Purple Dinos

Capstone Project Manual

CSC 394 & IS 376

DePaul University | WQ 2021-2022

TABLE OF CONTENTS

1.0 Introduction

1.1 Our Mission

The main purpose of this project was to create a flight booking platform that consolidates major airlines into one, easy to use place.

1.1.1 Competition

Like most businesses, the airline flight reservation business does have numerous competitors. Dino Travel is a new entrant in this space and we want to be able to help customers purchase flights at a cheaper rate while providing better customer satisfaction. Some of our competitors include Expedia, Priceline, and Travelocity.

1.1.2 Relevant Links

- Our Website (https://daniel-mccarthy.github.io/DinoTravelFrontend)
- Frontend code (https://github.com/Daniel-McCarthy/DinoTravelFrontend)
- Backend code (https://github.com/DPUPurpleDinos/dino-travel)

1.2 The Plan

We used Google Drive for all planning and documentation. There are 2 main files that we used, both Google Sheets.

1) Timelog (See 5.0)
2) Calendar

A general plan was created at the beginning of the quarter and a more specific plan was created weekly during the mandatory team meeting.

1.2.1 The Calendar

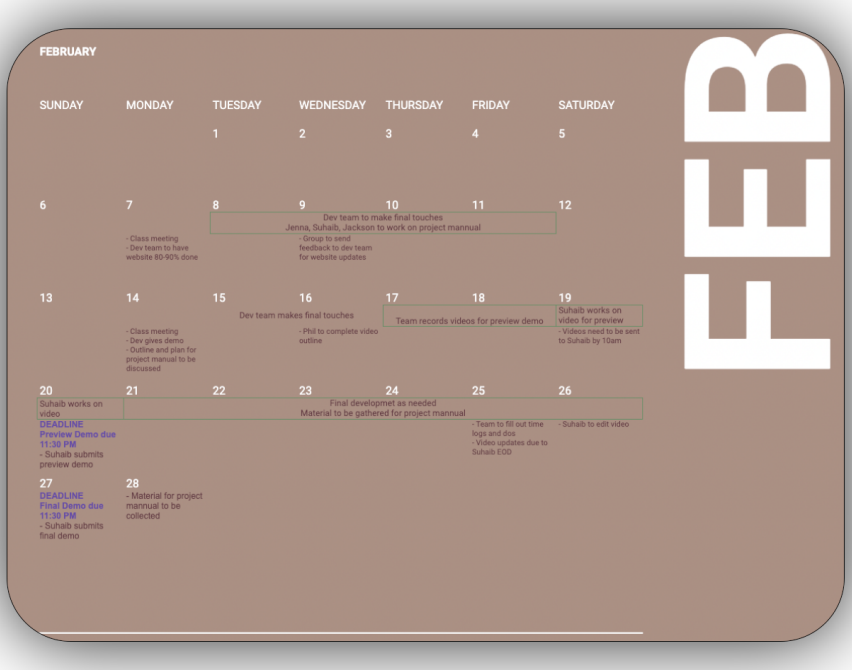
The Calendar holds important deadlines, tasks, and descriptions. This is where you will find group members to do lists and the modules.

1.2.1.1 Month View

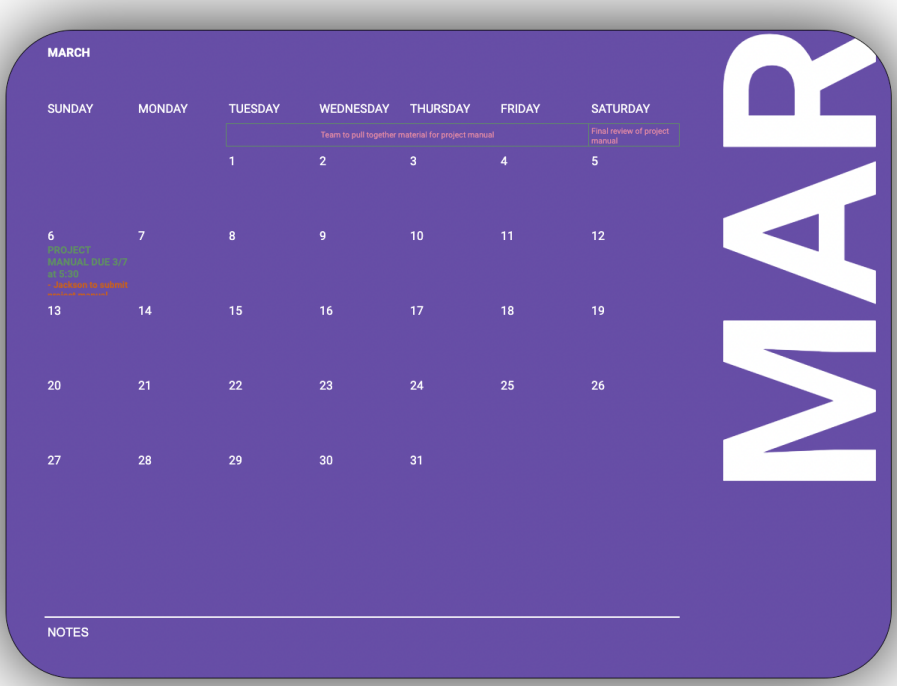
This shows submission deadlines and timeframes of when things are going to be worked on. A common workflow that our team used the week before a submission deadline is -

- ☐ Video outline done by Wednesday EOD
- ☐ Record your part of the video Thursday - Friday
- ☐ Send Video Manager your video by Friday EOD or early Saturday
- ☐ Suhaib works on the video on Saturday
- ☐ Suhaib send video to group on Sunday for feedback
- ☐ Video is submitted by EOD Sunday

1.2.1.1 Calendar View of January



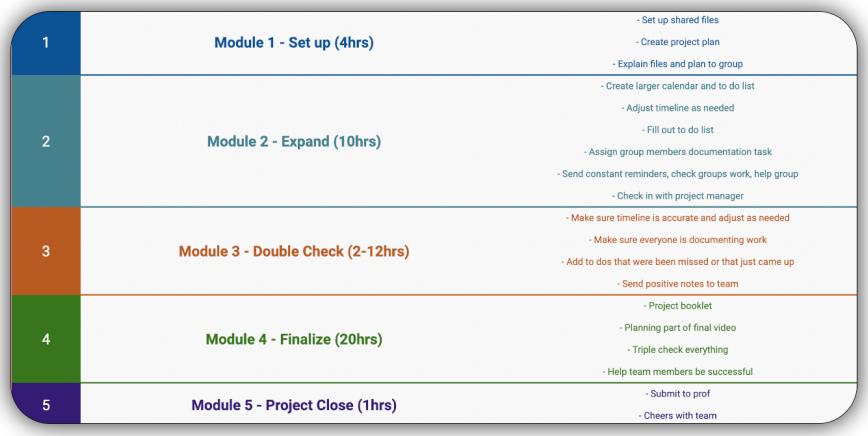
1.2.1.1 Calendar View of February



1.2.1.1 *Calendar View of March*

1.2.1.2 Modules

Each part of the project has been broken down into modules.The modules have submodules that follow the 5 hour rule. There is an estimated amount of time attached to each module.

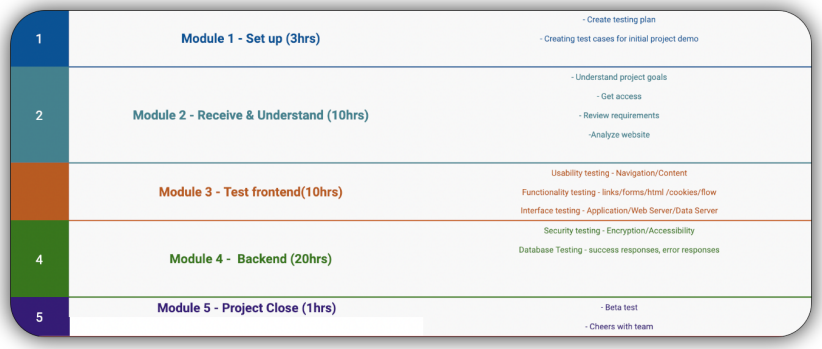


1.2.1.2 Planning - Modules on the left, Sub-Modules on the right

1.2.1.2 Front End Modules - Modules on the left, Sub-Modules on the right

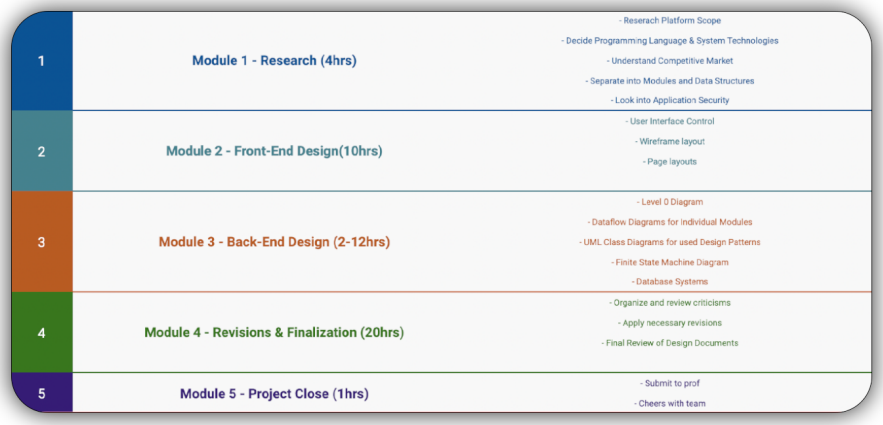
1	Module 1 - Set up (20 hrs)	-Set up AWS server, with MySQL database - Start Repo -Set up the basics of the spring boot server -Aquired a valid SSL certificate -Aquired an Amadeus API key -Find an appropriate API to use for data
2	Module 2 - Funcutionality (50 hrs)	-Fix any issues that arise from connecting the frontend to the backend - Add response codes to the Reservations and Users tables -Allow for requesting all locations, and air ports -Allow for requesting the cheapest flights with a given time period -Allow for requesting a seat map for a certain flight -Allow for booking a certain flight - Update Schema. Add baggage to reservations. - Add Complaints and Review REST controllers -Allow for canceling a certain flight
3	Module 3 - Quality (10 hrs)	- Comments throughout the Java classes - Comments in the testing module -Cache requests to lower database/amadeus api calls
4	Module 4 - Security (20hrs)	-Secure the server and database - Gmail user login verification? - Send user token through header for API requests
5	Module 5 - Project Close (1hrs)	- Submit to prof -Clean up code and remove unneeded dependencies - Cheers with team

1.2.1.2 Back End - Modules on the left, Sub-Modules on the right

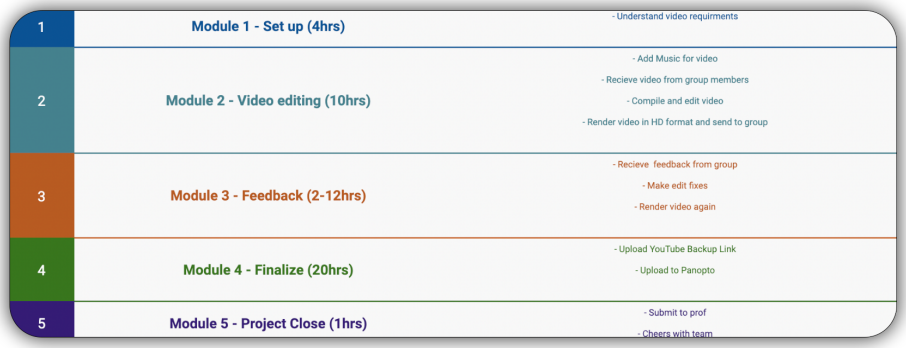


1	Module 1 - Set up (3hrs)	- Create testing plan - Creating test cases for initial project demo
2	Module 2 - Receive & Understand (10hrs)	- Understand project goals - Get access - Review requirements -Analyze website
	Module 3 - Test frontend(10hrs)	Usability testing - Navigation/Content Functionality testing - links/forms/html /cookies/flow Interface testing - Application/Web Server/Data Server
4	Module 4 - Backend (20hrs)	Security testing - Encryption/Accessibility Database Testing - success responses, error responses
5	Module 5 - Project Close (1hrs)	- Beta test - Cheers with team

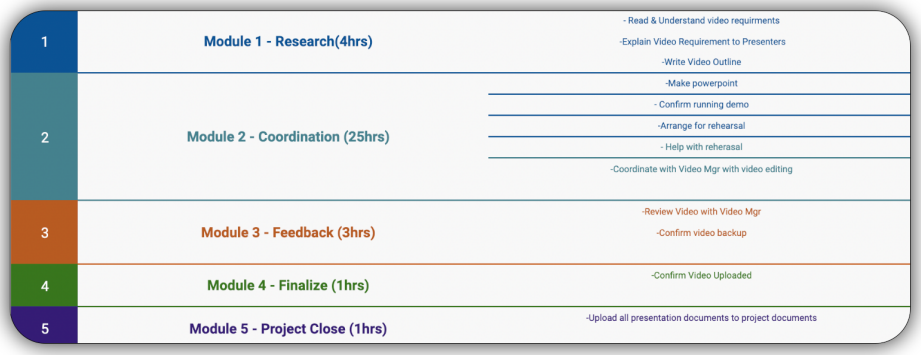
1.2.1.2 Testing - Modules on the left, Sub-Modules on the right



1.2.1.2 Design - Modules on the left, Sub-Modules on the right



1.2.1.2 Video - Modules on the left, Sub-Modules on the right

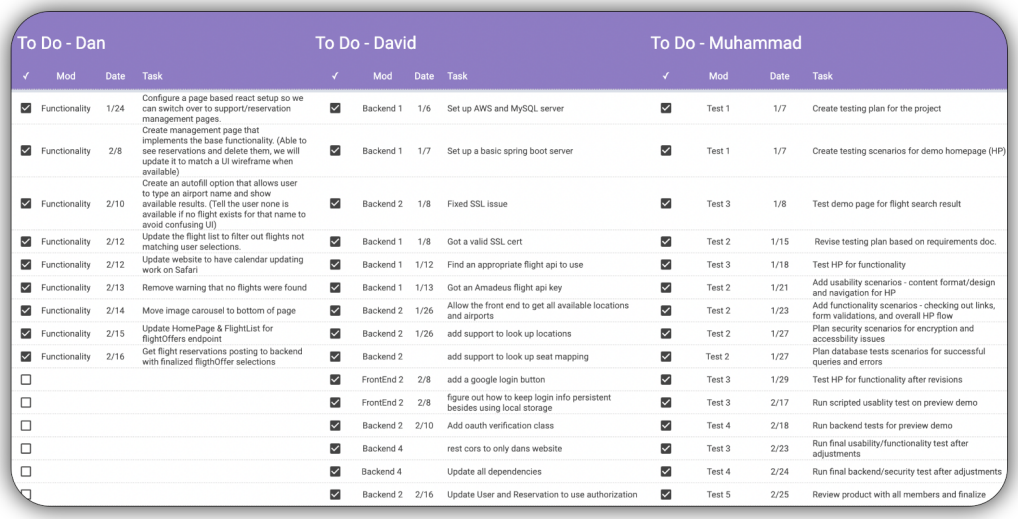


1.2.1.2 Presentation

1.2.1.3 To Do List

The submodules can be broken down even farther. The to-do list is where more specific and tedious tasks are located. Knowing that every team member has a different work style, team members were encouraged to use the to-do list as much or as little as possible. However, everyone had to try the to-do list once and the timelogs are mandatory.

The to-do list notes the date a task was worked on, what module and sub-module it applies to, and a description of the task.



To Do - Dan

✓	Mod	Date	Task
☑	Functionality	1/24	Configure a page based react setup so we can switch over to support/reservation management pages.
☑	Functionality	2/8	Create management page that implements the base functionality. (Able to see reservations and delete them, we will update it to match a UI wireframe when available)
☑	Functionality	2/10	Create an autofill option that allows user to type an airport name and show available results. (Tell the user none is available if no flight exists for that name to avoid confusing UI)
☑	Functionality	2/12	Update the flight list to filter out flights not matching user selections.
☑	Functionality	2/12	Update website to have calendar updating work on Safari
☑	Functionality	2/13	Remove warning that no flights were found
☑	Functionality	2/14	Move image carousel to bottom of page
☑	Functionality	2/15	Update HomePage & FlightList for flightOffers endpoint
☑	Functionality	2/16	Get flight reservations posting to backend with finalized fligthOffer selections
☐			
☐			
☐			
☐			
☐			
☐			

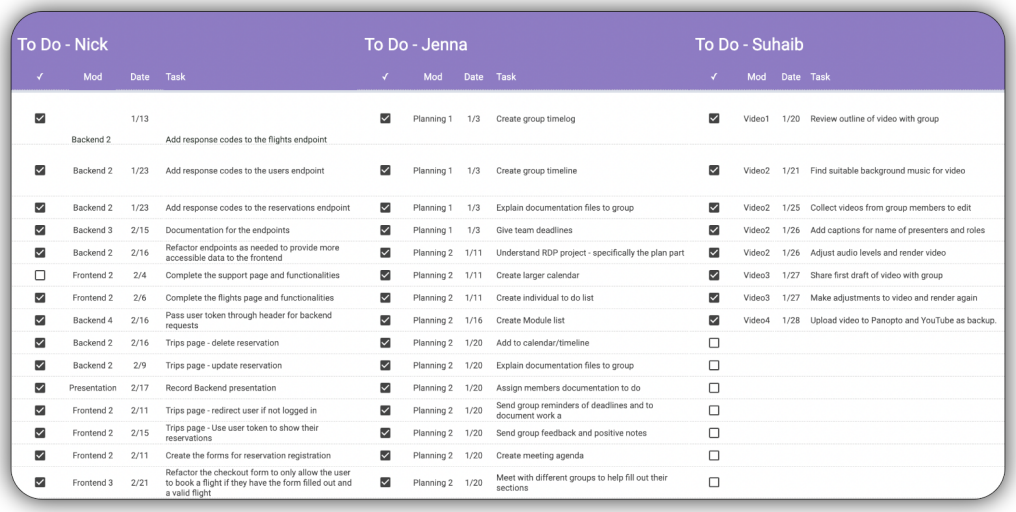
To Do - David

✓	Mod	Date	Task
☑	Backend 1	1/6	Set up AWS and MySQL server
☑	Backend 1	1/7	Set up a basic spring boot server
☑	Backend 2	1/8	Fixed SSL issue
☑	Backend 1	1/8	Got a valid SSL cert
☑	Backend 1	1/12	Find an appropriate flight api to use
☑	Backend 1	1/13	Got an Amadeus flight api key
☑	Backend 2	1/26	Allow the front end to get all available locations and airports
☑	Backend 2	1/26	add support to look up locations
☑	Backend 2		add support to look up seat mapping
☑	FrontEnd 2	2/8	add a google login button
☑	FrontEnd 2	2/8	figure out how to keep login info persistent besides using local storage
☑	Backend 2	2/10	Add oauth verification class
☑	Backend 4		rest cors to only dans website
☑	Backend 4		Update all dependencies
☑	Backend 2	2/16	Update User and Reservation to use authorization

To Do - Muhammad

✓	Mod	Date	Task
☑	Test 1	1/7	Create testing plan for the project
☑	Test 1	1/7	Create testing scenarios for demo homepage (HP)
☑	Test 3	1/8	Test demo page for flight search result
☑	Test 2	1/15	Revise testing plan based on requirements doc.
☑	Test 3	1/18	Test HP for functionality
☑	Test 2	1/21	Add usability scenarios - content format/design and navigation for HP
☑	Test 2	1/23	Add functionality scenarios - checking out links, form validations, and overall HP flow
☑	Test 2	1/27	Plan security scenarios for encryption and accessbility issues
☑	Test 2	1/27	Plan database tests scenarios for successful queries and errors
☑	Test 3	1/29	Test HP for functionality after revisions
☑	Test 3	2/17	Run scripted usablity test on preview demo
☑	Test 4	2/18	Run backend tests for preview demo
☑	Test 3	2/23	Run final usability/functionality test after adjustments
☑	Test 4	2/24	Run final backend/security test after adjustments
☑	Test 5	2/25	Review product with all members and finalize

1.2.1.3 Team To-Do List Sample 1



To Do - Nick

✓	Mod	Date	Task
☑	Backend 2	1/13	Add response codes to the flights endpoint
☑	Backend 2	1/23	Add response codes to the users endpoint
☑	Backend 2	1/23	Add response codes to the reservations endpoint
☑	Backend 3	2/15	Documentation for the endpoints
☑	Backend 2	2/16	Refactor endpoints as needed to provide more accessible data to the frontend
☐	Frontend 2	2/4	Complete the support page and functionalities
☑	Frontend 2	2/6	Complete the flights page and functionalities
☑	Backend 4	2/16	Pass user token through header for backend requests
☑	Backend 2	2/16	Trips page - delete reservation
☑	Backend 2	2/9	Trips page - update reservation
☑	Presentation	2/17	Record Backend presentation
☑	Frontend 2	2/11	Trips page - redirect user if not logged in
☑	Frontend 2	2/15	Trips page - Use user token to show their reservations
☑	Frontend 2	2/11	Create the forms for reservation registration
☑	Frontend 3	2/21	Refactor the checkout form to only allow the user to book a flight if they have the form filled out and a valid flight

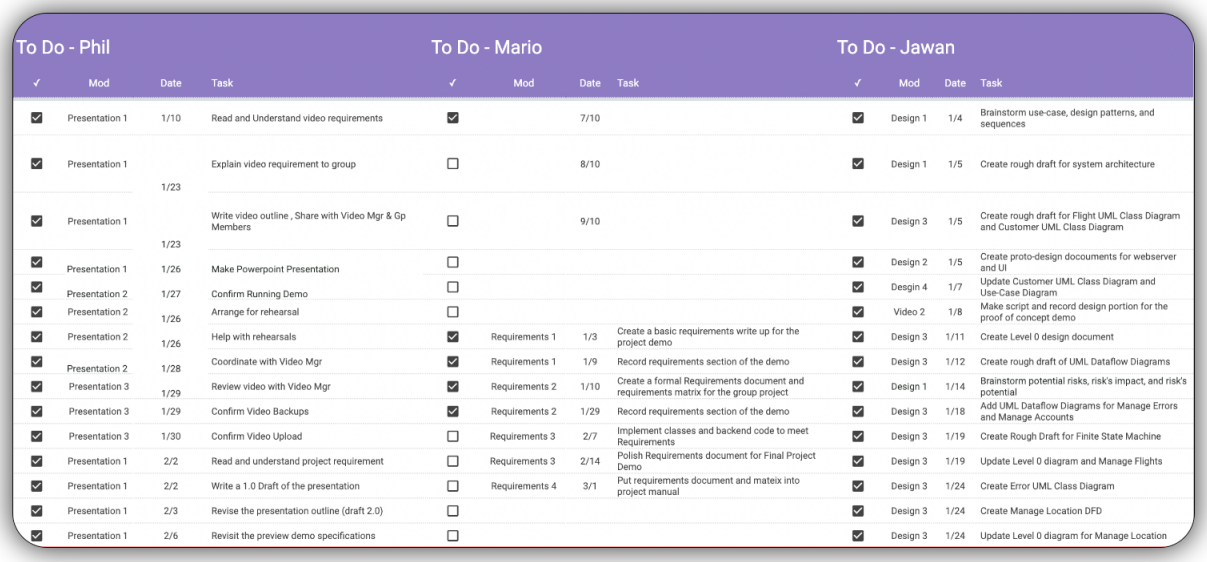
To Do - Jenna

✓	Mod	Date	Task
☑	Planning 1	1/3	Create group timelog
☑	Planning 1	1/3	Create group timeline
☑	Planning 1	1/3	Explain documentation files to group
☑	Planning 1	1/3	Give team deadlines
☑	Planning 2	1/11	Understand RDP project - specifically the plan part
☑	Planning 2	1/11	Create larger calendar
☑	Planning 2	1/11	Create individual to do list
☑	Planning 2	1/16	Create Module list
☑	Planning 2	1/20	Add to calendar/timeline
☑	Planning 2	1/20	Explain documentation files to group
☑	Planning 2	1/20	Assign members documentation to do
☑	Planning 2	1/20	Send group reminders of deadlines and to document work a
☑	Planning 2	1/20	Send group feedback and positive notes
☑	Planning 2	1/20	Create meeting agenda
☑	Planning 2	1/20	Meet with different groups to help fill out their sections

To Do - Suhaib

✓	Mod	Date	Task
☑	Video1	1/20	Review outline of video with group
☑	Video2	1/21	Find suitable background music for video
☑	Video2	1/25	Collect videos from group members to edit
☑	Video2	1/26	Add captions for name of presenters and roles
☑	Video2	1/26	Adjust audio levels and render video
☑	Video3	1/27	Share first draft of video with group
☑	Video3	1/27	Make adjustments to video and render again
☑	Video4	1/28	Upload video to Panopto and YouTube as backup.
☐			
☐			
☐			
☐			
☐			
☐			
☐			

1.2.1.3 Team To-Do List Sample 2



To Do - Phil

✓	Mod	Date	Task
☑	Presentation 1	1/10	Read and Understand video requirements
☑	Presentation 1	1/23	Explain video requirement to group
☑	Presentation 1	1/23	Write video outline , Share with Video Mgr & Gp Members
☑	Presentation 1	1/26	Make Powerpoint Presentation
☑	Presentation 2	1/27	Confirm Running Demo
☑	Presentation 2	1/26	Arrange for rehearsal
☑	Presentation 2	1/26	Help with rehearsals
☑	Presentation 2	1/28	Coordinate with Video Mgr
☑	Presentation 3	1/29	Review video with Video Mgr
☑	Presentation 3	1/29	Confirm Video Backups
☑	Presentation 3	1/30	Confirm Video Upload
☑	Presentation 1	2/2	Read and understand project requirement
☑	Presentation 1	2/2	Write a 1.0 Draft of the presentation
☑	Presentation 1	2/3	Revise the presentation outline (draft 2.0)
☑	Presentation 1	2/6	Revisit the preview demo specifications

To Do - Mario

✓	Mod	Date	Task
☑		7/10	
☐		8/10	
☐		9/10	
☐			
☐			
☐			
☑	Requirements 1	1/3	Create a basic requirements write up for the project demo
☑	Requirements 1	1/9	Record requirements section of the demo
☑	Requirements 2	1/10	Create a formal Requirements document and requirements matrix for the group project
☑	Requirements 2	1/29	Record requirements section of the demo
☐	Requirements 3	2/7	Implement classes and backend code to meet Requirements
☐	Requirements 3	2/14	Polish Requirements document for Final Project Demo
☐	Requirements 4	3/1	Put requirements document and mateix into project manual
☐			
☐			

To Do - Jawan

✓	Mod	Date	Task
☑	Design 1	1/4	Brainstorm use-case, design patterns, and sequences
☑	Design 1	1/5	Create rough draft for system architecture
☑	Design 3	1/5	Create rough draft for Flight UML Class Diagram and Customer UML Class Diagram
☑	Design 2	1/5	Create proto-design docouments for webserver and UI
☑	Desgin 4	1/7	Update Customer UML Class Diagram and Use-Case Diagram
☑	Video 2	1/8	Make script and record design portion for the proof of concept demo
☑	Design 3	1/11	Create Level 0 design document
☑	Design 3	1/12	Create rough draft of UML Dataflow Diagrams
☑	Design 1	1/14	Brainstorm potential risks, risk's impact, and risk's potential
☑	Design 3	1/18	Add UML Dataflow Diagrams for Manage Errors and Manage Accounts
☑	Design 3	1/19	Create Rough Draft for Finite State Machine
☑	Design 3	1/19	Update Level 0 diagram and Manage Flights
☑	Design 3	1/24	Create Error UML Class Diagram
☑	Design 3	1/24	Create Manage Location DFD
☑	Design 3	1/24	Update Level 0 diagram for Manage Location

1.2.1.3 Team To-Do List Sample 3

1.3 Meet the Team

The Dino Travel project team consists of ten members; half the group consisting of Computer Science majors, the other half consisting of Information Systems majors. This project was the culmination of a 10-week Capstone Course taught at DePaul University in the Winter Quarter of 2022.

1.3.1 Project Manager

Jackson Meyer
Project Manager

The project manager is responsible for the whole project. They guarantee that all other responsibilities are met, and if not, will initiate reassignments. They must solve problems for the group, and guarantee that meetings will be held, and are efficient with an agenda. While managing to be on track for the deadlines in the project, they must also manage the different development styles among group members. They are the liaison to the instructor and must make tough decisions about deadlines, personnel, and group structure.

1.3.2 Video Manager

Suhaib Mikbel
Video Manager

The Video Manager works under stress to deliver demo videos on time and running. They are also reliable with backups, early running "safety" versions, and deadlines. They are also knowledgeable about video recording and editing packages, and production, or willing to become so. The Video Manager works well with Presentation Manager and Project Manager. They are also good with organization to make sure component pieces are delivered in time for the full production. The Video Manager is able to help others make local videos and stills remotely and can get access to software to produce videos.

1.3.3 Planning/Documentation Manager

Jenna Dahl-Crimmin
Documentation Manager, Planning Manager

The Documentation and Planning Manager must create, early in the quarter, the formal, online, planning document with dates, hours, modules, tasks, names of group members responsible for each task, and dependencies of one task on another in it. Regularly communicates to report on the status of the plan by keeping everyone informed of the progress of all aspects of the group project. Makes updates to the plan as tasks are completed through the quarter. Works with the group and the project manager when the plan must be dynamically altered. Responsible that the time logs of all group members are updated, and available online throughout the quarter and for the final, printed, deliverable of the project group: the project booklet.

1.3.4 Interface Design Manager/Testing Manager

Muhammad Fahad
Interface Design Manager, Testing Manager

The Interface Design Manager is responsible for managing all aspects of the HCI component of the project. Works with the design manager and the requirements manager to influence the design according to valid HCI principles. Responsible for the formal usability study. Also Responsible for the testing plan for the group project and manages the testing group. The Testing Manager was responsible for the testing plan for the group project and managed the testing groups. The Testing Manager produces formal tests of the project for the project booklet.

1.3.5 Implementation Manager

David Gawel
Implementation Manager

The Implementation Manager is responsible for making sure that the project will actually run on each of the days that deliverables are due. They also manage the development team: assesses the strengths of members of the team, guarantees that everyone's code will run, and is compatible with the work of others as well as manage the translation of the design into running code. They work with the designer on technical issues relating to the running project. Responsible for the passing grade of every group member and is the bottom line on whether or not any particular code will make it into the official final project. Stands behind and *guarantees* that any code shown in demonstration will run. Manages the backup systems such that they are ready to come online on a moment's notice during demonstration and communicates with the project manager, the designer, and the planner, among others, about how the development is proceeding.

1.3.6 Webmaster

Dan McCarthy
Webmaster

The Webmaster and Testing Manager was responsible for the central communication hub(s), and the "face" of the project group. They worked with the implementation manager when web-related services were needed for the group project, as well as with the presentation manager when web resources were required for the demonstrations. They also worked with the Documentation Manager to translate the web resources into printed format for the project booklet.

1.3.7 Design Manager

Jawan Luangnikone
Design Manager

The Design Manager is responsible for producing the full, technical, design from which the project code will be written. They also work with the group to turn the initial ideas into a technical document that the group can follow, as well as work with the implementation manager to select the hardware, database, client, server, and implementation platforms. Performs the due diligence research to see what has been done before, what is available, what can be improved, how the design will solve problems for which there is a market. As well as producing the formal design document. The Design Manager works with the presentation manager to prepare the design for the RDP demo. Make sure there are not any faulty designs, and in the worst case, can guarantee there are no failures.

1.3.8 Presentation Manager

Philip Reeves
Presentation Manager

The presentation manager is in charge of formatting how the presentations are going to look, and in what order as well as guaranteeing that the running project will be shown at the demo. The Presentation Manager manages the critical, extensive, demonstrations of the group's work and has executive control over the look, and effectiveness, of the group during the critical minutes of the presentation.

1.3.9 Systems Programmer Manager/Remote Collaboration Manager

Nick Clark

Systems Programming Manager, Remote Collaboration Manager

The Systems Programming Manager is responsible for managing all aspects of the unix system, if one is used, including security, logins and backups. They also teach the rest of the group members about unix tools, as needed. The Remote Collaboration Manager should be strong technically and available during the first few weeks of the quarter. They are patient with group users not familiar with particular communication packages. They are willing to explore new packages, install them and give them a try before making recommendations to the group.

1.3.10 Requirements Manager

Mario DiBartomoleo
Requirements Manager

The requirements manager is responsible for the extensive formal requirements for the group project. They manage the wolf and thief groups and prepare the formal requirements document. They also prepare the requirements section of the RDP demo and work with the Documentation Manager to prepare the requirements document for inclusion in the project booklet.

2.0 Overview/RDP

2.1 Requirements Document

The Requirements Document section includes information about the Purple Dino's goals and requirements set between the client and development team. The purpose of this document is to arrange an agreement between the client and development team of what is expected to be in the final product of the application being developed. (Note: The numbering system of each section in the requirements is separate from the system used in the project manual.)

Another aspect of the Requirements Document was the accompanying Requirements Matrix, though due to its size and unique formatting, including the entire Matrix in this Project Manual didn't flow well. Instead, we've included the first portion of each matrix at the end of the section that they correspond to.

Functional Requirements

1.1 User Class

1.1.1 Account Registration

The user is able to create an account using their Gmail credentials. This account will give the user the ability to delete and update their specific reservations.

1.1.2 User Login

If the user has a registered account, the user should be able to login to Dino Travel.

1.1.3 Dashboard

If the user is logged in, the dashboard will allow the user to manage a reservation they created by updating or deleting their own.

1.1.4 Support

The user is able to access a support page that provides information on technical issues. Troubleshooting etc…

1.1.5 About

The user is able to access a page that provides information about Dino Travel's services.

1.1.6 Trips

Given that the user has a registered account, they should be able to access the trips functionality where they can manage their reservations by updating or deleting a reservation instance.

1.1.7 Round Trip

When selected, the flight search engine will only search for all available round trip flights.

1.1.8 One Way

When selected, the flight search engine will only search for all available one way flights.

1.1.9 Multi-City

When selected, the flight search engine will provide all available multi-city flight itineraries

1.1.10 Number of Adult Travelers

The user should be able to select the amount of adults making a reservation up to the capacity of the flight.

1.1.11 Number of Child Travelers

The user should be able to select the amount of children making a reservation up to the capacity of the flight.

1.1.12 Flight Class

The user should be able to select which flight class they will be purchasing a ticket for. Classes Include: First, Economy, Premium Economy, and Business.

1.1.13 Departure City

The user is able to search and select a departure city. The city list will only contain all airports that are registered with an IATA code.

1.1.14 Destination City

The user is able to search and select a destination city. The city list will only contain all airports that are registered with an IATA Code.

1.1.15 Departing Date

The user is able to select a date to depart from the selected departure city

1.1.16 Returning Date

The user (if necessary) is able to select a date to return back to the departure city.

1.1.17 Submit

The user should be able to submit and validate their request for a flight reservation. Users will be prohibited from booking flights that would put the flights beyond full capacity.

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
1.1.1	1	The user is able to create an account using Gmail credentials. This account will give the user the ability to delete and update reservations.	Ok passes bad client attack	Dev is lazy and doesn't check if a user deletes or edits a different users reservation	NO	NOT APPROVED David Gawel 1/23/2022
1.1.1	2	The user is able to create an account using their Gmail credentials. This account will give the user the ability to delete and update their specific reservations.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
1.1.2	1	If the user has a registered account, the user should be able to login to Dino Travel.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED Nick Clark 1/23/2022
1.1.3	1	If the user is logged in, the dashboard will allow the user to manage a reservation	Clients could argue the dashboard doesn't provide enough options to manage their reservation. What functions does a user have to manage a reservation?	Which reservations can they manage, their own or others?	NO	NOT APPROVED David Gawel 1/23/2022
1.1.3	2	If the user is logged in, the dashboard will allow the user to manage a reservation they created by updating or deleting their own.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
1.1.4	1	The user is able to access a support page that provides information on technical issues. Troubleshooting etc...	Clients could argue the support page doesn't provide enough information. What topics would be covered?	Ok passed bad dev attack	NO	NOT APPROVED Nick Clark 1/23/2022
1.1.4	2	The user is able to access a support page that provides information on technical issues such as login troubleshooting, and how to manage a	Ok passed bad client attack	Ok passed bad dev attack	YES	APPROVED Nick Clark 1/25/2022
1.1.5	1	The user is able to access a page that provides information about Dino Travel's services.	Ok passed bad client attack	Ok passed bad dev attack	YES	APPROVED Nick Clark 1/23/2022
1.1.6	1	Given that the user has a registered account, they should be able to access the trips functionality where they can manage their reservations.	What functions does a user have to manage their reservation?	Ok passed bad dev attack	NO	NOT APPROVED Nick Clark 1/23/2022
1.1.6	2	Given that the user has a registered account, they should be able to access the trips functionality where they can manage their reservations by	Ok passed bad client attack	Ok passed bad dev attack	YES	APPROVED Nick Clark 1/24/2022

Example of 1.0 Functional Requirements Matrix

Technical and Testing Requirements

2.1 Response Time

2.1.1 Search Time

Measurements obtained from 1000 random simple searches through development during testing, the response time should not be longer than 2 seconds for 80% of the time.

