# Project Proposal – UTI

**By Ayesha Rahman**
**Summer Internship**

The project is being done on behalf of UTI Mutual Fund company for a summer internship of 2 months in order to make the work of the analysts at the company easier.

This is <u>PROJECT 2 – Flights</u> at UTI.

## Contents

## Introduction:

In the past, researchers and scientists faced challenges in accessing relevant data for their ideas or thoughts. The scarcity of available information made it difficult for them to conduct comprehensive research (1). However, with the advent of the World Wide Web, a vast amount of data has become easily accessible. To utilize this information, researchers often resort to web scraping or manual data extraction through traditional copy-and-paste methods.

Nevertheless, certain scenarios, such as extracting financial data or hotel prices, can be tiresome and repetitive when done manually. To alleviate this issue, various methods, including APIs and web scraping, have been employed. This proposal aims to focus on utilizing Python to facilitate legal and ethical web scraping techniques, particularly for individuals who lack access to APIs or cannot afford their usage.

# Objectives

The initial phase of the project entails the utilization of web scraping techniques to gather valuable insights on flight prices. The data will be collected based on various parameters, including the departure location, arrival location, departure and arrival dates, and whether it is a one-way or round-trip journey. The resultant data will be meticulously compiled and stored in an Excel spreadsheet format for further analysis.

Conducting extensive research on the subject matter, I embarked on web scraping Google Flights to extract a table containing the requisite data for analytical purposes. This endeavour was guided and supported by my mentors, who have expertly advised me on the multifaceted aspects of this project, encompassing both Finance and Computer Science domains. For the purpose of this proposal, my focus lies exclusively on the Computer Science aspect.

The ultimate objective of this project is to deliver a comprehensive and organized table that presents a detailed overview of flight prices. By employing web scraping techniques, we seek to streamline the process of data collection and presentation, enabling analysts to access and analyse the pricing details efficiently and effectively. Through this endeavour, we aspire to contribute to the optimization of the travel planning process and facilitate data-driven decision-making in the realm of flight bookings.

