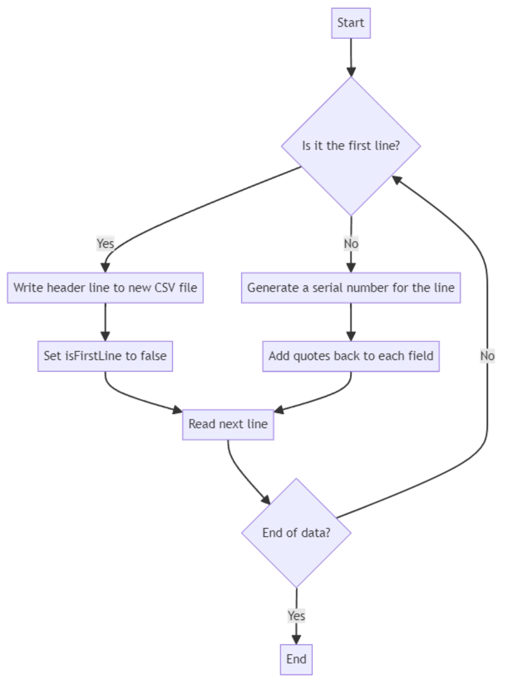
**U16A2**

## *To-Do List Application*

The goal of the To-do List app is to give users a simple, yet powerful solution for task management that utilizes a graphical user interface. Users are able to add, modify, remove and mark jobs as finished. The application will also allow you to examine tasks according to their level of completion, you can sort by all tasks or just incomplete tasks. Given that it includes developing a graphical user interface (GUI), processing user input and organising task data, this task is of average complexity.

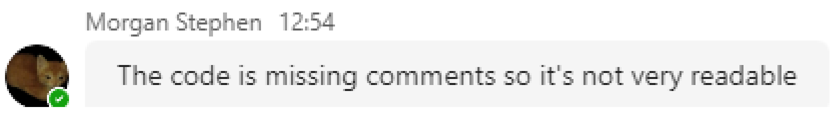
The first requirement is that the application must have a graphical user interface (GUI) that is made with Blazor, this may be a constraint due to the fact the interface must be user-friendly and may require design plans. When tasks are kept in memory, the data is deleted after closing the application. For users who need permanent task storage, this limitation restricts the efficiency and usability of the application.


Start
Is it the first line?
Yes
No
Write header line to new CSV file
Generate a serial number for the line
Set isFirstLine to false
Add quotes back to each field
No
Read next line
End of data?
Yes
End

| Test Number | Steps | Description | Expected Results | Actual Results | Comments |
|---|---|---|---|---|---|
| 1 | Add a task. | To see if adding a Task is successful | Task is added. | Task is added. | N/A |
| 2 | Try to edit the description of a task | To see if Editing a Task is successful | Description is edited. | Description is edited. | N/A |
| 3 | Delete a task | To see if Deleting a Task is successful | Task is deleted. | Task is deleted. | N/A |
| 4 | Try to mark a task complete | Marking a Task as Completed/Incomplete | Task is marked complete. | Task is marked complete. | N/A |

