

Amartya Choudhury

JU-BCSE IV

Roll – 001610501026

Internet Technology Lab

amartyachowdhury98@gmail.com

Abstract

In the digital age , every bit of information is available to us on the internet ,ranging from entertainment timings like movie timings ,sports timings to work like flight and train schedules .This is a humble attempt a building a flight schedule website using Servlets and Jsp. HTML was also used for front end development.

AIRLINE TRAVEL SITE

An airline travel web application using servlet and jsp

Problem Statement:

Implement a web application for “Travel Thru Air” using servlets to support the following two use cases

1. A list of current special deals must appear on the home page. Each special deal must display the departure city, the arrival city, and the cost. These special deals are set up by the marketing department and change during the day, so it can’t be static. Special deals are only good for a limited amount of time.

2. A user may search for flights, given a departure city, time and an arrival city. The results must display the departure city, the arrival city, the total cost, and how many legs the flight will have.

State and explain why and where you have used design patterns.

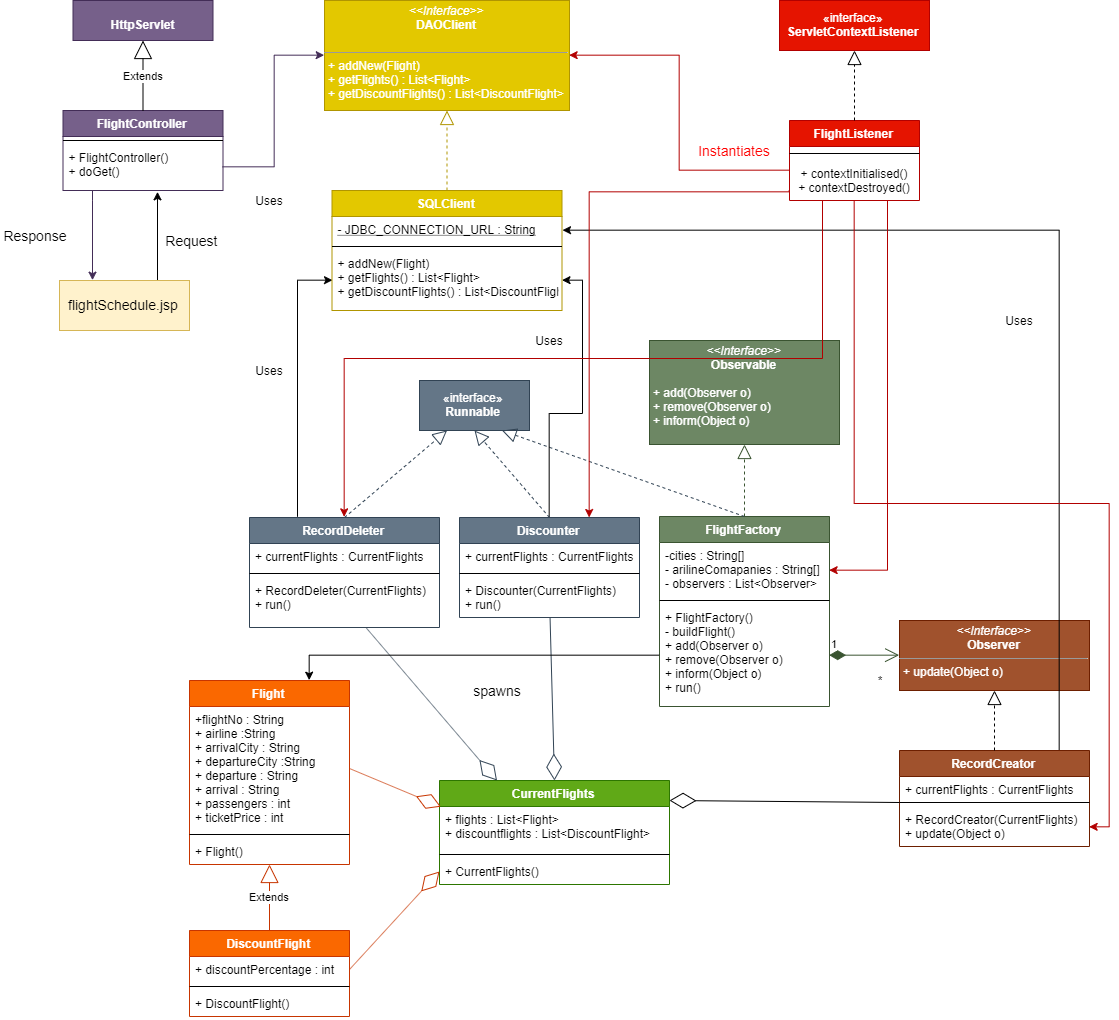
Design:

Fig 1 : Class Diagram

* The MVC design pattern is followed .
* flightSchedule.jsp is the view , FlightController.java is the controller and NormalFlights and DiscountFlights is the model to be displayed .
* User submits form given the departure , arrival ,date and airlines company .The controller handles this request and fetches NormalFlight records and discounted flightrecords with the help of the dao (SQLClient) .
* FlightListener implements Servlet context listener and its task is to –

1. Instatiation of Dao
2. Spawning of FlightFactory thread (Observable) and adding an instance of RecordCreator to it ( Observer).
3. Spawing a RecordDeleter thread
4. Spawning a Discounter thread
5. The RecordCreator , RecordDeleter and the Discounter thread all share an instance of CurrentFlight , which holds all non-discounted and discounted-flights that have not yet started its journey .