| | DepartedLocation | ArrivalLocation | DepartedDate | DepartureTime | ArrivalTime | Company | Duration | Stops | Emissions | emission_comparison | price | departure_airport | arrival_airport | category |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | MUMBAI | GOA | 06-08-2023 | 11:05 | 10:30+1 | Ethiopian, ITA | 26 hrs 55 min | 2 stops | 493 kg CO2 | +6% emissions | ₹60,101 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | best_flights |
| 3 | MUMBAI | GOA | 06-08-2023 | 21:55 | 10:30+1 | Etihad, ITA | 16 hrs 5 min | 2 stops | 474 kg CO2 | Avg emissions | ₹101,981 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | best_flights |
| 4 | MUMBAI | GOA | 06-08-2023 | 00:50 | 12:50 | Delayed 35 min | 15 hrs 30 min | 1 stop | 393 kg CO2 | -15% emissions | ₹113,115 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | best_flights |
| 5 | MUMBAI | GOA | 06-08-2023 | 02:25 | 13:25 | KLMOperated by KLM Cityhopper | 14 hrs 30 min | 1 stop | 514 kg CO2 | +11% emissions | ₹116,528 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | best_flights |
| 6 | MUMBAI | GOA | 06-08-2023 | 11:05 | 14:30+1 | Ethiopian, ITA | 30 hrs 55 min | 2 stops | ~Unknown emissions | | ₹60,101 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 7 | MUMBAI | GOA | 06-08-2023 | 04:05 | 10:30+1 | Ethiopian, ITA | 33 hrs 55 min | 2 stops | 493 kg CO2 | +6% emissions | ₹66,149 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 8 | MUMBAI | GOA | 06-08-2023 | 10:10 | 10:30+1 | Emirates, ITA | 27 hrs 50 min | 2 stops | 497 kg CO2 | +7% emissions | ₹98,644 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 9 | MUMBAI | GOA | 06-08-2023 | 17:05 | 10:30+1 | Etihad, ITA | 20 hrs 55 min | 2 stops | 506 kg CO2 | +9% emissions | ₹101,981 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 10 | MUMBAI | GOA | 06-08-2023 | 17:05 | 14:30+1 | Etihad, ITA | 24 hrs 55 min | 2 stops | ~Unknown emissions | | ₹101,981 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 11 | MUMBAI | GOA | 06-08-2023 | 17:05 | 18:35+1 | Etihad, ITA | 29 hrs | 2 stops | ~Unknown emissions | | ₹101,981 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 12 | MUMBAI | GOA | 06-08-2023 | 14:30+1 | 14:30+1 | Etihad, ITA | 20 hrs 5 min | 2 stops | ~Unknown emissions | | ₹101,981 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 13 | MUMBAI | GOA | 06-08-2023 | 21:55 | 18:35+1 | Etihad, ITA | 24 hrs 10 min | 2 stops | ~Unknown emissions | | ₹101,981 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 14 | MUMBAI | GOA | 06-08-2023 | 21:55 | 22:45+1 | Etihad, ITA | 28 hrs 20 min | 2 stops | ~Unknown emissions | | ₹101,981 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 15 | MUMBAI | GOA | 06-08-2023 | 10:10 | 22:45 | Emirates, ITA | 16 hrs 5 min | 2 stops | ~Unknown emissions | | ₹108,346 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 16 | MUMBAI | GOA | 06-08-2023 | 21:00 | 12:50+1 | Air India, Lufthansa, Air Dolomiti | 19 hrs 20 min | 2 stops | 485 kg CO2 | Avg emissions | ₹112,525 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 17 | MUMBAI | GOA | 06-08-2023 | 23:05 | 10:30+1 | Air India, Etihad, ITA | 14 hrs 55 min | 2 stops | 481 kg CO2 | Avg emissions | ₹112,603 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 18 | MUMBAI | GOA | 06-08-2023 | 23:05 | 14:30+1 | Air India, Etihad, ITA | 18 hrs 55 min | 2 stops | ~Unknown emissions | | ₹112,603 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 19 | MUMBAI | GOA | 06-08-2023 | 23:05 | 18:35+1 | Air India, Etihad, ITA | 23 hrs | 2 stops | ~Unknown emissions | | ₹112,603 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 20 | MUMBAI | GOA | 06-08-2023 | 23:05 | 22:45+1 | Air India, Etihad, ITA | 27 hrs 10 min | 2 stops | ~Unknown emissions | | ₹112,603 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 21 | MUMBAI | GOA | 06-08-2023 | 15:35 | 12:50+1 | Air India, Lufthansa, Air Dolomiti | 24 hrs 45 min | 2 stops | 452 kg CO2 | Avg emissions | ₹115,156 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 22 | MUMBAI | GOA | 06-08-2023 | 16:00 | 12:50+1 | Air India, Lufthansa, Air Dolomiti | 24 hrs 20 min | 2 stops | 495 kg CO2 | +6% emissions | ₹115,156 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 23 | MUMBAI | GOA | 06-08-2023 | 19:00 | 12:50+1 | Air India, Lufthansa, Air Dolomiti | 21 hrs 20 min | 2 stops | 452 kg CO2 | Avg emissions | ₹119,434 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 24 | MUMBAI | GOA | 06-08-2023 | 22:45 | 13:25+1 | IndiGo, KLMOperated by KLM Cityhopper | 18 hrs 10 min | 2 stops | 506 kg CO2 | +9% emissions | ₹120,679 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 25 | MUMBAI | GOA | 06-08-2023 | 17:50 | 17:45+2 | IndiGo, Qatar Airways, Vueling | 51 hrs 25 min | 2 stops | 475 kg CO2 | Avg emissions | ₹121,562 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 26 | MUMBAI | GOA | 06-08-2023 | 00:50 | 18:40 | Delayed 35 min | 21 hrs 20 min | 2 stops | ~Unknown emissions | | ₹131,466 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 27 | MUMBAI | GOA | 06-08-2023 | 00:50 | 18:40 | Delayed 35 min | 21 hrs 20 min | 2 stops | ~Unknown emissions | | ₹131,466 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 28 | MUMBAI | GOA | 06-08-2023 | 00:50 | 22:45 | Delayed 35 min | 25 hrs 25 min | 2 stops | ~Unknown emissions | | ₹131,466 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 29 | MUMBAI | GOA | 06-08-2023 | 00:50 | 22:45 | Delayed 35 min | 25 hrs 25 min | 2 stops | ~Unknown emissions | | ₹131,466 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 30 | MUMBAI | GOA | 06-08-2023 | 00:50 | 22:45 | Delayed 35 min | 25 hrs 25 min | 2 stops | ~Unknown emissions | | ₹131,466 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 31 | MUMBAI | GOA | 06-08-2023 | 04:30 | 22:45 | Emirates, ITA | 21 hrs 45 min | 2 stops | ~Unknown emissions | | ₹136,066 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 32 | MUMBAI | GOA | 06-08-2023 | 04:30 | 22:45 | Emirates, ITA | 21 hrs 45 min | 2 stops | ~Unknown emissions | | ₹136,066 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 33 | MUMBAI | GOA | 06-08-2023 | 19:20 | 22:45+1 | Emirates, ITA | 30 hrs 55 min | 2 stops | ~Unknown emissions | | ₹136,066 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 34 | MUMBAI | GOA | 06-08-2023 | 19:20 | 22:45+1 | Emirates, ITA | 30 hrs 55 min | 2 stops | ~Unknown emissions | | ₹136,066 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 35 | MUMBAI | GOA | 06-08-2023 | 22:20 | 18:35+1 | Emirates, ITA | 23 hrs 45 min | 2 stops | ~Unknown emissions | | ₹136,066 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 36 | MUMBAI | GOA | 06-08-2023 | 22:20 | 22:45+1 | Emirates, ITA | 27 hrs 55 min | 2 stops | ~Unknown emissions | | ₹136,066 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |
| 37 | MUMBAI | GOA | 06-08-2023 | 22:20 | 22:45+1 | Emirates, ITA | 27 hrs 55 min | 2 stops | ~Unknown emissions | | ₹136,066 | Chhatrapati Shivaji Maharaj International Airport | Genova City Airport | other_flights |

