**Problem 1**

**Deliverables:**

Scrape list of doctors at Kaiser Permanente, Northern California Region within the Redwood City Office.

<https://healthy.kaiserpermanente.org/northern-california/doctors-locations#/search-result>

**Description:**

Please write a scraper with which you can scrape the following details. Get at least 50 Physician with following details. Show the code and implementation details.

**Physician Name:** [Stella Sarang Abhyankar, MD](https://healthy.kaiserpermanente.org/physicians/resourceid/7773762)

**Physician Specialty:** Hospital Medicine

**Practicing Address:**

Redwood City Medical Center

1150 Veterans Blvd

Redwood City, CA 94063

**Phone:** 650-299-2000

*Challenges encountered to get the data:*

*------------------------------------------------*

*Research done with the data before scraping the data from the above url given.*

*These are the website I visited to grab the data from:*

*These are all the api’s used for searching doctors:*

*(Had challenges as could not find all the data needed, but only some of them could find).*

[*https://rockhealth.com/kaiser-permanente-launches-open-api/*](https://rockhealth.com/kaiser-permanente-launches-open-api/)

<https://developer.betterdoctor.com/>

<https://developer.betterdoctor.com/documentation15>

<https://app.drchrono.com/api-docs/v2016_06/tutorial#>

<https://rockhealth.com/work-with-us/>

ttps://developer.betterdoctor.com/

<https://developer.betterdoctor.com/documentation15>

<https://app.drchrono.com/api-docs/v2016_06/tutorial#>

<https://rockhealth.com/work-with-us/>

Implementation of the code on Jupyter notebook:

*!pip install splinter*

*# Dependencies*

*# https://splinter.readthedocs.io/en/latest/drivers/chrome.html*

*from selenium import webdriver*

*from splinter import Browser*

*from bs4 import BeautifulSoup*

*import requests*

*import tweepy*

*import yaml*

*import pandas as pd*

*import time*

*import re*

*import pymongo*

*#driver = webdriver.Chrome('/path/to/chromedriver')*

*executable\_path = {'executable\_path': '/usr/local/bin/chromedriver'}*

*browser = Browser('chrome', \*\*executable\_path, headless=False)*

*# Scrape the Kaiser Redwood city office site in northern california.*

*kaiser\_url = "https://healthy.kaiserpermanente.org/northern-california/doctors-locations#/search-form"*

*response = requests.get(kaiser\_url)*

*# Create BeautifulSoup object; parse with 'html.parser'*

*soup = BeautifulSoup(response.text, 'html.parser')*

*# To understand the data before collecting from the web.*

*print(soup.prettify())*

*# Visit website using Splinter*

*browser.visit(kaiser\_url)*

*# Select California - Northern from dropdown menu*

*browser.select("Region", "NCA")*

*# Select Redwood City from City dropdown*

*browser.select("city-dropdown-li", "Redwood City")*

*# Select doctors at Kaiser Permanente, Northern California Region within the Redwood City Office*

*browser.click\_link\_by\_id('searchButton')*

*# list of physicians dictionary*

*list\_of\_physicians = {}*

*#loop through all the physicians 20 \* 3 to get the 60 doctors. 20 from each page.*

*browser.visit(url)*

*for x in range(3):*

*#loop through all the 20 Physicians from each pagination.*

*#results = soup.find\_all('div', class\_='tab content')*

*results = soup.find\_all("div")*

*print(results)*

*Practicing\_Address = []*

*for r in results:*

*el = r.find\_all("div", class\_="result-list")*

*each\_dr = el.find("div", class\_="detail-data") #each dr info*

*# Create BeautifulSoup object; parse with 'html.parser'*

*html = browser.html*

*soup = BeautifulSoup(html, 'html.parser')*

*Physician Name = soup.find("a", class\_="bold-font doctorTitle").get\_text()*

*Physician Specialty = soup.find("div", class\_="specialtyMarginlineSpacing").strip()*

*medical\_location = soup.find("span")[0].text*

*street\_address = soup.find("span")[1].text*

*city = soup.find("span")[2].text*

*state = soup.find("span")[3].text*

*zipcode = soup.find("span")[4].text*

*# Keep a dictionary for each hemisphere. The dictionary contains the title and the feature image.*

