KOLEJ PROFESIONAL MARA BERANANG

DIPLOMA IN COMPUTER SCIENCE

COURSE NAME	:	**OBJECT ORIENTED PROGRAMMING**
COURSE CODE	:	**CSC2744**
ACADEMIC SESSION	:	**SESSION 2 2024/2025**
TYPE OF ASSESSMENT	:	**FINAL ASSIGNMENT**
DURATION	:	**10/10/2024 - 30/10/2024**

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	**SYAFIQ IZZUDDIN BIN RAZAZLI**
I/D Number	**BCS2307-093**
Class	**4A**
Lecturer	**NUR AKMAL HAFIZAN BINTI KAMAL IQBAL**

Section / Question No.	**Marks**
1	
2	
3	
4	
5	
6	
Total	**/ 50**

Question

In JavaScript development, **Electron** is a framework that allows developers to build cross-platform desktop applications using web technologies like HTML, CSS, and JavaScript. It wraps a web application (which would normally run in a browser) inside a desktop environment, enabling it to run as a native application on platforms like Windows, macOS, and Linux.

Many popular desktop applications, like Visual Studio Code, Slack, Discord, and Spotify (desktop version), are built using Electron. As part of your software development course, you have been tasked to develop a desktop application using Electron with web technologies like HTML, CSS, JavaScript and utilizes the third-party data from one of the given API. You also need to incorporate CRUD (Create, Read, Update, Delete) operations to enhance user interaction and usability.

Name of Application	Key Features:
World Explorer	• Helps travelers plan trips by providing key information about their destinations and nearby countries they can visit. • Display key information like area, continent, region and subregion, capital city, language spoken, time zone, flag & coat of arms, maps, location, population. • CRUD – personalized travel itineraries and keep important country-specific info. https://restcountries.com/v3.1/all
WeatherWise	• Weather insights for outdoor activities which provide recommendations based on weather forecast like hiking, paragliding etc. • Display key information like location name, region, country, local time, current temperature, forecast temperature, icon, wind speed, humidity and related data, weather condition, chances of snow/rain • CRUD - outdoor activities-related information / checklist of the required equipment for the activities. https://api.weatherapi.com/v1/forecast.json?key=32804b24a847407391c53709241010&q=London

Name of Application	Key Features:
Traveller Advisories	• Travel advisories that give real-time and forecast weather conditions and clothing recommendations based on weather. • Display key information like location name, region, country, local time, current temperature, feels like temperature, forecast temperature, sunset, sunrise, icon, wind speed, humidity, weather description. • CRUD - personalized travel itineraries and activities related to the weather information. https://api.weatherapi.com/v1/forecast.json?key=32804b24a847407391c53709241010&q=London
Beauty Finder	• Personalized product recommendations based on favorite brands and price range. • Display key information like brand, name, price, product image, website link, product link, product description, rating and product type. • CRUD – create a wish list or favorite list for products that suit the user preferences. http://makeup-api.herokuapp.com/api/v1/products.json?brand=maybelline
Meal Finder	• Meal suggestion and recipe filtered by ingredients and meal category. • Display key information like meal name, category, category description, area, measurement and ingredients, instructions on how to cook, picture of the meal, YouTube source. • CRUD – meal planner and groceries list https://www.themealdb.com/api.php

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. Include explanation of the information retrieved from the given API and CRUD process.
    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 and include the link in the report.

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.	
		• Unable to display all 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.	• Able to display extra data from the API beyond the application requirements.	• Able to display extra data from the API beyond the application requirements with the description in the report		
		• 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	• 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	• 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	

				concept of the application.	atmosphere that expresses the concept of the application.	produce an atmosphere that expresses the concept of the application.	
Convey	5. Produce a report on your application. Include the following: i. Overview of the application and its purpose. Include explanation of the information retrieved from the given API and CRUD process.	• The overview of the application and is vague	• The overview of the application and is very brief.	• The overview of the application and its purposes is clearly described.	• Include explanation of the information retrieved from the API and CRUD process with its purposes.		
	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 and include the link in the report.	• Does not submit complete electron files in GitHub	• Completely submit all the electron file in GitHub and include the link in the report.				
Total Marks Earned							**/50**
Total Percentage (40%)							**/40%**

Produce a report on your application. Include the following:

a. Overview of the application and its purpose. Include explanation of the information retrieved from the given API and CRUD process..

WeatherWise is an application designed to provide users with insights and recommendations for outdoor activities based on current weather forecasts. This application aims to assist users in planning their activities such as hiking, paragliding, and other outdoor adventures by supplying relevant weather information.

The Weather API (in this case, from weatherapi.com) provides a wealth of information that is vital for making informed decisions about outdoor activities. When you send a request to the endpoint.

The information retrieved from the API will be used to:

- Display a weather summary for users, allowing them to assess whether the conditions are suitable for their planned activities.
- Provide recommendations for activities based on current and forecasted weather conditions.
- Help users prepare for their outings by understanding potential weather-related challenges.

