

WEB SCRAPING FOR FLIGHT TICKET INFORMATION

Submitted in partial fulfilment of the required for the award of

Bachelor of Computer Science

by

THANGAVEL. D

(REGISTER NO 40290103)



DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF SCIENCE AND HUMANITIES

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

**Accredited with Grade "A" by NAAC | 12B Status by UGC|
Approved by AICTE**

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600 119

MAY 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

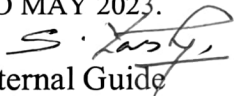
Accredited "A" Grade by NAAC | 12B Status by UGC I Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **THANGAVEL. D (40290103)** who carried out the project entitled "WEB SCRAPING FOR FLIGHT TICKET INFORMATION" under our supervision from JANUARY 2023 TO MAY 2023.

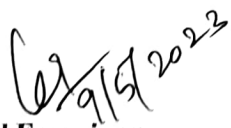

Internal Guide

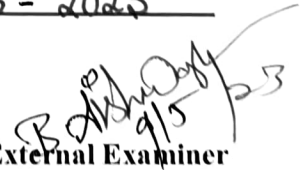
Ms. YUVASNEHA.S, M.Sc., Assistant Professor


DEAN AND HEAD OF THE DEPARTMENT

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.

Submitted for Viva voce examination held on 09-05-2023


Internal Examiner


External Examiner

DECLARATION

I am, **THANGAVEL. D (40290103)** here by declare that the Project Report entitled **“WEB SCRAPING FOR FLIGHT TICKET INFORMATION”** done by us under the guidance of **Ms. YUVASNEHA.S, M.Sc.,**(Assistant Professor) Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY,** Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600 119 is submitted in partial fulfillment of the requirements for the award of Master of Science degree in Computer Science.

THANGAVEL. D



DATE:09.05.2023

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. REKHA CHAKRAVARTHI M.E., Ph.D.** **Dean of Arts and Science** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sence of gratitude to my project guide **Ms. YUVASNEHA.S, M.Sc.**, Assistant Professor Department of Computer Science for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff member of the Department of Computer Science who were helpful in many ways for the completion of the project

THANGAVEL. D

ABSTRACT

This web scraping project aims to extract flight ticket information from various airline websites. The project involves the development of a Python script that will automate the process of collecting flight data such as flight number, departure and arrival times, airlines, origin and destination cities, and ticket prices. The script utilizes various web scraping libraries and techniques to extract flight information from airline websites. The data is then stored in a structured format, such as CSV or JSON, for further analysis and processing. The extracted data can be used for a variety of purposes such as price comparison, trend analysis, and forecasting. This project has the potential to provide valuable insights into the airline industry and help travelers make informed decisions when booking flights.

TABLE CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	5
	INTRODUCTION	8
1	1.1 DOMAIN INTRODUCTION	8
	1.2 OBJECTIVE OF THE PROJRCT	10
	1.3 PROJECT INTRODUCTION	10
	LITERTURE SURVEY	11
2	2.1 REVIEW ON LITERATURE WORK	11
	2.2 EXISTING SYSTEM	13
	2.3 PROPOSED SYSTEM	14
	SOFTWARE DESCRIPTION	15
3	3.1 PYTHON	15
	3.2 PYCHARM COMMUNITY EDITION	16
	3.3 SELENIUM	17
	3.4 BEAUTIFULSOUP	18
	3.5 PANDAS	19
	3.6 DATETIME	20
	3.7 REQUESTS	21
	3.8 JSON	22
	3.9 CHROMEDRIVER	23
	METHODOLOGY	24
4	4.1 REQUIREMENT ANALYSIS	24
	4.1.1 HARDWARE ENVIRONMENT	24

4	4.1.2 SOFTWARE REQUIRMENT	24
	4.2 SYSTEM DESIGN	24
	4.3 ARCHITECTURE DIAGRAM	25
	4.4 USECASE DIAGRAM	26
	4.5 SEQUENTIAL DIAGRAM	27
	4.6 IMPLEMENTATION	28
5	RESULT AND DISCUSSION	29
	5.1 OUTPUT CSV FILE	29
	5.2 USES OF THE EXTRACTED INFORMATION	29
6	CONCLUSION AND FUTURE WORK	30
	6.1 CONCLUSION	30
	6.2 FUTURE WORK	30
	APPENDICES	31
	SOURCE CODE	31
	OUTPUT SCREENSHOT	38
	REFERENCES	40

CHAPTER 1

INTRODUCTION

1.1 DOMAIN INTRODUCTION

Web scraping, also known as web data extraction, is a technique used to extract data from websites. It involves the automated process of collecting data from web pages by sending a request to a website and then extracting the desired information from the HTML code of the page. Web scraping can be done manually, but it is more commonly done using automated software tools or scripts, often written in programming languages like Python or JavaScript. Web scraping can be used to collect a wide range of data, including product information, customer reviews, news articles, financial data, and more. The collected data can be used for a variety of purposes, such as market research, price comparison, trend analysis, and forecasting. However, it is important to note that web scraping can be a legally sensitive area, as some websites may have policies that prohibit or limit the use of automated data collection tools. It is important to check the terms of use for each website before engaging in any web scraping activity.

Types of Web Scraping

There are several types of web scraping techniques, including:

HTML Parsing

This technique involves parsing the HTML structure of a website to extract the desired information. The data can be extracted using libraries like BeautifulSoup, lxml, and html.parser.

Web API Scraping

Some websites offer APIs (Application Programming Interfaces) that allow developers to access their data in a structured way. Web API scraping involves sending requests to these APIs and extracting the data returned in JSON or XML format.

Automated Browser Scraping

This technique involves using a headless browser, like Puppeteer or Selenium, to simulate a user's actions on a website, and then extracting the data from the resulting page source.

DOM Scraping

This technique involves manipulating the Document Object Model (DOM) of a webpage to extract the desired data. Libraries like jQuery and Cheerio can be used for this purpose.

Text Pattern Matching

This technique involves searching for specific patterns of text on a webpage to extract the desired data. Regular expressions can be used for this purpose.

Each technique has its advantages and limitations, and the choice of technique depends on the type and complexity of the data being scraped, as well as the technical capabilities and resources of the scraper.

1.2 OBJECTIVE OF THE PROJECT

The aim of our project is to extract flight information and save it in different csv file based on airport code.

1.3 PROJECT INTRODUCTION

Web scraping is the process of extracting data from websites using automated software programs, commonly known as bots or crawlers. When it comes to flight ticket information, web scraping can be a valuable tool for gathering real-time data on prices, schedules, availability, and other relevant details from different airline and travel aggregator websites. By using web scraping to collect flight ticket information, businesses and individuals can monitor prices and availability across multiple websites, identify the best deals and opportunities, and make informed decisions about when and where to book flights. Additionally, web scraping can be used to track changes in flight schedules or routes, which can be helpful in planning itineraries or anticipating disruptions. Web scraping for flight ticket information can be particularly useful for travel agencies, airlines, and online travel aggregators who need to stay competitive in a fast-paced and dynamic industry. By automating the process of collecting and analyzing flight data, these companies can quickly adapt to market trends, adjust pricing strategies, and optimize their offerings to meet the demands of customers. Overall, web scraping for flight ticket information provides a powerful tool for businesses and individuals to stay informed, make data-driven decisions, and save time and money in the travel planning process.

