**REDBUS DATA SCRAPING PROJECT**

**ANTONY JOSHUA. A**

**BATCH NO: MDTM-24**

**PROJECT INTRODUCTION**

**Objective:** The goal of this project is to build a comprehensive data scraping and visualization system for Redbus, a popular online bus ticket booking platform. The system will leverage web scraping techniques to extract real-time data from Redbus, store it in a PostgreSQL database, and provide an interactive web interface using Streamlit for users to query and visualize the data.

**Components:**

1. **Data Scraping with Python and Selenium:**
   * **Purpose:** To extract detailed information about bus routes, timings, availability, and pricing from the Redbus website.
   * **Method:** Use Selenium, a powerful tool for automating web browsers, to navigate through the Redbus website, interact with search forms, and retrieve data from the resultant pages. The data extracted includes bus operators, departure and arrival times, ticket prices, and bus types.
   * **Implementation:** Write Python scripts that use Selenium to automate browser interactions.
2. **Data Storage with PostgreSQL:**
   * **Purpose:** To store and manage the scraped data in a structured format for efficient querying and analysis.
   * **Method:** Use PostgreSQL, a robust relational database management system, to create tables for storing bus route details, pricing information, and availability status. Implement database schema design to ensure data integrity and optimize query performance.
   * **Implementation:** Develop SQL scripts or use an ORM (Object-Relational Mapping) library like SQLAlchemy to interface with the PostgreSQL database. Schedule regular data updates using cron jobs or a similar scheduling tool to keep the database current.
3. **Interactive Visualization with Streamlit:**
   * **Purpose:** To provide an intuitive and user-friendly web interface for visualizing and interacting with the bus data.
   * **Method:** Use Streamlit, a Python framework for building interactive web applications, to create dashboards that display key metrics such as bus availability, price trends, and route information.
   * **Implementation:** Develop Streamlit apps that connect to the PostgreSQL database, retrieve data, and present it in various formats such as tables, charts, and maps. Include features like search filters.

**Process Workflow:**

1. **Scraping Automation:** run the Python-Selenium script to scrape data from Redbus.
2. **Data Ingestion:** Store the scraped data into PostgreSQL tables, ensuring the data is clean and accurately reflects the current state of the website.
3. **Data Presentation:** Use the Streamlit app to provide a real-time interface where users can view and analyze the bus data, with functionalities such as filtering by date, route, or price range.

**PROJECT MODULES**

**SCRAPING MODULE:**

**Objective:** To automate the extraction of bus-related data from Redbus and prepare it for storage and visualization.

**Key Functionalities:**

1. **Initialization:**
   * **Setup WebDriver:** Initialize the Selenium WebDriver to control a web browser (e.g., Chrome, Firefox).
   * **Configure Settings:** Set up browser options and configurations to handle dynamic content and ensure smooth scraping.
2. **Navigate to Redbus:**
   * **Open Web Page:** Use Selenium to open the Redbus website and navigate to the relevant pages or sections where the bus data is displayed.
3. **Interact with Page Elements:**
   * **Search for Buses:** Automate interactions like filling out search forms (e.g., entering departure and arrival locations, dates) and submitting queries.
   * **Handle Dynamic Content:** Wait for elements to load or become interactable if the page content is loaded dynamically with JavaScript.
4. **Extract Data:**
   * **Locate Data Elements:** Use Selenium's methods to find and extract text, attributes, or other relevant information from HTML elements (e.g., bus schedules, prices, operator names).
   * **Store Extracted Data:** Collect the extracted data into a structured format (e.g., lists, dictionaries) for further processing.

**USER INTERFACE MODULE:**

**Objective:** To create an interactive web interface using Streamlit that allows users to search for buses, view detailed results, and analyze bus data scraped from Redbus.

**Key Features:**

1. **Search Interface:**
   * Allow users to input departure and destination locations, as well as the travel date.
   * Provide a search button to retrieve bus data based on user inputs.
2. **Display Results:**
   * Show a table of available buses including details such as bus operator, departure and arrival times, and ticket prices.
   * Implement filters and sorting options for user convenience.