In addition to providing weather data, **WeatherWise** will allow users to manage outdoor activity-related information through CRUD operations:

- **Create:**
    - **Purpose:** To add new activities or checklists for specific outdoor events.
    - **Example:** Users can create a new hiking event, specifying details like the date, location, and required equipment.
    - **API Interaction:** Use a POST request to save this information in the database.

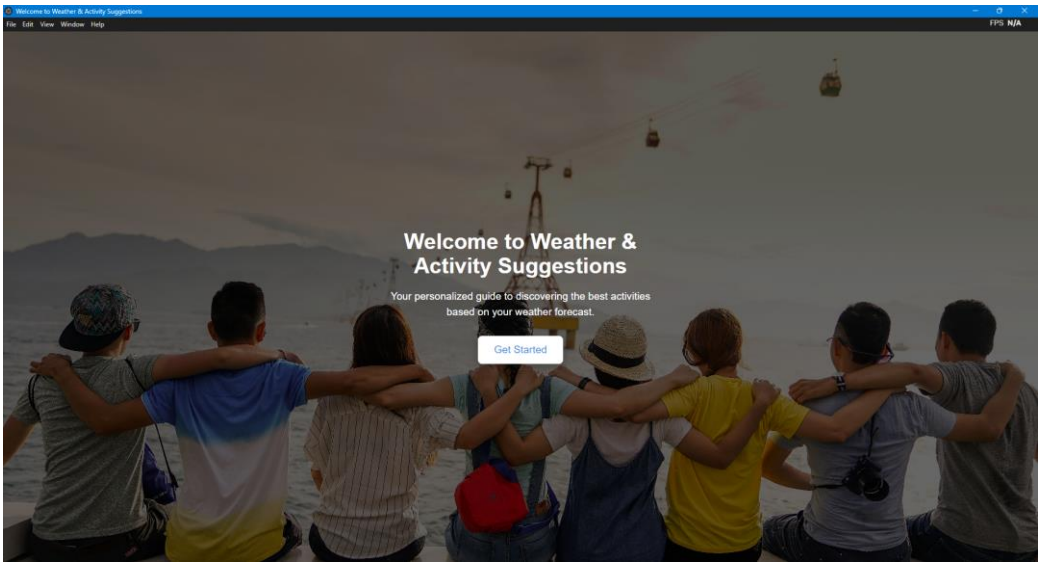
- **Read:**
    - **Purpose:** To retrieve and display existing outdoor activities and their related checklists.
    - **Example:** Users can view their planned activities, including details like location, date, and equipment needed.
    - **API Interaction:** Use a GET request to fetch this information from the database.

- **Update:**
    - **Purpose:** To modify existing activities or checklists.
    - **Example:** Users can update the date or location of a hiking trip or add/remove equipment from their checklist.
    - **API Interaction:** Use a PUT or PATCH request to send updated information.

- **Delete:**
    - **Purpose:** To remove activities or checklists that are no longer needed.
    - **Example:** Users can delete a canceled hiking event or remove an item from their equipment checklist.
    - **API Interaction:** Use a DELETE request to remove this information from the database.

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

Welcome Page:



Functionalities and Features

Welcome Page:

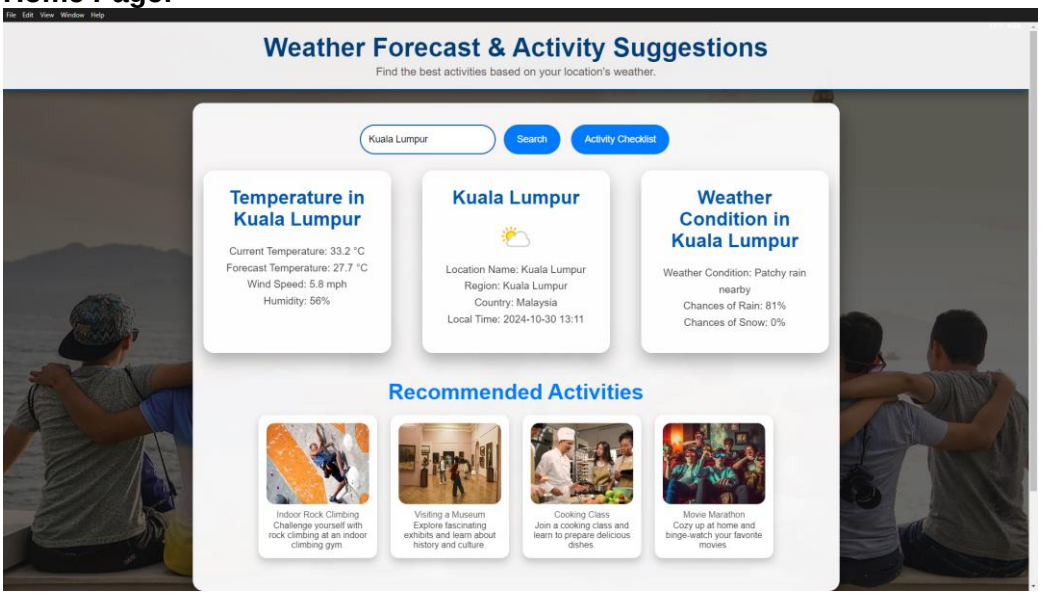
- **Welcome Message:** The homepage features a warm welcome message that introduces users to the application's purpose.
- **Personalized Guide:** A brief description informs users that the application will help them discover the best activities based on their local weather.
- **Call to Action:** A prominent "Get Started" button encourages users to begin their journey by navigating to the search page.

