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:

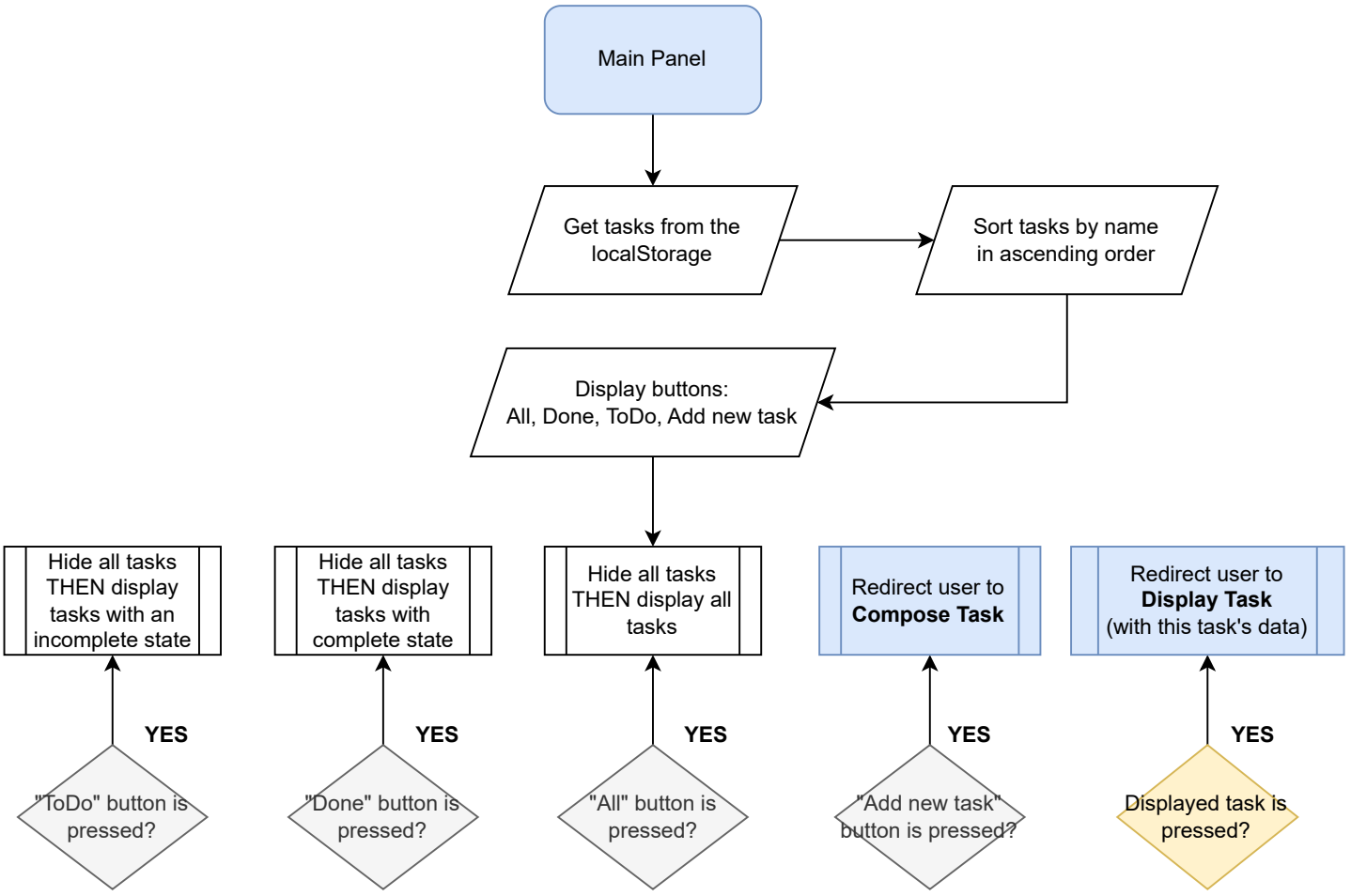
- Name : Attribute is restricted to 30 characters, and
- Details : Attribute is restricted to 500 characters.

