Trinity College Dublin

The University of Dublin

INTERNET APPLICATIONS

CSU44000-202223

ASSIGNMENT – 1

Name: Mudit Garg

TCD ID: 21355125

E-Mail ID: gargmu@tcd.ie

Mobile: +353892050748

DISCLAIMER:

All the codes are mentioned here in this document for reference purposes. To run the project in its intended manner:

- First have node.js installed and have node command globally available on your system for the server to be started easily.
- Extract the Assignment 1.zip file onto your system.
- Open Terminal in the folder where you have extracted the code.
- Type – "npm start" to start the server
- Now the website is up and running for testing on https://localhost:5510

Note: To run the code easily, just use the folder that is obtained by extracting Assignment 1.zip. Copying code from this pdf will result in many different problems and will require additional steps since you need to:

- Replicate the project structure that was intended while making the application that is:
    - **Root Directory:** index.html, package.json, server.js, "stylesheets" folder, "scripts" folder.
    - **"stylesheets" Directory in Root Directory:** app.css, button.css, scrollbar.css
    - **"scripts" Directory in Root Directory:** app.js
- Install the various node modules in order for the server.js to work. You can do that by typing "npm install" on the terminal

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title> Weather Bucket </title>
    <script src="https://unpkg.com/vue@next"></script>
    <link rel="stylesheet" type="text/css" href="stylesheets/app.css">
    <link rel="stylesheet" type="text/css" href="stylesheets/scrollbar.css">
    <link rel="stylesheet" type="text/css" href="stylesheets/button.css">
    <link
href="https://fonts.googleapis.com/css2?family=Kanit:wght@600&family=Space+Grotesk:w
ght@700&display=swap" rel="stylesheet">
  </head>
  <body>
    <h2 id="title">Weather Bucket</h2>
    <hr>
    <div id="app">
      <div id="input">
```

```
            <br>
            Enter the city you wish to visit:
             
             
            <input type="text" v-model="user_city" v-on:keyup.enter="responseCheck"
placeholder="e.g.Dublin, IE" spellcheck="false">
             
            <br>
            <br>
            <br>
            <button id="go" v-on:click="responseCheck"><span>Go</span></button>
            <button id="cls" v-on:click="clearInput"><span>Reset</span></button>
        </div>
        <br>
        <div v-if="content_show" id="weather_status">
            <div v-if="content_show" id="location">
                <img src="https://cdn-icons-
png.flaticon.com/32/819/819865.png"/> {{printLocation()}}
            </div>
            <br>
            <div id="indicators">
                <div id="weather_icon">
                    <img v-bind:src="weather_icon_url"/>
                </div>
                <img v-if="pollution_show" id="mask_icon" src="https://cdn-icons-
png.flaticon.com/128/3167/3167474.png"/>
            </div>
            <br>
            <h2><b>{{feelsLike}}</b></h2>
            <div v-if="pollution_show" id="pollution">
                PM 2.5 levels are going to be hazardous. Please wear a mask !!
            </div>
            <h3>{{umbrellaCheck}}</h3>
            <br>
            <br>
        </div>
        <div v-if="error_show" id="error">
            Code 404: City not found! Try again
        </div>
        <table v-if="content_show" id="summary">
            <div v-if="content_show" id="table_head">
                Weather Forecast for the next few days
            </div>
            <hr v-if="content_show">
            <thead>
                <tr>
```