* FlightFactory thread creates flights at regular intervals of time ( 3 seconds) . On flight creation , the update method of a RecordCreator is called , which adds the record to a database and to currentflights as normal flights .
* RecordDeleter scans the list of of normal and discounted currentflights and deletes the flight Records that have already started .The database is also updated .
* Discounter is a thread that runs at 5 minutes interval and discounts flights with a probability of 0.5 .The discounted flights are taken from flights lists and put into discount flight list .The database is updated with this information.

Design Patterns :

* The SQLClientDao has been made a Singleton , since creating a new connection everytime is extremely expensive .
* The RecordCreator “observes” the FlightFactory for new FlightRecords and then stores the flightRecord in database . So , the observer pattern is used here.
* For simulating real flight system , we need to create dummy flight records at regular intervals of time . So we created a FlightFactory whose job is to create flights after a time interval . The factory method was employe here .
* Dependency injection was used .The FlightController and other threads do not know which instance of Dao they are using .That has been configured in the flightListener . Also a FlightFactory (an Observable) does not know what are its Observers .It updates them after new flight is created nonetheless.

View :

flightSchedule.jsp

|  |
| --- |
| **<%@** page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"**%>**  **<%@**page import="java.util.List" **%>**  **<%@**page **import** ="**com.achoudhury.flightmanagement.\***" **%>**  <!DOCTYPE html>  <html>  <head>  <meta charset="ISO-8859-1">  <title>TraveThruAir</title>  </head>  <body>  <h1>Welcome to Travel Thru Air !</h1>  <h3>Search for Flights :</h3>  <form action="FlightController" method="get">    <!--<input name="departure" "type="text" placeholder="City">    <input name="arrival" "type="text" placeholder="City">-->  <label>Flying From :</label>  <select name="departure">  <option value="Delhi">Delhi</option>  <option value="Kolkata">Kolkata</option>  <option value="Bangalore">Bangalore</option>  </select>  <label>Flying To:</label>  <select name="arrival">  <option value="Delhi">Delhi</option>  <option value="Kolkata">Kolkata</option>  <option value="Bangalore">Bangalore</option>  </select>  <label>Airlines:</label>  <select name="airlines">  <option value="Any">Any</option>  <option value="Indigo">Indigo</option>  <option value="AirAsia">AirAsia</option>  <option value="Air India">Air India</option>  <option value="GoAir">GoAir</option>  <option value="Vistara">Vistara</option>  </select>  <label>Departing:</label>  <input name="date" type="date">  <button>Search</button>  </form>  <h2>Discounts</h2>  <table border="1">  <tr>  <th>Flight Number</th>  <th>Airline</th>  <th>Departing</th>  <th>Arrival</th>  <th>TicketPrice(Rs.)</th>  <th>Passengers</th>  </tr>  **<%**  **if**(request.getAttribute("discounts") != **null**){  List<DiscountFlight> discountedFlights= (List<DiscountFlight>)request.getAttribute("discounts");  **for**(DiscountFlight **df:** discountedFlights){  out.print("<tr>");  out.print("<td>");  out.print(df.flightNo);  out.print("</td>");  out.print("<td>");  out.print(df.airLine);  out.print("</td>");  out.print("<td>");  out.print(df.departure.toString());  out.print("</td>");  out.print("<td>");  out.print(df.arrival.toString());  out.print("</td>");  out.print("<td>");  out.print("<strike>"+Integer.toString(df.ticketPrice)+"</strike>");  out.print(df.ticketPrice\*(**100**-df.discountPercentage)/**100**);  out.print("</td>");  out.print("<td>");  out.print(df.passengers);  out.print("</td>");  out.print("</tr>");  }  }  **%>**  </table>    <h2>Normal Rate</h2>  <table border="1">  <tr>  <tr>  <th>Flight Number</th>  <th>Airline</th>  <th>Departing</th>  <th>Arrival</th>  <th>TicketPrice(Rs.)</th>  <th>Passengers</th>  </tr>  **<%**  **if**(request.getAttribute("normals") != **null**){  List<Flight> flights= (List<Flight>)request.getAttribute("normals");  **for**(Flight **f:** flights){  out.print("<tr>");  out.print("<td>");  out.print(f.flightNo);  out.print("</td>");  out.print("<td>");  out.print(f.airLine);  out.print("</td>");  out.print("<td>");  out.print(f.departure.toString());  out.print("</td>");  out.print("<td>");  out.print(f.arrival.toString());  out.print("</td>");  out.print("<td>");  out.print(f.ticketPrice);  out.print("</td>");  out.print("<td>");  out.print(f.passengers);  out.print("</td>");  out.print("</tr>");  }  }  **%>** </table>  </body>  </html> |

