**REDBUS DATA SCRAPING WITH SELENIUM &DYNAMIC FILTERING USING STREAMLIT**

**TABLE OF CONTENTS**

**CHAPTER NO TITLE PAGE NO**

1. **INTRODUCTION** 2
   1. OBJECTIVES 2
   2. SCOPE 2
   3. OUTCOMES 2
2. **IDEALOGY** 3
   1. PROBLEM STATEMENT 3
   2. SOLUTION OVERVIEW 3
3. **PROCEDURES** 4
   1. WEB SCRAPING 4

3. 2. SQL DATABASE INTEGRATION. 4

3.3. STREAMLIT APP DEVELOPMENT 5

1. **IMPLEMENTATION AND CODING 6**

**EXPLANATION**

4.1.WEB SCRAPING 6

4.2. SQL DATABASE INTEGRATION 8

4.3. STREAMLIT APP DEVELOPMENT 9

**5 STREAMLIT APPLICATION INTERFACE 10**

**6 CONCLUSION 12**

**CHAPTER 1**

**INTRODUCTION**

**1.1.OBJECTIVE:**

Develop a web scraper to automate the extraction of bus route details, schedules, and relevant information from the RedBus website for multiple states. Store the data in an SQL database and visualize it using a Streamlit app.

**1.2.SCOPE:**

* **Data Extraction:** Scrape bus route links, names, and detailed information for each route, including bus name, type, departing time, duration, reaching time, star rating, price, and seat availability.
* **Automation:** Automate navigation through multiple pages and states on the RedBus website.
* **Data Storage:** Store the scraped data in an SQL database.
* **Visualization:** Develop a Streamlit app to visualize and analyze the stored data.

**1.3.OUTCOMES**

The RedBus data scraper and visualization project successfully extracted and stored bus route details, schedules, and pricing information from the RedBus website. The data was then visualized in a Streamlit app, allowing users to explore and analyze the information through maps, charts, and tables. This tool provides valuable insights for travel planning, market analysis, and operational efficiency in the transportation industry.

**CHAPTER 2**

**IDEOLOGY**

**2.1. PROBLEM STATEMENT**

The "Redbus Data Scraping and Filtering with Streamlit Application" aims to revolutionize bus travel planning by automating data extraction and providing powerful visualization tools. Using Selenium, the project collects comprehensive bus route information, enabling data-driven decision-making and improved operational efficiency in the transportation industry.

**2.2.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.

**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.

**3. Streamlit App Development**

**Approach:**

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

**4.Data Analysis/Filtering using Streamlit:**

**Approach:**

* 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

**CHAPTER 3**

**PROCEDURES**

**3.1.WEB SCRAPING**

**Step by step procedure:**

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.

**3. 2. SQL DATABASE INTEGRATION.**

**Step by step procedure:**

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 | VARCHAR (100) |
| Duration | TEXT |
| Reaching Time | VARCHAR (100) |
| Star Rating | FLOAT |
| Price | FLOAT |
| Seat availability | INT |

**3. 3. STREAMLIT APP DEVELOPMENT**

**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:**

* **Search for bus routes:** Quickly find available bus routes based on departure and arrival cities, dates, and other filters.
* **View detailed route information:** Access detailed information about each route, including bus name, type, departure time, duration, reaching time, star rating, price, and seat availability.
* **Compare routes:** Compare multiple routes side-by-side to identify the best options based on your preferences.
* **Visualize data:** Explore data visualizations, such as maps and charts, to gain insights into bus route trends and patterns.
* **Export data:** Export the data in different formats (e.g., CSV, Excel) for further analysis or use.

**CHAPTER 4**

**IMPLEMENTATION AND CODING EXPLANATION**

**1. WEB SCRAPING**

**Implementation**

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.

**Coding Explanation.**

Initially the code defines a function state\_name\_link\_route that extracts bus route links and corresponding route names from the Redbus website.Here's a breakdown of its key components:

**Initialization**:

* Imports necessary Selenium modules for web interaction and time management.
* Defines a wait object for explicit waits.

**Function Definition:**

* Takes a path argument as input (likely an XPath expression).
* Initializes empty lists LINKS\_STATE\_NAME and ROUTE\_STATE\_NAME to store extracted links and route names.

