

**KOLEJ PROFESIONAL MARA BERANANG**

**DIPLOMA IN COMPUTER SCIENCE**

|  |  |  |
| --- | --- | --- |
| **COURSE NAME** | : | **OBJECT ORIENTED PROGRAMMING** |
| **COURSE CODE** | : | **CSC2744** |
| **ACADEMIC SESSION** | : | **SESSION 1 2023/2024** |
| **TYPE OF ASSESSMENT** | : | **FINAL ASSIGNMENT** |
| **DURATION** | : | **20/6/2023-10/07/2023** |

**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 given due date will not be accepted.
2. Report should be written using:

Font type: Arial

Size: 12 pts

Line Spacing: 1.5

1. Coding format:

Font type: Consolas

Size: 10 pts

Line Spacing: Single

|  |  |
| --- | --- |
| **Personal Details** | |
| **Name** | NUR ALIA NATASYA BT MOHD YUSRI |
| **I/D Number** | BCS2207-113 |
| **Class** | 4B |
| **Lecturer** | PN. NINI ANIZA BT ZAKARIA |

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

**Question**

Electron is a powerful framework that enables developers to create cross-platform applications using web technologies such as HTML, CSS, and JavaScript. You need to choose one of the applications below to develop a desktop application using Electron that integrates the given API. The application needs to be developed with specific requirements or functionalities.

|  |  |  |
| --- | --- | --- |
| **Name of Application** | **Description** | **Requirements** |
| Dictionary and Thesaurus | An app that lists words in groups of synonyms and related concept. | * Word search. * Information Output: give the meaning of the searched word for different part of speech (noun/adjective), antonyms and the example of word usage, sounds and related URL for the searched word. * CRUD words of the day   <https://api.dictionaryapi.dev/api/v2/entries/en/digital> |
| Meal planner | An app that displays suggestion of recipe based on food item entered by user | * Suggest recipe based on food item. * Information Output: One recipe suggestion that comprises of ingredients, instruction on how to cook and URL of the related site for the food and the link on how to prepare the food. * CRUD meal planner   <https://www.themealdb.com/api/json/v1/1/search.php?s=shawarma> |
| Makeup Box | An app that finds makeup products based on brand and category entered. | * Display the product info according to search criteria. * Information Output: product description based on brand and name, product image, product website and related link for the searched product. * CRUD makeup top 5 list   <http://makeup-api.herokuapp.com/api/v1/products.json?brand=maybelline> |

