**REDBUS PROJECT**

# 1. Project Overview

**Title**:Redbus Data Scraping with Selenium & Dynamic Filtering using Streamlit

**Objective**:

The project aims to automate the scraping of bus travel data from Redbus and provide real-time insights through a user-friendly Streamlit interface.

# 2. Technical Skills and Technologies Used

● Web Scraping

● Selenium

● Streamlit

● SQL

● Data Analysis

● Python

# 3. Business Use Cases

● Travel Aggregators: Providing real-time bus schedules and seat availability for customers.

● Market Analysis: Analyzing travel patterns and preferences for market research.

● Customer Service: Enhancing user experience by offering customized travel options based on data insights.

● Competitor Analysis: Comparing pricing and service levels with competitors.

# 4. Project Approach

1. Data Scraping:

○ Use Selenium to automate the extraction of Redbus data including routes, schedules, prices, and seat availability.

2. Data Storage:

○ the scraped data into a SQL database.

3. Streamlit Application:

○ a Streamlit application to display and filter the scraped data.

○ Implement various filters such as bustype, route, price range, star rating, availability

4. Data Analysis/Filtering using Streamlit:

○ 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

# 5. Technical Details

Code snippets

Python Code for scraping data from redbus.here the details.

I can use the same code to scrape 10 bus details.i can only change the website link

in webiee and csv files names

#code .

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

from selenium.common.exceptions import NoSuchElementException, TimeoutException

import time

import pandas as pd

# URL of the website

WEBIEE = "https://www.redbus.in/online-booking/kaac-transport"

# Initialize the driver

def initialize\_driver():

driver = webdriver.Chrome()

driver.maximize\_window()

return driver

# Load the webpage

def load\_page(driver, url):

driver.get(url)

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

# Scrape bus routes

def scrape\_bus\_routes(driver):

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

ALL\_BUS\_ROUTES\_LINK = [route.get\_attribute('href') for route in route\_elements]

ALL\_BUS\_ROUTES\_NAME = [route.text.strip() for route in route\_elements]

return ALL\_BUS\_ROUTES\_LINK, ALL\_BUS\_ROUTES\_NAME

# Scrape bus details for each route

def scrape\_bus\_details(driver, url, route\_name):

bus\_details = []

try:

driver.get(url)

time.sleep(5) # Allow the page to load

# Click the "View Buses" button if available

try:

view\_buses\_button = WebDriverWait(driver, 10).until(

EC.element\_to\_be\_clickable((By.CLASS\_NAME, "button"))

)

driver.execute\_script("arguments[0].click();", view\_buses\_button)

time.sleep(5) # Wait for buses to load

# Scroll down to load all bus items

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

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

# Extract bus details

bus\_name\_elements = driver.find\_elements(By.CLASS\_NAME, "travels.lh-24.f-bold.d-color")

bus\_type\_elements = driver.find\_elements(By.CLASS\_NAME, "bus-type.f-12.m-top-16.l-color.evBus")

departing\_time\_elements = driver.find\_elements(By.CLASS\_NAME, "dp-time.f-19.d-color.f-bold")

duration\_elements = driver.find\_elements(By.CLASS\_NAME, "dur.l-color.lh-24")

reaching\_time\_elements = driver.find\_elements(By.CLASS\_NAME, "bp-time.f-19.d-color.disp-Inline")

star\_rating\_elements = driver.find\_elements(By.CLASS\_NAME, "rating-sec.lh-24")

price\_elements = driver.find\_elements(By.CLASS\_NAME, "fare.d-block")

seat\_availability\_elements = driver.find\_elements(By.XPATH, "//div[contains(@class, 'seat-left m-top-30') or contains(@class, 'seat-left m-top-16')]")

for i in range(len(bus\_name\_elements)):

bus\_detail = {

"ROUTE\_NAME": route\_name,

"ROUTE\_ID": url,

"BUS\_NAME": bus\_name\_elements[i].text,

"BUS\_TYPE": bus\_type\_elements[i].text if i < len(bus\_type\_elements) else 'N/A',

"DEPARTING\_TIME": departing\_time\_elements[i].text if i < len(departing\_time\_elements) else 'N/A',

"DURATION": duration\_elements[i].text if i < len(duration\_elements) else 'N/A',

"REACHING\_TIME": reaching\_time\_elements[i].text if i < len(reaching\_time\_elements) else 'N/A',

"RATING": star\_rating\_elements[i].text if i < len(star\_rating\_elements) else '0',

"PRICE": price\_elements[i].text if i < len(price\_elements) else 'N/A',

"SEAT\_AVAILABILITY": seat\_availability\_elements[i].text if i < len(seat\_availability\_elements) else '0'

}

bus\_details.append(bus\_detail)

except Exception as e:

print(f"Error occurred while scraping bus details for {url}: {str(e)}")

return []

except (NoSuchElementException, TimeoutException) as e:

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

return []

return bus\_details

# Scrape all pages and bus routes

def scrape\_all\_pages():

