**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 import ActionChains

from selenium.webdriver.common.by import By

from selenium.common.exceptions import TimeoutException, NoSuchElementException, ElementClickInterceptedException

from selenium.webdriver.support.ui import WebDriverWait

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

import pandas as pd

import os

import time

# State links

state\_links = [

"https://www.redbus.in/online-booking/ksrtc-kerala/?utm\_source=rtchometile",

"https://www.redbus.in/online-booking/apsrtc/?utm\_source=rtchometile",

"https://www.redbus.in/online-booking/tsrtc/?utm\_source=rtchometile",

"https://www.redbus.in/online-booking/ktcl/?utm\_source=rtchometile",

"https://www.redbus.in/online-booking/rsrtc/?utm\_source=rtchometile",

"https://www.redbus.in/online-booking/south-bengal-state-transport-corporation-sbstc/?utm\_source=rtchometile",

"https://www.redbus.in/online-booking/hrtc/?utm\_source=rtchometile",

"https://www.redbus.in/online-booking/astc/?utm\_source=rtchometile",

"https://www.redbus.in/online-booking/uttar-pradesh-state-road-transport-corporation-upsrtc/?utm\_source=rtchometile",

"https://www.redbus.in/online-booking/wbtc-ctc/?utm\_source=rtchometile"

]

# Initialize the WebDriver

driver = webdriver.Chrome()

wait = WebDriverWait(driver, 20)

# Helper function to scroll to an element and click it

def scroll\_to\_and\_click(driver, element):

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

time.sleep(2) # Ensure some time for scrolling

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

# Function to retrieve bus routes for a state

def get\_state\_bus\_routes(state\_link, path="//a[@class='route']"):

driver.get(state\_link)

time.sleep(3)

driver.maximize\_window()

links = []

routes = []

for i in range(1, 4): # Adjust the range based on the number of pages

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

# Collect route links and names

for link in paths:

href = link.get\_attribute("href")

route\_name = link.text.strip() # Ensure full route name is collected

if href and route\_name: # Ensure both name and link are not empty

links.append(href)

routes.append(route\_name)

try:

# Check if next page exists and navigate

pagination = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@class="DC\_117\_paginationTable"]')))

next\_button\_xpath = f'//div[@class="DC\_117\_pageTabs " and text()="{i+1}"]'

next\_button = driver.find\_element(By.XPATH, next\_button\_xpath)

scroll\_to\_and\_click(driver, next\_button)

except NoSuchElementException:

print(f"No more pages for state: {state\_link}")

break

# Return DataFrame with full route names and links

return pd.DataFrame({"Route\_name": routes, "Route\_link": links})

# Initialize final DataFrame to collect all state data

all\_states\_data = pd.DataFrame()

# Iterate through each state and collect data

for state\_link in state\_links:

df\_state = get\_state\_bus\_routes(state\_link)

all\_states\_data = pd.concat([all\_states\_data, df\_state], ignore\_index=True)

# Check if file path is accessible before saving

output\_path\_routes = r"C:\Users\lokes\OneDrive\Documents\bus\_data\all\_routes1.csv" # Change path if needed

if os.path.exists(output\_path\_routes):

if not os.access(output\_path\_routes, os.W\_OK):

print(f"File at {output\_path\_routes} is not writable. Please check permissions.")

else:

print(f"Saving routes data to {output\_path\_routes}")

all\_states\_data.to\_csv(output\_path\_routes, index=False, encoding='utf-8-sig')

# Function to collect detailed bus information from each route link

def get\_bus\_details(df):

Bus\_names = []

Bus\_types = []

Start\_Time = []

End\_Time = []

Total\_Duration = []

Prices = []

Seats\_Available = []

Ratings = []

Route\_links = []

Route\_names = []

for i, row in df.iterrows():

link = row["Route\_link"]

route = row["Route\_name"]

driver.get(link)

time.sleep(2)

try:

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

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

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

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

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

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

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

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

for bus in bus\_name\_elements:

Bus\_names.append(bus.text)

Route\_links.append(link)

Route\_names.append(route)

for bus\_type in bus\_type\_elements:

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

for start\_time in start\_time\_elements:

Start\_Time.append(start\_time.text)

for end\_time in end\_time\_elements:

End\_Time.append(end\_time.text)

for duration in total\_duration\_elements:

