

Software Specification

Upon application's start, user is met with the **main-panel** page, which shows **ALL** saved tasks. User is able to scroll through tasks vertically using the scroll wheel, and by clicking the **task**, the user gets redirected to that task's **detailed-page**.

On the **main-panel**, user is also able to sort and create new task's using the available **options** at the top of the screen.

Options:

- All : Display every task, including finished tasks,
 - Done : Display completed tasks only,
 - ToDo : Display unfinished tasks only, and
 - NewTask : Upon it's click action, user is redirected to a **compose-task** page.
-

Compose-task:

The purpose of this page is to provide a form for the **task's creation**. The form includes two attributes:

- Name : Attribute is restricted to 30 characters, and
- Details : Attribute is restricted to 500 characters.

Upon submitting the form, a task is created only once all fields are populated with adequate characters length as specified above. Fields must not be empty for successful submission. Additionally, a new task's "Date of Creation" is passed to the **detailed-page** with the user, and newly created task is saved in a local storage.

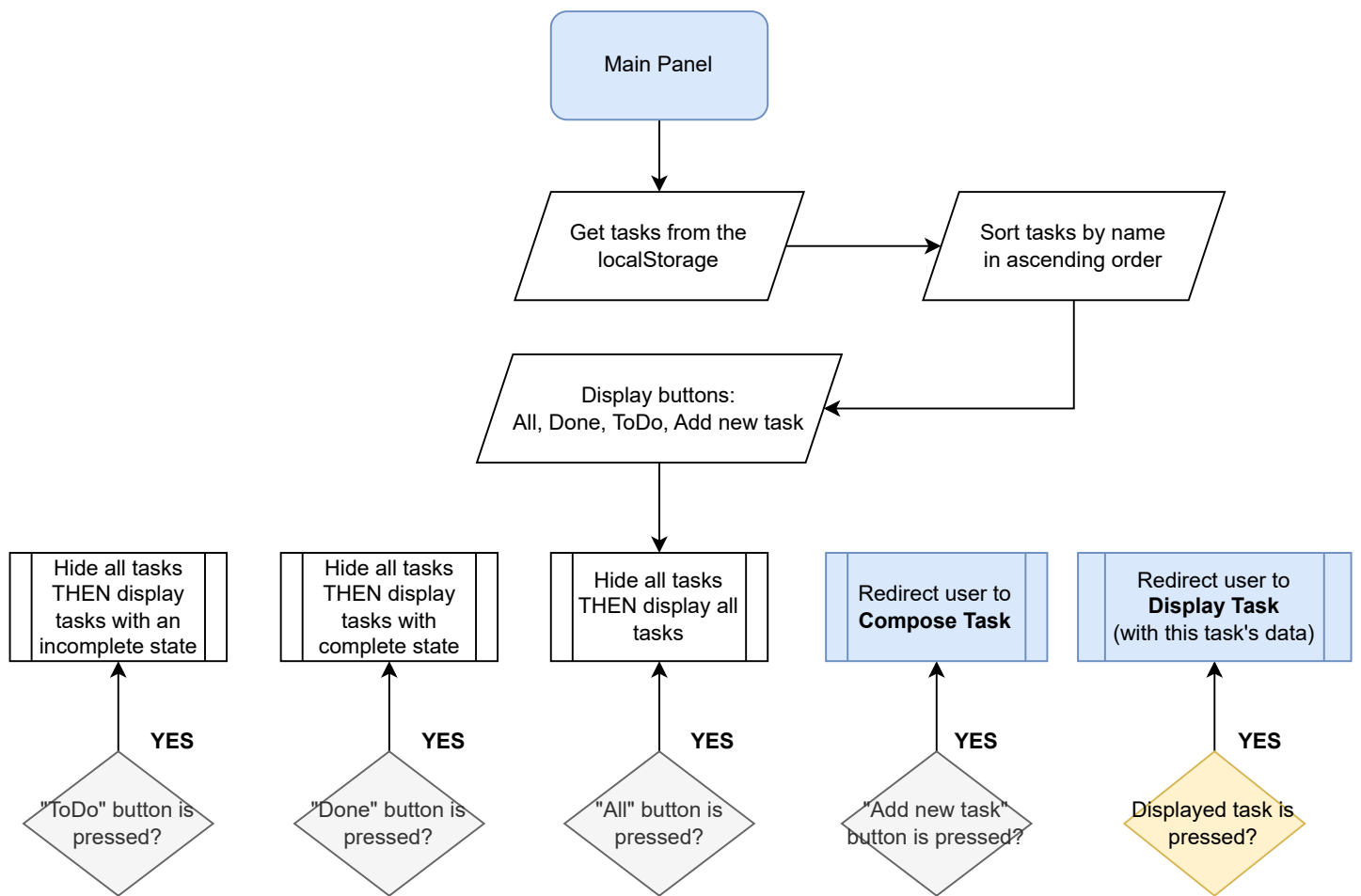
Detailed-page:

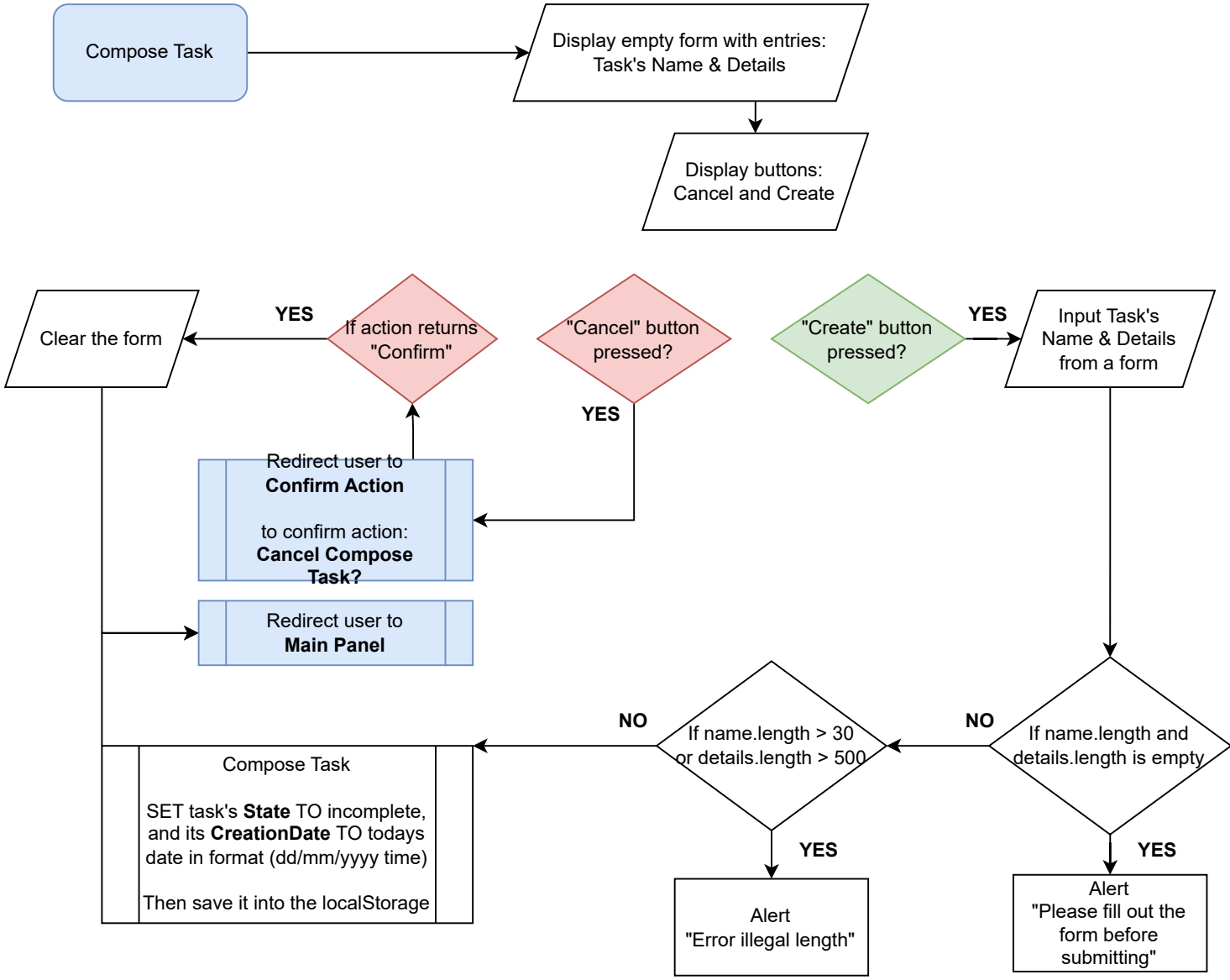
The purpose of this page is to display task's additionally specified information, like task's "Name", "Detail" and "Date of Creation", and available **manage-options** in order to either delete the task or change it's state.