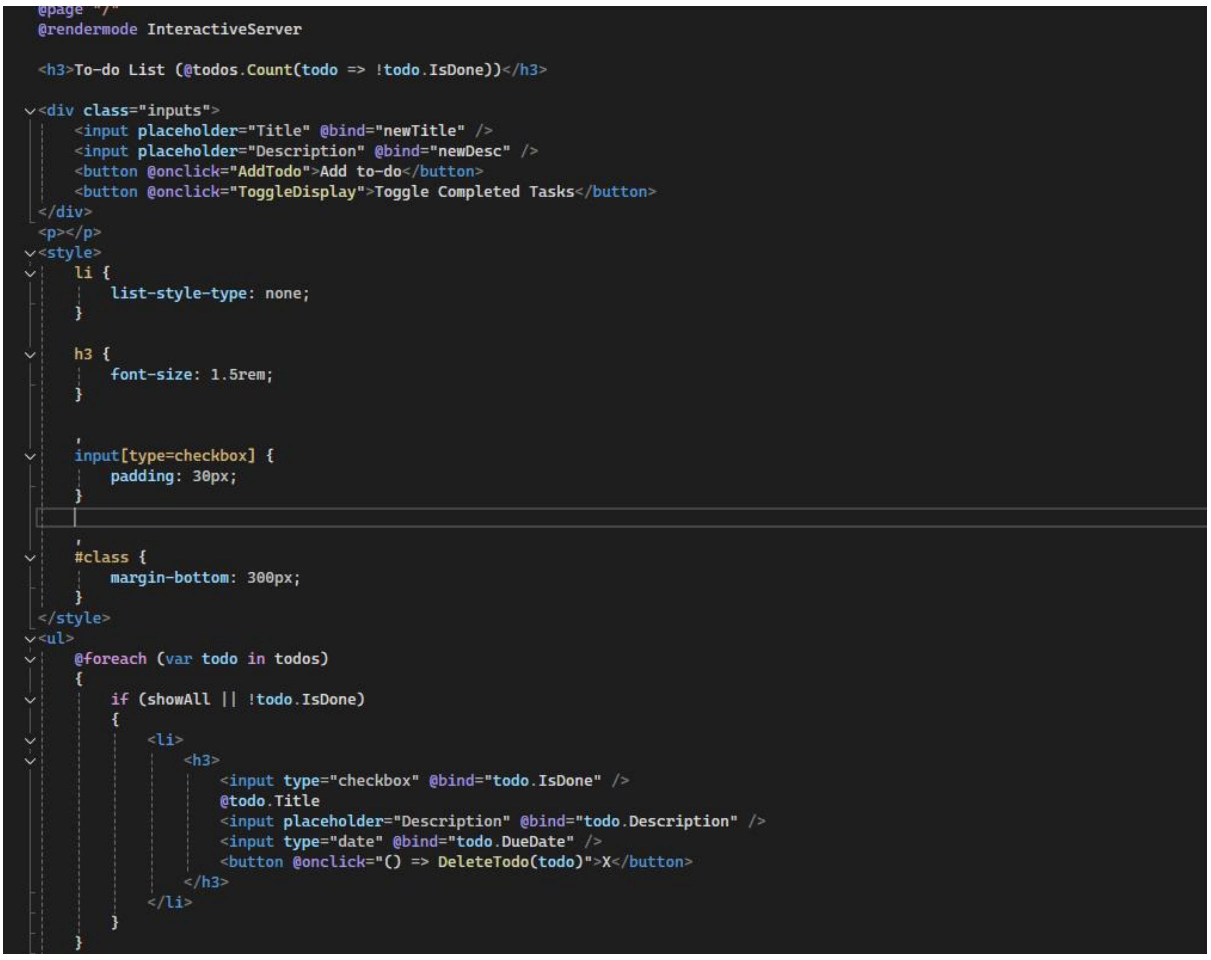
| | | | | | |
|---|---|---|---|---|---|
| 5 | Toggle view incomplete tasks | Viewing All Tasks/Incomplete Tasks | Visibility is accurate. | Visibility is accurate. | N/A |
| 6 | Try to add a blank task | Error Handling | Blank tasks cannot be added. | Blank tasks cannot be added. | N/A |