Total\_Duration.append(duration.text)

for rating in rating\_elements:

Ratings.append(rating.text)

for price in price\_elements:

Prices.append(price.text)

for seat in seats\_elements:

Seats\_Available.append(seat.text)

except NoSuchElementException:

print(f"Failed to extract data for route: {route}")

continue

# Create DataFrame from collected bus details

bus\_data = pd.DataFrame({

"Bus\_name": Bus\_names,

"Bus\_type": Bus\_types,

"Start\_time": Start\_Time,

"End\_time": End\_Time,

"Total\_duration": Total\_Duration,

"Price": Prices,

"Seats\_Available": Seats\_Available,

"Ratings": Ratings,

"Route\_link": Route\_links,

"Route\_name": Route\_names

})

return bus\_data

# Get detailed bus information from the routes

bus\_details\_df = get\_bus\_details(all\_states\_data)

# Check if file path is accessible before saving bus details

output\_path\_bus\_details = r"C:\Users\lokes\OneDrive\Documents\bus\_data\redbus.csv" # Change path if needed

if os.path.exists(output\_path\_bus\_details):

if not os.access(output\_path\_bus\_details, os.W\_OK):

print(f"File at {output\_path\_bus\_details} is not writable. Please check permissions.")

else:

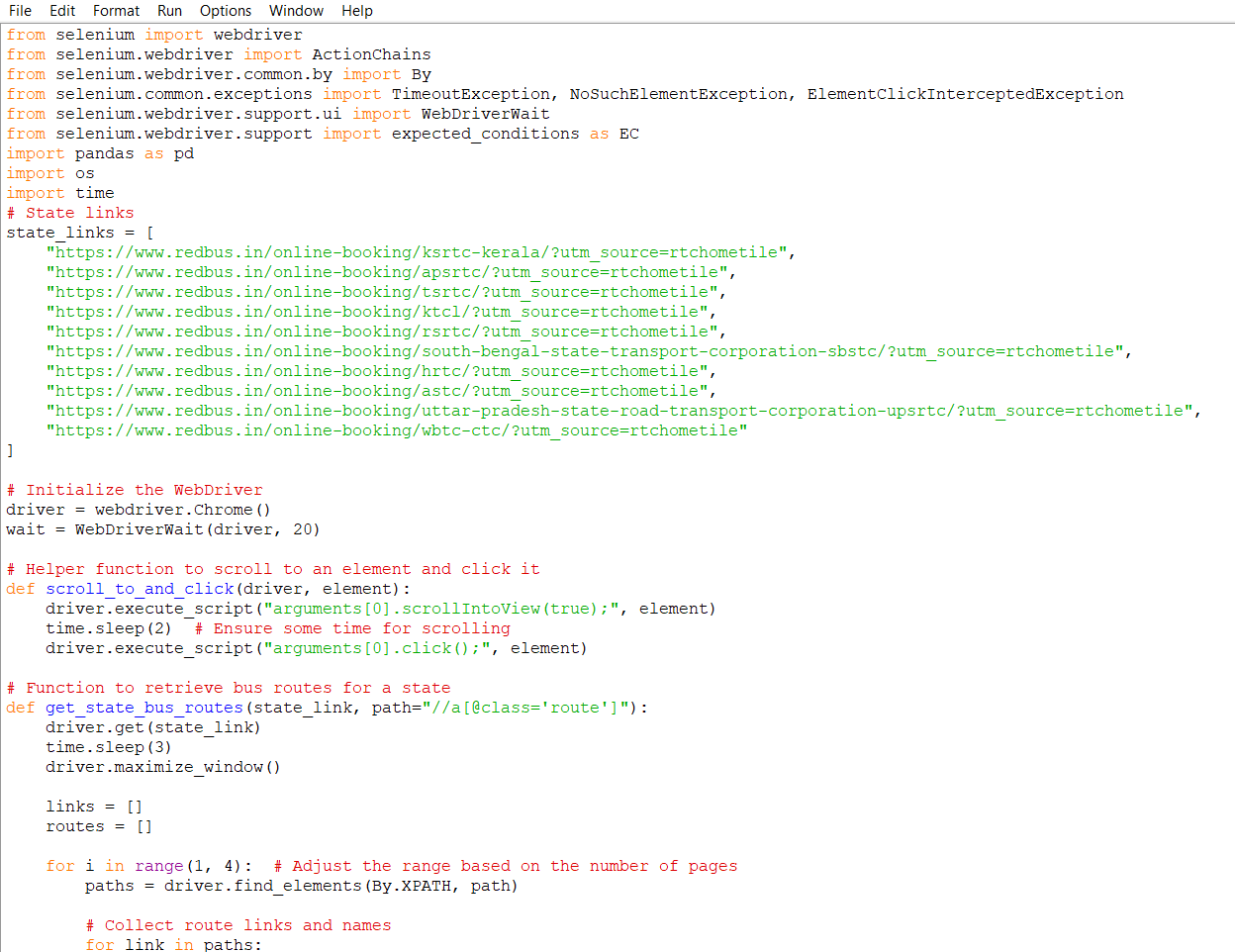
print(f"Saving bus details to {output\_path\_bus\_details}")

