REDBUS DATA SCRAPING WITH SELENIUM & DYNAMIC FILTERING USING STREAMLIT

BY
NIVETHA BALASUBRAMANIAM
MDTE11

INTRODUCTION

1.Data scraping in redbus

2.Data store in SQL and create Table

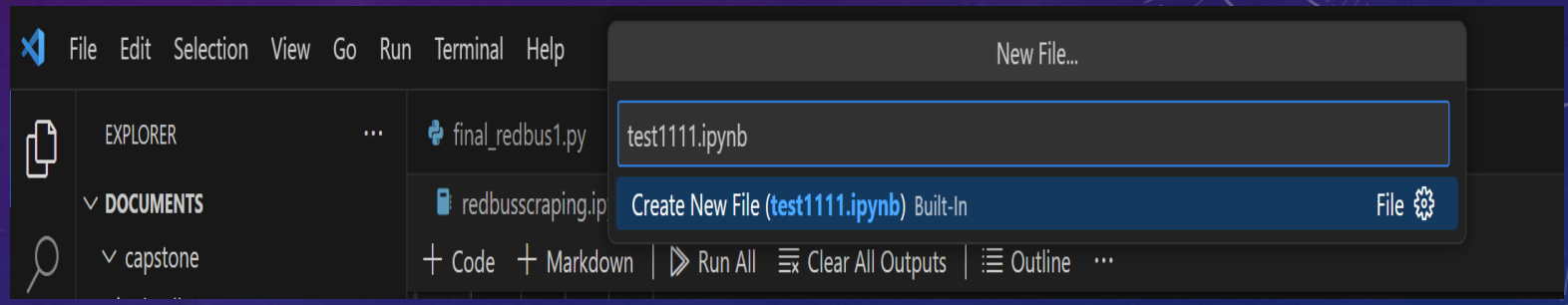
3.Streamlit app

INTRODUCTION

- What is data scraping?
- Why we use data scraping?
- How to scrap the data in redbus?
- MYSQL connection, create table and store the data.
- Using Streamlit app create.
- Create Github
- Upload Github

