Requirements Analysis Document (RAD)

Prepared for

World Plane, Inc. (WPI)

Prepared by

Team C

Akaash Varatharajan, Evan Arenburg, Gokul Srinivasan, and Joseph Lombardi.

May 1st, 2023

V3 - Final

Requirements Analysis Document (RAD)

1 Introduction

1.1 Purpose of the system

Team C will be delivering to World Plane Incorporated a proof-of-concept software that provides Retail Customers with the ability to book their Flight Reservations, instead of calling a Travel Agent. The software will utilize the current World Plane Incorporated database to search for flights that match search queries entered by Retail Customers through Team C's software. The software shall provide advanced functionalities including, but not limited to, displaying the Retail Customer suitable layover options for travel, converting local and destination time, and searching within a window of departure and arrival dates [1]. By using the software delivered, World Plane Incorporated will be able to expand its customer base and reduce overhead costs on Travel Agents [2].

1.2 References

1. Statement of Work (SOW) provided by World Plane, Inc.
2. SOW Introduction Video, Ekalb Noslen, Grand Poohbah & CEO of World Plane Inc.
3. Meeting with Eklab Noslen, 3/2/2023.
4. Meeting with Eklab Noslen, 4/17/2023.

1.3 Scope of the system

The airline ticket booking system facilitates the process of booking airline tickets the customers online. Using this software, customers can search for flights, choose their itinerary based on the price, select their preferred category of seat, and reserve their flights for travel. The software provides real-time updates on prices and availability of flights and seatings thereby enabling customers to plan their travel plans accordingly.

1.4 Core System Functionalities

The software delivered by Team C will primarily act as a ticket reservation system, that performs the same functionalities that Travel Agents do currently for World Plane Incorporated. The software will be able to search a database for flights, find flights that fit a user query and take the user up through reserving a ticket. The software will provide alternative flight options that a Travel Agent would also describe to Retail Customers, such as searching for flights within a window of departure and arrival dates, choosing between layover options, and selecting seating within the First Class or Coach categories. The system will act as an interface for Retail Customers to interact constructively and intuitively with the current WPI database.

1.5 Objectives and Success Criteria of the Project

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

- A user will be able to book a flight using the software for each applicable leg of the flight. One-way and round-trip flights will also be able to be booked by the user. The user will select an arrival and departure airport, and will be shown a list of possible flights that satisfy their criteria.
- The software will ensure that the arrival and departure times for each leg of the flight are only allowed to be booked if the times are realistic. The user can only book a flight if there are sufficient layover times in between the legs of the flight if applicable.
- Times will be displayed to the user in their local time.
- When the user books a flight, they will be able to reserve either first-class seating or coach seating for each leg of the flight if the seating is available. The software will let the user know if their seating choice is not available on certain legs of the flight so the user can select accordingly.
- The user will be able to search for flights using departure date, arrival date, or price.
- The user will be able to sort connection flights based on travel time.
- The price of the user's selected flight will be displayed to them by the software.
- The user will be able to select flights and confirm the selection before the reservation is booked.

2 Current System

2.1 Existing System

The current Flight Reservation system implemented by World Plane Incorporated is made up of many Travel Agents that make reservations for Retail Customers who phone in their requests. Retail Customers state to the Travel Agent the place they want to leave from, the location they want to travel to, and other pertinent flight information such as whether the customers want to book a one-way flight or a round-trip. The Travel Agents act as consultants for customers.

World Plane Incorporated wishes to downsize the amount of Travel Agents they have, by making it possible for Retail Customers to book their flights using an application on the internet.

2.2 Current Operations

To book a flight a Retail Customer calls World Plane Inc., and the call is directed to a Travel Agent. The Retail Customer gives the Travel Agent the details of their desired trip. Details would include their intended area or airport of departure, their intended arrival area or airport, scheduling information on when they want to take the flight, their desired seating class, and whether the trip would be a one-way flight or a round-trip. The Travel Agent then searches for possible flights that match the Retail Customer's query and provides options that fit. [3] Fundamentally, working with World Plane Incorporated Travel Agents to book a trip is a conversation with a back-and-forth of queries, sorting, and finally booking a flight.

3 Proposed System

3.1 Overview

The proposed system will perform the functions of a Travel Agent, and eventually be able to shunt customer load from Travel Agents to the proposed system. The system will provide a front for Retail Customers to interface with World Plane Incorporated's database so that customers can search for flights, see details about flights, and reserve flights.

