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

### **1. Introduction**

**Problem Statement:**

The "Redbus Data Scraping and Filtering with Streamlit Application" aims to revolutionize the transportation industry by providing a comprehensive solution for collecting, analyzing, and visualizing bus travel data. By utilizing Selenium for web scraping, this project automates the extraction of detailed information from Redbus, including bus routes, schedules, prices, and seat availability. By streamlining data collection and providing powerful tools for data-driven decision-making, this project can significantly improve operational efficiency and strategic planning in the transportation industry.

**Data Set:**

* **Source:** Data will be scraped from the Redbus website.

Link- <https://www.redbus.in/>

* **Format:** The scraped data will be stored in a SQL database.

**Required Fields:** Bus routes Link,Bus route Name, Bus name, Bus Type(Sleeper/Seater),  Departing Time, Duration, Reaching\_Time, Star-rating, Price, Seat\_availability.

**Data Set Requirements & Explanation:**

The scraped dataset for this project should contain detailed information about bus services available on Redbus, covering various aspects critical to travelers and service providers. Here is a breakdown of the fields required:

* **Bus Routes Name:** This field captures the start and end locations of each bus journey, providing crucial information about the routes serviced.
* **Bus Routes Link**: Link for all the route details.
* **Bus Name:** The name of the bus or the service provider, which helps in identifying the specific operator.
* **Bus Type (Sleeper/Seater/AC/Non-AC):** This field specifies whether the bus is a sleeper or seater type, indicating the seating arrangements and comfort level offered.
* **Departing Time:** The scheduled departure time of the bus, essential for planning travel schedules.
* **Duration:** The total duration of the journey from the departure point to the destination, helping passengers estimate travel time.
* **Reaching Time:** The expected arrival time at the destination, allowing for better planning of onward travel or activities.
* **Star Rating:** A rating provided by passengers, indicating the quality of service based on factors such as comfort, punctuality, and staff behavior.
* **Price:** The cost of the ticket for the journey, which can vary based on factors like bus type and demand.

**Seat Availability:** The number of seats available at the time of data scraping, giving real-time insight into the occupancy levels.

**Project Overview:**

For this project, data will be scraped using Selenium from the Redbus website, stored in a SQL database, and a Streamlit application will be created to dynamically filter and display the data. The objective is to offer an intuitive user interface for data exploration and automate the data extraction procedure.

**Objectives:**

* To scrape bus route details from Redbus.
* To store the scraped data in an SQL database.
* To create a Streamlit application for dynamic filtering and visualization of the data.

### **2. Tools and Technologies**

* **Python:** For scripting and data manipulation.
* **Selenium:** For web scraping.
* **MySQL:** For data storage.
* **Streamlit:** For creating the web application.
* **Jupyter Notebook:** For webscraping and data storage.
* **Pycharm**: For streamlit application

### **3. Project Setup**

**Prerequisites:**

* Python installed on your machine.
* Required Python libraries: selenium, sqlalchemy, streamlit, pandas, pymysql.
* Chrome WebDriver for Selenium.

**Installation:**

pip install selenium streamlit

pip install pandas

pip install pymysql

pip install streamlit

pip install webdriver-manager

### **4. Web Scraping using Selenium**

The code performs web scraping of bus transport data from the Redbus website and stores the scraped data in a MySQL database. It utilizes the Selenium library to interact with the web pages, extract relevant information, and handle dynamic content. The pymysql library is used to connect to the MySQL database and store the scraped data.

### **Web Scraping Process**

1. Initialization:

The Chrome WebDriver is initialized and maximized to ensure proper rendering of the web pages.