**Looping Through Pages**:

* Iterates through a specified range (adjust based on the number of pages).
* Finds elements matching the provided path using driver.find\_elements(By.XPATH, path).
* Extracts links and route names from the found elements.
* Attempts to find the pagination element and check if there's a next button.
* If a next button exists, scrolls into view and clicks it using JavaScript.
* If there's no next button, breaks out of the loop.
* Handles potential exceptions like NoSuchElementException and ElementClickInterceptedException.

**Return Values:**

* Returns the LINKS\_STATE\_NAME and ROUTE\_ STATE\_NAME lists containing the extracted links and route names.

After getting the route link and route name,this code retrieves bus details for each route link extracted in the previous snippet.

**Initialization:**

* Creates a new Chrome driver (driver\_k) for this specific task.
* Defines empty lists to store bus details (names, types, start times, etc.).

**Iterating Through Links:**

* Iterates through each row in the df\_k dataframe, accessing the "Route\_link" and "Route\_name" columns.
* For each link:
* Navigates to the link using driver\_k.get(link).
* Waits for a short time.
* Tries to click elements to reveal bus details (optional).
* Tries to click a button to view buses (optional).
* Scrolls through the page using ActionChains until there are no more new elements.
* Extracts bus details using XPaths for various elements (bus name, type, start time, etc.).
* Appends extracted details to corresponding lists.
* Printing Success Message:

**2. SQL DATABASE INTEGERATION.**

**Implementation.**

* + Used mysql-connector-python to create a database and define table structures.
  + Whether check datatypes are same and did some modification.

**Coding Explanation.**

1. **Data Cleaning:**

* The code defines functions to clean and standardize the data in columns like "Total\_duration", "Price", "Seats\_Available", and "Ratings".
* It handles missing values and converts data to appropriate types (e.g., converting duration strings to time formats, prices to floats, and ratings to floats).

1. **Data Preparation:**

* code applies the cleaning functions to the dataframe, cleaning and preparing the data for insertion.
* It removes rows with missing values in both "Bus\_name" and "Bus\_type" to ensure data integrity.

1. **Database Connection:**

* The code establishes a connection to the MySQL database using the provided credentials.
* It handles potential connection errors gracefully.

1. **Data Insertion:**

* The code iterates through each row in the cleaned dataframe.
* It executes an SQL INSERT statement to insert the data into the "bus\_routes" table, mapping the dataframe columns to the corresponding table columns.
* It handles potential errors during insertion.
* It commits changes to the database and closes the connection.

1. **Error Handling:**

* The code incorporates try-except blocks to handle potential errors during data cleaning, database connection, and insertion.
* This helps prevent unexpected failures and provides informative messages.

**3. STREAMLIT APP DEVELOPMENT.**

**Implementation.**

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

**Coding explanation**

**Data Fetching:**

* Retrieves bus route information from a MySQL database, formats time columns for better readability, and creates a Pandas DataFrame for data manipulation.

**Streamlit App:**

* Sets up a user-friendly interface with a title, an image, and a link to the Redbus website, providing context and additional information.

**Sidebar Filters:**

* Creates interactive filters in the sidebar of the app, allowing users to select specific route names and adjust price, rating, and seat availability ranges based on their preferences.