**SYSTEM DEPLOYMENT:**

To develop and deploy the Redbus data scarping application, we’ll require files and applications. The required things and their steps to use are listed down below.

**1. Install Python**

**Windows:**

1. **Download Python:**
   * Go to the official Python website and download the latest version of Python for Windows.
2. **Run the Installer:**
   * Open the downloaded installer.
   * Check the box "Add Python to PATH" before clicking "Install Now".
   * Follow the prompts to complete the installation.

**2. Install Streamlit**

**Using pip:**

pip install streamlit

**3. Install pgAdmin**

pgAdmin is a graphical interface for managing PostgreSQL databases.

**Windows/macOS/Linux:**

1. **Download pgAdmin:**
   * Visit the pgAdmin website and download the installer for your operating system.
2. **Run the Installer:**
   * Follow the installation instructions for your operating system.

**4. Install psycopg2**

psycopg2 is a PostgreSQL adapter for Python.

**Using pip:**

pip install psycopg2-binary

**5. Install SQLAlchemy**

SQLAlchemy is an SQL toolkit and Object-Relational Mapping (ORM) library for Python.

**Using pip:**

pip install sqlalchemy

**6. Install Selenium**

Selenium is a tool for automating web browsers.

**Using pip:**

pip install selenium

**For Chrome:**

* Download ChromeDriver from ChromeDriver's site.
* Extract the downloaded file and place it in a directory that is included in your system's PATH.

**Scrap.py:**

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.common.alert import Alert

import time

import psycopg2

driver = webdriver.Chrome()

url="https://www.redbus.in/bus-tickets/goa-airport-to-calangute-goa?fromCityId=197221&toCityId=253650&fromCityName=Goa%20Airport&toCityName=Calangute%20(goa)&busType=Any&srcCountry=IND&destCountry=IND&onward=21-Jul-2024"

driver.get(url)

driver.maximize\_window()

time.sleep(10)

button = driver.find\_element(By.CLASS\_NAME, "button")

button.click();

conn = psycopg2.connect(database = "postgres",

user = "postgres",

host= 'localhost',

password = "admin",

port = 5432)

cur = conn.cursor()

cur.execute("""CREATE TABLE redbus(

id SERIAL PRIMARY KEY,

State VARCHAR(50),

route\_name VARCHAR(100),

route\_link VARCHAR(600),

busname VARCHAR (255),

bustype VARCHAR (255),

departing\_time VARCHAR (10),

duration VARCHAR (10),

reaching\_time VARCHAR (10),

star\_rating VARCHAR (5),

price VARCHAR (5),

seats\_available VARCHAR (35));

""")

conn.commit()

cur.close()

conn.close()

ins\_qry = "INSERT INTO public.redbus(State, route\_name, route\_link, busname, bustype, departing\_time, duration, reaching\_time, star\_rating, price, seats\_available) VALUES ("

for x in range(20):

busname = driver.find\_elements(By.XPATH, "//div[contains(@class,'column-two')]/div[contains(@class,'travels')]")

bustype = driver.find\_elements(By.XPATH, "//div[contains(@class,'column-two')]/div[contains(@class,'bus-type')]")

departing\_time = driver.find\_elements(By.XPATH, "//div[contains(@class,'column-three')]/div[contains(@class,'dp-time')]")

duration = driver.find\_elements(By.XPATH, "//div[contains(@class,'column-four')]/div[contains(@class,'dur ')]")

reaching\_time = driver.find\_elements(By.XPATH, "//div[contains(@class,'column-five')]/div[contains(@class,'bp-time')]")

star\_rating = driver.find\_elements(By.XPATH, "//div[contains(@class,'column-six')]//div[contains(@class,'rating-sec')]//span")

price = driver.find\_elements(By.XPATH, "//div[contains(@class,'column-seven')]//div[@class='fare d-block']/span")

seats\_available = driver.find\_elements(By.XPATH, "//div[contains(@class,'column-eight')]//div[contains(@class,'seat-left')]")

for i in range(len(departing\_time)):

data = "'KTCL','goa-airport-to-calangute-goa'"+ ",'" + url + "', '" + busname[i].text + "', '" + bustype[i].text + "', '" + departing\_time[i].text + "', '" + duration[i].text + "', '" + reaching\_time[i].text + "', '" + star\_rating[i].text + "', '" + price[i].text + "', '" + seats\_available[i].text + "')"

qry = ins\_qry + data

print(qry)

cur.execute(qry)

conn.commit()

driver.execute\_script("window.scrollBy(0,300)")

time.sleep(3)

cur.close()

conn.close()

