FLOWMASTERY DESCRIPTION:

FlowMastery is a web application that offers a workflow technique, provides organisational and common tools.

Its main function is the pomodoro timer. This timer is meant to be used as a workflow technique to enhance productivity. It works in time/break intervals by setting a time for a focused work session and prompting for a break once the work timer runs out.

The timer incorporates a break timer to encourage the user not procrastinate.

When the break timer runs out, it's time to get back to work.

Along side the pomodoro timer's main feature, FlowMastery also has organisational and common tools to help the user remain organized, minimize cross app utilization and on top of his tasks and deadlines.

The website contains 8 Javascipt files, each one responsible for a specific functionality.

UX FUNCTIONALITY GUIDELINE

UX DEV : Mohammed El Ghali

1. Calculator : calc.js

It's a basic calculator code :

- Addition, subtraction, multiplication, and division operations using standard arithmetic symbols.
- Clearing the calculator display by pressing the "C" button.
- Deleting the last character entered on the calculator display using the backspace button.
- Evaluating the mathematical expression entered on the calculator display by pressing the "=" button.

2. Dark Mode toggler : darkMode.js

Switches between light and dark themes on the web page:

- An event listener is then added to the darkModeBtn button element that listens for a "click" event.
- If the current theme is "light", the theme is switched to "dark" by setting the theme data attribute of the documentElement to "dark". Otherwise, if the current theme is "dark", it is switched to "light" by setting the theme data attribute to "light".
- The body element is toggled between light and dark modes by adding or removing the "dark-mode" class, which is retrieved using CSS.
- Finally, the text content of the darkModeBtn is updated to display "Light mode" or "Dark mode" depending on whether the body element currently has the "dark-mode" class.

3. Deadline list : deadline.js

Deadline list functionality that allows the user to add a new deadline by entering the name then pressing the Enter key:

- adds an event listener to it that listens for a 'keypress' event, When the user presses the Enter key, the addDeadline() function is called.
- When the addDeadline() function is called, it checks if the input value is empty. If it is, an alert message is displayed to the user, asking them to enter a deadline.
- If the input value is not empty, It selects the element with class "deadline_tasks" and appends a new task element to it using the innerHTML property. The task element contains the task name and a delete button.
- When a delete button is clicked, it removes the corresponding task from the list using the remove() method.

4. Tools handler: handleTools.js

Provides a simple way to switch between different tools on a page based on user input.

- This code defines **a function handleTools** that takes a parameter btn representing the button that was clicked by the user.
- Next, the code uses a series of if/else statements to determine which tool should be displayed based on the button clicked. If the button is "calc", it removes the "hidden" class from the $calc element and adds the "hidden" class to the $note and $feedback elements, effectively displaying the calculator tool and hiding the other two tools. Similarly, if the button is "notes" or "feedback", the code displays the corresponding tool and hides the other two tools.

5. Notepad.js

provides a simple way to allow users to download the contents of a notepad element as a file.

- Defines an event listener for a download button with the ID "download". When the button is clicked, the code retrieves the contents of a notepad element with the ID "notepad" using the value property.
- Next, the code creates a new Blob object containing the notepad data with the MIME type set to "text/plain". This creates a file that can be downloaded by the user.

6. Pomodoro Timer :timer.js

provides a basic implementation of a timer that can be customized by users.

- This code creates a timer that counts down from a set number of minutes and seconds. Users can choose the countdown time by clicking on preset buttons (25,5 or 60 minutes) or by submitting a custom time using a form.
- The code also includes event listeners to handle user interaction with the preset buttons and the form. It uses clearInterval() to stop the timer when the stop or reset button is clicked.

Functions :

- start - This function starts the timer when the 'Start' button is clicked. It first checks if the timer is already running, and if not, it sets an interval to call the timer function every second.

- stopInterval - This function stops the timer when the 'Stop' button is clicked. It clears the interval set by the start function.
- reset - This function resets the timer to the default value of 25 minutes when the 'Reset' button is clicked. It also resets the counter to 0 and stops the timer.
- timer25, timer5, and timer60 - These functions set the timer to a pre-defined value of 25, 5, and 60 minutes, respectively, when the corresponding buttons are clicked.
- document.customForm - This is an event listener for the 'Submit' button on the custom form. It prevents the default form submission behavior and sets the timer to the value entered in the form.
- timer - This function is called every second when the timer is running. It decrements the seconds and minutes remaining, and if the timer reaches 0, it increments the counter and resets the timer to the default value of 25 minutes.

7. To do list : todo.js

simple todo list application that allows users to add and delete tasks. It uses localStorage to store the list of tasks so that the user can see their list even if they close the browser.

- The tasks array is used to store the list of tasks, and the localStorage API is used to store the list of tasks in the browser's localStorage. When the page loads, the code checks if there are any tasks stored in localStorage and if there are, it loads them into the tasks array and displays them on the screen.
- The createDiv function is used to create a new task element with a unique id based on the task's index in the tasks array. The addTask function is called when the user presses the "Enter" key in the input field or clicks the "Add" button. It creates a new task element, adds it to the screen, adds it to the tasks array, and saves the tasks array to localStorage.
- The delTask function is called when the user clicks the delete button for a task. It finds the index of the task in the tasks array, removes it from the array, saves the tasks array to localStorage, and removes the task element from the screen.
- The localStorage API is used to store and retrieve data in the browser's localStorage. The setItem method is used to store data with a key and value, and the getItem method is used to retrieve data by its key. The JSON.stringify method is used to convert the tasks array to a JSON string before storing it in localStorage, and the JSON.parse method is used to convert the JSON string back to an array when retrieving it from localStorage.

8. Weather API : weather.js

Uses the OpenWeatherMap API to fetch and display the current weather for Ottawa in Celsius. It defines an object weather with two methods:

- fetchweather() - This method uses the fetch() function to retrieve weather data from the OpenWeatherMap API. It then calls the displayWeather() method to display the data on the webpage.
- displayWeather(data) - This method takes the weather data as a parameter and extracts the temperature information from it. It then updates the temperature displayed on the webpage.
- The apikey property of the weather object contains the API key that is required to make requests to the OpenWeatherMap API. The fetch() function is used to make a GET request to the API endpoint, passing in the API key and other query parameters. The response from the API is in JSON format, so the response.json() method is called to parse the response into a JavaScript object. The data object is then passed to the displayWeather() method to extract and display the temperature information.