CSCI 4131 – Internet Programming

Homework Assignment 5 - Introduction to Node.JS

Due Date: 3pm Friday, March 29th

Late Submissions accepted with Penalty after Due Date through Saturday March 30th at 6pm

Submissions after 6pm March 30th will not be accepted

This an individual assignment. Do your own work – as explicitly specified in the Syllabus. See the instructor if you have questions.

1 Description

The objective of this assignment is to introduce web-server development with Node.js. We will provide most of the client-side code and some of the server-side code for this assignment to you, and you are required to add/complete certain functions to complete the assignment. Node.js is basically JavaScript running a Web- server. It uses an event-driven, non-blocking I/O model. So far, in this course we have used JavaScript for client-side scripting. For this assignment, we will use JavaScript for server-side scripting. Essentially, instead of writing the server code in Python, we will develop a basic web-server using JavaScript.

In this assignment, use either JavaScript or jQuery to request data using Asynchronous JavaScript and XML (AJAX) and manipulate the Document Object Model of the Webpage making the AJAX request. AJAX is used on the client-side to create asynchronous web applications. As discussed in class and the assigned reading, it is an efficient means of requesting data from the server, receiving data from the server, and updating the web page without reloading the entire web-page.

If you want to use jQuery, which is a JavaScript library, a good tutorial to start with is available at w3schools at the link: https://www.w3schools.com/jquery/

2 Preparation and Provided Files

I. The first step will be to get Node.js running on CSE lab machines. This can be accomplished as follows:

1. Log into a CSE lab machine.
2. Most of the CSE lab machines run version 8.11.4 of Node.js (as of 03/06/2019) which is the most current version.
3. Open the terminal and type the following command to add the Node.js module:

```
module add soft/nodejs
```

4. The next step is to check the availability of Node.js. Type the following command to check the version of Node.js on the machine:

```
node -v
```

5. This will display the current installed version.

II. The second step is to create a Node.js project for this assignment as follows:

Open a terminal on a CSE lab machine, then:

1. Create a directory named <x500id_hw05> by typing the following command:
   mkdir yourx500id_hw05

2. Go inside the directory by typing the following command:
   cd yourx500id_hw05

3. Having a file named *package.json* in Node.js project makes it easy to manage module dependencies and makes the build process easier. To create *package.json* file, type the following command:
   npm init

4. This will prompt you to enter the information. Use the following guideline to enter the information (The things that you need to enter are in bold. Some fields can be left blank.):

   package name: (yourx500id_hw08) **yourx500id_hw05**

   version: (1.0.0) **<Leave blank>**

   description: **Assignment 5**

   entry point: (index.js) **<Leave blank>** **(**We will provide an index.js file for your use**)**

   test command: **<Leave blank>**

   git repository: **<Leave blank>**

   keywords: **<Leave blank>**

   author: **yourx500id**

   license: (ISC) **<Leave blank>**

5. After filling in the above information, you will be prompted to answer the question: "`Is this ok? (yes)`". Type **yes** and hit enter.
6. Now copy all the files present that are provided for this assignment to this directory: *yourx500id_hw05*
7. Change the permissions of all files and folders inside *yourx500id_hw05 directory **(including the files in the client directory)** to 777*.
8. Listing (`ls -al`) all the available files in your ***HW5*** directory should display the following:

   drwxrwxrwx 2 madis180 CSEL-student 6 Mar 6 11:51 client/
   -rwxrwxrwx 1 madis180 CSEL-student 715 Mar 6 11:54 createServer.js
   -rwxrwxrwx 1 madis180 CSEL-student 219 Mar 6 11:56 package.json
   -rwxrwxrwx 1 madis180 CSEL-student 627 Mar 6 11:45 schedule.json