CHAPTER 2

LITERATURE SURVEY

2.1 REVIEW ON LITERATURE WORK

Paper 1: Web Scraping for Airline Pricing Intelligence

Author Name: A. Silventoinen and P. Järveläinen

Year of Publish: 2018

Abstract:

This study examines the potential of web scraping techniques for airline pricing intelligence. The authors collected data on flight schedules, prices, and availability from multiple airlines and travel websites using web scraping and analyzed it to identify patterns and trends in airline pricing strategies. The study highlights the challenges and limitations of web scraping, such as website design changes and data accuracy issues. The results suggest that web scraping is a useful tool for airline pricing intelligence, providing valuable insights into pricing strategies and market trends. The study emphasizes the importance of using accurate and reliable web scraping techniques to ensure the quality of the data collected.

Paper 2: Airfare Data Collection Using Web Scraping and API Integration

Author Name: A. Tahir et al.

Year of Publish: 2018

Abstract:

This study examines the use of web scraping and API integration for collecting airfare data from multiple airlines and travel websites. The authors compare the effectiveness of different web scraping tools and APIs for collecting data and analyze the data to identify trends in airfare prices. The results demonstrate the potential of web scraping and API integration for collecting and analyzing airfare data, providing valuable insights into pricing strategies and market trends. The study emphasizes the importance of using accurate and reliable web scraping tools and APIs for data collection, and the need for ongoing monitoring of website design changes and data accuracy issues.

Paper 3: Scraping Airline Ticket Prices: A Comparative Study of Five Web Scraping Tools

Author Name: M. B. Hanif et al.

Year of Publish: 2021

Abstract:

This study compares the effectiveness of five different web scraping tools for collecting airline ticket prices. The authors analyze the data to identify the best tool for collecting airfare data and discuss the potential applications of web scraping in the travel industry. The results demonstrate the usefulness of web scraping for collecting and analyzing airfare data, providing valuable insights into pricing strategies and market trends. The study highlights the importance of using accurate and reliable web scraping tools for data collection, and the need for ongoing monitoring of website design changes and data accuracy issues to ensure the quality of the data collected.

Paper 4: Airfare Prediction Using Web Scraping and Machine Learning

Author Name: R. Chandrasekar et al

Year of Publish: 2019

Abstract:

This study explores the use of web scraping and machine learning techniques for predicting airfare prices. The authors collected data on airfare prices and other relevant features using web scraping, and trained machine learning models to predict future airfare prices. The results demonstrate the potential of web scraping and machine learning for predicting airfare prices, providing insights into pricing trends and potential savings for travelers. The study highlights the importance of using accurate and reliable web scraping techniques for data collection and the need for appropriate machine learning algorithms for predictive analysis. The findings suggest that web scraping and machine learning techniques can be a useful tool for the travel industry, providing valuable insights for both customers and service providers.

Paper 5: A study on web scraping of online travel agents for airfare data collection

Author Name: K. Ramya and B. Priyanga

Year of Publish: 2020

Abstract:

This study examines the use of web scraping techniques for collecting airfare data from online travel agents. The authors collected data on airfare prices and other relevant features using web scraping, and analyzed the data to identify trends in pricing strategies and market trends. The study also discusses the challenges and limitations of web scraping, such as website design changes and data accuracy issues. The results demonstrate the potential of web scraping for collecting and analyzing airfare data, providing valuable insights for both customers and service providers in the travel industry. The study highlights the importance of using accurate and reliable web scraping techniques for data collection, and the need for ongoing monitoring of website design changes and data accuracy issues to ensure the quality of the data collected. The findings suggest that web scraping can be a useful tool for the travel industry, providing valuable insights for both customers and service providers.

2.2 EXISTING SYSTEM

Web scraping projects that focus on extracting flight information often rely solely on either BeautifulSoup or Selenium. However, with the frequent changes that occur on websites and the software used for web scraping, many of these projects become outdated over time. This can result in errors or incomplete data extraction, as the code may no longer be compatible with the updated website structure or software version. To combat this issue, it is important for web scraping projects to adapt to changes and updates in the website and software. This can be achieved by utilizing the latest versions of the software and keeping up-to-date with any changes made to the website structure. Additionally, using a combination of BeautifulSoup and Selenium can provide a more comprehensive and adaptable approach to web scraping, as mentioned in the previous answer. Ultimately, it is important for web scraping projects to stay flexible and adaptable in order to provide accurate and up-to-date flight information for users. This requires staying on top of changes in website structure and software updates, and utilizing the most effective tools and techniques for web scraping.

2.3 PROPOSED SYSTEM

This project takes advantage of both BeautifulSoup and Selenium to maximize flexibility and efficiency in web scraping. By using the latest versions of the software and adapting to the current structure of the website, the project is able to work seamlessly and efficiently. BeautifulSoup is used for parsing the HTML code and extracting data from the webpage, while Selenium is used for interacting with dynamic web elements and handling user interactions. This combination allows for a more comprehensive web scraping approach, where Selenium can handle complex interactions like clicking buttons and filling forms, and BeautifulSoup can extract the data from the HTML code without rendering the entire page. Overall, the use of both BeautifulSoup and Selenium in this project provides a robust and adaptable solution for efficient web scraping.

CHAPTER 3

SOFTWARE DESCRIPTION

3.1 PYTHON

Python is a high-level, interpreted programming language that was first introduced in 1991. The language's versatility enables it to cater to various applications, including web development, scientific computing, data analysis, artificial intelligence, and machine learning. One of the unique aspects of Python is its syntax, which is designed to be easily readable and writable, making it an excellent choice for developers of all skill levels. Python supports several programming paradigms, including object-oriented, functional, and procedural programming, providing developers with multiple options to tackle various programming tasks. The language has a vast and active community of developers who contribute to its development, resulting in a plethora of libraries, frameworks, and tools that streamline the creation of complex programs with minimal effort. Python's standard library contains a vast range of modules that simplify various tasks, such as file management, networking, and data processing. As an open-source language, Python is free to use, distribute, and modify. It runs seamlessly on multiple platforms, including Windows, macOS, and Linux, which makes it an excellent choice for cross-platform development. Furthermore, Python has several implementations, including CPython, Jython, IronPython, and PyPy, each with unique features and advantages. Overall, Python is a powerful and easy-to-learn programming language with numerous applications. Its ability to cater to various programming paradigms, vast community support, and extensive standard library make it a top choice for developers, data scientists, and researchers worldwide.

