ThreadBlend

Team members - Team 02:

Adam Wuth

Aaron Semones

Jake Mcknight

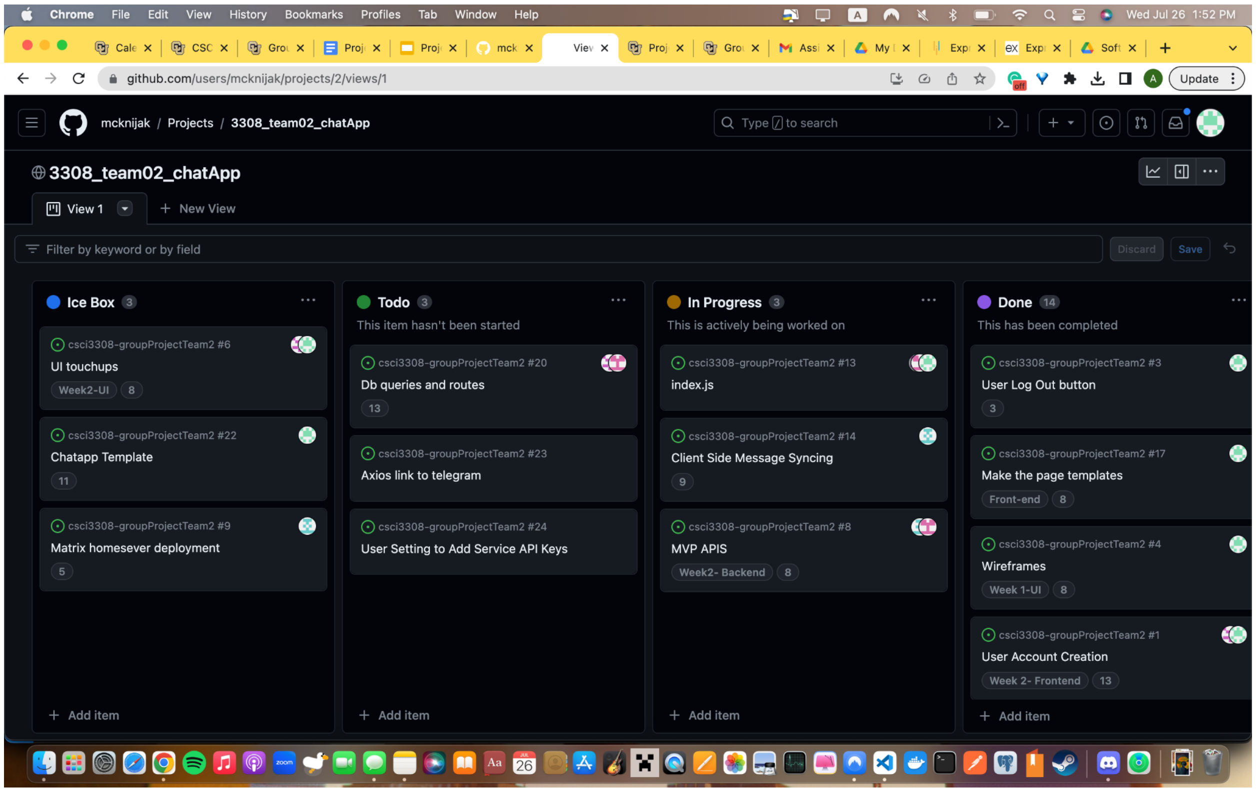
Henry Radovich

Project Description

ThreadBlend is a tool meant to help organize and stay up to date on the important conversations in a user's life by eliminating the need to toggle between multiple chat apps. The app is intended for people who are forced to check multiple sources to stay current on their conversations, even if they only use a certain app for a single conversation - eg keeping Facebook Messenger around solely for one group chat even though the user prefers to use another service. ThreadBlend is intended as a one-stop-shop for viewing chats. After setting up their account, a user will be able to pull in conversations from a third party app to send and receive messages on demand. ThreadBlend will sync with the conversation so that not all participants in the conversation need to be using ThreadBlend, nor are there any privacy concerns in regards to message storage. ThreadBlend doesn't store your messages; it is an interface for communicating across multiple chat platforms so that a user can see what's important to them in one site - rather than needing to switch between many sites. User accounts will store standard, basic identifying information about a user, and also any relevant API keys needed to interface with supported third party apps. This one account can then be used to send/receive messages for any supported third party apps - currently only Slack. Also stored is a list of conversations the user is importing into ThreadBlend, so past or inactive conversations aren't cluttering the interface.

Project Tracker

https://github.com/users/mcknijak/projects/2/views/1



mcknijak / Projects / 3308_team02_chatApp

3308_team02_chatApp

View 1 | New View

Filter by keyword or by field

Ice Box 3

- csci3308-groupProjectTeam2 #6 — UI touchups — Week2-UI — 8
- csci3308-groupProjectTeam2 #22 — Chatapp Template — 11
- csci3308-groupProjectTeam2 #9 — Matrix homesever deployment — 5

Todo 3 — This item hasn't been started

- csci3308-groupProjectTeam2 #20 — Db queries and routes — 13
- csci3308-groupProjectTeam2 #23 — Axios link to telegram
- csci3308-groupProjectTeam2 #24 — User Setting to Add Service API Keys

In Progress 3 — This is actively being worked on

- csci3308-groupProjectTeam2 #13 — index.js
- csci3308-groupProjectTeam2 #14 — Client Side Message Syncing — 9
- csci3308-groupProjectTeam2 #8 — MVP APIS — Week2- Backend — 8