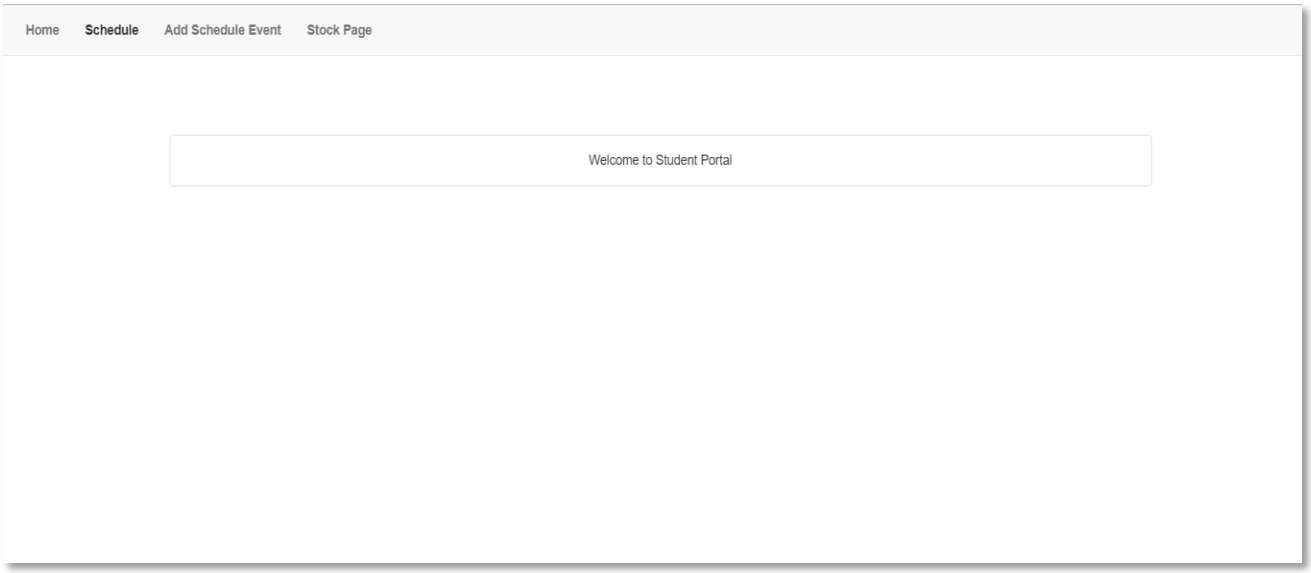
9. The project setup is now complete, and you are ready to start the server.

III. To start the server, type the following command:

```
node createServer.js
```

This starts the server and binds it to port 9001. Now, using in your browsers URL bar (i.e., address bar), type: **http:// localhost:9001**

The following page should be displayed (below, and shown again in the screenshots below):



The following files are provided for this assignment:

You will only need to add code to the files highlighted in red below.

1. **createServer.js**: This file contains the partially complete code for the node.js server.
2. **client/index.html**: Home page for this application.
3. **client/schedule.html**: Page which displays the list of schedule events. You need to fill in the TODO which would send a GET request to the Node.JS server via AJAX to fetch the data in the file ***schedule.json*** and then dynamically add the data to display a table on the schedule..html page.
4. **client/addSchedule.html**: Form to add details about a new place. When the form is submitted it will send a POST request with the data entered on the form to your Node.JS server
5. **client/stock.html** : Page which takes in the company as request and displays the stock report. You need to fill in the TODO to make a get request using the **alphavantage API**. Details are discussed further below.
6. **schedule.json**: This file contains the list of schedule events in JSON format.

3 Functionality

Note: It is advisable to complete the code changes for server before changing the code for client. All the server endpoints (APIs) can be tested using POSTMAN or curl.

Client

All the resources related to client have been provided in the client folder. The client folder has four HTML files (`index.html, schedule.html, addSchedule.html, and stock.html`).

`schedule.html` has a table (***id = "scheduleTable")*** whose body is empty. You need to add code to *the TODO section* that dynamically populates the contents of the table after getting the list of schedule details (a string containing the items in the table in JSON format) from the server. You need to implement the following functionality in schedule.html file:

1. Request the list of schedule entries from the ***getSchedule*** endpoint of your Node.js server using AJAX with the GET method.
2. Upon successful completion of the asynchronous AJAX GET request, your Node.js server will return the list of schedule entries.
3. Use the response returned to dynamically add rows to the table with the id ***'scheduleTable'*** present in schedule.html page (Create a JSON object out of the list returned and then build/render an HTML table to display the entries in the schedule)
4. You can either jQuery or JavaScript (or a mix of both) to achieve this.

Tasks for Stock.html

1. Visit https://www.alphavantage.co/ and create a Free API key.
2. On the button "Click", use AJAX to do a GET request to the `TIME_SERIES_DAILY` endpoint of the alphavantage API for the company that was selected in the dropdown list box (select).
3. Display the generated JSON response in textareas as indicated in the screenshot below.
4. You can use of jQuery or JavaScript (or a mix of both) to achieve this.

