Software Requirements Specification

for

<Efficient Flight Itinerary>

Version <2.0 draft 1>

Prepared by

Group Name: <*West-Central Airlines*>

<Lance D.>	**<018>**	**<dever018@cougars.csusm.edu>**
<Bradley T.>	**<403>**	**<tran403@cougars.csusm.edu>**
<Lawrence S.>	**<NA>**	**<lawrencesantander.gmail.com>**

Instructor: <*Dr. Asif Imran*>

Course: <CS 441 – Software Engineering>

Lab Section: <*Tues/Thurs 4:00 – 5:15 PM*>

Date: <*10/21/2022*>

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft 2.0	Bradley T, Lance D, Lawrence S	Initial product SRS draft	10/21/22

1 Introduction

1.1 Document Purpose

This is SRS document draft 1, v2.0 constructed by West-Central Airlines software development team that examines/contains in detail the varying core aspects of the Software Requirements Specifications (SRS) such as an overview of our softwares design & purpose, use cases, and critical requirement specifications.

Being that our product is heavily software focused in functionality, the primary benefit of this document is the simplification and assistance it will provide towards an efficient development process that will also streamline the maintenance side of the software.

1.2 Product Scope

Development of the Efficient Flight Itinerary (EFI) software is to simplify the flight booking/management process so that customers/passengers have a convenient and user friendly experience when trying to find and book flights that will best suit their preferences. The system will be a user-friendly API that incorporates an algorithm designed to find a flight itinerary that best matches a customer's needs as pulled from a database of existing and available flights. Our database will be limited in scope to between two airport locations (Los Angeles, CA/Dallas, TX) and within a maximum booking time frame of a week for available flights so that overhead is kept to a manageable workload for a small scale team of three software developers.

1.3 Intended Audience and Document Overview

As this Efficient Flight Itinerary (EFI) software is being developed as a prototype for academic reasons within the boundaries permitted by CSUSM, this software will be utilized and shared for educational purposes with professional peers while being guided under the supervision of Instructor, Dr. Asif Imran. Outside of the boundaries of CSUSM as an educational institution of governance, EFI software upon completed development will be a beneficial service for customers and flight management teams.

1.4 Definitions, Acronyms and Abbreviations

- API - Application Programming Interface
- DFW - Dallas Fort Worth International Airport
- EFI - Efficient Flight Itinerary
- LAX - Los Angeles International Airport
- SRS - Software Requirements Specification
- TBD - To Be Determined

2 Overall Description

2.1 Product Overview

Our Efficient Flight Itinerary system provides a service that simplifies and manages flight choices most applicable to a passengers predetermined itinerary & preferences. This product is self-contained as EFI is a software that will be able to function independently without the need for support from any other application. Plans for this product are that it will be an improvement to current existing flight management systems by allowing users to input specialized/specific preferences to better pinpoint flights that most suit their needs related to factors such as time, cost, and layover tolerances.

Provided through Figure 2.1 is a high level illustration of how users and our product interact with one another:

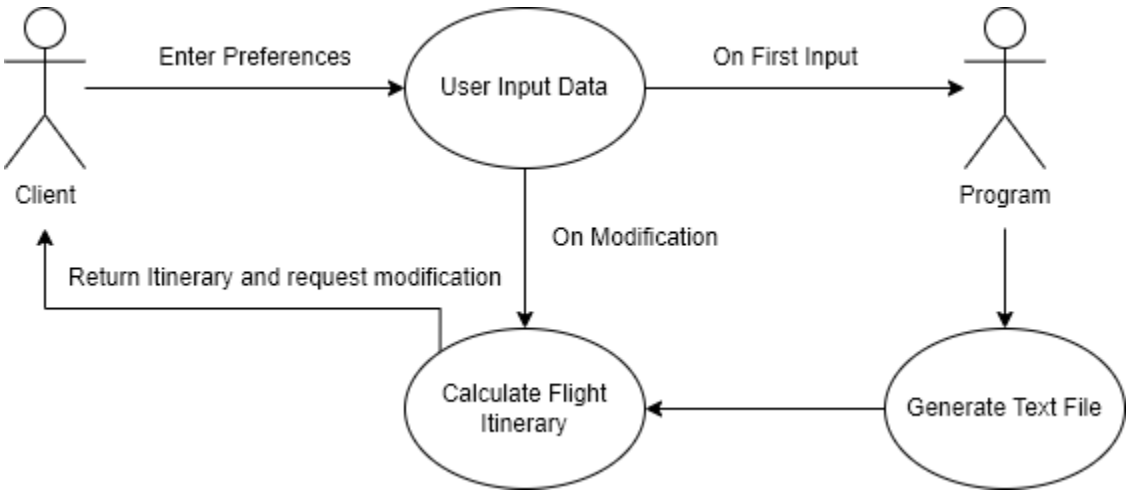


Figure 2.1: General Product Interaction Diagram

2.2 Product Functionality

1. Algorithm that calculates an efficient itinerary for a customer
    1.1. Weighted by user inputs
2. User-friendly operation and display
3. Integrate within our website
4. Modifiable after calculation
    4.1. Requests if user would like to modify original inputs