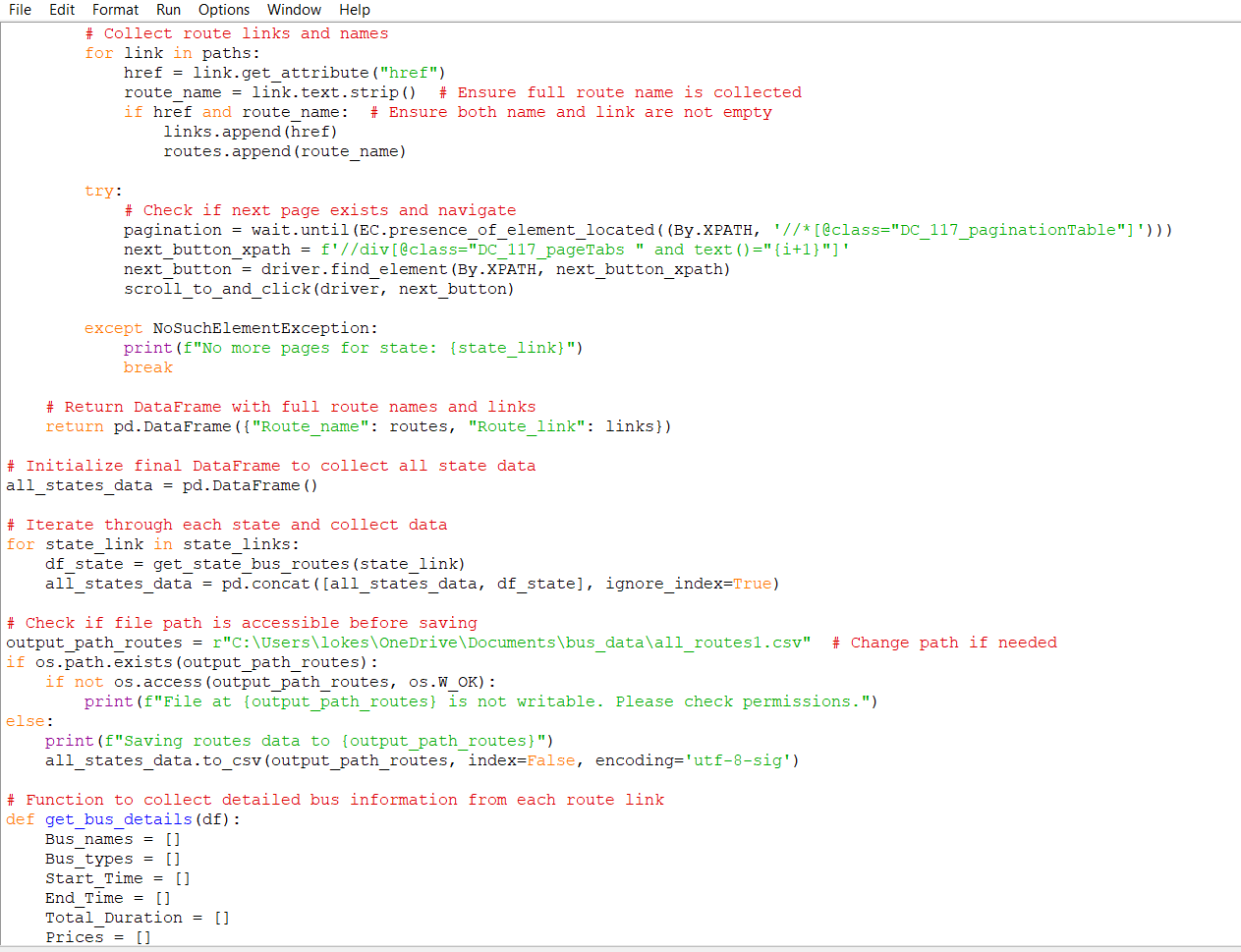
bus\_details\_df.to\_csv(output\_path\_bus\_details, index=False, encoding='utf-8-sig')

