**Introduction:**

Redbus, founded in 2006, is India's leading online bus booking platform, revolutionizing intercity travel with technology and ease. Redbus serves over 20 million consumers annually, connecting them to 2,500+ operators over 220,000+ routes, guaranteeing seamless travel experiences. The platform covers more than six nations and includes live monitoring, customizable seat selection, and real-time information. Redbus has transformed the global transport business by integrating a user-friendly interface with cutting-edge technology.

**Problem statement:**

The "Redbus Data Scraping and Filtering with Streamlit Application" project seeks to automate data gathering from Redbus, store it in a database, and provide an interactive user interface for filtering and visualizing bus journey data. The system improves customer service, travel analysis, and business intelligence within the transportation industry.

**Packages Used and Installation Commands:**

**1. Selenium** is used for web scraping. It automates the browser to extract dynamic data from the Redbus website, such as bus schedules, prices, and seat availability.

**2. Streamlit** is a Python-based library that creates interactive and visually appealing web applications. It displays and filters the scraped data in a user-friendly interface.

**3. PyMySQL** is used to connect and interact with the SQL database where the scraped data is stored. It enables executing queries, fetching filtered results, and maintaining database communication.

**4. Pandas** is a powerful library for data manipulation and analysis. It is used to structure, filter, and display data retrieved from the SQL database.

**5. NumPy** is used for numerical computations, such as replacing missing star ratings with NaN for better data handling and analysis.

**6. QRCode** is used to generate QR codes for additional app features like linking to the Redbus Play Store download page.

**7. Pillow** is used to handle image processing, like creating and displaying QR codes or images in the Streamlit app.

**8. Time** is part of Python's standard library. It is used to add delays and improve the user experience in Streamlit, such as loading animations.

**9. io** is a Python built-in module used to handle byte streams, like converting QR code images for display in the application.

**10. PyMySQL.error (Error Handling)** handles database-related errors during connections or queries, ensuring graceful handling of failures.

**Installation Commands:**

* **Selenium:** pip install selenium
* **Streamlit:** pip install streamlit
* **Mysql:** pip install pymysql
* **Pandas:** pip install pandas
* **Numpy:** pip install numpy
* **QR Code:** pip install qrcode[pil]
* **Pillow:** pip install pillow

**Code Flow:**

**Environment Setup:**

1. Create a folder named **red\_bus\_project**, then open Visual Studio and open the folder.
2. Then create a new file called **red\_bus\_project.py** (or manually create it through your editor).
3. If using VS Code, press Ctrl + ~ or go to Terminal > New Terminal.
4. In the terminal, create a virtual environment within the project folder: **python -m venv venv.**
5. Activate the virtual environment using: **Windows** -.\venv\Scripts\activate, **Mac** - ./mysql\_env/bin/activate or source mysql\_env/bin/activate
6. Install the necessary **libraries** like Streamlit, PyMySQL, pandas, qrcode, and pillow by running: pip install streamlit pymysql pandas qrcode pillow.

**Selenium**: Web scraping

1. **Import Libraries**: Import necessary libraries (selenium, time, pandas, csv).
2. **Launch and Maximize Browser**: Open the browser (e.g., Chrome) and maximize it.
3. **Scroll Page**: Scroll down to load hidden elements (e.g., "View All" button).
4. **Click "View All"**: Click the "View All" button to reveal additional options.
5. **Extract State Names and Links**: Loop through states, extract names and links, and store them in a list/DataFrame.
6. **Visit State-Specific Pages**: Loop through each state and collect route names and links.
7. **Extract Route Names and Links**: For each state, collect route names and links.
8. **Visit Route Pages**: Loop through routes, visiting each for more details.
9. **Extract Bus Information**: On each route page, extract bus details (name, type, time, price, availability).
10. **Store Data**: Append state, route, and bus data to Pandas DataFrames.
11. **Save to CSV**: Save the collected data to CSV files (route\_data.csv and bus\_data.csv).
12. **Close Browser**: Close the browser after scraping is complete.