4.2. Offers recommendations or secondary options

5. Above requirements are ordered by priority

2.3 Design and Implementation Constraints

Users/Personnel:

- Users that would like to use our system will require access to the internet and a operating device
- Small scale Project requires small scale team for development.

Data:

- Only consider West-Central Airlines flights between two airports (LAX, DFW).
- Only confer a database of flights compiled from strictly a 7-day period (1 week).
- With these hard set constraints, this will allow for a manageable overhead of data.

System:

- Limited integrability due to being constrained by company and geographical area (by design to keep overhead manageable).

2.4 Assumptions and Dependencies

With the vast number of flights that occur from a multitude of locations, it is assumed that our data for testing purposes will be sourced from a precompiled list of flights between Dallas, TX and Los Angeles, CA. It is also assumed that these flights will only be within a one week timeframe, no more and no less. The week of flights to compile a data bank from is still TBD.

This will be a suitable volume of manageable data to handle for a small scale team of three developers. Additional assumptions will be that the passenger/user is familiar with handling a keyboard and mouse to utilize our service.

3 Specific Requirements

3.1 External Interface Requirements

The only requirement of this type is the program being interoperable with the existing website of the client.

3.2 Functional Requirements

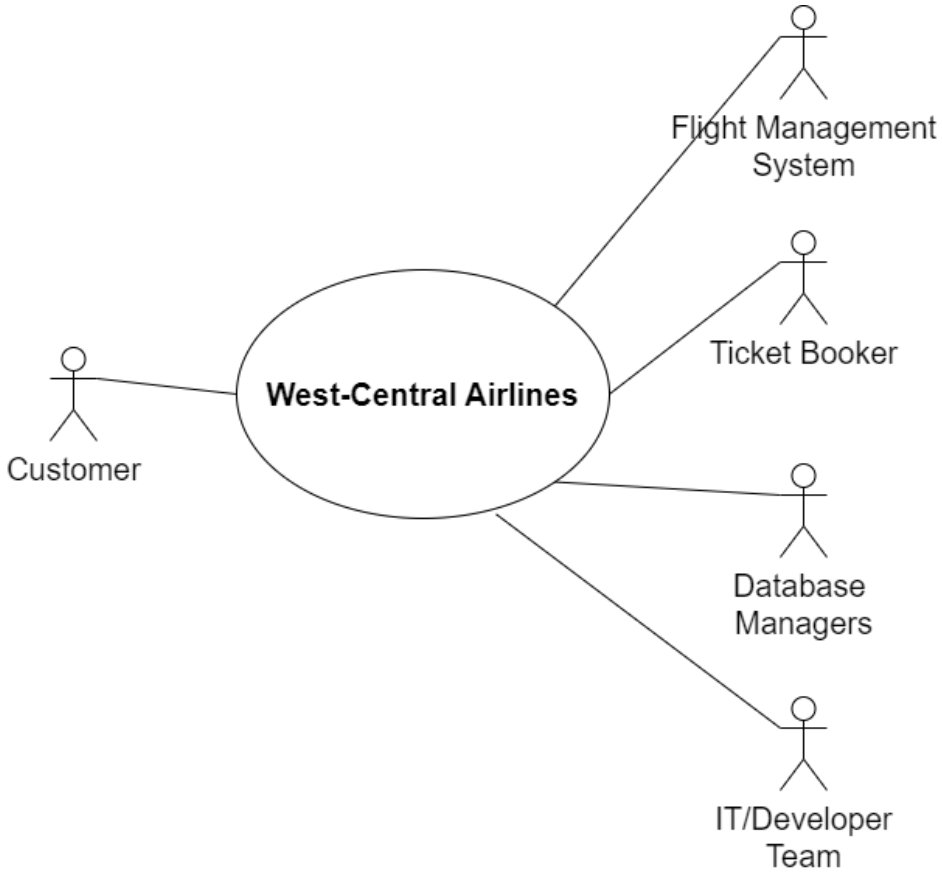
3.2.1 F1: The system shall …

- Take in a number of user inputs to set as the preferences
    - Fixed inputs for each use case will be the origin and destination ports and whether the flight is one or two-way
    - Preferences should be max acceptable numbers of transfers as well as all other inputs as numbers on a 1-5 scale
- Export the preferences as a text file for further use
- Take exported preferences to an integrated algorithm to generate an itinerary
    - Algorithm should be a matrix that draws upon imputed preference weights and compare to available options in flight database
    - Database should be generated going between LA and Dallas airports with minor airports as options for layover flights
        - Time span should be limited to a week
- Display the generated itinerary to the user
- Allow the user to edit preferences after being displayed itinerary until satisfied with result
- Allow user to save the itinerary as a file

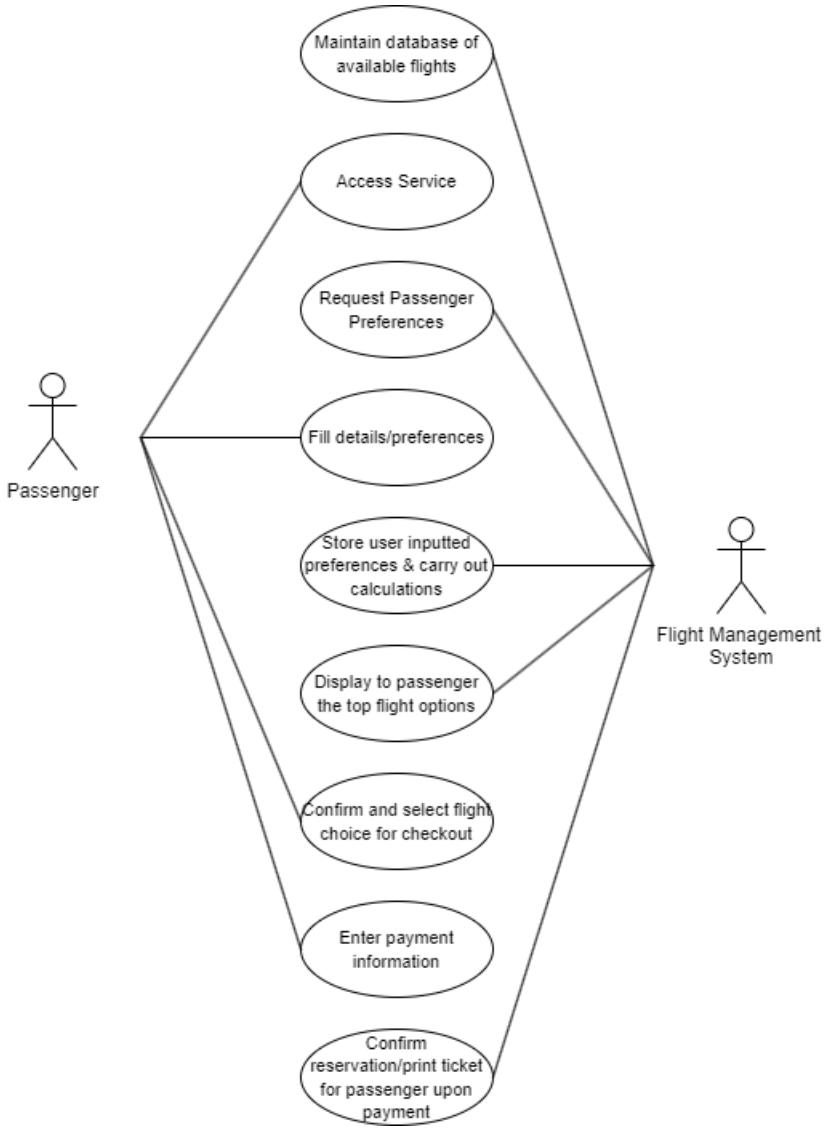
3.3 Use Case Models

For this section, varying Use Case Scenarios will be evaluated through the provided diagrams: Use Case Level 0 & Use Case Level 1 for West-Central Airlines Efficient Flight Itinerary (EFI).