ALL\_BUS\_DETAILS = []

for page in range(1, 6): # Iterate over the 5 pages

try:

driver = initialize\_driver()

load\_page(driver, WEBIEE)

if page > 1:

pagination\_tab = WebDriverWait(driver, 10).until(

EC.presence\_of\_element\_located((By.XPATH, f"//div[contains(@class, 'DC\_117\_pageTabs')][text()='{page}']"))

)

driver.execute\_script("arguments[0].scrollIntoView();", pagination\_tab)

driver.execute\_script("arguments[0].click();", pagination\_tab)

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

ALL\_BUS\_ROUTES\_LINK, ALL\_BUS\_ROUTES\_NAME = scrape\_bus\_routes(driver)

# Iterate over each route link and scrape bus details

for link, name in zip(ALL\_BUS\_ROUTES\_LINK, ALL\_BUS\_ROUTES\_NAME):

bus\_details = scrape\_bus\_details(driver, link, name)

if bus\_details:

ALL\_BUS\_DETAILS.extend(bus\_details)

except Exception as e:

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

return ALL\_BUS\_DETAILS

# Scrape routes and details from all pages

ALL\_BUS\_DETAILS = scrape\_all\_pages()

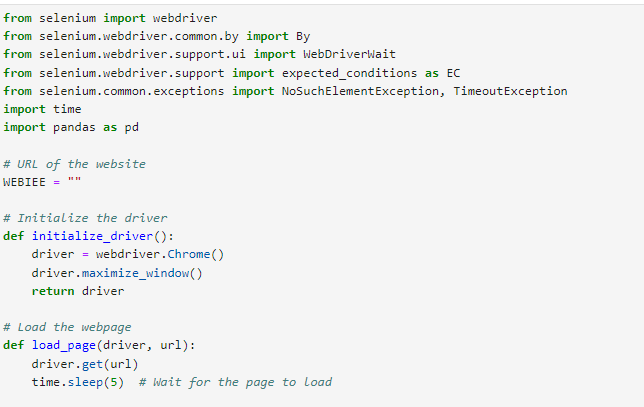
# Convert the list of dictionaries to a DataFrame

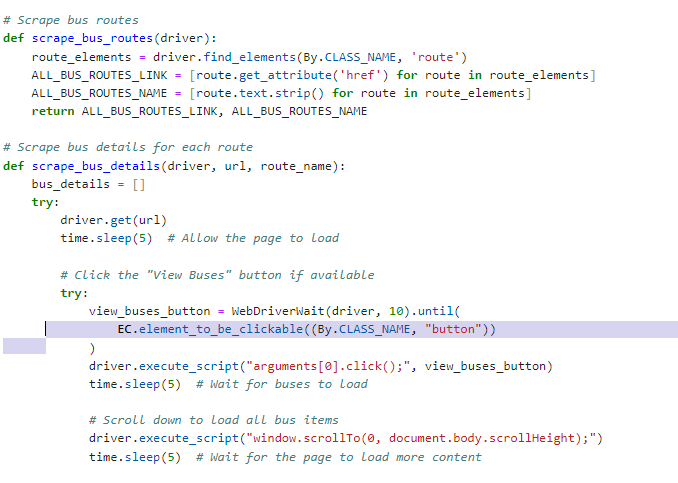
df = pd.DataFrame(ALL\_BUS\_DETAILS)

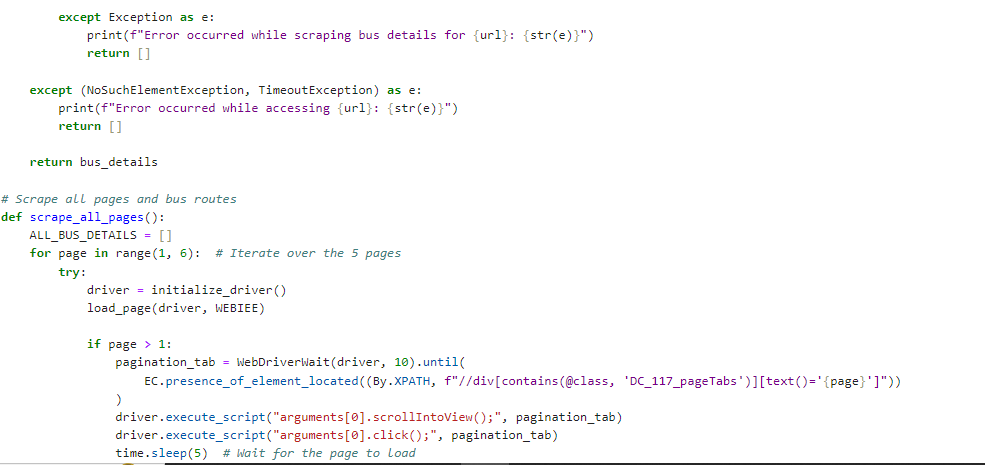
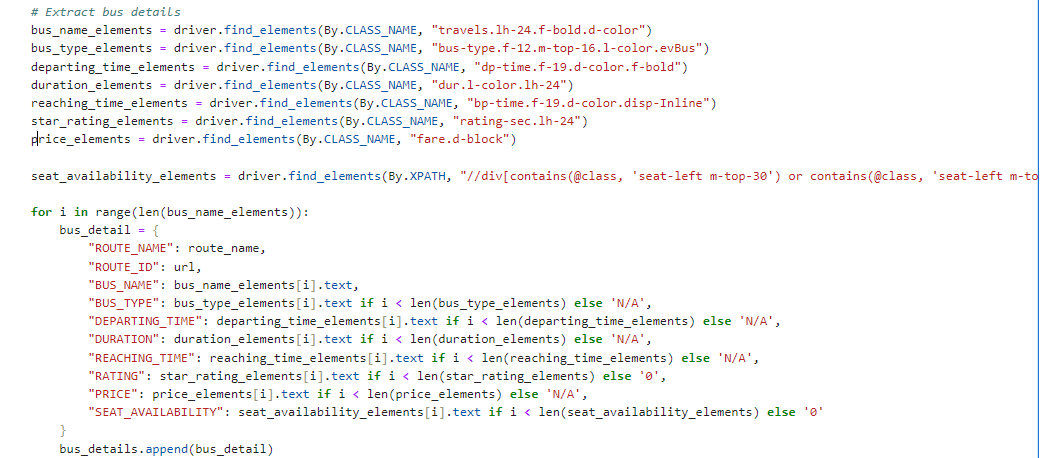
# Save the DataFrame to a CSV file

