Requirements Analysis Document (RAD)

Prepared for

World Plane, Inc. (WPI)

Prepared by

Team Dos

Colin Mettler, Joshua Castro

05.06.2025

v0.3

Major updates from v0.2: Section 1.3, 3.1, 3.4, 4.1

Requirements Analysis Document (RAD)

1 Introduction

1.1 Purpose of the system

World Plane, Inc. (WPI) is considering transitioning the travel reservation system to a Retail Customer airline system. The current system is manual and dated, so WPI is asking for a 'proof of concept' software application. The requested application should function similarly to an existing web-based airline reservation system, such as kayak.com or expedia.com. The desired application will not be full-scale and the behavior will be simple, but should function well enough to provide evidence that WPI's travel reservation system can be fully converted without loss of functionality or downtime.

1.2 References

- Statement of Work (SOW) provided by World Plane, Inc.
- Proof of Concept video introduction provided by Larz White
- RAD Template
- Test Plan Template

1.3 Scope of the system

This software will be browser-based with a GUI and should allow customers to select an origin and destination for travel. Users can select flights, make reservations, and view departure and arrival times of flights, including connections (which must be reasonably timed). Reservations can also be one-way or round-trip. The system should have different options for searching, including departure date, arrival dates, and time windows, as well as for sorting by time (arrival, departure, or duration). No user information should be entered or stored, nor should reservations be able to be deleted. Timely responses to user actions should be prioritized, and WPI has provided a sample database for testing purposes. The sample database will be accessed through a restful API which has documentation provided. After further evaluation of the provided database, there is no information provided for pricing, seat assignments, or seat classes. While we had initially imagined having access to this information, our project will be simplified. We have added a SeatsAvailable column to the testing database which is set to 100 by default and decrements every time a reservation for the specific flight is made.

1.4 Core System Functionalities

- Browser-based with GUI included
- Results will come from a single server database
- Personal information will not be stored
- Search
    - Users can search for flights by origin and destination airport

    - Results should display only realistic flight options with sufficient layover times
    - Each leg should be displayed with local airport arrival and departure times
- Sort
    - Flights can be displayed sorted by:
        - Departure time
        - Arrival time
        - Travel time
- Reserve
    - Users can reserve flights for either one-way or round-trip travel
    - Flights can be selected and must be confirmed for a reservation to be made
    - Reservations can not be canceled

1.5 Objectives and Success Criteria of the Project

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

- The primary objective will be to successfully deliver the above core functionalities
- The system should function in a timely manner
- The system should be scalable
- The system should provide a realistic proof-of-concept for an improvement over the existing manual travel reservation system
- The system should have robust error-checking and testing

2 Current System

2.1 Existing System

Currently, the reservation system in place is manual and dated. The Travel Agency system relies on users to communicate with agents to view flights and make reservations. This requires labor, which is costly, and the system is generally inconvenient for many users. In the existing system, the travel agent would search the database for flights connecting two locations on a date provided by the users over the phone. The improved system would remove the need for an agent and also would allow users to book through the web instead of over the phone. The underlying database will remain the same, but the process will be entirely different. The user can enter information themselves and search and sort the results provided from the database through the GUI system. Overall, the user experience will be enhanced and labor costs will be reduced by removing the need for phone agents.

2.2 Current Operations

In the current model of business, the problem of purchasing plane tickets is solved via human workers and manual database searches. Users must call a physical number to connect with a travel agent. This agent will ask the user for information about their trip and perform a manual search of the database for relevant flight options. The worker will then communicate these options to the user over the phone, which can be confusing especially when connections are included. Finally, the user needs to inform the agent of their desired flight and the agent will make the reservation. These workers are solely responsible for the coordination of the purchase of tickets. This is a labor-intensive and costly solution to the issue of ticket purchasing due to staffing and office requirements, as well as all associated utility and bureaucratic costs. This is

also a slow and inefficient system, with the availability of travel agents creating a limit on the frequency and amount of tickets being sold. Additionally, the customer is forced to book during a specific time when the agency is open and has available agents, and the customer is left at the mercy of the efficiency and skill of their assigned travel agent. Clients who are assigned travel agents with less efficiency might find their ticket-purchasing experience to be extended and drawn out. Even worse, clients who are assigned travel agents with less skill might find themselves on flights that do not match their trip specifications. Overall, the two biggest constraints are the human factor as a go-between connecting the user and the database, as well as the lack of visual implementation which would simplify communications.

