CS4530 FINAL PROJECT

TEAM 902: PAOLO LANARO, MATTHEW ELLIS, STEPHEN WILLIAMS, JAKE DWYER

CodeFlow

Source Code ↗

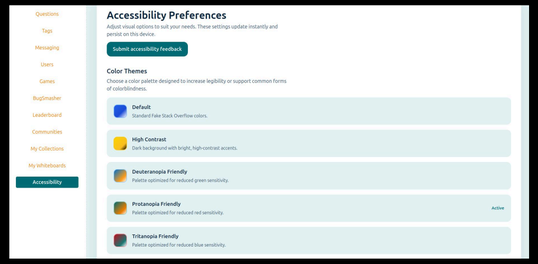
Deployment ↗

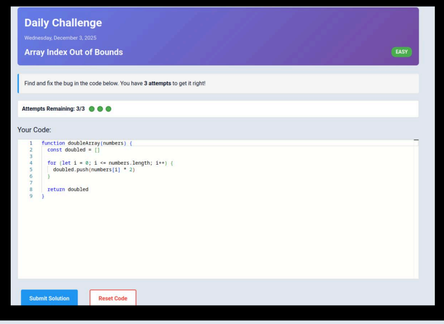
Our Features

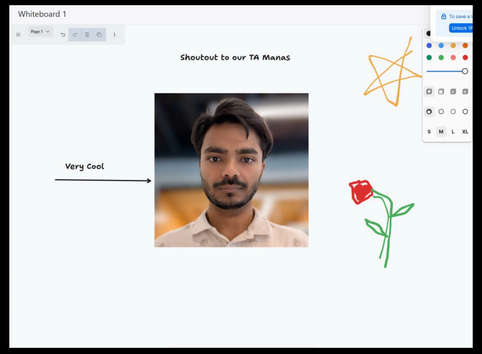
Our extended FakeStackOverflow platform focuses on collaboration, daily practice, and personalized accessibility. We introduced an end-to-end Accessibility Center where users can configure themes, font scaling, tooltip behavior, keyboard shortcuts, and full screen-reader support with live route announcements.

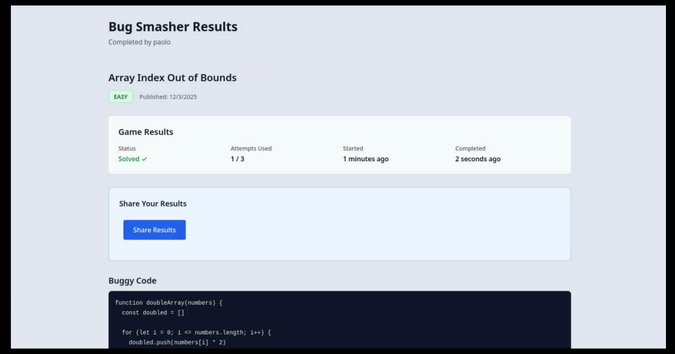
To make the platform more interactive, we built Bug Smasher, a daily debugging challenge that gives users a buggy code snippet, tracks their attempts, and scores their solution for accuracy and correctness. Users can share their results publicly and compare performance with others.

We also introduced real-time collaborative whiteboards, allowing learners to sketch, diagram, and solve problems together. Each board supports presence indicators, access requests and approvals, live syncing, and continuous snapshot persistence. Combined with an updated global Problems hub, the system encourages learners to practice, collaborate, and improve their debugging and problem-solving skills in a structured environment.









Tech Stack & Design

The system is built primarily with TypeScript, with a Vite and React client and an Express and Mongoose backend that share types for consistent validation. All API traffic passes through Swagger and OpenAPI validated routes.

Real-time collaboration is powered by Socket.IO, which manages whiteboard rooms, presence, and sync intervals. Whiteboard sessions use tldraw for canvas rendering and persist server snapshots every few seconds. Bug Smasher uses VM2 sandboxing to safely execute and validate user-submitted code within strict memory and time limits.

The frontend integrates a reusable socket provider along with global contexts for accessibility and user preferences. Settings such as theme, font size, and screen-reader configuration are persisted on the server to ensure a consistent experience across devices. The application follows a modular service and controller architecture, supported by dedicated testing coverage.

Future Work

Future improvements include expanding Bug Smasher with multiplayer races, streak tracking, and enhanced analytics so users can visualize accuracy trends over time. Whiteboards could support branching versions, time-travel history, and richer permission modes for classrooms or group study sessions.

We also aim to extend accessibility features with voice-driven navigation and higher-fidelity screen-reader cues.