LAB 10 - API

Objective: Students will practice

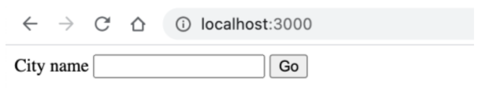
- Node.js File System module
- HTTP request and HTTP response message with Express.js framework
- Collect form data with the body-parser middleware
- Axios framework to work with APIs

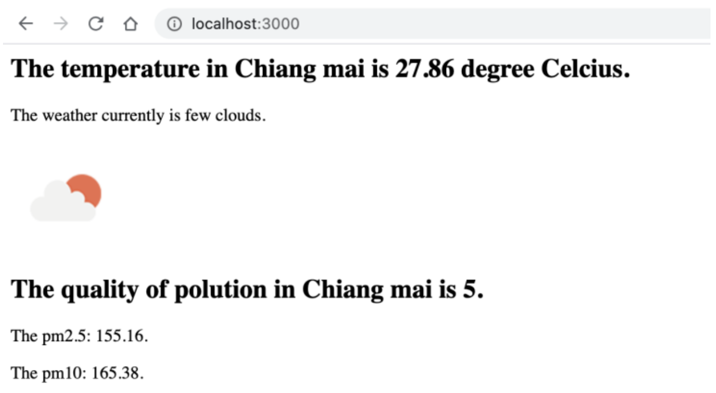
Lab instruction

- The LAB10 instruction and lab resources are posted on MS Teams channel LAB10 – APIs of subject 953262 (your section).
- Download the zipped file of resources-lab10.
- There are 2 assignments according to the LAB10 sheet posted on the channel.
- The LAB10 is worth 20 points in total.
- Score criteria: full point (for output correct); -1 (for output does not correct); -1 (for not follow problem constraint)
- **Assignment Submission:**
    - Upload your solutions to MS Team assignments. The submission later than the 'due date' will get 50% off your score. At the 'close date', you cannot submit your assignment to the system.
    - Post your 'Name and Student ID' on the MS Teams channel 'LAB10 – APIs. You should also be prompt for TA calling to verify your work on your computer.

1. Current weather and air pollution application (10 points)

This is a Web application working with the OpenWeatherMap APIs, from https://openweathermap.org/api, that allows users to search for the current weather and air pollution of a particular city. **You need to sign-up for the OpenWeatherMap account to obtain the application key (app-id).** The UIs of the project are shown below.





The temperature in Chiang mai is 27.86 degree Celcius.

The weather currently is few clouds.

The quality of polution in Chiang mai is 5.

The pm2.5: 155.16.

The pm10: 165.38.

Do the following steps.

1. Open Terminal and use the command line to make a new folder called **WeatherProject** inside your directory.
2. Change Directory to this new folder
3. Inside the WeatherProject folder, create two new files called **app.js** and **index.html**, respectively.
4. Set up the project folder with a new NPM package
5. Use NPM install the **express, body-parsers** and **axios** modules
6. Require **express, body-parser and axios** in your **app.js**
7. Setup express and boby-parser
8. Spin up our server on port **3000** with app.listen
9. Run server with **nodemon**

10. Modify the **index.html** with the following code.

```
<body>
    <form action="/" method="post">
        <label for="cityInput">City name</label>
        <input type="text" name="cityName">
        <button type="submit">Go</button>
    </form>
</body>
```

11. Inside the **app.js**, **add a '/' route GET method** to send the **index.html** file as the response.
12. **Add another '/' route POST method** to parse data from the index.html.
    - The callback function of the `app.post()`
        - Use **axios(config)** to make a request GET massage to obtain a json data back form the current weather api.
        - Example of a GET request for remote json data in node.js

```
const axios = require('axios');
axios({
        method: 'get',    //default
        url: 'https://api.kanye.rest/',
        responseType: 'json'   //default
})
 .then(response => {   console.log(response.data); })
 .catch(error => {   console.log(error); });
```

