Requirements Analysis Document (RAD)

Prepared for

World Plane, Inc. (WPI)

Prepared by

Team jellycat

Dongrui Qi, Wei Xiong, Zhuoqun Yuan, Yueyue Zhong

March 28, 2019

[version 2]

Requirements Analysis Document (RAD)

# Introduction

## Purpose of the system

WPI (World Plane, Inc) has an airline flight database running on an Internet accessible server, which is not comprehensible for common clients who want to book flights based on that. Our proof of concept software is browser-based and is aimed to transitioning the Travel Agency airline travel reservation system to a Retail Customer airline reservation system with feasibility. And it’s aimed to be conceptually similar to the existing web-based airline reservation systems (like google flights). It’s designed such that all new customers can understand all flights options easily and clearly and make flight reservations based on their personal preferences.

## References

* Statement of Work (SOW) provided by World Plane, Inc. 24 January 2019 and dated 05/10/2019.
* Lecture slides
* Object-Oriented Software Engineering Using UML, Patterns, and Java (3rd Ed) by Bruegge, Bernd, Dutoit, Allen H.

## Scope of the system

This system designs and implements concepts of an airplane reservation system. Supported by WPI airline flight database, all available air flight information can be accessed easily through a standard HTTP GET Application Programming Interface (API). A friendly customer interface is provided so that various combinations of search criteria can be fetched from the customer and corresponding database search results can be generated and sorted if prompted by customers. Customers can make reservations on the system based on their specific preferences. The system provides a solution allowing the customer to search for desired travel (like the departure airport and arrival airport), flight date (local date and time for both departure and arrival), one-way trip or round way trip. And customers can sort the results by price (in dollars), departure time, arrival time, travel time, or seat type (coach seat or first-class seat). Then the customer can reserve seats on preferable flights.

## Core System Functionalities

1. Create a proof of concept system that can be used by customers to reserve an airline flight.
2. Ensure that airline tickets reservation system is easy and customer friendly for customers.
3. Widely Search and Precise Search for locating flight list and applying customer preference.
4. Display all needed information for each flight based on customers’ requests.
5. Choose one-way or round-trip flight.
6. Display the local departure time and arrival time of each leg.
7. Customers can sort the flights results based on all-together price.
8. Customers can sort the flights results based on departure time.
9. Customers can sort the flights results based on arrival time.
10. Customers can sort the flights results based on travel time.
11. Customers can reserve first class or coach seating for travel through the system.
12. Select flights and confirm the selection prior to the reservation being made.
13. Select flights and confirm before the reservation being made.
14. Customers can exit the system and return to the initial state.

## Objectives and Success Criteria of the Project

The success of the application depends upon meeting the following core set of objectives:

1. Deliver a proof of concept system that can be used by customers to reserve an airline flight.
2. The system is easy and customer friendly for customers.
3. The system can make reasonable response to unsuccessful search.
4. Timeliness response to customer’s actions is guaranteed.
5. Ensure an easy to use and conflict-free reservation system for customers to reserve a flight.

# Current System

## Existing System

The current system is operated by travel agents employed by World Plane Inc (WPI.) When travelers want to fly from departure airport to destination airport, they will engage with a WPI travel agent who uses existing proprietary software to book the flights for the customer.

And an API to connect with the Database. Through current API, the system allows customers to make query about airports, airplanes and airports arrive and depart airports, and also allow developers to reset the database. After improvement, customers can make queries in a more customer-friendly way, and can also add or modify data through input. The original system will still be used to get the source data.

## Current Operations

Currently, travel agents employed by World Plane Inc (WPI.) When travelers want to fly from departure airport to destination airport, they will engage with a WPI travel agent who uses existing proprietary software to book the flights for the customer.

And there is an API where we can make queries, and get XML containing queried information. More current operations have not been provided.

# Proposed System

## Overview

This system represents the initial version of the Airline Reservation System. At a high level, this system will allow a customer to check flights, sort flight results, and reserve seats on flights. The goal is to allow customers greater and easier access to the airline’s booking system at any time.