3.2 Conceptual Model - User Scenarios

<u>Scenario 1: The Budgeter</u>
John Doe is trying to travel on a budget. He works remotely, so he has a lot of flexibility in his schedule. He is an avid vacationer with a penchant for pinching pennies, which is in turn the reason he can travel so much. John wants to go to Las Vegas, NV, because it is very cold in Manchester, NH. John loads the Flight Application and searches for flights that depart from his home of Manchester with airport MHT, and arrive in Las Vegas at LAS. He specifies a window of dates that he would be comfortable departing in. Since winter has just started and he loathes the cold, he does not specify a return date. Flights populate, and he finds that after sorting by price, some flights with layovers at odd times during the night are cheaper than direct flights. None of the layovers seem to be unreasonably long, so he books a coach seat on the cheapest one.

<u>Scenario 2: The Work Trip</u>
Martha Moss is a New York City-based businessperson with an important meeting in Boston, Massachusetts during the upcoming week. She does not have enough time to take the bus, train, or drive her car, and needs to book a round-trip flight for a single day. She opens the Flight Application and chooses to depart from LGA and arrive in BOS, and then return from BOS to LGA on the same day. The Flight Application returns a list of applicable flights and their prices. She sorts the departing flights by arrival time to be certain that she will make it in time for her meeting, and sorts her return flight by departure time, to be sure that the meeting will have concluded before having to fly back to New York. She chooses to fly First Class to minimize the time spent embarking and disembarking the plane. After finding everything satisfactory and selecting the appropriate flights, she is shown the cost, and she makes a reservation.

3.3 Functional Model - Use Case Model

The following section communicates the functionality of Team C's proposed software system.

Name	Search for One Way Trip
Actor(s)	Retail Customer, Database
Precondition(s)	Retail Customer has just started the Flight Application and is located in the initial or 'home' state.
Flow of Events	1. Retail Customer acknowledges that the program is ready for input by reading a greeting message at the terminal. 2. Retail Customer selects "Plan a one way trip" by typing "one way" and pressing enter on their keyboard. 3. Software prints out the format for the search, including how to format terminals and dates. 4. Retail Customer enters their departure airport, and a date or window of dates to search through. The given date is assumed to be in the local time of the given airport code. 5. Retail Customer enters their arrival airport. 6. Software builds possible Trips according to the "Build Trips" use case that match the Retail Customer's request. 7. Search results are displayed to the Retail Customer at the terminal, sorted by default filter of price, according to the "Sort Trips by Price" use case.
Postcondition(s)	Returned search data is stored locally and available for Flight Application to sort through.
Quality Requirements	

Name	Search for Round Trip
Actor(s)	Retail Customer, Database
Precondition(s)	Retail Customer has just started the Flight Application and is located in the initial or 'home' state.
Flow of Events	1. Retail Customer acknowledges that the program is ready for input by reading a greeting message at the terminal. 2. Retail Customer selects "Plan a round trip" by typing "round trip" and pressing enter on their keyboard. 3. Software prints out the format for the search, including how to format terminals and dates. 4. Retail Customer enters their departure airport, and a date or window of dates to search through. The given date is assumed to be in the local time of the given airport code. 5. Retail Customer enters their arrival airport. 6. Retail Customer enters their return date or window of dates.

	7. Software builds possible Trips according to the “Build Round Trips” use case that match the Retail Customer’s request. 8. Search results are displayed to the Retail Customer at the terminal, sorted by default filter of price, according to the “Sort Trips by Price” use case.
Postcondition(s)	Returned search data is stored locally and available for Flight Application to sort through.
Quality Requirements	

Name:	Server Query
Actor:	Database
Preconditions:	Information is obtained from the retail customer.
Flow of Events:	1. The information given by the retail customer is used build queries and send a request from the client to the server, to which the server then responds with the requested information.
Postconditions:	The required information is acquired from the server.
Quality Requirements:	

Name	Build Trips
Actor(s)	Database
Precondition(s)	Software Application has a departure date or window of dates and an airport. Software Application has an arrival airport.
Flow of Events	1. Software performs “Build a Trip” use case on each date within the window. 2. Software performs “Build a Layover Trip” use case on each date within the window. 3. Software adds the Trips that were built in steps 1 and 2 to a new Trips object. 4. Software returns the Trips object with the flights that match the search.
Postcondition(s)	Software now has Trips object.
Quality Requirements	

Name	Build Round Trips
Actor(s)	Database

Precondition(s)	Software Application has a departure date or window of dates and an airport. Software Application has an arrival airport. Software Application has an arrival date or window of dates.
Flow of Events	1. Software performs “Build a Trip” and “Build a Layover Trip” use case on each date within the departure window. 2. Software performs “Build a Trip” and “Build a Layover Trip” use case on each date within the arrival window. 3. Software adds the Trips that were built in steps 1 and 2 to a new Trips object for arrival and departure Trips.
Postcondition(s)	Software now has Trips object.
Quality Requirements	Build must complete in a reasonable amount of time.

