1. Source Code

a. Data Scraping Script (scrape_bus_data.py)

This script scrapes bus data from the Redbus website and stores it in a MySQL database.

python

```
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 TimeoutException, NoSuchElementException
import time
import mysql.connector
import re

# MySQL database connection parameters
db_config = {
    'user': 'your_username',
    'password': 'your_password',
    'host': 'localhost',
    'database': 'your_database'
}

# Function to insert data into MySQL database
def insert_bus_data(bus_data):
    try:
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor()
        insert_query = """
        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)
```

```
        """

        for data in bus_data:
            cursor.execute(insert_query, data)

        conn.commit()
        print("Data inserted successfully.")

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        cursor.close()
        conn.close()

# Initialize WebDriver
driver = webdriver.Chrome()

# Initialize bus_data
bus_data = []

try:
    driver.get("https://www.redbus.in/")
    wait = WebDriverWait(driver, 20)

    src_input = wait.until(EC.element_to_be_clickable((By.ID, "src")))
    src_input.send_keys("Bangalore")

    dest_input = wait.until(EC.element_to_be_clickable((By.ID, "dest")))
    dest_input.send_keys("Chennai")
```

```
    date_picker_button = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.ID,
'onwardCal')))
    date_picker_button.click()

    search_button = wait.until(EC.element_to_be_clickable((By.ID, "search_button")))
    search_button.click()

    def scroll_to_bottom(driver):
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")

    i = 0
    while i < 12:
        try:
            wait.until(EC.presence_of_element_located((By.CLASS_NAME, "bus-items")))
            bus_items = driver.find_elements(By.XPATH, '//ul[@class="bus-items"]/div/li')
            for bus_item in bus_items:
                try:
                    bus_name = bus_item.find_element(By.CLASS_NAME, "travels").text
                    bus_type = bus_item.find_element(By.CLASS_NAME, "bus-type").text
                    departure = bus_item.find_element(By.CLASS_NAME, "dp-time").text
                    arrival = bus_item.find_element(By.CLASS_NAME, "bp-time").text
                    price_text = bus_item.find_element(By.CLASS_NAME, "fare").text
                    available_seats = bus_item.find_element(By.CLASS_NAME, "seat-left").text
                    rating_text = bus_item.find_element(By.CLASS_NAME, "rating").text

                    price = re.sub(r'[^\d.]', '', price_text)
                    price = float(price) if price else 0.0

                    rating = re.sub(r'[^\d.]', '', rating_text)
                    star_rating = float(rating) if rating else 0.0
```

```
                available_seats = int(re.sub(r'[^\d]', '', available_seats)) if available_seats else 0

                route_link = "N/A"
                duration = "N/A"
                departing_time = departure
                reaching_time = arrival

                bus_data.append((route_link, route_link, bus_name, bus_type, departing_time, duration, reaching_time, star_rating, price, available_seats))
                if len(bus_data) > 20:
                    break
            except NoSuchElementException as e:
                print(f"Error finding element in bus item: {e}")

        scroll_to_bottom(driver)
        time.sleep(1)
        i += 1
    except Exception as e:
        print(f"An error occurred: {e}")

finally:
    driver.quit()
    print("WebDriver closed.")

if bus_data:
    insert_bus_data(bus_data)
else:
    print("No bus data was scraped.")
```

b. Streamlit Application (app.py)

This script creates a Streamlit application for data filtering and visualization.

python

```
import streamlit as st
import mysql.connector
import pandas as pd

# Connect to MySQL
conn = mysql.connector.connect(
    host="localhost",
    user="your_username",
    password="your_password",
    database="your_database"
)

# Query data from MySQL
def get_data(query):
    cursor = conn.cursor()
    cursor.execute(query)
    rows = cursor.fetchall()
    columns = [i[0] for i in cursor.description]
    cursor.close()
    return pd.DataFrame(rows, columns=columns)

# Filter options
st.title("Redbus Schedule")

bustype = st.selectbox("Select Bus Type", ["All", "AC", "Non-AC"])
price_range = st.slider("Price Range", 0, 5000, (100, 3000))
route = st.text_input("Enter Route")
```

```
query = "SELECT * FROM bus_routes WHERE 1=1"

if bustype != "All":
    query += f" AND bustype = '{bustype}'"

if route:
    query += f" AND (busname LIKE '%{route}%' OR bustype LIKE '%{route}%')"

query += f" AND price BETWEEN {price_range[0]} AND {price_range[1]}"

df = get_data(query)
st.dataframe(df)

conn.close()
```

2. Documentation

a. Overview

- **Data Scraping**: The scrape_bus_data.py script uses Selenium to scrape bus data from the Redbus website. It extracts details like bus name, type, departure and arrival times, price, rating, and available seats, and then stores this data in a MySQL database.
- **Streamlit Application**: The app.py script creates a web application using Streamlit. It allows users to filter bus data based on bus type, price range, and route, and displays the filtered results in a table.