Explanation:

The **WeatherWise** application's homepage is designed to be visually appealing and user-friendly, providing essential information while inviting users to explore further. The combination of engaging text, effective call-to-action buttons, and animations ensures a positive first impression, encouraging users to delve into the features that will help them plan their outdoor activities based on weather forecasts.

To fully realize the application's potential, additional pages (such as weather details and activity management) would need similar attention to functionality and design, ensuring a cohesive user experience throughout.

Home Page:



Functionalities and Features

Home Page:

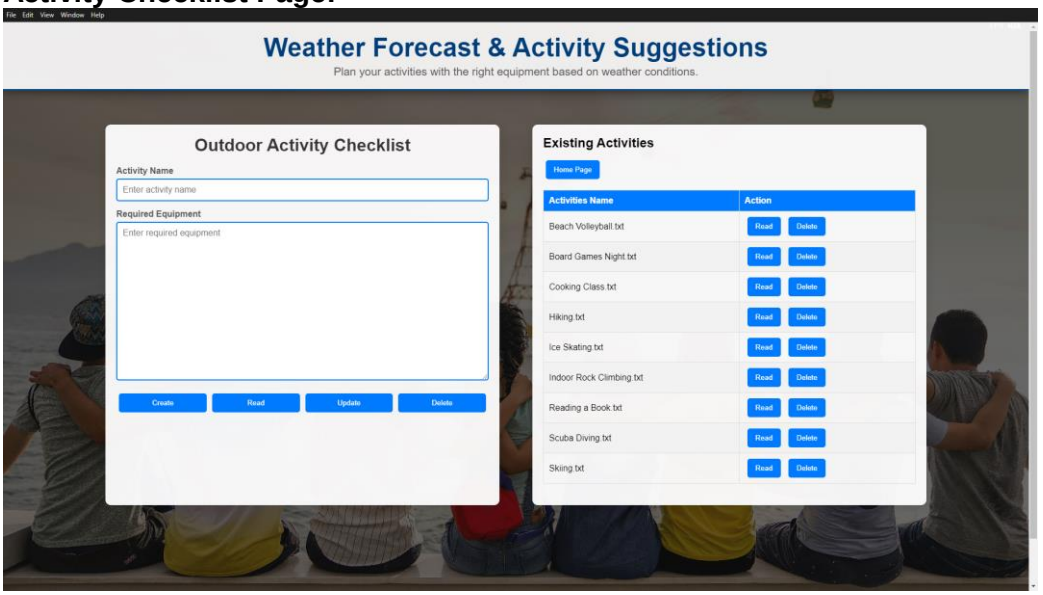
- **Dynamic Weather Fetching**: Upon entering a city and clicking "Search," the app fetches weather data from the Weather API (using fetch()) and displays relevant details.
- **Error Handling**: If the API fetch fails (due to an invalid city name or network issues), an error message is shown to the user.
- **Activity Display Logic**: Depending on the weather conditions returned, the corresponding activities are displayed. The categorization is based on keywords in the weather condition description (e.g., "rain," "snow," "cloud").

Explanation:

The Weather & Activities page is your go-to resource for real-time weather updates and personalized activity suggestions based on the current conditions in your chosen city. The layout is designed to be intuitive, featuring a convenient search bar for entering your city and dedicated sections that display the latest weather info along with recommended activities.

This page is all about making your experience seamless and enjoyable. By combining live weather updates with tailored activity ideas, it ensures that you have the most relevant information at your fingertips. Whether you're planning a sunny day hike or looking for indoor activities on a rainy day, the Weather & Activities page is here to enhance your outdoor adventures and help you make the most of every outing!

Activity Checklist Page:



Functionalities and Features

Activity Checklist Page:

- **Creating Activities**: Users can create new activities by entering an activity name and required equipment, which are saved as text files.
- **Reading Activities**: The application can read existing activities from the file system and populate the table with the activity names, providing buttons for further actions.
- **Updating Activities**: Users can select an existing activity to update its details, overwriting the previous content.
- **Deleting Activities**: Users can delete an activity, which removes it from the file system and updates the displayed list.

Explanation:

The Outdoor Activity Checklist application is designed to help outdoor enthusiasts easily manage their adventures. With this user-friendly tool, you can quickly input details about your activities and specify the equipment you'll need, all tailored to different weather conditions.