Name	Build a Layover Trip
Actor(s)	Database
Precondition(s)	Software Application has a departure date and an airport. Software Application has an arrival airport.
Flow of Events	1. Software performs “Search for Departing Flights” use case. 2. Software filters out direct flights (if any) 3. Software performs “Search for Departing Flights” use case on each flight returned from filter in Step 2. 4. Software filters flights returned by using “Search for Arriving Flights” on each set of flights returned from Step 3. 5. Software builds a Trip object out of each possible layover returned. 6. Software filters out long layovers using the “Filter Layovers” use case. 7. Software returns Trips object with applicable trips.
Postcondition(s)	Software now has Trips object that can be sorted.
Quality Requirements	

Name	Search for Departing Flights
Actor(s)	Database
Precondition(s)	Software Application has a departure date and an airport. Software assumes the given date is in local time.

Flow of Events	1. Software parses the input from the terminal. 2. Software calculates the next date to the given date. 3. Software generates a search call to the database to find flights for the given date and the next date. 4. Database returns a list of flights that depart the airport on the given date and the next day. 5. The software checks if the flights are departing between 00:00 hours and 23:59 hours on localtime of the given date. This is done only for the flight departing from the airport given by the user. 6. Software parses the XML into a Flights object.
Postcondition(s)	Returned search data is stored locally and available for Flight Application to sort through.
Quality Requirements	

Name	Search for Arriving Flights
Actor(s)	Database
Precondition(s)	Software Application has an arrival date and an airport. Software Application has a Flights object to search through. Software assumes the given date is in local time.
Flow of Events	1. Software calculates the next date to the given date. 2. Software generates a search call to the database to find flights for the given date and the next date. 3. Database returns a list of flights that depart the airport on the given date and the next day. 4. The software checks if the flights are departing between 00:00 hours and 23:59 hours on localtime of the given date. This is done only for the flight departing from the airport given by the user. 5. Software parses the XML into a Flights object.
Postcondition(s)	Program execution is turned over to the calling function.
Quality Requirements	

Name	Calculate layover time
Actor(s)	Database
Precondition(s)	The system has the flights within a Trip
Flow of Events	1. The system calculates the layover time using the arrival time of the first flight, and the departure time of the next flight of a Trip.
Postcondition(s)	The system returns the layover time.
Quality Requirements	

Name	Filter Layovers
Actor(s)	
Precondition(s)	The system has a list of Trips that have layovers.
Flow of Events	1. The system calculates the layover time according to the use case, “Calculate Layover Time”. 2. The system checks if the layover time is greater 2 hours. 3. The system removes Trips that are below the criteria in Step 3. 4. If no flights are available within that window, the system will return no flight found.
Postcondition(s)	The system returns Trips that fit the layover criteria.
Quality Requirements	The layover time should be equal to or greater than two hours.

Name	GMT to Local Time
Actor(s)	Database
Precondition(s)	The system has times and aiport code. The system has access to CSV file for airport time zones Time entered considered as local airport time
Flow of Events	1. The system obtains the time zones of each airport using the csv file. 2. The system converts the departure and arrival times into local time using the gmt offset, the timezone abbreviation of the given airport from the csv file. 3. The system displays the local time at arrival and departure airports
Postcondition(s)	Returned local time data is stored locally and available for Flight Application.
Quality Requirements	

Name	Display time
Actor(s)	Retail Customer, Database
Precondition(s)	The system has the arrival and departure times of the flight Access to GMT to local time function
Flow of Events	1. The system obtains the arrival and departure times from the database in GMT 2. The system calls the GMT to local time function using GMT time in the database as argument. 3. Displays the time

Postcondition(s)	GMT time from the server is converted correctly to local time
Quality Requirements	

Name	Sort flight details
Actor(s)	Retail Customer, Database
Precondition(s)	Retail Customer logged on to Flight Application Retail Customer has selected arrival and departure airports.
Flow of Events	1. The passenger selects the sorting criteria for the flights (price, arrival time, departure time, or travel time). 2. The system sorts the list of flights based on the criteria selected. 3. The system displays the sorted list of flights to the passenger.
Postcondition(s)	The system displays the sorted list of flights
Quality Requirements	The list of flights should be in a sorted manner

