Delhi Technological University

Department of Information Technology

IT 406: Web Technology

Project

Web Scrapping on MakeMyTrip and Paytm using Python and Selenium

Submitted By:

Yahya Bare Hajon(2K19/IT/145)

Ezatullah Amin(2K19/IT/047)

Submitted To: Ms Priyanka Meel

ABSTRACT

From the advancement of the World Wide Web, the situation of the web client and information trade has vastly changed. As average citizens join the web and begin to utilize it, heaps of new systems are elevated to help up the system. Simultaneously to improve PCs and system office, new innovations were brought which results in a consequent reduction in the cost of equipment and site's connected expenses. Business, academician, scientists all share their data on the web with the goal that they can be associated with individuals vastly with no problem at all. Because of exchange, share, and storage options for the information on the web, another issue emerged that how to deal with such information overload and how the client will get or get to the best data in the least endeavours. To settle these issues, specialist spot out a new procedure called Web Scraping. Web scraping, or web scratching, is a procedure which is utilized to create organized information based on accessible unstructured information on the web. Created organized information at that point is then put away in focal database to be dissected in spreadsheets. Presently, there are heaps of tools accessible in the market for web scratching.

Web scraping is a very popular and growing field. The presence of web scraping in travel related websites is gigantic. The advantages and difficulties of web information scrapping are significant to everybody in the travel business. Meta look locales, OTAs and travel providers are altogether being rejected and the most fascinating truth is that more than 30% of individuals who visit travel industry sites are web scrappers. Through a generalised web scraping mechanism using selenium web driver people can easily scrap the website data without even opening the browser. Selenium is one of the most popular tools used nowadays to scrap the websites. Today, there are more advanced web data scrapping techniques and skills that enable you to visit the websites and get the data comfortably and easily (as mentioned above selenium is one of those web drivers). There is also software that scraps the web automatically and records the data. For a very long time now, Bing has indexed travel suppliers. It has been also able to provide a direct inventory links to the resulting search results. The same route has been followed by Google as well. Fraud bot, which is bad, takes advantage of your system by looking at the vulnerable areas. These are those expensive hackers that fool websites by clicking ads, filling forms and look for other weaknesses in website.

This project is centred around the web scrapping of MakeMyTrip and Paytm websites using data extraction method and implemented using python.

INTRODUCTION

Web scraping is a most general and widely used field in the current day scenario. Mostly, web scraping in the current day scenario is done in order to save a lot of time. As a matter of fact, it does save a huge amount of time. In our project we attempt to scrap some relevant travel related websites such as makemytrip.com, yatra.com, goibibo.com etc using he corresponding classes in order to extract the website data. The scraping tool used to scrap the website data is selenium web driver. The user interface we make is using the tkinter library, which is the most popular and commonly used python library. It provides a robust and highly profound method in order to make user interfaces. According to our implementation the user will enter the flight details and get a sorted output based on the searched query for the flight availability.

In Web Scraping or Web Harvesting is a software technology aims at extracting information from websites. Web scraping typically simulate human exploring of the World Wide Web by creating a low-level Hyper Text Transfer Protocol or implementing a Suitable Web Browser. It is closely related to Web Indexing, an information extracting technique used by multiple search engines to index-data on the Web using human programmed bots. In comparison, web scraping stresses on transforming unstructured information (usually in HTML format) on the web into structured information that can be saved and processed in a centralised database. Web scraping is closely related to web automation as well, which uses computer software to simulate Internet browsing by a computer. Web scraping is mostly used for price comparison online, webpage interface change detection, weather forecast information, web information integration, webpages mix ups or mashups, and web surveys. Currently, there are multiple software gadgets available that aim to apply scraping techniques to personalize your website.

In Information Extraction, textual content extracting is used to scrap applicable facts outside of textual content documents via way of means of counting on language related and statistical algorithms. Web seeks and facts extraction is normally accomplished via way of means of extracting tools Web crawlers. A Web crawler is an application automatic code that browses the World Wide Web (WWW) in a methodical and automatic manner.

This project is aimed to build a web scrapper to retrieve the data from the websites MakeMyTrip and Paytm. The web scrapper will be developed using the Python programming language, along with the libraries Tkinter and Selenium. The web scrapper will be used to gather data from the two websites and store them in an organized manner for easier access and usage. The data collected from the websites will be used for various purposes such as getting the latest travel deals, comparing costs of flights and hotels, and getting the latest rates of different products. The web scrapper will be able to retrieve data from both the websites and store them in an organized manner. This will help in making the data more accessible and user-friendly.

PROBLEM STATEMENT

To generalise a web scraping mechanism for travel related websites. A popular demonstration of the capability of automating websites using selenium drivers. It is used for automating web applications for testing purposes, specifically for travel related websites in this problem. There is need to provide a successful and a user-friendly interface to users so that they can easily without even opening the website be able to get the flight details and make a comparison between the flight fares and book a good flight accordingly to their needs and requirements. This saves a huge amount of time for the customer as there is no need for the customer to visit different travel related websites and check for the corresponding flight fares and details, instead of that a single query inputted by the user would him/her a lot in order to view a broad range of categories.

This web scraping mechanism should be able to extract the necessary information from various travel websites such as flight fares, flight details, etc. The system should also be able to compare the flight fares of different websites and provide an overall view of the lowest fares available to the customers. Additionally, the system should be able to book the flight tickets automatically, without the user having to visit the particular website. The system should also be able to provide the customers with information about the available discounts and offers from each of the different travel websites. Furthermore, the system should also be able to provide the customers with reviews and ratings of the different travel websites, so that they can make an informed decision on which website to book their flight from.