# Close the browser

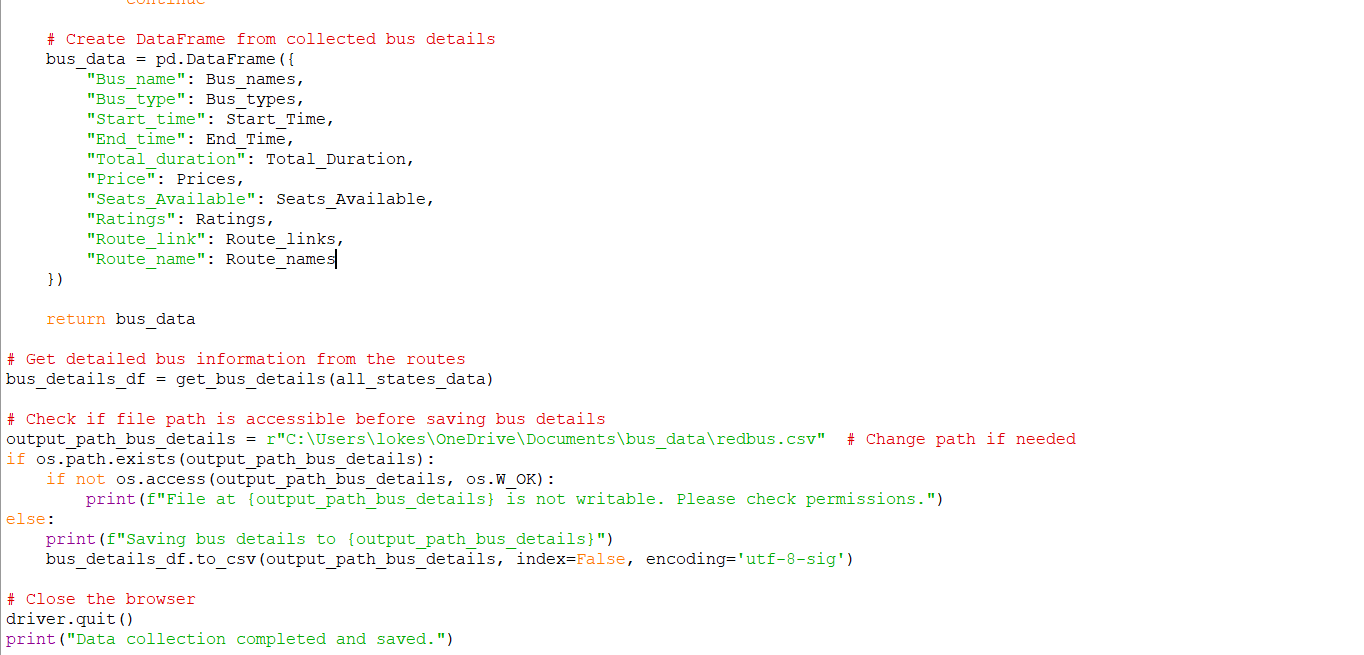
driver.quit()

print("Data collection completed and saved.")



****

****

****

***All route website links*:**

* **kerala:**[**https://www.redbus.in/online-booking/ksrtc-kerala/?utm\_source=rtchometile**](https://www.redbus.in/online-booking/ksrtc-kerala/?utm_source=rtchometile)
* **telegana:**[**https://www.redbus.in/online-booking/tsrtc/?utm\_source=rtchometile**](https://www.redbus.in/online-booking/tsrtc/?utm_source=rtchometile)
* **kadamba:**[**https://www.redbus.in/online-booking/ktcl/?utm\_source=rtchometile**](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**](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**](https://www.redbus.in/online-booking/hrtc/?utm_source=rtchometile)
* **assam:**[**https://www.redbus.in/online-booking/astc/?utm\_source=rtchometile**](https://www.redbus.in/online-booking/astc/?utm_source=rtchometile)
* **westbengal:**[**https://www.redbus.in/online-booking/wbtc-ctc/?utm\_source=rtchometile**](https://www.redbus.in/online-booking/wbtc-ctc/?utm_source=rtchometile)
* **chandigarh:**[**https://www.redbus.in/online-booking/chandigarh-transport-undertaking-ctu**](https://www.redbus.in/online-booking/chandigarh-transport-undertaking-ctu)
* **punjab:**[**https://www.redbus.in/online-booking/pepsu/?utm\_source=rtchometile**](https://www.redbus.in/online-booking/pepsu/?utm_source=rtchometile)
* **Jammu and kashmir:**[**https://www.redbus.in/online-booking/jksrtc**](https://www.redbus.in/online-booking/jksrtc)