# Imports and Functions:

## Imports used:

1. from playwright.sync_api import sync_playwright
2. from selectolax.lexbor import LexborHTMLParser
3. import json, time
4. import pandas as pd
5. import tkinter as tk
6. from tkinter import Label, Entry, Button, Tk
7. import tkinter.messagebox as mbox

# Functions Created:

1. **Functions for Handling Lock File:**

    a. **`log_action(action)`:**

    The function `log_action(action)` is a simple logging utility designed to record actions or events into a log file. When called with an `action` as its parameter, the function appends the `action` to a file named 'log.txt'. Each `action` is written as a new line in the log file, ensuring that each action is recorded sequentially. This function can be useful in various applications, such as tracking user interactions, monitoring system events, or debugging processes. By maintaining a log file, developers and system administrators can review the recorded actions later to gain insights into the behavior of a program or system.

2. **Custom Function for Displaying Popup and user input:**

    a. **`get_trip_type()`:**

    The `get_trip_type()` function is designed to obtain user input for selecting either a one-way or a round trip option using a popup window. The function creates a simple graphical user interface (GUI) with two buttons labeled "One-way" and "Round trip" for the user to choose from. When the user clicks on one of the buttons, the corresponding function (`on_one_way()` for one-way or `on_round_trip()` for round trip) is triggered. Each of these functions sets the `window.choice` variable to either "1" for one-way or "2" for round trip, and then closes the popup window.

    The main functionality is encapsulated in the `Tk()` window, a class provided by the Tkinter library for creating GUI applications in Python. The function then returns the value of `window.choice`, which represents the user's selection (either "1" or "2") after the GUI event loop (`window.mainloop()`) finishes running. This value can be used to determine the user's preference for the type of trip and further process the input in the rest of the program.

    b. **`get_user_input(trip_type)`:**

    The `get_user_input(trip_type)` function is designed to obtain user input using a graphical user interface (GUI) window. Its primary purpose is to collect information related to a travel booking, particularly the trip type (either one-way or round trip). The function creates a GUI window to provide a user-friendly interface. It includes entry fields and labels to gather essential details like departure date and destination.

    If the trip type selected by the user is round trip, an additional entry field is added to gather the return date. This ensures the function caters to both one-way and round trip booking scenarios. Once all necessary elements are set up, the function runs the GUI event loop, allowing the user to input the required information and then proceed with further actions like booking the trip based on the user's choices. The GUI window facilitates a seamless and interactive user experience throughout the travel booking process.

3. **Web Scraping Function:**

    a. **`scrape_google_flights(parser)`:**

The function `scrape_google_flights(parser)` is designed to extract flight data from the Google Flights website using a parser. The parser is expected to have the capability to parse HTML and CSS elements.

The function starts by initializing an empty dictionary named `data` to store the extracted flight information. It then uses the parser to identify two sets of elements: `categories` and `category_results`. Each category corresponds to a different class of flights (e.g., economy, business, first class), and the `category_results` contain the flight details for each category.

