FlyAway.com

Airline Booking Portal

Prototype of the Application

Name: RAJDEEP KUMAR

GAUTAM

GitHub: https://github.com/rajdeepkgautam/JavaFSDP2.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 must put his/her details according to the number of persons.

Sprint 6:

- In the registration page, there will be an 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.

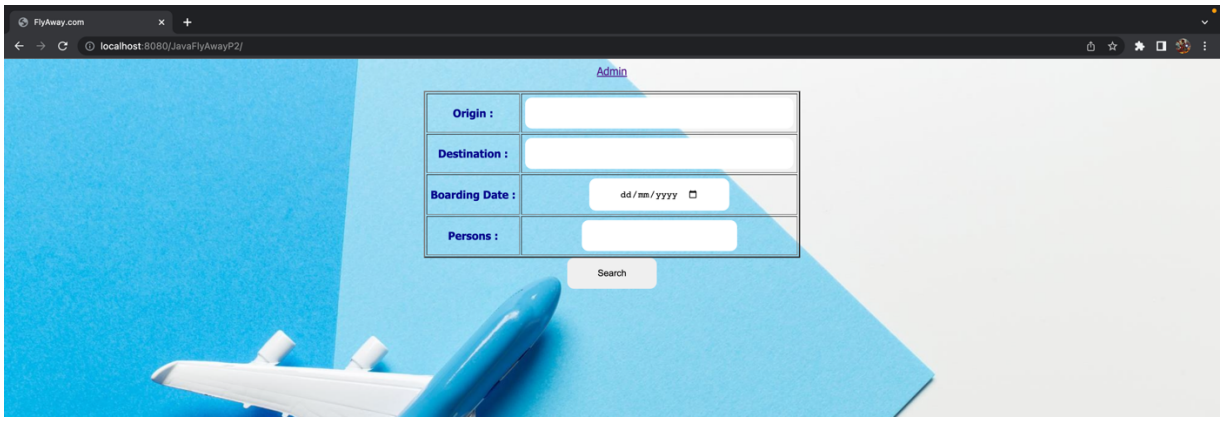
Sprint 7:

- 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.

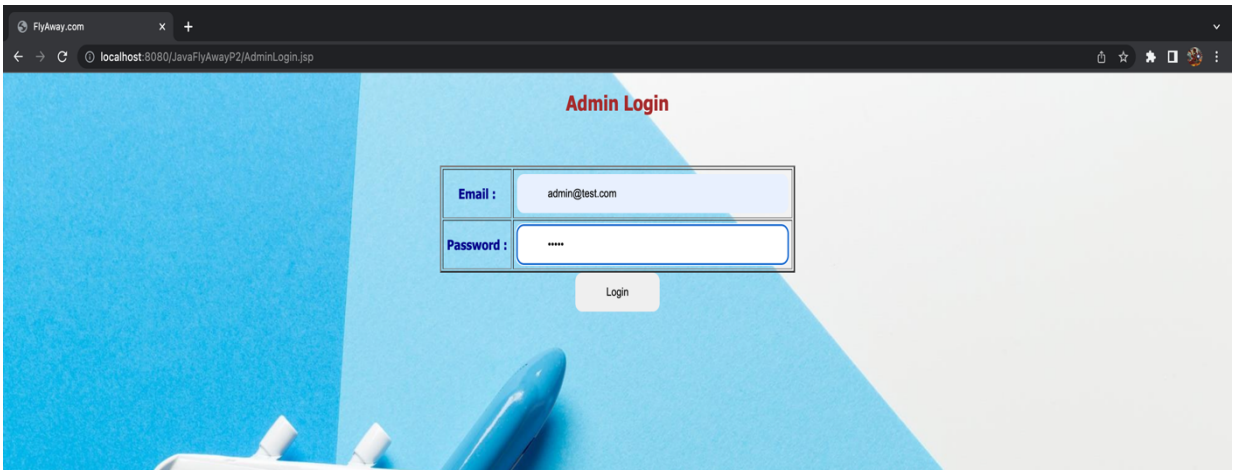
Documentation of the functionality:

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

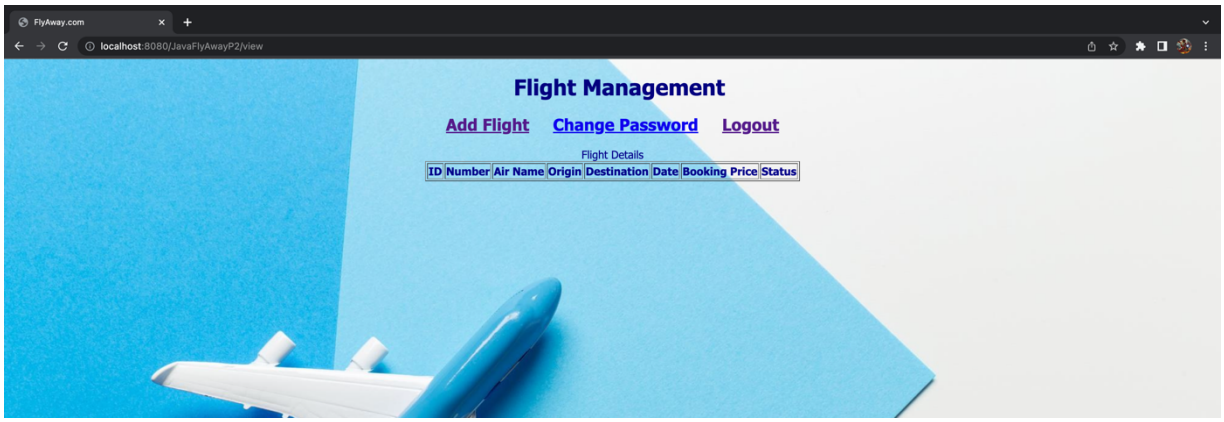
1: Home Page



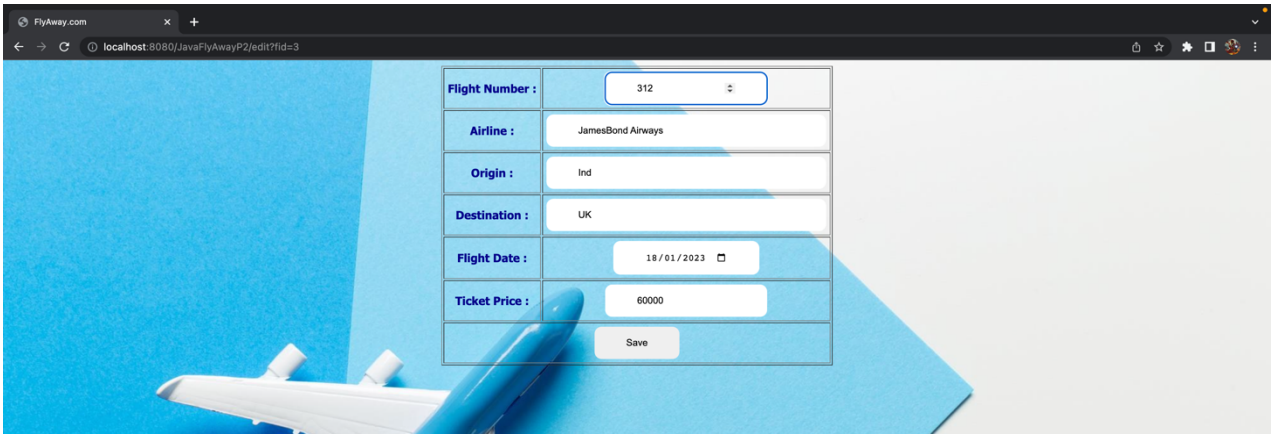
2: Admin Login Page (From admin side)



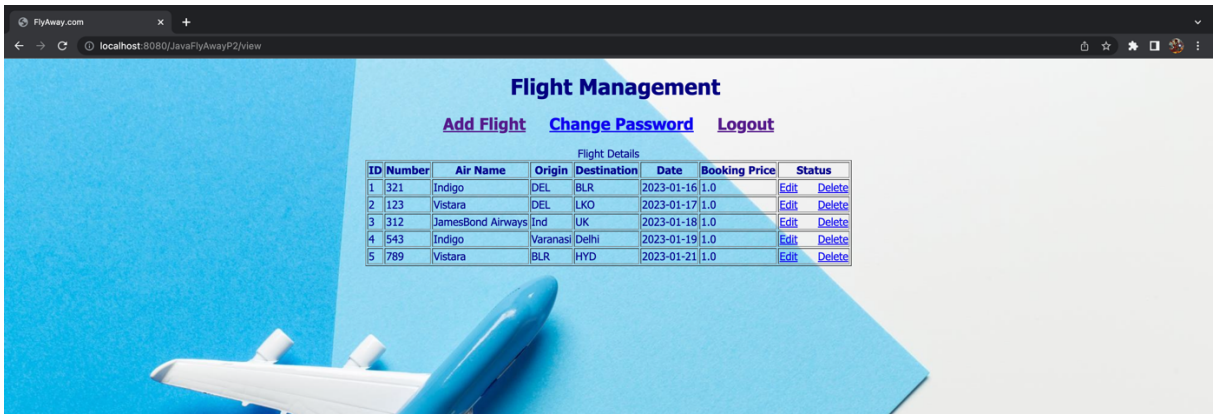
3: Admin main page (From admin side)



4: Add Flight Page (From admin side)

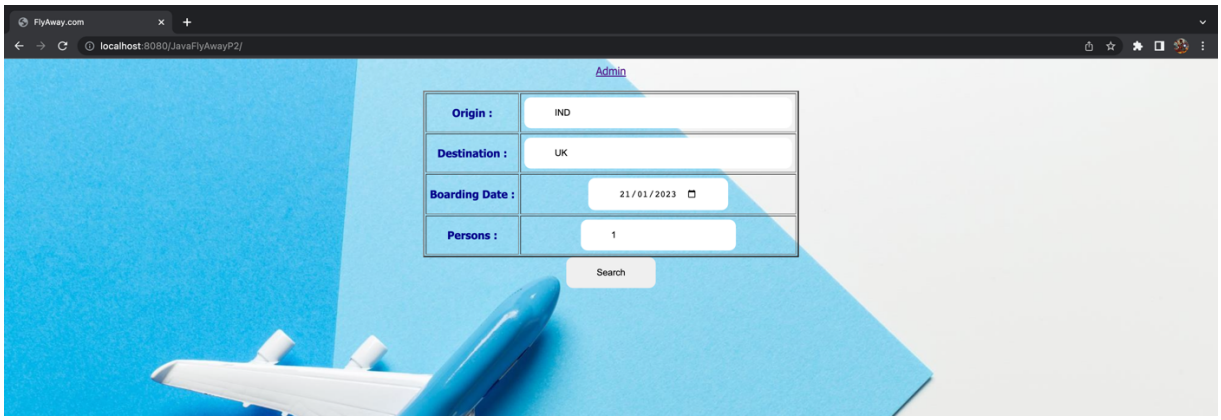


*Note: Once the user saves it after inserting details, it will be inserted into the database.

