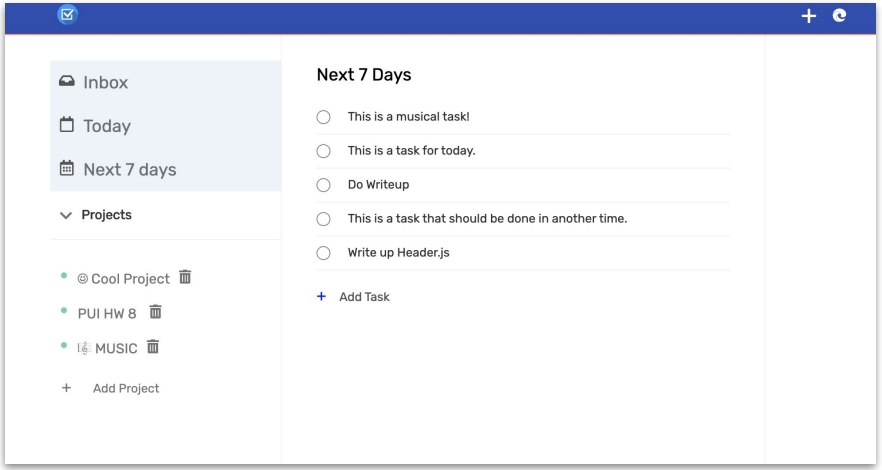
Matt's Task Manager

PUI HW 8 Writeup



Part 1: Summary

I. Purpose

Create a task manager that could separate their tasks by **projects** and **timeframe** so people could be more organized and accountable with their tasks

II. Content Conveyed

My goal is to make it clear to users that they can create tasks that can be categorized into different projects, and into different timeframes (today, tomorrow, next 7 days).

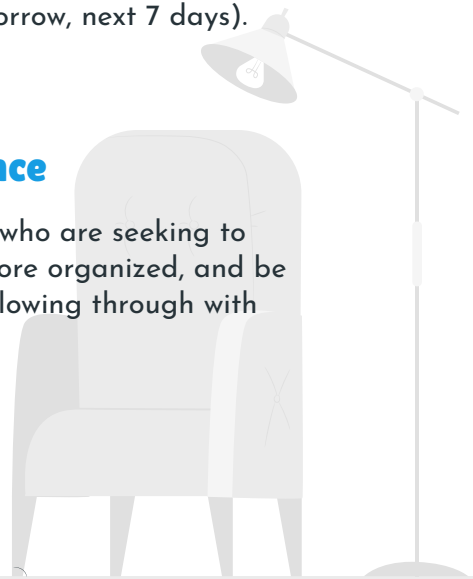
III. Value

Being able to holistically visualize all your tasks, and have them in a compact organizational system can be very exciting, and may motivate users to follow through with their tasks.

There's also a cool dark mode for users who are night-lurkers!

IV. Target Audience

The audience is people who are seeking to record their tasks, be more organized, and be more accountable in following through with their tasks.



Part 2: User Interactions

- **Seeing the list of tasks that are scheduled in a certain period of time**
  - *Click on "Next 7 Days" to see all your tasks scheduled for the next 7 days (tasks will be ordered by date)*
- **Adding a Project**
  - *Click on the "+add project" button to name and add your project*
- **Adding a Task**
  - *Click on the "+add task" button either within a project or within a timeframe to add a task*
  - *Click on the calendar icon to schedule your task today, tomorrow, or for the next 7 days*
  - *Click on the business card icon to put your task into a certain project*
- **QuickAdd Task (for mobile devices)**
  - *On the header, click on the "+" icon to add a task (has the same functionality as a normal way to add task)*
- **Changing to Darkmode**
  - *Click on the Swirl icon in the header to make the app into a darkmode setting*

Part 3: Tools Used

1. **React**
    a. I chose to use it because the ability to write components would make reusing elements much easier. Also, it's a full-stack application that is used commonly industry.
    b. I took full advantage of the components, and also used React Hooks to implement my code.
    c. The value of using React was that it provides me far more flexibility and options to implementing things than using Javascript.
2. **Firebase**
    a. I chose to use it so I can have a database to store my tasks.
    b. I hooked up my task-manager app to the firebase so I could not only test my code, but also so I wouldn't need to have a JSON file to store tasks.
    c. This adds value because we can just store all our tasks into a backend server, rather than having a file keep track of all the tasks.
3. **Moment.js**
    a. I chose to use this library because it would be a convenient way to help me format time.
    b. I used this to help me filter and order the time for user tasks.
    c. This adds value because it helps users order tasks by time.

Part 4: Iterations from Mockups

Understanding what the Quick-add Task would do

- Initially, I didn't think too seriously what a "quick-add task" button would do. Would the task just be added to the inbox? Would the user have complete flexibility to add it to a specific project? A timeframe?
- Afterwards, I decided to have the quick-add-task button have the same functionality of the normal add-task button. The ability to add a task should be consistent across all boards. The only difference is that a modal appears to help a user make a task.

Part 5: Challenges & Lessons Learned

I. Learning React+ BEM

- I had only worked with class-based components and styling with normal CSS syntax in the past. Working with function-based components and React Hooks was definitely time-consuming. I also wanted to challenge myself with BEM naming convention to do CSS styling. I had to devote a significant amount of time watching YouTube videos to understand these concepts. It's worth it because using these are best practices in modern industry!

II. Understanding Code vs Accepting it For What It is

- Sometimes when you use libraries or external sources (Firebase, moment, etc) it's tempting to understand what each line of code means. However, I've learned that in the interest of time, it might be best to just accept what that code does, and instead, focus more on understanding code that is important to the structure of your app, such as useEffect() and useState() within React.

III. Learning how to Debug Efficiently

- It was very tempting to try to just search through my code on VScode and look for my bugs - this was very ineffective. Then, I learned that it would be more effective to use developer tools. If I were still stumped, I would look up my error online and see if people had similar experiences to me. Finally, if I were still stumped, I would ask my question on Stackoverflow as a last resort. I've found this process was much more efficient than me trying to search for my error on VScode.