Redbus Data Scraping with Selenium & Dynamic Filtering using Streamlit

Domain: Transportation

1. Problem Statement

The goal of this project is to create a **dynamic bus booking platform** for users to scrape **real-time bus data** from RedBus using **Selenium**. The scrapped data will be filtered dynamically using **Streamlit** to provide users with specific bus options based on their requirements. This project aims to address the challenges of **manually searching** for buses, and **filtering results** of **real-time bus data**.

2 . Approach

1. **Web Scraping**: We use **Selenium** to scrape real-time bus data from the **RedBus** website. The data includes **bus name**, **departure time**, **price**, **availability**, and more.
2. **Dynamic Filtering**: Once data is scraped, we use **Streamlit** to create an interactive interface. Users can apply dynamic filters based on **bus type**, **price**, **departure time**, etc.
3. **Data Storage**: The data can be saved in a **MySQL database** for future use and analysis.
4. **User Interface**: Streamlit provides a simple and intuitive UI to present the scraped data in a structured and readable format.

3. Technology Used

- **Python**: The main programming language used for this project.
- **Selenium**: A tool used for web scraping to extract real-time bus data.
- **Streamlit**: A framework used to create the web interface and dynamically filter the scraped data.
- **MySQL**: A relational database used to store the bus data for future analysis.
- **Pandas**: Used for data manipulation and analysis.

4. Use Cases

1. **Real-time Bus Search**: Users can search for buses between any two cities and see real-time data on bus availability, prices, departure times, etc.
2. **Filter Options**: Users can filter bus results by **departure time**, **price**, **bus type**, etc., to find the most suitable bus for their journey.
3. **Pricing Analysis**: Users can compare prices across different buses to choose the best option.

4. **Booking Assistance**: The app can be extended to integrate with the booking process, providing users with a seamless experience.