***SQL :***

-- Step 1: Create the Database

CREATE DATABASE redbus\_travel;

-- Step 2: Use the created database

USE redbus\_travel;

-- Step 3: Create the 'project\_info' table for bus details

CREATE TABLE project\_info (

id INT AUTO\_INCREMENT PRIMARY KEY, -- Make 'id' auto-incrementing for easier insertion

Bus\_name VARCHAR(50),

Bus\_type VARCHAR(50),

Start\_time TIME,

End\_time TIME,

Total\_duration VARCHAR(50),

Price DECIMAL(10, 2),

Seats\_Available INT,

Ratings FLOAT,

Route\_link VARCHAR(500), -- Typo correction: "Root\_link" should be "Route\_link"

Route\_name VARCHAR(50)

);

-- Step 4: Create the route tables for each state

CREATE TABLE kerala\_routes (

Route\_name VARCHAR(255),

Route\_link VARCHAR(500)

);

CREATE TABLE andhra\_routes (

Route\_name VARCHAR(255),

Route\_link VARCHAR(500)

);

CREATE TABLE telangana\_routes (

Route\_name VARCHAR(255),

Route\_link VARCHAR(500)

);

CREATE TABLE kadamba\_routes (

Route\_name VARCHAR(255),

Route\_link VARCHAR(500)

);

CREATE TABLE rajasthan\_routes (

Route\_name VARCHAR(255),

Route\_link VARCHAR(500)

);

CREATE TABLE southbengal\_routes (

Route\_name VARCHAR(255),

Route\_link VARCHAR(500)

);

CREATE TABLE haryana\_routes (

Route\_name VARCHAR(255),

Route\_link VARCHAR(500)

);

CREATE TABLE assam\_routes (

Route\_name VARCHAR(255),

Route\_link VARCHAR(500)

);

CREATE TABLE uttarpradesh\_routes (

Route\_name VARCHAR(255),

Route\_link VARCHAR(500)

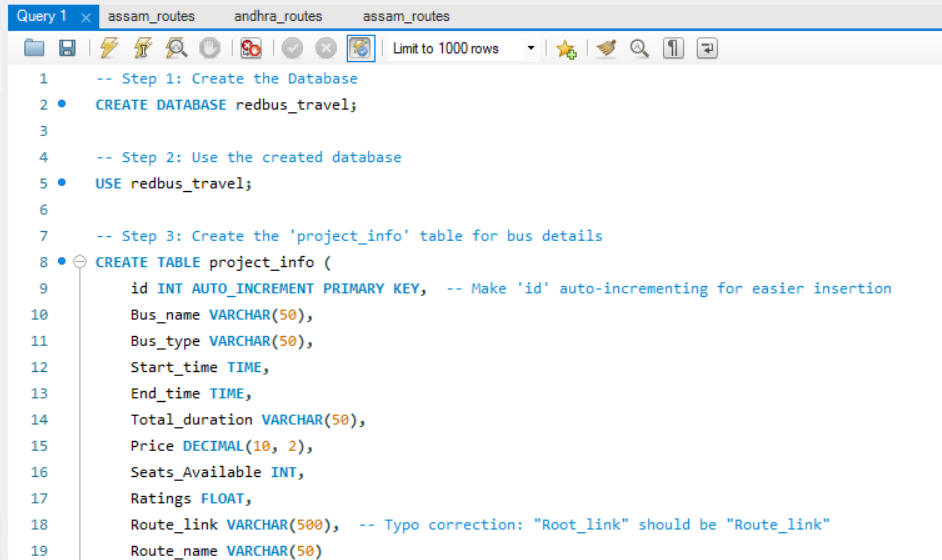
);