Name	See flight details
Actor(s)	Retail Customer, Database
Precondition(s)	Arrival and departure dates are given Arrival and departure airports are given List of available flights is presented in a sorted manner
Flow of Events	1. The retail customer selects a flight from the list of available flights 2. The system displays flight details (arrival and departure time, layover time, stops, and total travel time)
Postcondition(s)	The system displays the flight details and the retail customer reviews the flight details
Quality Requirements	The passenger should have the necessary details for travel

Name	Select Flight
Actor(s)	Retail Customer, Database
Precondition(s)	Flights are presented to the retail customer to select from.
Flow of Events	1. Retail Customer selects the flight they wish to travel in. 2. System responds in the form of confirmation. The data is loaded into the current flight builder.

Postcondition(s)	The system displays the flight details and the price.
Quality Requirements	

Name	Select Seating Class
Actor(s)	Retail Customer, Database
Precondition(s)	Retail Customer has selected the flight based on the presented details
Flow of Events	1. Retail Customer selects the seating class they wish to travel in. 2. System provides a response in the form of a confirmation or indication of an unavailable seat. 3. If a seating class is unavailable, the system provides alternative options.
Postcondition(s)	Customer receives confirmation of seating.
Quality Requirements	If the option is unavailable, the system should provide alternative options.

Name	Display Price
Actor(s)	Retail Customer, Database
Precondition(s)	The Retail Customer has selected a flight from the list of available flights for their given requirement.
Flow of Events	1. The system displays the price for the current flight selected.
Postcondition(s)	The price is displayed to the Retail Customer.
Quality Requirements	

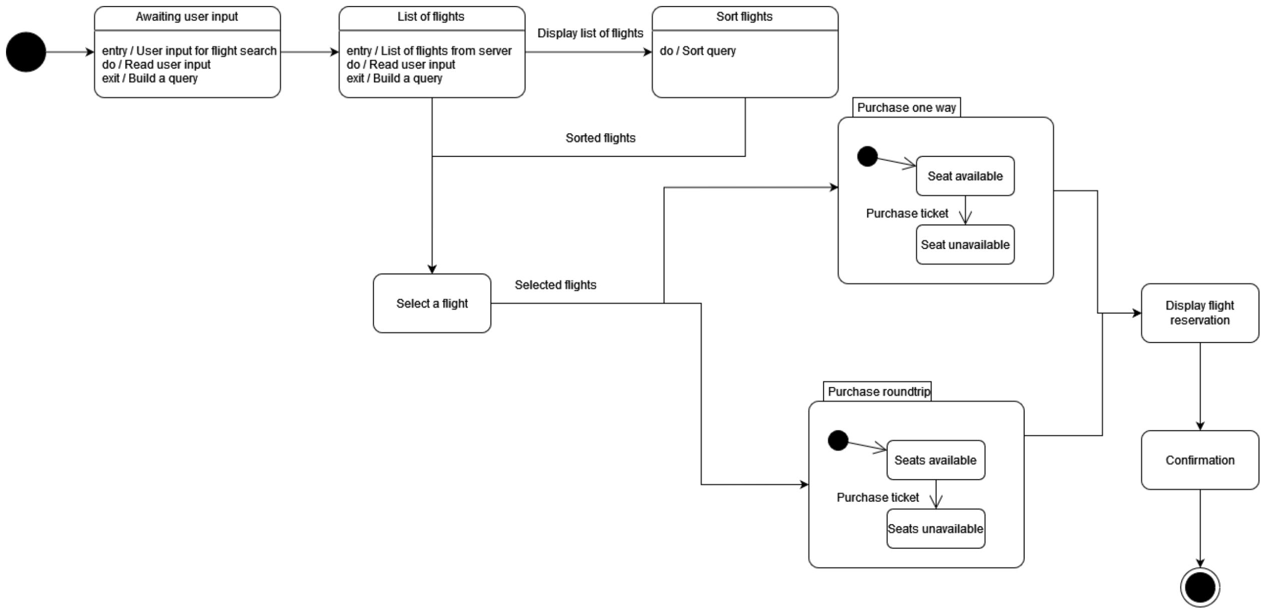
Name:	Reserve Trip
Actor:	Retail Customer, Database
Preconditions:	The Retail Customer has selected a flight and its seating class for each leg.
Flow of Events:	1. The Retail Customer requests to reserve a flight with their selected options. 2. The system validates that this is a valid reservation and allows the Retail Customer to continue to the confirmation unless the reservation is not valid, then the system alerts the Retail Customer of the issue and does not allow them to proceed.
Postconditions:	The system has brought the Retail Customer to the confirmation screen, or the Retail Customer has been informed of the issue.