2.1.2 Reservation Email

Measurements obtained from 200 random reservations during testing, the response time for receiving emails should not be longer than 60 seconds 100% of the time. Measurements will be carried out by development through automated tests. The fields for the reservations will be varied so all testing will be diverse.

2.2 System Dependability

2.2.1 Error

The user will be notified through an error message displayed on the webpage. If the system loses connection, javascript is not enabled, no flights are found, frontend is unable to post to the server, internal server error occurs, invalid arguments, not enough arguments, or unable to connect to the backend.

2.3 Performance/Usability

2.3.1 Flight Search

Departure and destination cities should be clearly visible from the homepage. The input boxes will also have an auto-complete feature in order to make the user find the airports/cities quicker.

2.3.2 City Auto-Fill

While inputting a city in the departure and destination search bars, it should suggest cities or airports in no more than 1 second if it is a listed city that has an airport with a registered IATA Code. Measurements obtained through 1000 random searches.

2.3.3 Departing/Returning Date

Departing and Returning date calendar should appear within .5 seconds of clicking on the button. Measurements will be obtained through 100 separate calendar clicks.

2.3.4 Disk Space

The system will require at least 8 GB of AWS storage space in order to function optimally.

2.4 Security

2.4.1 Security through Obscurity

All default port mappings shall be changed and there will be random user IDs and passwords for production.

2.4.2 Database Connection

Back-end connection to the database is encrypted with SSL.

2.4.3 Front-end Connection

Front-end connection to the back-end is encrypted with SSL.

2.4.4 HTTPS/SSL

All API requests and user authentication shall be encrypted with SSL .

2.4.5 Valid SSL Certificate

The website shall be secured using Valid SSL certificate.

2.4.6 CORS Enabled

The website shall be protected against cross site scripting attacks and protected from third party data miners using CORS. All text boxes are sanitized.

2.4.7 AWS Security

The website shall include firewalls to prevent unrecognized connections on certain ports.

2.5 Maintainability

2.5.1 Extendibility

The application should be considered easily extendable by 90% of developers on the PurpleDinos team. Allows for adjustments and new implementations to be changed after they have been agreed upon between the client and developers.

2.5.2 Testability

All functions within the application should be able to be tested by the developer team.

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
2.1.1	1	Measurements obtained from 1000 simple searches during testing, the response time should not be longer than 2 seconds for 80% of the time.	Client could argue the measurements are biased. Does the testing involve participating users or just developers?	Dev knows ahead of time the simple searches and only those searches pass, normally the searches are slow	NO	NOT APPROVED David Gawel 1/23/2022
2.1.1	2	Measurements obtained from 1000 random simple searches through development during testing, the response time should not be longer than 2 seconds	Ok passed bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
2.1.2	1	Measurements obtained from 200 reservations during testing, the response time for receiving emails should not be longer than 60 seconds 100% of the time.	Client could argue the measurements are biased. Does the testing involve participating users or just developers?	Dev knows ahead of time the email used and just makes sending emails to that email address fast.	NO	NOT APPROVED David Gawel 1/23/2022
2.1.2	2	Measurements obtained from 200 random reservations during testing, the response time for receiving emails should not be longer than 60	Ok passed bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
2.2.1	1	The user will be notified if the system loses connection, javascript is not enabled, no flights are found, frontend is unable to post to the server or, unable to connect to the backend.	Client could argue the notification wasn't sufficient. How are users notified? Error message, pop-up, etc?	User is not notified of an internal server error, invalid arguments, or not enough arguments given.	NO	NOT APPROVED David Gawel 1/23/2022
2.2.1	2	The user will be notified through an error message displayed on the webpage. If the system loses connection, javascript is not enabled, no flights are	Ok passed bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
2.3.1	1	Departure and destination cities should be clear and easy to input for searching.	What makes these functions easy?	Ok passed bad dev attack	NO	NOT APPROVED Nick Clark 1/23/2022
2.3.1	2	Departure and destination cities should be clearly visible from the homepage. The input boxes will also have an auto-complete feature in order to make the	Ok passed bad client attack	Ok passed bad dev attack	YES	APPROVED Nick Clark 1/24/2022
2.3.2	1	While inputting a city in the departure and destination search bars, it should suggest cities or airports in no more than 1 second.	Client could argue <unsupported city> never get a response	How many tests? How many times should it work? Dev only makes it work once	NO	NOT APPROVED David Gawel 1/23/2022
2.3.2	2	While inputting a city in the departure and destination search bars, it should suggest cities or airports in no more than 1 second if it is a listed city that has an airport with a registered IATA Code.	Ok passed bad client attack	Dev only makes it work once	NO	NOT APPROVED David Gawel 1/25/2022
2.3.2	3	While inputting a city in the departure and destination search bars, it should suggest cities or	Ok passed bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel

Example of 2.0 Technical and Testing Requirements Matrix

Risk Management

3.1 Mitigation Table

Risk	Potential Impact	Mitigation Strategy
MySQL Injection	There is a potential impact to all users as the data can be deleted or changed.	Using JPA allows us to use MySQL with parameterized safeguards.
Sensitive data exposure	Confidential information being sent over http, hackers can intercept and steal the data.	Encrypted HTTPS, organize, delete or prevent storage of data.
API Theft	Unauthorized programs can fetch our data and post to our servers.	Using CORS
Cross site scripting	Hacker can access the site from JS scripts	Sanitizing input, delete session IDs after 30 minutes
API Key Theft	People can steal our API keys and use them without our authorization	Established our own private servers in order to hide the API keys

3.2 Risk Management Plan

3.2.1 Management Process

Risks are identified per module by the PurpleDino project manager. To minimize the severity of each risk, we will be actively identifying them in a step by step process that includes risk identification, risk analysis, potential impact, risk reporting, planning, and monitoring.

3.2.2 Risk Identification

The PurpleDinos risk management team will evaluate risks by monitoring environmental factors and project scope. Project deliverables, assumptions, constraints, and key project documents will be given logs for members to note risk factors.

3.2.3 Risk Analysis

Outcomes from the project platform and quality of life will be used to determine most important risks to address.

3.2.4 Potential Impact

Potential impact of a risk will be determined through qualitative and quantitative analysis.

Qualitative Risks:

- Risks with probability greater than 60% of occurrence are considered to have a severe impact.
- Risks with probability 30%-60% of occurrence are considered to have a moderate impact
- Risks with probability 1%-30% of occurrence are considered to have a minimal impact.

Quantitative Risks:

- Analysis of risk events that are using the qualitative risk process will be prioritized by an estimated numerical rating.

3.2.5 Risk Planning

Top 5 Risks to the application include:

1. Website failure (moderate risk, priority ~ 1)
    - Website crashes
    - No longer responding

2. Database Corruption: (moderate risk, priority ~ 3)
    - Loss of data
    - SQL tables overwritten

3. DDoS attack (minimal risk, priority ~ 2)
    - Websites ability to respond to multiple users
    - An attempt by a malicious actor to overflood the website

4. API limit reached (moderate risk, priority ~ 4)
    - Reached maximum amount of GET requests for the Amadeus API.

5. False reservation (moderate risk, priority ~ 1)
    - The customer makes a reservation, but it doesn't reach the servers
    - Flight reservation isn't accepted by the flight provider

3.2.6 Risk Monitoring and Reporting

The PurpleDinos team will be monitoring risks that include:

Website failure:

- Check website status on AWS or another service
- Database corruption
- Keep track of the data tables in SQL

DDoS attack:

- Understand the minimum and maximum amount of users the website can handle
- Apply appropriate measures to limit the number of users (Ex: timeout sessions)

API limit:

- Monitor GET request on Amadeus API website

False reservation:

- Testing to make sure reservations are received by the database
- If any troubles, have users contact us about details.

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
3.2.1	1	Risks are identified per module by the PurpleDino project manager. To minimize the severity of each risk, we will be actively identifying them in a step by	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED Nick Clark 1/23/2022
3.2.2	1	Team will evaluate risks by monitoring environmental factors and project scope. Project deliverables, assumptions, constraints, and key project documents will be given logs for members to note risk factors.	Ok passes bad client attack	Who is the team made of? The people who made the project scope.	NO	NOT APPROVED David Gawel 1/23/2022
3.2.2	2	The PurpleDinos risk management team will evaluate risks by monitoring environmental factors and project scope. Project deliverables, assumptions, constraints, and key project documents will be given logs for members to note risk factors.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
3.2.3	1	Outcomes from the project platform and quality of life will be used to determine most important risks to address.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED Nick Clark 1/23/2022
3.2.4	1	Potential impact of a risk will be determined through qualitative and quantitative analysis. Qualitative Risks: - Risks with probability greater than 60% of occurrence are considered to have a severe impact. - Risks with probability 30%-60% of occurrence are considered to have a moderate impact - Risks with probability 1%-30% of occurrence are considered to have a minimal impact. Quantitative Risks: - Analysis of risk events that are using the qualitative risk process will be prioritized by an estimated numerical rating.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED Nick Clark 1/23/2022
3.2.5	1	Website failure: moderate risk, priority ~ 1 -Website crashes -No longer responding Database Corruption: moderate risk, priority ~ 3 -Loss of data -SQL tables overwritten	N/A	What counts as a website failure? whole shut down or just a picture with no loading?	NO	NOT APPROVED Nick Clark 1/23/2022

Example of 3.0 Risk Management Requirements Matrix

Constraints

4.1 Reservation Process

The reservation application should provide access for non-users to view flight options with a limited scope. Non-Users will only be able to add a reservation. The application should give existing users full access to all details in the application and will provide a way to set up a flight reservation management page (dashboard) that will allow users to add, update, and delete their current and future flight reservations.

4.2 Ease of Use

All functional parts of the website should be labeled and pages should have a relevant title to their content.

4.3 Constraint Table

Constraint	Impact	Strategy
Scalability	Network and Distribution will only be in the U.S.	Finish the initial product before expansion
Mobile Application	Our Services will not be compatible with mobile devices	two phase development, where a web application will be fully developed and then a mobile application will be implemented.
User Identity	Duplicate accounts, Wasted Storage space in the database	Verification through google accounts

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
4.1.1	1	The reservation application should provide access to non-users to view flight options with a limited scope. The application should give existing users full access to all	Clients could argue there wasn't enough support. What functionalities does a non user have access to? What functionalities does a user have to manage	What is limited scope? Dev makes it so that people without accounts can't see pricing. What is full	NO	NOT APPROVED Nick Clark 1/23/2022
4.1.1	2	The reservation application should provide access for non-users to view flight options with a limited scope. Non-Users will only be able to add a reservation. The	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
4.2.2	1	All information should be easy to access and use.	Client could argue some information wasn't easy to access or use. What makes the information clear and	What is easy to use and access. Dev thinks the command line is	NO	NOT APPROVED Nick Clark

Example of 4.0 Constraints Requirements Matrix

Project Planning

5.1 Change Management

5.1.1 Module implemented in planning to double check work that has been done. If a change is required it will be documented under To Do. Project management will be making final decisions if change is required. Entire team will be notified of said change.

5.2 Maximum Hours Limit

5.2.1 Each module of the project will be allotted an estimated amount of hours. Actual hours spent are documented in a time log.

5.3 Assigned Tasks

5.3.1 Minor tasks will be self-assigned in a To-Do checklist.

5.3.2 Larger tasks will be assigned in the timeline by planner.

5.4 Deadlines

5.4.1 Deadlines will be set by project planner on the group calendar on google docs.

5.4.2 Deadlines will be seen through and fulfilled by the project manager.

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
5.1.1	1	5.1.1 Module implemented in planning to double check work that has been done. If a change is required it will be documented under To Do. Project	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
5.2.1	1	Each module of the project will be allotted an estimated amount of hours. Actual hours spent are documented in a time log.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
5.3.1	1	Minor tasks will be self-assigned in a To-Do checklist.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
5.3.2	1	Larger tasks will be assigned in the timeline by the planner.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
5.4.1	1	Deadlines will be set by project planner on the group calendar on google docs.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
5.4.2	1	Deadlines will be seen through and fulfilled by the project manager.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022

Example of 5.0 Project Planning Requirements Matrix

Project Group Management

6.1 Management Strategy

6.1.1 The Purple Dinos management strategy consists of clear communication through discord and zoom.

6.1.2 Management is involved with all aspects of the project through oversight and decision making.

6.1.3 Management is requiring weekly one on one meetings with each member of the group to discuss progress and needs.

6.2 Conflict Resolution

6.2.1 Through the weekly one on one meetings, management will allow open discussion of any problems related to the project and or group members in the project.

6.2.2 If such a problem arises, it will be handled promptly through the concerned parties.

6.3 Proactive Tasks of Management

6.3.1 Using each group meeting and one on one meeting, management will be able to poll for any problems or concerns amongst the group.

6.3.2 Management will be monitoring how much work is done, and if any schedule slippage is noticed, management will potentially delegate tasks to whoever is next most capable of completing the task.

6.4 Progress Reports

6.4.1 Informally conducted once a week during one on ones with the project manager.

6.5 Accountability

6.5.1 Each individual is accountable for what is required of their role they have been assigned. If an individual is incapable of performing the duties required by the role. The project manager is ultimately responsible for making sure that the work is completed one way or another.

6.5.2 Team members will be required to track their progress and hours spent working on their tasks in order to aid in accountability of said tasks.

6.5.3 Each role and their responsibilities can be found in Section 7.0 of the requirements document.

6.6 Acknowledgement

6.6.1 At both the weekly and in the one on one meetings, the work done by each group member is going to be evaluated and recognized according to what was completed

6.7 Agenda

6.7.1 Work will be tracked through google sheets using a time log document and a calendar document.

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
6.2.1	1	Through the weekly one on one meetings, management will allow open discussion of any problems related to the project and or group	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
6.2.2	1	If such a problem arises, it will be handled promptly through the concerned parties.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
6.3.1	1	Using each group meeting and one on one meeting, management will be able to poll for any problems or concerns amongst the group.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
6.3.2	1	Management will be monitoring how much work is done, and if any schedule slippage is noticed, management will potentially delegate tasks to	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
6.4.1	1	Informally conducted once a week during one on ones with the project manager	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
6.5.1	1	Each individual is accountable for what is required of their role they have been assigned. If an individual is incapable of performing the duties required by the	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
6.5.2	1	Team members will be required to track their progress and hours spent working on their tasks in order to aid in accountability of said tasks.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
6.5.3	1	Each role and their responsibilities can be found in Section 7.0 of the requirements document.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
6.6.1	1	At both the weekly and in the one on one meetings, the work done by each group member is going to be evaluated and recognized according to what was	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
6.7.1	1	Work will be tracked through google sheets using a time log document and a calendar document.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022

Example of 6.0 Project Management Requirements Matrix

Project Roles and Specifications

7.1 Project Manager

7.1.1 The project manager is responsible for the entire project. Their job is to guarantee that all other responsibilities are met, or otherwise will reassign work/positions.

7.1.2 Must dispute problems that come up in the group and is the bottom line for making final decisions about deadlines, personnel, and group structure.

7.2 Implementation manager

7.2.1 Responsible for the project to actually run on the day the deliverable is due.

7.2.2 Manages the development team. Manages the translation of the design into running code.

7.2.3 Makes the final decision on which code will make it into the official project.

7.3 Collaboration Software Manager

7.3.1 Strong technical skills and is very available and is able to explore new packages, install them, and test them, before making recommendations to the group.

7.4 Video and Media Manager

7.4.1 Knowledgeable about recording and editing packages. Is able to produce a video that presents the product to companies.

7.4.2 Must have back-ups, and safety release versions to make sure there is something to present on deadlines.

7.4.3 Is able to help others make local videos and stills remotely.

7.5 Planner

7.5.1 Responsible for group planning. Must be able to create a document with dates, hours, modules, tasks, and dependencies of each task.

7.5.2 Communicates regularly to report on the status of the plan

7.5.3 Prepares the final plan for the project booklet in collaboration with the documentation manager.

7.5.4 Works with the presentation manager to have the plan ready for the RDP demo.

7.5.5 Is able to make last minute adjustments to the plan.

7.6 Webmaster

7.6.1 Responsible for the central communication hub and the “face” of the project group.

7.6.2 Works with the implementation manager when web-related services are needed for the group project.

7.6.3 Works with the presentation manager when web resources are required for the demonstrations.

7.6.4 Works with the documentation manager to translate the web resources into printed format for the project booklet

7.7 Documentation manager

7.7.1 Responsible for the time logs of all group members being updated and available online.

7.7.2 Responsible for the final printed deliverable of the project group.

7.8 Requirements Manager

7.8.1 Responsible for creating extensive formal requirements for the group project.

7.8.2 Prepare a formal requirement document and prepare the requirements section of the RDP demo.

7.9 Design Manager

7.9.1 Responsible for producing the full, technical, design from which the project code will be written.

7.9.2 Works to turn ideas into a document that the group can follow.

7.9.3 Works with the implementation manager to select the hardware, database, client, server, and implementation platforms.

7.10 Testing Manager

7.10.1 Responsible for creating a testing plan for the group project.

7.10.2 Produces formal tests of the project for the project booklet.

7.11 Presentation Manager

7.11.1 Controls the look and effectiveness of the group while creating a formal presentation.

7.11.2 Guarantees a showcasing of the running project.

7.12 Usability Manager

7.12.1 Responsible for managing all aspects of the HCI component of the project.

7.12.2 Works with the design manager and the requirements manager to influence the design according to valid HCI principles.

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
7.1.1	1	The project manager is responsible for the entire project. Their job is to guarantee that all other responsibilities are met, or otherwise will reassign	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
7.1.2	1	Must dispute problems that come up in the group and is the bottom line for making final decisions about deadlines, personnel, and group structure.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
7.2.1	1	Responsible for the project to actually run on the day the deliverable is due.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
7.2.2	1	Manages the development team. Manages the translation of the design into running code.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
7.2.3	1	Makes the final decision on which code will make it into the official project.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
7.3.1	1	Strong technical skills and is very available.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
7.3.2	1	Able to explore new packages, install them, and test them, before making recommendations to the group	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
7.4.1	1	Knowledgeable about recording and editing packages. Is able to produce a video that presents the product to companies.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
7.4.2	1	Must have back-ups, and safety release versions to make sure there is something to present on deadlines.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
7.4.3	1	Is able to help others make local videos and stills remotely.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022

Example of 7.0 Project Roles and Specifications Requirements Matrix

Client Developer Management

8.1 Conflict Resolution

8.1.1 Conflict Resolution will be handled from both parties with an assigned liaison and a hired negotiator for both parties.

8.1.2 Should significant conflict arise from either party all production of affected areas will be postponed until a resolution is reached.

8.1.3 Should an agreement not be made, Clients will be charged accordingly for the work already developed and not work to be developed.

8.2 Assumptions

8.2.1 Clients and Developers will be required to communicate through English unless specified and agreed by both parties before finalization of development.

8.3 Developer Liaison

8.3.1The developer team will be assigning Muhammad Fahad as the acting Liaison between the client and developers. Muhammad will be responsible for determining what the consumer really wants, what the developers will build to suit that desire.

8.3.2 The liaison will be communicating with the development team regularly to stay updated and relay requirements through comments on the code and in meetings.

8.3.3 The liaison will be communicating with the clients regularly, keeping them updated and relay the development process.

8.4 Developer Negotiator

8.4.1 The developer team will be assigning Jackson Meyers to the negotiator role. This role will be responsible for identifying and resolving disputes and misunderstandings ahead of time.

8.4.2 Maintain open communication with clients and developers in order to mitigate any disputes or misunderstandings as quickly and efficiently as possible.

8.5 Communication for Stakeholders

8.5.1 The developer team will be using google groups and google drive to communicate. Should any parties be concerned with getting in touch with the team a company email will be provided in the project manual.

8.5.2 The team plans on having monthly meetings to briefly showcase progress made in development and any changes/adjustments that are required in the coming weeks.

8.5.3 Development team emails will be checked daily given the day is in the business week.
8.5.4 Project logs will be managed by the project planner and kept in a google calendar document.

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
8.1.1	1	Conflict Resolution will be handled from both parties with an assigned liaison and a hired negotiator for both parties.	Ok passes bad client attack	Ok passes bad developer attack	YES	APPROVED David Gawel 1/23/2022
8.1.2	1	Should significant conflict arise from either party all production of affected areas will be postponed until a resolution is had.	Ok passes bad client attack	Ok passes bad developer attack	YES	APPROVED David Gawel 1/23/2022
8.1.3	1	Should an agreement not be made, Client will be charged accordingly for the work already developed and not work to be developed.	Ok passes bad client attack	Ok passes bad developer attack	YES	APPROVED David Gawel 1/23/2022
8.2.1	1	Clients and Developers will be required to communicate through English unless specified and agreed by both parties before finalization of	Ok passes bad client attack	Ok passes bad developer attack	YES	APPROVED David Gawel 1/23/2022
8.3.1	1	The developer team will be assigning Muhammad Fahad as the acting Liaison between the client and developers. Muhammad will be responsible for	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
8.3.2	1	The liaison will be communicating with the development team regularly to stay updated and relay requirements through comments on the code and in meetings.	N/A	N/A		
8.3.4	1	The liaison will be communicating with the clients regularly, keeping them updated and relay the development process.	N/A	N/A		
8.4.1	1	The developer team will be assigning Jackson Meyer to the negotiator role. This role will be responsible for identifying and resolving disputes	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
8.4.2	1	Maintain open communication with clients and developers in order to mitigate any disputes or misunderstandings as quickly and efficiently as	Ok passes bad client attack	Ok passes bad developer attack	YES	APPROVED David Gawel 1/23/2022
8.5.1	1	The developer team will be using google groups and google drive to communicate. Should any parties be concerned with getting in touch with the	N/A	Ok passes bad developer attack	YES	APPROVED David Gawel 1/23/2022

Example of 8.0 Client Developer Management Requirements Matrix

Project Phases

9.1 Backend Modules

9.1.1 Setup AWS ec2 instance running Ubuntu.
9.1.2 AWS instance will run a MySQL server.
9.1.3 Create initial code for java spring boot server to run.
9.1.4 Add SSL and make sure CORS functions properly with the website.
9.1.5 Obtain an Amadeus API key.
9.1.6 When API calls fail, send an error message that explains the cause of the failure.
9.1.7 Create an API call that allows developers to get all available airports with an IATA code.
9.1.8 Verify emails when an account is made.
9.1.9 Setup a cache for reducing the amount of Amadeus API calls.

9.2 Front End Modules

9.2.1 Create initial code for the base react website.
9.2.2 Build HTML for the homepage based on the initial design we have made.
9.2.3 Implement calls to the backend for looking up flight pricing, all available airports, reserving flights, and user data.
9.2.4 Develop flight registration workflow based on the initial design made by the PurpleDinos team.
9.2.5 Refine flight booking UI
9.2.6 Develop flight management page
9.2.7 Build login handling and user management
9.2.8 Develop support page

9.3 Video Creation Editing

9.3.1 Reviewing all videos that are submitted.
9.3.2 Organizing the videos in the editor.
9.3.3 Creating transitions between video clips.
9.3.4 Adjusting audio levels for all clips to be the same
9.3.5 Captioning roles and names in each video.
9.3.6 Rendering video
9.3.7 Video is shown to the group to assure quality.
9.3.8 Uploading to Panopto and YouTube for back-up.

9.4 Testing

9.4.1 Backend Testing Modules

- Test Users flights and reservations API.
- Test location API calls.
- Test for the status codes for the API.
- Test User table.

9.4.2

- Test usability
- Test functionality
- Test application
- Surveys
- Miscellaneous
- Polish web experience.

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
9.1.1	1	Setup AWS server instance	N/A	Developer just sets up an ec2 instance nothing is running	NO	Not APPROVED David Gawel 1/23/2022
9.1.1	2	Setup AWS ec2 instance running Ubuntu.	N/A	Ok passes bad developer attack	YES	APPROVED David Gawel 1/25/2022
9.1.2	1	Setup Database	N/A	What database? Does it even run?	NO	Not APPROVED David Gawel 1/23/2022
9.1.2	2	AWS instance will run a MySQL server.	N/A	Ok passes bad developer attack	YES	APPROVED David Gawel 1/25/2022
9.1.3	1	Create starter code to test application	N/A	started code on what?	NO	Not APPROVED David Gawel 1/23/2022
9.1.3	2	Create initial code for java spring boot server to run.	N/A	Ok passes bad developer attack	YES	APPROVED David Gawel 1/25/2022
9.1.4	1	Add SSL and fixed CORS issues	N/A	What are CORS issues?	NO	Not APPROVED David Gawel 1/23/2022
9.1.4	2	Add SSL and make sure CORS functions properly with the website.	N/A	Ok passes bad developer attack	YES	APPROVED David Gawel 1/25/2022
9.1.5	1	Obtain an API key	N/A	What API key?	NO	Not APPROVED David Gawel 1/23/2022
9.1.5	2	Obtain an Amadeus API key.	N/A	Ok passes bad developer attack	YES	APPROVED David Gawel 1/25/2022
9.1.6	1	Create error handling	N/A	error handling for what	NO	Not APPROVED David Gawel 1/23/2022

Example of 9.0 Project Phases Requirements Matrix

Stakeholders

10.1 Creators

10.1.1 The creators and primary stakeholders of this application belong to the PurpleDino group.

10.1.2 Each member of the PurpleDino group will be investing a minimum of 4 hours every week into the building of the flight reservation application

10.2 Investors

10.2.1 The investors and stakeholders of the flight application are DePaul University and Clark Elliot.

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
10.1.1	1	The creators and primary stakeholders of this application belong to the PurpleDino group.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
10.1.2	1	Each member of the PurpleDino group will be investing a minimum of 4 hours every week into the building of the flight reservation application	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022
10.2.1	1	The investors and stakeholders of the flight application are DePaul University and Clark Elliot.	N/A	N/A	YES	APPROVED Mario DiBartolomeo 1/24/2022

Example of 10.0 Stakeholders Requirements Matrix

Technical Scope

11.1 Project Management

11.1.1 Manages the scope, schedule, cost, resources, and quality.

11.2 Coordination and meetings

11.2.1 There will be weekly hour-long meetings with the team. Task progression and concerns will be discussed in these hour long meetings.

11.2.2 Meant to organize and track progress of the application

11.3 Documentation Management

11.3.1 All deliverables will be stored and organized in a google drive. All forums will be shared with all group members.

11.4 Quality Assurance

11.4.1 Quality assurance will be provided throughout the duration of the project in order to deliver a quality product on schedule.

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
11.1.1	1	Project management manages the scope, schedule, cost, resources, and quality.	N/A	N/A	YES	
11.2.1	1	There will be weekly hour long meetings with the team. Task progression and concerns will be discussed in these hour long meetings.	N/A	N/A	YES	
11.2.2	1	Meant to organize and track progress of the application	N/A	N/A	YES	
11.3.1	1	All deliverables will be stored and organized in a google drive. All forums will be shared with all group members.	N/A	N/A	YES	
11.4.1	1	Quality assurance will be provided throughout the duration of the project in order to deliver a quality product on schedule.	N/A	N/A	YES	

Example of 11.0 Technical Scope Requirements Matrix

User Profile

12.1 Skill Level

12.1.1 Users must have a basic understanding of how to browse the web and how to operate a computer in order to navigate the website.
12.1.2 Users must understand English
12.1.3 Users must be literate and able to read and write at novice level.

12.2 User Training

12.2.1 Users will be able to learn how to navigate the website and troubleshoot any difficulties through the help feature that will be displayed on the website's home page.

12.3 User Types

12.3.1 The set of different user types will include customer, managerial, and administrator. All user types will be required to have the skills stated in 12.3.2 Skill level

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
12.1.1	1	Users must have a basic understanding of how to browse the web and how to operate a computer in order to navigate the website.	Ok passes bad client attack	Ok passes bad developer attack	YES	APPROVED David Gawel 1/23/2022
12.1.2	1	Users must understand English	Ok passes bad client attack	Ok passes bad developer attack	YES	APPROVED David Gawel 1/23/2022
12.1.3	1	Users must be literate and able to read and write at a novice level.	Ok passes bad client attack	Ok passes bad developer attack	YES	APPROVED David Gawel 1/23/2022
12.2.1	1	Users will be able to learn how to navigate the website and troubleshoot any difficulties through the help feature that will be displayed on the	Ok passes bad client attack	Ok passes bad developer attack	YES	APPROVED David Gawel 1/23/2022
12.3.1	1	The set of different user types will include customer, managerial, and administrator. All user types will be required to have the skills stated in	Ok passes bad client attack	Ok passes bad developer attack	YES	APPROVED David Gawel 1/23/2022

Example for 12.0 User Profile Requirements Matrix

Testing Requirements

13.1 Usability Testing

13.1.1 Test Navigation of the website and if it is visible and consistent. Web Page functionality should not change through appearance updates.
13.1.2 Verify that all links correspond to the pages they were intended to link to.

13.2 Functionality Testing

13.2.1 Test all links
13.2.2 Test all forums and validation.
13.2.3 Ensure HTML and CSS is working properly.
13.2.4 Test overall workflow to ensure the website is functioning as intended.

13.3 Interface Testing

13.3.1 Application test requests are sent correctly.
13.3.2 Test the web server is handling all the application requests.
13.3.3 Test that the web server is handling all the application requests as expected.

13.4 Security Testing

13.4.1 Test unauthorized access. Only developers with valid credentials should be able to edit the database. Logged in users should only be able to edit their reservation.
13.4.2 Terminate inactive user sessions. Inactive user sessions will be determined by an agreed amount of time between the client and developer.
13.4.3 Test that all sensitive data is encrypted using SSL.

13.5 Database Testing

13.5.1 Test queries are functioning properly.
13.5.2 Test status codes. Verify success codes are returned when intended. Verify error codes are returned when intended.
13.5.3 Test SQL injections do not work.
13.5.4 Test data integrity while creating, updating, and deleting data.
13.5.5 Check response times of application methods.
13.5.6 Test data is retrieved and displayed accurately.

Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
13.1.1	1	Test Navigation of the website is visible and consistent.	Client could argue there were some inconsistencies. What's the expectation for a consistent web page?	How do you measure visibility and consistency?	NO	NOT APPROVED Nick Clark 1/24/2022
13.1.1	2	Test Navigation of the website and if it is visible and consistent. Web Page functionality should not change through appearance updates.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
13.1.2	1	There shouldn't be grammatical Errors and content should be legible.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
13.2.1	1	Test all links	Client could argue the testing was insufficient. How are links tested?	What links are tested?	NO	NOT APPROVED Nick Clark 1/24/2022
13.2.1	2	Verify that all links correspond to the pages they were intended to link to.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
13.2.2	1	Test all forums and validation.	Client could argue the testing was insufficient. How are forums tested?	What forums and validation are tested?	NO	NOT APPROVED Nick Clark 1/24/2022
13.2.4	1	Test overall workflow to ensure that the website is functioning as intended.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
13.3.1	1	Application test requests are sent correctly.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
13.3.2	1	Test that the web server is handling all the application requests.	Ok passes bad client attack	Make sure that it handles them correctly not just handles it	NO	NOT APPROVED David Gawel 1/25/2022
13.3.2	2	Test that the web server is handling all the application requests as expected.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022

Example of 13.0 Testing Requirements Matrix

Documentation Requirements

14.1 Technical Maintenance

14.1.1 Should anything malfunction developers from the appropriate role will fix the problem.

14.2 Administrators of the System

14.2.1 There will be a developer account provided to all developers of the application

14.2.2 Only requests that have Dan's domain name will be able to access the applications API.

14.3 Commented Code

14.3.1 Complicated code will be commented on.

14.3.2 All REST endpoints will be commented to explain their purpose, parameters, and return variable. Developers can add additional comments as they see fit.

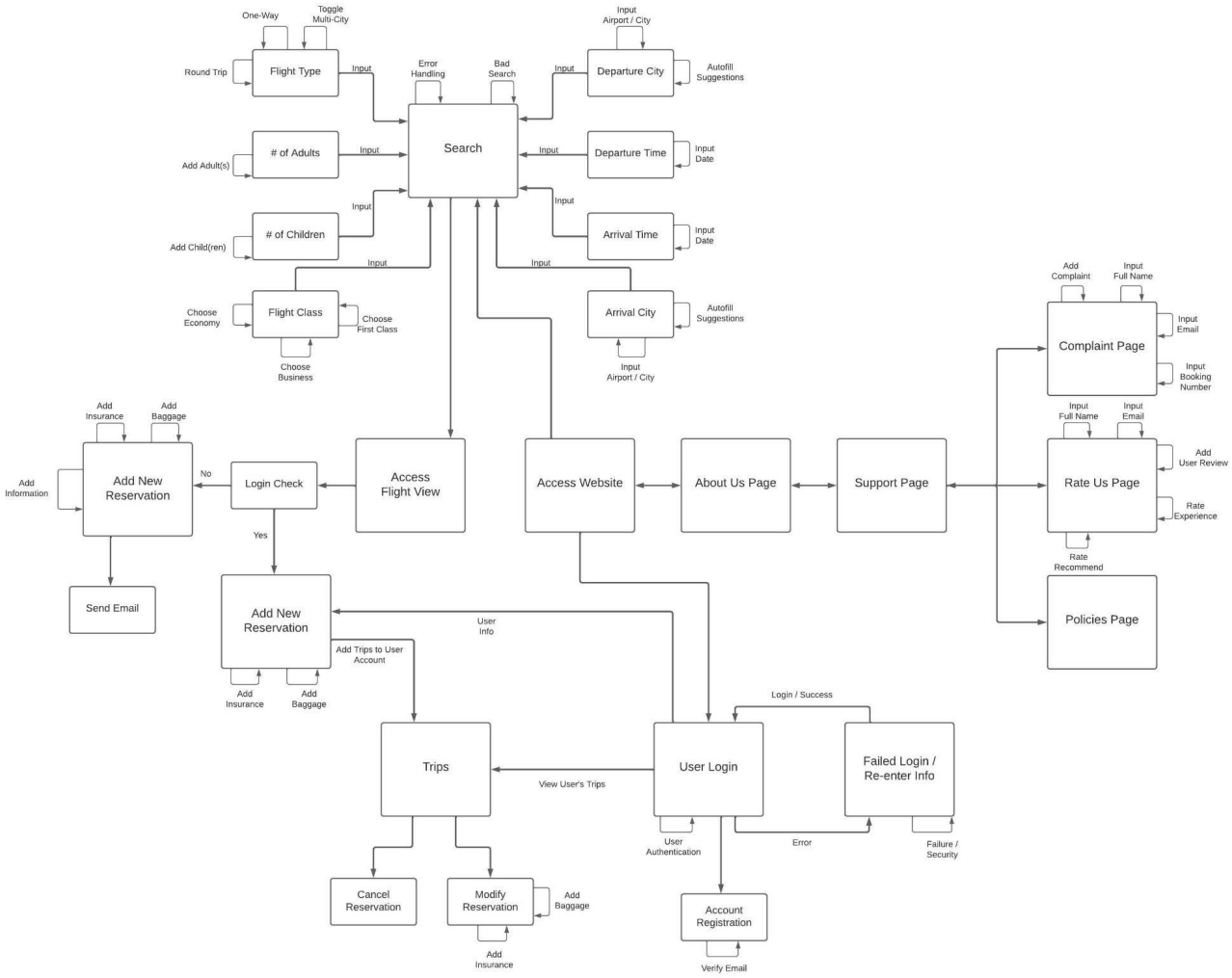
Number	Revision	Requirement Description	Bad Client Attack	Bad Developer Attack	Approved	Sign-Off Name
14.1.1	1	Should anything malfunction, developers from the appropriate role will fix the problem.	N/A	What if no one has that specific role who takes care of it?	NO	NOT APPROVED David Gawel 1/25/2022
14.2.1	1	There will be a developer account provided to all developers of the application	N/A	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
14.2.2	1	Only users that have Dan's domain name will be able to access the applications API.	N/A	Only request sent from dans domain can access the api	NO	NOT APPROVED David Gawel 1/25/2022
14.2.2	2	Only requests that have Dan's domain name will be able to access the applications API.	N/A	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
14.3.1	1	Complicated code will be commented on.	Client could argue some code isn't commented. What makes the code complicated?	What counts as complicated? dev may argue that all the code is simple and does not need	NO	NOT APPROVED Nick Clark 1/24/2022
14.3.1	2	All REST endpoints will be commented to explain their purpose, parameters, and return variable. Developers can add additional comments as they	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022
14.3.2	1	Utilize Javadocs to explain functions, methods, and algorithms.	Ok passes bad client attack	Ok passed bad dev attack	YES	APPROVED David Gawel 1/25/2022

Example of 14.0 Documentation Requirements Matrix

2.2 Design

2.2.1 Finite State Machine

A finite state machine represents the various resting states of the system and provides an abstract perspective on the flow of states. The user inputs their activity and arrives at certain states depending on the sequence of actions. The finite state machine depicted in this project manual represents the functionality flight reservation system and multiple states that occur within the system. The system requirements presented in the requirements document are visually represented through the finite state machine diagram and the relation between system requirements are shown through this abstract perspective.

