CS4530 Final Project: NUBoard

Group 309: Aayush Turakhia, Lawrence Osher, Jylah Bah and Jason Leon

Our Feature: NUBoard

In the original build that NUBoard was created off of, dubbed "Fake StackOverflow", users are able to post questions to a common forum where other user may answer them. They are also able to play games of Nim with other users and begin direct messages. The primary feature however, is the question-answer system that allows users to overcome blockers in projects they may be working on. In this state, the program does not support the capability of actually tracking the progress you make on a specific project, and users need to use external software to do so. What if users could track their progress through a native project-planning dashboard?

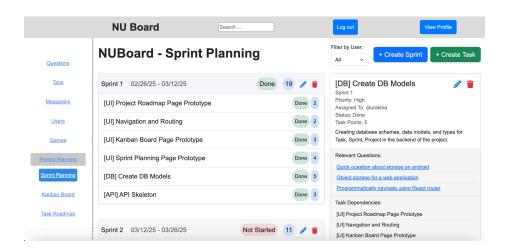
To remedy this, we developed a new features, called NUBoard, that does exactly that. It creates a Jira-esque project planning dashboard that allows users to create sets of tasks, assign them to sprints/backlog, and adjust their statuses as they make progress accordingly. We also have the ability to construct a roadmap of tasks using dependencies as edges.

Demo and Source

Our demo site can be found at:

https://cs4530-s25-309.onrender.com and our code can be found at:

https://github.com/neu-cs4530/spring25-team-project-spring25-project-group-309



Our Technology Stack and Design

We added Project, Sprint, and Task schemas, models, and types with the Task having relevantQuestions as a field that connects to the questions asked in the existing FakeStackOverflow architecture. Then, we created services, controllers, and corresponding automated tests with mockingoose and jest for each of our three new data models. On the client-side, we utilized React Bootstrap for clean and consistent styling, React DND Kit for drag-and-drop functionality, React Redux for application-wide state management, and vis-network for our Roadmap Graph. Redux came in useful for ensuring that we used consistent project data throughout the three different screens and so that our code was cleaner by not having to pass props and callback functions all throughout multiple components. "Prop drilling" can be error-prone and messy, and Redux helped to avoid this. We have a pipeline to run an automated test suite, and have deployed the software on render connected to a MongoDB.

Future Work

There are some proposed extensions to our product that could be built. First, support for multiple projects - we have made our code extensible for such a change, by having the projects endpoints currently return a list of projects, but the UI cannot support displaying multiple or switching between them. Furthermore, the ability to add attachments/documents via upload to different tasks could also be beneficial in adding context to each task.

