**Project Illustration**

**Title: Redbus Data Scraping with Selenium & Dynamic Filtering using Streamlit**

**Objective:** Develop a web scraper to automate the extraction of bus route details like bus name, bus links, bus type, departing time, duration, reaching time, star rating, price and relevant information from the RedBus website for bus from Bangalore to Chennai. Store the data in an SQL database and visualize it using a Streamlit app**.**

**SCOPE:**

* **DATA EXTRACTION:** Scarpe bus route, names, and detailed information for each route, including bus name, bus type, departing time, duration, reaching time, star rating, price, and seat availability.
* **DATA STORAGE:** Store the scraped data in an SQL database.
* **VISUALIZATION:** Develop a Streamlit app to visualize and analyze the stored data.

**Solution Overview**

The solutions involve three main components: web scraping, SQL database integration, and Streamlit app development.

**WEB SCRAPING**

**1. Required Libraries**

The script uses the following libraries:

* **selenium**: For web scraping and automating browser interactions.
* **pandas**: For handling structured data (though not used in this specific script, it's imported for potential use).
* **mysql.connector**: For connecting to and interacting with a MySQL database.
* **time**: To introduce delays for the webpage to load content properly.

Additional installations:

* **webdriver\_manager.chrome**: Automatically manages the ChromeDriver required by Selenium.

**2. Connecting to MySQL**

The connect\_to\_mysql() function establishes a connection to a local MySQL database.

**Parameters**:

* + - host: The database server, here set to "localhost".
    - user: Database username, here "root".
    - password: Database password, here "root".
    - database: The name of the target database (redbus\_project).

**3. Inserting Bus Details into MySQL**

The insert\_bus\_details() function inserts bus route details into a MySQL table, ensuring no duplicate entries are added.

**Input**:

* bus\_details: A list of dictionaries, each containing details about a bus.
* connection: MySQL database connection object.

**Process**:

**For each bus in bus\_details:**

* Check if a similar record already exists in the table using SELECT COUNT(\*).
* If no match exists, insert the details into the bus\_routes table using an INSERT query.
* Rollback in case of errors.

**Field in MySQL Table**:

* route\_name: Combination of source and destination (e.g., "Bangalore-Chennai").
* busname: Name of the bus service.
* bustype: Type of bus.
* departing\_time: Departure time of the bus.
* duration: Duration of the trip.
* reaching\_time: Arrival time.
* Star\_rating: Bus rating, if available.
* price: Ticket price.
* seats\_available: Number of seats available.

>>>Service=Service(ChromeDriverManager().install())

>>>driver = webdriver.Chrome(service=service)

**Explanation for the above code:**

* The script uses **Selenium WebDriver** to interact with the **RedBus** website and extract data
* Installs and configures ChromeDriver using webdriver\_manager.chrome.

>>>driver.get("https://www.redbus.in/")

**Explanation for the above code:**

* This line instructs the Selenium WebDriver to open the RedBus website (https://www.redbus.in/) in the browser controlled by the WebDriver.

>>>driver.maximize\_window()

**Explanation for the above code:**

* Maximizes the browser window and navigates to RedBus.

**Steps:**

* + Input departure city (e.g., "Bangalore") and destination city (e.g., "Chennai").
  + Select a travel date. For simplicity, the script selects the next available date.
  + Click the **Search** button to display the available buses.
  + Scroll down the page to load all search results dynamically (handled by JavaScript).
  + Extract bus details for each bus listing.

**4. Data Extraction**

* **Target Elements**:
  + **Bus Name**: Located using the travels class.
  + **Price**: Extracted from the fare class.
  + **Bus Type**: From the bus-type class (e.g., AC, Sleeper).
  + **Departure Time**: Using the dp-time class.
  + **Duration**: Extracted from the dur class.
  + **Arrival Time**: From the bp-time class.
  + **Rating**: From the rating class.
  + **Seats Available**: Found in the seat-left class, extracting the numeric portion of the text.
* **Handling Missing Data**:
  + If a specific field is unavailable, the script assigns "N/A" (for text fields) or None (for numeric fields).

>>> bus\_details.append({

        "RouteName":f"{from\_str}-{to\_str}",

        "Name": name,

        "Price": price,

        "Type": bus\_type,

        "Departure Time": departure\_time,

        "Duration": duration,

        "Reaching Time": reaching\_time,

        "Rating": rating,

        "Seats Available": seats,

    })

**Explanation for the above code:**

The data is stored in a MySQL table (bus\_routes) with the following schema:

* route\_name (TEXT): Combined source and destination.
* busname (TEXT): Name of the bus service.
* bustype (TEXT): Type of the bus
* departing\_time (TIME): Departure time.
* duration (TIME): Trip duration.
* reaching\_time (TIME): Arrival time.
* star\_rating (FLOAT): Bus rating.
* price (DECIMAL): Ticket price.
* seats\_available (INT): Number of seats available.

