Flight Reservation System

1. Introduction

The aim of this project is to develop a web-based Flight Reservation System (FRS). The FRS serves both business and airline customers. Business customers can search and book flights with different routes to choose and when the booking is completed, an email which contains booking information will be sent to him/her. On the other hand, airline staff can do modifications to their corresponding routes, at the same time, they can set specific routes to active/inactive. Once route is set inactive, business customers will not be able to search that route. The owner of the FRS remembers traveler's ID and from which airline the booking was made and its commission from that booking. All the information will be stored in the database and displayed via webpages.

2. Requirement Specification

- **Airline Customers:**

1. Airline customers need to be logged in.
2. Airline customers can update both their own routes and fares.

- **Business Customers:**

1. Business customers need to be logged in.
2. Business customers can search routes and make bookings
3. Business customers need to do payment via PayPal.
4. An email which contains booking information shall be sent once booking is completed.

- **Website Owner:**

1. Owner's commission will be made, which is 20 percent of the fare from each booking.

3. System Design

This project adopted Hibernate to process data from the database and PayPal is integrated to make transactions for business customers.

The FRS application is divided into 3 parts. Business Customers, Airline Customers and Website's owner. The application contains 12 JSPs, 6 servlets, and 5 session beans.

JSPs acquire user inputs and pass them to servlets and data will get further processed from session beans and servlets, the result will be returned to JSPs to display to users.

Figure 1 as a flow chart shows how the application implements.

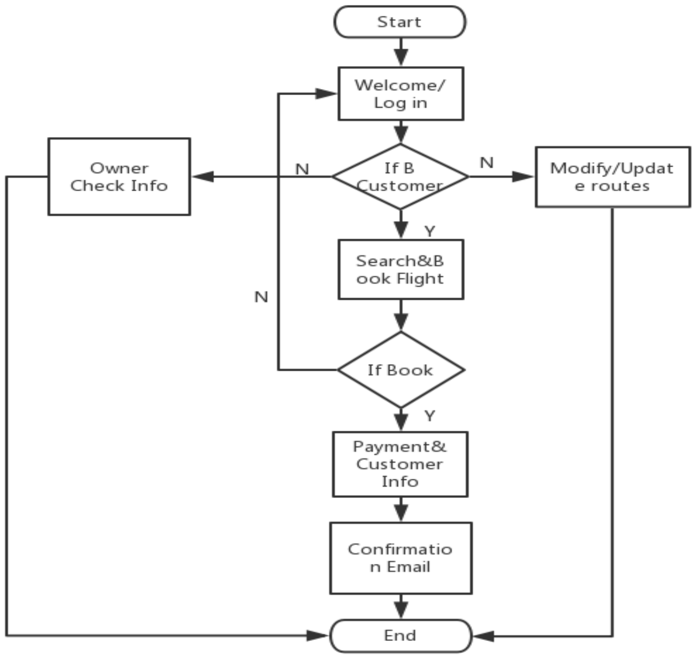

Figure 1

In total, 6 tables were made. “Users”, “Airplanes”, “Airports”, “Routes”, “Travelers”, and “Owner”.

In “Routes” table, “airplaneid” is the foreign key which bonds with “id” in “Airplanes” table. In “Traveler” table, “routeid” is bonded with “id” in table “Routes”. “travelerID” in Owner table is bonded with “id” in table “Traveler”.

Figure 2 as an EER diagram shows table relationships.