**Tasks:**

1. Create desktop app using electron and apply third party data fetched from API and the requirements given. Your application 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 as given in the requirements.
3. Apply HTML and CSS for user interface and provide evidence for application.
4. GUI Elements:
   1. Apply GUI elements that assist users in using application.
   2. The application’s ‘look and feel’ is attractive and informative.
5. Produce a report on your application functionalities and features. Include the following:
   1. Overview of your application with a brief description.
   2. Screenshots of the application with explanations on how to use it.
   3. 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 app using electron and apply third party data fetched from API and the requirements given. You may add extra functionality or features of your choice to the application. | 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 data from the API and does not fulfill the requirements. | Able to display sufficient data from the API that meet with the requirements together with the description. | Able to display sufficient data from the API that meet with the requirements together with the description. | Able to display extra data from the API beyond the application requirements together with no description. | Able to display extra data from the API beyond the application requirements together 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. | The data from the API is meaningful and reflect the purpose of the application developed. | The data from the API is meaningful to come up with extra idea for the application developed. |  |
| The developed application does not fulfill the requirements stated for the chosen application. | The developed application fulfills all the requirements stated for the chosen application with no extra functionalities. | The developed application fulfills all the requirements stated for the chosen application.  Add extra functionality or features to the application. | The developed application fulfills all the requirements stated for the chosen application.  Add extra functionality or features to the application that enhances the user experience or adds value to the application. | The developed application fulfills all the requirements stated for the chosen application that utilizes the data from the API  Add extra functionality or features to the application that enhances the user experience or adds value to the application. |  |
| 2. Implement CRUD (create, read, update, delete) process to the application as given in the requirements. | * Able to create only 2 of the CRUD processes according to the requirements. * No feedback for the CRUD process. * The design for data input is poor | * Able to create only 3 of the CRUD processes according to the requirements. * 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 according to the requirements. * No feedback for the CRUD process. * Well-designed data input for CRUD process. | * Able to perform all the CRUD processes according to the requirements. * No feedback for the CRUD process. * Well-designed data input for CRUD process. | * Able to perform all the CRUD processes according to the requirements. * 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 is in need of polish in its visual design and is not appropriate for the target audience. * **Color**   Choice of colors and combinations are not suitable. | * The application is in need of 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 concept of the application. | * 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. |  |
| **Convey** | 5. Produce a report on your application functionalities and features that includes:   1. Overview of the application. 2. Screenshots of the application with explanations on how to use it. | The overview of the application is vague.  The user guide is incomplete and cannot be recognized as a user guide. | The overview of the application is very brief and does not describe the whole functionalities of the application.  The user guide provides limited information with no screenshots of the application. | The overview of the application is clearly described the whole application and its functionalities.  The user guide provides basic information with limited screenshots of the application. | The overview of the application is clearly described the whole application and its functionalities.  The user guide provides adequate information with complete screenshots of the application. | The overview of the application is clearly described the whole application and its functionalities.  The user guide provides extensive information with complete screenshots and labelling of the application. |  |
|  | 1. Program codes of the system | HTML, CSS and JavaScript codes attached are not complete.  The codes are hardly read.  Does not submit complete electron files in GitHub | HTML, CSS and JavaScript codes attached are complete.  The codes are hardly read.  Completely submit all the electron file in GitHub. | HTML, CSS and JavaScript codes attached are complete.  The codes are readable but not organized.  Completely submit all the electron file in GitHub. | HTML, CSS and JavaScript codes attached are complete.  The codes are readable and organized.  Completely submit all the electron file in GitHub. | HTML, CSS and JavaScript codes attached are complete and include comments for the important parts of the codes.  The codes are readable and organized.  Completely submit all the electron file in GitHub. |  |
| **Total Marks Earned** | | | | | | | **/50** |
| **Total Percentage (40%)** | | | | | | | **/40%** |

1. Produce a report on your application functionalities and features. Include the following:
   1. Overview of your application with a brief description.

**MEAL PLANNER**

A meal planner application is a digital tool designed to help users plan and organize their meals effectively. These applications are typically available on various platforms, including smartphones, tablets, and computers, making it convenient for users to access their meal plans anytime, anywhere. Here's an overview of a meal planner application :

1. **Recipe**

It will display the name of the meals, category, image, meal instructions and youtube link on how to prepare it based on the ingredients that user choose.

1. **Meal Planning:**

Users can create customized meal plans by selecting recipes from the application's database. They can plan meals for everyday.

