PROJECT REPORT

AIRLINE RESERVATION SYSTEM

CS – 6360 001 Database Design

Department of Computer Science

Eric Jonsson School of Engineering and Computer Science

Submitted by: Binal Kamani

bxk131030@utdallas.edu

03/30/2015

Problem Description:

This programming project involves developing an Airline Reservation System for use by travel professionals (i.e. not passengers). It is a Java-based GUI application that interfaces with a backend MySQL database to manage its data.

Database Schema is based on the fig 3.8 (page 82) of the book – Fundamentals of Database Systems. An ER diagram for this schema can be found in Figure 7.20 (page 236), which provides a conceptual model.

Solution:

The tools used for this project are as below:

- Programming Language: JAVA
- Frontend IDE: NetBeans
- Backend Database management: MySQL

I have used java.sql.* libraries in JAVA to implement connection between Frontend which is JAVA GUI here and backend which is MySQL Airline database.

Backend Implementation:

I have used **MySQL** relational database management system (RDBMS) which is most widely used open-source RDBMS.

Operating System platform: Windows 8.1

MySQL version: 5.6.23

<u>Installation procedure:</u>

- Downloaded the current version for Windows (x86, 64-bit) from http://dev.mysql.com/downloads/mysql/
- Extract mysql files in C:\Program Files\
- Add mysql\bin directory path to the Path environment variable on Windows
- Open and run cmd as administrator and type mysqld –install

Start Database Service:

- Open and run cmd as administrator and type mysqld –-console
- Open and run new cmd as administrator and type mysqld –u root –p
- It ask for passwords: press enter as there's no password

Create Airline Reservation System DB:

This is one of the major part of backend as based on schema and ER diagram queries to create tables should be appropriate considering factors such as primary key, foreign key, constraints etc.

Below is the list of queries I used for creating Database:

1. drop database AIRLINE;

2. create database AIRLINE;

3. use AIRLINE;

4. create table AIRPORT(
   Airport_code varchar(3) PRIMARY KEY,
   Name varchar(50),
   City varchar(20),
   State varchar(2));

5. create table FLIGHT (
   Flight_number INT(4) PRIMARY KEY,
   Airline varchar(2),
   Weekdays varchar(30),
   Departure_Airport_Code VARCHAR(20),
   Scheduled_departure_time TIME,
   Arrival_airport_code VARCHAR(20),
   Scheduled_arrival_time TIME,
   Flight_duration varchar (10),

```
    FOREIGN KEY(DEPARTURE_AIRPORT_CODE) REFERENCES
    AIRPORT(AIRPORT_CODE) ON UPDATE CASCADE,
    FOREIGN KEY(ARRIVAL_AIRPORT_CODE) REFERENCES
    AIRPORT(AIRPORT_CODE) ON UPDATE CASCADE);
```

6. create table AIRPLANE_TYPE (

```
    Airplane_type_name varchar(3) PRIMARY KEY,
    Max_seats int(3),
    Company varchar(15));
```

7. create table FARE(

```
    Flight_number int(4),
    Fare_code varchar(20),
    Amount FLOAT(4),
    Restrictions varchar(20),
    PRIMARY KEY (Flight_number,Fare_code,Amount),
    FOREIGN KEY(FLIGHT_NUMBER) REFERENCES FLIGHT(FLIGHT_NUMBER));
```

8. create table AIRPLANE(

```
    Airplane_id int(4) PRIMARY KEY,
    Total_number_of_seats int(2),
    Airplane_type varchar(3),
    FOREIGN KEY(AIRPLANE_TYPE) REFERENCES
    AIRPLANE_TYPE(AIRPLANE_TYPE_NAME));
```

9. create table FLIGHT_INSTANCE(

```
    Flight_number int(4),
    Date date,
    Number_of_available_seats int(3),
    Airplane_id int(4),
    Departure_time TIME,
    Arrival_time TIME,
    PRIMARY KEY(Flight_number,Date),
    FOREIGN KEY(Airplane_id) REFERENCES AIRPLANE(Airplane_id),
    FOREIGN KEY(Flight_number) REFERENCES FLIGHT(Flight_number));
```

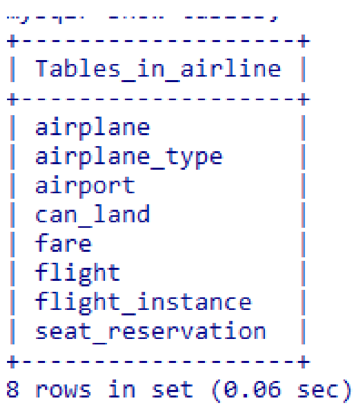
10. CREATE TABLE SEAT_RESERVATION (
    FLIGHT_NUMBER INT(4),
    DATE DATE,
    SEAT_NUMBER varchar(3),
    CUSTOMER_NAME VARCHAR(80),
    CUSTOMER_PHONE CHAR(10),
    PRIMARY KEY(FLIGHT_NUMBER, DATE, SEAT_NUMBER),
    FOREIGN KEY(FLIGHT_NUMBER, DATE) REFERENCES
    FLIGHT_INSTANCE(FLIGHT_NUMBER,DATE)
    );

11. CREATE TABLE CAN_LAND (
    AIRPLANE_TYPE_NAME VARCHAR(3),
    AIRPORT_CODE VARCHAR(3),
    PRIMARY KEY(AIRPLANE_TYPE_NAME, AIRPORT_CODE),
    FOREIGN KEY(AIRPLANE_TYPE_NAME) REFERENCES
    AIRPLANE_TYPE(AIRPLANE_TYPE_NAME),
    FOREIGN KEY(AIRPORT_CODE) REFERENCES AIRPORT(AIRPORT_CODE)) ;

Insert data for Airline Reservation System:

Some tables were provided by the instructor and 2 of the tables were supposed to be created on own. I used the http://www.mockaroo.com/ to create data based on attributes and datatypes given.

After all tables are created, it looks as shown below:



```
+--------------------+
| Tables_in_airline  |
+--------------------+
| airplane           |
| airplane_type      |
| airport            |
| can_land           |
| fare               |
| flight             |
| flight_instance    |
| seat_reservation   |
+--------------------+
8 rows in set (0.06 sec)
```

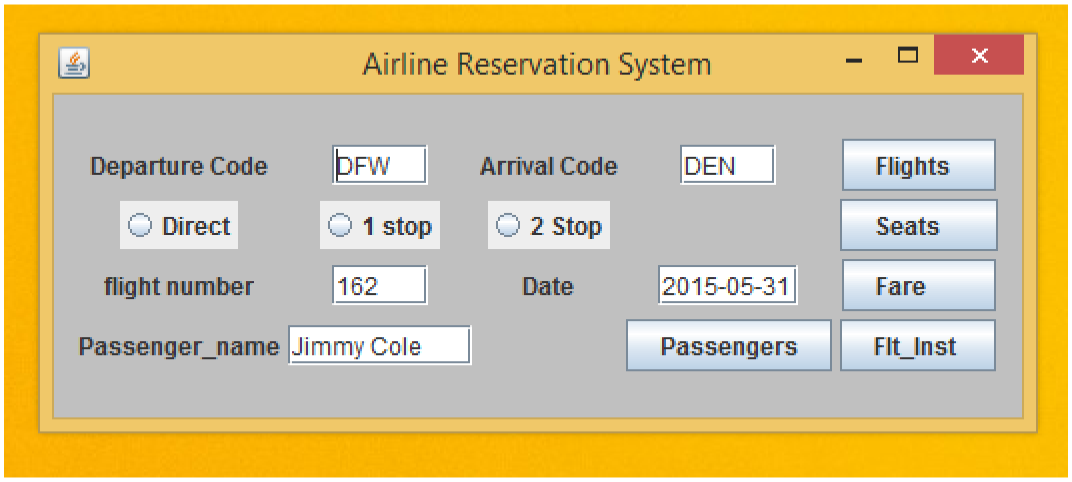
Data is attached with the zip folder for the reference with mysql Insert queries.

Front End Implementation:

I have chosen JAVA as a language to program native GUI for this project. The libraries used for implementing GUI are:

1) Java.awt.*
2) Java. Swing.*

GUI has textbox, labels, buttons, radio buttons, frame and panel which looks as shown below:



Textboxes – User will enter data into
Labels – For indicating what textbox is for what purpose
Radio Button – To select one of the option
Buttons – To request data based on input data

USE CASES – Project Requirements:

USE CASE 1:

Input: Departure Airport Code and Arrival Airport Code

Output: flight numbers and weekdays of all flights

Approach: There can be 3 possibilities based on data in database.

- Direct Flight from source to Destination
- One stop between Source and Destination
- Two Stop between source and Destination

As you may see in the image below, radio buttons are used to select one of the options mentioned above. Which states there are three inputs required for the query.