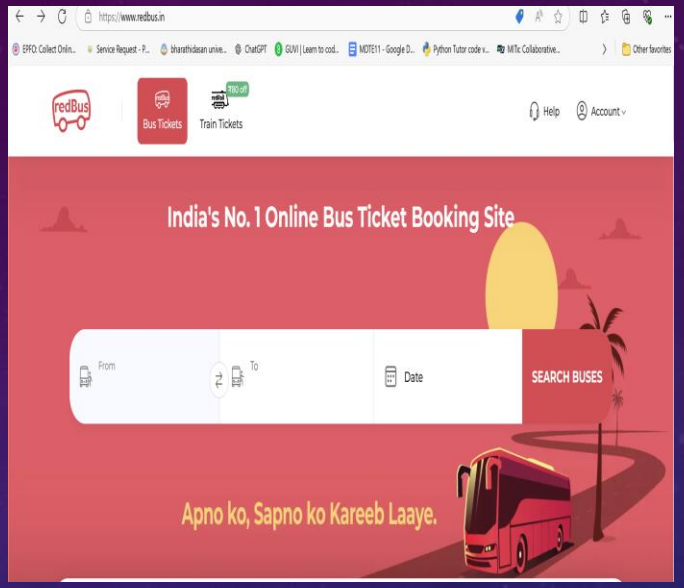
VISUAL STUDIO CODE

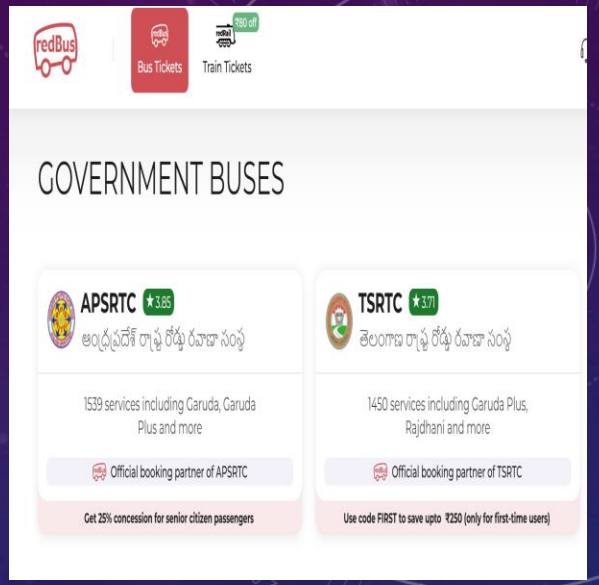
- New file open .ipynb
- !pip install pandas
- Set up the chrome driver
- Driver = webdriver Chrome()
- Driver get(URL)
- Driver.maximize_window()
- Time sleep(5)