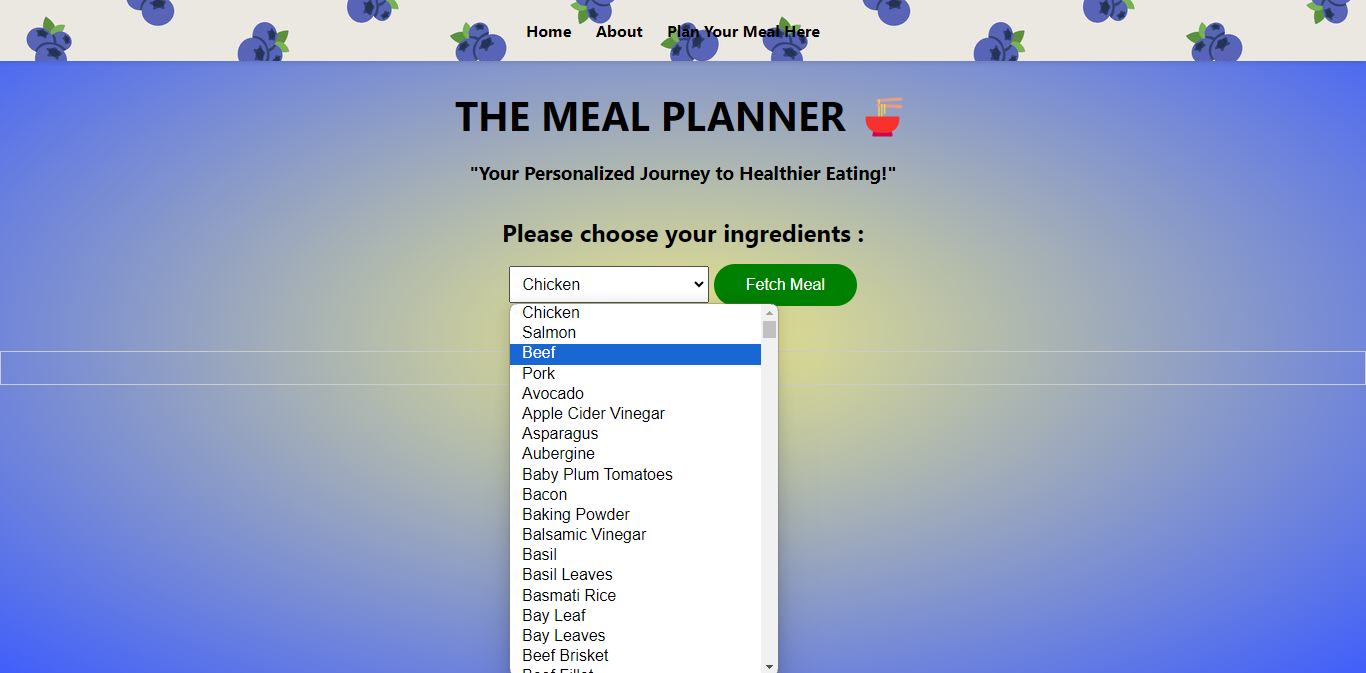
* 1. Screenshots of the application with explanations on how to use it.

**INDEX PAGE**

A screenshot of a menu

Description automatically generated

* This is the first page that user will see after run the application. (npm start)