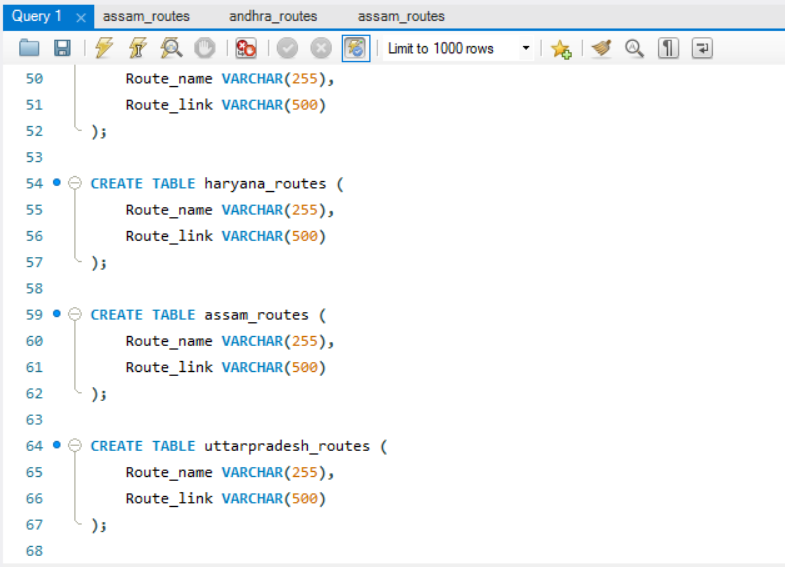
CREATE TABLE westbengal\_routes (

Route\_name VARCHAR(255),

Route\_link VARCHAR(500)

);

****



***STREAMLIT APPLICATION***

import streamlit as st

import pymysql

import pandas as pd

# Connect to MySQL database

def get\_connection():

return pymysql.connect(host='localhost', user='root', passwd='Lokesh#7', database='redbus\_travel')

# Function to fetch route names from a specific state table starting with a specific letter

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

query = f"SELECT DISTINCT Route\_name FROM {state}\_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 the project\_info table based on selected ROUTE\_NAME

def fetch\_data(connection, route\_name):

query = "SELECT \* FROM project\_info WHERE Route\_name = %s"

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['Ratings'].isin(ratings) & df['Bus\_type'].isin(bus\_types)]

return filtered\_df

# Terms and Conditions Content

def terms\_and\_conditions():

st.header("Terms and Conditions")

st.write("""

1. General Terms

- All users must provide accurate personal information for booking.

- Bookings are non-refundable unless otherwise specified.

2. Payment Terms

- Payment must be completed online using valid payment methods.

- The system will display total pricing, including applicable taxes.

3. Cancellation and Refunds

- Cancellations are subject to the terms set by each bus operator.

- Refunds will be processed based on the cancellation policy of the operator.

4. User Responsibilities

- Users are responsible for providing correct travel details and adhering to the operator's policies.

""")

# FAQ Content

def faq\_section():

st.header("Frequently Asked Questions (FAQ)")

st.write("""

1. How do I book a bus ticket?

- You can book a bus ticket by selecting your desired route and bus type, then following the booking process on the homepage.

2. What payment methods do you accept?

- We accept major credit cards, debit cards, and online banking.

3. Can I cancel my booking?

- Yes, you can cancel your booking through the "Manage Booking" section. However, cancellations are subject to the operator's policies.

4. How will I receive my booking confirmation?

- Once your booking is successful, you will receive a confirmation email with your ticket details.

""")

# Main Streamlit app

def main():

st.sidebar.title("Navigation")

# Sidebar - Option to choose between Booking, Terms and Conditions, and FAQ

page = st.sidebar.radio("Go to", ["Bus Booking", "Terms and Conditions", "FAQ"])

if page == "Bus Booking":

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')

# Sidebar - Selectbox for state

states = ['kerala', 'andhra', 'telangana', 'kadamba', 'rajasthan',

'southbengal', 'haryana', 'assam', 'uttarpradesh', 'westbengal']

selected\_state = st.sidebar.selectbox('Select State', states)

# Fetch route names starting with the specified letter

if starting\_letter and selected\_state:

route\_names = fetch\_route\_names(connection, starting\_letter.lower(), selected\_state)

if route\_names:

# Sidebar - Selectbox for ROUTE\_NAME

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

if selected\_route:

# Fetch data based on selected ROUTE\_NAME

data = fetch\_data(connection, selected\_route)

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['Ratings'].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)

# Select number of seats

available\_seats = filtered\_data['Seats\_Available'].sum()

if available\_seats > 0:

num\_seats = st.number\_input('Select Number of Seats', min\_value=1, max\_value=available\_seats)

# Display price information

price\_per\_seat = filtered\_data['Price'].mean() # Average price for selected buses

total\_price = num\_seats \* price\_per\_seat

st.write(f"Price per seat: ₹{price\_per\_seat:.2f}")

st.write(f"Total price for {num\_seats} seats: ₹{total\_price:.2f}")

# "Book Now" button

if st.button('Book Now'):

# Success message for successful booking

st.success(f"Successfully booked {num\_seats} seats for {selected\_route} at ₹{total\_price:.2f}!")

else:

st.warning("No seats available for the selected filters.")

else:

st.write(f"No data found for Route: {selected\_route}.")

else:

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

finally:

connection.close()

elif page == "Terms and Conditions":

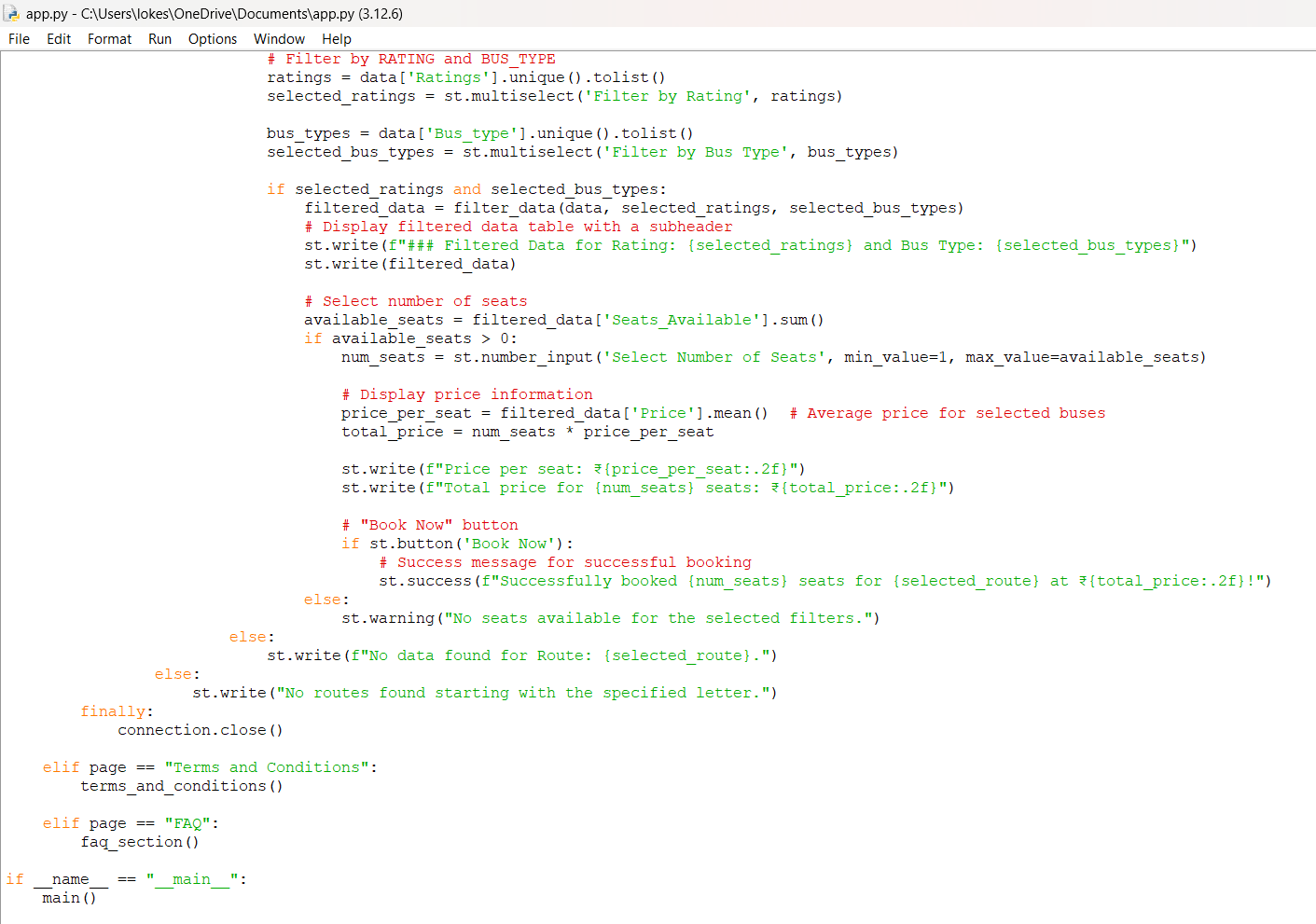
terms\_and\_conditions()

