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
3. Coding format:
   Font type: Consolas
   Size: 10 pts
   Line Spacing: Single

Personal Details	
Name	NADIA SYAZWIN BINTI SUHAIME
I/D Number	BCS2207 – 052
Class	DCS 4C
Lecturer	PUAN NUR AKMAL HAFIZAN BINTI KAMAL IQBAL

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:
    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 functionalities and features. Include the following:
    i. Overview of your application with a brief description.
    ii. Screenshots of the application with explanations on how to use it.
    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 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	The developed application fulfills all the requirements stated for the chosen application with	The developed application fulfills all the requirements stated for the chosen application.	The developed application fulfills all the requirements stated for the chosen application.	The developed application fulfills all the requirements stated for the chosen application that	

		application.	no extra functionalities.	Add extra functionality or features to the application.	Add extra functionality or features to the application that enhances the user experience or adds value to the application.	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	• **Text** - Most text used is clear but does not describe the content well. • **Graphics** - Graphics are related to the theme/purpose of the application	• **Text** - All text used is clear but does not describe the content well. • **Graphics -** Graphics are related to the theme/purpose of the	• **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	

			of low quality, OR distract the reader.	and are of excellent quality.	application, are of excellent quality and enhance reader interest or understanding	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. i.	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**	• The application is in need of polish in its visual design, but it is still appropriate for the target audience. • **Color**	• The application mostly follows good visual design principles (e.g.: alignment, contrast,	• The application demonstrates good visual design principles (e.g.: alignment, contrast,	• The application clearly demonstrates good visual design principles (e.g.: alignment,	

		Choice of colors and combinations are not suitable.	Choice of colors and combinations do not match the concept of the application.	easily read text) and is appropriate for the target audience. • **Color** Choice of colors and combinations match the concept of the application.	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.	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: i. Overview of the application. ii. 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.	
	iii. Program codes of the 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	HTML, CSS and JavaScript codes attached are complete and include comments for the important parts of the codes.	

		Does not submit complete electron files in GitHub	Completely submit all the electron file in GitHub.	Completely submit all the electron file in GitHub.	organized. Completely submit all the electron file in GitHub.	The codes are readable and organized. Completely submit all the electron file in GitHub.	
Total Marks Earned							**/50**
Total Percentage (40%)							**/40%**

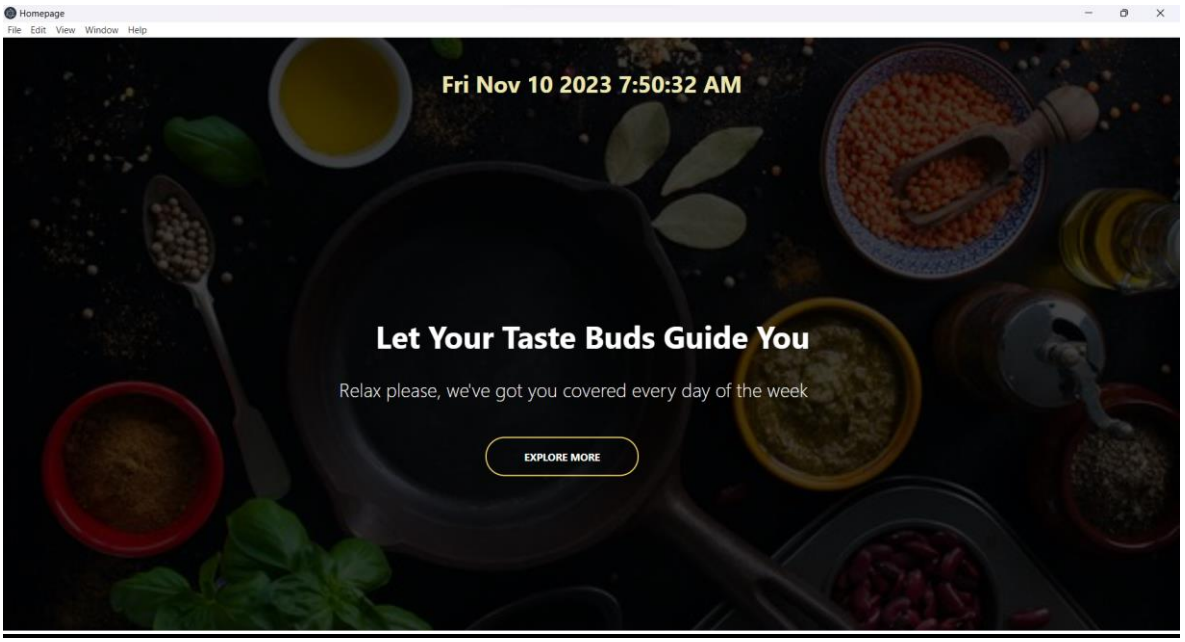
OVERVIEW OF THE APPLICATION

“Meal Planner” is a user-friendly system that simplifies the process of meal planning and recipe discovery. This versatile platform offers two main features, first, it allows users to explore an extensive range of recipes categorized by their preferences, such as “Side” or “Starter” upon selecting a category, the system presents a list of suggested recipes in boxes, each accompanied by detailed information fetched from the API, including a list of ingredients, cooking instructions, a link to a helpful YouTube video, and the source of the recipe. Second, “Meal Planner” empowers users to effortlessly plan their meals. With this system, users can create, read, and update their meal plans, providing an organized and flexible approach to managing their dietary choices. In essence, “Meal Planner” serves as a user-friendly tool for both discovering new recipes and streamlining the meal planning process, catering to the diverse needs of its users.

SCREENSHOTS OF THE APPLICATION WITH EXPLANATIONS ON HOW TO USE IT. (USER GUIDE)

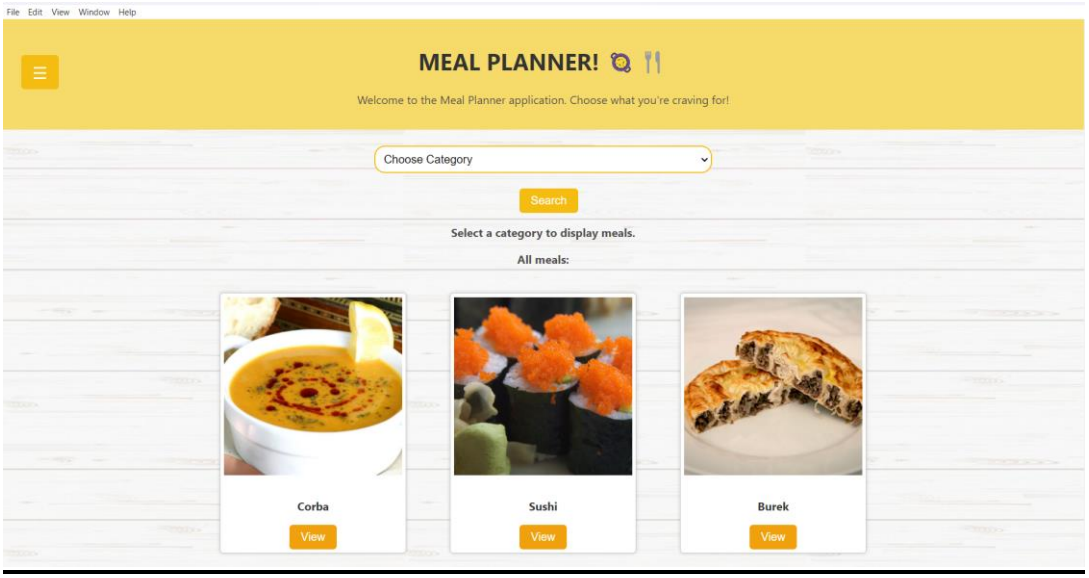
File name: home.html

This is the ‘home.html’ file, known as the ‘Home’ page in my system. This page is the first thing a user will see when they view my application. On this page, there is button “explore more” , and when user click on it will take them to the view recipe page.