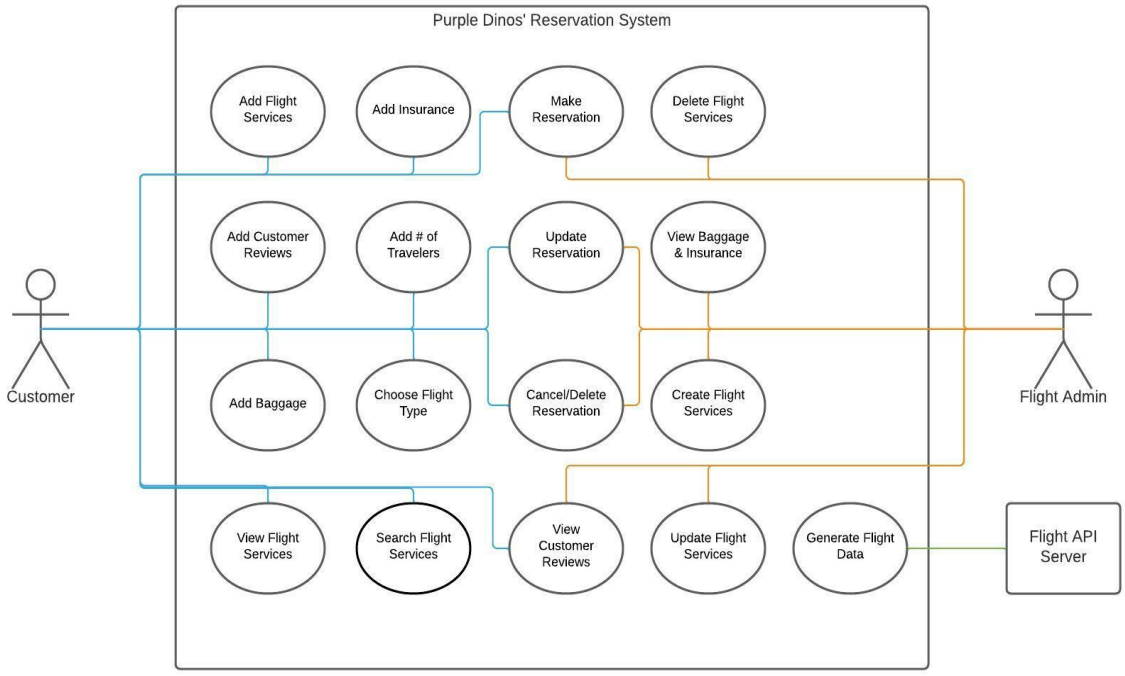


2.2.1 Finite State Machine Diagram

2.2.2 UML

2.2.2.1 Use-Case Diagram

When starting the design of our software, it was important to understand the various interactions that could occur within our system. Such examples would be, A Customer searching for flights or a Customer adding the number of travelers for the flight reservation. Using this document, we can design what was needed from the system. Therefore, a use-case diagram was drafted so we could understand the direction of the system. The use-case diagram present seen within the document represented the first ideas about the flight reservation system. As we moved further into the project, we realized that the flight administrator didn't need many use-cases presented in the system.



2.2.2.1 Use-Case Diagram

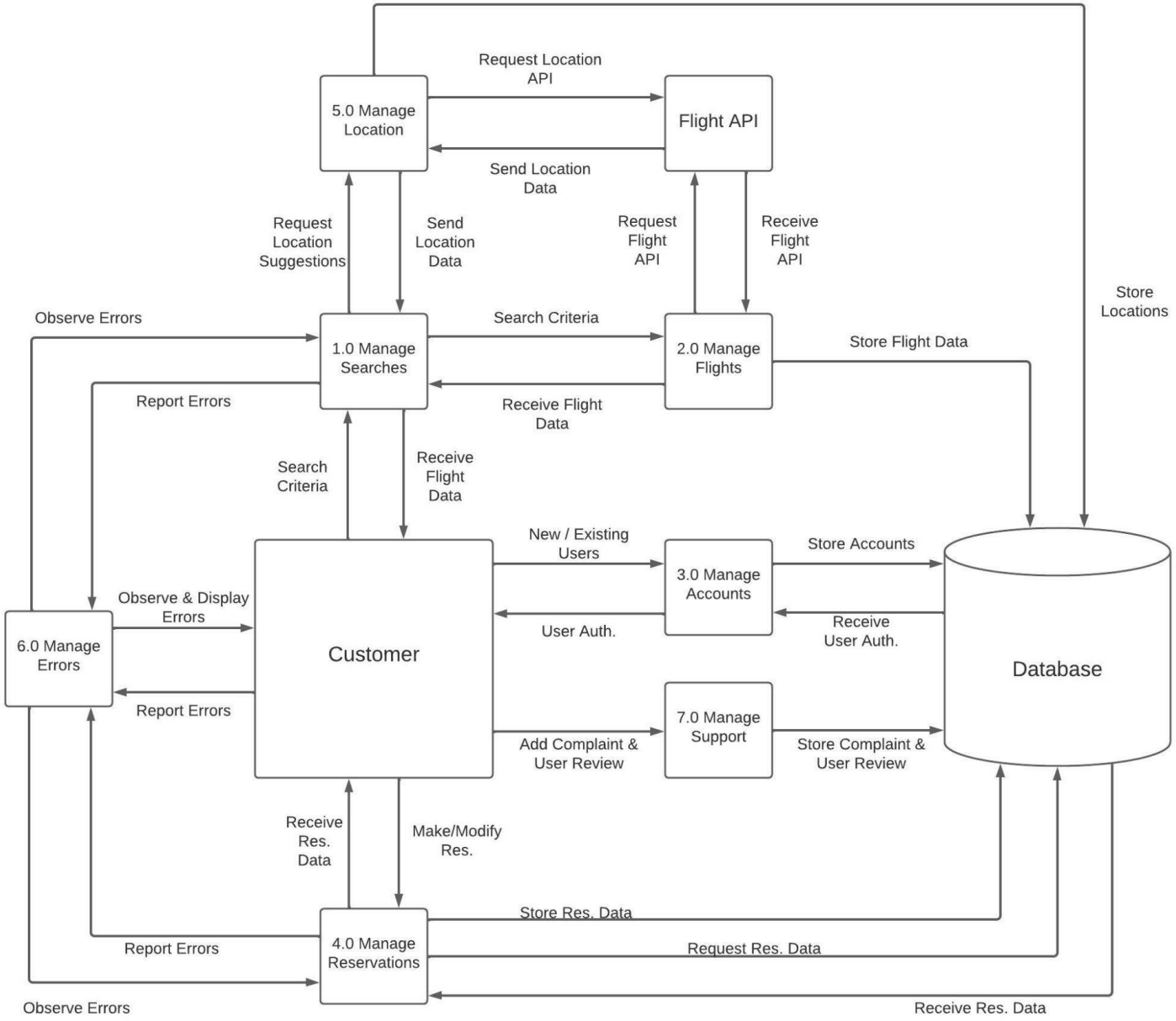
2.2.2.2 Level 0 Diagram

This level 0 diagram shows a high level overview of the Purple Dinos flight reservation system. The diagram broadly represents the 7 major modules and the relationships between the customer, database, and other modules. There are three components of the flight reservation system that are seen at the high level: Utilizing the Flight API, Interacting with the Customer/User, and Database Management.

At the top of the diagram is Flight API; of which, the entity is an external resource used for the support of the flight reservation system. The Flight API we're using is called Amadeus API and it is utilized by three modules—Manage Searches, Manage Flights, Manage Location. The initial search criteria are sent to the Manage Searches Module which then delegates parts of the search to the module Manage Location and the module Manage Flights. Both Manage Location and Manage Flights receive and utilize the data received from Amadeus API to find a flight that matches the search criteria

With the customer at the center of the level 0 diagram, the relationships between the customer and the system are an important aspect. The flight reservation system is a web application that was made to be utilized by the Customer/User. For the web application, it should support web browsers such as Google Chrome, Microsoft Edge, Firefox, and Safari. The customer can interact with Manage Reservations, Manage Accounts, and Manage Searches to utilize the web application. The Manage Errors module also interacts with the customer by observing the status customer and the modules they interact with.

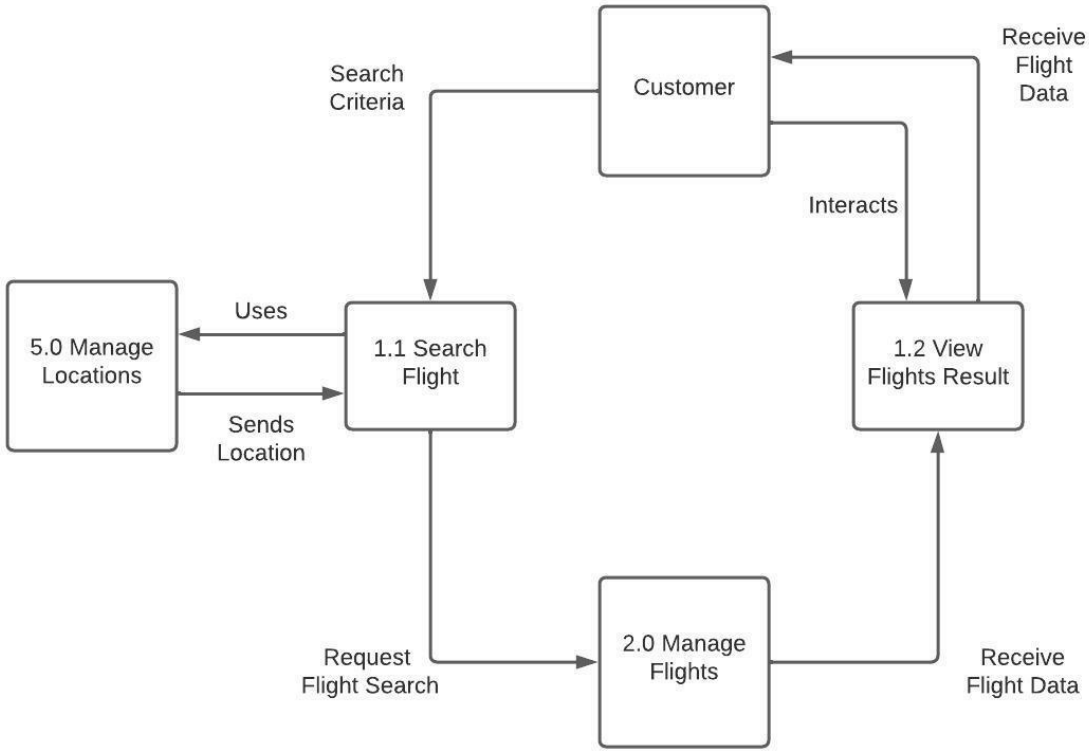
Database management is an important component of the flight reservation system because of the system's reliance on the SQL database to store and transfer data. Any necessary data that can be used later is saved in the database such as user authentication, reservations, location, and flights. In the Level 0 diagram, the modules that handle the necessary data also interact with the database.



2.2.2.2 Level 0 Diagram

2.2.2.3 Manage Searches DFD

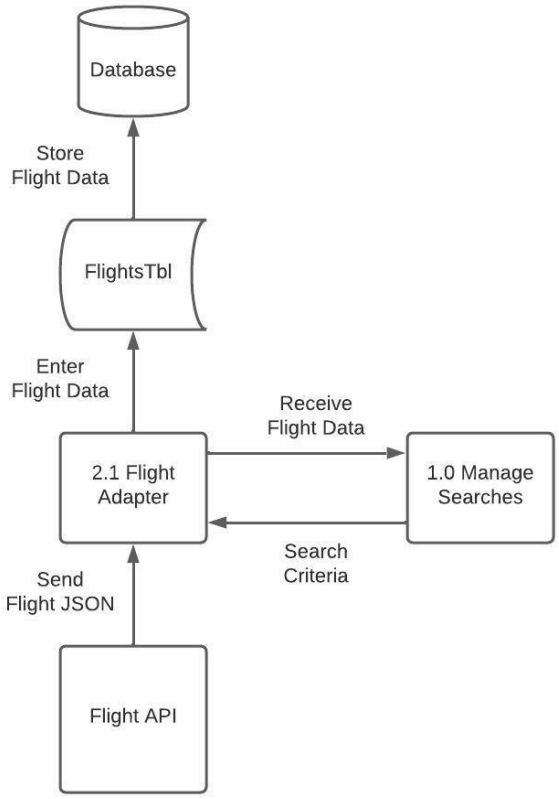
The Manage Searches module takes input from the Customer/User which contains the search criteria, and the data that flow through this module revolves around the information in the search criteria. The module would begin its dataflow once it takes input from the Customer/User. The first contact is with the sub-module Search Flight where it would clean up the user input into an object that can be found in the Amadeus API. The sub-module uses Manage Location to help suggest potential locations and airports that the Customer/User is searching for; meanwhile, the sub-module interacts with Manage Flights by requesting flight data using the necessary information. Once the major module Manage Flights is done processing the flight data that matches the search criteria, it sends it to the sub-module View Flights Result. View Flight Results receives the flight data and presents the data to the Customer/User who then can interact with the flights.



2.2.2.3 Manage Searches Data Flow Diagram

2.2.2.4 Manage Flights DFD

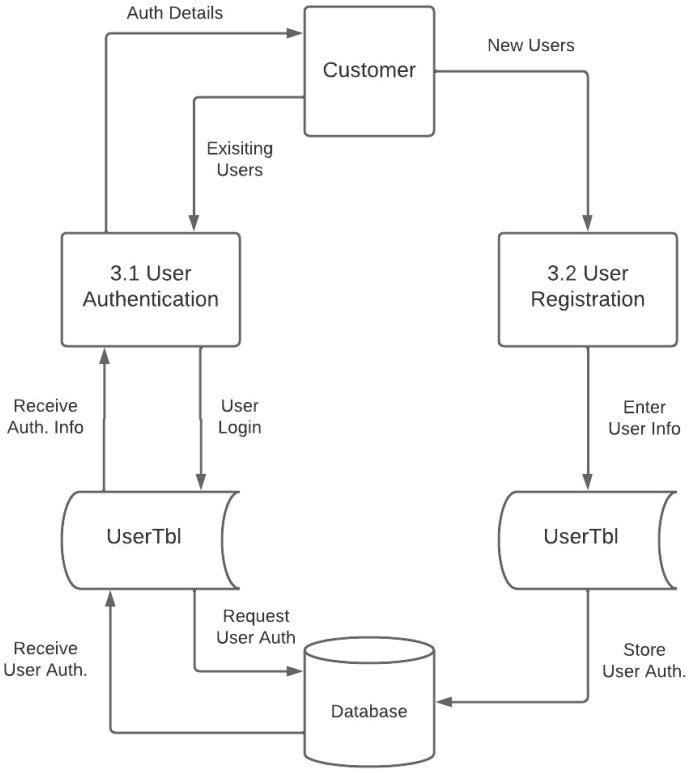
The Manage Flights module is used by the Manage Searches module to search for related flights according to the search criteria. It interacts with the Flight API we're using to support our system (Amadeus API). In the sub-module, Flight Adapter is given the search criteria for the flight from Manage Searches. Using the search criteria, the sub-module would make the appropriate requests to the Amadeus API and receive the flight data in the form of a JSON Array of JSON Objects. The Flight Adapter sub-module would then parse through the JSON and acquire all flight data that relates to the search requirements. Sending all related flight data to the Manage Searches while also storing the flight data in the database within the Flight Table.



2.2.2.4 Manage Flight Data Flow Diagram

2.2.2.5 Manage Accounts DFD

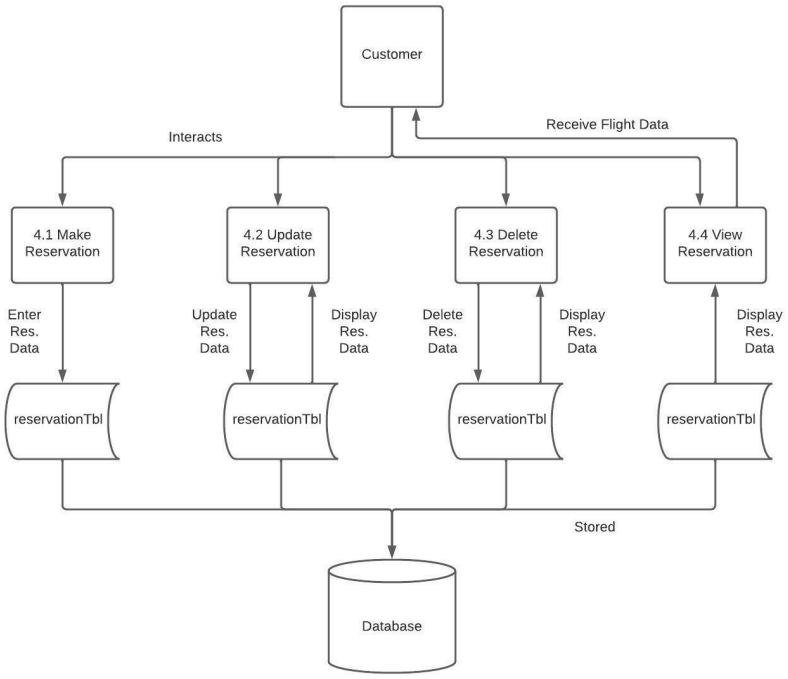
In the module, Manage Accounts, new and existing users would interact and either create or login to their accounts. If the Customer/User is creating a new account then they would flow into the User Registration sub-module. In User Registration, we plan to have the new users register with their Gmail accounts. Verifying that the Gmail belongs to them and confirming they want to use that Gmail for their account. The login information is then stored in the User Table of the database. If logging in, then the existing user would be sent to the User Authentication sub-module. The sub-module would search for the user's information in the User Table of the database and find the related information. Receiving the authentication info from the User Table in the database, the sub-module would compare the login details and either fail or pass the existing user into the account.



2.2.2.5 Manage Accounts Data Flow Diagram

2.2.2.6 Manage Reservation DFD

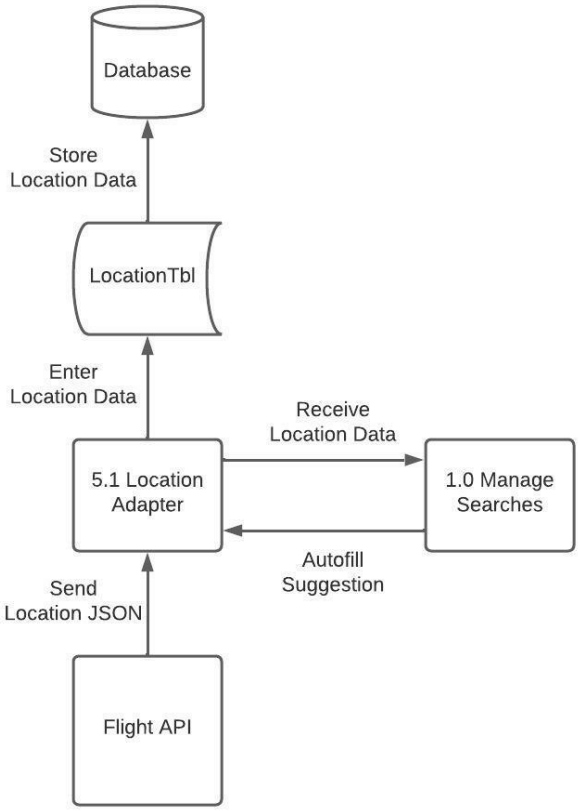
The major module, Manage Reservations, manages both the creation and the modification of reservations. The Customer/User can interact with the Manage Reservation module in many ways. Beginning with the first sub-module, Make Reservation, this allows the Customer/User to create a new reservation. After the customer creates a reservation, the reservation data is entered into the reservation table and stored in the database. The second sub-module, Update Reservation, sees the user make changes to a flight reservation they had completed prior. Using the reservation data stored in the database, the user then makes any changes they feel necessary. Once the user is done modifying the reservation, the sub-module would update the reservation table with the new reservation data. The third sub-module, Delete Reservation, allows the user to cancel any reservation made prior that hasn't expired. Reservation data is pulled from the reservation table and the Customer/User is shown which flight reservation they want to cancel. Once they cancel a flight reservation, the sub-module deletes the reservation data from the database. The last sub-module in Manage Reservations, View Reservation, allows the Customer/User to view all flight reservations that they've made prior. The sub-module gathers the reservation data from the reservation table in the database and displays it. The Customer/User would then receive the information of the flight reservations they've made.



2.2.2.6 Manage Reservation Data Flow Diagram

2.2.2.7 Manage Location DFD

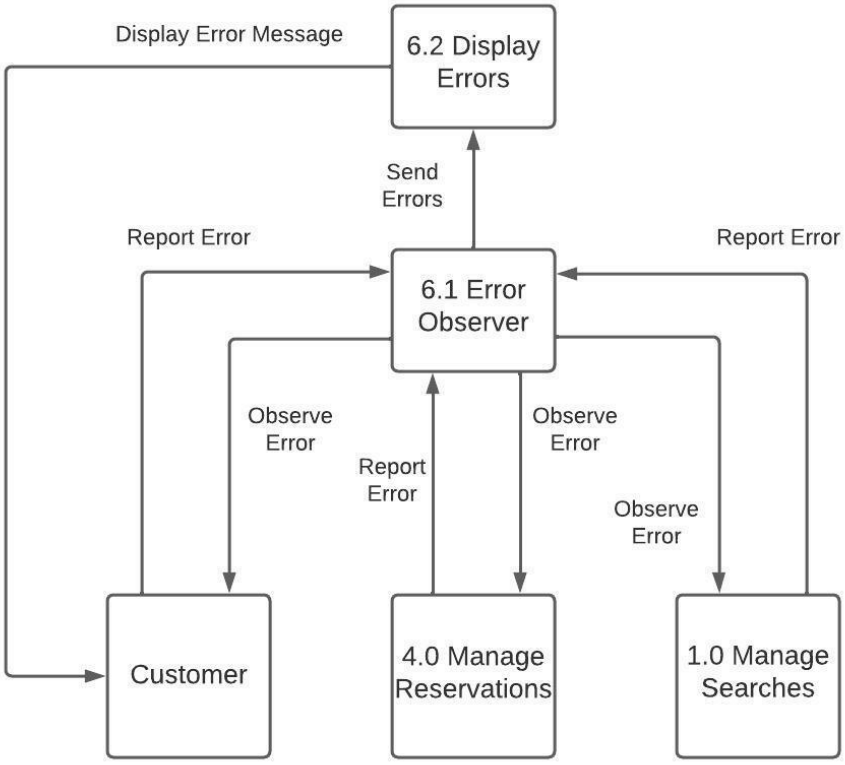
The Manage Location module is used by the Manage Searches module to search for related locations according to the search criteria. It interacts with the Flight API we're using to support our system (Amadeus API). In the sub-module, Location Adapter, it is given the partially filled information for the city or airport from Manage Searches. Using the information, the sub-module would make the appropriate requests to the Amadeus API and receive suggested location data in the form of a JSON Array of JSON Objects. The Location Adapter sub-module would then parse through the JSON and acquire all airports and cities that contain or relate to the information sent. Sending all related location data to the Manage Searches while also storing the location data in the database within the Location Table.



2.2.2.7 Manage Location Data Flow Diagram

2.2.2.8 Manage Errors DFD

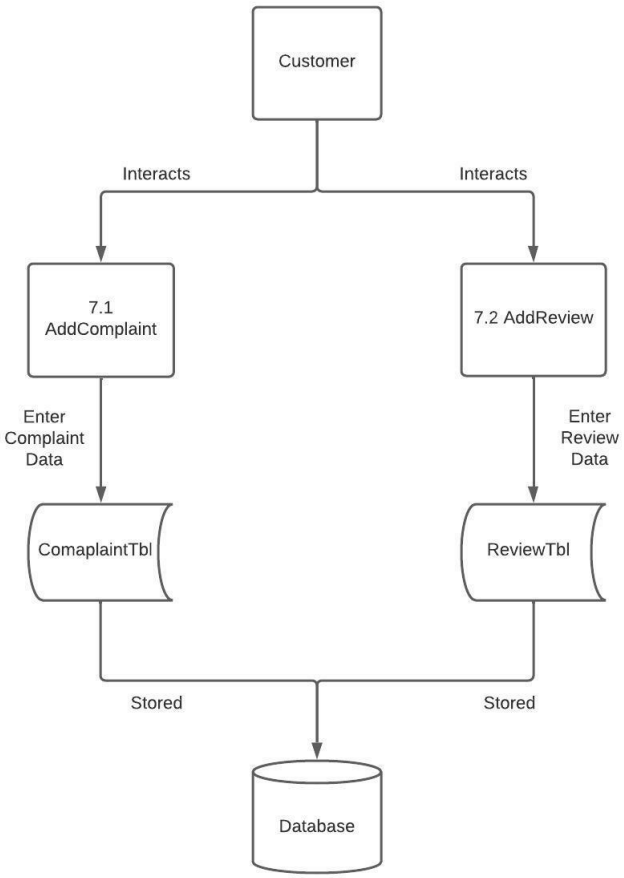
The Manage Errors module is used to observe and handle errors that may occur in error-prone modules such as Manage Searches and Manage Reservations. Through the sub-module, Error Observer, it can observe and receive reports from multiple objects within the system. For Customer/User, it monitors whether the user is still connected to the network. For Manage Reservation and Manage Searches, it monitors the information inputted for the reservation or searches and receives reports if required info is missing. Once the Error Observer receives an error from an object or module, it would then send the errors to the Display Errors sub-module. Within the Display sub-module, it would recognize the error and send an error message to the Customer/User that details the specific error.



2.2.2.8 Manage Errors Data Flow Diagram

2.2.2.9 Manage Support DFD

The Manage Support module is used to handle support related features such as complaints and reviews. It focuses on taking input from the Customer, for a complaint or review, and sending the data into the database. There are two sub-module that are contained in this major module: Add Complaint and Add Review. Both sub-modules require the customer's interaction before data can be stored into the database. The complaints or reviews that are stored in the database are used as feedback to fix issues or improve the quality of the flight reservation system.



2.2.2.9 Manage Support Data Flow Diagram

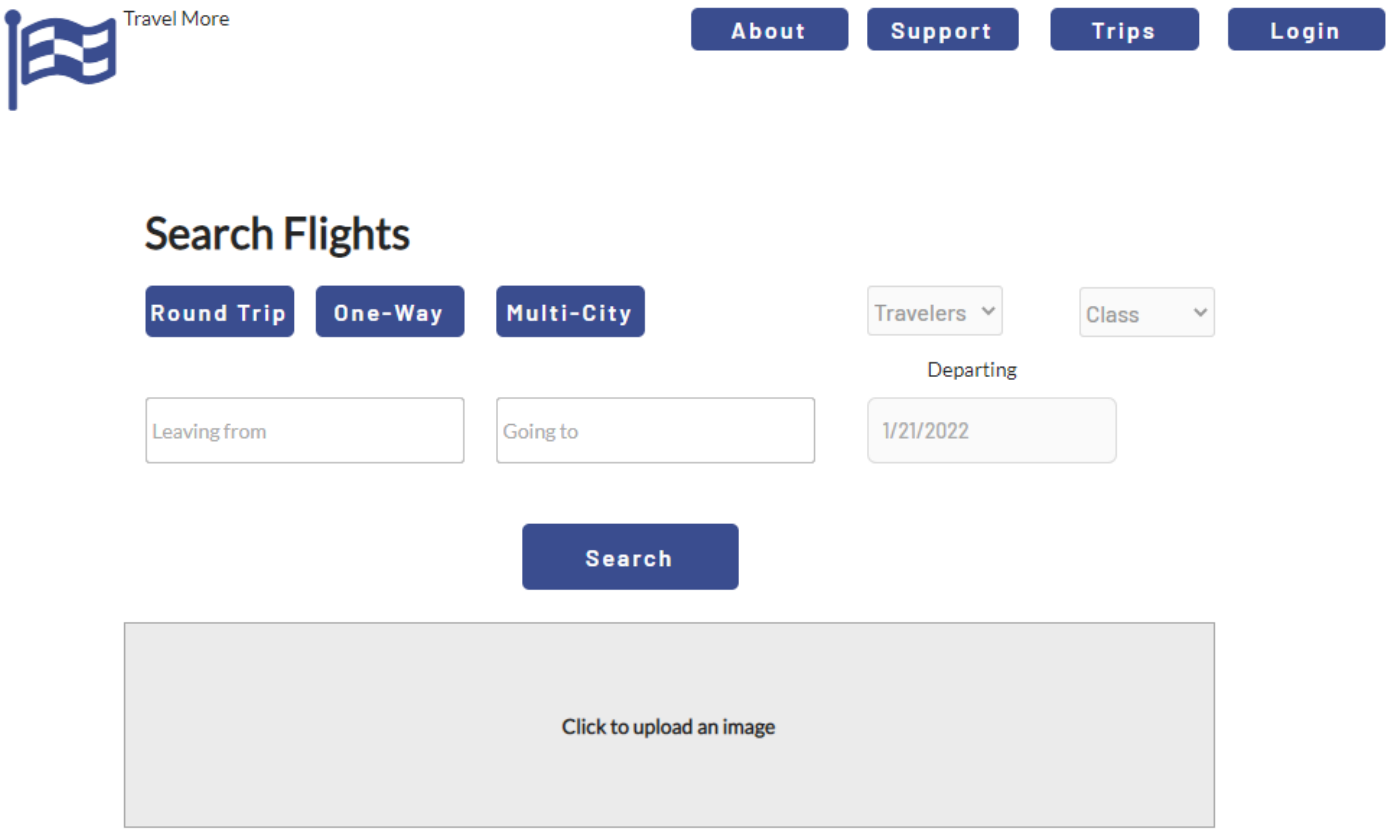
2.2.3 Conceptual Redesigns

Since design documents aren't perfect from the beginning, most of the UML diagrams have undergone revisions. Revisions took place whenever feedback was received on a previous draft or new requirements were discussed. The concept of the flight reservation system's design, both front-end and back-end, are bound to change as time moves on; regardless of if those changes require the inclusion of new features or the exclusion of others. With revisions, the design documents were able to represent the flight reservation system that was being built.

2.2.4 Front End Design

2.2.4.1 Original Front Page Design

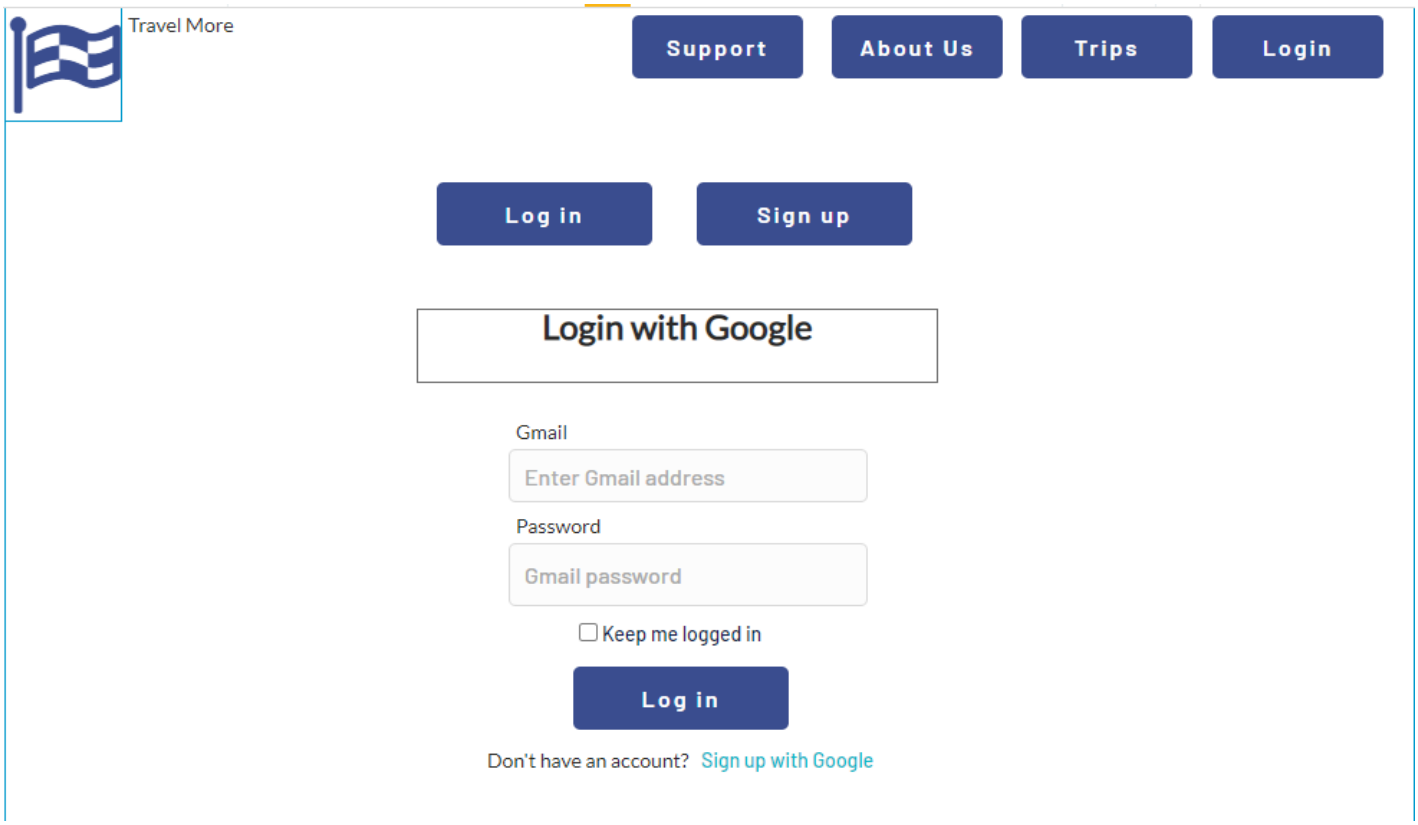
Overall design of the website homepage with placement logo. The homepage is designed to be fitted within one view for the user to be able to take note of the actions they need to take without any distractions. The user is able to click on the various buttons on the top right but will be restricted until they are logged in. The user can select from three types of flights.