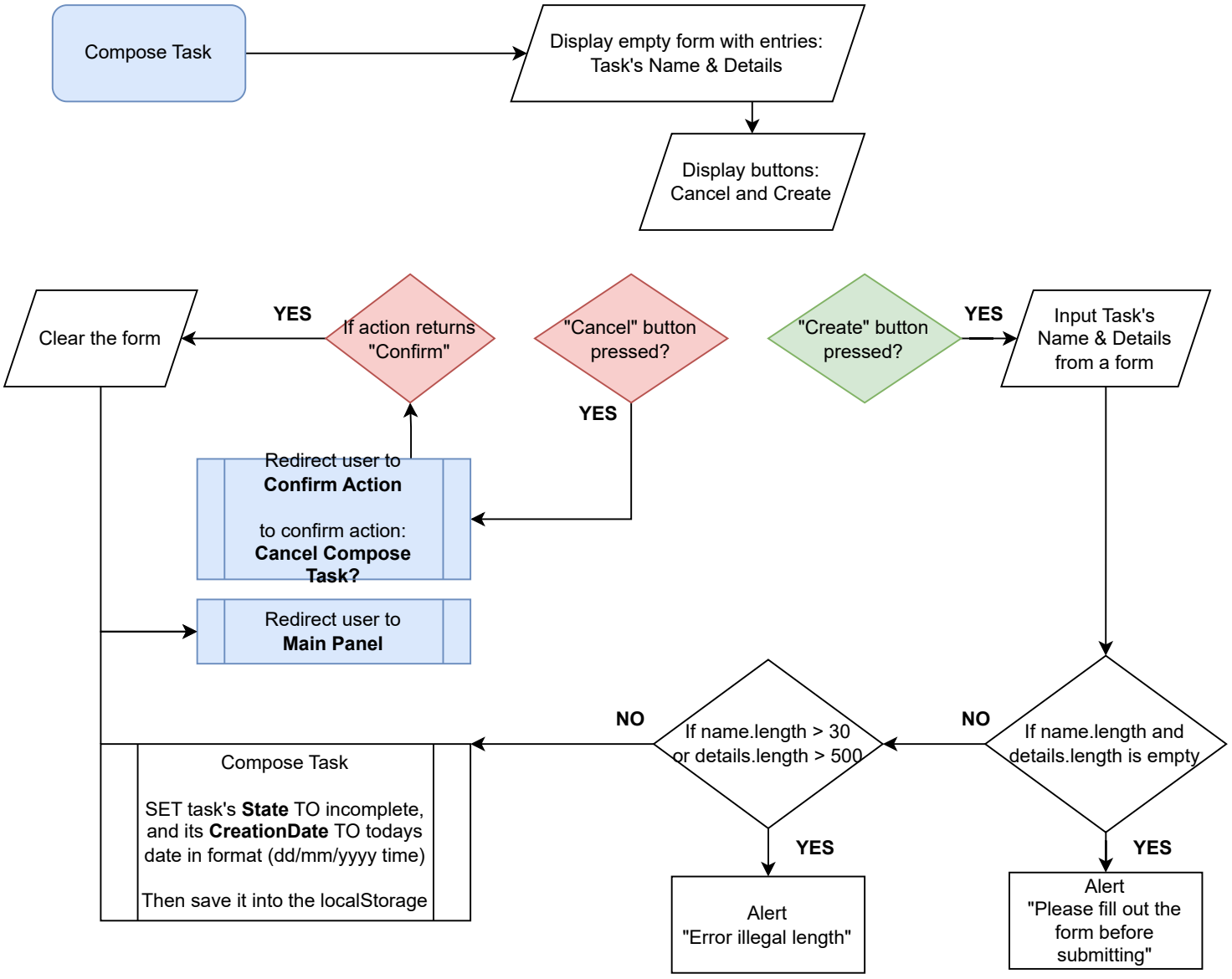
Upon submitting the form, a task is created only once all fields are populated with adequate characters length as specified above. Fields must not be empty for successful submission. Additionally, a new task's "Date of Creation" is passed to the **detailed-page** with the user, and newly created task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

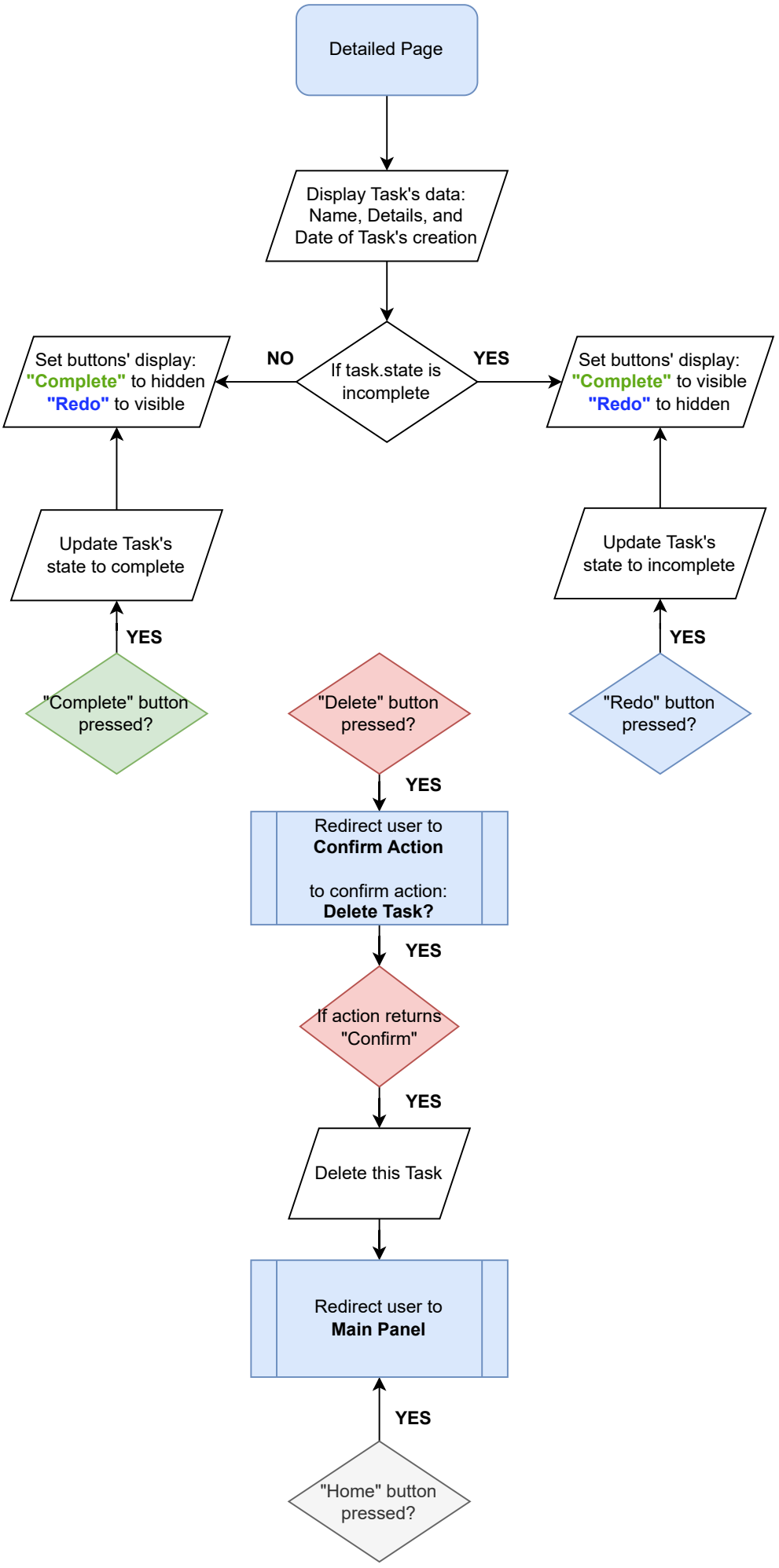
Detailed-page:
The purpose of this page is to display task's additionally specified information, like task's "Name", "Detail" and "Date of Creation", and available **manage-options** in order to either delete the task or change it's state.

