TripGuider: A Flight Management System

CIS 375 – Final Project Report

17th December 2017

Akshita Joshi
Department of Computer and Information Science
University of Michigan - Dearborn
Dearborn, MI
akshitaj@umich.edu

Luke Patton
Department of Computer and Information Science
University of Michigan - Dearborn
Dearborn, MI
pattonl@umich.edu

Rakshit Bhatt
Department of Computer and Information Science
University of Michigan - Dearborn
Dearborn, MI
rjbhatt@umich.edu

Alexander Reno
Department of Computer and Information Science
University of Michigan - Dearborn
Dearborn, MI
areno@umich.edu

Kenneth Rogale
Department of Computer and Information Science
University of Michigan - Dearborn
Dearborn, MI
krogale@umich.edu

This document is a detailed description of our term project for CIS 375 – Software Engineering (I), carried out under the guidance of our instructor, Dr. Marouane Kessentini.

I. INTRODUCTION

Flight booking modules were first introduced in the late 1950s as individual modules to oversee flight inventory, maintain flight schedules, seat assignments and aircraft loading. Today's flight reservation applications are an all-inclusive collection of products to provide a system that assists with a variety of airline management tasks and service customer needs from the time of initial reservation through completion of the flight.

One of the most common modes of travel is traveling by air. Customers who wish to travel by air nowadays have a wide variety of airlines and a range of timings to choose from. Nowadays, competition is so fierce between airlines that there are lot of discounts and a lot of luxuries given to customers that will give an edge to that particular airline.

The World Wide Web has become tremendously popular over the last four years, and currently most of the airlines have made provision for online reservation of their flights. The Internet has become a major resource for people looking for making reservations online without the hassle of meeting travel agents. Our project intends to serve these purposes. It intends to check the available databases for our chosen airlines, and returns a string of results, which can help potential passengers in their travel plans.

The objective of this project is to create a flight booking system where a traveler can request all flight information as per their preferred journey dates. They can get information regarding time, cost, available classes, etc., all at the same time and place. The system enables passengers to customize their flight path based on the number of connections, and also provides them with the option to choose a direct flight if available. Our customer login portal offers registered users the

convenience to access their flight itinerary and view any current delays. The system displays all the available airlines, schedules and prices. This system would help our airline partners to better serve their customers by catering to their needs. Our website uses a database to hold this information as well as the latest pricing and availability information for the airlines.

II. PROBLEM STATEMENT

Our project has the following goals:

A. *Primary Goal*

Our primary goal is to allow customers to book their itinerary conveniently, without having to visit a travel agent or an inefficient travel booking website to get it done. As it exists today, airline reservation systems are inefficient and provide a poor user experience for the following main reasons; making modifications are complicated, the flight search and query process can be time consuming, and websites can be error prone and unsecured. TripGuider hopes to alleviate these nuisances and provide customers with a safe and easy way to book their travel.

B. *Secondary Goals*

Our secondary goals include the following:

a. To allow passengers to modify their itinerary anytime.
b. To provide potential passengers with the option to filter through available options to narrow down the best available trip.
c. To allow passengers to view current delays at their airport and the current weather via the FlightAware API.
d. To give potential users access to past user testimonials.

III. METHODOLOGY

A. *Implementation Overview*

Originally, we planned to create a responsive ASP.Net MVC Bootstrap website using a straightforward C# and CSHTML implementation to create interactive web layouts integrated with the Razor Framework for ASP.Net. We were able to successfully create the Front-End layout using this approach; however, we ran into issues integrating our Front-End with a database. We started out attempting to add a database object directly to the source code. In C#, you can add a database object directly to your project and interact with it similar to if you created a connection to a mySQL/Oracle instance. When we tried to push our source code through GitHub to each member, the database object was considered local AppData and could not be uploaded to the common repository. When this did not work, we tried to use AWS to host our database. You can create a remote database instance that has an endpoint available to all public connections via port and address token. We were able to connect on one computer to the RDS instance using the AWS plugin for Visual Studio, however we still couldn't access the data via this connection. Furthermore, if more than one person needed to connect we had to add a SQL connection in the Web.config file for our C# project. We were unable to find where to place the config file due to Visual Studio creating multiple instances of config files throughout the architecture of a MVC5 web application. After these attempts, we decided to take on a new approach and take the individual CS.HTML files and convert them to PHP and HTML. We then hosted these files on an actual web server using a PHP backend for the database connections and data management.

The website is live and available via the following link: TripGuider Website.