2.2.4.1 Homepage wireframe

2.2.4.2 Original Login Page Design

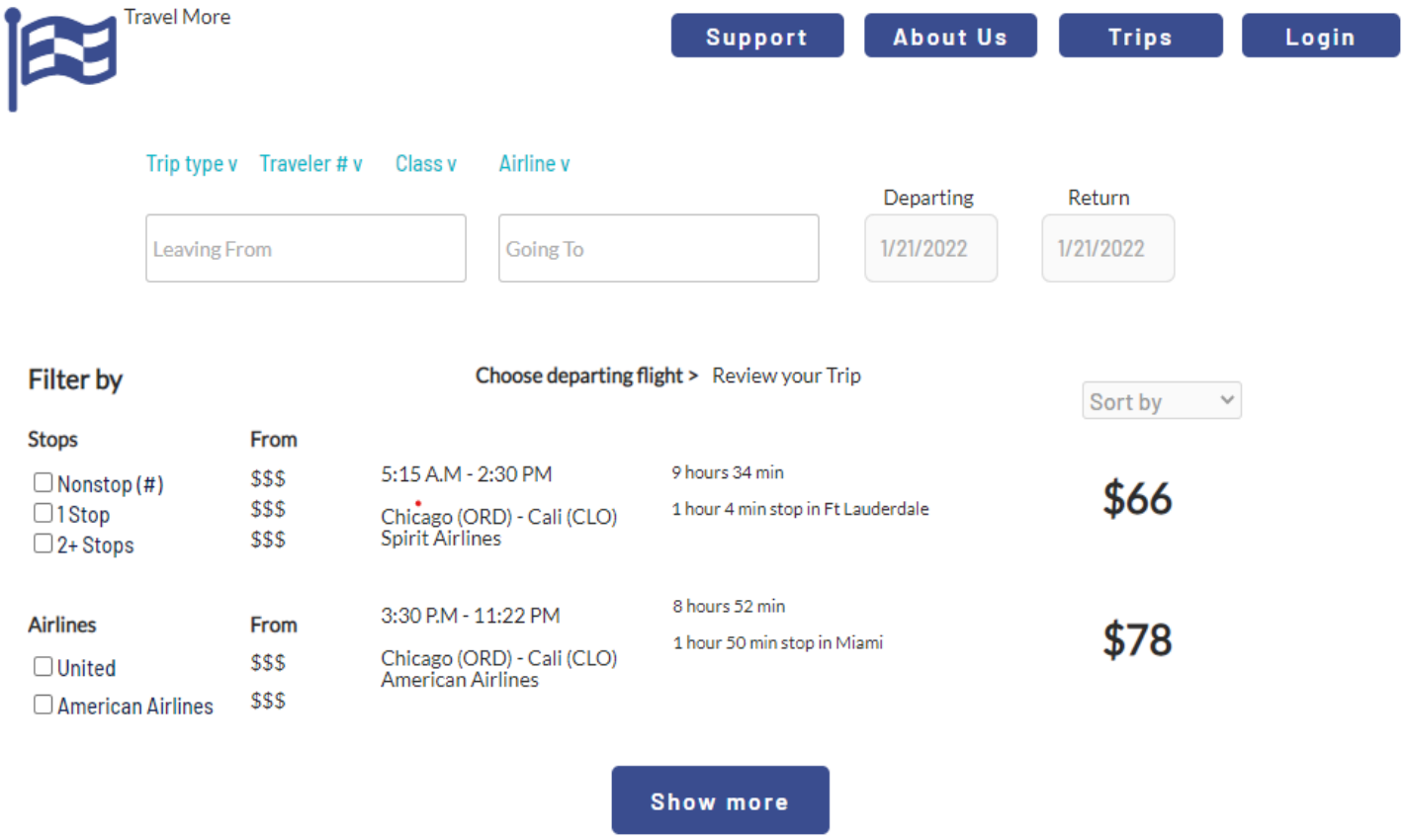
Original login page design that lets the user log in by entering their gmail account or clicking on the Google button to use google's login with authentication. The user was also supposed to be able to Sign Up for the website and stay logged in if the input was selected.



2.2.4.2 Original Login Wireframe

2.2.4.3 Search Page Design

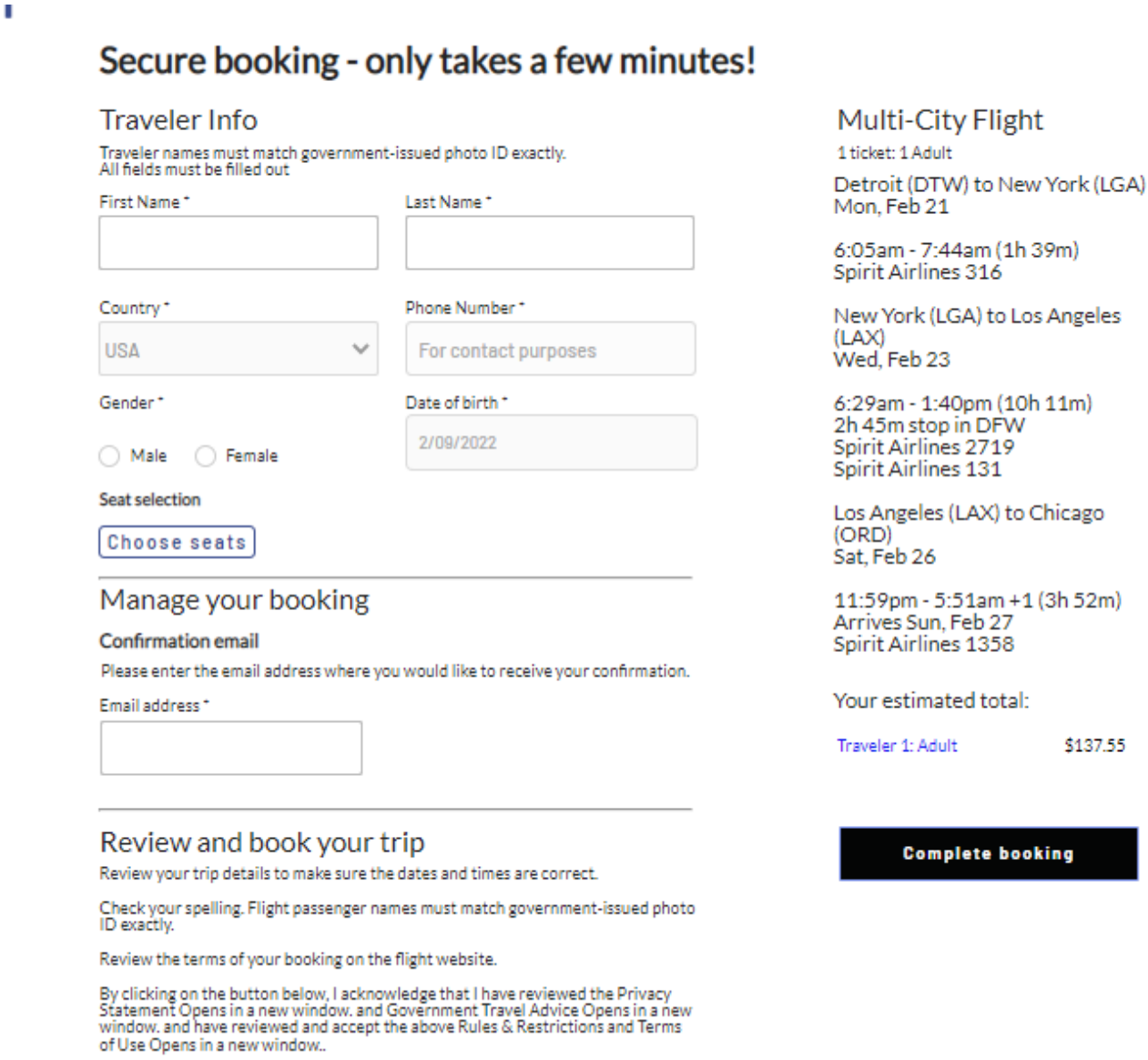
One-way search page wireframe with filters and selection bolded to clarify which flight we are choosing. The user is able to view relevant flight information on this page such as the price, dates, time, as well as having a variety of filters to make their selection easier to narrow down.



2.2.4.3 Original One-way search Wireframe

2.2.4.4 Original Review page Design

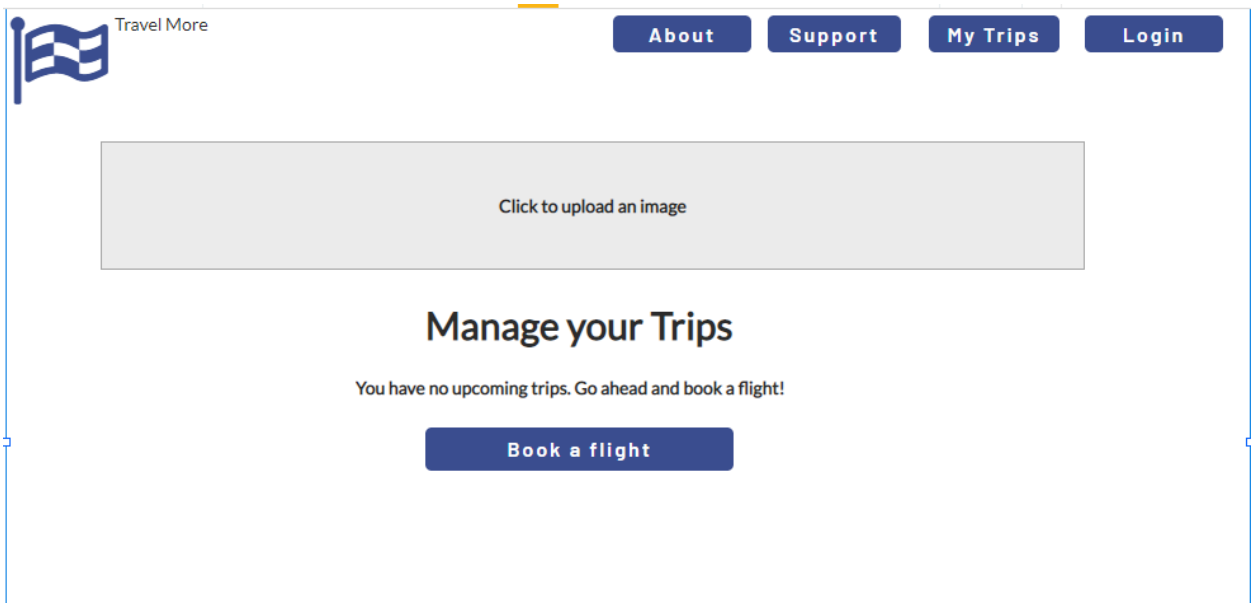
Conceptual design of the review page where the user will go on to confirm their flight selection and complete their booking. The page is designed to collect the users information and allow the inputs to gain their traveler info. The design includes aspects such as email address that will be used to forward confirmation as well as being able to select available seats. Users can also see detailed flight information on the right with their estimated total.



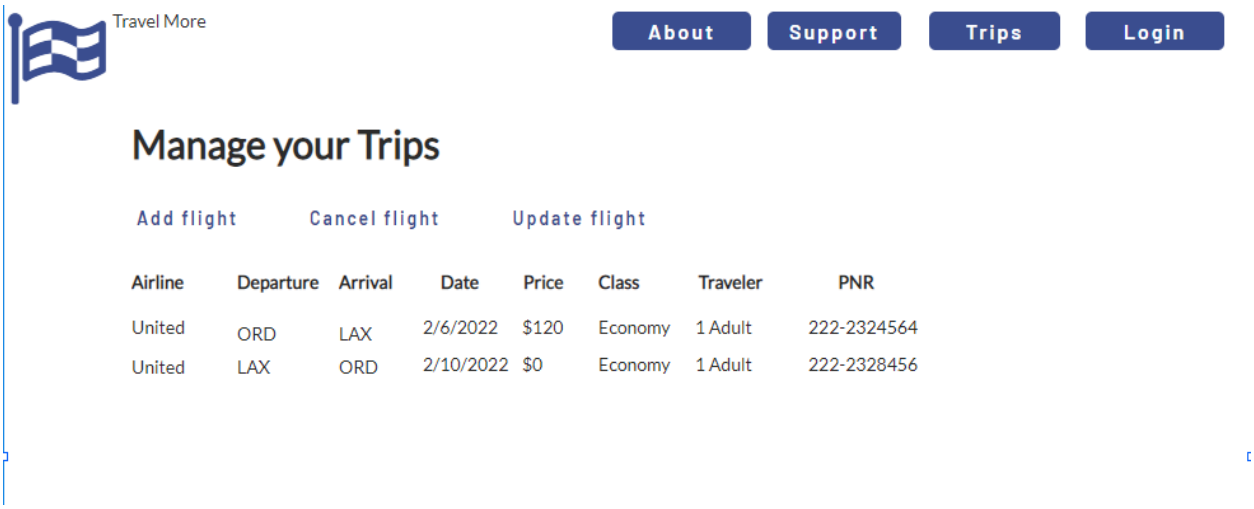
2.2.4.4 Original Review Page Wireframe

2.2.4.5 Original Trips Page Design

The Trips page is not accessible without being logged in first. The user gets a page that tells them to log in first if needed. Otherwise, the user is able to view their booked flights and have the option to add a flight, edit a flight with certain changes, and cancel a flight. Once the user has made a reservation, the trips page is accessible with all of their flights.



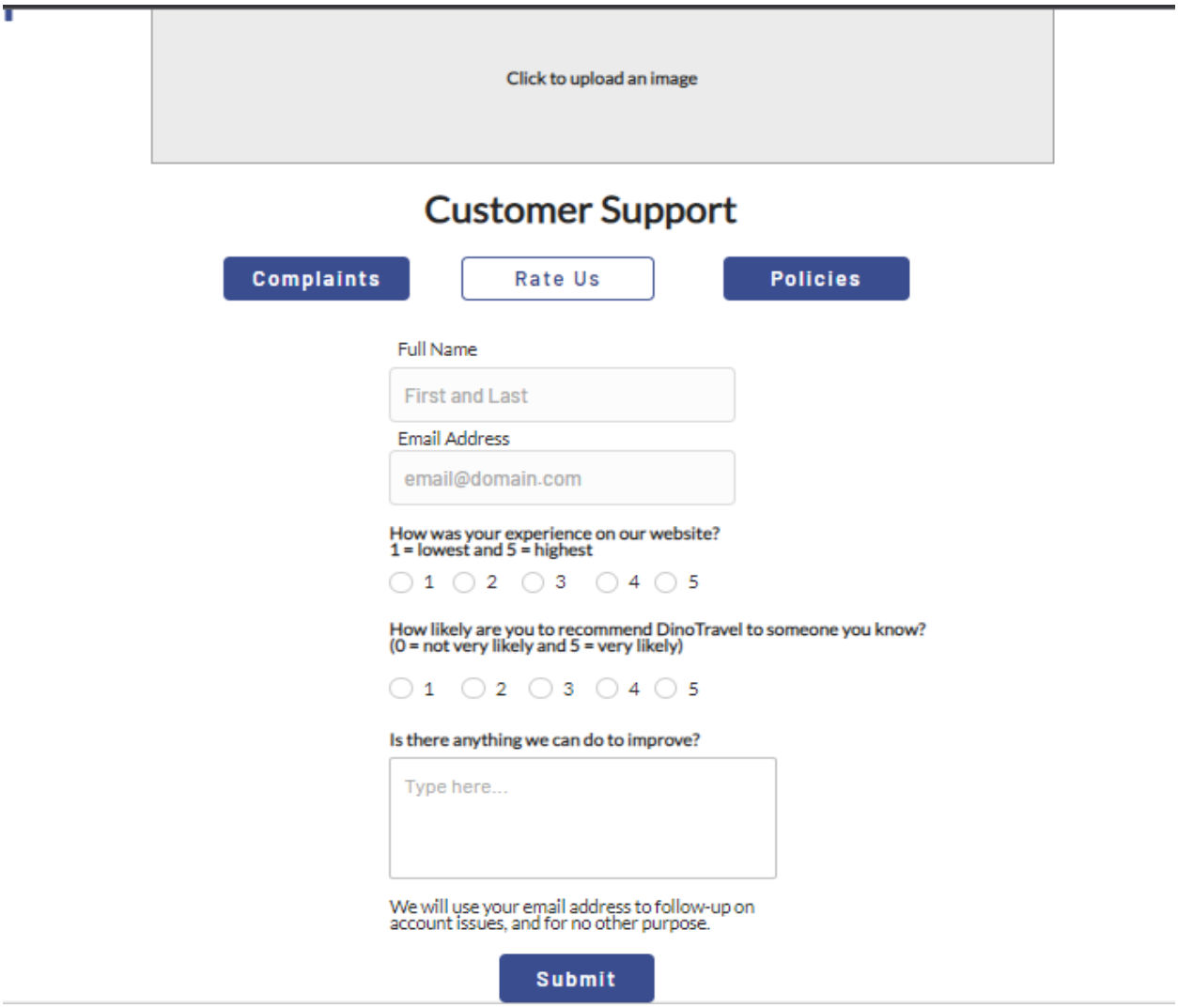
2.2.4.5 Original Trips Page (No login) Wireframe



2.2.4.5 Original Trips Page Wireframe

2.2.4.6 Original Support Page Design

The support page has a simple design with 3 clickable buttons. There are two forms to fill out for either a complaint in DinoTravels services, and a feedback form that lets us know how well our website is doing based on customer reviews. The third button lets us view policies for using the website as well as guidelines on travel.



2.2.4.6 Original Review Page Wireframe

2.3 Presentations

There were 4 Major presentations over the 10-week duration of this Capstone course.

1. Proof of Concept Demo
2. Requirements, Design, and Planning Demo (RDP Demo)
3. Preview Demo
4. Final Demo

A presentation outline was created for each of these demos, with each outline containing a comprehensive breakdown of what was to be covered in the presentation.

2.3.1 Presentation Outlines

2.3.1a Proof of Concept Demo Outline

DINO TRAVEL CONCEPT DEMO PRESENTATION OUTLINE

I. Intro: PHIL

a. Suhaib shows the site logo (**2 secs**)
b. Phil Introduces group members (Pictures and Roles) (**40 secs**)

II. Content

a. Jackson Gives an overview of the project (**50 secs**)
b. Jenna goes over the Time log and Calendar (**1 min 15 secs**)
c. Jawan talks about the UML diagram (**1 min 30 secs**)
d. Mario talks about the project requirements (**1 min 30 secs**)
e. Mohammed explains the wireframe (**1 min 25 secs**)
f. Dan does the testing of the frontend and demonstrates (**1 min 45 secs**)
g. David demonstrates the backend and talks about our data backup system (**1min 45 secs**)
h. Suhaib and Nick Compile all on the video no later than Saturday January 8, 2022, at 10:00pm.

NOTE: All videos should be in MP4 format and sent to Suhaib. Suhaib, please coordinate with Nick to start working on the video and sound editing once you have received the videos. Also, you can use less than the allotted time, but you cannot exceed the time you are given.

Please Reintroduce yourself when you are doing your recording.

2.3.1b RDP Demo Outline

REQUIREMENTS, DESIGN, AND PLANNING (RDP) PRESENTATION OUTLINE

I. Suhaib shows site logo across the screen (1 sec)

a. Suhaib shows pictures of Jenna, Jawan, and Mario with their roles **(1 sec)**

II. Jenna briefly introduces herself stating her name and role (5mins)

a. Show the group view of the calendar by clicking on any part of the calendar to show details of who is doing what on a specific date.
b. Show the individual view of one group member so that the viewer can clearly see the collection of tasks for which that individual is responsible, and what their path through the calendar looks like.
c. Show the dependencies for an interesting module, showing what task must be completed first, and which other modules depend on this module being complete. For example, Mohammed must complete his wireframe before Dan can create the front end of the website.
d. Now a random person is going to pick a section of the calendar (This is a hypothetical scenario): Show the group view of the calendar by clicking on any part of the calendar to show details of who is doing what on a specific date.
e. This is a hypothetical scenario with an audience member picking: Let an audience member pick an individual and show us the view of that member so that we can clearly see the collection of tasks for which he is responsible, and what their path through the calendar looks like.
f. This is a hypothetical scenario with an audience member picking a module: Show the dependencies for a module that is chosen, showing what else must be completed first, and which other module(s) depend on this module being complete.
g. Demonstrate how your plan acknowledges completed tasks in the group forum.
h. Give a static (or dynamic if you have software) example of replanning, and how dependencies are rescheduled.

i. PASS THE PRESENTATION ON TO JAWAN.

III.Jawan Briefly introduces himself stating his name and role (7mins 25 secs)

a. Present a top-level slide (e.g., level 0 DFD, context diagram)
b. Show the full project design (Very high level, very complete, global coverage) and discuss each briefly.
c. Teach viewers three or four main components of the design.
d. Clicks on" a portion of the design [which contains a hyperlink] to retrieve a more detailed design representation [level 1] of a portion of the project, and briefly explain how that portion of the project design works.

e. If you have level-two diagrams, click on a portion of the more detailed design to retrieve a very detailed design representation (level 2) of a very specific part of the project, and explain briefly.
f. Click on a portion of the very specific design representation and retrieve the code (HTML) implementation for that module.
g. Returns to the top level and asks an audience member to pick ANY displayed module, at each level, and repeats the cycle above, down to the level of code, and explain briefly. NOTE: this is just a simulation of selecting an audience member. You can be creative here**.**
h. **PASS THE PRESENTATION ONTO MARIO**

IV. Mario briefly introduces himself stating his name and role (7 Mins 25 secs)

a. Give an overview of all the requirements.
b. Show an overview of each individual requirement
c. Give an overview of your full requirements test suite (on website)
d. Show viewers how *individual* tests are specified to highlight the quality of the testing design
e. Show how the requirements are matched with the tests (use the matrix)
f. Show repeated tests of a requirement (show failure and success if necessary)
g. Select some interesting individual requirements and show testing
h. Discuss the formal schedule for the Bad Client review work. Who, dates, revisions, responses (show using Matrix).
i. Discuss the formal schedule for the Bad Dev review work. Who, dates, revisions, responses (show using Matrix).
j. Discuss the work of the requirements editor. Show example history of revisions based on, e.g., the work of the Bad Client and Bad Dev (show using Matrix)

V. Suhaib shows all the group members with their roles on a single slide (8 secs)

a. Edit the video and audio to meet project requirement (20mins)
b. The video should be uploaded, and backups should be made available
c. Update Phil if you if you have software issues when editing the video and audio
d. Update Phil of any issues you have with backups and uploading the video to D2L

NOTE: Jenna video should be turned in to Suhaib NLT Thursday, January 27, 2022 at 10:30pm, while Mario and Jawan video should be submitted NLT Saturday, January, 29, 2022 at 10:30pm. Point mouse pointer to where you are to help the viewers follow along. Incorporate a story while presenting to make the presentation more interesting. At the bottom of the page are the basic requirements needed for this phase if Mario would like to reference them. Also, if Mario and Jawan decide to dedicate a segment of their presentation to a group member, it is your duty to allocate some of your 7 mins 25 secs to that individual. Let group members know if you have a few minutes or seconds that you didn't use so they can use it. Lastly, anyone added

to the presentation other than Jenna, Jawan, and Mario should also be included in the intro slides by Phil. Let Phil know ASAP if you do intend to have someone helping with a segment of your presentation so he can include them into the intro slide. IF YOU WILL SHOW YOURSELF IN THE VIDEO, DRESS APPROPRIATELY. LET PHIL KNOW IF YOU NEED HELP WITH REHEARSAL.

2.3.1c Preview & Final Demo Outline

DINO TRAVEL PREVIEW/FINAL DEMO OUTLINE (20 mins)

I. Dino Travel logo displays 1 Sec
II. Display Image of Jackson, Jenna, Jawan, Mario, Muhammad, Dan, David, and Nick with their name & position 2 Secs
III. Transition word HOW IT STARTED 1 Sec
IV. Give a brief overview of our project

a. Business Plan **(Jackson) 1 Min PASS IT ON TO JENNA**
b. Plan shown and a brief explanation since it has been discussed previously in detail in RDP demo **(Jenna) 45 Secs**
c. Show a more convincing and detailed time log **(Jenna) 45 Secs PASS IT ON TO JAWAN**
d. Design presented and a brief explanation since it has been discussed previously in detail in RDP demo **(Jawan) 1 min 25 Secs PASS IT ON TO MARIO**
e. Requirements shown and a brief explanation since it has been briefly discussed previously Focus on **Bad Dev/ Bad Client, Revisions** & etc. **(Mario) 1 Min 45 Secs PASS IT ON TO THE MUHAMMAD**

V. Transition word HOW IS IT GOING

a. Human Computer Integration (HCI) usability study **(Muhammad) 2 Mins 30 Secs**
b. Discuss how client side is implemented **(Muhammed**) **2 Mins 30 Secs PASS IT ON TO DAN**
c. Show code successfully and discuss its relevance to the project **(Dan) 1 Min 30 Secs PASS IT ON TO DAVID**
d. Discuss the security model **(David) 2:00 mins**
e. Show Apparent Backups **(David) 2:00 mins PASS IT ON TO NICK**
f. Discuss how the full CRUD system is implemented (**Nick) 2:00 mins**
g. Discuss how sever side is implemented **(Nick) 2:00 Mins**

VI. **Suhaib**

a. Shows all the group members with their roles **2 secs**
b. Display a **THANK YOU 1 Sec**
c. Edit the video and audio to meet project requirement (20mins)
d. The video should be uploaded, and backups should be made available
e. Update Phil if you if you have software issues when editing the video and audio
f. Update Phil of any issues you have with backups and uploading the video to D2L

NOTE: Jackson, Jenna, Jawan, and Mario video should be turned in to Suhaib NLT Friday, February 18, 2022, at 11:30pm, while Muhammad, Dan, David and Nick video should be submitted NLT Saturday, February 19, 2022, at 11:30pm. Point mouse pointer to where you are

to help the viewers follow along. Incorporate a story while presenting to make the presentation more interesting. IF YOU WILL SHOW YOURSELF IN THE VIDEO, DRESS APPROPRIATELY. LET PHIL KNOW IF YOU NEED HELP WITH REHEARSAL.

2.3.2 Videos

The Video Manager was in charge of collaborating with every member of the group in order to make sure everything was formatted correctly and receiving everyone's video in order to start editing. The programs used to edit videos were Adobe After Effects, Sony Vegas Pro, and VN Editor. Adobe Photoshop was used for all image editing and opening and closing cards. An audio editing program called Audacity was used to make sure audio was consistent and high-quality.

Before starting to edit anything, a meeting with the Presentation Manager would take place in order to finalize and get the presentation outline. This was used to create a well-paced video which also stuck to the specific time constraints of each demo. After finalizing the outline for the video, the next step was finding royalty-free music through YouTube's Audio Library. After the music was found, all the clips for the video were imported and formatted to ensure all videos were the same aspect ratio. Then each clip was passed over to make sure they all had the same audio level. Once all formatting was completed, the music was added in. Once all video and audio aspects were complete, transitions and titles would be added for each member. After the first draft of the video was completed, it would be rendered and shared with the group for feedback. If there were comments or suggestions from group members for fixes, they would be addressed, re-rendered, and finally approved for submission. Afterwards, they were uploaded to Panopto (as per the requirements from the Professor), and YouTube as a backup.

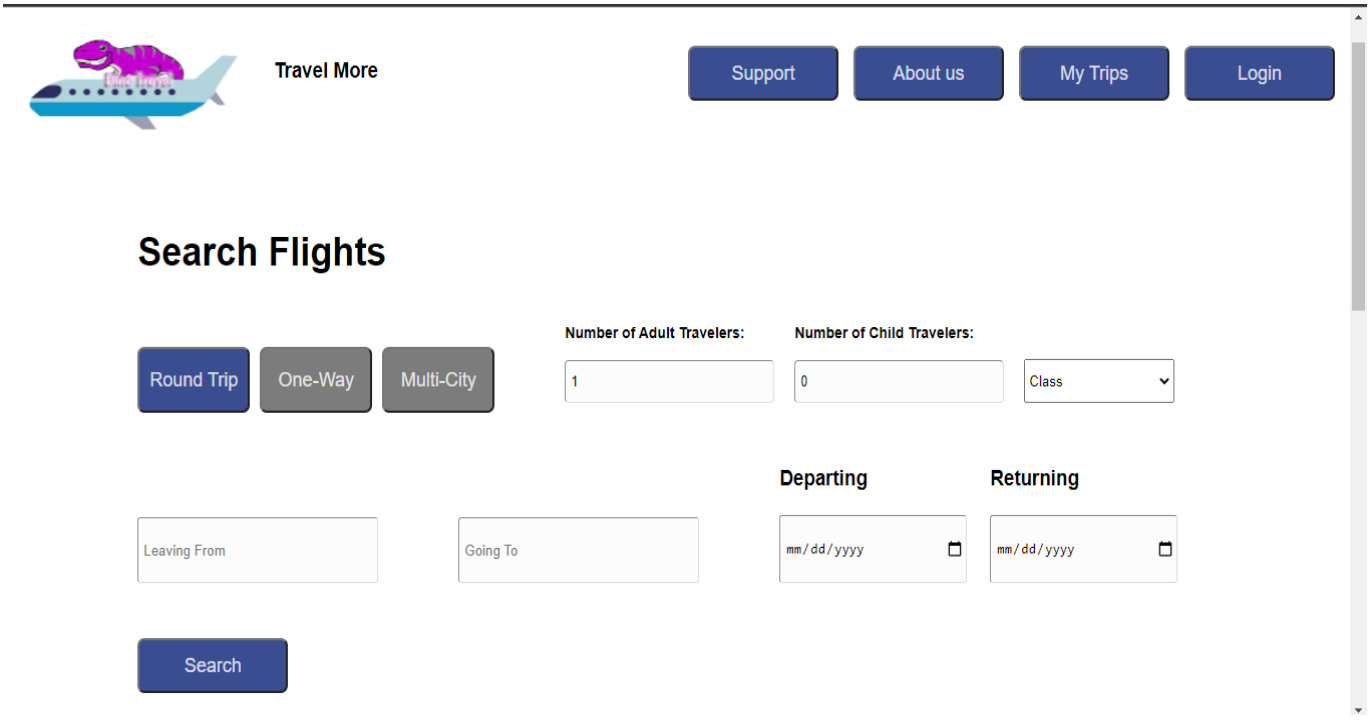
Concept Demo Video Link | **RDP Video Link** | **Preview Demo Video Link**

Final Demo Video Link

3.0 Final Product

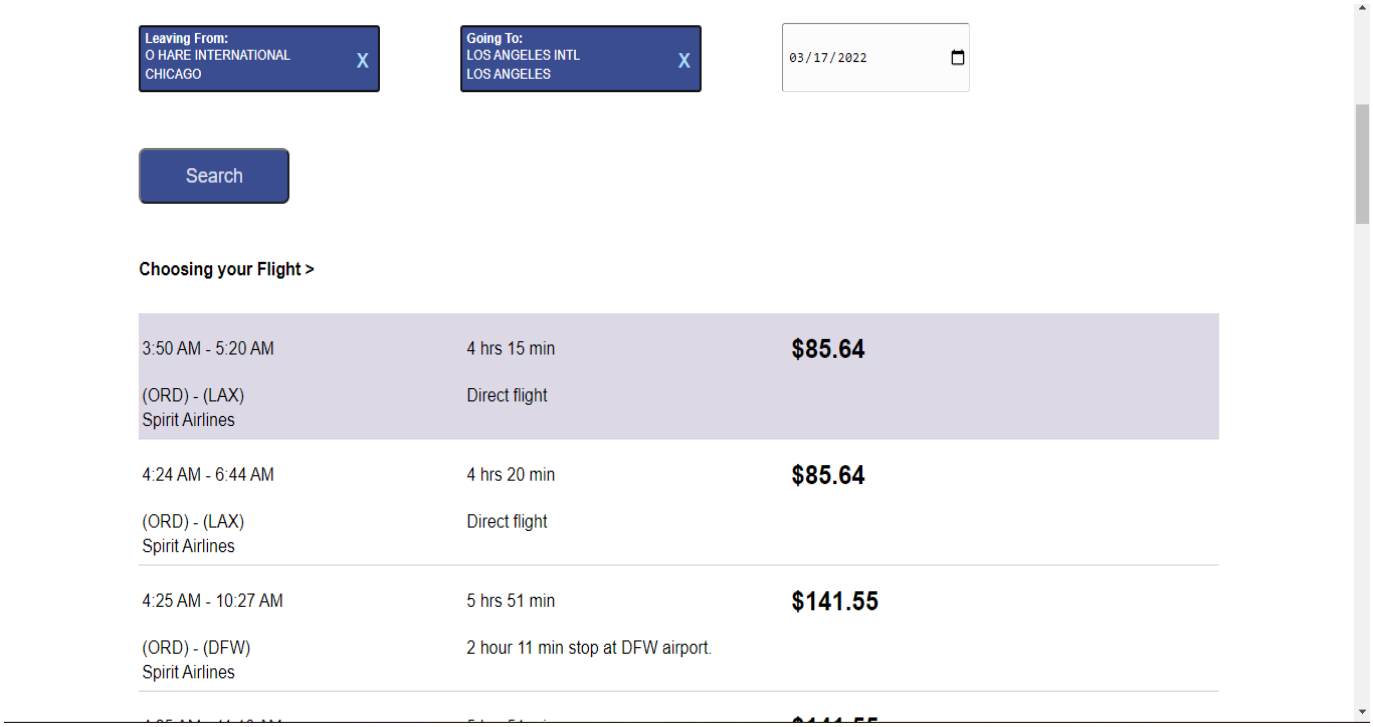
3.1 Final Website Design

3.1.1 Front Page



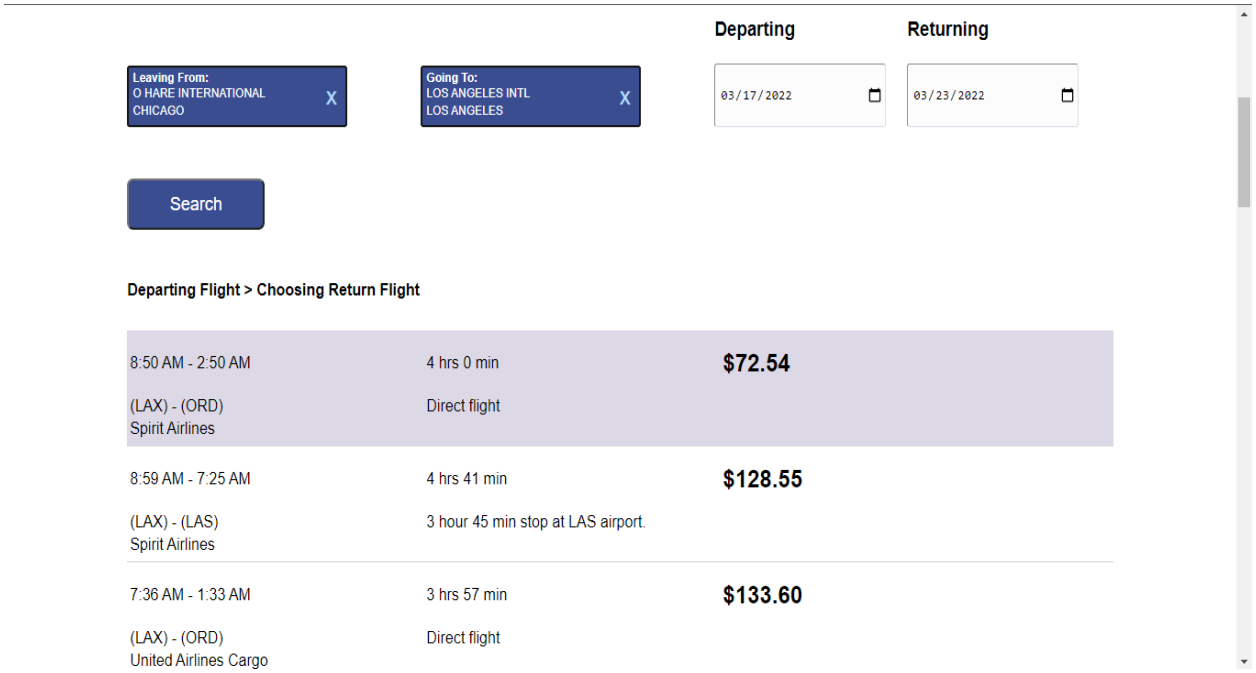
3.1.1 DinoTravel Homepage

3.1.2 One Way Flight Search Results List



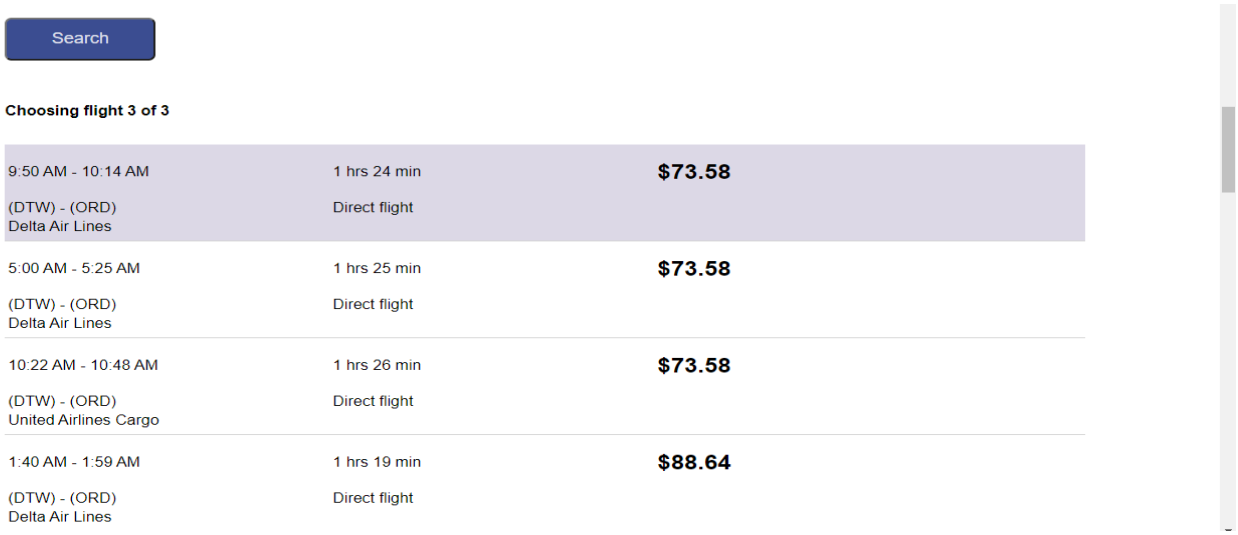
3.1.2 One-Way Search Results

3.1.3 Round Trip Flight Search Results



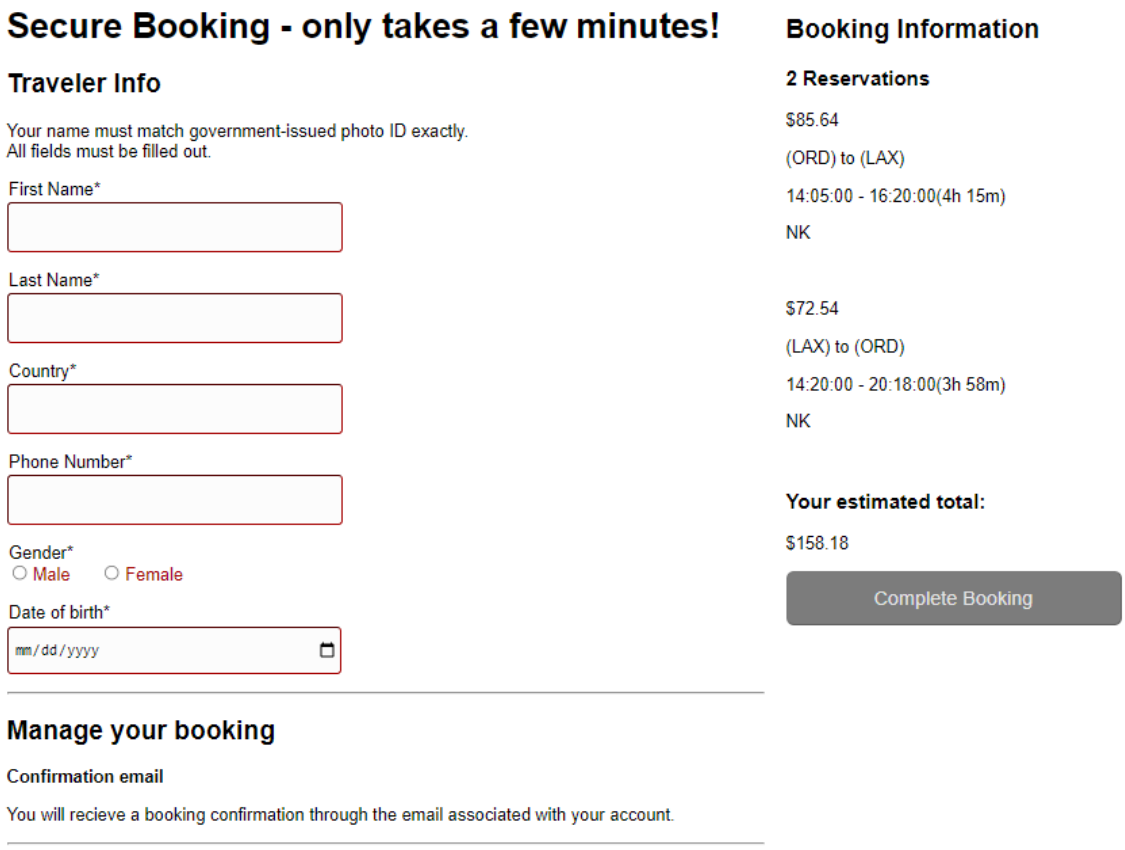
3.1.3 Roundtrip Search Results (Return)

