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

**Introduction**

The "Redbus Data Scraping and Filtering with Streamlit Application" project aims to enhance the transportation industry by automating the process of scraping, analyzing, and visualizing bus travel data. Using Selenium for web scraping and Streamlit for interactive data visualization and filtering, this project collects essential information from Redbus about bus routes, schedules, prices, and seat availability. It aims to streamline data collection processes and support data-driven decision-making to improve operational efficiency within the transportation sector.

**Features and Goals**

* Selenium Web Scraping: Collect detailed data from Redbus, covering both Government State Bus Transport and private bus services.
* Structured Database: Store the scraped data in an organized SQL database for efficient storage and retrieval.
* Streamlit Application: Provide an interactive, user-friendly web application that allows users to filter and view bus travel information based on different parameters.
* Data Visualization and Filtering: Offer powerful filtering options based on bus routes, schedules, prices, seat availability, and bus type.

**Solution Overview**

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

**1. Web Scraping**

**Approach:**

* Use Selenium to automate web browsing and data extraction from the RedBus website.
* Handle dynamic content loading, pagination, and potential pop-ups.

**Steps:**

1. **Initialize Web Driver:** Open and maximize the browser, and navigate to the RedBus website.
2. **Load Web Page:** Load the specific URL for the target state, handling any loading delays.
3. **Scrape Bus Routes:** Identify and extract all bus route links and names on the page, managing pagination to capture all routes.
4. **Scrape Bus Details:** Navigate to each bus route link and extract detailed information about available buses, such as name, type, departing time, duration, reaching time, star rating, price, and seat availability.
5. **Handle Errors:** Implement error handling for missing elements or loading failures, logging errors and continuing the scraping process.

**2. SQL Database Integration**

**Approach:**

* Use Python's sqlite3 or another SQL database connector (like mysql-connector-python for MySQL) to store the scraped data.

**Steps:**

1. **Database Setup:** Create a database and define a table structure to store bus route and schedule details.
2. **Data Insertion:** Insert the scraped data into the SQL database, ensuring data integrity and handling duplicates or errors.

|  |  |
| --- | --- |
| **Column names** | **Data types** |
| Id | INT PRIMARY KEY AUTO INCREMENT |
| Route Name | TEXT |
| Route Link | TEXT |
| Bus Name | TEXT |
| Bus Type | TEXT |
| Departing Time | TEXT |
| Duration | TEXT |
| Reaching Time | TEXT |
| Star Rating | TEXT |
| Price | TEXT |
| Seat availability | TEXT |

**3. Streamlit App Development**

**Approach:**

* Develop a Streamlit app to query and visualize the data from the SQL database.

**Steps:**

1. **Database Connection:** Establish a connection to the SQL database.
2. **Query Data:** Fetch data from the database to be displayed in the app.
3. **Filtering:** Use Streamlit components to filter the bus route name, price and star rating

**Streamlit App Features:**

* Display a table of bus routes and schedules.
* Provide filters for searching by route name, bus type, departing time, etc.

**Implementation Steps**

**1. Web Scraping**

1. **Initialize Web Driver:**
   * Use Selenium to open the browser and navigate to the target URL.
   * Handle page loading and pop-ups.
2. **Scrape Bus Routes and Details:**
   * Identify elements containing bus route links and names.
   * Navigate to each route link and extract detailed information.

**2. SQL Database Integration**

1. **Setup Database:**
   * Use sqlite3 to create a database and define table structures.
   * Whether check datatypes are same.

**3. Streamlit App Development**

1. **Setup Streamlit:**
   * Install Streamlit (pip install streamlit).
   * Create a Streamlit script (app.py).
2. **Database Connection:**
   * Use sqlite3 or another connector to connect to the database.
3. **Query and Display Data:**
   * Fetch data from the database and use Streamlit components to display it.
4. **Add Filters and Analysis:**
   * Use Streamlit widgets to add filters for route name, bus type, etc.
   * Use Streamlit to analyse data

By following this approach, you can automate the scraping of bus route details, store the data in an SQL database, and create an interactive Streamlit app for data analysis.

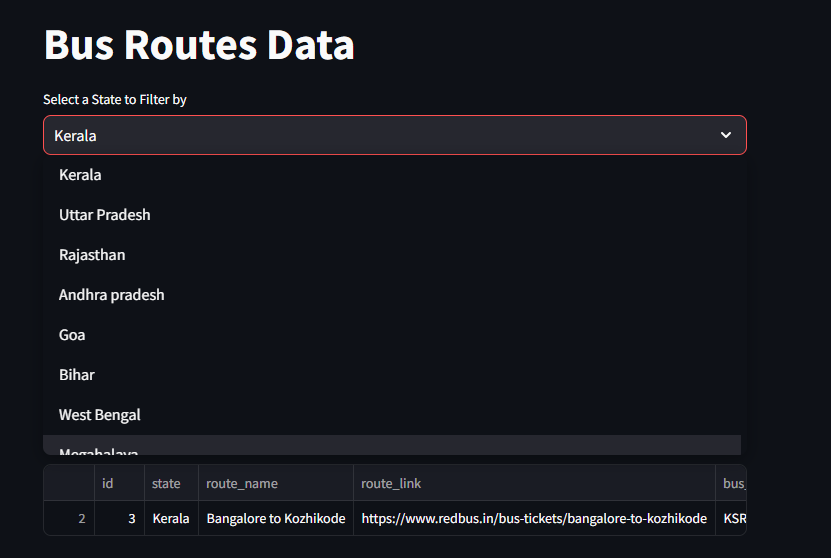
**Conclusion**

The **"Redbus Data Scraping and Filtering with Streamlit Application"** project combines data scraping, structured storage, and interactive filtering to provide a comprehensive solution for the transportation industry. This software enhances operational efficiency and strategic planning by providing real-time bus data and analysis tools.