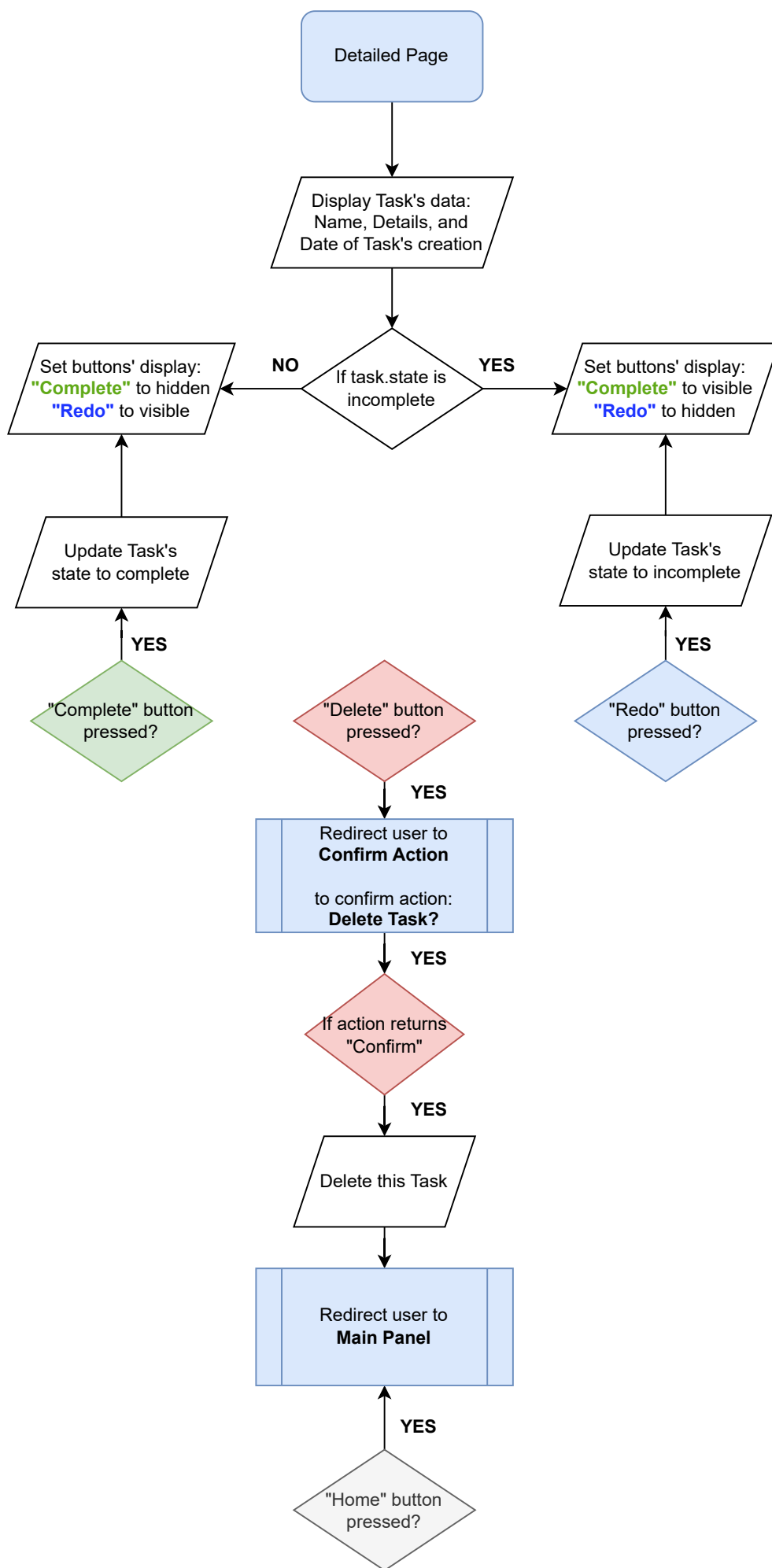
Manage options:

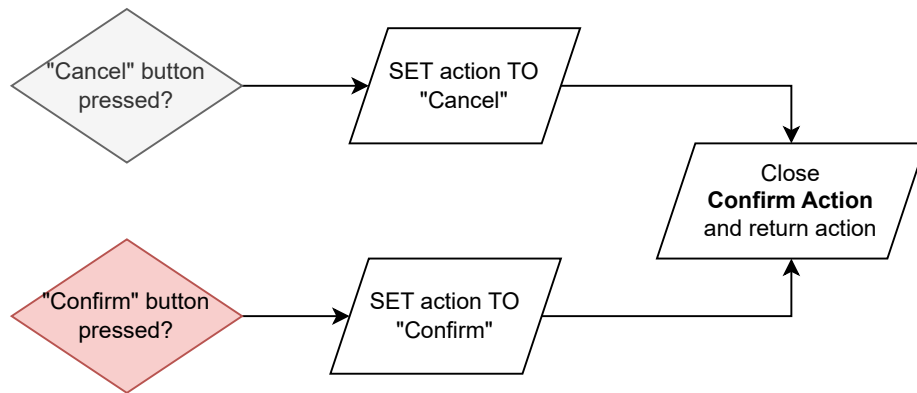
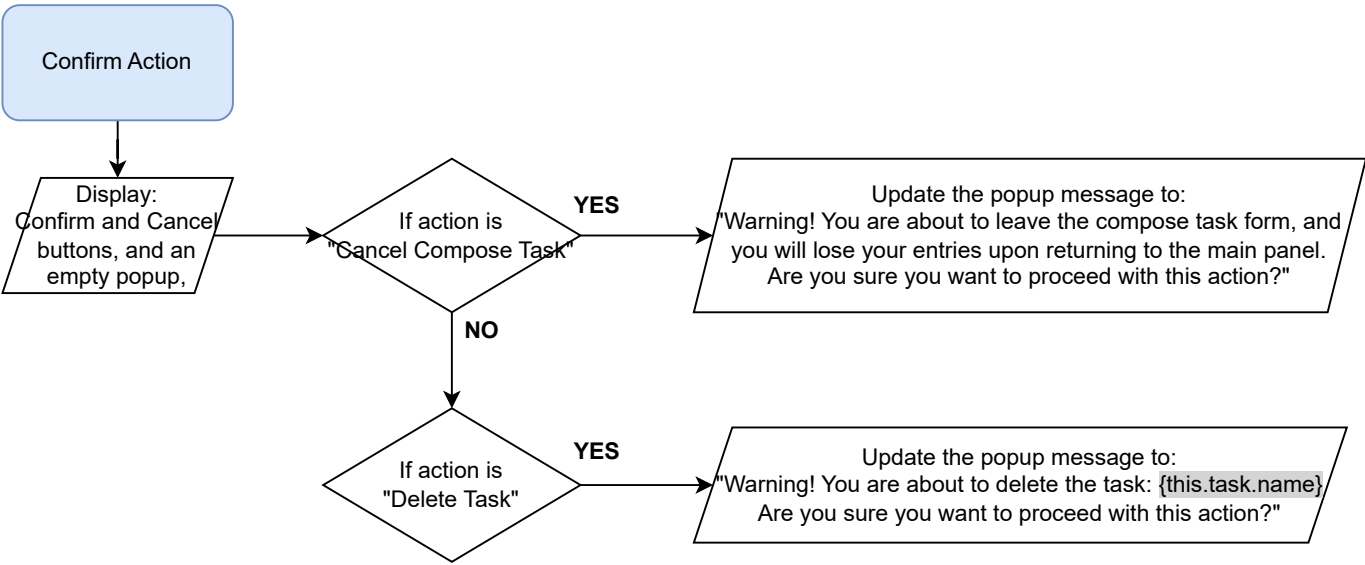
- Complete : Update task's state to **Done** (finished),
 - Redo : Update task's state to **ToDo** (unfinished), and
 - Delete : Redirect user to **confirm-action** page.
-