3.1.4 Multi-City Search Results



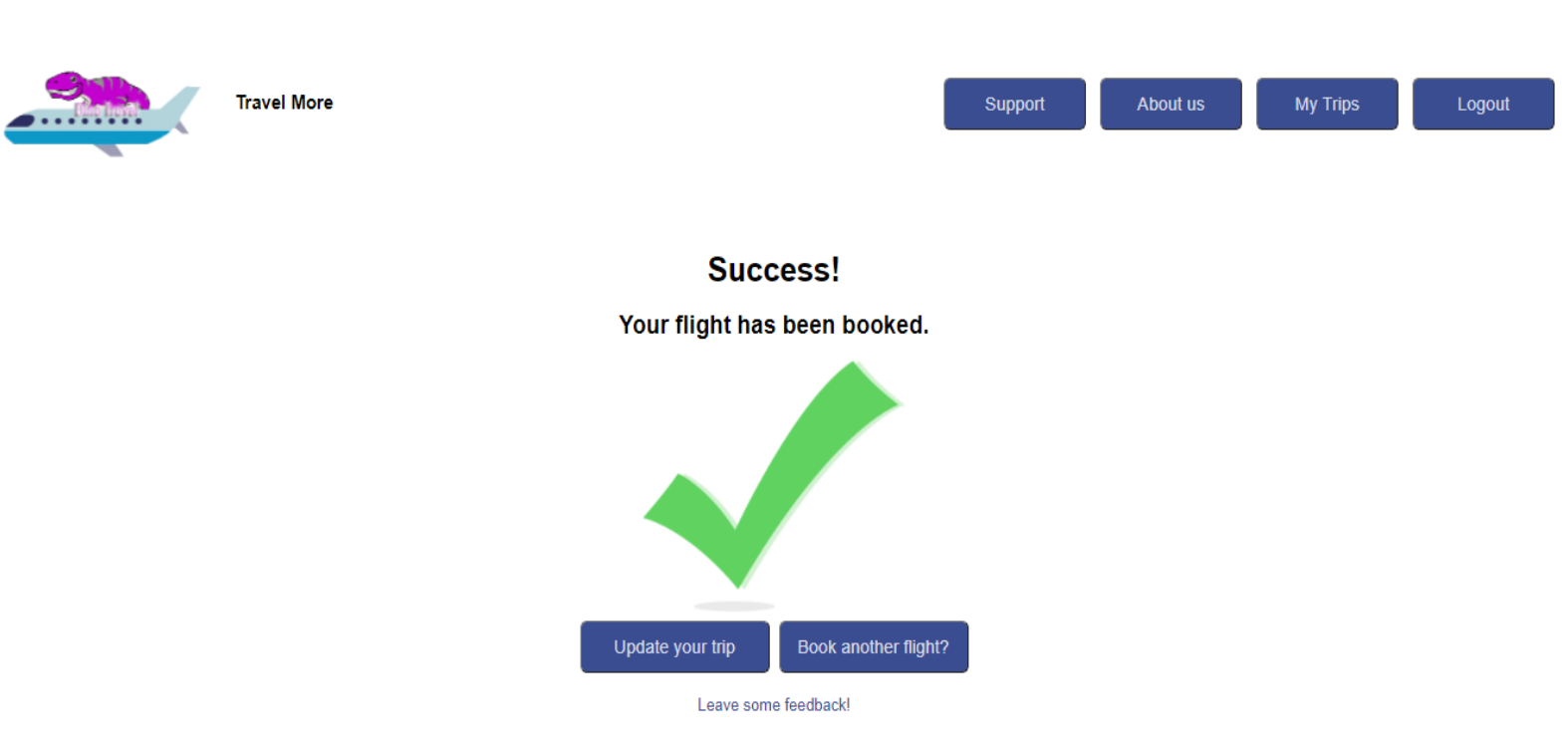
3.1.4 Multi-City Search Input

3.1.5 Customer Information Input Page



3.1.5 Customer Information Form & Flight Review (Round-trip)

3.1.6 Booking Complete Page



3.1.6 Confirmation Page

3.1.7 Support Screen

3.1.7 Customer Support, Complain, and Policy Page

3.2 Front End

Development of the website was done in TypeScript with the React frontend framework. Webpack was used for bundling the files and compiled code for the website.

MomentJS is used for handling dates and calculating flight and stop times from departure and landing times. This allows us to give the user more clear information on how long the flight will be and how long each stop (if any) the flight may make.

10:00 AM - 8:12 AM	11 hrs 2 min	**$311.20**
(ORD) - (SEA) Alaska Airlines	4 hour 10 min stop at SEA airport.	
8:00 AM - 8:12 AM	11 hrs 2 min	**$311.20**
(ORD) - (SEA) Alaska Airlines	6 hour 10 min stop at SEA airport.	

3.2 Example of flight options displaying pertinent flight information

TypeScript classes are used to split each page into an individual React component that can be rendered and swapped out. These pages are then split into subcomponents where useful, such as the image banner with rotating images is it's own component which can be reused.

By default, React is designed with a single page application setup in mind, however for our use case we wanted to have separate pages, and loosely coupled page components so that we could work individually on each page. To support this we used React Router to render a different page component depending on the URL the user is on (with the HomePage being the default url).

```
render() {
    return (
        <Router>
            <Routes>
                <Route path="/trips" element={<TripsPage id_Token={this.state.IDToken} isLoggedIn={this.state.isLoggedIn}/>} />
                <Route path="/support" element={<SupportPage isLoggedIn={this.state.isLoggedIn} />} />
                <Route path="/about" element={<AboutPage isLoggedIn={this.state.isLoggedIn} />} />
                <Route path="/checkout" element={<CheckoutPage id_Token={this.state.IDToken} isLoggedIn={this.state.isLoggedIn} reservedFlightOffers={this.state.reservedFlight
                <Route path="/success" element={<SuccessPage isLoggedIn={this.state.isLoggedIn} />} />
                <Route path="/" element={<HomePage id_Token={this.state.IDToken} isLoggedIn={this.state.isLoggedIn} onReservedFlightsFinalized={this.updateReservedFlights} />}
                <Route path="/login" element={<LoginPage updateIDToken={this.updateIdToken} isLoggedIn={this.state.isLoggedIn}/>} />
            </Routes>
        </Router>
    )
}
```

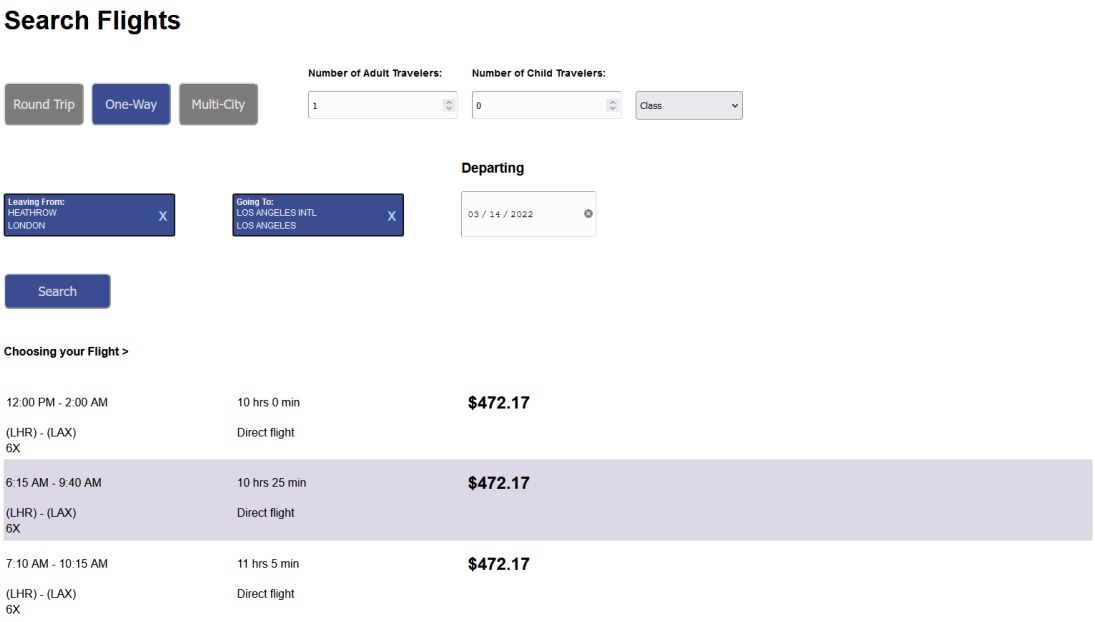
3.2 Demonstration of the page components being bound to URLs using React Router

3.2.1 Front End Pages

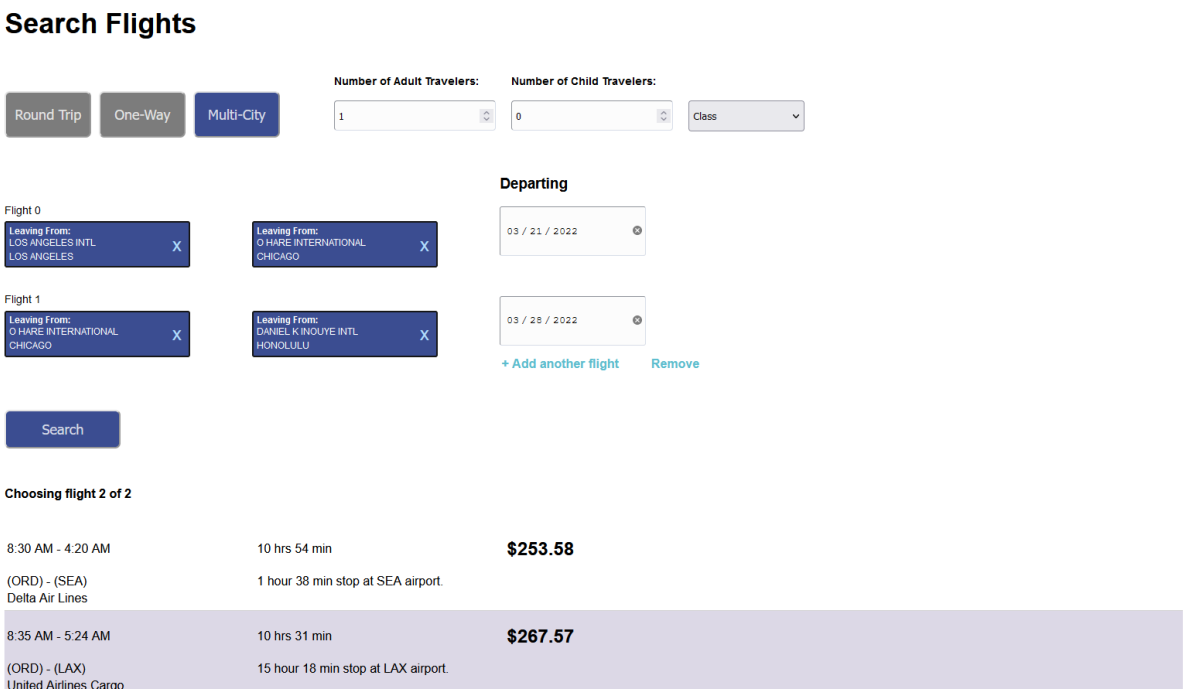
3.2.1a Home Page

The Home Page allows the user to select the flights they wish to book. They can do this through 3 presented modes, “Round Trip”, “One Way”, and “Multi City”. Round Trip and One Way both use the same UI flow to enter a departure airport and destination airport, as well as the departure date, and return date if Round Trip is selected. Multi City allows adding numerous flights that can be added and removed for whatever amount of flights they want to book.

A search button is available to return a list of flights from their selections and select their favorite option for each flight they plan to book. Once this is complete, they can submit which will bring them to the Checkout Page.



3.2.1a Example of Homepage flight selection in One Way mode



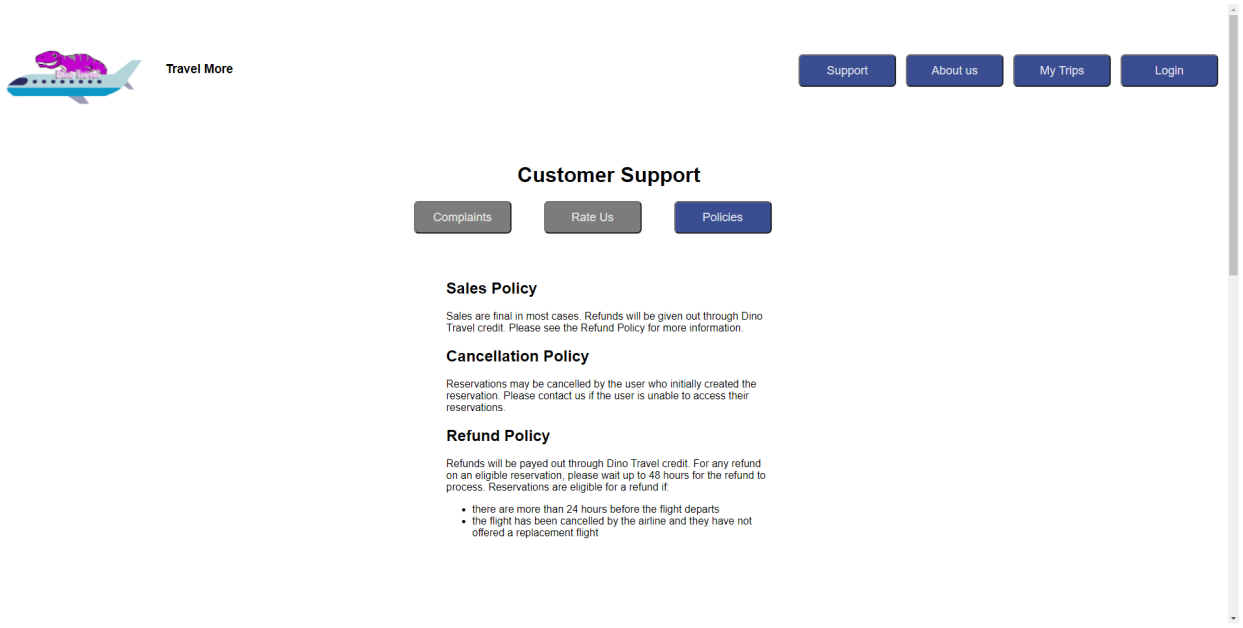
3.2.1a Example of Homepage flight selection in Multi City mode

The homepage airport selector components reduce the amount of API calls and searches being made by making use of debouncing wait until the user has stopped typing for a period of time before making the API call. This avoids performing a search every time a character is entered and can make a singular search if the user types in one go.

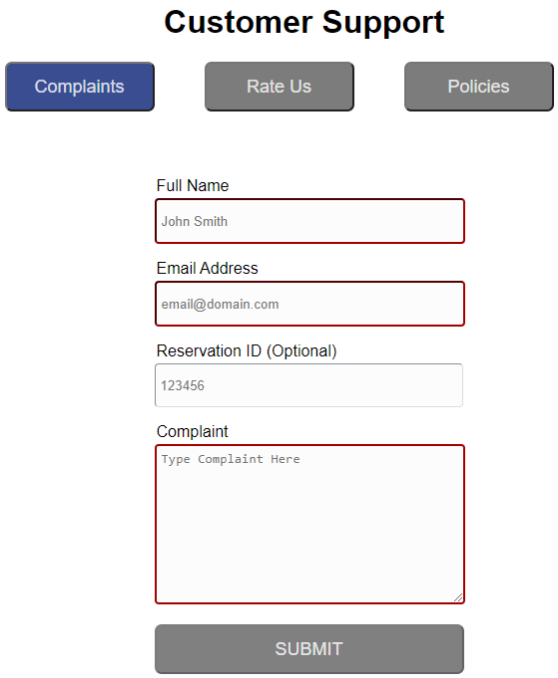
3.2.1a Demonstration of searching and populating results once the user has finished typing

3.2.1b Support Page

The support page provides users with 3 buttons to click on that change the page's content. By default, the policies will show when the support page is loaded. If the user clicks on "Complaints" or "Rate us," a form will appear for the user to fill out.



3.2.1b User view after clicking the Support button



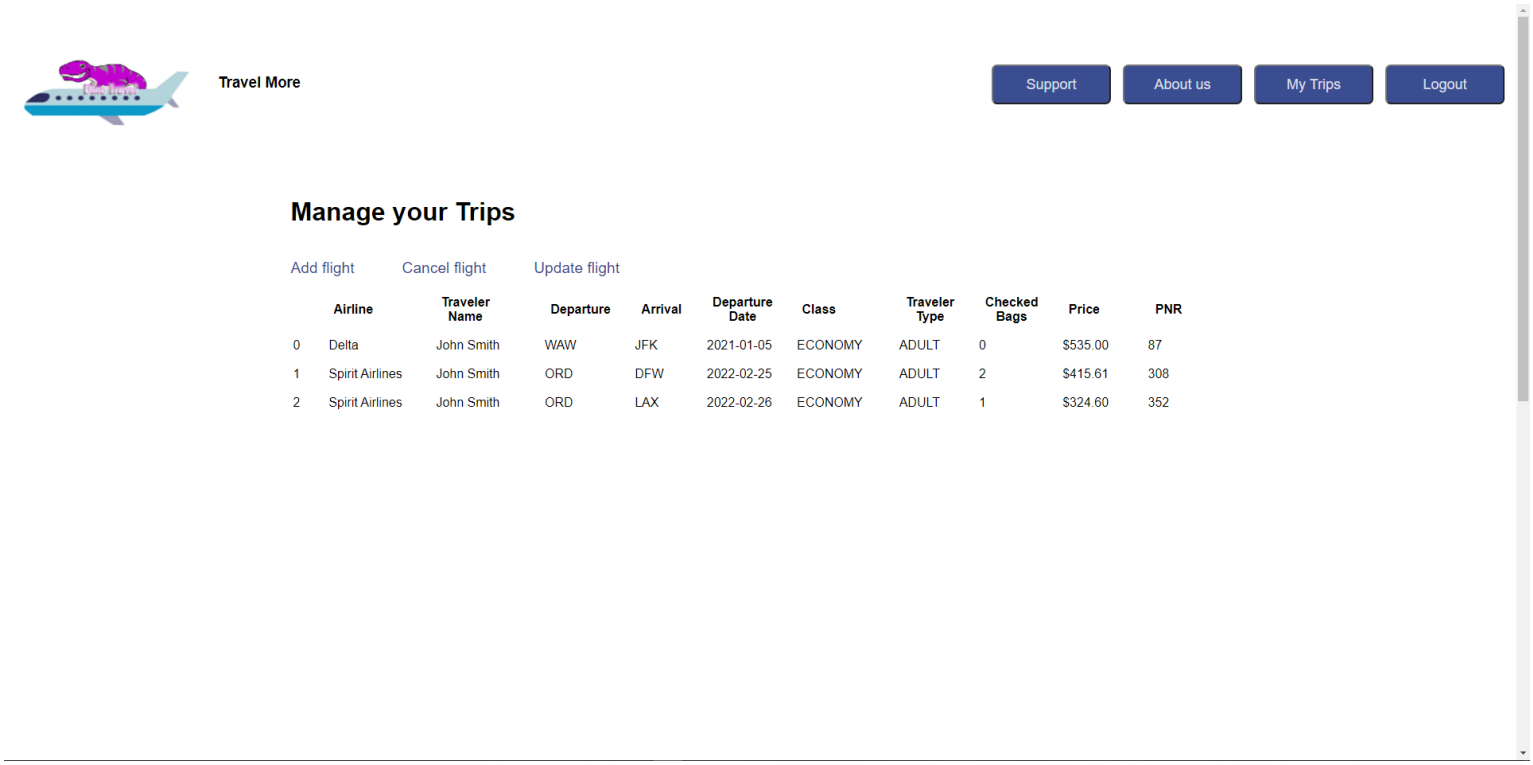
3.2.1b View when user has selected the "Complaints" form

3.2.1b View when user has selected the "Rate Us" form

For each of the forms, the required fields are highlighted with a red border. When the fields get filled out, the red border disappears. Then, when all of the required fields are filled out the submit button is enabled. All of the information that was inputted into the form gets sent to our MySQL database into the respective table.

3.2.1c Trips Page

The trips page shows all flights that a logged in user has made. If a user is not logged in when attempting to access the page, they will be notified that they have to sign in first. On the page, the user has access to 3 actions they can click on. The first is “Add flight.” This will redirect users to the homepage where they can select additional flights.

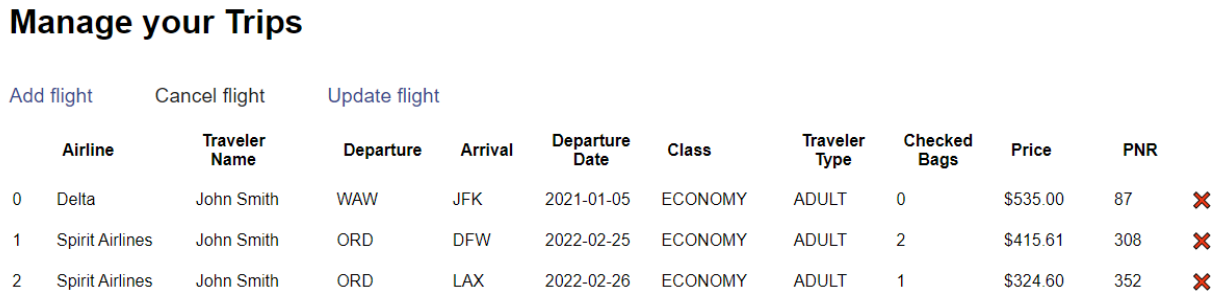


	Airline	Traveler Name	Departure	Arrival	Departure Date	Class	Traveler Type	Checked Bags	Price	PNR
0	Delta	John Smith	WAW	JFK	2021-01-05	ECONOMY	ADULT	0	$535.00	87
1	Spirit Airlines	John Smith	ORD	DFW	2022-02-25	ECONOMY	ADULT	2	$415.61	308
2	Spirit Airlines	John Smith	ORD	LAX	2022-02-26	ECONOMY	ADULT	1	$324.60	352

3.2.1c Example User view from the “My Trips” page

Canceling a Flight

When the “Cancel flight” text is clicked, a red “x” appears next to the reservations allowing users to delete a reservation. When the “x” is clicked, a prompt asks the user if they’re sure they want to delete a flight. If they are, the reservation is deleted from the reservations database table and the flights page is refreshed.



	Airline	Traveler Name	Departure	Arrival	Departure Date	Class	Traveler Type	Checked Bags	Price	PNR	
0	Delta	John Smith	WAW	JFK	2021-01-05	ECONOMY	ADULT	0	$535.00	87	✖
1	Spirit Airlines	John Smith	ORD	DFW	2022-02-25	ECONOMY	ADULT	2	$415.61	308	✖
2	Spirit Airlines	John Smith	ORD	LAX	2022-02-26	ECONOMY	ADULT	1	$324.60	352	✖

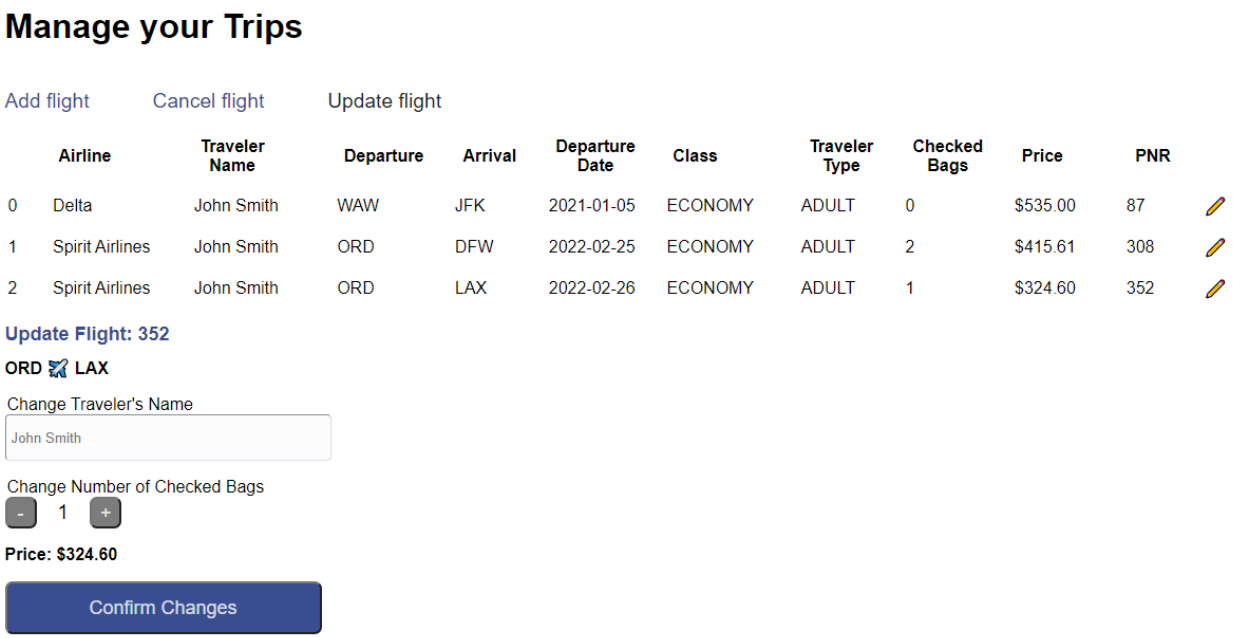
3.2.1c Example User view after clicking “Cancel Flight”

```
const handleDelete = async (resId: number) => {
    if (window.confirm("Are you sure you want you cancel your flight?

        if (idToken !== null) {
            deleteReservation(resId, idToken);

            await new Promise(r => setTimeout(r, 1500));
            window.location.reload()
        }
    }
}
```

3.2.1c Front End code handling deletion of flight

Updating a Flight

When the “Update flight” button is clicked, a pencil icon will appear next to all of the user’s reservations. Clicking on one of the icons will allow the user to update the traveler’s name or number of checked bags for that reservation. Changing the amount of checked bags will also increase/decrease the total price by $35 so it will get updated as well. Once the user makes their changes and clicks “Confirm Changes,” the reservation will be updated in our reservations database table and the flights page will refresh to show the new changes.



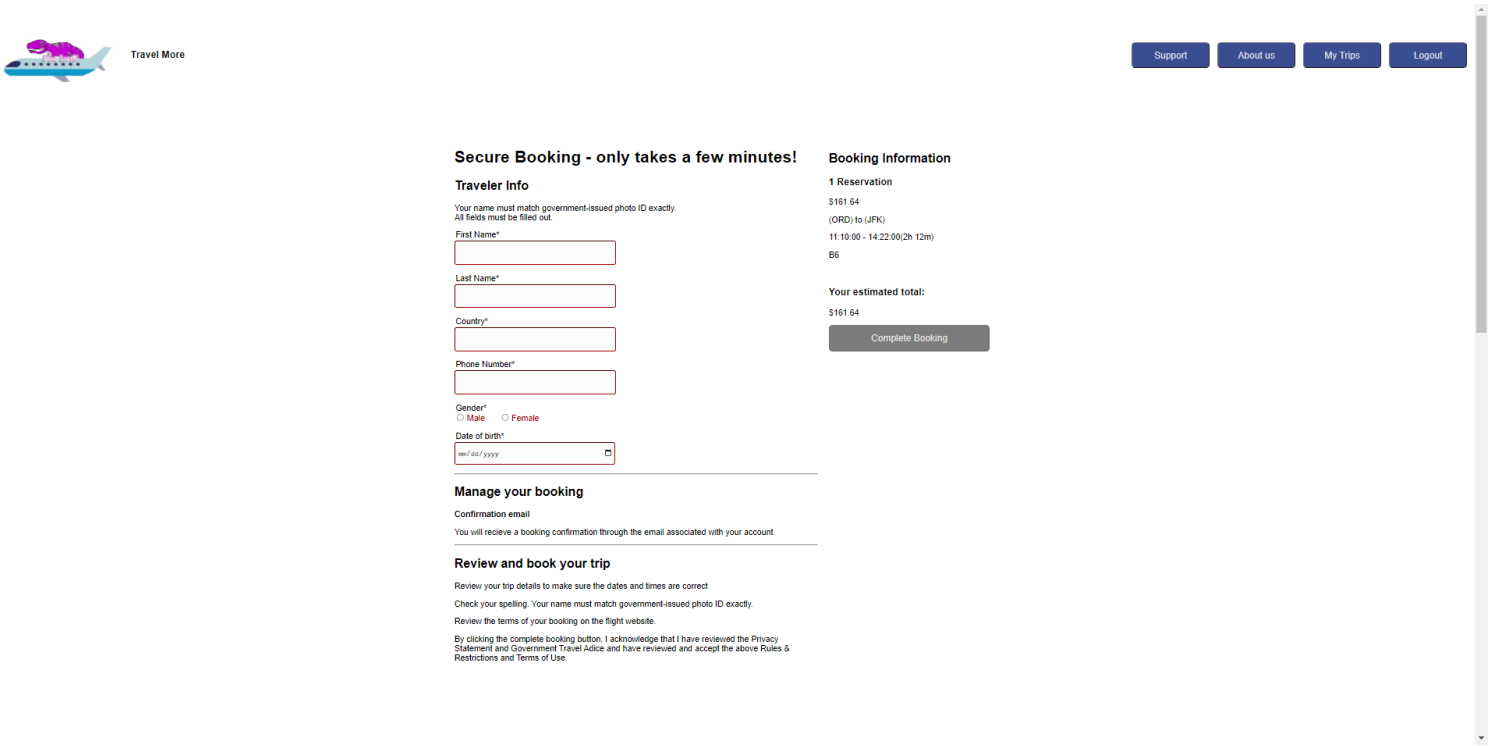
3.2.1c Example User view after clicking “Update Flight”

```
useEffect(() => {
    setBags(updateItem.numCheckedBags);
    setName(updateItem.travelerName);
    setNewPrice(updateItem.price);
}, [updateItem])
```

3.2.1c Front End code for Updating a user’s flight

3.2.1d Checkout Page

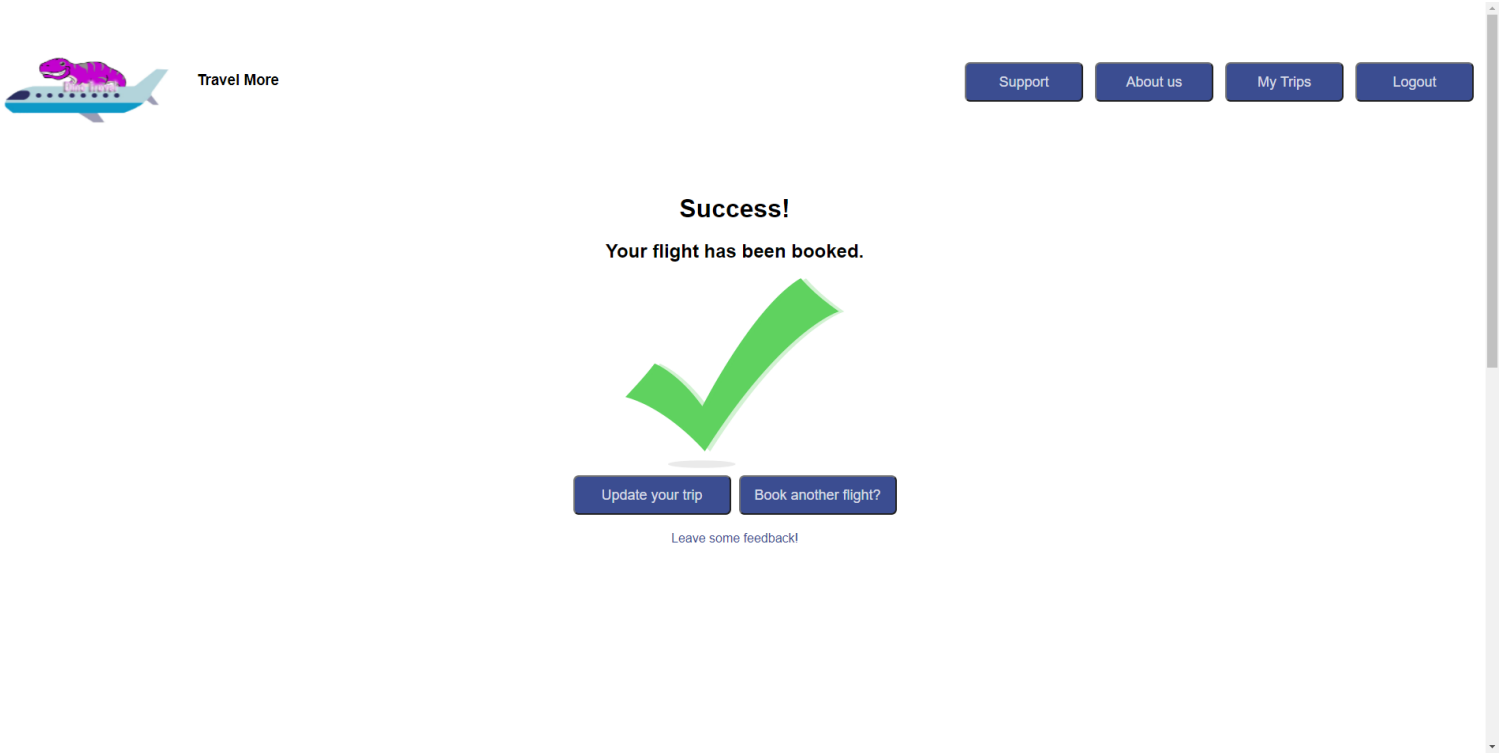
The checkout page appears after a user has selected all of their flights. Similar to the support page forms, the checkout page form highlights all of the required fields with a red border. As long as the user has selected at least 1 flight and has filled out of the fields, the “Complete Booking” button will become enabled. Once the user completes their booking, the user’s information from the form will be added to the users table in the database. Additionally, all of their selected flights will be added to the reservations and flights tables in the database. Then, the user will be redirected to the success page.



