REDBUS PROJECT

**Objectives**:

* To Successfully scrape data from Redbus website using Selenium.
* 17 State’s Government Bus Transport
* Also the private bus information for the selected routes.
* To Store the data in a structured SQL database.
* To Develop an interactive Streamlit application for data filtering.
* To Ensure the application is user-friendly and efficient.

**Scope:**

**Data Scraping:**

Selenium Script is used to Scrape Bus Details from each page on redBus Website. This script automates the process of scraping bus route and bus details from the booking section on the redBus website using Selenium. The data from multiple pages are collected and saved into CSV files for further analysis. The script is scalable and can be used for scraping similar data from different states, resulting in multiple CSV files which are later merged for further processing.

**Libraries Used:**

* **Selenium (selenium.webdriver):** For browser automation and interaction with web elements.
* **Pandas (pandas):** For organizing data into a DataFrame and saving it to CSV files.
* **Time (time)**: For adding delays to ensure web pages load before scraping data.

**Functions:**

1. **Initialize the WebDriver:** initialize\_driver():

Sets up the Chrome WebDriver and maximizes the browser window.

Returns the initialized WebDriver object.

1. **Load Web Page:** load\_page(driver, url):

Takes a URL and loads the page in the provided WebDriver.

Waits for 5 seconds to ensure the page is fully loaded.

1. **Scrape Bus Routes:**scrape\_bus\_routes(driver):

Scrapes all bus routes by finding route URLs and names.

Returns two lists

* One containing the route URLs.
* One containing the corresponding route names.

1. **Scrape Bus Details:** scrape\_bus\_details(driver, url, route\_name):

* Loads the bus route URL.
* Clicks on the "View Buses" button.
* Scrolls the page to load all available buses.
* Collects the following details for each bus.

Bus Name

Bus Type

Departure Time

Duration

Reaching Time

Star Rating

Price

Seat Availability

**The data is stored as dictionaries and appended to a list.**

1. **Scrape All Pages:**

* scrape\_all\_pages(): Handles pagination for loading the next set of pages.

Iterates through the first 5 pages of bus route listings.

* For each route, it calls scrape\_bus\_routes() and scrape\_bus\_details():
* Scrapes the bus details and appends them to a list.

1. **Save Data:**

* Once all the bus details are scraped, the data is saved into a Pandas DataFrame.
* Then exported to a CSV file.

**7. Closing the WebDriver:**

* The script terminates by calling driver.quit() to close the browser window

and end the WebDriver session.

**Key Functions:**

***initialize\_driver():*** Sets up the Chrome WebDriver.

***load\_page(driver, url):*** Loads the specified URL in the browser.

***scrape\_bus\_routes(driver):*** Scrapes bus routes from the current page.

***scrape\_bus\_details(driver, url, route\_name):*** Scrapes detailed bus information from the selected route.

***scrape\_all\_pages():*** Iterates through multiple pages, scrapes routes, and extracts bus details.

***pd.DataFrame() and df.to\_csv():*** Organizes data into a DataFrame and saves it as a CSV.

**Merging Multiple DataFrames:**

Once the bus data for all 17 states is scraped and saved as individual CSV files, the script merges them into a single DataFrame for easier analysis.

**Steps for Merging:**

* **Read CSV Files:**

The individual CSV files are read using pd.read\_csv().

* **Concatenate DataFrames:**

All DataFrames are merged using pd.concat()

the argument ignore\_index=True to reset the index.

* **Save the Merged Data:**

The consolidated DataFrame is saved to a new CSV file using df.to\_csv().

**Data Cleaning Steps:**

Data cleaning is performed to handle missing values, convert time formats, and ensure the data is in the correct structure for analysis.

**Steps:**

* **Handling Missing Values:**

The columns "Price" and "Star\_Rating" are filled with default values (0) using fillna().

* **Time Format Conversion:**

The columns "Departing\_Time" and "Reaching\_Time" are converted from strings to datetime objects using pd.to\_datetime().The times are then formatted to a 24-hour format (HH:MM) using strftime().

**Key Functions:**

**fillna():** Replaces missing values in specified columns with a default value (e.g., 0).

**pd.to\_datetime():** Converts string data into datetime objects.

**strftime('%H:%M'):** Formats datetime objects into a readable time format (HH:MM).

**Saving DataFrame to CSV:**

The final cleaned DataFrame is saved to a CSV file. The process ensures data is stored in an easily readable and shareable format.

**Steps:**

* **Specify File Path:**

Define the path where the CSV will be saved (e.g., C:/Users/Downloads/bus\_routes.csv).

* **Save to CSV:**

The DataFrame is saved using df.to\_csv(path, index=False).

**Inserting CSV Data into MySQL Database:**

The script reads the data from CSV files and inserts it into a MySQL database for more efficient querying and manipulation.

**Key Steps:**

* **Connect to MySQL:**

Usepymysql.connect() to connect to the database.

* **Create Table:**

Create a table (dbusdata) to store the bus details, ensuring the table structure matches the data.

* **Insert Data:**

Use cursor.execute() to insert rows of data from the CSV file.

* **Commit and Close:**

Commit the changes to the database and close the connection.

**Connecting to MySQL and**

**Loading Data into Pandas DataFrame:**

Data from the MySQL database is retrieved into a Pandas DataFrame for further manipulation or visualization.

**Key Steps:**

* **Execute SQL Query:**

A SQL query is executed to fetch data from the dbusdata table.

* **Convert to DataFrame:**

The result is converted into a DataFrame and unnecessary columns are dropped.

**Streamlit Integration:**

The final step involves using Streamlit to create an interactive web app for tracking bus information.

**Key Steps:**

* **State Selection:**

Users select a state, and the corresponding CSV data is loaded.

* **Route and Bus Type Selection:**

Data is filtered based on selected routes and bus types.

* **Fare Range:**

Users can filter data by price ranges.

* D**ata Display:**

Filtered data is displayed dynamically on the Streamlit app.

**Conclusion:**

This project involves scraping bus data from the redBus website, organizing it, and storing it in CSV and MySQL databases. The data is cleaned and made accessible for analysis using Streamlit for dynamic filtering and visualization**.**