## Conceptual Model - Customer Scenarios

|  |  |
| --- | --- |
| *Scenario name* | searchOneWayTrip |
| *Participating actor instance* | Bob: customer |
| *Flow of events* | 1. Bob lives in Worcester and wants to fly to his parents’ house in Los Angeles for a break in May. 2. He is thinking about taking the trip on May 10th, flying from Boston to Los Angeles because he hasn’t decided when to return, he just wants to look up a one-way trip and he is just a student, he only wants to take the coach seat. 3. He types in the date he wants to take the trip: May 10th, and departure airport BOS, and arrival airport LAX, and choose one-way trip and coach seat. 4. From all the available flights from BOS to LAX, Bob wants to sort the results by total flight price, he clicked “sort by price” button. He doesn’t care about the number of layovers, he just wants to buy the cheapest flight, so he chooses the flight with the lowest price and checked that each layover period is with 30 to 120 minutes which is enough for him to make transfer, so finally he makes the reservation. |

|  |  |
| --- | --- |
| *Scenario name* | searchRoundTrip |
| *Participating actor instance* | Joe: customer |
| *Flow of events* | 1. Joe is a businessman and wants to take a business trip from Boston to New York in May 11th and return in May 13th. All the expenses are covered by his company, so he doesn’t care about the price. 2. He wants to make the most use of his time, so he wants to leave as early as possible and return as late as possible. Besides he doesn’t want the trip to have layovers. 3. He types in the date he wants to leave: May 11th , departure airport: BOS, and arrival airport JFK, seat type: first-class, and also he chooses the trip type to “Round Trip”. Then he had to fill out the return date: May 13th 4. From all the available flights from BOS to JFK, Joe first sorts the flight by departure time and choose the one that leaves at 8am with no layovers and confirm the flight. 5. Then he also sorts all the return flights from JFK to BOS by departure time and choose the latest with no layovers and confirm this return flight. 6. After choosing a whole trip from BOS to JFK and from JFK to BOS, he finally clicks the “reserve” button and make a reservation of the whole trip. |

|  |  |
| --- | --- |
| *Scenario name* | cancelTrip |
| *Participating actor instance* | Ann: customer |
| *Flow of events* | 1. Ann lives in Boston and plans to have some fun in Las Vegas in May, and she just wants to look up the prices because she doesn’t have much savings. 2. She randomly chooses a departing date in May 5th and departing airport: BOS and the arrival airport: LAS. She chooses one-way trip and coach seat. 3. From all the available flights from BOS to LAS, she sorts the results by price because she wants to know whether she has enough money to cover the vacation. 4. She realized that even the cheapest price is way too expensive for her, so she cancels the search by clicking “cancel” and exit the system. |

## Functional Model - Use Case Model

### Initiate System

|  |  |
| --- | --- |
| **Name:** | Initiate System |
| **Actor:** | Customer |
| **Entry**  **Conditions:** | System is running. |
| **Flow of**  **Events:** | 1.The customer clicks on the app/ tries to enter the website.  2.The system shows front page in reasonable time response. |
| **Exit**  **Conditions:** | * The system displays front page and preferences for customers to choose. |

### Entry of search parameters

|  |  |
| --- | --- |
| **Name:** | Entry of Search Parameters |
| **Actor:** | Customer |
| **Entry**  **Conditions:** | Customer has opened the main website or app. |
| **Flow of**  **Events:** | 1. The customer identifies One-way. Then enter departure airport arrival airport, and departure date, seating class and departure time windows. 2. The customer identifies Round-trip. Then one more choice will be displayed to the customer identified as return date and return time windows. |
| **Exit**  **Conditions:** | * All parameters have been identified (except time windows which can be left empty). * The system listens that the customer clicks on the “Search” Button. |

### Getting flights from DB Server

|  |  |
| --- | --- |
| **Name:** | Getting flights from DB Server |
| **Actor:** | Database Server, customer |
| **Entry**  **Conditions:** | Customer clicked “search” after entering search parameters. |
| **Flow of**  **Events:** | 1. Customer clicks “search” 2. The system will process the customer’s search parameters to HTTP query. 3. Database Server get the parameters from the system. 4. Database Server send back the information based on the query from the system. 5. System processes the raw information including converting time zones and determining available seating. 6. System shows the list of satisfying flights to the customer (satisfying flights mean there are available seats in first class or coach seating, or both left for customers to choose from). |
| **Exit**  **Conditions:** | * Viable legs of flights found * No flights path found * Customer canceled search |