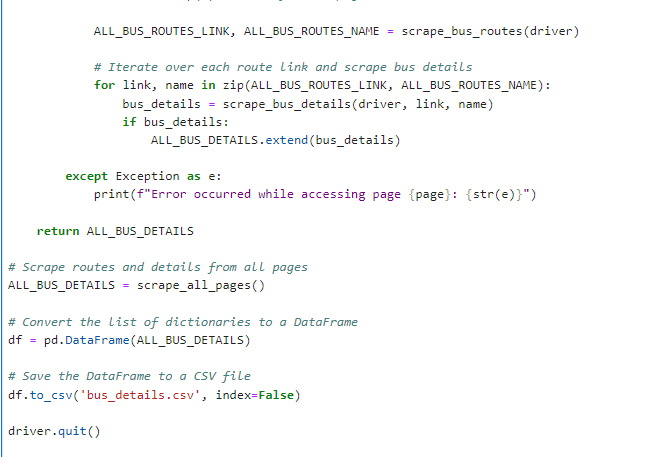
df.to\_csv('assam\_kaac\_bus\_details.csv', index=False)

driver.quit()





****

****

All route website links:

* kerala:<https://www.redbus.in/online-booking/ksrtc-kerala/?utm_source=rtchometile>
* telegana:<https://www.redbus.in/online-booking/tsrtc/?utm_source=rtchometile>
* kadamba:<https://www.redbus.in/online-booking/ktcl/?utm_source=rtchometile>
* northbengal(westbengal2):<https://www.redbus.in/online-booking/south-bengal-state-transport-corporation-sbstc/?utm_source=rtchometile>
* Himachal pradesh:<https://www.redbus.in/online-booking/hrtc/?utm_source=rtchometile>
* assam:<https://www.redbus.in/online-booking/astc/?utm_source=rtchometile>
* westbengal:<https://www.redbus.in/online-booking/wbtc-ctc/?utm_source=rtchometile>
* chandigarh:<https://www.redbus.in/online-booking/chandigarh-transport-undertaking-ctu>
* punjab:<https://www.redbus.in/online-booking/pepsu/?utm_source=rtchometile>
* Jammu and kashmir:<https://www.redbus.in/online-booking/jksrtc>

Python mysql connection:

import pandas as pd

import pymysql

# List of CSV file paths

csv\_files = ["wbengal2\_bus\_details.csv", "chandigarh\_bus\_details.csv", "hp\_bus\_details.csv",

"kadamba\_bus\_details.csv", "KERALA\_bus\_details.csv", "punjab\_bus\_details.csv",

"tl\_bus\_details.csv", "wbengal\_bus\_details.csv", "northbengal\_bus\_details.csv",

"jk\_bus\_details.csv"]

# Read each CSV file into a DataFrame and store it in a list

df\_list = [pd.read\_csv(file) for file in csv\_files]

# Concatenate all DataFrames in the list

combined\_df = pd.concat(df\_list, ignore\_index=True)

# Add id column

combined\_df['id'] = range(1, len(combined\_df) + 1)

# Replace 'INR ' in 'Price' column (if exists) and handle NaN values

if 'Price' in combined\_df.columns:

combined\_df['Price'] = combined\_df['Price'].fillna('').astype(str).str.replace('INR ', '', regex=False)

# Extract digits from 'Seat\_Availability' column (if exists) and handle NaN values

if 'Seat\_Availability' in combined\_df.columns:

combined\_df['Seat\_Availability'] = combined\_df['Seat\_Availability'].fillna('').str.extract(r'(\d+)')