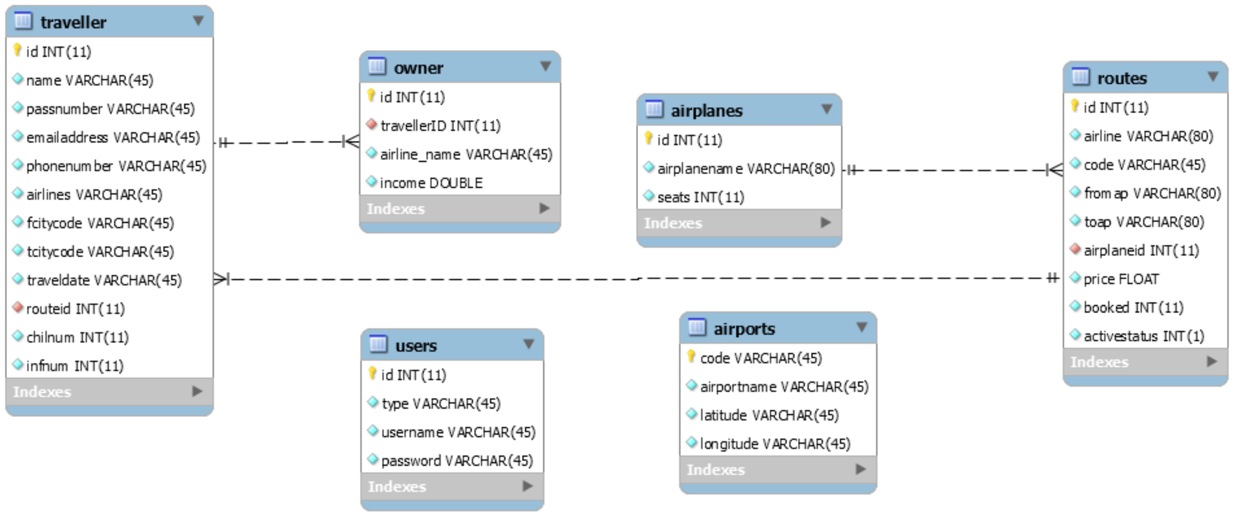

Figure 2

4. System Implementation

4.1 Login/Register

All users need to be logged in to have further control to other functions. If a user does not have an account, they can choose to become a business customer or airline customer and input username(unique) and password, then an account will be created.

4.2 Update Routes and Fare

Airline customers can have access to their corresponding routes, they can add, delete, modify and search their routes. As well as they can set routes to be active/inactive. Once route is set as inactive, B-customers will not be able to find it. In addition, if one route was used for a booking, airline customer will not able to delete or modify it.

4.3 Make a Booking

Business customers can search from airport of departure to airport of destination, with these information, multiple routes will be returned (if there is one or more). Customers can select which route they are willing to choose, then they need to do transactions. Based on traveler's age, corresponding discounts will be given, for instance, child, infant. When transaction is completed, B-customer is required to input personal information, for instance, name, email-address. Then an email which contains booking information will be sent to the email-address which was provided by the customer.

4.4 Transaction

Fare is stored in DB in table "route", when transaction is needed, it will be used as a parameter passing to PayPal, and PayPal will withdraw money from user's PayPal account.

4.5 Website's Owner's Commission

When one booking is made, the "owner" table will record from which business customer the booking is made and which airline was used for this booking. The information is stored in DB and can be statistically use in the future. And from each booking, the website's owner gets 20% of the fare.

5. Testing Results

Table 1

Functions	Results
Register	Pass
Log-in	Pass
Search Routes	Pass
Modify Route	Pass
Delete Route	Pass
Add Route	Pass
PayPal	Pass
Email	Pass
Store Owner Commission	Pass

6. Discussion

During the implementation of this project, two challenges have occurred. The first one is adopting Hibernate. Due to lack of experience, instead of using SQL query to access and process DB, using HQL was a challenge that I faced. Another challenge that I encountered was using PayPal. PayPal was integrated into one of the JSPs, and the fare was passed to it as a parameter. However, during implementation, the transaction kept failing. After debugging for a while, the problem was that the fare has too many decimal numbers, and PayPal could not process such numbers. The solution was trimming the fare with only two digits left after the decimal points. Then the problem was solved.

7. Conclusion

During this project, a clearer way of knowing how web application's components are interacted and their own usage was learnt. Also, I have learnt how to build a web-based, cross-platform application with Hibernate by using NetBeans IDE. Moreover, this project helps me to understand how to think as a developer to satisfy clients' needs and how to convert the needs into programming.

8. Reference

[1] K.Venkatesh, Tejas.Deshpande. (2017). DBridge: Translating Imperative Code to SQL. *SIGMOD '17*. 4 (Pages 1663-1666), 1.