The function iterates through the `categories` and their corresponding `category_results`. For each flight result within a category, it extracts various attributes such as `departure_date`, `arrival_date`, `company`, `duration`, `stops`, `emissions`, `emission_comparison`, `price`, and `price_type`. These attributes represent different aspects of the flights.

The extracted data for each individual flight is stored in a dictionary named `flight_data`. Depending on the structure of the result element, the function extracts additional information such as `service`, or if not available, `departure_airport` and `arrival_airport` information.

The data for each category is collected and stored in the `data` dictionary, where the keys are the lowercase category names with spaces replaced by underscores. Finally, the function returns the complete `data` dictionary containing flight information for different categories from Google Flights.

4. **Function to navigate Google Flights page:**
   a. `get_page(playwright, from_place, to_place, departure_date, return_date, trip_type)`:

   The function "get_page" is designed to automate the process of navigating to the Google Flights page and entering user inputs to search for flights. It uses the Playwright library to launch a headful Chromium browser instance and create a new page.

   Depending on the selected trip type (one-way or round trip), the function performs different actions to input the departure and arrival cities, departure and return dates, and other necessary details. For a one-way trip, it selects "One-way" in the dropdown, types the "From" city, "To" city, and "Departure date," and clicks on "Explore" to search for flights. For a round trip, it additionally types the "Return date" and follows the same steps as the one-way trip.

   After interacting with the page, the function then parses the page content using the LexborHTMLParser and closes the page. Overall, this function simplifies the process of searching for flights on Google Flights by automating the data entry and search steps.

5. `run(playwright)`:

   The function `run(playwright)` is designed to scrape flight data from Google Flights based on user input, and then save the scraped data to an Excel file. It first calls the `get_trip_type()` function to determine the type of trip (e.g., one-way or round trip). It then prompts the user to input necessary details related to the trip (e.g., departure and return locations, dates) using `get_user_input(trip_type)`. The function checks if the user has provided the required inputs. If so, it proceeds with the next steps. It calls the `get_page()` function to use Playwright (a

browser automation tool) to fetch the Google Flights page corresponding to the user's trip details. The `scrape_google_flights()` function is called to extract flight data from the Google Flights page. The scraped data is organized into a DataFrame using pandas, with additional information such as the flight category (e.g., economy, business) added to each entry. The DataFrame is then saved to an Excel file named in the format "{departure_date}-{from_place}.xlsx" to the current directory. If the data is successfully saved, it logs the action and displays a success message using `mbox.showinfo()`. If any errors occur during the process, they are caught, logged, and likely displayed to the user.

# Project Scope

The scope of this project revolves around the development of a web scraping tool specifically tailored to Google Flights. The primary objective is to extract relevant flight price data based on user-provided inputs such as arrival location, departure location, and date of travel. The tool will be designed to scrape only one URL, which is the Google Flights website.

To achieve this, the code will incorporate some key functionalities.

1. User Input Processing: The tool will be equipped to accept user inputs, including the departure location, arrival location, and desired travel date. These inputs will serve as the parameters for the web scraping process.

2. Data Extraction from Google Flights: Using web crawling techniques, the Python code will navigate the Google Flights website and extract the necessary flight price information based on the provided inputs. The extraction process will encompass data related to one-way and round-trip options for the specified travel date.

3. Intelligent Data Scraping: The web scraping mechanism will be intelligently designed to accurately extract flight prices based on the departure and arrival locations, ensuring comprehensive data collection. The scraped data will include details such as departure times, arrival times, airlines, and trip type.

4. Output in Excel Format: The resulting flight price data will be presented in the form of an Excel spreadsheet. This output format ensures ease of accessibility and manipulation for analysts, as they can employ familiar spreadsheet tools to further analyze the extracted data.

5. Structured Tabular Presentation: The Excel sheet will be organized in a tabular format, resembling a clear and well-structured table. The table will present flight options, departure and arrival details, corresponding prices, and travel durations, providing analysts with a comprehensive overview of available flight choices.

6. User-Friendly Experience: The tool will be designed to offer a seamless and user-friendly experience. It will allow users to easily update and manage input data by reading and processing website links and HTML classes from an Excel sheet. This approach adds flexibility and convenience to the data extraction process.

7. Limitations: The scope of this project is limited to web scraping from Google Flights only. Additionally, the tool will be focused solely on flight price data extraction and will not cover other aspects of the travel planning process.