B. *Technical Overview*

a. *Back-End*

i. *PHP*

PHP is a server-side scripting language created in 1994, it is designed for web development but also used as a general-purpose programming language. PHP was used as our primary back-end engine for the database implementation.

b. *Front-End*

i. *HTML*

Hypertext Markup Language (HTML) created in 1990, is the standard markup language for creating web pages and web applications. HTML formed the barebones framework for our Front-End implementation.

ii. CSS

Cascading Style Sheets (CSS), created in 1996, is a style sheet language used for describing the presentation of a document written in a markup language. CSS was very useful in helping us easily control the layout of our Front-End implementation and was an integral tool in helping shape our website design.

iii. JavaScript

JavaScript is a lightweight object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well. JavaScript served as the focal point in the implementation of many menu-driven user interface options for our project such as the drop down menu(s), graphics animations, and interactive buttons throughout our website.

C. Tools

a. Filezilla FTP Server

FileZilla is a free software, cross-platform FTP application, consisting of FileZilla Client and FileZilla Server. We used the Filezilla FTP Server to connect to the web server files and allowed us to directly interact and update our system files. It also served as a form of source control for our project similar to GitHub.

b. Atom/Notepad++ Text Editor

Atom is a free and open-source text and source code editor for macOS, Linux, and Microsoft Windows with support for plug-ins written in Node.js, and embedded Git Control, developed by GitHub. We used Atom and Notepad++ to edit our PHP files and commit them to Filezilla to update our website files.

c. PhpMyAdmin

It is a free and open source administration tool for MySQL and MariaDB. We used PHP MyAdmin as our main web server central client to add databases and allow connections.

d. FlightAware API

FlightAware is a global aviation software and data services company based in Houston, Texas. The company operates a website and mobile application which offers free flight tracking of both private and commercial aircraft in the United States, Canada, Australia, and New Zealand. We used their open source API to embed a live airport view tool for customers to see current delays and temperature conditions at their origin and departure airports.

IV. RESULTS

At the end of this document is a table consisting of our test cases, with an explanation of our test case scenarios. Please refer to *Table 1* at the end of this document.

While executing our test cases (as mentioned in *Table 1* in the end of this document), we found that all of our tests are black box tests. White box testing does not really work the best in PHP since most of the data is “in-line” versus making classes, structures and methods for something like this. Also, all of the unit tests were performed as edits were made to the page. One example to convey this would be PHP testing for the $sql strings, as a result of which our customer purchase database has around 100 entries in it.

To improve existing code smells, we can aim to use inheritance-type structure for common classes. Also, for better maintainability of our code, we can utilize the factory design pattern to organize and structure the code better so it is maintainable for future development.

V. THREATS TO VALIDITY

We observed some expedient limitations with our project, most of which were because of the time constraint that we had for this project.

The following is a list of our limitations with TripGuider:

A. Inability in implementing a SSL Certificate.

We were unable to implement a SSL Certificate to encrypt details like a passenger's credit card information. We understand that it is very important for passengers to be able to securely enter their financial information for making a purchase for their selected itinerary. However, with limited time for this project, we were unable to implement a SSL platform, thereby not being able to facilitate secure payments successfully.

B. The system is not optimized on mobile devices.

TripGuider has not been developed for use on mobile platforms. We lack a mobile application for our system, which in turn fails to provide further convenience for passengers to make a booking using their mobile devices.

C. The system lacks an organized account page.

We were unable to implement an organized account page for customers, however customers are still able to view their purchased tickets via entering their last name and phone number.

D. The existing FlightAware API lacks additional features.

We embedded the FlightAware API into our 'Search Flights' module to offer customers a view of the current weather at their origin and destination airports. The FlightAware API however lacks additional features such as the ability for much smaller airports or a separate module to book flights.

VI. CONCLUSION

TripGuider was developed to be used as a bug free, easy to use flight booking tool for consumers within the United States.

Originally, we planned to create a ASP.Net MVC website using C# and CSHTML, however when we ran into obstacles with integrating our database we decided to pivot and take a different approach. We were instead able to create a highly scalable and interactive system using a Full-Stack combination of PHP, HTML, and JavaScript. A snapshot of some of the features that set TripGuider apart from its competitors are it's highly robust and intuitive layout making navigating the site a breeze for potential customers, and the inclusion of an external API to track forecast conditions of airports all around the US.

While executing our test cases, we found that all of our tests are black box tests. White box testing does not really work the best in PHP since most of the data is "in-line" versus making classes, structures and methods for something like this. To improve existing code smells we can aim to use inheritance-type structure for common classes. Also, for better maintainability of our code, we can utilize the factory design pattern to organize and structure the code better so it is maintainable for future development.

TripGuider is far from a finished product, although we were able to implement the core features and functionalities, due to time constraints we were unable to implement some features that would help boost the customer experience. In the future, we plan to include a login module for customers to track their flight purchases and reward them via points for frequent purchases. We also plan to implement incentives for new customers to register for our rewards system via a separate module to track these incentives. Additionally, we plan to include the option to filter flights via the number of desired connections, and future development to partner with even more airline carriers to provide the most options to prospective customers.