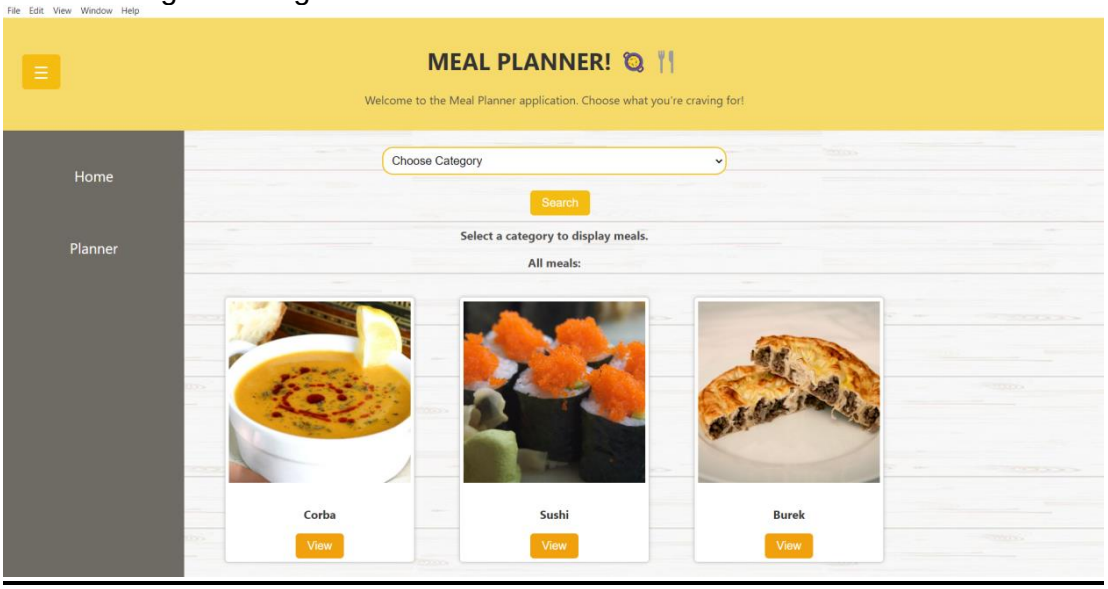


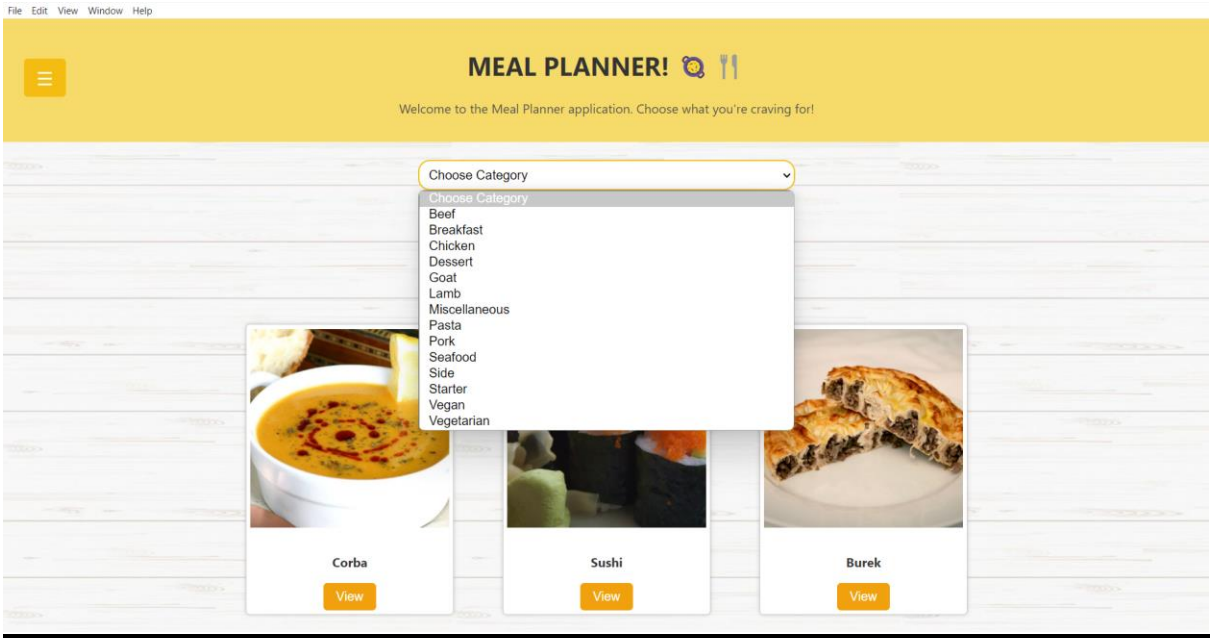
File name: index.html

This is the ‘index.html’ file, known as the ‘View Recipe’ page in my system. On this page, user will see a random meals displayed and there is navigation button menu on top-left of the website with options such as ‘Home’ and ‘Planner’. When a user clicks ‘Home’ in the navigation menu, it will display this page. Clicking ‘Planner’ will take them to the meal planner page. Users also are provided with search fields to allow them to search for any category item and view suggested recipes based on their search. When users click the search fields, a dropdown appears containing a list of category items. After selecting a category and clicking the ‘Search’ button, the system will display all the suggested recipes. Users can click on the button “view” of any suggested recipe to access details about the meal, including ingredients, instructions, the source, and a YouTube URL for the recipe.

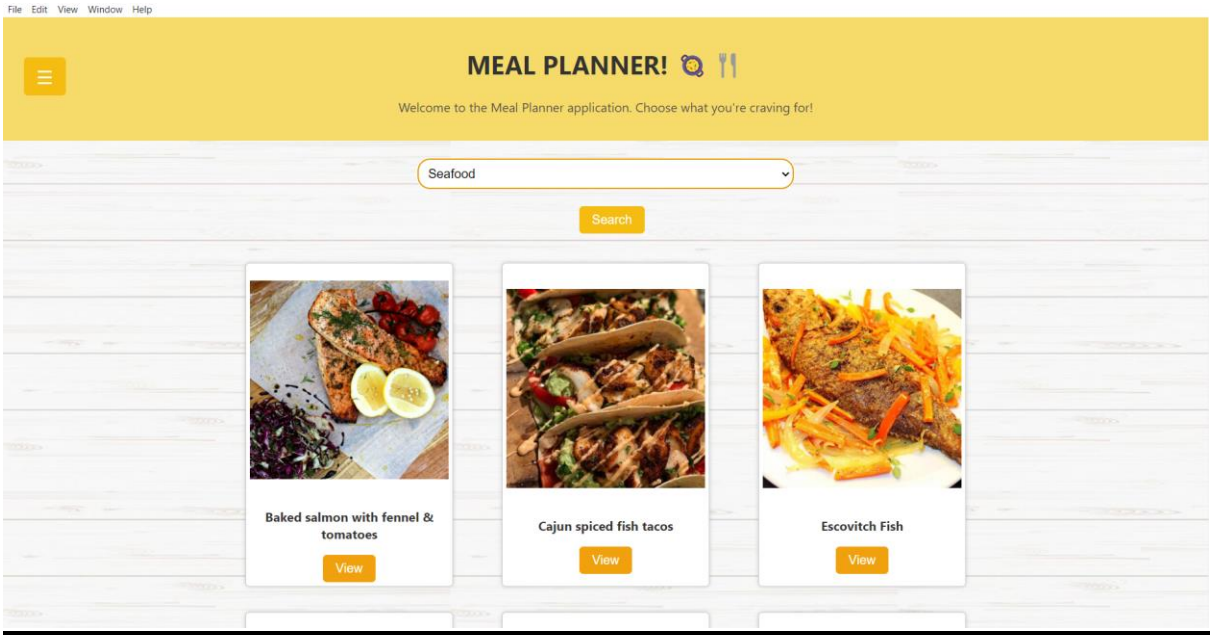


After clicking the navigation button:

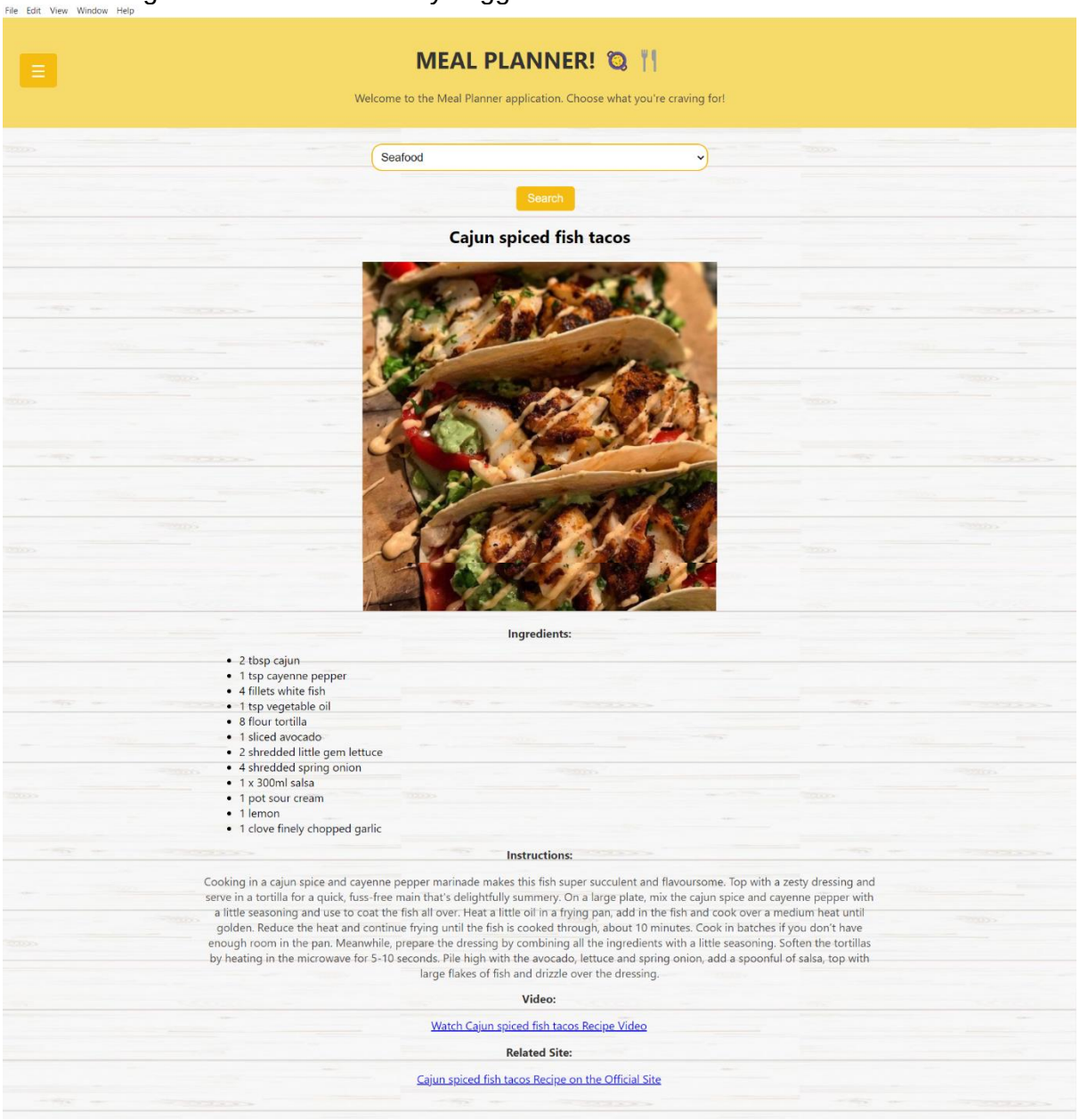




After clicking the ‘Search’ button:



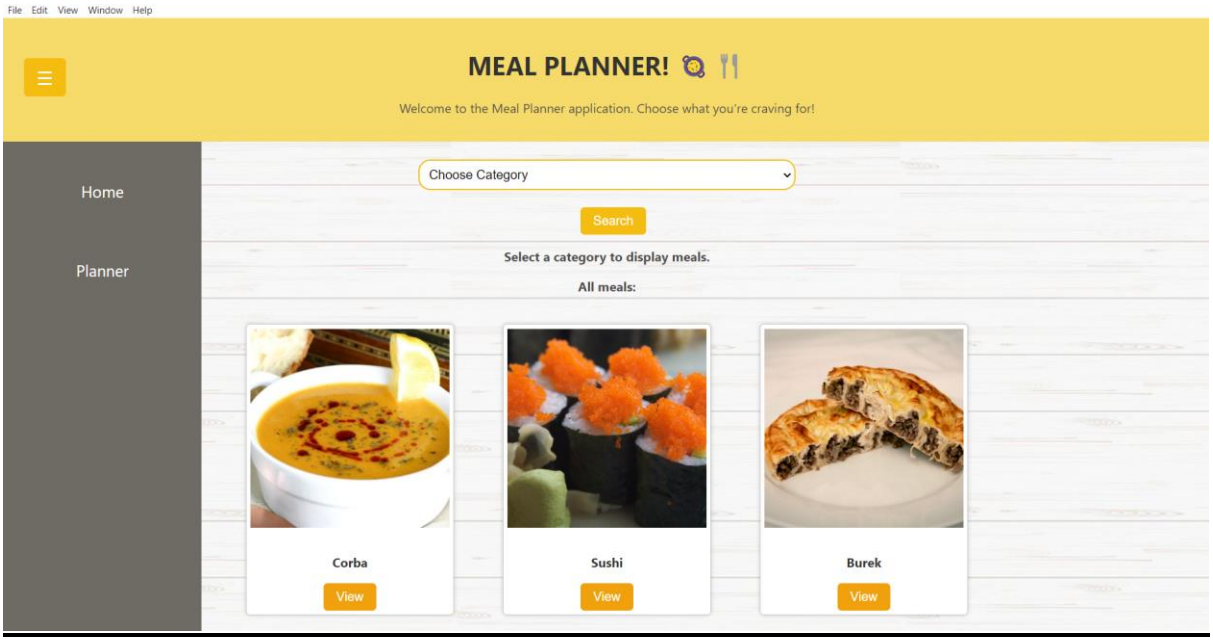
After clicking the ‘view’ button of any suggested meal:



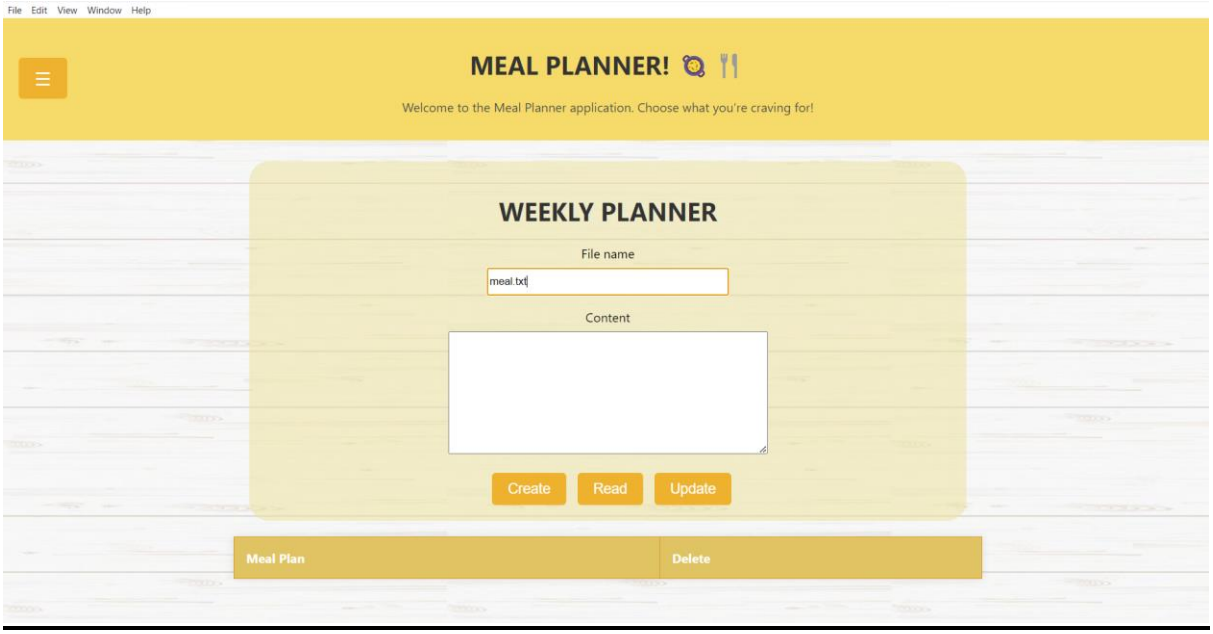
File name: CRUD.html

This is the ‘CRUD.html’ file, known as the ‘Plan Meal’ page in my system. On this page, users are provided with a form that allows them to enter their planned meals in the content fields. In the form, user need to enter the file name they want to use. If user want to create a new file, all they need to do is enter the file name and click on ‘create’ button. The system will display an alert such as, ‘meal.txt text file was created!’. After the user fills out the content form, if they click the ‘Create’ button again, the system will display an alert, ‘meal.txt text file has been added!’ to alert the user that the content has been successfully created and added to the file.