Flight.java

|  |
| --- |
| **package** com.achoudhury.flightmanagement;  **import** **java.util.Date**;  **public** **class** **Flight** {  **public** String flightNo;  **public** String airLine;  **public** String departureCity;  **public** String arrivalCity;  **public** Date departure;  **public** Date arrival;  **public** **int** passengers;  **public** **int** ticketPrice;  Flight(String flightNo,String airLine,String departureCity,String arrivalCity,Date departure,Date arrival,**int** passengers,**int** ticketPrice){  **this**.flightNo = flightNo;  **this**.airLine =airLine;  **this**.departureCity = departureCity;  **this**.arrivalCity = arrivalCity;  **this**.departure = departure;  **this**.arrival = arrival;  **this**.passengers = passengers;  **this**.ticketPrice = ticketPrice;  }  } |

DiscountFlight.java

|  |
| --- |
| **package** com.achoudhury.flightmanagement;  **import** **java.util.Date**;  **public** **class** **DiscountFlight** **extends** Flight {  **public** **int** discountPercentage;  DiscountFlight(String flightNo, String airLine, String departureCity, String arrivalCity, Date departure,  Date arrival, **int** passengers,**int** ticketPrice,**int** discountPercentage) {  **super**(flightNo, airLine, departureCity, arrivalCity, departure, arrival,passengers, ticketPrice);  **this**.discountPercentage = discountPercentage;  }  } |

FlightFactory.java

|  |
| --- |
| **package** com.achoudhury.flightmanagement;  **import** **java.util.Date**;  **import** **java.util.Calendar**;  **import** **java.util.List**;  **import** **java.util.ArrayList**;  **import** **java.lang.Math**;  **public** **class** **FlightFactory** **implements** Runnable,Observable{  String[] cities = {  "Kolkata","Delhi","Bangalore"} ;  String[] airlinesCompanies = {  "Indigo","Air India","Spicejet","AirAsia","Vistara","GoAir"  };  List<Observer> observers;    FlightFactory(){  }  **private** Flight **buildFlight**(){  Flight flight = **null**;  **try** {  **int** index1 = (**int**)( Math.random()\***100** ) % cities.length;    String temp = cities[index1];  cities[index1] = cities[cities.length-**1**];  cities[cities.length -**1**] = temp;  **int** index2 = (**int**)( Math.random()\***100** ) % (cities.length -**1**);  String departureCity = cities[cities.length -**1** ];  String arrivalCity = cities[index2];  String airline = airlinesCompanies[(**int**)(Math.random()\***100**)%airlinesCompanies.length];  Date timeNow = **new** Date();  Calendar c = Calendar.getInstance();  c.setTime(timeNow);  c.add(Calendar.DATE,**1** +(**int**)(Math.random()\***100**)%**7**);  Date departure = c.getTime();  System.out.println(departure);  c.add(Calendar.HOUR,(**int**)( Math.random()\***100** ) % **24**);  c.add(Calendar.MINUTE,(**int**)( Math.random()\***100** ) % **24**);  Date arrival = c.getTime();  **int** passengers = **1** + (**int**)(Math.random()\***1000**) % **256**;  **int** cost = **2000** + (**int**)(Math.random()\***10000**) % **8000**;  String flightNo ="";  flightNo += (**char**)((**int**)(Math.random()\***26**)+**65**);  flightNo += (**char**)((**int**)(Math.random()\***26**)+**65**);  flightNo += " ";  flightNo += Integer.toString((**int**)(**1** + Math.random()\***1000**));  flight = **new** Flight(flightNo,airline,departureCity,arrivalCity,departure,arrival,passengers,cost);  }  **catch**(Exception e) {  e.printStackTrace();  }  **return** flight;  }  **@Override**  **public** **void** **add**(Observer o) {  **if**(observers == **null**) observers = **new** ArrayList<Observer>();  observers.add(o);  }  **@Override**  **public** **void** **remove**(Observer o) {  observers.remove(o);  }  **@Override**  **public** **void** **inform**(Object obj) {  **for**(Observer obs : observers) {  obs.update(obj);  }  }  **@Override**  **public** **void** **run**() {  **while**(**true**) {  Flight flight = buildFlight();  inform(flight);  **try** {  Thread.sleep(**3**\***1000**);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  }    } |