support Promise fulfiled

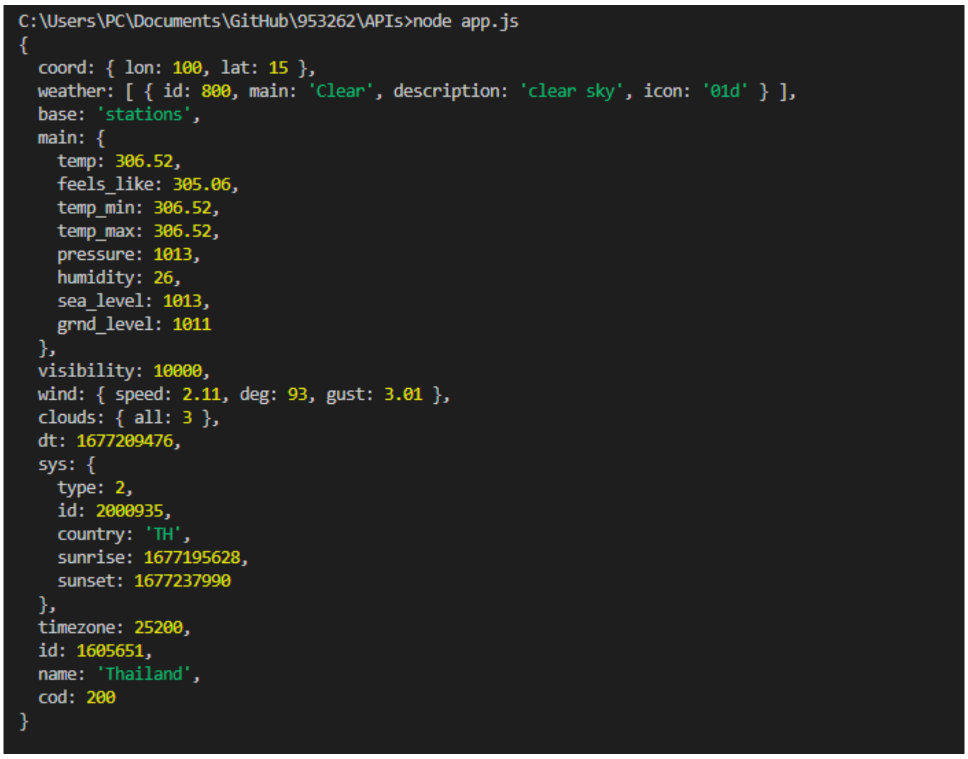
support Automatic JSON data handling in response

support Promise rejected

Axios reference: https://axios-http.com/docs/intro

Promise reference: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

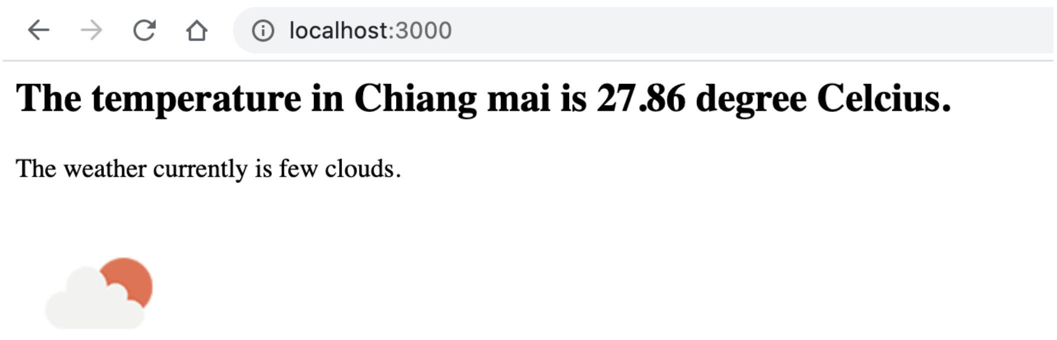
- **The endpoint of the Current Weather by city name api** is api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}. The link to this API is https://openweathermap.org/current#name
- Examine the logged current weather data in the Promise fulfiled `.then(response => console.log(response.data))` ; you will see the following output.



```
C:\Users\PC\Documents\GitHub\953262\APIs>node app.js
{
  coord: { lon: 100, lat: 15 },
  weather: [ { id: 800, main: 'Clear', description: 'clear sky', icon: '01d' } ],
  base: 'stations',
  main: {
    temp: 306.52,
    feels_like: 305.06,
    temp_min: 306.52,
    temp_max: 306.52,
    pressure: 1013,
    humidity: 26,
    sea_level: 1013,
    grnd_level: 1011
  },
  visibility: 10000,
  wind: { speed: 2.11, deg: 93, gust: 3.01 },
  clouds: { all: 3 },
  dt: 1677209476,
  sys: {
    type: 2,
    id: 2000935,
    country: 'TH',
    sunrise: 1677195628,
    sunset: 1677237990
  },
  timezone: 25200,
  id: 1605651,
  name: 'Thailand',
  cod: 200
}
```

