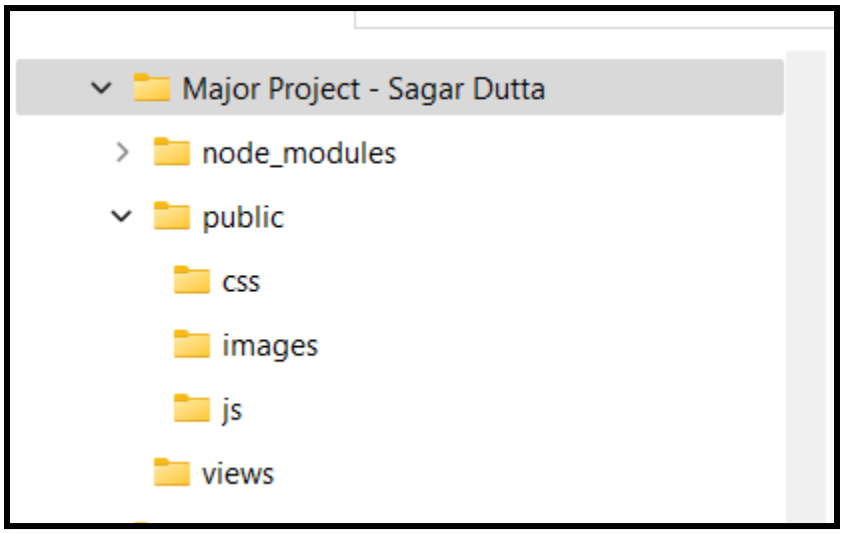
NAME: Sagar Dutta

BATCH: December 2022

PROJECT: Weather Forecast App - Major Project

Step 1: Create a folder. Let us name it '*Major Project - Sagar Dutta*'. Now create two sub-folder named *public* and *views* respectively. The public folder will contain static files like images, audio, .css, .js files. The views folder will contain .ejs files.



Step 2: Open command prompt or (new terminal in case of VS Code) and type *npm init* and a folder named *node_modules* will be created. Now install *express, body-parser, ejs, dotenv* packages.

NOTE: this project needs node js for running. It is not an only html file. To run the project, redirect the console to the ***Major Project - Sagar Dutta*** directory and then type '***node server.js***' in the terminal.

Step 3: Create a .ejs file named index (views -> index.ejs) and we will write HTML code inside it. The webapp will be divided into 3 parts, namely, header, body and footer.

Step 4: Include the cdn links to connect with cloudflare icons, bootstrap, jquery. Also create a file app.js (public -> js -> app.js) and another file named style.css (public -> css -> style.css). Now link these static files with index.ejs file using link tag inside head tag.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Weather Forecast App</title>
        <script src="js/app.js"></script>
        <link rel="stylesheet" type="text/css" href="css/style.css">
        <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.13.0/css/all.min.css" rel="stylesheet">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.2/css/all.min.css" />
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css">
        <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.1/dist/jquery.slim.min.js"></script>
        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js"></script>
    </head>
```

Step 5: Header section of the webapp contains a navbar and is created using the nav class of bootstrap. It contains a textfield and a related button for the user to enter the city name.

```
<body onload=" GetWeather(DefaultPlace)">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <!-- <a class="navbar-brand" href="#">LOGO</a> -->
          <i class="navbar-brand fas fa-umbrella"></i>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupported
        <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <h1>Weather Forecast App</h1>
        <table class="searchcitybar">
            <tr>
            <th><input class="form-control mr-sm-2" type="text" id="userInput"></h5></th>
            <td><input value="Search" class="btn btn-secondary" type="button" id="searchBtn" onclick="citySubmit()"></td>
            </tr>
        </table>

        </div>
    </nav>
    <div class="container">
        <div class="row">
            <div class="col-4" id="nowcastWeather">
                <div class="card" style="width: 100%; height: auto;">
                    <div class="card-body">
                        <h1 class="card card-title" id="cityName"></h1>

                      <table>
                        <% const date = new Date(); %>
```

Step 6: Body section is gridded into two subparts using row and column classes. The smaller subpart contains a card which will display the present day weather of the searched city while the other subpart will contain a table which will display the weather forecast in a tabular format.

Step 7: The footer contains the app name, designer's name and the links to the social handles of the designer.

```
        <footer>
            <div class="App-Logo">
                <i class="navbar-brand fas fa-umbrella"></i>
                <p id="App_Name"><b>Weather<span> Forecast </span>App</b></p>
            </div>
            <div class="DesignerInfo">
                <p><b>Designed and Developed By <span id="Name">Sagar Dutta</span></b></p>
            </div>
            <div class="SocialMedia">
                <a href="https://github.com/SagarDuttaSays" target="_blank" title="Github-Profile"><i
                        class="fa-brands fa-github"></i>
                    <a href="https://www.instagram.com/theeliopsalms?igshid=ZDdkNTZiNTM=" target="_blank" title="Instagrap-Profile"><i
                            class="fa-brands fa-instagram"></i>
                        <a href="https://www.linkedin.com/in/duttasagar" target="_blank"
                            title="Linkdin-Profile"><i class="fa-brands fa-linkedin-in"></i>

            </div>
        </footer>

    </body>
</html>
```

Step 8: the css styling sheet present in the css subfolder in the public folder, is used to control the layout, formatting, and visual appearance of web pages and applications. It provides the designer with the ability to style and arrange elements, such as text, images, and forms, on a web page, making the web content visually appealing and user-friendly. CSS can also be used to create responsive designs that adapt to different screen sizes and devices, ensuring that the web page looks good on any device.

