Redbus Project 1

By. Amey Patil – MDE92

1. Introduction

The "Redbus Data Scraping and Dynamic Filtering with Streamlit Application" project aims to modernize and streamline the collection, analysis, and visualization of bus travel data in the transportation industry. The transportation sector is a critical component of any economy, and access to accurate, up-to-date data is essential for various stakeholders, including travel aggregators, market analysts, and customer service teams.

The motivation behind this project is to address the challenges associated with manual data collection and analysis in the transportation industry. Traditionally, gathering bus travel data, such as routes, schedules, prices, and seat availability, has been a time-consuming and error-prone process. By leveraging modern web scraping techniques, this project automates the extraction of critical information from Redbus, a popular online bus ticketing platform, thereby enhancing efficiency and accuracy.

Objectives

The primary objectives of this project are:

1. **Automate Data Extraction**: Use Selenium, a powerful web automation tool, to scrape bus travel data from Redbus, including bus routes, schedules, prices, and seat availability.
2. **Store Data Efficiently**: Store the scraped data in a structured SQL database to ensure efficient data management and retrieval.
3. **Develop an Interactive Application**: Create an interactive application using Streamlit, an open-source framework, to allow users to filter and explore the bus travel data dynamically.
4. **Provide Valuable Insights**: Enable various stakeholders to gain valuable insights from the collected data, facilitating data-driven decision-making in the transportation industry.

Benefits

By automating the data collection process and providing an interactive platform for data analysis, this project offers several benefits:

- **Travel Aggregators**: Can access real-time schedules and seat availability, allowing them to offer more accurate and timely information to customers.

- **Market Analysts**: Can study travel patterns and trends, aiding in market research and strategic planning.
- **Customer Service Teams**: Can offer customized travel options based on real-time data, improving customer satisfaction and loyalty.
- **Competitor Analysis**: The application supports competitor analysis by comparing pricing and service levels, enabling companies to stay competitive in the market.

Project Overview

This project showcases the integration of modern web scraping techniques, data storage solutions, and interactive data visualization tools, demonstrating a comprehensive approach to data-driven decision-making in the transportation industry. The methodology involves several key steps, each contributing to the overall goal of transforming how bus travel data is collected, stored, and analyzed.

By following this methodology, the project successfully automates the data collection from Redbus, ensuring that the information is accurate, up-to-date, and easily accessible. The data is then stored in a structured SQL database, allowing for efficient management and retrieval. The Streamlit application provides an intuitive interface for users to explore and filter the data dynamically, delivering valuable insights that can be used to improve services, analyze market trends, and enhance customer experiences.

In summary, the "Redbus Data Scraping and Dynamic Filtering with Streamlit Application" project exemplifies how modern technologies can be leveraged to streamline processes and enhance decision-making in the transportation industry. Through the integration of web scraping, data storage, and interactive visualization, this project sets a new standard for how bus travel data is collected, analyzed, and utilized.

2. Methodology

The methodology for the "Redbus Data Scraping and Dynamic Filtering with Streamlit Application" project involves several key steps: data scraping, data storage, application development, and dynamic filtering. Here is a detailed breakdown of each step:

1. Data Scraping

Tools Used:

- **Selenium**: Selenium is a powerful web automation tool that is commonly used for web scraping. It allows users to programmatically control a web browser, enabling the extraction of data from web pages.

Steps:

1. **Initialize WebDriver**:
   - Create a Selenium WebDriver instance to interact with the Redbus website. This involves specifying the browser driver (e.g., ChromeDriver) and setting up any necessary options.

```
python
Copy code
from selenium import webdriver
driver = webdriver.Chrome()
```

2. **Navigate to Redbus**:
   - Use the WebDriver to navigate to the Redbus URL. This is done using the get method, which loads the specified web page.

```
python
Copy code
driver.get('https://www.redbus.in/online-booking/rtc-directory')
```

3. **Fetch Government Bus Links**:
   - Locate and collect URLs of government bus services listed on Redbus. This is achieved by finding elements using XPath and extracting their href attributes.

```
python
Copy code
gov_links = driver.find_elements(By.XPATH, "//li[@class='D113_item_rtc']/a")
gov_link_list = [link.get_attribute('href') for link in gov_links]
```

4. **Iterate Through Links**:
   - For each link, navigate to the page and handle pagination to ensure all available data is collected. This involves clicking through pagination buttons and scraping data from each page.

```
python
Copy code
for link in gov_link_list:
   driver.get(link)
```

```
    time.sleep(3)
    # (Additional scraping logic here)
```

5. **Extract Data**:
   - Scrape specific data points such as route names, bus names, bus types, departure times, durations, arrival times, star ratings, prices, and seat availability. This involves locating elements using XPath and extracting their text content.

```
python
Copy code
buses = driver.find_elements(By.XPATH, '//div[@class="clearfix bus-item"]')
for bus in buses:
    bus_name = bus.find_element(By.XPATH, './/div[@class="travels lh-24 f-bold d-color"]').text
    # (Extract other data points similarly)
```

6. **Store Data Temporarily**:
   - Store the scraped data in a pandas DataFrame for further processing and eventual storage in a SQL database.

```
python
Copy code
df = pd.DataFrame(columns=['route_name', 'route_link', 'route_start', 'route_end', 'bus_name', 'bustype', 'departing_time', 'duration', 'reaching_time', 'star_rating', 'price', 'seats_available'])
df = df.append({'route_name': route_name, 'route_link': route_link, 'bus_name': bus_name, 'bustype': bustype, 'departing_time': departing_time, 'duration': duration, 'reaching_time': reaching_time, 'star_rating': star_rating, 'price': price, 'seats_available': seats_available}, ignore_index=True)
```

References:

- Selenium Documentation
- Web Scraping with Selenium and Python

2. Data Storage

Tools Used:

- **MySQL**: MySQL is a popular relational database management system used for storing and managing structured data.

Steps:

1. **Create Database and Table**:
   - Use SQL scripts to create a database and a table with the required schema. This involves specifying the database name and defining the table structure with appropriate data types.

```
python
Copy code
import mysql.connector
con = mysql.connector.connect(host="localhost", user="root", password="your_password")
cursor = con.cursor()
cursor.execute("CREATE DATABASE IF NOT EXISTS Redbus")
cursor.execute("USE Redbus")
cursor.execute("""CREATE TABLE IF NOT EXISTS bus_routes (
                    route_name VARCHAR(200),
                    route_link VARCHAR(500),
                    route_start VARCHAR(200),
                    route_end VARCHAR(200),
                    bus_name VARCHAR(200),
                    bustype VARCHAR(200),
                    departing_time TIME,
                    duration VARCHAR(200),
                    reaching_time TIME,
                    star_rating DECIMAL(2, 1),
                    price DECIMAL(10, 2),
                    seats_available INT
                )""")
```

2. **Clean and Prepare Data**:
   - Clean and format the scraped data to match the SQL table schema. This may involve handling missing values, converting data types, and performing any necessary transformations.

```
python
Copy code
df = pd.read_excel("Redbus_data.xlsx")
df = df.replace({np.nan: None})
df['seats_available'] = df['seats_available'].str.split(' ', expand=True)[0]
df['bustype'] = df['bustype'].apply(lambda x: 'Non A/C' if 'Non' in x else 'A/C')
```

3. **Insert Data**:
   - Insert the cleaned data into the SQL table using SQL queries. This involves iterating through the DataFrame and executing insert statements for each row.

```
python
Copy code
query = "INSERT INTO bus_routes VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
for index, row in df.iterrows():
    cursor.execute(query, tuple(row))
    con.commit()
```

References:

1. MySQL Documentation
2. Python MySQL Connector

3. Application Development

Tools Used:

- **Streamlit**: Streamlit is an open-source framework for building interactive web applications with Python. It allows for quick development of data-driven apps with minimal effort.

Steps:

1. **Set Up Streamlit App**:
   - Initialize a Streamlit application and configure the layout. This involves importing the Streamlit library and setting the page configuration.

```
python
Copy code
import streamlit as st
st.set_page_config(layout="wide")
```

2. **Connect to Database**:
   - Establish a connection to the MySQL database to fetch the stored data. This is done using a function that returns a database connection object.

```
python
Copy code
def get_connection():
    connection = mysql.connector.connect(
        host='localhost',
        user='root',
        password='your_password',
        database='Redbus'
    )
    return connection
```

3. **Load Data**:
   - Load the data into a pandas DataFrame for easy manipulation and filtering. This involves executing a SQL query to retrieve the data and storing it in a DataFrame.

```
python
Copy code
def load_data():
    conn = get_connection()
    query = "SELECT * FROM bus_routes"
    df = pd.read_sql(query, conn)
    conn.close()
    return df
data = load_data()
```

4. **Create Filters**:
   - Implement various filters (e.g., route name, bus type, price range, star rating) using Streamlit widgets. This allows users to select criteria for filtering the data.

```
python
Copy code
route_names = ['All'] + list(data['route_name'].unique())
selected_route_name = st.sidebar.selectbox("Route Name", route_names)
```

5. **Display Data**:
   - Display the filtered data dynamically based on user input. This involves updating the displayed DataFrame as the user adjusts the filters.

```
python
Copy code
filtered_data = data[data['route_name'] == selected_route_name] if selected_route_name != 'All' else
data
st.dataframe(filtered_data)
```

References:

- Streamlit Documentation
- Building Streamlit Apps

4. Dynamic Filtering

Steps:

1. **User Input**:
   - Users interact with the Streamlit app, selecting filter criteria from the sidebar. Streamlit widgets are used to capture user inputs for various filters.

```
python
Copy code
selected_bustype = st.sidebar.selectbox("Bus Type", bustypes)
```

2. **Data Filtering**:
   - The application dynamically filters the data based on the selected criteria using pandas. This involves applying conditional statements to filter the DataFrame.

```
python
Copy code
filtered_data = data[(data['bustype'] == selected_bustype if selected_bustype != 'All' else True) &
                     (data['price'] >= price_range[0]) &
                     (data['price'] <= price_range[1])]
```

3. **Real-time Display**:
   - The filtered data is displayed in real-time, allowing users to explore and analyze the data interactively. The st.dataframe method is used to render the DataFrame in the Streamlit app.

```
python
Copy code
st.dataframe(filtered_data)
```

References:

- Pandas Documentation
- Interactive Data Filtering with Streamlit

4. Summary

The "Redbus Data Scraping and Dynamic Filtering with Streamlit Application" project exemplifies the transformative power of modern technologies in the transportation industry. This project leverages web scraping, data storage, and interactive data visualization to streamline the collection, analysis, and utilization of bus travel data, thereby addressing several key challenges and providing numerous benefits.