequestDispatcher rd = request.getRequestDispatcher("FlightDetails.jsp");
191     rd.forward(request, response);
192 }
193
194
195 private void showNewForm(HttpServletRequest request, HttpServletResponse response)
196 throws ServletException, IOException {
197
198     RequestDispatcher rd = request.getRequestDispatcher("AddFlight.jsp");
199     rd.forward(request, response);
200 }
201
202
203 private void insertDetails(HttpServletRequest request, HttpServletResponse response) throws IOException {
204
205     int fnumber = Integer.parseInt(request.getParameter("fnumber"));
206     String fname = request.getParameter("name");
207     String forigin = request.getParameter("forigin");
208     String ftarget = request.getParameter("ftarget");
209     String date = request.getParameter("date");
210     float fprice = Float.parseFloat(request.getParameter("fprice"));
211     Flight f1 = new Flight(fnumber, fname, forigin, ftarget, date, fprice);
212     ob.insertFlightInDB(f1);
213     response.sendRedirect("view");
214 }
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 = ob1.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
234 private void getflightDetailsById(HttpServletRequest request, HttpServletResponse response)
235 throws ServletException, IOException {
236
237     int id = Integer.parseInt(request.getParameter("fid"));
238 }
```

```
# D MasterServlet.java ×
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("name");
208     String forigin = request.getParameter("forigin");
209     String ftarget = request.getParameter("ftarget");
210     String date = request.getParameter("date");
211     float fprice = Float.parseFloat(request.getParameter("fprice"));
212     Flight f1 = new Flight(fnumber, fname, forigin, ftarget, date, fprice);
213     ob.insertFlightInDB(f1);
214     response.sendRedirect("view");
215 }
216
217
218 private void passengerFlightDetails(HttpServletRequest request, HttpServletResponse response)
219 throws ServletException, IOException {
220
221     String origin = request.getParameter("origin");
222     String target = request.getParameter("target");
223     String date = request.getParameter("date");
224     num = Integer.parseInt(request.getParameter("qty"));
225
226     List<Flight> list = ob1.getAllDetailsByOriginDate(origin, date, target);
227     // System.out.println(list);
228     // request.setAttribute("num", num);
229     request.setAttribute("SelectedList", list);
230     RequestDispatcher rd = request.getRequestDispatcher("PassengerFlights.jsp");
231     rd.forward(request, response);
232 }
233
234
235 private void getflightDetailsById(HttpServletRequest request, HttpServletResponse response)
236 throws ServletException, IOException {
237
238     int id = Integer.parseInt(request.getParameter("fid"));
239 }
```

```
MasterServlet.java ×
232
233 }
234
235* private void getflightDetailsById(HttpServletRequest request, HttpServletResponse response)
236     throws ServletException, IOException {
237
238     int id = Integer.parseInt(request.getParameter("fid"));
239     FlightDAO x = new FlightDAO();
240     f = x.getFlightById(id);
241     // request.setAttribute("num", num);
242     RequestDispatcher rd = request.getRequestDispatcher("booking");
243     rd.forward(request, response);
244 }
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
```

```
MasterServlet.java ×
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("AdminHome.jsp");
296     } else {
297         RequestDispatcher rd = request.getRequestDispatcher("ResetPage.jsp");
298         PrintWriter out = response.getWriter();
299         rd.include(request, response);
300         out.println("<center> <span style='color:red'> Invalid Credentials!!! </span></center>");
301     }
302
303 }
```

```
MasterServlet.java ×
292
293     String newpwd = request.getParameter("newpwd");
294     String confpwd = request.getParameter("confpwd");
295
296     if (newpwd.compareTo(confpwd) == 0) {
297         pd = newpwd;
298         response.sendRedirect("AdminLogin.jsp");
299     } else {
300         RequestDispatcher rd = request.getRequestDispatcher("ResetPage.jsp");
301         PrintWriter out = response.getWriter();
302         rd.include(request, response);
303         out.println("<center> <span style='color:red'> Invalid Credentials!!! </span></center>");
304     }
305
306
307* private void login(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
308
309     String email = request.getParameter("email");
310     String pwd = request.getParameter("pwd");
311
312     RequestDispatcher rd = null;
313
314     if (email.equalsIgnoreCase("admin@test.com") && pwd.equals(pd)) {
315         rd = request.getRequestDispatcher("view");
316         rd.forward(request, response);
317     } else {
318         rd = request.getRequestDispatcher("AdminLogin.jsp");
319         PrintWriter out = response.getWriter();
320         rd.include(request, response);
321         out.println("<center> <span style='color:red'> Invalid Credentials!!! </span></center>");
322     }
323
324 }
```

## 6: Program Structure and pom.xml

The following three screenshots show the program structure and the pom.xml file for three different versions of the project.

**Screenshot 1 (Top):**

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com</groupId>
  <artifactId>project2</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>project2 Maven Webapp</name>
  <url>http://www.example.com</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId> servlet-api</artifactId>
      <version>3.0-alpha-1</version>
    </dependency>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.13</version>
    </dependency>
  </dependencies>