This app is perfect for anyone who loves spending time outdoors, making it simple to organize your plans and ensure you have everything you need for a great outing. Its clean and intuitive interface means you won't have to navigate through complicated menus—just create, view, and update your activities with ease.

c. Program codes of your system

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome to Weather & Activity Suggestions</title>
    <link rel="stylesheet" href="index.css">
</head>
<body>
    <div class="overlay">
        <div class="homepage-container">
            <h1>Welcome to Weather & Activity Suggestions</h1>
            <p>Your personalized guide to discovering the best activities based on your
weather forecast.</p>
            <button onclick="window.location.href='search.html'">Get Started</button>
        </div>
    </div>
</body>
</html>
```

Index.css

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: Arial, sans-serif;
}

body, html {
  height: 100%;
  width: 100%;
  overflow: hidden;
}

body {
  background: url('https://images.unsplash.com/photo-1529156069898-49953e39b3ac') no-repeat
center center/cover;
  display: flex;
  align-items: center;
  justify-content: center;
}

.overlay {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(0, 0, 0, 0.6);
```

```
  display: flex;
  align-items: center;
  justify-content: center;
}

.homepage-container {
  text-align: center;
  max-width: 600px;
  padding: 40px;
  border-radius: 10px;
  color: #ffffff;
  animation: fadeIn 2s ease-in-out;
}

.homepage-container h1 {
  font-size: 2.5em;
  margin-bottom: 20px;
  animation: slideIn 1.5s ease-out;
}

.homepage-container p {
  font-size: 1.2em;
  margin-bottom: 30px;
  line-height: 1.5;
}

button {
  padding: 15px 30px;
  font-size: 1.2em;
  color: #4a90e2;
  background-color: #ffffff;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.3s;
  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.3);
}

button:hover {
  background-color: #f0f0f0;
  transform: scale(1.05);
}

@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}

@keyframes slideIn {
  from { transform: translateY(-50px); opacity: 0; }
  to { transform: translateY(0); opacity: 1; }
}
```

Index.js

```
const { app, BrowserWindow } = require('electron');
const path = require('node:path');
const fs = require('fs');

var mainWindow
var childWindow

// Handle creating/removing shortcuts on Windows when installing/uninstalling.
if (require('electron-squirrel-startup')) {
  app.quit();
}

const createWindow = () => {
  // Create the browser window.
  const mainWindow = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      nodeIntegration: true,
      contextIsolation: false,
      preload: path.join(__dirname, 'preload.js'),
    },
  });

  // and load the index.html of the app.
  mainWindow.loadFile(path.join(__dirname, 'index.html'));

  // Open the DevTools.
  //mainWindow.webContents.openDevTools();
};

// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
app.whenReady().then(() => {
  createWindow();

  // On OS X it's common to re-create a window in the app when the
  // dock icon is clicked and there are no other windows open.
  app.on('activate', () => {
    if (BrowserWindow.getAllWindows().length === 0) {
      createWindow();
    }
  });
});

// Quit when all windows are closed, except on macOS. There, it's common
// for applications and their menu bar to stay active until the user quits
// explicitly with Cmd + Q.
app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});
```

Search.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Weather & Activities</title>
    <link rel="stylesheet" href="search.css">
</head>
<body onload="buttonClicked()">
    <div class="overlay"></div>
    <div class="header">
        <h1>Weather Forecast & Activity Suggestions</h1>
        <p>Find the best activities based on your location's weather.</p>
    </div>

    <div class="main-content">
        <div class="search-container">
            <input type="text" id="city_input" placeholder="Enter City" value="Kuala
Lumpur">
            <button onclick="buttonClicked()">Search</button>
            <button onclick="window.location.href='crud.html'">Activity
Checklist</button>
        </div>

        <div class="weather-container">
            <div class="weather-info left">
                <h2>Temperature in <span id="title2"></span></h2>
                <div class="weather-details">
                    <p id="main5"></p>
                    <p id="main6"></p>
                    <p id="main8"></p>
                    <p id="main9"></p>
                </div>
            </div>

            <div class="weather-info center">
                <h2><span id="title1"></span></h2>
                <div class="weather-details">
                    <div id="main7" class="icon"></div>
                    <p id="main1"></p>
                    <p id="main2"></p>
                    <p id="main3"></p>
                    <p id="main4"></p>
                </div>
            </div>

            <div class="weather-info right">
                <h2>Weather Condition in <span id="title3"></span></h2>
                <div class="weather-details">
```

```
                    <p id="main10"></p>
                    <p id="main11"></p>
                    <p id="main12"></p>
                </div>
            </div>
        </div>

        <div class="activities">
            <h2>Recommended Activities</h2>
            <div class="activity-list" id="activity-list"></div>
        </div>
    </div>

    <footer>
        <p>&copy; 2024 Weather Forecast. All rights reserved.</p>
    </footer>