### Converting time zones

|  |  |
| --- | --- |
| **Name:** | Converting time zones |
| **Actor:** | Database Server |
| **Entry**  **Conditions:** | Database sends in flight information including GMT and longitude. |
| **Flow of**  **Events:** | 1. Database sends in flight information including GMT and longitude. 2. System get flight information from Database server including GMT and airport longitude. 3. System get the GMT offset from API by giving the longitude and latitude of the airport. 4. System calculate the local time using GMT offset and GMT time. |
| **Exit**  **Conditions:** | * The calculation of local time is completed. |

### Determining available seating

|  |  |
| --- | --- |
| **Name:** | Choosing available seating |
| **Actor:** | Customer, Database Server |
| **Entry**  **Conditions:** | There are already valid results from previous flight search. |
| **Flow of**  **Events:** | 1. System looked for occupied seating number (the seat class is already identified by the user) for each flight (to trip and return trip separately) and make comparison with the total available seating for the airplane model. 2. Only the flights with available desired seating will be displayed to the customer. |
| **Exit**  **Conditions:** | * The flights are displayed in the screen. * No valid results-nothing is showed. |

### Sorting of flight data

|  |  |
| --- | --- |
| **Name:** | Sorting of flight data |
| **Actor:** | Customer |
| **Entry**  **Conditions:** | Valid flights are displayed, customer select one of the sorting buttons based on: price (low to high), departure time (earliest to latest), arrival time (earliest to latest), travel time (shortest to longest). |
| **Flow of**  **Events:** | 1. Customer select one of the sorting buttons based on: price, departure time, arrival time, travel time for both to trip and return trip (if round trip applied) respectively. 2. The system will use the flight information to sort the specific term and display the new sorted flight list to the customer. 3. If price: sorting the flights by low to high price. 4. If departure time: sorting the flights by earlier to late depature time. 5. If arrival time: sorting the flights by earlier to late arrivaltime. 6. If travel time: sorting the flights by travel time from least to most. |
| **Exit**  **Conditions:** | * Flight list is showed. * Customer cancel the sort condition. |

### Reserving a travel

|  |  |
| --- | --- |
| **Name:** | Reserving a travel |
| **Actor:** | Customer, Database Server |
| **Entry**  **Conditions:** | Customer click on the confirmation/ reserve button after he/she choose the desired legs of flight. |
| **Flow of**  **Events:** | 1.Customer click on the confirmation/ reserve button.  2.System gathers the flight numbers of all the legs in this flight.  3.System locks the Database Server.  4.System send the reserve query one by one.  5.System unlock the Database Server.  6.Customer will be acknowledged of successfully reserving the tickets. |
| **Exit**  **Conditions:** | * Reservation failed (database locked by others). * Reservation canceled by customer. * Reservation succeeded. |

### Error processing for no available flights

|  |  |
| --- | --- |
| **Name:** | Error processing for no available flights |
| **Actor:** | Database Server, Customer |
| **Entry**  **Conditions:** | After the customer entering all the needed parameters and clicks “search” button. |
| **Flow of**  **Events:** | 1. System looked up for available flights from database. 2. Database return a blank list or false information. 3. System recognize the blank list or false flags. 4. System displayed “No flight available found!” information to the customer. 5. System displayed a “Restart” button to the customer. |
| **Exit**  **Conditions:** | * Customer select “Restart” button. * Search canceled by customer. |