# Save the cleaned DataFrame to CSV

combined\_df.to\_csv("bus\_routes.csv", index=False)

# Connect to MySQL database

myconnection = pymysql.connect(host='127.0.0.1', user='root', passwd='34566@25', database="redbus")

# Drop rows with missing values

combined\_df = combined\_df.dropna()

# Convert DataFrame dtypes to MySQL compatible datatypes

dtype\_map = {

'float64': 'FLOAT',

'object': 'TEXT',

'int64': 'INT'

}

# Create MySQL table schema from DataFrame dtypes

columns\_and\_types = ", ".join(f"{col} {dtype\_map[str(dtype)]}" for col, dtype in combined\_df.dtypes.items())

table\_name = "bus\_routes"

# Create table if not exists

with myconnection.cursor() as cursor:

cursor.execute(f"CREATE TABLE IF NOT EXISTS {table\_name} ({columns\_and\_types})")

# Insert data into MySQL table

with myconnection.cursor() as cursor:

for i in range(len(combined\_df)):

# Use parameterized queries for security and performance

sql = f"INSERT INTO {table\_name} VALUES ({','.join(['%s'] \* len(combined\_df.columns))})"

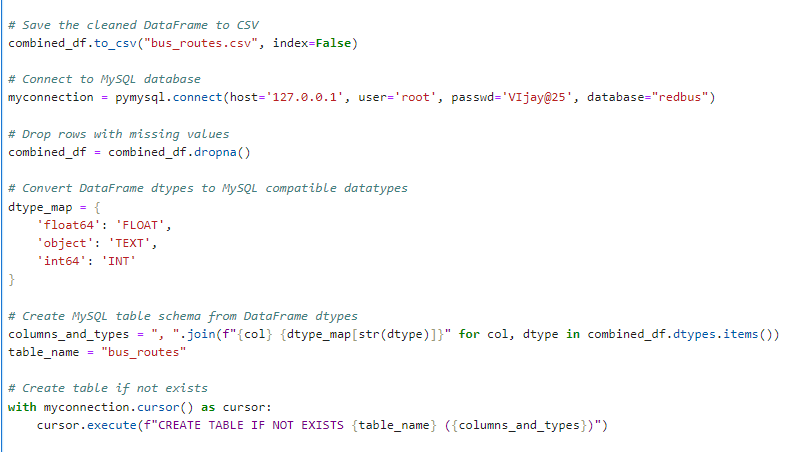
cursor.execute(sql, tuple(combined\_df.iloc[i]))

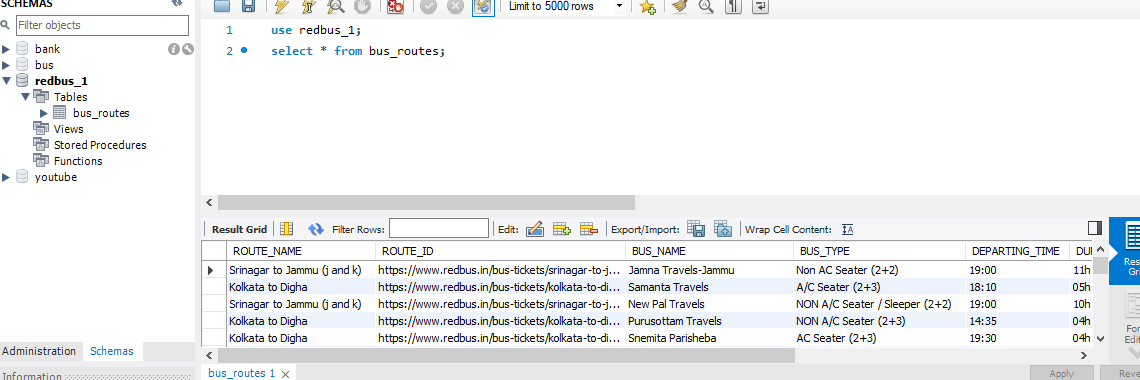
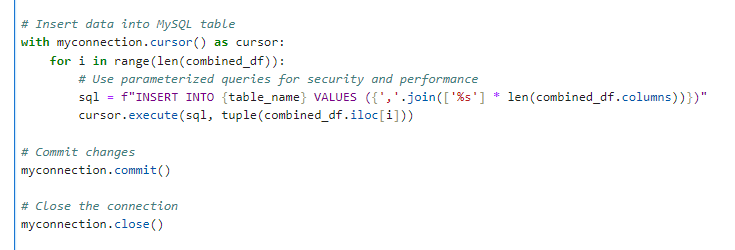
# Commit changes

myconnection.commit()

# Close the connection

myconnection.close()



****

Streamlit code:

import streamlit as st

import pymysql

import pandas as pd

# Connect to MySQL database

def get\_connection():

return pymysql.connect(host='127.0.0.1', user='root', passwd='VIjay@25', database='redbus\_1')

