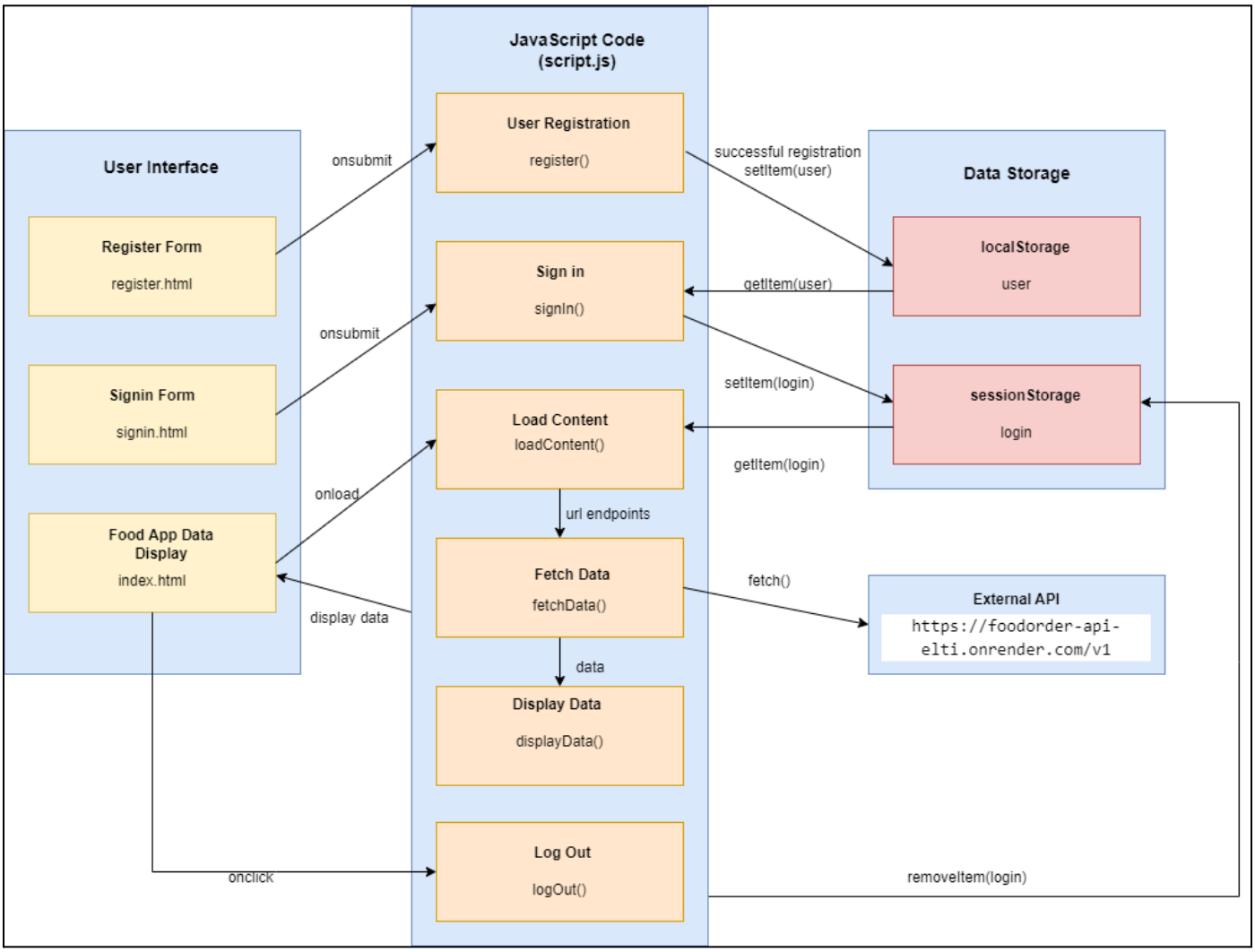
Advanced JavaScript - Assessment Project

Introduction

This week, we will be building **a basic web-based Food Order application**. The goal of this project is to create a web-based application that allows users to register, sign in, and view a list of available food items and restaurants.