VII. ACKNOWLEDGEMENT

A great debt of our gratitude goes towards our instructor, Dr. Marouane Kessentini, for we were highly comfortable in working with this project right since the beginning because of his excellent method of instruction in class, and his guidance throughout this project.

With due respect, we are also grateful to one another for having been so considerate towards each other's opinions and ideas, and hence for having been able to work on this assignment together whole-heartedly.

VIII. REFERENCES

1. "Flight Status API / Flight Tracking API / FlightAware API ✈ Commercial Services ✈ FlightAware." *FlightAware - Flight Tracker / Flight Status / Flight Tracking.* Web. http://flightaware.com/commercial/flightxml/

2. "JavaScript | MDN." *MDN Web Docs.* Web. http://developer.mozilla.org/en-US/docs/Web/JavaScript.

3. "PHP: What is PHP? - Manual ." *PHP: Hypertext Preprocessor.*Web. http://php.net/manual/en/intro-whatis.php.

4. "Air ticket reservation system presentation ." *Share and Discover Knowledge on LinkedIn SlideShare.* Web. http://www.slideshare.net/smitpatel10192/air-ticket-reservation-system-presentation

5. "CSS Introduction." *W3Schools Online Web Tutorials.* Web. http://www.w3schools.com/css/css_intro.asp

6. "ASP.NET MVC Reference." *Learn to Develop with Microsoft Developer Network | MSDN.* Web. http://msdn.microsoft.com/en-us/library/dd566232(v=vs.100).aspx

7. "PhpDocumentor." *PhpDocumentor.* Web. http://www.phpdoc.org/

Table 1: Test Cases and Results

Test Scenario	Test Case	Test Steps	Test Data	Expected Result	Actual Results
Website Link: About	Check if website links work.	Load up webpage, click on links at the top menu bar	N/A: Mouse only required. UI functionality.	Page navigates to the "About" page and displays all relevant data.	Page displays all data using the correct css format.
Website Link: Home	Check if website links work.	Load up webpage, click on TripGuider name at the top menu bar	N/A: Mouse only required. UI functionality.	Page navigates to the "Home" page and displays all relevant data.	Page displays all data using the correct css format.
Website Link: Testimonials	Check if website links work.	Load up webpage, click on links at the top menu bar	N/A: Mouse only required. UI functionality	Page navigates to the "Testimonials" page and displays all relevant data.	Page displays all data using the correct css format.
Website Link: Search Flights	Check if website links work.	Load up webpage, click on links at the top menu bar	N/A: Mouse only required. UI functionality	Page navigates to the "Search Flights" page and displays all relevant data.	Page displays all data using the correct css format.
Website Link: FAQ	Check if website links work.	Load up webpage, click on links at the top menu bar	N/A: Mouse only required. UI functionality	Page navigates to the "FAQ" page and displays all relevant data.	Page displays all data using the correct css format.
Website Link: My Tickets	Check if website links work.	Load up webpage, click on links at the top menu bar	N/A: Mouse only required. UI functionality	Page navigates to the "My Tickets" page and displays all relevant data.	Page displays all data using the correct css format.
Check Blank Data for First Name	See if form will accept blank data in "Search Flight forms"	Fill in all information besides "First Name"	First Name: -"Blank" Last Name: -"Doe" Phone #: -"19891234567" Date: -default date # of Adults -1 # of Children -0 Seat Tier -Economy	Error message displays on the box that doesn't have data, IE "Is Blank"	Page displays error message on blank First Name

Check Blank Data for Phone #	See if form will accept blank data in “Search Flight forms”	Fill in all information besides “Phone Number”	First Name: -“John” Last Name: -”Doe” Phone #: -”Blank” Date: -default date # of Adults -1 # of Children -0 Seat Tier -Economy	Error message displays on the box that doesn’t have data, IE “Is Blank”	Page displays error message on blank Phone Number
Check Blank Data for Last Name	See if form will accept blank data in “Search Flight forms”	Fill in all information besides “Last Name”	First Name: -“John” Last Name: -”Blank” Phone #: -”19891234567” Date: -default date # of Adults -1 # of Children -0 Seat Tier -Economy	Error message displays on the box that doesn’t have data, IE “Is Blank”	Page displays error message on blank Last Name
Check Blank Data for # of Adult Tickets	See if form will accept blank data in “Search Flight forms”	Fill in all information besides “Number of Adults”	First Name: -“John” Last Name: -”Doe” Phone #: -”19891234567” Date: -default date # of Adults -blank # of Children -0 Seat Tier -Economy	Error message displays on the box that doesn’t have data, IE “Is Blank”	Page displays error message on Blank Number of Adults
Check Blank Data for # of Child Tickets	See if form will accept blank data in “Search Flight forms”	Fill in all information besides “Number of Child”	First Name: -“John” Last Name: -”Doe” Phone #: -”19891234567” Date: -default date # of Adults -1 # of Children -”Blank” Seat Tier -Economy	Error message displays on the box that doesn’t have data, IE “Is Blank”	Page displays error message on Blank Number of Children