# Initialize the WebDriver

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

driver.maximize\_window()

1. Open the desired Website:

The WebDriver navigates to the specific route link and waits for the page to load.

driver.get("https://www.redbus.in/online-booking/rsrtc/?utm\_source=rtchometile")

time.sleep(3) # Allow time for the page to load

1. Function to retrive the route links and names:

# Function to retrieve bus route links and route names

def link\_route(path):

LINKS = []

ROUTE = []

wait = WebDriverWait(driver, 10)

while True:

try:

paths = driver.find\_elements(By.XPATH, path)

for links in paths:

d = links.get\_attribute("href")

if d:

LINKS.append(d)

for route in paths:

ROUTE.append(route.text)

1. Navigating to the next pages:

try:

active\_page\_element = driver.find\_element(By.XPATH, "//div[@class='DC\_117\_pageTabs DC\_117\_pageActive']")

active\_page\_number = active\_page\_element.text

next\_page\_number = str(int(active\_page\_number) + 1)

next\_page\_button\_xpath = f"//div[@class='DC\_117\_paginationTable']//div[text()='{next\_page\_number}']"

next\_page\_button = wait.until(EC.presence\_of\_element\_located((By.XPATH, next\_page\_button\_xpath)))

driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)

time.sleep(1)

next\_page\_button.click()

print(f"Navigating to page {next\_page\_number}")

time.sleep(10)

except (NoSuchElementException, TimeoutException):

print("No more pages to paginate or pagination element not found")

break

except Exception as e:

print(f"Error occurred: {str(e)}")

break

return LINKS, ROUTE

1. Retrieve the route links:

Links, route = link\_route("//a[@class='route']")



1. Save route data:

df\_routes = pd.DataFrame({"Route\_name": ROUTE, "Route\_link": LINKS})

df\_routes.to\_csv("RSRTC.csv", index=False)

print("Route details saved successfully.")

# Close the first driver

driver.quit()

1. Initialize second WebDriver for bus details:

The script scrapes the route names and their links by manually clicking through each page and extracting the data.

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

driver.maximize\_window()

1. Scraping Bus Details:

The script iterates through each route link to scrape detailed bus information such as bus name, type, departing and reaching times, star rating, price, and seat availability.

* # Bus\_names = []
* Bus\_types = []
* Departure = []
* Arrival = []
* Ratings = []
* Total\_Duration = []
* Prices = []
* Seats\_Available = []
* Route\_names = []
* Route\_links = []
* # Loop through route links to extract bus details
* for i, r in df\_routes.iterrows():
* link = r["Route\_link"]
* routes = r["Route\_name"]
* driver.get(link)
* time.sleep(2)
* try:
* view\_buses\_button = driver.find\_element(By.XPATH, "//div[@class='button']")
* view\_buses\_button.click()
* except:
* continue
* time.sleep(2)
* scrolling = True
* while scrolling:
* old\_page\_source = driver.page\_source
* ActionChains(driver).send\_keys(Keys.PAGE\_DOWN).perform()
* time.sleep(5)
* new\_page\_source = driver.page\_source
* if new\_page\_source == old\_page\_source:
* scrolling = False

1. Extract bus details:

bus\_name = driver.find\_elements(By.XPATH, "//div[@class='travels lh-24 f-bold d-color']")

bus\_type = driver.find\_elements(By.XPATH, "//div[@class='bus-type f-12 m-top-16 l-color evBus']")

start\_time = driver.find\_elements(By.XPATH, "//\*[@class='dp-time f-19 d-color f-bold']")

end\_time = driver.find\_elements(By.XPATH, "//\*[@class='bp-time f-19 d-color disp-Inline']")

total\_duration = driver.find\_elements(By.XPATH, "//\*[@class='dur l-color lh-24']")

price = driver.find\_elements(By.XPATH, '//div[@class="fare d-block"]//span')

seats = driver.find\_elements(By.XPATH, "//div[contains(@class, 'seat-left')]")

try:

rating = driver.find\_elements(By.XPATH, "//div[@class='clearfix row-one']/div[@class='column-six p-right-10 w-10 fl']")

except:

rating = []

1. Append data to lists:

for bus in bus\_name:

Bus\_names.append(bus.text)

Route\_links.append(link)

Route\_names.append(routes)

for bus\_type\_elem in bus\_type:

Bus\_types.append(bus\_type\_elem.text)

for start\_time\_elem in start\_time:

Departure.append(start\_time\_elem.text)

for end\_time\_elem in end\_time:

Arrival.append(end\_time\_elem.text)

for total\_duration\_elem in total\_duration:

Total\_Duration.append(total\_duration\_elem.text)

for ratings\_elem in rating:

Ratings.append(ratings\_elem.text if ratings\_elem else "N/A")

for price\_elem in price:

Prices.append(price\_elem.text)

for seats\_elem in seats:

Seats\_Available.append(seats\_elem.text)

print("Bus details extracted successfully.")