FlightController.java

|  |
| --- |
| package com.achoudhury.flightmanagement;  import java.io.IOException;  import java.sql.SQLException;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import java.util.Date;  import java.util.List;  import java.util.ArrayList;  import java.text.ParseException;  import java.text.SimpleDateFormat ;  @WebServlet("/FlightController")  public class FlightController extends HttpServlet {  private static final long serialVersionUID = 1L;  public FlightController() {  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String departureCity = request.getParameter("departure");  String arrivalCity = request.getParameter("arrival");  String date = request.getParameter("date");  String airlines = request.getParameter("airlines");  SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");  try {  SQLClient sqlClient = SQLClient.getInstance();  Date dt = formatter.parse(date);  List<Flight> normalFlights = sqlClient.getnormalFlights(departureCity,arrivalCity,dt,airlines);  List<DiscountFlight> discountFlights = sqlClient.getdiscountFlights(departureCity,arrivalCity,dt,airlines);  request.setAttribute("normals",normalFlights);  request.setAttribute("discounts",discountFlights);  request.getRequestDispatcher("/flightSchedule.jsp").forward(request, response);  }  catch (SQLException e1) {  e1.printStackTrace();  }  catch (ParseException e) {  e.printStackTrace();  }  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  } |

CurrentFlights.java

|  |
| --- |
| **package** com.achoudhury.flightmanagement;  **import** **java.util.ArrayList**;  **import** **java.util.List**;  **import** **java.util.Vector**;  **public** **class** **CurrentFlights** {  **public** List<Flight> flights;  **public** List<DiscountFlight> discountflights;  CurrentFlights(){  flights = **new** ArrayList<>();  discountflights = **new** ArrayList<>();  }  } |

Observable.java

|  |
| --- |
| **package** com.achoudhury.flightmanagement;  **public** **interface** **Observable** {  **public** **void** **add**(Observer o);  **public** **void** **remove**(Observer o);  **public** **void** **inform**(Object o);  } |

Observable.java

|  |
| --- |
| **package** com.achoudhury.flightmanagement;  **public** **interface** **Observer** {  **public** **void** **update**(Object o);  } |

RecordCreator.java

|  |
| --- |
| **package** com.achoudhury.flightmanagement;  **import** **java.sql.SQLException**;  **public** **class** **RecordCreator** **implements** Runnable,Observer{  CurrentFlights currentflights;  RecordCreator(CurrentFlights currentflights){  **this**.currentflights = currentflights;  }  **public** **void** **update**(Object o) {  Flight f = (Flight)o;  **synchronized**(currentflights) {  currentflights.flights.add(f);  **try** {  SQLClient sqlClient = SQLClient.getInstance();  sqlClient.addnew(f);  System.out.println("added flight " + f.flightNo);  }  **catch** (SQLException e) {  e.printStackTrace();  }  }  }  **@Override**  **public** **void** **run**() {    }  } |

RecordDeleter.java

|  |
| --- |
| **package** com.achoudhury.flightmanagement;  **import** **java.sql.SQLException**;  **import** **java.util.Date**;  **import** **java.util.Iterator**;  **public** **class** **RecordDeleter** **implements** Runnable{  CurrentFlights currentFlights;  RecordDeleter(CurrentFlights currentflights){  **this**.currentFlights = currentflights;  }  **@Override**  **public** **void** **run**() {  **try**{  SQLClient sqlclient= SQLClient.getInstance();  sqlclient.removeCompletedFlights(**new** Date());  }  **catch**(SQLException e) {  e.printStackTrace();  }  **while**(**true**) {  **try** {  SQLClient sqlClient = SQLClient.getInstance();    Date timeNow = **new** Date();  **synchronized**(currentFlights){  **for**(Iterator<Flight> itr = currentFlights.flights.iterator();itr.hasNext();) {  Flight flight = itr.next();  **if**(timeNow.after(flight.departure)) {  sqlClient.removeflight(flight.flightNo);  itr.remove();  System.out.println("Deleted record for flight" + flight.flightNo);  }  }  **for**(Iterator<DiscountFlight> itr = currentFlights.discountflights.iterator();itr.hasNext();) {  DiscountFlight df = itr.next();  **if**(timeNow.after(df.departure)){  sqlClient.removeflight(df.flightNo);  itr.remove();  System.out.println("Deleted record for flight" + df.flightNo);  }  }  }  }  **catch** (SQLException e) {  e.printStackTrace();  }  }    }  } |