```
                <th>Date</th>
                <th>Time</th>
                <th></th>
                <th>Description</th>
                <th>Temperature (°C)</th>
                <th>Feels Like (°C)</th>
                <th>Min Temp (°C)</th>
                <th>Max Temp (°C)</th>
                <th>Windspeed (m/s)</th>
                <th>Rainfall (mm)</th>
            </tr>
        </thead>
        <tbody>
            <tr v-for="item in items">
                <td>{{new Date(item.dt_txt).toDateString()}}</td>
                <td>{{new Date(item.dt_txt).toLocaleTimeString()}}</td>
                <td><img v-bind:src="retrieveImg(item.weather[0])"/></td>
                <td><b>{{item.weather[0].description}}</b></td>
                <td>{{item.main.temp}}°</td>
                <td>{{item.main.feels_like}}°</td>
                <td>{{item.main.temp_min}}°</td>
                <td>{{item.main.temp_max}}°</td>
                <td>{{item.wind.speed}}</td>
                <td v-if="item.rain == undefined">0</td>
                <td v-else>{{item.rain["3h"]}}</td>
            </tr>
        </tbody>
    </table>
</div>
<script src="scripts/app.js"></script>
</body>
</html>
```

APP.JS

```
var app = Vue.createApp({
    data() {
        return {
            api_id: 'aa3551b69f041b72a4094701dbc65f92',
            user_city: '',
            lat:'',
            lon:'',
            content_show: false,
```

```
        pollution_show: false,
        error_show: false,
        umbrellaCheck: 'Info not retrieved!!',
        feelsLike: '',
        items: [],
        weather_icon_url:''
    }
},
methods: {
    cityCapitalise() {
        // a minor function just to capitalise the first letter of the city
        this.user_city = this.user_city[0].toUpperCase()+this.user_city.substr(1)
    },
    responseCheck: async function() {
        //Function to check if the current city entered, exists or not
        let response = await this.retrieveTodayWeather()
        if(response.cod == '200') {
            this.content_show = true
            this.error_show = false
            this.cityCapitalise(response)
            console.log(`Weather bucket is looking at the forecast of ${this.user_city}`)
            this.compute(response)
        } else if (response.cod == '404') {
            this.content_show = false
            this.error_show = true
        }
    },
    retrieveTodayWeather: async function() {
        //Fetching the weather data for today !!
        let origin = "http://localhost:5510/"
        let server_response = await fetch(origin+"current_weather/"+this.user_city)
        server_response = await server_response.json()
        if(server_response.data == "404") {
            let error_response = {"cod":'404', "message":"City not found!"}
            console.log("========================")
            console.log(error_response)
            return error_response
        }
        console.log("========================")
        let server_data = server_response
        console.log("Current Weather JSON Data:")
        console.log(server_data)
        return server_data
    },
    forecastFetch: async function() {
        //Fetching the forecast data for 4 days !!
```

```
    let origin = "http://localhost:5510/"
    let server_response = await fetch(origin+"future_forecast/"+this.user_city)
    let data = await server_response.json()
    console.log("Forecast JSON Data:")
    console.log(data)
    this.items = data.list
    this.rainStatus()
},
retrievePollution: async function() {
    //Fetching the pollution data for 4 days !!
    let origin = "http://localhost:5510/"
    let server_response = await fetch(origin+"pollution/"+this.lat+","+this.lon)
    let data = await server_response.json()
    console.log("Pollution JSON Data:")
    console.log(data)
    let list = data.list
    for(let i=0;i<92;i++) {
        let pm_lvl = list[i].components.pm2_5
        if(pm_lvl >= 10) {
            this.pollution_show = true
        }
    }
},
compute(response) {
    // Main Function calling other functions to do every task required
    this.posCalc(response)
    this.weather_icon_url = this.retrieveImg2x(response.weather[0])
    this.feelsLikeStatus(response)
    this.forecastFetch()
    this.retrievePollution()
},
posCalc(response) {
    // Storing the lattitude and longitude of the current city
    this.lat = response.coord.lat
    this.lon = response.coord.lon
},
retrieveImg(weather) {
    // Retrieving icon based on the current weather
    let icon = weather.icon
    let url = "https://openweathermap.org/img/wn/"+icon+".png"
    return url
},
retrieveImg2x(weather) {
    // Retrieving icon based on the current weather double the size
    let icon = weather.icon
    let url = "https://openweathermap.org/img/wn/"+icon+"@2x.png"
```