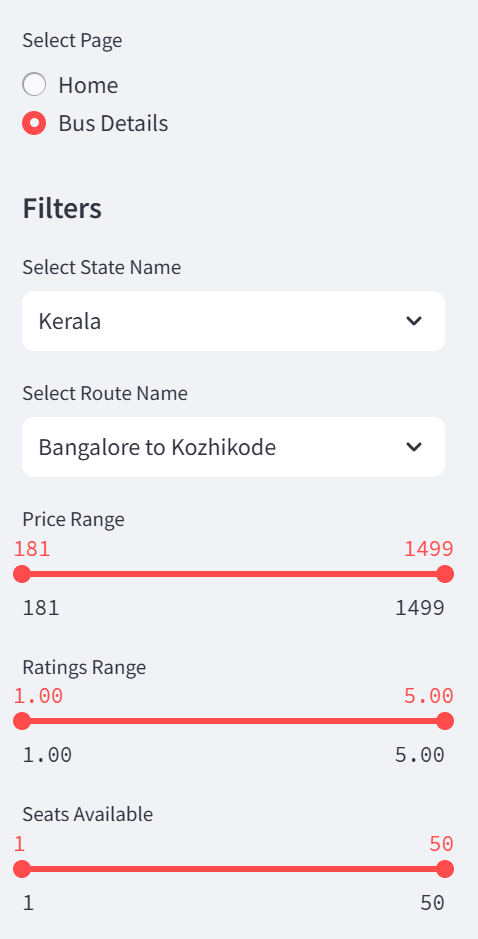
**Fetch Button:**

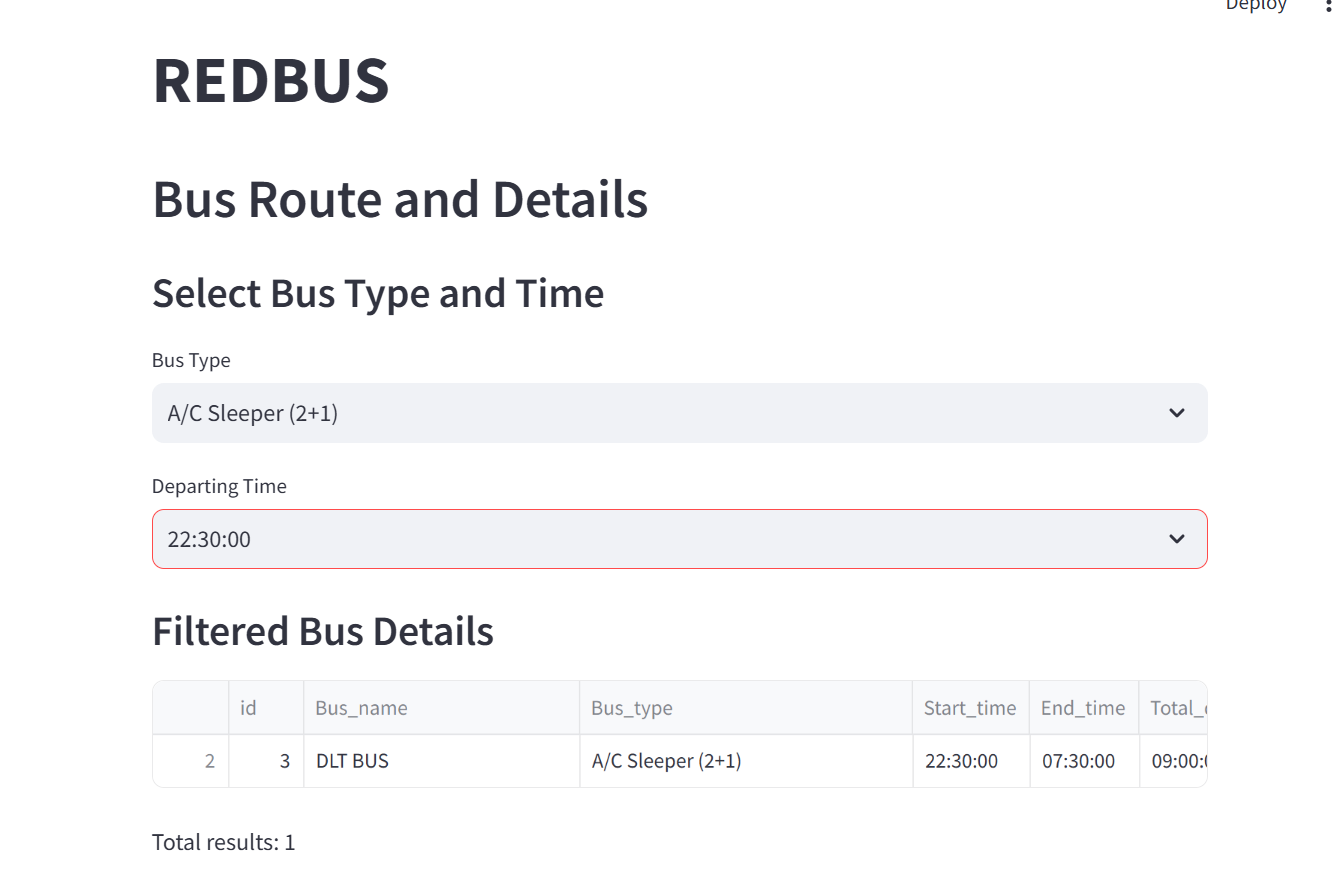
* When the "Fetch Bus Details" button is clicked, the app filters the data based on the selected filters and displays the filtered results in a clear and concise format, providing relevant information about the matching bus routes.