Discounter.java

|  |
| --- |
| **package** com.achoudhury.flightmanagement;  **import** **java.sql.SQLException**;  **import** **java.util.Iterator**;  **import** **java.lang.Math**;  **public** **class** **Discounter** **implements** Runnable{  CurrentFlights currentflights;  Discounter(CurrentFlights currentflights){  **this**.currentflights = currentflights;  }  **@Override**  **public** **void** **run**() {  **while**(**true**) {  **try** {  SQLClient sqlClient = SQLClient.getInstance();  **synchronized**(currentflights) {  **for**(Iterator<Flight> itr = currentflights.flights.iterator();itr.hasNext();) {  **int** random = (**int**)(Math.random()\***100**);  Flight flight = itr.next();  **if**(random < **50**) {  itr.remove();  DiscountFlight df = **new** DiscountFlight(flight.flightNo,flight.airLine,flight.departureCity,flight.arrivalCity,flight.departure,flight.arrival,flight.passengers,flight.ticketPrice,(**int**)(Math.random()\***100**)+**1**);  currentflights.discountflights.add(df);  sqlClient.discount(df.flightNo,df.discountPercentage);  System.out.println("discounted flight " + df.flightNo);  }  }  }  Thread.sleep(**5**\***60**\***1000**);  }  **catch** (SQLException e) {  e.printStackTrace();  }  **catch** (InterruptedException e) {  e.printStackTrace();  }  }    }  } |

FlightListener.java

|  |
| --- |
| **@WebListener**  **public** **class** **FlightListener** **implements** ServletContextListener {  /\*\*  \* Default constructor.  \*/  **public** **FlightListener**() {  // TODO Auto-generated constructor stub  }  **public** **void** **contextDestroyed**(ServletContextEvent arg0) {  // TODO Auto-generated method stub  }  /\*\*  \* @see ServletContextListener#contextInitialized(ServletContextEvent)  \*/  **public** **void** **contextInitialized**(ServletContextEvent arg0) {  // TODO Auto-generated method stub  CurrentFlights currentflights = **new** CurrentFlights();  FlightFactory factory = **new** FlightFactory();  RecordCreator rc = **new** RecordCreator(currentflights);  factory.add(rc);  Thread flightProduction = **new** Thread(factory);  flightProduction.start();  Thread recordDeleter = **new** Thread(**new** RecordDeleter(currentflights));  recordDeleter.start();      Thread recordCreator = **new** Thread(rc);  recordCreator.start();  Thread discounter = **new** Thread(**new** Discounter(currentflights));  discounter.start();  System.out.println("all threads started");    }    } |

