CS4530 Final Project: POST IT

Project overview

postIT expands on the previous setup by focusing on user interactions with the app. We wanted to make postIT an easy-to-use platform where users can easily create posts, quickly find what they are looking for, and interact with their communities in a meaningful way.

Through adding push and database notifications, users are instantly notified when their content is engaged with or their requests have been answered.

Communities are now also more interactive, through the addition of community moderators and level badges. Moderators and admins handle requests such as users joining a private community and flagged community content. Level badges create a fun leveling-up game in a community where members can earn points through posting new content.

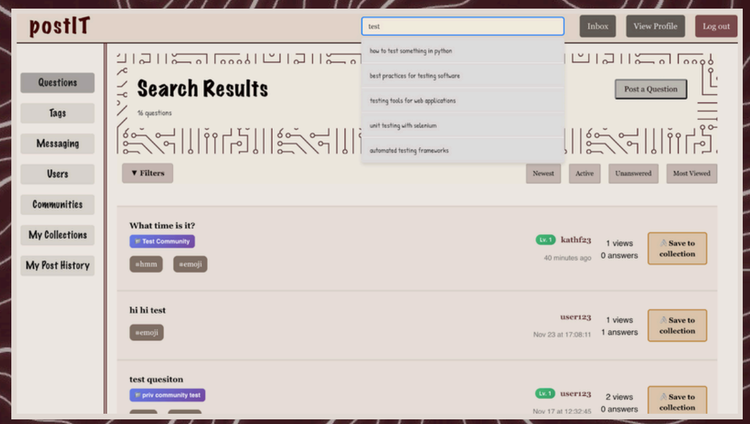
We improved user engagement with posted content through the addition of @ing users, threaded comments, and upvoting answers and comments.

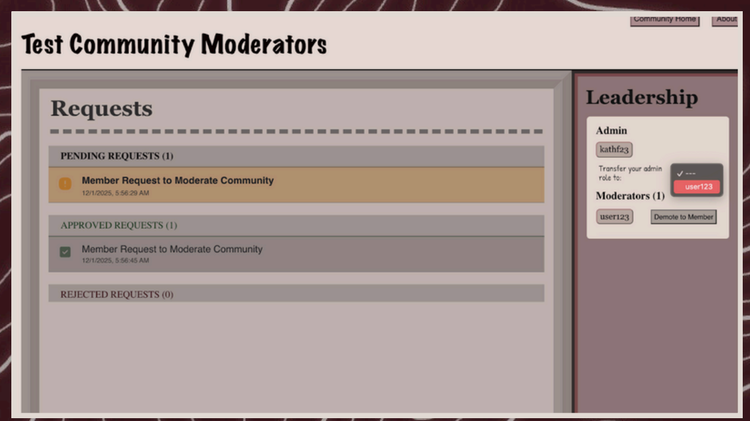
In terms of usability, our features allow users to access their previous posts through a dedicated, User History Page. Furthermore, through the use of autocompleted searches and additional category-specific filtering, sorting, and searching, postIT allows for efficient information retrieval.

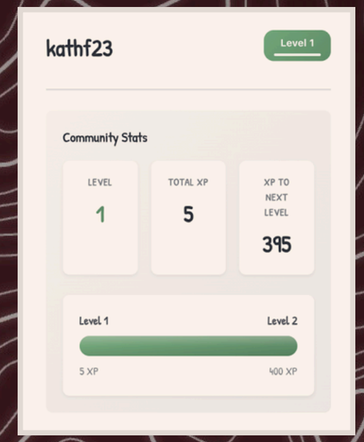
We also improved our security protocol, ensuring that users are properly authenticated and don.t have security concerns.

Future Work

In the future, we would like to include suggested communities and questions for users based on their previous searches and interactions, as well as suggestions based on the item they are currently viewing. This would also include, recent communities, and recent questions, tab in the side bar. We also plan to implement a bookmarking system, friend requests, and group chats. Lastly, we think it would be interesting to also include a trending posts section.







Tech Stack

The frontend is built with React and TypeScript, and the backend uses Express.js with MongoDB for data storage.

Authentication is handled through Auth0. We built a custom middleware that syncs Auth0 users to our MongoDB database. This middleware also handles migration of existing users who had bcrypt-hashed passwords by detecting their username and linking it to their new Auth0 account. Sessions use rolling timeouts, and the client shows a warning 5 minutes before session expiration. Real-time features use Socket.io. The user post history page needed to aggregate data from different MongoDB collections and sorts everything by date. We used MongoDB's aggregation pipeline for this and added indexes on username and date fields to make the queries faster when users have thousands of posts.

Push notifications are sent through Firebase Cloud Messaging using a service worker, and are then saved to the database for future access. For the frontend, we used a mix of CSS styling of the React elements and Chakras UI, along with a preselected color palette.

Group 112: Kathryn, Lea, Percy

Demo: https://cs4530-f25-112.onrender.com/g

Source: https://github.com/neu-cs4530/fall25-project-fall25-project-group-112