KOLEJ PROFESIONAL MARA BERANANG

DIPLOMA IN COMPUTER SCIENCE

COURSE NAME	:	**OBJECT ORIENTED PROGRAMMING**
COURSE CODE	:	**CSC2744**
ACADEMIC SESSION	:	**SESSION 3 2023/2024**
TYPE OF ASSESSMENT	:	**FINAL ASSIGNMENT**
DURATION	:	**13/02/2024 - 05/03/2024**

CLO 3: Employ third party data in object oriented application development using graphical user interface (GUI) application framework

INSTRUCTION TO CANDIDATES:

1. Late submissions after the given due date will not be accepted.
2. Report should be written using:
   Font type: Arial
   Size: 12 pts
   Line Spacing: 1.5
3. Coding format:
   Font type: Consolas
   Size: 10 pts
   Line Spacing: Single

Personal Details	
Name	NUR ASMA' BINTI FAIZAL SANUSI
I/D Number	BCS2211-041
Class	DCS4B
Lecturer	PUAN NINI ANIZA BINTI ZAKARIA

Section / Question No.	**Marks**
Total	**/ 50**

Question

As part of your software development course, you have been tasked with creating a desktop application using Electron, a framework for building cross-platform desktop applications with web technologies like HTML, CSS, and JavaScript. Additionally, you've been provided access to a specific API that offers various functionalities and data sets. Your goal is to design a meaningful application that leverages one of the provided API and incorporates CRUD (Create, Read, Update, Delete) operations to enhance user interaction and usability.

List of API	Requirements
https://restcountries.com/v3.1/all	• Information Output: Common and official name for the searched country, capital city, currencies, region, languages, population, flag, location, continent, google map.
https://www.omdbapi.com/?apikey=c1aecf62&i=[imdbid]	• Information Output: Movie title, year, genre, director, actors, plot, poster, rating, awards, writer.
https://api.openweathermap.org/data/2.5/weather?q=[city]&appid=9fd7a449d055dba26a982a3220f32aa2	• Information Output: City name, country, coordinate, weather description, current temperature, min & max temperature, humidity, wind speed, sunrise and sunset.

Tasks:

1. Create desktop application using electron framework with the integration of the given API. Your application needs to output the information required above. It should have at least 2 pages and you may add extra functionality or features of your choice to the application.
2. Implement CRUD (create, read, update, delete) process to the application to make your application more meaningful.
3. Apply HTML and CSS for user interface and provide evidence for application.
4. GUI Elements:
    i. Apply GUI elements that assist users in using application.
    ii. The application's 'look and feel' is attractive and informative.
5. Produce a report on your application. Include the following:
    i. Overview of the application and its purpose.

ii. Description of the functionalities and features of the developed application with print screen of the pages and explanation.

iii. Program codes of your system

6. Submit files in GitHub.

Assessment Rubric

ATTRIBUTES	CRITERIA	POOR (1 mark)	FAIR (2 marks)	GOOD (3 marks)	VERY GOOD (4 marks)	EXCELLENT (5 marks)	Mark Obtained
Reproduce and Process Information	1. Create desktop application using electron framework with the integration of the given API. Your application needs to output the information required above.	• The application is an extensive collection and rehash of other people's ideas, products, and images. There is no evidence of new thought.	• The application is somewhat a collection and rehash of other people's ideas, products, and images. There is little evidence of new thought or inventiveness.	• The application is a minimal collection or rehash of other people's ideas, products, and images. There is a few evidence of new thought or inventiveness.	• The application shows a lot of evidence of originality and inventiveness.	• The application shows significant evidence of originality and inventiveness. • Most of the content and many of the ideas are fresh, original, and inventive.	
		• Able to display part of the required data from the API and does not fulfill the requirements.	• Able to display all the required data from the API that meet with the requirements with no description.	• Able to display extra data from the API beyond the application requirements with no description.	• Able to display all the required data from the API that meet with the requirements with the description.	• Able to display extra data from the API beyond the application requirements with the description.	
		• The data from the API does not reflect the whole purpose of the application developed.	• The data from the API is sufficient but does not reflect the whole purpose of the application developed.	• The data from the API is meaningful but does not reflect the purpose of the application developed.	• Able to utilize the data fetched from the API to come up with meaningful functionalities that reflect the purpose of the application developed.	• Able to utilize the data fetched from the API to come up with meaningful functionalities that reflect the purpose of the application developed. • Come up with extra idea for the	

						application developed that enhances the user experience or adds value to the application.	
	2. Implement CRUD (create, read, update, delete) process to the application to make your application more meaningful.	• Able to create only 2 of the CRUD processes. • No feedback for the CRUD process. • The design for data input is poor	• Able to create only 3 of the CRUD processes. • No feedback for the CRUD process. • The design for data input is good with some room for improvement	• Able to perform all the CRUD processes. • No feedback for the CRUD process. • Well-designed data input for CRUD process.	• Able to perform all the CRUD processes. • No feedback for the CRUD process. • Well-designed data input for CRUD process.	• Able to perform all the CRUD processes. • Appropriate feedback for the CRUD process. • Well-designed and user-friendly data input for CRUD process.	
	3. Apply HTML and CSS for user interface and provide evidence for application.	• **Text -** All text used is too small to view or the font type is wrongly chosen. • **Graphics** - Graphics seem randomly chosen, are of low quality, OR distract the reader.	• **Text** – Some of the text used is too small to view or the font type is wrongly chosen. • **Graphics -** Graphics seem randomly chosen, are of low quality, OR distract the reader.	• **Text** - Most text used is clear but does not describe the content well. • **Graphics** - Graphics are related to the theme/purpose of the application and are of excellent quality.	• **Text** - All text used is clear but does not describe the content well. • **Graphics -** Graphics are related to the theme/purpose of the application, are of excellent quality and enhance reader interest or understanding	• **Text** - All text used is clear and able to describe the content well. • **Graphics** - Graphics are related to the theme/purpose of the application, are thoughtfully cropped, are of high quality and enhance reader interest or understanding.	

Curate	4.GUI Elements: i. Apply GUI elements that assist users in using application.	• Not able to curate for required content. • **Layout** - The HTML elements in the application are cluttered looking or confusing. • **Navigation** Links do not take the reader to the sites/ pages described. User typically feels lost.	• Limited curation for required content. • **Layout** - The HTML elements in the application is messy, may appear busy or boring. • **Navigation** Links seem to be missing and don't allow the user to easily navigate.	• Satisfactory curation for required content. • **Layout** -The HTML elements are suitable. • **Navigation** Links allow the reader to move from page to page, but some links seem to be missing.	• Good curation for required content. • **Layout** - The HTML elements are suitable and usable. • **Navigation** Links are labelled and allow the user to easily move from page to page.	• Excellent curation for required content. • **Layout** - The HTML elements are well structured, attractive, and usable layout. • **Navigation** Links are clearly labelled, consistently placed, and allow the user to easily move from page to page.	
	ii. The application's 'look and feel' is attractive and informative.	• The application needs polish in its visual design and is not appropriate for the target audience. • **Color** Choice of colors and combinations are not suitable.	• The application needs polish in its visual design, but it is still appropriate for the target audience. • **Color** Choice of colors and combinations do not match the concept of the application.	• The application mostly follows good visual design principles (e.g.: alignment, contrast, easily read text) and is appropriate for the target audience. • **Color** Choice of colors and combinations match the concept of the application.	• The application demonstrates good visual design principles (e.g.: alignment, contrast, easily read text) and is appropriate for the target audience. • **Color** Appropriate colors used to produce an atmosphere that expresses	• The application clearly demonstrates good visual design principles (e.g.: alignment, contrast, easily read text) and is appropriate for the target audience. • **Color** Appropriate colors used to produce an atmosphere that expresses the concept of the application.	

					the concept of the application.		
Convey	5. Produce a report on your application. Include the following: i. Overview of the application and its purpose.	• The overview of the application and its purpose is vague.	• The overview of the application and its purpose is very brief.	• The overview of the application and its purpose is clearly described.			
	ii. Description of the functionalities and features of the developed application with print screen of the pages and explanation.	• The description of the functionalities and features is poor and unorganized. • Incomplete print screen of the application.	• The description of the functionalities and features is very brief. • Incomplete print screen of the application.	• The description of the functionalities and features is complete. • Complete screenshot of the pages.	• The description of the functionalities and features is complete with extensive information. • Complete screenshot of the pages.	• The description of the functionalities and features is complete with extensive information. • Complete screenshot of the pages and the labelling of the features in the application.	
	iii. Program codes of your system	• HTML, CSS, and JavaScript codes attached are not complete. • The codes are hardly read.	• HTML, CSS, and JavaScript codes attached are complete. • The codes are hardly read.	• HTML, CSS, and JavaScript codes attached are complete. • The codes are readable but not organized.	• HTML, CSS, and JavaScript codes attached are complete. • The codes are readable and organize.	• HTML, CSS, and JavaScript codes attached are complete and include comments for the important parts of the codes. • The codes are readable and organized.	
	6. Submit files in GitHub.	• Does not submit complete electron files in GitHub	• Completely submit all the electron file in GitHub.				
Total Marks Earned							**/50**
Total Percentage (40%)							**/40%**

Overview:

The provided code creates a web page for travel planning with a focus on managing and setting reminders for events. Users can add, edit, delete, and search for events, and the information is stored in local storage. The page has a visually appealing design with responsive layout features.