3.3.1 Use Case Diagram (Level 0)



3.3.2 Use Case Diagram (Level 1)



3.3.3 Use Case #1

Author – Lawrence Santander

Purpose - Using the software systems to build and suggest flight itineraries to users

Requirements Traceability – Identify all requirements traced to this use case

Priority - High priority

Preconditions - System must be connected to a database of flights between the two airports

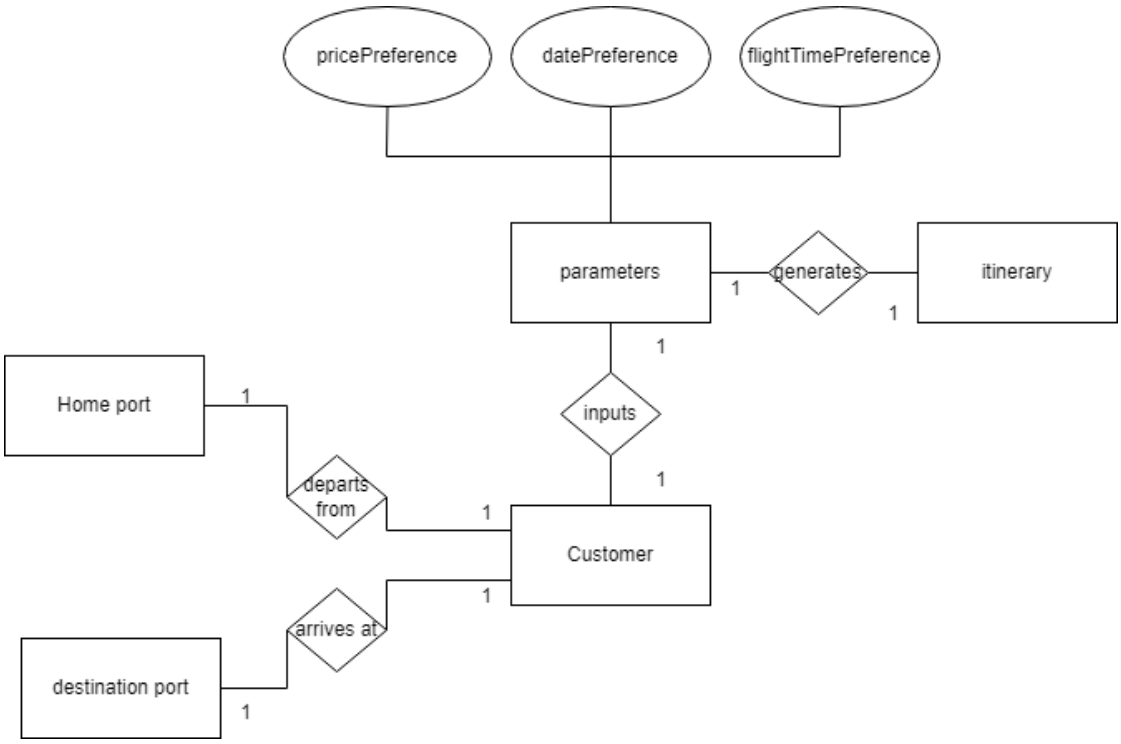
Post conditions - Flight itinerary can be edited afterwards with new preferences