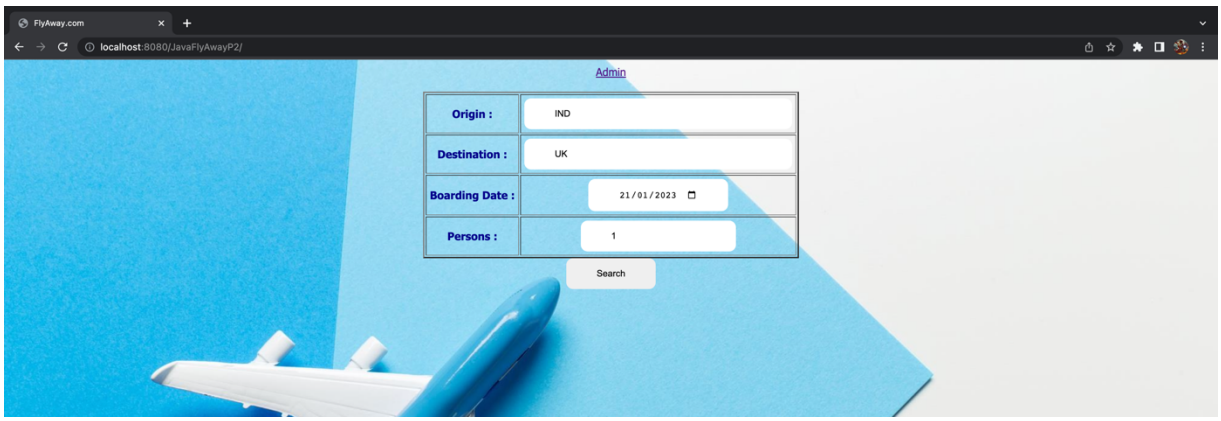


After, Admin can logout back to the homepage.

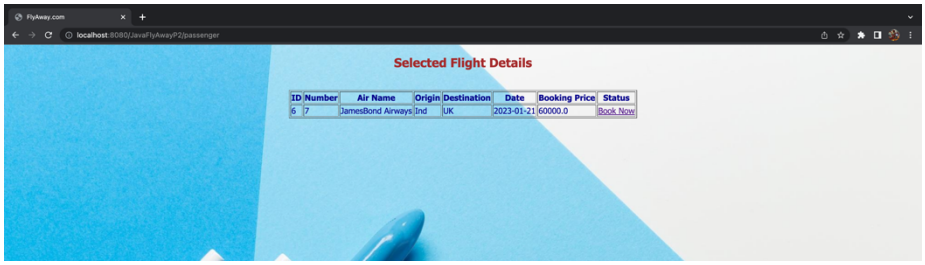
5: Insert the booking details from client side.



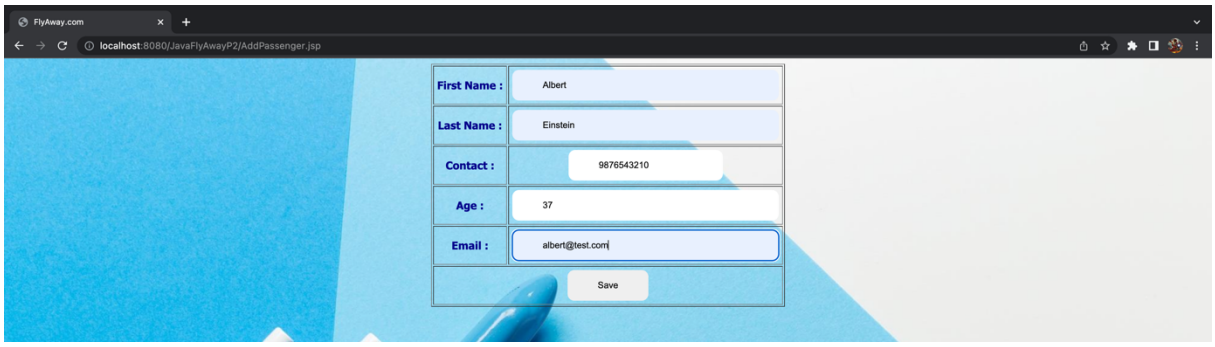
6: After inserting booking details, when user click the “search” button.

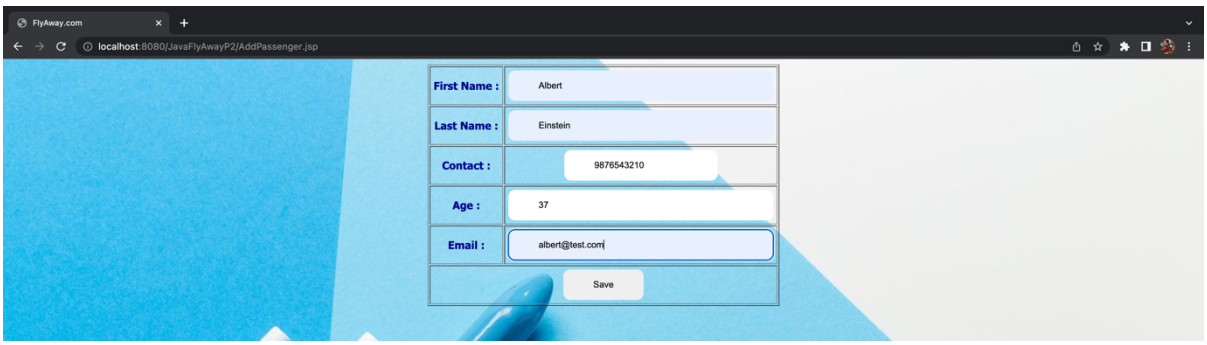


7: After filtering flight details and user pressed the “Book Now” button. It will surf to Passenger Details.