# Function to fetch route names starting with a specific letter, arranged alphabetically

def fetch\_route\_names(connection, starting\_letter):

query = f"SELECT DISTINCT ROUTE\_NAME FROM bus\_routes WHERE ROUTE\_NAME LIKE '{starting\_letter}%' ORDER BY ROUTE\_NAME"

route\_names = pd.read\_sql(query, connection)['ROUTE\_NAME'].tolist()

return route\_names

# Function to fetch data from MySQL based on selected ROUTE\_NAME and price sort order

def fetch\_data(connection, route\_name, price\_sort\_order):

price\_sort\_order\_sql = "ASC" if price\_sort\_order == "Low to High" else "DESC"

query = f"SELECT \* FROM bus\_routes WHERE ROUTE\_NAME = %s ORDER BY RATING DESC, PRICE {price\_sort\_order\_sql}"

df = pd.read\_sql(query, connection, params=(route\_name))

return df

# Function to filter data based on RATING and BUS\_TYPE

def filter\_data(df, ratings, bus\_types):

filtered\_df = df[df['RATING'].isin(ratings) & df['BUS\_TYPE'].isin(bus\_types)]

return filtered\_df

# Main Streamlit app

def main():

st.header('Easy and Secure Online Bus Tickets Booking')

connection = get\_connection()

try:

# Sidebar - Input for starting letter

starting\_letter = st.sidebar.text\_input('Enter Starting Letter of Route Name', 'A')

# Fetch route names starting with the specified letter

if starting\_letter:

route\_names = fetch\_route\_names(connection, starting\_letter.upper())

if route\_names:

# Sidebar - Selectbox for ROUTE\_NAME

selected\_route = st.sidebar.radio('Select Route Name', route\_names)

if selected\_route:

# Sidebar - Selectbox for sorting preference

price\_sort\_order = st.sidebar.selectbox('Sort by Price', ['Low to High', 'High to Low'])

# Fetch data based on selected ROUTE\_NAME and price sort order

data = fetch\_data(connection, selected\_route, price\_sort\_order)

if not data.empty:

# Display data table with a subheader

st.write(f"### Data for Route: {selected\_route}")

st.write(data)

# Filter by RATING and BUS\_TYPE

ratings = data['RATING'].unique().tolist()

selected\_ratings = st.multiselect('Filter by Rating', ratings)

bus\_types = data['BUS\_TYPE'].unique().tolist()

selected\_bus\_types = st.multiselect('Filter by Bus Type', bus\_types)

if selected\_ratings and selected\_bus\_types:

filtered\_data = filter\_data(data, selected\_ratings, selected\_bus\_types)

# Display filtered data table with a subheader

st.write(f"### Filtered Data for Rating: {selected\_ratings} and Bus Type: {selected\_bus\_types}")

st.write(filtered\_data)

else:

st.write(f"No data found for Route: {selected\_route} with the specified price sort order.")

else:

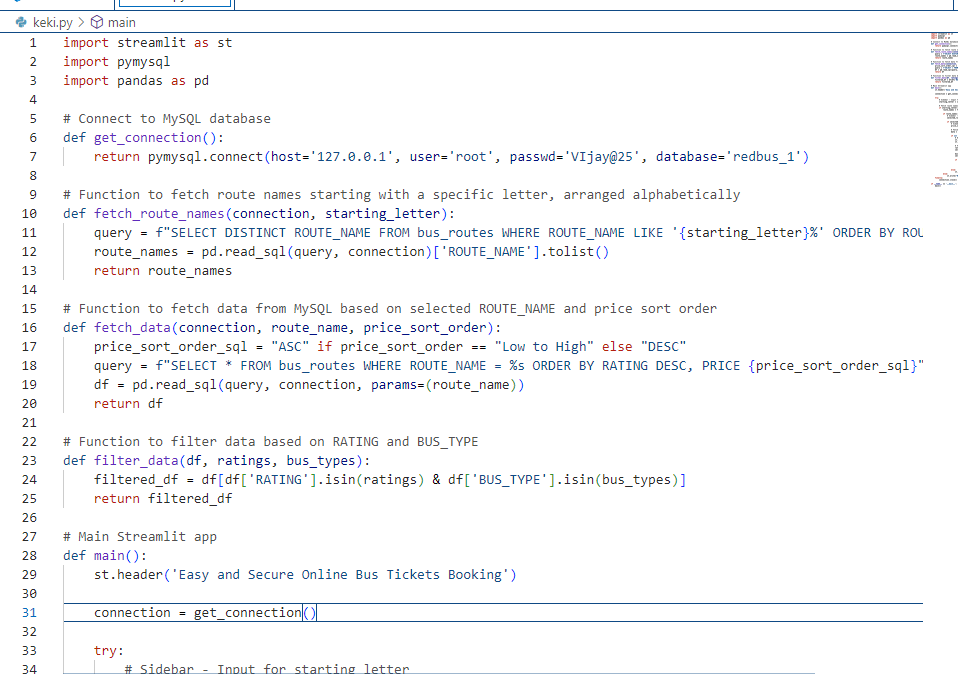
st.write("No routes found starting with the specified letter.")