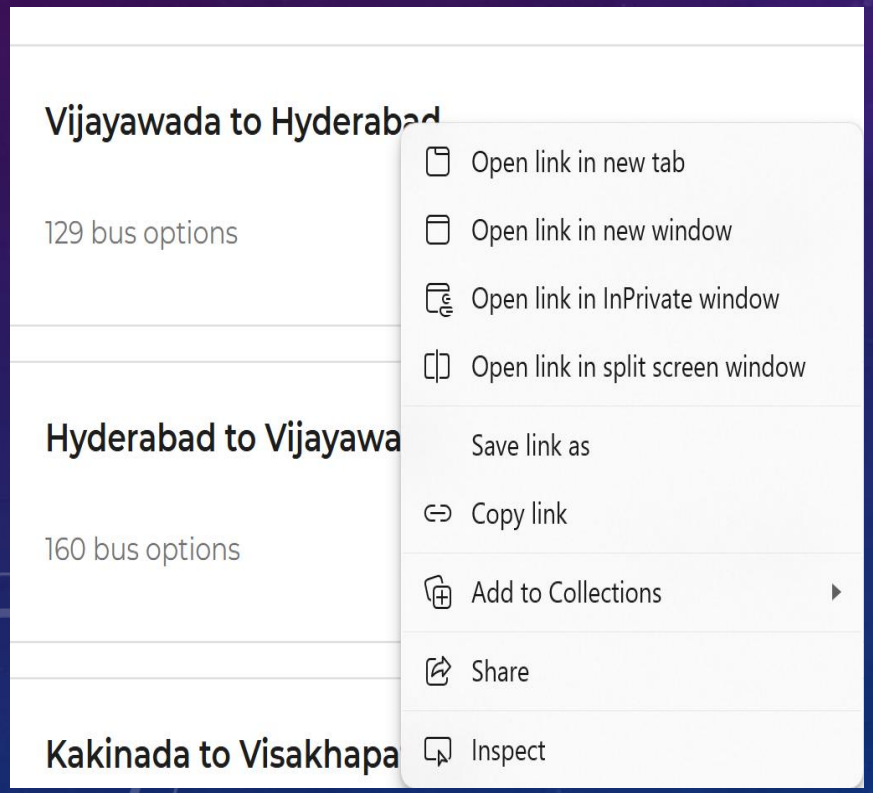


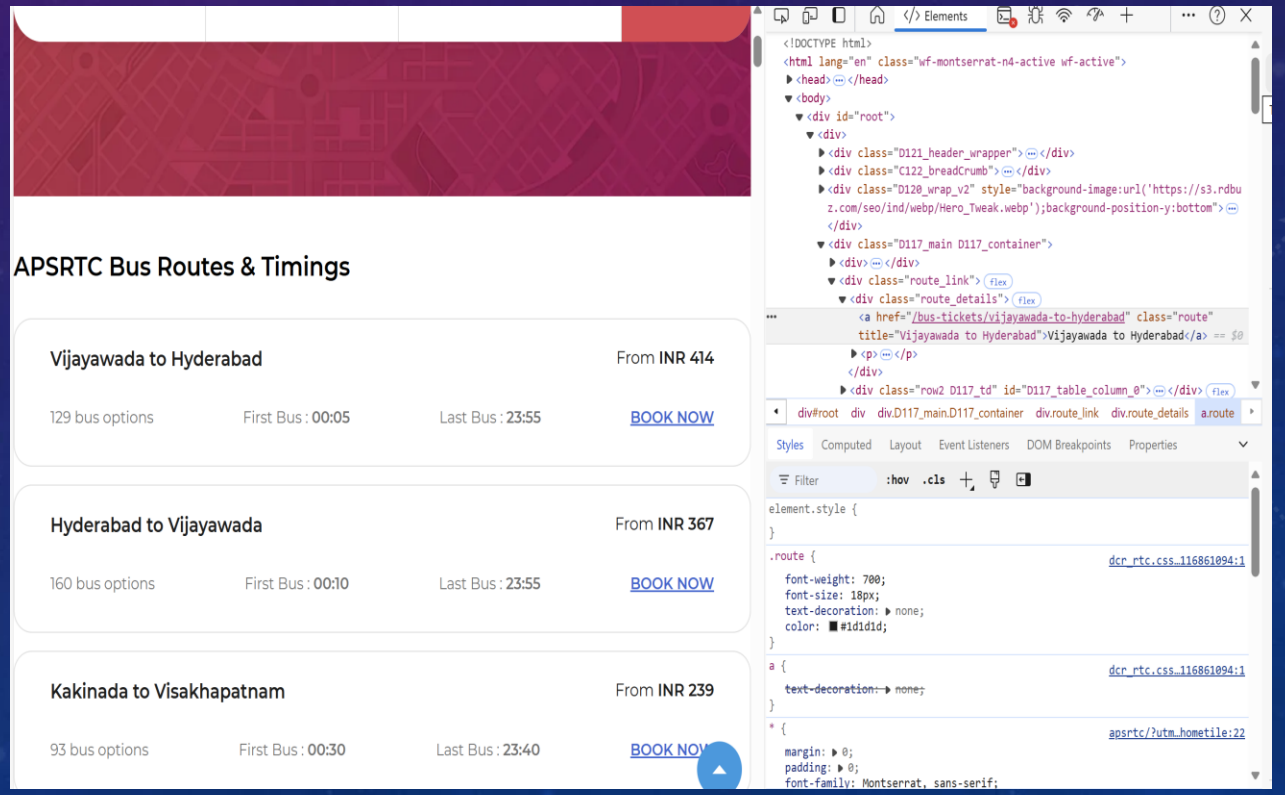
DATA SCRAPING

- Open URL:"https://www.redbus.in
- Government bus detail scrape like(Bus name,Bus link etc)