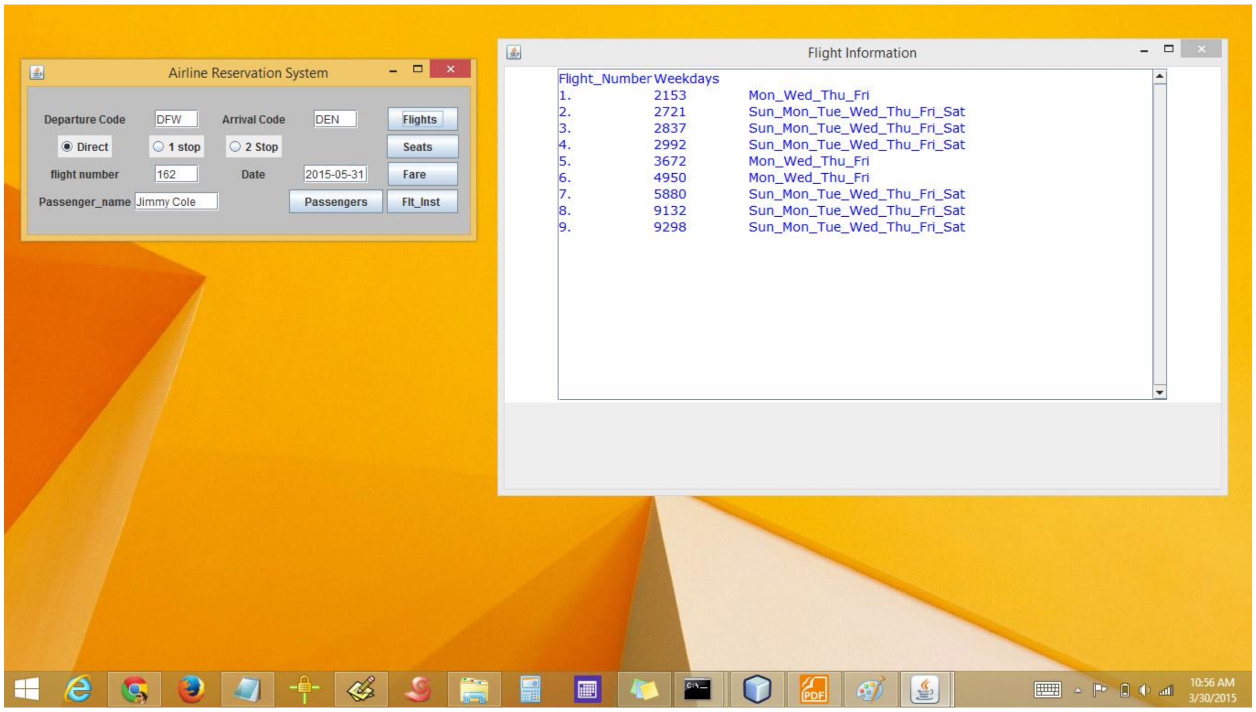
There's switch case in Airplane.java file where each time a new request is detected from user,

- The javasql connection gets established
- Run the query from Netbeans to access database in mysql
- Show the output in a new JFrame

Here based on 3 different options available to user, there are 3 different queries as shown below.

1) **Direct Flight:**
   SELECT flight_number, Weekdays **FROM** flight **WHERE** Departure_airport_code='" + input1 + "'and Arrival_airport_code='" + input2 + "';"

   Output looks as shown in figure below:



Airline Reservation System

Departure Code	DFW	Arrival Code	DEN	Flights
[✓ Direct]	1 stop	2 Stop		Seats
flight number	162	Date	2015-05-31	Fare
Passenger_name	Jimmy Cole		Passengers	Flt_Inst

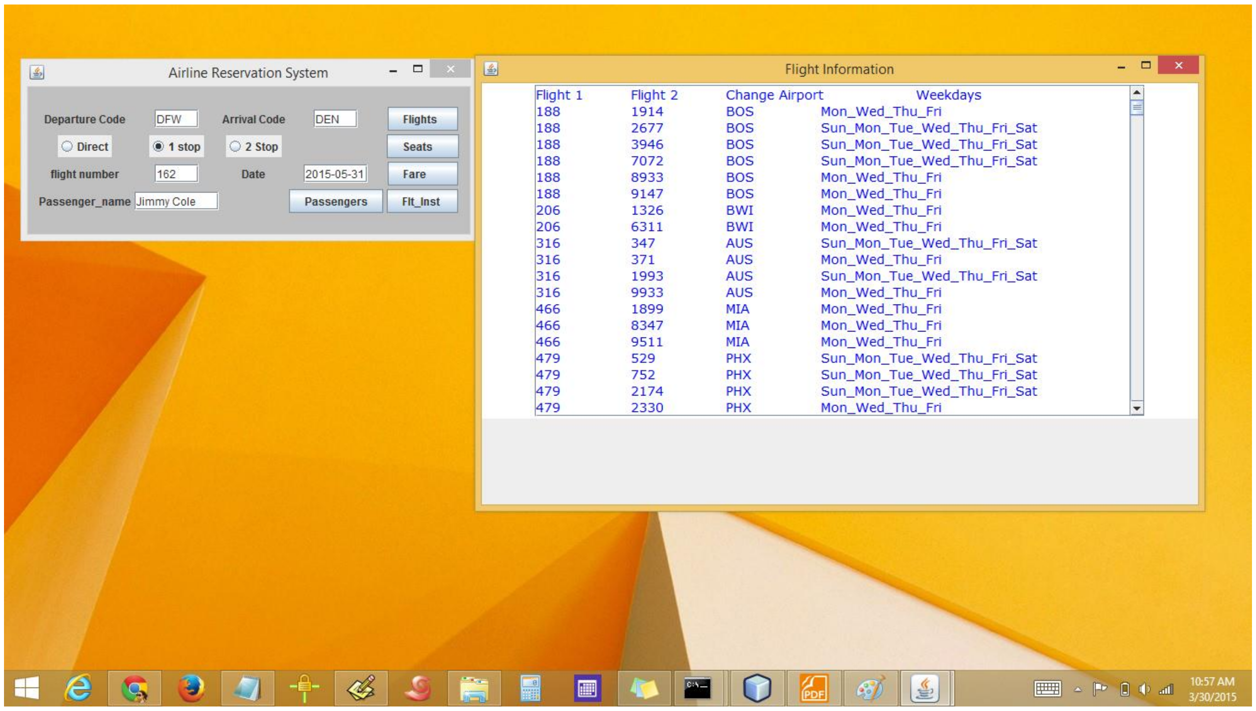
Flight Information

```
Flight_Number Weekdays
1.        2153        Mon_Wed_Thu_Fri
2.        2721        Sun_Mon_Tue_Wed_Thu_Fri_Sat
3.        2837        Sun_Mon_Tue_Wed_Thu_Fri_Sat
4.        2992        Sun_Mon_Tue_Wed_Thu_Fri_Sat
5.        3672        Mon_Wed_Thu_Fri
6.        4950        Mon_Wed_Thu_Fri
7.        5880        Sun_Mon_Tue_Wed_Thu_Fri_Sat
8.        9132        Sun_Mon_Tue_Wed_Thu_Fri_Sat
9.        9298        Sun_Mon_Tue_Wed_Thu_Fri_Sat
```