**Feedback**

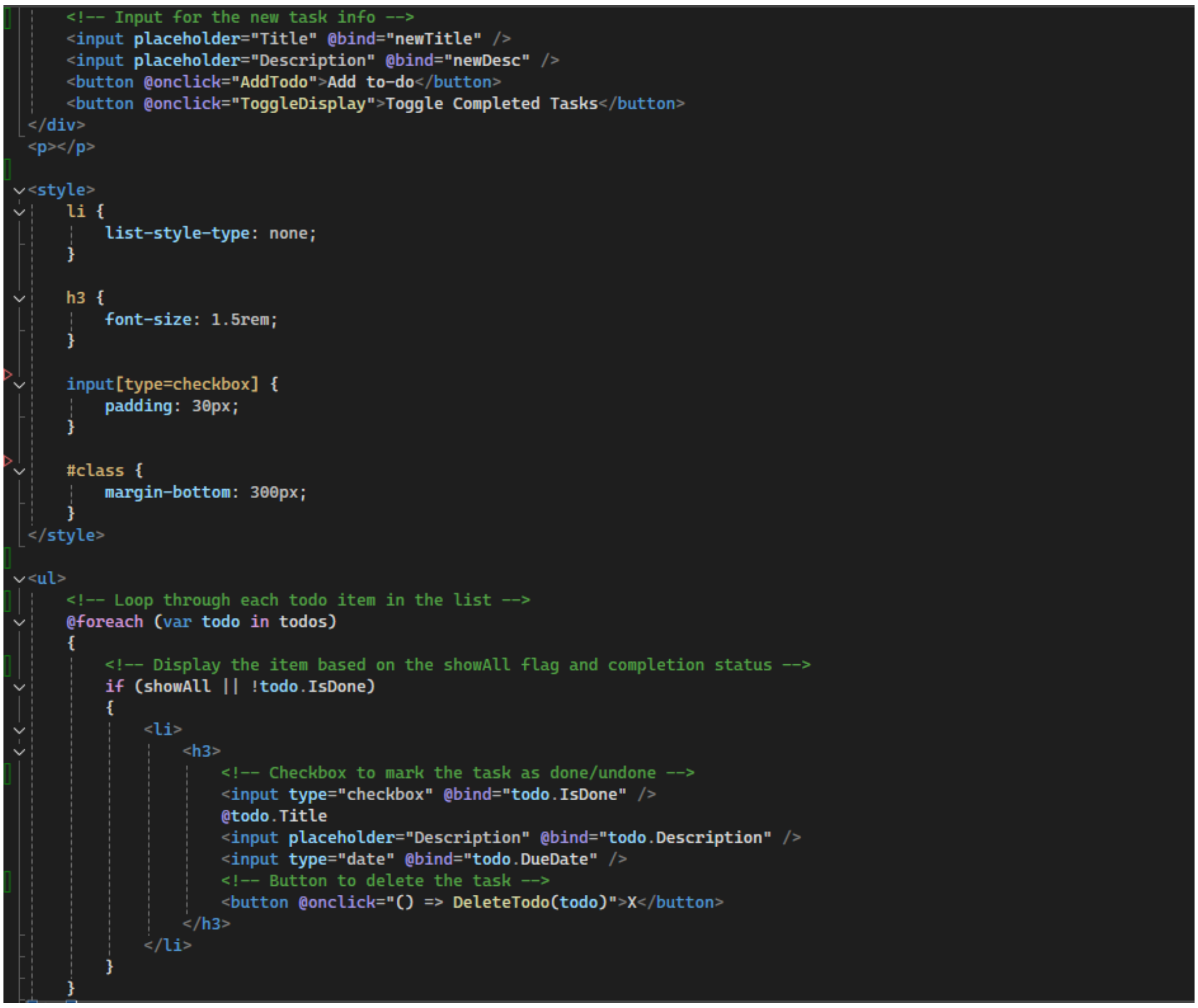

Morgan Stephen 12:54

The code is missing comments so it's not very readable

Before Optimization:

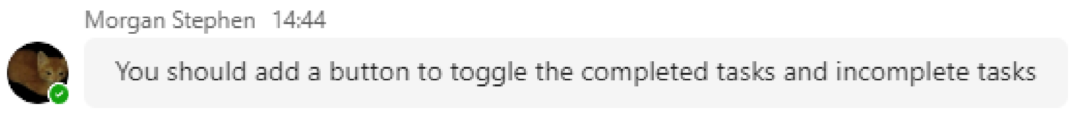

```
@page "/"
@rendermode InteractiveServer

<h3>To-do List (@todos.Count(todo => !todo.IsDone))</h3>

<div class="inputs">
    <input placeholder="Title" @bind="newTitle" />
    <input placeholder="Description" @bind="newDesc" />
    <button @onclick="AddTodo">Add to-do</button>
    <button @onclick="ToggleDisplay">Toggle Completed Tasks</button>
</div>
<p></p>
<style>
    li {
        list-style-type: none;
    }

    h3 {
        font-size: 1.5rem;
    }


    input[type=checkbox] {
        padding: 30px;
    }



    #class {
        margin-bottom: 300px;
    }
</style>
<ul>
    @foreach (var todo in todos)
    {
        if (showAll || !todo.IsDone)
        {
            <li>
                <h3>
                    <input type="checkbox" @bind="todo.IsDone" />
                    @todo.Title
                    <input placeholder="Description" @bind="todo.Description" />
                    <input type="date" @bind="todo.DueDate" />
                    <button @onclick="() => DeleteTodo(todo)">X</button>
                </h3>
            </li>
        }
    }
```

