Redbus Travel Insights: Data Scraping, Filtering Using Streamlit Web Application

ABSTRACT

The **"Redbus Data Scraping and Filtering with Streamlit Application"** project leverages modern technologies to streamline and optimize the collection, analysis, and visualization of bus travel data. By utilizing **Selenium**, the project automates the extraction of critical information from the Redbus platform, including bus routes, schedules, pricing, and seat availability. The scraped data is then processed and presented through an interactive **Streamlit-based web application**, enabling users to filter, analyze, and visualize the information effortlessly.

This project offers a robust solution for stakeholders in the transportation industry, such as operators, planners, and passengers, to make data-driven decisions. It enhances operational efficiency by automating data collection, supports strategic planning through insightful analytics, and improves user experience with real-time data accessibility. The integration of automation and intuitive visualization makes this project a vital tool for revolutionizing the way bus travel data is utilized in the industry.

INTRODUCTION

In today's fast-paced world, efficient transportation systems are essential for connecting people and places. Platforms like **Redbus** play a significant role by offering a convenient way to search and book bus tickets. However, the vast amount of data available on such platforms, including routes, schedules, pricing, and seat availability, often remains underutilized due to the lack of accessible tools for analysis and visualization.

The **"Redbus Data Scraping and Filtering with Streamlit Application"** addresses this gap by providing an automated solution to extract and analyze this valuable data. Leveraging **Selenium** for web scraping, the project ensures accurate and timely data collection from the Redbus platform. This data is then processed and displayed through an interactive **Streamlit-based application**, allowing users to filter and visualize key insights effectively.

This project aims to empower stakeholders—such as bus operators, transportation planners, and passengers—by delivering actionable insights that can improve decision-making and

operational efficiency. Combining automation, and data filtering, this system offers a powerful tool to enhance the management and optimization of bus transportation services.

BUSINESS USE CASES:

Dynamic Pricing Strategies:

The solution can help operators analyze demand patterns and implement dynamic pricing models to maximize revenue during peak and off-peak travel times.

Route Optimization:

By analyzing route performance and occupancy data, transportation companies can adjust schedules, add or remove routes, and optimize bus utilization for greater efficiency.

Demand Forecasting:

Historical data on travel patterns can be used to predict future demand, allowing operators to prepare for seasonal fluctuations or special events.

Fleet Management:

The system provides insights into the performance and utilization of individual buses, helping operators manage maintenance schedules, reduce downtime, and improve fleet efficiency.

Partnership Development:

Travel platforms can use the data to identify potential partnerships with bus operators based on service quality and coverage, expanding their network and service offerings.

Regulatory Compliance and Reporting:

The data collected can be leveraged to ensure compliance with transportation regulations and to generate reports for government agencies or stakeholders.

APPROACH:

1. Data Scraping:
    - Use Selenium to automate the extraction of Redbus data including routes, schedules, prices, and seat availability.
2. Data Storage:
    - Store the scraped data in an SQL database.
3. Streamlit Application:
    - Develop a Streamlit application to display and filter the scraped data.

- Implement various filters such as bustype, route, price range, star rating, availability.

4. Data Analysis/Filtering using Streamlit:
    - Use SQL queries to retrieve and filter data based on user inputs.
    - Use Streamlit to allow users to interact with and filter the data through the application.

DATA SCRAPING USING SELENIUM

Data scraping is a core component of the project, and Selenium is employed for its capability to interact with dynamic web pages effectively. Redbus, being a **dynamic platform,** uses JavaScript to load content, making Selenium an ideal choice for extracting data such as routes, schedules, prices, and seat availability.

Steps to Implement Data Scraping with Selenium

1. Setup and Environment Configuration:
    - Install necessary libraries: Selenium, and pandas (for data handling).
    - Download and configure the appropriate WebDriver (e.g., ChromeDriver for Google Chrome).
2. Navigate to Redbus Website:
    - Use Selenium's webdriver to open the Redbus platform.
    - Interact with the page elements like search bars, dropdown menus, and buttons to input the required travel details (source, destination, and travel dates).
3. Handle Dynamic Content:
    - Wait for the content to load using explicit waits (WebDriverWait) to ensure fully rendered elements like schedules and prices.