In our project, the data can be scraped from the website using selenium web driver and stored in the database. The web scraper can be used to retrieve the data from the website and store it in the database. The data can be then used to display the information to the user in a sorted manner. The web scraper can be used to automate the process of extracting data from the website. This can be done by creating a script that will fetch the data from the website and store it in the database. The data can then be used to generate reports and display the information to the user. The web scraper can also be used to detect any changes in the website content and update the data accordingly. This will help in keeping the data up to date and accurate

BASIC CONCEPTS/ TECHNOLOGY USED

This project utilizes many concepts and is been made with the help of various tools and technologies. The required resource analysis is as follows;

Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

JUPYTER Lab - Project Jupyter is a non-profit organization created to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

Web Scraping (also termed Screen Scraping, Web Data Extraction, Web Harvesting etc.) is a concept utilized to gather information from sites whereby the information is extricated and spared to a record in your PC or to a database in table (spreadsheet) design [2]. Below, some systems similar to the current project are mentioned in order to highlight the contrast between the current project and these mentioned software tools. Import.io is an online instrument for extracting information from a site without composing code. In the event when the client needs a quick outcome, at that point he will go after this approach with the goal that he can change data of the site in a brief timeframe. For extricating information, the client enters URL and the application naturally gets the information which client needs if the client doesn't keen on the programmed extraction, the point and click interface helps to select data fields on the site [1]. As the information extraction is finished, the separated dataset is stored on Import.io cloud server and further downloaded in CSV, Exceed expectations, JSON design.

TKINTER

Tkinter, a popular and a standard Graphical user Interface library for the python. When tkinter library is combined or integrated with python it provides a very quick and simple mechanism to create a Graphical User Interface Application. It provides a highly powerful object-oriented interface to the GUI toolkit of Tk.

Tkinter provides a number of different GUI toolkits such as buttons, labels, frames, menus, and so on. It also provides some basic widgets like text boxes and check boxes to help you build a complete GUI application. Tkinter can also be used to create graphical user interfaces for applications that run on multiple platforms, such as Windows, Linux, and Mac OS X. There are many advantages to using Tkinter. It is easy to learn and use, and you can quickly create a basic application. It is also platform independent, meaning that you can write your code once and it will run on any supported platform. Additionally, it is open source, so it is free to use and modify. Finally, it is very well documented, so you can find answers to any questions you have.

The mechanism for creating a simple GUI application on tkinter.

a. Import the module tkinter.

b. Create a Graphical user interface application main window.

c. One can add the discussed modules or widgets to the application GUI.

d. Then one has to enter an event loop so that a suitable action against the event can be taken which is triggered by the user.

Functionalities provided by tkinter: -

1. It provides classes for displaying and controlling the widgets.

2. Properties of widgets are specified using with keyword arguments.

3. Keyword arguments have the same name as the corresponding resource under Tk.

4. Top level widgets are Tk.

5. Widgets are being positioned with managers of geometry like Place, Pack or Grid. Methods such as place, pack and grid are called by these managers.

SELENIUM

Selenium is the tool used in order to automate testing of web applications. It is most commonly known as Selenium 2.0. This web driver uses an underlying framework which is very different when compared to selenium RC, as selenium RC uses a JavaScript embedded selenium core which has some number of considerable limitations. This selenium web driver interacts with the web browser directly without the use of any sort of intermediary, whereas selenium RC has a limitation as it is completely dependent on the server.

Selenium provides a platform for automating test cases. It allows the user to write scripts in different programming languages like Java, C#, Python, Ruby and JavaScript and then use them to execute automation tests. Selenium is used to automate web applications, desktop applications and mobile applications. It can be used to test the UI of applications, web services, and APIs. It also supports cross-browser testing, meaning that it can be used to test applications across different browsers.

The context in which selenium web driver comes into action is as follows:

1. It has the ability to handle multiple frames, popups, multi browser window, and some alerts.

2. Ability to navigate complex web pages.

3. Some advances navigation techniques such as drag and drop.

4. AJAX based User Interface elements.

5. It is also used in the multiple browser testing including an improved functionality for the browsers which is not at all well – supported by selenium.