- Extract the following **weather data properties: coord.lon, coord.lat, weather.description, and weather.icon and** write a code to send the HTML content back via the **res object** of the express app. As you want to send the HTML content, you need to set Content-type of the Header's res object. `res.setHeader("Content-type", "text/html");` and use `res.write()` and `res.end()` to send multiple times to provide successive parts of the body. Note that the weather icons are stored on the server at the following address. `http://openweathermap.org/img/wn/" + icon +"@2x.png>`
- Don't forget to catch the Promise rejected to see an error if it happens. `.catch(err => console.log(err))`

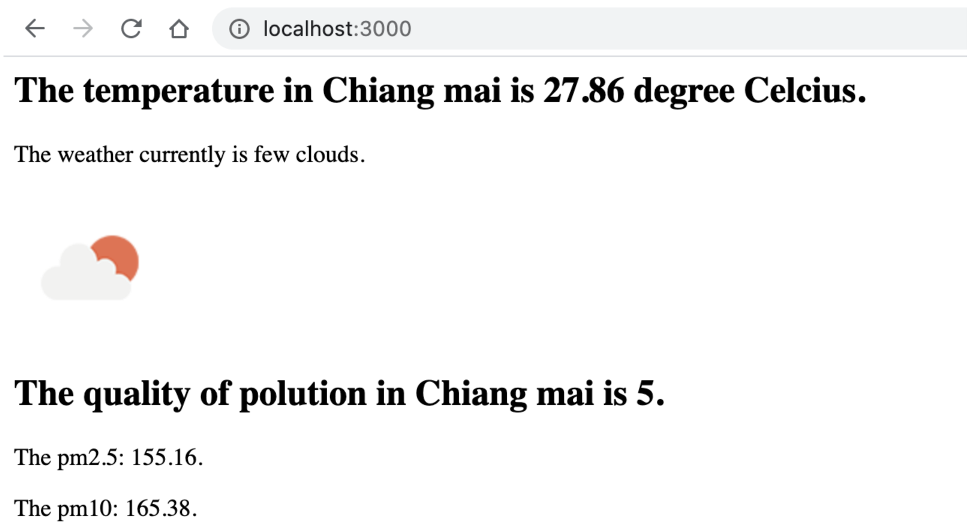
The expected output after the user submits the city name form is the following.


localhost:3000

The temperature in Chiang mai is 27.86 degree Celcius.

The weather currently is few clouds.

13. Continue working inside the **the callback function** of `axios()` to call another the Current pollution api with axios(). The endpoint of the api is http://api.openweathermap.org/data/2.5/air_pollution?lat={lat}&lon={lon}&appid={API key} The link to this API is https://openweathermap.org/api/air-pollution
14. Examine the logged current pollution data in the Promise fulfiled `.then(response => console.log(response.data))` and extract the following pollution properties: **api, pm2_5, and pm10.** Write the code to send the data back as the res object of the express app. Don't forget to catch the Promise rejected to see an error if it happens. `.catch(err => console.log(err))`
15. The expected output after the user submits the city name form is the following.


localhost:3000

The temperature in Chiang mai is 27.86 degree Celcius.

The weather currently is few clouds.

The quality of polution in Chiang mai is 5.

The pm2.5: 155.16.