Server

When the server starts, it listens for incoming connections on port 9001. This server is designed to handle only GET and POST requests.

GET requests:

1. The server has been designed to serve four different HTML pages to clients: *index.html, schedule.html, addSchedule.html and stock.html*

2. The server can also read and write to the list of schedule entries (in JSON format) by accessing *schedule.json* file.

3. GET request for the index.html: The code for this has already been provided to you in *createServer.js* file where the server is listening on the endpoint "/" and "/index.html". **You do not need to add any code for this.**

4. GET request for the *schedule.html* page:
    a. When the **Schedule Tab** is clicked on the browser, a request is sent to the server to fetch the schedule.html file.
    b. You need to write code in createServer.js to listen for requests to the Server's endpoint "./schedule.html" and return the file 'client/schedule.html' to the client

5. GET request from the *schedule.html* page:
    a. You need write code to listen on an endpoint for the GET request from schedule.html (the request will be seeking the contents of the ***schedule.json*** file)
    b. You need to write code in createServer.js to **fetch json data** from the **schedule.json** file and return the json **data to *schedule.html* (which will then be parsed and displayed by schedule.html** in table format)

6. GET request for the addSchedule.*html* page:
    a. When the **Add Schedule Event Tab** is clicked on the browser, a request is sent to the server to fetch the ***addSchedule.html*** file.
    b. You need to write code in createServer.js to listen to for requests to the endpoint ***"./addSchedule.html"*** and return the file: ***'client/addSchedule.html'*** to the requesting client.

7. GET request for the stock.*html* page:
    a. When the **Stock Tab** is clicked on the browser, AJAX should be used to send a GET request is sent to the your server to fetch the file: ***addSchedule.html***
    b. You need to write code in createServer.js to listen for requests to the endpoint ***"./stock.html"*** and return the ***'client/stock.html'*** file to the requesting client.

8. GET request for any other resource: If the client requests any resource other than those listed above, the server should return 404 error. The implementation is already provided in the code we've provided for you.

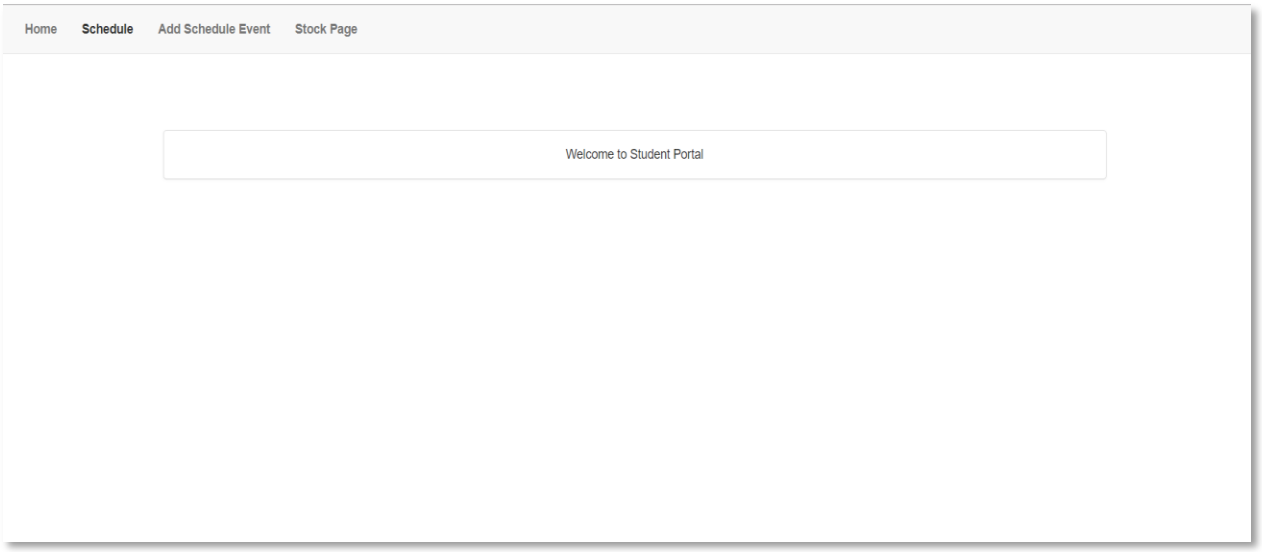
9. POST requests:
    - The server should process the form data posted by client. The form we've provided, in the file ***addSchedule.html*** enables a user to enter details about a new schedule event and update the list of schedule events. The user enters the **Event Name**, **Location**, **the Day of week**, **Start -time,** and **End-time** in the form.
    - Details for a few events are pre-populated in the ***schedule.json*** file. Your job is to add code that appends the details of a new schedule event sent via a POST of the data entered on the form to this file and redirect the user to the ***schedule.html*** page after successful addition of the new course.
    - To accomplish this, your server needs to listen for requests to the "/postScheduleEntry" endpoint for a POST request from the ***addSchedule.html*** file.