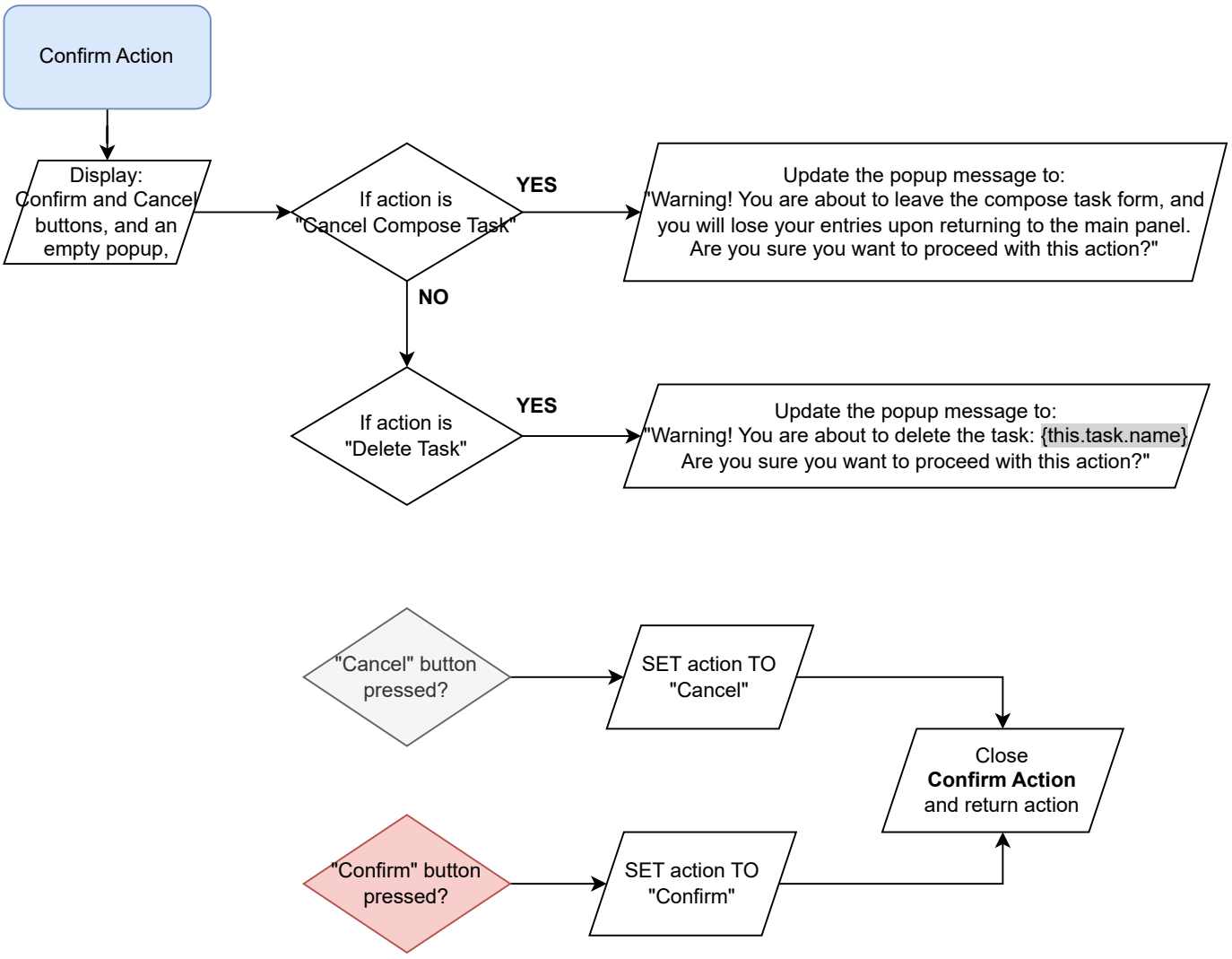
Manage options:
- Complete : Update task's state to **Done** (finished),
- Redo : Update task's state to **ToDo** (unfinished), and
- Delete : Redirect user to **confirm-action** page.