Overall, this project aims to develop an efficient and reliable web scraping tool that empowers users to obtain valuable flight price information from Google Flights, enabling data-driven decision-making for travel arrangements.
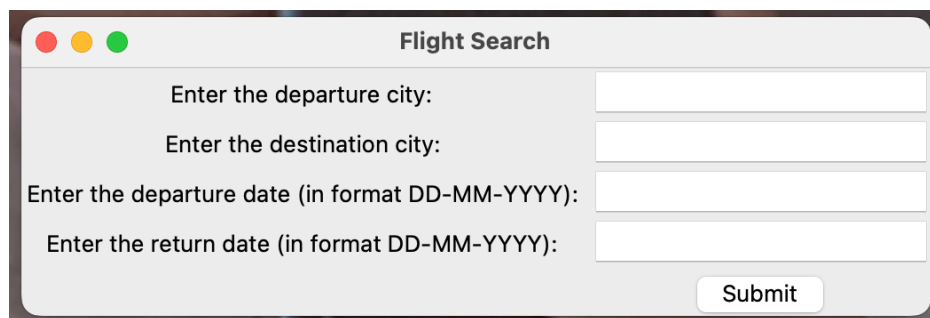
# Methodology

In this project, the development of a web scraping tool for extracting Google Flight details will follow a systematic and iterative approach. The methodology will be structured into three key iterations, each focusing on enhancing the functionality, usability, and error-handling capabilities of the tool.

## Iteration 1

During the initial iteration of the project, the primary focus will be on establishing the core functionality of the web scraping tool with a specific emphasis on extracting round-trip flight data from Google Flights. This iterative phase will involve a series of key steps and features designed to create a robust and user-friendly tool.

To begin with, advanced web scraping techniques will be implemented using Python code, enabling the tool to navigate the complex structure of the Google Flights website and extract comprehensive flight details. These details will be based on the user-provided inputs, which include the departing date, return date (for round trips), departing location, and arrival location. The aim is to capture all relevant information related to available flights, including airline options, departure and arrival times, flight durations, and, most importantly, pricing details.



Once the extraction process is completed, the obtained flight data will be meticulously organized and systematically saved in an Excel file. To ensure easy tracking and management of the collected data, the file will be named according to the departure date and departure location. This structured approach to data storage facilitates efficient access to the extracted flight details, streamlining further analysis and decision-making.

Additionally, a log file will be generated during the web scraping process. This log file will serve as a valuable record of the entire scraping operation, capturing essential information such as timestamps, the success status of each data extraction, and any encountered issues or errors during the process. This log file not only helps in debugging and refining the tool but also ensures transparency and traceability, providing a clear audit trail of the web scraping activities.

To enhance the user experience and simplify the input process, a user-friendly popup will be incorporated into the tool. This popup will serve as an intuitive interface that allows users to provide their desired details for a round-trip journey conveniently. Within this popup, users will be prompted to enter their departing date, return date, departing location, and arrival location. The interactive nature of the popup ensures a seamless user experience, making it easier for users to interact with the tool and initiate the data extraction process effortlessly.

This first iteration marks a foundational phase of the project, where the core functionality for round-trip data extraction from Google Flights is established. The subsequent iterations will build
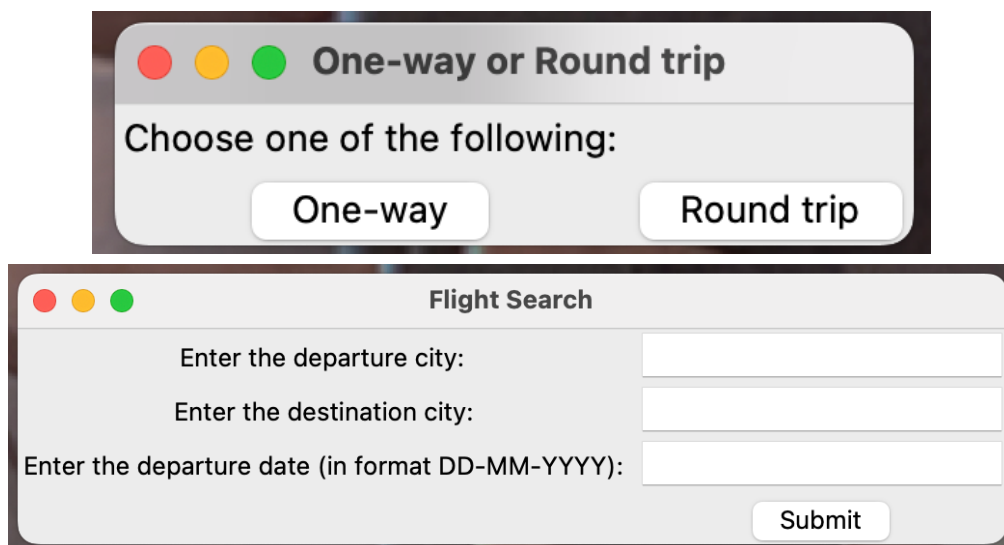
upon this foundation, extending the tool's capabilities to include one-way trip data extraction, additional error handling and GUI features, as well as more refined data output formats. The iterative approach allows for gradual improvements, providing a flexible and adaptable solution that aligns with the evolving needs of users seeking valuable flight information from Google Flights.