- You need to write code to
  i. read the data "posted" (i.e., the data the user has entered on each field) to the form)
  ii. add the new information to schedule.json file,
  iii. and redirect the file **schedule.html**.

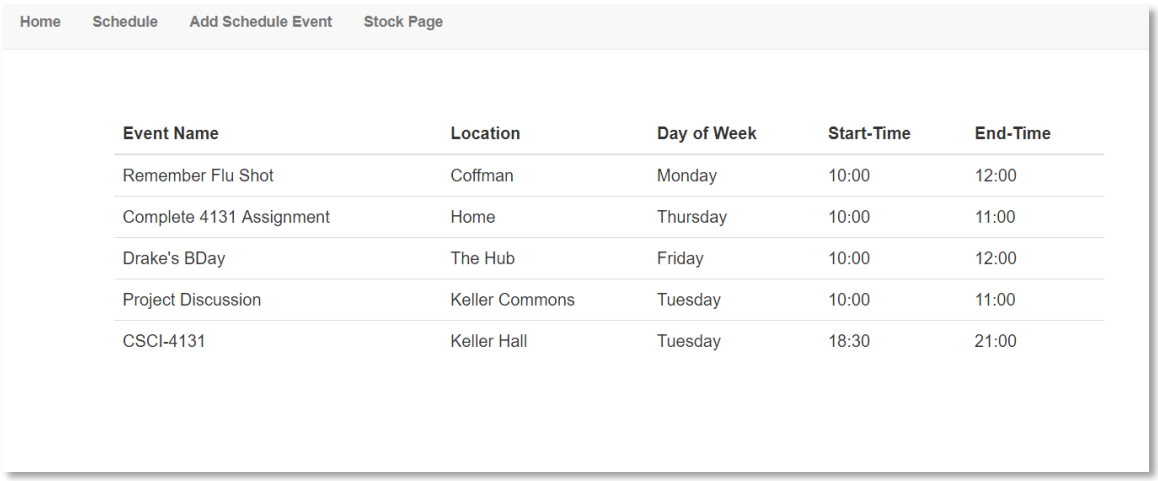
The code for redirection is 302.
Ensure that the newly added data does not change the format of the schedule.json file.

