# The Methodology used for scraping data from the website

My method for extracting data from the website comprised of two distinct applications. Initially, I created a program to retrieve the company links from the webpage. Subsequently, I developed an additional software application to access every company webpage and extract the required data via scraping. The necessary information could be systematically and effectively extracted from the website thanks to this two-step methodology.

# Why I chose the framework and the libraries used

## Selenium

I chose to use Selenium because of its exceptional ability to manage dynamic websites with lots of interactive features. To put it another way, Selenium excels at complex websites that have the feel of vibrant marketplaces, but other tools are great for simpler tasks. It is the best option for navigating through such difficult web pages because of its capacity to adjust to these dynamic environments and interact with them without any problems. Therefore, Selenium is my first choice for managing these complex online scenarios due to its specialised capabilities, even though alternatives like Playwright and Beautiful Soup are also available.

## Json

You can work with JSON (JavaScript Object Notation) data using this Python library. It offers functions for converting Python objects into JSON strings and vice versa.

## Threading

Using a library called Threading, you can execute several tasks at once inside of a single process. It helps you multitask and makes your programme run more smoothly, particularly when you're working on tasks that can be completed on your own.

## Queue

With the help of the FIFO (First In, First Out) data structure provided by the Queue module, you can efficiently enqueue and dequeue items. It is frequently employed in multithreaded applications for task management.

## Webdriver

The Selenium library's webdriver module offers a collection of classes and methods for working with web browsers. It lets you to create instances of browsers and use your Python code to manipulate their behaviour.

## Service

In order for Selenium to control Google Chrome, this module offers a service class for handling the ChromeDriver server.

## By

Constants representing various methods of locating elements on a web page—by ID, class name, or XPath, for example—are contained in the By module.

## WebDriverWait

The WebDriverWait class offers methods for delaying the execution of your code until specific requirements are satisfied. It is frequently used to synchronise your code with the current state of a webpage, for example, when you want to wait for certain elements to become clickable or visible.

## Expected Conditions

WebDriverWait can wait for a set of predefined conditions found in this module. These conditions, which include elements being present, visible, or clickable, reflect typical situations you may run into when interacting with web pages.

## Options

This module offers a class for setting up preferences and settings for the Chrome browser instance that Selenium creates. It lets you adjust the way the browser behaves in a number of ways, including managing browser windows, turning on and off features, and setting user preferences. This helps the web scrapping faster by reducing the loading time and consumes less data

## Code Explanation

## 1.Link Extraction

First of I have imported all the necessary libraries like

- Selenium
- Webdriver
- Service
- By
- WebDriverWait
- Time
- Expected Condition (EC)

```
batches=[]
for i in range(2,41):

path="/html/body/div/div[2]/section[2]/div/div[2]/div[1]/div/d
iv[10]/div["+str(i)+"]/label/input"
    batches.append(path)
```

I have created an empty list named “batch”.Then started appending the Xpath of each tickboxes near the batch names.When I looked up the xpath of each batch I noticed that instead of using clicking each tickboxes and extracting data by only changing one number in the xpath so that I can get the Xpath of each tickboxes near the batch name and added everything to a list.

```
for batch in batches:

select=WebDriverWait(driver,20).until(EC.presence_of_element_l
ocated((By.XPATH,batch)))
    select.click()

    time.sleep(7)
```

Now I have appended each and every xpaths and iterating through each batch and waiting for 7 seconds for the webpage to load completely

```
def extract_hrefs_from_column(xpath):
        service = Service('chromedriver.exe')
        driver = webdriver.Chrome(service=service)
        driver.get("https://www.ycombinator.com/companies")


        WebDriverWait(driver,
10).until(EC.presence_of_element_located((By.XPATH, xpath)))


        def scroll_to_bottom():
                driver.execute_script("window.scrollTo(0,
document.body.scrollHeight);")
```

```
                time.sleep(2)

        scroll_to_bottom()


        elements = driver.find_elements(By.XPATH, xpath)


        while True:

                scroll_to_bottom()
                new_elements = driver.find_elements(By.XPATH,
xpath)
                if len(new_elements) == len(elements):
                        break
                elements = new_elements


        hrefs = [element.get_attribute("href") for element in
elements]


        driver.quit()

        return hrefs
```

Inside the function to extract hrefs the the column that take the argument as column Xpath.I have assigned chromedriver to the variable name 'service' and using the driver to get the webpage.Then I have selected the column from which I want to collect the hrefs.

Then I created a function named scroll to bottom to execute the scrolling of the webpage with a sleep time of 2 second for the complete herfs to load.Then an infinite loop to execute the scroll until the end of the webpage.