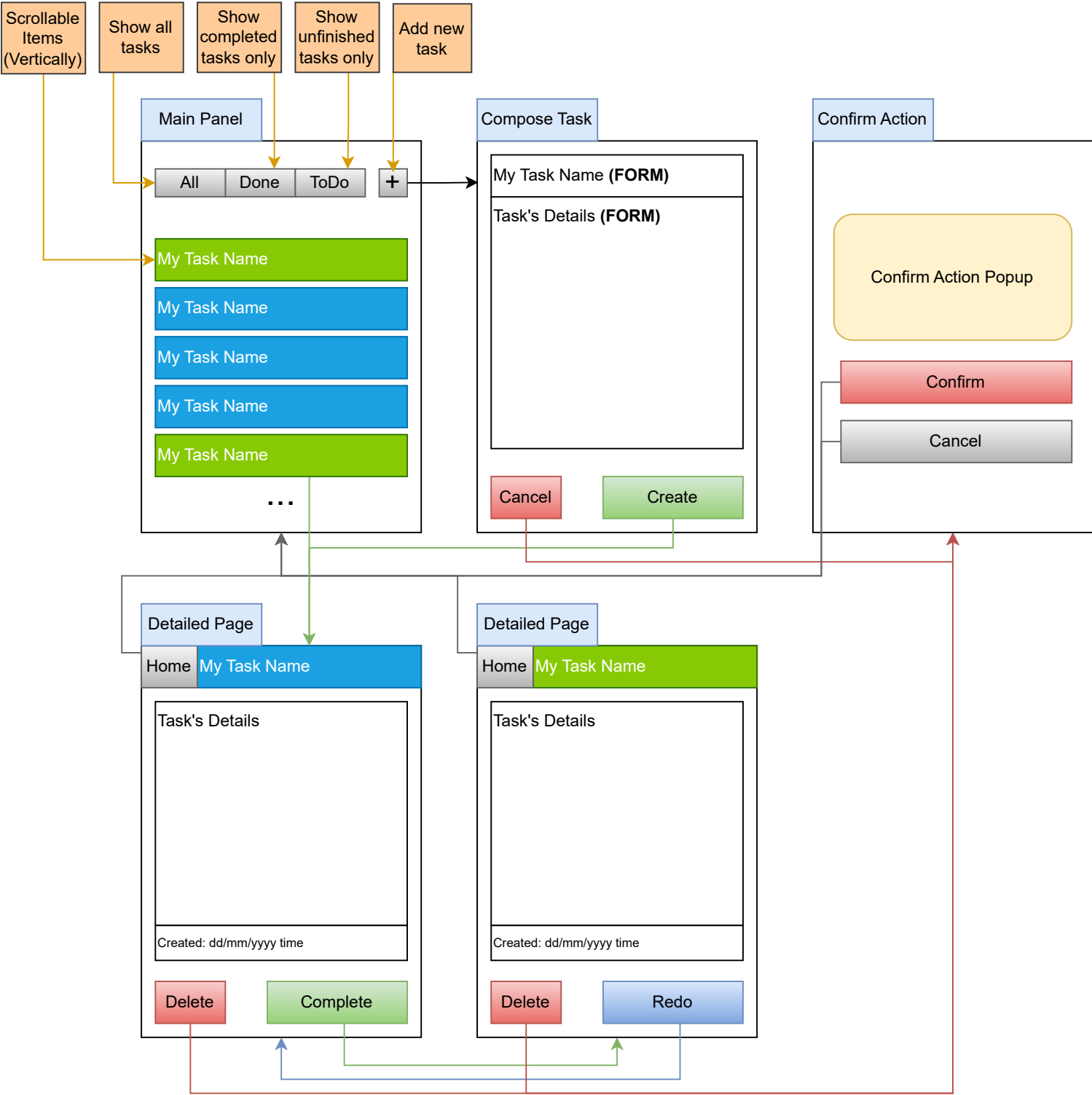
Confirm-action's page purpose is to prompt the user to verify their intent. It is a safety precaution in case user triggers a dangerous action by accident that could have negatively impact the user's experience.