1. Save the bus data:

data = {

'Route\_name': Route\_names,

'Route\_link': Route\_links,

'Bus\_name': Bus\_names,

'Bus\_type': Bus\_types,

'Departing\_time': Departure,

'Total\_duration': Total\_Duration,

'Reaching\_time': Arrival,

'Star\_Rating': Ratings,

'Price': Prices,

'Seats\_Available': Seats\_Available

}

df\_buses = pd.DataFrame(data)

1. Convert the saved data into csv files:

df\_buses.to\_csv("redbus1\_details.csv", index=False)

print("Bus details saved successfully.")

1. Close the webdriver:

Driver.quit()

**SQL Data Storage Process**

1. Connecting to the MySQL Database:

The script establishes a connection to the MySQL database using pymysql.

* # Connect to the MySQL database
* from mysql import connector
* connection =connector.connect(
* host="localhost",
* user="root",
* password="Mona@999"
* )
* mycursor = connection.cursor()

1. Creating the Database Schema:

The script creates a table named bus\_details if it doesn't already exist. The table schema is designed to accommodate the scraped data, with appropriate data types for each column.

* # query="create database if not exists redbusproject"
* mycursor.execute(query)

#query ="use redbusproject"

mycursor.execute(query).

#mycursor.execute("""

CREATE TABLE IF NOT EXISTS bus\_details (

ID INT AUTO\_INCREMENT PRIMARY KEY,

Route\_name VARCHAR(255) NULL,

Route\_link VARCHAR(255) NULL,

Bus\_name VARCHAR(255) NOT NULL,

Bus\_type VARCHAR(255) NOT NULL,

state VARCHAR(255) NOT NULL,

Departing\_time VARCHAR(255) NOT NULL,

Total\_duration VARCHAR(255) NOT NULL,

Reaching\_time VARCHAR(255) NOT NULL,

Star\_Rating FLOAT NULL,

Price DECIMAL(10,2),

Seats\_available VARCHAR(255) NOT NULL

)

""")

print("Table created successfully")

* Explanation of Table Columns:
  + id: An auto-incrementing primary key to uniquely identify each record.
  + route\_name: The name of the bus route.
  + route\_link: The URL link to the bus route page.
  + busname: The name of the bus operator.
  + bustype: The type of bus (e.g., AC, Non-AC, Sleeper).
  + departing\_time: The departure time of the bus, stored in DATETIME format.
  + duration: The duration of the bus journey.
  + reaching\_time: The arrival time of the bus, stored in DATETIME format.
  + star\_rating: The star rating of the bus, stored as a FLOAT.
  + price: The price of the bus ticket, stored as a DECIMAL with precision up to two decimal places.
  + seats\_available: The number of seats available on the bus, stored as an INT.
* d) Inserting Data into the Database:

The script iterates through the list of bus details (bus\_details) and inserts each record into the bus\_details table.