**App.py:**

**import pandas as pd**

**import streamlit as st**

**import streamlit.components.v1 as components**

**from pandas.api.types import (**

**is\_categorical\_dtype,**

**is\_datetime64\_any\_dtype,**

**is\_numeric\_dtype,**

**is\_object\_dtype,**

**)**

**import warnings**

**from sqlalchemy import text**

**import psycopg2**

**import pandas as pd**

**warnings.filterwarnings('ignore')**

**conn = st.connection("postgresql", type="sql")**

**df = pd.DataFrame(conn.query("SELECT \* FROM redbus"))**

**st.title("Redbus Data Scraping Project")**

**def filter\_dataframe(df: pd.DataFrame) -> pd.DataFrame:**

**modify = st.checkbox("Add filters")**

**if not modify:**

**return df**

**df = df.copy()**

**for col in df.columns:**

**if is\_object\_dtype(df[col]):**

**try:**

**df[col] = pd.to\_datetime(df[col])**

**except Exception:**

**pass**

**if is\_datetime64\_any\_dtype(df[col]):**

**df[col] = df[col].dt.tz\_localize(None)**

**modification\_container = st.container()**

**with modification\_container:**

**to\_filter\_columns = st.multiselect("Filter dataframe on", df.columns)**

**for column in to\_filter\_columns:**

**left, right = st.columns((1, 20))**

**if is\_categorical\_dtype(df[column]) or df[column].nunique() < 10:**

**user\_cat\_input = right.multiselect(**

**f"Values for {column}",**

**df[column].unique(),**

**default=list(df[column].unique()),**

**)**

**df = df[df[column].isin(user\_cat\_input)]**

**elif is\_numeric\_dtype(df[column]):**

**\_min = float(df[column].min())**

**\_max = float(df[column].max())**

**step = (\_max - \_min) / 100**

**user\_num\_input = right.slider(**

**f"Values for {column}",**

**min\_value=\_min,**

**max\_value=\_max,**

**value=(\_min, \_max),**

**step=step,**

**)**

**df = df[df[column].between(\*user\_num\_input)]**

**elif is\_datetime64\_any\_dtype(df[column]):**

**user\_date\_input = right.date\_input(**

**f"Values for {column}",**

**value=(**

**df[column].min(),**

**df[column].max(),**

**),**

**)**

**if len(user\_date\_input) == 2:**

**user\_date\_input = tuple(map(pd.to\_datetime, user\_date\_input))**

**start\_date, end\_date = user\_date\_input**

**df = df.loc[df[column].between(start\_date, end\_date)]**

**else:**

**user\_text\_input = right.text\_input(**

**f"Substring or regex in {column}",**

**)**

**if user\_text\_input:**

**df = df[df[column].astype(str).str.contains(user\_text\_input)]**

**return df**

**st.dataframe(filter\_dataframe(df))**

**CONCLUSION:**

In my project I have concluded that

1. using selenium data can be collected efficiently from Red Bus websites, significantly reducing the time and effort compared to manual collection.

2. This allows for scaling up the data collection process to gather large datasets across multiple pages and sources without manual intervention.

3. precise control over what data to scrape and ensuring that only relevant information is collected

4. comparatively it is budget friendly as there is no need for third party purchase and scraping is the cost effective method for obtaining data.

5. my study ensures consistent data collection methods, leading to more uniform and high-quality datasets.

6. It prevents the human errors associated with manual data entry and collection.

7. Analysis of customer reviews and feedback can provide valuable insights into customer preferences and areas for improvement.