# Front-End Technologies Basics

## DOM Manipulation

**Use the provided skeleton to solve this problem.**

**Write the missing functionality** of this user interface.

### Application structure

You are provided with an HTML application - **Smoothie Order System** with the following structure:

A screenshot of a computer

Description automatically generated

Open the index.html to start the application. Its purpose is to order smoothies. The application looks like this:



Look at the html structure in the index.html file. You will notice that there are 2 hidden elements:





The preview element is supposed to be displayed after the 'Customize Smoothie' button is clicked. The element **with id="order-success"** is supposed to be displayed after the order is finalized.

### Your Task

**Write the missing JavaScript code in the app.js file** to make the **Smoothie Order System** work as expected:

### Customize Smoothie Button Functionality

A screenshot of a phone

Description automatically generated

When the **[Customize Smoothie]** button is clicked, you need to validate that all fields are filled. If this is not the case, do not proceed.

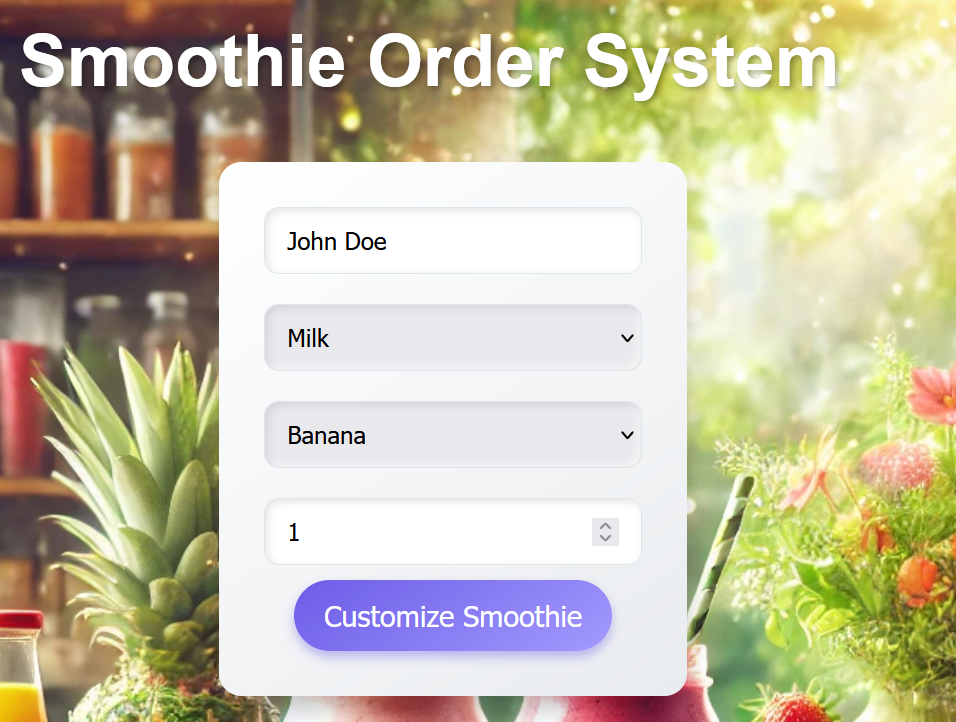
**Do NOT include any additional validation for the fields.**

If the validation is successful, the following functionality needs to be implemented:

* The information about the smoothie is filled in the preview element – name, base, fruit, sweetener.
* The preview element must be displayed.
* The 'Customize Smoothie' button must be disabled.
* The values of the fields in the form must be cleared.

### Edit Smoothie Info

**When the [**Edit] **button is clicked, all of the information is loaded back into the input fields from step 1, while the** [**Customize Smoothie**] **button is enabled again.**



The preview element must be hidden.

### Confirm Smoothie

**When the** [Confirm] **button is clicked,** you must hide the preview element and show the element with **id="order-success".**

A screenshot of a smoothie order system

Description automatically generated

### Back

**When the** [Back] **button is clicked,** you must **hide the element with id="order-success** and enable the [**Customize Smoothie**] **button**.

## JS Application Testing

You have been given a JavaScript application. All the necessary setup is complete, allowing you to start writing your tests. Your goal is to write end-to-end (Front-End) tests using Playwright.

### App Introduction

The **application for testing** is called **"Recipe Sharing Hub"**. It's a user-friendly application offering functionalities and permissions based on user login status:

1. For **guest user**:

* **Home Page**: View a brief introduction to the app.
* **Discover page**: See all created recipes.
* **Search Page**: Search for recipes by name.
* **Register/Login page**: Register a new account or log in.

1. For **Logged-In user**:

* **Create Recipe Page**: Create new recipes.
* **Discover**: See all created recipes (both their own and others').
* **Search Page**: Search for recipes by name.
* **Logout**: Log out of the app.

1. **Recipe details functionality**:

* **Detail View**: Only logged-in users can see detailed information about a recipe.
* **Edit/Delete Buttons**: Only the owner of a recipe can see and use the Edit and Delete buttons in the detail view to modify or remove the recipe.

1. **Navigation bar**: Provides easy access to application functionalities based on your login status (logged in or guest).