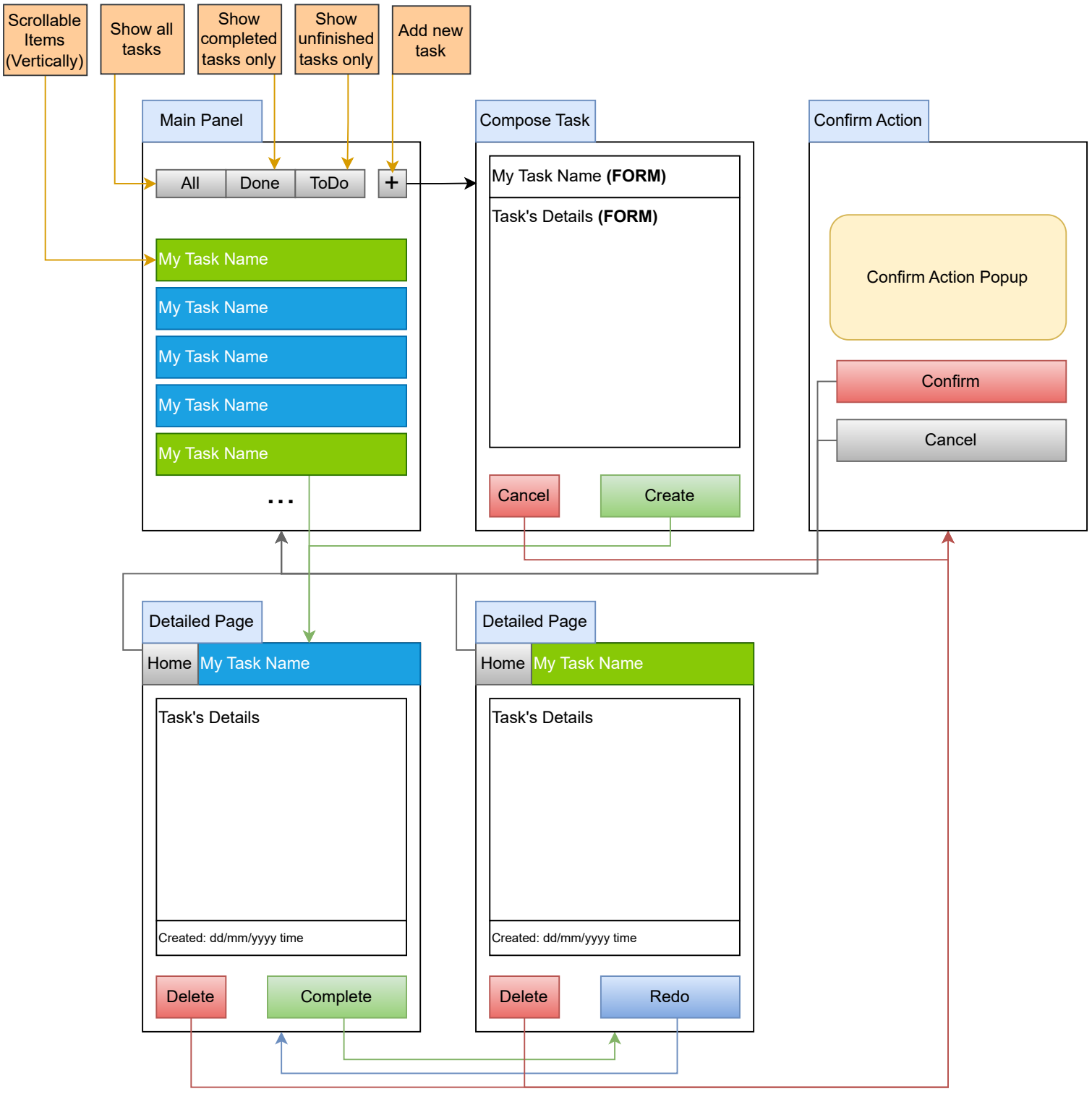
Confirm-action's page purpose is to prompt the user to verify their intent. It is a safety precaution in case user triggers a dangerous action by accident that could have negatively impact the user's experience.


Main Panel
Get tasks from the localStorage
Sort tasks by name in ascending order
Display buttons: All, Done, ToDo, Add new task
Hide all tasks THEN display tasks with an incomplete state
Hide all tasks THEN display tasks with complete state
Hide all tasks THEN display all tasks
Redirect user to **Compose Task**
Redirect user to **Display Task** (with this task's data)
YES
"ToDo" button is pressed?
YES
"Done" button is pressed?
YES
"All" button is pressed?
YES
"Add new task" button is pressed?
YES
Displayed task is pressed?


Compose Task
Display empty form with entries:
Task's Name & Details
Display buttons:
Cancel and Create
"Create" button pressed?
YES
Input Task's Name & Details from a form
"Cancel" button pressed?
YES
If action returns "Confirm"
YES
Clear the form
Redirect user to
Confirm Action
to confirm action:
Cancel Compose Task?
Redirect user to
Main Panel
If name.length and details.length is empty
NO
YES
Alert
"Please fill out the form before submitting"
If name.length > 30 or details.length > 500
NO
YES
Alert
"Error illegal length"
Compose Task
SET task's State TO incomplete, and its CreationDate TO todays date in format (dd/mm/yyyy time)
Then save it into the localStorage


Detailed Page
Display Task's data:
Name, Details, and
Date of Task's creation
If task.state is
incomplete
NO
YES
Set buttons' display:
"Complete" to hidden
"Redo" to visible
Set buttons' display:
"Complete" to visible
"Redo" to hidden
Update Task's
state to complete
Update Task's
state to incomplete
YES
YES
"Complete" button
pressed?
"Delete" button
pressed?
"Redo" button
pressed?
YES
Redirect user to
Confirm Action
to confirm action:
Delete Task?
YES
If action returns
"Confirm"
YES
Delete this Task
Redirect user to
Main Panel
YES
"Home" button
pressed?


Confirm Action
Display:
Confirm and Cancel
buttons, and an
empty popup,
If action is
"Cancel Compose Task"
YES
Update the popup message to:
"Warning! You are about to leave the compose task form, and
you will lose your entries upon returning to the main panel.
Are you sure you want to proceed with this action?"
NO
If action is
"Delete Task"
YES
Update the popup message to:
"Warning! You are about to delete the task: {this.task.name}
Are you sure you want to proceed with this action?"
"Cancel" button
pressed?
SET action TO
"Cancel"
Close
Confirm Action
and return action
"Confirm" button
pressed?
SET action TO
"Confirm"


Scrollable Items (Vertically)
Show all tasks
Show completed tasks only
Show unfinished tasks only
Add new task
Main Panel
All
Done
ToDo
+
My Task Name
My Task Name
My Task Name
My Task Name
My Task Name
...
Compose Task
My Task Name (FORM)
Task's Details (FORM)
Cancel
Create
Confirm Action
Confirm Action Popup
Confirm
Cancel
Detailed Page
Home
My Task Name
Task's Details
Created: dd/mm/yyyy time
Delete
Complete
Detailed Page
Home
My Task Name
Task's Details
Created: dd/mm/yyyy time
Delete
Redo

Pre-Implementation Decisions

Table of Contents

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.