Purposes:

1. **Event Management:**
    - Users can add events with details such as date, time, place, and things to do.
    - Existing events can be edited or deleted.
    - Events are displayed in a visually organized manner.
2. **Search Functionality:**
    - Users can search for events based on the entered place, enhancing the ability to find specific information.
3. **Data Persistence:**
    - Local storage is used to store event data, ensuring that user-added events are retained even when the page is reloaded.
4. **WeatherMaster Integration:**
    - The page is part of the WeatherMaster application, as indicated by the header, and it suggests integration with weather-related features.

Navigation Classification Explanation and Function:

1. **Home (index.html):**
    - ***Overview:*** Likely the main landing page providing general information or options.
    - ***Function:*** Directs users to the home page.
2. **Travel (travel.html):**
    - ***Overview:*** Possibly a section for travel-related information or features.
    - ***Function:*** Navigates users to the travel-related content.
3. **Local Events (event.html):**
    - ***Overview***: Focuses on local events and possibly integrates with the calendar.
    - ***Function:*** Takes users to the local events page.
    - ***Dropdown (Reminder - todo.html):***
        - *Overview:* A subsection or related feature, possibly related to reminders.
        - *Function:* Offers a dropdown with a link to the reminder page.
4. **Reminder (todo.html):**
    - ***Overview:*** Dedicated to managing reminders and events with a weather-related theme.
    - ***Function:*** Opens the reminder page for event management and integrates with WeatherMaster features.

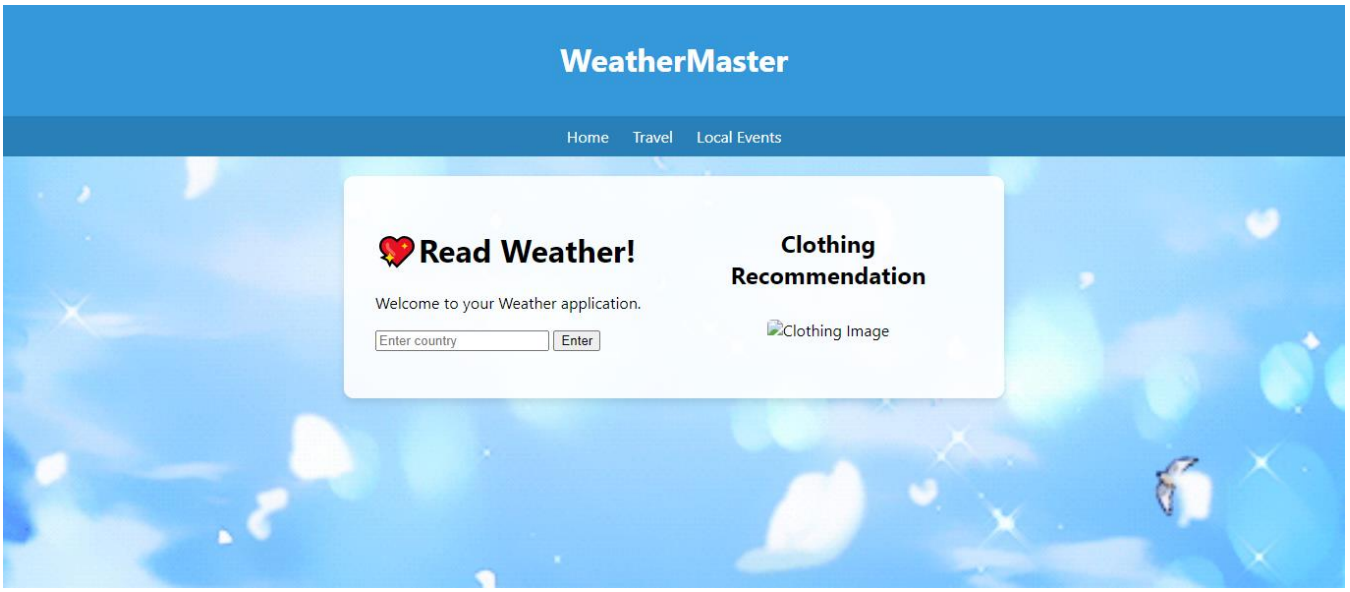
WeatherMaster App's Products: While the provided code is part of the WeatherMaster application, the specific products or features of WeatherMaster are not explicitly detailed in the code. However, based on the page structure and content, WeatherMaster might offer:

- **Weather Information Integration:**
  - Users could potentially access weather information for selected destinations and dates during travel planning.

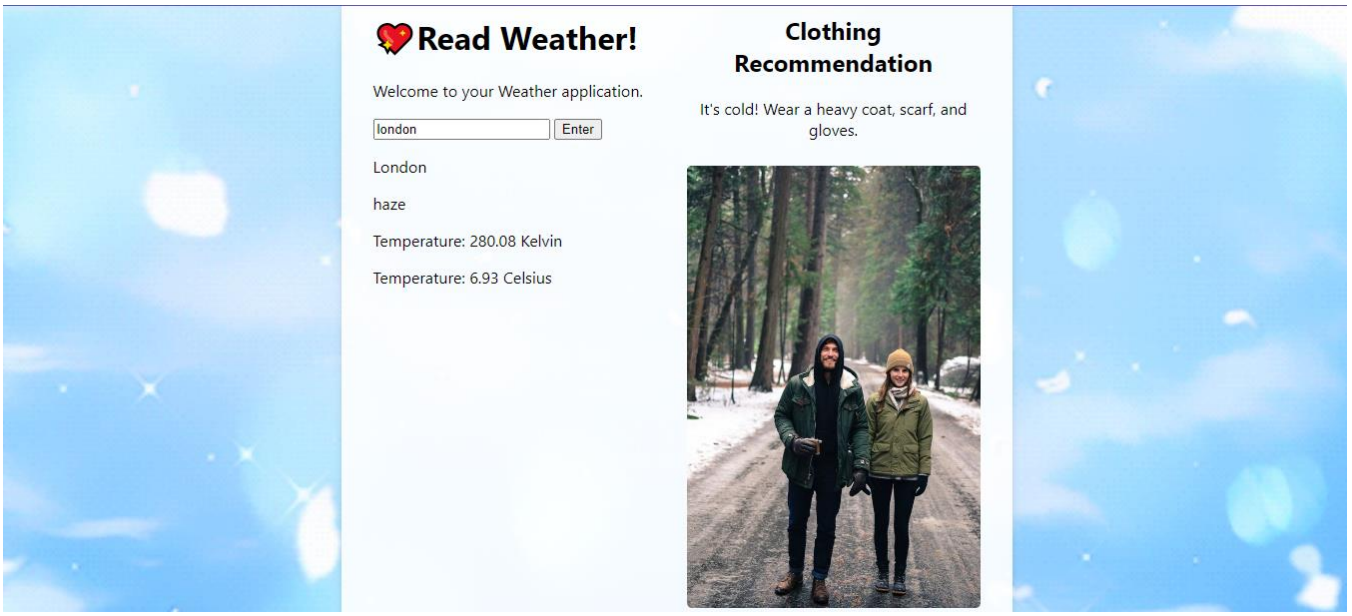
- **Event-Weather Integration:**
  - The reminder and event management features may integrate weather details, allowing users to plan activities based on weather conditions.

- **User-Friendly Interface:**
  - WeatherMaster aims to provide a user-friendly interface for planning trips with confidence, suggesting a focus on travel-related information and weather considerations.