6 Screenshots

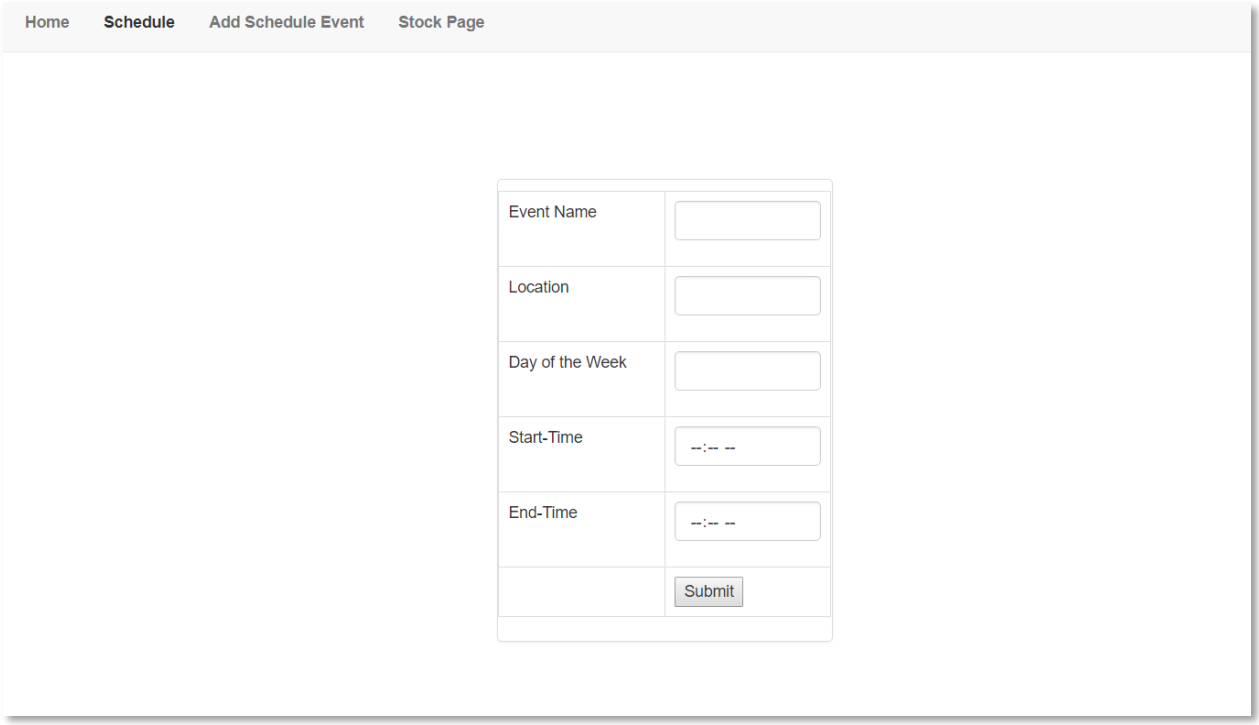
index.html (Should be displayed when you type: **http:// localhost:9001** in your browser's URL bar after starting your Node.js server)



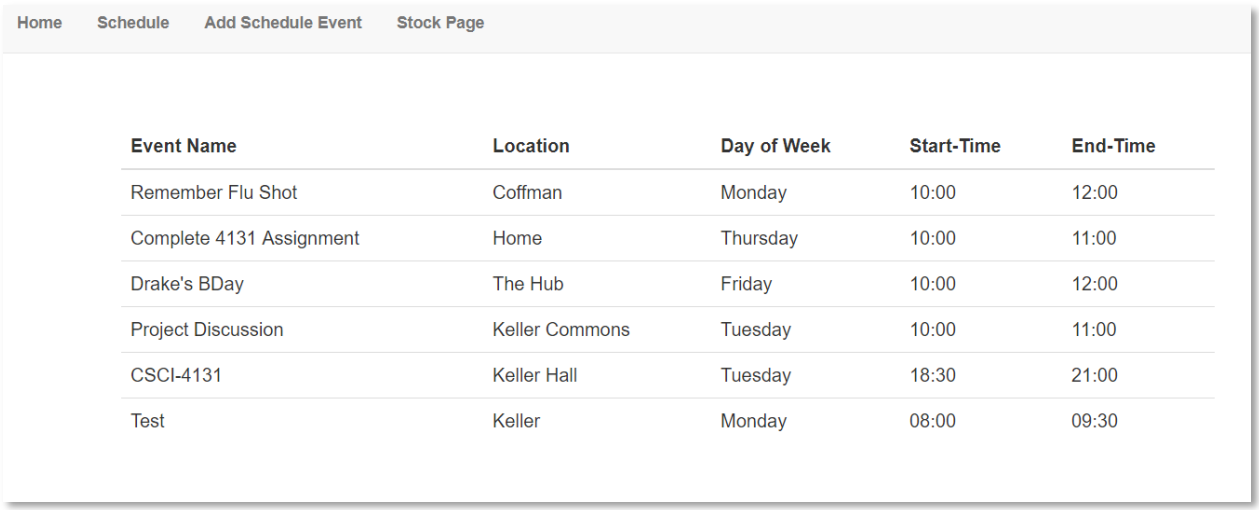
Initial display for schedule.html (displayed when user selects **Schedule** menu item**)**



Add details for a new schedule event (Form displayed when **Add Schedule Event** is selected**).**