Quality Requirements:	

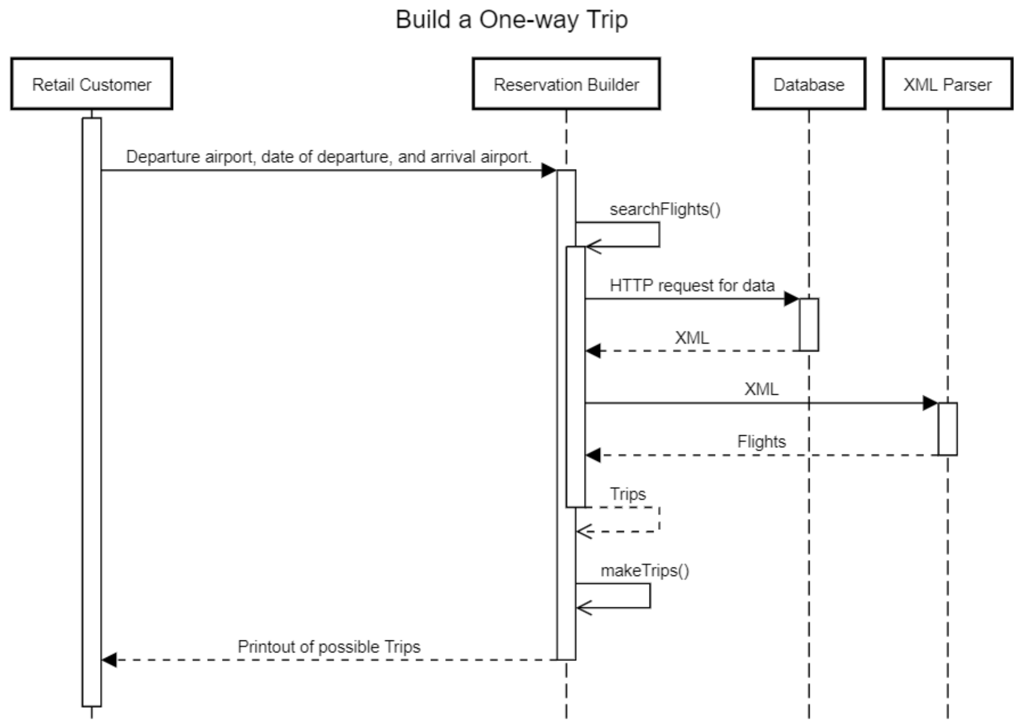
Name:	Confirm Selection
Actor:	Retail Customer, Database
Preconditions:	The Retail Customer has requested to reserve a flight.
Flow of Events:	1. The Retail Customer selects to confirm their reservation. 2. The system first locks the database to prevent different clients from booking the same flight 3. The system logs this reservation into the database. 4. The system unlocks the database so that the database can be used again. 5. The system reports back to the Retail Customer that the reservation was booked successfully. The system will alert the Retail Customer if any errors occur.
Postconditions:	The Retail Customer is informed that their reservation has been successfully booked, or the Retail Customer is informed of the issue.
Quality Requirements:	

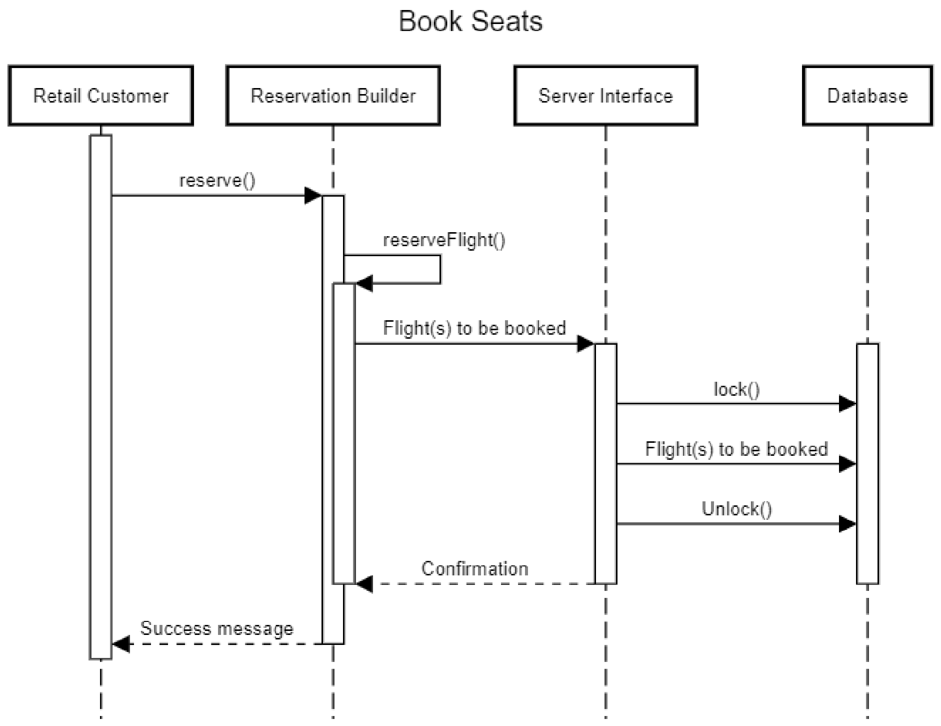
Name:	Parse XML
Actor:	Database
Preconditions:	Information is retrieved from the Server
Flow of Events:	1. Information is retrieved from the server in XML format. 2. This XML response is parsed to make it readable for the retail customer.
Postconditions:	The parsed XML is displayed to the Retail Customer.
Quality Requirements:	

