Opus – Task Management Web Application

Opus is the name of my task management web application, and this is my final write up for the CVWO Assignment AY2021/22.

Roadmap Changes

Previously, I wrote a roadmap of features and have completed them except for project collaboration. Implementing it would lead to more complexed problems like conflict management due to concurrent edits. Hence, I wanted to lay down a more stable foundation before I make such a leap since I was rushing to implement features in the past.

Therefore, I laid down some goals and this was what I achieved since the previous write up, red-coloured goals are the deviations from the original write up.

1. Task Deadlines 2. Undo User Actions

3. Error Handling UI 4. Complete User Authentication 5. Accessibility Improvements 6. CSRF Prevention

Challenges Faced

Since this is my first attempt at web development, it took time to familiarise with some libraries and realise ideas I envisioned. The following are the few significant hurdles I faced.

Undo User Actions

This was a fun challenge since undo implementation with redux I see online did not felt applicable. Usually, a history of user states is stored, and rollbacks are done by replacing the entire current user state. However, this implementation would require me to update the database with the full project tasks layout & data as the app is unaware of the specific changes. Storing large project layout may cause memory issues too.

Hence, I wrote an enhancer on top of my project layout reducer to track the actions made, undo actions then used my pre-existing actions as much as possible (eg. 'update task info' is used to update task state and hence can be reused by the undo action to reverse tasks changes). Writing actions that does very specific actions (eg. 'complete task', 'set task name') would be complicated as I would need a 1 to 1 mapping for every action to make any action undoable.

Production Environment

Writing the Dockerfile and docker-compose was relatively straightforward for the PostgreSQL and rails, but I soon learnt I had to serve the static assets myself in production. I had to setup nginx for the purpose of serving my static assets and redirecting all other requests to rails. It was a relatively fun learning process, especially on the motivation behind separating web and application server, and the rails asset pipeline between development and production environment.

CSRF & CORS Prevention

Preventing CSRF using authenticity token felt like it was going against RESTful principles as it was stateless. As a small trade-off, I settled on storing a CSRF token as a meta tag in the HTML view and use that as an authentication token for all requests from the site. It felt like I understood CORS and how to implement it, however I have not yet understood why the rails middleware is not working as expected, I will be reading up or implementing it on a fresh project in hope for some fresh perspective.

Looking Back

On hindsight, there are some fundamental changes that I feel would benefit Opus.

Local state and database synchronisation

When I first thought about the project, I was worried about synchronisation between the user local state (what they view) and the database state. Hence, all user actions first updates the database, then make local state changes on success. The opposite felt more complicated since I would have to revert local state changes if the database failed to update.

However, there are some implications for this decision. The time lag between action fired and view changes would be significant if the request to the backend server took time to update the changes and return a response. Lag time really does not make for good user experience since the application just feels slow to response consistently.

Organising my styles

As I developed, it soon became apparent that I could have organised my css styling better. I was utilising *styled-components* for my individual components, general *.css* files for mainly application wide styles and soon I was using *tailwind css* as I utilised *headlessui* for some of my components. Now that I am clearer of the general user interaction designs and feels for the app, I will be improving on this aspect in the near future as I work on this further.

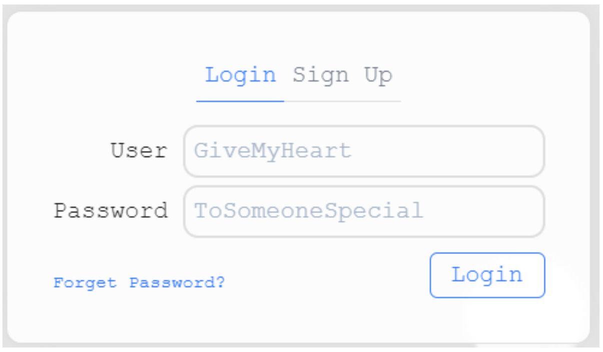
Conclusion

I am generally proud of how far I came with this project. However, I wished I spent more time on fundamental web practices (etc. production deployment, security practices), occasionally it was a more interesting topic to me. Furthermore, there are some implementations mentioned above that I am unsure if there exist better implementations, I will explore further. Nonetheless, I am glad I attempted this assignment.