Easy to learn: Python has a simple and intuitive syntax that is easy to learn and understand, making it an ideal language for beginners.

High-level language: Python is a high-level language, which means that it abstracts away low-level details such as memory management, allowing developers to focus on the problem at hand.

Large standard library: Python has a large and comprehensive standard library that provides built-in modules and functions for a wide range of tasks, such as data processing, networking, and file I/O.

Cross-platform compatibility: Python is a cross-platform language, which means that it can run on different operating systems such as Windows, Linux, and macOS.

Open-source: Python is an open-source language, which means that it is free to use, distribute, and modify.

Extensible: Python is a highly extensible language that supports the integration of other programming languages such as C and C++, making it suitable for writing high-performance applications.

Data analysis and visualization: Python has several powerful libraries such as NumPy, Pandas, and Matplotlib that make it an ideal language for data analysis and visualization.

Community: Python has a large and active community of developers and users who provide support, documentation, and code examples.

Machine learning and AI: Python has several powerful libraries such as TensorFlow, Keras, and Scikit-Learn that make it an ideal language for developing machine learning and AI applications.

Web development: Python has several powerful web frameworks such as Django and Flask that make it an ideal language for web development.

3.2 PYCHARM COMMUNITY EDITION

PyCharm Community Edition is an integrated development environment (IDE) for Python programming language that is both free and open-source. JetBrains, the developer of PyCharm, designed it to provide Python developers with a range of features that make it easier to create Python applications. The IDE has a user-friendly interface that simplifies

coding and debugging processes. PyCharm Community Edition supports various Python frameworks and technologies such as Django, Flask, and Pyramid, among others. The IDE provides several productivity features like code completion, code inspections, and quick-fixes that help developers write high-quality code. PyCharm Community Edition comes with a built-in debugger that makes it easier for developers to debug their code. The debugger provides features like breakpoints, watches, and step-by-step code execution. Additionally, PyCharm integrates with popular version control systems like Git, Mercurial, and Subversion, making it easy to manage project source code. Other notable features of PyCharm Community Edition include support for multiple languages, integration with databases, support for unit testing frameworks, and a powerful refactoring system. Overall, PyCharm Community Edition is an excellent IDE for Python developers who want to work with a feature-rich and intuitive tool without any financial cost. The IDE has many features that make it easier to write, test, and debug Python code. Its integration with popular tools and frameworks makes it a top choice for many developers.

3.3 SELENIUM

Selenium is a popular open-source framework used for automating web browsers. It allows users to control a browser and simulate user actions, such as clicking links, filling out forms, and scrolling through pages. Selenium can be used for web scraping, testing, and general browser automation tasks. In Python, the Selenium package provides a Python interface to the Selenium framework, allowing users to control web browsers using Python code. The Selenium package provides a range of tools and methods to help automate browser actions, including selecting and interacting with web elements, handling pop-ups and alerts, and capturing screenshots of web pages. Selenium can interact with a wide range of web browsers, including Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari, among others. The Selenium package can also integrate with other Python packages, such as BeautifulSoup and Pandas, to help with web scraping and data analysis tasks.

Cross-browser compatibility testing: Selenium can test web applications across different browsers, including Chrome, Firefox, Safari, and Internet Explorer.

Automated testing: Selenium can automate repetitive tasks such as filling out forms, clicking buttons, and navigating through web pages, reducing manual testing efforts and time.

Wide range of programming languages: Selenium supports several programming languages, including Python, Java, C#, Ruby, and JavaScript, making it accessible to developers with different language preferences.

Integration with other tools: Selenium can be integrated with other testing frameworks and tools, such as TestNG, JUnit, and Jenkins, for further testing and continuous integration.

Real-time testing: Selenium can perform real-time testing, simulating user interactions with web applications, making it useful for testing complex applications and workflows.

Large community: Selenium has a large and active community of users and contributors, providing support, documentation, and code examples.

Scalability: Selenium can scale to handle large and complex web applications, making it suitable for testing enterprise-level applications.

3.4 BEAUTIFUL SOUP

Beautiful Soup is a Python library used for web scraping tasks. It is designed to make it easier to extract data from HTML and XML documents by providing a set of tools for navigating and searching the document's structure. With Beautiful Soup, developers can parse an HTML or XML document and extract specific information from it. The library provides methods to search for tags and attributes, navigate the document tree, and extract the text and attributes of specific HTML or XML elements. One of the key advantages of Beautiful Soup is its ability to handle poorly structured HTML and XML documents. It can work with documents that have missing or incomplete tags and can also handle documents with invalid markup. Beautiful Soup can repair the broken HTML or XML and parse it correctly, making it an ideal choice for web scraping tasks where the data source is not always well-formed. Beautiful Soup can work with different parsers, including the built-in Python parser, lxml, and html5lib, among others. This makes it a versatile library that can handle different types of HTML and XML documents. Overall, Beautiful Soup is a powerful library for web scraping tasks that simplifies the process of extracting data from HTML and XML documents. Its ability to handle poorly structured documents and work with different

parsers makes it an ideal choice for many developers and businesses.

Easy to use: BeautifulSoup has a simple and intuitive syntax that makes it easy for beginners to use.

Robust parser: BeautifulSoup uses a robust parser to navigate through HTML and XML documents, which makes it easy to extract data even from poorly structured documents.

Extensible: BeautifulSoup can be easily extended with new parsers and functionalities using plugins and extensions.

Cross-platform: BeautifulSoup works on multiple platforms, including Windows, Linux, and macOS.

Large community: BeautifulSoup has a large and active community of users and contributors who provide support, documentation, and code examples.

Integration with other tools: BeautifulSoup can be easily integrated with other Python libraries, such as Pandas and NumPy, for further data analysis and manipulation.

Fast and efficient: BeautifulSoup is fast and efficient in handling large and complex web scraping tasks, making it suitable for processing large amounts of data.

3.5 PANDAS

Pandas is a versatile and widely used open-source data analysis library for the Python programming language. Its core functionalities are built around two main data structures: Series and DataFrames, which enable users to perform various operations on structured data, such as cleaning, transforming, aggregating, and visualizing data. With Series and DataFrames, Pandas offers numerous data manipulation and analysis tools. It provides efficient methods for filtering and selecting data, handling missing data, merging and reshaping data, and performing statistical calculations. Pandas also includes extensive time-series functionality that can handle irregularly sampled and missing data with ease.

Additionally, Pandas is renowned for its compatibility with a vast range of data sources. It can read and write data in various file formats, including CSV, Excel, SQL databases, and JSON files, as well as access and process data from various web APIs. Pandas also offers powerful data visualization tools for creating custom and interactive plots and charts, using Matplotlib and other libraries. Overall, Pandas is a crucial library for data analysts and data scientists working on structured data in Python. Its flexibility, ease of use, and extensive set of tools make it an indispensable tool for data analysis in Python.

Efficient data handling: Pandas provides fast and efficient data handling tools, allowing users to work with large and complex datasets with ease.