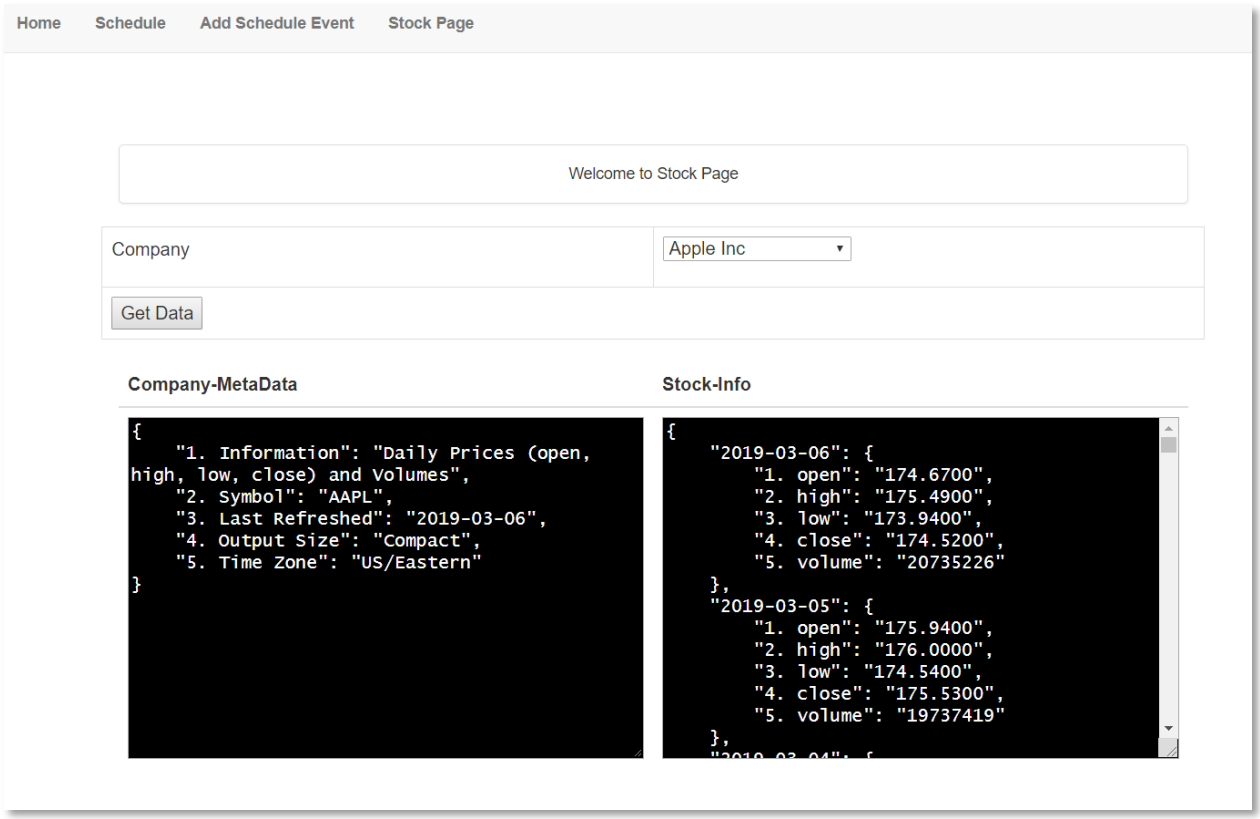


schedule.html page after adding a new schedule event (after completed form is submitted with the information shown in the last row of the Schedule displayed below)



Event Name	Location	Day of Week	Start-Time	End-Time
Remember Flu Shot	Coffman	Monday	10:00	12:00
Complete 4131 Assignment	Home	Thursday	10:00	11:00
Drake's BDay	The Hub	Friday	10:00	12:00
Project Discussion	Keller Commons	Tuesday	10:00	11:00
CSCI-4131	Keller Hall	Tuesday	18:30	21:00
Test	Keller	Monday	08:00	09:30

Stock.html after Apple Inc is selected and the Get Data button is "Clicked".



4 Submission Instructions

Zip your entire project directory - and the name of the zipped folder should be your **x500id_nodejs**.

5 Evaluation

Your submission will be graded out of 100 points on the following items:

1. *schedule.html* is successfully returned by the server (**15 points**).
2. *addSchedule.html* is successfully returned by the server (**15 points**).
3. Client successfully gets the list of schedule events from the server. The schedule events are dynamically added to the table present in schedule.html page. (**30 points**)
4. *POST endpoint* successfully adds the details of the new schedule entry to schedule.json file (**30 points**).
5. User is redirected to *schedule.html* page after successful addition of a new place (**10 points**).
6. **Stock.html is has the required functionality (10 points) Bonus**