User Guide

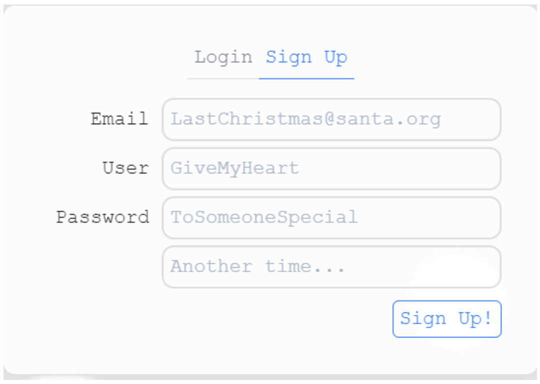
Login / Sign Up

On entering the site, you will be greeted by the **Login / Sign up** page, simply fill in the necessary details to enter the task page.



User GiveMyHeart
Password ToSomeoneSpecial

Login

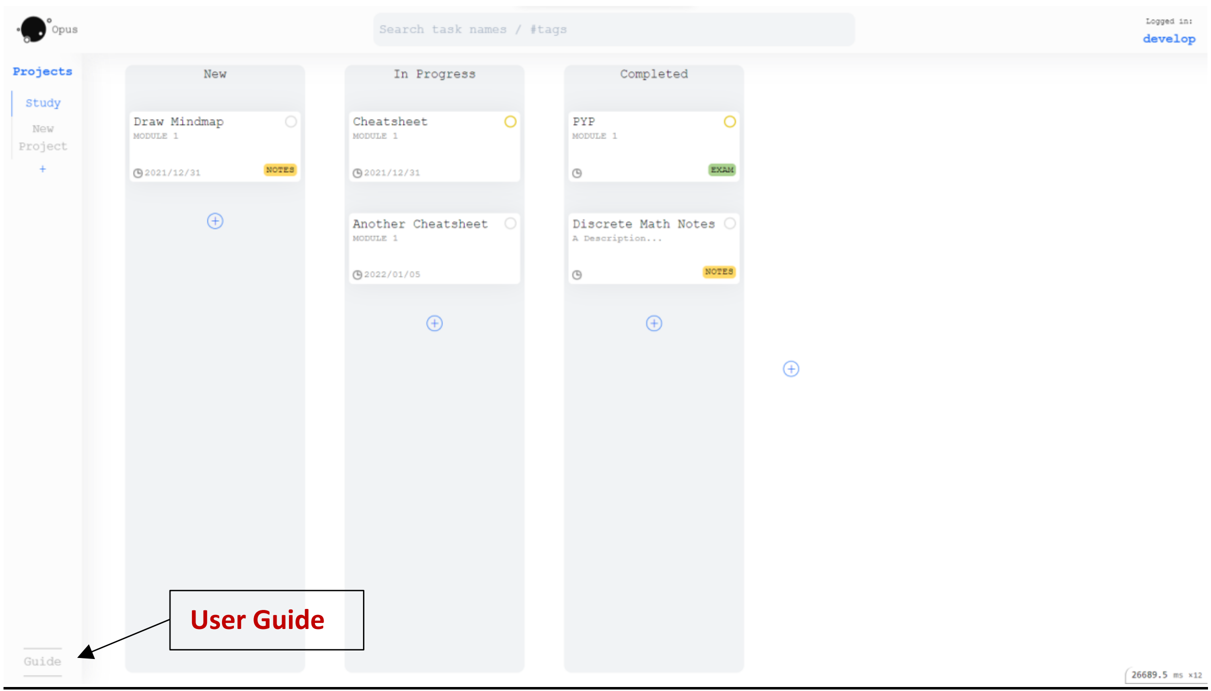


Email LastChristmas@santa.org
User GiveMyHeart
Password ToSomeoneSpecial
Another time...

Sign Up

Click ‘Forget Password’ to reset password for your account.

Task Page



New	In Progress	Completed
Draw Mindmap, MODULE 1, 2021/12/31, NOTES	Cheatsheet, MODULE 1, 2021/12/31	PYP, MODULE 1, EXAM
	Another Cheatsheet, MODULE 1, 2022/01/05	Discrete Math Notes, A Description..., NOTES

26689.5 ms x12

View the User Guide at the bottom left of the site for a comprehensive guide on how to navigate the app!