2) **One stop:** Here function called TIMEDIFF is used in query to evaluate output result based on the time difference which is 1 hour. This says that after flight lands at first stop there is at least one hour duration to catch second flight.

   In addition to which, there needs to be a check that the second flight is starting after first flight lands hence the scheduled departure time of second flight is more than scheduled arrival time of first flight.

SELECT F1.flight_number, F2.flight_number, F1.Weekdays, F2.Weekdays, F1.Departure_airport_code, F2.Departure_airport_code, F1.Arrival_airport_code, F2.Arrival_airport_code, F1.Scheduled_arrival_time, F2.Scheduled_departure_time
FROM flight F1, flight F2 where F1.Departure_airport_code = '" + input1 + "' and F2.Arrival_airport_code = '" + input2 + "'
and F1. Arrival_airport_code = F2.Departure_airport_code
and TIMEDIFF(F2.Scheduled_departure_time, F1.Scheduled_arrival_time) > '01:00:00' "
+ "and F2.Scheduled_departure_time > F1.Scheduled_arrival_time");



Departure Code DFW
Arrival Code DEN
Direct [✓ 1 stop] 2 Stop
flight number 162
Date 2015-05-31
Passenger_name Jimmy Cole

```
Flight 1     Flight 2     Change Airport          Weekdays
188          1914         BOS          Mon_Wed_Thu_Fri
188          2677         BOS          Sun_Mon_Tue_Wed_Thu_Fri_Sat
188          3946         BOS          Sun_Mon_Tue_Wed_Thu_Fri_Sat
188          7072         BOS          Sun_Mon_Tue_Wed_Thu_Fri_Sat
188          8933         BOS          Mon_Wed_Thu_Fri
188          9147         BOS          Mon_Wed_Thu_Fri
206          1326         BWI          Mon_Wed_Thu_Fri
206          6311         BWI          Mon_Wed_Thu_Fri
316          347          AUS          Sun_Mon_Tue_Wed_Thu_Fri_Sat
316          371          AUS          Mon_Wed_Thu_Fri
316          1993         AUS          Sun_Mon_Tue_Wed_Thu_Fri_Sat
316          9933         AUS          Mon_Wed_Thu_Fri
466          1899         MIA          Mon_Wed_Thu_Fri
466          8347         MIA          Mon_Wed_Thu_Fri
466          9511         MIA          Mon_Wed_Thu_Fri
479          529          PHX          Sun_Mon_Tue_Wed_Thu_Fri_Sat
479          752          PHX          Sun_Mon_Tue_Wed_Thu_Fri_Sat
479          2174         PHX          Sun_Mon_Tue_Wed_Thu_Fri_Sat
479          2330         PHX          Mon_Wed_Thu_Fri
```

3) **Two Stop:** As the previous option, this option also has use of TIMEDIFF function in mysql query as well as there are more checks like, arrival airport code of first flight is same as departure code of second flight also scheduled departure time of second flight is more than scheduled arrival time of first flight and similar checks for flight 2 and flight 3.