Let's understand the components of this application -



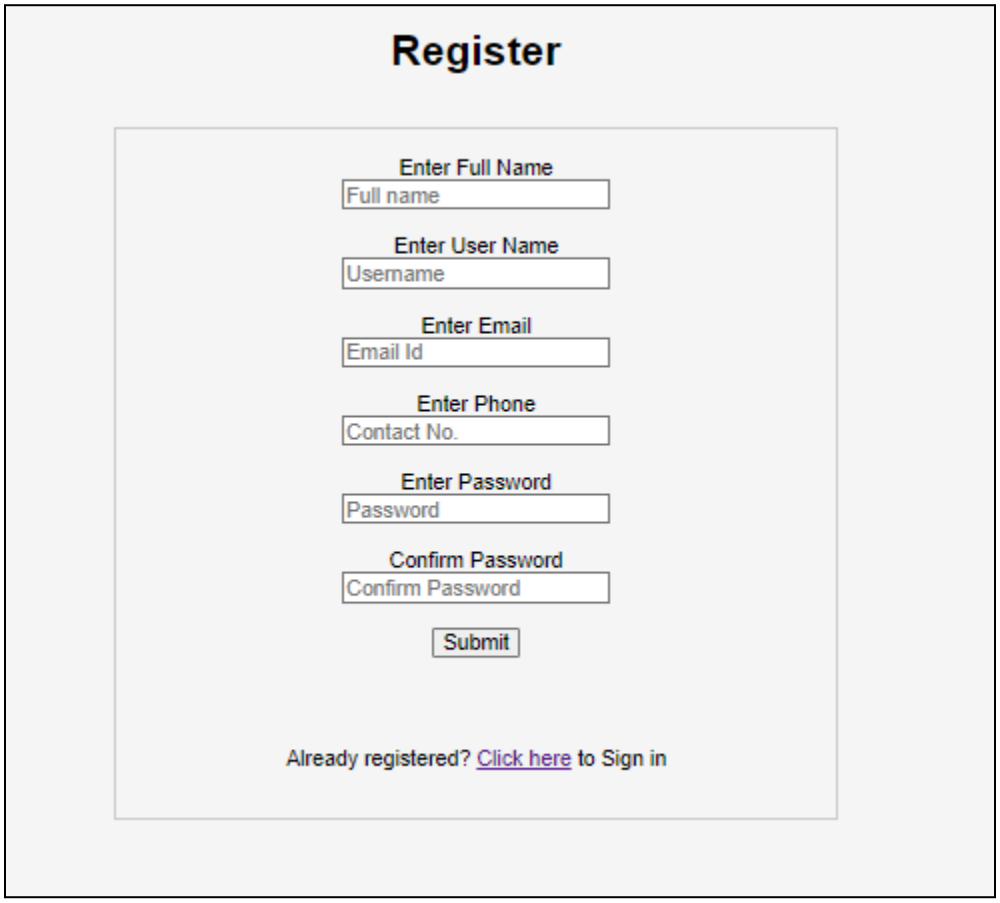
The details of this application are given below:

- When the user goes to the home page, one of the following two things may happen:

- If the user is registered and logged in, then the user can see the index.html page where all the cuisines and categories of food items and restaurants are displayed.

- If the user is not registered and logged in, then the **Sign In** page appears as below:

- If the user is trying to sign-in but not registered yet, he/she can click on the **Click here** button to go to the **Register** page.



- There's a Logout button on the home page. If the user wants to logout, he/she can click on the button and will be re-directed to the **Sign In** page.

While building this application, we will be using various concepts like user authentication, windows local storage and session storage, async await, fetch api, json(), error-handling etc.

- You will find the following sub-folders in the folder:
  - **html:** There are three files under this sub-folder.
    - i. **index.html:** This file contains the home page of the website where all the categories, cuisines and restaurants are displayed.
    - ii. **register.html:** This file contains the user registration form.
    - iii. **signin.html:** This file contains the user sign-in form.
  - **css:** There is one file under this sub-folder.
    - i. **style.css:** This file contains the styling for index.html.
  - **js:** There is one file under this sub-folder.
    - i. **script.js:** In this file, you have to add code to complete the tasks which are mentioned in the Problem Statement.

- To execute the app, run the index.html on the live server or copy its path and open in the browser.

Housekeeping points

- This is a minimal example and may not follow some standard practices.
- We focus on the main flow, and not much error handling.