3 Proposed System

3.1 Overview

Our proposed system will eliminate the human go-between and allow the user to directly access information from the flight database. All functionality from the existing system, such as searching for flights, sorting results, and making reservations should be kept, but the proposed system should allow the user to perform these actions themselves through the web without having to speak with a travel agent. While our system is simply a proof-of-concept, it should be scalable to a full-size implementation without major changes. Therefore, the proposed system must also be efficient and constructed as if it were a large-scale web application. The system that we have created satisfies all of the above goals while maintaining overall functionality. It should be able to be scaled by being connected to a full database and having hosting support incorporated.

3.2 Conceptual Model - User Scenarios

Scenario 1: Search (One-Way)

- Role: Customer
- Action: The customer enters a departure airport, a destination airport, and a departure date, and clicks "Search." This is because they want to book a flight.
- Result: The system retrieves and displays a list of available flights, with options for sorting.

Scenario 2: Search (Round-Trip)

- Role: Customer
- Action: The customer enters a departure airport, a destination airport, a departure date, and a return date, and clicks "Search." This is because they will want to book a round-trip flight.
- Result: The system retrieves and displays a list of available outbound and return flights, with options for sorting.

Scenario 3: Search (Multi-Leg)

- Role: Customer
- Action: The customer searches as described in scenario 1 or 2, but there are no direct flights available between the desired origin and destination, or the customer has reason to want a connecting flight (price or travel to a specific airport, for example).

- Result: The system displays flight options with layovers, including waiting time, and ensures the layover time is enough for baggage transfer and customer travel from arrival gate to departure gate.

Scenario 4: Sorting and Filtering Flight Search Results
- Role: Customer
- Action: After performing a flight search, the customer sorts results by departure time, arrival time, or total travel duration. The customer wants to find the most convenient flight based on their schedule.

Result: The system reorders the flight list dynamically based on the selected criteria.

Scenario 5: Selecting and Reserving a Flight
- Role: Customer
- Action: After searching for and finding a desired flight, the customer selects it and confirms their reservation.
- Result: The system stores the reservation (cannot be deleted) and communicates information to the user.

Scenario 6: Handling Multiple Clients Accessing the System Simultaneously
- Role: Multiple Customers
- Action: Several customers perform flight searches and make flight reservations at the same time. The system must handle concurrent users without conflicts.
- Result: The system correctly manages multiple requests. While there is no information provided about seat availability, we can hard-code a max number of tickets sold per flight which must not be exceeded.

Scenario 7: Ensuring Timely System Responses
- Role: Customer
- Action: The customer performs a flight search, selects a flight, and reserves a seat. The system should respond efficiently without delay.
- Result: The system processes searches and reservations within seconds, ensuring a smooth user experience.

3.3 Functional Model - Use Case Model

This section communicates the functionality the future system will provide and how the different functionalities (use-cases) relate to each other. These can be communicated using either the use case table or a use case diagram – or both.

The set of use cases, taken as a set, need to provide all functionality required to support the set of user scenarios identified in the conceptual model above.

Name:	Use Case 1: Managing Seat Assignments and Availability
Actor:	Actor: Customer
Entry Conditions:	Customer has accessed the system. Customer wants to book a flight.

Flow of Events:	1. Customer enters a departure airport, destination airport, and departure date. 2. Customer clicks "Search." 3. System retrieves available flights. 4. System displays flight options with sorting features.
Exit Conditions:	Customer sees a list of available flights.
Quality Requirements:	Search results load reasonably fast. System handles high user concurrency.

Name:	Use Case 2: Search for Round-Trip Flights
Actor:	Actor: Customer
Entry Conditions:	Customer has accessed the system. Customer wants to book a round-trip flight.
Flow of Events:	1. Customer enters a departure airport, destination airport, departure date, and return date. 2. Customer clicks "Search." 3. System retrieves available outbound and return flights. 4. System displays flight options with sorting features.
Exit Conditions:	Customer sees a list of available round-trip flights.
Quality Requirements:	Search results load reasonably fast. System handles high user concurrency.