* # insert\_query = """
* INSERT INTO bus\_details (
* Route\_name,
* Route\_link,
* Bus\_name,
* Bus\_type,
* state,
* Departing\_time,
* Total\_duration,
* Reaching\_time,
* Star\_Rating,
* Price,
* Seats\_available
* ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s,%s)
* """
* for index, row in dfbus.iterrows():
* mycursor.execute(insert\_query, (
* row['Route\_name'],
* row['Route\_link'],
* row['Bus\_name'],
* row['Bus\_type'],
* row['State'],
* row['Departing\_time'],
* row['Total\_duration'],
* row['Reaching\_time'],
* row['Star\_Rating'],
* row['Price'],
* row['Seats\_Available']
* ))
* connection.commit()
* print("Data inserted successfully!")
* Explanation of Insertion Process:
  + The cursor.execute method is used to execute the SQL INSERT statement for each record in the bus\_details list.
  + The %s placeholders are used to safely insert the data into the SQL query, preventing SQL injection attacks.
  + Each record from the bus\_details list is unpacked and inserted into the corresponding columns of the bus\_details table.

### **5. Creation of Streamlit Application**

The code that is supplied creates a Streamlit application that can retrieve bus transit data from a MySQL database, let users filter the data according to different standards, and then show the data that has been filtered.

**Creating the Streamlit App:**

1. Importing Required Libraries:

import streamlit as st

from sqlalchemy import create\_engine

import pandas as pd

streamlit: The main library used to create the interactive web application.

sqlalchemy: A library used to interact with the MySQL database.

pandas: A library used for data manipulation and analysis.

1. Database Connection Using SQLAlchemy:

db\_connection\_str = 'mysql+pymysql://root:Mona%40999@localhost/redbusproject'

engine=create\_engine(db\_connection\_str)

This code sets up a connection to the MySQL database using SQLAlchemy. Replace the credentials with your actual database username, password, host, port, and database name.

1. Fetching Data from the Database:

query = "SELECT \* FROM bus\_details"

data = pd.read\_sql(query, engine)

The pd.read\_sql function is used to execute the SQL query and fetch the data from the bus\_routes table into a pandas DataFrame named data.

1. Streamlit App Layout:

dfbus['Departing\_time'] = pd.to\_datetime(dfbus['Departing\_time'], format='%H:%M').dt.time

# Streamlit title and instructions

# Bus image URL

bus\_image\_url = "https://cdn-icons-png.flaticon.com/512/5030/5030991.png"

# Display the bus image along with the title using Markdown

st.markdown(f'<h1><img src="{bus\_image\_url}" width="50" style="vertical-align:middle;"> Redbus Data Scraping with Selenium & Dynamic Filtering using Streamlit</h1>', unsafe\_allow\_html=True)

st.write('This app allows you to filter and view bus route details. You can also download the filtered data.')

This line sets the title of the Streamlit application.

1. Filters:

state\_filter =st.multiselect('Select State:',options=dfbus['state'].unique() )

route\_filter = st.multiselect('Select Route:', options=dfbus['Route\_name'].unique())

bus\_type\_filter = st.sidebar.multiselect('Select Bus Type:', options=dfbus['Bus\_type'].unique())

price\_filter = st.sidebar.slider('Select Price Range:', min\_value=int(dfbus['Price'].min()),max\_value=int(dfbus['Price'].max()), value=(int(dfbus['Price'].min()), int(dfbus['Price'].max()))) time\_ranges = {

'6 AM to 10 AM': (6, 10),

'10 AM to 12 PM': (10, 12),

'12 PM to 3 PM': (12, 15),

'3 PM to 6 PM': (15, 18),

'6 PM to 9 PM': (18, 21),

'9 PM to 12 AM': (21, 24)

}

selected\_time\_range = st.selectbox('Select Departing Time Range:', list(time\_ranges.keys()))

# Apply filters

filtered\_data = dfbus

if state\_filter:

filtered\_data = filtered\_data[filtered\_data['state'].isin(state\_filter)]

if route\_filter:

filtered\_data = filtered\_data[filtered\_data['Route\_name'].isin(route\_filter)]

if bus\_type\_filter:

filtered\_data = filtered\_data[filtered\_data['Bus\_type'].isin(bus\_type\_filter)]

filtered\_data = filtered\_data[(filtered\_data['Price'] >= price\_filter[0]) & (filtered\_data['Price'] <= price\_filter[1])]

filtered\_data = filtered\_data[(filtered\_data['Star\_Rating'] >= rating\_filter[0]) & (filtered\_data['Star\_Rating'] <= rating\_filter[1])]