```
            return url
        },
        rainStatus() {
            // Rain Status Check
            this.umbrellaCheck = "⛅🌂 The skies are clear for a few days 🌂⛅"
            for(let i=0;i<this.items.length;i++) {
                let rain = this.items[i].rain
                if(rain != undefined) {
                    this.umbrellaCheck = "🌧️🌂 You should keep an umbrella with you 🌂🌧️"
                }
            }
        },
        feelsLikeStatus(response) {
            // Feels like Message construction
            let feels_like = response.main.feels_like
            this.feelsLike = "It feels like "+ (feels_like)+"°C"
            if(feels_like<=12) {
                this.feelsLike = "Pack for COLD. "+ this.feelsLike
            } else if(feels_like>12 && feels_like<24) {
                this.feelsLike = "Pack for MILD. "+ this.feelsLike
            } else if(feels_like>=24) {
                this.feelsLike = "Pack for HOT. "+ this.feelsLike
            }
        },
        clearInput() {
            // Clearing all variables
            this.content_show = false
            this.error_show = false
            this.pollution_show = false
            this.user_city = ''
            this.umbrellaCheck= 'Info not retrieved!!'
            this.feelsLike = ''
            this.items = []
            this.weather_icon_url = ''
        },
        printLocation() {
            // Printing Location Name and Lattitude and Longitude
            return this.user_city+" ("+this.lat+", "+this.lon+")"
        }
    }
})

app.mount('#app');
```

SERVER.JS

```
// Necessary Imports for the code
const express = require("express")
const axios = require("axios")

// Initialising the Express server
const app = express()

// Port number where the server would listen requests
const PORT = 5510

// Path where the files are stored
const PATH = __dirname

// OpenWeather API key free plan
const api_id = 'aa3551b69f041b72a4094701dbc65f92'

// Make sure that the server serves the appropriate files from the mentione path
app.use(express.static(PATH));

// Make the server listen to the requests on the port mentioned
app.listen(PORT, () => {
    console.log(`Weather Bucket ready to listen on address http://localhost:${PORT}/ !`)
})

// Sending the HTML file on calling the home address
app.get('/', (req,res) => {
    res.sendFile("/index.html")
})

// Sending the data of the current weather fetched from the OpenWeather API based on the
city entered
app.get('/current_weather/:city', async (req,res) => {
    let response
    try {
        response = await
axios.get("https://api.openweathermap.org/data/2.5/weather?q="+req.params.city+"&units=m
etric&APPID="+api_id)
        console.log("============================")
        console.log("Checking the recieved response from API call ...")
        let json = response.data
        // console.log(json)
```

```
        res.send(json)
        console.log("Weather data for "+req.params.city+" sent to the frontend for display!!")
    } catch(error) {
        let response = {
            "data":"404"
        }
        console.log("============================")
        console.log("Code 404: City not found! Try again !")
        // console.log(error)
        res.send(response)
    }
})

// Sending the data of the future weather fetched from the OpenWeather API based on the city entered
app.get('/future_forecast/:city', async (req,res) => {
    response = await axios.get("https://api.openweathermap.org/data/2.5/forecast?q="+req.params.city+"&units=metric&cnt=32&APPID="+api_id)
    let json = response.data
    // console.log(json)
    res.send(json)
    console.log("Future weather data for "+req.params.city+" sent to the frontend for display!!")
})

// Sending the data of the future air pollution fetched from the OpenWeather API based on the city entered
app.get('/pollution/:lat,:lon', async (req,res) => {
    response = await axios.get("https://api.openweathermap.org/data/2.5/air_pollution/forecast?lat="+req.params.lat+"&lon="+req.params.lon+"&appid="+api_id)
    let json = response.data
    // console.log(json)
    res.send(json)
    console.log("Pollution data for ("+req.params.lat+", "+req.params.lon+")")
})
```

PACKAGE.JSON