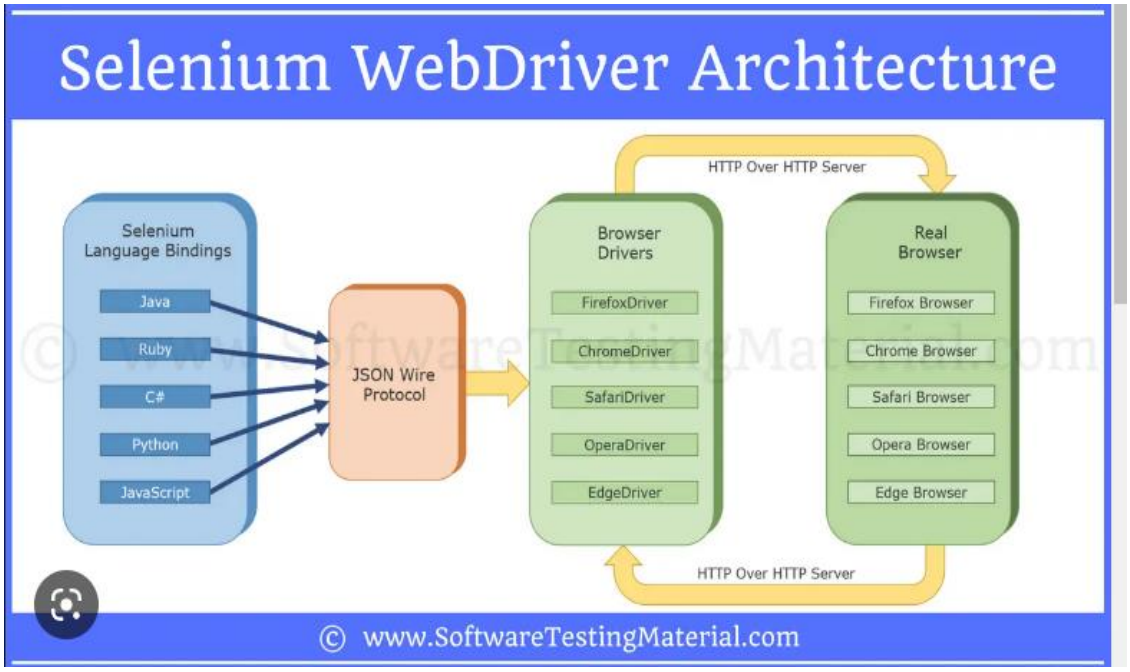


Fig 1. architecture of selenium web driver

METHODOLOGY/ MODEL TOOL

The proposed work centres around dissecting the website pages (HTML code). Right now, have built up working model. By utilizing this procedure weblink change into visual blocks. A visual block is actually section of web page. The framework is programmed top-down and deals to recognize web content structure. Basically, the block-based page content structure is obtained by using a python script in Beautiful Soup in order to further save it as a CSV file. Simulation of experimental work shown below

A. Installation of Beautiful Soup and Requests

B. Python scripting

C. Execution of the python code

D. Content structure construction

E. Saving it as a CSV file

Requirements

- Python 3.5 or higher
- Selenium library
- Tkinter library
- Google Chrome web browser
- Google Chrome Driver

Design

The web scrapper will be built using the Python programming language. The libraries used for the development of the web scrapper will be Tkinter and Selenium. The web scrapper will be used to collect data from the websites MakeMyTrip and Paytm. The web scrapper will be able to collect data from both the websites and store them in an organized manner. The web scrapper will use the Selenium library to retrieve data from the websites. The Selenium library will be used to automate the process of retrieving data from the websites. The web scrapper will be able to retrieve data from both the websites and store them in an organized manner. The data collected from the websites will be used for various purposes such as getting the latest travel deals, comparing costs of flights and hotels, and getting the latest rates of different products. The web scrapper will also use the Tkinter library to create a graphical user interface (GUI). The GUI will be used to provide a user-friendly interface to the web scrapper. The GUI will be used to provide users with options such as selecting the websites to scrape data from, selecting the type of data to be retrieved, and selecting the output format of the data. The GUI will also provide users with options to customize the data retrieved from the websites.

Implementation

The web scrapper will be implemented using the Python programming language. The libraries used for the development of the web scrapper will be Tkinter and Selenium. The web scrapper will be used to collect data from the websites MakeMyTrip and Paytm. The web scrapper will use the Selenium library to retrieve data from the websites. The Selenium library will be used to automate the process of retrieving data from the websites. The web scrapper will also use the Tkinter library to create a graphical user interface (GUI). The GUI will be used to provide a user-friendly interface to the web scrapper. The GUI will be used to provide users with options such as selecting the websites to scrape data from, selecting the type of data to be retrieved, and selecting the output format of the data. The GUI will also provide users with options to customize the data retrieved from the websites. Once the web scrapper is developed and tested, it can be deployed into a production environment. The web scrapper can then be used to collect data from the websites MakeMyTrip and Paytm. The data collected from the websites can then be used for various purposes such as getting the latest travel deals, comparing costs of flights and hotels, and getting the latest rates of different products.

The process of web scraping involves the following steps:

1. Selecting the target website for scraping: Firstly, we need to select the target website for scraping. In our project, we have chosen to scrape the travel websites such as makemytrip.com, yatra.com, goibibo.com etc.

2. Gathering the necessary data from the target website: Once the target website is selected, the next step is to gather the necessary data from the target website. We use the selenium web driver for this purpose. We first use the web driver to navigate to the website and then use the appropriate classes to extract the necessary data.

3. Processing the extracted data: Once the necessary data is extracted, the next step is to process the data. We use the python library tkinter to create a user interface to process the extracted data. The user interface allows the user to enter the search query and get the sorted output based on the searched query.

4. Storing the processed data: The last step is to store the processed data. We use the python library pandas to store the processed data. The panda's library provides a convenient way to store the processed data.

CODE

```
from tkinter import *
import csv


def create_standard_table(f, window):
    handle = csv.reader(f)
    length = len(next(handle))
    lengths = [0] * length
    for record in handle:

        for p,column in enumerate(record):
            if len(column) > lengths[p]:
                lengths[p] = len(column)+3

    f.seek(0)
    trow = 0
    table = Frame(window)
    for record in handle:
        for w,column in enumerate(record):
            Label(table,text=column,width=lengths[w],borderwidth=2,relief="groove",justify=LEFT,anchor=W, background='white'

        trow+=1

    return table

