Module 3: Asynchronous Programming

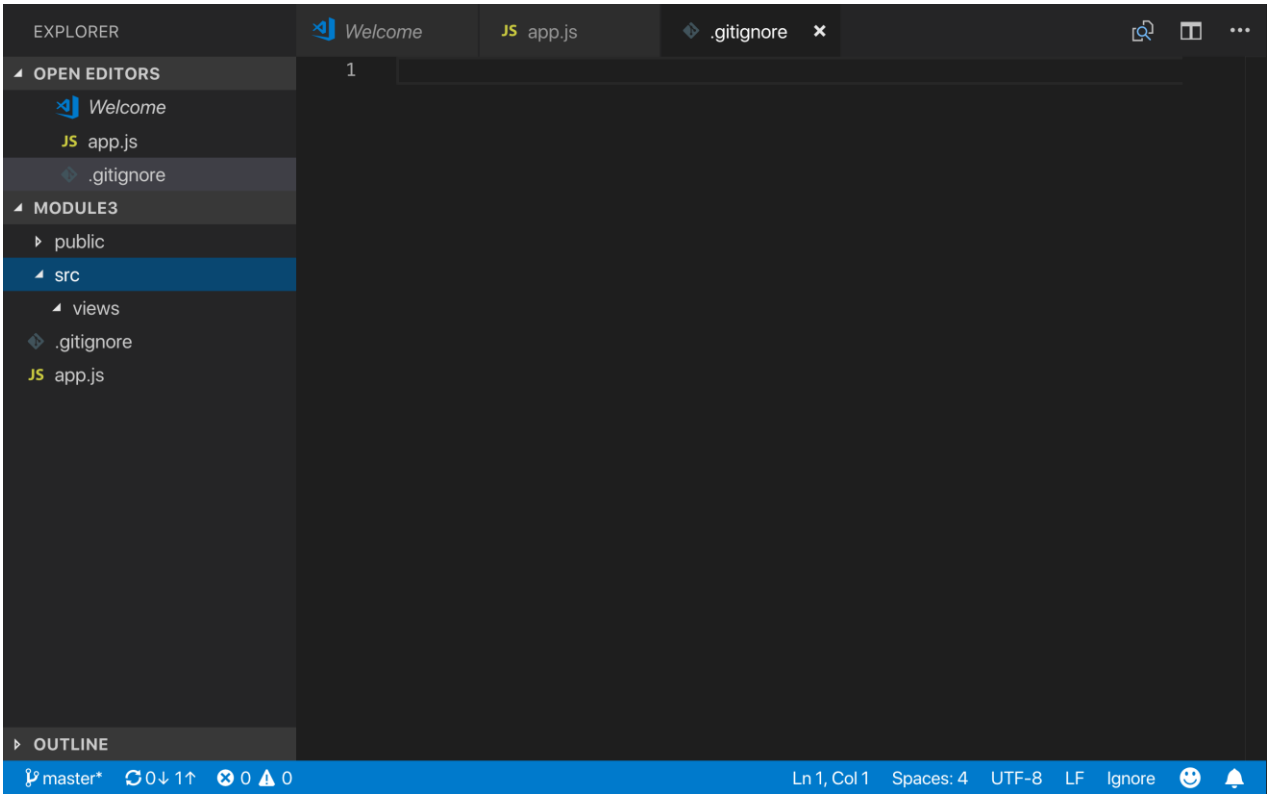
Demo Document 1

edureka!

Creating Weather Application Using Nodejs, Express, EJS, Promise And Callback

Api used is of OpenWeather : https://openweathermap.org/api

Step 1: Create new folder named **module3**. This time add 2 new folder named **public** and **src** for view and static files as well as added .gitignore file to stop pushing node_modules to GitHub after that navigate to the folder with terminal or command prompt.