****

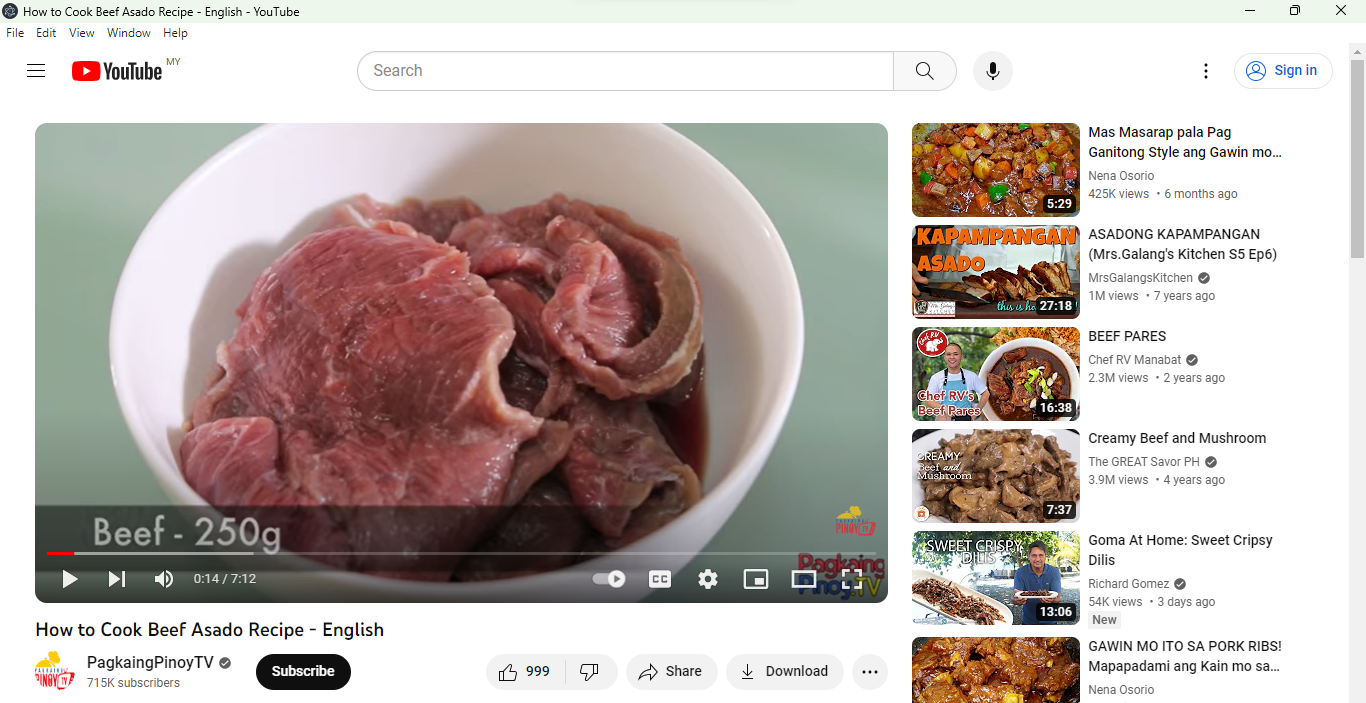
* User needs to choose the ingredients based on the list of ingredients that have been fetched by the API. For example, this user choose Beef. Then, click the fetch meal button.

A screenshot of a food

Description automatically generated



* After user click the fetch meal button, it will display the name of the meals, its category, instruction on how to prepare the meal, list of ingredients that user needs and a youtube video link on how to prepare the meal.



* If user click the link, it will directly bring the user to the youtube app which show the video on how to prepare the meals. For example, this user click the link on how to cook Beef Asado.

A screenshot of a menu

Description automatically generated

A screenshot of a menu

Description automatically generated

* At the header, there is a option to Plan Your Meal Here. By clicking that, user will go to the next page which is Meal Planner Page.





* For this page, user can create customized meal plans by selecting recipes from the application's database. User can put any name for the file, then choose the ingredients and put some notes in the contents. Click the Create button. It will give notice that the text file was created.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* To check if the file has created or not, go to the mealplanapp then click src. Check the folder name ‘Files’. It should have the created text file.

A screenshot of a computer

Description automatically generated

* For example, this is the file that user have created.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Next, user can choose to update the content of the file by clicking Update button. After clicking it, it will give notify that the file was updated.

A screenshot of a computer

Description automatically generated

* Example, user have update the content of the previous text file.

A screenshot of a computer

Description automatically generated

* If user want to delete the file that has been created, they just need to click the button Delete. And the file will deleted in the Folder “Files”.

A screenshot of a computer

Description automatically generated

* One more option, user can clear everyting that they have write in the box just by clicking Clear button in red. It will clear every words in the contents.
* User can go back to the first page by clicking the HOME link at left bottom of the page.

A screenshot of a menu

Description automatically generated

FETCHAPI

* Fecth the list of ingredients from the API

A close-up of a computer screen

Description automatically generated

* Fetch the meal type from the API

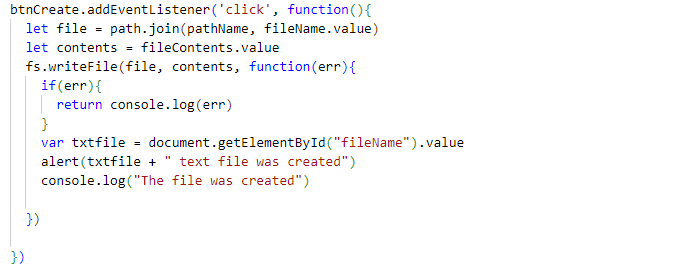
A close-up of a website

Description automatically generated

CRUD OPERATION

CREATE

* Create a new text file and saved it into folder ‘Files’.