## Iteration 2

In the second iteration of the project, we will undertake significant advancements to bolster the capabilities of our web scraping tool, enabling it to seamlessly handle the extraction of one-way trip data. This iterative phase is geared towards enriching the user experience and diversifying the tool's functionalities to cater to a broader range of travel preferences.

The centrepiece of this iteration is the augmentation of the Python code to adeptly scrape data for one-way trips. I will intricately fine-tune the web scraping algorithm to account for the nuances and variations in the Google Flights website when it comes to presenting information for one-way journeys. By intelligently adapting the scraping mechanisms, the code aims to ensure that users receive comprehensive and accurate flight details for their desired one-way travel itineraries.

Moreover, to furnish users with the utmost convenience and control over their queries, we will integrate an intuitive and user-friendly popup. This will serve as a trip-type selection interface, allowing users to effortlessly choose between one-way and round-trip journeys. The brilliance of this enhancement lies in its versatility, as the popup inputs will dynamically adapt based on the selected trip type. For example, if a user opts for a one-way trip, the popup will refrain from requesting a return date input, thereby streamlining the data entry process, and reducing any potential confusion.



A crucial aspect of automating the tool's functionality is the implementation of a dropdown selection mechanism for one-way trips. With the help of sophisticated code, the tool will be endowed with the capability to interact with the Google Flights website seamlessly. When necessary, the tool will skilfully navigate to the dropdown menu, allowing it to effortlessly select the one-way trip option. This intelligent automation mitigates the need for manual intervention, enhancing the overall efficiency and robustness of the tool.

Through the collective implementation of these enhancements, the second iteration aims to elevate the web scraping tool's performance, ensuring its adaptability to diverse travel scenarios and preferences. The objective is to empower users with a versatile and user-friendly interface that

effortlessly accommodates both one-way and round-trip journey queries. By continuously refining and improving our tool, we are committed to delivering an exceptional and valuable experience to our users seeking valuable flight details from Google Flights. The iterative approach we employ guarantees that our tool is continuously fine-tuned to meet the ever-evolving needs of analysts while maintaining the highest standards of accuracy and reliability in data extraction.

## Iteration 3

In the third iteration of this web scraping project, the focus shifts towards refining the tool's functionality to provide a seamless and user-friendly experience. One of the crucial aspects addressed in this phase is the implementation of robust error-handling mechanisms. By incorporating these error-handling features, the tool will be able to identify and manage potential issues or exceptions that may arise during the data extraction process. Whenever an error occurs, the tool will display relevant error messages in user-friendly popups, ensuring that users receive clear feedback and guidance on how to proceed.

Another key improvement in Iteration 3 involves enhancing the output format of the extracted flight data. For round-trip journeys, the Excel output file will be enriched with additional columns, namely 'DepartedLocation,' 'ArrivalLocation,' 'DepartedDate,' and 'ReturnDate.' These additional columns will significantly enhance the comprehensiveness of the output, as analysts will be presented with a structured and detailed representation of round-trip flight data. This organized presentation will facilitate better analysis and decision-making regarding travel plans.

Similarly, for one-way trips, the output Excel file will undergo improvements to include the essential 'DepartedLocation,' 'ArrivalLocation,' and 'DepartedDate' columns. This enhancement ensures that users receive a comprehensive overview of the extracted one-way flight data. Furthermore, recognizing that analysts often seek flexibility in their travel plans, users will have the opportunity to define the number of days for which they want flight data to be extracted. This user-defined data extraction feature is particularly beneficial for one-way trips, where analysts may want to explore flight prices for a range of travel dates to find the most suitable option.