>>>print("Available Buses and their Details count: ",len(bus\_details))

>>>for i, bus in enumerate(bus\_details, 1):

…  print(f"Bus {i}:")

…  for key, value in bus.items():

…        print(f"  {key}: {value}")

 …   print("-" \* 50)

**Explanation for the above code:**

* This section of the script is responsible for printing the extracted bus details in a structured and human-readable format.

>>>connection = connect\_to\_mysql()

>>>insert\_bus\_details(bus\_details, connection)

>>>connection.close()

**Explanation for the above code:**

* This block of code is responsible for establishing a connection to the MySQL database, inserting the extracted bus details into it, and then closing the connection

**5. Example Output**

When executed successfully, the script will:

* Print the total number of buses found and their details in the console.
* Insert unique bus details into the bus\_routes table in MySQL.
* Close the browser and database connection gracefully.

**SQL DATABASE INTEGRATION**

* The connect\_to\_mysql() function establishes a connection to a local MySQL database TO create a database and define table structures.

**STREAMLIT APP DEVELOPMENT**

**Required Libraries**

>>>import streamlit as st

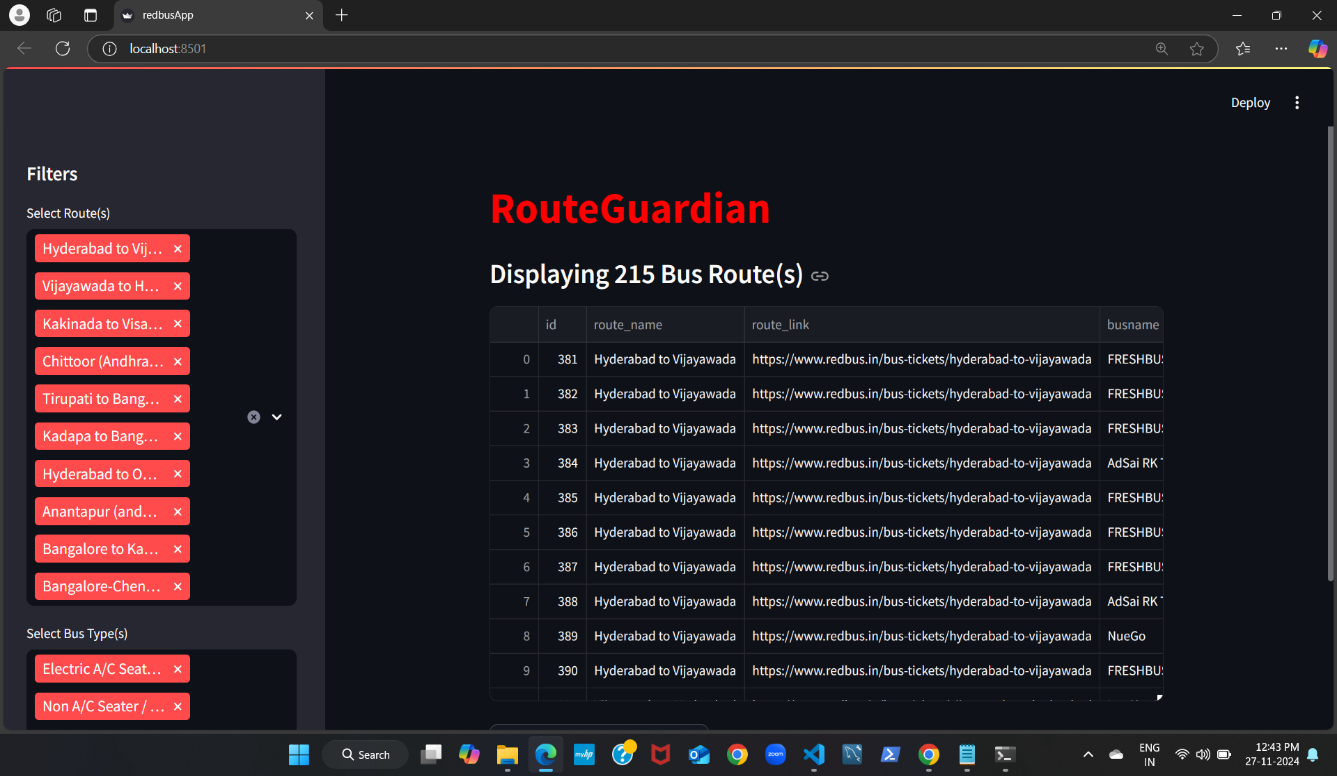
>>>import pandas as pd

>>>from sqlalchemy import create\_engine

This Streamlit application connects to a MySQL database using SQLAlchemy to retrieve and display data about bus routes. Here's a concise breakdown:

* **Database Connection**:
  + The get\_database\_connection function establishes a connection to a MySQL database.
  + Data is fetched with the fetch\_data function via an SQL query.
* **Filters**:
  + Users can filter bus routes in the sidebar by:
    - Route name
    - Bus type
    - Price range
    - Star rating
    - Minimum available seats
* **Data Display**:
  + The filtered results are displayed in an interactive table.
  + Users can export the filtered data as a CSV file.

By following this approach, you can automate the scraping of bus route details, store the data in an SQL database, and create an interactive Streamlit app for data analysis.

****