READ

* Read contents of the created text file.

A computer screen shot of a code

Description automatically generated

UPDATE

* Update the contents of the created text file.

A computer code with text

Description automatically generated

DELETE

* Delete the created text file.

A computer code with text

Description automatically generated

* 1. 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">

<link rel="stylesheet" href="index.css" />

<title>Meal Planner</title>

</head>

<body>

<nav>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="index.html">About</a></li>

<li><a href="mealprep.html">Plan Your Meal Here</a></li>

</ul>

</nav>

<h1>THE MEAL PLANNER 🍜</h1>

<h3>"Your Personalized Journey to Healthier Eating!"</h3>

<h4>Please choose your ingredients :</h4>

<select id="ingredient-dropdown"></select>

<button id="fetch-meal-btn">Fetch Meal</button>

<div id="meal-container"></div>

<script src="fetchapi.js"></script>

</body>

</html>

**FETCHAPI.JS**

document.addEventListener('DOMContentLoaded', async () => {

const ingredientDropdown = document.getElementById('ingredient-dropdown');

// Fetch the list of ingredients from the API

//List all the Ingredients

const response = await fetch('https://www.themealdb.com/api/json/v1/1/list.php?i=list');

const data = await response.json();

// Populate the dropdown with ingredients from the API response

data.meals.forEach(ingredient => {

const option = document.createElement('option');

option.value = ingredient.strIngredient;

option.textContent = ingredient.strIngredient;

ingredientDropdown.appendChild(option);

});

const fetchMealBtn = document.getElementById('fetch-meal-btn');

fetchMealBtn.addEventListener('click', async () => {

const selectedIngredient = ingredientDropdown.value;

const meals = await fetchMealData(selectedIngredient);

displayMealData(meals);

});

});

const fetchMealData = async (mealType) => {

//Search meal by name

const response = await fetch(`https://www.themealdb.com/api/json/v1/1/search.php?s=${mealType}`);

const data = await response.json();

return data.meals;

};

const displayMealData = (meals) => {

const mealContainer = document.getElementById('meal-container');

mealContainer.innerHTML = '';

if (meals) {

meals.forEach(meal => {

const mealDiv = document.createElement('div');

mealDiv.innerHTML = `

<h2>${meal.strMeal}</h2>

<p><strong>Category:</strong> ${meal.strCategory}</p>

<center><img src="${meal.strMealThumb}" alt="${meal.strMeal}" style="width: 400px; height: auto;"></center>

<h3>Meal Instruction:</h3>

${meal.strInstructions}

<h3>Ingredients:</h3>

<ul>

${getIngredientsList(meal)}

</ul>

<h3>Youtube Link:</h3>

<a href="${meal.strYoutube}" target="\_blank">Watch Video: How to Prepare</a>

`;

mealContainer.appendChild(mealDiv);

});

} else {

mealContainer.innerHTML = 'No meals found.';

}

};

const getIngredientsList = (meal) => {

const ingredientsList = [];

for (let i = 1; i <= 20; i++) {

const ingredient = meal[`strIngredient${i}`];

const measure = meal[`strMeasure${i}`];

if (ingredient && measure) {

ingredientsList.push(`<li>${measure} ${ingredient}</li>`);

}

}

return ingredientsList.join('');

};

document.addEventListener('DOMContentLoaded', () => {

const fetchMealBtn = document.getElementById('fetch-meal-btn');

const mealTypeDropdown = document.getElementById('meal-type-dropdown');

fetchMealBtn.addEventListener('click', async () => {

const mealType = mealTypeDropdown.value;

const meals = await fetchMealData(mealType);

displayMealData(meals);

