**Data-Scraping-with-Selenium-Dynamic-Filtering-using-Streamlit**

**Introduction**

Web scraping is the process of automatically extracting data from websites. In this context, we aim to scrape data from RedBus, a popular online bus ticket booking platform, to extract useful information such as bus routes, timings, prices, and seat availability.

One of the powerful tools to achieve web scraping is Selenium, a Python library that automates browser actions. Selenium is commonly used for automating tasks such as testing web applications and scraping dynamic web content that loads using JavaScript.

**Why Use Selenium for RedBus Scraping?**

RedBus, like many modern websites, dynamically loads content using JavaScript. Traditional scraping libraries like requests and BeautifulSoup may not work efficiently with such dynamic content. This is where Selenium excels—it can:

* Simulate human interactions: Clicking buttons, navigating through multiple pages, and filling out forms.
* Handle JavaScript-rendered content: Load and scrape data from websites that generate content dynamically.

**Domain**

* TRANSPORTATION

**SKILL-TAKEAWAY**

* Python scripting,Selenium,Data Collection,Data Management using SQL,Streamlit

**Steps to Scrape RedBus Data Using Selenium**

* Set Up Selenium:
* Install the Selenium package via pip install selenium.
* Download the appropriate WebDriver (e.g., ChromeDriver for Chrome) to interact with the browser.
* Identify Data to Scrape:
* The RedBus website provides useful information such as bus names, routes, bus types, departure and arrival times, seat availability, and ticket prices.
* These data points are typically loaded dynamically using JavaScript, making Selenium an ideal choice.
* Navigate the RedBus Website:
* We can automate actions like opening the website, searching for buses between cities, and navigating through multiple pages of results.
* Each result contains individual bus information that can be extracted.
* Extract the Data:
* After navigating the site, use Selenium to locate the HTML elements containing the desired information (e.g., bus name, price, etc.) and extract their text.
* Handle Pagination:
* If there are multiple pages of results, Selenium can be used to click through the pagination and scrape data from each page.
* Example Workflow of Scraping RedBus Data Using Selenium
* Below is an outline of the steps to extract bus details from the RedBus website using Selenium:
* Initialize Selenium and Open the RedBus Website:
* Set up a Selenium WebDriver to automate a browser (e.g., Chrome) and open the RedBus homepage.
* Search for Buses:
* Simulate entering a source and destination city and selecting a travel date.
* Extract Bus Details:
* Once the search results are loaded, extract details such as:
* Bus operator name
* Bus type (e.g., AC, Non-AC, Sleeper)
* Departure and arrival times
* Ticket price
* Seat availability
* Customer ratings
* Handle Pagination:
* If there are multiple pages of results, automate the clicking of pagination buttons and repeat the data extraction process.
* Store the Data:
* Save the extracted data in a structured format, such as a CSV file or a Pandas DataFrame, for further analysis or reporting.

**Store data in database:**

* The collected bus details data was transformed into pandas dataframes. Before that, a new database and tables were created.

**web app streamlit:**

* With the help of Streamlit, you can create an interactive application similar to RedBus by designing a user-friendly interface that allows users to search for bus routes, view available buses, and get details like departure times and prices

**Conclusion**

Scraping RedBus data using Selenium provides a way to automatically extract useful bus-related information from the dynamic website. By automating actions such as clicking buttons and navigating through multiple pages, you can gather and store data for analysis, reporting, or any other use case.