(420-718-AB) CLIENT-SIDE WEB PROGRAMMING
WINTER 2024

ASSIGNMENT 2# [Worth 5%]

Due: 30th January 2024 at 9.00 AM

Objective

- To create a registration form with native data validation;
- To read values from an input form and manipulate the DOM using JavaScript;
- To create a functional user interface for an online food delivery service.

Context

You are a free-lance developer working on a contract for an online food delivery service called “BAN the dishes”. You are responsible for adding two features:

- The new user registration form,
- The menu interface that restaurants will use to manage dishes being sold.

The project will be built in phases and a designer has already prepared the HTML file and a **draft** style-sheet.

You are allowed to change the html and the css file (see appendix A for information on the style-sheet). The designer is friendly and encourages input from developers.

Part 1: New User Registration Form

The new user registration form is accessible from the home page. This interface is intended for new users who want to register with the service.

The form must include the following fields:

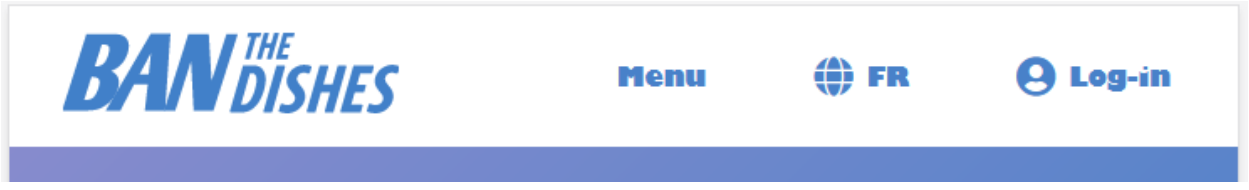
- First name (with an id of *first)*
- Last name (with an id of *second)*
- Email (with an id of *email)*
- Password (with an id of *pwd*, a min size of 8 and a max size of 12 characters)
- City (with an id of *city*)
- Province (with an id of *prov*, using a drop down list with all canadian provinces)

Requirements:

1. All fields are required and their input type should be set in order to provide native HTML data validation and reduce user error.
2. Include labels for all inputs.

Part 2: Login button

The second feature to be implemented is the Login button in the Menu page. This interface is intended for restaurant owners and managers who wish to login prior to updating the meals and prices of their restaurant's page.



1. Once the user clicks on the Log-in button the following should take place:
    - The “Log-in” text should change to “Log-out”;
    - The ‘*new meal information’* form should appear at the bottom of the screen.

The ‘*new meal information’* form is already present in the menu.html file (see * * around line 75), however, it is initially hidden (*display: none)*.

2. Once logged in, if the user clicks on the button “Log-out”, the following should happen:
    - The “Log-out” text should change to “Log-in” ;
    - The ‘*new meal information’* form should disappear at the bottom of the page.

3. You **must use an event listener** with the log-in button in order to listen for clicks.

4. All JavaScript files must be located in the **folder named “src”**.

Part 3: Removing Existing Cards

A meal description must be deleted via the delete button on the card.

Samples of the delete button are already present in the html file (see inside each *.card),* however, they are initially hidden (*display: none)*.

Requirements:

1. This functionality **must be implemented using event listeners.**
2. The delete buttons should only appear while the user is “logged in” and be hidden otherwise.
3. Clicking on the delete button or on the button’s icon should delete the cards.
4. Deleted cards should be removed from the DOM (rather than hidden).

Part 4: Adding New Cards

Restaurant administrators need to add new meals to their restaurant profile. This is done via the ‘*new meal information’* form.

1. In order for a new meal to be added, the conditions below must be met, otherwise, the form should display a message explaining what is wrong:
    - All the fields are required;
    - The price field must include a number with up to two decimal places;
    - The meal description must be less than 200 characters.
2. A new meal description is added once the + (plus) button is pressed.
3. The user should be able to add new cards even if there are no cards left in the page.
4. Placeholder text should be included in the form fields in order to guide the user on what kind of information is expected in which field.
5. Cards newly added via the ‘*new meal information’* form also need to have a functional delete button.

IMPORTANT: Submission guidelines

1. Submit a single zip folder with all files as per files folder as well as screenshots of your webpages.
2. Only the last submission will be evaluated.
3. **Failure to follow these guidelines will result in losing points**
4. Plagiarism and cheating will be strongly penalized as per college policies.
5. Do not submit any work via MIO or TEAMS.

Appendix A: Style-sheet organization

You are free to change the default styling found in the CSS file. The style-sheet is mostly organized in the order that elements appear in the page. A few notes:

CSS Variables

In order to keep the style consistent and avoid repetition, the style-sheet uses CSS variables for colors. You can make any color changes via a single variable and it will affect the entire project. You can learn more about CSS variables here.

Variables in CSS are declared within a scope. For a global scope you can use either the :root or the body selector. The variable name must begin with two dashes (--). For example:

```
:root {  --header-txt-color: #4881c9;  }
```

To use the variable, replace the css property value by **var(*name of variable*)**. For example:

```
.right-box {   color: var(--header-txt-color);  }
```

Utility Classes

The section named “utility classes” includes styling that repeats for elements in different locations. For example, buttons in the main section and in the input form should look the same, so they all receive the class of ‘*.btn*’ .