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

Search.css

```
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
    font-family: 'Arial', sans-serif;
}

body {
    color: #333;
    position: relative;
}

body::before {
    content: '';
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background: url('https://images.unsplash.com/photo-1529156069898-49953e39b3ac') no-
repeat center center/cover;
    background-size: cover;
    background-position: center;
    background-attachment: fixed;
    z-index: -2;
    opacity: 0.9;
}

.overlay {
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background: rgba(0, 0, 0, 0.6);
    z-index: -1;
}

.header {
    position: fixed;
    top: 0;
    width: 100%;
    background: rgba(255, 255, 255, 0.9);
    text-align: center;
    padding: 20px 0;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.5);
    z-index: 100;
    border-bottom: 2px solid #0056b3;
}

.header h1 {
    font-size: 3rem;
    color: #003d7a;
    margin-bottom: 5px;
    font-weight: bold;
    text-shadow: 1px 1px 2px rgba(255, 255, 255, 0.8);
```

```
}

.header p {
    font-size: 1.3rem;
    color: #555;
}

.main-content {
    width: 90%;
    max-width: 1200px;
    padding: 40px 20px;
    margin: 150px auto;
    min-height: calc(100vh - 150px);
    display: flex;
    flex-direction: column;
    align-items: center;
    background-color: rgba(255, 255, 255, 0.95);
    border-radius: 20px;
    box-shadow: 0px 20px 40px rgba(0, 0, 0, 0.4);
    text-align: center;
    z-index: 1;
}

.search-container {
    margin-bottom: 30px;
    display: flex;
    justify-content: center;
    gap: 10px;
}

.search-container input {
    padding: 15px;
    font-size: 1.1rem;
    border: 2px solid #0056b3;
    border-radius: 30px;
    transition: border-color 0.3s, box-shadow 0.3s;
}

.search-container input:focus {
    border-color: #004494;
    box-shadow: 0 0 5px rgba(0, 70, 150, 0.5);
    outline: none;
}

.search-container button {
    padding: 15px 25px;
    font-size: 1.1rem;
    color: #fff;
    background-color: #007bff;
    border: none;
    border-radius: 30px;
    cursor: pointer;
    transition: background-color 0.3s, transform 0.3s;
    margin: 0 5px;
}

.search-container button:hover {
```

```
    background-color: #0056b3;
    transform: scale(1.05);
}

.weather-container {
    display: flex;
    justify-content: space-between;
    width: 100%;
}

.weather-info {
    width: 30%;
    padding: 30px;
    margin-bottom: 20px;
    background: rgba(255, 255, 255, 0.9);
    border-radius: 20px;
    box-shadow: 0px 15px 30px rgba(0, 0, 0, 0.3);
    text-align: center;
    transition: transform 0.3s;
}

.weather-info:hover {
    transform: translateY(-5px);
}

.weather-info h2 {
    font-size: 2.2rem;
    color: #0056b3;
    margin-bottom: 15px;
}

.weather-details {
    font-size: 1.2rem;
    line-height: 1.6;
    color: #444;
    padding: 10px;
}

.activities {
    width: 90%;
    margin-top: 30px;
    text-align: center;
}

.activities h2 {
    font-size: 2.5rem;
    margin-bottom: 20px;
    color: #007bff;
    text-shadow: 1px 1px 2px rgba(255, 255, 255, 0.8);
    font-weight: bold;
}

.activity-list {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 25px;
```

```
}

.activity {
    width: 220px;
    text-align: center;
    background-color: #ffffff;
    border-radius: 15px;
    padding: 15px;
    box-shadow: 0px 8px 15px rgba(0, 0, 0, 0.2);
    transition: transform 0.3s, box-shadow 0.3s;
    overflow: hidden;
}

.activity:hover {
    transform: scale(1.05);
    box-shadow: 0px 12px 20px rgba(0, 0, 0, 0.3);
}

.activity img {
    width: 100%;
    height: 150px;
    object-fit: cover;
    border-radius: 15px;
    transition: transform 0.3s;
}

.activity:hover img {
    transform: scale(1.1);
}

.activity h3 {
    margin: 10px 0;
    font-size: 1.8rem;
    color: #0056b3;
    font-weight: bold;
}

.activity p {
    font-size: 1rem;
    color: #555;
    line-height: 1.4;
    margin-top: 5px;
}

footer {
    text-align: center;
    padding: 20px 0;
    background-color: #f1f1f1;
    position: relative;
    bottom: 0;
    width: 100%;
}

footer p {
    margin: 0;
}
```