*Practicing\_Adress.append({"medical\_location": medical\_location,*

*"street\_address": street\_address,*

*"city": city,*

*"state": state,*

*"zipcode": zipcode})*

*Phone = soup.find("div", class\_"doctorPhone")[none].text()*

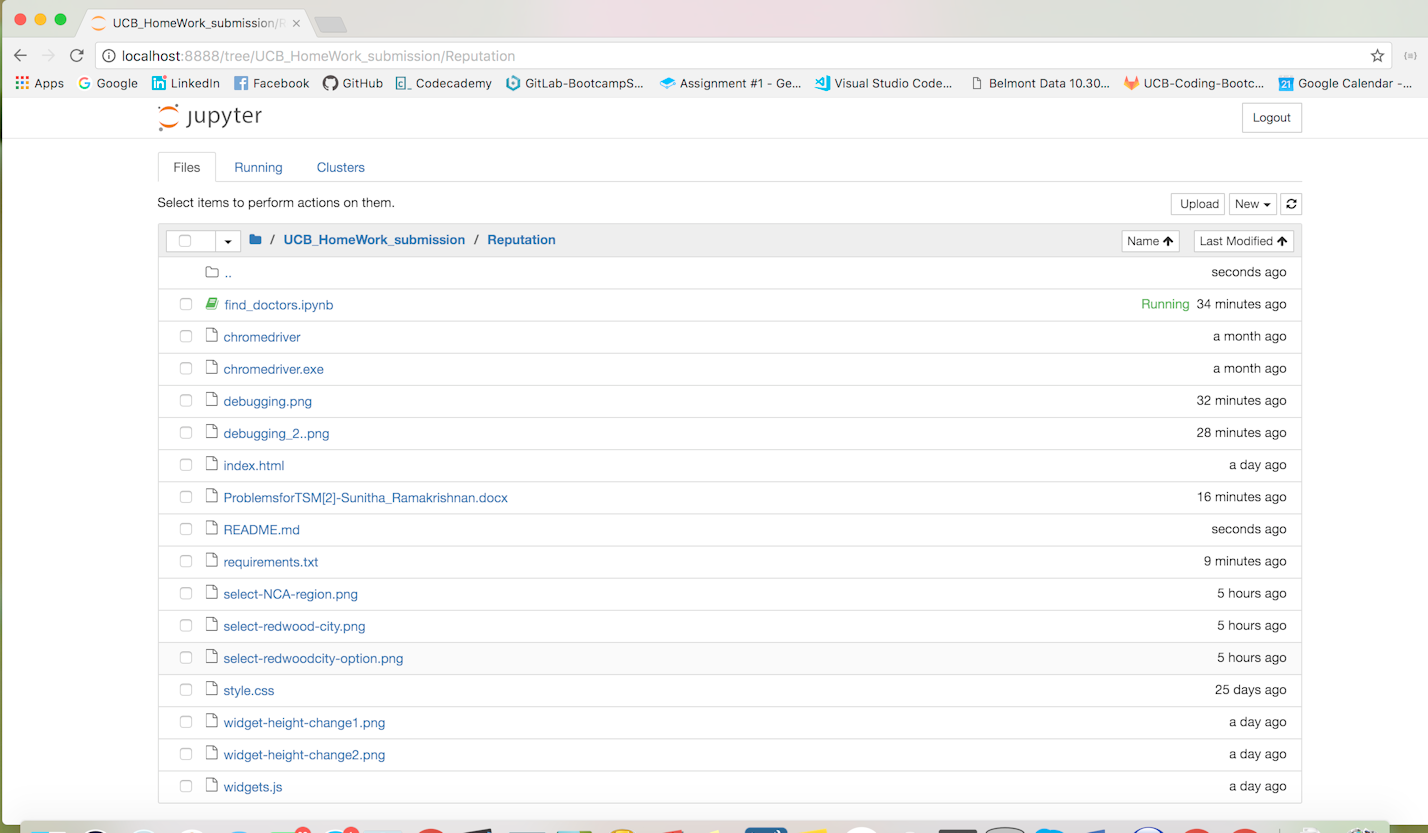
*# Add all the data collected to list\_of\_physicians dictionary*

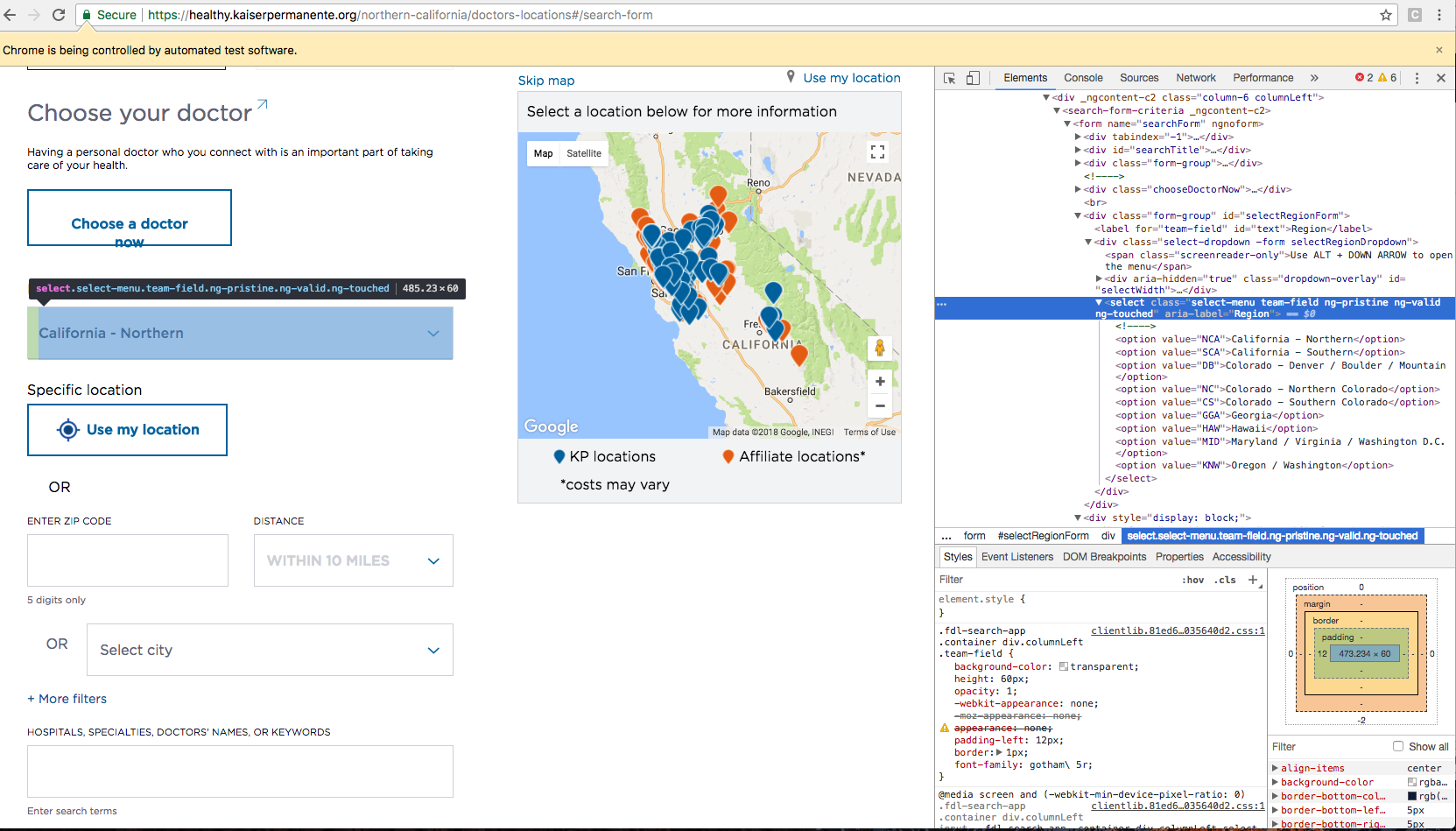
*list\_of\_physicians["Physician Name"] = Physician Name*