After:



```
        <!-- Input for the new task info -->
        <input placeholder="Title" @bind="newTitle" />
        <input placeholder="Description" @bind="newDesc" />
        <button @onclick="AddTodo">Add to-do</button>
        <button @onclick="ToggleDisplay">Toggle Completed Tasks</button>
    </div>
    <p></p>

<style>
    li {
        list-style-type: none;
    }

    h3 {
        font-size: 1.5rem;
    }

    input[type=checkbox] {
        padding: 30px;
    }

    #class {
        margin-bottom: 300px;
    }
</style>

<ul>
    <!-- Loop through each todo item in the list -->
    @foreach (var todo in todos)
    {
        <!-- Display the item based on the showAll flag and completion status -->
        if (showAll || !todo.IsDone)
        {
            <li>
                <h3>
                    <!-- Checkbox to mark the task as done/undone -->
                    <input type="checkbox" @bind="todo.IsDone" />
                    @todo.Title
                    <input placeholder="Description" @bind="todo.Description" />
                    <input type="date" @bind="todo.DueDate" />
                    <!-- Button to delete the task -->
                    <button @onclick="() => DeleteTodo(todo)">X</button>
                </h3>
            </li>
        }
    }
```

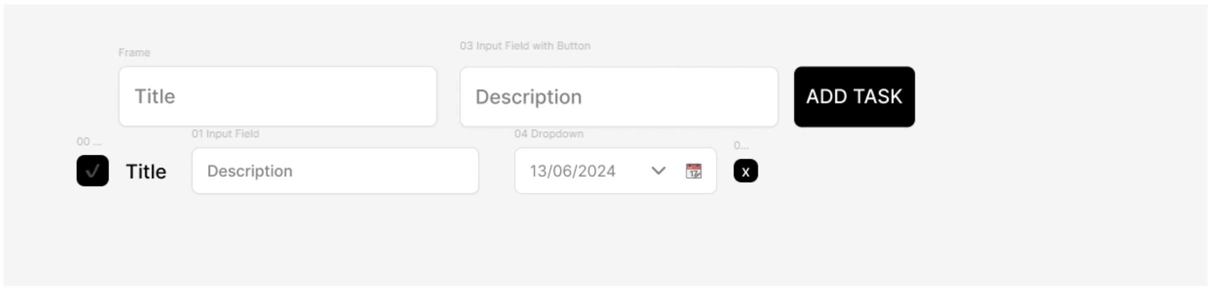
## Visual Design Plan Review



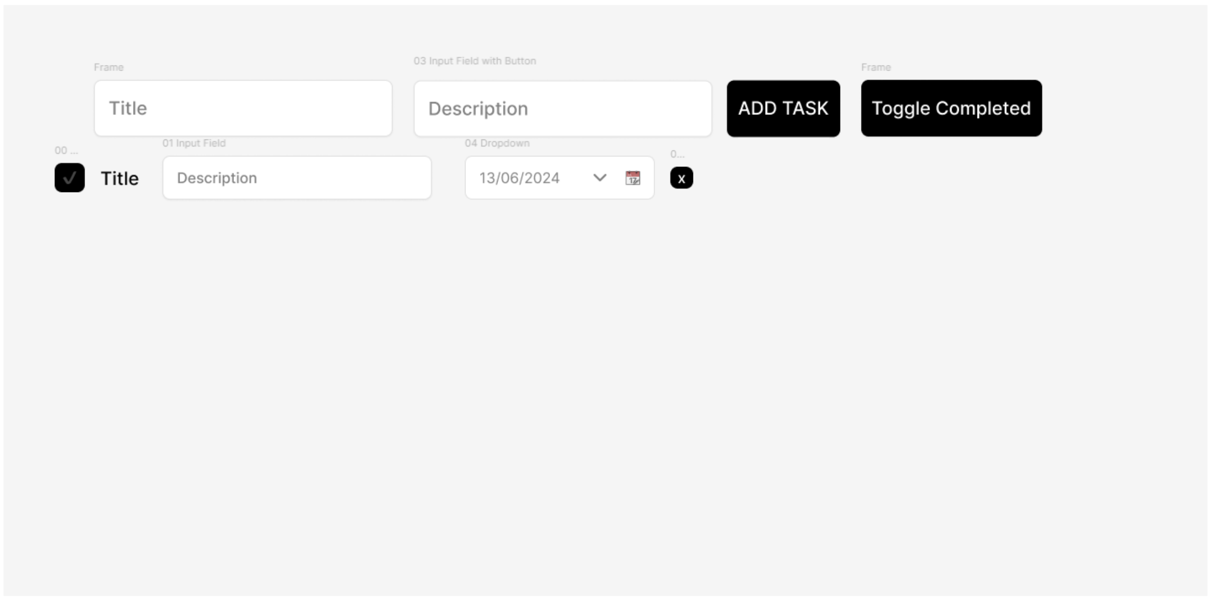
Morgan Stephen 14:44

You should add a button to toggle the completed tasks and incomplete tasks

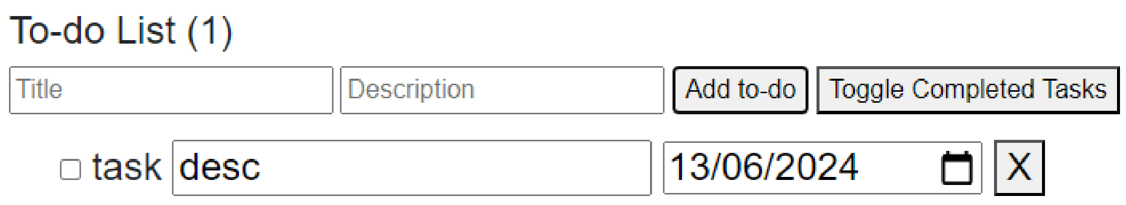
### Before Optimization:



### After:



**Final Design:**



### Design Justifications

This optimized visual design takes into consideration the clients requirements and provides all of the necessary design additions within a visually appealing graphical user interface. Before receiving peer assessment, my design lacked a button that was able to toggle the ability to view incomplete or completed tasks. After re-reading the brief I realized that the client did request this ability, and quickly corrected my mistake in the new design.

## Data Dictionary

| Variable | Type | Description |