3.4 Statechart Diagram

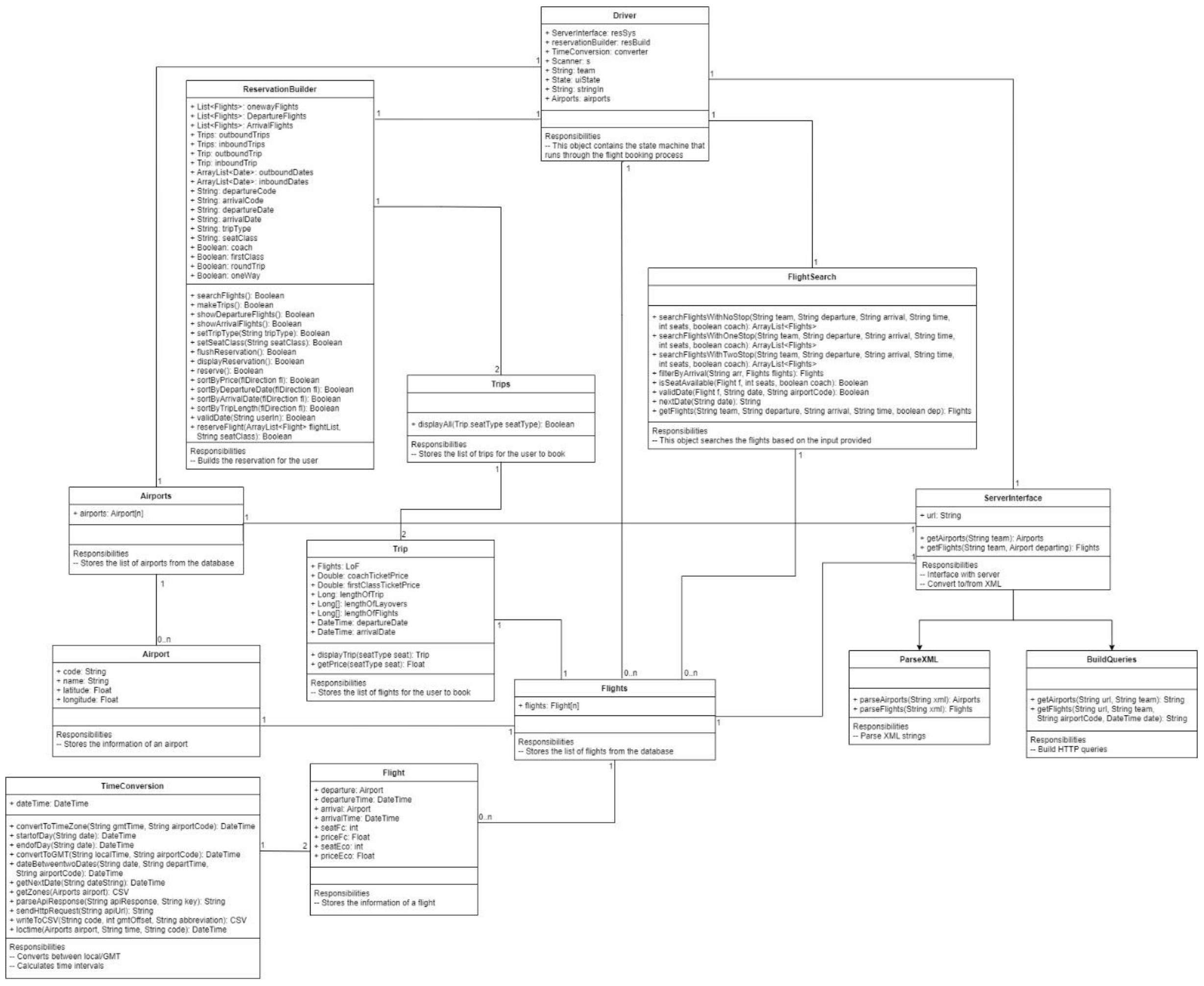


3.5 Behavioral Diagrams





3.6 Analysis Object Model


Driver
+ ServerInterface: resSys
+ reservationBuilder: resBuild
+ TimeConversion: converter
+ Scanner: s
+ String: team
+ State: uiState
+ String: stringIn
+ Airports: airports
Responsibilities
-- This object contains the state machine that runs through the flight booking process
ReservationBuilder
+ List<Flights>: onewayFlights
+ List<Flights>: DepartureFlights
+ List<Flights>: ArrivalFlights
+ Trips: outboundTrips
+ Trips: inboundTrips
+ Trip: outboundTrip
+ Trip: inboundTrip
+ ArrayList<Date>: outboundDates
+ ArrayList<Date>: inboundDates
+ String: departureCode
+ String: arrivalCode
+ String: departureDate
+ String: arrivalDate
+ String: tripType
+ String: seatClass
+ Boolean: coach
+ Boolean: firstClass
+ Boolean: roundTrip
+ Boolean: oneWay
+ searchFlights(): Boolean
+ makeTrips(): Boolean
+ showDepartureFlights(): Boolean
+ showArrivalFlights(): Boolean
+ setTripType(String tripType): Boolean
+ setSeatClass(String seatClass): Boolean
+ flushReservation(): Boolean
+ displayReservation(): Boolean
+ reserve(): Boolean
+ sortByPrice(flDirection fl): Boolean
+ sortByDepartureDate(flDirection fl): Boolean
+ sortByArrivalDate(flDirection fl): Boolean
+ sortByTripLength(flDirection fl): Boolean
+ validDate(String userIn): Boolean
+ reserveFlight(ArrayList<Flight> flightList, String seatClass): Boolean
Responsibilities
-- Builds the reservation for the user
Trips
+ displayAll(Trip.seatType seatType): Boolean
Responsibilities
-- Stores the list of trips for the user to book
FlightSearch
+ searchFlightsWithNoStop(String team, String departure, String arrival, String time, int seats, boolean coach): ArrayList<Flights>
+ searchFlightsWithOneStop(String team, String departure, String arrival, String time, int seats, boolean coach): ArrayList<Flights>
+ searchFlightsWithTwoStop(String team, String departure, String arrival, String time, int seats, boolean coach): ArrayList<Flights>
+ filterByArrival(String arr, Flights flights): Flights
+ isSeatAvailable(Flight f, int seats, boolean coach): Boolean
+ validDate(Flight f, String date, String airportCode): Boolean
+ nextDate(String date): String