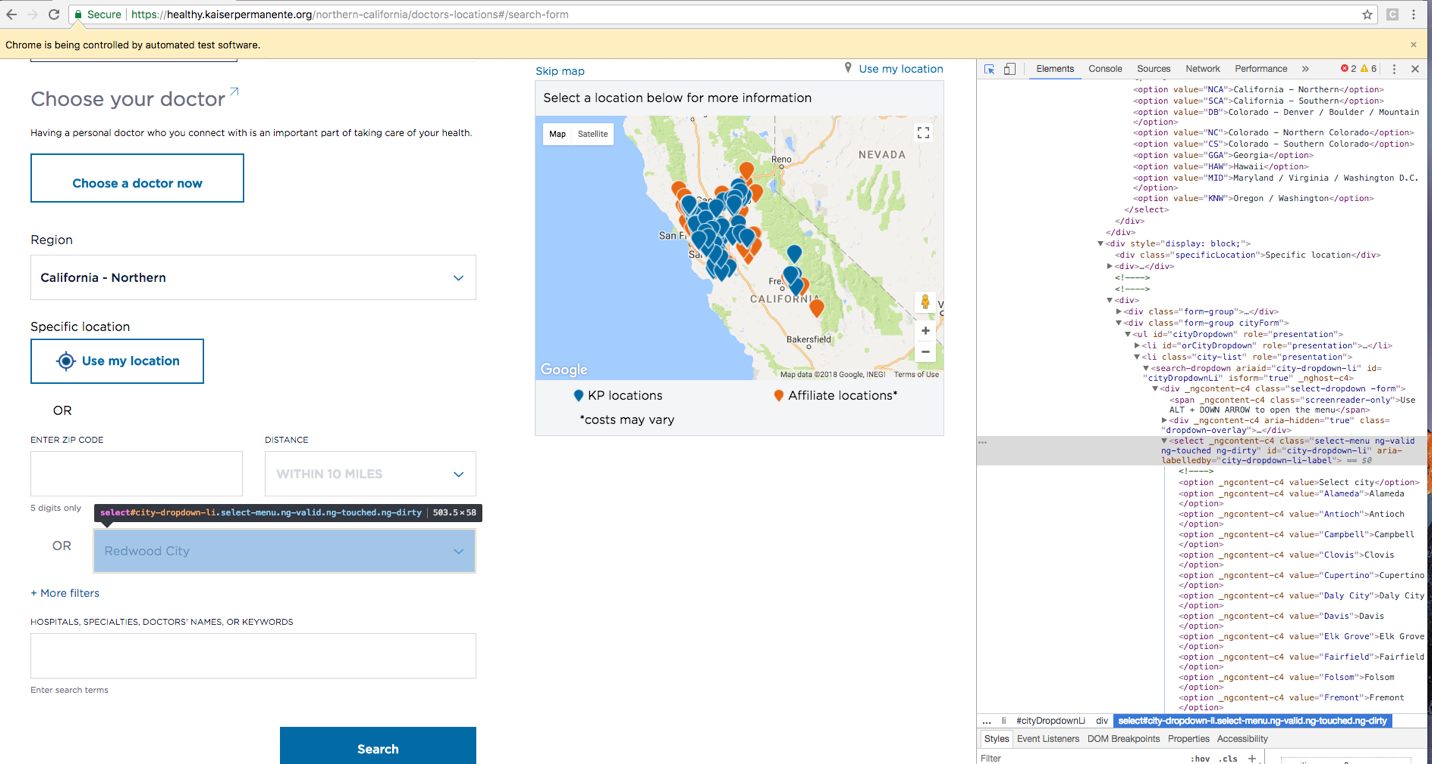
*list\_of\_physicians["Physician Specialty"] = Physician Specialty*