Data cleaning and preparation: Pandas has several built-in functions for data cleaning and preparation, such as handling missing values, removing duplicates, and transforming data types.

Powerful data analysis: Pandas has powerful data analysis tools that enable users to explore, filter, and aggregate data easily.

Integration with other tools: Pandas can be easily integrated with other data analysis and visualization tools, such as Matplotlib and Seaborn.

Flexibility: Pandas is highly flexible and can handle a wide variety of data formats, including CSV, Excel, SQL databases, and JSON.

Easy to learn: Pandas has a simple and intuitive syntax, making it easy for new users to learn and use.

Large community: Pandas has a large and active community of users and contributors, providing support, tutorials, and code examples.

3.6 DATETIME

The datetime package is a built-in Python module that provides classes for working with dates, times, and time intervals. It is widely used in Python applications for tasks such as

time-stamping events, working with time zones, and scheduling tasks.

The key classes in the datetime package include:

datetime: A class that represents a date and time object, including year, month, day, hour, minute, second, and microsecond. It provides methods for performing arithmetic operations on dates, comparing dates, and formatting dates as strings.

date: A class that represents a date object, including year, month, and day. It provides methods for working with dates, such as computing the difference between two dates, formatting dates as strings, and checking if a year is a leap year.

time: A class that represents a time object, including hour, minute, second, and microsecond. It provides methods for working with time, such as formatting time as a string and converting time to a timestamp.

timedelta: A class that represents a duration or time interval, such as the difference between two dates or times. It provides methods for performing arithmetic operations on time intervals, such as adding or subtracting days, hours, minutes, or seconds.

The datetime package also includes functions for working with time zones, such as `timezone` and `tzinfo`, which allow developers to handle daylight saving time and time zone conversions. Overall, the datetime package is a powerful tool for working with dates, times, and time intervals in Python, providing a wide range of functionalities that can be used for various applications, including data analysis, web development, and scientific computing.

3.7 REQUESTS

The requests package is a popular third-party library in Python that allows developers to make HTTP requests to web pages and APIs, and handle the responses in their Python code. It provides a simple and elegant way to interact with web services, including GET and POST requests, form submission, file uploads, and more. Using the requests package, developers can easily send HTTP requests and retrieve responses by using a simple and intuitive API. The response object can be parsed to extract data such as HTML content, JSON, and other formats.

The key features of the requests package include:

- Support for HTTP and HTTPS protocols
- Support for various HTTP methods such as GET, POST, PUT, DELETE, and more
- Easy handling of query parameters, request headers, and cookies
- Built-in support for authentication and session management
- Support for file uploads and downloads
- Customizable timeouts and error handling
- Compatibility with third-party libraries for parsing and handling response data, such as BeautifulSoup and JSON.

The requests package is widely used in various applications, including web scraping, automation, testing, and building web services. It is also used in data analysis and scientific computing, as it allows developers to easily access and process data from web APIs and services. Overall, the requests package is a powerful and versatile library in Python for making HTTP requests and handling responses, and it provides a simple and intuitive API for developers to work with.

3.8 JSON

JSON or JavaScript Object Notation is a lightweight format used to exchange data between different systems. It is a text-based format comprising of key-value pairs. The keys are always in string format while the values can be strings, numbers, booleans, null, arrays, or other JSON objects. JSON data can be easily parsed and converted into various programming languages, including Python.

Python has a built-in module known as 'json' that encodes and decodes JSON data. This module enables developers to convert Python objects to JSON strings and vice versa easily. The 'json' module also provides various methods for formatting JSON data in a readable format and handling exceptions during the encoding or decoding process.

Developers can use the 'json.dumps()' method to convert Python objects to a JSON string, and the 'json.loads()' method to convert a JSON string to a Python object. The 'json.load()' and 'json.dump()' methods can also be used to read and write JSON data to and from a file.

The json package in Python has several options for customizing the encoding and decoding process. For instance, developers can use the 'json.JSONEncoder' class to customize the way

Python objects get encoded into JSON, or the 'json.JSONDecoder' class to customize how JSON strings get decoded into Python objects.

In summary, the 'json' package in Python provides a simple and user-friendly interface for working with JSON data in Python applications. It is commonly used for exchanging data between client-side and server-side applications, logging data, and storing configuration files.

3.9 CHROMEDRIVER

ChromeDriver is a standalone executable program that enables the Google Chrome browser to communicate with the Selenium WebDriver. It is necessary for automated testing of web applications with Selenium and Chrome. ChromeDriver is available for different operating systems such as Windows, Linux, and macOS. ChromeDriver allows Selenium to interact with the Chrome browser just like a human user would, performing actions such as clicking buttons, filling out forms, and navigating through web pages. This makes it an essential component for automated testing of web applications, as it enables developers to test the functionality and performance of their web applications under different conditions. ChromeDriver can be downloaded and installed from the official Chromium project website. Once installed, it needs to be added to the system PATH variable so that the operating system can locate it. To use ChromeDriver with Selenium, developers can create a new instance of the ChromeDriver class in their Python code and pass it the path to the ChromeDriver executable file. This creates a connection between the Python code and the Chrome browser, allowing the developer to automate actions and test their web applications.

CHAPTER 4

METHODOLOGY

4.1 REQUIREMENT ANALYSIS

Requirement analysis determines the requirements of a new system. This project analyses the product and resource requirement, which is required for this successful system. The product requirement includes input and output requirements it gives the wants in term of input to produce the required output. The resource requirements give in brief about the software and hardware that are needed to achieve the required functionality.

4.1.1 Hardware Environment

Hard Disk	:	500GB and Above
RAM	:	4GB and Above
Processor	:	I3 and Above

4.1.2 Software Requirements

- ✓ Operating System : Windows 7 , 8, 10 (64 bit)
- ✓ Software : Python and PyCharm Community Edition
- ✓ Tools : Chrome driver and packages like Pandas, Selenium, BeautifulSoup

4.2 SYSTEM DESIGN

- ✓ Identify the target website: Determine the website from which the data will be scraped.
- ✓ Define the data to be extracted: Identify the data elements and structure that will be extracted from the website.
- ✓ Set up Selenium WebDriver: Configure Selenium WebDriver to open the website in a browser and interact with its elements.
- ✓ Navigate to target pages: Use Selenium to navigate to the target pages that contain the data to be scraped.
- ✓ Extract HTML content: Use BeautifulSoup to extract the HTML content of the target pages.

database.

- ✓ Set up error handling: Implement error handling to deal with common issues that may arise during the scraping process, such as connection errors or page load failures.
- ✓ Implement data processing and analysis: Implement data processing and analysis to clean, transform, and analyze the scraped data.
- ✓ Set up automated scraping: Set up a schedule or trigger for the scraping process to run automatically at regular intervals.

