Redbus Data Scraping and Filtering with Streamlit Application

1. Overview

The **"Redbus Data Scraping and Filtering with Streamlit Application"** project is designed to automate the process of collecting, analyzing, and visualizing bus travel data from Redbus. The project leverages Selenium for web scraping and Streamlit for interactive data filtering and presentation. This system aims to revolutionize the transportation industry by providing a comprehensive solution for obtaining real-time bus schedules, prices, seat availability, and other relevant travel information.

The system's goal is to streamline data collection from Redbus, perform automated filtering, and display the results in a user-friendly interface for improved decision-making and operational efficiency. By analyzing bus routes, schedules, and other critical data, the system empowers users to make data-driven decisions and optimize transportation strategies.

Technologies Used:

- **Selenium**: A browser automation tool used to scrape bus data from the Redbus website.
- **Streamlit**: A Python library used to build an interactive user interface for data visualization and filtering.
- **pymysql**: A MySQL connector for storing and managing scraped data in a MySQL database.
- **pandas**: A Python library for data manipulation and analysis, used to process the scraped data.

2. Problem Statement

The transportation industry faces challenges in obtaining real-time, accurate, and structured data on bus routes, schedules, prices, and seat availability. The **Redbus Data Scraping and Filtering with Streamlit Application** aims to address this problem by:

- **Automating Data Collection**: Using Selenium to scrape detailed bus information from the Redbus platform, ensuring accurate and up-to-date data.
- **Data Filtering and Analysis**: Using Streamlit, users can filter and explore the scraped data based on various criteria such as bus type, price range, route name, and star ratings.
- **Real-Time Insights**: Providing real-time insights into available buses, allowing for smarter transportation planning and operational decision-making.

---

3. Code Explanation

Web Scraping with Selenium

The data collection process is performed using Selenium, which automates the extraction of bus route data from the Redbus website. Below is the breakdown of the scraping code:

Web Driver Initialization:

```
driver = webdriver.Chrome()
```

1. The webdriver.Chrome() initializes the browser, allowing interaction with the webpage.
   Navigating to the Redbus URL:

   ```
   url = 'https://www.redbus.in/bus-tickets/kishangarh-to-jaipur'driver.get(url)
   ```

2. The driver navigates to the Redbus page for bus tickets from Kishangarh to Jaipur. This URL is parameterized to search for routes based on these cities.

Waiting for the Page to Load:

```
time.sleep(15)
```

3. The code waits for 15 seconds to allow the page to fully load. In production, dynamic waiting using WebDriverWait would be more efficient.

Extracting Bus Data:

```
bus_elements = driver.find_elements(By.XPATH, "//div[contains(@class, 'clearfix bus-item-details')]")
```

4. The find_elements() function locates all bus listing elements on the page using XPath.

Iterating Through Each Bus Element:

For each bus element, the following details are extracted using XPath:

- **Bus Name**
- **Bus Type**
- **Departure Time**
- **Duration**
- **Arrival Time**
- **Rating** (handled with exception for missing data)
- **Price**
- **Seat Availability**

Storing Data in a List:

```
 bus_data.append({
    "Route Name": "test",  # Placeholder, will be updated dynamically
    "Route Link": "test",  # Placeholder
    "Bus Name": bus_name,
    "Bus Type": bus_type,
    "Departure Time": departure_time,
    "Duration": duration,
    "Arrival Time": arrival_time,
    "Rating": rating,
    "Price": price,
    "Seats Available": seats_available,
    "route_id": "98"
})
```

5. The extracted bus data is stored in a list of dictionaries for further processing.

Closing the Browser:

```
 driver.quit()
```

6. After the data is collected, the browser is closed.

Storing Data in MySQL

Once the data is collected, it is stored in a MySQL database using pymysql:

Connecting to MySQL:

```
 connection = pymysql.connect(
    host='localhost',
    user='root',
    password='*****',
    database='******'
)
cursor = connection.cursor()
```

1. **Insert Query:**

```
            insert_query = """
INSERT INTO bus_routes (route_name, route_link, busname, bustype, departing_time,
duration, reaching_time, star_rating, price, seats_available, route_id)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
"""
for bus in bus_data:
    data_tuple = (
        bus["Route Name"],
        bus["Route Link"],
        bus["Bus Name"],
        bus["Bus Type"],
        bus["Departure Time"],
        bus["Duration"],
        bus["Arrival Time"],
        bus["Rating"],
        bus["Price"],
        bus["Seats Available"],
        bus["route_id"]
    )
    cursor.execute(insert_query, data_tuple)
```

```
connection.commit()
```

2. This code inserts the scraped data into the bus_routes table in the MySQL database.

Streamlit Application for Data Visualization and Filtering

The Streamlit application provides an interactive user interface for filtering and visualizing the bus data. Here's a breakdown of the application:

Streamlit Setup:

```
 import streamlit as st
import pandas as pd
```

1.
2. **Database Connection:** The Streamlit app fetches the bus data stored in the MySQL database using a SQL query. The data is loaded into a Pandas DataFrame.

3. **User Interface:**

    - **Filters**: The user can filter bus data based on bus type, route name, price range, and star rating.
    - **Table Display**: The filtered results are displayed as a table, providing users with a clear overview of available buses.
    - **Visual Enhancements**: Custom CSS is used to improve the visual appeal of the interface, including background images, font styling, and animations.

Data Filtering Logic: The app uses user input (e.g., bus type, route name, price range) to filter the dataset:

Display Results: The filtered results are displayed using the st.dataframe() function. If no data matches the filters, a warning is shown.

---

4. Data Collection Process

The data collection process involves the following steps:

1. **Scraping the Redbus Website:**
    - Using Selenium, the script opens a browser and navigates to the specified Redbus page for bus routes.
    - The script waits for the page to fully load and then locates all bus listing elements.
    - Bus details are extracted (name, type, time, duration, price, etc.) and stored in a list.
2. **Storing the Data in MySQL:**
    - The scraped data is stored in a MySQL database for further analysis and querying.
3. **Interactive Filtering and Analysis:**
    - The user can filter the bus data by bus type, route name, price range, and star rating in the Streamlit app, making it easier to find specific bus routes.

---

5. Application Usage

How to Use the Web Scraping Script:

1. Ensure you have **Selenium** and **ChromeDriver** installed.
2. Run the scraping script to collect data from Redbus.
3. The data will be stored in the specified MySQL database.

How to Use the Streamlit Application:

1. **Install Required Libraries:** Install the required libraries via pip:
   ```
   pip install streamlit pandas pymysql
   ```
2. **Run the Streamlit App:** Navigate to the directory containing the Streamlit app and run:
   ```
   streamlit run app.py
   ```
3. **Use the Filters:**

   - Use the sidebar filters to refine the displayed bus data based on bus type, route name, price range, and rating.
   - View the filtered bus routes displayed as a table in the main section of the app.

6. Conclusion

The **Redbus Data Scraping and Filtering with Streamlit Application** automates the collection, filtering, and visualization of bus data, enabling transportation professionals to make more informed decisions based on real-time data. By combining Selenium for web scraping with Streamlit for interactive analysis, this project provides a powerful solution for improving operational efficiency and strategic planning in the transportation industry.