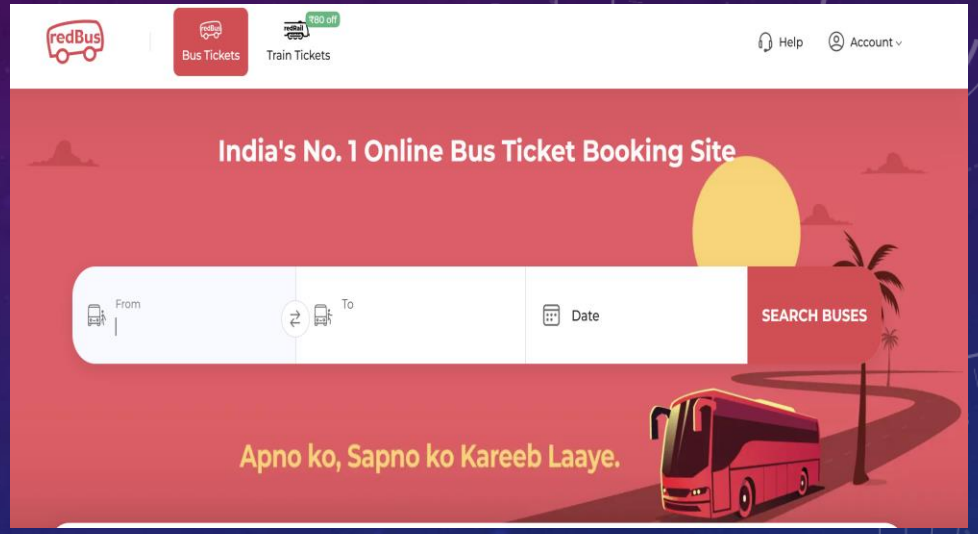




REDBUS DATA SCRAPING

WEST BENGAL

```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time
import pandas as pd
# URL of the website
URL = "https://www.redbus.in/online-booking/west-bengal-transport-corporation?utm_source=rtchometile"
# Set up the Chrome driver
driver = webdriver.Chrome()
driver.get(URL)
driver.maximize_window()
time.sleep(5)  # Wait for the page to load
# Function to scrape bus routes
def scrape_bus_routes():
route_elements = driver.find_elements(By.CLASS_NAME, 'route')
bus_routes_link = [route.get_attribute('href') for route in route_elements]
bus_routes_name = [route.text.strip() for route in route_elements]
return bus_routes_link, bus_routes_name
```