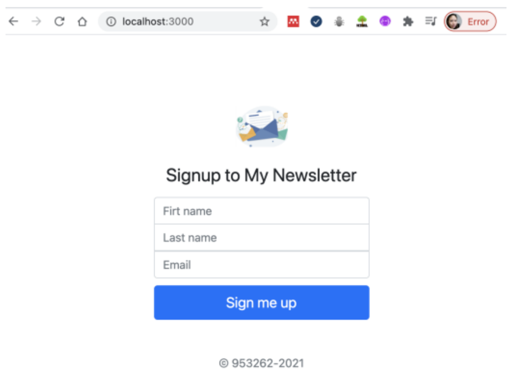
The pm10: 165.38.

Save the folder as **Q1 folder** and submit it to MS Teams.

2. Newsletter-Signup (10 points)

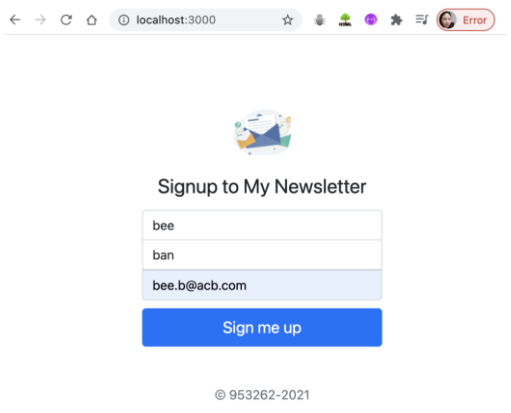
This project is to build a Newsletter-Signup Web application connected to the Mailchimp API to add a new subscriber to the Mailchimp server. The Web application allows any user to sign up for your newsletter. The following is a set of UIs.

Successful scenario


localhost:3000

Signup to My Newsletter

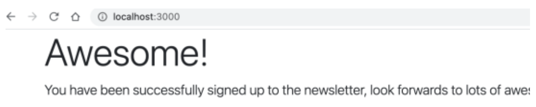
© 953262-2021


localhost:3000

Signup to My Newsletter

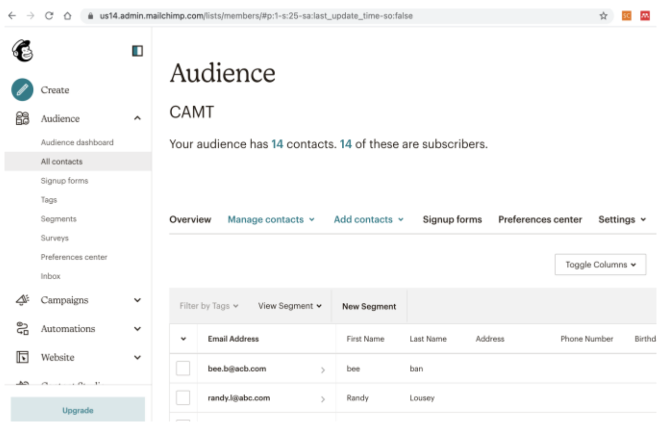
bee
ban
bee.b@acb.com

© 953262-2021


localhost:3000

Awesome!

You have been successfully signed up to the newsletter, look forwards to lots of awe


us14.admin.mailchimp.com/lists/members/#p:1-s:25-sa:last_update_time-so:false

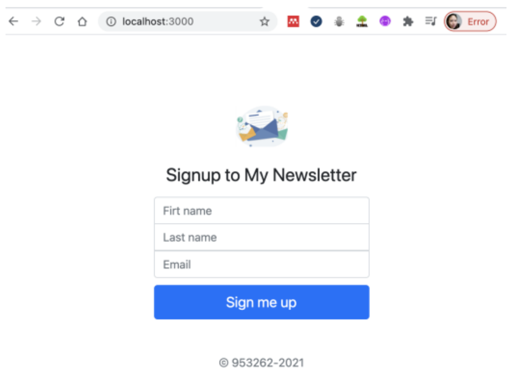
Audience

CAMT

Your audience has 14 contacts. 14 of these are subscribers.