Name:	Use Case 3: Search for Multi-Leg Flights
Actor:	Actor: Customer
Entry Conditions:	Customer wants to book a flight. No direct flights are available, or customer prefers a connecting flight.
Flow of Events:	1. Customer searches as described in Use Case 1 or 2. 2. System identifies no direct flights or user preference for a connecting flight. 3. System retrieves flight options with layovers. 4. System displays layover details, including wait times and gate transfer requirements.
Exit Conditions:	Customer sees a list of available multi-leg flights.

Quality Requirements:	Layover durations are sufficient for baggage transfer and passenger transit. System provides real-time availability and pricing.

Name:	Use Case 4: Sorting and Filtering Flight Search Results
Actor:	Actor: Customer
Entry Conditions:	Customer has performed a flight search. Customer wants to refine search results.
Flow of Events:	1. Customer applies sorting/filtering options. a. arrival time b. departure time c. travel time 2. System reorders the flight list dynamically based on selected criteria.
Exit Conditions:	Customer sees an updated list of flight options based on sorting preferences.
Quality Requirements:	Sorting updates occur reasonably fast. System ensures accuracy and consistency in results.

Name:	Use Case 5: Selecting and Reserving a Flight
Actor:	Actor: Customer
Entry Conditions:	Customer has performed a flight search and has found a desired flight.
Flow of Events:	1. Customer selects a flight. 2. Customer confirms reservation. 3. System stores the reservation permanently. 4. System decreases number of available seats.
Exit Conditions:	Flight is reserved and cannot be deleted.
Quality Requirements:	Number of seats available updates in real-time. Reservation process is secure and prevents overbooking.

Name:	Use Case 6: Handling Multiple Clients Accessing the System Simultaneously
Actor:	Actor: Multiple Customers
Entry Conditions:	Several customers are searching and booking flights simultaneously.
Flow of	1. Customers perform searches and make reservations. 2. System processes multiple requests concurrently.

Events:	3. System ensures seat reservations occur on a first-come, first-served basis.
Exit Conditions:	All reservations are handled correctly without conflicts.
Quality Requirements:	System prevents double-booking of seats. Performance remains stable under high traffic loads.

Name:	Use Case 7: Ensuring Timely System Responses
Actor:	Actor: Customer
Entry Conditions:	Customer performs a flight search, selects a flight, and reserves a seat.
Flow of Events:	1. System processes flight search request. 2. System processes flight selection. 3. System processes seat reservation.
Exit Conditions:	Customer completes booking smoothly without delays.
Quality Requirements:	System responds to searches and reservations within seconds. Booking confirmation is instant with minimal processing time.