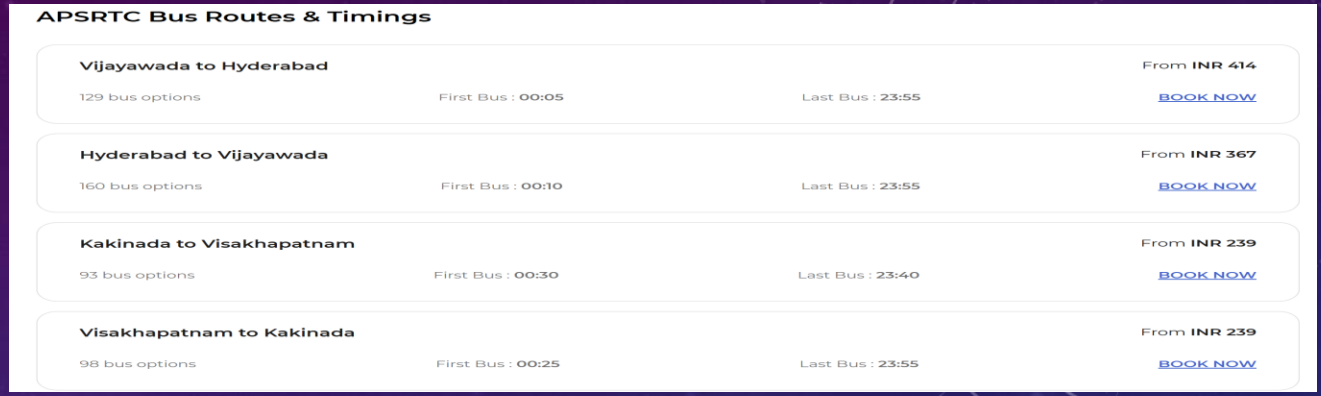
Continue……

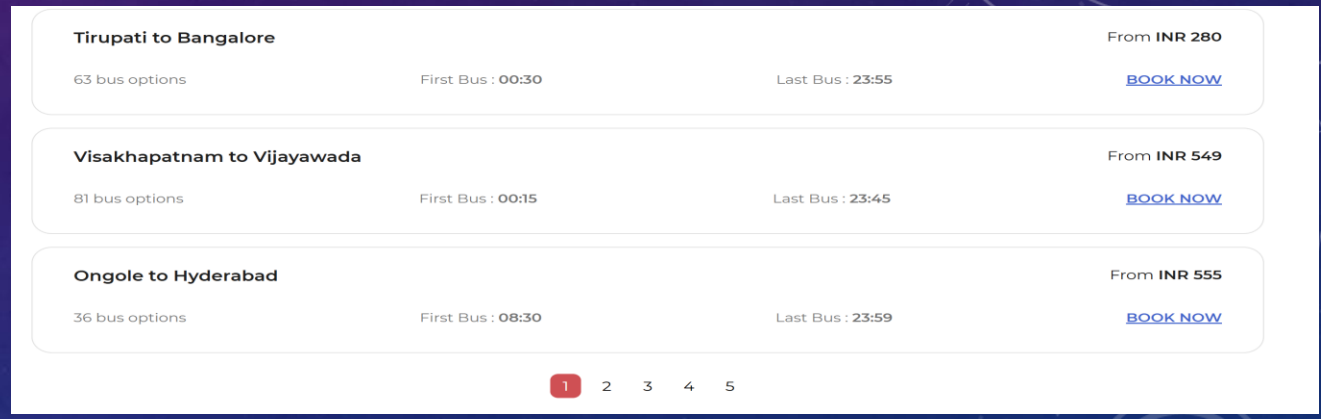
```
# Scrape the first page
all_bus_routes_link, all_bus_routes_name = scrape_bus_routes()
# Function to scrape bus details
def scrape_bus_details(url, route_name):
    try:
        driver.get(url)
        time.sleep(5)  # Allow the page to load
        # Scroll down to load all bus items
        last_height = driver.execute_script("return document.body.scrollHeight")
        while True:
            driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
            time.sleep(3)  # Wait for the page to load more content
            new_height = driver.execute_script("return document.body.scrollHeight")
            if new_height == last_height:
                break
            last_height = new_height
```

```
        # Find bus item details
        bus_name_elements = driver.find_elements(By.CLASS_NAME, "travels")
        bus_type_elements = driver.find_elements(By.CLASS_NAME, "bus-type")
        departing_time_elements = driver.find_elements(By.CLASS_NAME, "dp-time")
        duration_elements = driver.find_elements(By.CLASS_NAME, "dur")
        reaching_time_elements = driver.find_elements(By.CLASS_NAME, "bp-time")
        star_rating_elements = driver.find_elements(By.XPATH, "//div[@class='rating-sec lh-24']")
        price_elements = driver.find_elements(By.CLASS_NAME, "fare")
        seat_availability_elements = driver.find_elements(By.XPATH, "//div[contains(@class, 'seat-left m-top-
30') or contains(@class, 'seat-left m-top-16')]")
        bus_details = []
        for i in range(len(bus_name_elements)):
            bus_detail = {
                "Route_Name": route_name,
                "Route_Link": url,
                "Bus_Name": bus_name_elements[i].text,
                "Bus_Type": bus_type_elements[i].text,
                "Departing_Time": departing_time_elements[i].text,
                "Duration": duration_elements[i].text,
                "Reaching_Time": reaching_time_elements[i].text,
                "Star_Rating": star_rating_elements[i].text if i < len(star_rating_elements) else '0',
                "Price": price_elements[i].text,
                "Seat_Availability": seat_availability_elements[i].text if i < len(seat_availability_elements) else '0'}
```