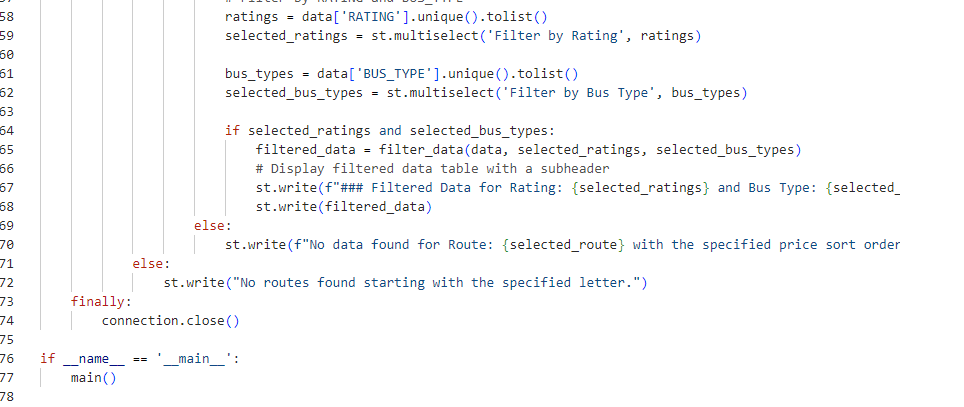
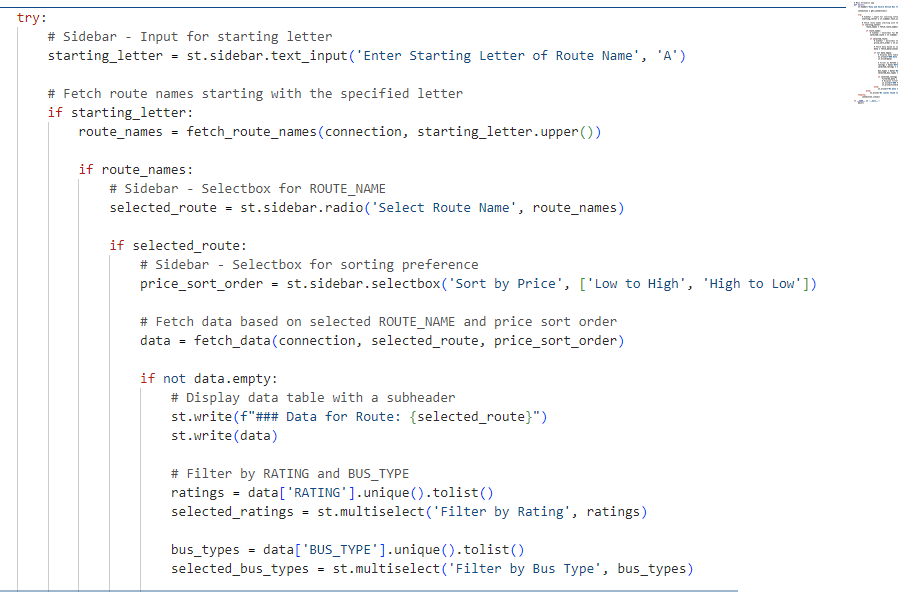
finally:

connection.close()

if \_\_name\_\_ == '\_\_main\_\_':

main()



****

# 6.results and analysis;

* Number of bus routes scrapped:3785
* Filtering functionalities provided : Enter Starting Letter of Route Name , Sort by Price,Data for Route: Agra to Chandigarh,Filter by Rating,Filter by Bus Type.

Streamlit significantly enhances the user experience by:

* Intuitive Interface: The application's clean and user-friendly design makes it easy for users to navigate and find the information they need.
* Real-time Data: Users can access up-to-date bus schedules and prices, ensuring they have the most accurate information for their travel plans.
* Customization: The ability to filter results based on various criteria, such as bus type, price, and departure time, empowers users to personalize their search and find the best options for their needs.
* Visualizations: Streamlit can provide visualizations, such as maps and charts, to help users visualize bus routes and understand pricing trends.
* Seamless Integration: By integrating with Redbus data, Streamlit can offer a comprehensive solution for bus travel planning and booking.

Here are some additional paragraphs to elaborate on how Streamlit enhances user experience:

* Efficiency: Streamlit's interactive nature eliminates the need for manual data entry and searching. Users can quickly and easily find the information they need, saving time and effort.
* Accessibility: The application's responsive design ensures it can be accessed on various devices, including smartphones and tablets, making it convenient for users on the go.
* Personalization: By allowing users to save their preferences and search history, Streamlit can provide a more personalized experience, tailoring recommendations to their individual needs.
* Engagement: The interactive elements of Streamlit, such as filters and visualizations, can make the process of searching for bus travel more engaging and enjoyable.

By incorporating these enhancements, Streamlit can provide a superior user experience for those planning their bus travel, making it a valuable tool for both individuals and businesses.