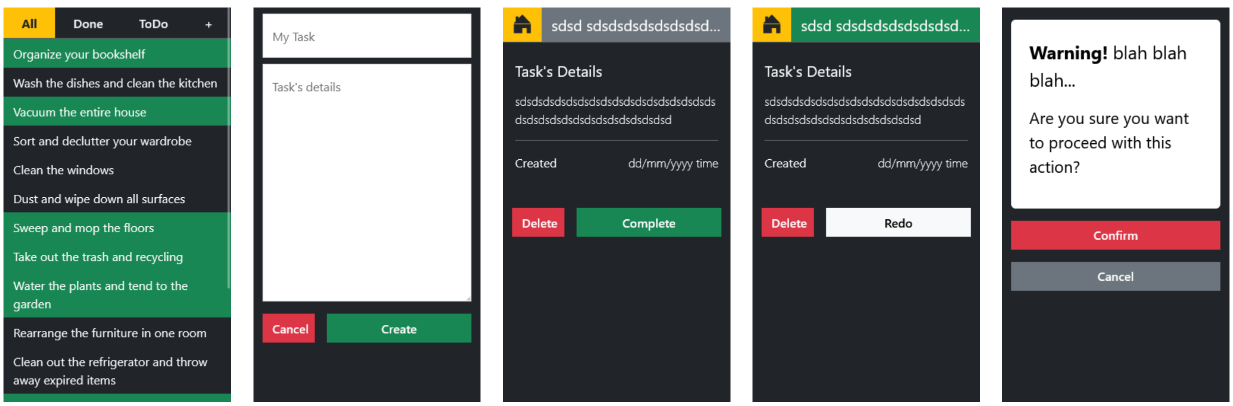
2.3 Effective Project Management

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

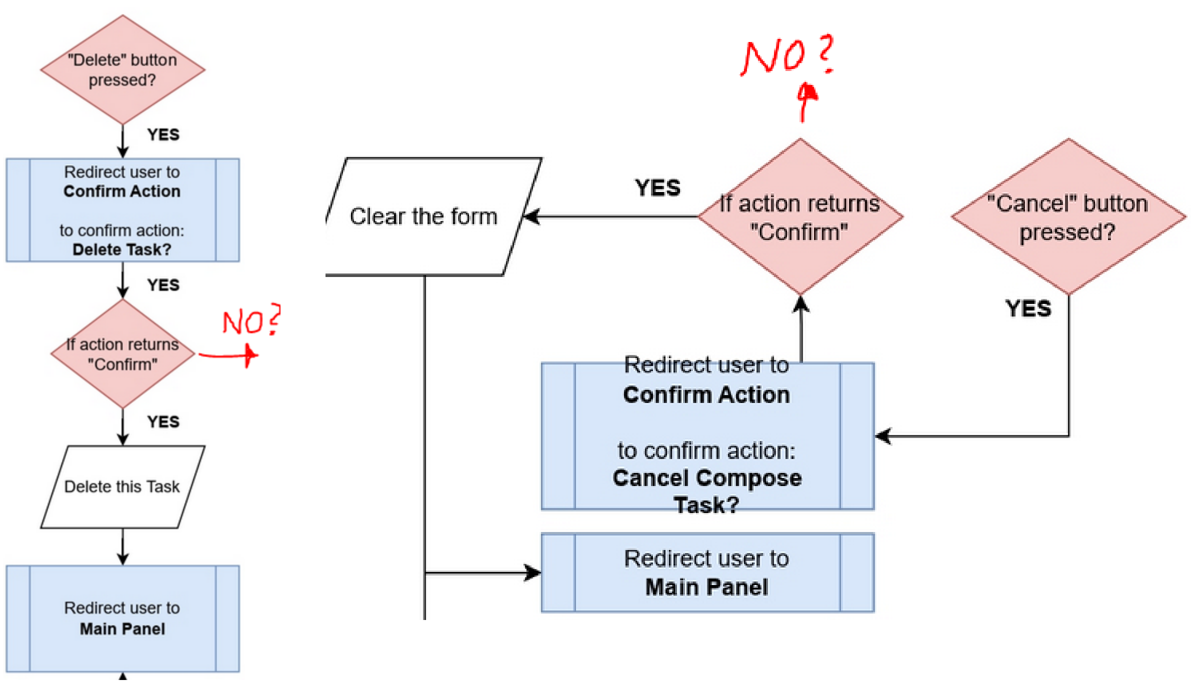
Prototype Implementation Decisions

Implemented Design: (27/06/2023)

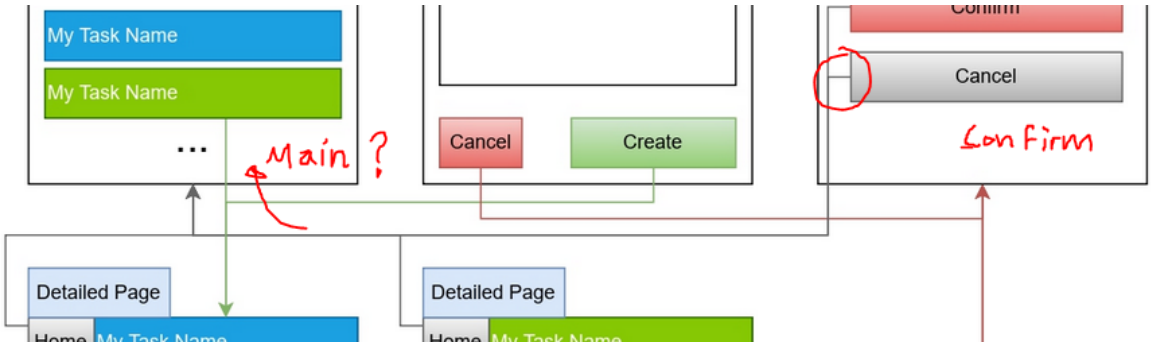
Main Panel | **Compose Task** | **Detailed Page (ToDo)** | **Detailed Page (Done)** | **Confirm Action**



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 wire-frames 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.