Step 2: On terminal run command "**npm init**" and at the end write yes to create file a **package.json**

```
Avyaans-MacBook-Pro:module3 avi$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (module3) weatherapp
version: (1.0.0)
description: Node App with EJS
entry point: (app.js)
test command:
git repository:
keywords: Node EJS
author: Edureka
license: (ISC)
About to write to /Users/avi/Desktop/folder/EdurekaApp/module3/package.json:

{
  "name": "weatherapp",
  "version": "1.0.0",
  "description": "Node App with EJS",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "Node",
    "EJS"
  ],
  "author": "Edureka",
  "license": "ISC"
}


Is this OK? (yes) yes
```

Step 3: Install packages locally in the folder

Package name: **Express, EJS and Request**

```
Avyaans-MacBook-Pro:module3 avi$ npm install express ejs request
```

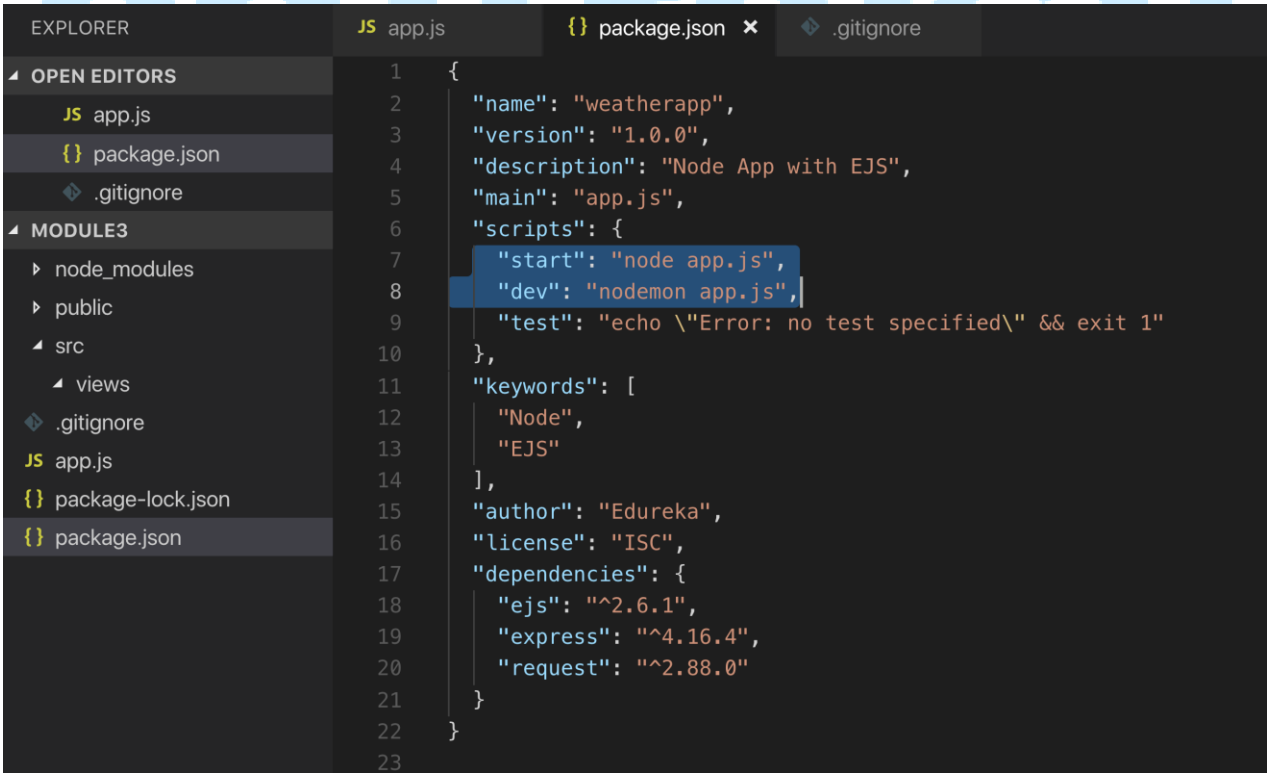
Step 4: Wait for package to install successfully and then navigate to Text Editor (Visual studio code) for implementation.

```
npm notice created a lockfile as package-lock.json. You
 should commit this file.
npm WARN weatherapp@1.0.0 No repository field.

+ request@2.88.0
+ ejs@2.6.1
+ express@4.16.4
added 92 packages from 88 contributors and audited 185
packages in 6.485s
found 0 vulnerabilities
```