```css
nav{
  height: 100px !important;
  background-image: linear-gradient(to right, black, #141010) !important;
  color: #f4a688 !important;
  opacity: 0.8;
}
.card{
  background-image:linear-gradient(to right, black, #141010)  !important;
  color: #f4a688;
  opacity: 0.7;
  box-shadow: 10px 5px 5px black;
}
body{
  background-image: url(/images/bodyimage.jpg);
  /* background-image: linear-gradient(to right, #fbefcc ,  #f9ccac, #f4a688, #e0876a) !important; */
}
.searchcitybar{
  position: absolute;
  right: 0px;
  padding: 5px;
  margin-right: 10px;
}
.container{
  margin-top: 20px;
  margin-bottom: 20px;
}
h1{
  text-align: center!important;
```

Step 9: The coding in the app.js file present in the js subfolder of the public folder. It is a JavaScript implementation for fetching weather data from an API. The code does the following:

- Defines two constants, DefaultPlace and CheckCity. DefaultPlace is set to "Delhi" and CheckCity is a regular expression that checks if the user input is a valid city name (alphabetic characters only).

- Defines a function citySubmit that is triggered when the user submits a city name. The function performs the following steps:
  Retrieves the value entered in the text field with the id 'userInput'
  Validates the input to ensure it is a valid city name. If the input is not valid, an error message is displayed with a red background color.
  If the input is valid, the function calls GetWeather and passes the input value as an argument.

- Defines a constant GetWeather that is a function that takes a city name as an argument. The function performs the following steps:
  Creates a URL string with the city name and an API key.
  Makes a GET request to the URL using the Fetch API.
  If the request is successful, the data is logged to the console and the ShowWeather function is called with the data as an argument.
  If the request fails, an error message is displayed with a red background color.

```
const DefaultPlace = "Delhi";
const CheckCity = /^[A-Za-z]+$/;
function citySubmit(){
    var textField = document.getElementById('userInput');
    console.log(textField.value);
    let len= textField.value;
    let ErrorMsg = document.getElementById("ErrorMsg");
            if(textField.value.match(CheckCity) && len.length != 0){
                GetWeather(textField.value);
            }
            else{
                ErrorMsg.style.opacity = 1;
                ErrorMsg.innerHTML ="Kindly Enter Valid Place Name !";
                ErrorMsg.style.backgroundColor ="#ff5d9e";
                HideErrorMsg();
                searchCity.focus();
                searchCity.value="";
            }
}

// Success function
function successFunction(value) {
    var lat = value.coords.latitude;
    var lon = value.coords.longitude;
    console.log(lat);
    console.log(lon);
    const url = `http://api.openweathermap.org/data/2.5/forecast?lat={lat}&lon={lon}&appid=bec85626bbe604b4eb861ba8b540c081`;
    fetch(url).then((resp) => resp.json())
```

Step 10: The code is creating a function that displays 5-day weather information for a specific city. The function takes an argument WeatherData, which should contain information about the weather in a certain city. The information from the WeatherData is then used to fill in the values of various HTML elements, such as the temperature, humidity, wind speed, and pressure, etc. The code gets references to these elements by calling document.getElementById(elementId) where elementId is a string representing the id of the HTML element to be updated. The updated information is then set to the innerHTML property of the elements, which updates the text content of the elements on the webpage.

Step 11: Now create a **server.js** file in the *Major Project - Sagar Dutta* directory. The code is a simple Express.js application setup. The following steps occur in the code:
Required modules are imported:

- express: The main Express.js framework for building web applications
- body-parser: Middleware for parsing incoming request bodies in a middleware before your handlers, available under the req.body property.
- request: A simplified HTTP request client

Express is initialized and configured:

- The app.use method is used to specify middleware that will be used in the application.
- The first app.use statement sets the public directory as the root directory and serves static files from it.
- The second app.use statement sets up the body-parser middleware to parse incoming request bodies.
- The app.set method sets the view engine to EJS.

The default GET endpoint is set up:

- The endpoint listens for GET requests to the root URL ("/") and returns the "index" template file without any data.

The application starts listening on port 1000 and logs a message to the console when the server starts.

```
const express = require('express');
const bodyParser = require('body-parser');
const request = require('request');
const app = express();

// Setup your express app and body-parser configurations
// Setup your javascript template view engine
// we will serve your static pages from the public directory, it will act as your root directory
app.use(express.static('public'));
app.use(bodyParser.urlencoded({ extended: true }));
app.set("view engine", "ejs");

// Setup your default display on launch
app.get("/", function (req, res) {
  // It will not fetch and display any data in the index page
  res.render("index");
});

app.listen(1000, function(){
    console.log('server started!');
});
```

OUTPUT


clear sky
Max Temp (°C) 17
Min Temp (°C) 16
Humidity (%) 82
Winds (kmh-1) 2.73
Pressure (atm) 1014
02-05 00:00:00 sky
2023-02-06 00:00:00 16 16 46 0.36 1011 clear sky
2023-02-07 00:00:00 17 17 49 1.23 1009 clear sky
2023-02-08 00:00:00 15 15 48 2.13 1012 clear sky
2023-02-09 00:00:00 15 15 36 1.96 1011 few clouds
Weather Forecast App
Designed and Developed By Sagar Dutta