3.2.1d Example of Checkout Page before user input

Success Page

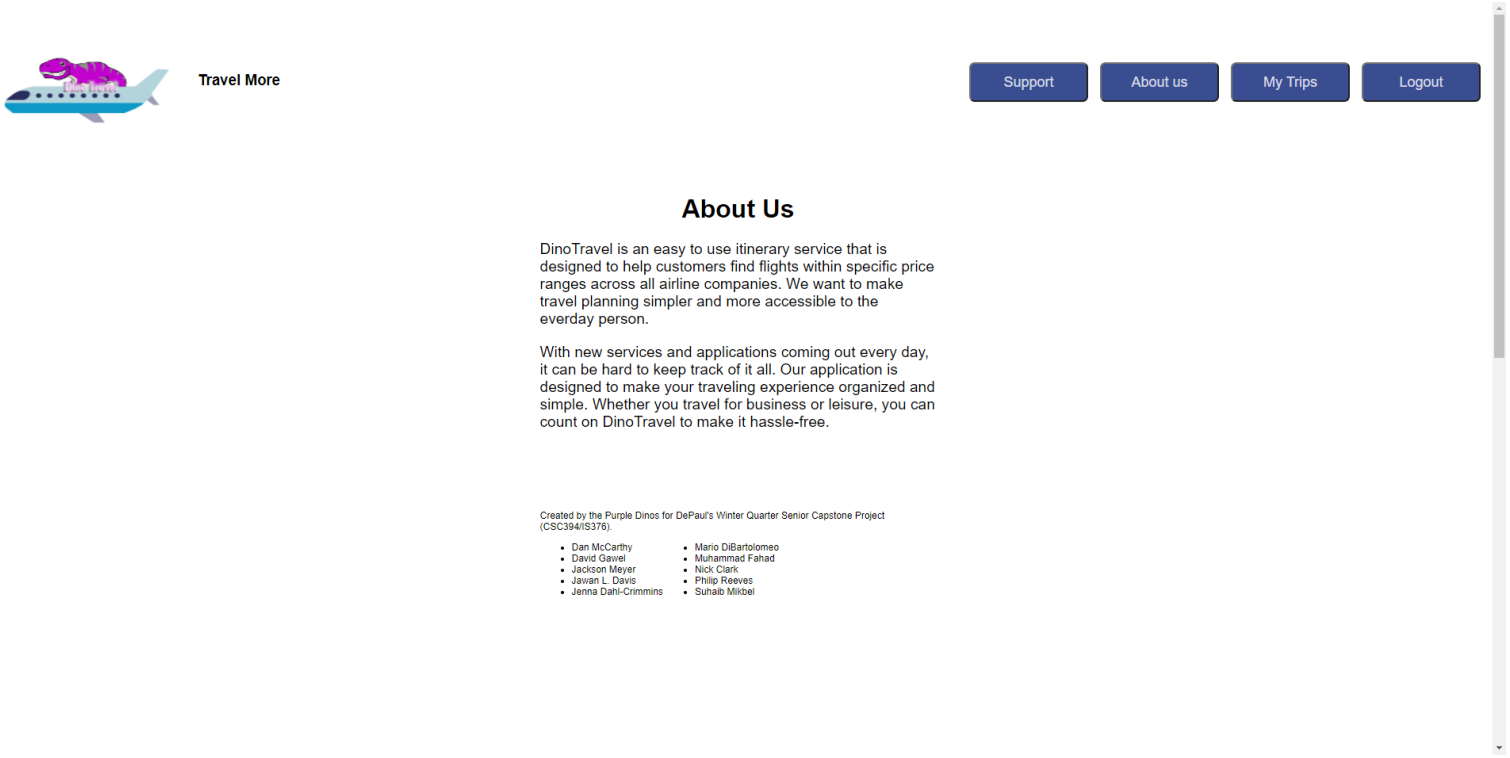
From the success page, the user has 3 actions. The “Update your trip” button will direct the user to the trips page where they can manage their reservations. The “Book another flight?” button will direct the user back to the home page where they can select more flights. Lastly, the “Leave some feedback!” link will direct the user to the support page where they can fill out the complaint or review form.



3.2.1d Example of the Success page after user has successfully booked a flight

3.2.1e About Us Page

The About Us page is a static page without functionality. It highlights DinoTravel's mission and lists out all of DinoTravel's team members.

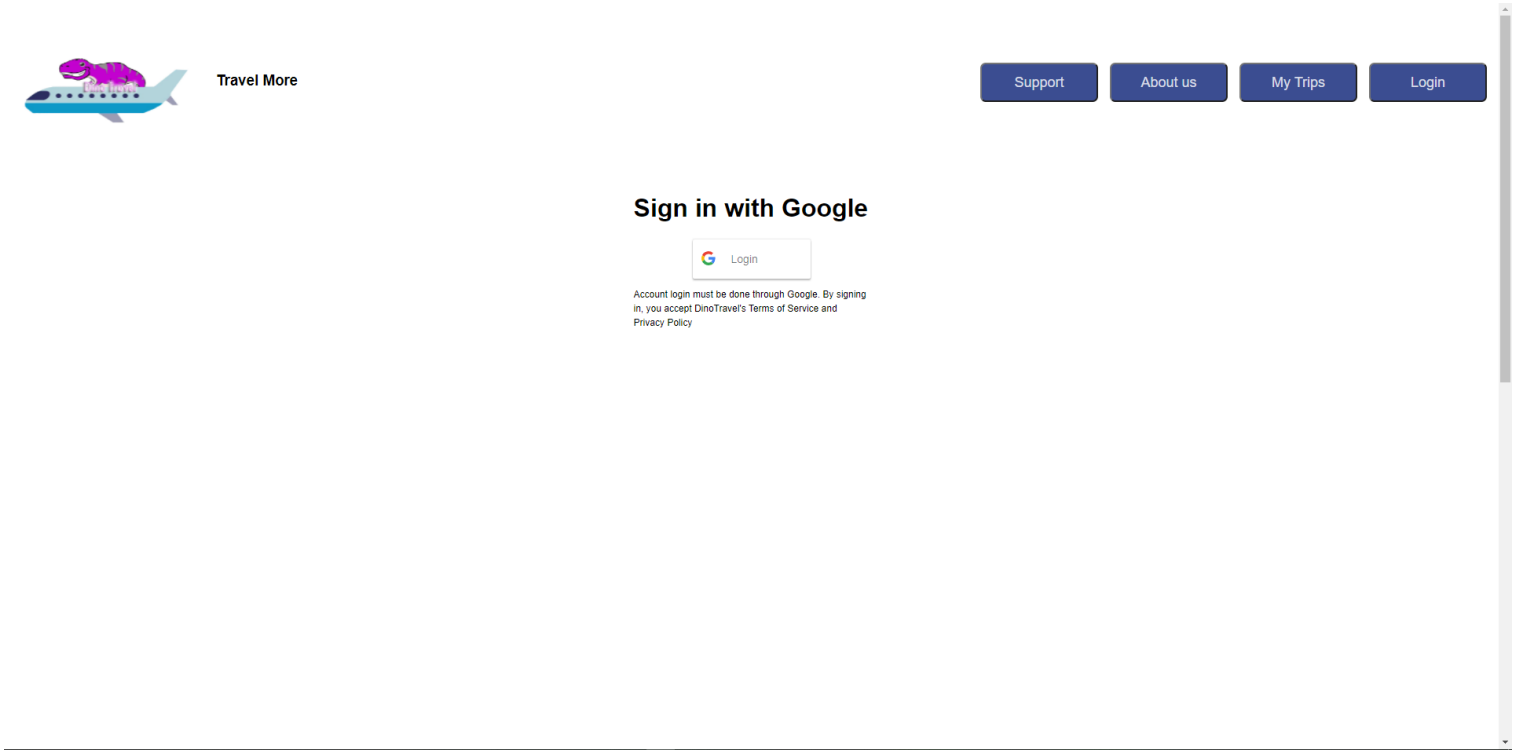


3.2.1e Image of the "About Us" page

3.2.1f Login Page

The login page features a Google login button that will prompt the user to sign in with one of their Gmail accounts. When a user logs in, the button and header changes from saying “Login” to “Logout” to reflect the user’s updated status. Once a user logs in, the webpage makes use of some of the user’s information:

- ☐ Logged in status - redirect a user if they’re not logged in
- ☐ ID Token - Authenticate requests to the backend to create or request reservations
- ☐ Email - Send a booking confirmation email



3.2.1f Initial view when a logged-out user clicks the Login button

The user's information is passed between pages through React props. The login page sends the information back up to the page router and the page router can share that information with all of the webpages. Some of the pages, such as the about page, don't need the user's ID token to show them the page's content. However, the trips page does require the user's ID token to show them all of the reservations that they have made.

```
<Router>
    <Routes>
        <Route path="/trips" element={<TripsPage id_Token={this.state.IDToken} isLoggedIn={this.state.isLoggedIn}/>} />
        <Route path="/support" element={<SupportPage isLoggedIn={this.state.isLoggedIn} />} />
        <Route path="/about" element={<AboutPage isLoggedIn={this.state.isLoggedIn} />} />
        <Route path="/checkout" element={<CheckoutPage id_Token={this.state.IDToken} isLoggedIn={this.state.isLoggedIn} reser
        <Route path="/success" element={<SuccessPage isLoggedIn={this.state.isLoggedIn} />} />
        <Route path="/" element={<HomePage id_Token={this.state.IDToken} isLoggedIn={this.state.isLoggedIn} onReservedFlights
        <Route path="/login" element={<LoginPage updateIDToken={this.updateIdToken} isLoggedIn={this.state.isLoggedIn}/>} />
    </Routes>
</Router>
```

3.2.1f Example code of React props used to pass data between pages

3.3 Testing/ HCI

Testing Objectives: The objective of the test was to identify areas within our website that could make the time shorter for the user to book their flight. We wanted to see what actions the user will take when given a set amount of time to and relay their experience as to how easy or difficult the process was, as well as what functionality we can improve on to create a simple and intuitive experience.

Format of study: The users were given a scripted test and were able to perform remotely. They were asked to book a flight of any type using our website with steps directing them on tasks that were to be completed accordingly. The users were also asked to rate how difficult each part of the booking procedure was on a scale from one through five (one = easy, five = hard). After the booking was complete, the users were asked to use DinoTravels Support page to relay their comments and concerns and logout.

User Info: The study included six participants over the age of twenty-one in order to relay the correct demographics for users that have travel experience and are less likely to commit errors with technology. Participants included four males and two females, and all were working full-time jobs with prior travel experience and online booking knowledge.

1. **Login user functionality (Rate 1-5)**

    1.1. Go to Dinotravel homepage

    1.2. Flights can’t be booked until logged in

    1.3. Click on login button

    1.4. Redirect to homepage with top right Login changed to Logout after user logs in

    1.5. User stays logged in

2. **Homepage user functionality (Rate 1-5)**

    2.1 User clicks on Round Trip and button turns blue to confirm selection

    2.2 User clicks on One-Way and button turns blue to confirm selection

    2.3 User clicks on Multi-City and button turns blue to confirm selection

    2.4 User hovers over Adult Travelers input and clicks arrow to increase increments

    2.5 User hovers over Child Traveler input and clicks arrow to increase increments

2.6 User clicks on Class drop down and selects a fare

2.7 User clicks on Leaving From input and selects current location

2.8 User clicks on Going To input and selects destination

2.9. User clicks Search button and search indicator appears

2.9.1 If no flights are available then notice message will be displayed

2.9.2 User sees login button if not logged in to proceed with Search

2.9.3 If logged in and flight selected, click on Next flight or submit

3. **Checkout review page functionality (Rate 1-5)**

3.2 All inputs must be filled out before booking is complete

3.2 If booking isn't able to be completed, user gets an alert

3.3 Price is reviewed along with flight information

3.4 Clicking Complete booking displays Alert confirmation/informs user of trips page

3.4.1 Complete booking button changes color when clicked

3.5 Clicking Complete booking takes you to submission page

4. **Trips page user functionality (Rate 1-5)**

5.1 User clicks on trips from homepage and is redirected

5.2 If user is not logged in, page will be inaccessible & direct them to login

5.3 User able to access trips page if logged in

5.5 Trips table receives flight data and user data from Users booked flights

5.6 User clicks on Add flight button and is redirected to flight search

5.7 User clicks on Cancel flight and is able to delete reservation with X icon

5.8 User clicks on Update flight and is able to update their reservation with Pencil icon

5.8.1 User is able to update Traveler name

5.8.2 User is able to update baggage and price accordingly

5.8.3 Clicking Confirm Changes updates table and flight reservation

5. **Support page user functionality (Rate 1-5)**

    4.1 User clicks on support from homepage and redirected

    4.2 User clicks on Complaints to reveal complaint form - please leave any complaints

    4.3 User clicks on Rate us to reveal feedback form - please leave us some feedback

    4.4 User clicks on Policies to reveal DinoTravel’s policies

6. **User logs out after completing test**

3.4 Back End

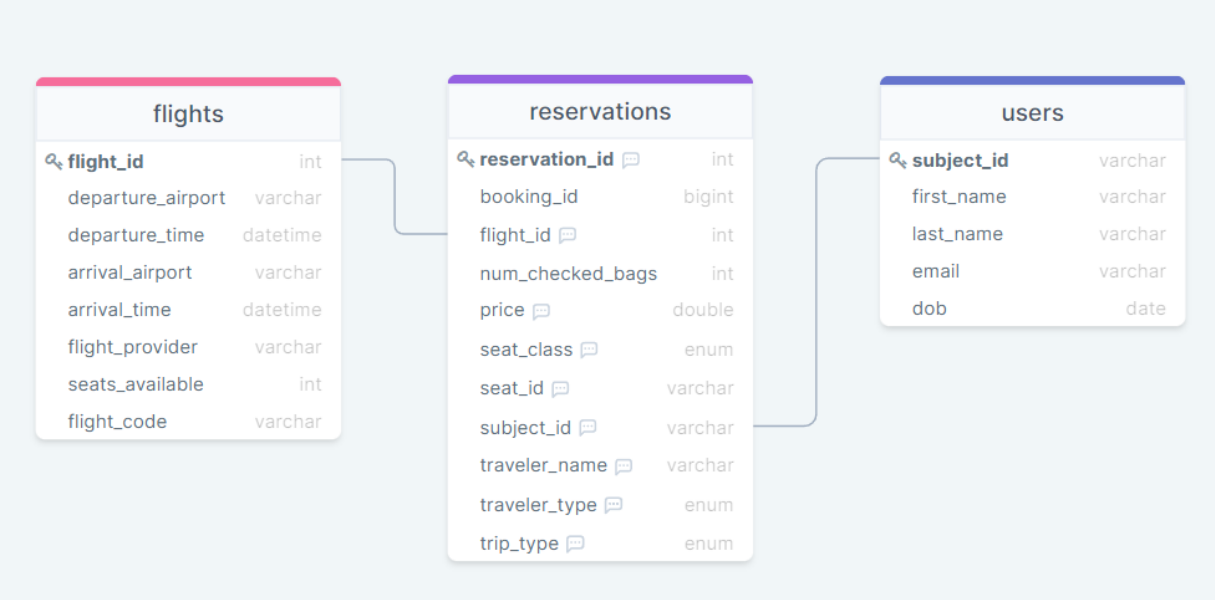
This portion of the Project Manual will cover the different aspects of the Back End, from the Database design to interfacing with the Amadeus Flight API. All code for the backend can be found at https://github.com/DPUPurpleDinos/dino-travel.

3.4.1 Software Used

This is a comprehensive list of all the software and technologies used in the development of the Back End for the Dino Travel project.

- ☐ Java --> Backend language
- ☐ Spring Boot --> REST controllers and endpoints
- ☐ AWS --> Cloud provider for our server
- ☐ JPA/Hibernate --> Mapping Java classes to a relational schema
- ☐ H2 --> Database for testing endpoints locally
- ☐ LetsEncrypt --> SSL certificate authority
- ☐ MySQL --> For live test data and production data
- ☐ JUnit --> Testing Java classes and methods
- ☐ JavaMail --> API for sending emails
- ☐ Google OAuth 2.0 --> Authentication API
- ☐ Git --> Version control system

3.4.2 Final Database Schema



3.4.2 Diagram of Final Database Schema

3.4.3 Back End Code

This section of the document will examine the key components of the Back End code. In whole, the backend code is comprised of 4 key parts:

- ☐ *User authentication through Google's OAuth 2.0 API*
- ☐ *Amadeus API controllers to get real flight data and locations*
- ☐ *Rest controllers to interact with the information stored in our database*
- ☐ *Send Email confirmation*

These separate parts work together to capture the relevant data queried for by the user on the front end and pass back that info to display it to the user, as well as store any reservations made by the user.

3.4.3.1 User Authentication

For user Authentication we use Google's OAuth 2.0 API. This allowed us to not have to deal with storing and hashing passwords, or authenticatication of users, that was all handled on Google's end.

On the front end the user logins through google and receives a response. This response has a lot of data but all we care about is the IDToken. This token has a lot of info about the user and is signed by google. We store this token in the routing prop so that other pages can easily access this data. We also store this in local storage so that when the user leaves the page and comes back they are still logged in.

Below is a screen shot of the frontend react code that handles login and logout.

```
onSignIn = (googleUser : any) => {
    //DO NOT SEND USER ID TO BACKEND
    localStorage.setItem('LoggedIn', 'true');
    localStorage.setItem('LoginAttempted', 'true');
    localStorage.setItem('Profile', JSON.stringify(googleUser.profileObj));
    localStorage.setItem('Token', JSON.stringify(googleUser.tokenObj));
    this.props.updateIDToken(googleUser.tokenObj.id_token);
    this.setState( state: {
        LoginAttempted: true,
        LoggedIn: true,
        Profile: googleUser.profileObj,
        Token: googleUser.tokenObj,
    })
}

onLogOut = () => {
    localStorage.setItem('LoggedIn', 'false');
    localStorage.setItem('LoginAttempted', 'false');
    localStorage.removeItem( key: 'Profile');
    localStorage.removeItem( key: 'Token');
    this.props.updateIDToken( NewID: null);
    this.setState( state: {
        LoginAttempted: false,
        LoggedIn: false,
    });
}
```

3.4.3.1 Front End code handling Login/Logout

On login the react router prop values are set and the localstorage is set. On logout the opposite is done the react router prop values are set to null and the localstorage values are removed.

There are some security risks with this though. Using local storage is not ideal because it can be called through javascript. This makes it possible for a cross site scripting attack where an attacker embeds malicious javascript code into our legitimate website. This makes it possible for them to steal these tokens and impersonate a user. This is not the biggest deal though because by default react sanitizes user input. On the backend as well we use jpa to parameterizes our queries to prevent sql injection. These tokens also expire after one hour and we do not refresh them.

We also use id tokens, not authentication tokens. This is not ideal either since IDTokens are supposed to be used when a user is authenticated, not to authenticate a user. It is better to use an authentication token in this instance. This allows for an attacker to steal the IDToken and get user data. There is also the issue that we can not see what scopes a user has allowed but we do not use scope. This is not the biggest security risk as long as an attacker does not get your token, but it is something to watch out for any future projects.

On the back end once we get these IDTokens we validate them using the google token verifier.

```
/***
 * Class for verifying if a given token string is valid
 * @param tokenString the token string to verify
 * @return Token Verifier Response. Returned if valid
 * @throws TokenInvalid Error thrown if the token is invalid
 */
public static TokenVerifierResponse verifyToken(String tokenString) throws TokenInvalid{
  GoogleIdToken idToken;
  //Try to verify the token return a null or token if no exceptions occur
  try {
    //automatically verifies the jwt signature, audience claim, the issuer claim, and the expiration
    //if any fail it will give a null. Most likely the token expired
    idToken = verifier.verify(tokenString);
    //have to catch this general exceptions for the verifier
  } catch (GeneralSecurityException | IOException e) {
    throw new TokenInvalid("A general google exception occurred reason: " + e.getMessage());
    //thrown if the token is too short or too long
  } catch (IllegalArgumentException e){
    throw new TokenInvalid("Invalid Input Length reason: " + e.getMessage());
    //catch any other exceptions
  } catch (Exception e){
    throw new TokenInvalid("An exception occurred reason: " + e.getMessage());
  }
  //Check if the token is not null if, if it is not everything is good
  //else it probably expired
  if (idToken != null) {
    return new TokenVerifierResponse(idToken.getPayload(), validity: true);
  }else{
    throw new TokenInvalid("Your token most likely expired.");
  }
}
```

3.4.3.1 Back end code showing IDToken validation

If any errors occur, such as an expired token being used, that info is sent to the client and is properly dealt with. This validation function is used every time any token has to be validated. The function call is very simple and also returns a response containing the subjectID and also the payload which has info like user email, name, profile picture, etc. If an error occurs, an exception is thrown and the process halts.

```
TokenVerifierResponse response = TokenVerifier.verifyToken(auth);
```

3.4.3.1 Invocation of the `verifyToken(String)` method

3.4.3.2 Database Controllers

All of the controllers are created using Java and the Spring Boot framework. There are 7 controllers in total:

- ☐ **Flights**
- ☐ **Users**
- ☐ **Reservations**
- ☐ **Complaints**
- ☐ **Reviews**
- ☐ **Flight Offers**
- ☐ **Flight Locations**

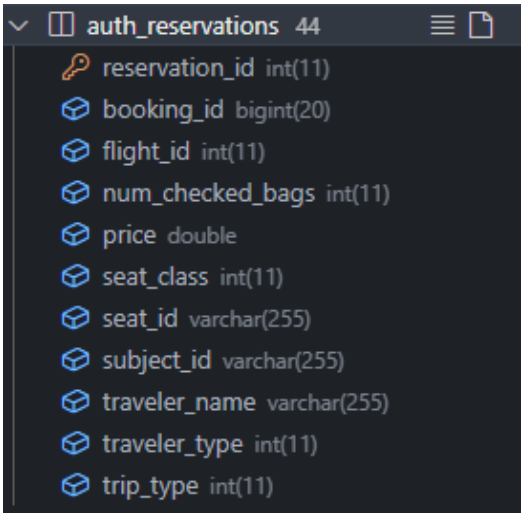
Each controller is created using a few parts. The schema, which would match Java classes and fields to a table's schema in the database. The Model Assembler, which formats the JSON response. A repository to create the connection between the Java classes to the MySQL fields. And then, any relevant Exceptions.

Each controller supports methods to get all items, get a single item, update an item, add a new item, and delete an item. Each controller uses at least 2 of these methods. For example, the complaint and review controller only support getting all items, and adding an item while the reservation controller uses all 5.

3.4.3.2a Reservation Controller

To get a better understanding of how the controllers work, take a closer look at the reservations controller.

```java
@Id
@GeneratedValue
@Column(name = "reservation_id", nullable = false, updatable = false)
public int reservation_id;

//the id of the booking the reservation is apart of
@Column(name = "booking_id", nullable = false, updatable = false)
public long booking_id;

// Foreign key
@Column(name = "subject_id", nullable = false, updatable = false)
public String subject_id;

@Column(name = "price", nullable = false)
public double price;

//enum ONEWAY, ROUNDTRIP, MULTICITY
@Enumerated(EnumType.ORDINAL)
@Column(name = "trip_type", nullable = false)
public tripType tripType;

// Foreign key
@Column(name = "flight_id", nullable = false)
public int flight_id;

//enum ADULT (12+), CHILD (2-12), INFANT (2-0)
@Enumerated(EnumType.ORDINAL)
@Column(name = "traveler_type", nullable = false)
public travelerType traveler_type;

//name of the traveler for the ticket
@Column(name = "traveler_name", nullable = false)
public String traveler_name;

// ID is saved as a String to use the seat's natural key (ex: "27A")
@Column(name = "seat_id", nullable = false)
public String seat_id;

//class of the seat
@Enumerated(EnumType.ORDINAL)
@Column(name = "seat_class", nullable = false)
public travelClass seat_class;

@Column(name = "num_checked_bags", nullable = false)
public int num_checked_bags;
```



3.4.3.2a Schema code denoted in Java (Left), MySQL Schema (Right)

The first required part is the schema. On the left, the java class denotes fields using the Column annotation and matches with a field name in the database. The primary key is marked using the ID annotation and it is automatically incremented with the generated value annotation

Then the model assembler creates the JSON response for all reservations. Below is an example of the response for 2 reservations.


```
[
    {
        "reservation_id": 87,
        "booking_id": 1645061873409,
        "subject_id": "                    ",
        "price": 535.0,
        "tripType": "ROUNDTRIP",
        "flight_id": 64,
        "traveler_type": "ADULT",
        "traveler_name": "Nick Clark",
        "seat_id": "27A",
        "seat_class": "ECONOMY",
        "num_checked_bags": 0
    },
    {
        "reservation_id": 138,
        "booking_id": 1645082259627,
        "subject_id": "                    ",
        "price": 231.64,
        "tripType": "ONEWAY",
        "flight_id": 137,
        "traveler_type": "ADULT",
        "traveler_name": "Nick",
        "seat_id": "",
        "seat_class": "ECONOMY",
        "num_checked_bags": 2
    }
]
```


3.4.3.2a Example of JSON response from query

The repository connects the reservation classes to the reservations table in our MySQL database. We can optionally add SQL commands to this object through the Query annotation. For example, the “find subject ID” method will return all reservations created by the logged in user which was helpful in creating the trips page.

```
public interface ReservationRepository extends JpaRepository<Reservation, Integer> {

    @Query(value = "SELECT * FROM reservations WHERE user_id=?1", nativeQuery = true)
    List<Reservation> findByUserId(int user_id);

    @Query(value = "SELECT * FROM auth_reservations WHERE subject_id=?1", nativeQuery = true)
    List<Reservation> findBySubjectId(String subject_id);

    @Query(value = "SELECT DISTINCT subject_id FROM auth_reservations WHERE booking_id=?1", nativeQuery = true)
    String findBookingOwner(long bookingId);

    @Query(value = "SELECT * FROM auth_reservations WHERE booking_id=?1", nativeQuery = true)
    List<Reservation> findByBookingId(long bookingId);
}
```

3.4.3.2a Example of generated SQL Commands

Manage your Trips

Add flight Cancel flight Update flight

	Airline	Traveler Name	Departure	Arrival	Departure Date	Class	Traveler Type	Checked Bags	Price	PNR
0	Delta	Nick Clark	WAW	JFK	2021-01-05	ECONOMY	ADULT	0	$535	87
1	B6	Nick Clark	ORD	JFK	2022-02-17	ECONOMY	ADULT	1	$196.64	138

3.4.3.2a Example of Queried Data displayed on the Front End

The reservations controller has several exceptions. This exception gets thrown if the user’s token is invalid.

```
/**
 * Generate a 401 error if the user is unauthorized to perform an action
 * @param ex InvalidCredentials
 * @return String explaning what the user cant do
 */
@ResponseBody
@ExceptionHandler(InvalidCredentials.class)
@ResponseStatus(HttpStatus.UNAUTHORIZED)
String invalidCredentials(InvalidCredentials ex) {return ex.getMessage();}
```

3.4.3.2a Example of exception handling

```
1   The provided token is invalid. Reason: Token Expired
```

3.4.3.2a Example of returned exception

Here are the GET mappings for the Reservations controller. The first endpoint returns all reservations created by a user. The second returns a specific reservation denoted by the reservation ID.

```
    /**
     * get all the reservation ids associated with a user
     * @param auth auth string
     * @return list of all the reservations associated with a user
     */
    @GetMapping("/user")
    List<Reservation> getReservationsByUser(@RequestHeader("Authorization") String auth){
        TokenVerifierResponse response = TokenVerifier.verifyToken(auth);
        return reservationRepository.findBySubjectId(response.getSubject());
    }

    /**
     * get a reservation from a specified reservation id
     * @param auth auth string
     * @param reservationId the specified reservation id
     * @return return the id with the specified reservation
     */
    @GetMapping("/{id}")
    Reservation getReservationById(@RequestHeader("Authorization") String auth, @PathVariable ("id") int reservationId) {
        TokenVerifierResponse response = TokenVerifier.verifyToken(auth);
        Reservation reservation = reservationRepository.findById(reservationId).orElseThrow(() -> new
            ReservationNotFoundException(reservationId));
        if (!reservation.getSubject_id().equals(response.getSubject())){
            throw new InvalidCredentials("You are not authorized to view the requested reservation.");
        } else {
            return reservation;
        }
    }
  return go(f, seed, [])
}
```

3.4.3.2a Code for GET mappings in the Reservations controller

This is the POST mapping. Multiple reservations and flight segments can be added through this endpoint.

```java
/**
     * Given a list of users and associated flights it will make new
     * reservations and new flights is not already present also takes away
     * seat availability from booked flights
     * @param auth auth string
     * @param requestedReservations the array of requested reservations
     * @return created status if everything got made
     */
    @PostMapping
    ResponseEntity<?> createReservation(@RequestHeader("Authorization") String auth, @RequestBody ReservationRequest []
            requestedReservations) {
        //verify token
        TokenVerifierResponse response = TokenVerifier.verifyToken(auth);
        //make matcher to match any flights that are exactly the same
        //we use a matcher because it is better than making a long sql
        //query with 5 params
        //and rand
        ExampleMatcher matcher = ExampleMatcher.matchingAll().withIgnorePaths("flight_id", "seats_available");
        Random rand = new Random();
        //get bookingID off system time
        long bookingID = System.currentTimeMillis();

        //for every given reservation book it
        for (ReservationRequest request : requestedReservations){
            //for every given flight check if
            for (Flight requestedFlight: request.getFlight_request_info()){
                List<Flight> flightMatches = flightRepository.findAll(Example.of(requestedFlight, matcher));
                int reservationFlightID;
                if (flightMatches.isEmpty()){
                    //the requested flight exists
                    requestedFlight.setSeats_available(rand.nextInt(10, 50));
                    flightRepository.save(requestedFlight);
                    reservationFlightID = requestedFlight.getFlight_id();
                }else{
                    //else remove a passenger from the flight
                    Flight matchedFlight = flightMatches.get(0);
                    if (matchedFlight.getSeats_available() > 0) {
                        matchedFlight.addSeats_available(-1);
                        flightRepository.save(flightMatches.get(0));
                        reservationFlightID = matchedFlight.getFlight_id();
                    }else{
                        throw new FlightIsFull(matchedFlight.getDeparture_airport(), matchedFlight.getArrival_airport());
                    }
                }
                //for every flight make the appropriate reservation for the customer
                Reservation newReservation = new Reservation(request, bookingID, reservationFlightID, response.getSubject());
                reservationRepository.save(newReservation);
            }
        }
        //return created status all good!
        Map<String, Long> ret = new HashMap<>();
        ret.put("booking_id", bookingID);
        return ResponseEntity.created(
            URI.create("https://www.purpledinoapi.link:8080/api/reservations/booking/" + bookingID))
            .body(ret);
    }
```

3.4.3.2a Code for POST mapping

Here is the PUT mapping. 1 or more changes to a reservation can be sent through here.

```
/**
     * Given a dict of changes, change the associated reservation
     * @param auth auth string
     * @param changes the fields to change
     * @param reservationId the id of the reservation to change
     * @return the newly changed reservation
     */
    @PutMapping("/{id}")
    Reservation changeReservation(@RequestHeader("Authorization") String auth,@RequestBody Map<String, String> changes,
            @PathVariable("id") int reservationId){
        TokenVerifierResponse Response = TokenVerifier.verifyToken(auth);

        Reservation existingReservation = reservationRepository.findById(reservationId).orElseThrow(() -> new
            ReservationNotFoundException(reservationId));

        existingReservation.update(changes);

        reservationRepository.save(existingReservation);

        return existingReservation;
    }
```

3.4.3.2a Code for PUT mapping

And finally, here are the DELETE mappings. The first endpoint will delete multiple reservations that are a part of a booking. The second endpoint will delete a single reservation.

```
/**
     * Delete a requested booking
     * @param auth auth string
     * @param bookingID the booking id to deleted
     * @return ok if every thing went well
     */
    @DeleteMapping("/booking/{id}")
    HttpStatus deleteBooking(@RequestHeader("Authorization") String auth, @PathVariable("id") long bookingID) {
        TokenVerifierResponse response = TokenVerifier.verifyToken(auth);
        String bookingOwner = reservationRepository.findBookingOwner(bookingID);
        if (!bookingOwner.equals(response.getSubject())){
            throw new InvalidCredentials("You are not authorized to delete the requested booking.");
        }
        List<Reservation> reservationsInBooking = reservationRepository.findByBookingId(bookingID);
        for (Reservation r : reservationsInBooking){
            reservationRepository.deleteById(r.getReservation_id());
        }

        return HttpStatus.OK;
    }

    /**
     * delete a requested reservation, this is singular
     * use booking for multiple
     * @param auth auth string
     * @param reservationId the id of the reservations to be deleted
     * @return ok if everything went well
     */
    @DeleteMapping("/{id}")
    HttpStatus deleteReservation(@RequestHeader("Authorization") String auth, @PathVariable("id") int reservationId){
        TokenVerifierResponse response = TokenVerifier.verifyToken(auth);
        Reservation reservation = reservationRepository.findById(reservationId).orElseThrow(() -> new
            ReservationNotFoundException(reservationId));
        //Check if the user owns the requested reservation to be deleted
        if (!reservation.getSubject_id().equals(response.getSubject())){
            throw new InvalidCredentials("You are not authorized to delete the requested reservation.");
        }
        reservationRepository.deleteById(reservationId);

        return HttpStatus.OK;
    }
```

3.4.3.2a Code for DELETE mappings

3.4.3.2b Amadeus Controllers

We use the Amadeus API to get locations and flight offers. The controllers for the API are structured slightly differently than our database rest controllers. The first difference is that we use GSON to serialize the responses that get returned by the api. The reason is that responses don't work with Spring boot's default serializer, Jackson. Additionally, one of GSON's strengths is that it easily converts java objects to and from JSON.

```
//The returned Objects are serialized with GSON, have to convert it to a string
return gson.toJson(Amadeus.Connect.getFlightOffers(p));
```

3.4.3.2b Example demonstrating use of GSON conversion to JSON

The Location Names controller is made up of 2 key parts. The first part is what connects to the actual api and makes a call with the given parameters.

```
/**
 * @param parameters the parameters for the function
 * @return returns an array of location objects
 * @throws ResponseException Amadeus response error
 */
public Location[] getLocationNames(Params parameters) throws ResponseException {
  return amadeus.referenceData.locations.get(parameters);
}
```

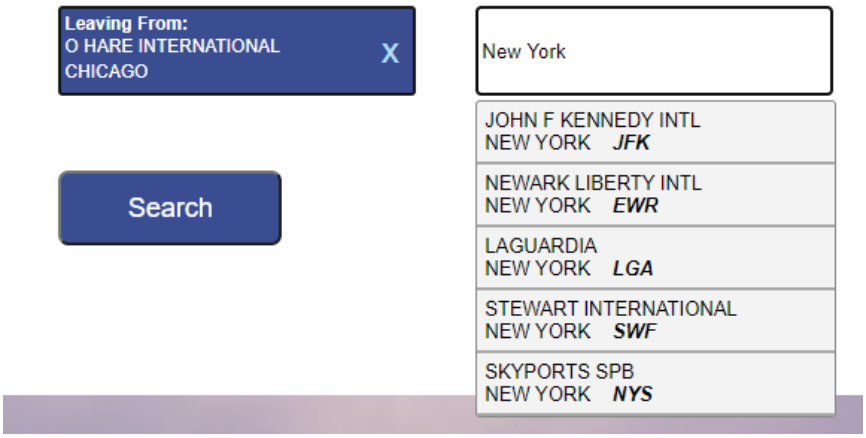
3.4.3.2b First part of Location controller connecting to API

The second part is the entrypoint for the frontend to interact with the API. It passes a keyword which would either be a city name, airport name, or airport code. By default, we are setting 3 parameters to limit the information we receive. Then we return the API's response through the controller.

```
  @GetMapping
  public String locations(@RequestParam( ) String keyword) throws ResponseException {
    if (keyword == ""){
      return "[]";
    }
    Params p = Params.with("keyword", keyword);
    p.and("subType", Locations.ANY);
    p.and("page[limit]", 100);
    p.and("view", "LIGHT");
    return gson.toJson(Amadeus.Connect.getLocationNames(p));
  }
}
```

3.4.3.2b Second part of Location Controller returning API response

On the frontend, the user will see a list of all the matching locations to choose from.