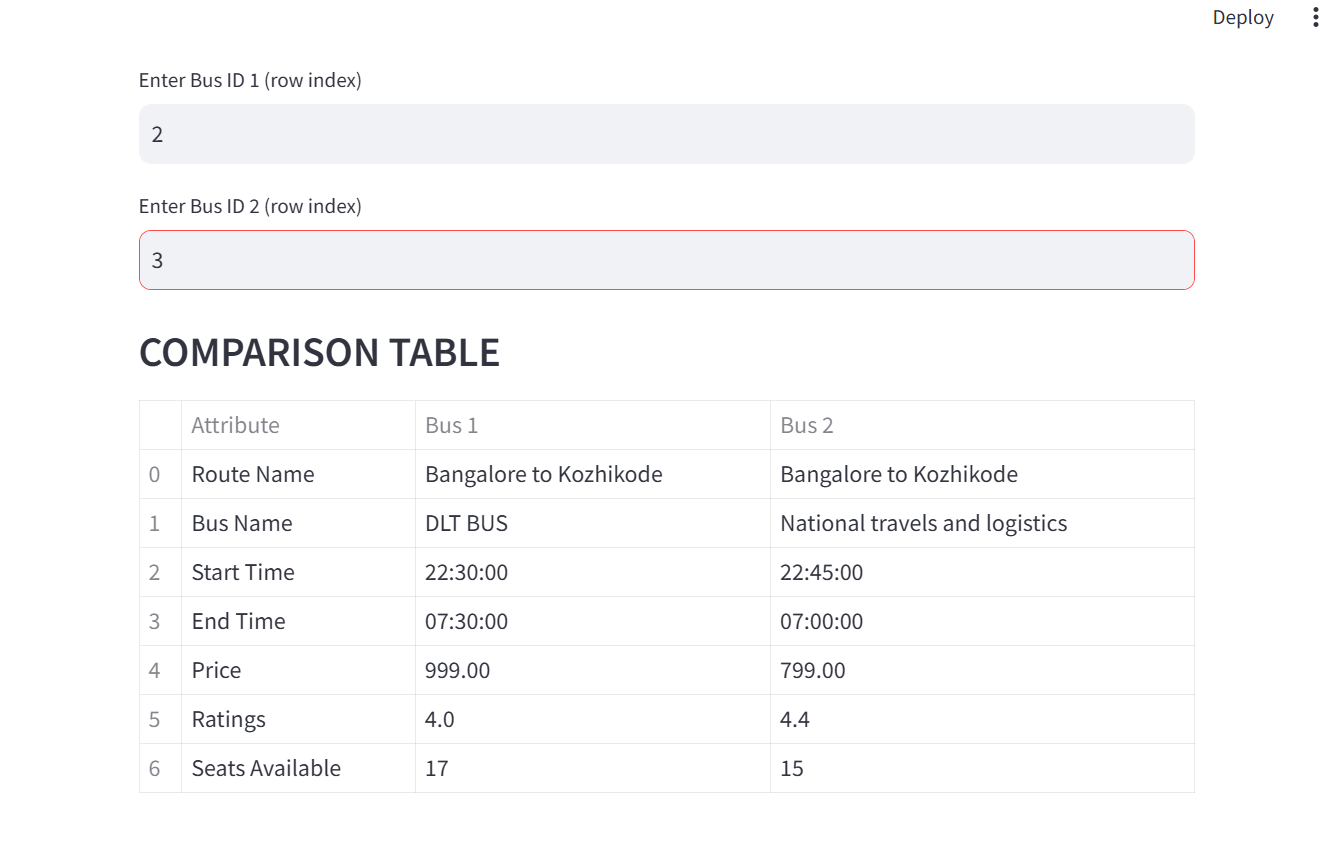
**CHAPTER 5**

**STREAMLIT APPLICATION INTERFACE**









**CHAPTER 6**

**CONCLUSION**

The "Redbus Data Scraping and Filtering with Streamlit Application" effectively addresses the challenges faced by the transportation industry by providing a comprehensive solution for data collection, analysis, and visualization. By automating the process of extracting bus route information using Selenium, the project significantly enhances operational efficiency and enables data-driven decision-making. The Streamlit application further empowers users to explore and filter the data, facilitating market analysis, customer service improvements, and competitive intelligence. The successful implementation of this project demonstrates the potential of leveraging technology to revolutionize the transportation sector.