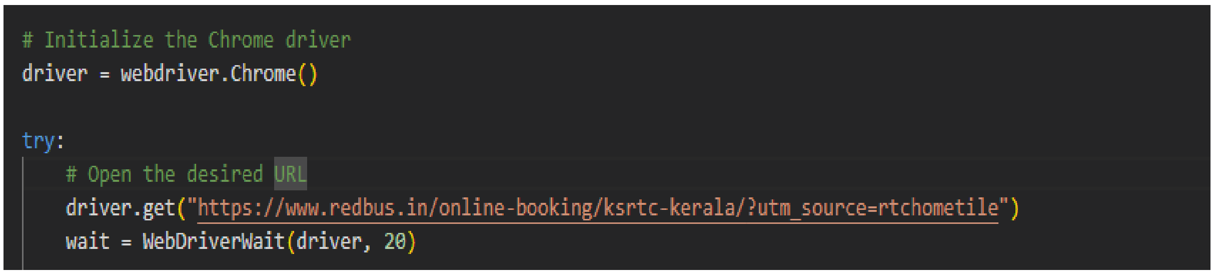
5. Code

1. Install dependencies

   Pip install -r requirements.txt

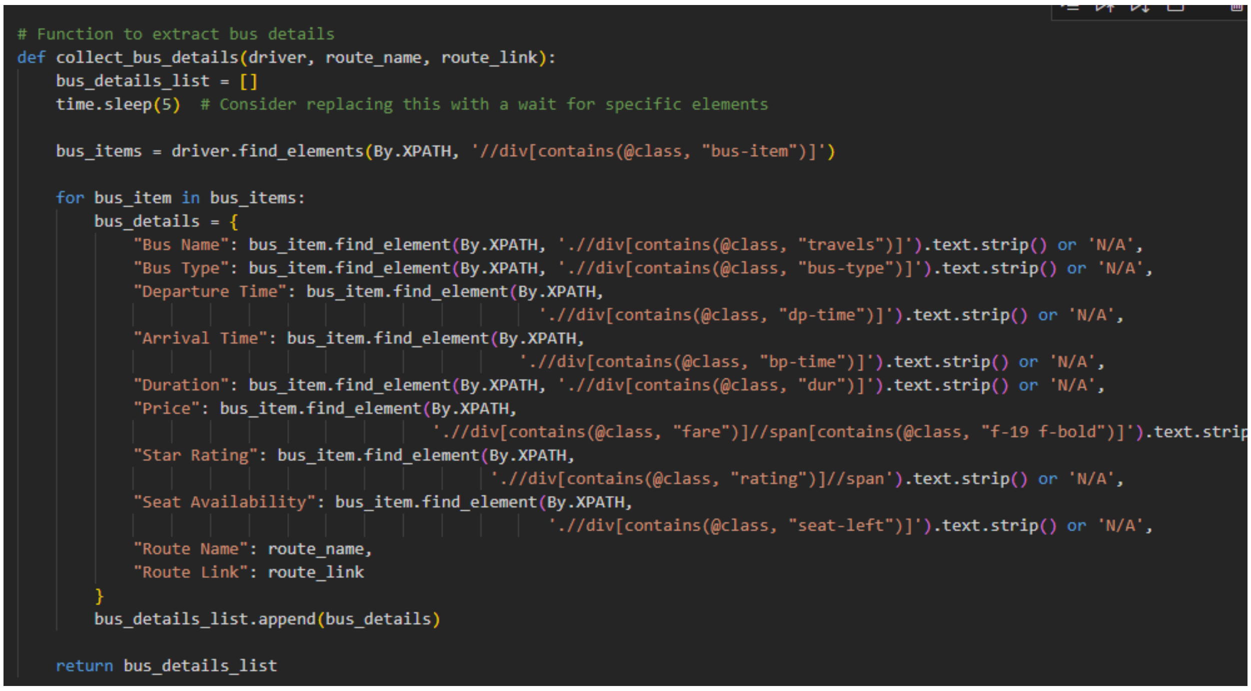
2. Data scrapping

   - Initialize WebDriver



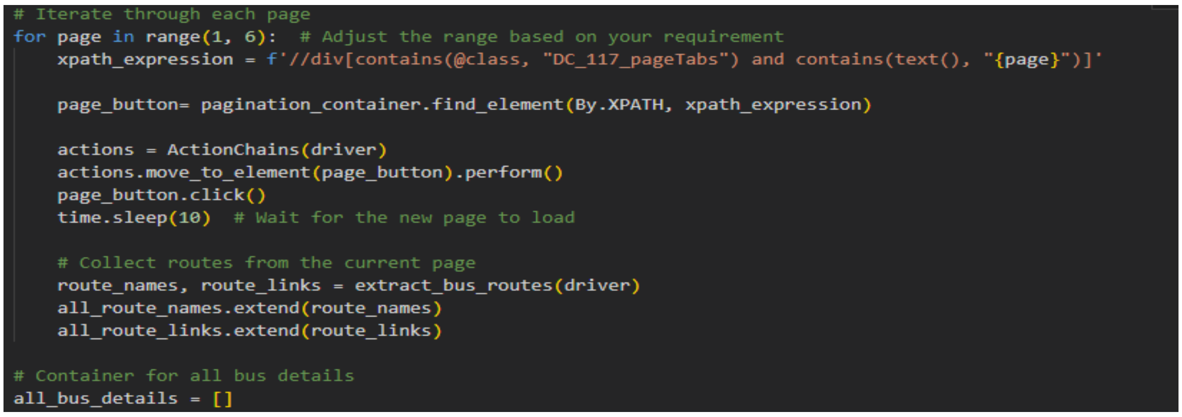
```python
# Initialize the Chrome driver
driver = webdriver.Chrome()

try:
    # Open the desired URL
    driver.get("https://www.redbus.in/online-booking/ksrtc-kerala/?utm_source=rtchometile")
    wait = WebDriverWait(driver, 20)
```