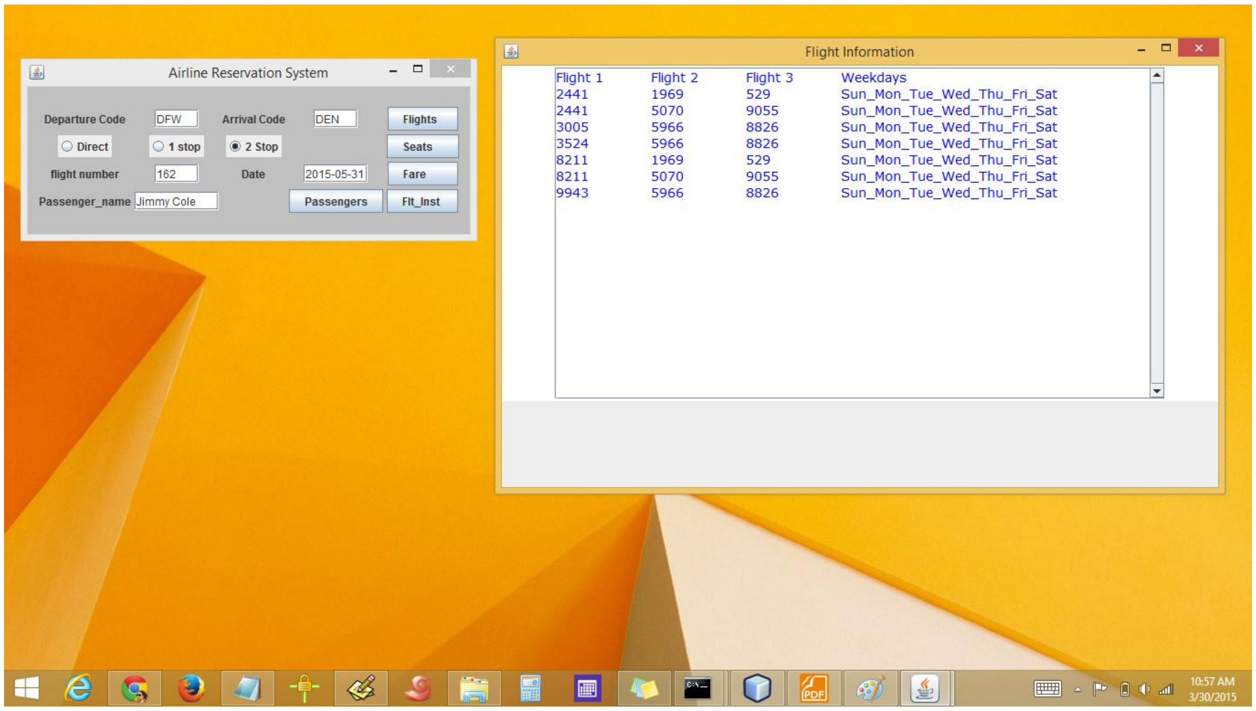
The additional check here is to make sure that

- F1.Departure_airport_code!=F2.Arrival_airport_code
- F2.Departure_airport_code!=F3.Arrival_airport_code

  There is a probability of results having destination of first flight same as final destination which is no use result.

One major concern in option 2 and 3 is to find connecting flights on the same day in a week, I have addressed as below:

- Parse Weekdays of all flights using Weekdays.split
- Assign appropriate value to integer WD_count
  For ex Mon = 1, Tue = 2, Wed = 4, Thu = 8 and so on..
- Keep on adding this value to WD_count for all Weekdays for single result
  Max value can be 127 which is all days
- Do the same procedure for connecting flight, let's call it WD_count1
- Compare WD_count and WD_count1 by AND operation
- In case of no common days, result will be 0
- Otherwise display output with flight numbers and weekdays.



Departure Code DFW
Arrival Code DEN
Direct [✓ 2 Stop] 1 stop
flight number 162
Date 2015-05-31
Passenger_name Jimmy Cole

```
Flight 1   Flight 2   Flight 3   Weekdays
2441       1969       529        Sun_Mon_Tue_Wed_Thu_Fri_Sat
2441       5070       9055       Sun_Mon_Tue_Wed_Thu_Fri_Sat
3005       5966       8826       Sun_Mon_Tue_Wed_Thu_Fri_Sat
3524       5966       8826       Sun_Mon_Tue_Wed_Thu_Fri_Sat
8211       1969       529        Sun_Mon_Tue_Wed_Thu_Fri_Sat
8211       5070       9055       Sun_Mon_Tue_Wed_Thu_Fri_Sat
9943       5966       8826       Sun_Mon_Tue_Wed_Thu_Fri_Sat
```

<u>USE CASE 2:</u>

Input: Flight Number, Date
Output: Number of available seats
Approach: The table Flight_instance has an attribute called Number_of_available_seats hence it does not seem difficult to retrieve it based on some JOIN condition.
But it's not only about retrieving this attribute, it is more than that.
I have implemented a check which checks if the database is updated, make changes to Number_of_available_Seats accordingly.