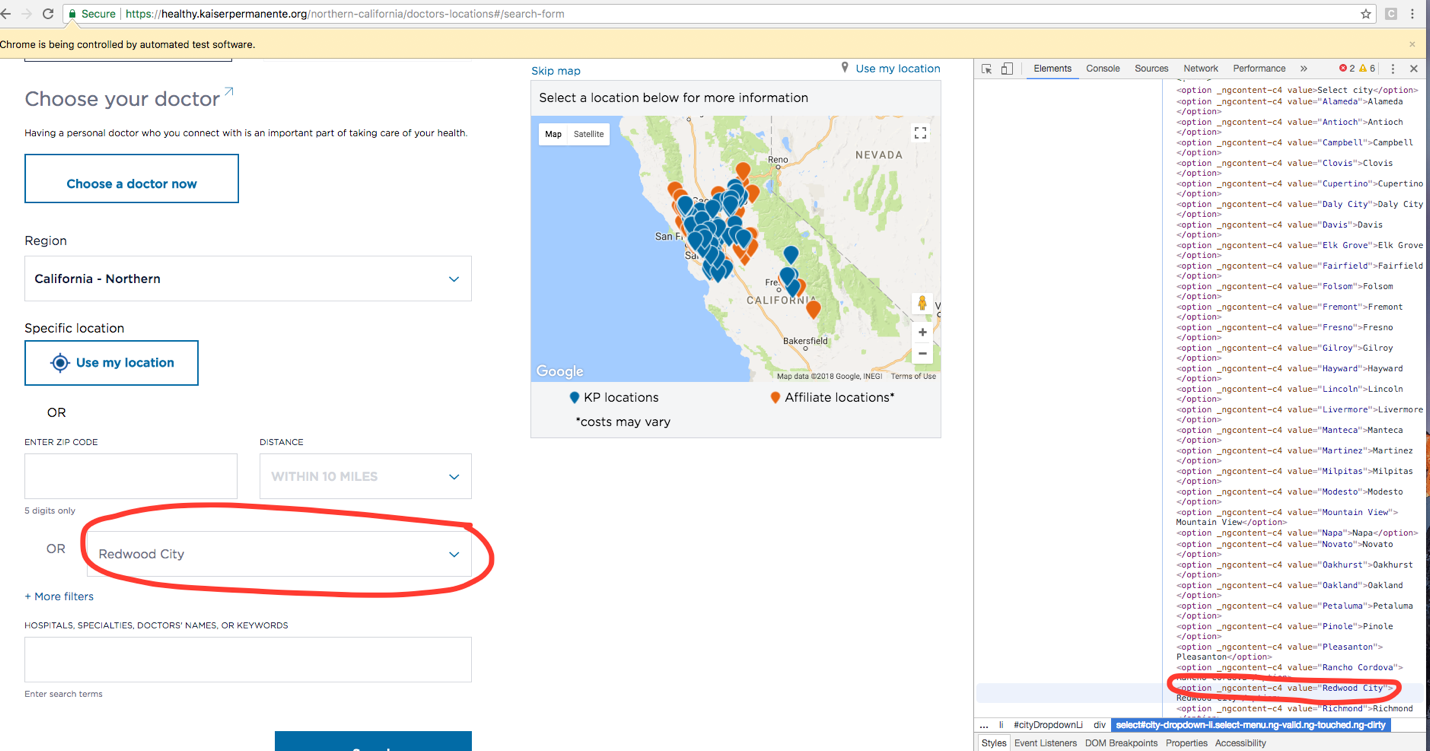
*list\_of\_physicians["Practicing\_Adress"] = Practicing\_Adress*

