Minimalist Notes App: Portfolio Part 3

Objectives

To develop a minimalist productivity app that allows users to manage their tasks and time in a minimalistic retro interface. The minimal notes app is a lightweight web app that has helpful tools to assist people with getting organized & planning their day.

Concept

The application was built to help users manage their daily tasks with a clean, retro-style interface**. It offers the following key features:** Write, save & delete personal notes; Manage to-do lists (with strikethrough function); Track task times with a retro stopwatch timer; Use the day progress meter to plan ahead; View Date/Time in the app; Plan the day using real-time weather data; Secure login via Google or email/password; Save your notes, to-do's & timers in your account; Switch between daylight and night mode; Bonus: Get Motivational Quotes via API.

Implementation

This application was developed in three phases. The **first phase** was used to go over the concept. The **second phase** was used to build the minimum viable product. After feedback and testing the **final phase** was built with feedback in mind.

Design Process

I researched applications on YouTube & GitHub and decided to bundle features together in a productivity app (that had plenty of existing documentation and guidelines). I added a retro style to it because it's a design-trend that's popular currently. Regrettably I didn't plan & just dove right in and started to code the features one-by-one next to one another on the page as I wasn't sure if I'd be able to create what I wanted. Once I had the basic components on the page I added styling and decided on NES style buttons, half-shadows and a retro Press 2P font.

Frontend Development

I developed the frontend using **HTML, CSS, and JavaScript**, and hosted it all on **GitHub Pages**. The majority of the features are rendered on the frontend, as it allows the app to be much quicker and lightweight. I had a bit of a struggle with the responsiveness as the app had issues when tested in Brave/Firefox and on larger screens than my own. I finally managed to get it working in a satisfactory way after using Flexbox and adding some Firefox specific fixes for the flip box. I got some design inspiration by searching for Retro apps and styling my font and buttons (and hover effects) in a nice retro-style. I enjoyed styling the individual components.

Backend Development

I got feedback in my phase two that I didn't have enough dynamic elements. As such, I decided to build more features into the app. I had one element that counted

as dynamic (the Weather API) and decided to add a quote feature (with a Quote API). I felt, however, that this wasn't sufficient and this was an ideal opportunity to learn how to implement some more backend features. I built a backend using **Node.js and Express**, hosted on **Render**. Note: Render takes 60 seconds to load on the free tier when the app has been idle for longer that 15 minutes. I added **User authentication** via **Google Sign-In** but it had several issues when tested in other browsers, on mobile and on Google itself so I added the regular **email/password login** to enable personal accounts. I used **MongoDB Atlas** as the database to store user-specific data (notes, tasks and timers). I now have three **external APIs** : the weather data (OpenWeatherMap), Authentication (Google), and Quotes (API Ninjas).

Testing & Debugging

I used the Inspector tool to test the app on mobile and tablet. I also asked a friend to test it on his Windows PC (Brave browser) & tested it on my brother's PC with Firefox. I spotted issues (Google Login, Flip Box and responsiveness) which I fixed until they worked with testing not only on my MacBook but also on their PC's.

Documentation & Finalization

I created clear documentation for the project, including: A **README** with setup instructions, test cases and feature overviews; **Inline code comments** for maintainability & a final **walkthrough video** to showcase the app.

Lessons Learned

This project taught me the value of starting with a clear plan. In retrospect I wish I had spent more time on design. I would have created a different multi-page design that would have saved me a lot of time debugging responsiveness issues. I also wish I had submitted the version with the backend in phase 2 as I could have received more feedback at that stage. Planning is crucial to save time during development.

I gained experience integrating **external APIs**, handling **authentication**, and deploying both frontend and backend components on GitHub. I now understand backend and database integration for the first time and it's a lot less complicated than I thought it would be. I enjoyed adding different features as well such as the dark mode & button hover effects. I also liked adding authentication and persistence features which brought everything together quite nicely in the end in my opinion.

Reflections & Future Improvements

The final app succeeded in delivering a retro minimalist productivity tool. For future iterations, I'd like to redesign the interface, make it multi-page & add more interactive design features like drag and drop. Perhaps different ways to sign in. Progress analytics- a page with charts and a dashboard showing how the user spent the day. Undo/Redo functions, voice notes & verbal input would be nice. An LLM tool to interact with. A Calendar syncing option as well or reminders via email/text app. Theme customization and auto-toggle on the dark mode would be neat, too.