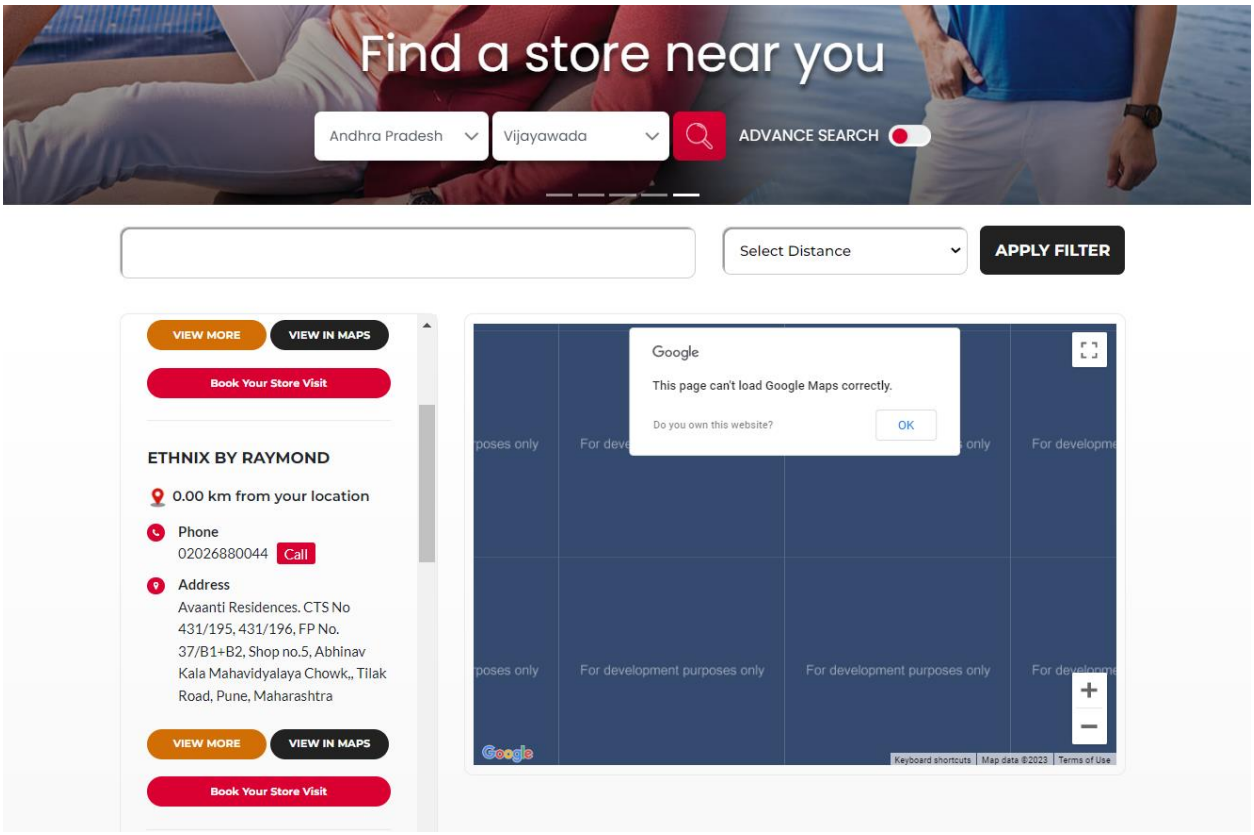
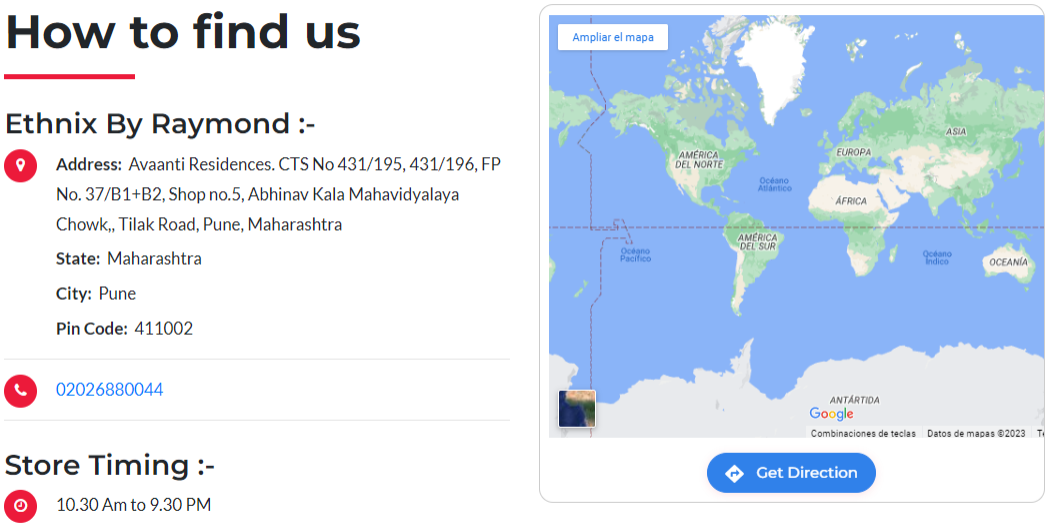
Store Selected: Raymond Clothing

I went to the Rayond Clothing website and Selected Stores near you tab on their page to get a list of stores that they had in India. A page with options to find all the different stores in India according to respective States and Cities was opened. A snapshot is provided below.



Approach 1: Looping though State and Cities options using Selenium.

In this approach I went though selecting a particular state and then all the cities corresponding to it. For each State and city combination, I copied the “VIEW MORE” link in the store section for all the available stores. The “VIEW MORE” link reference to a webpage which contained detailed information about the store, a snapshot is provided below.



Here I could scrape all the information about the store like Store Name, Address, Contact Number, Timings and Coordinates (coordinates could be found by scraping the link from "**Get Direction**" button, it is of the form "https://www.google.com/maps/search/?api=1&query=**latitude**,**longitude**".

Problems with Selenium: There were a total of 25 states with an average of 10 cities to cycle though. Combined with the wait times for populating the browser window with information about stores, the entire process was too slow.

Solution: I found that "VIEW MORE" button referenced to a page of the form "https://stores.myraymond.com/mystore/raymondstore.php?**store_id=1675**". On inspection I found that each store had a unique store_id and we could just loop though a range of ids (say 1 to 1700) to get information regarding all the stores.

Approach 2: Using requests and bs4 for store_id approach.

Using requests and bs4, I went though all the different web elements that held the required information and stored them in a list. Then I used pandas to create a dataframe and finally saved the entire information in a .csv file.

Problem: Since I was sending requests about each page sequentially, a lot of time was wasted since until the previous page was parsed for all the information, the program was sitting idle.

Solution: Async requests. I changed to aiohttp and async library to create a list of tasks to be completed. Using async calls a larger number of requests could be sent and the scraping task could be done as soon as the page was available for all the different tasks. The total time to scrape was brough down to around 25 seconds from 3 minutes.