1. Home Page (index.html, index.css, index.js)



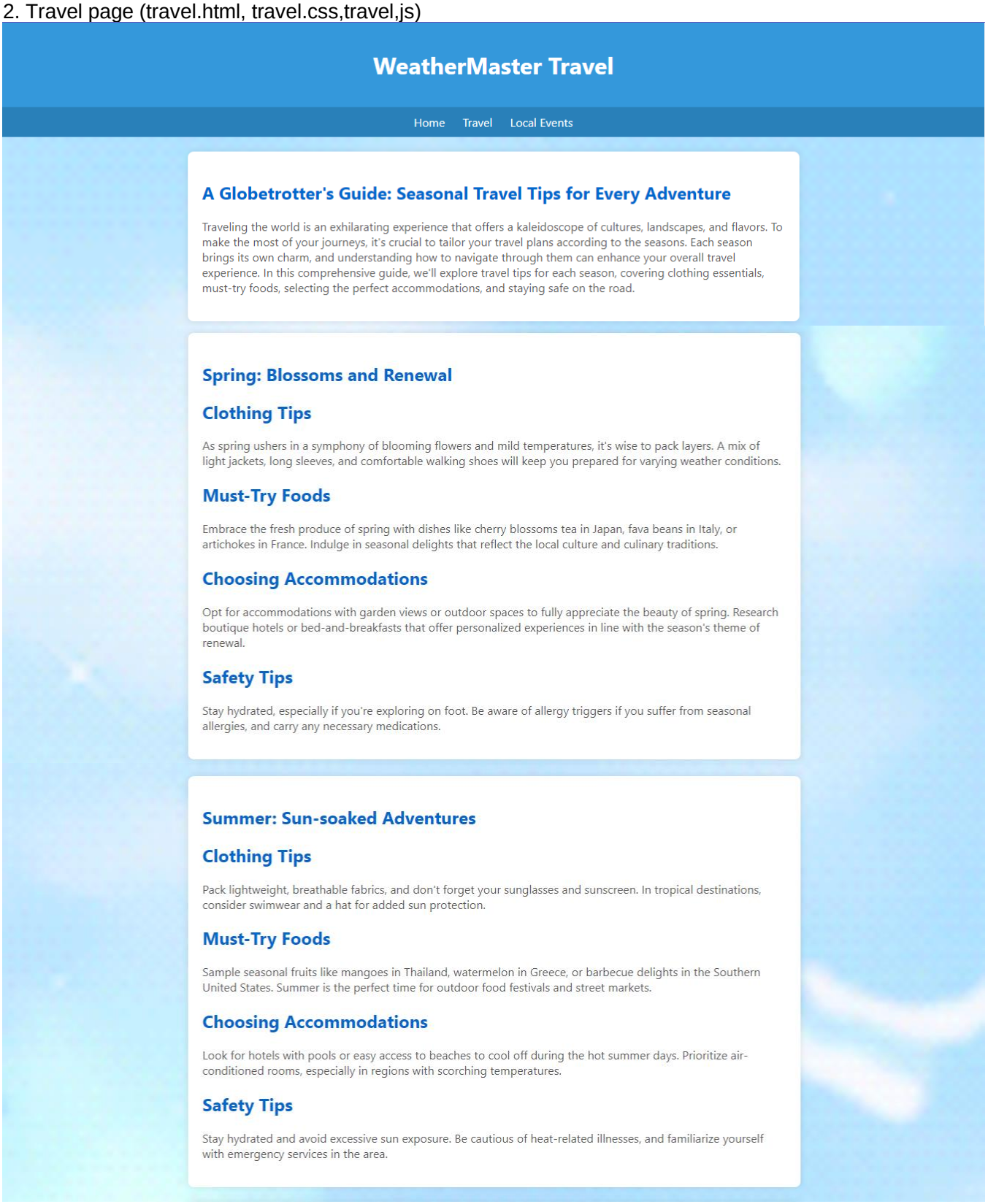
Firstly, user need to enter country to read weather forecasts and click enter

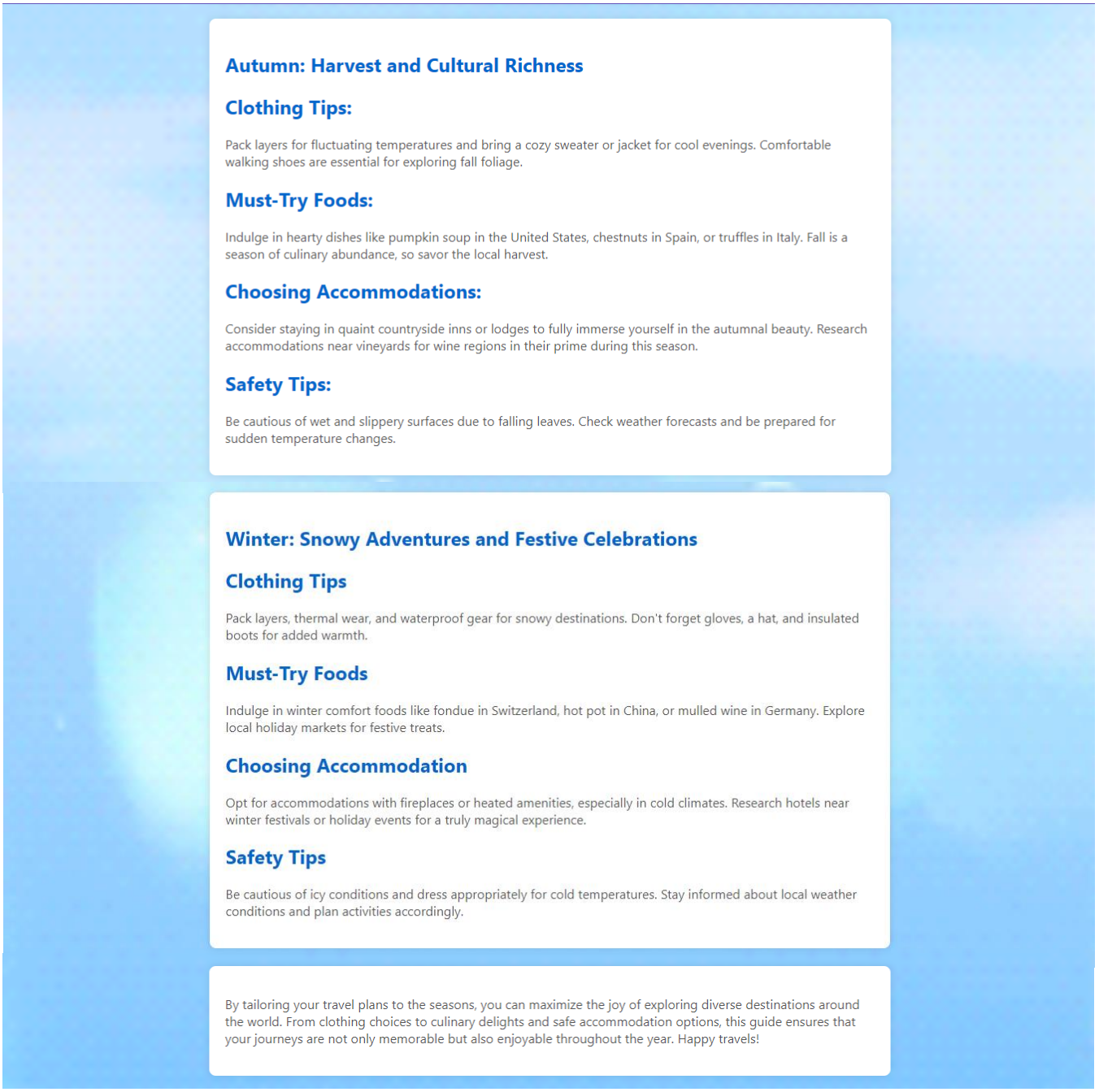


system display the weather forecast and cloth recommendation based on temperature.

The Home Page of the WeatherMaster app serves as the main entry point, offering a brief overview and navigation options for users. Here, users can find general information about the application and its features. The navigation bar provides links to key sections of the app, including Travel, Local Events, and Reminders. It acts as a central hub for users to explore the various functionalities available in WeatherMaster.

2. Travel page (travel.html, travel.css,travel,js)

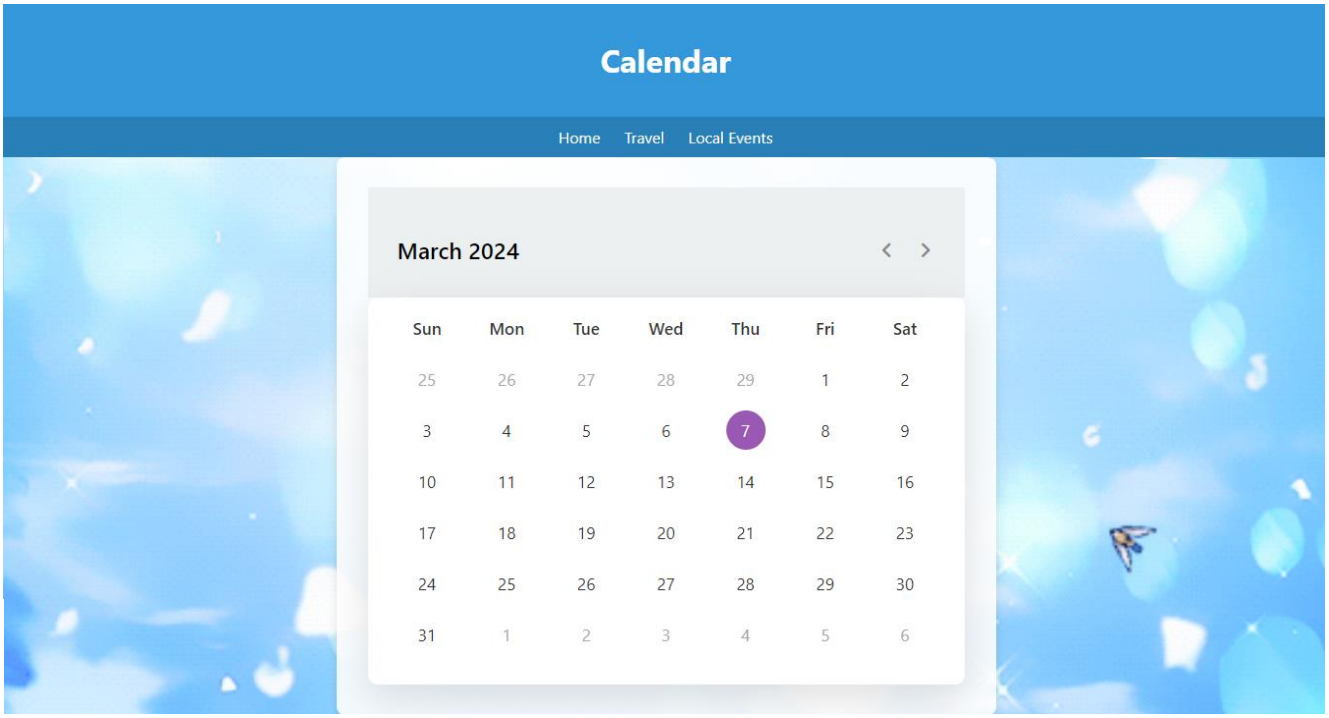




The system display the article about travel tips . this page only display for user guidance furthermore. Beside, this page should have features so that user can share their thoughts on travel to share for every user that use this WeatherMaster application.

