**Project-Bus Availability Finder**

Web Scraping using Selenium

# Overview:

This project is designed to scrape bus availability data from the RedBus website. The goal is to extract critical information such as bus names, types, prices, timings, seat availability, and user ratings for multiple routes. The extracted data is organized into a structured DataFrame, which can be further used for analysis or as a data source for applications like a Streamlit app.

# Tools and Libraries Used:

- Python: Core language for scripting.  
- Selenium: For web automation and scraping dynamic content.  
- Pandas: For organizing the scraped data into a DataFrame for analysis and storage.

# Key Components of the Project:

## 1. Web Scraping with Selenium:

- Website: The target website is RedBus-various states  
- Browser Automation: Selenium WebDriver is used to automate browser actions, including navigating pages, clicking on elements, and scrolling through dynamic content.  
- Data Scraping: Data is scraped by identifying specific HTML elements on the page (using XPaths) that contain relevant bus information.

## 2. Data Collected:

- Route Information: Route names and their corresponding URLs.  
- Bus Details (for each route):  
 - Bus Name: The name of the operator or bus company.  
 - Bus Type: The type of bus (e.g., AC, Sleeper, Semi-Sleeper).  
 - Start and End Time: Departure and arrival times of the bus.  
 - Total Duration: The total journey time.  
 - Price: The ticket fare for the journey.  
 - Seats Available: Number of seats available at the time of scraping.  
 - Ratings: User ratings (if available).

## 3. Pagination Handling:

The website has multiple pages for available routes. The scraper navigates through these pages using pagination controls. This ensures that all routes are covered in the scraping process.

## 4. Dynamic Content Handling:

The bus details on each route page are dynamically loaded after interacting with the website (clicking and scrolling). Selenium automates these actions to reveal all the necessary information.

# Steps Involved:

## 1. Initialization:

The project starts by importing necessary libraries, such as Selenium, Pandas, and time utilities. Selenium WebDriver is launched, and the target website is loaded.

## 2. Route Link Extraction:

The function `link\_route()` scrapes bus route names and their respective links from the pages of the website. These are stored in lists and later converted into a Pandas DataFrame.

## 3. Bus Details Extraction:

For each bus route, the script navigates to the corresponding link, clicks on elements to reveal bus details, performs scrolling actions to load dynamic content, and scrapes the following information: Bus names, types, timings, duration, ratings, prices, and seat availability.

## 4. Data Storage:

All the scraped data is stored in a Pandas DataFrame for further use or analysis. The resulting DataFrame contains columns such as Bus\_name, Bus\_type, Start\_time, End\_time, Total\_duration, Price, Seats\_Available, Ratings, Route\_link, and Route\_name.

# DataFrame Structure:

|  |  |
| --- | --- |
| Column | Description |
| Bus\_name | Name of the bus operator or service. |
| Bus\_type | Type of bus (e.g., AC, Sleeper, Semi-Sleeper). |
| Start\_time | Departure time of the bus. |
| End\_time | Arrival time of the bus. |
| Total\_duration | Total journey time from start to end. |
| Price | Ticket fare for the journey. |
| Seats\_Available | Number of seats available for booking. |
| Ratings | User ratings for the bus service (if available). |
| Route\_link | Hyperlink to the specific route page on the website. |
| Route\_name | Name of the bus route (e.g., "Gangtok to Siliguri"). |

# Challenges Encountered:

- Dynamic Loading: Many elements on the page (like bus details) load dynamically after clicking or scrolling. This was handled using Selenium's `ActionChains` and `WebDriverWait`.  
- Pagination: The website uses pagination for routes, so the scraper had to interact with the pagination elements to scrape all available routes.  
- Click Handling: In some cases, elements could not be clicked directly, requiring the use of JavaScript to scroll and click.

# Potential Use Cases:

- Streamlit App: The collected data can be used to power a Streamlit app that filters and displays bus availability based on user preferences (e.g., price, rating, or seat availability).  
- Data Analysis: Insights into bus services (e.g., most popular routes, price ranges, user satisfaction) can be derived from the DataFrame.  
- API: The data can be exported to an API for real-time use in travel booking platforms or other services.

# Future Improvements:

- Error Handling: Adding more robust error handling for cases where certain bus details or routes may be unavailable.  
- Performance: Optimization of the scraping process by reducing sleep intervals and using multithreading for faster execution.  
- Scalability: Extending the scraper to cover more routes and regions.

Bus Data Aggregation and Insertion into MySQL Database

# Overview

This focuses on aggregating bus-related data from various CSV files, cleaning and processing the data, and storing the cleaned information in a MySQL database. The dataset includes information on bus services such as bus names, types, timings, ticket prices, seat availability, and ratings. The ultimate goal is to create a structured database that can be queried for insights on bus availability and service details.

# Key Objectives

1. Merge multiple CSV files containing bus-related data.

2. Clean and preprocess the data for consistency.

3. Insert the cleaned data into a MySQL database.

4. Ensure that the data is stored in a well-structured format for future queries and analysis.

# Tools and Technologies Used

- Python: Core programming language used for data manipulation, processing, and database interaction.

- Pandas: Library used for reading, concatenating, and cleaning data from CSV files.

- MySQL: Database used for storing the cleaned and processed data.

- pymysql: Python library used to interact with the MySQL database.