Step 5: In package.json add “start” and “dev” command to run application in production and Development mode

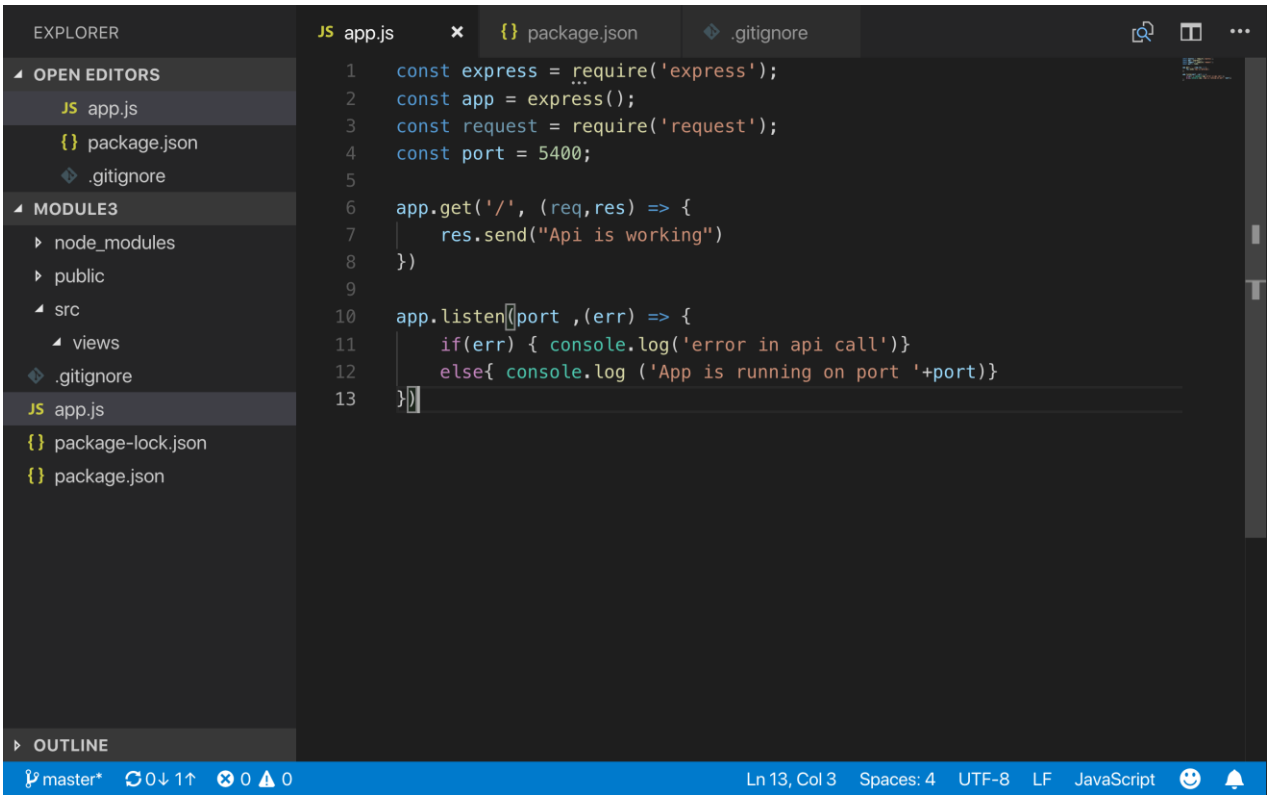
For Production use “**npm start**”

For Dev use “**npm run dev**”



Step 6: Create Server using express and use port number here we are using "**5400**"

And create one default Route as test route of the application. After creating navigate to terminal for running application.



Step 7: Run app using " npm run dev" and this will allow app to run with nodemon.

It keep on detecting the changes and restart our server this help in fast development.