The Travel Planning page is designed to assist users in organizing their trips efficiently. Users can access this section by clicking on the "Travel" link in the navigation bar. The page includes a search bar where users can input a specific place to gather relevant travel information. By integrating weather details, WeatherMaster empowers users to plan their trips with confidence, taking weather conditions into account for a seamless travel experience.

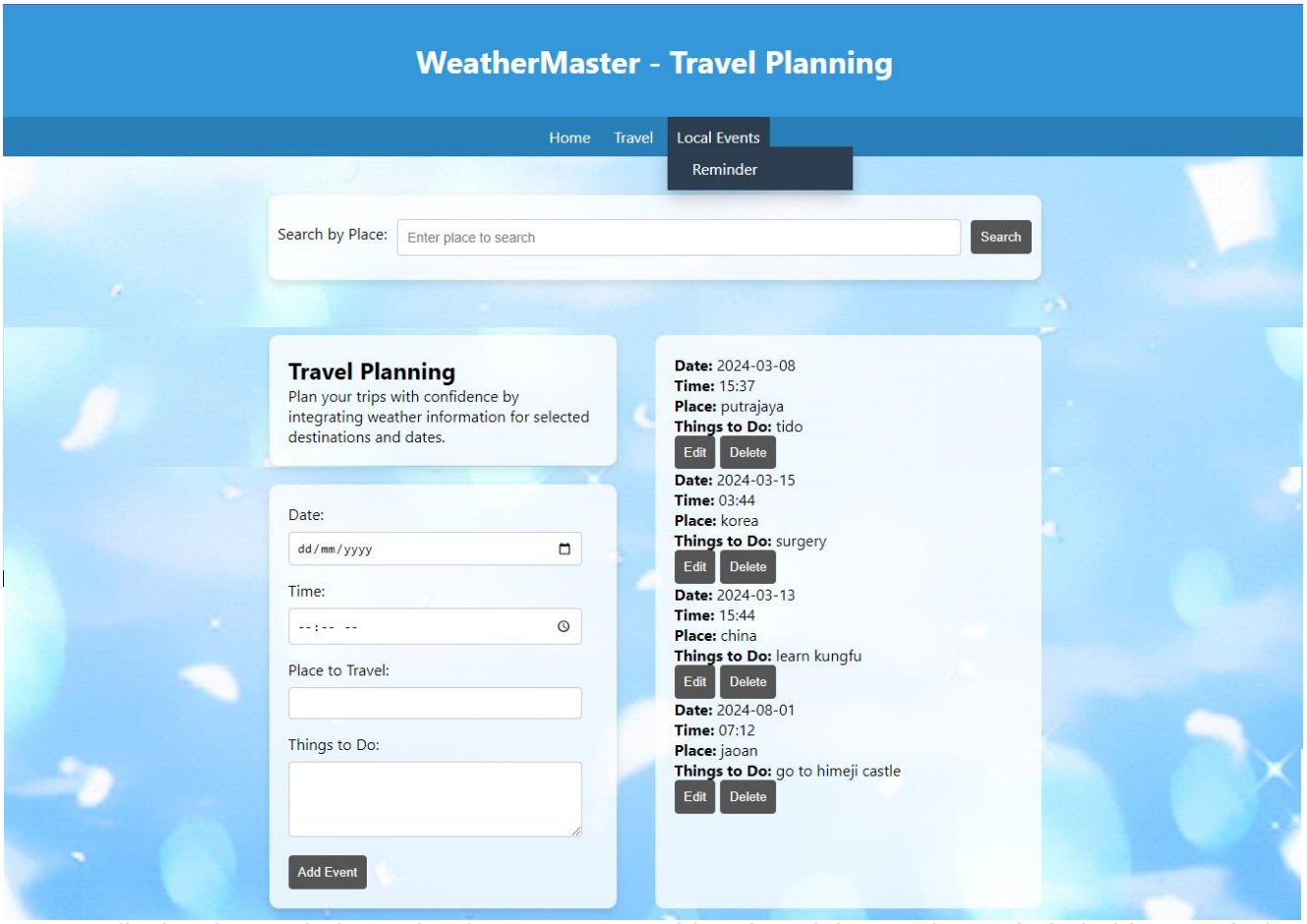
3. Local Event Page (event.html,event.css,event.js)



The system here only display calendar .

Navigating to the Local Events page via the "Local Events" link, users can explore and manage local events effortlessly. The page incorporates a visually appealing calendar layout, displaying events by date. Users can seamlessly scroll through the calendar, identify local events, and engage with the app's features. The dropdown menu within this section provides a link to the "Reminder" page, suggesting an integration of event reminders with WeatherMaster's capabilities

4.Reminder page (todo.html,todo.js,todo.css)



System display the reminder tools where user can add,update,delete and search their things to do for travel purpose only. This part user can add event through provided form.

The Reminder Management page, accessible through the "Reminder" link, serves as a dedicated space for users to add, edit, delete, and search for events. Users can utilize the search bar to filter events based on a specific place, enhancing the efficiency of event management. The page features a form for adding new events, including date, time, place, and things to do. Events are displayed in a calendar format, offering users a clear and organized view of their schedule.

User-Friendly Features and Integration:
WeatherMaster strives to provide a user-friendly interface, ensuring a seamless experience for users. The integration of weather information with travel and event planning enhances the app's functionality. The dropdown navigation and clearly labeled links facilitate easy access to different sections of the app. Users can navigate intuitively, leveraging the features of WeatherMaster to plan and manage their travel and events effectively.
In summary, WeatherMaster is a comprehensive application designed to simplify travel planning and event management. The user-friendly interface, clear navigation, and integration of weather details contribute to a cohesive and efficient user experience. Whether organizing a trip or managing local events, users can leverage WeatherMaster's features for a confident and well-informed planning process.

The code of Home Page :

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Home - WeatherMaster</title>
  <link rel="stylesheet" href="index.css">
</head>
<body>
  <header>
    <h1>WeatherMaster</h1>
  </header>

  <nav>
    <ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="travel.html">Travel</a></li>
      <li>
        <div class="dropdown">
          <a href="event.html">Local Events</a>
          <div class="dropdown-content">
            <a href="todo.html">Reminder</a>
          </div>
        </div>
      </li>
    </ul>
  </nav>

  <div class="weather-container">
    <div class="weather-input">
      <h1>Read Weather!</h1>
      <p>Welcome to your Weather application.</p>
      <input type="text" id="searchData" placeholder="Enter country">
      <button onclick="displayWeather()">Enter</button>
      <p id="country"></p>
      <p id="desc"></p>
      <p id="tempK"></p>
      <p id="tempC"></p>
    </div>

    <!-- To display cloth recommendation based on temperature -->
    <div class="clothing-recommendation">
      <h2>Clothing Recommendation</h2>
      <p id="clothing-recommendation-text"></p>
      <img id="clothing-image" src="" alt="Clothing Image">
    </div>
```

```
  </div>

  <script src="index.js" defer></script>
</body>
</html>
```

Index.css

```
body {
  font-family: 'Segoe UI', Roboto, Helvetica, Arial, sans-serif;
  margin: 0;
  padding: 0;
  background: url('images/day.gif') center/cover no-repeat; /* Replace with the path to
your GIF file */
}

header {
  background-color: #3498db;
  padding: 15px;
  text-align: center;
  color: white;
}

nav {
  display: flex;
  justify-content: space-around;
  background-color: #2980b9;
  padding: 10px;
}

nav a {
  color: white;
  text-decoration: none;
  padding: 10px;
  cursor: pointer;
}

nav a:hover {
  background-color: #2c3e50;
}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

li {
  display: inline;
```

```
}

.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #2c3e50;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  padding: 12px 16px;
  z-index: 1;
}