```

**Screenshot 2 (Middle):**

```

<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> servlet-api</artifactId>
    <version>3.0-alpha-1</version>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.13</version>
  </dependency>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId> hibernate-core</artifactId>
    <version>5.4.29.Final</version>
  </dependency>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId> hibernate-entitymanager</artifactId>
    <version>4.1.8.Final</version>
    <exclusions>
      <exclusion>
        <groupId>org.hibernate.javax.persistence</groupId>
        <artifactId> hibernate-jpa-2.0-api</artifactId>
      </exclusion>
    </exclusions>
  </dependency>

```

**Screenshot 3 (Bottom):**

```

<dependencies>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId> hibernate-entitymanager</artifactId>
    <version>4.1.8.Final</version>
    <exclusions>
      <exclusion>
        <groupId>org.hibernate.javax.persistence</groupId>
        <artifactId> hibernate-jpa-2.0-api</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
  <dependency>
    <groupId>javax.persistence</groupId>
    <artifactId> persistence-api</artifactId>
    <version>1.0.2</version>
  </dependency>
  <dependency>
    <groupId>commons-codec</groupId>
    <artifactId> commons-codec</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>org.glassfish.jaxb</groupId>
    <artifactId> jaxb-runtime</artifactId>
    <version>2.3.1</version>
  </dependency>
  <dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId> jaxb-api</artifactId>
    <version>2.3.1</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> jstl</artifactId>
    <version>1.2</version>
  </dependency>

```

The screenshot shows the Eclipse IDE interface with two main windows. On the left is the 'Project Explorer' view, which lists the project structure for 'Project2'. It includes a 'src' folder containing 'main' (with Java files like FlightDAO.java and PassengerDAO.java), 'webapp' (with JSP files such as AddFlight.jsp, AddPassenger.jsp, AdminLogin.jsp, FlightDetails.jsp, HomePage.jsp, PassengerFlights.jsp, PaymentPage.jsp, RegisterPage.jsp, ResetPage.jsp, and SummaryPage.jsp), and 'test' and 'target' folders. There are also documentation files like Documentation.docx and pom.xml. The right window is the 'pom.xml' editor, displaying the XML configuration for the Maven build. The code is as follows:

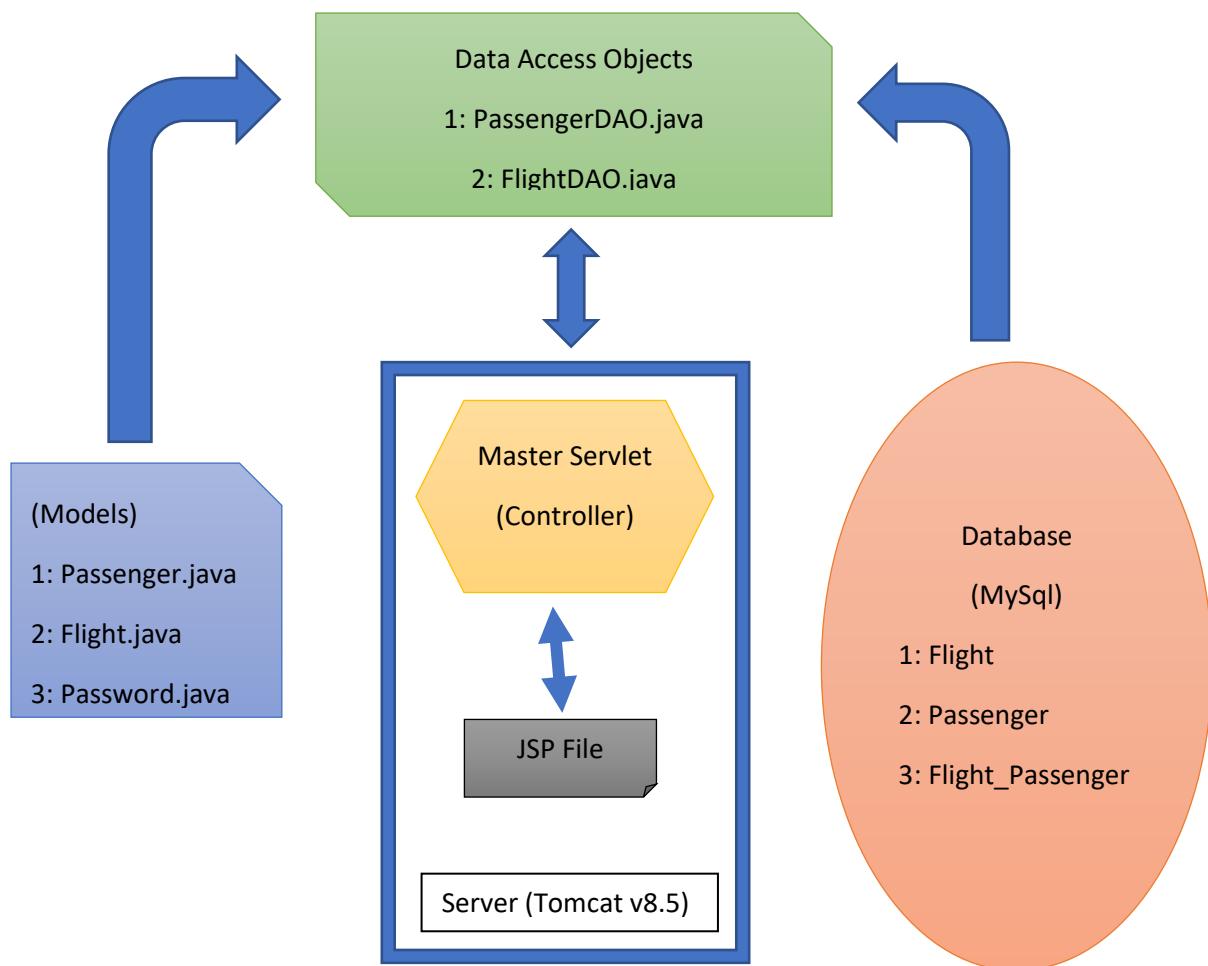
```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.2.</version>
    <scope>provided</scope>
</dependency>
</dependencies>
<build>
    <finalName>project2</finalName>
    <pluginManagement>
        <plugins>
            <plugin>
                <artifactId>maven-clean-plugin</artifactId>
                <version>3.1.0</version>
            </plugin>
            <plugin>
                <artifactId>maven-resources-plugin</artifactId>
                <version>3.0.2</version>
            </plugin>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.0</version>
            </plugin>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.22.1</version>
            </plugin>
            <plugin>
                <artifactId>maven-war-plugin</artifactId>

```

This screenshot is similar to the one above, showing the same project structure in the 'Project Explorer' and the 'pom.xml' editor in the 'Maven Dependencies' tab. The XML code in the editor has been updated to include additional plugins:

```
<pluginManagement>
    <plugins>
        <plugin>
            <artifactId>maven-clean-plugin</artifactId>
            <version>3.1.0</version>
        </plugin>
        <plugin>
            <artifactId>maven-resources-plugin</artifactId>
            <version>3.0.2</version>
        </plugin>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.0</version>
        </plugin>
        <plugin>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.22.1</version>
        </plugin>
        <plugin>
            <artifactId>maven-war-plugin</artifactId>
            <version>3.2.2</version>
        </plugin>
        <plugin>
            <artifactId>maven-install-plugin</artifactId>
            <version>2.5.2</version>
        </plugin>
        <plugin>
            <artifactId>maven-deploy-plugin</artifactId>
            <version>2.8.2</version>
        </plugin>
    </plugins>
</pluginManagement>
<build>
    <finalName>project2</finalName>
    <plugins>
        <plugin>
            <artifactId>maven-clean-plugin</artifactId>
            <version>3.1.0</version>
        </plugin>
        <plugin>
            <artifactId>maven-resources-plugin</artifactId>
            <version>3.0.2</version>
        </plugin>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.0</version>
        </plugin>
        <plugin>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.22.1</version>
        </plugin>
        <plugin>
            <artifactId>maven-war-plugin</artifactId>
            <version>3.2.2</version>
        </plugin>
        <plugin>
            <artifactId>maven-install-plugin</artifactId>
            <version>2.5.2</version>
        </plugin>
        <plugin>
            <artifactId>maven-deploy-plugin</artifactId>
            <version>2.8.2</version>
        </plugin>
    </plugins>
</build>
<project>
```

## 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.