4.3 ARCHITECTURE DIAGRAM

Web Scraping Steps

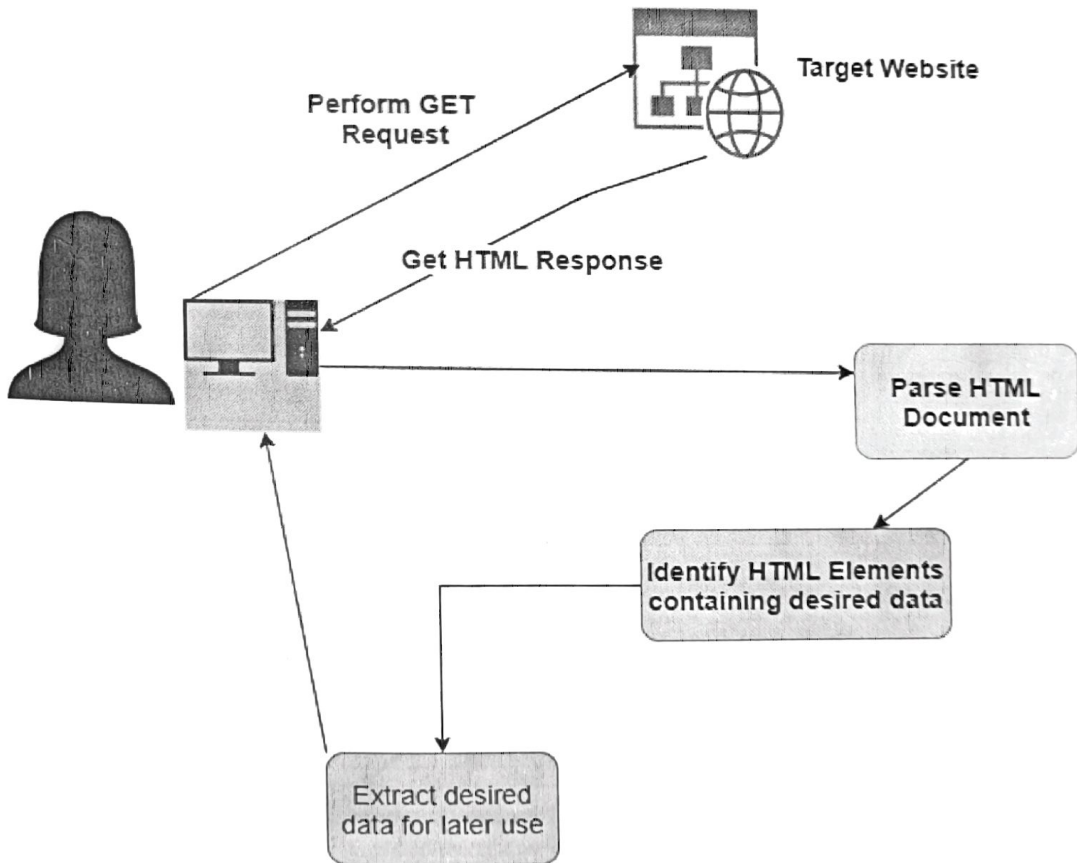


Fig 4.1

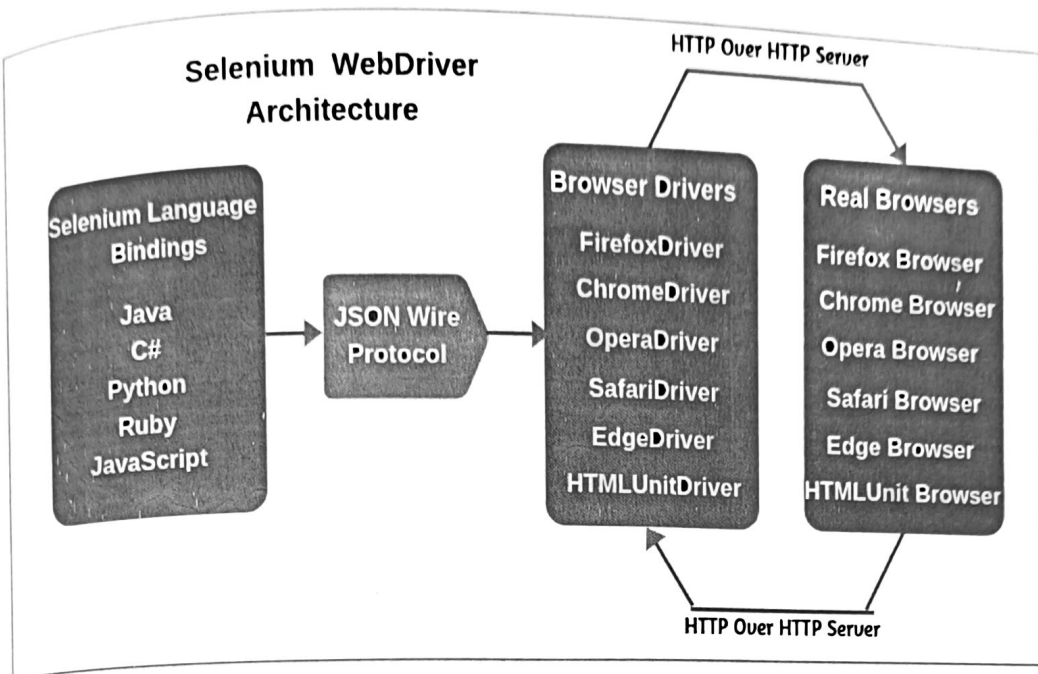


Fig 4.2

4.4 USECASE DIAGRAM



Fig 4.3

4.5 SEQUENTIAL DIAGRAM

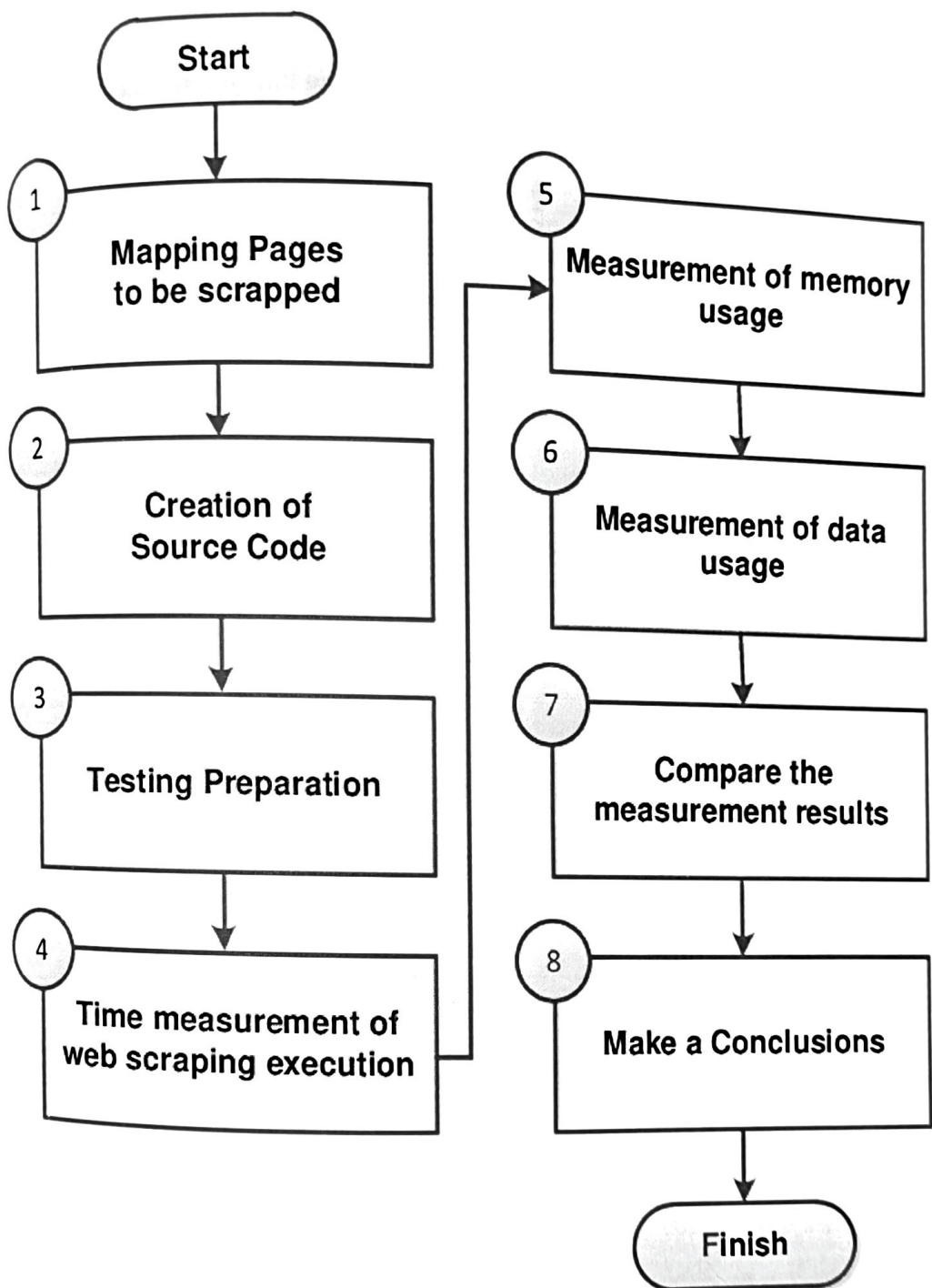


Fig 4.4

4.6 IMPLEMENTATION

✓ Install the necessary packages:

- Python (if not already installed)
- Selenium: **'pip install selenium'**
- BeautifulSoup: **'pip install beautifulsoup4'**
- Pandas: **'pip install pandas'**
- Download the appropriate driver for your browser (e.g., chromedriver for Chrome) and make sure it's in your system path.

✓ Open a browser using Selenium.

✓ Navigate to the webpage you want to scrape.

✓ Extract the HTML content of the webpage using Selenium.

✓ Parse the HTML content using BeautifulSoup.

✓ Find the elements you want to extract using BeautifulSoup.

✓ Extract the data you want from the elements and store it in a pandas DataFrame.

✓ Close the browser.

CHAPTER 5

RESULT AND DISCUSSION

5.1 OUTPUT CSV FILE

The csv file contains various information about flights such as destination city and country, duration of the flight travel, price of the flight ticket, airline company, airport code, depart date, return date, Link for a direct access and minimum price. The python program will generate multiple csv files specific to certain airport. Every csv file will contain information as mentioned above.

5.2 USES OF THE EXTRACTED INFORMATION

The extracted flight ticket information in CSV files can be utilized in various ways. One of the most common uses is for price comparison and analysis. Travel companies and individual travelers can use the data to compare prices across different airlines and travel dates, enabling them to make well-informed decisions about the best deals. Moreover, the information can be utilized for market analysis and forecasting. Airlines and travel agencies can study the prices over time to identify market trends and adjust their pricing strategies accordingly. The extracted data can also be used to predict future prices, aiding travelers in finding the best deals. Another use for the CSV files is to analyze customer behavior and preferences. By examining the search and purchase patterns, travel companies can gain insights into customer preferences, such as preferred travel dates and destinations. This information can help personalize marketing and enhance the overall customer experience. In conclusion, the extracted flight ticket information in CSV files through web scraping is a valuable resource for travel companies, individual travelers, and researchers. The data can help in decision-making, improve pricing strategies, and enhance the overall travel experience.