## **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.
* **Visual studio:** For development and testing.

### **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

### **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()

driver.maximize\_window()

1. Navigating to Redbus Website:

The WebDriver navigates to the Redbus homepage and waits for the page to load.

# Open Redbus website

driver.get("https://www.redbus.in")

time.sleep(5) # Wait for the page to load

1. Clicks the 'View All' Button:

The script locates and clicks the 'View All' button in the Government Bus Corporations section to access the list of bus corporations.

# Click 'View All' button in the Government Bus Corporations section

view\_all\_button = driver.find\_element(By.XPATH, '//\*[@id="homeV2-root"]/div[3]/div[1]/div[2]/a')

view\_all\_button.click()

time.sleep(5)

1. Switching to New Tab:

The script switches to the new tab that opens after clicking the 'View All' button.

# Switch to the new tab

driver.switch\_to.window(driver.window\_handles[1])

1. Scrolling to the Bottom of the Page:

The script scrolls to the bottom of the page to ensure all bus corporations are loaded.

* # Scroll down to the bottom of the page
* last\_height = driver.execute\_script("return document.body.scrollHeight")
* while True:
* driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")
* time.sleep(5) # Wait to load the page
* new\_height = driver.execute\_script("return document.body.scrollHeight")
* if new\_height == last\_height:
* break
* last\_height = new\_height

1. Selecting a Bus Corporation:

The script selects a specific bus corporation and clicks on its link.

# Select a bus corporation

try:

bus\_corp = WebDriverWait(driver, 10).until(

EC.element\_to\_be\_clickable((By.XPATH, '//\*[@id="root"]/div/article[2]/div/div/ul[3]/li[3]/a'))

)

# Scroll the element into view

driver.execute\_script("arguments[0].scrollIntoView(true);", bus\_corp)

time.sleep(2)

# Click the bus corporation link

bus\_corp.click()

time.sleep(5)

except ElementClickInterceptedException:

print("Element is not clickable, trying again...")

bus\_corp.click()

time.sleep(5)

1. Scraping Route Names and Links:

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

# Initialize list to store routes

routes = []

# Loop to manually click each page and scrape data

while True:

input("Please click the next page number button manually, wait for the page to load, then press Enter to confirm...")

# Scrape route names and their links

route\_elements = driver.find\_elements(By.CLASS\_NAME, 'route')

for route\_element in route\_elements:

route = route\_element.text # Get the route name

route\_link = route\_element.get\_attribute('href') # Get the route link

routes.append((route, route\_link))

# Check if there are more pages to click

more\_pages = input("Are there more pages to click? (yes/no): ")

if more\_pages.lower() != 'yes':

break

# Print the scraped routes to view them

print("Scraped Routes and Links:")

for route, link in routes:

print(f"Route: {route}, Link: {link}")

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.

* # Now use the scraped routes and links to get bus details
* bus\_details = []
* for route, route\_link in routes:
* driver.get(route\_link)
* time.sleep(5) # Wait for the page to load
* # Click the 'View Buses' button if present
* try:
* view\_buses\_button = driver.find\_element(By.CLASS\_NAME, 'button')
* view\_buses\_button.click()
* time.sleep(5)
* except NoSuchElementException:
* pass # No 'View Buses' button present
* # Scroll down to the bottom of the page
* last\_height = driver.execute\_script("return document.body.scrollHeight")
* while True:
* driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")
* time.sleep(2) # Wait to load the page
* new\_height = driver.execute\_script("return document.body.scrollHeight")
* if new\_height == last\_height:
* break
* last\_height = new\_height
* try:
* bus\_elements = driver.find\_elements(By.CSS\_SELECTOR, "div.bus-item")
* except NoSuchElementException:
* print("No bus elements found")
* continue
* for bus in bus\_elements:
* try:
* busname = bus.find\_element(By.CSS\_SELECTOR, "div.travels.lh-24.f-bold.d-color").text
* except NoSuchElementException:
* busname = "N/A"
* try:
* bustype = bus.find\_element(By.CSS\_SELECTOR, "div.bus-type.f-12.m-top-16.l-color.evBus").text
* except NoSuchElementException:
* bustype = "N/A"
* try:
* departing\_time = bus.find\_element(By.CSS\_SELECTOR, "div.dp-time.f-19.d-color.f-bold").text
* departing\_time\_dt = convert\_to\_datetime(departing\_time, datetime.now())
* except NoSuchElementException:
* departing\_time\_dt = None
* try:
* duration = bus.find\_element(By.CSS\_SELECTOR, "div.dur.l-color.lh-24").text
* except NoSuchElementException:
* duration = "N/A"
* try:
* reaching\_time = bus.find\_element(By.CSS\_SELECTOR, "div.bp-time.f-19.d-color.disp-Inline").text
* reaching\_time\_dt = convert\_to\_datetime(reaching\_time, datetime.now())
* if reaching\_time\_dt and departing\_time\_dt and reaching\_time\_dt < departing\_time\_dt:
* reaching\_time\_dt += timedelta(days=1)
* except NoSuchElementException:
* reaching\_time\_dt = None
* try:
* star\_rating = bus.find\_element(By.CSS\_SELECTOR, "div.rating-sec.lh-24").text
* star\_rating = float(star\_rating) if star\_rating != "N/A" else 0.0
* except NoSuchElementException:
* star\_rating = 0.0
* try:
* price = bus.find\_element(By.CSS\_SELECTOR, "span.f-19.f-bold").text
* price = float(price.replace('₹', '').replace(',', '').strip()) if price != "N/A" else None
* except NoSuchElementException:
* price = None
* try:
* seats\_available = bus.find\_element(By.CSS\_SELECTOR, "div.seat-left.m-top-16").text
* seats\_available = int(seats\_available.split()[0]) if seats\_available != "N/A" else 0
* except NoSuchElementException:
* seats\_available = 0
* bus\_details.append((route, route\_link, busname, bustype, departing\_time\_dt, duration, reaching\_time\_dt, star\_rating, price, seats\_available))
* # Print the scraped bus details to view them
* print("Scraped Bus Details:")
* for detail in bus\_details:
* print(detail)