There are more issues:

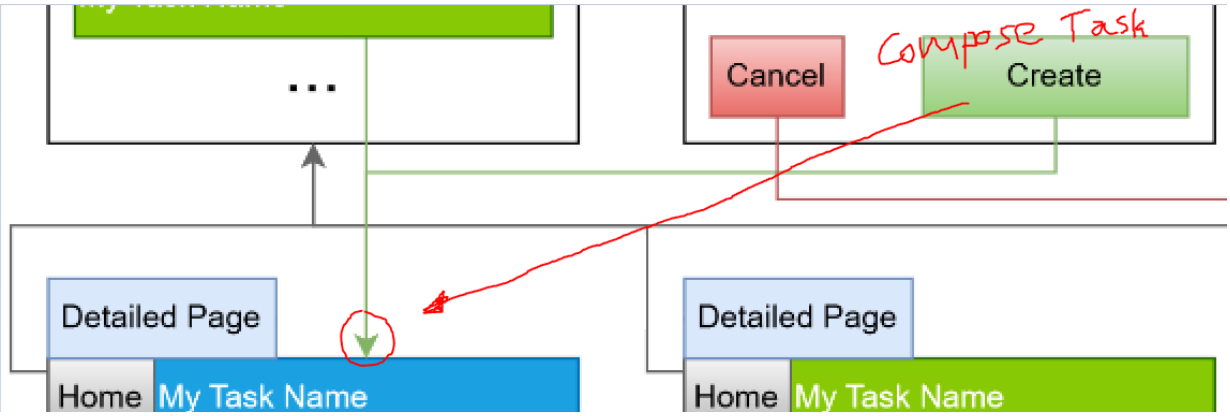
Compose-task:
The purpose of this page is to provide a form for the **task’s creation**. The form includes two attributes:

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

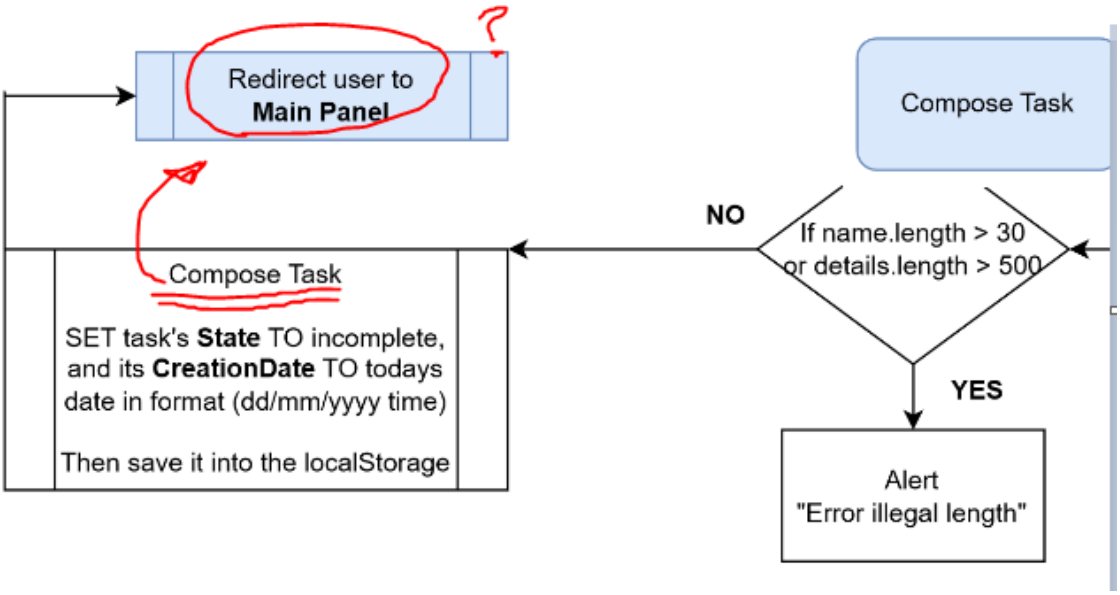
Upon submitting the form, a task is created only once all fields are populated with adequate characters length as specified above. Fields must not be empty for successful submission. Additionally, a new task’s “Date of Creation” is passed to the **detailed-page** with the user, and newly created task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

This is not an ideal approach for tackling the user's and newly created task's redirection. It's best to define, validate and create the new task before redirecting the user to the “Detailed Task” page. I also assume that redirecting intent is to display a newly created task. The confusion derives from both wire-frames and flow-charts diagrams as both re-directions are inconclusive.

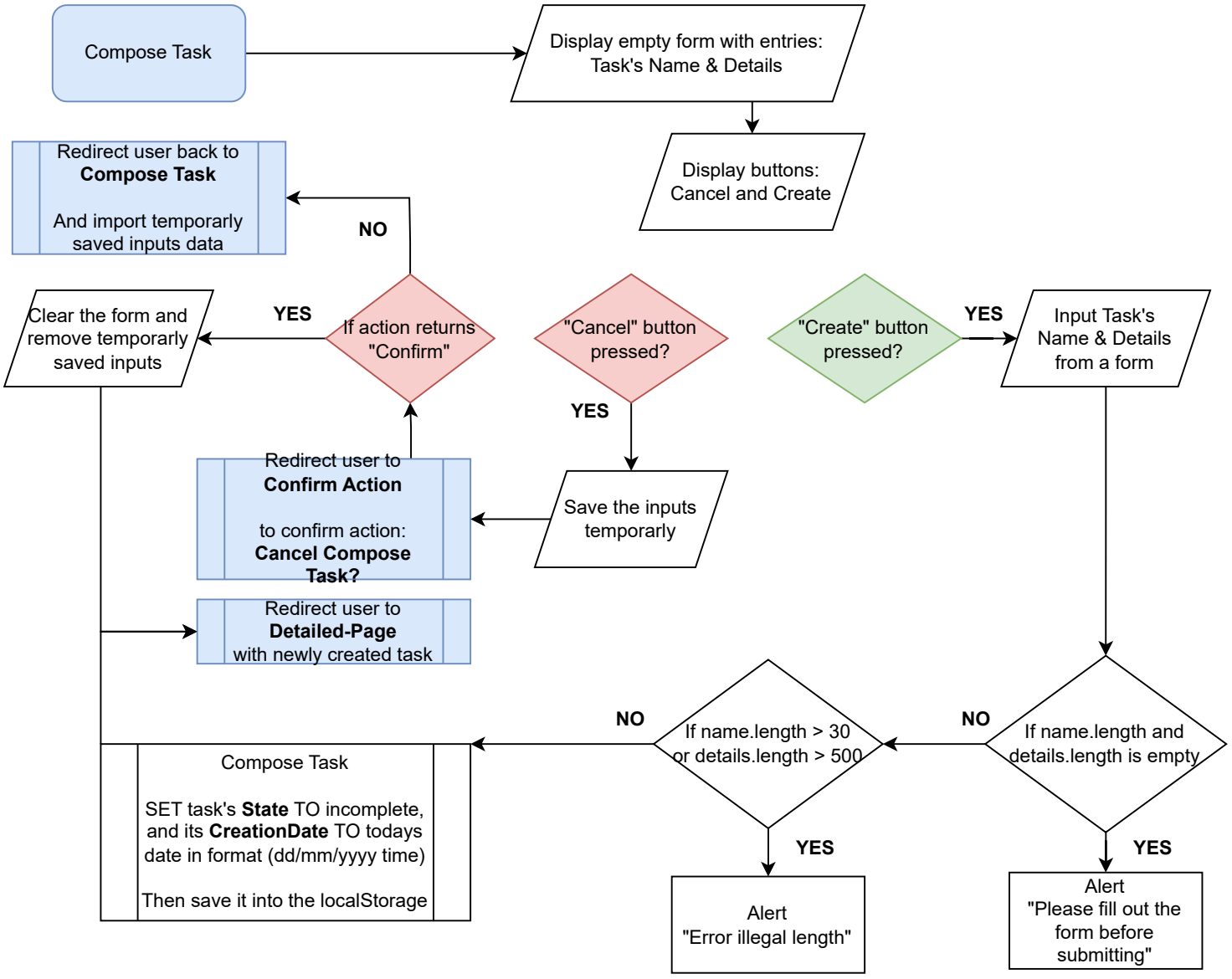
Wireframes:

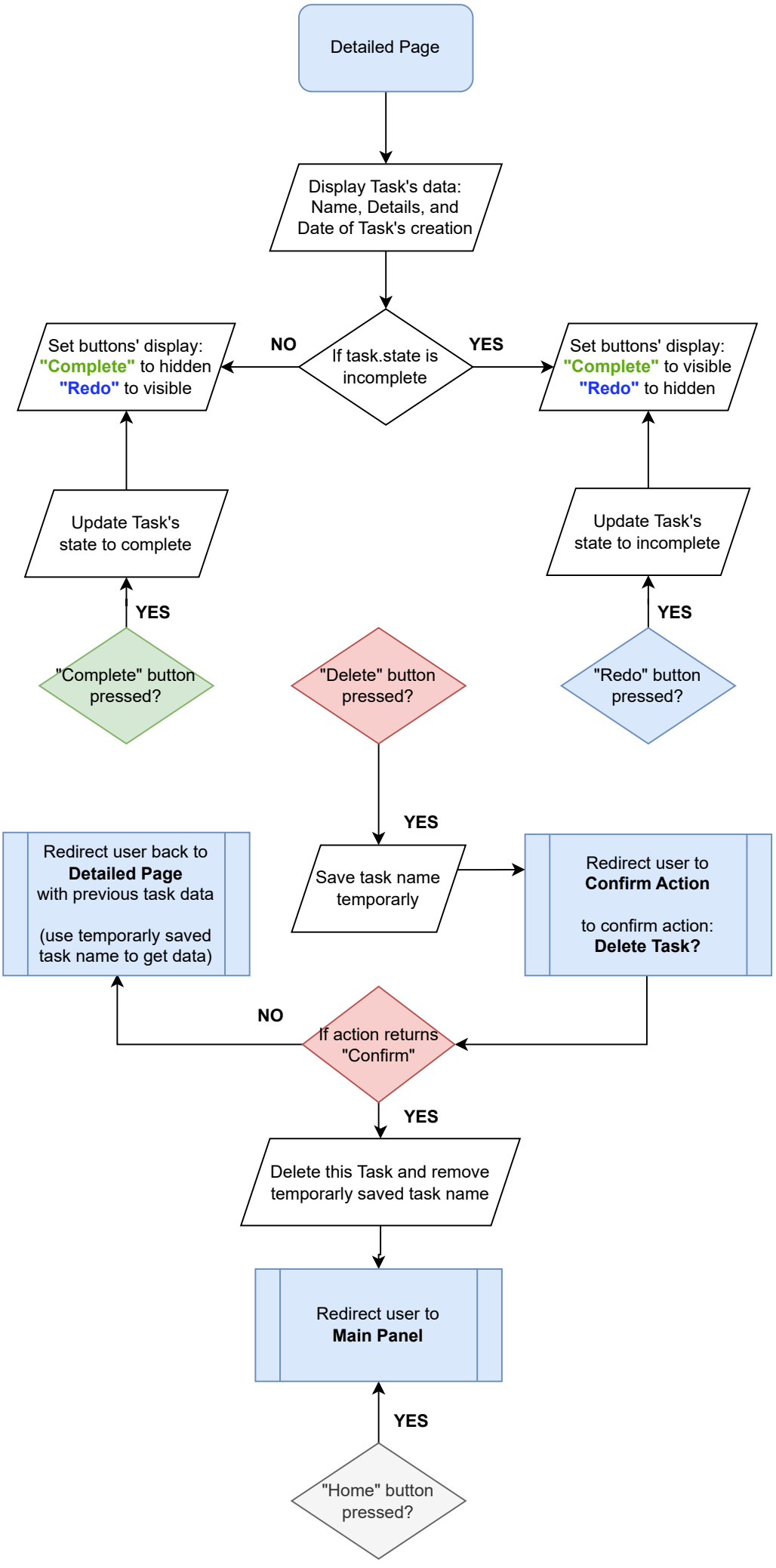


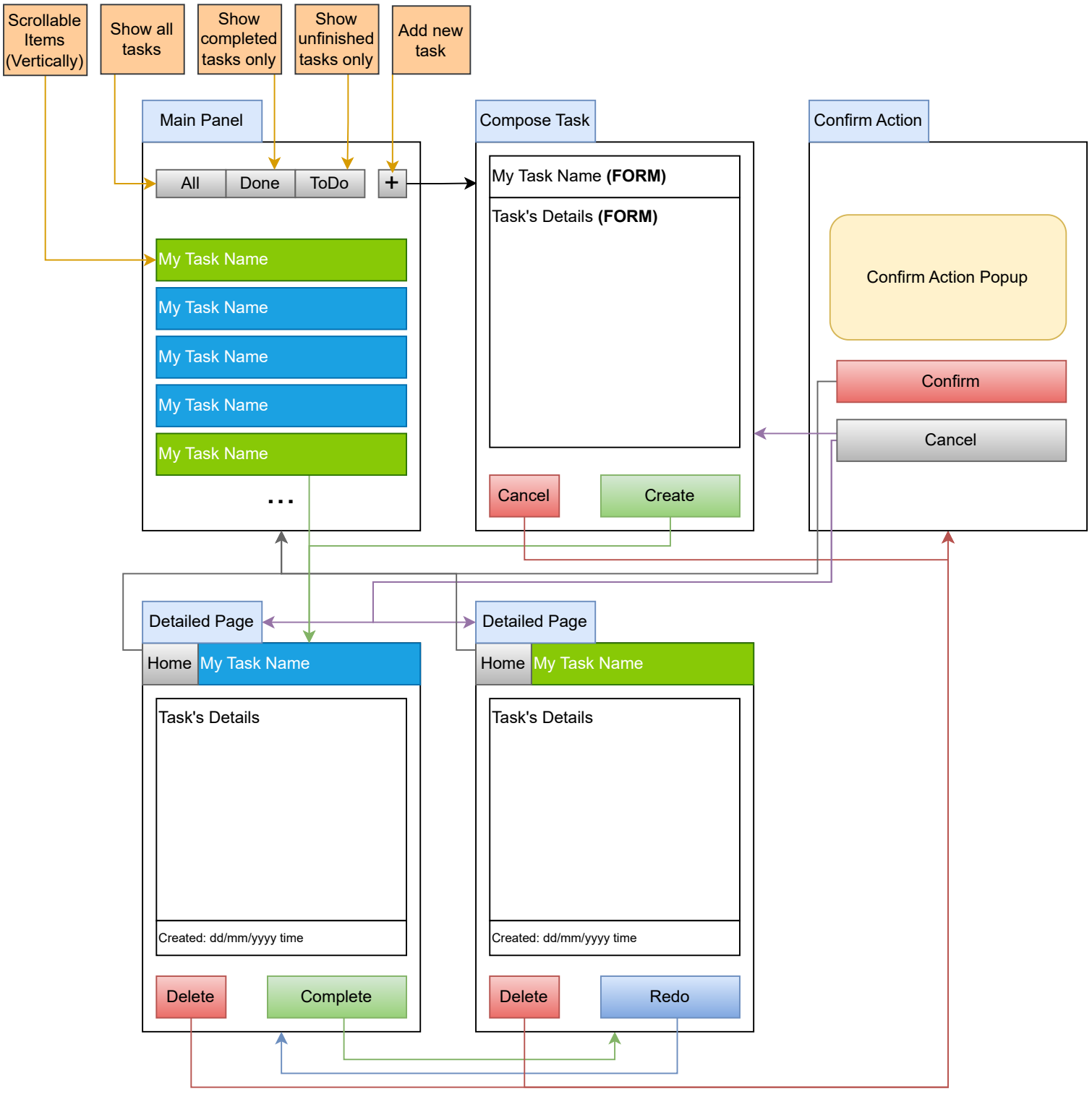
Flowchart-ComposeTask:



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.


Compose Task
Display empty form with entries:
Task's Name & Details
Display buttons:
Cancel and Create
"Create" button pressed?
YES
Input Task's Name & Details from a form
"Cancel" button pressed?
YES
Save the inputs temporarily
If action returns "Confirm"
YES
Clear the form and remove temporarly saved inputs
NO
Redirect user back to Compose Task
And import temporarly saved inputs data
Redirect user to Confirm Action
to confirm action: Cancel Compose Task?
Redirect user to Detailed-Page
with newly created task
If name.length and details.length is empty
NO
YES
Alert
"Please fill out the form before submitting"
If name.length > 30 or details.length > 500
NO
YES
Alert
"Error illegal length"
Compose Task
SET task's State TO incomplete, and its CreationDate TO todays date in format (dd/mm/yyyy time)
Then save it into the localStorage


Detailed Page
Display Task's data:
Name, Details, and
Date of Task's creation
If task.state is
incomplete
NO