Main.js

```
const activities = {
    sunny: [
        {
            name: "Scuba Diving",
            image: "https://wallpapers.com/images/hd/scuba-diving-with-silver-fishes-
uktw6k4uimc2qncm.jpg",
            description: "Explore the underwater world and discover colorful marine
life."
        },
        {
            name: "Hiking",
            image: "https://img.freepik.com/premium-photo/outdoor-adventure-with-people-
hiking-scenic-landscape-adventurous-style-wide-shot-vibrant-colors_248779-3018.jpg",
            description: "Enjoy breathtaking views and fresh air while hiking through
scenic trails."
        },
        {
            name: "Beach Volleyball",
            image:
"https://www.shutterstock.com/shutterstock/videos/1055634659/thumb/6.jpg?ip=x480",
            description: "Play a fun game of beach volleyball with friends on a sunny
day."
        },
        {
            name: "Picnic in the Park",
            image:
"https://st5.depositphotos.com/1829243/70606/i/450/depositphotos_706063998-stock-photo-
big-family-sitting-picnic-blanket.jpg",
            description: "Relax and enjoy a delightful picnic surrounded by nature."
        },
    ],
    rainy: [
        {
            name: "Indoor Rock Climbing",
            image: "https://cdn.outsideonline.com/wp-content/uploads/2018/09/24/ee-gym-
cover_s.jpg",
            description: "Challenge yourself with rock climbing at an indoor climbing
gym."
        },
        {
            name: "Visiting a Museum",
            image: "https://images.pexels.com/videos/7986423/apartment-art-chair-
collection-7986423.jpeg",
            description: "Explore fascinating exhibits and learn about history and
culture."
        },
        {
            name: "Cooking Class",
            image:
"https://www.shutterstock.com/shutterstock/videos/1090743727/thumb/1.jpg?ip=x480",
            description: "Join a cooking class and learn to prepare delicious dishes."
        },
        {
            name: "Movie Marathon",
            image: "https://png.pngtree.com/thumb_back/fh260/background/20230704/pngtree-
bunch-of-teenagers-enjoying-a-3d-movie-marathon-at-home-image_3761072.jpg",
```

```
            description: "Cozy up at home and binge-watch your favorite movies."
        },
    ],
    snowy: [
        {
            name: "Skiing",
            image: "https://images.alphacoders.com/462/thumb-1920-462944.jpg",
            description: "Hit the slopes for an exhilarating day of skiing."
        },
        {
            name: "Building a Snowman",
            image: "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcRgC5JxlsE_2IvtlefCieLT1e8iDZcZKVlRzw&s",
            description: "Gather your friends and create the perfect snowman."
        },
        {
            name: "Ice Skating",
            image: "https://todaysparent.mblycdn.com/tp/resized/2017/01/1600x900/what-to-
know-when-teaching-your-toddler-to-ice-skate-1280x960.jpg",
            description: "Glide across the ice and enjoy a fun day of ice skating."
        },
        {
            name: "Snowshoeing",
            image: "https://traventuria.com/wp-content/uploads/2016/10/snowshoeing-
bulgaria-1.jpg",
            description: "Explore snowy landscapes while snowshoeing through nature
trails."
        },
    ],
    cloudy: [
        {
            name: "Photography",
            image: "https://img.freepik.com/free-photo/portrait-senior-man-with-camera-
device-world-photography-day-celebration_23-2151657243.jpg",
            description: "Capture beautiful moments and scenery with your camera."
        },
        {
            name: "Reading a Book",
            image:
"https://img.pikbest.com/ai/illus_our/20230413/d05bedaec4f67d4313b0c6e872de3858.jpg",
            description: "Get lost in a good book while enjoying a cozy atmosphere."
        },
        {
            name: "Board Games Night",
            image: "https://images.stockcake.com/public/8/9/7/897606b2-0e74-45b4-9740-
31363b739566_large/family-game-night-stockcake.jpg",
            description: "Gather friends for an entertaining board games night."
        },
        {
            name: "Art and Craft Session",
            image: "https://images.squarespace-
cdn.com/content/v1/5a0acf45aeb625c125bc9c6b/8c22f9b1-38b1-44f6-95e9-
095d8db234d5/Arts+%26+Craft+workshop",
            description: "Unleash your creativity with an art and craft session at home."
        },
    ],
};
```

```
function buttonClicked() {
    const city = document.getElementById("city_input").value;

    fetch(`https://api.weatherapi.com/v1/forecast.json?key=32804b24a847407391c53709241010
&q=${city}`)
        .then((response) => {
            if (!response.ok) {
                throw new Error('Network response was not ok ' + response.statusText);
            }
            return response.json();
        })
        .then((data) => {
            document.getElementById("title1").innerHTML = `${data.location.name}`;
            document.getElementById("title2").innerHTML = `${data.location.name}`;
            document.getElementById("title3").innerHTML = `${data.location.name}`;
            document.getElementById("main1").innerHTML = `Location Name:
${data.location.name}`;
            document.getElementById("main2").innerHTML = `Region:
${data.location.region}`;
            document.getElementById("main3").innerHTML = `Country:
${data.location.country}`;
            document.getElementById("main4").innerHTML = `Local Time:
${data.location.localtime}`;
            document.getElementById("main5").innerHTML = `Current Temperature:
${data.current.temp_c} °C`;
            document.getElementById("main6").innerHTML = `Forecast Temperature:
${data.forecast.forecastday[0].day.avgtemp_c} °C`;
            document.getElementById("main7").innerHTML = `<img
src="https:${data.current.condition.icon}" alt="">`;
            document.getElementById("main8").innerHTML = `Wind Speed:
${data.current.wind_mph} mph`;
            document.getElementById("main9").innerHTML = `Humidity:
${data.current.humidity}%`;
            document.getElementById("main10").innerHTML = `Weather Condition:
${data.forecast.forecastday[0].day.condition.text}`;
            document.getElementById("main11").innerHTML = `Chances of Rain:
${data.forecast.forecastday[0].day.daily_chance_of_rain}%`;
            document.getElementById("main12").innerHTML = `Chances of Snow:
${data.forecast.forecastday[0].day.daily_chance_of_snow}%`;

            const weatherCondition =
data.forecast.forecastday[0].day.condition.text.toLowerCase();
            let conditionKey = 'sunny';

            if (weatherCondition.includes('rain')) {
                conditionKey = 'rainy';
            } else if (weatherCondition.includes('snow')) {
                conditionKey = 'snowy';
            } else if (weatherCondition.includes('cloud')) {
                conditionKey = 'cloudy';
            }

            displayActivities(conditionKey);
        })
        .catch((error) => {
            console.error("Error fetching weather data:", error);
```

```
            alert("Failed to retrieve weather data. Please check your input and try
again.");
        });
}

function displayActivities(weatherCondition) {
    const activityList = document.getElementById("activity-list");
    activityList.innerHTML = "";

    const recommendedActivities = activities[weatherCondition] || [];

    recommendedActivities.forEach(activity => {
        const activityDiv = document.createElement("div");
        activityDiv.className = "activity";

        const img = document.createElement("img");
        img.src = activity.image;
        img.alt = activity.name;

        const p = document.createElement("p");
        p.textContent = activity.name;

        const description = document.createElement("p2");
        description.textContent = activity.description;
        description.className = "activity-description";

        activityDiv.appendChild(img);
        activityDiv.appendChild(p);
        activityDiv.appendChild(description);
        activityList.appendChild(activityDiv);
    });
}
```