Moreover, a notable addition to the tool's functionalities in this iteration is the capability for users to select the directory where they wish to save the output Excel file. This enhanced feature allows users to conveniently organize and manage their extracted flight data in a location of their choice, contributing to a more personalized and efficient workflow. By empowering users to define the output directory, the tool caters to individual preferences and ensures that the process aligns seamlessly with their existing data management practices.

Throughout the entire development process, meticulous testing and feedback loops are continuously integrated to ensure the tool's reliability, accuracy, and user satisfaction. Each iterative enhancement builds upon the previous version, resulting in a sophisticated web scraping tool that is well-equipped to navigate Google Flights, extract pertinent flight details for both one-way and round-trip journeys, and present the information in a well-structured Excel format. By combining error handling, user-defined data extraction, and customizable output directory selection, this project aims to deliver a powerful and user-friendly solution for analysts and analysts seeking comprehensive flight price data for their travel planning needs.

## Running of the Application:

The running of the application will involve a user-friendly interface that guides users through the entire process. Upon launching the tool, a graphical interface will prompt users to input their preferences, such as trip type (one-way or round-trip), departing date, returning date (if applicable), departing location, and arrival location. Based on these inputs, the tool will initiate the web scraping process and present the extracted flight details in an easily navigable and structured format within the Excel output.

Throughout the project's development, regular testing and feedback loops will be employed to ensure the tool's efficiency, accuracy, and usability. The iterative approach will enable gradual improvements and refinements, resulting in a sophisticated web scraping tool that caters to the dynamic needs of users seeking valuable flight details from Google Flights.


## Legality and Ethics of Web Scraping:

Web scraping involves the automated extraction and organization of data from the web, using technical tools, with the purpose of analysing the data further (2). Although the use of web scraping software and tools has become widespread, the legal and ethical considerations associated with it are often overlooked (1). The abundance of web scraping tools has created the misconception that web scraping is entirely legal, when in reality, its legality remains uncertain and falls into a "grey area" within the legal field (3).

Since there are no specific laws directly addressing web scraping, it is governed by a set of related legal theories and laws, including copyright infringement, the Computer Fraud and Abuse Act (CFAA), breach of contract, and trespass to chattels (3). To effectively prevent web scraping, website owners can explicitly prohibit it in their publicly accessible "terms of use" policy. Violating these terms and conditions may constitute a breach of contract, enabling the website owner to pursue legal action against the user (2). Furthermore, republishing scraped data or information that is owned or explicitly copyrighted by a website can result in a copyright infringement case (2).

Website owners possess the capacity and authority to establish and enforce a wide range of measures, including but not limited to imposing comprehensive restrictions or erecting robust firewalls, with the specific aim of impeding the activities of web scrapers and web crawlers in their quest to extract data from their websites. These measures effectively serve as formidable barriers, meticulously designed to thwart the efforts of automated tools, even when the data being targeted is seemingly accessible to the public and falls within the legal boundaries (2).

By proactively implementing these restrictions and fortified firewalls, website owners assert their rights and exercise control over their digital domains, actively safeguarding the valuable assets contained within. These protective measures encompass a multifaceted approach, combining cutting-edge technologies, advanced algorithms, and meticulously crafted protocols that scrutinize and evaluate incoming requests in real-time. Through the strategic implementation of IP blocking mechanisms, stringent rate limiting strategies, multifactor authentication protocols, and other sophisticated security measures, website owners effectively raise the barriers for unauthorized data extraction, diminishing the chances of successful scraping endeavours.

The authority bestowed upon website owners enables them to institute a myriad of sophisticated measures that effectively deter and hinder web scrapers and web crawlers from extracting data from their websites, even if that data is ostensibly available to the public and meets legal criteria. Through the implementation of comprehensive restrictions, fortified firewalls, and cutting-edge security measures, website owners actively assert their control over their digital realms,

safeguarding their valuable assets, complying with legal obligations, protecting user privacy, and fortifying their intellectual property rights.

## Limitations

1. **Website changes:** Web scraping heavily relies on the structure of the website being scraped. Google Flights or any other website is subject to change their HTML structure, CSS classes, or underlying technologies at any time. This can lead to the scraper not being able to locate and extract the required data correctly.

2. **Legal concerns:** Web scraping might violate the terms of service or the copyright of the website being scraped. Google Flights, like many other websites, may have explicit terms that prohibit scraping their data. Engaging in web scraping without permission could lead to legal issues.