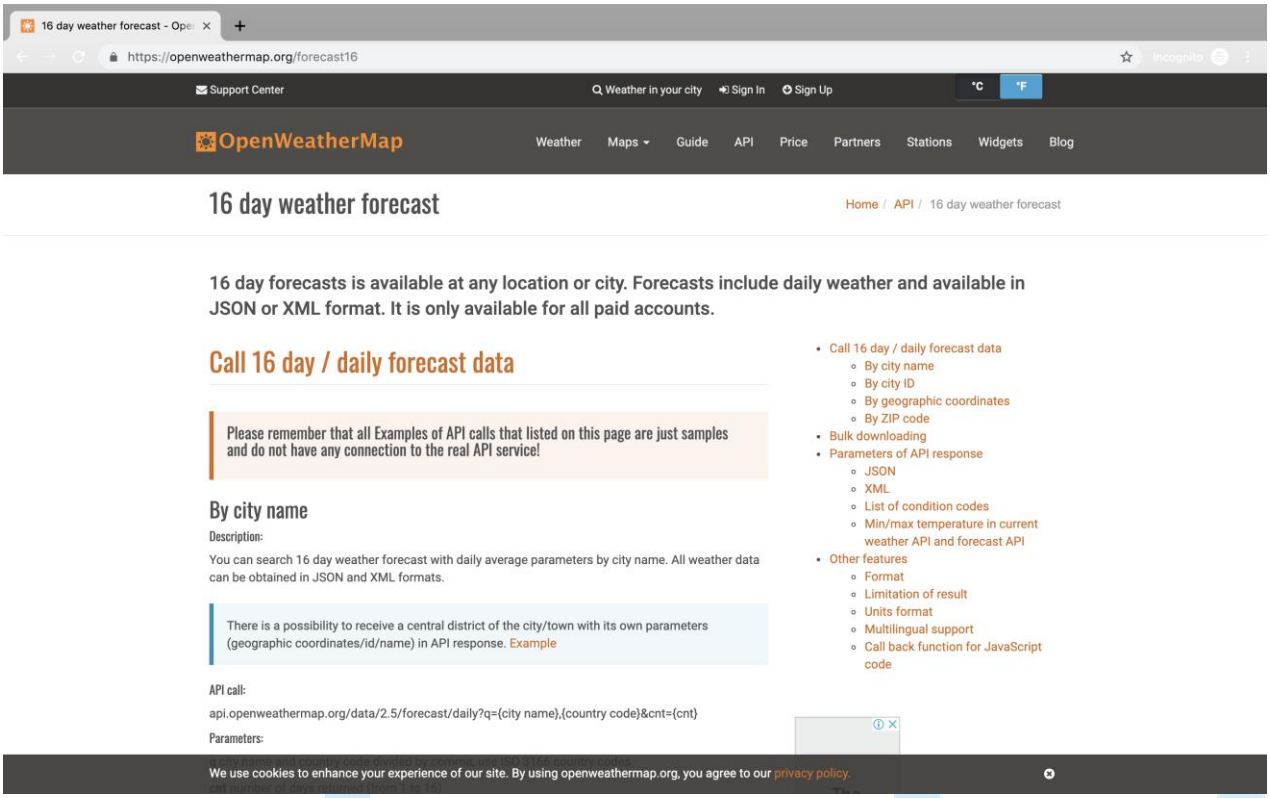
```
Avyaans-MacBook-Pro:module3 avi$ npm run dev

> weatherapp@1.0.0 dev /Users/avi/Desktop/folder/EdurekaApp/module3
> nodemon app.js

[nodemon] 1.18.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
App is running on port 5400
```

Step 8: Test App on browser using "http://localhost:5400" I will run default router with output.

Step 9: Create Account on open weather map website https://openweathermap.org/api

and get **api keys** for limited number of hits

Note: You must wait for few hours to get your key activated.