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In conclusion, web scraping can be a powerful tool for extracting information about flights from various websites. Using tools such as Beautiful Soup and Selenium, it is possible to navigate through web pages, extract data, and then store it in a structured format. This data can then be used for various purposes, such as analyzing trends in airfare prices, creating a comparison tool for flights, or providing recommendations for travelers. It is important to note, however, that web scraping may be subject to legal and ethical considerations, so it is important to ensure that you are scraping data in a responsible manner and in compliance with any applicable laws and website terms of service.

6.2 FUTURE WORK

Future studies could focus on improving the accuracy and reliability of web scraping techniques for collecting airfare data. This could involve developing more sophisticated algorithms for data collection and analysis, or using machine learning techniques to improve the accuracy of predictions for future airfare prices. Another possible direction for future research could be to explore the potential ethical and legal issues associated with web scraping in the travel industry. This could involve examining the privacy concerns associated with collecting and analyzing large amounts of personal data from travel websites, or considering the implications of using web scraping to gain a competitive advantage in the industry. Additionally, future studies could explore the potential for integrating web scraping with other data sources, such as social media or customer reviews, to provide a more comprehensive understanding of the travel market. This could involve developing new methods for data integration and analysis, or using natural language processing techniques to analyze text data from multiple sources. Overall, the papers mentioned in this chat provide a foundation for future research exploring the potential applications and limitations of web scraping for flight details, and offer a variety of avenues for further investigation in this area.

APPENDICES

SOURCE CODE

```
import json
from lib2to3.pgen2 import driver
from telnetlib import EC
```

```
import requests
import pandas as pd
import calendar
from datetime import datetime
from time import gmtime, strftime
```

```
from bs4 import BeautifulSoup
from selenium.webdriver.support.wait import WebDriverWait
```

```
def scrape_kayak(start="", end="", airport='OPO'):
```

```
    """
```

This function scrapes flight information from the kayak explore page.

Parameters:

start, end, airport - integer representing earliest possible departure date in YYYYMMDD format, integer representing latest return date, string with three letter code for starting airport. When both are start and end are left blank, results are returned from present date to one year in the future.