.dropdown:hover > .dropdown-content {
  display: block;
}

.weather-container {
  max-width: 38rem;
  margin: 20px auto; /* Adjust margin as needed */
  display: flex;
  justify-content: space-between;
  background-color: rgba(255, 255, 255, 0.9); /* Add transparency to the container
background */
  padding: 2rem;
  border-radius: 10px;
  box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
}

.weather-input {
  flex: 1;
}

.clothing-recommendation {
  flex: 1;
  margin-left: 20px; /* Adjust margin as needed */
  text-align: center;
}

#clothing-image {
  max-width: 100%;
  border-radius: 5px;
  margin-top: 10px;
}
```

Index.js

```
function displayWeather() {
  var country = document.getElementById("searchData").value;

  fetch(`https://api.openweathermap.org/data/2.5/weather?q=${country}&appid=9fd7a449d055dba26a982a3220f32aa2`)
      .then((response) => {
          if (!response.ok) {
              throw new Error('Network response was not ok');
          }
          return response.json();
      })
      .then((data) => {
          console.log(data);
          document.getElementById("country").innerHTML = data.name;
          document.getElementById("desc").innerHTML = data.weather[0].description;
          document.getElementById("tempK").innerHTML = "Temperature: " + data.main.temp + " Kelvin";

          var celsius = data.main.temp - 273.15;

          document.getElementById("tempC").innerHTML = "Temperature: " + celsius.toFixed(2) + " Celsius";

          // Call the function to display clothing recommendation
          displayClothingRecommendation(celsius);
      })
      .catch((error) => {
          console.error('Error during fetch operation:', error);
      });
}

function displayClothingRecommendation(temperature) {
  var recommendationText = "";
  var clothingImageSrc = "";

  if (temperature < 10) {
      recommendationText = "It's cold! Wear a heavy coat, scarf, and gloves.";
      clothingImageSrc = "images/winter.jpg";
  } else if (temperature < 20) {
      recommendationText = "It's cool. A light jacket or sweater should be enough.";
      clothingImageSrc = "images/autumn.jpg";
  } else if (temperature < 30) {
      recommendationText = "It's warm. T-shirt and shorts would be comfortable.";
      clothingImageSrc = "images/spring.jpg";
  } else {
      recommendationText = "It's hot! Stay cool with light and breathable clothes.";
```

```
        clothingImageSrc = "images/summer.jpg";
    }

    document.getElementById("clothing-recommendation-text").innerHTML =
recommendationText;
    document.getElementById("clothing-image").src = clothingImageSrc;
}
```

The explanation of Home page code :
This code is for a simple web page that allows users to check the weather for a specified country and provides clothing recommendations based on the temperature.

1. **HTML (index.html):**
    - The HTML file contains the structure of the webpage.
    - It includes a header, navigation bar, a weather input section, and a clothing recommendation section.
    - The navigation bar has links for Home, Travel, and Local Events (with a dropdown for Reminder).
2. **CSS (index.css):**
    - The CSS file defines the styles for the webpage.
    - It sets the background, styles the header, navigation bar, and various sections on the page.
    - It also includes styles for a dropdown menu and the weather container.
3. **JavaScript (index.js):**
    - The JavaScript file defines two functions: **displayWeather** and **displayClothingRecommendation**.
    - The **displayWeather** function is triggered when the user clicks the "Enter" button after entering a country in the input field. It fetches weather data from the OpenWeatherMap API based on the entered country.
    - The temperature is displayed in both Kelvin and Celsius, and the **displayClothingRecommendation** function is called to provide clothing recommendations based on the temperature.
    - The **displayClothingRecommendation** function sets a recommendation text and an image source based on temperature ranges, and updates the corresponding HTML elements.
4. **API Request:**
    - The weather data is fetched from the OpenWeatherMap API using the **fetch** function.
    - The API key is included in the URL for authentication.
    - The fetched data is processed in the **then** block, updating HTML elements with information such as country name, weather description, and temperature.
5. **Clothing Recommendations:**
    - The clothing recommendations are determined based on the temperature.
    - Different recommendations and images are provided for temperature ranges below 10°C (cold), between 10°C and 20°C (cool), between 20°C and 30°C (warm), and above 30°C (hot).

Overall, this code combines HTML for the structure, CSS for styling, and JavaScript for dynamic behavior, creating a simple webpage that fetches and displays weather information along with clothing recommendations.

The code of Travel Page :

Travel.html:

```
<!-- travel.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>WeatherMaster Travel</title>
  <link rel="stylesheet" href="travel.css">
</head>
<body>
  <header>
    <h1>WeatherMaster Travel</h1>
  </header>

  <nav>
    <ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="travel.html">Travel</a></li>
      <li>
        <div class="dropdown">
          <a href="event.html">Local Events</a>
          <div class="dropdown-content">
            <a href="todo.html">Reminder</a>
          </div>
        </div>
      </li>
    </ul>
  </nav>

  <section id="tripArticles">
    <article class="seasonal-article" id="introduction">
      <h2>A Globetrotter's Guide: Seasonal Travel Tips for Every Adventure</h2>
      <p>Traveling the world is an exhilarating experience that offers a kaleidoscope
of cultures, landscapes, and flavors. To make the most of your journeys, it's crucial
to tailor your travel plans according to the seasons. Each season brings its own charm,
and understanding how to navigate through them can enhance your overall travel
experience. In this comprehensive guide, we'll explore travel tips for each season,
covering clothing essentials, must-try foods, selecting the perfect accommodations, and
staying safe on the road.</p>
    </article>

    <article class="seasonal-article" id="spring">
        <h2>Spring: Blossoms and Renewal</h2>
        <h2>Clothing Tips</h2>
        <p>As spring ushers in a symphony of blooming flowers and mild temperatures,
it's wise to pack layers. A mix of light jackets, long sleeves, and comfortable walking
shoes will keep you prepared for varying weather conditions.</p>
```

```
        <h2>Must-Try Foods</h2>
        <p>Embrace the fresh produce of spring with dishes like cherry blossoms tea in Japan, fava beans in Italy, or artichokes in France. Indulge in seasonal delights that reflect the local culture and culinary traditions.</p>
        <h2>Choosing Accommodations</h2>
        <p>Opt for accommodations with garden views or outdoor spaces to fully appreciate the beauty of spring. Research boutique hotels or bed-and-breakfasts that offer personalized experiences in line with the season's theme of renewal.</p>
        <h2>Safety Tips</h2>
        <p>Stay hydrated, especially if you're exploring on foot. Be aware of allergy triggers if you suffer from seasonal allergies, and carry any necessary medications.</p>
    </article>

    <article class="seasonal-article" id="summer">
        <h2>Summer: Sun-soaked Adventures</h2>
        <h2>Clothing Tips</h2>
        <p>Pack lightweight, breathable fabrics, and don't forget your sunglasses and sunscreen. In tropical destinations, consider swimwear and a hat for added sun protection.</p>
        <h2>Must-Try Foods</h2>
        <p>Sample seasonal fruits like mangoes in Thailand, watermelon in Greece, or barbecue delights in the Southern United States. Summer is the perfect time for outdoor food festivals and street markets.</p>
        <h2>Choosing Accommodations</h2>
        <p>Look for hotels with pools or easy access to beaches to cool off during the hot summer days. Prioritize air-conditioned rooms, especially in regions with scorching temperatures.</p>
        <h2>Safety Tips</h2>
        <p>Stay hydrated and avoid excessive sun exposure. Be cautious of heat-related illnesses, and familiarize yourself with emergency services in the area.</p>
    </article>

    <article class="seasonal-article" id="winter">
        <h2>Autumn: Harvest and Cultural Richness</h2>
        <h2>Clothing Tips:</h2>
        <p>Pack layers for fluctuating temperatures and bring a cozy sweater or jacket for cool evenings. Comfortable walking shoes are essential for exploring fall foliage.</p>
        <h2>Must-Try Foods:</h2>
        <p>Indulge in hearty dishes like pumpkin soup in the United States, chestnuts in Spain, or truffles in Italy. Fall is a season of culinary abundance, so savor the local harvest.</p>
        <h2>Choosing Accommodations:</h2>
        <p>Consider staying in quaint countryside inns or lodges to fully immerse yourself in the autumnal beauty. Research accommodations near vineyards for wine regions in their prime during this season.</p>
        <h2>Safety Tips:</h2>
```

```
        <p>Be cautious of wet and slippery surfaces due to falling leaves. Check
weather forecasts and be prepared for sudden temperature changes.</p>
    </article>

    <article class="seasonal-article" id="winter">
        <h2>Winter: Snowy Adventures and Festive Celebrations</h2>
        <h2>Clothing Tips </h2>
        <p>Pack layers, thermal wear, and waterproof gear for snowy destinations. Don't
forget gloves, a hat, and insulated boots for added warmth.</p>
        <h2>Must-Try Foods </h2>
        <p>Indulge in winter comfort foods like fondue in Switzerland, hot pot in
China, or mulled wine in Germany. Explore local holiday markets for festive treats.</p>
        <h2>Choosing Accommodation </h2>
        <p>Opt for accommodations with fireplaces or heated amenities, especially in
cold climates. Research hotels near winter festivals or holiday events for a truly
magical experience.</p>
        <h2>Safety Tips </h2>
        <p>Be cautious of icy conditions and dress appropriately for cold temperatures.
Stay informed about local weather conditions and plan activities accordingly.</p>
    </article>