Step 10: Provide api url with key in your app server

Url Sample:
http://api.openweathermap.org/data/2.5/forecast/daily?q=London&mode=json&units=metric&cnt=5&appid=**your_api_key**

```
const express = require('express');
const app = express();
const request = require('request');
const port = 5400;

const url = "http://api.openweathermap.org/data/2.5/forecast/daily?q=Lon
```

Step 11: With the use of ‘request’ module we will call the api in callback response we will get

```
const express = require('express');
const app = express();
const request = require('request');
const port = 5400;

const url = "http://api.openweathermap.org/data/2.5/forecast/daily?q=Lon

app.get('/', (req,res) => {
    res.send("Api is working")
})

app.get('/weather',(req,res) => {
    request(url, (err,response,body) =>{
        if(err){
            console.log(err);
        } else {

            const output = JSON.parse(body);
            res.send(output);
        }
    });
})

app.listen(port ,(err) => {
    if(err) { console.log('error in api call')}
    else{ console.log ('App is running on port '+port)}
```

data according to city provided. This call is without promises a simple node callback

Step 13: Now we are calling same api with promise. The difference is syntax as well as with promises its easy to handle response and error as we are waiting for Api to get resolve with the promise Objet.

Below is the function that call Api with promise

EXPLORER

OPEN EDITORS
- app.js code
- package-lock.json code

MODULE3
- code
  - node_modules
  - public
  - src
  - .gitignore
  - app.js
  - package-lock.json
  - package.json
  - screenshot

