# Lab 1 FullStack Report

## Challenges:

1. **updateRecipe(id):** The updateRecipe(id) function faced challenges in dynamically updating the HTML table cells with textareas for editing recipe information. The difficulties included:

- Proper replacement of text content with textareas.
- Handling save and cancel actions effectively.

2. **createButton(text, onClick, id):** Creating dynamic buttons with specific text, onclick events, and IDs posed challenges such as:

- Ensuring proper attachment of onclick event handlers.
- Generating unique IDs for each button instance.

3. **Middleware**: Implementing Middleware for parsing JSON and URL-encoded data, as well as serving static files, presented the following challenges:

- Setting up middleware functions correctly in the Express.js application.
- Handling errors and exceptions within the middleware effectively.

## Solutions:

1. **updateRecipe**(id): To address the challenges with updateRecipe(id), the following solutions were implemented:

- Utilized querySelectorAll to select all table cells and replaced their content with dynamically generated textareas.
- Implemented functions to handle save and cancel actions separately, ensuring proper execution based on user interactions.

2. **createButton**(text, onClick, id): The challenges with createButton(text, onClick, id) were overcome with the following solutions:

- Created a reusable function to generate button elements with specified text, onclick events, and unique IDs.
- Ensured proper attachment of onclick event handlers to each button instance by utilizing the onclick property.

3. **Middleware**: To address challenges related to Middleware, the following solutions were implemented:

- Configured Express.js application to use middleware functions for parsing JSON and URL-encoded data using express.json() and express.urlencoded() respectively.

## How I would approach the problem differently

1- **updateRecipe(id)**:

Instead of dynamically updating HTML table cells with textareas directly, I would consider using a frontend framework like React or Vue.js. These frameworks provide efficient ways to manage dynamic UI updates and state changes.

By using a frontend framework, you can separate concerns more effectively, making it easier to handle save and cancel actions without directly manipulating the DOM.

Implementing a state management solution such as Redux (for React) or Vuex (for Vue.js) could simplify handling user interactions and updating the UI accordingly.

**2- createButton(text, onClick, id):**

Rather than manually managing onclick events and generating unique IDs, I would utilize the capabilities of modern JavaScript frameworks/libraries.

Frameworks like React or Vue.js offer components that handle event binding and unique ID generation automatically, reducing the likelihood of errors and simplifying code maintenance.

These frameworks also provide reusable component patterns, allowing you to define a button component once and reuse it throughout your application with ease.

**3- Middleware:**

For middleware implementation, I would stick with Express.js as it's a popular choice for building backend applications in Node.js.

To handle errors and exceptions within middleware effectively, I would implement robust error handling middleware using Express's error handling capabilities.

Additionally, I would explore using more specialized middleware packages for specific tasks (e.g., body parsing middleware like body-parser) to simplify middleware setup and reduce the likelihood of errors.