4. Extract Required Data:
    - Identify the required HTML elements using XPath, CSS selectors, or ID/Name attributes to scrape data fields like bus names, departure times, arrival times, prices, and seat availability.
    - Use Selenium commands such as find_element_by_xpath or find_elements_by_class_name to retrieve data.
5. Store Data:
    - Collect the scraped data and structure it into a suitable format (e.g., lists or dictionaries).
    - Use pandas to convert the structured data into a data frame for further processing.
6. Error Handling:
    - Implement error handling for cases like missing elements, page timeouts, or captchas.
7. Save Data:
    - Export the final dataset into formats like CSV, and JSON, or directly integrate it into the Streamlit application for real-time use.

Benefits of Selenium for Data Scraping

- Handles dynamic content rendered by JavaScript.
- Supports interaction with web elements (e.g., clicks, form filling).
- Automates complex navigation and data extraction processes.

DATA STORAGE USING SQL CONNECTOR IN PYTHON

After scraping data using Selenium, storing it in a structured database ensures efficient management, querying, and future retrieval. By integrating SQL Connector in Python, the scraped Redbus data can be stored in relational databases like MySQL or SQLite, enabling robust data handling for analysis.

Steps to Implement Data Storage

Setup the Database:

Install and configure the database system (e.g., MySQL or SQLite).

Use an SQL client or script to create a database and relevant tables to store the scraped data.

Install SQL Connector:

“pip install mysql-connector-python”

Define the Database Schema:

Decide on a schema to store data. For example:

Table Name: redbus_data

Columns:

1. id (Primary Key)
2. Bus Route Name (VARCHAR)
3. Bus Routes Link (VARCHAR)
4. bus_name (VARCHAR)
5. departure_time (TIME)
6. arrival_time (TIME)
7. price (DECIMAL)
8. Bus Type (Sleeper/Seater) (VARCHAR)
9. seat_availability (VARCHAR)
10. Star Rating (DECIMAL)

Connect to the Database:

Using the SQL connector, establish a connection between the Python application and the SQL database.

Insert Data into the Database:

Insert the scraped data into the database table, either in bulk or row-by-row.

Query and Retrieve Data:

Verify successful data storage by querying the database and displaying the results in Python.

Advantages of Using SQL for Data Storage:

Structured Data Management:

Relational databases organize data in tables with defined relationships, making it easy to manage and query.

Scalability:

SQL databases can handle large datasets efficiently, which is useful for extensive Redbus scraping projects.

Data Integrity:

SQL constraints (e.g., primary keys) help maintain data accuracy and consistency.

Flexibility in Querying:

Complex queries can be written to filter or aggregate data as needed for analytics or visualization.

STREAMLIT APPLICATION:

Streamlit is a powerful and easy-to-use framework for building interactive web applications. In this project, Streamlit can be used to display the scraped and stored Redbus data, allowing users to filter, explore, and visualize information such as bus routes, timings, seat availability, prices, and more.

Steps to Create the Streamlit Application

Install Streamlit "**pip install streamlit"**

Connect to the Database:

The app will retrieve data stored in your MySQL database using SQL queries. You can also load the data directly from CSV or other formats if needed.

Design the Layout:

The Streamlit app should have interactive filters, data visualizations, and tables to make it easy for users to explore bus schedules, prices, and availability.

Add Functionalities:

Filtering buses by route, type, price, availability, etc.

Sorting buses by price or rating.

Displaying detailed bus information.

Benefits of the Streamlit Application:

1. User-Friendly: Streamlit provides an easy-to-use interface that allows users to interact with data without needing complex software development skills.
2. Interactive Filters: Users can filter and customize data views based on their preferences (e.g., bus type, price range, availability).
3. Data Visualization: The app provides insightful visualizations like bar charts, helping users make informed decisions about their bus travel options.
4. Rapid Deployment: Streamlit apps are easy to deploy and host, providing a quick and flexible solution for real-time data analysis.

Data Analysis/Filtering Using Streamlit

In this step, the Streamlit application will display the data and allow for interactive data analysis and filtering. Users can filter bus routes based on various criteria, analyze trends (such as price or seat availability), and perform in-depth queries directly in the app. Streamlit's interactive widgets make it easy to enable real-time data filtering and analysis.