});

});

**INDEX.JS**

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

const fs = require('fs')

const path = require('path')

// Handle creating/removing shortcuts on Windows when installing/uninstalling.

if (require('electron-squirrel-startup')) {

// eslint-disable-line global-require

app.quit();

}

const createWindow = () => {

// Create the browser window.

const mainWindow = new BrowserWindow({

width: 800,

height: 600,

webPreferences: {

nodeIntegration: true,

contextIsolation: false,

}

});

// 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.on('ready', 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();

}

});

app.on('activate', () => {

// 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.

if (BrowserWindow.getAllWindows().length === 0) {

createWindow();

}

});

// In this file you can include the rest of your app's specific main process

// code. You can also put them in separate files and import them here.

**INDEX.CSS**

body {

font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Helvetica, Arial, sans-serif;

margin: 0;

padding: 0;

text-align: center;

background-image: radial-gradient(#ebe887, #3f5efb);

background-size: cover;

background-repeat: no-repeat;

background-attachment: fixed;

}

nav {

background-image: url(blueberry.jpg);

padding: 20px 0;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

ul {

list-style: none;

padding: 0;

margin: 0;

}

li {

display: inline;

margin-right: 20px;

}

a {

color: black;

text-decoration: none;

font-weight: bold;

transition: color 0.3s ease;

}

a:hover {

color: #ffc107; /\* Change the color on hover \*/

}

.container {

max-width: 800px;

margin: 0 auto;

padding: 2rem;

}

h1 {

font-size: 2.5rem;

margin-bottom: 1rem;

}

h2 {

font-size: 2rem;

margin-bottom: 1.5rem;

}

h4 {

font-size: 1.5rem;

margin-bottom: 1rem;

}

#ingredient-dropdown {

font-size: 1rem;

margin-bottom: 1rem;

padding: 0.5rem;

width: 100%;

max-width: 200px;

}

#fetch-meal-btn {

font-size: 1rem;

background-color: green;

color: white;

border: none;

border-radius: 25px;

padding: 0.75rem 2rem;

cursor: pointer;

transition: background-color 0.3s ease;

}

#fetch-meal-btn:hover {

background-color: #45a049;

}

#meal-container {

margin-top: 2rem;

border: 1px solid #ccc;

padding: 1rem;

text-align: left;

}

**MEALPREP.HTML**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8" />

<title>CRUD App Meal Prep</title>

<link rel="stylesheet" href="mealprep.css" />

</head>

<body>

<div class="mainWrapper">

<h1> 🍜 MEAL PREP 🍜</h1>

<h2>"Prepare Today, Thrive Tomorrow: Your Recipe for a Healthier, Hassle-Free Life"</h2>

<form>

<div class="form-group">

<label>File name :</label>

<input id="fileName" type="text" class="form-control">

</div>

<div class="form-group">

<label>Content :</label>

<select id="ingredient-dropdown">

<textarea id="fileContents" class="form-control" rows="5"></textarea>

</select>

</div>

</form>

<button id="btnCreate" class="btn btn-default">Create</button>

<button id="btnRead" class="btn btn-default">Read</button>

<button id="btnDelete" class="btn btn-default">Delete</button>

<button id="btnUpdate" class="btn btn-default">Update</button>

<button id="clear-btn">Clear</button>

</div>

<a href="index.html">HOME</a>

<script src="mealprep.js">

</script>

</body>

</html>

**MEALPREP.JS**

document.addEventListener('DOMContentLoaded', async () => {

const ingredientDropdown = document.getElementById('ingredient-dropdown');

// Fetch the list of ingredients from the API

//List all the Ingredients

const response = await fetch('https://www.themealdb.com/api/json/v1/1/list.php?i=list');

const data = await response.json();