3.4.3.2b Front end view after API response

The flight offers controller is created the same way. The first part connects to the API.

```
  /**
   * Amadeus api function to request flight offers
   * @param parameters the paramaters of the request
   * @return an array with all the offers available
   * @throws ResponseException Amadeus response error
   */
  public FlightOfferSearch[] getFlightOffers(Params parameters) throws ResponseException {
    return amadeus.shopping.flightOffersSearch.get(parameters);
  }
```

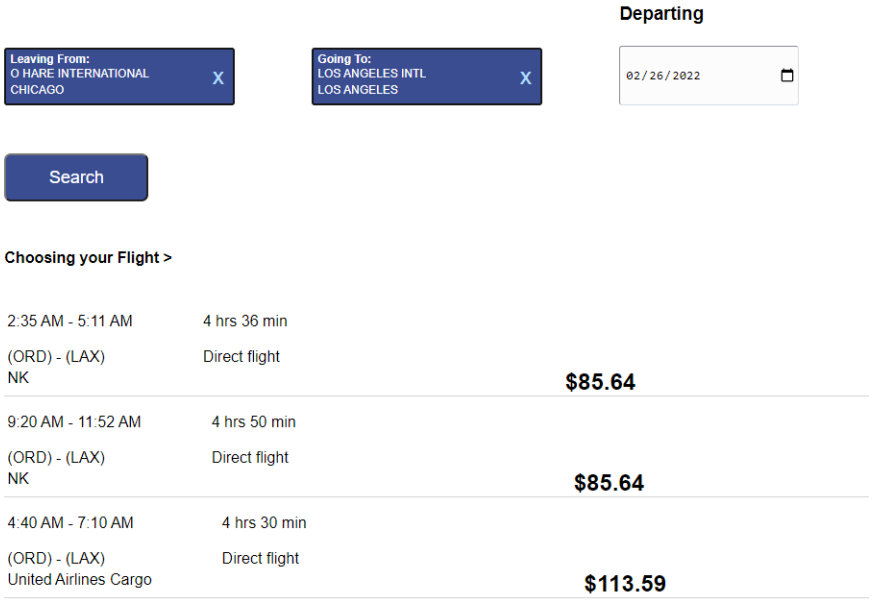
3.4.3.2b First part of Flight Offers controller connecting to API

The second part deconstructs all the parameters the Front End sent. Then, there is the case that one of the parameters is invalid, so we catch the exception. If no exceptions are thrown, then a list of all flight offers is returned.

```
  @GetMapping
  public String getFlight (@RequestParam Map<String, String> requestParameters) throws ResponseException
  {
    //Make a new parameter object and add the request to it
    Params p = Params.with("currencyCode", "USD");
    for(Map.Entry<String, String> pair : requestParameters.entrySet()){
      p.and(pair.getKey(), pair.getValue());
    }

    //Try and make the request if it fails the parameters are wrong and
    //raise an exception
    try{
      //The returned Objects are serialized with GSON, have to convert it to a string
      return gson.toJson(Amadeus.Connect.getFlightOffers(p));
    }catch (ClientException e){
      throw new flightOfferException(e.getMessage());
    }
  }
}
```

3.4.3.2b Second part of controller deconstructing parameters sent from Front End

On the frontend, this is the list of flights that get created.



3.4.3.2b List of flights displayed to user on the Front End

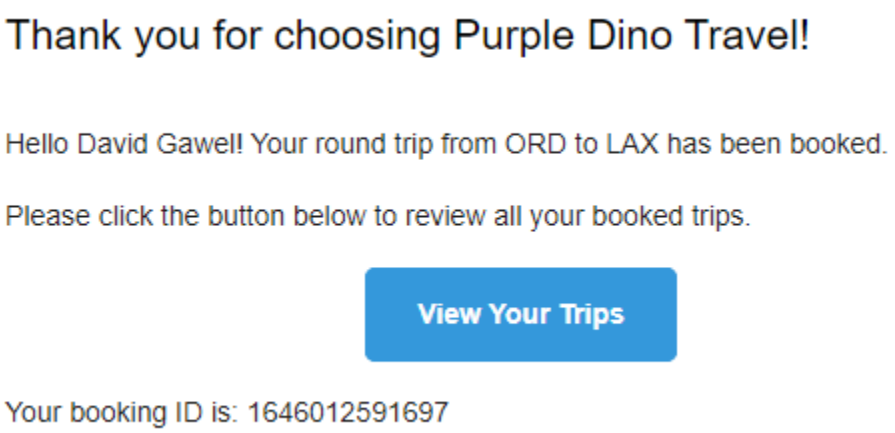
3.4.3.3 Email Confirmation

The way we send email confirmations to the user is simple. All we do is use the JavaMail API. We connect to the Google SMTP Mail server with SSL, and provide our username, and password. Whenever a valid reservation is made we just call the send mail function with the valid info. What this does is that it reads from a pre-made html template we have. We parse this template and stylize it with the user data and then send it over.

```java
/**
 * Method for sending confirmation emails
 * @param toEmail the email address to send the message to
 * @param Name the name to use is the email
 * @param BookingID the bookingID to use in the subject
 */
private void sendEmail(String toEmail, String Name, long BookingID, tripType type, String tripName){
    String from = "purpledinoair@gmail.com";
    try {
        // Create a default MimeMessage object.
        MimeMessage message = new MimeMessage(session);

        // Set From: header field of the header.
        message.setFrom(new InternetAddress(from));

        // Set To: header field of the header.
        message.addRecipient(Message.RecipientType.TO, new InternetAddress(toEmail));

        // Set Subject: header field
        message.setSubject("Purple Dino Travel Confirmation - " + BookingID);

        //get the html file
        //I use jsoup because it is easy to parse html files.
        //I tried using the default java formatter and a message
        //formatter, but they did not work. Jsoup was the best option to
        //make good customizable content.
        Document doc = Jsoup.parse(getClass().getResourceAsStream( name: "/templates/EmailMessage.html"),  charsetName: "UTF-8",  baseUri: "https://daniel-mccarthy.github.io");
        //set the name values
        Element tag = doc.select( cssQuery: "p.nameTag").first();
        tag.replaceWith(doc.createElement( tagName: "p")
            .appendText("Hello " + Name + "! Your " + type + " from " + tripName + " has been booked."));
        //set the booking id values
        tag = doc.select( cssQuery: "p.idTag").first();
        tag.replaceWith(doc.createElement( tagName: "p")
            .appendText("Your booking ID is: " + BookingID));

        // Now set the actual message
        message.setContent(doc.toString(),  type: "text/html; charset=utf-8");
        // Send message
        Transport.send(message);
    } catch (MessagingException | IOException mex) {
        mex.printStackTrace();
    }
```

3.4.3.3 Code of the `sendEmail()` *function*

The user receives this nicely styled email from purpledinoair@gmail.com with appropriate user data like Booking ID, their name, and the flight name they just ordered. The button takes them to the trips page to view all the trips that they have booked.



3.4.3.3 Example screenshot of Confirmation Email sent to user

3.4.4 Testing

Testing is done through the JUnit testing library. Since many of the backend endpoints require user authentication, it was important to ensure user tokens were working as intended. Here is the test class to test if the Token Verifier class was correctly validating tokens.

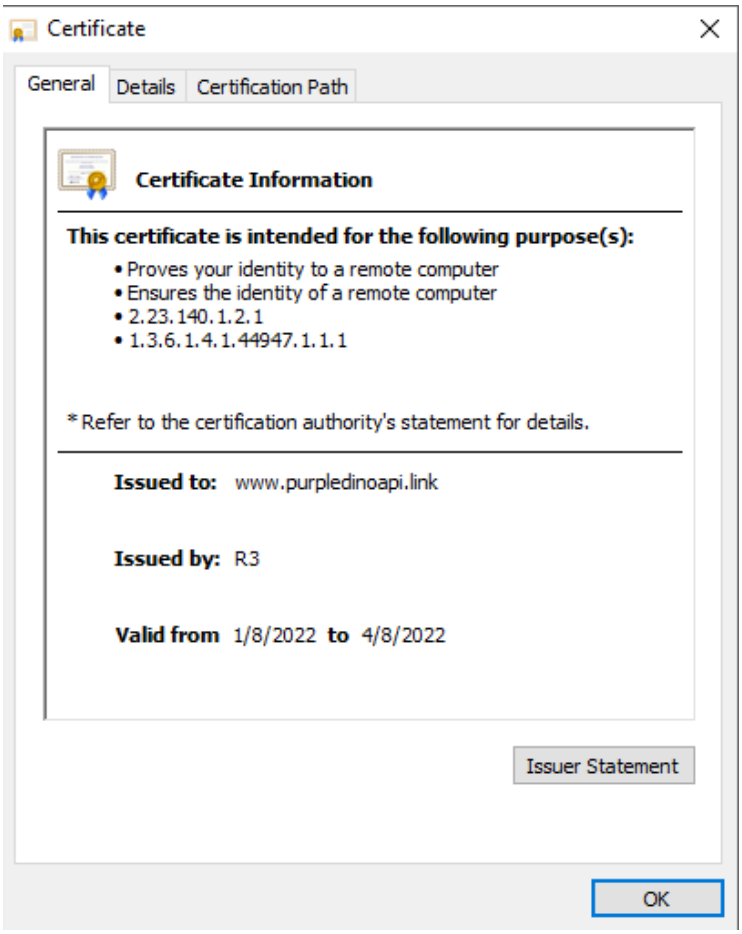
```
public class TokenVerifierTests {

  private final String ValidToken;
  private final String InValidToken;

  public TokenVerifierTests(){
    Properties properties = new Properties();
    //open the application-test file from the classpath
    try(InputStream is = getClass().getResourceAsStream("/application-test.properties")){
      properties.load(is);
    }catch (IOException e){
      e.printStackTrace();
    }
    //set valid and invalid
    ValidToken = properties.getProperty("ValidToken");
    InValidToken = properties.getProperty("InValidToken");
  }

  @Test
  void TestInValid(){
    Assertions.assertThrows(TokenInvalid.class, () -> {TokenVerifier.verifyToken(InValidToken);});
  }

  //If this test fails the token may have expired
  @Test
  void TestValid(){
      Assertions.assertTrue(TokenVerifier.verifyToken(ValidToken).isValid());
  }

  @Test
  void TestTooLong(){
    Assertions.assertThrows(TokenInvalid.class, () -> {TokenVerifier.verifyToken(ValidToken + "aaa");});
  }

  @Test
  void TestTooShort(){
    Assertions.assertThrows(TokenInvalid.class, () -> {TokenVerifier.verifyToken("a");});
  }
}
```

3.4.4 Example of some JUnit tests for verifying Tokens

3.4.5 Security

3.4.5a SSL Certificate

We have a SSL certificate authorized by let's encrypt. This secures the connection between the backend and the frontend protecting users from a man in the middle attack.



3.4.5a Image of valid SSL Certificate

3.4.5b CORs Policy

We have a cors policy to only allow requests from our website's domain. Though this doesn't completely prevent XSS attacks, it does help to prevent them.

```
public FilterRegistrationBean corsFilter() {
  UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
  CorsConfiguration config = new CorsConfiguration();
  config.setAllowCredentials(true);
  config.addAllowedOrigin("https://daniel-mccarthy.github.io");
  config.addAllowedHeader("*");
  config.addAllowedMethod("*");
  source.registerCorsConfiguration( pattern: "/**", config);
  FilterRegistrationBean bean = new FilterRegistrationBean(new CorsFilter(source));
  bean.setOrder(0);
  return bean;
}
```

3.4.5b Image of the CORs filter used in the Back End

3.4.5c REACT Input Sanitization & JPA Parameterization

By default REACT sanitizes input, this prevents users from giving the api malicious data like SQL injection attacks. Of course we can not trust that all requests come from the frontend so we also parameterize our queries. This also prevents SQL injection attacks.

3.4.5d App.properties

To protect our API keys and passwords we put all our important keys in an applications.properties file. This file is not shared on github and we only have access to it internally in our shared google drive.

3.4.6 AWS Backups

We have multiple backups of our EC2 instance. During the runtime of the project these backups got made daily, currently these backups get made every month. If anything terrible happens like all our data gets deleted or corrupted, we could always return it to a functioning state.

3.4.7 Budget alerts

While these servers are part of AWS free tier we have to be careful of any unexpected charges that could occur. As a precaution once the budget goes over $4 we are alerted of this and we can take any actions to get these errors remedied.

3.4.8 Errors Received During Development

When Hibernate connects Java classes and fields to a database, the variable name gets converted to snake case. So the database fields in the MySQL tables had to be snake-case as well, otherwise Hibernate would create an extra column in the table. Likewise, it was important to use snake-case when filling out the column annotations for the Java fields. If we didn't, hibernate would create a new column using the annotated column name and use that for data instead.

When we started working on the endpoints, Internal Server Errors would get thrown if a request didn't match what the endpoint expected. For example, an Internal Server Error would throw if a user requested an ID that didn't exist in the database. So, we created multiple exceptions for each of the controllers that covered various edge cases. Now instead of an Internal Server Error getting thrown, a Json response gets thrown outlining the reason why the request was invalid and a http status code.

4.0 Team Collaboration

4.1 Agendas

Every team meeting had the following agenda. A couple meetings required a more in depth agenda and they are shown below.

General Agenda

- ☐ **Discuss feedback from Professor**
- ☐ **Dev Team gives update**
  - ☐ **Walk through**
  - ☐ **What has been done**
  - ☐ **What needs to be done**
- ☐ **Week plan**
  - ☐ **Plan for next submission deadline**
- ☐ **Other class deadlines**
- ☐ **Open floor/Misc**

Main Topic - Development

1) **Discuss RPD feedback**
2) **Dev team to show progress**
   a) What is left?
   b) How can we help?
   c) What questions do you have for the group?
3) **Jackson to discuss creating and documenting development deadlines.**
   a) Do you guys want hard deadlines for specific tasks?
   b) Jackson will be filling out the calendar with development deadlines for class documentation purposes.
4) **Jackson to remind the group of other class deadlines.**
5) **Open floor**

Upcoming group deadlines-
Preview Demo due Feb 20, 2022 11:30 PM
Final Demo due Feb 27, 2022 11:30 PM

Team created deadlines -
Development to be mostly done Feb 7

Keep updating time log and todos!

4.1 Example Agenda used for 1/31 Meeting

5.0 Timelogs

Each time log depicts the number of hours each individual worked to make sure that they fulfill all of their assigned project responsibilities. It also highlights how other group members collaborated and helped each other to make sure that the project was a success.

Below is a timelog including all mandatory group meetings, as well as the combined total hours of the project as a whole.

Name	Date	Hours	FULL description of the task in 3 sentences or fewer.
Everyone	1/3/2022	1.5	First class meeting
Everyone	1/4/2022	1.25	Extra group meeting on group roles and what to do next
Everyone	1/5/2022	1	Dev team talked about plan on creating front and back end
Everyone	1/7/2022	1	Demo for team, talked about outstanding task before deadline, asked any questions anyone had
Everyone	1/10/2022	1	Second class meeting. Went over the agenda given by the professor.
Everyone	1/17/2022	1	MLK day unofficial meeting, went over planning and requirements doc and a bit of the schedule going forward
Everyone	1/24/2022	1	Class meeting, discussed what needs to be done for final sprint before RDP is due
Everyone	1/31/2022	1	Class meeting, went over RDP feedback, figured out what needs to be done for preview demo
Everyone	2/7/22	1	Class meeting, development update and discuss project manual
Everyone	2/14/2022	1	Class meeting, development update and discuss project manual
Everyone	2/21/2022	1	Class meeting, discussed plans for the final demo and what to do with the project manual
Everyone	2/28/2022	1	Class meeting, celebrated end of project. Status report on project manual.
Total		12.75	
Total (Meeting Hours)		127.5	
Total (Person Hours)		552.54	
Total (Cumulative)		680.04	

5.1 Jackson Meyer

Name	Date	Hours	FULL description of the task in 3 sentences or fewer.
Jackson Meyer	1/4/2022	2.5	Organized documents, read more on requirements, planning for the first week and demo. Second full group meeting (not mandatory).
Jackson Meyer	1/5/2022	1	Corresponded with Mario. Sorted out issues and questions about the Requirements Document.
Jackson Meyer	1/7/2022	2	Brainstormed ideas for how to distinguish our product. Did more reading on requirements for presentation. Fourth meeting, getting prepared for presentation.
Jackson Meyer	1/8/2022	1.5	Recorded intro clip for proof of concept demo, rewatched lecture material to get more clear idea about group roles
Jackson Meyer	1/9/2022	2.5	Finishing up checklist documents, coordinating with groupmates about final touches. Showed Suhaib how to upload to Panopto.
Jackson Meyer	1/10/2022	0.5	Prepared for the second group meeting. Made sure all submissions were completed. Corresponding with professor.
Jackson Meyer	1/10/2022	1.25	Second mandatory meeting. Re-organized group roles. Revised necessary documents and uploaded to D2L.
Jackson Meyer	1/16/2022	0.5	Brainstormed ways to be more proactive with the group, corresponded with Jenna for meeting planning.
Jackson Meyer	1/17/2022	1.25	Third meeting (unofficial). Prep for the meeting and brought up the idea of 1on1's.
Jackson Meyer	1/18/2022	1	Met with Mario to go over the PM aspect of the Requirements document and the Negotiator role.
Jackson Meyer	1/22/2022	1	Updating documents, talked with group members about deadlines and delegated work on Requirements Matrix.
Jackson Meyer	1/23/2022	2	Evaluated requirements document. Met with Mario to fill out pertinent sections of said document. Brought Testing Matrix to his attention and formed a plan to work on it.
Jackson Meyer	1/24/2022	0.75	Prepared for the fourth full-group meeting. Elaborated on 1 on 1's, sent out directions for showing me availability.
Jackson Meyer	1/24/2022	0.5	1on1 meeting with Nick Clark.
Jackson Meyer	1/26/2022	0.5	1on1 with Suhaib Mikbel.
Jackson Meyer	1/26/2022	0.5	1on1 with Jenna Dahl-Crimmins.
Jackson Meyer	1/26/2022	0.25	1on1 with Muhammad Fahad.
Jackson Meyer	1/27/2022	0.15	1on1 with Philip Reeves.
Jackson Meyer	1/27/2022	0.15	1on1 with Jawan Luangnikone Davis.
Jackson Meyer	1/27/2022	0.25	1on1 with David Gawel.
Jackson Meyer	1/27/2022	0.5	1on1 with Dan McCarthy.
Jackson Meyer	1/31/2022	0.65	Fifth full group meeting. Went over planning/deadlines and talked about fleshing them out. Reminded to schedule 1 on 1's. Props for a great RDP demo.

Jackson Meyer	2/2/2022	0.5	1on1 with Nick Clark.
Jackson Meyer	2/2/2022	0.35	1on1 with Jawan Luangnikone Davis.
Jackson Meyer	2/2/2022	0.5	1on1 with Muhammad Fahad.
Jackson Meyer	2/3/2022	0.75	1on1 with Suhaib Mikbel.
Jackson Meyer	2/3/2022	0.5	1on1 with Philip Reeves.
Jackson Meyer	2/3/2022	0.5	1on1 with Mario DiBartolomeo.
Jackson Meyer	2/5/2022	0.5	1on1 with Jenna Dahl-Crimmins.
Jackson Meyer	2/5/2022	0.5	1on1 with David Gawel.
Jackson Meyer	2/7/2022	1.25	6th full group meeting. Talked about beginning work on the Project Manual beyond the outline. Update from dev team on current progress, reminded of deadlines.
Jackson Meyer	2/9/2022	0.25	1on1 with Philip Reeves.
Jackson Meyer	2/9/2022	0.4	1on1 with Suhaib Mikbel.
Jackson Meyer	2/10/2022	0.5	1on1 with Nick Clark.
Jackson Meyer	2/10/2022	0.75	1on1 with Muhammad Fahad.
Jackson Meyer	2/11/2022	0.15	1on1 with Jawan Luangnikone Davis.
Jackson Meyer	2/11/2022	0.25	1on1 with David Gawel.
Jackson Meyer	2/11/2022	0.15	1on1 with Mario DiBartolomeo.
Jackson Meyer	2/12/2022	0.25	1on1 with Dan McCarthy.
Jackson Meyer	2/14/2022	0.5	1on1 with Jenna Dahl-Crimmins. Cemented planning for Project Manual
Jackson Meyer	2/14/2022	1	7th full group meeting. Status update on dev progress, status update on preview demo outline. Talked at length about planning for the project manual and who's doing what.
Jackson Meyer	2/16/2022	0.25	1on1 with Philip Reeves.
Jackson Meyer	2/16/2022	1.5	Recorded intro clip for preview demo, editing of said clip.
Jackson Meyer	2/16/2022	0.5	1on1 with Nick Clark.
Jackson Meyer	2/17/2022	0.25	1on1 with Mario DiBartolomeo.
Jackson Meyer	2/21/2022	0.5	1on1 with Jenna Dahl-Crimmins. Worked out some new deadlines for the Project Manual and the work coming up.
Jackson Meyer	2/21/2022	0.5	8th full group meeting. Props to a great Preview Demo, talked about the notes and deadlines for the remaining parts to be turned in for the Final demo. Deadlines for Project Manual covered again.
Jackson Meyer	2/21/2022	0.25	1on1 with Suhaib Mikbel. Covered what he needs to do for the Project Manual skeleton and deadline for it.
Jackson Meyer	2/22/2022	0.2	Quick follow up with Suhaib on skeleton of Project Manual, tips and gave an all good.
Jackson Meyer	2/23/2022	0.25	1on1 with Philip Reeves.
Jackson Meyer	2/23/2022	0.25	1on1 with Jawan Luangnikone Davis.

Jackson Meyer	2/23/2022	1	1on1 with Nick Clark.
Jackson Meyer	2/23/2022	0.25	1on1 with Muhammad Fahad.
Jackson Meyer	2/24/2022	0.25	1on1 with Mario DiBartolomeo.
Jackson Meyer	2/27/2022	0.4	Edited the formatting of the Project Manual, made it more uniform and easier to read. Formatted each header to be more easily distinguished from the body of text.
Jackson Meyer	2/28/2022	0.9	Final group meeting. Reviewed progress on the Project Manual as a whole and gave tips on formatting and what needs to be changed. Reminded everyone of deadlines for Project Manual.
Jackson Meyer	3/2/2022	1	Meeting with Suhaib to talk about Project Manual formatting and things to look at. Talked about revisions of the Introductions section, editing member's photos.
Jackson Meyer	3/4/2022	0.75	Another meeting with Suhaib to assess Project Manual progress. Assigned his work and what I have left to do.
Jackson Meyer	3/4/2022	0.5	Meeting with Mario to talk about formatting issues with Requirements documents and some ways to troubleshoot. Discussed my plans for formatting the Requirements Matrix to fit better.
Jackson Meyer	3/5/2022	1.5	Format sections of Project Manual, corresponding with group members about what needs to be fixed.
Jackson Meyer	3/6/2022	1.5	Inputting Back End section to main Project Manual document.
Jackson Meyer	3/6/2022	1	Meeting with Suhaib to assess final tweaks to be done on Project Manual.
Jackson Meyer	3/6/2022	3	Finished inputting Back End section, inputted Front End section into Project Manual document. General formatting of documents.
Jackson Meyer	3/6/2022	1	Removed entire Requirements Matrix, replaced each section with smaller images of beginning of each matrix (saves space, looks cleaner).
Jackson Meyer	3/6/2022	2.5	Major reformatting of Project Manual, unifying font and font size, fixing headers and grammar mistakes.
Jackson Meyer	3/6/2022	2	Final review of Project Manual, fixed up timelog formatting. Added in full-group timelog (thanks David for pointing that out). Reformatted "Videos" section.
	Total	52.75	

5.2 Nick Clark

Name	Date	Hours	FULL description of the task in 3 sentences or fewer.
Nick Clark	1/4/2022	0.2	Managing project files in the Google Drive and starting the GitHub repo to store the backend code.
Nick Clark	1/5/2022	2	Troubleshooting and experimenting with the Apache web server to host the project's code.
Nick Clark	1/5/2022	1.5	Meeting for designing and planning the concept demo
Nick Clark	1/6/2022	2	Creating production and testing databases on the MySql server. Added User, Reservation, and Flight tables and some example data to the test DB.
Nick Clark	1/7/2022	3.5	Began implementing REST endpoints for the Users, Flights, and Reservations tables in separate branches from the Main branch on GitHub.
Nick Clark	1/8/2022	1.5	Debugging REST endpoints and mySQL database entries. Extra columns were unintentionally added and the Users primary key was not incrementing as intended.
Nick Clark	1/12/2022	1.5	Researched how to implement response codes to endpoints with Spring and started to implement them to the flight-response-codes branch.
Nick Clark	1/13/2022	0.2	Added test data back to the testing DB (Flight, Reservation, and User tables) after its security was refactored.
Nick Clark	1/13/2022	0.5	Meeting with Mario and the front end team to cover the initial requirements for the requirements doc.
Nick Clark	1/13/2022	1.5	Finished working on the flight-response-codes branch to add response codes for the REST API. Created an exception to raise a 404 status code if an item is not found.
Nick Clark	1/16/2022	1	Investigated issue in which the backend api (version 0.6) would reject POST requests. Found that the API is at least able to accept curl requests.
Nick Clark	1/16/2022	1	Committed a change to the flight-response-codes branch to remove a redundant link that was a part of the GET request body. Also looked into removing the _embedded nesting without success.
Nick Clark	1/17/2022	1	Added initial documentation to the Flight, FlightController, and FlightModelAssembler classes in the flight-response-codes branch.
Nick Clark	1/19/2022	0.5	Resolved merge conflicts and merged the flight-response-codes branch into the main branch
Nick Clark	1/20/2022	0.3	Meeting with Jenna and David to add our first set of requirements to the planner
Nick Clark	1/22/2022	1.5	Starting implementing response codes for the reservations and user tables under the reservations-users-codes branch
Nick Clark	1/23/2022	1	Meeting with Mario, frontend, backend, testing, and creation teams to discuss project phases and the requirements document
Nick Clark	1/23/2022	1.25	First pass through the requirements document as the bad client
Nick Clark	1/23/2022	1.5	Finished adding response codes to the Reservations and Users endpoint in the reservations-users-codes branch. Also added documentation to their respective classes and

			methods. Merged branch into main
Nick Clark	1/24/2022	0.5	Continued with my first pass through the requirements document as the bad client for sections 9, 13, and 14.
Nick Clark	1/24/2022	1.5	Meeting with mario to add the second version of revisions to requirements I have not approved
Nick Clark	1/24/2022	0.5	Week 4 1on1 with Jackson to cover the branch I have pushed to the backend and plans for this week
Nick Clark	1/27/2022	1	Brainstorming ideas on how to store reserved seat details for the reservation endpoint. 1 reservation per seat vs 1 reservation for all seats.
Nick Clark	1/27/2022	0.5	Meeting with David to discuss our goals and plans for the next 1.5 weeks in order to hit the Feb 7th deadline.
Nick Clark	1/27/2022	1.5	Started adding support to send an array of reservations to the POST endpoint in the post-reservations-array branch
Nick Clark	1/29/2022	0.5	Finished working on the post-reservations-array branch. Merged into the main branch
Nick Clark	1/29/2022	1	Started a new branch, reservation-seat and added seat details to the reservations table. Merged the branch into main and added the new SQL schema to the design folder
Nick Clark	1/30/2022	0.75	Added seat details to the database to reflect the new schema created in the reservation-seat branch. Also added test data to these new columns.
Nick Clark	1/31/2022	0.75	Added a limit (50) to how many reservations could be added through the /api/reservations/multi endpoint.
Nick Clark	1/31/2022	1	Meeting with Dan and David to talk about the support, about us, login, and trips webpages. Cloned the frontend repo and verified that it worked on my machine.
Nick Clark	1/31/2022	0.5	Started the support page for the frontend under the support-page branch. It currently just has the header and example text.
Nick Clark	2/1/2022	1.5	Continued working on the support-page branch. Added the support page buttons and the complaint form.
Nick Clark	2/2/2022	0.5	Added flight_code field to the flights table and removed seats_available. Pushed changes to main.
Nick Clark	2/2/2022	2	Continued working on the support-page branch. Added functionality to hide/show the complaint form, review form, and policies depending on what the user has selected. Added styling to the complaint form.
Nick Clark	2/2/2022	0.5	Added a method to the reservations controller that filters all reservations created by a specific user. Pushed the changes to main.
Nick Clark	2/2/2022	1	Met with Dan to talk about an issue where the ToastMessage was not re appearing as intended. Then added the changes from the Toast-Manual-Close branch
Nick Clark	2/2/2022	0.5	Met with Jackson to have our week 5 1on1 to cover the support page progress and the new endpoint added to the reservations api.
Nick Clark	2/2/2022	0.5	Styling support page so that the buttons were spaced out more. Also made the policies paragraphs closer to the center of the page

Nick Clark	2/3/2022	2	Added a table with test data and the required buttons to the trips page. Styled the page and table to look like the design posted in the shared design folder. Committed changes to the flight-page branch
Nick Clark	2/4/2022	2	Merged changes from main and Toast-Manual-Close into the support-page branch. Added and styled the review form and added 3 policies under the policies tab. Submitted a PR to merge support-page into main.
Nick Clark	2/5/2022	0.5	Meeting with Muhammad to go over the fields for the trips page table. We covered the table headers and how the "cancel flight," "add flight," and "update flight" buttons should work.
Nick Clark	2/5/2022	2.5	Refactored trips page to send table data as a prop to the trips table. Added a fuction to get reservations by a user from the backend API. Tried pulling data from the function to send to the trips table - was not successful.
Nick Clark	2/6/2022	3.5	Adding functionality to the "cancel flight" and "update flights" buttons. Submitted a PR to merge trips-page into main
Nick Clark	2/6/2022	1	Added selected and unselected styling for the "cancel flight" and "update flight" buttons on the trip page. Also started looking into issue where submitting review or complaint form refreshes the whole support page
Nick Clark	2/6/2022	2.5	Meeting with Dan to get help on Promises<> and React props in order to display backend data to the trips table. Added interfaces to the reservation api requests and a function to create the initial table data to display.
Nick Clark	2/7/2022	3.5	Added get, update, and delete functions to the frontend reservations api. Then added a get function to the frontend flights api. Successfully loaded the backend data into the trips page table and added a function to cancel/update a flight on our DB.
Nick Clark	2/7/2022	1.5	Talked with David, Muhammad, and Dan about what remains to be completed on the frontend and backend to get the project working by Feb 16th.
Nick Clark	2/8/2022	0.15	On the support page, I made all forms hidden by default. Previously, the complaint form was shown when the page was loaded.
Nick Clark	2/8/2022	1.5	Added an update form on the trips page for a traveler to change their name and the amount of checked bags they have for a reservation.
Nick Clark	2/9/2022	2	Styled the update form on the trips page. Changed "update checked bags" textbox to be 2 seperate buttons to add/subtract bags. Added React states to hold the name, checked bags, and price.
Nick Clark	2/9/2022	1	Added functionality to the "confirm changes" button on the trips page to update data in the DB when a user changes their reservation details.
Nick Clark	2/9/2022	0.67	Meeting with Dan and Muhammad to talk about user authentication and how it affects current schemas. Also went over details for the trips page and reservation review page
Nick Clark	2/10/2022	0.5	Met with Jackson to have our week 6 1on1 to cover the updated trips page, new schema plans, and passing user info to the trips page.
Nick Clark	2/10/2022	0.5	Met with Mario to help him get the backend code cloned to his computer, build the app, and make a commit to the complaints-controller branch
Nick Clark	2/11/2022	0.5	Merged trips-page branch with Login_on_home_page to add user token props to the trips

			page. Also added links to the login page to link to the support and trips page.
Nick Clark	2/11/2022	2	For the reservation review, I added and styled a flight details component and flight summary component. Published changes to the reservation-review-components branch
Nick Clark	2/11/2022	0.5	Redirected the user to the login page if they're not logged in and try to access the trips page
Nick Clark	2/14/2022	0.75	Talking and planning with David about how the production Reservation table should be built.
Nick Clark	2/14/2022	0.75	Meeting with David, Muhammad, and Dan to discuss the final steps of the application. Also talked about the plan on how we are going to merge all of the branches
Nick Clark	2/14/2022	1	Created the "About Us" page and added content to it. Submitted a pr to merged it into main
Nick Clark	2/14/2022	1	Created presentation slides for the backend material I'll be covering
Nick Clark	2/15/2022	1	Started the checkout page. Added the form for user information. Committed changes to the checkout-page branch
Nick Clark	2/15/2022	1	Added screenshots of the rest controllers and code snippets to my presentation slides
Nick Clark	2/16/2022	0.75	Meeting with Dan to talk about the checkout page and how we should implement it.
Nick Clark	2/16/2022	0.5	Meeting with David to get PostMan working on my computer and make a few requests to the new database schema
Nick Clark	2/16/2022	1	Worked on the form on the checkout page. Got the form information to pass the values back up to its parent in order to be sent to the checkout/receipt component
Nick Clark	2/16/2022	2	Updated the trips page to use the new schema. Reworked some of the requests to send a user's token through the header to the backend. Merged changes into main
Nick Clark	2/16/2022	0.25	Went through the website and made sure all the header links were working since we merged multiple branches into main.
Nick Clark	2/16/2022	0.5	1on1 with Jackson to go over what the dev team has done today and plans to finish. Also talked about the preview presentation
Nick Clark	2/16/2022	0.75	Created the receipt portion of the checkout page. Also styled the checkout page so that the form and receipt would appear next to each other.
Nick Clark	2/16/2022	1.5	Added an method to update a user's info based off their information inserted into the checkout form. Also met with Dan to talk about how the receipt should display fight information
Nick Clark	2/17/2022	1.5	Got reservation data to appear on the checkout receipt.
Nick Clark	2/17/2022	1.25	Added and styled the success page. Added the changes to the checkout-page branch
Nick Clark	2/17/2022	1	Added exception and controller screenshots to the preview presentation. Then recorded the presentation.
Nick Clark	2/17/2022	0.25	Added a component to the trips page that tells the user they need to log in first if they try to access it while being logged out
Nick Clark	2/18/2022	1	Implemented changes from Muhammad's usability document. Reduced size of trips table, made "add flight" text smaller, updated review and complaint forms, and added text to login page.
Nick Clark	2/18/2022	1	Moved the header to its own component to make updating it easier and made a PR. Updated

			the header text and made the trips table slightly wider to accommodate for the cancel/update icons
Nick Clark	2/21/2022	0.5	Continued working on the usability changes. Fixed price to show 2 decimals on trips page and checkout page. Also changed the checkout receipt to only show the flight total once for all segments instead of every segment.
Nick Clark	2/21/2022	2	Refactored the checkout page to use all the form information. Also styled the page so that required sections have a red border until they're filled out.Then when the booking is completed, the page redirects to the success page.
Nick Clark	2/21/2022	2.5	Refactored checkout form to not ask for email since it's already gotten from the user when they sign in. Refactored flight and user validation on the checkout page so that the complete booking button is only enabled when they're valid. Continued adding styling to buttons and links across the website.
Nick Clark	2/23/2022	1	1on1 with Jackson to cover final presentation, final website changes, DPU github profile readme, and project booklet formatting
Nick Clark	2/23/2022	1	Working on the DPUPurpleDino profile readme to provide a brief overview of our repositories and what our objective was.
NIck Clark	2/26/2022	1.5	Adding content to the project booklet. Added "software used", "errors received," and info on the backend controllers
Nick Clark	3/2/2022	1.5	Adding content to the project booklet. Added page information and screenshots for the support, about us, trips, and checkout pages.
Nick Clark	3/2/2022	1	Adding content to the project booklet. Added page information and screenshots for the "login page." Also added some info about testing
	Total	98.02	

