Concept Document

Authors:
Steven Berdak, Juan Lopez, Stefin Racho, Cameron Martinez-Ramon, Connor Schooley, Hemant Singh

We propose a general purpose chat application that provides ways for people with shared interests to interact with one another. The application allows users to create chat rooms and tag them with keywords which will facilitate other users finding those chat rooms and joining them. Chat room state is saved to and synced from the web server, and users are able to interact via chat in a persistent chat room with options for filtering messages based on various criteria.

The project will contain a basic chat room interface with a user list and buttons that allow users to create new chat rooms, a search feature that users can use to find chat rooms based on tags that are supplied via the chat room configuration. Within a chat room a search feature will allow users to filter chat rooms based on read status, date and word token matching. Chat room creators will be able to moderate their own chat to mitigate unwanted activity from other chat room users.

The project will allow persistent connections via websockets for immediate state updates to the chatroom when new messages arrive. This could result in many different connections all existing at the same time and thus it is important to design the web server in such a way that it can still serve normal website and API traffic. To do this we will use multithreading to allow the web server to handle both simultaneously with minimal impact on performance.

High-Level Requirements Document

Authors:

Steven Berdak, Juan Lopez, Stefin Racho, Cameron Martinez-Ramon, Connor Schooley, Hemant Singh

Here at SEG1 Incorporated we pride ourselves on being number one! We have determined that our internal communication is sub-par and thus we believe that it is necessary to create an internal chat application for all of our 150,000 company employees to use for productivity, organization or simply bragging about whose team won the game last weekend. This document will provide a general outline for the application as we have determined based on the needs of the company.

We have found that existing chat applications do not meet our criteria, either for being too technical and clumsy, or else being too general and not providing ways to join channels that are dedicated to opposing teams so we can drop into the channel and dunk on them when our team was victorious over theirs. Therefore we need a chat application that allows users to create chat rooms, apply keywords and lets others join at their own leisure. The chat rooms should have basic chat room necessities such as user lists, chat room moderation and real-time communication.

An important feature that we feel is vital to the success of the chat rooms is the ability for chat room creators to add keyword tags to the chat room that can be subsequently used to locate chat rooms of shared interests. Given that chat rooms should not exist without a purpose, at least 1 tag is mandatory for creation of a chat room. Also, chat rooms should be able to be moderated by the creator of the chat room so that chat room creators can be self-administering to remove any company overhead that may be incurred having to hire employees to monitor them.

We also believe, to prevent misuse by disgruntled employees, that all users should be validated and authenticated using company email. Users should be able to create a basic profile, including name and a profile picture. Users should also be able to search and join chat rooms based on the keyword tags. We believe in transparency, so any employee should be able to join and remove themselves from any chat room as they wish. We also believe it is important that users are able to search through chat room message history by date, read status or words

to be able to recall messages that may be important later, such as who placed bets on which teams.

Technical Specifications Document

Authors:

Steven Berdak, Juan Lopez, Stefin Racho, Cameron Martinez-Ramon, Connor Schooley, Hemant Singh

The technical specifications for the application are as follows:

Backend:

1. The backend will be written in Java using the HttpServer, HttpExchange and the HttpHandler base class.
2. Authentication will be handled with Firebase Auth for both user validation flows and authentication for API calls using JSON Web Tokens (JWT).
3. The app will utilize an SQL database for handling persistent storage of information.
4. Websockets will be used to handle persistent connections for connected users who are participating in shared communication functions of the app.

Frontend:

1. The front end will be written in the React framework.
2. The front end will make use of modern web styling for delivering a rich content experience to the users utilizing an UI framework such as Tailwind or Material UI.
3. An email validation flow will validate new users and facilitate onboarding users into the platform.
4. A login flow that accepts multiple login methods (social sign-on, email password, etc).
5. Darkmode/lightmode selector for changing the display mode based on user preference.

Features:

1. Validated and authenticated user accounts.
2. User profiles that include a user name and a picture.
3. Chat room creation and moderation that can be performed by the chat room creator.
4. User roles to distinguish between a chat room member and the chat room creator/administrator.
5. A user list for determining who is present in a chat room.

6. Read receipt status of messages that can be used for both direct viewing on messages and also filtering.
7. A search feature that can search a specific chat room's content based on criteria such as date, read receipt status or keyword tags.

Testing:

1. Functional testing will be used to verify that the application meets the specifications outlined.
2. Regression testing will be used to prevent software regressions from one version to the next.
3. Load testing and stress testing to ensure that the application can maintain high throughput with multiple concurrent connections.
4. Unit testing to ensure functions behave as expected for given inputs, with careful attention to extreme values and edge-cases.
5. Integration testing to ensure that various parts of the application work together as expected.

DevOps:

1. Github will be used for version control, continuous integration and regression testing.
2. Asana will be used for Sprint planning and also as a Scrum board to ensure that tasks are completed on schedule.
3. Zoom will be used for virtual meetings to evaluate current project status and determine tasks to be completed during the next Sprint.