Actors – Client, Software System

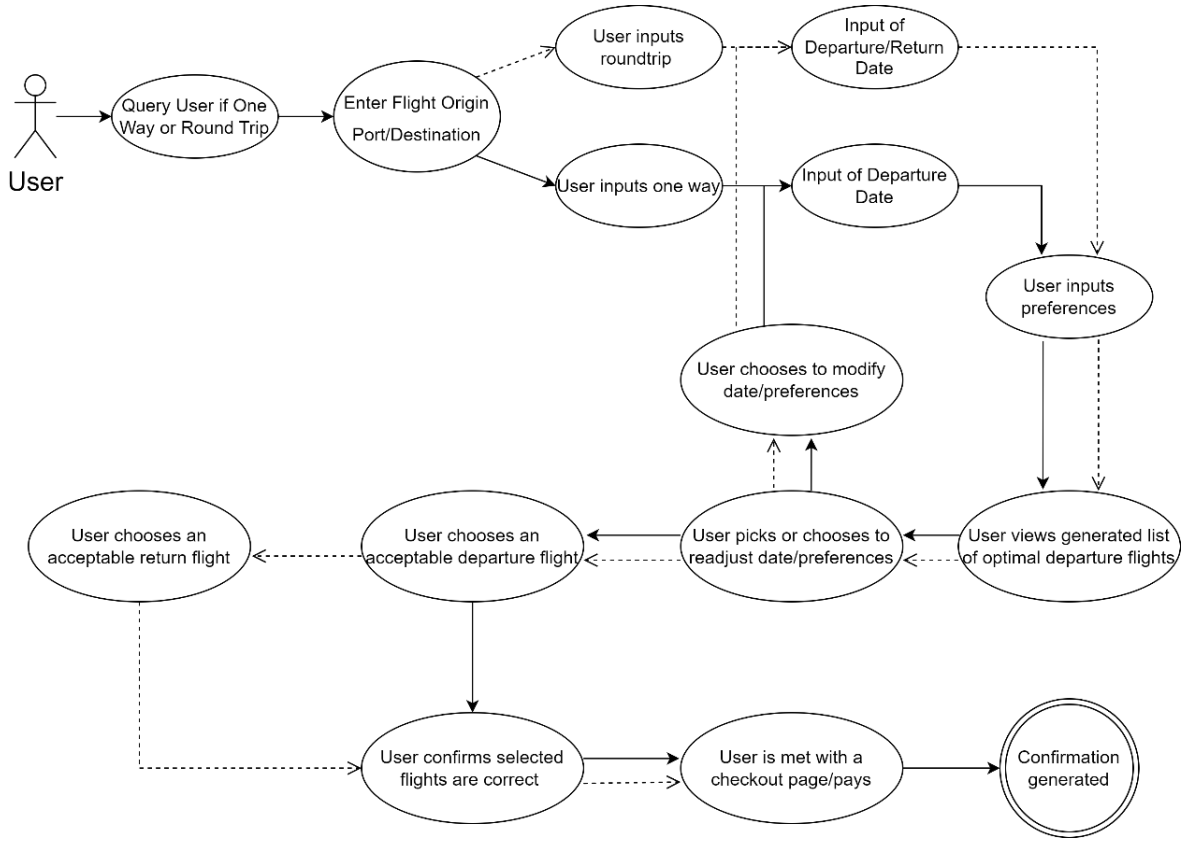
Flow of Events

1. User inputs flight preferences
2. Systems uses the user inputted preferences to create a flight itinerary to give to user
3. User can state their satisfaction with the flight itinerary to possibly generate a new/better one