   - Function to extract all necessary bus details of government states



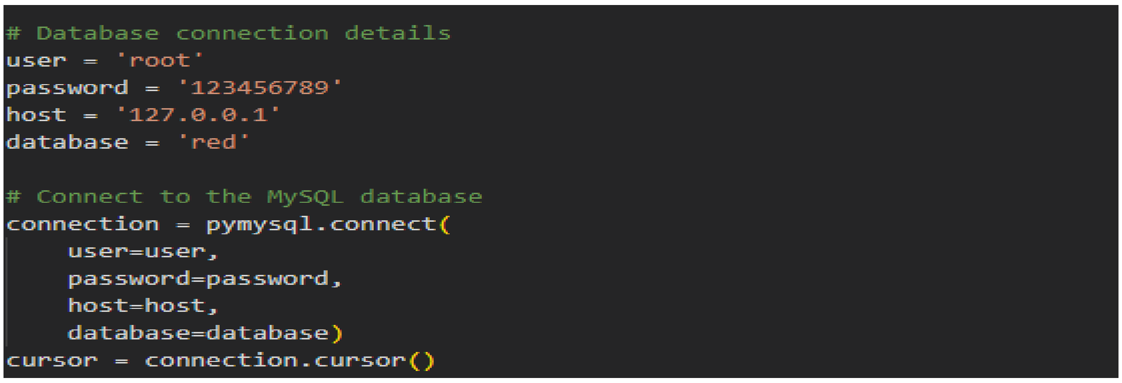
```python
# Function to extract bus details
def collect_bus_details(driver, route_name, route_link):
    bus_details_list = []
    time.sleep(5)  # Consider replacing this with a wait for specific elements

    bus_items = driver.find_elements(By.XPATH, '//div[contains(@class, "bus-item")]')

    for bus_item in bus_items:
        bus_details = {
            "Bus Name": bus_item.find_element(By.XPATH, './/div[contains(@class, "travels")]').text.strip() or 'N/A',
            "Bus Type": bus_item.find_element(By.XPATH, './/div[contains(@class, "bus-type")]').text.strip() or 'N/A',
            "Departure Time": bus_item.find_element(By.XPATH,
                                                    './/div[contains(@class, "dp-time")]').text.strip() or 'N/A',
            "Arrival Time": bus_item.find_element(By.XPATH,
                                                  './/div[contains(@class, "bp-time")]').text.strip() or 'N/A',
            "Duration": bus_item.find_element(By.XPATH, './/div[contains(@class, "dur")]').text.strip() or 'N/A',
            "Price": bus_item.find_element(By.XPATH,
                                           './/div[contains(@class, "fare")]//span[contains(@class, "f-19 f-bold")]').text.strip
            "Star Rating": bus_item.find_element(By.XPATH,
                                                 './/div[contains(@class, "rating")]//span').text.strip() or 'N/A',
            "Seat Availability": bus_item.find_element(By.XPATH,
                                                       './/div[contains(@class, "seat-left")]').text.strip() or 'N/A',
            "Route Name": route_name,
            "Route Link": route_link
        }
        bus_details_list.append(bus_details)

    return bus_details_list
```

- Iterate to all pages of redbus under each state.