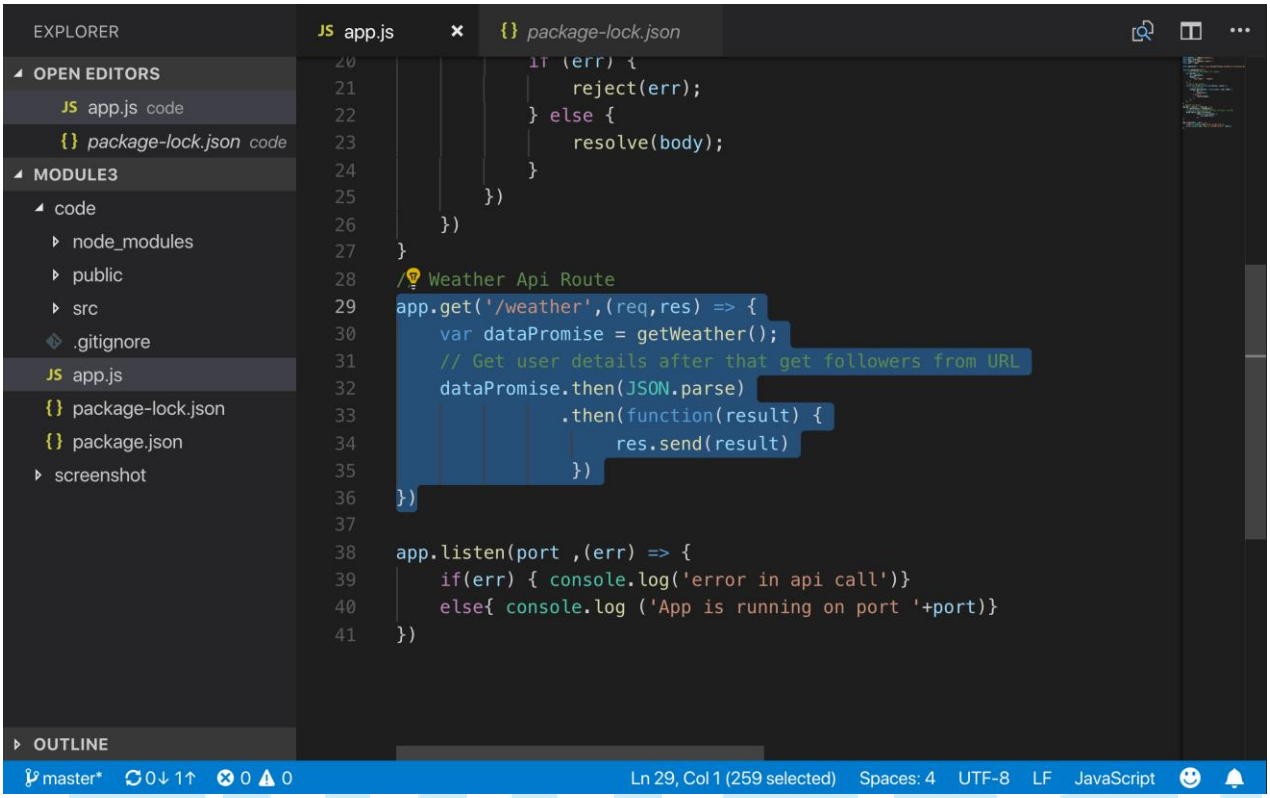
OUTLINE

app.js | package-lock.json

```javascript
const request = require('request');
const port = 5400;

const weatherUrl = "http://api.openweathermap.org/data/2.5/forecast/dail

function getWeather(url) {
    // Setting URL and headers for request
    var options = {
        url: weatherUrl,
        headers: {
            'User-Agent': 'request'
        }
    };
    // Return new promise
    return new Promise(function(resolve, reject) {
        // Do async job
        request.get(options, function(err, resp, body) {
            if (err) {
                reject(err);
            } else {
                resolve(body);
            }
        })
    })
}
// Weather Api Route
```

master* 0↓ 1↑ 0 0 | Ln 27, Col 2 (483 selected) | Spaces: 4 | UTF-8 | LF | JavaScript

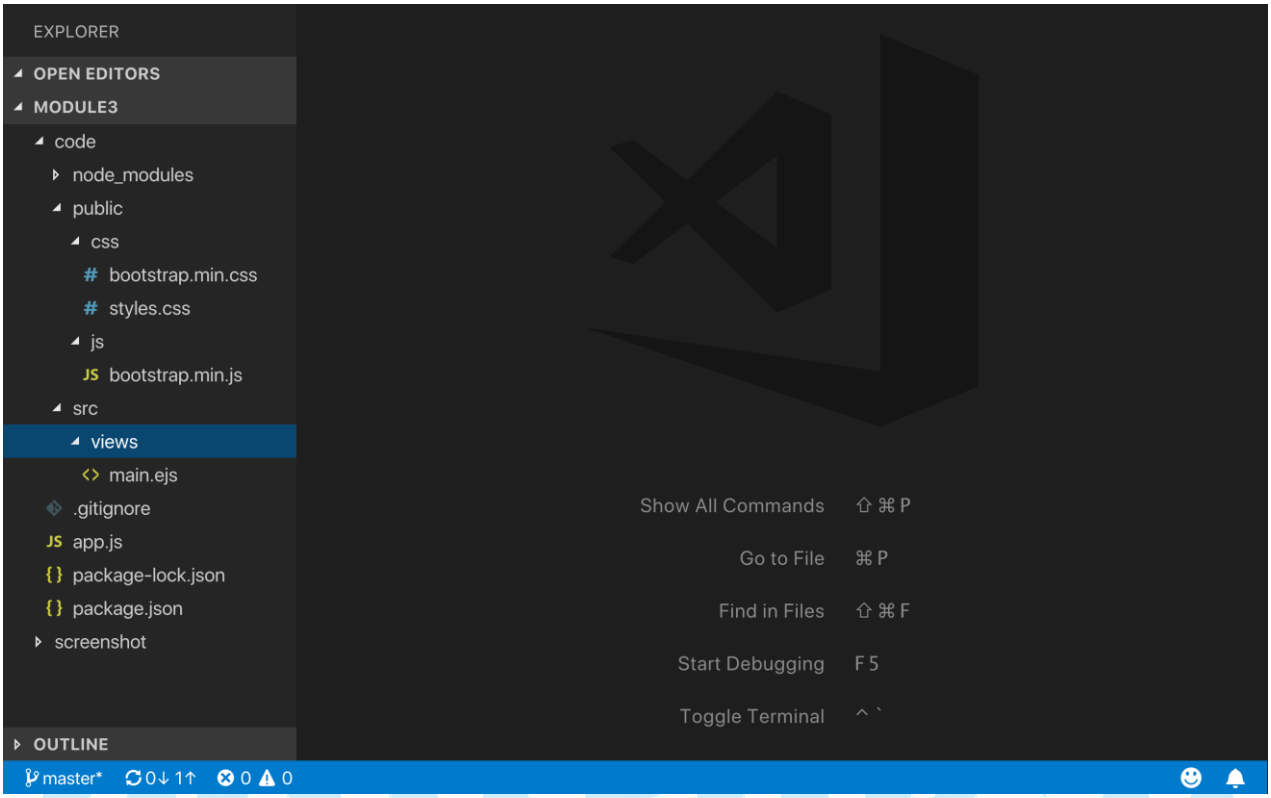
Step 14: Now as above function responses back data now we must resolve promise and get data out of promise