SQLClient.java

|  |
| --- |
| **package** com.achoudhury.flightmanagement;  **import** **java.sql.\***;  **import** **java.util.Date**;  **import** **java.util.List**;  **import** **java.util.ArrayList**;  **public** **class** **SQLClient** {  **private** **static** SQLClient instance;  **private** Connection connection;  **private** **static** **final** String JDBC\_CONNECTION\_URL ="jdbc:sqlserver://localhost\\MSSQLSERVER:60768;databaseName=travelThruAir;user=achoudhury98;password=1234";  **private** **static** **final** String INSERT\_NEW\_RECORD = " INSERT INTO flights VALUES(?,?,?,?,?,?,?,?,'False',NULL)";  **private** **static** **final** String GET\_NORMAL\_FLIGHT\_RECORDS = "SELECT \* FROM flights WHERE departure\_city=? AND arrival\_city=? AND CAST(departure AS Date) = ? AND discount='False'";  **private** **static** **final** String GET\_DISCOUNT\_FLIGHT\_RECORDS = "SELECT \* FROM flights WHERE departure\_city=? AND arrival\_city=? AND CAST(departure AS Date) = ? AND discount='True'";  **private** **static** **final** String GET\_NORMAL\_FLIGHT\_RECORDS\_FILTER\_AIRLINES = "SELECT \* FROM flights WHERE departure\_city=? AND arrival\_city=? AND CAST(departure AS Date) = ? AND airline = ? AND discount='False'";  **private** **static** **final** String GET\_DISCOUNT\_FLIGHT\_RECORDS\_FILTER\_AIRLINES = "SELECT \* FROM flights WHERE departure\_city=? AND arrival\_city=? AND CAST(departure AS Date) = ? AND airline = ? AND discount='True'";  **private** **static** **final** String REMOVE\_FLIGHT\_RECORD = "DELETE FROM flights WHERE flight\_no = ?";  **private** **static** **final** String DISCOUNT\_FLIGHT\_RECORD ="UPDATE flights SET discount\_percentage = ? , discount='True' WHERE flight\_no=?";  **private** **static** **final** String REMOVE\_COMPLETED\_FLIGHTS = "DELETE FROM flights WHERE departure < ?";  **private** **SQLClient**() **throws** SQLException{  **try** {  Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  **this**.connection = DriverManager.getConnection(JDBC\_CONNECTION\_URL);  }  **catch** (ClassNotFoundException e) {  e.printStackTrace();  }    }  **public** Connection **getConnection**() {  **return** **this**.connection;  }  **public** **static** SQLClient **getInstance**() **throws** SQLException{  **if**(instance == **null** || instance.connection.isClosed()) {  instance = **new** SQLClient();  }  **return** instance;  }    **public** **void** **addnew**(Flight f) **throws** SQLException{  Connection conn= **this**.getConnection();  PreparedStatement p = conn.prepareStatement(INSERT\_NEW\_RECORD);  p.setString(**1**, f.flightNo);  p.setString(**2**, f.airLine);  p.setString(**3**, f.departureCity);  p.setString(**4**, f.arrivalCity);  p.setTimestamp(**5**, **new** java.sql.Timestamp(f.departure.getTime()));  p.setTimestamp(**6**, **new** java.sql.Timestamp(f.arrival.getTime()));  p.setInt(**7**,f.passengers);  p.setInt(**8**,f.ticketPrice);  p.executeUpdate();  p.close();  }    **public** List<DiscountFlight> **getdiscountFlights**(String departure,String arrival,Date dt,String airlines) **throws** SQLException{  List<DiscountFlight> discountFlights = **null**;  Connection conn = **this**.getConnection();  PreparedStatement p;  **if**(airlines.equals("Any")) {  p = conn.prepareStatement(GET\_DISCOUNT\_FLIGHT\_RECORDS);  p.setString(**1**,departure);  p.setString(**2**,arrival);  p.setDate(**3**,**new** java.sql.Date(dt.getTime()));  }  **else** {  p = conn.prepareStatement(GET\_DISCOUNT\_FLIGHT\_RECORDS\_FILTER\_AIRLINES);  p.setString(**1**,departure);  p.setString(**2**,arrival);  p.setDate(**3**,**new** java.sql.Date(dt.getTime()));  p.setString(**4**, airlines);  }  ResultSet rs = p.executeQuery();  **while**(rs.next()) {  **if**(discountFlights == **null**) discountFlights = **new** ArrayList<>();  discountFlights.add(**new** DiscountFlight(rs.getString(**1**),rs.getString(**2**),rs.getString(**3**),rs.getString(**4**),**new** Date(rs.getTimestamp(**5**).getTime()),  **new** **Date**(rs.getTimestamp(**6**).getTime()),rs.getInt(**7**),rs.getInt(**8**),rs.getInt(**10**)));  }  **return** discountFlights;  }    **public** List<Flight> **getnormalFlights**(String departure,String arrival,Date dt,String airlines) **throws** SQLException{  List<Flight> flights = **null**;  Connection conn = **this**.getConnection();  PreparedStatement p;  **if**(airlines.equals("Any")) {  p = conn.prepareStatement(GET\_NORMAL\_FLIGHT\_RECORDS);  p.setString(**1**,departure);  p.setString(**2**,arrival);  p.setDate(**3**,**new** java.sql.Date(dt.getTime()));  }  **else** {  p = conn.prepareStatement(GET\_NORMAL\_FLIGHT\_RECORDS\_FILTER\_AIRLINES);  p.setString(**1**,departure);  p.setString(**2**,arrival);  p.setDate(**3**,**new** java.sql.Date(dt.getTime()));  p.setString(**4**, airlines);  }  ResultSet rs = p.executeQuery();  **while**(rs.next()) {  **if**(flights == **null**) flights = **new** ArrayList<>();  flights.add(**new** Flight(rs.getString(**1**),rs.getString(**2**),rs.getString(**3**),rs.getString(**4**),**new** Date(rs.getTimestamp(**5**).getTime()),  **new** **Date**(rs.getTimestamp(**6**).getTime()),rs.getInt(**7**),rs.getInt(**8**)));  }  p.close();  **return** flights;  }  **public** **void** **removeflight**(String flight\_no) **throws** SQLException{  Connection conn = **this**.getConnection();  PreparedStatement p = conn.prepareStatement(REMOVE\_FLIGHT\_RECORD);  p.setString(**1**, flight\_no);  p.executeUpdate();  p.close();  **return**;  }  **public** **void** **removeCompletedFlights**(Date dt) **throws** SQLException{  Connection conn = **this**.getConnection();  PreparedStatement p = conn.prepareStatement(REMOVE\_COMPLETED\_FLIGHTS);  p.setTimestamp(**1**, **new** java.sql.Timestamp(dt.getTime()));  System.out.println("removed "+ p.executeUpdate() + " outstanding records");    p.close();  **return**;  }  **public** **void** **discount**(String flight\_no,**int** discountPercentage) **throws** SQLException{  Connection conn = **this**.getConnection();  PreparedStatement p = conn.prepareStatement(DISCOUNT\_FLIGHT\_RECORD);  p.setString(**2**, flight\_no);  p.setInt(**1**, discountPercentage);  p.executeUpdate();  p.close();  **return**;  }  } |