### Error processing for no requested seats for all legs of flight

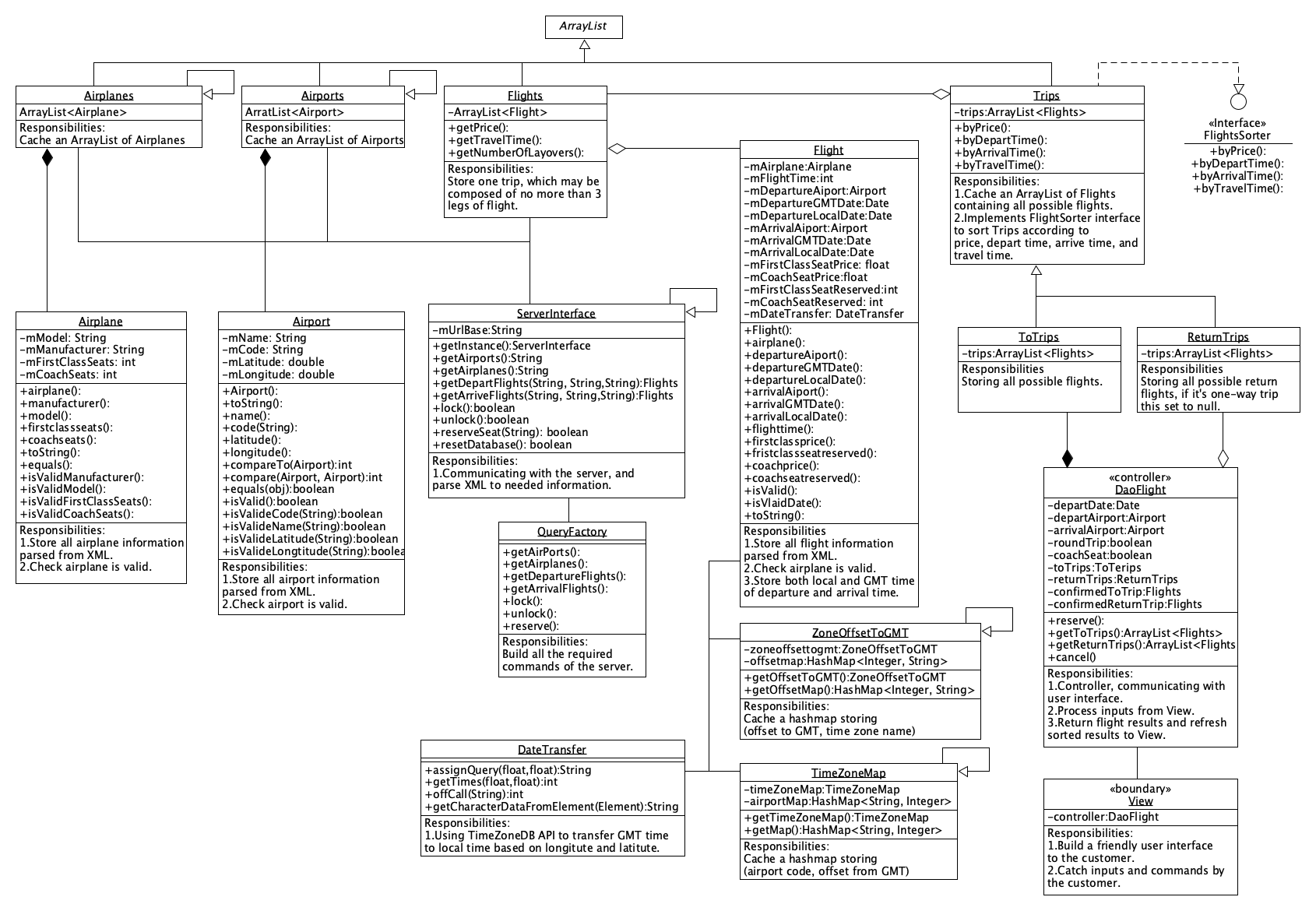
|  |  |
| --- | --- |
| **Name:** | Error processing for no requested seats for all legs of flight |
| **Actor:** | Customer, Database Server |
| **Entry**  **Conditions:** | There are already valid flight results for customer’s preferences. Customer wants to select first-class seating for all legs of flight on his/her one-way trip. |
| **Flow of**  **Events:** | 1.Customer select the first-class button.  2.System find no flight for all legs meets the requested class.  3.System shows “No available seat found, please choose another kind of seating or restart.” And a “Restart” button. |
| **Exit**  **Conditions:** | * Customer choose “coach seating”. * Customer click “Restart”. * Search canceled by customer. |

