Specification Document - FlyAway

Author: Trey Turley

1. Introduction

1.1 Purpose

FlyAway is a ticket booking portal that will allow customers to find and book flights.

1.2 Intended Audience

This application will be used by people looking to purchase tickets for traveling via airline to a destination.

1.3 Intended Use

This application will provide a web portal that users can utilize to find flights based on some criteria like origin, destination, and number of travelers.

1.4 Scope

Users will be able to search for a flight based on origin, destination, and number of travelers. Matching flights will be shown to the customer along with pricing information. The Customer may then proceed by entering the passenger details before purchasing tickets via a dummy payment portal. An admin portal will also be available for managing the available locations, airlines, flights, and prices. The admin may also change the login password (default pw is admin).

1.5 Definitions and Acronyms

N/A

1.6 GitHub

https://github.com/TreyT-FSD/Phase2-Project

2. Overall Description

2.1 User Needs

2.1.1 Customer

The customer is anyone looking to book a flight. FlyAway will allow them to search for flights. Once a flight is found they can enter the passenger details and purchase tickets.

2.1.2 Admin

The admin will be able to manage the list of locations, airlines, and flights that are available to book. The default admin password is admin.

2.2 Assumptions

2.2.1 Customer Ticket Purchase Limit

No specifications were made around ticket purchase limits for the customers. We are not asking the admin to specify limits on the number of tickets a customer can purchase in the admin portal. As such, we will set a global limit of 9 tickets per purchase on the customers. This is being imposed so that there is some amount of realism and so that our booking registration page (called flight booking page in the app) can have a limited number of fields for entering passenger details. In the real world, the number of tickets available for purchase for a given flight would be dictated by a number of other factors that we are not taking into account.

2.2.2 Booking Details are Not Saved

The specifications require that a customer can search for flights and purchase tickets. There are no requirements for a customer to be able to view previously purchased tickets. As such, after the customer completes the purchasing process and they leave the confirmation page, their only options will be to go to the homepage where they can search for and book additional flights if they so choose.

2.2.3 Booking The Same Flight

No specifications were made regarding the number of tickets that can be purchased. As such, a customer will potentially be able to book hundreds of tickets for the same flight by repeatedly going through the booking process. This is allowed as we are not keeping track of how many tickets the customer purchases and the only limit is that a max of 9 tickets for a flight can be purchased at a time.

2.2.4 Flight Capacity Constraints

No specifications were made regarding the number of tickets available for a given flight. Additionally no flight capacity constraints have been outlined. As such, we assume flights have infinite capacity. This means a customer could potentially book hundreds of tickets for the same flight. They will however be limited to purchasing 9 tickets at a time.

2.2.5 Running the application

As this is a project to demonstrate what we have learned in the simplilearn class and not a live production site, the application will have to be run locally. This will have some dependencies, specifically with mysql, as outlined in section 2.3.

2.2.6 Admin Credentials and Security

As this is a class project and not a real world production website. It's likely that the admin's credentials will be available in plaintext both on the website and in the database to aid in testing and developing the application. Basic security practices like masking the password input box will be used. Otherwise, the password will most likely be available in plaintext.

2.2.7 Flight Availability

In the real world, flight availability is determined by schedules, weather, airplane availability and a number of other factors. In our application, we will assume that as long as the customer enters a valid origin and destination, then the date they have selected for departure does not matter and a flight will be available. The time of the flight may be randomly generated,hardcoded, or not mentioned at all.

2.2.8 All flights are one way

We are going to assume that all tickets purchased are for one-way flights. Only a departure date is required to be entered when searching for flights. No return dates or round-trip style flights will be available.

2.2.9 URL Manipulation and General Application Security

During class we did not cover best practices for managing general application security. As such, it's possible that users could manipulate the urls of the application as well as the input fields. This will likely result in errors and/or undefined behavior of the application. Some basic error handling and parameter validation has been put into the application to handle some common issues that could occur.

2.3 Dependencies

2.3.1 Database

This application was developed with A MySql 8 DB for the backend. The connection string in the hibernate.cfg.xml is expecting a DB called flyaway to be available via the connection string - jdbc:mysql://localhost:3306/flyaway. Connection properties will need to be updated to work in local deployments of the app.

2.3.2 Libraries

This is a Maven project and as such all library dependencies will be defined in the pom.xml file. Eclipse will then handle integrating these dependencies with the application during development.

2.3.3 Web Server

The app was developed with a Tomcat v9.0 server.

2.4 Application Appearance

This application is a webapp and can be accessed at http://localhost:8080/FlyAway/index.jsp. It uses basic HTML elements for user interaction and for displaying data.

3. System Features and Requirements

3.1 Functional Requirements

3.1.1 Flight Searching

The customer will be able to input basic information like trip date, origin, destination, and number of travelers to find flights that match the criteria. The customer will be able to select the flight and complete some basic passenger details. Once that is done, payment is handled via a dummy payment button and then the final booking confirmation is shown.