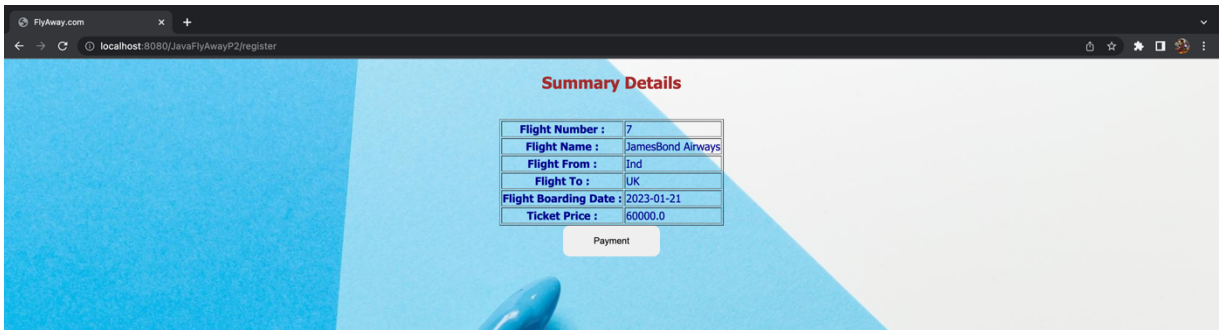


8: Add Passenger Page, Entering Passenger Details

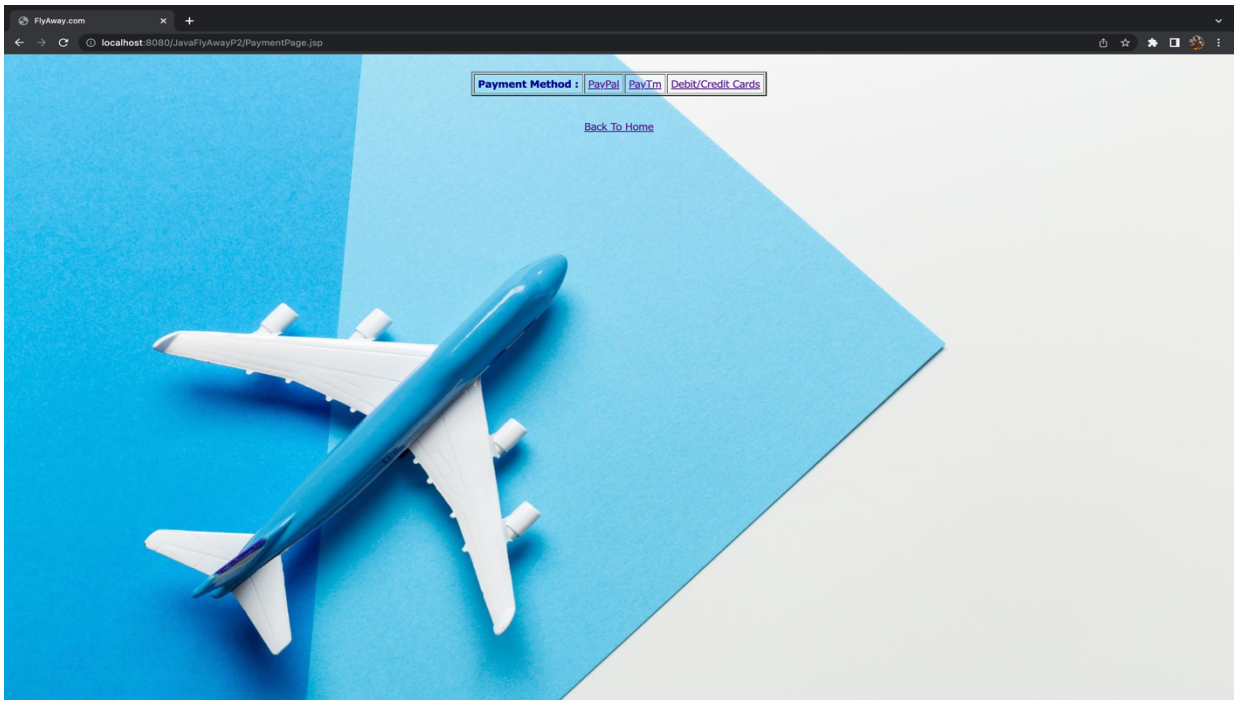




9: Summary Page



10: Payment Page


FlyAway.com
localhost:8080/JavaFlyAwayP2/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)
```

```java
package models;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;

@Entity
@Table(name = "Flight")
public class Flight {

    @Id
    @GeneratedValue(strategy = GenerationType.
IDENTITY)
    @Column(name = "flight_id")
    private int flightId;

    @Column(name = "flight_number")
    private int flightNumber;

    @Column(name = "flight_name")
    private String airline;

    @Column(name = "Source")
    private String origin;

    @Column(name = "Destination")
    private String target;

    @Column(name = "Boarding_Date")
```

```
41      private String dob;
42
43      @Column(name = "Ticket_price")
44      private float price;
45
46      public Flight() {
47      }
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      public void setFlightId(int flightId) {
 87          this.flightId = flightId;
 88      }
 89
 90      public int getFlightNumber() {
 91          return flightNumber;
 92      }
 93
 94      public void setFlightNumber(int flightNumber
    ) {
 95          this.flightNumber = flightNumber;
 96      }
 97
 98      public String getAirline() {
 99          return airline;
100      }
101
102      public void setAirline(String airline) {
103          this.airline = airline;
104      }
105
106      public String getOrigin() {
107          return origin;
108      }
109
110      public void setOrigin(String origin) {
111          this.origin = origin;
112      }
113
114      public String getTarget() {
115          return target;
116      }
```

```
117
118     public void setTarget(String target) {
119         this.target = target;
120     }
121
122     public String getDob() {
123         return dob;
124     }
125
126     public void setDob(String dob) {
127         this.dob = dob;
128     }
129
130     public float getPrice() {
131         return price;
132     }
133
134     public void setPrice(float price) {
135         this.price = price;
136     }
137
138     public List<Passenger> getPassenger() {
139         return passenger;
140     }
141
142     public void setPassenger(List<Passenger>
    passenger) {
143         this.passenger = passenger;
144     }
145
146 }
147
```