### **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
* conn = pymysql.connect(
* host='127.0.0.1',
* user='root',
* passwd='sripathi12345678',
* db='redbus\_data' # Make sure the database exists
* )
* cursor = conn.cursor()

1. Creating the Database Schema:

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

* # Create the bus\_routes table if it doesn't exist
* cursor.execute('''
* CREATE TABLE IF NOT EXISTS bus\_routes (
* id INT AUTO\_INCREMENT PRIMARY KEY,
* route\_name TEXT,
* route\_link TEXT,
* busname TEXT,
* bustype TEXT,
* departing\_time DATETIME,
* duration TEXT,
* reaching\_time DATETIME,
* star\_rating FLOAT,
* price DECIMAL(10, 2),
* seats\_available INT
* )
* ''')
* 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\_routes table.

* # Insert scraped data into the bus\_routes table
* for detail in bus\_details:
* cursor.execute('''
* INSERT INTO bus\_routes (
* route\_name, route\_link, busname, bustype, departing\_time,
* duration, reaching\_time, star\_rating, price, seats\_available
* ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
* ''', detail)
* 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\_routes table.
* e) Committing the Transaction and Closing the Connection:

After inserting all the data, the transaction is committed to the database to ensure the data is saved.

* # Commit the transaction and close the connection
* conn.commit()

The database connection is then closed to free up resources.

* conn.close()
* f) Printing Confirmation Message:

The script prints a confirmation message indicating that the data has been successfully saved to the database.

* print("Data has been successfully saved to the database.")

### **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. A download button to export the filtered data as a CSV file is also provided by the program. A thorough description of the code's operation may be found below.

**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:

engine = create\_engine('mysql+pymysql://root:sripathi12345678@127.0.0.1:3306/redbus\_data')

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\_routes"

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:

st.title('Redbus Routes Data Filtering and Analysis')

This line sets the title of the Streamlit application.

1. Filters:

bustype\_filter = st.multiselect('Select Bus Type:', options=data['bustype'].unique())

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

price\_filter = st.slider('Select Price Range:', min\_value=int(data['price'].min()), max\_value=int(data['price'].max()), value=(int(data['price'].min()), int(data['price'].max())))

star\_filter = st.slider('Select Star Rating Range:', min\_value=float(data['star\_rating'].min()), max\_value=float(data['star\_rating'].max()), value=(float(data['star\_rating'].min()), float(data['star\_rating'].max())))

availability\_filter = st.slider('Select Seat Availability Range:', min\_value=int(data['seats\_available'].min()), max\_value=int(data['seats\_available'].max()), value=(int(data['seats\_available'].min()), int(data['seats\_available'].max())))

Multiselect Filters:

* 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.
* availability\_filter: Allows users to select a seat availability range using a slider.