Pre-Implementation Decisions

Table of Contents

1 Technology Selection

1.1 **React**

Chosen for its familiarity and the opportunity to enhance React-specific skills.

1.2 **JavaScript**

Selected for its popularity, essential features, and wide range of applications.

1.3 **TypeScript**

Integrated with React for real-time validation and improved data flow tracking.

1.4 **Vite**

Included to compile scripts, reduce production size, and enable real-time changes.

1.5 **Bootstrap5**

Added for its pre-styled components, simplifying the styling process.

2 Project Flexibility and Maintainability

2.1 **React's Flexibility**

Highlighted the ability to integrate React into different frameworks and environments.

2.2 **Browser Rendering and Styling**

Utilizing browsers for rendering JSX and taking advantage of CSS3 for styling.

2.3 **Effective Project Management**

Using React's component-based architecture and splitting pages for efficient collaboration.

2.4 **Delivering the Project**

Ensuring flexibility and maintainability to meet client needs successfully.

1. Technology Selection

I decided to develop the project in React because of my relative experience from past projects and confidence in using JavaScript for the chosen library. This provides me with an ideal opportunity to polish my use case of React features, such as hooks like `useState()`. Additionally, this project will allow me to become more comfortable with JavaScript's modular programming, including imports and exports, and gain additional experience in deconstructing objects. These improved skills will be valuable for future projects.

1.1 React

React is a popular high-level JavaScript library that offers a wide range of features and benefits. It is well-known in the industry and its inclusion in various frameworks is easy to implement. For example, we can integrate our React project into Electron to create our final product as a desktop application. If the client decides to host the application online instead, we can integrate our React project with frameworks like Django. This flexibility allows us to adapt to different project requirements.

1.2 JavaScript

I chose JavaScript as it is a popular and easy-to-understand high-level language. It provides essential and fundamental features such as Object-Oriented Programming, Functional Modular and Procedural programming support, objects, arrays, booleans, spread and deconstructing, logic operators, ternary operators, conditional and unconditional loops, asynchronous and synchronous functions, returns, switch statements, access to DOM objects, DOM event-handlers, observers, global and local scopes, recursion support, math modules, and regexp support. JavaScript has various applications in different programming domains, from frameworks like Next.js and hosting servers using Node.js to managing APIs. The wide adoption of JavaScript also means there are plenty of frameworks built on top of it, offering even more options for our project.

1.3 TypeScript

To ensure timely project delivery and reduce error cases, I decided to integrate TypeScript with React. TypeScript provides real-time validation in IDE, enhancing the development process. The addition of strongly typed decorators allows me to track and understand the project's data flow, making it easier to identify and prevent potential issues.

1.4 Vite

I chose to include Vite with React to improve production efficiency. Vite compiles our scripts into a bundle, reducing the overall production size and increasing the project's performance. It also allows us to see real-time changes without losing the application's states, which further speeds up the development process.