Returns:

df - a data frame containing all destination cities and corresponding flight information returned by the scrape

```
    """
```

```
# Format the beginning and end dates to insert them into the URL
# start = '&depart=' + str(start)
# end = '&return=' + str(end)
```

```
#
"https://www.kayak.pt/s/horizon/exploreapi/destinations?airport=OPO&budget=&depart=2023060
l&return=20230630&tripdurationrange=4%2C7&duration=&flightMaxStops=&stopsFilterActive=
false&topRightLat=51.82490080841914&topRightLon=8.962652968749989&bottomLeftLat=28.6
36584579286538&bottomLeftLon=-
26.32543296874999&zoomLevel=5&selectedMarker=&themeCode=&selectedDestination="
```

```
format = "%Y%m%d"
```

```
res = False
```

```
try:
```

```
    res = bool(datetime.strptime(start, format))
```

```
except ValueError:
```

```
    res = False
```

```
if res:
```

```
    # print(res)
```

```
    url = "https://www.kayak.com/s/horizon/exploreapi/destinations?airport=" + airport +
```

```
"&budget=&depart=" + start + "&return=" + end +
```

```
"&tripdurationrange=4%2C7&duration=&flightMaxStops=&stopsFilterActive=false&topRightLat
```

```
=51.82490080841914&topRightLon=8.962652968749989&bottomLeftLat=28.636584579286538
```

```
&bottomLeftLon=-
```

```
26.32543296874999&zoomLevel=5&selectedMarker=&themeCode=&selectedDestination="
```

```
else:
```

```
    url = "https://www.kayak.com/s/horizon/exploreapi/destinations?airport=" + airport +
```

```
"&budget=&tripdurationrange=4%2C7&duration=&flightMaxStops=&stopsFilterActive=false&to
```

```
pRightLat=51.82490080841914&topRightLon=8.962652968749989&bottomLeftLat=28.63658457
```

```
9286538&bottomLeftLon=-
```

```
26.32543296874999&zoomLevel=5&selectedMarker=&themeCode=&selectedDestination="
```

```
response = requests.post(url).json()
```

```
df = pd.DataFrame(
```

```
    columns=['City', 'Country', 'Duration', 'Price', 'Airline', 'Airport', 'Depart', 'Return', 'Link'])
```

```
for i in range(len(response['destinations'])):
```

```

destination = response['destinations'][i]
row = list([destination['city']['name'], destination['country']['name'],
            destination['flightMaxDuration'],
            destination['flightInfo']['price'], destination['airline'],
            destination['airport']['shortName'], pd.to_datetime(destination['departd']).date(),
            pd.to_datetime(destination['returnd']).date(),
            str('http://kayak.com' + destination['clickoutUrl'])])
df.loc[i] = row

```

```

city_mins = df.groupby(['City']).idxmin().astype(int)
df['MinPrice'] = df.loc[city_mins['Price']].to_list().Price
df['is_MinPrice'] = df['Price'].eq(df['MinPrice']).astype(int)
# df = df.loc[city_mins['Price'].to_list()]
# There is a glitch where some flights are returned with unrealistically
# prices, so we'll remove those entries.
df = df.where(df['Price'] != 999999).dropna()

```

```

return df

```

```

# airport='OPO'
# start='20230601'
# end='20230630'
# data = datetime.strptime(start, '%Y%m%d')
# mes = datetime.strptime(start, '%Y%m%d').month
# calendar.month_name[data.month]
# str(data.year)

```

```

origins = ['MAD', 'MXP', 'NAP', 'LIS', 'MAA']
for origin in origins:

```

```

    # df = scrape_kayak(start, end, origin)
    df = scrape_kayak(airport=origin)
    # df.to_csv('data/' + strftime("%Y%m%d%H%M",

```

```

    gmtime())) + '_' + origin + '_' + calendar.month_name[mes] + '_2023' + '.csv', index=False)

```

```

rest_info = []
r = requests.get(URL)
soup = BeautifulSoup(r.content, 'html.parser')
all = soup.find("tbody")
row = all.findAll('tr')
for i in row:
    infos_row = i.findAll('td')
    for index, j in enumerate(infos_row):
        if index == 0:
            attack_number = j.text.replace('\n', "").replace('\t', "").replace('\r', "")
        if index == 1:
            narrations = j.text.replace('\n', "").replace('\t', "").replace('\r', "")
        if index == 2:
            date_of_incident = j.text.replace('\n', "").replace('\t', "").replace('\r', "")
        if index > 2:
            continue
    try:
        rest_info.append([attack_number, narrations, date_of_incident,
datetime.today().strftime('%Y-%m-%d %H:%M')])
    except:
        continue

df_pirates = pd.DataFrame(rest_info, columns=['attack_nr', 'text', 'date_of_incident',
'scrape_date'])
df_pirates['text'] = df_pirates["text"].apply(lambda x: x.split("Posn: ")[1])
df_pirates['location'] = df_pirates["text"].apply(
    lambda x: x.split(",")[1].split(".")[0] if ":" in x.split(",")[0] else x.split(".")[0])
df_pirates["geo_location"] = df_pirates["location"].apply(get_location_coordinates)

if jsonify:
    result = df_pirates.to_json(orient="split")
    parsed = json.loads(result)
    return json.dumps(parsed)
else:

```

```
df_pirates.to_csv("scraped_pirates.csv", mode='a', index=False, header=False)  
return
```

```
scrapeWeather()  
scrapePirates()
```

OUTPUT SCREENSHOT

When the python program gets executed, it will create different csv files for each airport and the files will contain specific airport code in its name (MAA is the code for Chennai airport)

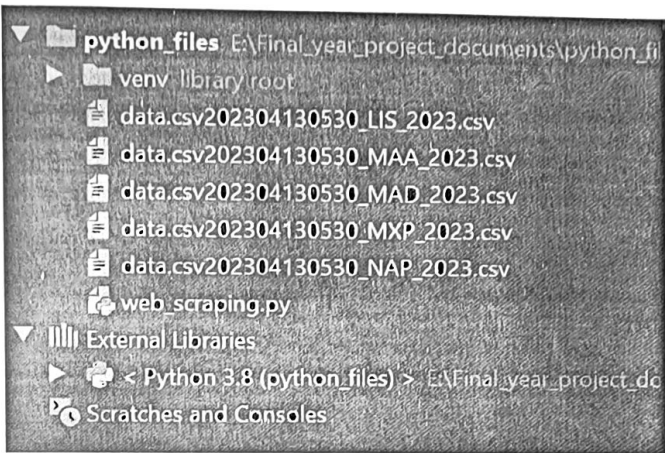


Fig 7.1

The files will contain information in the format mentioned in the screenshot.