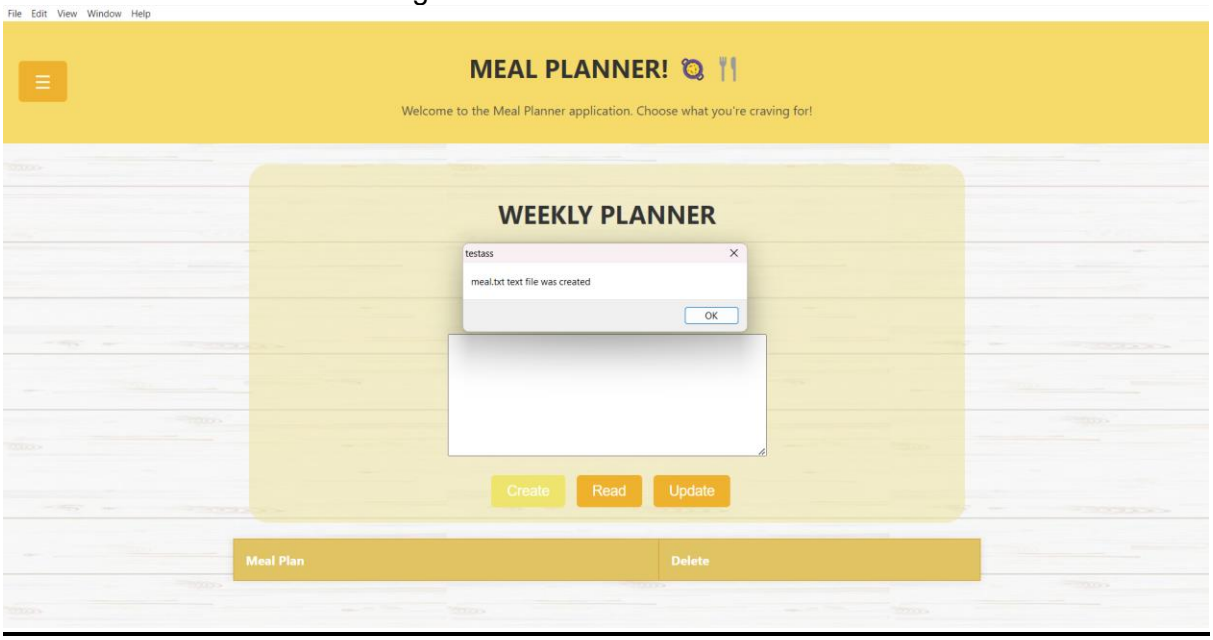
User clicking ‘Planner’ on navigation menu:



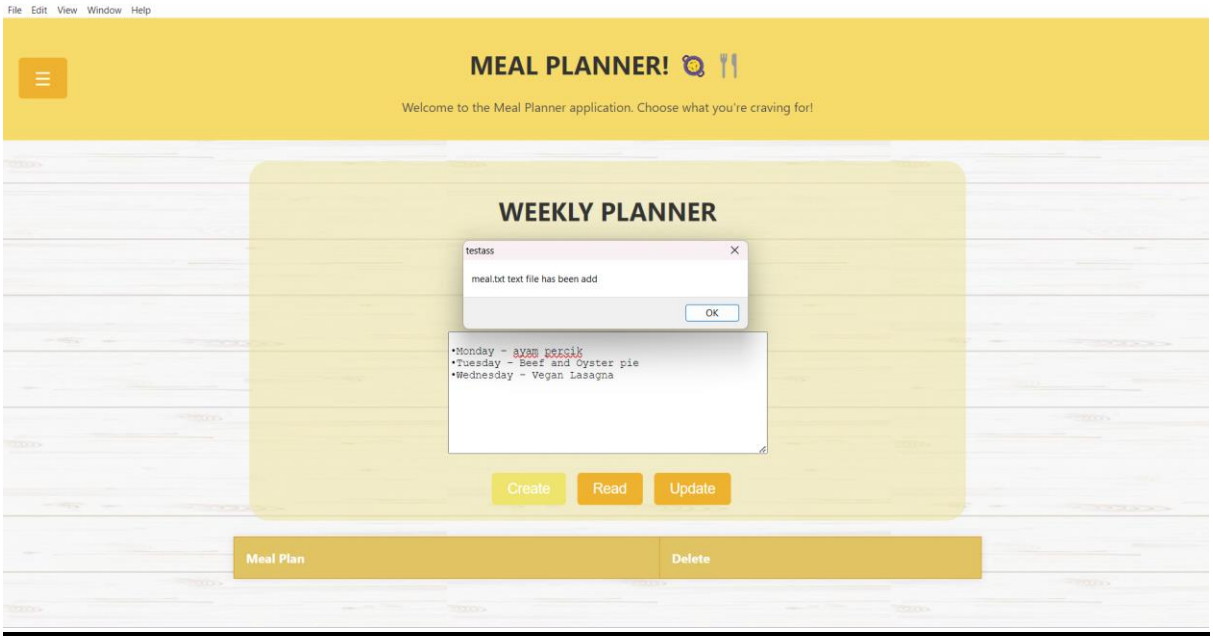
After clicking ‘Planner’ on navigation menu:



After enter file name and clicking the ‘Create’ button:

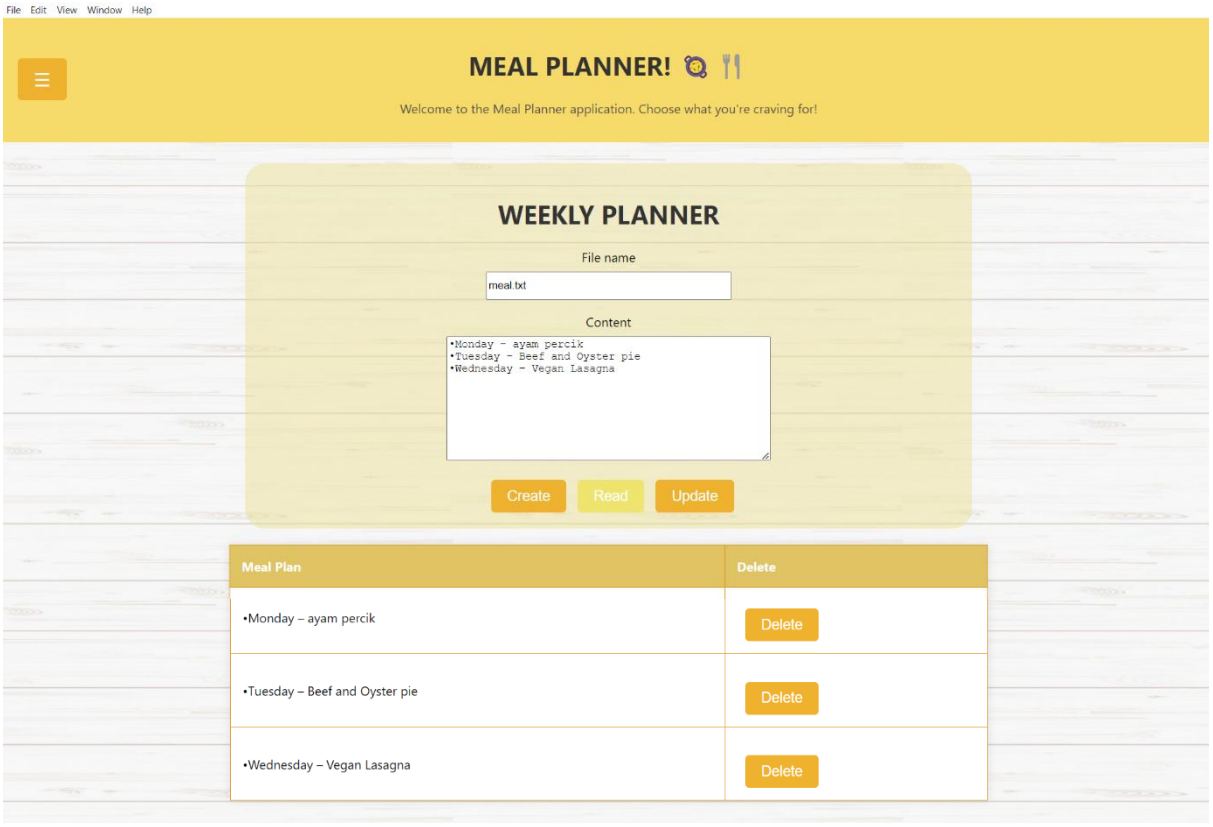


After user fills out the content fields and clicking the ‘Create’ button:

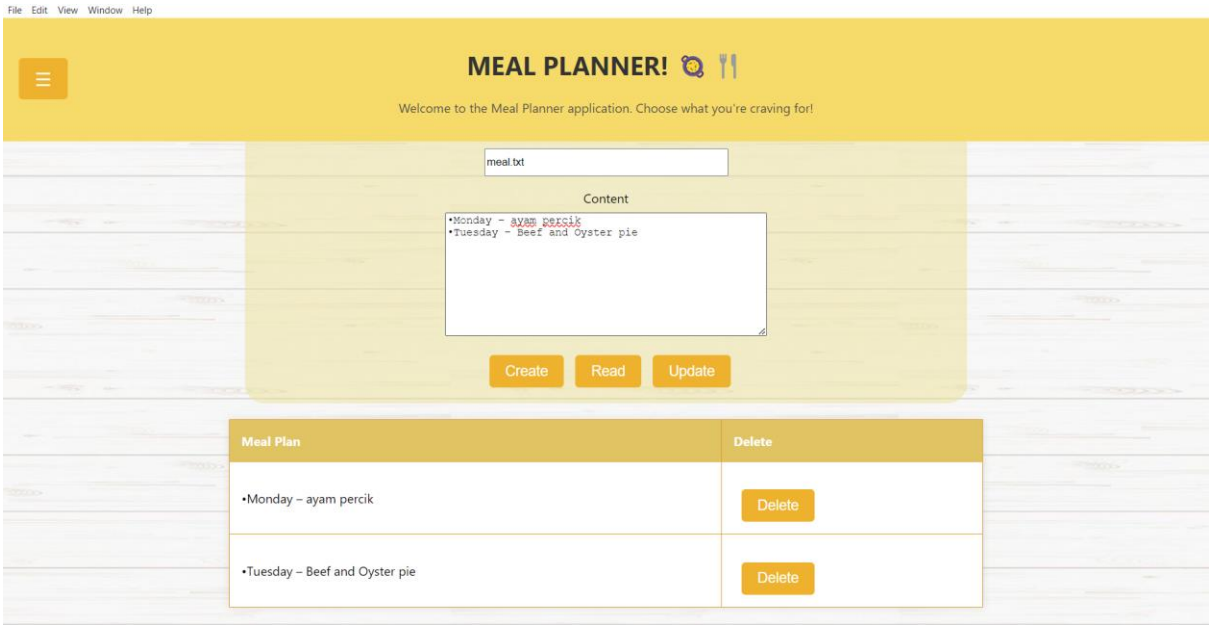


If the user clicks the ‘Read’ button, the content from ‘meal.txt’ will be displayed in the form also in the table below the form, allowing the user to read the content they had previously created and added. In the table, user can click the ‘Delete’ button if they want to delete any of the content in the ‘meal.txt’ like shown below:

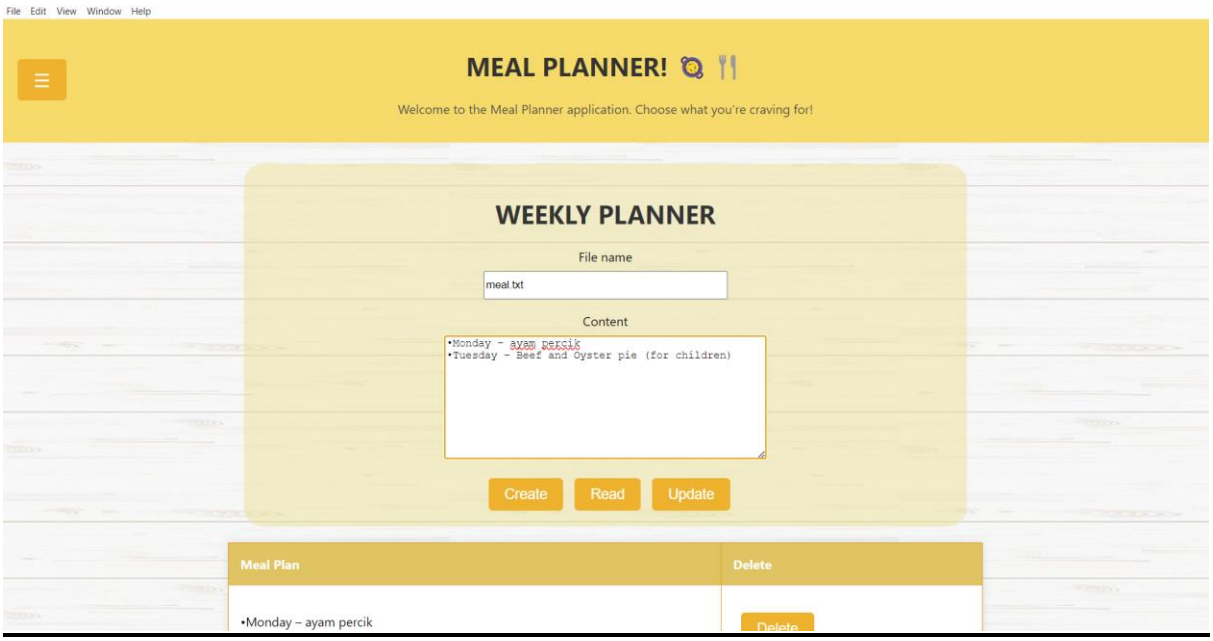
After click the “Read” button:



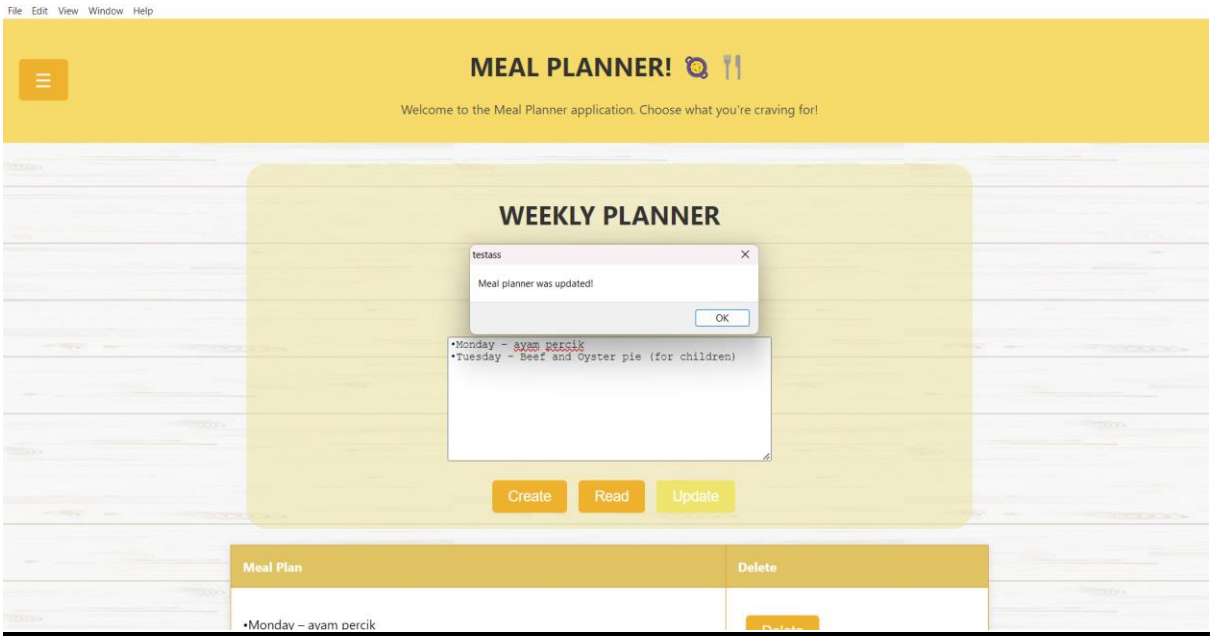
After click the “Delete” button:

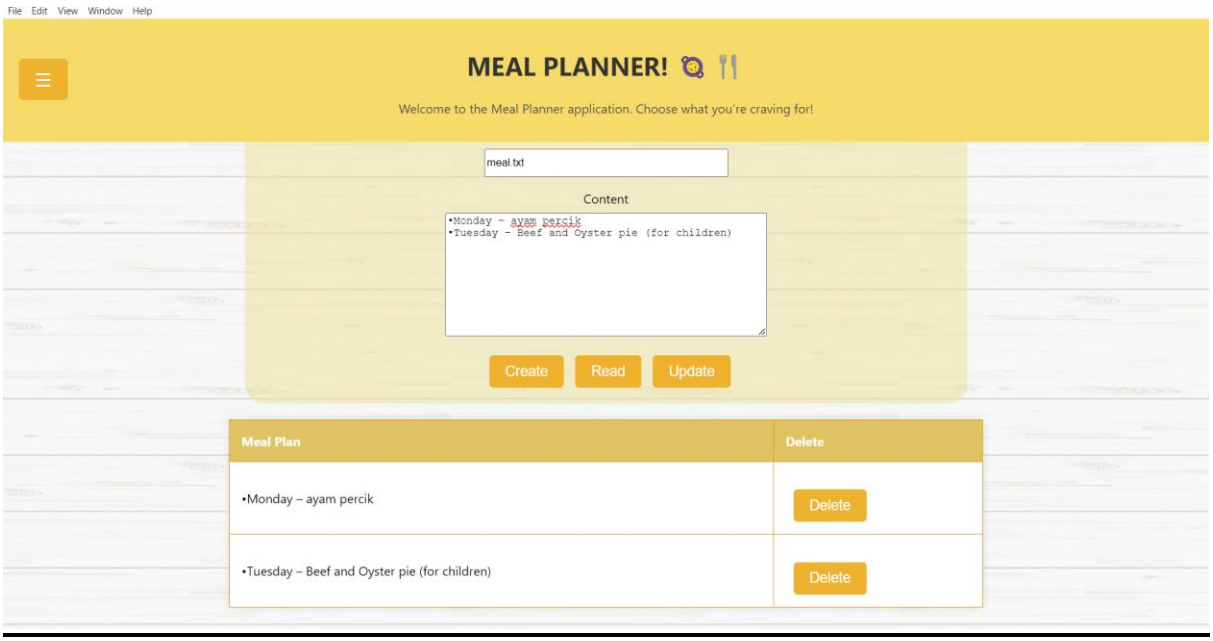


If a user wants to update the ‘meal.txt’ file, they need to click the ‘Read’ button first. After clicking ‘Read’ the system will display the content in the form. Once the content is displayed, the user can make updates within the form and then click the ‘Update’ button. After clicking ‘Update’ the user will receive an alert that says ‘Meal Planner was updated!’ to inform the user that the file has been successfully updated. Here’s an example of updated content in the form:



After clicking the “Update” button:




File Edit View Window Help
MEAL PLANNER!
Welcome to the Meal Planner application. Choose what you're craving for!
meal.txt
Content
•Monday – ayam percik
•Tuesday – Beef and Oyster pie (for children)
Create
Read
Update
Meal Plan
Delete
•Monday – ayam percik
Delete
•Tuesday – Beef and Oyster pie (for children)
Delete

PROGRAM CODE OF SYSTEM

home.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Homepage</title>
    <link rel="stylesheet" href="home.css" />
</head>
<body>
    <!--add menu sidebar at the top left website-->
    <div class="banner">
        <!--add date and time at the top of website-->
        <div id="datetime">
        </div>
        <!--add picture content of website-->
        <div class="content">
            <h1> Let Your Taste Buds Guide You </h1>
            <p style="color: whitesmoke;"> Relax please, we've got you covered
every day of the week </p>
            <!--add button to go to notes page-->
            <div>
                <a href="index.html">
                    <button type="button"><span id="learn"></span>EXPLORE
MORE</button>
                </a>
            </div>
        </div>
    </div>
    <script>
        // Get the current date and time
        function getCurrentDateTime() {
            var currentDate = new Date();
            var date = currentDate.toDateString();
            var time = currentDate.toLocaleTimeString();
            return date + ' ' + time;
        }
```

```
        // Update the date and time every second
        function updateDateTime() {
            var datetimeElement = document.getElementById('datetime');

            // Update the content
            datetimeElement.innerHTML = getCurrentDateTime();

            // Schedule the next update
            setTimeout(updateDateTime, 1000);
        }
        // Start updating the date and time
        updateDateTime();
    </script>
</body>
</html>
```

home.css

```
*{
  margin: 0;
  padding: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Helvetica,
Arial, sans-serif;
}
/* BACKGROUND SETTINGS */
.banner{
  width: 100%;
  height: 100vh;
  background-image: linear-
gradient(rgba(0,0,0,0.75),rgba(0,0,0,.75)),url(bghmp.jpg);
  background-size: cover;
  background-position: center;
}
/* MENU SETTINGS */
#menu{
  font-size: 30px;
  cursor: pointer;/*kalau lalu nnti keluar tangan*/
 position: absolute;
  top: 30px;
  left: 30px;}
```

```
/* DATE AND TIME SETTINGS */
#datetime {
  color: #ECE6B1;
  text-align: center;
  padding-top: 40px;
  font-weight: bold;
  font-size: 30px;
}
/* CONTENT SETTINGS */
.content{
  width: 95%;
  margin: 50px auto;
  position: absolute;
  top: 50%;
  transform: translateY(-50%);
  text-align: center;
  color:white;
}
.content h1{
  font-size: 40px;
  margin-top: 80px;
  margin-left: 80px;
}
.content p{
  margin: 30px auto;
  font-weight: 100;
  line-height: 25px;
  margin-left: 90px;
  margin-right: 60px;
}
/* BUTTON SETTINGS (GO TO NOTES PAGE)*/
button{
  width: 200px;
  padding: 15px 0;
  text-align: center;
  margin: 20px 10px;
  border-radius: 25px;
  font-weight: bold;
  border: 2px solid rgb(247, 219, 106);
  background-color: transparent;
```

```
  color: #fff;
  cursor: pointer;
  position: relative;
  overflow: hidden;
}
#learn{
  background: rgb(247, 219, 106);
  height: 100%;
  width: 0;
  border-radius: 25px;
  position: absolute;
  left: 0;
  bottom: 0;
  z-index: -1;
  transition: 0.5s;
}
button:hover #learn{
  width: 100%;

button:hover{
  border:none;
}
```

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Meal Categories</title>
    <link rel="stylesheet" href="index.css" />
  </head>
  <body>
    <header>
      <button id="sidebarToggle">☰</button>
      <h1>MEAL PLANNER! 🧑‍🍳 🍴</h1>
      <p>Welcome to the Meal Planner application. Choose what you're craving
for!</p>
    </header>
```

```
    <div id="sidebar" class="closed">
      <br><br>
      <br><br>
      <br><br>
      <br><br>
      <div id="closebutton">X</div>
      <a href="index.html">Home</a><br><br>
      <a href="CRUD.html">Planner</a><br><br>
    </div>

    <br><br>
    <br><br>
    <br><br>

    <div>
      <label for="searchData"></label>
      <select id="searchData">
        <option value="" disabled selected>Choose Category</option>
      </select>
      <br><br>
      <button id="searchButton">Search</button>
    </div>

    <div id="categories-list">
      <p><b>Select a category to display meals.</b></p>
      <p><b>All meals:</b></p>
    <div class="meals-container" id="allMealsContainer"></div>
    </div>

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

index.css

```
body {
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Helvetica,
Arial, sans-serif;
  margin: auto;
  max-width: 950px;
```

```
  background-image:url("wood3.jpg");
  background-color: rgba(255, 255, 255, 0.10);
  padding: 2rem;
  text-align: center;
}
header {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  background-color: rgb(247, 219, 106);
  padding: 1rem 0;
  z-index: 100;
h1 {
  color: #333;
  font-size: 2rem;
}
p {
  color: #4d4c4c;
  font-size: 1rem;
}
div {
  margin-top: 1rem;
}
label {
  display: block;
  font-size: 1rem;
  margin-bottom: 0.5rem;
}
select {
  width: 50%;
  padding: 0.5rem;
  border: 2px solid rgb(245, 188, 17);
  border-radius: 15px;
  font-size: 1rem;
  background: white;
}

button {
  background: rgb(245, 188, 17);
```

```
  color: white;
  border: none;
  padding: 0.5rem 1rem;
  border-radius: 5px;
  font-size: 1rem;
  cursor: pointer;
  margin-left: 1rem;
}
#categories-list {
  margin-top: 1rem;
  text-align: center;
}
.meals-container {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  padding: 5px;
}
.meal-row {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  width: 100%;
  margin: 0.5rem 0;