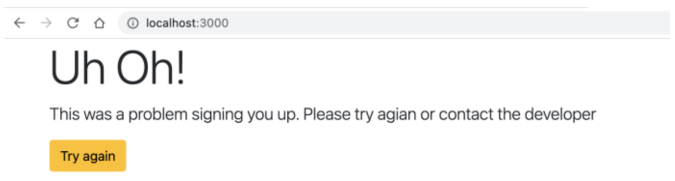
Email Address	First Name	Last Name	Address	Phone Number	Birthd
bee.b@acb.com	bee	ban			
randy.l@abc.com	Randy	Lousey			

Unsuccessful scenario


localhost:3000

Signup to My Newsletter

© 953262-2021


localhost:3000

Uh Oh!

This was a problem signing you up. Please try agian or contact the developer

Your job is to complete the following tasks.

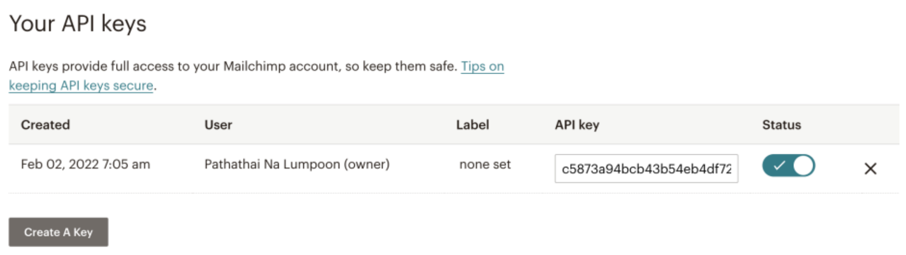
1. **Sign up for the Mailchimp accoun**t from https://mailchimp.com/ to obtain an API key.

   During the signup process, choose the **Free plan option**, **No option for having a list of email subscribers**, and **Not right now for finding your marketing path**. After you log in successfully, you will be taken to the Mailchimp dashboard. Do the following steps to obtain an API key.

   a. Click your **profile icon** (located at the bottom left corner) and then select **Account or click at this link** https://us14.admin.mailchimp.com/account/

   b. On your **Account page**, select the **Extras** tab and then go to the **API keys**.

   c. **Click the Create a key** button. The following picture is an example of the generated API key.



Your API keys

API keys provide full access to your Mailchimp account, so keep them safe. Tips on keeping API keys secure.

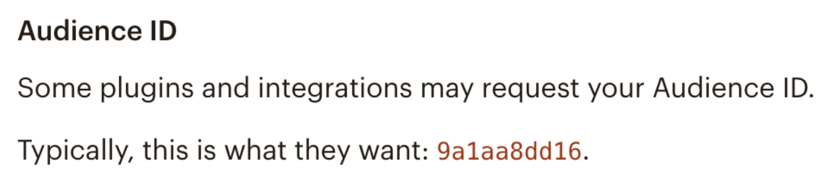
Created	User	Label	API key	Status
Feb 02, 2022 7:05 am	Pathathai Na Lumpoon (owner)	none set	c5873a94bcb43b54eb4df72	✓

   d. **Copy and paste your API key somewhere** to be ready to use when you code.

2. Get your **list_id** that you will use later (**in Step 4**) for calling **the Batch subscribe or unsubscribe API**. This list_id is the unique number that is used to identify the list of your subscribers (members or audiences). Do the following steps.

   a. On the **Mailchimp dashboard**, select the **Audience** on the left side menu and go to the **All contacts**.

   b. On the **Audience contacts page**, select the **Setting** tab and go to **Audience name and defaults.** You will see your **unique id** for your audiences (so-called list_id). The following picture is an example of the list id.



