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

**Objective**:

Create a web scraper to automate the extraction of bus route details, schedules, and other relevant data from the RedBus website across multiple states. The scraped data will be stored in an SQL database, with a user-friendly visualization and dynamic filtering enabled through a Streamlit app.

**Scope**:

* **Data Extraction**: Scrape detailed bus information, including route names, bus type, schedules, star ratings, prices, and seat availability.
* **Automation**: Automate navigation through various pages on the RedBus website, ensuring the collection of data across multiple states.
* **Data Storage**: Store the scraped data in a structured SQL database.
* **Visualization**: Develop a Streamlit app to visualize and interact with the scraped data through dynamic filters.

**Solution Overview**

The solution involves three primary components: **web scraping**, **SQL database integration**, and **Streamlit app development**.

1. **Web Scraping**:
   * **Approach**: Utilize Selenium for automating web interactions and data extraction from RedBus. The script will manage dynamic content loading and pagination.
   * **Steps**:
     + **Initialize Web Driver**: Launch the browser and navigate to the RedBus website.
     + **Load Web Page**: Open the URL of the target state, handling page load delays.
     + **Scrape Bus Routes**: Extract all bus route links and names, handling pagination to gather all data.
     + **Scrape Bus Details**: For each bus route, navigate to its page and scrape detailed data such as bus name, type, timings, ratings, price, and seat availability.
     + **Error Handling**: Implement error checks for missing elements or failed loading, ensuring continuity of the scraping process.
2. **SQL Database Integration**:
   * **Approach**: Store the scraped data using mysql-connector-python for MySQL.
   * **Steps**:
     + **Database Setup**: Create a database and define a table schema to store the bus route data.
     + **Data Insertion**: Insert the extracted data into the database, ensuring data integrity by handling any potential duplicates or errors.

**Database Schema:** Table: bus\_routes

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| id | INT | Primary Key (Auto-increment) |
| route\_name | TEXT | Bus Route information for each state transport |
| route\_link | TEXT | Link to the route details |
| busname | TEXT | Name of the bus |
| bustype | TEXT | Type of the bus |
| departing\_time | TIME | Departure time |
| duration | TEXT | Duration of the journey |
| reaching\_time | TIME | Arrival time |
| star\_rating | FLOAT | Rating of the bus |
| price | DECIMAL | Price of the ticket |
| seats\_available | INT | Number of seats available |

1. **Streamlit App Development**:
   * **Approach**: Build a Streamlit app to query the SQL database and visualize the data with dynamic filtering.
   * **Steps**:
     + **Database Connection**: Establish a connection to the SQL database from the Streamlit app.
     + **Query and Display Data**: Fetch and display the bus route data.
     + **Filter Implementation**: Implement filters in the app for route name, bus type, price range, star rating, and seat availability.

**Streamlit App Features**:

* + Display bus routes and schedule data in a table.
  + Provide dynamic filtering options such as route name, bus type, and price range.

**Implementation Steps**

1. **Web Scraping**:
   * **Initialize Web Driver**:
     + Launch a Selenium WebDriver to control the browser.
     + Navigate to the target URL and manage page loading issues.
   * **Scrape Bus Routes and Details**:
     + Identify the elements containing bus route information and extract the data.
     + For each bus route, click the link and scrape further details.
2. **SQL Database Integration**:
   * **Setup Database**:
     + Use mysql-connector-python to create the database and its structure.
     + Ensure proper data types and consistency across columns.
   * **Insert Data**:
     + Insert the scraped data into the SQL database, checking for duplicate entries and handling errors.
3. **Streamlit App Development**:
   * **Setup Streamlit**:
     + Install Streamlit (pip install streamlit) and set up a script (app.py).
   * **Database Connection**:
     + Connect to the SQL database using sqlite3 or mysql-connector-python.
   * **Query and Display Data**:
     + Fetch data from the SQL database and display it in the Streamlit app.
   * **Filtering and Analysis**:
     + Implement widgets for filtering the data by route name, bus type, price, etc.
     + Use the app to analyze the data interactively.

**Results**

The goal of the project is to:

* **Successfully scrape data** for at least 10 state-run bus routes, along with private bus routes.
* **Store the data** in a well-structured SQL database.
* **Develop an interactive Streamlit application** with efficient filtering and data display capabilities.
* **Ensure ease of use**: The application should be user-friendly and efficient.

**Data Set Requirements**

**Source**: Data will be scraped from the RedBus website at [RedBus](https://www.redbus.in/).

**Technical Tags**

* Web Scraping
* Selenium
* Streamlit
* SQL
* Data Analysis
* Python
* Interactive Applications