Output:

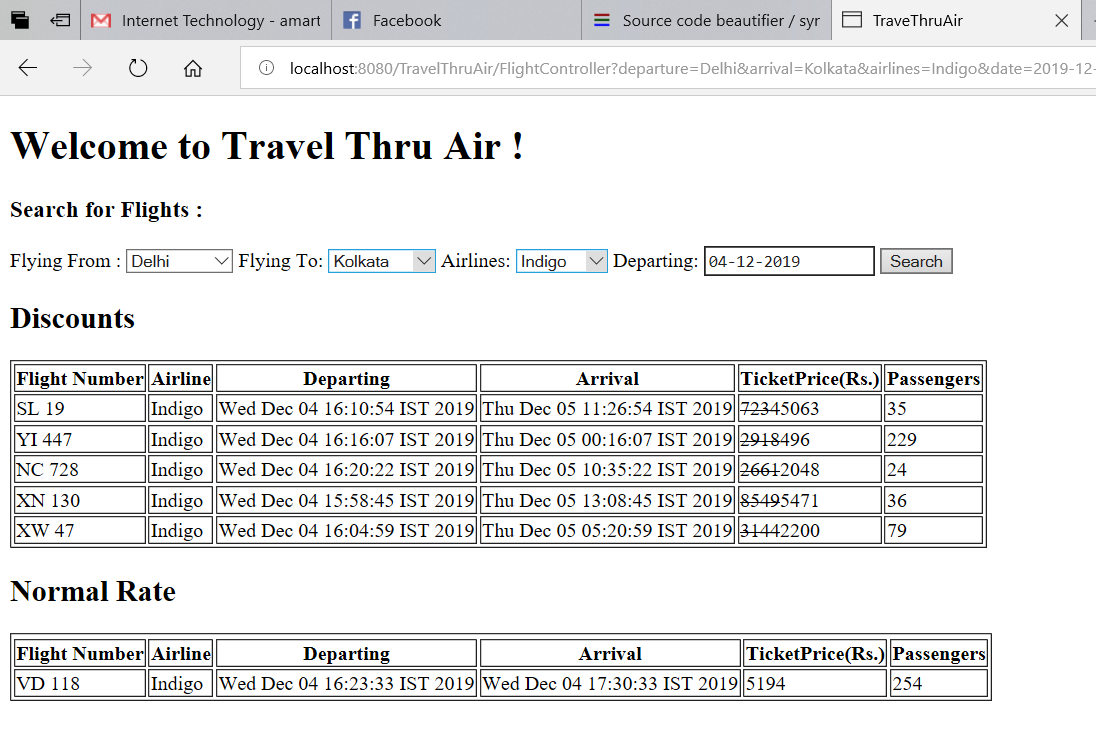


Fig 1A: Flight search demo

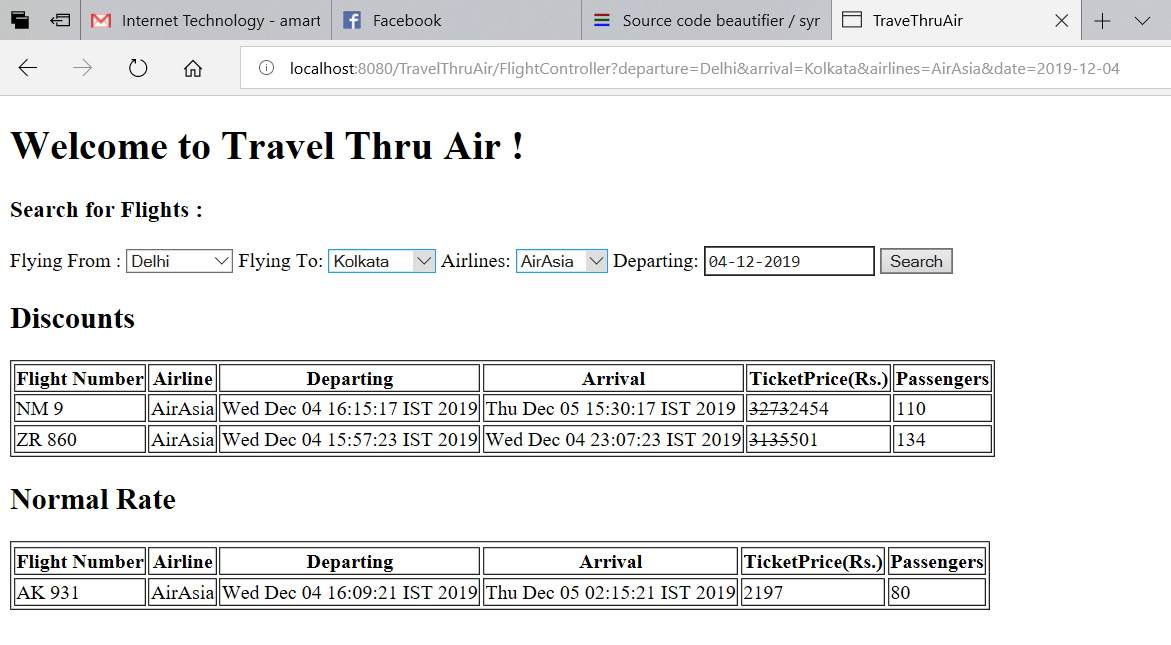


Fig 1B: Flight search demo