3. **Dynamic content:** Google Flights heavily relies on JavaScript to render some of its content dynamically. The scraper provided in the project does not handle JavaScript-driven content, so it might miss some relevant information that is populated after the initial page load.

4. **Rate limiting and IP blocking**: When performing web scraping, there's a risk of being rate-limited or blocked by the website's server if too many requests are made in a short period. Google, being a large tech company, may have measures in place to detect and prevent scraping attempts.

5. **Incomplete data:** Some flights or relevant details might be missing from the scraping results due to various reasons such as incomplete data on the website, network issues, or data not being available at the time of scraping.

6. **Scraping efficiency:** The current implementation of the scraper might not be optimized for efficiency, leading to slower performance, especially when dealing with a large number of flights.

7. **Data accuracy:** Web scraping is prone to errors, and the accuracy of the scraped data heavily depends on the robustness of the parsing logic. Changes in website layout or data formatting can introduce errors or inconsistencies in the scraped information.

8. **Maintenance overhead:** As the website changes over time, the scraper will require regular maintenance and updates to ensure it continues to work correctly.

9. **Dependency on third-party libraries:** The scraper's functionality relies on the underlying parsing library being used. Changes or issues with the library could impact the scraper's performance.

10. **Limited scope:** The current scraper is specifically designed for Google Flights and might not be easily adaptable to scrape data from other flight search engines or travel websites.

It is important to be aware of these limitations and consider the legal and ethical aspects of web scraping before using such a scraper in a production environment.

## Conclusion

Engaging in this project of scraping flight details from Google Flights has been a valuable learning experience and a significant contribution to my expertise in web scraping and data extraction. Throughout this endeavor, I have gained a deeper understanding of Python programming and the intricacies of parsing data from websites.

This project has allowed me to explore the fascinating intersection of technology and the travel industry. By working on this scraper, I have realized the immense potential of web scraping in gathering essential flight information efficiently and cost-effectively. The knowledge gained from this project can be applied to various other domains that rely on web data extraction.

One of the key takeaways from this project is the importance of adapting to changes. Websites are dynamic entities, and Google Flights can undergo modifications at any time, affecting the scraper's functionality. Hence, the project reinforced the significance of maintaining and updating the code regularly to ensure it stays relevant and functional.

Ethical considerations were at the forefront of the project's development. I made sure to abide by Google Flights' terms of service and avoid any unethical scraping practices. Respecting website policies, controlling the scraping frequency, and limiting data extraction to publicly available information were critical aspects to ensure a responsible and ethical approach.

The project's limitations further highlighted the challenges faced in web scraping. Understanding and mitigating these limitations have been an essential aspect of the development process. While the current scraper serves its purpose, it may require continuous improvements to stay robust and reliable.

Overall, this project has been a rewarding journey, offering a practical application of my skills in computer science while deepening my knowledge of web scraping and data extraction. It has equipped me with a valuable toolset that can be leveraged in various projects, empowering me to solve real-world problems through data analysis and automation. Moving forward, I will continue to explore and expand my expertise in web scraping, leveraging this valuable experience to excel in the field of data-driven technologies.

## References:

1. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=need+for+web+scraping&btnG=#d=gs_qabs&t=1689568686149&u=%23p%3DF0to5SeS9PEJ
2. https://www.researchgate.net/profile/Vlad-Krotov/publication/324907302_Legality_and_Ethics_of_Web_Scraping/links/5aea622345851588dd8287dc/Legality-and-Ethics-of-Web-Scraping.pdf
3. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=snell+and+menaldo+2016&btnG=#d=gs_qabs&t=1689570600409&u=%23p%3D018PufGTz_EJ
4. Reference for code:
   https://medium.com/@chukhraiartur/scrape-google-flights-with-python-2b1677e1ee2

## About the writer

Name - Ayesha Rahman
Student ID – 201522771

Course details:

Under-graduation in Computer Science specialising in Artificial Intelligence.

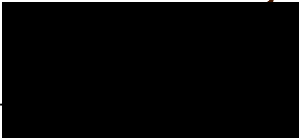Currently in year 3 (last year) at the University of Leeds.

Internship details:

Has done the internship at UTI in the summer of 2023.
(From July 10-2023 to September 03-2023)

# Acknowledgement from UTI

**Mentors:**

- Miss Pradnya S. Ganar: Associate Vice President (Fund Management)

- Mr Parag Chavan: Sr Associate Vice President (Fund Management)