Using the list comprehension I have extracted only the herfs by using get_attribute function and quit the driver.

```
elect = WebDriverWait(driver,
20).until(EC.presence_of_element_located((By.XPATH, batch)))
select.click()
```

This is a code to untick the batch that has been selected previously

```
def save_links_to_txt(hrefs):
        with open("company_links.txt", "a") as f:  # Open the
file in append mode ("a")
                for href in hrefs:
                        f.write(f"{href}\n")
```

Save_links_to_text function to append data into a txt file name company link.txt and adding each hrefs line by line

```
column_xpath = "//a[@class='_company_99gj3_339']"
hrefs = extract_hrefs_from_column(column_xpath)

save_links_to_txt(hrefs)
```

Specifing the column xpath and executing the code to save the herfs into the file.

# 2.Web Scrapping

Importing all the necessary libraries like

- Selenium
- Json
- Threading
- Service
- Options
- Time
- WebDriverWait
- Webdriver
- By
- Expected Conditions

```
def extract_data_from_website(url):
    chrome_options = Options()
    chrome_options.add_argument("--blink-
settings=imagesEnabled=false")
    chrome_options.add_argument("--headless")
    service =
webdriver.chrome.service.Service('chromedriver.exe')  #
importing the service
    driver =
webdriver.Chrome(service=service,options=chrome_options)  #
assigning the service
    driver.get(url)  # Use the passed URL instead of hardcoded
one

    # Initialize industry tags for each website
    industry_tags = []

    name_element = WebDriverWait(driver, 2).until(
        EC.presence_of_element_located((By.XPATH,
"//h1[@class='font-extralight']")))
    name = name_element.text.strip()
```

```
    try:
        tagline_element = WebDriverWait(driver,
2).until(EC.presence_of_element_located((By.CLASS_NAME, 'text-
xl')))
        tagline = tagline_element.text.strip()
    except Exception as e:
        tagline = None
```

In this function named extract data from the website it scrape data through each website and then extracting the data. Options are used to reduce the webpage loading time and make it faster by avoid loading the pictures etc.

Then importing the necessary services.Creating an empty list to add the industry tag like in the given format.Then giving identifying each element by providing sufficient time to load the element and scrap the information and also implementing error handling to not produce an error incase of absence of data.

```
def worker():
    while True:
        url = queue.get()
        if url is None:
            break
        data = extract_data_from_website(url)

        with open('data.json', 'a') as json_file:
            json.dump(data, json_file, indent=4)
            json_file.write(',\n\n')
        print(f"Data extraction completed for {url}")
        queue.task_done()
```

In the worker function an infinite loop is introduced and url queue is introduced to load 10 website at a time to increase to speed of webscrapping.If condition is given when there is no hrefs the loop is exited. Then writing every extracted data into a json file named data.json and appending the data into the file.

```
with open('Allcompany_links1.txt', 'r') as file:
    urls = file.read().splitlines()
```

This is code to iterate through each line of hrefs from the link extraction text file and it is read line by line

```
queue = Queue()

for url in urls:
    queue.put(url)
```

Assigning the queue functions and adding each hrefs to the queue for web scrapping

```
threads = []
for _ in range(10):
    t = threading.Thread(target=worker)
    t.start()
    threads.append(t)
```
Creating an empty list named threads .Within a for loop that execute 10 time the threading begin and executing the worker function and it is append to the list threads and executing the multi tasking for web scraping and the threading is being started.

```
queue.join()
```
This is the code to block the main program execution until all tasks in the queue have been completed and all threads have finished processing.

```
for _ in range(10):
    queue.put(None)
```
This is the code to stop the worker function.

```
for t in threads:
    t.join()
```
This is the code to join the threads

```
print("All data extraction completed.")
```
This is the code to display when the data extraction is completed.

# Challenges I have encountered while doing the project

- First main challenge was the selection of the best library, I have gone through several websites and referred chatgpt for help
- Second main challenge was regarding the hrefs extraction first when I have extracted the hrefs I got only 1000 hrefs because that was the limit of the webpage.Then I have extracted the hrefs batchwise I was able to solve the issue
- Third main challenge was regarding the duplication of the hrefs then I have to introduced sufficient sleep time for the elements to completely load
- Fourth main challenge that I come across was the xpath changing according to the orientation of the webpage later I discovered that and I started working accordingly.
- Fifth main challenge was the web scraping was taking too much of data then I started using an unlimited connection for data extraction.
- Sixth main challenge was regarding the time of web scraping as I was using selenium I was able to solve the problem to an extend by using the method of threading that helped me to solve that problem to an extend

- Seventh main challenge that I faced was due to connection time out. I had done the data extraction of the remaining hrefs after each time until the full extraction is complete.

# Summary of the Data Extracted

I had a great time learning new libraries and get much more familiar with web scraping and understood the main challenges that I will be facing when dealing with dynamic webpages.
The data extracted from the webpage gave me certain insight that mostly some of the information is not present on the website so error handling is important. The xpaths of the webpage changes in different and I have to with different xpath where the ideal location of the element and the only thing that the all company have is company name other than that there is something missing in each of the webpages.