Continue……

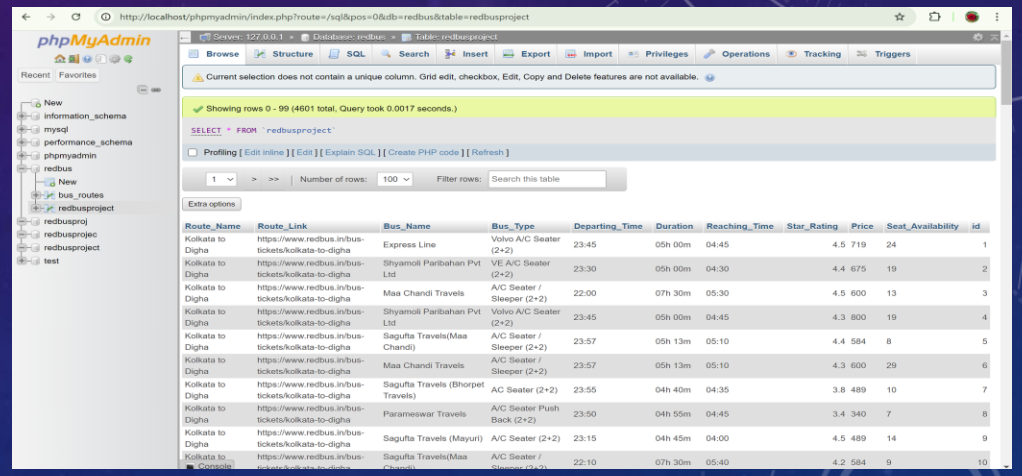
```
        bus_details.append(bus_detail)
        return bus_details

    except Exception as e:
        print(f"Error occurred while accessing {url}: {str(e)}")
        return []

# List to hold all bus details
all_bus_details = []

# Iterate over each bus route link and scrape the details
for link, name in zip(all_bus_routes_link, all_bus_routes_name):
    bus_details = scrape_bus_details(link, name)
    if bus_details:
        all_bus_details.extend(bus_details)

# Convert the list of dictionaries to a DataFrame
df = pd.DataFrame(all_bus_details)

# Save the DataFrame to a CSV file
df.to_csv('wb2_bus_details.csv', index=False)
# Close the driver
driver.quit()
```





MYSQL

```
➢  !pip install mysql-connector-python
import mysql.connector
mydb = mysql.connector.connect(
 host="localhost",
 user="root",
 password="",)
print(mydb)
mycursor = mydb.cursor(buffered=True)
➢ mycursor.execute("create database redbusprojec")
import pandas as pd
import mysql.connector  # Uncomment this when using MySQL
# List of CSV file paths
csv_files = [
    "wb2_bus_details.csv", "Telangana_bus_details.csv",
"rajasthan_bus_details.csv","ap_bus_details.csv","assam_bus_details.
csv", "chandigarh_bus_details.csv", "himachal_bus_details.csv",
"kerala_bus_details.csv", "kadamba_bus_details.csv",
"up_bus_details.csv"]
```



STREAMLIT

```
import streamlit as st
import pandas as pd
import mysql.connector
# Database connection
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="redbus"
)
# Create a cursor object
cursor = connection.cursor(buffered=True)
# Query to fetch the data
cursor.execute("SELECT * FROM REDBUSPROJECT")
out = cursor.fetchall()
# Convert the result to a pandas DataFrame
# Ensure to get column names
columns = [desc[0] for desc in cursor.description]
df = pd.DataFrame(out, columns=columns)
```

```
# Convert Price and Star_Rating columns to numeric types, forcing any errors
to NaN
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')
df['Star_Rating'] = pd.to_numeric(df['Star_Rating'], errors='coerce')
# Debugging: Print out the DataFrame columns and the first few rows
print("DataFrame columns:", df.columns)
print("DataFrame preview:", df.head())
# Streamlit application
st.set_page_config(page_title="RedBus Data Filtering",
page_icon="https://th.bing.com/th/id/OIP.6nU3XTAOJe8B07685FoXVQHaEK?w=305&h=18
0&c=7&r=0&o=5&dpr=2&pid=1.7", layout="wide")
# Add a background image
st.markdown( """  <style>.stApp { background-image:
url('https://miro.medium.com/v2/resize:fit:828/format:webp/1*S-
95TWd9jgxT87cKkZWnFg.jpeg');background-size: cover;} </style> """,
unsafe_allow_html=True)
st.title('RedBus Data Application')
```

