Contents

[Project Overview 1](#_Toc172589604)

[Purpose 1](#_Toc172589605)

[Functionality 1](#_Toc172589606)

[ Selections and Filtering: 1](#_Toc172589607)

[ Data Fetching: 1](#_Toc172589608)

[ Display and Interaction: 2](#_Toc172589609)

[ Error Handling: 2](#_Toc172589610)

[Technologies Used 2](#_Toc172589611)

[Future Enhancements 2](#_Toc172589612)

[Code Overview 2](#_Toc172589613)

[states\_link.py 2](#_Toc172589614)

[db\_operation.py 2](#_Toc172589615)

[**RedBus Class (in the main script)** 3](#_Toc172589616)

[RedBus Class Explanation 3](#_Toc172589617)

[Data Collection 4](#_Toc172589618)

[Application Usage 4](#_Toc172589619)

[app.py 4](#_Toc172589620)

[Summary 5](#_Toc172589621)

[Reference: 5](#_Toc172589622)

# Project Overview

## Purpose

The purpose of this Streamlit application is to provide users with dynamic bus information based on various criteria such as state, route, price range, star rating, and bus type. Users can select different options to filter and view bus details fetched from a MySQL database.

## Functionality

### Selections and Filtering:

Users can choose a state and select a specific route within that state.

Price range, star rating, and bus type can be further filtered to narrow down search results.

### Data Fetching:

**States and Routes:** 11 Available states and corresponding routes are fetched dynamically from the database.

**Bus Information:** Queries are executed based on user selections to fetch relevant bus details matching the criteria.

### Display and Interaction:

Results are displayed in a tabular format showing bus ID, route name, bus name, type, departure time, duration, reaching time, star rating, price, and available seats.

A count of buses found is shown at the bottom right corner aligned to the right.

### Error Handling:

Errors related to database queries or fetching are displayed as error messages to the user.

## Technologies Used

* **Python Libraries:** Pandas for data manipulation, Streamlit for building interactive web applications, and mysql-connector for connecting SQL to python.
* **Database:** MySQL database for storing and retrieving bus-related data.
* **Web Framework:** Streamlit for creating a user-friendly dashboard interface.
* **Selenium**: To fetch the data from website.

## Future Enhancements

* Implementing additional features such as sorting columns in the bus information table.
* Improving error handling and user feedback.

## Code Overview

### states\_link.py

This script retrieves the list of states and their corresponding links from a web page.

#### Purpose:

* **get\_states(url, driver)**: Retrieves a list of states and their links from a given URL using Selenium.

#### Usage:

* get\_states(url, driver): Returns a list of dictionaries, each containing a state name and its corresponding link.

### db\_operation.py

This script handles database operations such as creating tables, inserting data, and fetching results.

#### Purpose:

* **create\_table(bus\_details)**: Creates a MySQL table named bus\_routes and inserts bus details into it.

#### Usage:

* These functions are used to create tables and insert data into tables.

### **RedBus Class (in the main script)**

This is the main class responsible for scraping bus information from the RedBus website.

### RedBus Class Explanation

#### Initialization (\_\_init\_\_ method)

* **\_\_init\_\_(self, url, state)**: Initializes the class with a RedBus URL and a state name.
* **Attributes**:
  + routes\_details: List to store details of routes fetched.
  + bus\_details: List to store details of buses fetched.

#### get\_routes(self, state) Method

* **Purpose**: Retrieves all routes available for a given state from the RedBus website.
* **Steps**:
  + Uses Selenium to navigate to the provided state URL.
  + Finds pagination elements to iterate through pages.
  + Collects route names and their corresponding links.
  + Stores each route's details in self.routes\_details.

#### get\_bus\_details(self, state, url, route) Method

* **Purpose**: Retrieves bus details for a specific route and date.
* **Steps**:
  + Navigates to the provided route URL.
  + Checks for the availability of buses for the current date.
  + If no buses are found, checks the next three dates.
  + If buses are found, extracts details such as bus name, type, departing time, duration, reaching time, star rating, available seats, and price.
  + Stores each bus's details in self.bus\_details.