1.5 Bootstrap5

To simplify the styling process and increase productivity, I decided to include the Bootstrap5 library. By leveraging the pre-styled components provided by Bootstrap5, we can avoid the need for manual CSS styling. This allows developers to choose appropriate styles from the library, reducing the time and effort spent on design.

2. Project Flexibility and Maintainability

The chosen technologies are deliberately selected to promote the project's flexibility and maintainability.

2.1 React's Flexibility

React's wide industry adoption and its integration capabilities with other frameworks contribute to the project's flexibility. We can easily integrate React into different environments and frameworks, such as Electron and Django, to meet the specific requirements of our clients.

2.2 Browser Rendering and Styling

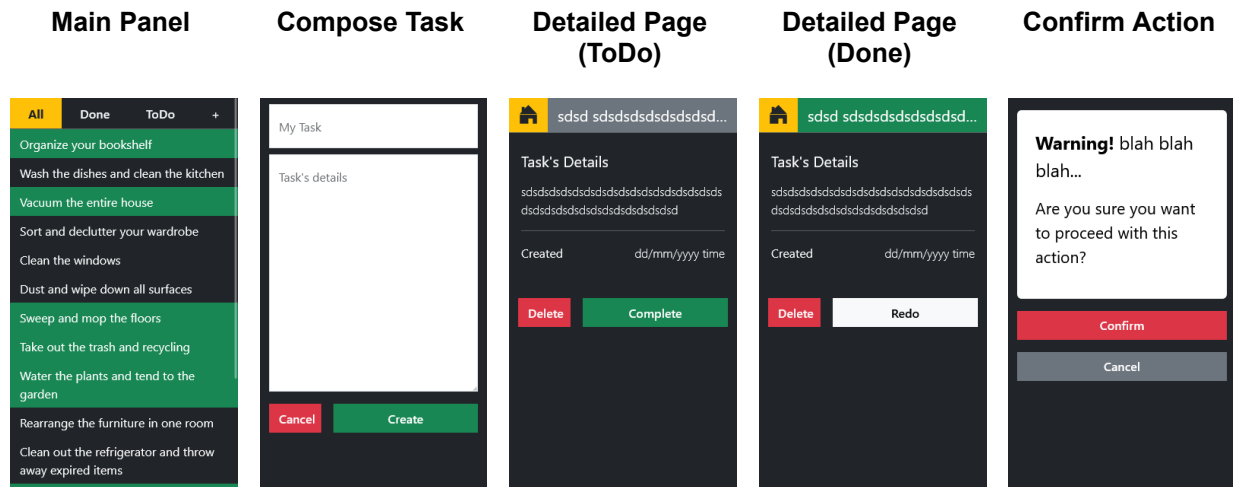
React requires a browser to render JSX, which is a component based on JavaScript and HTML5. This allows us to utilize the powerful styling capabilities of CSS3, making it ideal for front-end web development. With my experience in responsive design using HTML5 and CSS3, we can efficiently implement design improvements and create applications that are adaptive to different screen sizes.

