JS - Assessment 3

Introduction

As we continue building the **Food Order Application**, the next phase focuses on gradually integrating **JavaScript** into our web pages. In this stage, you will enhance the **interactivity** of the application by implementing **sign-in authentication**, performing **form data validation**, and **dynamically updating HTML tables** based on user input. This hands-on approach will strengthen your understanding of using **JavaScript** to transform static pages into **dynamic and responsive** user interfaces.

Objectives of the Assessment:

✔ Understand how to **implement sign-in authentication** using JavaScript.
✔ Learn how to **handle form submission** and **process form data**.
✔ **Dynamically generate table rows** using JavaScript data.
✔ **Add submitted form data into the table.**
✔ **Write clean and error-free backend** JavaScript logic.
✔ **Recognize the role and importance of JavaScript** in web development.

Problem Statement

1. **Sign-In Page**

   As the sign-in HTML page is given in the source code,

   - **Create** a file named **authentication.js** inside the **JS folder** and **link it** to the corresponding **HTML file** using the `<script>` tag.
   - **Fetch the username and password** entered in the form when the user **submits** it.
   - **Validate** that the username is "**great**" and the password is "**learning**" to allow access.
   - If the credentials are **correct**, use browsers location object to **redirect** to the admin page in the **same tab** and **prevent back navigation** to the login page.
   - If the credentials are **incorrect**, **display an error message** asking the user to try again.
   - **Allow unlimited attempts** for the user to log in.

2. **Add Fooditem**

   - Create a file named **`add_fooditem.js`** inside the **JS folder** and **link it** to the corresponding **HTML file** using the `<script>` tag.
   - **Capture the form data** when the user **clicks Submit**.
   - If the **Category** or **Cuisine** fields are **empty**, show an **alert** prompting the user to fill them.
   - After capturing the data, **create an object** named **newFoodItem** with the following properties: **name, description, inStock, isVeg, category, cuisine, and foodImage**.
   - **Reset the form** after the object is created.
   - Use **URLSearchParams(object).toString()** to convert the object into a **query string**.

- **Redirect** the user to **list_fooditems.html** and **pass the query string** along with the URL using **location** object**.**

3. **List Food Items**

   Go through the given list_foodItems.js file, you will see an Array of Objects.

   - Link this file to the corresponding **HTML file** using the `<script>` tag.
   - In this file, **create a function** named **addFoodItem** that takes two parameters — a **foodItemObject** and a **number** (used as the serial number).
   - Use **DOM manipulation** and **template strings** to **build a table row** that aligns with the table headers in **list_fooditems.html**, and **append it to the <tbody>** of the table.
   - Create another function called **loadAllFoodItems** that will **loop through the array** of food items and, for each item, **call addFoodItem** with the item and its serial number.
   - As discussed in the **mentor session**, use **URLSearchParams(window.location.search)** to **retrieve the query string**, and convert it into an object.
   - **Add this object** to the existing **array of food items**, and **display it** in the table by calling **addFoodItem** with the new object.

!!! Happy Coding !!!