```

```
# Importing Libraries - Tkinter, CSV, numpy and pandas
from tkinter import *
from tkinter import Tk
from tkinter import filedialog
import csv
from tables import createStandardTable as cst
import numpy as np
import pandas as pd
from selenium import webdriver

# Declaring Variables of list datastructure for sorting.
pr, tim, air, depart, arrival, source, destination, duration, stopage, prgo = [], [], [], [], [], [], [], [], [], []
timgo, airgo, sourcegp, destgo, dash, flightnumgo, airname = [], [], [], [], [], [], []
departgo, arrivego, airname, flynum, finalgo, stopage2, final, airname1, flight1 = [], [], [], [], [], [], [], [], []

# Creating TK() object
window = Tk()

# Declaring all the functions# Functions for taking inputs
def inputFrom():
  mText = fromCity.get()# mlabel1 = Label(tableFrame3, text = mText).grid()
def inputTo():
  mText = toCity.get()# mlabel1 = Label(tableFrame3, text = mText).grid()
def inputDepDate():
  mText = departDate.get()# mlabel1 = Label(tableFrame3, text = mText).grid()

def inputDepMon():
  mText = departMonth.get()# mlabel1 = Label(tableFrame3, text = mText).grid()
def mAbout():
  messagebox.showinfo(title = "About", message = "This is the about box")
def mQuit():
  mExit = messagebox.askyesno(title = "Quit", message = "Are you sure")
if mExit > 0:
  window.destroy()
```

```
# Functions For Sorting
def price_sort_increase(final):
  s = sorted(final, key = lambda x: x[7], reverse = False)
return s
def price_sort_decrease(final):
  s = sorted(final, key = lambda x: x[7], reverse = True)
return s
def sort_durationInc(flist):
  s = sorted(flist, key = lambda x: int(str(x[5][0]) + str(x[5][3]) + str(x[5][4])), reverse = False)
return s
def sort_durationDec(flist):
  s = sorted(flist, key = lambda x: int(str(x[5][0]) + str(x[5][3]) + str(x[5][4])), reverse = True)
return s

MMT = ["HYD", "MAA", "BOM", "DEL", "CCU", "AMD", "BLR", "NAG", "CCJ", "PNQ"]
Ptm = ["HYD-Hyderabad", "MAA-Chennai", "BOM-Mumbai", "DEL-Delhi", "CCU-Kolkata", "AMD-Ahmedabad", "BLR-Bengaluru", "NAG-Nagpu

# Main Function where all the process happens
def run1():
  mtext1 = fromCity.get()
mtext2 = toCity.get()
mtext3 = departDate.get()
mtext4 = departMonth.get()
ptext1 = Ptm[MMT.index(mtext1)]
ptext2 = Ptm[MMT.index(mtext2)]
url = "https://flights.makemytrip.com/makemytrip/search/O/O/E/1/0/0/S/V0/" + mtext1 + "_" + mtext2 + "_" + mtext3 + "-" + mte
url2 = "https://paytm.com/flights/flightSearch/" + ptext1 + "/" + ptext2 + "/1/0/0/E/2017-" + mtext4 + "-" + mtext3
chrome_path = "C:\\Users\\Lenovo\\Downloads\\Cyber2023\\Chromedriver\\chromedriver_win32"
driver = webdriver.Chrome(chrome_path)
driver2 = webdriver.Chrome(chrome_path)
driver.get(url)
driver2.get(url2)
```

```
## Getting data for make my trip
airline_info = driver.find_elements_by_class_name("airline_info_detls")
price = driver.find_elements_by_class_name("num")
time = driver.find_elements_by_class_name("time_info")
for post in price:
  price = post.text
price = price.replace(',', '')
price = int(price)
pr.append(price)
dash.append("-")
for post in airline_info:
  airline = post.text
airline = airline.replace('\n', ' ')
air.append(airline)
for post in time:
  t = post.text
tim.append(post.text)
conarray = [i + "\n" + j + "\n" + k
  for i, j, k in zip(tim[::3], tim[1::3], tim[2::3])
]
for element in conarray:
  parts = element.split('\n')
depart.append(parts[0])
source.append(parts[1])
arrival.append(parts[2])
destination.append(parts[3])
duration.append(parts[4])
stopage.append(parts[5])
for k in range(0, len(pr)):
  final.append([air[k], source[k], depart[k], destination[k], arrival[k], duration[k], stopage[k], pr[k], dash[k]])
newlist = sorted(final, key = lambda x: x[7], reverse = False)
```