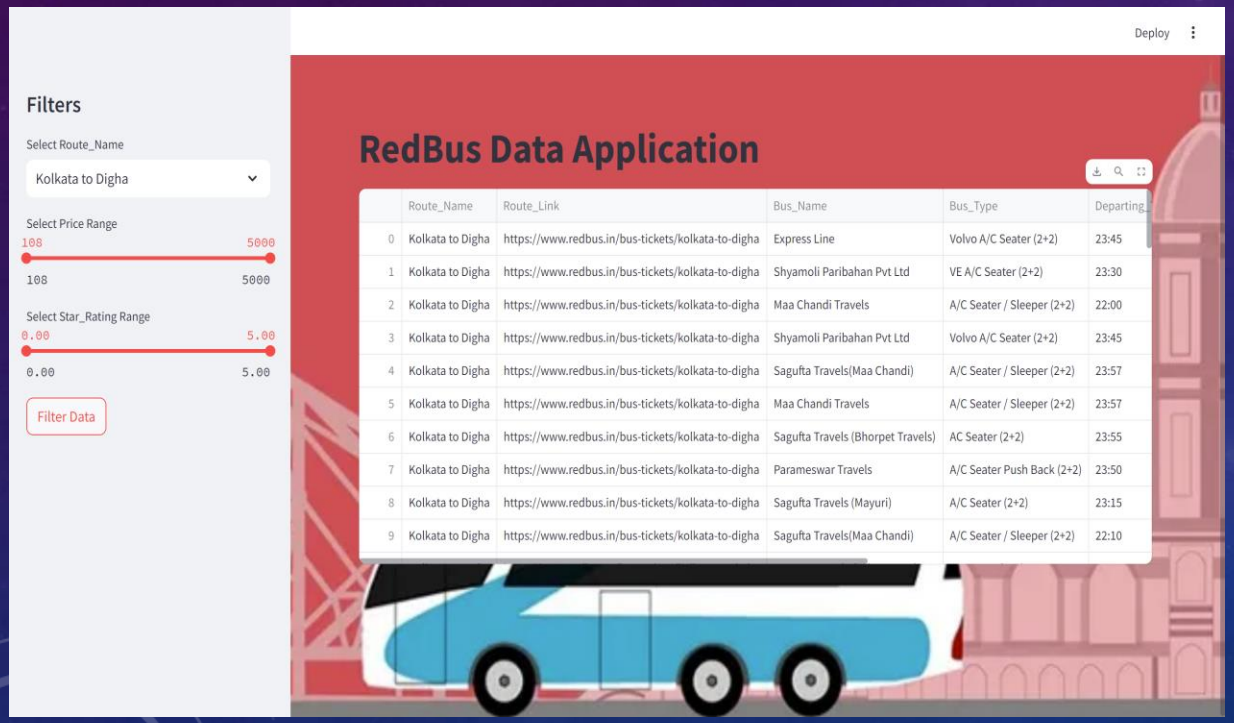
Continue……

```
# Sidebar filters
st.sidebar.title("Filters")
# Debugging: Check if 'Route_Name' is in the DataFrame columns
if 'Route_Name' in df.columns:
Route_Name_options = df["Route_Name"].unique()
selected_Route_Name = st.sidebar.selectbox("Select Route_Name",
Route_Name_options)
else:
  st.error("The column 'Route_Name' is not in the DataFrame")
# Ensure that the DataFrame has the 'Price' column before using it
if 'Price' in df.columns:
    price_min = int(df["Price"].min())
    price_max = int(df["Price"].max())
    selected_price = st.sidebar.slider("Select Price Range", price_min, price_max,
(price_min, price_max))
else:
st.error("The column 'Price' is not in the DataFrame")
```

```
# Ensure that the DataFrame has the 'Star_Rating' column before using it
if 'Star_Rating' in df.columns:
    Star_Rating_min = float(df["Star_Rating"].min())
    Star_Rating_max = float(df["Star_Rating"].max())
    selected_Star_Rating = st.sidebar.slider("Select Star_Rating Range", Star_Rating_min, Star_Rating_max,
(Star_Rating_min, Star_Rating_max))
else:
    st.error("The column 'Star_Rating' is not in the DataFrame")
# Filter button
if st.sidebar.button("Filter Data"):
    if 'Route_Name' in df.columns and 'Price' in df.columns and 'Star_Rating' in df.columns:
        filtered_df = df[
            (df["Route_Name"] == selected_Route_Name) &
            (df["Price"] >= selected_price[0]) &
            (df["Price"] <= selected_price[1]) &
            (df["Star_Rating"] >= selected_Star_Rating[0]) &
            (df["Star_Rating"] <= selected_Star_Rating[1])
        ]
        st.write(filtered_df)
else:
    st.write("Apply filters to see the data.")
```