#### remove\_duplicate(self, state, data) Method

* **Purpose**: Removes duplicate entries from self.bus\_details.
* **Steps**:
  + Converts self.bus\_details to a Pandas DataFrame for easy manipulation.
  + Drops duplicate rows based on all columns.
  + Saves the cleaned data to a CSV file named {state}\_scarpped\_data.csv.
  + Returns a list of dictionaries (current\_data) containing unique bus details.

#### scroll\_to\_bottom(self) Method

* **Purpose**: Scrolls to the bottom of the web page.
* **Steps**:
  + Continuously scrolls down until the bottom of the page is reached to get all the bus information
  + Uses Selenium's Keys.PAGE\_DOWN to simulate scrolling.

#### click\_view\_buses(self) Method

* **Purpose**: Clicks on "View Buses" to expand bus details on the webpage.
* **Steps**:
  + Finds and clicks on all "View Buses" elements on the page using Selenium.
  + Scrolls to each element and clicks to expand details.

#### extract\_bus\_details(self, state, url, route, current\_date) Method

* **Purpose**: Extracts details of each individual bus for a given route and date.
* **Steps**:
  + Finds and extracts bus details such as name, type, departing time, duration, reaching time, star rating, available seats, and price using Selenium.

#### get\_today\_date(self) Method

* **Purpose**: Returns today's date in a specific format (day month).

## Data Collection

* **Scraping Logic**: Uses Selenium to interact with the RedBus website, navigating through pages, clicking elements, and extracting bus information dynamically.
* **Storage**: Stores fetched data in lists (self.routes\_details and self.bus\_details) and stored in the MYSQL database and eventually in a CSV file ({state}\_scarpped\_data.csv).

## Application Usage

* **Setup**: Requires Chrome WebDriver (webdriver.Chrome()) and Streamlit (import streamlit as st).
* **Execution**: Runs the RedBus class for each state and fetches bus details.
* **Interface**: Provides a Streamlit-based interface (app.py) with pages for dashboard and bus information display.
* **Navigation**: Users can navigate between pages to view summary/dashboard or detailed bus information.

### app.py

This script integrates the Streamlit framework to create a user interface for interacting with the bus data.

It contains two pages

Dashboard(welcome.py) : It displays the overview of project.

#### View Buses(view\_buses.py) :

* **It is used to fetch the stored bus information from the MySQL database and handle the filtering operations**
* **get\_result(results)**: Converts fetched database results into a Pandas DataFrame for display purposes.
* **get\_states\_list()**: Fetches a list of states available in the bus\_routes table.
* **get\_routes\_list(state)**: Fetches a list of routes for a specific state from the bus\_routes table.
* **get\_bus\_data\_from\_db(state, route, min\_price, max\_price, min\_rating, max\_rating, bus\_type)**: Fetches bus data based on user-defined filters from the bus\_routes table.
* **get\_min\_max\_value(value)**: Helper function to parse user-selected filter values.

#### Purpose:

* Integrates the UI components using Streamlit for displaying bus information based on user inputs such as state, route, price range, rating, and bus type.

#### Usage:

* Run streamlit run app.py to launch the web application.
* Select state, route, price range, star rating, and bus type from the dropdown menus to filter bus information dynamically.

**URL LINK:** <http://localhost:8501/>

Files: main.py, db\_operation.py, states\_link.py, app.py, welcome.py, view\_buses.py

Output Files: <state\_name>\_scrapped\_data.csv (11 files)

SQL File: datascrapping\_sql\_queries.sql (it contains some sample queries)

## Summary

* **Purpose**: Automates the retrieval of bus details from RedBus for multiple states.
* **Technologies**: Uses Python, Selenium for web scraping, Pandas for data manipulation, and Streamlit for creating a web interface.
* **Operation**: Scrapes data, removes duplicates, stores in CSV, and provides a user-friendly interface to view bus information.

## Reference:

* <https://www.redbus.in/>
* <https://docs.google.com/document/d/1lWJU0W7BQC2x3_wv5Wlu6l6nHUgobxgXaL4V8vFu3FQ/edit#heading=h.58k9i7iv24ca>
* [**https://docs.streamlit.io/get-started/installation**](https://docs.streamlit.io/get-started/installation)
* [**https://www.selenium.dev/documentation/webdriver/elements/locators/**](https://www.selenium.dev/documentation/webdriver/elements/locators/)
* <https://www.datacamp.com/tutorial/streamlit>