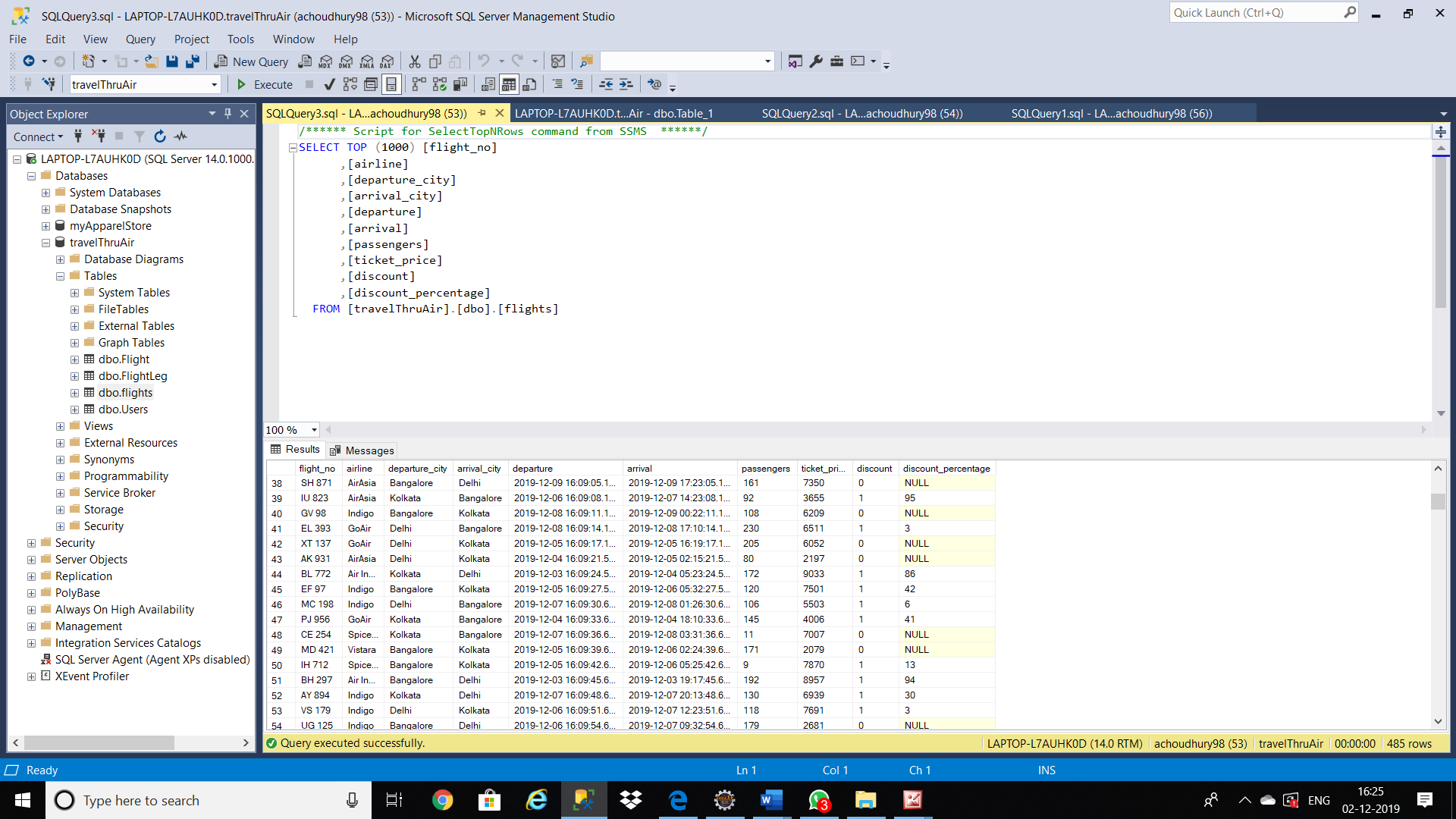


Fig 2 : Flight table

References:

1. Head First Servlet JSP
2. Telusko Video lectures
3. https://www.javatpoint.com/servlet-tutorial