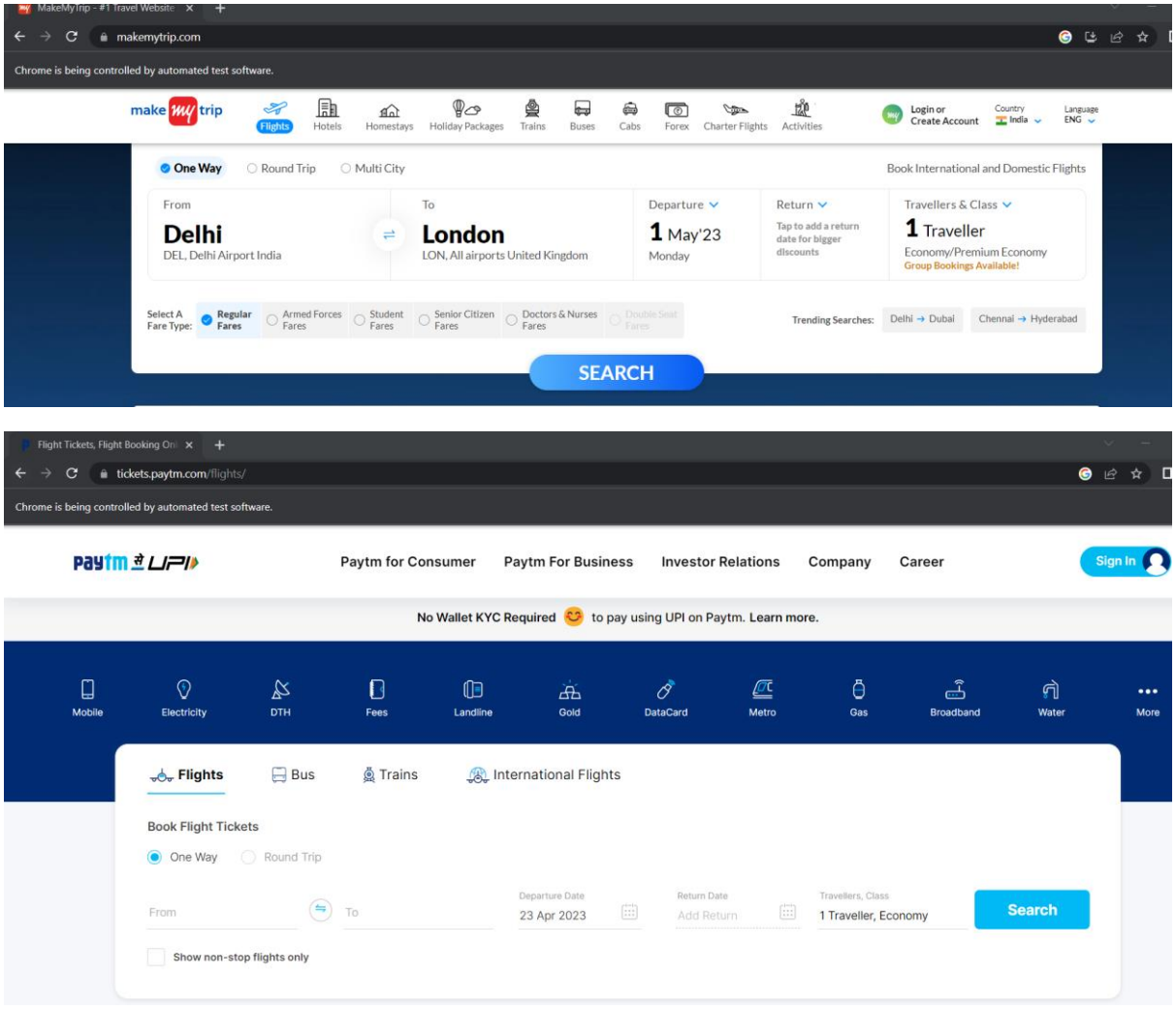
```
## Getting data for Paytm

airline_info2 = driver2.find_elements_by_class_name("_3H-S")
price2 = driver2.find_elements_by_class_name("_2gMo")
fly = driver2.find_elements_by_class_name("NqXj")
time2 = driver2.find_elements_by_class_name("vY4t")
stopgo = driver2.find_elements_by_class_name("_7BOG")
for post in price2:
  pri = post.text
pri = pri.replace(',', '')
pri = int(pri)
prgo.append(pri)
for post in stopgo:
  stopage2.append(post.text)
for post in airline_info2:
  airgo.append(post.text)
for post in time2:
  timgo.append(post.text)
for post in fly:
  flynum.append(post.text)

airname = airgo[::3]
departgo = airgo[1::3]
arrivego = airgo[2::3]
flightnumgo = flynum[::4]
flightnumgo = [w.replace(' ', '') for w in flightnumgo]
sourcego = flynum[1::4]
destgo = flynum[2::4]
```

```
conarray2 = [i + " " + j
  for i, j in zip(airname, flightnumgo)
]
for i in range(0, len(prgo)):
  finalgo.append([conarray2[i], sourcego[i], departgo[i], destgo[i], arrivego[i], timgo[i], stopage2[i], prgo[i]])
newlist2 = sorted(finalgo, key = lambda x: x[7], reverse = False)

## Checking and storing price for flights in Paytm into the array
for j in range(0, len(newlist)):
  for i in range(0, len(newlist2)):
    if (newlist2[i][0] == newlist[j][0]):
      newlist[j][8] = (newlist2[i][7])
      break

  ## Exporting all the values into individual CSV files
finallist = newlist[: 15]

my_df1 = pd.DataFrame(finallist)
my_df1.to_csv('finallist.csv', index = False, header = False)

price_sort_increase1 = price_sort_increase(finallist)

my_df2 = pd.DataFrame(price_sort_increase1)
my_df2.to_csv('priceSortIncrease.csv', index = False, header = False)

price_sort_decrease1 = price_sort_decrease(finallist)

my_df3 = pd.DataFrame(price_sort_decrease1)
my_df3.to_csv('priceSortDecrease.csv', index = False, header = False)

sortDurationInc1 = sort_durationInc(finallist)

my_df4 = pd.DataFrame(sortDurationInc1)
my_df4.to_csv('sort_durationInc.csv', index = False, header = False)
```

```
sortDurationDec1 = sort_durationDec(finallist)

my_df5 = pd.DataFrame(sortDurationDec1)
my_df5.to_csv('sort_durationDec.csv', index = False, header = False)

## Creating Frames for input, Sorting buttons and output
tableFrame = Frame(window, bg = 'cyan', width = 450, height = 50, pady = 10)
tableFrame2 = Frame(window, bg = 'gray2', width = 450, height = 40, padx = 50, pady = 20)
tableFrame3 = Frame(window, bg = 'red', width = 450, height = 300, padx = 50, pady = 20)

# Creating the Variables for Inputs
fromCity = StringVar()
toCity = StringVar()
departDate = StringVar()
departMonth = StringVar()

# Declaring the title for the window
window.title('Web Mining - WEB SCRAPING FOR MAKE-MY-TRIP and PAYTM')

def tableFrames():
  tableFrame3.grid()
tableFrame2.grid()
tableFrame.grid()

# create a global newtable grid.
f = open("my_csv.csv")
newtable = cst(f, tableFrame)

def createTableFrame():
  f = open("sort_durationDec.csv")
global newtable
newtable = cst(f, tableFrame)
newtable.grid()