Step 15: Lets add view layer using EJS we will use EJS as middle ware and add path of public directory.

```
const express = require('express');
const app = express();
const request = require('request');
const port = 5400;

const weatherUrl = "http://api.openweathermap.org/data/2.5/forecast/dail

// Static file path
app.use(express.static(__dirname+'/public'));
// Html or rending Path
app.set('views', './src/views');
// View engine specification
app.set('view engine', 'ejs');


function getWeather(url) {
    // Setting URL and headers for request
    var options = {
        url: weatherUrl,
        headers: {
            'User-Agent': 'request'
        }
    };
    // Return new promise
    return new Promise(function(resolve, reject) {
        // Do async job
```

Step 16: Now we will create one folder for view as well as public folder where we can keep our



css and Js as well as any other static files.

Step 17: Instead of res.send we will use res.render to display data on view. Here will provide the path as well as send data as JSON.

EXPLORER

OPEN EDITORS
- app.js code
- main.ejs code/src/views

MODULE3
- code
  - node_modules
  - public
    - css
      - bootstrap.min.css
      - styles.css
    - js
      - bootstrap.min.js
  - src
    - views
      - main.ejs
  - .gitignore
  - app.js
  - package-lock.json
  - package.json
  - screenshot

OUTLINE

app.js | main.ejs

```
30                  } else {
31                      resolve(body);
32                  }
33              })
34          })
35      }
36      // Weather Api Route
37      app.get('/weather',(req,res) => {
38          var dataPromise = getWeather();
39          // Get user details after that get followers from URL
40          dataPromise.then(JSON.parse)
41                      .then(function(result) {
42                          res.render('main',{result,title:'***Weather App***'}
43                      })
44      })
45
46      //Weather Api Without promise
47      app.get('/weatherwithoutpromise',(req,res) => {
48          request(url, (err,response,body) =>{
49              if(err){
50                  console.log(err);
51              } else {
52
53                  const output = JSON.parse(body);
54                  res.send(output);
55              }
```

master* 0↓ 1↑ 0 0 | Ln 41, Col 15 (118 selected) | Spaces: 4 | UTF-8 | LF | JavaScript | 1

