Community Overflow

CS 4530 - Group 212 - Final Project

Anshul, Sumer, Kaushik, & Immanuel

Our Feature: Community Overflow

One of the major motivations for building out communities was due to the lack of engagement within the platform. Users did not have a way to communicate with others and share their thoughts with each other.

Group Chats

Members in a community are automatically added to a group chat. Users are able to react to messages, delete/restore messages, see active users, upload files, and see which users are typing.

Interactive Q+A

Users are able to post Q+A in markdown, receive notifications, view community stats, post anonymously, bookmark questions, receive a daily email digest. An LLM runs at midnight to group questions to a community.

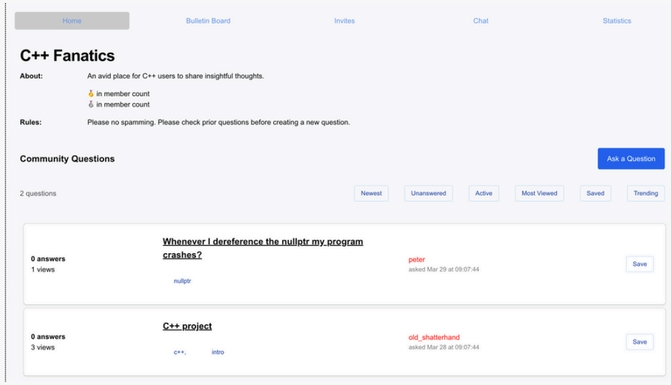
Communities

Users can preview/create/join a community and invite their friends, Upon joining, users can visit the bulletin board, voice chat, group messaging chat, and view main community statistics.

Demo & Source

Our demo can be viewed here.

Our source code can be viewed here.



Community Home Page

Our Technology Stack & Design

We implemented the Community Overflow features in the existing Fake StackOverflow codebase. Each of the communities are represented as a Community object which are dynamically created. Group chats, Q+A, bulletin board, notifications, video chats, and statistics all live inside of a community as their own objects.

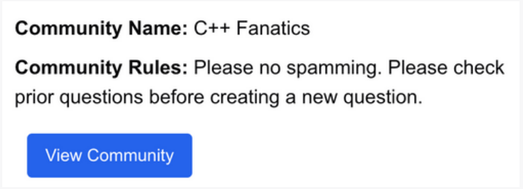
The majority of our frontend design is built in React with Chakra components in Typescript. The backend is built with node as our package manager. Live updates with group chats, notifications, and the bulletin board are handled with socket-io. The email digest is sent with the nodemailer API. Bulletin board is created with tldraw and the voice chats are done with Jitsi meet. Filtering of questions to an assigned community is handled with a Gemini LLM. Uploading images to the cloud is handled with Bytescale. Emails and the LLM are triggered via nodecron at midnight.

Our continuous integration pipeline runs an automated test suite on the frontend and backend components, and then deploys the site using Render.

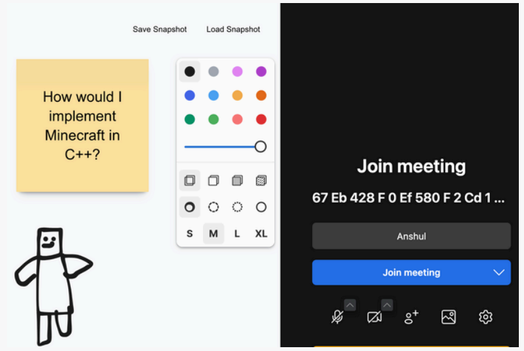
Future Work

In the future, we could work on revamping our UI to make it fully integrated and easier to use and see. For the chat messaging system, we could enhance the UI to easily integrate more features in the future. Some possible features are community-wide polls, sending and running source code, and allowing users to reply to specific messages to create a thread.

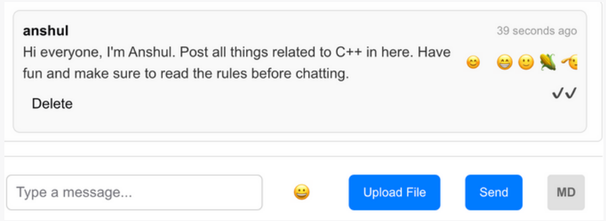
In terms of communities, we could allow for community moderation tools for the admin. They could be able to pin messages, mute users, or review flagged content. They could also be given access to a community analytics dashboard, which could give deeper insights into growth, engagement, and popular content over time.



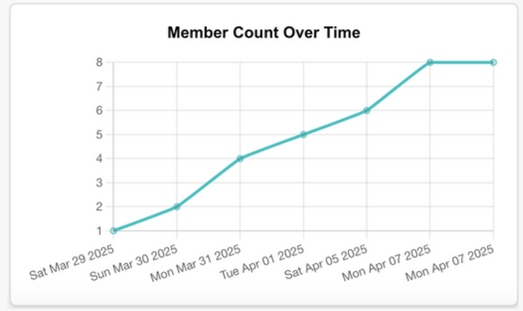
Join/View/Preview a Community



Create edits on the bulletin board & join the video calls



Post messages in the chat



View Community Statistics