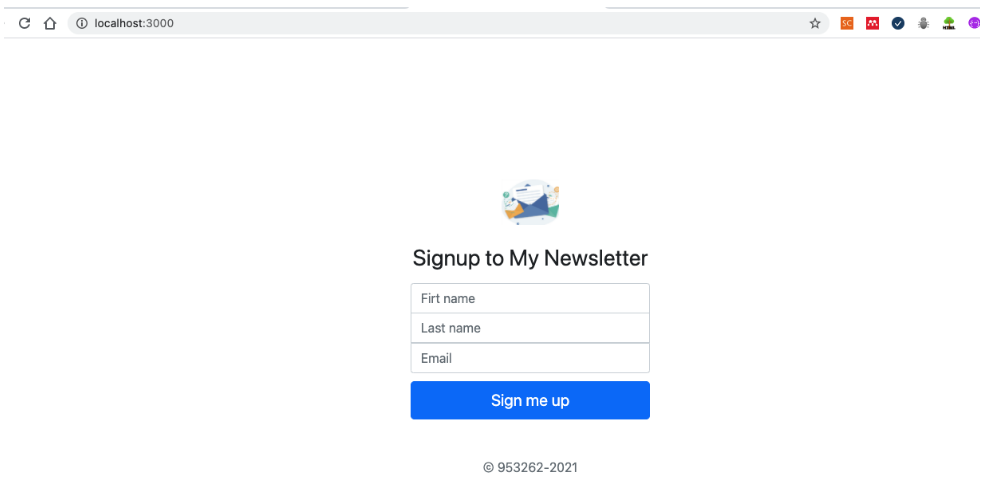
Audience ID

Some plugins and integrations may request your Audience ID.

Typically, this is what they want: `9a1aa8dd16`.

   c. **Copy and paste your list id somewhere** to be ready to use when you code.

3. Set up the sign-up page. Do the following steps.
   a. Download and unzip the LAB9-resource file. Inside the folder, there are **signup.html**, **success.html**, **failure.html**, and **a public directory** that contains **css and images folders.** <u>Note that the public folder keeps the static files to be loaded to the server.</u>
   b. Change the folder name from LAB9-resource file to **Newsletter-Signup**
   c. Change directory path to this folder
   d. Inside the Newsletter-Signup folder, create new file **app.js.**
   e. Create **package.json** file and install **express, body-parse and axios** modules.
   f. Require the **express, body-parser, axios** modules in the app.js
   g. Set up express and body-parser
   h. Set up serving static files in express with the following code
      ```
      app.use(express.static("public"));
      ```
      This makes Express looks up the files relative to the static public directory, so the name of the static public directory is not part of the URL.
   i. Make server running on port 3000 and run the server with **nodemon**
   j. **Complete the signup.html** and **create a '/' route get** to send the signup.html file as the response. The expected output is as follows.



4. Posting data to Mailchimp's servers via their API

   The **Mailchimp API** that you are going to use is the **Batch subscribe or unsubscribe API in Lists/Audience category under the Mailchimp Marketing API**. A link to the API is https://mailchimp.com/developer/marketing/api/lists/batch-subscribe-or-unsubscribe/

   The detail of the Batch subscribe or unsubscribe API is as follows.

   Name: Batch subscribe or unsubscribe API

   Description: Arrange subscription of a list of members. For example, adding new subscribers, with each member's signup information to the server.

   Endpoint: https://<dc>.api.mailchimp.com/3.0/

   <dc> is your server id in which you can find it from your App Key. For example, my App key is c1457a94fdg43b54ec23es2714set58-**us14**, so **my dc is us14**.

   Auth: 'app-key'

   API: POST Endpoint/lists/{list_id}

   Path : list_id

   list_id is your unique ID for the list members/audiences. You are already gotten your list_id from Step 2.

Body parameter: [members]

[members] is an array of objects, each representing an email address and the subscription status for a specific list. There are many member properties (e.g, email address, email_type, stutus, merge_fields, interest and etc) that you can specify for each member. However, according to your signup form (in Step 3), you need only 3 properties of each member object. **The example of a member object is as follows**.

```
members: [
                {
                        email_address: email,
                        status: "subscribed",
                        merge_fields: {
                                FNAME: firstName,
                                LNAME: lastName,
                                },
                },
        ],
```