// Populate the dropdown with ingredients from the API response

data.meals.forEach(ingredient => {

const option = document.createElement('option');

option.value = ingredient.strIngredient;

option.textContent = ingredient.strIngredient;

ingredientDropdown.appendChild(option);

});

const fetchMealBtn = document.getElementById('fetch-meal-btn');

fetchMealBtn.addEventListener('click', async () => {

const selectedIngredient = ingredientDropdown.value;

const meals = await fetchMealData(selectedIngredient);

displayMealData(meals);

});

});

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

const fs = require('fs')

const path = require('path')

var btnCreate = document.getElementById('btnCreate')

var btnRead = document.getElementById('btnRead')

var btnDelete = document.getElementById('btnDelete')

var btnUpdate = document.getElementById('btnUpdate')

var fileName = document.getElementById('fileName')

var fileContents = document.getElementById('fileContents')

let pathName = path.join(\_\_dirname, 'Files')

btnCreate.addEventListener('click', function(){ //creating text file when user click CREATE button

let file = path.join(pathName, fileName.value)

let contents = fileContents.value

fs.writeFile(file, contents, function(err){ //param1: textfile yg kita nak write param2: apa yg kita nak write ke text file

if(err){

return console.log(err)

}

var txtfile = document.getElementById("fileName").value

alert(txtfile + " text file was created")

console.log("The file was created")

})

})

btnRead.addEventListener('click', function(){ //read contents of the created text file

let file = path.join(pathName, fileName.value)

fs.readFile(file, function(err, data){

if(err){

return console.log(err)

}

fileContents.value = data

console.log("The file was read!")

})

})

btnDelete.addEventListener('click', function(){

let file = path.join(pathName, fileName.value)

fs.unlink(file, function(err){

if(err){

return console.log(err)

}

fileName.value = ""

fileContents.value = ""

console.log("The file was deleted!")

})

})

btnUpdate.addEventListener('click', function(){

let file = path.join(pathName, fileName.value)

let contents = fileContents.value

fs.writeFile(file, contents, function(err){

if(err){

return console.log(err)

}

var txtfile = document.getElementById("fileName").value

alert(txtfile + " text file was updated")

console.log("The file was updated")

})

})

let clearBtn = document.getElementById('clear-btn')

clearBtn.addEventListener('click', function(){

fileName.value = ""

fileContents.value = ""

})

**MEALPREP.CSS**

body {

font-family: 'Courier New', Courier, monospace;

margin: auto;

max-width: 38rem;

padding: 2rem;

background-image: linear-gradient(#3f5efb , #ebe887);

background-repeat: no-repeat;

background-size: cover;

background-attachment: fixed;

}

.mainWrapper {

text-align: center;

}

h1 {

font-size: 2.5rem;

margin-bottom: 1.5rem;

color: #fff;

}

.form-group {

margin-bottom: 1.5rem;

text-align: left;

}

label {

display: block;

font-weight: bold;

margin-bottom: 0.5rem;

color: #fff;

}

.form-control {

width: calc(100% - 2rem);

padding: 0.75rem;

font-size: 1rem;

border: 1px solid #ccc;

border-radius: 0.25rem;

color: #333;

}

.btn {

background-color: #57cd84;

color: #fff;

border-radius: 20px;

padding: 0.75rem 1.5rem;

cursor: pointer;

transition: background-color 0.3s ease, transform 0.2s ease, box-shadow 0.3s ease;

margin-right: 1rem;

border: none;

box-shadow: 0 2px 4px rgba(87, 205, 132, 0.2); /\* Add a subtle shadow \*/

}

.btn:hover {

background-color: #41a165;

transform: scale(1.05); /\* Add a slight scale effect on hover \*/

box-shadow: 0 4px 8px rgba(87, 205, 132, 0.3); /\* Increase the shadow on hover \*/

}

#clear-btn {

background-color: #f44336;

color: #fff;

}

#clear-btn:hover {

background-color: #d32f2f;

}