```

```
def Header():
  f = open("my_csv.csv")
global newtable
newtable = cst(f, tableFrame)
newtable.grid()

def createTableFrame2():
  f = open("my_csv.csv")
global newtable
newtable = cst(f, tableFrame)
newtable.grid()

def createTableFrame3():
  f = open("priceSortIncrease.csv")
global newtable
newtable = cst(f, tableFrame)
newtable.grid()

def createTableFrame4():
  f = open("priceSortDecrease.csv")
global newtable
newtable = cst(f, tableFrame)
newtable.grid()

def createTableFrame5():
  f = open("sort_durationInc.csv")
global newtable
newtable = cst(f, tableFrame)
newtable.grid()

def ct():
  global newtable
newtable.destroy()
createTableFrame3()
```

```
def th():
  global newtable
newtable.destroy()
createTableFrame()
tableFrame.grid()

def t2h():
  global newtable
newtable.destroy()
createTableFrame4()
tableFrame.grid()

def t3h():
  global newtable
newtable.destroy()
createTableFrame5()
tableFrame.grid()

def sorting():
  Button(tableFrame2, text = "Show Price Sort Increase", command = ct).grid(row = 0, column = 0, padx = 15, pady = 10)
Button(tableFrame2, text = "Show Price Sort Decrease", command = t2h).grid(row = 0, column = 1, padx = 15, pady = 10)
Button(tableFrame2, text = "Show Sort Duration Increase", command = t3h).grid(row = 0, column = 2, padx = 15, pady = 10)
Button(tableFrame2, text = "Show Sort Duration Decrease", command = th).grid(row = 0, column = 3, padx = 15, pady = 10)

# Input
mLabel1 = Label(tableFrame3, text = 'FROM')
mLabel1.grid(row = 0, column = 0, sticky = W)
mEntry1 = Entry(tableFrame3, textvariable = fromCity).grid(row = 0, column = 1, padx = 20, pady = 10)
mButton1 = Button(tableFrame3, text = 'OK', command = inputFrom).grid(row = 0, column = 2)

mLabel2 = Label(tableFrame3, text = 'TO').grid(row = 1, column = 0, sticky = W)
mEntry2 = Entry(tableFrame3, textvariable = toCity).grid(row = 1, column = 1, pady = 10)
mButton2 = Button(tableFrame3, text = 'OK', command = inputTo).grid(row = 1, column = 2)

mLabel3 = Label(tableFrame3, text = 'DATE').grid(row = 2, column = 0, sticky = W)
mEntry3 = Entry(tableFrame3, textvariable = departDate).grid(row = 2, column = 1, pady = 10)
mButton3 = Button(tableFrame3, text = 'OK', command = inputDepDate).grid(row = 2, column = 2)

mLabel4 = Label(tableFrame3, text = 'MONTH').grid(row = 3, column = 0, sticky = W)
mEntry4 = Entry(tableFrame3, textvariable = departMonth).grid(row = 3, column = 1, pady = 10)
mButton4 = Button(tableFrame3, text = 'OK', command = inputDepMon).grid(row = 3, column = 2)

mRun = Button(tableFrame3, text = 'RUN', command = run1).grid(row = 4, column = 0, sticky = W)

tableFrames()
sorting()
Header()
createTableFrame2()

window.mainloop()
```