    <article class="seasonal-article" id="conclusion">
      <p>By tailoring your travel plans to the seasons, you can maximize the joy of
exploring diverse destinations around the world. From clothing choices to culinary
delights and safe accommodation options, this guide ensures that your journeys are not
only memorable but also enjoyable throughout the year. Happy travels!</p>
    </article>
  </section>
</body>
</html>
```

Travel.css:

```
body {
    font-family: 'Segoe UI', Roboto, Helvetica, Arial, sans-serif;
    margin: 0;
    padding: 0;
    background: url('images/day.gif') center/cover no-repeat; /* Replace with the path
to your GIF file */
  }
header {
    background-color: #3498db;
    padding: 15px;
    text-align: center;
    color: white;
}
/* Navigation Bar Styles */
nav {
    display: flex;
    justify-content: space-around;
```

```
    background-color: #2980b9;
    padding: 10px;
}

nav a {
    color: white;
    text-decoration: none;
    padding: 10px;
    cursor: pointer;
}

nav a:hover {
    background-color: #2c3e50;
}

ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

li {
    display: inline;
}

.dropdown {
    position: relative;
    display: inline-block;
}

.dropdown-content {
    display: none;
    position: absolute;
    background-color: #2c3e50;
    min-width: 160px;
    box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);
    padding: 12px 16px;
    z-index: 1;
}

.dropdown:hover > .dropdown-content {
    display: block;
}

.seasonal-article {
    max-width: 800px;
    margin: 20px auto;
    background-color: #ffffff; /* White article background */
    border-radius: 8px;
```

```
    padding: 20px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.seasonal-article h2 {
    color: #0066cc; /* Deep blue heading color */
}

.seasonal-article p {
    color: #555;
}
```

The explainantion of Trave page code:

This code is for a webpage that provides travel tips for different seasons, including information on clothing, must-try foods, choosing accommodations, and safety tips.

1. **HTML (Travel.html):**
    - The HTML file defines the structure of the travel webpage.
    - It includes a header, navigation bar, and a section (**tripArticles**) containing multiple articles with information about seasonal travel tips.
    - Each article covers a specific season (Spring, Summer, Autumn, Winter), including relevant tips and recommendations.
2. **CSS (Travel.css):**
    - The CSS file defines the styles for the travel webpage.
    - It sets the background, styles the header, navigation bar, and the articles.
    - The navigation bar styles include a dropdown menu with hover effects.
    - The articles have a white background, rounded corners, and a subtle box shadow for a card-like appearance.
    - Heading and paragraph styles are specified for the articles, with specific colors for better readability.
3. **Content (Travel.html - Articles):**
    - Each article within the **tripArticles** section provides information for a specific season (Spring, Summer, Autumn, Winter).
    - Information is categorized into Clothing Tips, Must-Try Foods, Choosing Accommodations, and Safety Tips for each season.
    - The articles provide practical advice for travelers based on the characteristics of each season.

In summary, this code creates a visually appealing and informative webpage for travelers, offering insights into how to plan for different seasons. The layout is clean, and the use of CSS enhances the visual presentation of the content. The articles provide practical tips that can help travelers make the most of their journeys throughout the year.

The code of Local Events page :

Event.html :

```
<!DOCTYPE html>
<!-- Coding By CodingNepal - youtube.com/codingnepal -->
<html lang="en" dir="ltr">
<head>
  <meta charset="utf-8">
  <title>Local Event | Calendar</title>
  <link rel="stylesheet" href="event.css">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <!-- Google Font Link for Icons -->
  <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Material+Symbols+Rounded:opsz,wght,FILL,
GRAD@20..48,100..700,0..1,-50..200">
  <script src="event.js" defer></script>
</head>
<body>
    <header>
        <h1>Calendar</h1>
      </header>

      <nav>
        <ul>
          <li><a href="index.html">Home</a></li>
          <li><a href="travel.html">Travel</a></li>
          <li>
            <div class="dropdown">
              <a href="event.html">Local Events</a>
              <div class="dropdown-content">
                <a href="todo.html">Reminder</a>
              </div>
            </div>
          </li>
        </ul>
      </nav>
    <div class="cal-container">
        <div class="wrapper"></div>
            <div class="custom-header">
            <p class="current-date"></p>
                <div class="icons">
                    <span id="prev" class="material-symbols-
rounded">chevron_left</span>
                    <span id="next" class="material-symbols-
rounded">chevron_right</span>
                </div>
            </div>
            <div class="calendar">
```

```
            <ul class="weeks">
                <li>Sun</li>
                <li>Mon</li>
                <li>Tue</li>
                <li>Wed</li>
                <li>Thu</li>
                <li>Fri</li>
                <li>Sat</li>
            </ul>
            <ul class="days"></ul>
            </div>
        </div>
    </div>
</body>
</html>
```

Event.js:

```
const daysTag = document.querySelector(".days"),
currentDate = document.querySelector(".current-date"),
prevNextIcon = document.querySelectorAll(".icons span");

// getting new date, current year and month
let date = new Date(),
currYear = date.getFullYear(),
currMonth = date.getMonth();

// storing full name of all months in array
const months = ["January", "February", "March", "April", "May", "June", "July",
              "August", "September", "October", "November", "December"];

const renderCalendar = () => {
    let firstDayofMonth = new Date(currYear, currMonth, 1).getDay(), // getting first
day of month
    lastDateofMonth = new Date(currYear, currMonth + 1, 0).getDate(), // getting last
date of month
    lastDayofMonth = new Date(currYear, currMonth, lastDateofMonth).getDay(), //
getting last day of month
    lastDateofLastMonth = new Date(currYear, currMonth, 0).getDate(); // getting last
date of previous month
    let liTag = "";

    for (let i = firstDayofMonth; i > 0; i--) { // creating li of previous month last
days
        liTag += `<li class="inactive">${lastDateofLastMonth - i + 1}</li>`;
    }

    for (let i = 1; i <= lastDateofMonth; i++) { // creating li of all days of current
month
        // adding active class to li if the current day, month, and year matched
```

```
        let isToday = i === date.getDate() && currMonth === new Date().getMonth()
                     && currYear === new Date().getFullYear() ? "active" : "";
        liTag += `<li class="${isToday}">${i}</li>`;
    }

    for (let i = lastDayofMonth; i < 6; i++) { // creating li of next month first days
        liTag += `<li class="inactive">${i - lastDayofMonth + 1}</li>`
    }
    currentDate.innerText = `${months[currMonth]} ${currYear}`; // passing current mon
and yr as currentDate text
    daysTag.innerHTML = liTag;
}
renderCalendar();

prevNextIcon.forEach(icon => { // getting prev and next icons
    icon.addEventListener("click", () => { // adding click event on both icons
        // if clicked icon is previous icon then decrement current month by 1 else
increment it by 1
        currMonth = icon.id === "prev" ? currMonth - 1 : currMonth + 1;

        if(currMonth < 0 || currMonth > 11) { // if current month is less than 0 or
greater than 11
            // creating a new date of current year & month and pass it as date value
            date = new Date(currYear, currMonth, new Date().getDate());
            currYear = date.getFullYear(); // updating current year with new date year
            currMonth = date.getMonth(); // updating current month with new date month
        } else {
            date = new Date(); // pass the current date as date value
        }
        renderCalendar(); // calling renderCalendar function
    });
});
```

Event.css:

```
/* Import Google font - Poppins */
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600&display=swap');

body {
    font-family: 'Segoe UI', Roboto, Helvetica, Arial, sans-serif;
    margin: 0;
    padding: 0;
    background: url('images/day.gif') center/cover no-repeat; /* Replace with the path
to your GIF file */
  }

  header {
    background-color: #3498db;
    padding: 15px;
```

```
  text-align: center;
  color: white;
}

nav {
  display: flex;
  justify-content: space-around;
  background-color: #2980b9;
  padding: 10px;
}

nav a {
  color: white;
  text-decoration: none;
  padding: 10px;
  cursor: pointer;
}

nav a:hover {
  background-color: #2c3e50;
}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

li {
  display: inline;
}

.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #2c3e50;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  padding: 12px 16px;
  z-index: 1;
}

.dropdown:hover > .dropdown-content {
  display: block;
```

```
  }
  .cal-container {
    max-width: 38rem;
    margin: auto;
    background-color: rgba(255, 255, 255, 0.9); /* Add transparency to the container
background */
    padding: 2rem;
    border-radius: 10px;
    box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
  }
.wrapper {
    width: 450px;
    margin: auto;
  }
.custom-header {
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 25px 30px 10px;
  background-color: #ecf0f1;
}