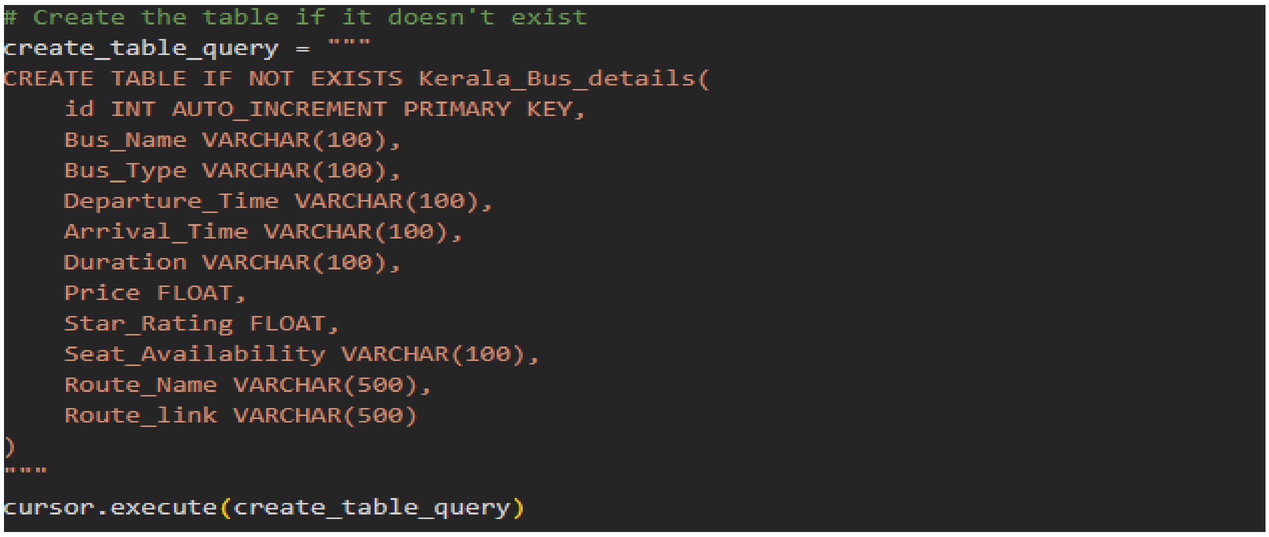
```python
# Iterate through each page
for page in range(1, 6):  # Adjust the range based on your requirement
    xpath_expression = f'//div[contains(@class, "DC_117_pageTabs") and contains(text(), "{page}")]'

    page_button= pagination_container.find_element(By.XPATH, xpath_expression)

    actions = ActionChains(driver)
    actions.move_to_element(page_button).perform()
    page_button.click()
    time.sleep(10)  # Wait for the new page to load

    # Collect routes from the current page
    route_names, route_links = extract_bus_routes(driver)
    all_route_names.extend(route_names)
    all_route_links.extend(route_links)

# Container for all bus details
all_bus_details = []
```

- Converting all extracted details to dataframe and stored in database by establishing the MySQL connection.



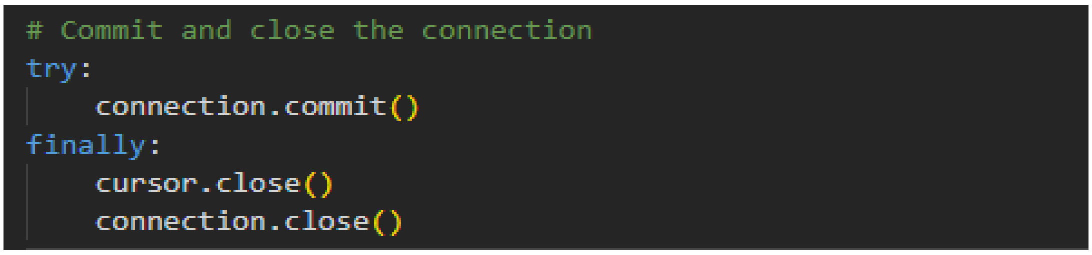
```python
# Database connection details
user = 'root'
password = '123456789'
host = '127.0.0.1'
database = 'red'

# Connect to the MySQL database
connection = pymysql.connect(
    user=user,
    password=password,
    host=host,
    database=database)
cursor = connection.cursor()
```

