CS4530 Final Project: ‘JobOverflow’

Group 209 : Ryan, Hailee, Abdurrahman, Minghui

Our Feature: JobOverflow

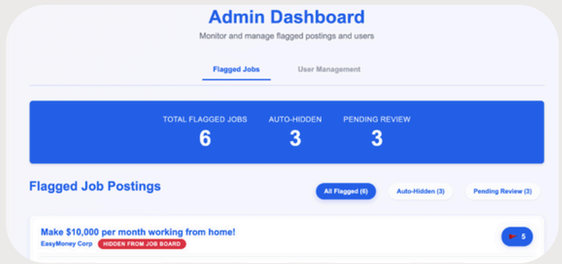
With the initial FakeStackOverflow we developed the front facing, and backend services to facilitate users to make collections and interact as a community. However, with only a subset of features implemented from the real StackOverflow we were met with a large swath of different options we could explore. Given that many of us are in the process of finding a job, we had some ideas to implement some topical features.

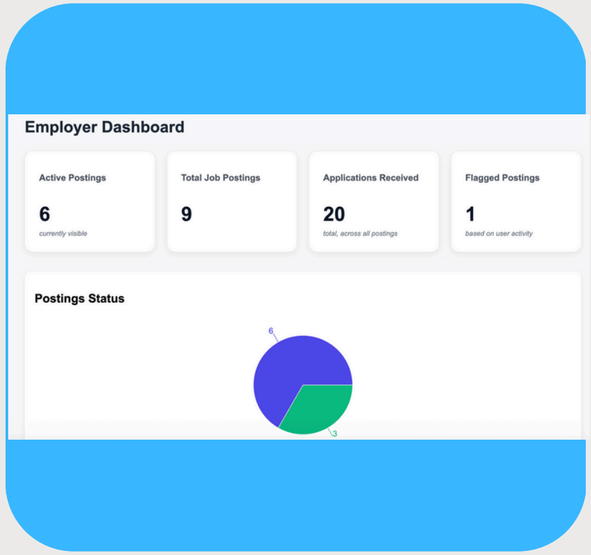
We developed a new feature to extend FakeStackOverflow into **JobOverflow** to incorporate a key part of the real service, with a job board which facilitates employers to post and customise jobs, and developers to apply to jobs with documents, taking ownership of their job seeking experience. To make sure all parties behaved, we also debuted a *Staff* role to audit, moderate, manage, and cleanup users, applications, and postings that don’t fit our community guidelines and fair practices.

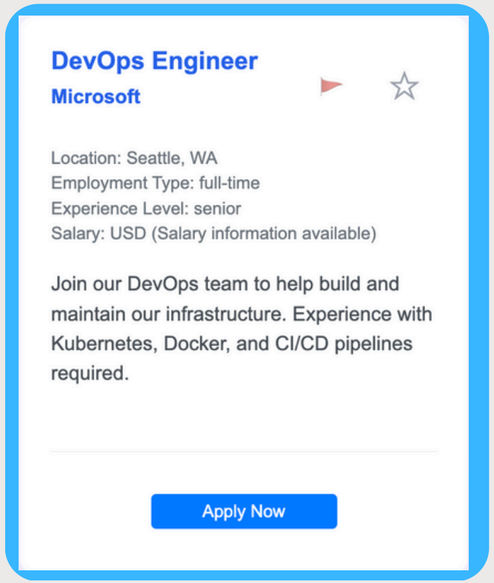
With these new features, JobOverflow evolves from a simple Q&A platform into a job-seeking ecosystem where developers, employers, and admins can coordinate safely and efficiently.

Demo and Source

Our demo site is available at:
https://fall25-project-fall25-project-group-209.onrender.com
and our code at: https://github.com/neu-cs4530/fall25-project-fall25-project-group-209.git







Our Technology Stack & Design

We used many technologies to help achieve our goal of making a functioning job board. Our backend is built with Node.js and MongoDB where we added schemas for job postings, applications, notifications and moderation flags, so developers, employers, and admins can safely share the same APIs. We implemented OAuth through Google Workspace, so user can sign in safely without us storing passwords. The frontend is written in Typescript/React, with separate views for all roles that all talk to the backend via REST endpoints for filtering, sorting, and status updates. CI/CD pipeline runs an automated test suite to check formatting and runs all the tests. Deployment of both frontend and backend is done using Render.

Future Work

We are currently utilising some shorthand practices to allow us to iterate frequently and move fast in development. In the future we would want to utilise a Relational Caching service, such as Redis, to streamline our caching logic to improve efficiency, availability, and privacy.

In addition, our current Express setup limits the file size that a user could submit for their application. While we’ve already expanded the cap for file sizes, future work for us could include making the user experience with file sizes to be more seamless.

Overall, these future enhancements will move JobOverflow closer to a refined platform that scales smoothly while staying secure, fair, and easy to use for every role.