+ getFlights(String team, String departure, String arrival, String time, boolean dep): Flights
Responsibilities
-- This object searches the flights based on the input provided
Airports
+ airports: Airport[n]
Responsibilities
-- Stores the list of airports from the database
ServerInterface
+ url: String
+ getAirports(String team): Airports
+ getFlights(String team, Airport departing): Flights
Responsibilities
-- Interface with server
-- Convert to/from XML
Trip
+ Flights: LoF
+ Double: coachTicketPrice
+ Double: firstClassTicketPrice
+ Long: lengthOfTrip
+ Long[]: lengthOfLayovers
+ Long[]: lengthOfFlights
+ DateTime: departureDate
+ DateTime: arrivalDate
+ displayTrip(seatType seat): Trip
+ getPrice(seatType seat): Float
Responsibilities
-- Stores the list of flights for the user to book
Airport
+ code: String
+ name: String
+ latitude: Float
+ longitude: Float
Responsibilities
-- Stores the information of an airport
Flights
+ flights: Flight[n]
Responsibilities
-- Stores the list of flights from the database
ParseXML
+ parseAirports(String xml): Airports
+ parseFlights(String xml): Flights
Responsibilities
-- Parse XML strings
BuildQueries
+ getAirports(String url, String team): String
+ getFlights(String url, String team, String airportCode, DateTime date): String
Responsibilities
-- Build HTTP queries
TimeConversion
+ dateTime: DateTime
+ convertToTimeZone(String gmtTime, String airportCode): DateTime
+ startofDay(String date): DateTime
+ endofDay(String date): DateTime
+ convertToGMT(String localTime, String airportCode): DateTime
+ dateBetweentwoDates(String date, String departTime, String airportCode): DateTime
+ getNextDate(String dateString): DateTime
+ getZones(Airports airport): CSV
+ parseApiResponse(String apiResponse, String key): String
+ sendHttpRequest(String apiUrl): String
+ writeToCSV(String code, int gmtOffset, String abbreviation): CSV
+ loctime(Airports airport, String time, String code): DateTime
Responsibilities
-- Converts between local/GMT
-- Calculates time intervals
Flight
+ departure: Airport
+ departureTime: DateTime
+ arrival: Airport
+ arrivalTime: DateTime
+ seatFc: int
+ priceFc: Float
+ seatEco: int
+ priceEco: Float
Responsibilities
-- Stores the information of a flight
1
2
0..n