# Steps Involved

## 1. Data Aggregation

- **CSV File Reading**: Multiple CSV files containing bus information (e.g., `ap.csv`, `bsrtc.csv`, etc.) are read into Pandas DataFrames using `pd.read\_csv()`.

- **Merging Data**: All the DataFrames are concatenated into a single DataFrame using `pd.concat()`. This step combines data from different sources into a unified dataset for further processing.

## 2. Data Cleaning

- **Dropping Missing Data**: Rows containing missing or `NaN` values are removed using `dropna()`.

- **Column Removal**: Unnecessary columns (such as `Unnamed: 0`) are dropped to keep the dataset clean and focused on relevant information.

- **Price Cleaning**: The `Price` column is cleaned by extracting the numeric value of the ticket price.

- **Ratings Cleaning**: The `Ratings` column is cleaned by extracting only the first part (numeric rating).

## 3. Data Preprocessing

The script iterates through the DataFrame, ensuring that the cleaned data is in a consistent format suitable for insertion into the MySQL database.

## 4. MySQL Database Interaction

- Database Connection: Using the `pymysql` library, a connection is established with a MySQL database.

- Table Creation: A table named `buses` is created within the `redbusproject` database. The columns include `Bus\_name`, `Bus\_type`, `Start\_time`, `End\_time`, `Total\_duration`, `Price`, `Seats\_Available`, `Ratings`, `Route\_link`, and `Route\_name`.

- Data Insertion: For each row in the cleaned DataFrame, the bus data is inserted into the `buses` table.

## 5. Output

The final cleaned dataset is saved into a CSV file (`combined.csv`), and the data is inserted into the MySQL database for further use and analysis.

# Data Fields

- \*\*Bus\_name\*\*: Name of the bus operator.

- \*\*Bus\_type\*\*: Type of bus (AC, Sleeper, etc.).

- \*\*Start\_time\*\*: Departure time of the bus.

- \*\*End\_time\*\*: Arrival time of the bus.

- \*\*Total\_duration\*\*: Duration of the journey.

- \*\*Price\*\*: Ticket price.

- \*\*Seats\_Available\*\*: Number of seats available for booking.

- \*\*Ratings\*\*: User rating of the bus service.

- \*\*Route\_link\*\*: URL to the bus route page (if applicable).

- \*\*Route\_name\*\*: Name of the bus route (e.g., 'Hyderabad to Bangalore').

# Challenges Encountered

- **Data Inconsistencies**: Different CSV files had variations in how information was structured (e.g., price values with extra words). Special handling was required to ensure consistent formatting.

- **Data Cleaning**: Some columns contained unwanted or irrelevant data that had to be cleaned or dropped to maintain data quality.

- **Database Insertion**: Ensuring that the cleaned data was inserted into the MySQL table in the correct format required careful handling of data types.

# Future Enhancements

- **Error Handling**: Improving error handling during the data cleaning and database insertion phases to handle unexpected data issues or connection errors more gracefully.

- **Data Expansion**: Expanding the dataset to include additional routes, more bus operators, and real-time data scraping.

- **Performance Optimization**: Enhancing the performance of the script by optimizing data processing and reducing the time taken for database transactions.

Streamlit Application Building

# Overview

A web application built using Streamlit that helps users search for available buses based on specific routes, price range, ratings, and seat availability. The application processes a pre-aggregated dataset of bus information, allowing users to interactively filter and visualize relevant data for their journey. The filtered results include clickable route links for more detailed information.

# Objectives

The primary objectives of this project are:

1. To provide users with an intuitive interface to filter buses based on:

- Route Name

- Price Range

- User Ratings

- Seat Availability

2. To clean and preprocess bus data from various sources for consistency and usability.

3. To integrate bus route links as clickable hyperlinks for easy access to more details about a particular route.

# Tools and Technologies

- **Python**: For data manipulation and backend processing.

- **Pandas**: For data cleaning, preprocessing, and filtering.

- **Streamlit**: For building the web-based interface and displaying results interactively.

# Steps Involved

## 1. Data Loading

Data is loaded from the `combined.csv` file into a Pandas DataFrame. The `Price` and `Ratings` columns are cleaned by converting them into appropriate numeric formats for filtering. Additionally, the `Seats\_Available` column is processed to extract the number of available seats.

## 2. Sidebar Filters

The sidebar of the application provides multiple filters to the user for narrowing down their search. These filters include:

- **Route**: Allows the user to select a specific bus route.

- **Price Range**: Allows filtering buses based on ticket prices.

- **Minimum Rating**: Allows filtering buses with a minimum user rating.

- **Seats Available**: Allows filtering buses based on available seat counts.

## 3. Data Filtering and Results Display

Based on the user’s selected filters, the data is filtered to display buses that meet the criteria. The results are displayed in a table format, including the following columns: `Bus\_name`, `Bus\_type`, `Start\_time`, `End\_time`, `Total\_duration`, `Price`, `Seats\_available`, `Ratings`, and `Route\_link`. The `Route\_link` column contains clickable links that open the bus route details in a new tab.

# Conclusion

This project allows users to easily search and filter available bus services, providing them with valuable insights into bus timings, prices, seat availability, and ratings. The integration of clickable route links enhances the user experience by providing direct access to further details about each bus route.