```java
package models;

public class Password {

    private static String pwd;

    public Password() {
        pwd = "admin";
    }

    public static String getPwd() {
        return pwd;
    }

    public static void setPwd(String pwd) {
        Password.pwd = pwd;
    }

}

```

```java
package models;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.ManyToMany;

@Entity
@Table(name = "Passenger")
public class Passenger {

    @Id
    @GeneratedValue(strategy = GenerationType.
IDENTITY)
    @Column(name="passenger_id")
    private int pId;

    @Column(name="passenger_fname")
    private String fname;

    @Column(name="passenger_lname")
    private String lname;

    @Column(name="passenger_age")
    private int age;

    @Column(name="passenger_mob")
    private long contact;

    @Column(name="passenger_email")
    private String email;

```

```
41      @ManyToMany(mappedBy = "passenger")//, cascade
     = CascadeType.MERGE)
42      private List<Flight> flight = new ArrayList<
    Flight>();
43
44      public Passenger() {
45      }
46
47      public Passenger(int pId, String fname, String
    lname, int age, long contact, String email) {
48          super();
49          this.pId = pId;
50          this.fname = fname;
51          this.lname = lname;
52          this.age = age;
53          this.contact = contact;
54          this.email = email;
55      }
56
57
58      public Passenger(String fname, String lname,
    int age, long contact, String email) {
59          super();
60          this.fname = fname;
61          this.lname = lname;
62          this.age = age;
63          this.contact = contact;
64          this.email = email;
65      }
66
67      public int getpId() {
68          return pId;
69      }
70
71      public void setpId(int pId) {
72          this.pId = pId;
73      }
74
75      public String getFname() {
76          return fname;
77      }
```

```java

    public void setFname(String fname) {
        this.fname = fname;
    }

    public String getLname() {
        return lname;
    }

    public void setLname(String lname) {
        this.lname = lname;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public long getContact() {
        return contact;
    }

    public void setContact(long contact) {
        this.contact = contact;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public List<Flight> getFlight() {
        return flight;
    }

```

```
119     public void setFlight(List<Flight> flight) {
120         this.flight = flight;
121     }
122 
123 }
124 
```

```java
package Controller;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.SQLException;
import java.util.List;

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

import models.Flight;
import models.Passenger;
import transferobjectaccess.FlightDAO;
import transferobjectaccess.PassengerDAO;

@WebServlet("/")
public class MasterServlet extends HttpServlet {

    private FlightDAO ob;
    private Flight f;
    private PassengerDAO ob1;
    private Passenger p;
    private int num;
    private String pd = null;
    private int count, flag;
    private List<Passenger> list1;

    private static final long serialVersionUID = 1L
;

    public void init() {

        ob = new FlightDAO();
        ob1 = new PassengerDAO();
        pd = "admin";
        count = 0;
        flag = 0;
```

```
41          list1 = null;
42
43      }
44
45      public MasterServlet() {
46          count = 0;
47          list1 = null;
48      }
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             }
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 = ob1.getAllDetails
```

```
115 ();
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             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         Passenger p = ob1.getPassengerById(id);
```

```
144         ob1.deletePassenger(p);
145         response.sendRedirect("booking");
146 
147     }
148 
149     private void EditDetails(HttpServletRequest
    request, HttpServletResponse response)
150             throws SQLException, ServletException
    , IOException {
151 
152         int id = Integer.parseInt(request.
    getParameter("fid"));
153         Flight f = ob.getFlightById(id);
154         RequestDispatcher dispatcher = request.
    getRequestDispatcher("AddFlight.jsp");
155         request.setAttribute("f", f);
156         dispatcher.forward(request, response);
157 
158     }
159 
160     private void updateDetails(HttpServletRequest
    request, HttpServletResponse response)
161             throws SQLException, IOException {
162 
163         int fid = Integer.parseInt(request.
    getParameter("fid"));
164         int fnumber = Integer.parseInt(request.
    getParameter("fnumber"));
165         String fname = request.getParameter("fname
    ");
166         String forigin = request.getParameter("
    forigin");
167         String ftarget = request.getParameter("
    ftarget");
168         String date = request.getParameter("fdate"
    );
169         float fprice = Float.parseFloat(request.
    getParameter("fprice"));
170         Flight fl = new Flight(fid, fnumber, fname
    , forigin, ftarget, date, fprice);
171         ob.updateFlight(fl);
```

```
172         response.sendRedirect("view");
173 
174     }
175 
176     private void deleteDetails(HttpServletRequest
    req, HttpServletResponse resp) throws IOException
     {
177 
178         int id = Integer.parseInt(req.getParameter
    ("fid"));
179         Flight f = ob.getFlightById(id);
180         ob.deleteFlight(f);
181         resp.sendRedirect("view");
182 
183     }
184 
185     private void showAllDetails(
    HttpServletResponse response, HttpServletRequest
    request)
186             throws ServletException, IOException {
187 
188         List<Flight> list = ob.getAllDetails();
189         System.out.println(list);
190         request.setAttribute("list", list);
191         RequestDispatcher rd = request.
    getRequestDispatcher("FlightDetails.jsp");
192         rd.forward(request, response);
193 
194     }
195 
196     private void showNewForm(HttpServletRequest
    request, HttpServletResponse response)
197             throws ServletException, IOException {
198 
199         RequestDispatcher rd = request.
    getRequestDispatcher("AddFlight.jsp");
200         rd.forward(request, response);
201 
202     }
203 
204     private void insertDetails(HttpServletRequest
```

```
204 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 date = request.getParameter("fdate"
    );
211         float fprice = Float.parseFloat(request.
    getParameter("fprice"));
212         Flight fl = new Flight(fnumber, fname,
    forigin, ftarget, date, fprice);
213         ob.insertFlightInDB(fl);
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
```