**Note:**

* You need a **web driver** compatible with your browser to run the automation software (e.g., **ChromeDriver** for Google Chrome). Download it from [ChromeDriver](https://sites.google.com/chromium.org/driver/) and ensure it matches your browser version.
* To run **python red\_bus\_project.py**

**PyMySQL:** Warehouse

**Creation of Database:**

1. **Import Libraries**: Import pymysql and pandas.
2. **Creation of database:**

* **Open MySQL Workbench** and connect to the server.
* To create a database, run the query: **CREATE DATABASE red\_bus;**

1. **Database Connection**: Connect to MySQL using pymysql.connect(), providing connection parameters (host, user, password, database).
2. **Check Connection**: Verify the connection status.
3. **Create Cursor**: Use conn.cursor() to create a cursor for executing queries.
4. **Drop Existing Tables**: Drop any existing tables (route\_data, bus\_data) with DROP TABLE IF EXISTS.
5. **Create Tables**: Create new tables (route\_data, bus\_data) with appropriate columns using CREATE TABLE.
6. **Insert Data**: Insert data from the Pandas DataFrame into MySQL using cursor.executemany() and an INSERT INTO query.
7. **Commit Transaction**: Save changes with conn.commit().
8. **Close Cursor**: Close the cursor with cursor.close().
9. **Close Connection**: Close the database connection with conn.close().
10. **Optional - Verify Data**: Optionally, run a SELECT query to verify data insertion.

**Streamlit: Web application**

1. **Import Libraries**: Import necessary libraries: streamlit, pymysql, pandas, qrcode, etc.
2. **Configure Streamlit Page**:
   * Set page title, icon, layout, and initial sidebar state using st.set\_page\_config().
3. **Database Connection**:
   * Create functions to connect to MySQL database using pymysql.connect().
   * Fetch data (state names, route names, bus details) based on user input.
4. **Sidebar Navigation**:
   * Create a sidebar with options like "Home" and "Search Bus" using st.radio() or st.selectbox().
5. **Home Page**:
   * Display platform info and a QR code linking to the mobile app for download.
6. **Search Bus Page**:
   * Use st.selectbox() for filtering by state, route, bus type, price, and ratings.
   * Query the database and display results using st.dataframe().
7. **Display Data**:
   * Display the filtered bus data in a table format using st.dataframe().
   * Show a message if no data matches the filter.
8. **Generate QR Code**:
   * Create and display a QR code for the mobile app download link using qrcode and PIL.

**CSS Styling:**

1. **Style Buttons**:
   * Customize button appearance (background color, padding, border-radius) using CSS.
2. **Title and Sidebar Styles**:
   * Set title font size, color, and margin.
   * Style sidebar elements (e.g., select boxes, sliders).
3. **Style Data Table**:
   * Style the data table with custom colors, borders, and hover effects.
4. **General Page Styling**:
   * Set background color for the page and text colors for the body.
5. **Image and QR Code Layout**:
   * Use Flexbox to align images and QR codes properly on the page.
6. **Icon Layout**:
   * Arrange icons (e.g., Play Store and RedBus) using Flexbox for a clean layout.

**Launch application:**

* **Run the App**:
  + In the terminal, run the command **streamlit run streamlit\_app.py.** The app will open in your browser at http://localhost:8501.
* **Sidebar Options**:

**Home**:

View the Redbus introduction and scan the **QR Code** to download the Redbus mobile app.

**Search bus:**

Allows you to search for buses based on filters like state, route, bus type, price range, and star rating.

* + **State**: Select a state from the dropdown menu to filter buses.
  + **Route**: Choose a route based on the selected state.
  + **Bus Type**: Select the bus type (e.g., Sleeper, Seater) to narrow down the search.
  + **Price Range**: Use the slider to set the price range of buses.
  + **Star Rating**: Adjust the slider to select the minimum star rating for buses.
* **Search for Buses**:
  + Click the **Search** button to view the bus results based on the selected filters.
* **View Results**:
  + The app will display a table with bus details such as route, bus type, price, available seats, and rating.
* **Close**:
  + To stop the app, press **Ctrl + C** in the terminal.

**Conclusion:**

The Redbus Data Scraping and Filtering with Streamlit project collects bus data from the Redbus website using **Selenium** for web scraping, saves the data in a **MySQL** database using **PyMySQL**, and provides an interactive user interface using **Streamlit**. The platform allows users to filter bus details based on status, route, and pricing, and it provides an easy-to-use interface for data exploration. Additionally, the software creates QR codes that link to the Redbus mobile app. This project optimizes trip planning by automating data gathering, allowing for quick filtering and analysis, and enriching the user experience through interactive visualizations.