COMP3011 Web Services and Web Data
Coursework 2
Aggregator Implementation Report

Kalina Nikolova

Table of Contents

Introduction

The purpose of this report is to describe the development and implementation details of a flight aggregator client application. It is the second part of an assignment aimed at creating a flight booking and management system in collaboration with 12 other students. In the first part, a design for the solution was carefully crafted so that each team member could use it as a reference and implement a web service or a client during the second phase of the project. Our system is called SkySavers and it consists of 5 airlines, 4 payment providers and 4 aggregators. The author of this report was tasked with the creation of one of the aggregators.

Requirements

This section goes through the functional and non-functional requirements set for the development of the console application.

Functional

- The app must allow users to search for a flight by entering departure and arrival destination as well as preferred date of travel.
- Users should be able to view a list of available flights matching their preferences and sorted by price (lowest price on top).
- Users should be able to book a flight by entering their personal details such as names, email and phone number.
- Users should be provided with two alternative payment options - card payment and payment by email and password.
- Users should be able to view an existing booking.
- Users should be able to change the name associated with a booking.
- Users should be able to cancel a booking.

Non-functional

- The app must be fast and responsive.
- Appropriate error handling must be in place and clear error messages provided to the user when an issue occurs.
- The app must be easy to use and intuitive, providing clear instructions and prompts for the user.

Design

The design of the aggregator client was focused on creating an intuitive user experience and ensuring that the final solution is robust and efficient.

User Interface

First of all, a simple user interface had to be created. Following the specification document prepared during the initial phase of this project, it had to provide the user with the ability to search, book, pay and manage flights. In order to make the process of inputting data and viewing outputs easy for the users, a menu containing appropriate options for each available feature was constructed.

Program flow

All steps and actions that the program takes in response to user input had to be presented in a clear and logical way. When initially started, the program greets the user and asks them to choose an action amongst booking a flight, managing an existing booking and exiting. The first two options were designed to follow separate flows as they offer distinct functionality.

Booking a flight allows either a partial completion, in which case the user reserves their seat for a chosen journey without paying, or a full completion, where a payment is required to confirm the booking. In either case, users are presented with a summary containing important details such as their reference number which can later be utilised for the booking management features.

Users can benefit from the booking management features that the client provides given that they have made a reservation and retrieved a reference number. Cancelling a flight, changing the name associated with a booking, viewing an existing booking and paying for unconfirmed reservation are the options provided by the management menu.

Payment can either be part of the booking process or completed via the management menu. Overall, both options follow the same flow, however users are required to provide less identification information in the second case as they would have already supplied most of the needed data while reserving their journey. The application allows users to pay in two ways - by card and via a pre-existing payment profile by supplying email and password instead.

Regardless of the chosen flow, the users are always presented with either a meaningful confirmation message or a summary of the results of their actions.

Error handling

Making sure that users understand in case something goes wrong and how to fix it, as well as gracefully handling occurring errors is essential for console applications. The client has been designed to output clear messages whenever a functionality does not exhibit expected behaviour so that users can easily rectify any issues. What is more, an option to end the session is present at all times which again limits the opportunities for error-prone unexpected

behaviour. In addition to this, all user input provided to the web services is firstly validated on the client side through pattern matching with expected data format.

Implementation

Programming language and external libraries

The client has been implemented with the Python programming language. Some of the required features are provided through external libraries. All described components function by sending requests to and receiving responses from the web services - airlines and payment providers. To achieve this, a library called `requests` [1] was used. It allows seamless communication with all services on the condition that appropriate request parameters, http method and format of data sent are specified.

A useful tool called `re` [2] was utilised for pattern matching the supplied user input. As described in the specification document created during part one of the project, we decided to adopt regular expressions to ensure that the provided data conforms to the expected format. In the implementation of the client described in this report, the feature `fullmatch` of the module `re` served as a method for verifying the correctness of user data.

It was considered important to have a document containing flight details being generated at the end of a booking process. In this way, the user is not required to take a note of their booking reference number since a pdf file with the necessary information is automatically saved to their device. This was achieved by adopting a module for pdf generation - `pdfgen` [3].

Key features

The primary functionalities of the system revolved around reserving and handling flight bookings. Both can be accessed through a main menu which is presented to the users when interaction with the client begins. Users pick an option by supplying a number associated with it and pressing enter. Based on the input an appropriate function that handles the desired action is triggered.