5.3 Jenna Dahl-Crimmins

Name	Date	Hours	FULL description of the task in 3 sentences or fewer.
Jenna Dahl-Crimmins	1/4/2022	1	Planned week 1, creating project timeline
Jenna Dahl-Crimmins	1/4/2022	1.5	Updated timeline, created tabs on timelog, planned week 2-10
Jenna Dahl-Crimmins	1/8/2022	0.5	Created my part of the video for first demo
Jenna Dahl-Crimmins	1/11/2022	2	Created google sheet for plan part of RDP Demo
Jenna Dahl-Crimmins	1/16/2022	3	Added due dates and work to calendar, created module list
Jenna Dahl-Crimmins	1/16/2022	0.5	Made meeting agenda
Jenna Dahl-Crimmins	1/20/2022	2	Met with individuals to go over modules and answer any planning questions
Jenna Dahl-Crimmins	1/25/2022	0.5	Updated calendar and sent group important deadlines and reminders
Jenna Dahl-Crimmins	1/26/2022	0.5	Met with project manager to discuss current strengths and struggles
Jenna Dahl-Crimmins	1/27/2022	1.5	Got all material ready for RDP and recorded my part of the video
Jenna Dahl-Crimmins	1/31/2022	0.5	Created team meeting agenda
Jenna Dahl-Crimmins	2/5/2022	0.5	Met with project manager to discuss any adjustments needed to the plan
Jenna Dahl-Crimmins	2/7/2022	1	Updated Calendar with more task and deadlines for Feb
Jenna Dahl-Crimmins	2/8/2022	0.5	Started planning content for project manual
Jenna Dahl-Crimmins	2/11/2022	1	Met with Suhaib to go over project manual outline
Jenna Dahl-Crimmins	2/14/2022	0.5	Met with project manager to discuss project manual plan
Jenna Dahl-Crimmins	2/18/22	1	Update calendar and recorded my part of video
Jenna Dahl-Crimmins	2/21/22	0.5	Met with project manager to discuss plan for remaining to dos
Jenna Dahl-Crimmins	2/27/22	1.5	Updated timelog, to dos, and project manual
Jenna Dahl-Crimmins	3/3/22	0.75	Updated Project Manual
	Total	20.75	

5.4 Mario DiBartolomeo

Name	Date	Hours	FULL description of the task in 3 sentences or fewer.
Mario DiBartolomeo	1/4/2022	2	Writing up a basic requirements guide for the group. Requirements lecture.
Mario DiBartolomeo	1/7/2022	2	Organized Project Overview and Basic Requirements for the group.
Mario DiBartolomeo	1/8/2022	2.5	Finished writing project overview and recorded video to submit for demo
Mario DiBartolomeo	1/12/2022	1	Created a timeline for writing each section of the requirements document and mapped out which group mates I will need to consult while writing each section.
Mario DiBartolomeo	1/13/2022	1.5	Created a first draft of the technical requirements section with David and Jawan. Created requirements related to Response Time, System Dependability, Performance/Usability/Security/Maintainability.
Mario DiBartolomeo	1/14/2022	2	Created Risk management requirements section of the document.
Mario DiBartolomeo	1/15/2022	1.5	Finished Risk management requirements section of the document.
Mario DiBartolomeo	1/17/2022	4	Met with a group to review requirements finished thus far. Also created a Constraints, project planning and project roles requirements section to the requirements document.
Mario DiBatolomeo	1/18/2022	1	Created project manager requirements section with Jackson. Also did 1 on 1 meeting discussing progress of the requirements document.
Mario Dibartolomeo	1/22/2022	2	Created sections 10,11, and 12 of the requirements document. Reviewed Matrix
Mario Dibartolomeo	1/23/2022	2	Created sections 9, 13, and 14. Revised sections 6 and 8 with jackson.
Mario Dibartolomeo	1/24/2022	1	Finished filling out the first iteration of requirements matrix, created testing matrix.
Mario DiBartolomeo	1/24/2022	1.5	Reviewed and revised All Bad Client Attacks with Nick.
Mario DiBartoloeo	1/25/2022	0.5	Revised Section 3.2.5 and 3.2.6 with jawan to make more understandable
Mario DiBartolomeo	1/25/2022	2	Reviewed and revised all Bad dev attacks with David
Mario DiBartolomeo	02/10/2022	2	met with David and Nick individually to get assigned a task to help with the backend. I will be completing the complaints class for the application. They also helped me set up my machine properly.
Mario DiBartolomeo	02/14/2022	3.5	Implemented the Complaints API

Mario DiBartolomeo	02/15/2022	2	Implemented the review API
Mario DiBarotlomeo	02/15/2022	1	Recording Requirements section of the presentation
Mario DiBartolomeo	02/25/2022	0.5	Jackson 1 on 1
Mario DiBartolomeo	02/28/2022	2	Completed requirements section of the project manual.
	Total	37.5	

5.5 Philip Reeves

Name	Date	Hours	FULL description of the task in 3 sentences or fewer.
Philip Reeves	1/5/2022	2.5	Research various wordpress themes, plugins, and watched numerous youtube tutorial videos on how to build a functional airline reservation site.
			Also setup group meeting to link up with group members via zoom
Philip Reeves	1/5/2022	2.5	Helped with the design and planning of the project implementation.
Philip Reeves	1/8/2022	1.3	Wrote the presentation outline for the group and made sure everyone understood what they had to present, when it should be completed, and how much time they had to complete the presentation.
Philip Reeves	1/8/2022	1	Made a powerpoint for the group including every group member and their roles. Also did a presentation introducing the group members.
Philip Reeves	1/10/2022		Attend group meeting
Philip Reeves	1/15/2022		Attend group meeting
Philip Reeves	1/16/2022	0.3	Create the Requirement Demo Rubric
Philip Reeves	1/18/2022	2	Brainstorm ideas of how to make the presentation outline inorder for the presentation to be more effective.
Philip Reeves	1/23/2022	0.25	Check the website to make sure it was running.
Philip Reeves	1/24/2022	2.5	Created the presentation Outline and share with group
Philip Reeves	1/25/2022	0.5	Made a powerpoint for the presentation showing every group member and their roles.
Philip Reeves	1/27/2022	0.01	Send out a message on group discord reminding group members about presentation rehearsal
Philip Reeves	1/27/2022	0.15	Met with Jackson one-on-one
Philip Reeves	1/29/2022	0.03	Check with Suhaib to make sure if he has received videos from the group members, and the video has been edited, backups have been created, and link shared with group members
Philip Reeves	1/30/2022	0.03	Check with the Video manager to confirm that the video has been uploaded to D2L and link shared with the professor.
Philip Reeves	2/2/2022	0.3	Read the preview demo specification to brainstorm the presentation outline
Philip Reeves	2/2/2022	0.3	Wrote a Draft 1.0 of the presentation outline
Philip Reeves	2/3/2022	0.3	Revised the presentation outline and wrote a 2.0 draft.
Philip Reeves	2/3/2022	0.5	Met with Jackson one-on-one
Philip Reeves	2/6/2022	0.5	Read the preview demo specification again to make sure my 3.0 draft meets all of the specifications.
Philip Reeves	2/7/2022	0.05	Presented the 3.0 draft to the team members and asked for any suggestions.
Philip Reeves	2/9/2022	0.25	Met with Jackson one-on-one

Philip Reeves	2/11/2022	0.45	Brainstorm on how many minutes should be allocated to each group member based on what they are going to present.
Philip Reeves	2/16/2022	0.3	Completed the final Preview Demo Outline and shared with group members via Discord and Google shared drive
Philip Reeves	2/16/2022	0.25	Met with Jackson one-on-one
Philip Reeves	2/17/2022	0.05	Coordinated with Suhaib (Video Manager) to talk about a last minute tweak to the preview demo video.
Philip Reeves	2/17/2022	0.3	Made a rough draft of the Presentation Slides
Philip Reeves	2/18/2022	0.45	Completed powerpoint slides and shared with video manager
Philip Reeves	2/23/2022	0.25	Met with Jackson one-on-one
Philip Reeves	2/28/2022	1.45	Put in the presentation portion and the competition portion in the project manual
Philip Reeves	3/06/2022	0.45	Put in everyone updated time logs into the project manual and corrected grammatical errors
	Total	17.32	

5.6 Suhaib Mikbel

Name	Date	Hours	FULL description of the task in 3 sentences or fewer.
Suhaib Mikbel	1/4/22	0.5	Created concept for company logo and discussed with group
Suhaib Mikbel	1/5/22	0.75	Created logo and showed to group for revisions
Suhaib Mikbel	1/7/22	1	Created new logo and showed group for more feedback
Suhaib Mikbel	1/7/22	0.5	Accepted Feedback and changed positioning of words in Dino Travel Logo
Suhaib Mikbel	1/8/22	2	Accepted four out of 8 videos required for demo and started editing them
Suhaib Mikbel	1/9/22	2	Accepted the four missing video and edited them and added transitions and group names and roles
Suhaib Mikbel	1/9/22	0.25	Rendered video and shared the first draft of the video with group
Suhaib Mikbel	1/9/22	1	Reviewed feedback and adjusted audio levels for two members of the group, and then I rendered it
Suhaib Mikbel	1/9/22	1	Group approved the video and I uploaded it to Panopto, YouTube and posted it.
Suhaib Mikbel	1/10/22	2.5	Started outline for Project Manual so group members can get a head start
Suhaib Mikbel	1/11/2022	1.5	Completed rough draft for outline for the Project Manual
Suhaib Mikbel	1/18/2022	0.25	Reviewed Requirements outline, and made slight grammar changes
Suhaib Mikbel	1/22/2022	1	Created and Updated matrix for requirements document
Suhaib Mikbel	1/23/2022	1	Created and Updated matrix for requirements test document
Suhaib Mikbel	1/26/2022	0.5	Met with Project Manager to discuss current standing in group/ how everything is going
Suhaib Mikbel	1/27/2022	0.5	Searched and found background music for RDP Video
Suhaib Mikbel	1/28/2022	1	Started compiling videos for RDP Video
Suhaib Mikbel	1/29/2022	0.5	Met with Project Manager and discussed tasks for the week
Suhaib Mikbel	1/29/2022	1.5	Finished Editing and rendering video the first time.
Suhaib Mikbel	1/29/2022	1	Received feedback for video and re rendered
Suhaib Mikbel	1/29/2022	0.5	Met with the Project Manager to discuss tasks for the week.
Suhaib Mikbel	1/29/2022	1	Processed RDP Video and uploaded to YouTube and Panopto
Suhaib Mikbel	2/12/2022	1	Edited structure of project manual with Jenna.
Suhaib Mikbel	2/17/2022	1	Started compiling videos for Preview Video
Suhaib Mikbel	2/18/2022	1	Made transition cards for group members in video
Suhaib Mikbel	2/18/2022	0.5	Made end card for preview demo
Suhaib Mikbel	2/19/2022	2	Edited video, added music, adjusted audio levels, included transition cards, previewed the final product.
Suhaib Mikbel	2/19/2022	1.5	Rendered video and uploaded to YouTube and Panopto.
Suhaib Mikbel	2/21	1.5	Created new structure of project manual with linked table of contents

Suhaib Mikbel	2/22/2022	0.5	Met with project manager to discuss goals for the week, and upcoming preview demo
Suhaib Mikbel	3/1/2022	1	Met with project manager to discuss upcoming project manual
Suhaib Mikbel	3/4/2022	0.75	Met with project manager to assign formatting for project manual
Suhaib Mikbel	3/4/2022	1	Formatting the project manual: reformatting messy looking text and writing
Suhaib Mikbel	3/5/2022	4	Formatting the project manual: fixing images for better quality and added text
Suhaib Mikbel	3/6/2022	1	Met with project manager to discuss final version of project manual
	Total	38.5	

5.7 Muhammad Fahad

Name	Date	Hours	FULL description of the task in 3 sentences or fewer.
Muhammad F.	1/5/2022	1.5	Design and plan meeting for implementation
Muhammad F.	1/5/2022	1	Wireframe design for home page
Muhammad F.	1/7/2022	2	Analyzing user friendly designs/implementing key points to make application easy to use and effective
Muhammad F.	1/7/2022	1	Wireframe design 2 for flight results page
Muhammad F.	1/13/2022	1	Group meeting discussing user requirements and website layout
Muhammad F.	1/15/2022	1	Create basic homepage layout from design requirements
Muhammad F.	1/17/2022	1	Analyzing website user accounts profile and support structure/layouts
Muhammad F.	1/20/2022	0.5	Meeting with planner to discuss testing modules and to do requirements
Muhammad F.	1/20/2022	1.5	Research testing methods and figuring out required tests to do for usability study
Muhammad F.	1/21/2022	2.5	Create One-way/Multi-city/Round Trip front page & One-way/Roundtrip search page design
Muhammad F.	1/21/2022	0.5	Meet with Front end to discuss interactive components and websites flow
Muhammad F.	1/23/2022	1	Filling out requirements doc and understanding website backend api
Muhammad F.	1/26/2022	0.8	Meet with the project manager and update the test module + requirements doc.
Muhammad F.	1/27/2022	2	Wire frame designs for search pages and review page
Muhammad F.	1/29/2022	1	Update testing requirements for homepage functionality
Muhammad F.	2/1/2022	1	Design initial support page and modify login page for google sync
Muhammad F.	2/2/2022	1.5	Design trips wireframe page and functionality and meet with proj, manager to go over weekly tasks
Muhammad F.	2/5/2022	1	Meeting with backend to discuss trips page and flow plus table functionality
Muhammad F.	2/6/2022	0.5	Briefly test the website and note adjustments to make on login, CSS, home and page structure
Muhammad F.	2/7/2022	1	Meeting with dev team to discuss search page structure and overall website functionality
Muhammad F.	2/8/2022	2.5	Design review page and connect it to the websites flow
Muhammad F.	2/9/2022	2	Design checkout page and meet with devs to discuss website requirements
Muhammad F.	2/10/2022	0.8	Meet with project manager to discuss final website designs and testing requirements
Muhammad F.	2/12/2022	1.5	Creating testing requirement scenarios for search page functionality
Muhammad F.	2/12/2022	0.9	Meeting with dev team to discuss final website + page functionality and see what can and can't be done within schedule
Muhammad F.	2/15/2022	1	Creating final wireframe designs for submissions and static pages

Muhammad F.	2/17/2022	1	Testing out front end to smooth out kinks for preview demo
Muhammad F.	2/18/2022	3	Test out every page and document all usability, functionality, and CSS minor adjustments to be made
Muhammad F.	2/19/2022	2	Create usability script and record video
Muhammad F.	2/23/2022	0.2	Meet with project manager and go over final video and project manual
Muhammad F.	2/26/2022	0.8	Record final video
Muhammad F.	2/27/2022	2	Fill out the project manual with HCI elements
	Total	41	

5.8 David Gawel

Name	Date	Hours	FULL description of the task in 3 sentences or fewer.
David Gawel	1/5/2022	1	Researched how to connect the backend to the frontend and what technologies to use
David Gawel	1/5/2022	1.5	Design and plan meeting for our implementation
David Gawel	1/6/2022	2	Set up the mysql database server
David Gawel	1/7/2022	1	Set up phpmyadmin on the database server
David Gawel	1/7/2022	2	Set up the rest endpoint on the server, currently does nothing but it works
David Gawel	1/7/2022	1.5	Pushed the code to github and helped others with compatibility
David Gawel	1/7/2022	0.25	talked with Nick on troubles with the database
David Gawel	1/7/2022	2.5	Set up the connection with the database in the rest server
David Gawel	1/8/2022	1.5	Modified nicks code so that it would link up with our database
David Gawel	1/8/2022	0.5	put the new code on the server
David Gawel	1/8/2022	1	tried to fix the cors issue
David Gawel	1/8/2022	1	made the rest endpoint use https
David Gawel	1/8/2022	2	got a valid ssl certificate
David Gawel	1/9/2022	0.5	Recorded the video for the backend
David Gawel	1/10/2022	1	Researched how other groups made their backend and looked at different tech stacks
David Gawel	1/11/2022	0.5	Researched free flight apis available
David Gawel	1/12/2022	0.5	fixed cors issue finally
David Gawel	1/12/2022	2	configured SSL on the database and changed privileges
David Gawel	1/12/2022	1	tried to reset root password on database
David Gawel	1/12/2022	2	messed up sql database and had to reinstall lost data and rebuilt a portion of it
David Gawel	1/13/2022	1	was in a meeting with Mario to discuss the requirements doc
David Gawel	1/13/2022	0.5	was in a meeting with the front end and Mario to discuss the requirements doc
David Gawel	1/13/2022	0.5	Signed up for a API key and overview some of our code
David Gawel	1/18/2022	0.5	Overviewed Nicks code with error generation
David Gawel	1/18/2022	1	added the code for connecting to the flight API
David Gawel	1/19/2022	0.5	Overviewed the requirements doc
David Gawel	1/19/2022	0.5	Hid the api keys in the config file
David Gawel	1/19/2022	1	Restructured project to have better file structure and file names
David Gawel	1/20/2022	0.25	Meeting with Jenna going over the planning doc
David Gawel	1/23/2022	1	Meeting with Mario to discuss requirements Matrix

David Gawel	1/23/2022	1	Doing Bad Dev attack on the requirements Matrix
David Gawel	1/25/2022	1.5	Worked with Mario critiquing and fixing the bad dex attacks
David Gawel	1/26/2022	1	Played around with the Amadeus API seeing how it worked
David Gawel	1/26/2022	1	Thought about how to allow requests for the flight api
David Gawel	1/26/2022	1	Implemented the flight offer endpoint with error handling
David Gawel	1/26/2022	0.5	fixed the gson serialization issue
David Gawel	1/26/2022	0.5	added a basic location lookup endpoint
David Gawel	1/26/2022	0.5	wrote API documentation for flight offers and location lookup
David Gawel	1/26/2022	0.5	fixed the issue with the jar not finding the applications file
David Gawel	1/26/2022	0.5	uploaded the jar onto the server and send out discord message of updates
David Gawel	1/30/2022	0.5	reviewed video and looked at Nick's code
David Gawel	1/31/2022	1	Meeting with Nick and Dan going over the frontend and how it works
David Gawel	2/1/2022	0.5	Updated API and set CORS to *
David Gawel	2/2/2022	0.5	Setup front end environment on my end
David Gawel	2/2/2022	1.5	Researched and set up Oauth login
David Gawel	2/3/2022	1	Implemented a logout button and a bit of a visual representation for user data
David Gawel	2/3/2022	1	Added token authentication in the java backend
David Gawel	2/5/2022	0.25	One on one with Jackson to show progress
David Gawel	2/5/2022	1	modified login to be on the front page and made the returned values better
David Gawel	2/6/2022	2	Attempted and researched how to transfer data between pages
David Gawel	2/6/2022	0.5	Looked more into using local storage and potentially jwt
David Gawel	2/6/2022	1	Converted google login button to a component
David Gawel	2/7/2022	2	Changed token verifier and experimented with authorization
David Gawel	2/7/2022	1	Meeting with Dan, Nick and Muhammad going over problems with the front and plans for the future
David Gawel	2/8/2022	3	Created auth for the users class, started thinking about its design and researched a bit more of google oauth
David Gawel	2/8/2022	1	Tested the tokenverifierclass to make sure that is works
David Gawel	2/9/2022	2	Made the users class more concise and easier to use the get, put, and post mapping. Also added delete
David Gawel	2/9/2022	1	meeting with Dan going over some typescript questions and children in react
David Gawel	2/9/2022	0.75	meeting with Nick and Muhammad going over the plans for the reservations class and bits of the front end
David Gawel	2/10/2022	2	Worked on implementing login button and making it a child of the react router
David Gawel	2/10/2022	1	Helped Mario with getting up to speed on the backend and what he has to do

David Gawel	2/12/2022	1	Tested the User class to make sure it works before I work on the reservations class
David Gawel	2/12/2022	1	Messed around with postman to help with testing with auth codes
David Gawel	2/12/2022	0.25	Meeting with Jackson going over deadlines and what we are going to do over the weekend
David Gawel	2/14/2022	1	Worked on making the new database schema and thought about what to add for the new reservations class
David Gawel	2/14/2022	1	Made the new reservations database schema
David Gawel	2/14/2022	1	Meeting with Muhammad, Dan and Nick going over the plan for this week
David Gawel	2/14/2022	2	Helped Mario with the server code and postman
David Gawel	2/15/2022	2	Worked on making the new post and get methods for the reservations class
David Gawel	2/15/2022	2	Worked on the post mapping of the new reservations controller
David Gawel	2/16/2022	1	Worked on put and delete of the new reservations controller
David Gawel	2/16/2022	0.5	Added documentation and reviewed Marios complaint controller
David Gawel	2/16/2022	0.5	Made api documentation for the reservation controller
David Gawel	2/16/2022	1	Worked on price bug and return booking id in the reservations controller
David Gawel	2/16/2022	0.45	Helped Nick with postman
David Gawel	2/16/2022	0.5	Made backups of the aws ec2 instance
David Gawel	2/16/2022	1.5	updated the website to use post requests for the complaints page
David Gawel	2/16/2022	0.5	reviewed Marios branch and put the fixed code on the server
David Gawel	2/17/2022	0.25	thought about what to include in the preview demo video
David Gawel	2/18/2022	1	Made a script and recorded my preview demo
David Gawel	2/19/2022	1	Figured out how to send emails with java mail
David Gawel	2/20/2022	0.5	Reviewed preview demo and fixed /all security issue in reservations
David Gawel	2/21/2022	1.5	Creating reservations now sends confirmation emails
David Gawel	2/23/2022	1	Worked on making html email customizable
David Gawel	2/23/2022	1	Finished making html emails customizable
David Gawel	2/23/2022	0.5	Attempted to get a production key for the Amadeus api
David Gawel	2/24/2022	1	Reviews page now connects to the database and sends data
David Gawel	2/24/2022	0.25	Updated the email message to be centered and put it on the server
David Gawel	2/27/2022	1.5	Worked on the authentication and email sending portion of the project manual
David Gawel	3/3/2022	0.5	Setup aws budget alerts so that in the future this server does not cost me money
David Gawel	3/4/2022	0.25	Updated dependencies, will have to update the code in the future to remove stuff we do not need
David Gawel	3/4/2022	1.5	Worked on the Security portion of the project manual
David Gawel	3/6/2022	0.75	Overviewed the project manual and added some comments

David Gawel	3/6/2022	1.5	Updated the backend to the final build and made the README nice
	Total	95.7	
Total with required meetings		108.45	

5.9 Dan McCarthy

Name	Date	Hours	FULL description of the task in 3 sentences or fewer.
Dan McCarthy	1/3/2022	2.5	Set up wordpress and a domain name for the project. Learning more about wordpress as a CMS for the project.
Dan McCarthy	1/5/2022	1	Writing up a testing document to cover areas we will need to test and provide an overview of what we are checking.
Dan McCarthy	1/5/2022	1.5	Design and plan meeting for our implementation (UI frontend & backend planning)
Dan McCarthy	1/5/2022	2.5	Working on building base frontend project in React, made repository & added everyone to repository
Dan McCarthy	1/6/2022	2.5	Developing the homepage UI in react, got the user inputs added and styled to match the UI design
Dan McCarthy	1/7/2022	4	Updating frontend for feedback from meeting, configuring button interactivity and tracking values with React state.
Dan McCarthy	1/7/2022	1	Adding alert system in react to display error/info/or success messages at the top of the page in React.
Dan McCarthy	1/7/2022	2	Connecting the front end to access flights API data, display the data on the home page, and making selection interactive.
Dan McCarthy	1/8/2022	2.5	Creating a Flight list component in React to display flights retrieved from the database, connecting to the home page. Also formatting dates and rendering each data element as needed.
Dan McCarthy	1/8/2022	1	Polishing frontend for concept demo, updating flight list to be selectable, and publishing all changes to repository and live site.
Dan McCarthy	1/8/2022	1	Recording video for concept demo and sharing with Presentation manager
Dan McCarthy	1/9/2022	2	Creating test cases for concept demo frontend and validating they are working. Found and fixed defect with flight duration calculation/label.
Dan McCarthy	1/12/2022	1.5	Investigating JSON.parse issue with endpoint from backend for API calls. It seems to be related to CORS issues we were having previously. Update site to improve debugging for backend.
Dan McCarthy	1/13/2022	1.5	Adding handling of all reservations API endpoints and connecting it to frontend for submit button. Still in progress.
Dan McCarthy	1/15/2022	2	Working on POST API endpoint for reservation submission and handling of reservation IDs. Had issues with post requests due to 500 internal server errors.
Dan McCarthy	1/16/2022	2.75	Issues with POST reservation endpoint fixed and configured to register with all attributes user selected. Updating selection system to be able to send price to endpoint as well.
Dan McCarthy	1/18/2022	2.25	Improved handling of flight selection in React so HomePage has a copy of the selected flight data. Submit button now submits all filter/selected flight data for reservation backend API endpoint.

Dan McCarthy	1/19/2022	2	Updating repository and website with current changes. Updating UI/logo with suggestions given to freshen up the look for the next demo. (Troubleshooting Webpack/TypeScript issues with image asset importing)
Dan McCarthy	1/20/2022	1	Polishing up image banners & updating to look consistent on Chrome/Firefox. Deployed changes to the website.
Dan McCarthy	1/23/2022	3	Starting to update the homepage to display differently for each mode round trip/oneway/multicity etc. Updating frontend to contain and use a set of US airports for autocomplete purposes.
Dan McCarthy	1/25/2022	1	Updated website to not display data/filter warning on load. Fixed date input on Safari not working. Multi-city is now automatically disabled on One Way mode.
Dan McCarthy	1/26/2022	3.5	Developing Multi-city add-able and remove-able flight selections for multiple flight registration flow.
Dan McCarthy	1/27/2022	1	Implemented add and remove button functionality for managing airport selection rows in multi-city flow.
Dan McCarthy	1/28/2022	3	Fixing handling of add/remove for React state. Adding UUIDs to track dynamically created JSX elements. Updated homepage to handle OneWay/RoundTrip/MultiCity has 3 distinct modes.
Dan McCarthy	1/29/2022	1.5	Configured React project with React-router allowing for a multi-page set up and to locally link between React pages via button. Added the initial page component that will become Login Page and made it accessible from the HomePage login button.
Dan McCarthy	1/29/2022	3	Creating a react component for airport search & dropdown list. Added handling of calling backend locations endpoint to pull in locations for queries. Handling of loading data, displaying it as dropdown, and UI handling of events.
Dan McCarthy	1/30/2022	2.5	Making airport search dropdown items clickable & adding airport selection handling to UI. Fixing issues with dropdown expand/contract when focus moves to child element (e.g. clicking dropdown list item)
Dan McCarthy	1/31/2022	1	Meeting with Nick and David to get them introduced to the frontend, react, and our homepage code base. Additionally went over the status of the project and what is left to be done.
Dan McCarthy	2/2/2022	3	Updated homepage to move banner to bottom as per suggestion. Looking into resolving issues where banner images are no longer loading. (It's due to relative urls and react-router)
Dan McCarthy	2/2/2022	2	Resolving issue where dropdown menu had race condition causing it to close instantly when clicking a list item but without the click event firing.
Dan McCarthy	2/2/2022	1	Fixing CSS issues where elements on the home page were moving off-center when changing states, i.e. going from unselected to selected.
Dan McCarthy	2/3/2022	1.5	Helping Nick with ToastMessage component issues. Updated toast message to be closed manually by the user and fixed issue where the toast would not reappear after being closed the first time.
Dan McCarthy	2/4/2022	4	Working on getting support for the /api/flightOffers/ endpoint from frontend, defining types to represent the data structure that is returned from the API to be used via TypeScript.
Dan McCarthy	2/5/2022	0.25	Updating Number of Adult Travelers and Number of Child Travelers inputs to update react

			HomePage state when values are changed, and pull in value from state to display.
Dan McCarthy	2/5/2022	0.25	Fixing issue where date for departure and arrival were not setting state for each input, but rather both inputs were overwriting the return flight date.
Dan McCarthy	2/5/2022	3	Finishing typing and api endpoint retrieval for /api/flightOffers/ so that we can pull in the data. Updating HomePage to handle new flightOffers data instead of flightsData directly.
Dan McCarthy	2/5/2022		Updating FlightList component to be able to handle showing multi-stop flights as per the flightOffer itineraries data.
Dan McCarthy	2/6/2022	2.5	Helping Nick with TypeScript difficulties/complicated type and asynchronous handling issues.
Dan McCarthy	2/6/2022	4	Working on updating flightlist to load in and display flightOffer data. Updating HomePage to use flightOffer data instead of flight data (different endpoint). Working on multi-city multiple-flight selection
Dan McCarthy	2/7/2022	5	Working on a multi-city workflow, handling changes to filters in real time so we can go back and reselect pages. Updates so we can get airline names from IATA code.
Dan McCarthy	2/9/2022	2.5	Had a meeting with David regarding how we can store our login info in a parent component to the pages in React router and pass them down. Made a new branch with an example, pushed it public, and shared it with David to look at after the meeting.
Dan McCarthy	2/9/2022	2	Updating flight count handling for how many to select to use prior simpler system since it's compatible with design after talking with Muhammad.
Dan McCarthy	2/10/2022	3	Continued working on configuring the handling of multiple page flight selection in React state & connecting to Flight List for multiple-selections/starting a second/third, etc.
Dan McCarthy	2/11/2022	4	Refinement to handling multiple flight searches, switching between state and resetting flight search state.
Dan McCarthy	2/13/2022	3.5	Finalizing FlightList component, fixing bugs with representation of data for flights, handling segment/itinerary data. Parsing of dates and reformatting for UX purposes, CSS enhancements for view-ability.
Dan McCarthy	2/14/2022	6	Rebuilt Multi-city section to use AirportSelector components, become stateful with the filters, and bubble up those filter selections to the HomePage component. Updating HomePage state to fully handle tracking multiple-flight selections and which flight is currently being selected/which is next.
Dan McCarthy	2/15/2022	10.5	Built home page flow to dynamically handle Next/Submit buttons depending on current flight selections, what is next, and getting them refetching when moving onto next flight. Handling of resetting search state, asynchronous handling of loading icons when waiting for data, among other UX and flow improvements.
Dan McCarthy	2/16/2022	0.75	Meeting with Nick to organize splitting up of Checkout page, how to implement certain things in React & merging in pull requests & fixing merge conflicts.
Dan McCarthy	2/16/2022	8	Merging changes from all developer branches, implementing reservation posting for finalized flightOffer selections, sharing the data with Checkout page, and helping & organizing with Nick. Handling of asynchronous events and allowing transition from HomePage to checkout page, and fixing behavioral issues with different mode states/hand off to next page.

	Total	122.75	

5.10 Jawan Luangnikone Davis

Name	Date	Hours	FULL description of the task in 3 sentences or fewer.
Jawan L Davis	1/4/2022	1.25	Brainstorm use-case, design patterns for DFD, and sequences. Created a rough draft of a use-case diagram.
Jawan L Davis	1/5/2022	2.5	Create UML Design Rough Draft Documents for System Architecture and Flight DFD. Created UML Customer DFD.
Jawan L Davis	1/5/2022	2.75	Create proto-design documents for web server and UI.
Jawan L Davis	1/5/2022	1.5	Meeting with front-end and back-end developers for design and planning of the concept demo.
Jawan L Davis	1/7/2022	1	Updated the Customer DFD and Use-Case Diagram.
Jawan L Davis	1/8/2022	0.5	Made script and recorded my portion of the presentation.
Jawan L Davis	1/11/2022	1	Created the Level 0 design document rough draft
Jawan L Davis	1/12/2022	1.25	Created first draft UML Dataflow Diagrams for Manage Reservation, Manage Searches, and Manage Flights.
Jawan L Davis	1/13/2022	1	Discussed technical requirements with Requirements Manager
Jawan L Davis	1/13/2022	0.75	Drafted Finite State Machine in notebook. Thinking about possible states.
Jawan L Davis	1/14/2022	1	Drafted updated Level 0 diagram, Manage Error DFD, and Manage Account DFD.
Jawan L Davis	1/14/2022	0.75	Brainstorm potential risks, the risk's impact, and the risk's potential.
Jawan L Davis	1/18/2022	1.5	Create Manage Accounts DFD and Manage Errors DFD. Updated Level 0 Diagram and Manage Flights DFD.
Jawan L Davis	1/19/2022	2	Created the rough draft of the Finite State Machine.
Jawan L Davis	1/24/2022	1	Updating and revising work on DFD and Level 0 diagram
Jawan L Davis	1/24/2022	0.5	Added Manage Locations DFD and inserted it into Level 0 diagram
Jawan L Davis	1/26/2022	1.25	Started working on the oral presentation of the design in the RDP demo. Written a brief explanation for the Level 0 diagram.
Jawan L Davis	1/26/2022	1	Written an explanation for 3 Major Modules: Manage Searches, Manage Flights, and Manage Location
Jawan L Davis	1/27/2022	1.25	Written an explanation for 3 Major Modules: Manage Accounts, Manage Reservations, and Manage Errors
Jawan L Davis	1/28/2022	2.25	Ruminated on visual aid and decided on Prezi. Created Prezi to be a visual aid for my RDP Design presentation.
Jawan L Davis	1/29/2022	0.5	Recorded my portion of the RDP presentation
Jawan L Davis	2/16/2022	0.75	Updated Level 0 Diagram and Finite State Machine to include Manage Support.

			Created Manage Support DFD Rough Draft
Jawan L Davis	2/17/2022	0.15	Written script and timed oral presentation for the Design portion of the Preview Demo.
Jawan L Davis	2/18/2022	0.25	Recorded my portion of the Preview presentation
Jawan L Davis	2/22/2022	1	Written content for Level 0 diagram, Use-case, and DFD for the project manual
Jawan L Davis	2/25/2022	0.25	Inserted content (both diagrams and written text) for MAIN project manual
Jawan L Davis	3/2/2022	0.35	Written content for Use-Case and Finite State Machine. Inserted content into MAIN project manual
	Total	29.25	