# Filter by selected departing time range

start\_hour, end\_hour = time\_ranges[selected\_time\_range]

filtered\_data = filtered\_data[filtered\_data['Departing\_time'].apply(lambda x: start\_hour <= x.hour < end\_hour)]

# Display filtered data

st.write('Filtered Bus Data:')

st.dataframe(filtered\_data)

# Download button for filtered data as CSV

if not filtered\_data.empty:

st.download\_button(

label="Download Filtered Data",

data=filtered\_data.to\_csv(index=False),

file\_name="filtered\_bus\_details.csv",

mime="text/csv"

)

else:

st.warning("No data available with the selected filters.")

Multiselect Filters:

* state\_filter: Allows users to select multiple selects from a dropdown.
* bustype\_filter: Allows users to select multiple bus types from a dropdown.
* route\_filter: Allows users to select multiple routes from a dropdown.

Slider Filters:

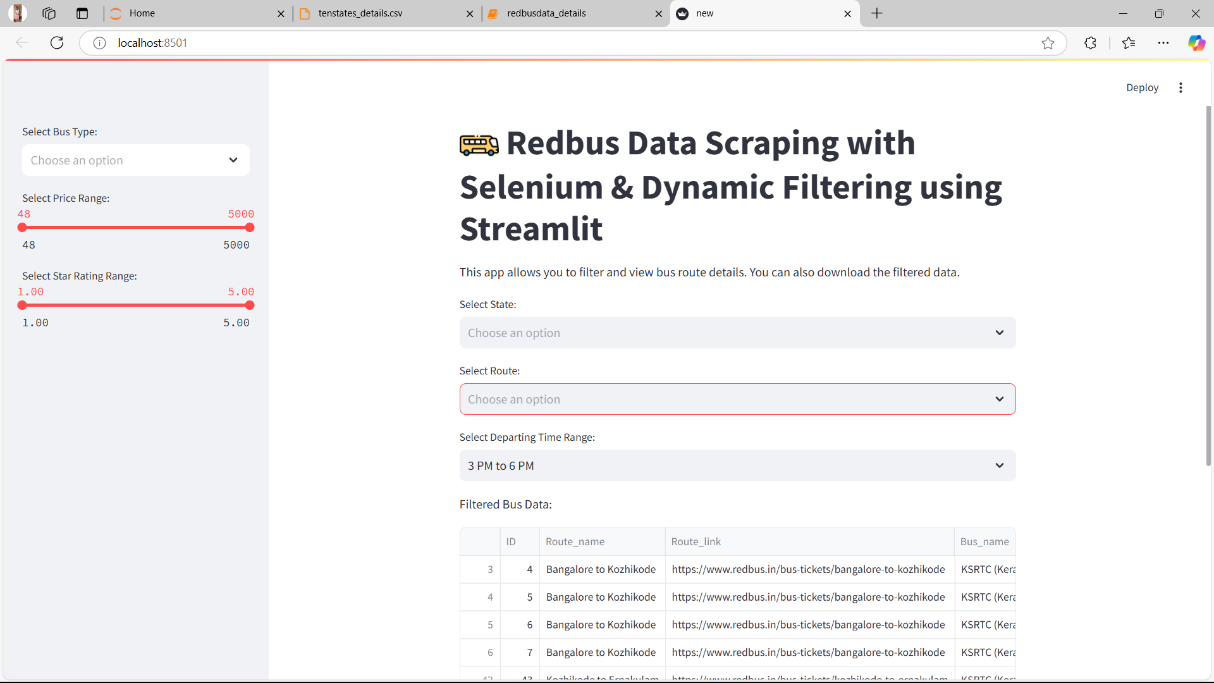
price\_filter: Allows users to select a price range using a slider.