.custom-header .icons {
  display: flex;
}

.custom-header .icons span {
  height: 38px;
  width: 38px;
  margin: 0 1px;
  cursor: pointer;
  color: #878787;
  text-align: center;
  line-height: 38px;
  font-size: 1.9rem;
  user-select: none;
  border-radius: 50%;
}

.custom-header .icons span:last-child {
  margin-right: -10px;
}

.custom-header .icons span:hover {
  background: #f2f2f2;
}

.custom-header .current-date {
  font-size: 1.45rem;
```

```
  font-weight: 500;
}

.calendar {
  padding: 20px;
  background-color: #fff;
  border-radius: 10px;
  box-shadow: 0 15px 40px rgba(0, 0, 0, 0.12);
}

.calendar ul {
  display: flex;
  flex-wrap: wrap;
  list-style: none;
  text-align: center;
}

.calendar .days {
  margin-bottom: 20px;
}

.calendar li {
  color: #333;
  width: calc(100% / 7);
  font-size: 1.07rem;
}

.calendar .weeks li {
  font-weight: 500;
  cursor: default;
}

.calendar .days li {
  z-index: 1;
  cursor: pointer;
  position: relative;
  margin-top: 30px;
}

.days li.inactive {
  color: #aaa;
}

.days li.active {
  color: #fff;
}

.days li::before {
  position: absolute;
```

```
  content: "";
  left: 50%;
  top: 50%;
  height: 40px;
  width: 40px;
  z-index: -1;
  border-radius: 50%;
  transform: translate(-50%, -50%);
}

.days li.active::before {
  background: #9B59B6;
}

.days li:not(.active):hover::before {
  background: #f2f2f2;
}
```

The explanation of Local Events page code :

This code is for a web page that serves as a local events calendar. Users can navigate through the months, view the days of the week, and click on specific days to potentially view local events or reminders.

1. **HTML (Event.html):**
    - The HTML file defines the structure of the local events page.
    - It includes a header, navigation bar, and a calendar container.
    - The calendar container has a wrapper for styling, a custom header displaying the current date and navigation icons, and a calendar section with lists for weeks and days.
2. **JavaScript (Event.js):**
    - The JavaScript file handles the dynamic functionality of the calendar.
    - It retrieves elements using **document.querySelector** for the days, current date, and navigation icons.
    - It initializes the current date and month and defines a function **renderCalendar** to display the calendar dynamically based on the current date and month.
    - The script adds event listeners to the previous and next icons, allowing users to navigate through months.
    - The **renderCalendar** function generates the HTML for the calendar, including inactive days from the previous and next months, and highlights the current day.
    - The Google Material Symbols Rounded font is used for chevron icons.
3. **CSS (Event.css):**
    - The CSS file defines styles for the entire page, including the body, header, navigation bar, dropdown, and the calendar container.
    - It styles the custom header, navigation icons, and the calendar with specific colors, fonts, and layouts.
    - The calendar has a responsive design with a transparent background and a box shadow for a clean and visually appealing look.
    - Hover effects are added to the navigation icons and inactive days in the calendar.

Overall, the code combines HTML for structure, CSS for styling, and JavaScript for dynamic behavior, creating a functional and visually appealing local events calendar. The calendar allows users to navigate through months and highlights the current day, providing a potentially interactive interface for displaying local events or reminders.

The code for Reminder page :

todo.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Travel Planning - WeatherMaster</title>
  <link rel="stylesheet" href="todo.css">
</head>
<body>
  <header>
    <h1>WeatherMaster - Travel Planning</h1>
  </header>

  <nav>
    <ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="travel.html">Travel</a></li>
      <li>
        <div class="dropdown">
          <a href="event.html">Local Events</a>
          <div class="dropdown-content">
            <a href="todo.html">Reminder</a>
          </div>
        </div>
      </li>
    </ul>
  </nav>

  <div class="search-container">
    <label for="searchPlace">Search by Place:</label>
    <input type="text" id="searchPlace" placeholder="Enter place to search">
    <button onclick="searchEvents()" id="searchButton">Search</button>
  </div>

  <div class="weather-container">
    <div class="form-container">
      <div class="background-container">
        <h2>Travel Planning</h2>
        <p>Plan your trips with confidence by integrating weather information for
selected destinations and dates.</p>
      </div>

      <!-- Form for adding a new event -->
      <form id="eventForm">
        <label for="eventDate">Date:</label>
        <input type="date" id="eventDate" required>
```

```
        <label for="eventTime">Time:</label>
        <input type="time" id="eventTime" required>

        <label for="eventPlace">Place to Travel:</label>
        <input type="text" id="eventPlace" required>

        <label for="thingsToDo">Things to Do:</label>
        <textarea id="thingsToDo" rows="4" required></textarea>

        <button type="button" onclick="addEvent()">Add Event</button>
      </form>
    </div>

    <!-- Displaying events in a calendar -->
    <div class="event-container" id="eventCalendar"></div>
  </div>

  <script src="todo.js" defer></script>
</body>
</html>
```

todo.js:

```
document.addEventListener('DOMContentLoaded', function () {
    // Load events from local storage on page load
    loadEvents();

    // Add event listener to the form for adding events
    document.getElementById('eventForm').addEventListener('submit', function (event) {
        event.preventDefault();
        addEvent();
    });

    // Initialize the search button event listener
    document.getElementById('searchButton').addEventListener('click', searchEvents);

    // Add event listener to the download button
    document.getElementById('downloadButton').addEventListener('click',
downloadEvents);
});

// Function to add a new event
function addEvent() {
    const date = document.getElementById('eventDate').value;
    const time = document.getElementById('eventTime').value;
    const place = document.getElementById('eventPlace').value;
    const thingsToDo = document.getElementById('thingsToDo').value;

    if (date && time && place && thingsToDo) {
        const newEvent = {
```

```
            date: date,
            time: time,
            place: place,
            thingsToDo: thingsToDo,
        };

        // Get existing events from local storage
        let events = getEventsFromLocalStorage();

        // Check if the event already exists based on the date
        const existingEventIndex = events.findIndex(event => event.date === date);

        if (existingEventIndex !== -1) {
            // If the event already exists, update it
            events[existingEventIndex] = newEvent;
            alert('Event updated successfully!');
        } else {
            // If the event does not exist, add it to the array
            events.push(newEvent);
            alert('Event added successfully!');
        }

        // Save the updated events array to local storage and text file
        saveEvents(events);

        // Reload events and clear the form
        loadEvents();
        clearForm();
    } else {
        alert('Please fill in all fields.');
    }
}

// Function to save events to a text file and local storage
function saveEvents(events) {
    saveEventsToFile(events);

    // Optional: Save events to local storage as well
    localStorage.setItem('events', JSON.stringify(events));
}

// Function to download events as a text file
function downloadEvents() {
    const events = getEventsFromLocalStorage();
    saveEventsToFile(events);
}

// Function to save events to a text file
function saveEventsToFile(events) {
```

```
    const eventsText = events.map(event => `${event.date}, ${event.time},
${event.place}, ${event.thingsToDo}`).join('\n');
    const blob = new Blob([eventsText], { type: 'text/plain' });
    const link = document.createElement('a');
    link.href = URL.createObjectURL(blob);
    link.download = 'events.txt';
    link.click();
}

// Function to load events from local storage and display them
function loadEvents() {
    const eventCalendar = document.getElementById('eventCalendar');
    eventCalendar.innerHTML = '';

    // Get events from local storage
    const events = getEventsFromLocalStorage();

    // Display events in the calendar
    events.forEach(function (event) {
        const eventItem = document.createElement('div');
        eventItem.classList.add('event-item');
        eventItem.innerHTML = `<strong>Date:</strong> ${event.date}<br>
                               <strong>Time:</strong> ${event.time}<br>
                               <strong>Place:</strong> ${event.place}<br>
                               <strong>Things to Do:</strong> ${event.thingsToDo}<br>
                               <button
onclick="editEvent('${event.date}')">Edit</button>
                               <button
onclick="deleteEvent('${event.date}')">Delete</button>`;
        eventCalendar.appendChild(eventItem);
    });
}

// Function to edit an event
function editEvent(date) {
    // Get existing events from local storage
    const events = getEventsFromLocalStorage();

    // Find the event with the specified date
    const selectedEvent = events.find(event => event.date === date);

    if (selectedEvent) {
        // Populate the form with the selected event's details
        document.getElementById('eventDate').value = selectedEvent.date;
        document.getElementById('eventTime').value = selectedEvent.time;
        document.getElementById('eventPlace').value = selectedEvent.place;
        document.getElementById('thingsToDo').value = selectedEvent.thingsToDo;
    } else {
        alert('Event not found.');
```

```
    }
}

// Function to delete an event
function deleteEvent(date) {
    // Get existing events from local storage
    let events = getEventsFromLocalStorage();

    // Remove the event with the specified date
    events = events.filter(function (event) {
        return event.date !== date;
    });

    // Save the updated events array to local storage
    saveEventsToLocalStorage(events);

    // Reload events
    loadEvents();
}

// Function to search events based on the entered place
function searchEvents() {
    const searchPlace = document.getElementById('searchPlace').value.toLowerCase();

    // Get events from local storage
    const events = getEventsFromLocalStorage();

    // Filter events based on the entered place
    const filteredEvents = events.filter(function (event) {
        return event.place.toLowerCase().includes(searchPlace);
    });

    // Display filtered events
    const eventCalendar = document.getElementById('eventCalendar');
    eventCalendar.innerHTML = '';

    filteredEvents.forEach(function (event) {
        const eventItem = document.createElement('div');
        eventItem.classList.add('event-item');
        eventItem.innerHTML = `<strong>Date:</strong> ${event.date}<br>
                               <strong>Time:</strong> ${event.time}<br>
                               <strong>Place:</strong> ${event.place}<br>
                               <strong>Things to Do:</strong> ${event.thingsToDo}<br>
                               <button
onclick="editEvent('${event.date}')">Edit</button>
                               <button
onclick="deleteEvent('${event.date}')">Delete</button>`;
        eventCalendar.appendChild(eventItem);
    });
```

```
}

// Function to get events from local storage
function getEventsFromLocalStorage() {
    const eventsString = localStorage.getItem('events');
    return eventsString ? JSON.parse(eventsString) : [];
}

// Function to save events to local storage
function saveEventsToLocalStorage(events) {
    localStorage.setItem('events', JSON.stringify(events));
}

// Function to clear the form after adding or editing an event
function clearForm() {
    document.getElementById('eventForm').reset();
}
```