Test database queries for view flights	See if DB returns correct information and is displayed nicely.	Fill in all information on the previous page and press search	First Name: -“John” Last Name: -”Doe” Phone #: -”19891234567” Date: -12-5-2017 # of Adults -1 # of Children -”1” Seat Tier -Economy	Page displays 1 available flight, from PHL to JFK, with correct pricing information of $230 dollars, flight number, carrier, flight time and duration.	Page displays the correct information from the database and passes the query test for information that exists.
Test database queries for view flights	See if DB returns error message for no available flights found given the input parameters.	Fill in all information on the previous page and press search	First Name: -“John” Last Name: -”Doe” Phone #: -”19891234567” Date: -12-31-2017 # of Adults -1 # of Children -”1” Seat Tier -Economy	Page displays the “Sorry” message for no flights available on the given data of 12-31-17.	Page moves to viewflights and displays all information correctly. A “Sorry, there are no flights available on that date” is displayed instead of the table of information from the DB.
Website Link: View Flights	Check if “search” works on search flights, and brings to view flights.	Fill in all information and click search.	First Name: -“John” Last Name: -”Doe” Phone #: -”19891234567” Date: -default date # of Adults -1 # of Children -”1” Seat Tier -Economy	View flights loads with selected flight data” OR moves to the page and displays no results found.	Search button does move to view flights, and displays the information relevant to entered information.
Website Link: buytickets.php	Check if “Book!” grabs the data and sends it to the next page where it displays it properly for purchasing.	Select “Book” on a flight option.	N/A. Just using an UI element to see if the information stored from the previous test case (see View Flights) is displayed correctly on the next linked page.	Buy Tickets loads with selected flight data and can enter information into text boxes.	Information correctly passes into the newly loaded Buy Tickets page. User can interact with additional UI elements on screen.

Website Link: Tickets.php	Check if "Purchase Now!" grabs the data and sends it to the next page where it displays it properly as a record of purchase.	Select "Purchase Now!" at the bottom of the CC info & payment.	Passenger Name, Ticket#, Flight#, Departure and Arrival Code, Seat type, Date and Price all need to be sent. Data is default to First Name: -"John" Last Name: -"Doe" Ticket#: -"76663" Flight#: -"42715" Depart. Code: -"PHX" Arrival. Code: -"DFW" Seat Tier -Economy Date: -"2017-12-09" Price: -"$200"	Tickets.php loads correctly, uses the information from last page and queries purchase DB to display all relevant information.	Information correctly passes into the new page, and all data is the same as the input test data seen to the left. Display is correct and orderly, all themes and UI elements are interactable and correct.
Website Link: Tickets.php	Check if "Cancel" brings user back to search flights page.	Select "Cancel" at the bottom of the CC info & payment.	N/A. Just click a button to check the UI interaction.	Search flights correctly displays after the button is clicked.	User is taken back to Search flights to start a new flight search from scratch. All UI elements work correctly and themes are correct.
My Tickets functionality. 0 Tickets	Check if "My Flights" will find available tickets given last name and phone number of purchase.	Select "Find Flights" at the bottom of the text box for Phone Number.	Last Name: -"Patton" Phone Number: -"1234500000"	No tickets should be displayed and an error message saying none were found should occur, since this phone doesn't exist in the database under last name "Patton".	No tickets display, error message is displayed, website links to alltickets.php and the Search again link works.
My Tickets functionality. 1 Tickets	Check if "My Flights" will find available tickets given last name and phone number of purchase.	Select "Find Flights" at the bottom of the text box for Phone Number	Last Name: -"Patton" Phone Number: -"1234567890"	1 ticket should be displayed, since 1 tricket is stored under "Patton" using this phone number.	1 ticket displays in the correct format of the form table and all information correctly pulled from the database.

My Tickets functionality. 2+ Tickets	Check if "My Flights" will find available tickets given last name and phone number of purchase.	Select "Find Flights" at the bottom of the text box for Phone Number	Last Name: -"Doe" Phone Number: -"2147483647"	Many tickets should be displayed since this was the phone# and last name used for error checking.	20+ tickets display, which match the amount of entries in the database for this last name and phone# combination. All information is correct after a long review and is displayed correctly in a scrolling table.