### Cancel before confirmation

|  |  |
| --- | --- |
| **Name:** | Cancel Before Confirmation |
| **Actor:** | Customer |
| **Entry**  **Conditions:** | There are valid flights based on his/her preferences for the customers to choose. Customer select “cancel” before reserving a flight. |
| **Flow of**  **Events:** | 1.There are valid flights based on his/her preferences for the customers to choose.  2.Customer clicks “reserve” for a desired flight.  3.System pop out a dialog with two buttons: “Confirm” and “Cancel”  4.Customer select “Cancel” button.  5.System returned to valid flights page as in step 1. |
| **Exit**  **Conditions:** | * Customer clicks “Cancel”. * Customer clicks “Confirm”. |

## Analysis Model – Object Model

## The picture below is the object model of the system.



# Requirements

## Functional Requirements

### World Plane Inc allow customers to specify departure airport, arrival airport, departure date, class type (coach seat or first-class seat), one-way trip/round-trip, which are all specified by customers. And the system will display all satisfying flights that are no more than 2 layovers and with each stopover time between 30 minutes and 2 hours.

### The system can transform the GMT time to local time according to airport’s longitude and latitude.

### The system can sort flights according to departure time, arrival time, flight time, price based on customer’s request.

### After the customer confirmed the desired legs of flights, the system will be able to update the information on the server side.

### The system may not display the flight if seats are not available for any legs of the flight.

### Customers can cancel the order before they confirmed it and they can exit the system.

## Nonfunctional Requirements

### Usability

* Customers shall be able to input and get the local time and local time zone but not unified GMT time.
* Reasonable and convenient UI shall be provided for customers.

### Reliability

### The system will be able to return information of ‘No seat available’ when requested seat is not available for all legs of flights.

### The system will be able to return the information of ‘server is busy’ to customer when database is locked by others.

### Performance

### Response time for any requested actions will reasonable. Operations in excess of 3 seconds will provide indication to the customer the system is operating.

### A request of customer shall be answered within seconds, and the system shall be able to process several requests at the same time.

### Accurate and complete results shall be output according to the customer request.

### Supportability

### The application will use the JAVA programming language for platform independence.

### The application shall be run on software environment and get the same performance.

### The application shall have the ability to be maintained to fix defects or add functions.

# Interaction diagram

### 

### This sequence diagram depicts the whole user case for customer to book ticket. Here we assume there is suitable flight for the customer’s request and there are remaining seats, and finally customer choose to book a seat. We just try to depict the whole procedure but not contain all the possibility such as unsuccessful user case.

### 

### This communication diagram depicts the same information with the above sequence diagram, one advantage of communication diagram is to remove the need for geometrical constraints on the objects and results in a more compact diagram.

# State Machine Diagram

### 

# Glossary

Definitions, Acronyms & Abbreviations

|  |  |
| --- | --- |
| *Reservation* | A seat on a specific flight specifying either “First Class” or “Economy” seating section of the plane. A reservation does not specify a particular seat number for the flight. |
| *One-way* | A one-way flight means a trip with a departure airport and an arrival airport. |
| *Round-trip* | Round-trip flights mean two trips: 1. To trip: departure airport to arrival airport. 2. Return trip: destination airport to arrival airport |
| *Stopover* | The airport connecting two flights. |
| *GMT* | Greenwich Mean Time |
| *First class* | Seating will be better and price higher. |
| *Coach seat* | Ordinary seating conditions with ordinary price. |
| *Concurrency* | Multiple customers will operate on the system at the same time but will not conflict with each other. |
| *Leg of flight* | From a certain departure airport to an arrival airport, there may not be a direct flight to choose. Several flights connecting from this specific departure airport to the arrival airport can also satisfies travel preferences. Each of these connecting flights are called leg of flight. |
| *Travel time* | The total time (including all legs of flights and layover time) spent on a trip from departure airport to arrival airport. |
| *Layover time* | A time period between 2 connecting flights, and is strictly between 30 minutes and 2.5 hours. |
| *Time window* | A time span for customers to choose, including a start time and an end time (they should be in the same day), the flight time of the first leg of flights will be limited to after this start time and before this end time. |