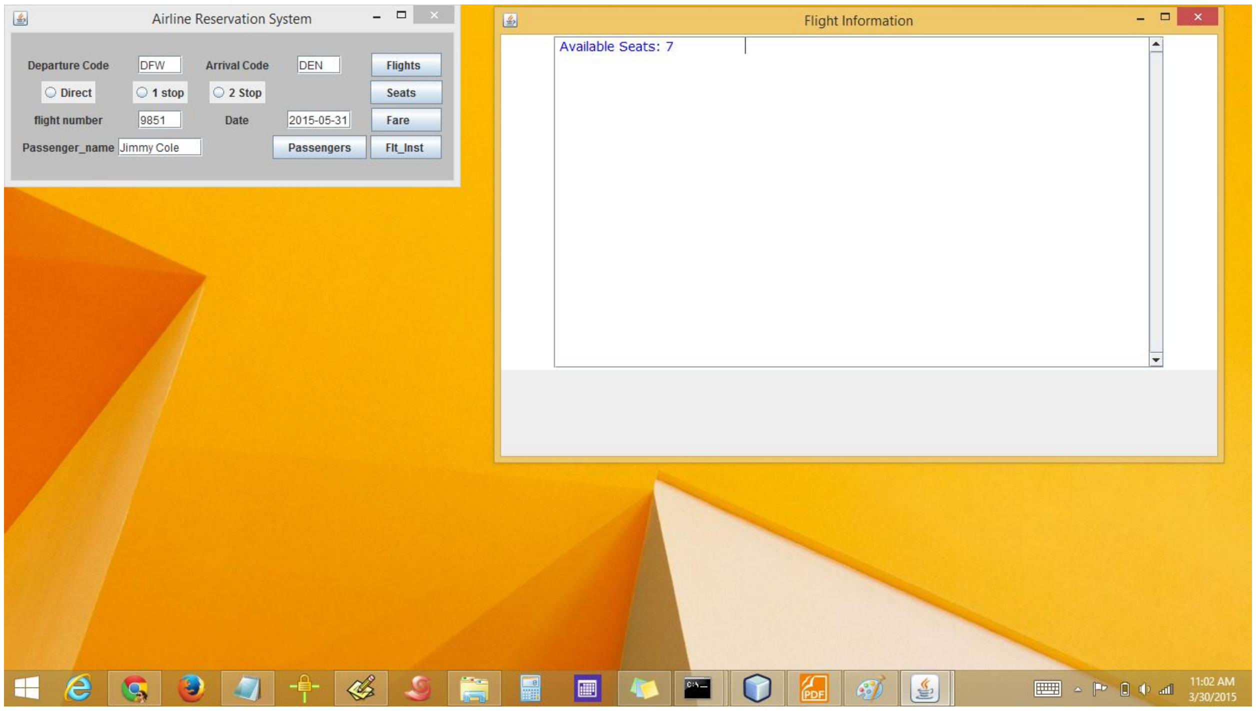
It might often happen that a professional who is going to use this application also use command line to insert some new data and let's say the new data is in Seat Reservation table such as new passenger is on XYZ flight, hence one seat should be reduced from number of available seats in Flight instance table.

It is more professional to check this data every time the data on available seat is required. In JAVA, I have included this check after the query shown below:

SELECT F.Number_of_Available_seats, A.Total_number_of_seats - count(*), F.flight_number **FROM** Airplane A, Flight_instance F , Seat_reservation S
WHERE F.Flight_number=S.Flight_number and A.Airplane_id=F.Airplane_id and F.Flight_number='" + input1 + "' and F.Date='" + input2
+ "' GROUP BY S.flight_number;

If there's a change in the Seat Reservation then the calculated available seats will not be same as one in flight instance table. In this case it will run update query to change value in the flight_instance accordingly and show correct result in output.

The output result looks like one shown below:



```
Available Seats: 7
```

Available seats flight instance table = rs1.getInt("Number_of_Available_seats");
Calculated available seats= rs1.getInt("A.Total_number_of_seats - count(*)");