YES
Set buttons' display:
"Complete" to hidden
"Redo" to visible
Set buttons' display:
"Complete" to visible
"Redo" to hidden
Update Task's
state to complete
Update Task's
state to incomplete
YES
YES
"Complete" button
pressed?
"Delete" button
pressed?
"Redo" button
pressed?
YES
Redirect user back to
Detailed Page
with previous task data
(use temporarily saved
task name to get data)
Save task name
temporarly
Redirect user to
Confirm Action
to confirm action:
Delete Task?
NO
If action returns
"Confirm"
YES
Delete this Task and remove
temporarly saved task name
Redirect user to
Main Panel
YES
"Home" button
pressed?


Scrollable Items (Vertically)
Show all tasks
Show completed tasks only
Show unfinished tasks only
Add new task
Main Panel
All
Done
ToDo
+
My Task Name
My Task Name
My Task Name
My Task Name
My Task Name
...
Compose Task
My Task Name (FORM)
Task's Details (FORM)
Cancel
Create
Confirm Action
Confirm Action Popup
Confirm
Cancel
Detailed Page
Home
My Task Name
Task's Details
Created: dd/mm/yyyy time
Delete
Complete
Detailed Page
Home
My Task Name
Task's Details
Created: dd/mm/yyyy time
Delete
Redo