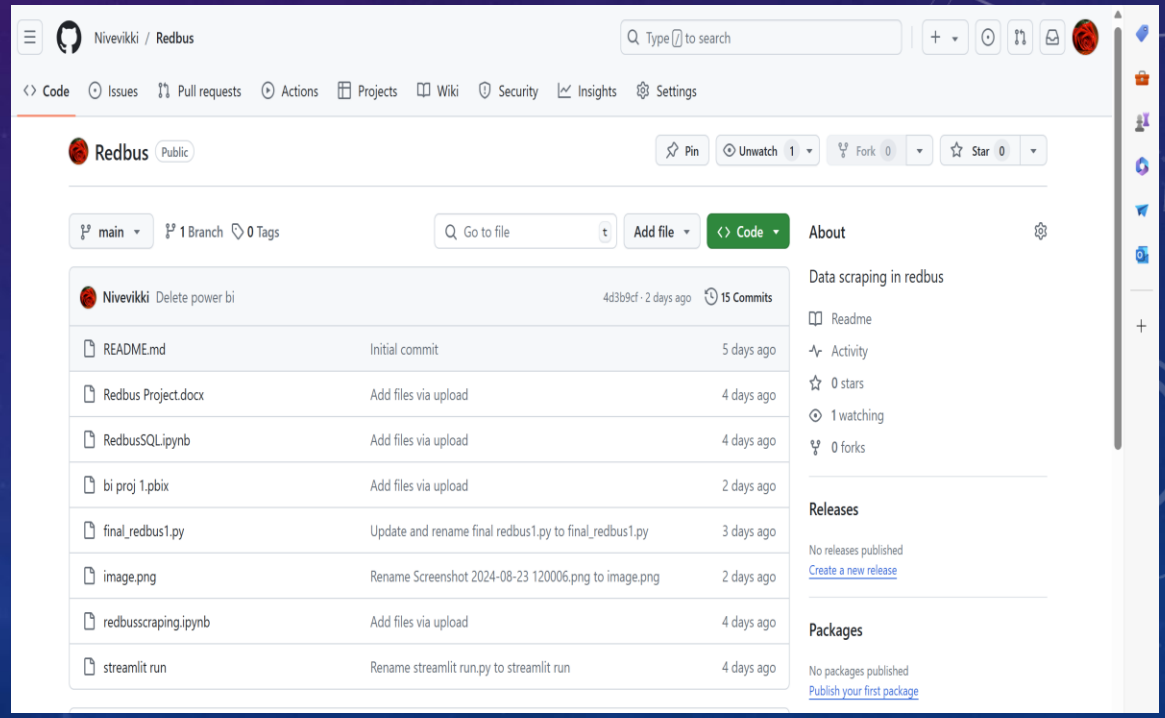
Continue……

STREAMLIT OUTPUT



GITHUB

- Create Github
- Paste the file in the repository




Thank
you