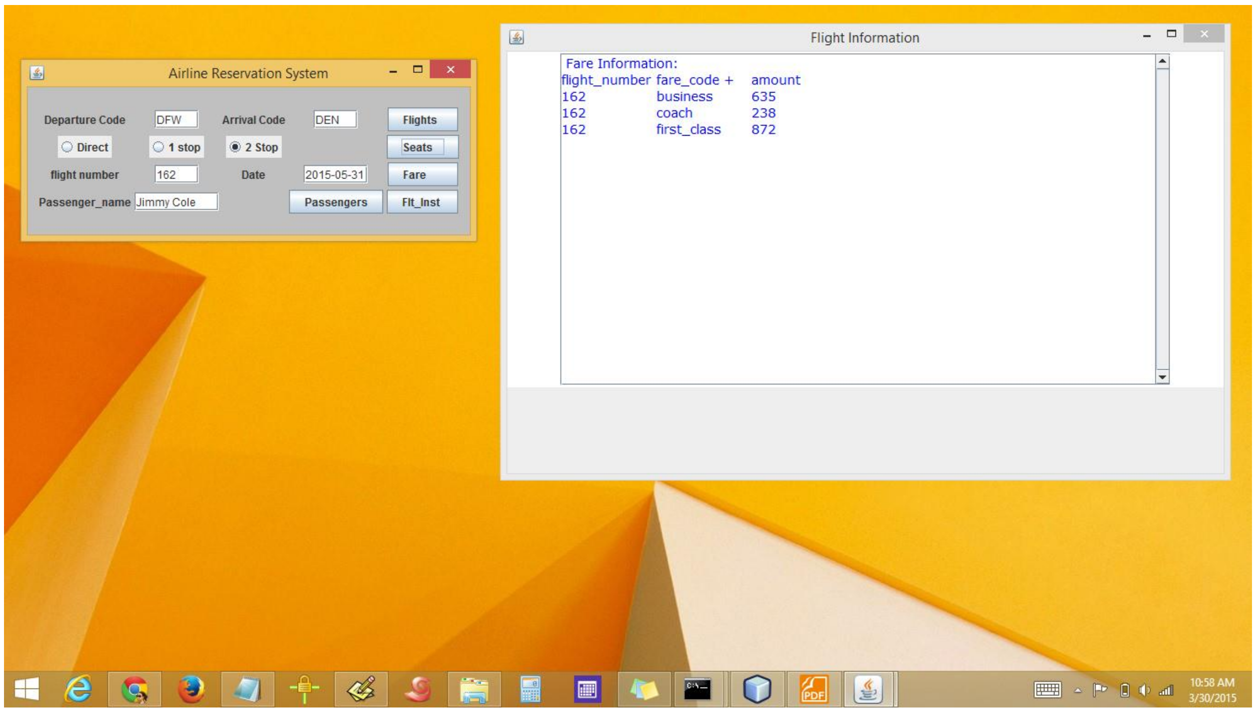
Update Query:

UPDATE flight_instance **SET** Number_of_available_seats='" + avail_seat_calc + "' **WHERE** Flight_number='" + input1 + "';

USE CASE 3:
Input: Flight Number
Output: Fare
Approach: In this use case, application will show fare for all class (Business, first_class, Economy) for entered flight number.

The query is as below:
"**SELECT** flight_number, fare_code, amount **FROM** fare **WHERE** flight_number='" + input1 + "';"

The output for this use case is shown here:



```
Fare Information:
flight_number fare_code +   amount
162           business      635
162           coach         238
162           first_class   872
```

USE CASE 4:

