CY CHAT APP

Done By: Hassan Medhat, Rawan Osama, Ahmed Qasim

Contents:

Sizing:

- Total Number of Users who will use the chat is 100 User
- Each Client Will use the App to send an average 8,640 Messages per Day = 10 Message Attachment, 8,630 Text Message
- If we considered that the Maximum size of the message is 1 KB
- If We added Later on Attachment messages for our system. The Image Size will be 10 MB = 10000 KB
- We need to keep the messages for 1 year, then it will be deleted.
- Total Sizing Needed:
- 8,630 Text Message * 1 KB * 100 User * 365 Day = 314,995,000 KB = 315 GB
- 10 Attachment Message * 10000 KB *100 User *365 Day = 3,650 GB
- Total Sizing in Giga Bytes: 315 + 3,650 = 3,965 GB approx. = 4,000 GB = 4 TB. As Margin we may Provide 5 TB for [database indexes, metadata, and log files].

Load of the System in One Hour, and On One Day [Average Message/Sec]:

- Assume Each User Sends 4 Messages in 40 Sec.
- If I have 100 User
- 100 * 4 = 400 Messages In 40 Sec
- 400 Message / 40 Sec = 10 Message Per 1 Sec for the 100 User. [10 Message / Sec]
- The Maximum Load in 1 Hour
- 1 Hour = 60 min * 60 sec = 3600 sec
- For 100 User in 1 hour = 10 * 3600 = 36,000 Message / 1 Hour
- For 100 User in 1 Day = 36,000 * 24 Hour = 864,000 Message / 1 Day

System Design Consideration:

- As Per my Calculation I may have Bottle neck for my File Upload.

- Hence, when we design our system technologies that support chat Application. the system technologies that we use should support sending 10 Messages / Sec For 100 User.

- Assuming that at peak time the system may send 5 Attachments with 10 MB. 5 Attachment Messages with 10 MB it will mean 50 MB for the File Upload for each file per second as Peak Throughput. Hence, to ensure the System Performance I will need to Have SSD Hard Drive to Store the Upload of the Files as Separate File System and the Location of this File System will be stored in the Database.

- Moreover. When we think about the Technologies that we may let our system work with. It should be libraries that supports the Asynchronization I/O.

- As we have here Attachment Upload to the SSD Hard Drive or external file system, and the data stores successfully into the Database. The System Should accept the asynchronization

process to not let the system crash or hang until it receives the confirmation that the process is done. For example, the system should accept the data provided by the user for attachment for example, and it will take its time till it is uploaded and stored in the SSD Drive, the user should not feel or see there is hang in the system until it finishes its write process.

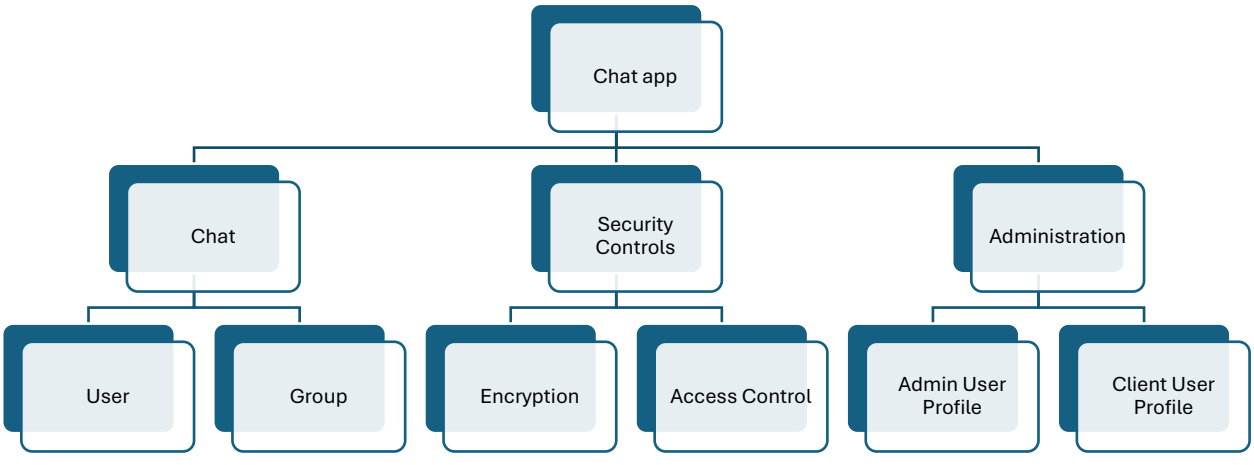
- The System will Accept the Attachment and it will notify the user back that the Data Uploaded Successfully. However, it takes time in the background till the data be uploaded to the SSD not the RAM.