|---|---|---|
| todos | List | A list that stores all the to-do items. Each item is of type TodoItem |
| newTitle | string | A string that holds the title of the new to-do item being added. |
| newDesc | string | A string that holds the description of the new to-do item being added. |
| showAll | boolean | A Boolean variable showing whether to display all tasks or only incomplete tasks. |

|  |  |  |
|---|---|---|
| TodoItem.Title | string | A property of TodoItem representing the title of the to-do item. |
| TodoItem.Description | string | A property of TodoItem representing the description of the to-do item. |
| TodoItem.IsDone | boolean | A property of TodoItem that shows the to-do item is completed. |
| TodoItem.DueDate | DateTime | A property of TodoItem that shows when the due date of the to-do item is. |

## *Review against client requirements*

The client has asked for an application to be produced using WPF or Blazor, in this case I utilized Blazor in order to meet the following criteria:

- **Creation and Deletion of Tasks**

  I have met the following criteria using the method "AddTodo", in which I allow the user to input a newTitle and newDesc which will cause the creation of a task visible to the user. I have also given the user the ability to delete tasks using the "DeleteTodo" method.

- **Tracking done state and allowing tasks to be set to complete**

  The user is able to toggle which tasks they wish to see, only incomplete tasks or all tasks using the information provided by the checkboxes (which mark an item as "IsDone".

- **Supporting title, description, due date, completed with description and due date being mutable.**

  All of these are presented to the user, with only the description and date being mutable using Blazor's @bind attribute..

- **Displaying a list of tasks**

  The tasks are displayed with their Title, Description, DueDate, and a checkbox for the IsDone status.

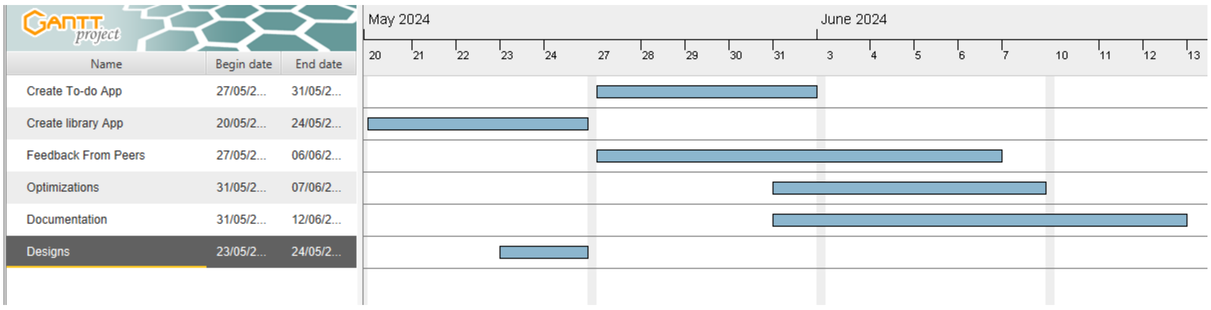
- **Toggling whether all tasks or only incomplete tasks are displayed**

  The ToggleDisplay method toggles the showAll flag between true and false.