Crud.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Outdoor Activity Checklist</title>
    <link rel="stylesheet" href="crud.css">
</head>
<body>
    <div class="overlay"></div>
    <div class="header">
        <h1>Weather Forecast & Activity Suggestions</h1>
        <p>Plan your activities with the right equipment based on weather conditions.</p>
    </div>
    <div class="wrapper">
        <div class="mainWrapper">
            <h1 class="main-title">Outdoor Activity Checklist</h1>
            <form>
                <div class="form-group">
                    <label for="fileName">Activity Name</label>
                    <input id="fileName" type="text" class="form-control"
placeholder="Enter activity name">
                </div>
                <div class="form-group">
                    <label for="fileContents">Required Equipment</label>
                    <textarea id="fileContents" class="form-control" rows="15"
placeholder="Enter required equipment"></textarea>
                </div>
                <div class="button-group">
                    <button type="button" id="btnCreate" class="btn btn-
default">Create</button>
                    <button type="button" id="btnRead" class="btn btn-
default">Read</button>
                    <button type="button" id="btnUpdate" class="btn btn-
default">Update</button>
                    <button type="button" id="btnDelete" class="btn btn-
default">Delete</button>
                </div>
            </form>
        </div>

        <div class="existing-files-wrapper">
            <h2>Existing Activities</h2>
            <br>
            <button class="btn btn-home"
onclick="window.location.href='search.html'">Home Page</button>
            <table id="fileTable" class="file-table">
                <thead>
                    <tr>
```

```
                    <th>Activities Name</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <!-- activities -->
            </tbody>
        </table>
    </div>
</div>
<footer>
    <p>&copy; 2024 Outdoor Activity Checklist. All rights reserved.</p>
</footer>
<script src="crud.js"></script>
</body>
</html>
```

Crud.css

```
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
    font-family: 'Arial', sans-serif;
}

body {
    position: relative;
    overflow-y: auto;
    display: flex;
    flex-direction: column;
    min-height: 100vh;
}

body::before {
    content: '';
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background: url('https://images.unsplash.com/photo-1529156069898-49953e39b3ac') no-repeat
center center/cover;
    z-index: -2;
    opacity: 0.9;
}

.overlay {
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background: rgba(0, 0, 0, 0.6);
    z-index: -1;
}

.wrapper {
    display: flex;
    justify-content: space-between;
    width: 80%;
    margin: 10% auto;
}

.mainWrapper,
.existing-files-wrapper {
    background: rgba(255, 255, 255, 0.95);
    border-radius: 10px;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
    padding: 20px;
    width: 48%;
    max-height: 80vh;
    overflow-y: auto;
}

.header {
    position: fixed;
    top: 0;
```

```
    width: 100%;
    background: rgba(255, 255, 255, 0.9);
    text-align: center;
    padding: 20px 0;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.5);
    z-index: 100;
    border-bottom: 2px solid #0056b3;
}

.header h1 {
    font-size: 3rem;
    color: #003d7a;
    margin-bottom: 5px;
    font-weight: bold;
    text-shadow: 1px 1px 2px rgba(255, 255, 255, 0.8);
}

.header p {
    font-size: 1.3rem;
    color: #555;
}

h1.main-title {
    font-size: 2rem;
    color: #333;
    margin-bottom: 20px;
    text-align: center;
}