Also, there will be seeded data for 3 recipes, that will always be loaded when you start the application.

### Instructions

A **folder named "tests"** is prepared for you. There is **a single file named "e2e.test.js"**. In it, you must write your front-end tests with the Playwright framework.

As for the execution of the tests, **you need to start the back-end server of the application and the HTTP server**. Everything is configured for you, so you need just **to execute three commands in two terminals**:

* "**npm install**" (or "**npm i** ") – to install the dependencies for your app.
* **"npm run server"** – to start the application back-end server.
* **"npm start" (in another terminal)** – to start the HTTP server.

**Note**: You will need to use **a third Terminal window for the execution of Playwright tests**. Use the command **"npm test"**.

### Front-End Testing with Playwright

You are provided with predefined configurations in the e2e.test.js file:

* Needed imports for Playwright:



* Predefined variables that you can use:

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

* Before and after test configurations:

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

* Test suits:

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

Use them and write the following e2e tests with Playwright for the "Recipe Sharing Hub" application.

**IMPORTANT: You are NOT allowed to delete nor add any test, use the tests exactly as they are provided.**

**NOTE: The tests are dependent on each other, so you should run them all together.**

#### Registration with Valid Data (Authentication Functionality)

1. Create a test scope.
2. **Go to** [**http://localhost:3000**](http://localhost:3000)
3. **Locate and click on the Register button**.
4. **Wait for the register form to load**.
5. **Create a unique email value**.
6. **Locate and fill the input field for email.**
7. **Locate and fill the input field for password**.
8. **Locate and fill the input field for confirm password**.
9. **Press the submit button**.
10. **Assert that you are redirected to the home page and the Logout button is visible**.

**Hint**: Use the predefined user object to hold and reuse user data.

#### Login with Valid Data (Authentication Functionality)

1. Create a test scope.
2. **Go to** [**http://localhost:3000**](http://localhost:3000)
3. **Locate and click on the Login button**.
4. **Wait for the login form to load**.
5. **Locate and fill the input field for email.**
6. **Locate and fill the input field for password**.
7. **Press the submit button**.
8. **Assert that you are redirected to the home page and the Logout button is visible**.

#### Logout from the Application (Authentication Functionality)

1. **Create a test scope**.
2. **Go to** [**http://localhost:3000**](http://localhost:3000)
3. **Log in to the application**.
4. **Click the Logout button**.
5. **Assert that the Login button is visible.**
6. **Assert that the URL is for home page.**

#### Navigation for Logged-In User Testing

1. Create a test scope.
2. **Go to** [**http://localhost:3000**](http://localhost:3000)
3. **Log in to the application**.
4. **Assert** **that "Home", "Discover", "Search", "Create Recipe" and "Logout" buttons are visible**, and **"Login" and "Register" buttons are hidden.**

#### Navigation for Guest User Testing

1. Create a test scope.
2. **Go to** [**http://localhost:3000**](http://localhost:3000)
3. **Assert that "Home", " Discover", "Search", "Login" and "Register" buttons are visible, and "Create Recipe" and "Logout" buttons are hidden.**

#### Create a Recipe Testing (CRUD Functionality)

1. Create a test scope.
2. **Go to** [**http://localhost:3000**](http://localhost:3000)
3. **Log in to the application**.
4. **Locate and click the "Create Recipe" button**.
5. **Wait for the create recipe form to load**.
6. **Generate random recipe name and save it in predefined variable.**
7. **Locate and fill the input field for name with random generated value**.
8. **Locate and fill the input field for recipe image.**
9. **Locate and fill the input field for preparation time.**
10. **Locate and fill the input field for shared by.**
11. **Locate and fill the input field for cuisine type.**
12. **Locate and fill the input field for steps**.
13. **Press the submit button**.
14. **Assert that the url is** [**http://localhost:3000**](http://localhost:3000)**/discover**.
15. **Assert that a recipe with the name you just added is present in the list**.

#### Edit a Recipe Testing (CRUD Functionality)

1. Create a test scope.
2. **Go to** [**http://localhost:3000**](http://localhost:3000)
3. **Log in to the application**.
4. **Locate and click on "Search" button**.
5. **Locate and fill the input field for search. (Search for the recipe that you created)**
6. **Locate and click on "Search" button for searching result.**
7. **Locate and click on the name of the recipe that you created from the results**.
8. **Locate and click on Edit button**.
9. **Wait for the Edit form to load**.
10. **Locate and fill the name field in the edit form with new value to edit a recipe.**
11. **Press the submit button**.
12. **Assert that the name’s value is as expected with** **edited value**.

#### Delete a Recipe Testing (CRUD Functionality)

1. Create a test scope.
2. **Go to** [**http://localhost:3000**](http://localhost:3000)
3. **Log in to the application**.
4. **Locate and click on "Search" button.**
5. **Locate and fill the input field for search. (Search for the recipe that you created)**
6. **Locate and click on "Search" button for searching result**.
7. **Locate and click on the name of the recipe that you created from the results**.
8. **Press the delete button**.
9. **Assert that the url is** [**http://localhost:3000**](http://localhost:3000)**/discover**.
10. **Assert that a recipe with the name you just deleted is NOT present in the list**.