1. Filtering Data Based on User Inputs:

filtered\_data = data

if bustype\_filter:

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

if route\_filter:

filtered\_data = filtered\_data[filtered\_data['route\_name'].isin(route\_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'] >= star\_filter[0]) & (filtered\_data['star\_rating'] <= star\_filter[1])]

filtered\_data = filtered\_data[(filtered\_data['seats\_available'] >= availability\_filter[0]) & (filtered\_data['seats\_available'] <= availability\_filter[1])]

The original DataFrame data is filtered by this code according to the user's choices.Rows are filtered using the isin technique according to the bus kinds and routes that have been chosen.Rows are filtered according to the chosen price range, star rating range, and seat availability range using the between condition.

1. Displaying Filtered Data:

st.write('Filtered Data:')

st.dataframe(filtered\_data)

This code displays the filtered data in the Streamlit application.

1. Download Button:

if not filtered\_data.empty:

st.download\_button(

label="Download Filtered Data",

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

file\_name="filtered\_data.csv",

mime="text/csv"

)

else:

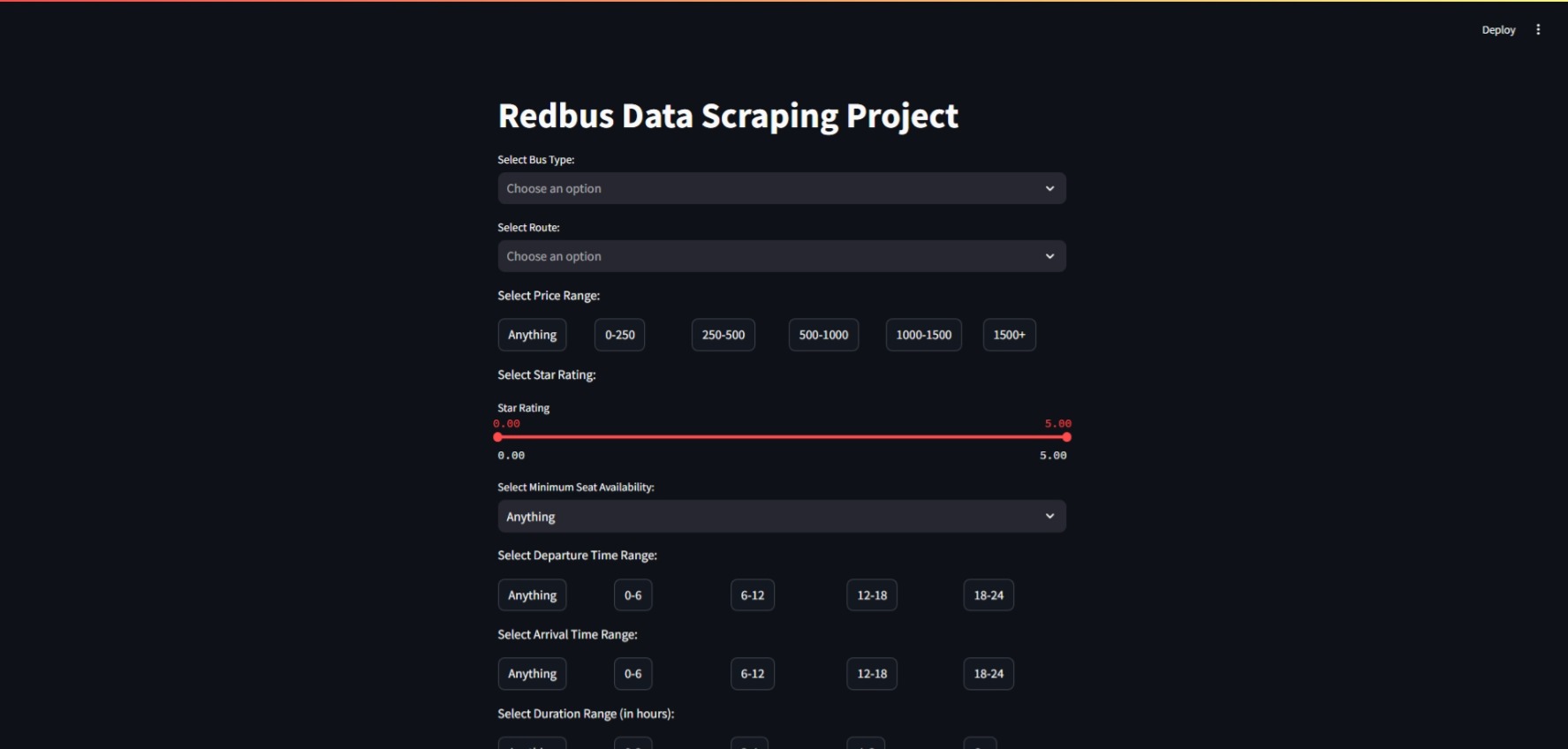
st.warning("No data available with the selected filters.")