```
{
  "name": "weather-bucket",
  "description": "A simple app to use the OpenWeather API",
```

```
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "nodemon server.js"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/muditgarg48/weather_bucket.git"
  },
  "author": "Mudit Garg",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/muditgarg48/weather_bucket/issues"
  },
  "homepage": "https://github.com/muditgarg48/weather_bucket#readme",
  "dependencies": {
    "express": "^4.18.2",
    "nodemon": "^2.0.20",
    "axios": "^1.1.3"
  }
}
```

APP.CSS

```
html {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 100%;
}
body {
    font-family: 'Kanit', sans-serif;
    background-image: url("https://images.theconversation.com/files/232705/original/file-
20180820-30593-1nxanpj.jpg?ixlib=rb-
1.1.0&q=45&auto=format&w=1200&h=900.0&fit=crop");
    background-repeat:no-repeat;
    background-size: cover;
    padding: 0 5%;
    width: 100%;
    height: 100%;
    padding-bottom: 10vh;
}
```

```
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
table {
    width: 100%;
    padding: 10%;
    background: rgba(0,0,0,0.2);
    backdrop-filter: saturate(180%) blur(10px);
    box-shadow: 0 3px 10px rgb(0 0 0 / 0.2);
}
th, td {
    padding: 7px;
    text-align: center;
}
input[type=text] {
    border: none;
    border-bottom: 2px solid black;
    background-color: transparent;
    font-size: 30px;
    outline: none;
    text-align: center;
}

#error {
    color: red;
    text-align: center;
    font-size: 3vh;
}

#pollution {
    color: yellow;
    text-align: center;
    font-size: 3vh;
}

#table_head {
    font-size: 3vh;
}
#title {
    font-size: 8vh;
}

#title, #input, #weather_status {
    text-align: center;
```

```
}

#indicators{
    display: flex;
    justify-content: center;
    align-items: center;
}

#weather_icon {
    border-radius: 50%;
    box-shadow: 0 0 50px grey;
    display: flex;
    width: min-content;
    background: grey;
    backdrop-filter: saturate(180%) blur(10px);
}
```

SCROLLBAR.CSS

```
body {
    margin: 0;
    overflow-x: hidden;
}

::-webkit-scrollbar {
  width: 10px;
  background-color: #F5F5F5;
}

::-webkit-scrollbar-track {
  box-shadow: inset 0 0 6px rgba(0,0,0,0.3);
  background-color: #F5F5F5;
  border-radius: 10px;
}

::-webkit-scrollbar-thumb {
  background-image: -webkit-gradient(linear,
                    left bottom,
                    left top,
                    color-stop(0.44, rgb(122, 157, 217)),
                    color-stop(0.72, rgb(73,125,189)),
                    color-stop(0.86, rgb(28,58,148)));
  border-radius: 20px;
```

```
}

::-webkit-scrollbar-thumb:hover {
  background: rgba(146,187,255,1);
}
```

BUTTON.CSS

```
button {
    font-family: 'Space Grotesk', sans-serif;
    display: inline-block;
    border-radius: 100px;
    background: rgba(0,0,0,0.2);
    border: none;
    color: black;
    text-align: center;
    font-size: 20px;
    padding: 10px;
    width: 200px;
    transition: all 0.5s;
    cursor: pointer;
    margin: 5px;
}

button span {
    cursor: pointer;
    display: inline-block;
    position: relative;
    transition: 0.5s;
}

button span:after {
    position: absolute;
    opacity: 0;
    top: 0;
    right: -20px;
    transition: 0.5s;
}

#go span:after {
    content: '\00bb';
    color: green;
```

```
}

#cls span:after {
    content: 'x';
    color: red;
}

button:hover span {
    padding-right: 25px;
}

button:hover span:after {
    opacity: 1;
    right: 0;
}
```