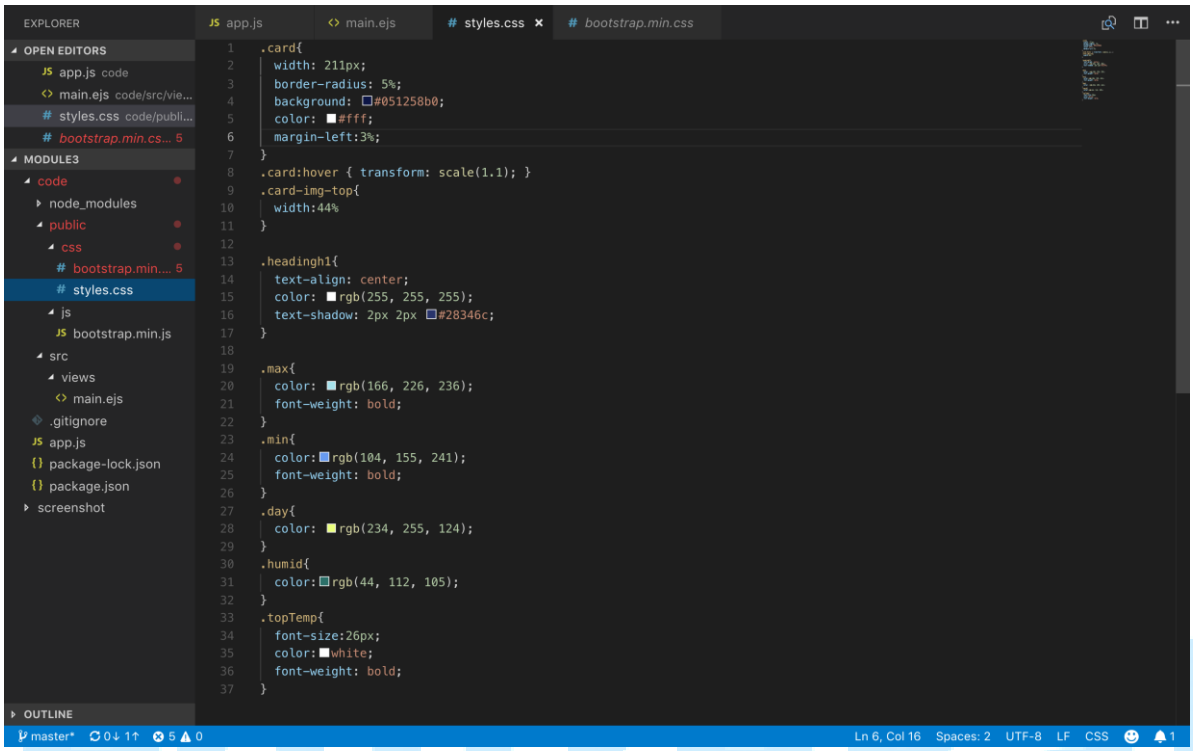
Step 18: We can include any of the css framework to make our app look. Here we are using Bootstrap for csss as well as adding For Loop using EJS to iterate the data

```
EXPLORER
OPEN EDITORS
    app.js code
    main.ejs code/src/vie...
    styles.css code/publi...
    bootstrap.min.cs... 5
MODULE3
    code
        node_modules
        public
            css
                bootstrap.min.... 5
                styles.css
            js
                bootstrap.min.js
        src
            views
                main.ejs
        .gitignore
        app.js
        package-lock.json
        package.json
        screenshot
OUTLINE
```

```
10      <link href="//netdna.bootstrapcdn.com/font-awesome/3.2.1/css/font-awesome.min.css" rel="stylesheet">
11
12
13  </head>
14  <body>
15      <div class="container">
16              <div >
17                  <center>
18                      <h1><%=title%></h1>
19                  </center>
20                  <h2>City: <%=result.city.name%></h2>
21              </div>
22              <hr/>
23              <div class="row">
24                      <%for(let i=0; i<result.list.length;i++){%>
25                      <div class="card col-md-2">
26                          <div class="row">
27                          <img class="card-img-top" src='https://openweathermap.org/img/w/<%=result.list[i].weath
28                          <span class="topTemp"><%=result.list[i].temp.day%>°C</span>
29                          </div>
30                          <div class="card-body">
31                          <span class="max"><%=result.list[i].temp.max%></span>   /
32                                  <span class="min"><%=result.list[i].temp.min%>°C</span>
33                              <h4 class="card-title"><%=result.list[i].weather[0].main%></h4>
34                              <p class="card-text">
35                                  <p class="day"><%=Date(result.list[i].dt)%> </p>
36
37                                  <p>Humidity: <%=result.list[i].humidity%>%</p>
38
39                              </p>
40                          </div>
41                      </div>
42                      <%}%>
43                  </div>
44
45      </div>
46
47  </body>
```

master* 0↓ 1↑ 5 0 Ln 42, Col 30 (1170 selected) Spaces: 4 UTF-8 LF HTML 1

Step 19: Below is CSS file its completely open to use any css properties as well as any style as you like.



Step 20: One everything is setup this will be final response and this response may vary according to your style.