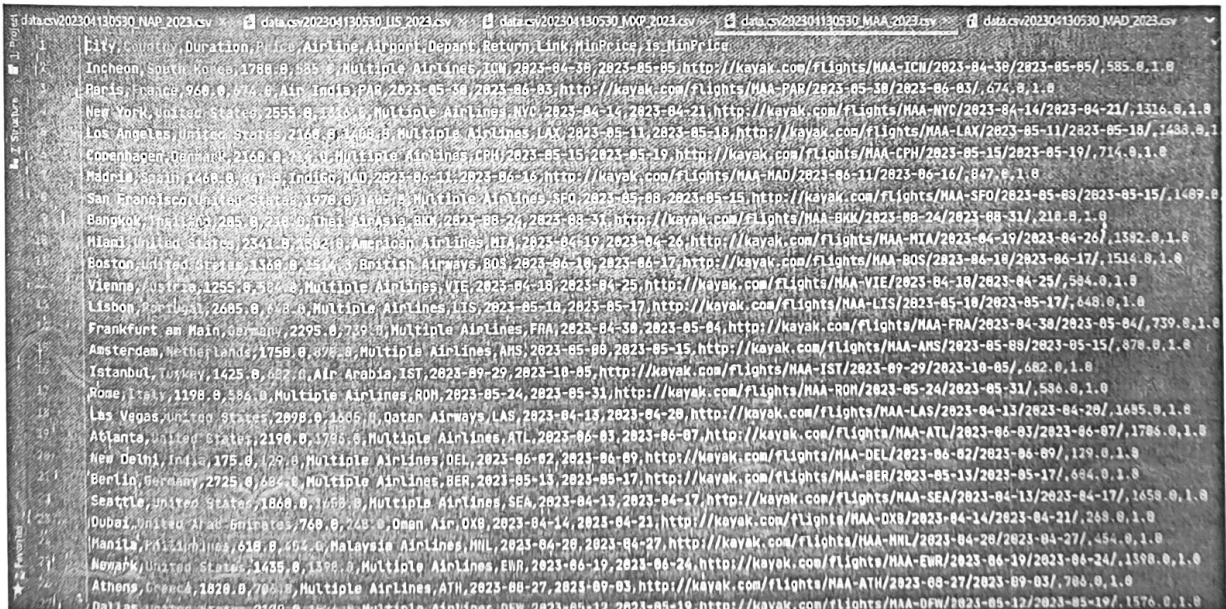


Fig 7.2

These csv files can be visualized in a structured way using Microsoft Excel.

City											
A	B	C	D	E	F	G	H	I	J	K	L
City	Country	Duration	Price	Airline	Airport	Depart	Return	Link	MinPrice	Is_MinPrice	
Incheon	South Korea	1780	585	Multiple Airlines	ICN	4/30/2023	5/5/2023	http://kay	585	1	
Paris	France	960	674	Air India	PAR	5/30/2023	6/3/2023	http://kay	674	1	
New York	United States	2555	1316	Multiple Airlines	NYC	4/14/2023	4/21/2023	http://kay	1316	1	
Los Angeles	United States	2160	1488	Multiple Airlines	LAX	5/11/2023	5/18/2023	http://kay	1488	1	
Copenhagen	Denmark	2160	714	Multiple Airlines	CPH	5/15/2023	5/19/2023	http://kay	714	1	
Madrid	Spain	1460	847	IndiGo	MAD	6/11/2023	6/16/2023	http://kay	847	1	
San Francisco	United States	1970	1409	Multiple Airlines	SFO	5/8/2023	5/15/2023	http://kay	1409	1	
Bangkok	Thailand	205	210	Thai AirAsia	BKK	8/24/2023	8/31/2023	http://kay	210	1	
Miami	United States	2341	1302	American Airline	MIA	4/19/2023	4/26/2023	http://kay	1302	1	
Boston	United States	1360	1514	British Airways	BOS	6/10/2023	6/17/2023	http://kay	1514	1	
Vienna	Austria	1255	504	Multiple Airlines	VIE	4/18/2023	4/25/2023	http://kay	504	1	
Lisbon	Portugal	2605	648	Multiple Airlines	LIS	5/10/2023	5/17/2023	http://kay	648	1	
Frankfurt am Main	Germany	2295	739	Multiple Airlines	FRA	4/30/2023	5/4/2023	http://kay	739	1	
Amsterdam	Netherlands	1750	870	Multiple Airlines	AMS	5/8/2023	5/15/2023	http://kay	870	1	
Istanbul	Turkey	1425	602	Air Arabia	IST	9/29/2023	10/5/2023	http://kay	602	1	
Rome	Italy	1190	586	Multiple Airlines	ROM	5/24/2023	5/31/2023	http://kay	586	1	
Las Vegas	United States	2098	1605	Qatar Airways	LAS	4/13/2023	4/20/2023	http://kay	1605	1	
Atlanta	United States	2190	1706	Multiple Airlines	ATL	6/3/2023	6/7/2023	http://kay	1706	1	
New Delhi	India	175	129	Multiple Airlines	DEL	6/2/2023	6/9/2023	http://kay	129	1	
Berlin	Germany	2725	604	Multiple Airlines	BER	5/13/2023	5/17/2023	http://kay	604	1	
Seattle	United States	1860	1650	Multiple Airlines	SEA	4/13/2023	4/17/2023	http://kay	1650	1	
Dubai	United Arab Emira	760	268	Oman Air	DXB	4/14/2023	4/21/2023	http://kay	268	1	
Manila	Philippines	610	454	Malaysia Airlines	MNL	4/20/2023	4/27/2023	http://kay	454	1	
Newark	United States	1435	1398	Multiple Airlines	EWB	6/19/2023	6/24/2023	http://kay	1398	1	
Athens	Greece	1820	706	Multiple Airlines	ATH	8/27/2023	9/3/2023	http://kay	706	1	
Dallas	United States	2179	1576	Multiple Airlines	DFW	5/12/2023	5/19/2023	http://kay	1576	1	

Fig 7.3

REFERENCES

- [1] Damian Trilling and Natali Helberger. "Web Scraping: A Large-Scale Study of Techniques and Tools." Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations, 2019, pp. 42-47.
- [2] Gregory Saxton. "Python for Large-Scale Web Scraping." Journal of Information Technology and Politics, vol. 16, no. 2, 2019, pp. 152-164.
- [3] Jingjing Li and Yuanting Zhang. "Web Scraping and Crawling Techniques for Monitoring Online Price Changes." Journal of Business Research, vol. 132, 2021, pp. 1-12.
- [4] Yan Long. "Web Scraping with Python and BeautifulSoup for Social Science Research." Journal of Open Research Software, vol. 8, no. 1, 2020.
- [5] Adriana Rios-Ramirez and Elvira Castillo-Gallegos. "Web Scraping and Data Analysis with Python for Business Applications." International Journal of Computational Intelligence and Applications, vol. 19, no. 2, 2020, pp. 1-19.
- [6] Steven Altham. "Web Scraping with Python: A Practical Guide." Apress, 2018.
- [7] Katie McLaughlin. "Web Scraping with Python and Beautiful Soup." O'Reilly Media, 2019.
- [8] Seppe vanden Broucke and Bart Baesens. "Practical Web Scraping for Data Science: BestPractices and Examples with Python." Apress, 2019.
- [9] Alexander Obinna Okafor. "Web Scraping with Python." Packt Publishing, 2020.