CS4530 Final Project Poster

Group 903: Kaylie, Roslyn, Natalia, and Nick

Our Features:

Our platform allows users to ask technical questions, explore solutions, and collaborate with one another. In the original version of the site, however, we noticed several gaps in the user experience. Some users with accessibility needs found it difficult to customize the interface in ways that made reading and interacting with content comfortable. Others who wanted to collaborate on code had no shared workspace to write or experiment together. And for users who encountered long or complex answers, it was often unclear where to begin or how to quickly understand what others had written.

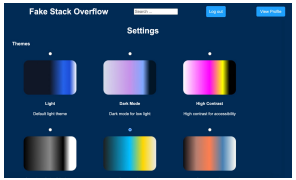
To improve this experience, we introduced a set of new features that enhance accessibility, collaboration, and learning. For accessibility, we now provide customizable interface options such as adjustable text size, color themes, and optional text-to-speech. These settings are organized into clear sections, making them easy to find and modify so users can interact with the platform in a way that fits their needs.

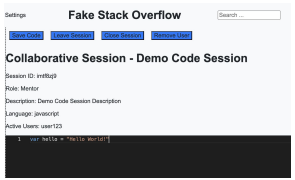
For collaboration, we developed real-time coding spaces where multiple users can edit code simultaneously, see updates instantly, and work together more effectively. Admins control who can join a session and can manage participation as the session evolves.

We also integrated AI assistance to help users understand content more easily. AI-generated answers and summaries provide quicker guidance, making complex explanations more accessible and helping users learn more efficiently.

Demo & Source:

Our demo site is available at https://fall25-project-903.onrender.com, and our code at https://github.com/neu-cs4530/fall25-project-903





Our Technology Stack & Design:

For our accessibility settings, we utilized React's Context API to manage global UI preferences, with persistence in both backend and local storage.

For our code-editor, we integrated Monaco Editor with Socket.IO for real-time synchronization. Each session is managed by the Session model, allowing data and permissions to persist across edits.

To support AI-generated answers, we added an AI service layer that interacts with OpenAI's GPT-4o-mini model. A controller uses REST endpoints with database schemas to track AI metadata

Our test suite features automated backend tests for service logic and controllers. Our frontend is tested through Cypress, which verifies accessibility settings, collaborative editing permissions, and session workflows. Our architecture is structured to support continuous integration.

Future Work:

- Let admins assign more granular roles with different editing or viewing permissions.
- Show which user is typing or editing specific parts of the code in real time.