Book a flight

Flight booking is constructed as a three-step process. Firstly, users search for a journey by supplying departure and arrival destinations as well as their preferred date of travel. These details are added to a dictionary and later used as one of the request's parameters. All available flights are retrieved by iterating through a list of airline urls and sending a `GET` request with user specified data to the appropriate endpoint `(/searchFlights)` of each one. In case that no flights are present, users get an indicative message and prompt to try again with different details. On a

successful retrieval, all flights are added to a list, sorted by price and then displayed to the users. A flight can be chosen by entering its ID.

Choosing a number associated with a flight, initiates the next phase - booking. During this step users enter their names, phone number and email. This time a `POST` request containing the user data, as well as the chosen flight id and date of travel, is sent only to the airline offering the chosen journey. Any occurring errors are handled and displayed to the user by also providing clear details on ways to resolve them. A successful response from an airline indicates that the booking has been saved. After that, a summary is displayed to the user and saved to a pdf file.

The final step of the process involves completing a payment for the chosen journey. Users can choose to do that later via the booking management menu or finalise the reservation in the current session. Completing a payment is a two-step process on its own - users are asked to pick a payment provider and then payment option. Similarly to the booking process, appropriate `POST` requests are sent to a user-chosen payment provider. Depending on the chosen payment method, users need to supply either credit card details such as number, cvv and expiry date or an email and password. Once payment is confirmed by the payment provider, an additional `POST` request containing a booking reference and valid payment id follows to the flight provider. This step concludes the process and guarantees that a booking has occurred.

Manage booking

Opening the booking management menu results in a prompt for booking reference number, last name of the user and name of the airline flight provider. The details are verified by sending a `GET` request to the appropriate airline which results in a success only if the user supplied data exists. Once verified, users are provided with the option to change their name, cancel a flight, pay (if they have not already) and view a summary of existing booking.

Changing a name is achieved through a `PUT` request containing new first and last names supplied by the user. Choosing to cancel triggers a `DELETE` request to the specified airline's database. Picking to view an existing booking does not require an additional request as the booking details are retrieved at the start of the management session and displayed to the user if necessary. Finally, the pay feature results in the same payment process described in the 'Book a flight' section above. For each option, users are presented with either a success message or error code and reason for failing to satisfy their request.

Results

All requirements were met and the console application offers the functionality defined as part of the specification document created in phase one of the project. The entire system has been collectively tested with the available solutions of other team members. This section explores some of the strengths and limitations of the described solution.

Strengths

The resulting client is a user-friendly and well-functioning application. One of its main strengths is that a confirmation step is included before taking an action (i.e sending a request to a web client). In this way, the user is offered an opportunity to review the details and catch any mistakes before proceeding. This helps for preventing accidental actions which can save time and reduce the risk of unintentional errors occurring. What is more, by integrating multiple flight and payment providers, the client offers users a wider variety of options to consider and choose from.

Limitations

A limitation of the client is that all urls of web services are hard-coded. A better approach would have been to maintain them in a deployed database instance so that they are automatically retrieved, and the working of the client is not affected in case any changes occur. In addition to that, it was discovered that the refund feature has not been well defined in the specification document. Even though such an endpoint has been created by the payment providers, it could not be adapted as a fully functioning feature due to a lack of integration with the airline services.

Conclusion

In conclusion, the developed solution has been implemented with all required functionality and offers users a range of useful features. Additionally, the console client provides informative messages and robust error handling so that users are made aware of any issues that may arise. Overall, the application offers a comprehensive and user-friendly solution for flight booking and management, making it a valuable option for users seeking streamlined and efficient travel booking experience.

References

1. Reitz, K. 2019. Requests: HTTP for Humans™ — Requests 2.27.1 documentation. *requests.readthedocs.io*. [Online]. [Accessed 12 May 2023]. Available from: https://requests.readthedocs.io/en/latest/.
2. Python 2009. re — Regular expression operations — Python 3.7.2 documentation. *Python.org*. [Online]. [Accessed 12 May 2023]. Available from: https://docs.python.org/3/library/re.html.
3. Driscoll, M. 2018. ReportLab Docs. *docs.reportlab.com*. [Online]. [Accessed 12 May 2023]. Available from: https://docs.reportlab.com/.