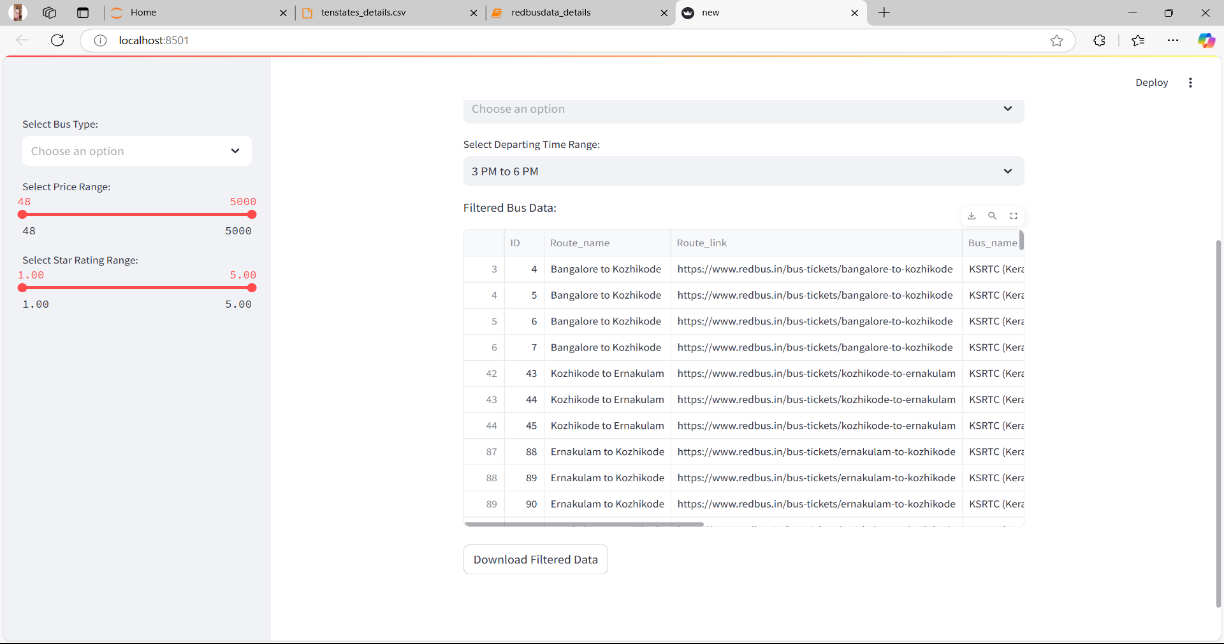
* star\_filter: Allows users to select a star rating range using a slider.

**Running the Streamlit App:**

streamlit run your\_script\_name.py

**Screenshots :**

****

****

**Local URL:** [**http://localhost:8502**](http://localhost:8502)

**Network URL:** [**http://192.168.1.4:8502**](http://192.168.1.4:8502)

### 

### **6. Results**

**Expected Outcomes:**

### Use Selenium to successfully scrape at least 10 Government State Bus Transport data points from the Redbus website. Include information on private buses on the chosen routes as well.

### Put the information in an organized SQL database.

### Create an interactive data filtering application using Streamlit.

### Make sure the application is effective and easy to use.

### **7. Project Evaluation Metrics**

**Data Scraping Accuracy:**

* Completeness and correctness of the scraped data.

**Database Design:**

* Effective and efficient database schema.

**Application Usability:**

* User experience and ease of use of the Streamlit application.

**Filter Functionality:**

* Effectiveness and responsiveness of data filters.

**Code Quality:**

* Adherence to coding standards and best practices.

**8. Technical Tags:**

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

### **9. Conclusion**

**Summary:** Summarize the project, the process of scraping data, storing it, and displaying it using Streamlit.

**Future Work:**

* Improvements in data scraping.
* Adding more features to the Streamlit app.

### **10. References**

* Links to resources and documentation used in the project.

<https://www.redbus.in/>

[Selenium Documentation](https://www.selenium.dev/documentation/)

[Streamlit Documentation](https://docs.streamlit.io/)

[PyMySQL Documentation](https://pymysql.readthedocs.io/en/latest/)