```
229 );
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         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
```

```
262 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");
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 }
307
```

```java
package Configuration;

import java.util.Properties;

import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.
StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.cfg.Environment;
import org.hibernate.service.ServiceRegistry;

import models.Flight;
import models.Passenger;

public class HibernateConfig {

    private static SessionFactory sFactory;

    public static SessionFactory getSessionFactory
() {
        if(sFactory == null) {
            try {
                Configuration cfg = new
Configuration();

                Properties settings = new
Properties();
                settings.put(Environment.DRIVER, "
com.mysql.cj.jdbc.Driver");
                settings.put(Environment.URL, "jdbc
:mysql://localhost:3306/demo?useSSL=false");
                settings.put(Environment.USER, "
root");
                settings.put(Environment.PASS, "
cisco");
                settings.put(Environment.DIALECT, "
org.hibernate.dialect.MySQL5Dialect");
                settings.put(Environment.SHOW_SQL,
"true");
                settings.put(Environment.
CURRENT_SESSION_CONTEXT_CLASS, "thread");
```

```
31                  //settings.put(Environment.
   HBM2DDL_AUTO, "update");
32                  settings.put(Environment.
   HBM2DDL_AUTO, "create");
33
34                  cfg.setProperties(settings);
35                  cfg.addAnnotatedClass(Flight.class
   );
36                  cfg.addAnnotatedClass(Passenger.
   class);
37                  ServiceRegistry servReg = new
   StandardServiceRegistryBuilder()
38                          .applySettings(cfg.
   getProperties()).build();
39                  sFactory = cfg.buildSessionFactory(
   servReg);
40              } catch (Exception e) {
41                  e.printStackTrace();
42              }
43          }
44          return sFactory;
45      }
46 }
47
```

```java
package transferobjectaccess;

import java.util.List;

import org.hibernate.Session;
import org.hibernate.Transaction;

import Configuration.HibernateConfig;
import models.Flight;
import models.Passenger;

public class FlightDAO {

    public void insertFlightInDB(Flight f)  {

        Transaction transaction = null;
        Session dbSession = null;
        try {
            dbSession = HibernateConfig.
  getSessionFactory().openSession();
            transaction = dbSession.
  beginTransaction();
            dbSession.save(f);
            transaction.commit();
        }catch (Exception e) {
            if(transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }finally {
            dbSession.close();
        }

    }

    @SuppressWarnings("unchecked")
    public  List<Flight> getAllDetails()    {

        Session dbSession = null;
        List<Flight> list= null;
        try {
```

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

```
 76          Transaction transaction = null;
 77          Session dbSession = null;
 78          try {
 79              dbSession = HibernateConfig.
    getSessionFactory().openSession();
 80              transaction = dbSession.
    beginTransaction();
 81              dbSession.update(f);
 82              transaction.commit();
 83          }catch (Exception e) {
 84              if(transaction != null) {
 85                  transaction.rollback();
 86              }
 87              e.printStackTrace();
 88          }finally {
 89              dbSession.close();
 90          }
 91          return f;
 92
 93      }
 94
 95      public Flight deleteFlight(Flight f) {
 96
 97          Transaction transaction = null;
 98          Session dbSession = null;
 99          try {
100              dbSession = HibernateConfig.
    getSessionFactory().openSession();
101              transaction = dbSession.
    beginTransaction();
102              dbSession.delete(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
116     public void relation(Flight f, List<Passenger
    > list) {
117
118         Transaction transaction = null;
119         Session dbSession = null;
120         try {
121             dbSession = HibernateConfig.
    getSessionFactory().openSession();
122             transaction = dbSession.
    beginTransaction();
123             f.setPassenger(list);
124             dbSession.save(f);
125             transaction.commit();
126         }catch (Exception e) {
127             if(transaction != null) {
128                 transaction.rollback();
129             }
130             e.printStackTrace();
131         }finally {
132             dbSession.close();
133         }
134
135     }
136
137 }
138
```

```java
package transferobjectaccess;

import java.util.List;

import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.query.Query;

import Configuration.HibernateConfig;
import models.Flight;
import models.Passenger;

public class PassengerDAO {

    @SuppressWarnings("unchecked")
    public List<Flight> getAllDetailsByOriginDate(String origin, String date, String target) {

        Transaction transaction = null;
        Session dbSession = null;
        List<Flight> list = null;
        try {
            dbSession = HibernateConfig.getSessionFactory().openSession();
            transaction = dbSession.beginTransaction();
            @SuppressWarnings("rawtypes")
            Query query = dbSession
                    .createQuery("from Flight f where f.origin = :origin and f.target = :target and f.dob = :date");
            query.setParameter("origin", origin);
            query.setParameter("target", target);
            query.setParameter("date", date);
            list = query.list();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        } finally {
```

```
37              dbSession.close();
38          }
39          return list;
40
41      }
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          return list;
 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 }
124 
```

```
 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 class="container" align="center">
11         <a href="AdminLogin.jsp">Admin</a> <br>
12         <br>
13         <form action="passenger" method="post">
14             <table border="2" cellpadding="5"
   postion="bottom">
15                 <tr>
16                     <th>Origin :</th>
17                     <td><input type="text" name="
   origin" size="45" required>
18                     </td>
19                 </tr>
20                 <tr>
21                     <th>Destination :</th>
22                     <td><input type="text" name="
   target" size="45" required>
23                     </td>
24                 </tr>
25                 <tr>
26                     <th>Boarding Date :</th>
27                     <td><input type="date" name="
   date" size="45" required>
28                     </td>
29                 </tr>
30                 <tr>
31                     <th>Persons :</th>
32                     <td><input type="number" name="
   qty" size="45" required>
33                     </td>
34                 </tr>
35             </table>
```