b. Running the Scripts

1. **Data Scraping**:
    - Install necessary libraries: selenium, mysql-connector-python, and pandas.
    - Download and set up ChromeDriver.
    - Update db_config in scrape_bus_data.py with your MySQL credentials.
    - Run the script:

bash

```
python scrape_bus_data.py
```

2. **Streamlit Application**:
    - Install Streamlit and required libraries: streamlit, mysql-connector-python, and pandas.
    - Update MySQL credentials in app.py.
    - Run the Streamlit app:

bash

```
streamlit run app.py
```

c. Data Collection

- **Data Source**: Redbus website.
- **Data Collected**: Bus name, type, departure time, arrival time, price, available seats, and star rating.

3. Database Schema

a. SQL Script to Create the Database and Table

sql

```
-- Create database
CREATE DATABASE redbus_db;
-- Use database
USE DATABASE redbus_db;

-- Use the database
USE redbus_db;

-- Create table
CREATE TABLE bus_routes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    route_name TEXT,
    route_link TEXT,
    busname TEXT,
    bustype TEXT,
    departing_time TIME,
```

duration TEXT,

reaching_time TIME,

star_rating FLOAT,

price DECIMAL(10, 2),

seats_available INT

);

b. SQL Script to Populate the Database

This step is not applicable directly as data is inserted via the scraping script. However, if needed, you could insert sample data manually:

sql

INSERT INTO bus_routes (route_name, route_link, busname, bustype, departing_time, duration, reaching_time, star_rating, price, seats_available)

VALUES

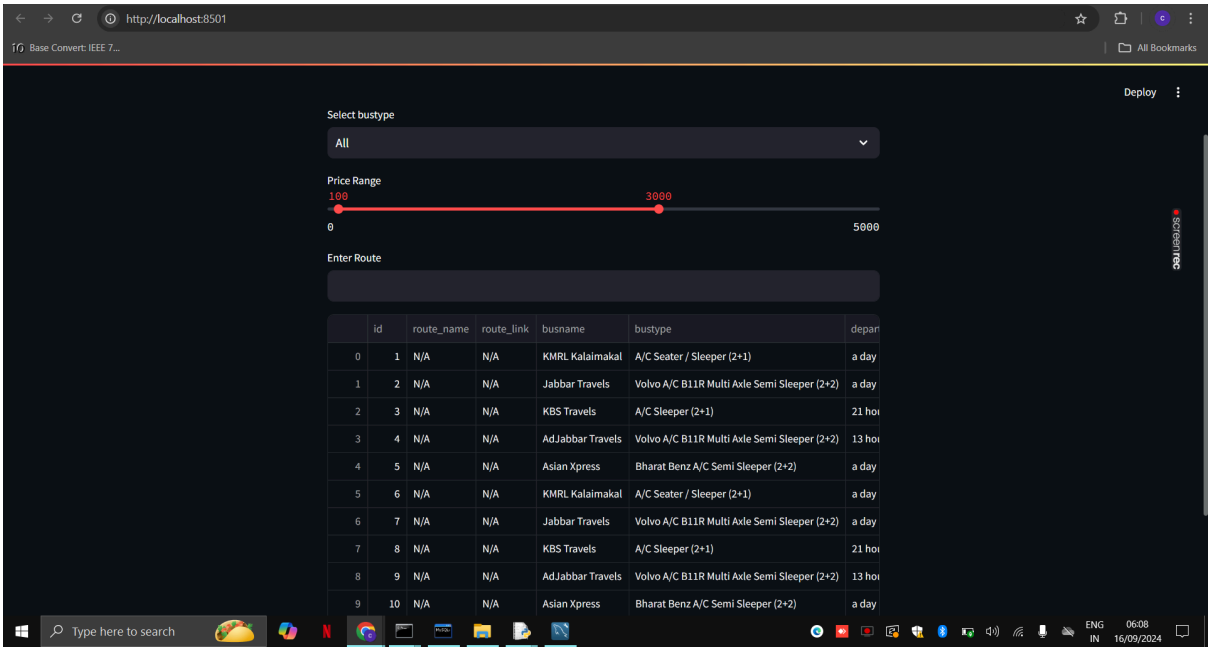
('Route 1', 'link1', 'Bus A', 'AC', '10:00:00', '5h', '15:00:00', 4.5, 1500.00, 20),

('Route 2', 'link2', 'Bus B', 'Non-AC', '12:00:00', '4h', '16:00:00', 3.5, 800.00, 30);

4. Application Using Streamlit

a. Screenshots

- **Screenshot 1**: Streamlit Application Home Page



- **Screenshot 2**: Filtered Results (e.g., filtering by AC buses and a price range)