- Tables are created using SQL query to insert all bus details in state wise



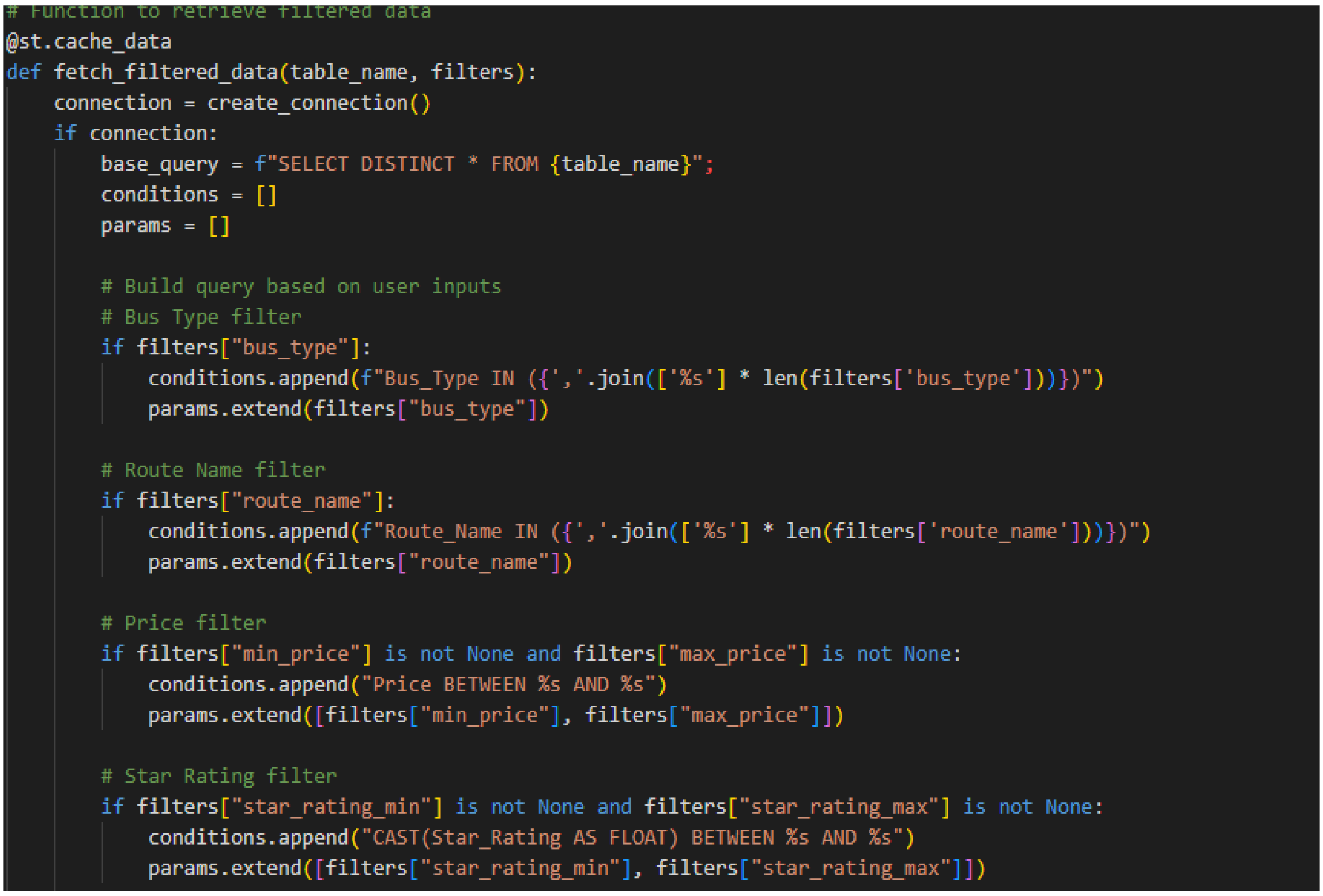
```python
# Create the table if it doesn't exist
create_table_query = """
CREATE TABLE IF NOT EXISTS Kerala_Bus_details(
    id INT AUTO_INCREMENT PRIMARY KEY,
    Bus_Name VARCHAR(100),
    Bus_Type VARCHAR(100),
    Departure_Time VARCHAR(100),
    Arrival_Time VARCHAR(100),
    Duration VARCHAR(100),
    Price FLOAT,
    Star_Rating FLOAT,
    Seat_Availability VARCHAR(100),
    Route_Name VARCHAR(500),
    Route_link VARCHAR(500)
)
"""
cursor.execute(create_table_query)
```

- Commit and close the connection



```
# Commit and close the connection
try:
    connection.commit()
finally:
    cursor.close()
    connection.close()
```

3. Streamlit application development and dynamic filtering of analysed data insights from redbus.



```
# Function to retrieve filtered data
@st.cache_data
def fetch_filtered_data(table_name, filters):
    connection = create_connection()
    if connection:
        base_query = f"SELECT DISTINCT * FROM {table_name}";
        conditions = []
        params = []

        # Build query based on user inputs
        # Bus Type filter
        if filters["bus_type"]:
            conditions.append(f"Bus_Type IN ({','.join(['%s'] * len(filters['bus_type']))})")
            params.extend(filters["bus_type"])

        # Route Name filter
        if filters["route_name"]:
            conditions.append(f"Route_Name IN ({','.join(['%s'] * len(filters['route_name']))})")
            params.extend(filters["route_name"])

        # Price filter
        if filters["min_price"] is not None and filters["max_price"] is not None:
            conditions.append("Price BETWEEN %s AND %s")
            params.extend([filters["min_price"], filters["max_price"]])

        # Star Rating filter
        if filters["star_rating_min"] is not None and filters["star_rating_max"] is not None:
            conditions.append("CAST(Star_Rating AS FLOAT) BETWEEN %s AND %s")
            params.extend([filters["star_rating_min"], filters["star_rating_max"]])
```