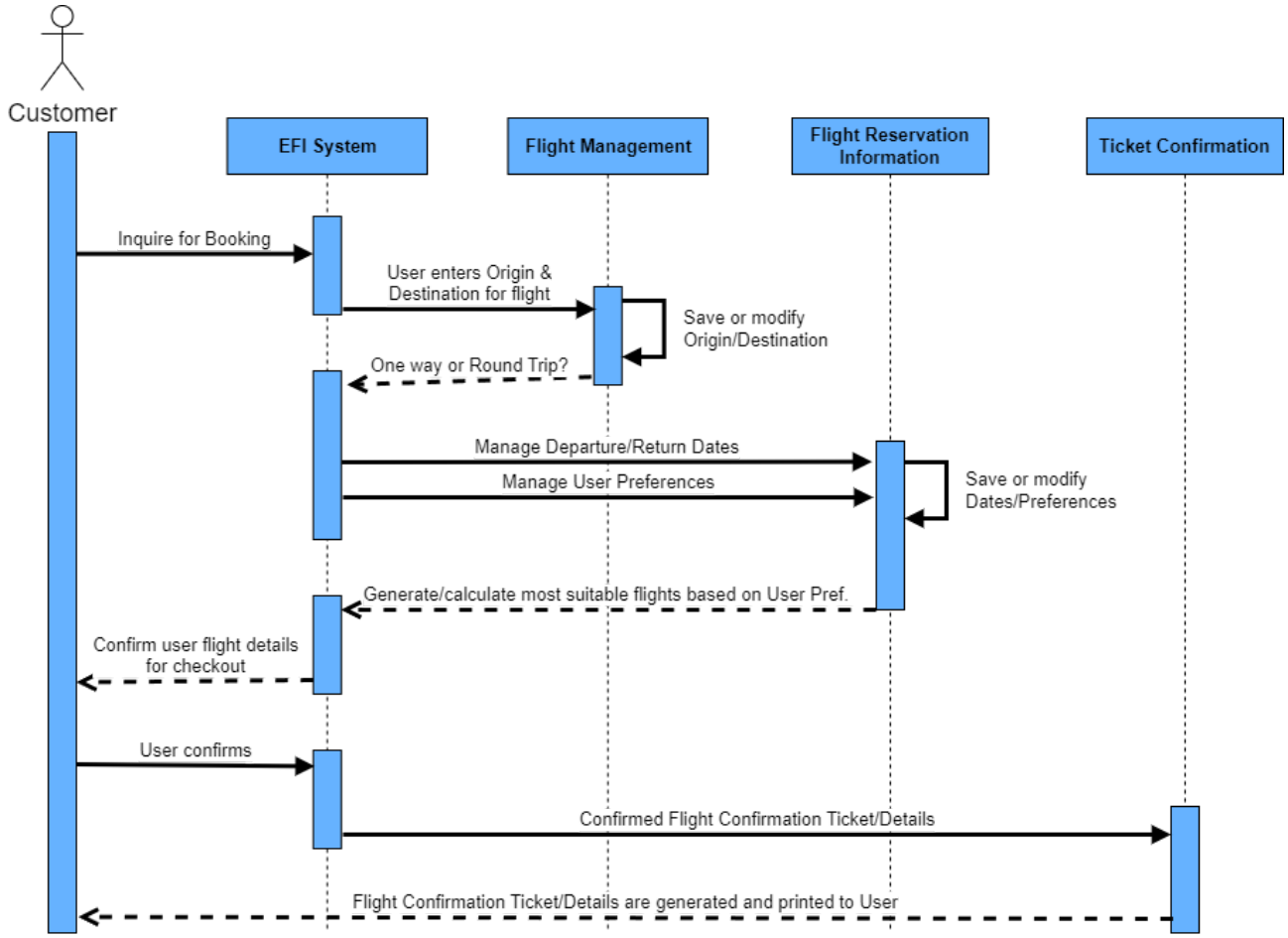
3.4 Activity Diagram



3.5 Flow Chart Diagram



3.6 Sequence Diagram



4 Other Non-functional Requirements

4.1 Performance Requirements

- The software is expected to be able to perform its functions within its section of the host website
- The software should have an uptime identical to that of the website it will service (24/7 except for scheduled maintenance)
- Software should be able to run smoothly and quickly across all major operating systems and their browsers (Windows, iOS)
- Software should be able to easily accessible and operable from the most common devices and platforms (desktops and smartphones)

4.2 Safety and Security Requirements

- The software will be hosted on a web page so it must be secure enough to not allow users to get any data other than what the application is supposed to serve
- The software must not divulge any user data to any third parties
- The software will only take in information directly inputted from the user
    - i.e. will not automatically take in user's location to help in preferences submission, etc
- No sensitive data will be stored in backups
- User inputs and displayed itineraries will be regularly cleared
    - Ideally as soon as the user downloads the image or exits the page (whichever comes first)
- Text and outputted itinerary files should implement file level encryption
- All downloadable files outputted will only be relevant to the itinerary

4.3 Software Quality Attributes

4.3.1 Reliability

- Software will not crash under normal operation and repeated reinputs of user preferences
    - Stress case will be 10 requests to redo the input process in each phase of the two way flight
    - Target will be zero crashes in at least 10 trials

4.3.2 Usability

- Software will stress ease of learning over minimization of displayed information
    - Every request for user input will display what is being requested and how to format each request, no matter how many times the process is repeated

4.3.3 Adaptability

- Software should be made to be able to work as built with updating database of available flights for future use

5 Software Evolution

5.1 Significant Changes

- Realization of complexity of database technologies in conjunction with our timeline for the project, it was decided that the data for the flights that will be read by the application will be read out of a text file rather than connecting to a database that we create. It is assumed that our clients will already have a database to connect our application to.
- Considered a more diverse mix of languages for each section of the software
    - Javascript, HTML, and PHP for use on the front end (as it is made to fit on a website)
    - SQL used to manage the database
    - Idea was dropped due to the all Java requirement of the product
- Simplified approach to storing database of available flight options to customer
    - Rather than creating a database via SQL, we will be storing all flight details within a .txt file
    - Flight information will be read in by our program to carry out calculations necessary for finding the most efficient flight options for the customer based on their inputted preferences
- Software originally did not handle two-way flights, users would have to run the program a second time to set this up
    - the program now has user flag if the flight is a two-way flight, and handles both flights in the same instance
- Complexity of creating a Graphical User Interface is also a component that likely will not be able to be reached within the timeline of the project so the outputs of our program will come out straight to the terminal

5.2 Existing Components Re-usability & Design Patterns

- The goal of the team is to show the functionality of the software with a currently small sample size of flights, but to build it for handling a larger database in the future

5.3 Time and Resource Overview

-

5.4 Software Change Management

5.5 Project Learnings & Takeaways

6 Alternative Approaches

6.1 Design Patterns

-

6.2 Additional Diagram Models

-