In conclusion, I have efficiently followed client requirements in order to make a working code that is visually appealing and easy for any users to navigate and control. Along with taking my own initiative in order to optimize the code and provide additions that make the code more efficient and easy to understand and edit in future.

**Planning and Managing My Time**

My progress has been greatly enhanced by setting deadlines and goals. I divided more complex tasks into smaller jobs with specific due dates. This method enabled me to efficiently manage my workload. I was also sure to review my work with peers throughout the project and gain feedback. This allowed me to constantly optimize the code and visual designs and meet client requirements more effectively. I decided to create a Gantt chart in order to better visualize my deadlines:



| Name | Begin date | End date |
|---|---|---|
| Create To-do App | 27/05/2... | 31/05/2... |
| Create library App | 20/05/2... | 24/05/2... |
| Feedback From Peers | 27/05/2... | 06/06/2... |
| Optimizations | 31/05/2... | 07/06/2... |
| Documentation | 31/05/2... | 12/06/2... |
| Designs | 23/05/2... | 24/05/2... |

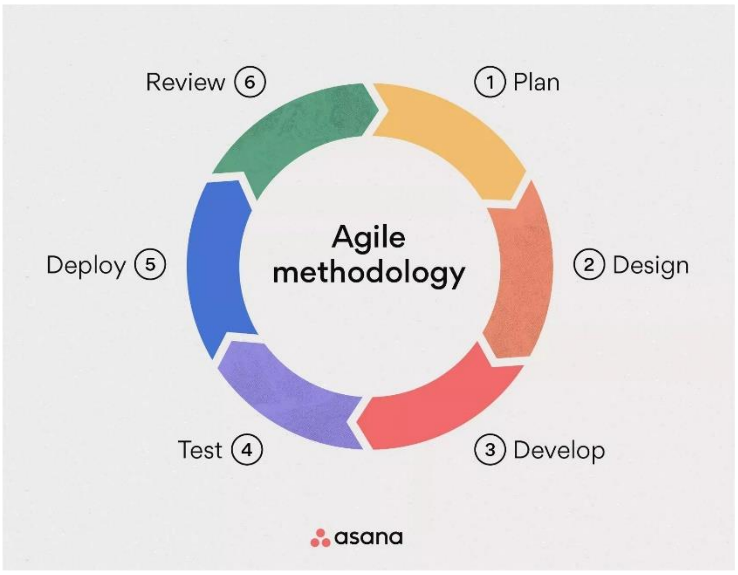
## Evaluation

The page shows a to-do list application that uses Blazor and efficiently exhibits an intuitive and user-friendly interface for task management. Input options for task details, including title, description and due date, are included in the layout along with buttons to add new tasks and toggle the display of completed tasks. The user can easily navigate the application thanks to my graphic design.

Even though the existing solution successfully satisfies the requirements for basic task management, there is room for development. An area that could use improvement is the lack of input validation. Implementing input validation to the application will make for a much more well rounded code and prevent issues within the to-do list where the user is able to input incorrect information. In conclusion, this Blazor-based application exhibits a responsive user interface in addition to fulfilling its functional needs.

### *Methodology Evaluation*

Within this project I utilized the Agile methodology, which involves breaking the project into phases and involves a cycle of planning, execution and evaluation (Source: Atlassian)



*(Source: Asana)*

**Plan:** I understood the client's needs, which included the ability to create, delete and manage tasks in addition to certain GUI requirements utilising Blazor. I then deconstructed the requirements into tasks that needed to be accomplished inside a Gantt chart that I created.

**Design:** Using the client's specifications for a Blazor-based GUI, I designed a user-friendly interface. Input fields for task details, buttons for actions like adding tasks, and toggling display were strategically placed for ease of use.

**Develop:** I efficiently developed the code for the application using Blazor following the Gantt chart's strategic planning.

**Test:** To ensure that every implemented feature complies with client requirements, I ran functional tests. Task addition, task deletion, task editing, task marking as completed, and task view switching were all included in this.

**Deploy:** After testing the application, I successfully deployed the application to GitHub.

**Review:** I received feedback from peers multiple times and detailed how I used this feedback for optimization and evaluated the whole project.