elif page == "FAQ":

faq\_section()

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:***

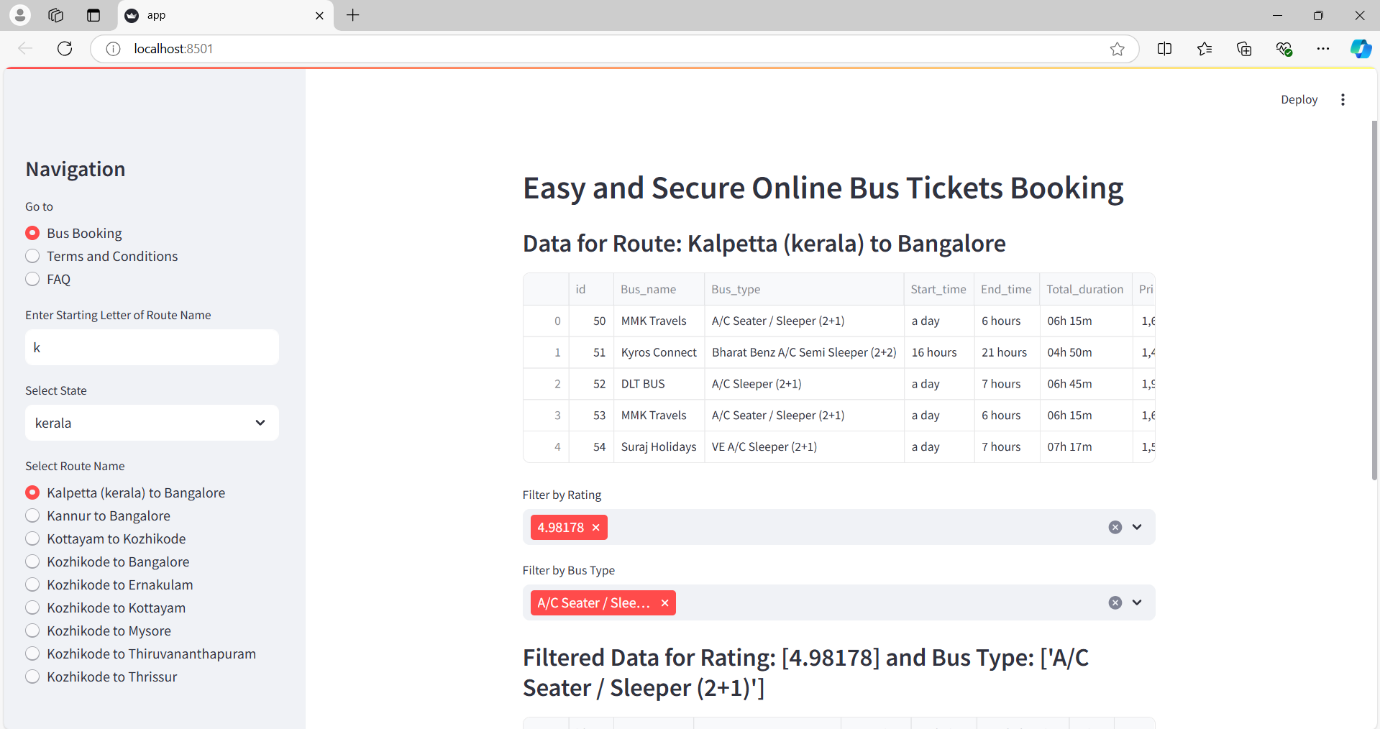
Handling dynamic content on the Redbus website, such as dynamically generated elements and JavaScript-based interactions.

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.

THANK YOU