.meal {
  margin: 1rem;
  padding: 5px;
  border: 1px solid #ccc;
  border-radius: 5px;
  background: white;
  text-align: center;
  box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
  flex: 0 0 calc(30% - 2rem);
  max-width: calc(30% - 2rem);
  display: flex;
  flex-direction: column;
  justify-content: center;
}
img {
```

```
  max-width: 100%;
  height: auto;
}
h3 {
  font-size: 1rem;
  color: #333;
}
.view-button {
  background: rgb(240, 161, 13);
  color: white;
  border: none;
  padding: 0.5rem 1rem;
  border-radius: 5px;
  font-size: 1rem;
  cursor: pointer;
  margin: 0 auto;
  display: block;
}
ul {
  text-align: left;
}
ul li {
  list-style-type: disc;
  margin-left: 10px;
 }
.mealpic {
  width: 500px;
  margin: auto;
}
#categories-list {
  margin-top: 1rem;
  text-align: center;
}
#sidebar {
  position: fixed;
  top: 0;
  left: -250px; /* Hidden by default */
  width: 250px;
  height: 100%;
  background-color: rgb(109, 107, 99);
```

```
  z-index: 1;
  transition: 0.3s;
}

#closebutton {
  cursor: pointer;
  position: absolute;
  top: 10px;
  right: 10px;
  font-size: 24px;
  color: black;
}

#closebutton:hover {
  color: white;
}

#sidebar a {
  padding: 15px 25px;
  text-decoration: none;
  font-size: 20px;
  color: white;
  display: block;
  transition: 0.3s;
}

#sidebar a:hover {
  color: white;
  font-size: 22px;
}

#sidebar.closed {
  left: -250px;
}

#sidebar.open {
  left: 0;
}

button#sidebarToggle {
```

```
  position: fixed;
  top: 50px;
  left: 10px;
  font-size: 24px;
  cursor: pointer;
  color: #fff;
  z-index: 2;
}

button#sidebarToggle:hover {
  color: #777;
}
```

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, 'home.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
```

fetchapi.js

```
document.addEventListener('DOMContentLoaded', () => {
  const allMealsContainer = document.getElementById('allMealsContainer');
  const apiUrl = 'https://www.themealdb.com/api/json/v1/1/search.php?s=';
  let page = 1;
  const displayedMeals = new Set();

  const fetchAllMeals = () => {
    fetch(`${apiUrl}&page=${page}`)
```

```
.then((response) => response.json())
   .then((data) => {
     const meals = data.meals;
     if (meals) {
       meals.forEach((meal) => {
         if (!displayedMeals.has(meal.strMeal)) {
           displayedMeals.add(meal.strMeal);

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

           const mealImage = document.createElement('img');
           mealImage.src = meal.strMealThumb;
           mealImage.alt = meal.strMeal;

           const mealDetails = document.createElement('div');
           mealDetails.className = 'meal-details';

           const mealName = document.createElement('h3');
           mealName.textContent = meal.strMeal;

           const viewButton = document.createElement('button');
           viewButton.className = 'view-button';
           viewButton.textContent = 'View';
           viewButton.dataset.mealName = meal.strMeal;

           mealDetails.appendChild(mealName);
           mealDetails.appendChild(viewButton);

           mealDiv.appendChild(mealImage);
           mealDiv.appendChild(mealDetails);

           allMealsContainer.appendChild(mealDiv);
         }
       });

       // Fetch the next page if available
       if (meals.length > 0) {
         page++;
```

```
            fetchAllMeals();
          }
        } else {
          allMealsContainer.innerHTML = '<p>No meals found.</p>';
        }
      })
      .catch((error) => {
        console.error('Error fetching data:', error);
        allMealsContainer.innerHTML = '<p>Error fetching meals.</p>';
      });
  };

  fetchAllMeals();
});

function fetchMealCategories() {
    fetch("https://www.themealdb.com/api/json/v1/1/list.php?c=list")
      .then((response) => response.json())
      .then((data) => {
        const categoryDropdown = document.getElementById("searchData");

        data.meals.forEach((category) => {
          const option = document.createElement("option");
          option.value = category.strCategory;
          option.textContent = category.strCategory;
          categoryDropdown.appendChild(option);
        });
      })
      .catch((error) => {
        console.error("Error fetching meal categories: " + error);
      });
  }

  // Call the function to populate the dropdown
  fetchMealCategories();

  // Add an event listener for the category dropdown
  const searchData = document.getElementById("searchData");
  searchData.addEventListener("change", function () {
```

```
    const selectedCategory = searchData.value;
    fetchCategoryMeals(selectedCategory);
  });

  // Function to fetch meals based on the selected category
  function fetchCategoryMeals(category) {
    const categoriesList = document.getElementById("categories-list");
    categoriesList.innerHTML = "Loading meals for the selected category...";

    fetch(`https://www.themealdb.com/api/json/v1/1/filter.php?c=${category}`)
      .then((response) => response.json())
      .then((data) => {
        if (data.meals) {
          const mealsHTML = data.meals.map((meal) => {
            return `
              <div class="meal">
                <div class="meal-image">
                  <img src="${meal.strMealThumb}" alt="${meal.strMeal}">
                </div>
                <div class="meal-details">
                  <h3>${meal.strMeal}</h3>
                  <button class="view-button" data-meal-
name="${meal.strMeal}">View</button>
                </div>
              </div>
            `;
          }).join('');
          categoriesList.innerHTML = `<div class="meals-
container">${mealsHTML}</div>`;
        } else {
          categoriesList.innerHTML = `No meals found for the category
"${category}".`;
        }
      })
      .catch((error) => {
        categoriesList.innerHTML = "Error fetching data.";
        console.error("Error:", error);
      });
  }
  // Add an event listener for the "View" buttons
```

```
  const categoriesList = document.getElementById("categories-list");
  categoriesList.addEventListener("click", function (event) {
    if (event.target.classList.contains('view-button')) {
      const mealName = event.target.getAttribute('data-meal-name');
      fetchMealDetails(mealName);
    }
  });

  // Function to fetch detailed meal information
  function fetchMealDetails(mealName) {
    const categoriesList = document.getElementById("categories-list");
    categoriesList.innerHTML = "Loading meal details...";

    fetch(`https://www.themealdb.com/api/json/v1/1/search.php?s=${mealName}`)
      .then((response) => response.json())
      .then((data) => {
        if (data.meals && data.meals.length > 0) {
          const meal = data.meals[0];
          const ingredients = [];

          // Loop through the ingredients and add them to the ingredients array
          for (let i = 1; i <= 20; i++) {
            const ingredient = meal[`strIngredient${i}`];
            const measure = meal[`strMeasure${i}`];
            if (ingredient && measure) {
              ingredients.push(`${measure} ${ingredient}`);
            }
          }

          const mealDetailsHTML = `
          <div class="meal-details">
            <h2>${meal.strMeal}</h2>
            <div class="mealpic"><img src="${meal.strMealThumb}"
alt="${meal.strMeal}"></div>
            <h3>Ingredients:</h3>
            <ul>
              ${ingredients
                .map((ingredient) => `<li>${ingredient}</li>`)
                .join('')}
            </ul>
```

```
            <h3>Instructions:</h3>
            <p>${meal.strInstructions}</p>
            <h3>Video:</h3>
            <p><a href="https://www.youtube.com/results?search_query=${mealName}"
target="_blank">Watch ${mealName} Recipe Video</a></p>
            <h3>Related Site:</h3>
            <p><a href="${meal.strSource}" target="_blank">${meal.strMeal} Recipe
on the Official Site</a></p>
          </div>
          `;
          categoriesList.innerHTML = mealDetailsHTML;
        } else {
          categoriesList.innerHTML = `No details found for the meal:
"${mealName}".`;
        }
      })
      .catch((error) => {
        categoriesList.innerHTML = "Error fetching data.";
        console.error("Error:", error);
      });
  }
  document.addEventListener("DOMContentLoaded", function () {
    const sidebar = document.getElementById("sidebar");
    const sidebarToggle = document.getElementById("sidebarToggle");
    const closebutton = document.getElementById("closebutton");

    sidebarToggle.addEventListener("click", function () {
      sidebar.classList.toggle("open");
    });
    closebutton.addEventListener("click", function () {
      sidebar.classList.toggle("open");
    });
  });
```