3.4 Analysis Model – Object Model

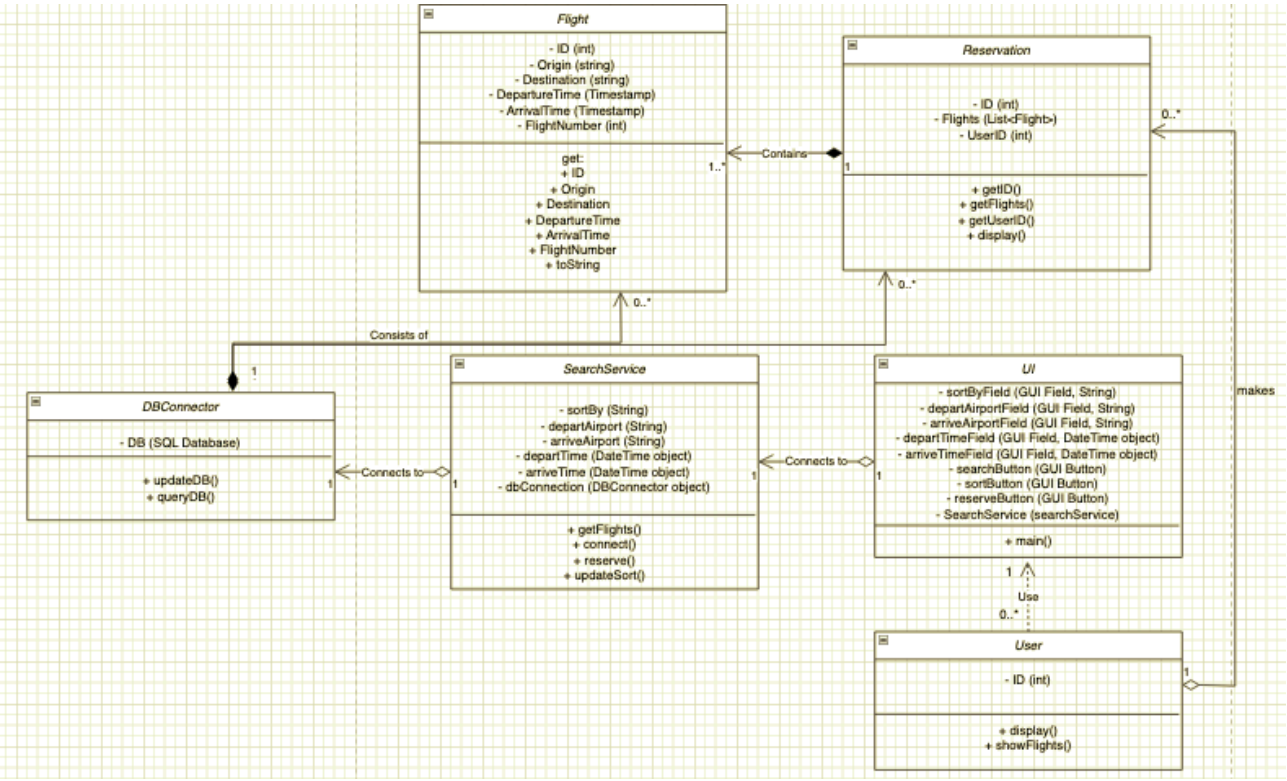
Classes/actions:

- User
- Flight
  - An individual flight with many attributes
    - ID, Number, Origin, Destination, Departure time, Arrival time, Duration
- Reservation
  - Contains flight selections (can be multiple) and booking details
  - Max number of reservations per flight is capped
- SearchService
  - Connects to UI, processes flight searches and reservations.
  - Ensures smooth user experience with quick response times.
- UI
  - Gets info on the screen from the user
  - Displays options for sorting and for finding flight to user
- DBConnector
  - Handles concurrent access from multiple customers, loads and saves DB

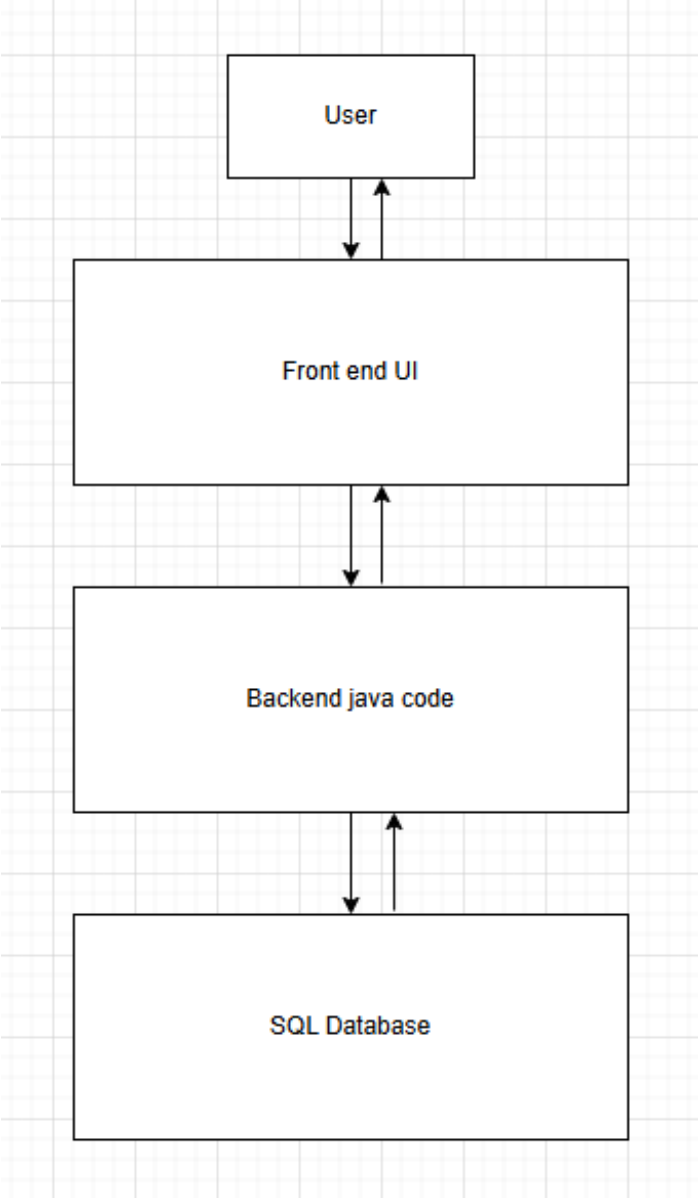
Relationships/Associations:

- The one UI is used by 0 to many users
- The one UI connects to one SearchService
- The one SearchService connects to the one DB
- The one DB has 0 to many reservations and 0 to many flights
- One customer can make zero or more reservations
- One reservation contains one or more flights

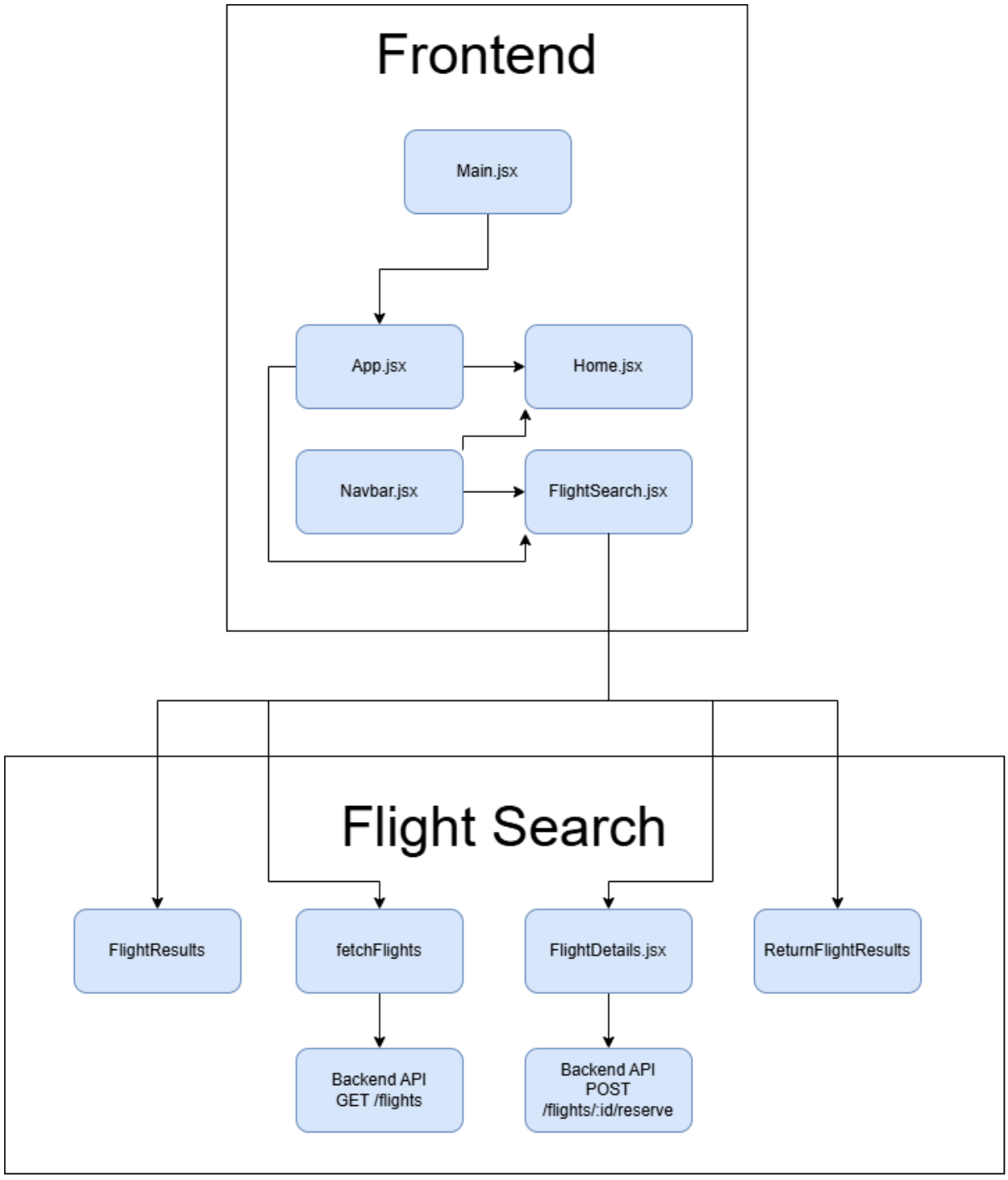
Class Diagram:



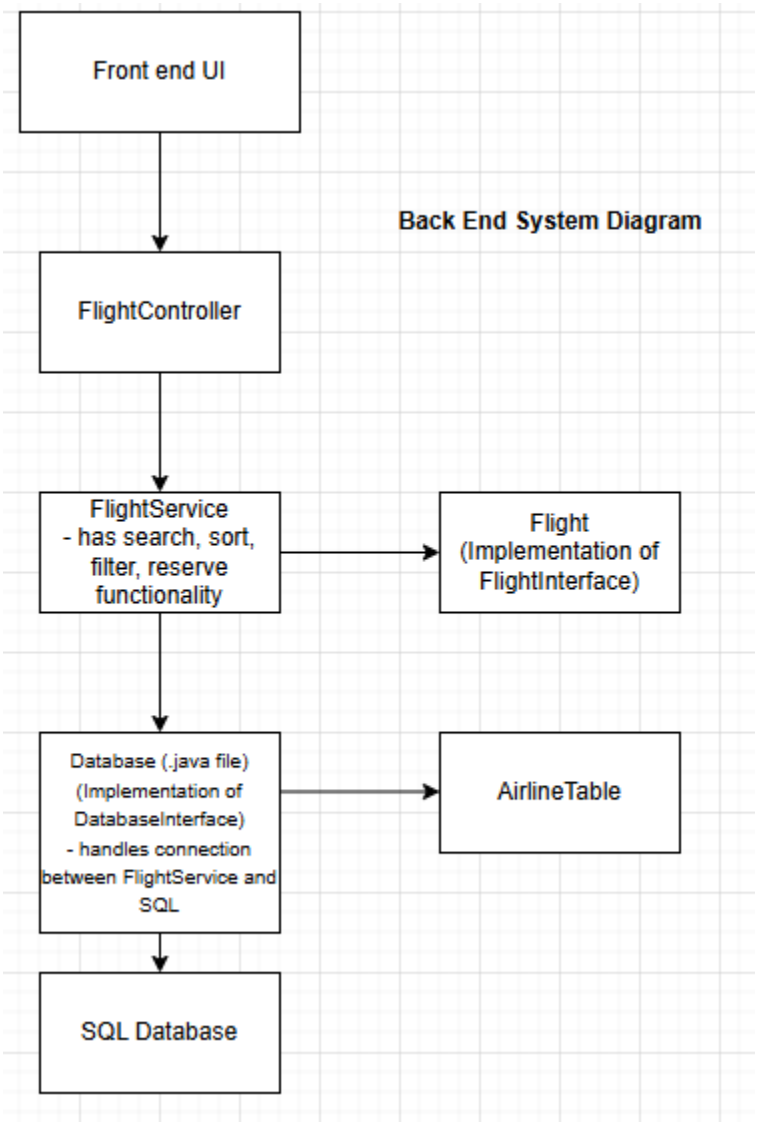
Systems Organization (Architecture) Diagram