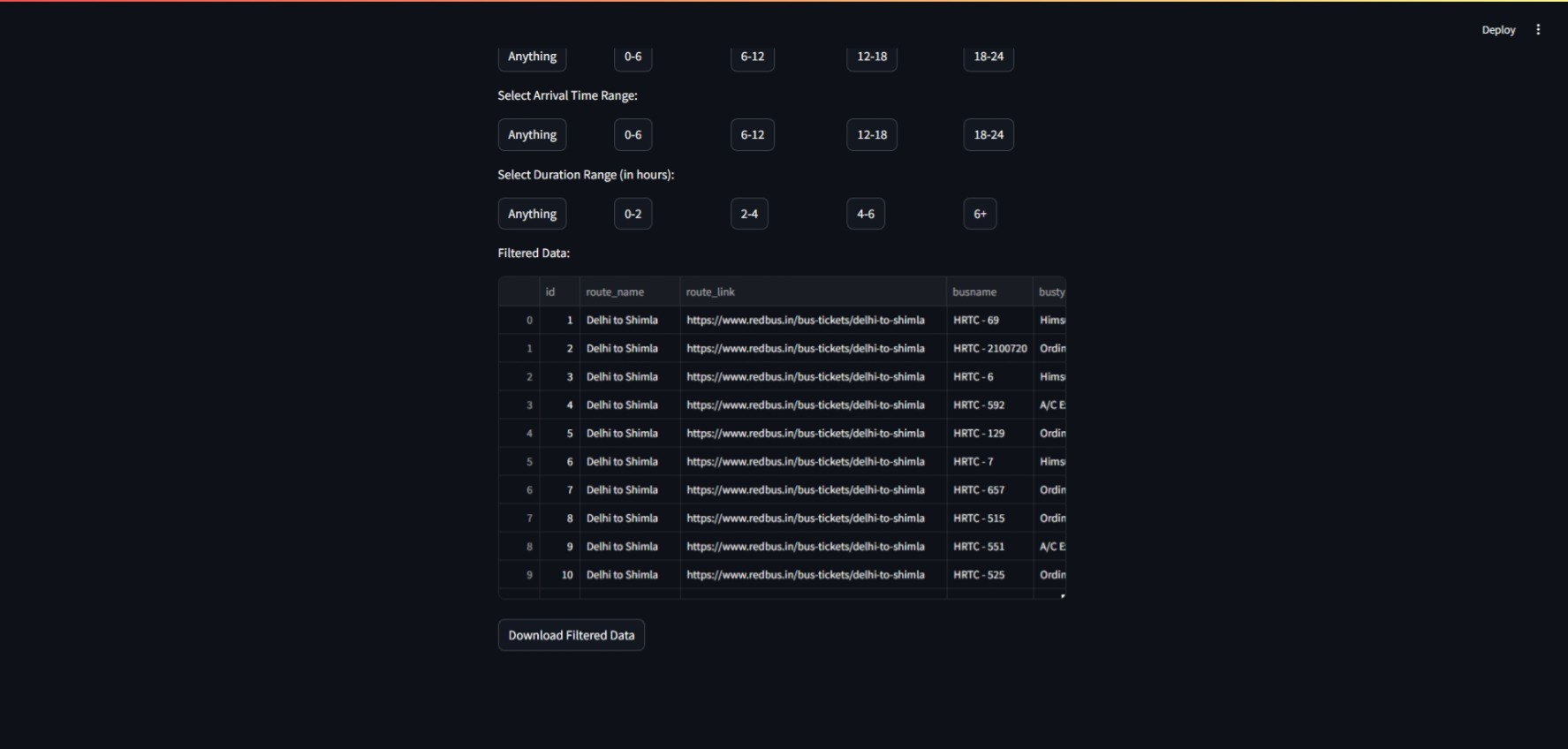
This code adds a download button to the Streamlit application, allowing users to download the filtered data as a CSV file. If the filtered data is empty (i.e., no data matches the selected filters), a warning message is displayed instead.

**Running the Streamlit App:**

streamlit run your\_script\_name.py

**Screenshots :**

****

****

**Link :**

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

**Network URL:** [**http://192.168.3.192:8502**](http://192.168.3.192:8502)

### **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/)