HINT:

Using xpath we can scrape the data.

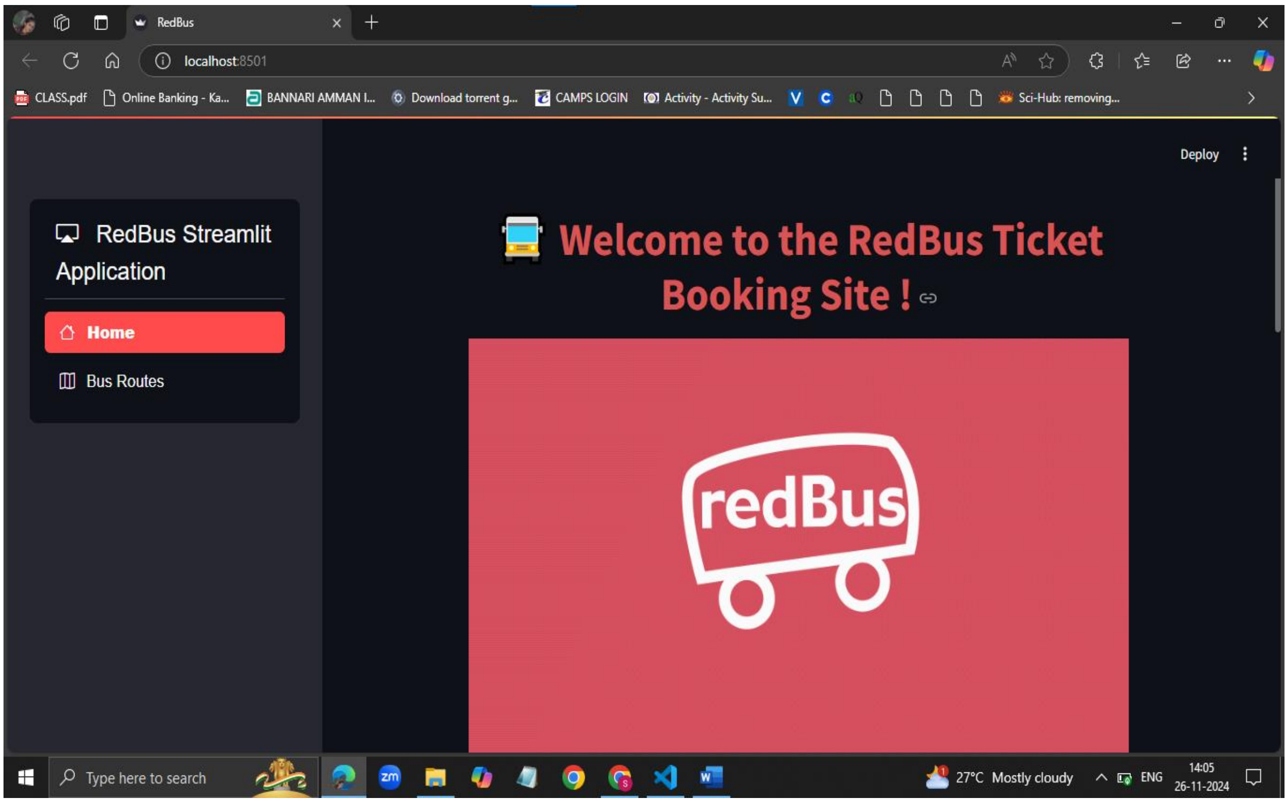
yourdriver.find_elements(By.XPATH,"//div[@class='']")

TO run code:

Streamlit run app.py

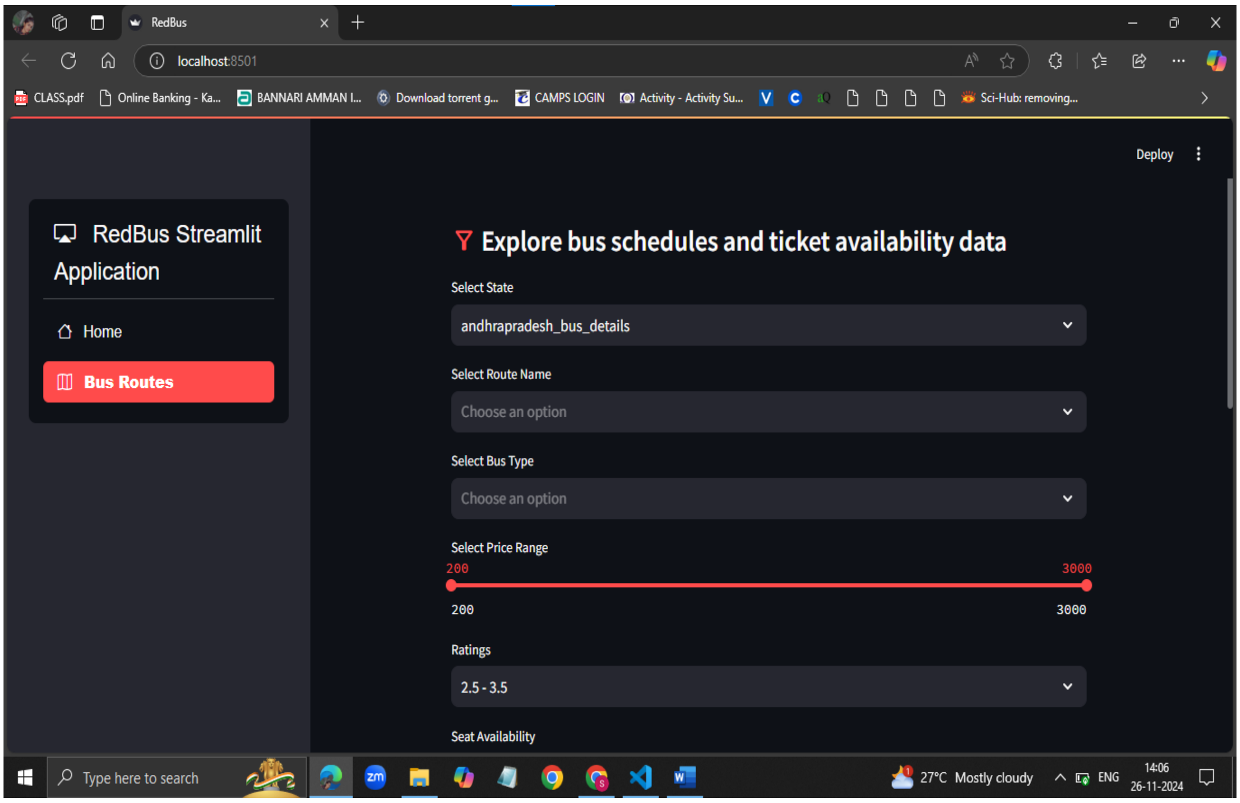
6. Output:

1. HOME PAGE



Welcome to the RedBus Ticket Booking Site !