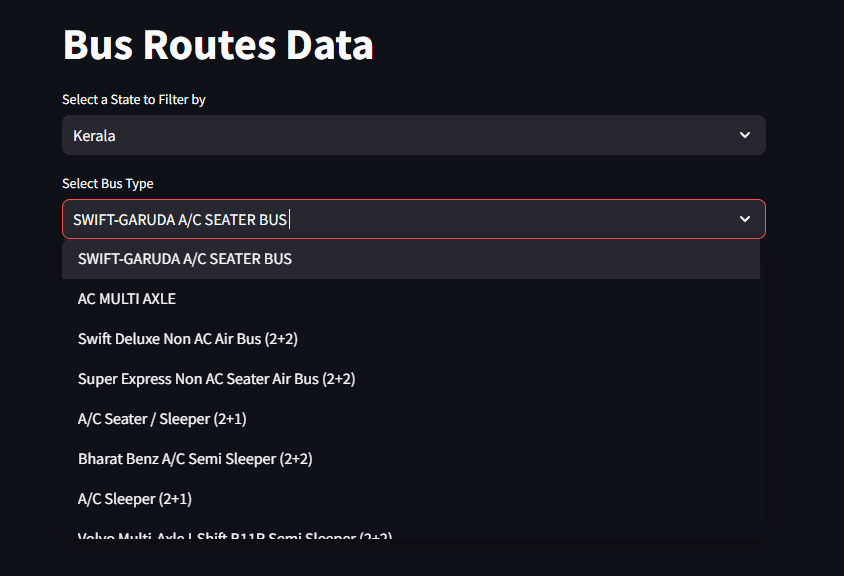
By automating data collection and offering robust filtering and visualization options, the application empowers stakeholders to make data-driven decisions for route optimization, pricing strategies, and capacity planning.

Streamlit UI screenshots

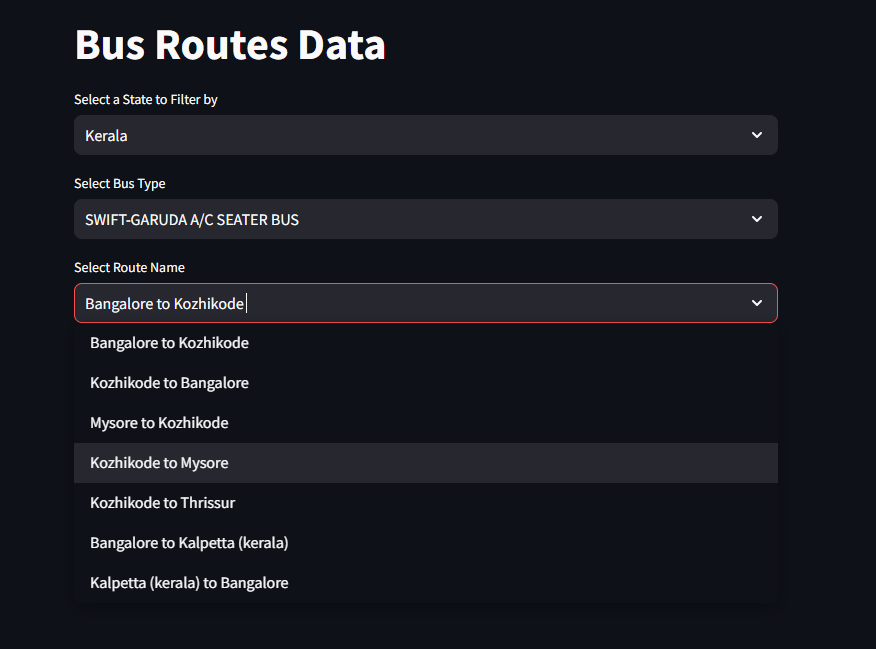
1. State Transport Operators list



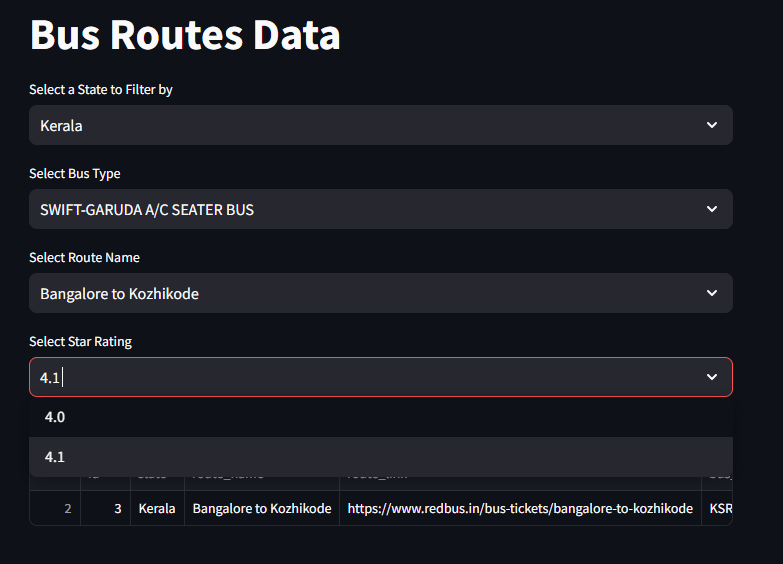
1. Bus Type:



1. Route Name:



1. Star Rating:



1. Data for the filtered item:

