

Redbus Data Scrapping with Selenium & Dynamic Filtering using Streamlit

This documentation outlines the process for scraping bus data from the Redbus website, storing it in a MySQL database, and creating a Streamlit application for data filtering and analysis. The project involves the following key components:

1. **Web Scrapping using Selenium:** Automating the browser to scrape data from the Redbus website.
2. **Database Interaction:** Using MySQL to store the scraped data.
3. **Streamlit Application:** Creating an interactive application to filter and analyse the data.

Prerequisites

- Python 3.x
- Selenium
- MySQL
- Streamlit
- WebDriver for the browser (Chrome Driver for Google Chrome)

Setup Instructions

1. Install Dependencies

Installed the required Python packages using pip:

```
pip install selenium mysql-connector-python streamlit
```

2. MySQL Database Setup

Created a MySQL database named Redbus and a table named bus_routes using the following SQL script:

```
query="create database if not exists Redbus"  
cursor.execute(query)
```

```
query="use Redbus"  
cursor.execute(query)
```

```
query="""create table if not exists bus_routes(  
            id INT PRIMARY KEY AUTO_INCREMENT,  
            route_name TEXT,  
            route_link TEXT,  
            busname TEXT,  
            bustype TEXT,  
            departing_time TEXT,  
            duration TEXT,  
            reaching_time TEXT,  
            star_rating FLOAT,  
            price DECIMAL,  
            seats_available INT  
        )"""  
cursor.execute(query)
```

Code Explanation

Database Connection

The **get_database_connection** function establishes a connection to the MySQL database.

```
# Connect to the MySQL database
def get_database_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="12345678",
        database="Redbus"
    )
```

Creating the Table

The **create_table** function creates the bus_routes table if it does not already exist.

```
# Create the table if it doesn't already exist
def create_table(cursor):
    query = """
    CREATE TABLE IF NOT EXISTS bus_routes (
        id INT PRIMARY KEY AUTO_INCREMENT,
        route_name TEXT,
        route_link TEXT,
        busname TEXT,
        bustype TEXT,
        departing_time TEXT,
        duration TEXT,
        reaching_time TEXT,
        star_rating FLOAT,
        price DECIMAL,
        seats_available TEXT
    )
    """
    cursor.execute(query)
```

Initializing the WebDriver

The `get_driver` function initializes and returns a Selenium WebDriver instance.

```
#Initializes the WebDriver  
def get_driver():  
    driver = webdriver.Chrome()  
    driver.maximize_window()  
    return driver
```

Scraping Bus Routes and Links

The `scrape_current_page` function scrolls through the current page to load all bus routes and extracts the route names and links.

```
#Scrapes the current page for bus routes and links  
def scrape_current_page(driver, bus_list, max_scroll_attempts=3):  
    scrolling = True  
    scroll_attempts = 0  
  
    while scrolling and scroll_attempts < max_scroll_attempts:  
        old_page_source = driver.page_source  
        body = driver.find_element(By.TAG_NAME, "body")  
        body.send_keys(Keys.PAGE_DOWN)  
  
        time.sleep(2)  
        new_page_source = driver.page_source  
  
        if new_page_source == old_page_source:  
            scrolling = False  
            scroll_attempts += 1  
  
    WebDriverWait(driver, 10).until(  
        EC.presence_of_all_elements_located(  
            (By.XPATH, "//div[@class='D117_main D117_container']/div[@class='route_link']")  
        )  
    )  
    for bus in driver.find_elements(  
        By.XPATH, "//div[@class='D117_main D117_container']/div[@class='route_link']"):  
        route_name = bus.find_element(By.CSS_SELECTOR, "a.route").text  
        route_link = bus.find_element(By.CSS_SELECTOR, "a.route").get_attribute('href')  
        bus_list.append({  
            'route': route_name,  
            'link': route_link  
        })
```

Clicking the Next Page Button

The `click_next_page` function navigates to the next page of bus routes.

```
#Clicks the next page button to navigate to the next page
def click_next_page(driver, current_page):

    try:
        next_page = driver.find_element(
            By.XPATH, f"//div[@class='DC_117_pageTabs ' and text()='{{current_page + 1}}']")
        driver.execute_script("arguments[0].scrollIntoView(true);", next_page) # Scroll to the element
        time.sleep(1) # Wait for the scroll to complete
        driver.execute_script("arguments[0].click();", next_page) # Click using JavaScript
        time.sleep(2)
    except Exception as e:
        print(f"Error clicking next page: {e}")
```

Scraping Bus Details

The `scrape_bus_details` function extracts detailed information about each bus on the route.

```
#Scrapes the bus details for each route in the bus list
def scrape_bus_details(driver, cursor, bus_list):

    for bus in bus_list:
        try:
            driver.get(bus['link'])

            WebDriverWait(driver, 20).until(
                EC.presence_of_element_located((By.XPATH, "//div[@class='clearfix bus-item']")))
        )

        for bus_detail in driver.find_elements(
            By.XPATH, "//div[@class='clearfix bus-item']/div[@class='clearfix bus-item-details']/div[@class='clearfix row-one']"):
            try:
                busname = bus_detail.find_element(
                    By.XPATH, ".*div[@class='travels lh-24 f-bold d-color']").text
                bustype = bus_detail.find_element(
                    By.XPATH, ".*div[@class='bus-type f-12 m-top-16 l-color evBus']").text
                departing_time = bus_detail.find_element(
                    By.XPATH, ".*div[@class='dp-time f-19 d-color f-bold']").text
                duration = bus_detail.find_element(
                    By.XPATH, ".*div[@class='dur l-color lh-24']").text
                reaching_time = bus_detail.find_element(
                    By.XPATH, ".*div[@class='bp-time f-19 d-color disp-Inline']").text

                star_rating_str = bus_detail.find_element(
                    By.XPATH, ".*div[@class='rating-sec lh-24']").text
                star_rating = float(star_rating_str.split()[0]) if star_rating_str else 0.0

                price_str = bus_detail.find_element(By.XPATH, ".*div[@class='fare d-block']").text
                price = int(price_str.replace('INR', '').replace(', ', '').strip()) if price_str else 0

                try:
                    seat_availability_str = bus_detail.find_element(
                        By.XPATH, ".*div[@class='seat-left m-top-30']").text
                    seats_available = int(seat_availability_str.split()[0]) if seat_availability_str else 0
                except NoSuchElementException:
                    seats_available = 0

                cursor.execute("""
                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)""",
                    (bus['route'], bus['link'], busname, bustype, departing_time,
                     duration, reaching_time, star_rating, price, seats_available))

            except NoSuchElementException as e:
                print(f"Error scraping bus details: {e}")
            except ValueError as ve:
                print(f"Error converting data: {ve}")

    except Exception as e:
        print(f"Error scraping bus details: {e}")
```


Main Function

The main function manages the entire process, from initializing the database and WebDriver to scraping data and inserting it into the database.

#Main function to execute the scraping and database insertion process

```
def main():

    con = get_database_connection()
    cursor = con.cursor()
    create_table(cursor)

    driver = get_driver()

    urls = [
        'https://www.redbus.in/online-booking/apsrtc/?utm_source=rtchometile',
        'https://www.redbus.in/online-booking/ksrtc-kerala/?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-sbstrc/?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',
    ]

    for url in urls:
        driver.get(url)

        WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.XPATH, "//div[@class='D117_main D117_container']"))
        )

        bus_list = []
        current_page = 1
        total_pages = len(driver.find_elements(By.XPATH, "//div[@class='DC_117_pageTabs '])) + 1

        while current_page <= total_pages:
            scrape_current_page(driver, bus_list)
            if current_page < total_pages:
                click_next_page(driver, current_page)
                current_page += 1

            scrape_bus_details(driver, cursor, bus_list)
            con.commit()

        cursor.close()
        con.close()

        driver.quit()
        print("Data scraped and inserted into MySQL successfully.")

if __name__ == "__main__":
    main()
```

Technical Tags:

- Web Scraping
- Selenium
- Streamlit
- SQL
- Data Analysis
- Python
- Interactive Application

Running the Script

To run the script, executed the following command terminal:

```
python scrap.py
```

Streamlit Application

Created a streamlit.py file for the Streamlit application:

Data Analysis/Filtering using Streamlit:

Used SQL queries to retrieved and filtered data based on user inputs.

Used Streamlit to allow users to interact with and filter the data through the application.

Streamlit Application:

The Streamlit application to display and filter the scraped data.

Implemented various filters such as bustype, route, price range, star rating, availability.


```

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

# Function to connect to the MySQL database
def get_database_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="12345678",
        database="Redbus"
    )

# Function to retrieve data from the database with dynamic filters
def get_filtered_data(bustype, route, price_range, star_rating, availability):
    con = get_database_connection()
    cursor = con.cursor(dictionary=True)

    query = "SELECT * FROM bus_routes WHERE 1=1"
    params = []

    # Apply filters dynamically
    if bustype:
        query += " AND bustype LIKE %s"
        params.append(f"%{bustype}%")
    if route:
        query += " AND route_name LIKE %s"
        params.append(f"%{route}%")
    if price_range:
        query += " AND price BETWEEN %s AND %s"
        params.extend(price_range)

```

```

    if star_rating:
        query += " AND star_rating <= %s"
        params.append(star_rating)
    if availability:
        query += " AND seats_available > 0"

    cursor.execute(query, params)
    result = cursor.fetchall()
    con.close()
    return result

# Streamlit application layout
st.title("Redbus Data Filtering and Analysis")

# User inputs for filters
bustype = st.selectbox("Select Bus Type", options=["", "A/C Seater", "A/C Sleeper", "Non A/C Seater", "Non A/C Sleeper"])
route = st.text_input("Enter Route")
price_range = st.slider("Select Price Range", 0, 5000, (0, 5000))
star_rating = st.slider("Select Minimum Star Rating", 0.0, 5.0, 0.0)
availability = st.checkbox("Show only buses with available seats")

# Fetch and display filtered data
if st.button("Apply Filters"):
    data = get_filtered_data(bustype, route, price_range, star_rating, availability)
    if data:
        df = pd.DataFrame(data)
        st.write(f"Showing {len(data)} results:")
        st.dataframe(df)
    else:
        st.write("No results found.")

# Display all data initially
else:
    data = get_filtered_data(None, None, (0, 5000), 0.0, False)
    df = pd.DataFrame(data)
    st.write(f"Showing all {len(data)} results:")
    st.dataframe(df)

```

Database Schema:

Table Name: bus_routes

Primary Key: To ensure each record is unique, an auto-incrementing primary key (id) is used.

Column Name	Data Type	Description
id	INT	Primary Key (Auto-increment)
route_name	TEXT	Bus Route information for each state transport
route_link	TEXT	Link to the route details
busname	TEXT	Name of the bus
bustype	TEXT	Type of the bus
departing_time	TEXT	Departure time
duration	TEXT	Duration of the journey
reaching_time	TEXT	Arrival time
star_rating	FLOAT	Rating of the bus
price	DECIMAL	Price of the ticket
seats_available	INT	Number of seats available

GitHub Link:

<https://github.com/vishnupriya08-hub/redbus-data-scraping>