4 Requirements

4.1 Functional Requirements

- Customers shall be able to specify the departure airport they wish to travel from and the arrival airport they wish to travel to.
- The Software must calculate layover time for trips with layovers.
- Departure and arrival times shall be shown in the local airport time
- The Software shall be able to reserve one-way or round Trips.
- Customers shall be able to reserve first-class or coach seating when selecting a flight.
- When seats are not available for a Customer to reserve, the software will not allow the customer to do so. Instead, the Software will throw an error message and return the user to their search.
- The Software will allow Customers to search for flights using departure and arrival dates and time windows between each flight.
- The Software shall sort Flights and Trips by price, arrival date and time, departure date and time, and travel time.
- The Software shall confirm the selection to the customer before a reservation is locked into the database.
- The Software shall convert server-side GMT to local/user time.
- The Software shall parse XML data located on the server representing flights and airports into other understandable forms of data.
- The Software shall use HTTP to interface with World Plane Incorporated’s Server.
- The Software shall lock the database before reserving a seat, and will unlock the database after the operation has completed.
    - Once the database is locked, only the Software will be able to make changes to the database.
- The program will provide for sufficient layover time to get from one gate to the next

4.2 Nonfunctional Requirements

4.2.1 Usability

- Customers shall be able to specify the departure airport and arrival airport from a list of possible airports provided by the system.
- A reservation will not be able to be modified or deleted once made.
- The software will not support payment processing.

4.2.2 Reliability

- All classes will be unit-tested using Junit test cases developed in parallel with application software

4.2.3 Performance

- Response time for any requested actions will reasonable. Operations over 3 seconds will indicate to the customer the system is operating.

4.2.4 Supportability

- The application will use the JAVA programming language for platform independence.
- Personal information will not be stored by the program.
- When unused, the database will automatically unlock after a specified period.

5 Glossary

Reservation	A seat on a specific flight specifies either the ‘First Class’ or ‘Coach’ seating section of the plane. A reservation does not specify a particular seat number for the flight.
Travel Agent	World Plane Incorporated’s current employees provide Retail Customers with Trip planning and Flight reserving services.
Retail Customer	A person logged in to the Client Software, searching for a Trip.
User	See *“Retail Customer”*
Trip	A travel itinerary that can be either One-Way or Round-Trip made up of Flights.
Flight	A planned route for an Airplane between two Airports after a certain duration of Flight Time.
Connecting Flight	A planned route for an Airplane that between a Layover and another airport after a certain duration of Flight Time.
Layover	A stop between the Departure Airport and the Arrival Airport of a Trip, due to the unavailability or undesirability of a direct flight.
One-Way	A Trip that is plotted from one Airport to another.
Round-Trip	A Trip is identified uniquely by its return to the arrival airport after a specified amount of time by a Retail Customer.
Client Software	The program provided by Team C to World Plane Incorporated fulfills the requirements identified in this document.
Airport	A transit hub for Airplanes and Passengers seeking to travel.
Gate	A numbered station at an Airport where an Airplane “docks” to load and unload passengers and cargo.
Airplane	A large vehicle, similar to a bus with wings, capable of using aerodynamic principles and high-powered engines to travel through the air carrying passengers and luggage from destination to destination safely.
Flight Time	The total duration of time that an airplane spends transiting between the Gates of the Departure and Arrival Airports
HTTP	Hypertext Transfer Protocol, an application layer protocol used to transfer data between networked devices.
XML	Extensible Markup Language, a format for files that store arbitrary data.
GMT	Greenwich Mean Time, an international standard for observation-based time.