Brief Report on Website Information Extraction Project

**Introduction**

The goal of this project was to develop a Python script to extract specific information from a list of websites and save the data into a CSV file. The extracted information included social media links, tech stack, meta title, meta description, payment gateways, website language, and category of the website.

**Approach**

1. Research and Learning

Since this type of project was new to me, the first step was to research and learn about the necessary packages and their uses for web scraping. The key packages used were:

- Selenium: For rendering dynamic web pages.

- Beautiful Soup: For parsing HTML and extracting data.

- Requests: For sending HTTP requests.

- CSV: For saving the extracted data into a CSV file.

2. Setting Up the Environment

Setting up the development environment involved installing the necessary packages and ensuring the Chrome WebDriver was correctly configured for Selenium to work in headless mode.

3. Script Development

The script was developed in stages:

- Defining Functions: Functions were created to extract specific pieces of information from the HTML content.

- Main Extraction Function: A main function `extract\_info` was defined to call the individual extraction functions and compile the data into a dictionary.

- Saving Data to CSV: The script iterates through a list of websites, calls the `extract\_info` function for each website, and writes the data into a CSV file.

4. Testing and Debugging

The script was tested on multiple websites to ensure it worked correctly and handled various scenarios, including websites with different structures and technologies.

Challenges Encountered

1. Learning Curve

This type of project was new to me, so the initial challenge was to understand and learn about the necessary packages and how they could be used for web scraping. This learning process took some time but was crucial for the project's success.

2. SQL Integration

One of the requirements was to store the extracted data in an SQL database. However, my system was not allowing me to connect my SQL server (PC server) environment to the one in Visual Studio Code. Despite multiple attempts and troubleshooting, I had to give up on this due to time constraints and decided to save the data into a CSV file instead.

3. Website Connection Issues

While scraping websites, there were instances where the connection to the website timed out, resulting in failure to retrieve information. This issue required implementing error handling and retry mechanisms to ensure the script continued running despite occasional connection failures.

4. Categorizing Websites

Another challenge was categorizing the websites accurately. The initial list of categories was limited, and expanding this list to include more values required additional effort and refinement.

**Conclusion**

Despite the challenges encountered, this project was a valuable learning experience. The script developed is efficient in extracting valuable information from a list of websites and saving it into a CSV file. It demonstrates proficiency in using Python for web scraping, handling dynamic web content with Selenium, and parsing HTML with Beautiful Soup. The project also highlights the importance of error handling and adaptability in the face of unforeseen issues.