```
36              <input type="submit" value="Search">
37          </form>
38      </div>
39  </body>
40  </html>
```

File - /Users/rajgauta/Desktop/RKG/JavaFSD_P2/src/main/webapp/AddFlight.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         <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 <%@ 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             
13         <br><br>
14         <form action="reset" method="post">
15             <table border="2" cellpadding="5"
   postion="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>
```

```
 1 <%@ page language="java" contentType="text/html;
   charset=ISO-8859-1"
 2     pageEncoding="ISO-8859-1" %>
 3
 4
 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         <h2>Admin Login</h2>
15             
16         <br><br>
17         <form action="login" method="post">
18             <table border="2" cellpadding="5"
   postion="bottom">
19                 <tr>
20                     <th>Email :</th>
21                     <td><input type="email" name="
   email" size="45" required>
22                     </td>
23                 </tr>
24                 <tr>
25                     <th>Password :</th>
26                     <td><input type="password" name
   ="pwd" size="45" required>
27                     </td>
28                 </tr>
29             </table>
30             <input type="submit" value="Login">
31         </form>
32     </div>
33 </body>
34 </html>
```

```
 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        
15         <table border="2" cellpadding="5" postion="
   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><br>
24         <a href="HomePage.jsp">Back To Home</a>
25     </div>
26 </body>
27 </html>
```

```
 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         <h2>Summary Details</h2>
15             
16         <form action="PaymentPage.jsp" method="post
   ">
17             <table border="1">
18                     <tr>
19                         <th>Flight Number :</th>
20                         <td><c:out value="${f.
   flightNumber}" /></td>
21                     </tr>
22                     <tr>
23                         <th>Flight Name :</th>
24                         <td><c:out value="${f.
   airline}" /></td>
25                     </tr>
26                     <tr>
27                         <th>Flight From :</th>
28                         <td><c:out value="${f.
   origin}" /></td>
29                     </tr>
30                     <tr>
31                         <th>Flight To :</th>
32                         <td><c:out value="${f.
   target}" /></td>
33                     </tr>
34                     <tr>
```

```
35                          <th>Flight Boarding Date
    :</th>
36                          <td><c:out value="${f.dob}"
    /></td>
37                      </tr>
38                      <tr>
39                          <th>Ticket Price :</th>
40                          <td><c:out value="${f.price
    * n}" /></td>
41                      </tr>
42              </table>
43              <input type="submit" value="Payment">
44          </form>
45      </div>
46 </body>
47 </html>
```

```
 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         <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='${p.lname
   }' />" required /></td>
34              </tr>
35              <tr>
36                  <th>Contact :</th>
37                  <td><input type="number" name="
   contact" size="45"
38                      value="<c:out value='${p.
   contact}' />" required /></td>
39              </tr>
40              <tr>
41                  <th>Age :</th>
42                  <td><input type="text" name="age"
   size="45"
43                      value="<c:out value='${p.age
   }' />" required /></td>
44              </tr>
45              <tr>
46                  <th>Email :</th>
47                  <td><input type="email" name="email
   " size="45"
48                      value="<c:out value='${p.email
   }' />" required /></td>
49              </tr>
50              <tr>
51                  <td colspan="2" align="center"><
   input type="submit"
52                      value="Save" /></td>
53              </tr>
54          </table>
55          </form>
56      </div>
57 </body>
58 </html>
```

File - /Users/rajgauta/Desktop/RKG/JavaFSD_P2/src/main/webapp/RegisterPage.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     <h1 align="center">Passenger Details</h1>
14     <h2 align="center">
15         <a href="AddPassenger.jsp">Add Passenger</a
   >    
16     </h2>
17     <div align="center">
18         <table border="1">
19             <caption>Passenger Details</caption>
20             <tr>
21                 <th>ID</th>
22                 <th>First Name</th>
23                 <th>Last Name</th>
24                 <th>Contact</th>
25                 <th>Age</th>
26                 <th>Email</th>
27             </tr>
28             <c:forEach var="p" items="${list}">
29                 <tr>
30                     <td><c:out value="${p.pId}"
    /></td>
31                     <td><c:out value="${p.fname}"
    /></td>
32                     <td><c:out value="${p.lname}"
    /></td>
33                     <td><c:out value="${p.contact}"
    /></td>
34                     <td><c:out value="${p.age}"
```

```
34  /></td>
35                      <td><c:out value="${p.email}"
    /></td>
36                      <td><a href="deletePassenger?id
   =<c:out value='${p.pId}' />">Delete</a></td>
37                  </tr>
38              </c:forEach>
39          </table>
40          <a href="register">Confirm</a>
41      </div>
42 </body>
43 </html>
```

File - /Users/rajgauta/Desktop/RKG/JavaFSD_P2/src/main/webapp/FlightDetails.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     <h1 align="center">Flight Management</h1>
14     <h2 align="center">
15         <a href="add">Add Flight</a>
16            
17         <a href="ResetPage.jsp">Change Password</a>
18            
19         <a href="HomePage.jsp">Logout</a>
20     </h2>
21     <div align="center">
22         <table border="1">
23             <caption>
24                 Flight Details
25             </caption>
26
27             <tr>
28                 <th>ID</th>
29                 <th>Number</th>
30                 <th>Air Name</th>
31                 <th>Origin</th>
32                 <th>Destination</th>
33                 <th>Date</th>
34                 <th>Booking Price</th>
35                 <th>Status</th>
36             </tr>
37             <c:forEach var="f" items="${list}">
38                 <tr>
```

```
39                      <td><c:out value="${f.flightId
   }" /></td>
40                      <td><c:out value="${f.
   flightNumber}" /></td>
41                      <td><c:out value="${f.airline}"
    /></td>
42                      <td><c:out value="${f.origin}"
    /></td>
43                      <td><c:out value="${f.target}"
    /></td>
44                      <td><c:out value="${f.dob}"
    /></td>
45                      <td><c:out value="${f.price}"
    /></td>
46                      <td>
47                       <a href="edit?fid=<c:out value
   ='${f.flightId}' />">Edit</a>
48                           
49                       <a href="delete?fid=<c:out
   value='${f.flightId}' />">Delete</a
   >
50                      </td>
51                  </tr>
52              </c:forEach>
53          </table>
54      </div>
55 </body>
56 </html>
```