# **7.Challenge:**

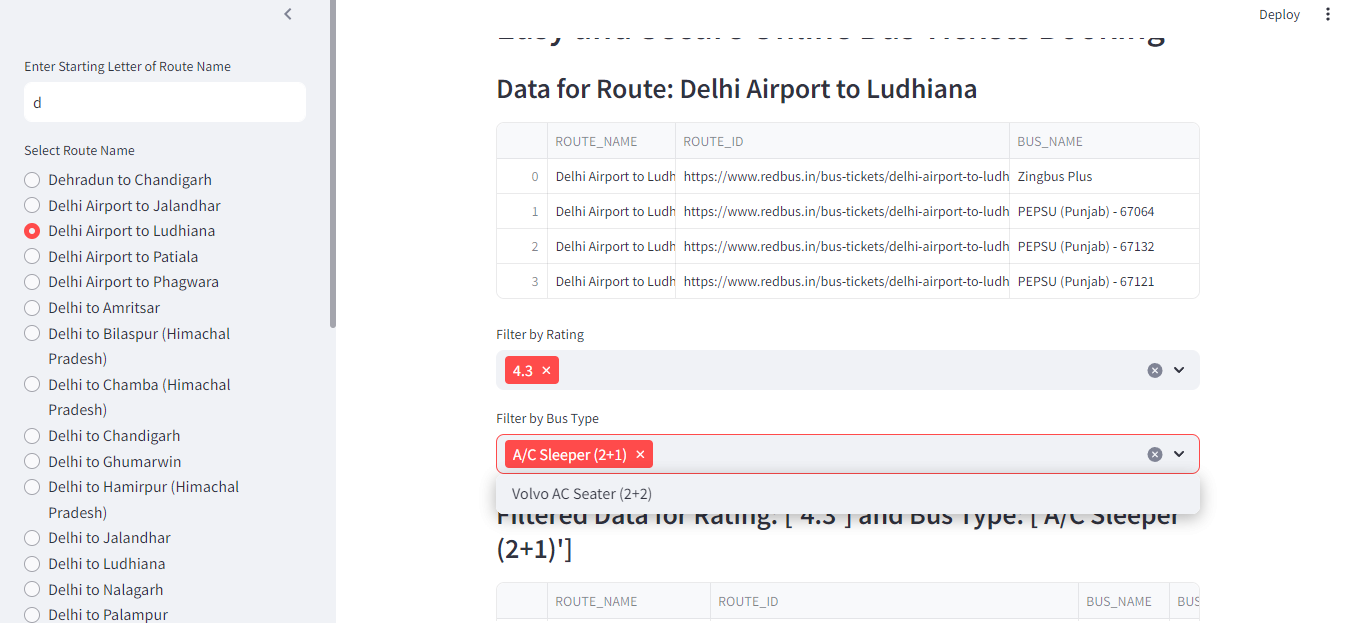
# Handling dynamic content on the Redbus website, such as dynamically generated elements and JavaScript-based interactions.

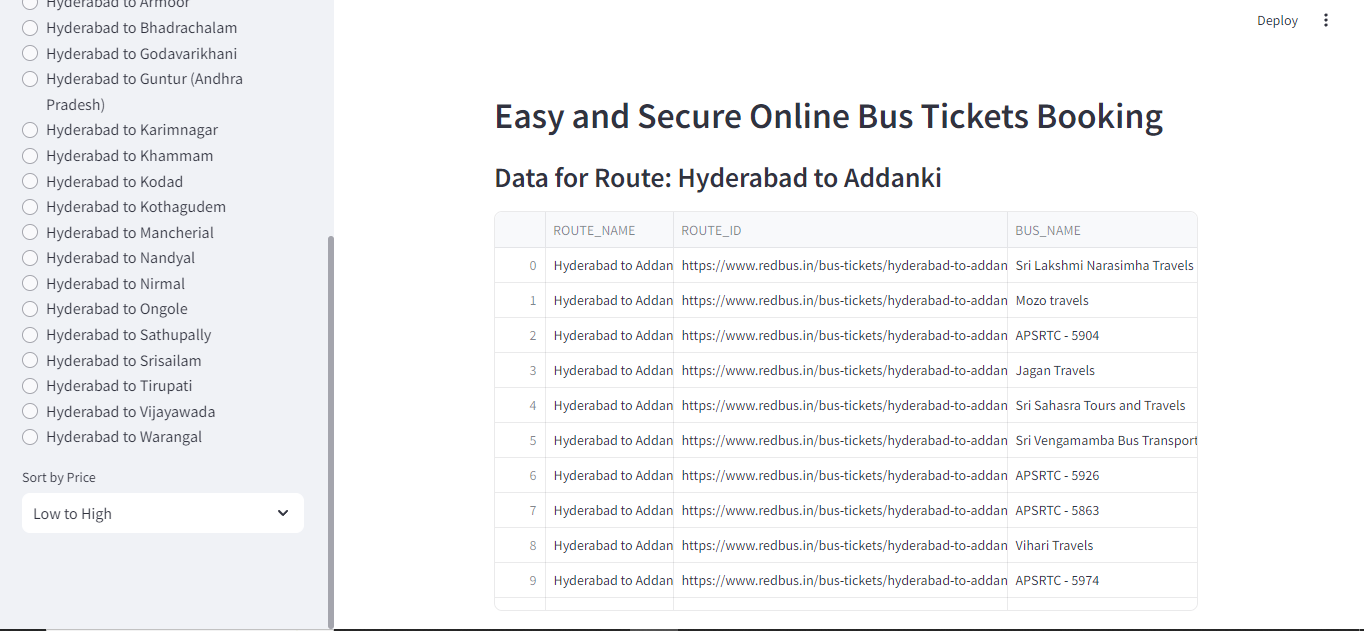
# Solution:

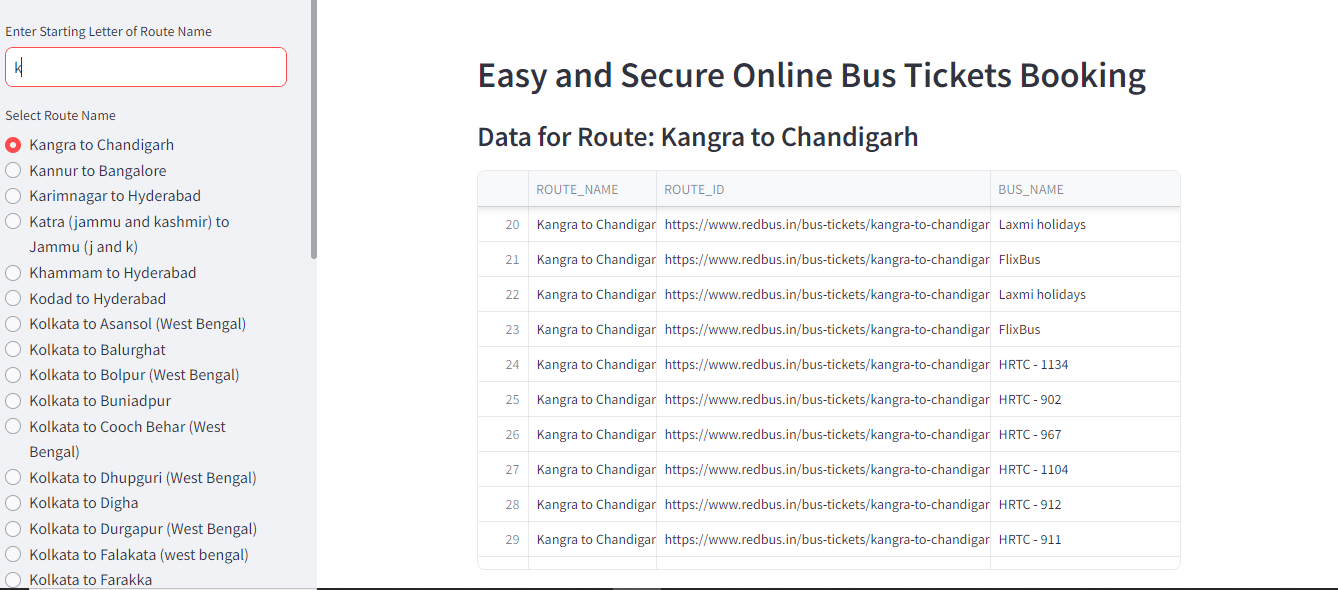
* **Selenium WebDriver:** Use Selenium's WebDriverWait and expected\_conditions to handle dynamic elements and wait for them to load before interacting.
* **Python script execution:** Employ python execution through Selenium to interact with elements that are dynamically added or modified.
* **Regular Expression:** Utilize regular expressions to extract specific data from the HTML content, even if it's dynamically generated

# 8. Screenshots:

Screenshot of my streamlit application:

****

****

****

# 9. Conclusion

Summary of the Project:

The Redbus Streamlit application successfully automates the scraping of bus travel data from Redbus and provides a user-friendly interface for filtering and analyzing the information. By leveraging Selenium, Streamlit, and SQL, the project offers a valuable tool for travel aggregators, market researchers, and individuals planning their bus journeys.

Potential Impact:

The application has the potential to significantly impact the transportation industry in several ways:

* Improved User Experience: By providing real-time, personalized information, the application enhances the user experience for those planning their bus travel.
* Data-Driven Decision Making: The application empowers businesses with data-driven insights to optimize their operations and marketing strategies.
* Increased Efficiency: The automation of data scraping and analysis reduces manual effort and improves efficiency.

Future Directions:

Future enhancements to the application could include:

* Integration with Booking Platforms: Integrating the application with online booking platforms would provide a seamless user experience for travelers.
* Real-time Alerts: Implementing real-time alerts for changes in bus schedules or prices could keep users informed and help them make timely decisions.
* Expansion to Other Transportation Modes: Expanding the application to include data from other transportation modes, such as trains or flights, could provide a more comprehensive travel planning solution.

By continuing to develop and refine the Redbus Streamlit application, it can become a valuable asset for the transportation industry and improve the overall travel experience for users.