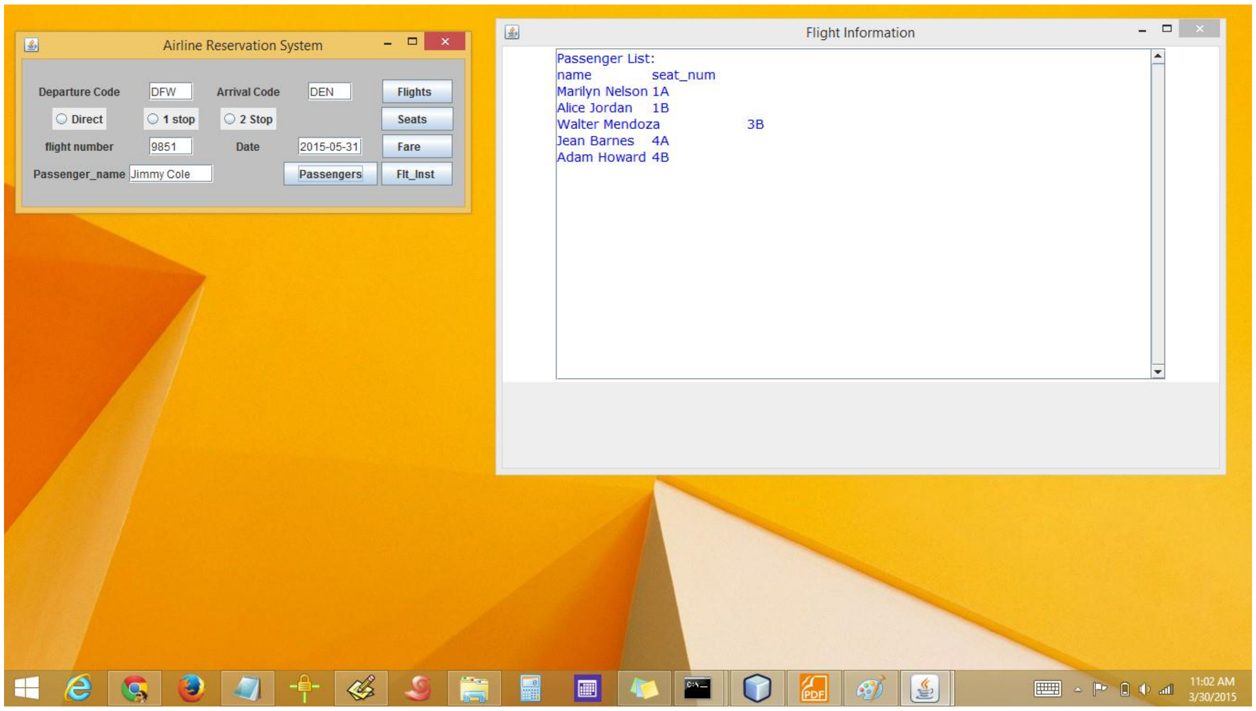
Input: Flight Number, Passenger Name

Output: Flight instances, list of passengers

Approach: Given the flight number, run query to get list of passengers on this flight, the query is as show below.

SELECT Customer_name, Seat_number **FROM** Seat_Reservation **WHERE** Flight_number = '" + input1 + "';

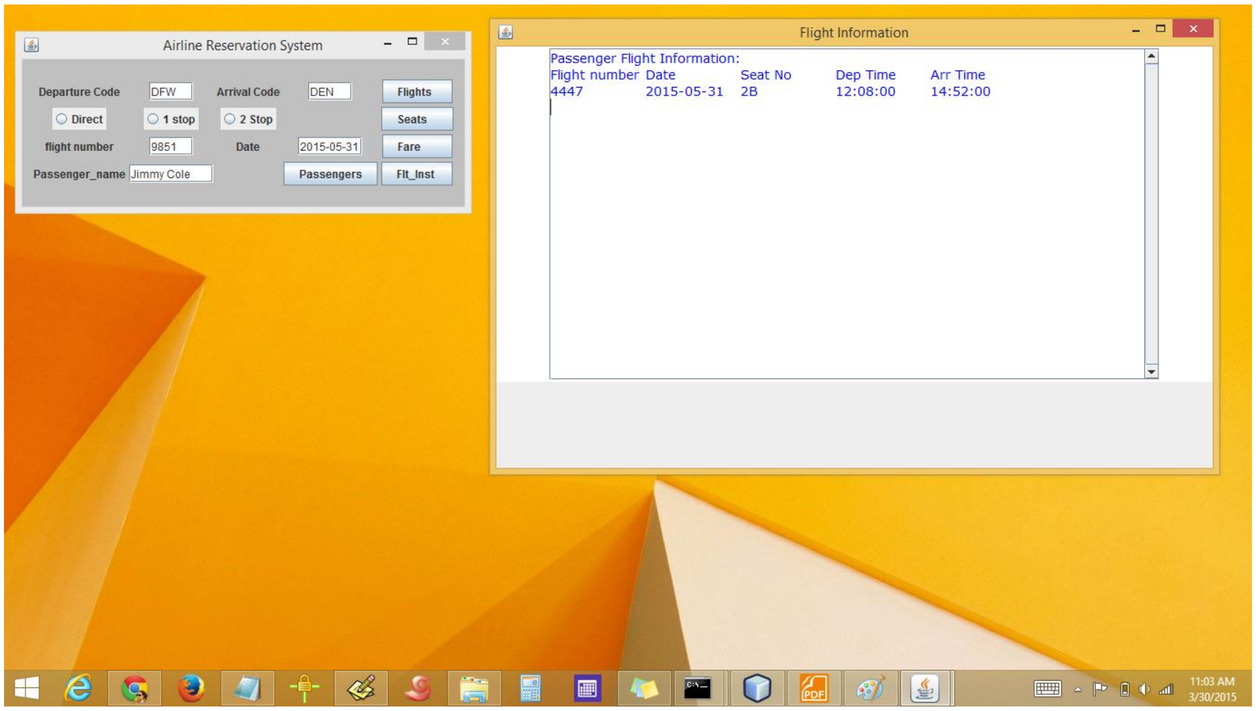
Result is shown below:



```
Passenger List:
name         seat_num
Marilyn Nelson 1A
Alice Jordan    1B
Walter Mendoza              3B
Jean Barnes    4A
Adam Howard 4B
```

For the second case: given the input of Passenger name, search for all flight instances possible.

Query : "**SELECT** S.Flight_Number, S.Date, S.Seat_number, F.Departure_time, F.Arrival_time **FROM** Flight_Instance F, Seat_Reservation S **WHERE** S.Flight_number=F.Flight_number AND F.Date=S.Date AND Customer_name='" + input1 + "';"



```
Passenger Flight Information:
Flight number Date        Seat No     Dep Time    Arr Time
4447          2015-05-31  2B          12:08:00    14:52:00
```