.form-group {
    margin-bottom: 15px;
}

label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
    color: #555;
}

.form-control {
    width: 100%;
    padding: 10px;
    border: 2px solid #007bff;
    border-radius: 5px;
    font-size: 1rem;
    transition: border-color 0.3s;
}

.form-control:focus {
    border-color: #0056b3;
    outline: none;
    box-shadow: 0 0 5px rgba(0, 86, 179, 0.5);
}

.button-group {
    display: flex;
    justify-content: space-between;
    margin-top: 20px;
}
```

```
.btn {
    background-color: #007bff;
    color: white;
    border: none;
    padding: 10px 15px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
    flex: 1;
    margin: 0 5px;
}

.btn:hover {
    background-color: #0056b3;
}

.btn:active {
    transform: scale(0.98);
}

.file-table {
    width: 100%;
    margin: 20px 0;
    border-collapse: collapse;
}

.file-table th, .file-table td {
    padding: 10px;
    text-align: left;
    border: 1px solid #ddd;
}

.file-table th {
    background-color: #007bff;
    color: white;
}

.file-table tr:nth-child(even) {
    background-color: #f2f2f2;
}

.file-table tr:hover {
    background-color: #e0e0e0;
}

footer {
    text-align: center;
    padding: 20px 0;
    background-color: #f1f1f1;
    width: 100%;
    margin-top: auto;
}

footer p {
    margin: 0;
}
```

Crud.js

```
const { app, BrowserWindow } = require('electron');
const fs = require('fs');
const path = require('path');

// Get references to buttons and input fields
var btnCreate = document.getElementById('btnCreate');
var btnRead = document.getElementById('btnRead');
var btnUpdate = document.getElementById('btnUpdate');
var btnDelete = document.getElementById('btnDelete');
var fileName = document.getElementById('fileName');
var fileContents = document.getElementById('fileContents');
var fileTable = document.getElementById('fileTable').getElementsByTagName('tbody')[0];

// Define the path for storing files
let pathName = path.join(__dirname, 'Files');

// Create the "Files" folder if it doesn't exist
if (!fs.existsSync(pathName)) {
    fs.mkdirSync(pathName, { recursive: true });
}

// Function to load existing files into the table
function loadFiles() {
    fs.readdir(pathName, (err, files) => {
        if (err) {
            console.error(`Error reading directory: ${err}`);
            return;
        }
        fileTable.innerHTML = ''; // Clear the table before populating
        files.forEach(file => {
            if (file.endsWith('.txt')) {
                var row = fileTable.insertRow();
                var cellName = row.insertCell(0);
                var cellAction = row.insertCell(1);
                cellName.textContent = file; // Display file name
                cellAction.innerHTML = `<button class="btn btn-default"
onclick="readFile('${file}')">Read</button>
                                        <button class="btn btn-default"
onclick="deleteFile('${file}')">Delete</button>`;
            }
        });
    });
}

// Event listener for creating a text file
btnCreate.addEventListener('click', function() {
    const name = fileName.value.trim();
    if (!name) {
        alert('Please enter a file name.');
        return;
    }

    let file = path.join(pathName, name + '.txt'); // Ensure .txt extension
    let contents = fileContents.value;

    fs.writeFile(file, contents, function(err) {
        if (err) {
            console.error(`Error writing file: ${err}`);
            return alert('Error creating file: ' + err.message);
```

```
        }
        alert(name + " text file was created");
        fileContents.value = ''; // Clear contents after creation
        fileName.value = ''; // Clear filename field
        loadFiles(); // Refresh the file list
    });
});

// Function to read a file and populate input fields
function readFile(name) {
    let file = path.join(pathName, name);
    fs.readFile(file, 'utf8', function(err, data) {
        if (err) {
            console.error(`Error reading file: ${err}`);
            return alert('Error reading file: ' + err.message);
        }
        fileContents.value = data; // Populate textarea with file contents
        fileName.value = name.replace('.txt', ''); // Set filename without .txt
    });
}

// Event listener for updating a text file
btnUpdate.addEventListener('click', function() {
    const name = fileName.value.trim() + '.txt'; // Get the filename
    if (!name) {
        alert('Please enter a file name to update.');
        return;
    }

    let file = path.join(pathName, name);
    let contents = fileContents.value;

    fs.writeFile(file, contents, function(err) {
        if (err) {
            console.error(`Error updating file: ${err}`);
            return alert('Error updating file: ' + err.message);
        }
        alert(name + " text file was updated");
        fileContents.value = ''; // Clear contents after update
        fileName.value = ''; // Clear filename field
        loadFiles(); // Refresh the file list
    });
});

// Function to delete a file
function deleteFile(name) {
    let file = path.join(pathName, name);
    fs.unlink(file, function(err) {
        if (err) {
            console.error(`Error deleting file: ${err}`);
            return alert('Error deleting file: ' + err.message);
        }
        alert(name + " text file was deleted");
        loadFiles(); // Refresh the file list after deletion
    });
}

// Load existing files when the application starts
loadFiles();
```

Submit files in GitHub and include the link in the report.

https://github.com/SyafiqLuckyS/BCS2307-093_OOP_FINAL.git