3.2 External Interface Requirements

N/A - no external API

3.3 System Features

The application can be accessed at http://localhost:8080/FlyAway/index.jsp

3.4 Non-Functional Requirements

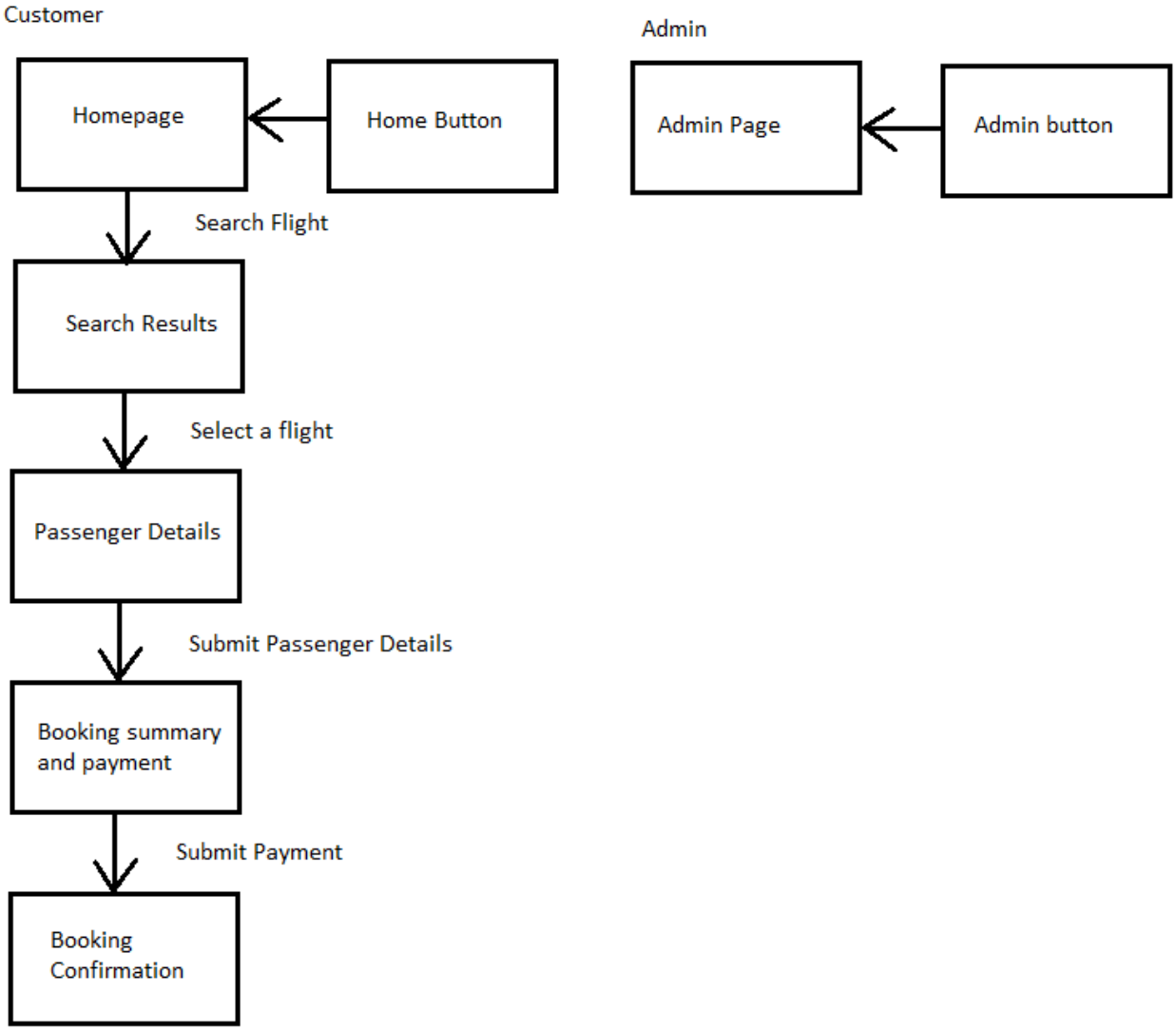
No specifications were provided for any non-functional requirements such as SLAs for user interactions or response times. However, an effort will be made to use efficient algorithms where needed.

3.5 Future Enhancements

3.5.1 Flyaway enhancements

- Search via flight number
- Find all flights for a certain airline
- Filter flights to a specific airline
- Filter by price
- Find all flights with the specified origin

3.6 System Interaction Flow Chart



4. Application Details

4.1 Java Concepts

Basic OOP principles like classes, objects, and encapsulation are used for the locations, airlines, and flights.

4.2 Data Structures

Lists are used to store data that is being retrieved from the DB.

5. Sprint Planning

5.1 Sprint 1 (1 Week)

5.1.1 Story 1: Application Design

As a user, I want to be able to search for flights and book tickets.

5.1.1.1 Task 1: Implement classes for flights, passengers, locations, and airlines. Annotate so that hibernate will create them for us in the DB at runtime.

5.1.2 Story 2: Admin Page

As an admin, I want to be able to add locations, airlines, and flights.

5.1.2.1 Task 2: Implement the DAO layer for the entities so that the admin can add and delete locations, airlines, and flights from the DB.

5.2 Sprint 2 (1 Week)

5.2.1 Story 3: Implement Front-End for Customer Happy Path

As a user, I want to be able to search for flights and book tickets.

5.2.1.1 Task 3: Implement front-end for flight searching, passenger detail entry, and displaying flight details and purchase confirmation.

5.2.1.2 Task 4: Implement the servlets that will assist with the booking flow.

5.2.2 Story 4: Admin Page

As an admin, I want to be able to add locations, airlines, and flights.

5.2.2.1 Task 5: Implement front-end for admin page to enable adding and deleting location, airlines, and flights.