Front End Systems Diagram



Back End Systems Diagram



4 Requirements

4.1 Functional Requirements

- Customers shall be able to search for flights within the database
- Customers shall be able to specify the destination airport they wish to travel to
- Customers shall be able to specify the departure airport they wish to travel from
- Customers shall be able to specify the maximum number of stopovers when traveling from departure to destination airports.
- Customers shall be able to specify if the trip is one-way or round-trip
- Customers shall be able to specify the return date (if round-trip)
- Customers shall be able to specify the time window in which the departure flight should take place
- Customers shall be able to specify the time window in which the return flight should take place
- Customers shall be able to specify the airline they wish to use
- Customers shall be able to sort results by departure time
- Customers shall be able to sort results by arrival time
- Customers shall be able to sort results by total travel duration
- The system shall retrieve available flights from the WPI flight database using a RESTful API
- The system shall ensure all flights displayed have sufficient layover times for passenger and baggage transfer
- The system shall display the departure time in local airport time
- The system shall display the arrival time in local airport time
- The system shall display the airline for each flight
- The system shall display the total number of stops and layovers for each flight
- The system shall display the total travel duration for each flight
- The system shall display available reservations (tickets) for each flight
- The system shall allow users to select a flight
- The system shall allow users to make a reservation
- The system shall require customer confirmation before finalizing reservation
- The system shall not allow cancellations once a reservation is made
- The system shall not store personal information of passengers

4.2 Nonfunctional Requirements

This section lists the constraints on the system. Nonfunctional requirements specify the way the system must fulfill the functional requirements above. They are usually categorized by usability, Reliability, Performance, and Supportability.

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.
- The system shall provide a browser-based GUI that is intuitive and easy to navigate.
- The system shall provide clear indications of available flights and seats

- The system shall provide clear indications of layover times

4.2.2 Reliability

- All classes will be unit tested using Junit test cases developed in parallel with application software
- The system shall maintain data integrity when multiple clients interact with the same database
- The system shall prevent duplicate or inconsistent flight reservations

4.2.3 Performance

- Response time for any requested actions will be reasonable. Operations in excess of 3 seconds will provide indication to the customer the system is operating.
- The system shall handle concurrent access from multiple clients without significant degradation in response time
- The system shall efficiently retrieve and display sorted flight options

4.2.4 Supportability

- The application will use the JAVA programming language for platform independence.
- Documentation for the APIs, including preconditions, post conditions, invariants and side effects will be provided to all development teams supporting this project.
- This proof of concept software shall be designed to support future enhancement and scalability with the possibility of evolving into a fully-fledged client software.

5 Glossary

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.
Flight	A scheduled journey from an origin airport to a destination airport operated by an airline
One-Way Ticket	A flight reservation for a single outbound flight without a return trip
Round-Trip Ticket	A flight reservation that includes both outbound and inbound flights
Single-Leg Flight	A flight reservation wherein the journey is directly from the departure airport to the destination airport
Multi-Leg Flight	A flight reservation consisting of multiple connecting flights, starting at the departure airport and ending at the destination airport but with one or more stops at airports in between.
Layover	The period of time a passenger spends at an intermediate airport between connecting flights.
Sorting Criteria	Parameters by which flights can be ordered.
Seat Assignment	The process of allocating specific seats to passengers via reservation
GUI	Graphical User Interface (GUI) is a visual interface allowing users to interact with the system via prompts, buttons, menus, and forms.
RESTful API	A web service interface that allows software applications to communicate over the internet using HTTP methods.
Scalability	The ability of a system to handle increases in user traffic and data, as well as overall system and software complexity, without significant degradation in performance.

Concurrency	The system's ability to handle multiple clients interacting with the same database, coordinating data changes.
Overbooking	The practice of selling more tickets than available seats
World Plane, inc.	The company for which the Airline Reservation System is being developed.
HTTP	HTTP (Hypertext Transfer Protocol) is a protocol for communication between web browsers and servers, used to transmit data over the internet.
Functional Requirement	A specific feature, capability, or behavior the system must have.
Non-Functional Requirement	A constraint or quality attribute that defines how a system should perform.
Usability	A measure of how easy and intuitive a system is for users.
Reliability	The ability of a system to perform its required functions without failure or degradation in performance, data quality, or result.
Performance	A measure of how efficiently a system responds to user interactions and workloads.
Use-Case	A scenario describing how a user interacts with a system to achieve a desired goal.