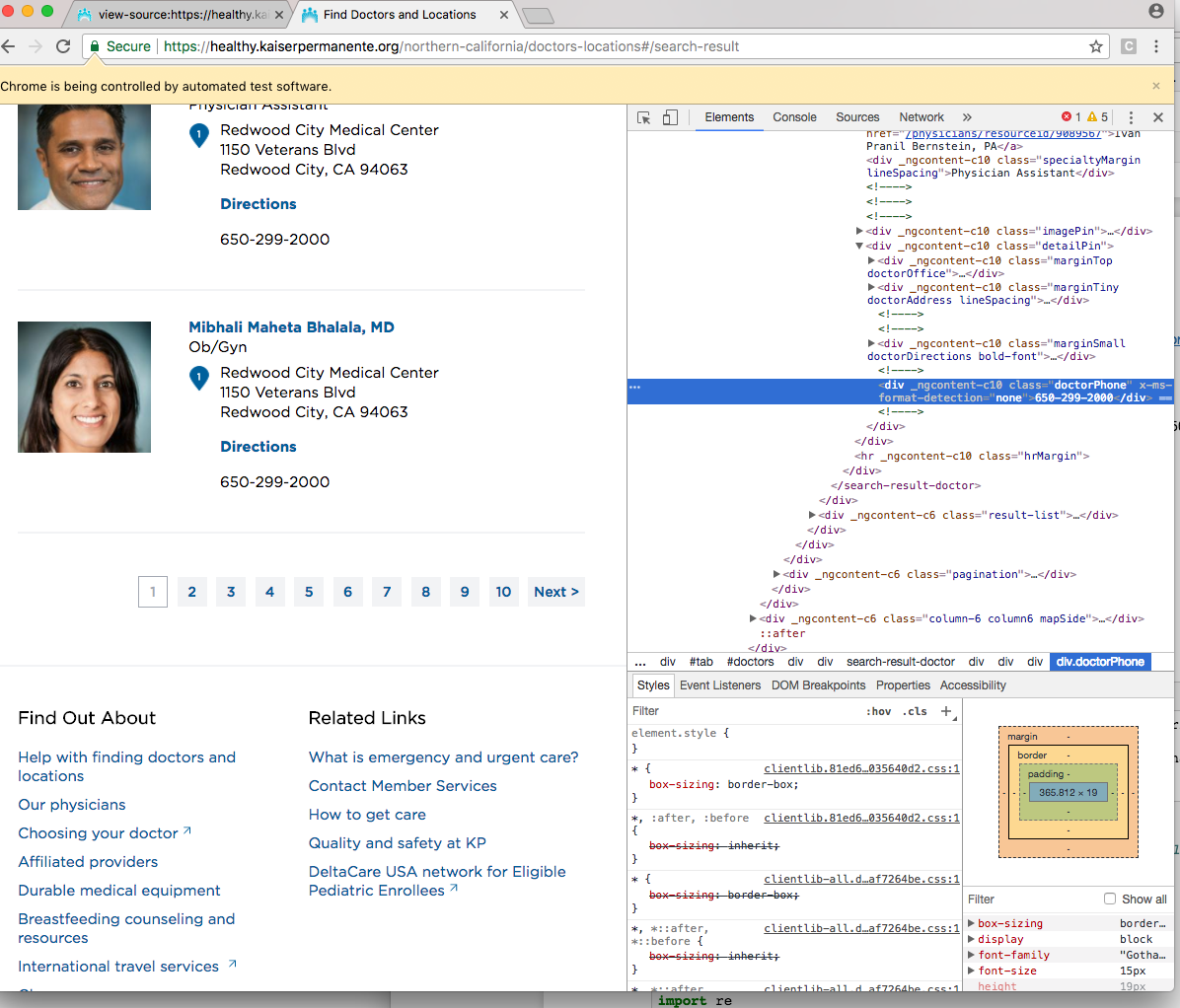
*list\_of\_physicians["Phone"] = Phone*

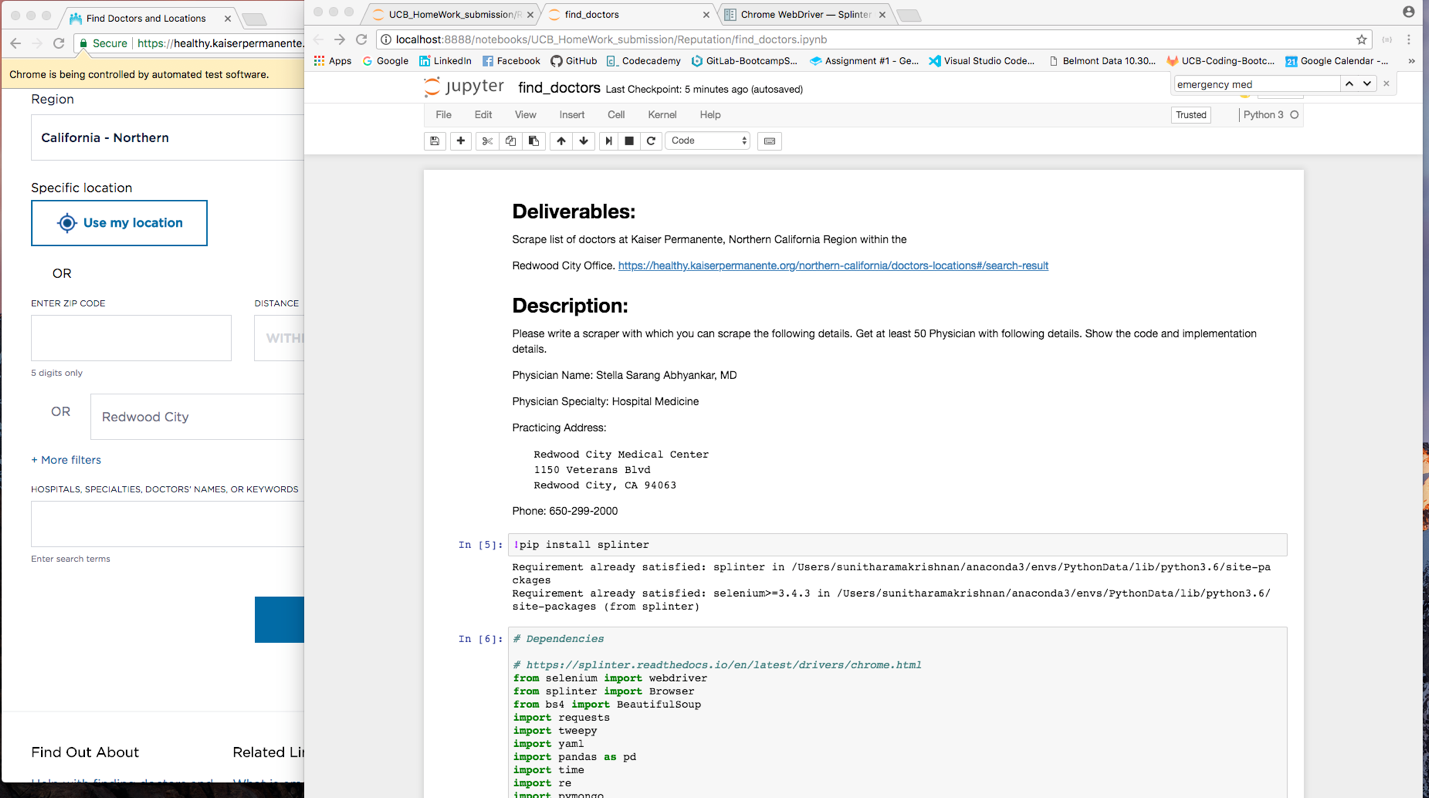
**

**

**

**

**

**

**Problem 2:**

**Description**

This is an IFRAME script that renders reviews. Build an HTML page that will render this script.

State how you will change the height of the widget.

