PROJECT REPORT

AIM:

The project aims to extract structured data from the Y Combinator website, specifically focusing on information about companies listed on the platform. The goal is to gather comprehensive details about each company, including its name, description, founding year, team size, location, batch, industry tags, social media links, and founders' information

Methodology Used for Scraping:

The scraping methodology involved using a combination of Selenium and BeautifulSoup to extract data from the Y Combinator website. The script starts by using Selenium to navigate to the target webpage containing a list of companies. Then it clicks on the "See All" option to reveal all batches. It compiles batches of checkboxes that represent filters (e.g., Winter, Summer, or International). The script scrolls down to the bottom of the page to load all the companies. It extracts the URLs of individual company profiles from the loaded page and these URLs are written to a text file for later use. Once the URLs of all companies are collected, it proceeds to scrape each company's website individually using BeautifulSoup and request libraries. The scraped data is structured into a JSON format. Finally, the collected data is saved into a JSON file.

Framework and Libraries:

BeautifulSoup: Beautiful Soup is a Python library used for web scraping HTML and XML files. It allows the script to parse the downloaded HTML content of company web pages to extract relevant information.

Requests: The Requests library is used for making HTTP requests in Python. It enables the script to fetch the HTML content of web pages by sending HTTP GET requests to the specified URLs.

Time: The Time module provides various time-related functions. It is used for introducing delays in the script, primarily for waiting between interactions with the web page to ensure elements are loaded properly.

Regular Expressions(re): Regular expressions are used for pattern matching and manipulation of strings. In the script, regex is used for matching specific patterns in the text content of elements, particularly in filtering checkboxes based on certain criteria.

JSON: JSON was used for storing the extracted data in a structured format for further analysis and future use.

Selenium: Selenium WebDriver was chosen for its ability to automate web browser interactions, making it suitable for scenarios like infinite scrolling. It is used in conjunction with a WebDriver (in this case, Chrome WebDriver) to automate interactions with web pages, such as clicking on elements, scrolling, and extracting data from dynamic content.

Tqdm: It is used in loops to visualize the progress of iterative tasks, such as iterating through batches of checkboxes or URLs, providing a visual indication of the script's progress.

Code Explanation:

The code consists of two main functions:

1.yc_links_extractor():

- make_driver(): it sets up a headless Chrome WebDriver.
- get_url(): Visit Y Combinator website.
- click_see_all_options(): Clicks on the "See All" options to reveal all filters.
- compile_batches(): Iterating through batches by clicking their checkboxes.
- scroll_to_bottom(): Scrolls down to the end of each displayed list to ensure all companies are loaded.
- fetch_url_paths(): It extracts URLs of individual company profiles from the loaded page.
- write_urls_to_file(): Function to write the extracted URLs to a text file.
- read_urls_from_file(): Function to read URLs from the text file.

2.scrape_company_website():

Function to scrape relevant information from individual company profiles.

- Fetches the HTML content of the company webpage using requests.

- Parses the HTML content with BeautifulSoup to extract various company details like name, tagline, description, industry, location, website, founding year, team size, social media links, founder information (including names, biographies, and social media profiles).
- Builds a dictionary structure containing all the scraped information for a single company.
- Saves the company data as a formatted JSON string.

Challenges encountered and Solutions:

One of the main challenges faced in the project was dealing with infinite scrolling on the Y Combinator website. To overcome this challenge, the project implemented a custom scrolling function using the Selenium WebDriver to ensure that all companies' information is loaded before extracting their URLs. Additionally, batch-wise link extraction posed some difficulties but managed by iterating through batches of links and processing them sequentially. Difficulties arose due to varied HTML structures for founder information. Founders' names were consistently extractable, but their roles were inconsistently structured. To mitigate this, focus was placed solely on extracting founders' names, omitting roles where HTML structure varied significantly.

Summary of Extracted Data:

The extracted data includes company names, taglines, descriptions, founding years, team sizes, locations, batch information, company types, industry tags, social media URLs, founders' information, and biographies. All extracted data is stored in a structured JSON format, making it suitable for further analysis and processing. Overall, the scraping process involved a combination of automation, web parsing techniques, and data extraction to gather comprehensive information about Y Combinator companies. The resulting dataset provides valuable insights into various startups associated with Y Combinator and their respective details.