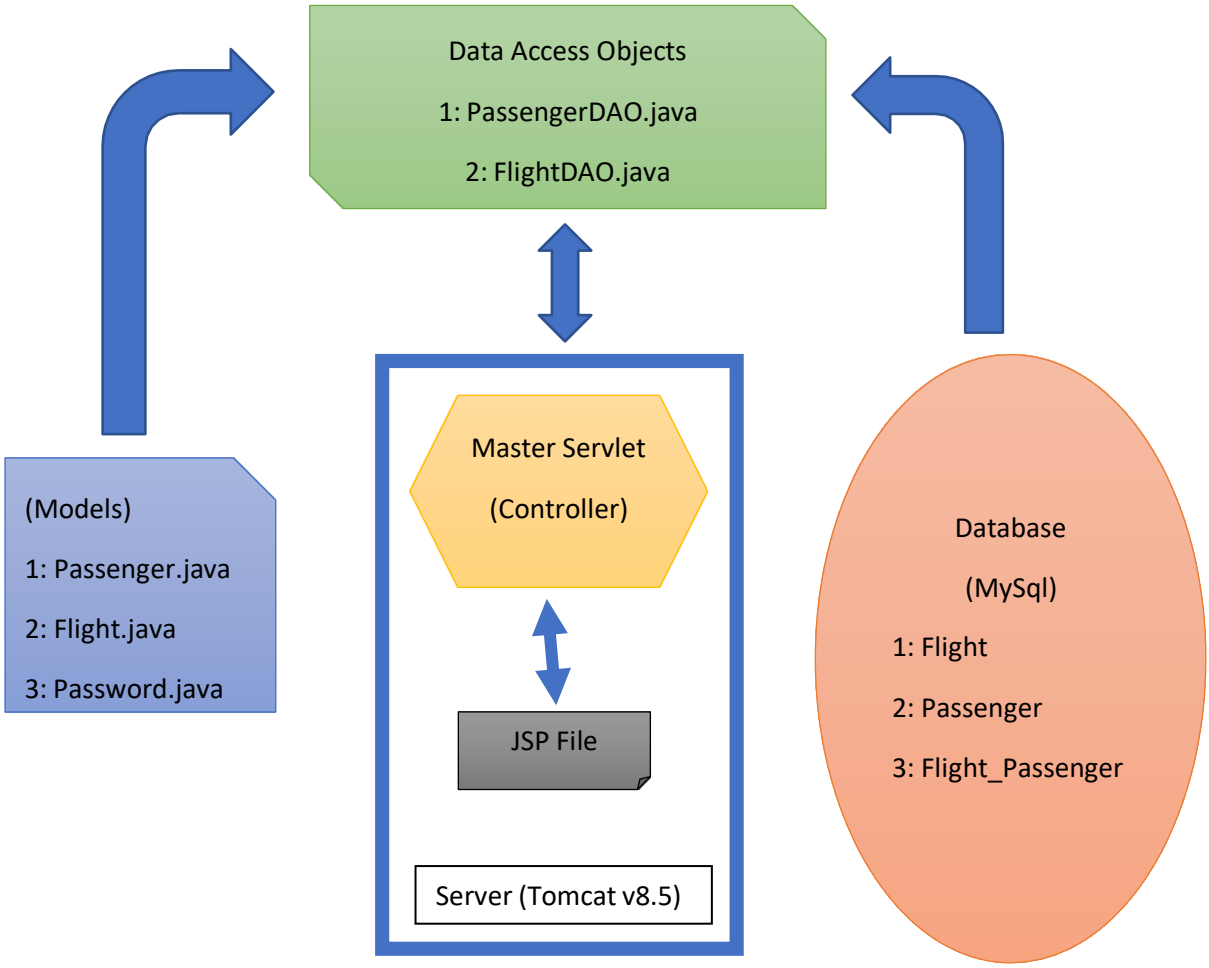
File - /Users/rajgauta/Desktop/RKG/JavaFSD_P2/src/main/webapp/PassengerFlights.jsp

```
 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
 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             
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}"
```

```
34  /></td>
35                      <td><c:out value="${f.dob}"
    /></td>
36                      <td><c:out value="${f.price}"
    /></td>
37                      <td><a href="find?fid=<c:out
   value='${f.flightId}'/>">Book Now</a></td>
38                  </tr>
39              </c:forEach>
40          </table>
41      </div>
42 </body>
43 </html>
```

```
 1 <!DOCTYPE web-app PUBLIC
 2  "-//Sun Microsystems, Inc.//DTD Web Application 2.
   3//EN"
 3  "http://java.sun.com/dtd/web-app_2_3.dtd" >
 4
 5 <web-app>
 6   <display-name>Archetype Created Web Application</
   display-name>
 7   <servlet>
 8     <servlet-name>MasterServlet</servlet-name>
 9     <display-name>MasterServlet</display-name>
10     <description></description>
11     <servlet-class>Controller.MasterServlet</
   servlet-class>
12   </servlet>
13   <servlet-mapping>
14     <servlet-name>MasterServlet</servlet-name>
15     <url-pattern>/MasterServlet</url-pattern>
16   </servlet-mapping>
17   <welcome-file-list>
18     <welcome-file>HomePage.jsp</welcome-file>
19   </welcome-file-list>
20 </web-app>
21
```

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.