Sure, let's create a simple personal project: a "**To-Do List**" web application. This project will help you practice using your knowledge of HTML, CSS, React, and possibly even some basic back-end functionality. Here are the details:

**\*\*Project: To-Do List Web Application\*\***

**\*\*Description:\*\***

Create a web application that allows users to create, manage, and complete tasks on a to-do list. The application will provide a clean and intuitive user interface for adding, editing, and marking tasks as complete.

**\*\*Technologies:\*\***

- Front-End: HTML, CSS, React

- Back-End (optional for storing tasks): Node.js, Express.js, MongoDB

**\*\*Features:\*\***

1. \*\*Task List Display:\*\* Display a list of tasks with their titles, due dates (if any), and completion status.

2. \*\*Add Task:\*\* Allow users to add new tasks to the list, including a title and an optional due date.

3. \*\*Edit Task:\*\* Enable users to edit existing tasks, including updating the title and due date.

4. \*\*Complete Task:\*\* Allow users to mark tasks as completed and change their status.

5. \*\*Delete Task:\*\* Provide an option to remove tasks from the list.

6. \*\*Task Filtering:\*\* Implement filtering options to view all tasks, completed tasks, or pending tasks.

7. \*\*Responsive Design:\*\* Ensure the application works well on different devices, such as mobile phones and desktops.

**\*\*User Interface:\*\***

- The main page will display the list of tasks.

- Each task will have an "Edit," "Complete," and "Delete" button.

- Users can click on a task to edit its details inline.

- There will be a form at the top of the page to add new tasks.

**\*\*Development Steps:\*\***

1. Set up your project: Create a new React project using tools like Create React App.

2. Design the user interface: Create HTML and CSS files to build the UI for your to-do list.

3. Create a task component: Design a component that represents a single task in the list.

4. Implement task list display: Create a component that displays a list of tasks using the task component.

5. Add task functionality: Implement the ability to add new tasks through a form.

6. Edit task functionality: Enable editing of task details in place.

7. Complete task functionality: Add the ability to mark tasks as completed.

8. Delete task functionality: Implement the option to delete tasks.

9. Task filtering: Create buttons or tabs to switch between different task lists based on their status.

10. Responsive design: Ensure your application looks good and functions well on various devices.

**\*\*Optional Back-End Steps:\*\***

1. If you want to store tasks persistently, create a Node.js and Express.js back-end.

2. Set up a MongoDB database to store tasks.

3. Implement API endpoints for adding, updating, deleting, and retrieving tasks.

4. Integrate the back-end with your front-end React application using API calls.

**\*\*Note:\*\***

This project can be as simple or as complex as you want it to be. Feel free to add more features, enhance the user interface, or explore additional technologies as you become more comfortable with the development process.

Remember that the primary goal of this project is to gain hands-on experience and apply your knowledge. As you work on this project, you'll not only improve your technical skills but also develop problem-solving abilities and gain confidence in building web applications.