Success response: HTTP Status 200 - Batch update List members

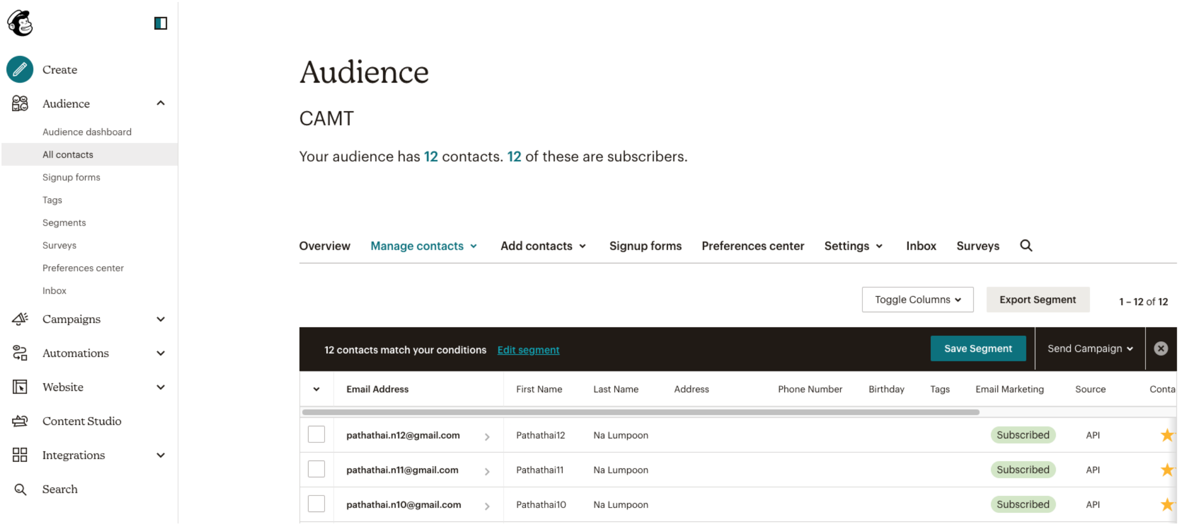
[new_members], [updated_ members], [errors] and etc. You can find an example of a JSON response from https://mailchimp.com/developer/marketing/api/lists/batch-subscribe-or-unsubscribe/

Do the following steps to post data to Mailchimp's servers via their API.

a. **Create a '/' route POST method** to parse data from the signup form. Store the signup data using the **above BODY parameter format.**
b. **To call the API with data**, use **axios(config)** to make a request POST massage to create new subscriber to the {list_id} form the the API POST Endpoint/lists/{list_id}
c. Set a **config object** to contain **url, method, auth and data** properties like example below.

```
{
        url: < Endpoint/lists/{your list_id}> ,
        method: "post",
       auth: {
        username: <anyname>,
        Password: <your app-key>
        },
        Data: {
                members: [
                        {
                                email_address: email,
                                status: "subscribed",
                                merge_fields: {
                                        FNAME: firstName,
                                        LNAME: lastName,
                                        },
                        },
                ],
        }
    }
```

d. Come back to write a Promise fulfiled function of the axios() to log the response object. If calling the API successfully from the signup.html, the [new_member] is logged in the console, and your list of audiences on the Mailchimp's server is updated. The following is the example of the Mailchimp audience system showing the updated list of audiences that are sent from my Newsletter signup Web app.



Audience

CAMT

Your audience has 12 contacts. 12 of these are subscribers.

1 – 12 of 12

12 contacts match your conditions

Email Address	First Name	Last Name	Address	Phone Number	Birthday	Tags	Email Marketing	Source	Conta
pathathai.n12@gmail.com	Pathathai12	Na Lumpoon					Subscribed	API	★
pathathai.n11@gmail.com	Pathathai11	Na Lumpoon					Subscribed	API	★
pathathai.n10@gmail.com	Pathathai10	Na Lumpoon					Subscribed	API	★

5. Adding success and failure pages. Do the following steps.

   a. Check the status code and the data.error_count of the response object returned. if the status code 200 and error_count 0 are returned, the response object of the express app returns success.html file. If not, the response object of the express app returns failure.html file. Don’t forget to catch the Promise rejected to see an error if it happens. `.catch(err => console.log(err))`
   b. **Add another `‘/failure’` route with POST method** to redirect to the root. Use the following code. `res.redirect('/');`
   c. Test your Web app for successful and failure scenarios. Note that to test failure scenario, comment on the line of code that sends data to the API.
6. Save the folder as **Q2 folder** and submit it to MS Teams.