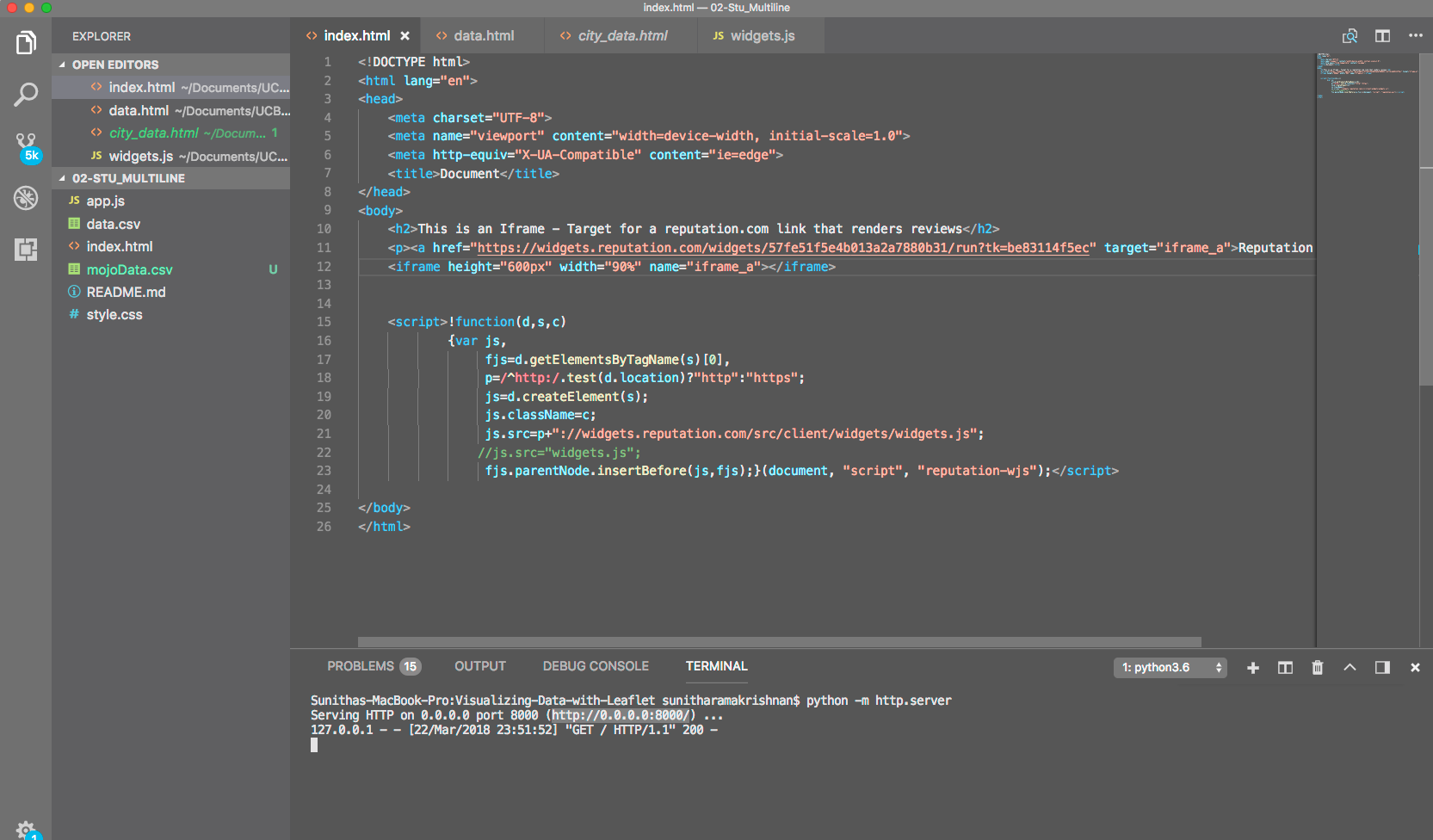
<a class="reputation-widget" target="\_blank" href="https://widgets.reputation.com/widgets/57fe51f5e4b013a2a7880b31/run?tk=be83114f5ec" data-tk="be83114f5ec" data-widget-id="57fe51f5e4b013a2a7880b31" env="">Reputation Reviews</a>

<script>!function(d,s,c){var js,fjs=d.getElementsByTagName(s)[0],p=/^http:/.test(d.location)?"http":"https";js=d.createElement(s);js.className=c;js.src=p+"://widgets.reputation.com/src/client/widgets/widgets.js";fjs.parentNode.insertBefore(js,fjs);}(document,"script","reputation-wjs");</script>