Problem Statement

In this project, we will build a **web-based E commerce application** that helps users to register, sign in and view information of available cuisines, categories and restaurants. The application should have the following features :

1. **User Registration:** You have to write code for the function **register().**
    - This function helps in getting the values of the full name, username, email, contact information, password, and confirm the password etc. from the **register.html** page.
    - The function also ensures password confirmation matches the entered password before allowing registration.
    - Stores user information locally using appropriate data storage techniques (e.g., local storage). And redirects to the **signin.html** page.
    - Provides informative error messages to guide users in case of passwords mismatch.

2. **User Authentication:** You have to write code for the function **signIn().**
    - This function designs a sign-in functionality where registered users can log in with their username and password.
    - Validates user credentials against the stored user data.
    - Implements session management using session storage upon successful login to maintain the user's login status.
    - Implements error handling for incorrect login credentials or any unexpected issues during data fetching or processing.
    - Provides informative error messages to guide users in case of registration or login failures.

3. **Cuisines, Categories and Restaurants Data Fetch:** You have to write code for **fetchData()** function.
    - This is a common function to fetch data from an API.
    - This function is called by the **loadContent()** function to load cuisines, categories and restaurants data.
    - The **loadContent()** function is called on the onload event of the **index.html** page. This function is already implemented.

- The data will be fetched from an external API **https://foodorder-api-elti.onrender.com/v1**.
- The URL endpoints to fetch cuisines, categories and restaurants information are **https://foodorder-api-elti.onrender.com/v1/cuisines**, **https://foodorder-api-elti.onrender.com/v1/categories** and **https://foodorder-api-elti.onrender.com/v1/restaurants** respectively.
- The **fetchData()** function is called thrice by the **loadContent()** function to fetch cuisine data, category data and restaurants data respectively.
- The above mentioned URL endpoints are already mentioned in the **loadContent()** function.
- The **fetchData()** function takes two arguments: URL endpoints and the respective list elements from the index.html page.
- The function is declared as async as its fetching data asynchronously.
- It also handles errors in case the data could not be fetched.

4. **Cuisines, Categories and Restaurants Data Display:** You have to write code for **displayData()** function.
    - This is a common function to display data. This is called by the **fetchData()** function to display data after fetching it.
    - It takes two arguments: data and list element.
    - It traverses over the data objects and adds the respective images and names to the innerHTML of the list element.
    - This is how the data gets displayed on the **index.html** page.

5. **Log Out:** You have to write code for the **logout()** function.
    - This function sets the login status to false.
    - It also redirects to the signin.html page.

Functions Which are Already Implemented

1. **loadContent():**
    a. This function is responsible for loading the cuisines, categories and restaurants data on the index.html(home page).
    b. The function calls two other functions.
    c. If the user has logged in, it calls **fetchAndLoadData()** function to fetch the data from the external API and load it.
    d. Else, it calls the **loadSignInPage()** function.

2. **fetchAndLoadData():** This function is responsible for
    a. Getting all the URLs.
    b. Getting all the HTML list elements
    c. Fetching data

    It calls three functions for doing the above tasks:
    a. fetchURLs()

b. **loadListElements()**
c. **fetchData()**

3. **fetchURLs():** This function is responsible for fetching and returning the cuisine, category and restaurant URL.

4. **loadListElements():** This function is responsible for fetching the HTML list elements for data display.

5. **loadSignInPage():** This function is responsible for loading the signin.html page.

6. **fetchRegisterPageData():** This function is responsible for fetching the form values from register.html page and returning them. Call this function in register() to fetch values.

7. **fetchSignInPageData():** This function is responsible for fetching the form values from signin.html page and returning them. Call this function in signIn() to fetch values.

Program Organization

- You will be getting a zip folder containing a folder named **Advanced JavaScript_Assessment_For Coders.**
- The **Advanced JavaScript_Assessment_For Coders** folder has three subfolders namely html, css and js and five files under these sub-folders namely index.html, register.html, signin.html, script.js and style.css.
- You are required to add functionalities to complete the tasks (stated above in the problem statement) in **script.js** file.

Evaluation Rubric

Total Project Points: 60

- Correctness:
  Correctness of implementation
  - Problem statement - point 1 : **10 Points**
  - Problem statement - point 2 : **10 Points**
  - Problem statement - point 3 : **25 Points**
  - Problem statement - point 4 : **10 Points**
  - Problem statement - point 5 : **5 Points**

Program Instructions

- Make sure you zip the **Advanced JavaScript_Assessment_For Coders** folder before submitting the project.
- Project will not be evaluated if the submitted project is not in the zip/rar format.