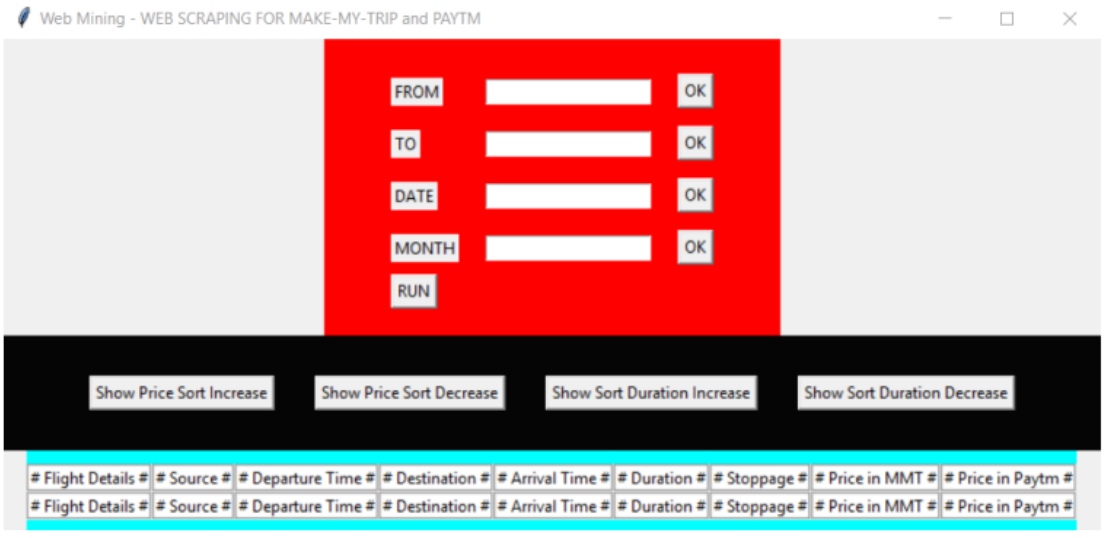
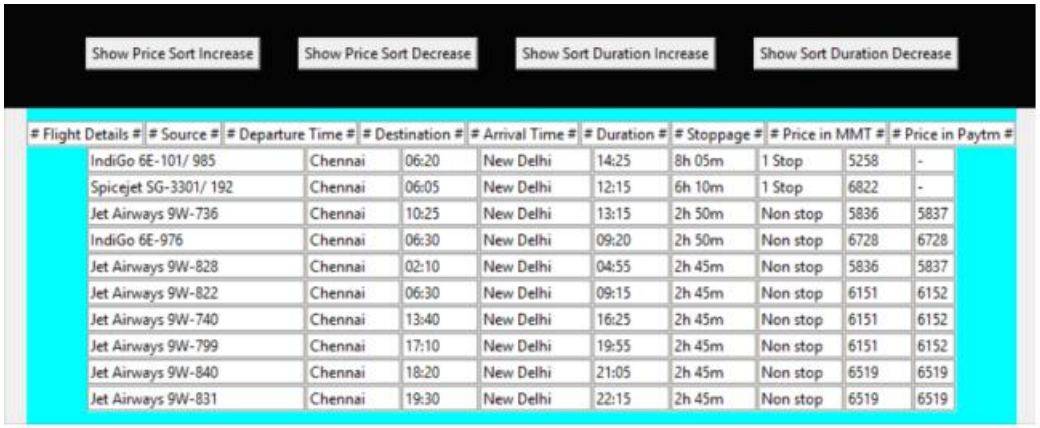
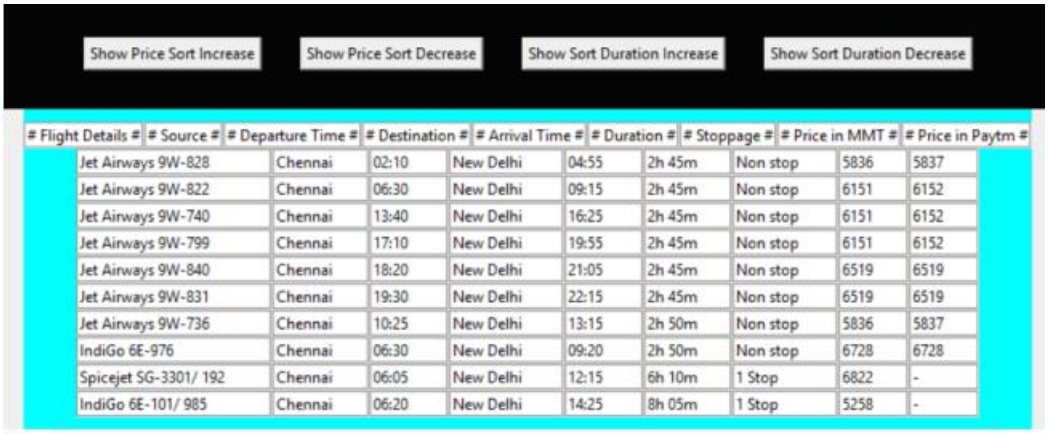


Figure 1 - Main Screen – Dashboard



# Flight Details #	# Source #	# Departure Time #	# Destination #	# Arrival Time #	# Duration #	# Stoppage #	# Price in MMT #	# Price in Paytm #
IndiGo 6E-101/ 985	Chennai	06:20	New Delhi	14:25	8h 05m	1 Stop	5258	-
Spicejet SG-3301/ 192	Chennai	06:05	New Delhi	12:15	6h 10m	1 Stop	6822	-
Jet Airways 9W-736	Chennai	10:25	New Delhi	13:15	2h 50m	Non stop	5836	5837
IndiGo 6E-976	Chennai	06:30	New Delhi	09:20	2h 50m	Non stop	6728	6728
Jet Airways 9W-828	Chennai	02:10	New Delhi	04:55	2h 45m	Non stop	5836	5837
Jet Airways 9W-822	Chennai	06:30	New Delhi	09:15	2h 45m	Non stop	6151	6152
Jet Airways 9W-740	Chennai	13:40	New Delhi	16:25	2h 45m	Non stop	6151	6152
Jet Airways 9W-799	Chennai	17:10	New Delhi	19:55	2h 45m	Non stop	6151	6152
Jet Airways 9W-840	Chennai	18:20	New Delhi	21:05	2h 45m	Non stop	6519	6519
Jet Airways 9W-831	Chennai	19:30	New Delhi	22:15	2h 45m	Non stop	6519	6519