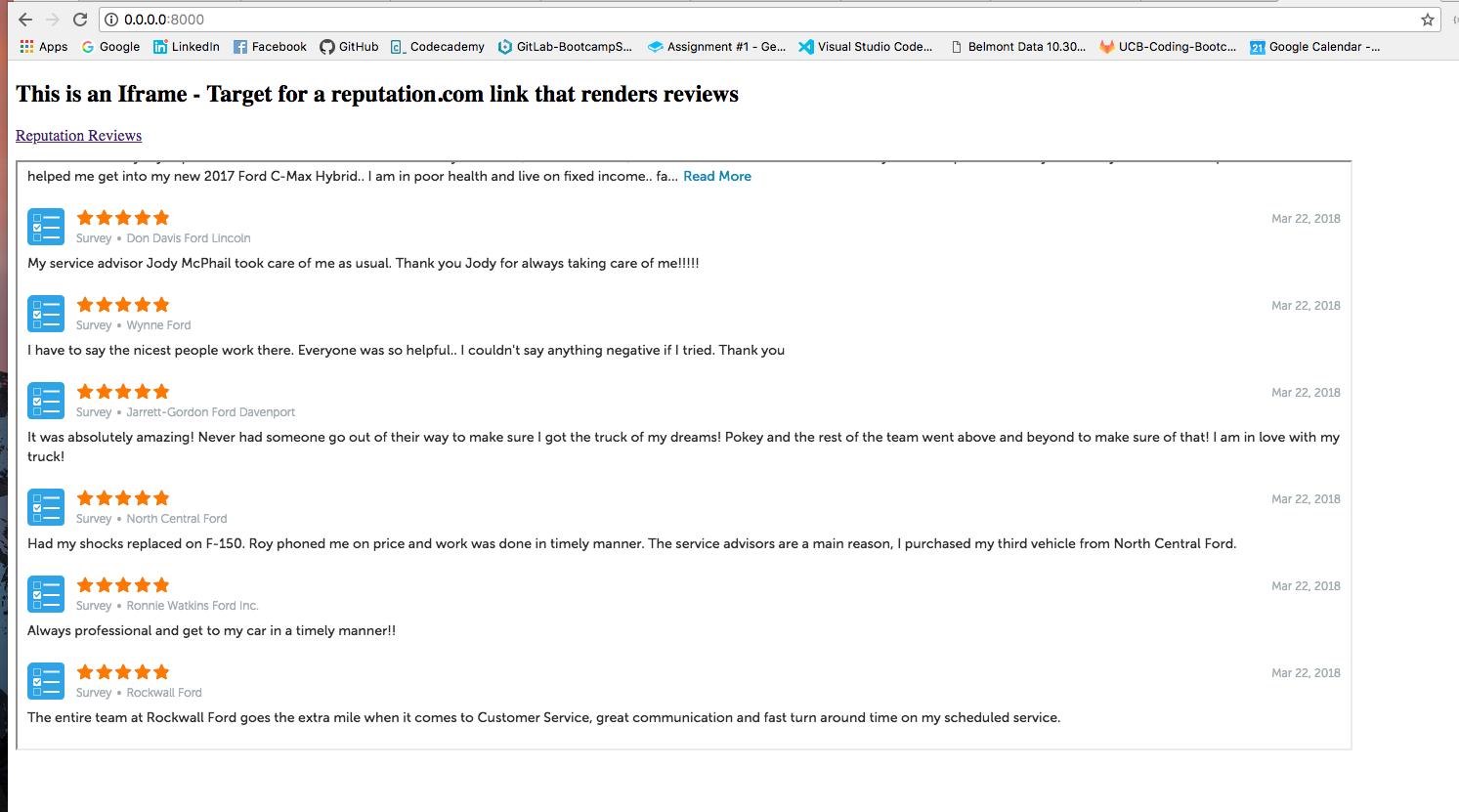
**Deliverables:**

HTML code that wraps the script and renders a page with 3rd party reviews

I will be attaching my “index.html” code to the email .



This is the output of the HTML file which renders page with 3rd party reviews. I’ve a scroll bar inside the iframe tag:

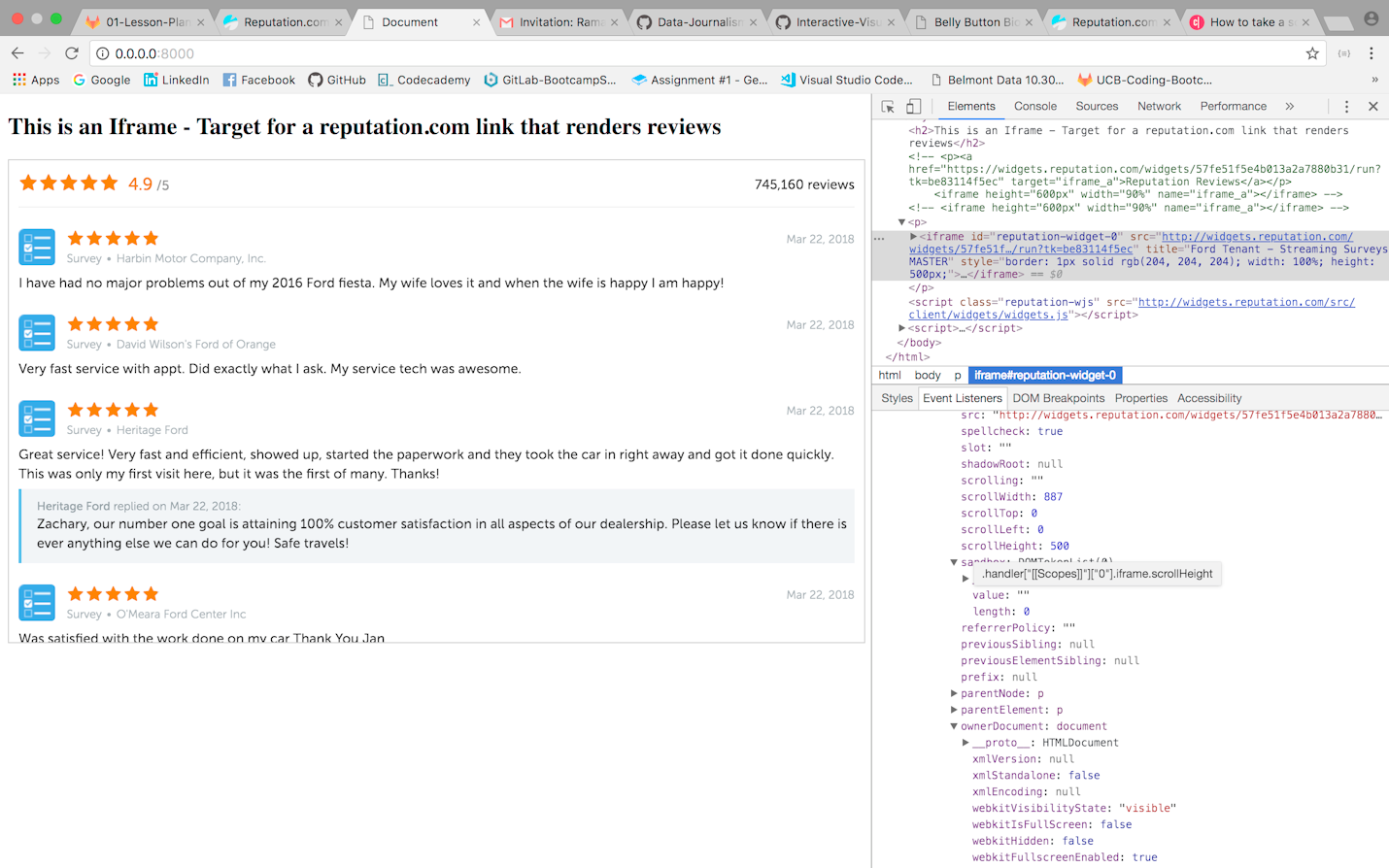


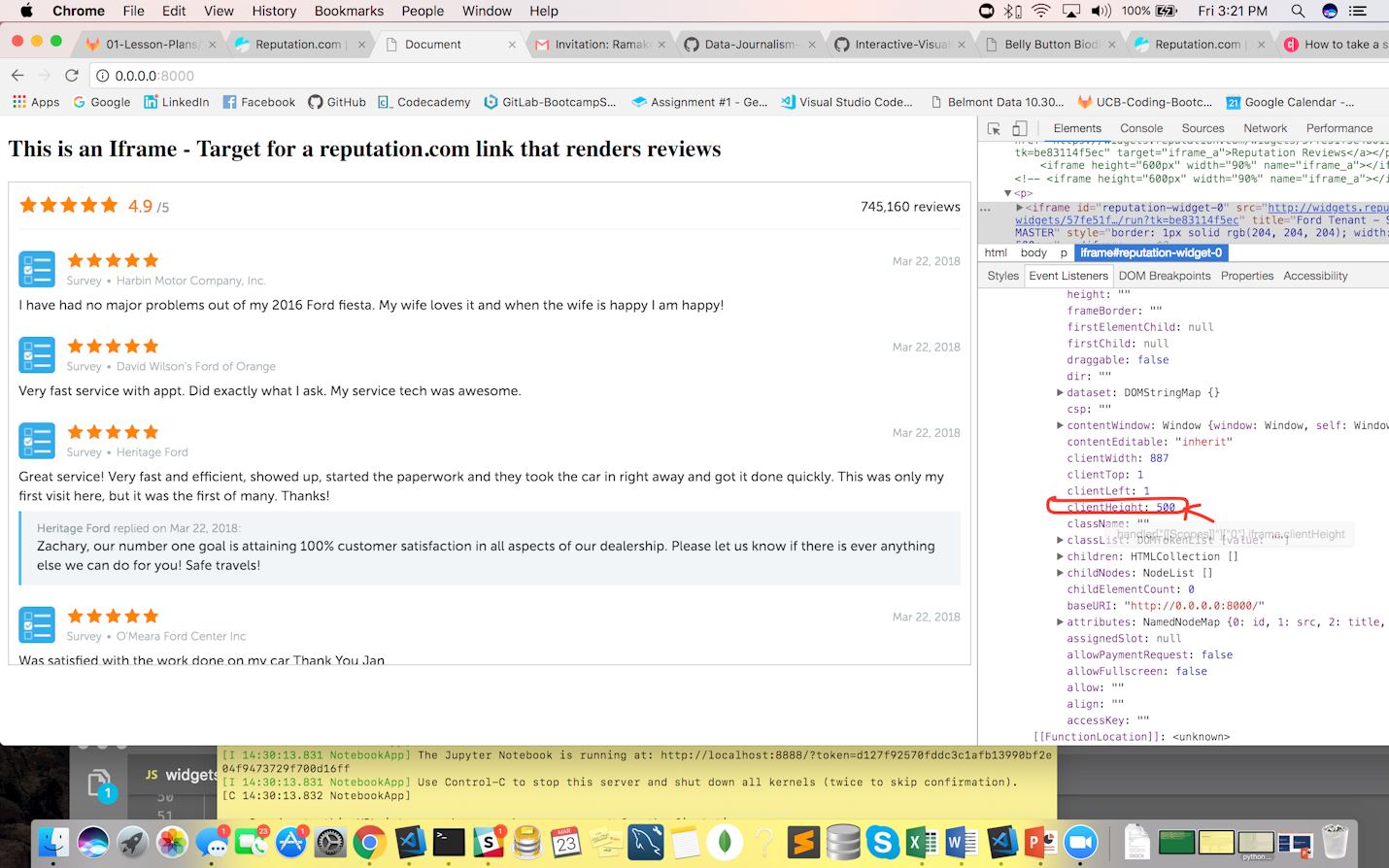
**Extra Points:**

Describe different variables used in this implementation

1. Height of the widget can be changed with the following:

<iframe height="600px" width="90%" name="iframe\_a"></iframe>





1. **data-widget-id**="57fe51f5e4b013a2a7880b31"

you need to include the data-widget\_id parameter to generate/building the http or https url to go third party and fetch the data.

1. **getElementsByTagName(s)[0]**

get elements of DOM \*Document Object Model with getElementsByTagName method(s)

(s) : is a string and can be any class

[0]: html content of the first <a class>

element (index 0) in the document.

1. **<a href :**

Is a anchor tag with reference to the url link

1. **class="reputation-widget"**

class is a attribute specifies one or more class names for html element. The class names can be used by CSS and javascript to perform certain tasks.

1. **target="\_blank"**

An iframe is used to display a web page within a frame like webpage.

An iframe can be used for a target with link.

1. **env="">Reputation Reviews” and data-tk="be83114f5ec"**

specific to the third party api’s identifiers/to fetch the data.

1. **p=/^http:/.test(d.location)?"**

Building up the url location to visit the 3rd party review site. Searching the string with starting with first characters /^http:/ and test if it does have a document location, then assign that value to ‘P’.

1. **d.createElement**

It is an web api to create document element.