2.3 Effective Project Management

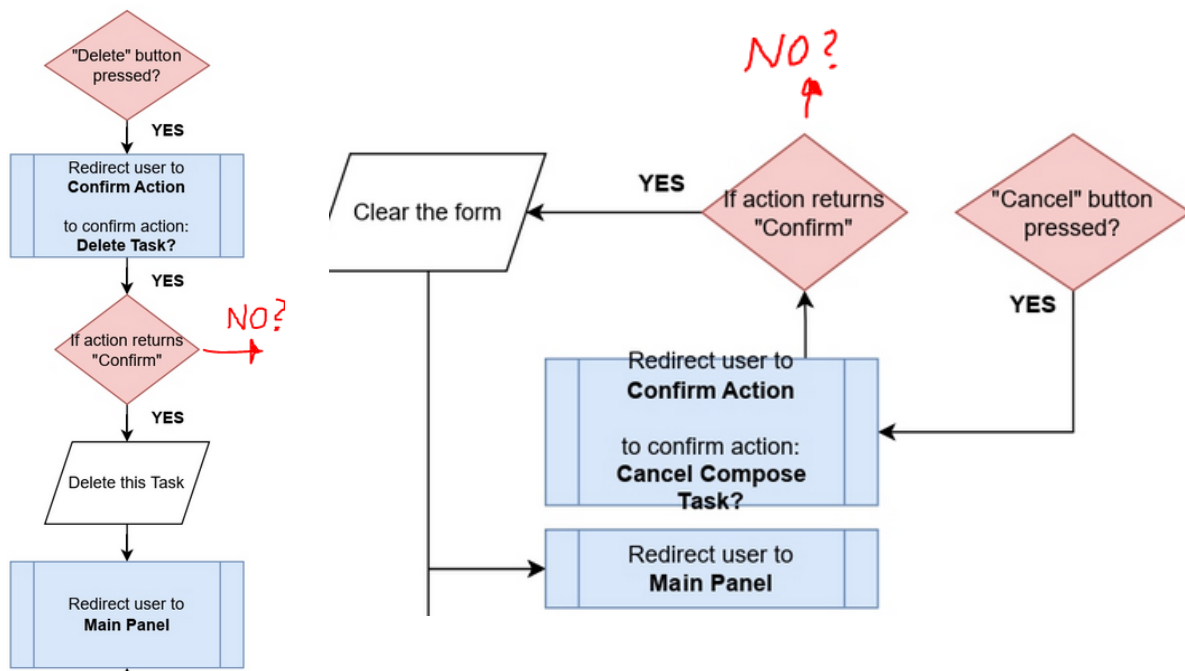
By hosting applications locally, React enables us to test our application on multiple devices with distinct display sizes without the need

Prototype Implementation Decisions

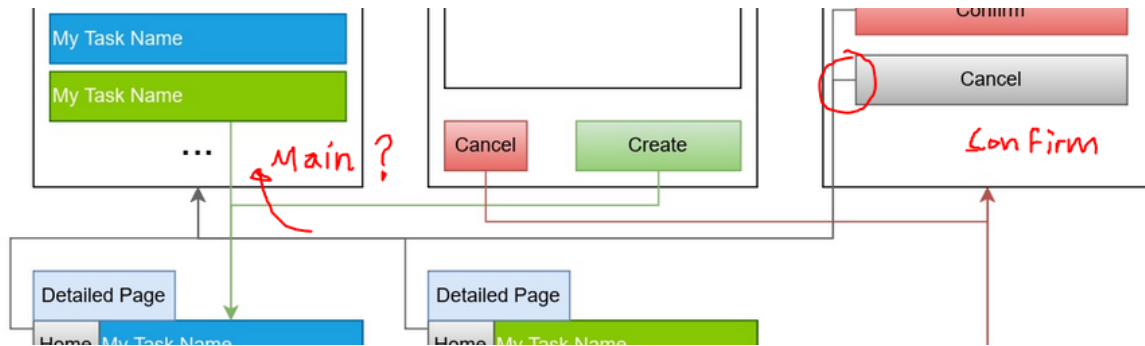
Implemented Design: (27/06/2023)



We have implemented an appropriate design to meet the specified software-development-process documentation requirements. However, we require an iteration of some flow charts related to redirecting the user to the "Confirm Action" page because there is a lack of an exit point after confirming the action's cancellation and questionable redirection to the "Main" page upon cancellation at "Confirm Action" wire-frame.



It is expected that the user is redirected back to their progress upon cancelling critical action such as post form from a standard UI procedure. In our case, user composing a task can cancel their progress, which prompts them to confirm their intent on the "Confirm Action" page. Hence, upon cancellation of critical action, the user should be redirected back to continue composing their task, which is not the case according to our wireframes documentation.



Additionally, the prompting to "Confirm Action" page deletes the user's progress. There are a couple of solutions that can help us solve this issue, mainly local storage and cookies. In our case, cookies can be beneficial as they contain an expiry date. However, saving data before prompting to the "Confirm Action" page is not specified.

In short, iteration is required to fix the application's navigation and data flow in terms of temporary savings as we implement critical functionality for the final product.