- Accordingly, and as we are willing to have our system to work as Web Application, we need to depend on Technologies that the support the following:

    - Lightweight Web Frame as it only works for 100 User & and Support Asynchronization.
    - Database that Could mount data up to 5 TB.
    - SSD Hard Drive that has high Writing Processing and Enough RAM Specs.

- There For in our Design we will Depend on the Following:
    - FAST API as Web Framework for our backend.
    - FrontEnd based on HTML, CSS, TailWind UI Technologies and JavaScript.
    - Database That Stores Data up to 5 TB [SQL database] => Storing metadata (sender, timestamp, message text and the file path).
    - SSD Hard Drive in case I need to Upload the Files that were Provided by the Backend to be stored.

Domains of System Design:



User Stories:

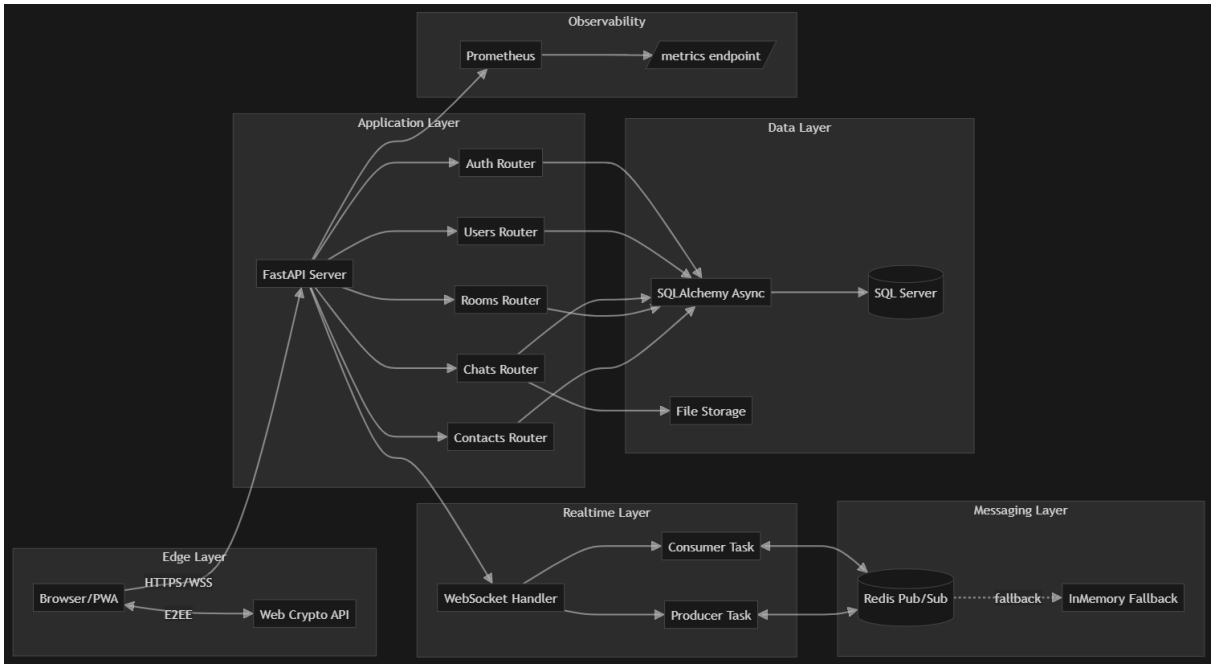
Users:

- As user I want to create an account so I can access the application
- As user I want to send end to end encrypted message so I be sure the messages are confidential and accurate.
- As user I want to create a group chat so I can manage and send messages to different users at the same time.
- As user I want to be able to attach different media files as attachments and be able to share these attachments with the users individually and within groups.
- As user I want to join groups so I can send messages to different users at the same time in an existence group.
- As user I want to leave chat groups to not receive any more messages from specific groups anymore.
- As user I want to see my recent chat messages with individual users and the chat messages in the groups I joined.
- As user I want to have multiple application appearance so I can swipe between.