2. DYNAMIC FILTERING OF DATA



Explore bus schedules and ticket availability data

Select State: andhrapradesh_bus_details

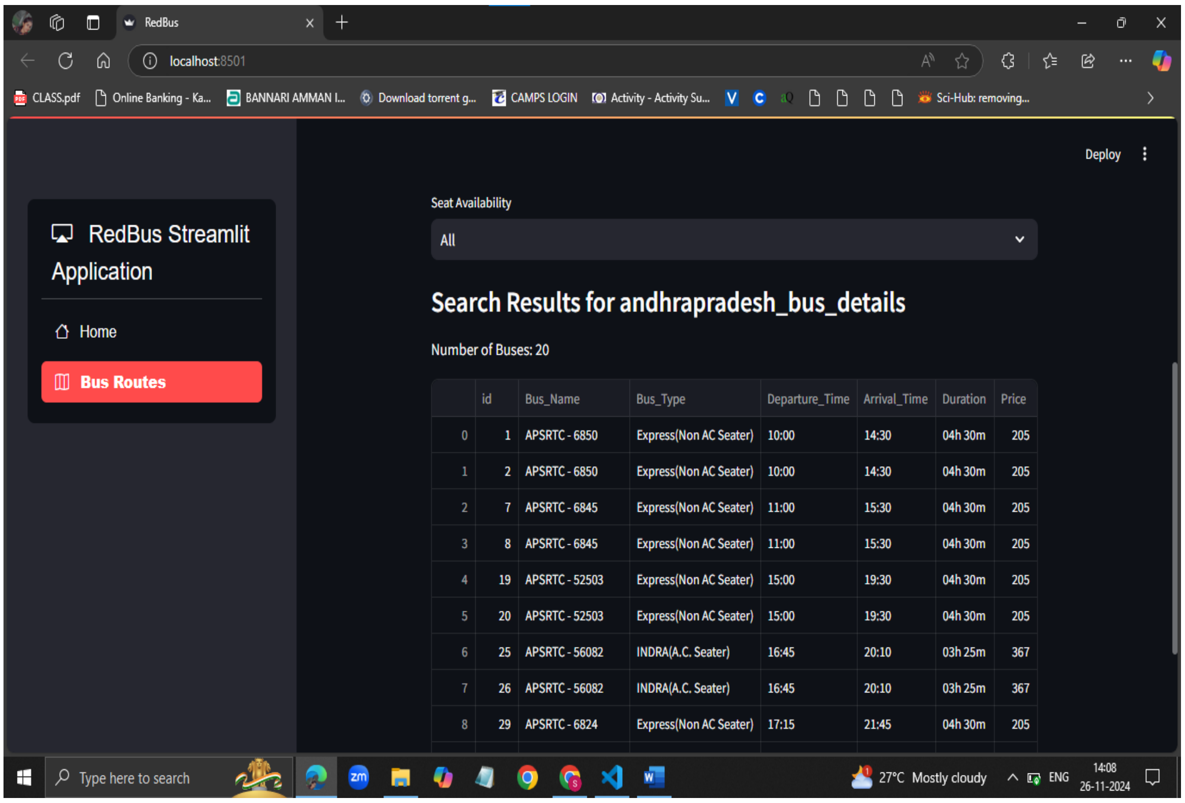
Select Route Name: Choose an option

Select Bus Type: Choose an option

Select Price Range: 200 – 3000

Ratings: 2.5 - 3.5

Seat Availability



Seat Availability

All

Search Results for andhrapradesh_bus_details

Number of Buses: 20

	id	Bus_Name	Bus_Type	Departure_Time	Arrival_Time	Duration	Price
0	1	APSRTC - 6850	Express(Non AC Seater)	10:00	14:30	04h 30m	205
1	2	APSRTC - 6850	Express(Non AC Seater)	10:00	14:30	04h 30m	205
2	7	APSRTC - 6845	Express(Non AC Seater)	11:00	15:30	04h 30m	205
3	8	APSRTC - 6845	Express(Non AC Seater)	11:00	15:30	04h 30m	205
4	19	APSRTC - 52503	Express(Non AC Seater)	15:00	19:30	04h 30m	205
5	20	APSRTC - 52503	Express(Non AC Seater)	15:00	19:30	04h 30m	205
6	25	APSRTC - 56082	INDRA(A.C. Seater)	16:45	20:10	03h 25m	367
7	26	APSRTC - 56082	INDRA(A.C. Seater)	16:45	20:10	03h 25m	367
8	29	APSRTC - 6824	Express(Non AC Seater)	17:15	21:45	04h 30m	205

7. Final Analysis

This application allows you to view bus details scraped from RedBus and stored in a MySQL database. The RedBus Streamlit Application is a demo project to showcase how to build a simple and interactive web application using Streamlit. It leverages real-time data scraping and integrates it with a MySQL database for efficient data retrieval and display.