Done 14 — This has been completed

- csci3308-groupProjectTeam2 #3 — User Log Out button — 3
- csci3308-groupProjectTeam2 #17 — Make the page templates — Front-end — 8
- csci3308-groupProjectTeam2 #4 — Wireframes — Week 1-UI — 8
- csci3308-groupProjectTeam2 #1 — User Account Creation — Week 2- Frontend — 13

Video Demonstration

VCS

https://github.com/mcknijak/csci3308-groupProjectTeam2

Contributions

Adam

I was in charge of the front end. I made the templates, except for the chat.ejs, and did most of the functionality endpoints. I also worked on the index.js file, and troubleshooting errors with the docker yaml to get our site to a point where it would run and display on local host. I then worked closely with other group members to add forms and endpoints to get features, such as connecting your api and signing key, working. A large amount of my time was spent resolving errors where people's code didn't quite connect or pages weren't displaying the correct values.

Aaron

I managed the database component of the project. I created the database erd model, implemented the model in sql, and handled any sort of issues of the database raised by any other team members. I added and altered tables and columns as needed. I also did minor work on the index.js for basic routing capabilities and handled dev-ops/deployment for the team. I created and bugfixed the dockerfile and ensured that the image worked locally and then once deployed to the cloud. Finally, I wrote up most of the project presentation

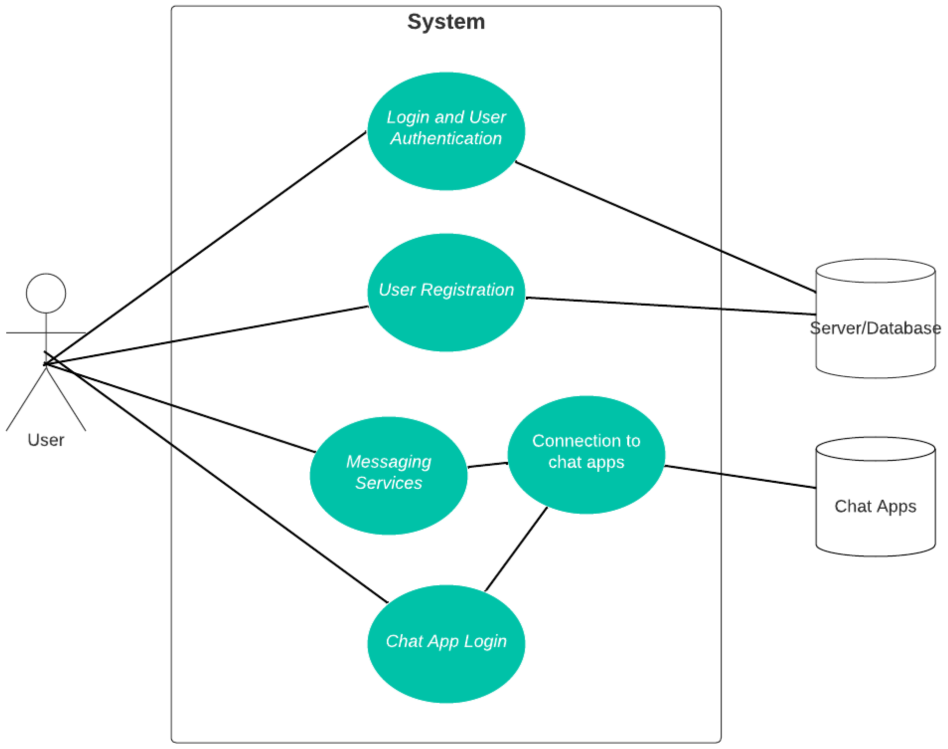
Jake

I worked on the third party APIs that were needed to access different chat services and also built the chat page that is used to display the messages within each chat. To start, I researched different methodologies of integrating chat services and found one SDK that had what we were looking for, but unfortunately the bridge tools it provided were only modular within its own ecosystem. Then I pivoted to working with React components since they live refresh, not realizing immediately the project specs said we couldn't use such frameworks. Then, I pivoted to working with the socket.io node module to force client / server communication. Unfortunately, I think I tended towards thinking about complex solutions and missed some more simple ones.

Henry

During my involvement in the project, I played a significant role in the development of the index.js file, where I focused primarily on working with APIs. My main areas of expertise revolved around enhancing and fixing various APIs, with special attention given to the sign-up API and multiple Chat API functionalities. Additionally, I did minor work in the management of the database, where I worked with tables to ensure the smooth functionality of the APIs. I mostly used Postman and the localhost to debug and fix errors with the API's.

Use Case Diagram



Test Result:

Deployment:

https://threadblend3.azurewebsites.net

Deployment details

To deploy the application we first ensured that the app ran locally within a container. We started the project within a container so from the start we knew that our docker.yaml was correctly formatted. Next was the task of ensuring the app ran locally in a docker image. We researched how to write dockerfiles and wrote a dockerfile from scratch. However there were pathing issues and issues with copying modules over from the dev environment to the deployment image. So we shifted the pathing in the dockerfile and the repo to ensure it matched. Then we added a small script to reinstall the npm modules on the server. Once the application was deployed we had issues with accessing the database for database calls. We noticed that the dockerfile was automatically ignoring .env files so we added the database info to the top of the index.js file to combat the issue as we lacked the time to bugfix the dockerignore.