todo.css

```
body {
  font-family: 'Segoe UI', Roboto, Helvetica, Arial, sans-serif;
  margin: 0;
  padding: 0;
  background: url('images/day.gif') center/cover no-repeat; /* Replace with the path to
your GIF file */
}

  header {
    background-color: #3498db;
    padding: 15px;
    text-align: center;
    color: white;
  }

  /* Navigation Bar Styles */
  nav {
    display: flex;
    justify-content: space-around;
    background-color: #2980b9;
    padding: 10px;
  }

  nav a {
    color: white;
    text-decoration: none;
    padding: 10px;
    cursor: pointer;
  }
```

```css
nav a:hover {
  background-color: #2c3e50;
}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

li {
  display: inline;
}

.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #2c3e50;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);
  padding: 12px 16px;
  z-index: 1;
}

.dropdown:hover > .dropdown-content {
  display: block;
}

/* Search Bar Styles */
.search-container {
  margin-top: 40px;
  margin-bottom: 30px;
  padding: 10px;
  background-color: rgba(255, 255, 255, 0.8);
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  display: flex;
  align-items: center;
  justify-content: center;
  width: 780px;
  margin-left: auto;
  margin-right: auto;
}
```

```css
#searchPlace {
  flex: 1;
  margin-right: 10px;
  margin-left: 10px;
  margin-top: 15px;
  padding: 10px; /* Adjusted padding */
  font-size: 14px;
  width: 150px;
}

button {
  padding: 10px;
  background-color: #555;
  color: white;
  cursor: pointer;
  border: none;
  border-radius: 4px;
}

button:hover {
  background-color: #333;
}

/* Weather Container and Layout Styles */
.weather-container {
  max-width: 800px;
  margin: 20px auto;
  display: flex;
  justify-content: space-between;
}

.form-container {
  flex: 1;
  margin-right: 20px;
}

.event-container {
  flex: 1;
  background-color: rgba(255, 255, 255, 0.8);
  padding: 20px;
  border-radius: 10px;
  margin-top: 20px;
  margin-left: 20px; /* Adjust as needed */
}

/* Background Container Styles */
.background-container {
  background-color: rgba(255, 255, 255, 0.8);
  padding: 20px;
```

```
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  margin-bottom: 20px;
  margin-top: 20px;
}

/* General Form and Event Styles */
.form-container h2,
.event-container h2,
.background-container h2,
.background-container p {
  margin: 0;
}

form {
  max-width: 400px;
  margin-left: auto;
  margin-right: auto;
  background-color: rgba(255, 255, 255, 0.8);
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

label {
  display: block;
  margin-bottom: 8px;
}

input,
textarea {
  margin-bottom: 15px;
  width: calc(100% - 16px);
  padding: 8px;
  box-sizing: border-box;
  border: 1px solid #ccc;
  border-radius: 4px;
}
```

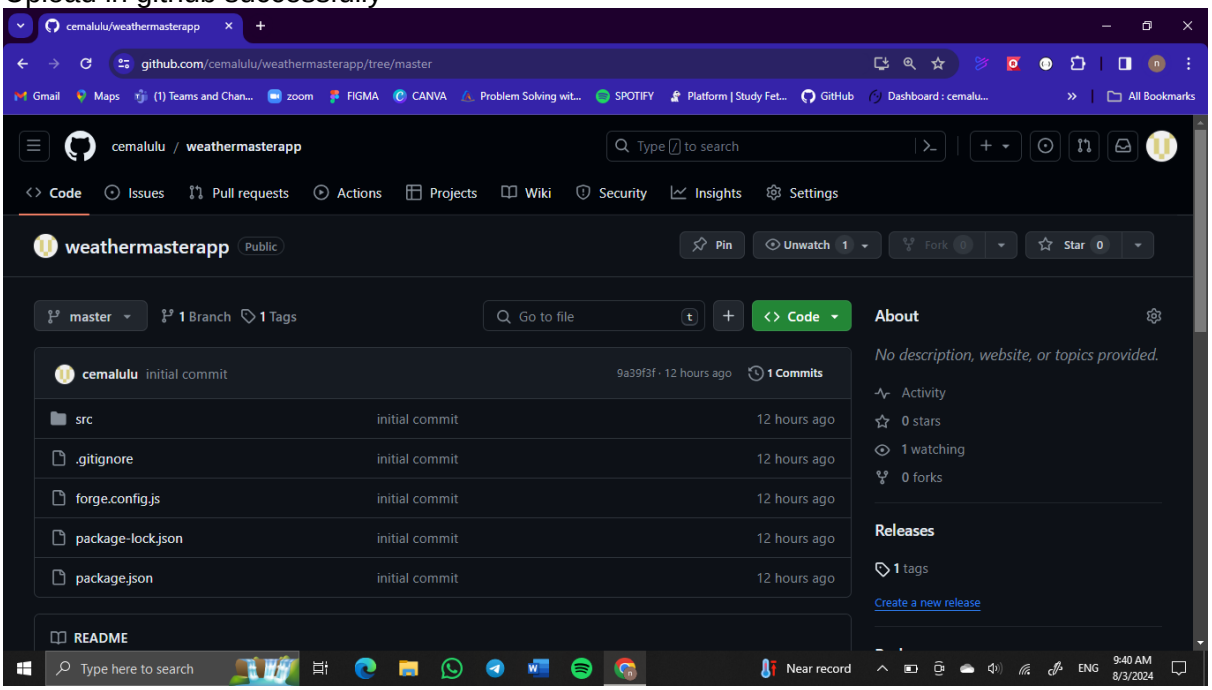
The explanation of the Reminder page code :

This code creates a web page for managing and planning travel events with a focus on reminders.

1. **HTML (todo.html):**
    - The HTML file structures the travel planning page.
    - It includes a header, navigation bar, search container, weather container with a form for adding events, and a container for displaying events.
    - The form includes fields for date, time, place, and things to do.
    - The page references an external stylesheet (todo.css) and JavaScript file (todo.js).
2. **JavaScript (todo.js):**
    - The JavaScript file adds functionality to the travel planning page.
    - It waits for the DOM to be fully loaded (**DOMContentLoaded**) before executing the code.
    - Event listeners are added to handle form submission, searching events, and downloading events.
    - Functions are defined for adding, editing, deleting, and searching events.
    - The code uses local storage to store and retrieve events, and it provides functionality to save events as a text file.
    - The **loadEvents** function dynamically displays events on the page.
    - The **editEvent** function populates the form with details of the selected event for editing.
    - The **deleteEvent** function removes an event from the list.
    - The **searchEvents** function filters events based on the entered place and displays the results.
3. **CSS (todo.css):**
    - The CSS file styles the entire page, including the body, header, navigation bar, dropdown, search container, weather container, form, and event container.
    - It defines styles for the search bar, buttons, layout, and background containers.
    - The layout is designed to be responsive, with adjustments for padding, margins, and widths.
    - Different sections of the page have specific background colors, border-radius, and box-shadow for a visually appealing design.
4. **General Styling:**
    - The body has a background image (day.gif).
    - The header and navigation bar have consistent styling with blue tones.
    - The dropdown in the navigation bar has a dark background color.
    - The search container has a semi-transparent white background with rounded corners and a box shadow.
    - The weather container and form container have specific widths and margins for layout.

Overall, this code creates a user-friendly travel planning page where users can add, edit, delete, and search for events. The styling enhances the visual appeal and user experience of the page. The code utilizes JavaScript to handle dynamic functionality, and local storage for data persistence.

Upload in github successfully



The link to the github:
https://github.com/cemalulu/weathermasterapp.git