Figure 8 - Price Sort Duration Decrease

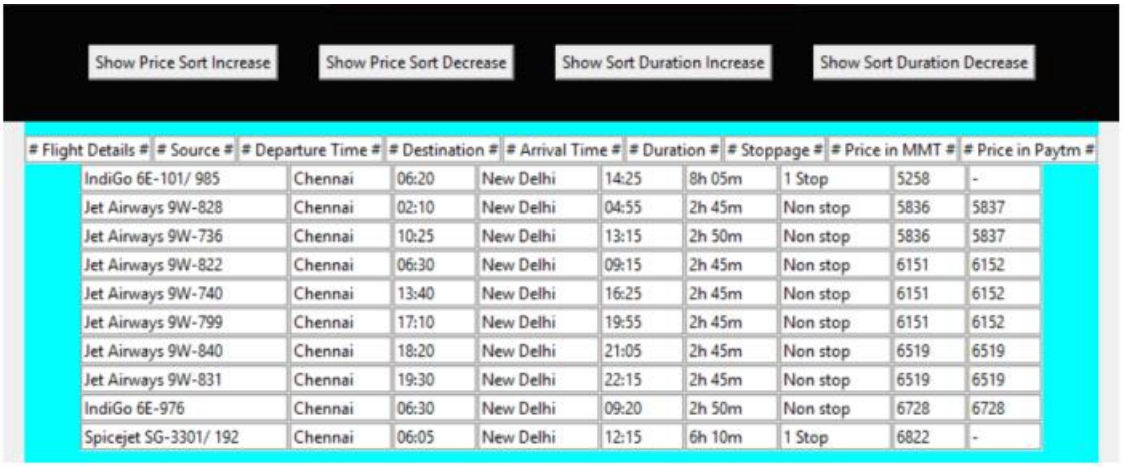


# Flight Details #	# Source #	# Departure Time #	# Destination #	# Arrival Time #	# Duration #	# Stoppage #	# Price in MMT #	# Price in Paytm #
Jet Airways 9W-828	Chennai	02:10	New Delhi	04:55	2h 45m	Non stop	5836	5837
Jet Airways 9W-822	Chennai	06:30	New Delhi	09:15	2h 45m	Non stop	6151	6152
Jet Airways 9W-740	Chennai	13:40	New Delhi	16:25	2h 45m	Non stop	6151	6152
Jet Airways 9W-799	Chennai	17:10	New Delhi	19:55	2h 45m	Non stop	6151	6152
Jet Airways 9W-840	Chennai	18:20	New Delhi	21:05	2h 45m	Non stop	6519	6519
Jet Airways 9W-831	Chennai	19:30	New Delhi	22:15	2h 45m	Non stop	6519	6519
Jet Airways 9W-736	Chennai	10:25	New Delhi	13:15	2h 50m	Non stop	5836	5837
IndiGo 6E-976	Chennai	06:30	New Delhi	09:20	2h 50m	Non stop	6728	6728
Spicejet SG-3301/ 192	Chennai	06:05	New Delhi	12:15	6h 10m	1 Stop	6822	-
IndiGo 6E-101/ 985	Chennai	06:20	New Delhi	14:25	8h 05m	1 Stop	5258	-

Figure 7 - Price Sort Duration Increase

# Flight Details #	# Source #	# Departure Time #	# Destination #	# Arrival Time #	# Duration #	# Stoppage #	# Price in MMT #	# Price in Paytm #
Spicejet SG-3301/ 192	Chennai	06:05	New Delhi	12:15	6h 10m	1 Stop	6822	-
IndiGo 6E-976	Chennai	06:30	New Delhi	09:20	2h 50m	Non stop	6728	6728
Jet Airways 9W-840	Chennai	18:20	New Delhi	21:05	2h 45m	Non stop	6519	6519
Jet Airways 9W-831	Chennai	19:30	New Delhi	22:15	2h 45m	Non stop	6519	6519
Jet Airways 9W-822	Chennai	06:30	New Delhi	09:15	2h 45m	Non stop	6151	6152
Jet Airways 9W-740	Chennai	13:40	New Delhi	16:25	2h 45m	Non stop	6151	6152
Jet Airways 9W-799	Chennai	17:10	New Delhi	19:55	2h 45m	Non stop	6151	6152
Jet Airways 9W-828	Chennai	02:10	New Delhi	04:55	2h 45m	Non stop	5836	5837
Jet Airways 9W-736	Chennai	10:25	New Delhi	13:15	2h 50m	Non stop	5836	5837
IndiGo 6E-101/ 985	Chennai	06:20	New Delhi	14:25	8h 05m	1 Stop	5258	-

Figure 6 - Price Sort Decrease



# Flight Details #	# Source #	# Departure Time #	# Destination #	# Arrival Time #	# Duration #	# Stoppage #	# Price in MMT #	# Price in Paytm #
IndiGo 6E-101/ 985	Chennai	06:20	New Delhi	14:25	8h 05m	1 Stop	5258	-
Jet Airways 9W-828	Chennai	02:10	New Delhi	04:55	2h 45m	Non stop	5836	5837
Jet Airways 9W-736	Chennai	10:25	New Delhi	13:15	2h 50m	Non stop	5836	5837
Jet Airways 9W-822	Chennai	06:30	New Delhi	09:15	2h 45m	Non stop	6151	6152
Jet Airways 9W-740	Chennai	13:40	New Delhi	16:25	2h 45m	Non stop	6151	6152
Jet Airways 9W-799	Chennai	17:10	New Delhi	19:55	2h 45m	Non stop	6151	6152
Jet Airways 9W-840	Chennai	18:20	New Delhi	21:05	2h 45m	Non stop	6519	6519
Jet Airways 9W-831	Chennai	19:30	New Delhi	22:15	2h 45m	Non stop	6519	6519
IndiGo 6E-976	Chennai	06:30	New Delhi	09:20	2h 50m	Non stop	6728	6728
Spicejet SG-3301/ 192	Chennai	06:05	New Delhi	12:15	6h 10m	1 Stop	6822	-

Figure 5 - Price Sort Increase

Conclusion

This project is aimed to build a web scrapper to retrieve the data from the websites MakeMyTrip and Paytm. The web scrapper will be developed using the Python programming language, along with the libraries Tkinter and Selenium. The web scrapper will be used to gather data from the two websites and store them in an organized manner for easier access and usage. The data collected from the websites will be used for various purposes such as getting the latest travel deals, comparing costs of flights and hotels, and getting the latest rates of different products. The web scrapper will be able to retrieve data from both the websites and store them in an organized manner. This will help in making the data more accessible and user-friendly. Once the web scrapper is developed and tested, it can be deployed into a production environment. The web scrapper can then be used to collect data from the websites MakeMyTrip and Paytm. The data collected from the websites can then be used for various purposes such as getting the latest travel deals, comparing costs of flights and hotels, and getting the latest rates of different products.