CRUD.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
```

```
        <title>CRUD App</title>
        <link rel="stylesheet" href="CRUD.css" />
        <link rel="stylesheet" href="photon.min.css" />
    </head>
    <body>
        <header>
            <button id="sidebarToggle">☰</button>
            <h1>MEAL PLANNER! 🧑‍🍳 🍴</h1>
            <p>Welcome to the Meal Planner application. Choose what you're craving
for!</p>
          </header>

          <div id="sidebar" class="closed">
            <br><br>
            <br><br>
            <br><br>
            <br><br>
            <div id="closebutton">✕</div>

            <a href="index.html">Home</a><br><br>
            <a href="CRUD.html">Planner</a><br><br>
          </div>

          <br><br>
          <br><br>
          <br><br>

        <div class="mainWrapper">
            <div class="transparent-container">
                <h1>WEEKLY PLANNER</h1>
                <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>
                        <textarea id="fileContents" class="form-control"
rows="5"></textarea>
```

```
                    </div>
                </form>

                <button id="btnCreate" class="btn btn-default">Create</button>
                <button id="btnRead" class="btn btn-default">Read</button>
                <button id="btnUpdate" class="btn btn-default">Update</button>
            </div>

            <table id="fileTable">
                <thead>
                    <tr>
                        <th>Meal Plan</th>
                        <th>Delete</th>
                    </tr>
                </thead>
                <tbody id="fileTableBody"></tbody> <!-- Table for displaying meal
plans -->
            </table>
        </div>

        <script>
            document.addEventListener("DOMContentLoaded", function () {
                const sidebar = document.getElementById("sidebar");
                const sidebarToggle = document.getElementById("sidebarToggle");
                const closebutton = document.getElementById("closebutton");
                sidebarToggle.addEventListener("click", function () {
                    sidebar.classList.toggle("open");
                });

                closebutton.addEventListener("click", function () {
                    sidebar.classList.toggle("open");
                });
            });
        </script>

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

CRUD.css

```
body {
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Helvetica,
Arial, sans-serif;
    margin: auto;
    max-width: 950px;
    background-image:url("wood3.jpg");
    background-color: rgba(255, 255, 255, 0.10);
    padding: 2rem;
    text-align: center;
  }

  header {
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    background-color: rgb(247, 219, 106);
    padding: 1rem 0;
    z-index: 100;}

  h1 {
    color: #333;
    font-size: 2rem;
  }

  p {
    color: #4d4c4c;
    font-size: 1rem;
  }

  div {
    margin-top: 1rem;
  }
  label {
    display: block;
    font-size: 1rem;
    margin-bottom: 0.5rem;
  }
```

```
select {
  width: 50%;
  padding: 0.5rem;
  border: 2px solid rgb(245, 188, 17);
  border-radius: 15px;
  font-size: 1rem;
  background: white;

}

button {
  background: rgb(245, 188, 17);
  color: white;
  border: none;
  padding: 0.5rem 1rem;
  border-radius: 5px;
  font-size: 1rem;
 cursor: pointer;
  margin-left: 1rem;
}

#categories-list {
  margin-top: 1rem;
  text-align: center;
}

.meals-container {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  padding: 5px;
}

.meal-row {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  width: 100%;
  margin: 0.5rem 0;  }
```

```
.meal {
  margin: 1rem;
  padding: 5px;
  border: 1px solid #ccc;
  border-radius: 5px;
  background: white;
  text-align: center;
  box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
  flex: 0 0 calc(30% - 2rem);
  max-width: calc(30% - 2rem);
  display: flex;
  flex-direction: column;
  justify-content: center;
}

img {
  max-width: 100%;
  height: auto;
}

h3 {
  font-size: 1rem;
  color: #333;
}

#sidebar {
  position: fixed;
  top: 0;
  left: -250px;
  width: 250px;
  height: 100%;
  background-color: rgb(109, 107, 99);
  z-index: 1;
  transition: 0.3s;
}

#closebutton {
  cursor: pointer;
  position: absolute;
  top: 10px;
```

```
  right: 10px;
  font-size: 24px;
  color: black;
}

#closebutton:hover {
  color: white;
}

#sidebar a {
  padding: 15px 25px;
  text-decoration: none;
  font-size: 20px;
  color: white;
  display: block;
  transition: 0.3s;
}

#sidebar a:hover {
  color: white;
  font-size: 22px;
}

#sidebar.closed {
  left: -250px;
}

#sidebar.open {
  left: 0;
}

button#sidebarToggle {
  position: fixed;
  top: 30px;
 left: 10px;
  font-size: 24px;
  cursor: pointer;
  color: #fff;
  z-index: 2;
}
```

```
  button#sidebarToggle:hover {
    color: #777;
  }

  #fileName {
    width: 300px;
    height: 30px;
  }

  #fileContents {
    width: 400px;
    height: 150px;
  }

table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
 overflow: hidden; /* Hide overflowing content */
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Add a subtle shadow */
}

table th, table td {
  border: 1px solid rgb(223, 151, 35);
  padding: 15px; /* Increase padding for better spacing */
  text-align: left;
  transition: background-color 0.3s; /* Add smooth transition */
}

table tr {
  margin-bottom: 10px;
  background-color: #fff;
table th {
  background-color: rgb(224, 195, 98);
  color: #fff;
}

#fileTableBody {
  background-color: #d5b76c;
}
```

```
button {
  background-color: rgb(238, 178, 47);
  color: #fff;
  border: none;
  border-radius: 5px;
  padding: 10px 20px;
  font-size: 18px;
  margin-top: 20px;
  margin-left: 10px;
  cursor: pointer;
  transition: background-color 0.3s;
}

button:hover {
  background-color: rgb(239, 229, 110);
}

.button {
  border: none;
  border-radius: 60px;
  padding: 10px 20px;
  font-size: 18px;
  margin-top: 20px;
  margin-left: 10px;
  cursor: pointer;
  transition: background-color 0.3s;
}

main {
  width: 80%;
  margin: 0 auto;
  padding: 20px;
  text-align:center;
}

.transparent-container {
  background-color: rgba(236, 226, 151, 0.5); /* Adjust the alpha channel for
transparency */
  padding: 20px;
```

```
  border-radius: 25px;
  margin: 20px;
}
```

CRUD.js

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

//BUTTON
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')

//CREATE BUTTON
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")
    })
})

//READ BUTTON
btnRead.addEventListener('click', function() {
    let file = path.join(pathName, fileName.value);
```

```
    fs.readFile(file, 'utf8', function(err, data) {
        if (err) {
            console.error(err);
            return console.log(err);
        }
        fileContents.value = data;
        const lines = data.split('\n')
        const tableBody = document.getElementById('fileTableBody');
        tableBody.innerHTML = '';
        lines.forEach(function(line) {
            addContentToTable(line);
        });
        console.log("Meal planner has been Read!");
    });
});

//UPDATE BUTTON
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);
        }
        console.log("Meal planner was updated!")
        alert( "Meal planner was updated!")
        })
})

//DELETE BUTTON
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("Meal planner was deleted!")
    })
})

//TABLE
    function addContentToTable(content) {
        const newRow = document.createElement('tr');

        const contentCell = document.createElement('td');
        contentCell.textContent = content;

        const deleteCell = document.createElement('td');
        const deleteButton = document.createElement('button');
        deleteButton.textContent = 'Delete';

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

            const row = this.parentNode.parentNode;
            const rowIndex = row.rowIndex;

            row.remove();

            const file = path.join(pathName, fileName.value);
            fs.readFile(file, 'utf8', function(err, data) {
                if (err) {
                    console.error(err);
                    return console.log(err);
                }
                const lines = data.split('\n');
                lines.splice(rowIndex - 1, 1);
                const updatedData = lines.join('\n');
                fs.writeFile(file, updatedData, 'utf8', function(err) {
                    if (err) {
                        console.error(err);
                    }
                });
            });
        });

        deleteCell.appendChild(deleteButton);
```

```
        newRow.appendChild(contentCell);
        newRow.appendChild(deleteCell);

        const tableBody = document.getElementById('fileTableBody');
        tableBody.appendChild(newRow);
    }
```

preload.js

```
// See the Electron documentation for details on how to use preload scripts:
// https://www.electronjs.org/docs/latest/tutorial/process-model#preload-scripts
```

Link : https://github.com/naddysyazwn/testass.git