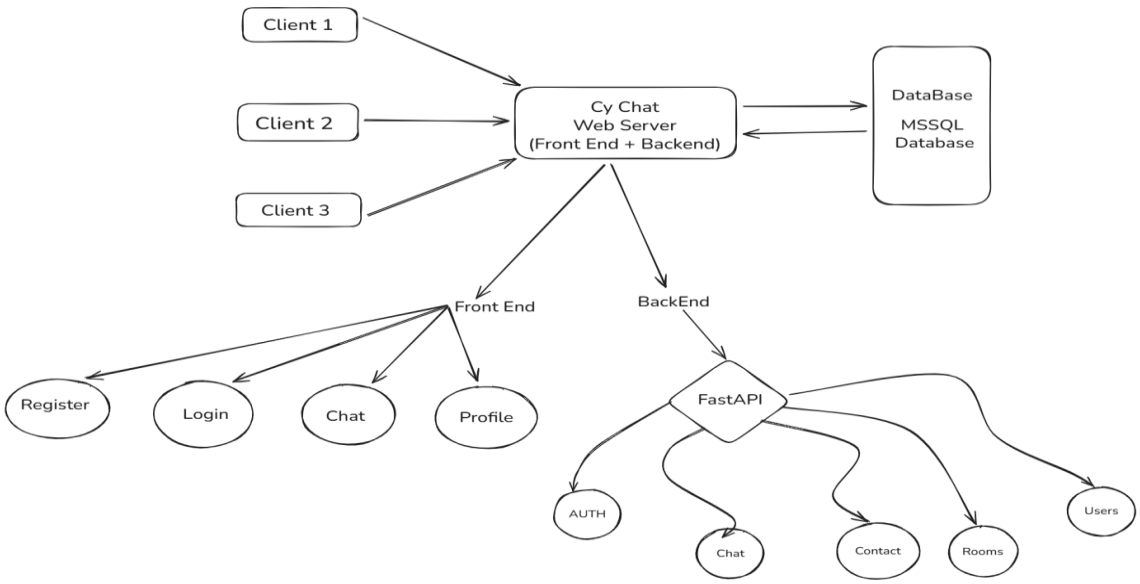
Admin:

- As Admin of the Application, I need to be able to check the users, groups created on the system.
- As Admin of the Application, I need to create groups for the users within the Application.
- As Admin of the Application, I need to edit the groups and be able to control the users who Join or banned from the groups.
- As Admin of the Application, I need to control the Pipeline of the new features within the Application.
- As Admin of the Application, I need to be able to add more users to any group.
- As Admin of the Application, I need to be able to specify Admin for each Group who is able to create, delete users from groups, and delete messages within the group.
- As Admin, I will need to delete messages and physical SSD files older than 1 year to ensure we stay within our resources limits which is 5 TB storage limit.

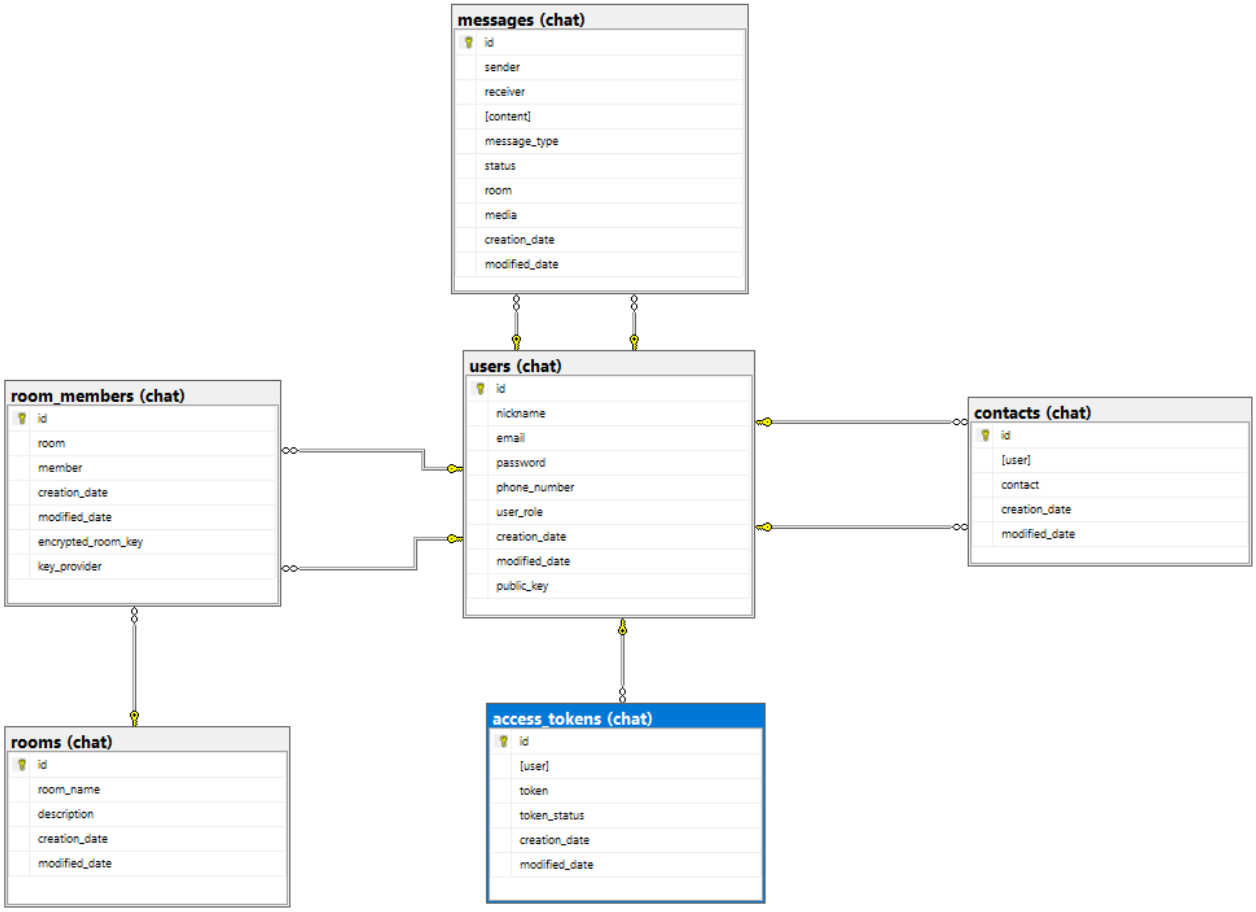
System Architecture:



System Diagram:



Database Design:



Security Model

To secure our chat app we need to **define what our app store**. In our database we store emails, passwords, messages and attachments which need to be secured. We need to set our security models that keep these assists protected.

Desired Security Model (controls) aspects that should be covered to achieve CIA

- Authentication ➔ Done without MFA
- Authorization and User roles policy ➔ Done
- On group chats
- On database access
- Users only access their data
- Data protection
- Passwords stored hashed ➔ Done
- Only the database admin can access the database ➔ Done
- Messages are stored in database encrypted ➔ Done
- The attachments are stored on file server with proper access ➔ Done

- HTTPS for transmission ➔ Not Done
- End to end encryption for messages ➔ Done
- Inputs and attachments upload security validation ➔ Not Done
- Penetration testing for the application according to OWASP ➔ Not Done
- Logging to trace errors and monitor suspicious behaviors ➔ Not Done
- Add infrastructure security controls. ➔ Not Done

Current Security Implementation

- We Provided User Authentication for our Application.
- No User should add his chat peer first to be able to receive his messages, otherwise it will keep him in the other messages.
- The applications run locally so we don't need HTTPs used till now [Beta Version].
- We kept Data protection layer by encrypting the messages as End-to-end encryption for messages between the clients.

Weaknesses "still not implemented"

- We don't have different replica of the database. Hence, this considered as Single Point of Failure. We need to have multiple databases
- There is proxy node that work as load balancer ahead the backend side which is (nginx for example)
- No logging for the system, we didn't create log file.
- Administration to manage the app to be able to create groups, users, and set their privileges.
- If we deployed this Chat Application in real world Environment we need to apply infrastructure security control.

Possible improvements

- Last seen service
- Multi factor authentication on Login page
- HTTPS for transmission by adding proxy node like nginx before the backend
- Files to be saved encrypted in its datastore.
- Inputs and attachments upload security validation.
- Plan our System to be more scalable to accept messages and connections from more than 100 Users.
- deploy all the Services Components in the containerization environment to be easily deployed in production.
- Create administration portal to manage the users
- Create users role
- Accounts can be deactivated and blocked
- Delete the users and messages and rooms
- Leave room

High availability

- Create Replica for database
- Deploy load balancer for your system to load balance the traffic load in case we later on scaled our production system to accept connection and messages from more than 100 users.

Testing

We agreed to have our Testing Done Based on the following Test Cases:

- Unit Testing.
- Regression Testing.

Unit Testing:

- Testing if each user has capability to Register himself to the system or not.
- Testing of each user able to login to the system with the Email and the Password he registered with.
- Testing If each user able to add more contacts within his circles to chat with.
- Testing if each user able to create new group, or get himself out of the group.
- Testing if each user able to Join existing group or not.
- Testing if each user able to edit his info on the system or not.
- Testing if the User able to logout from the system or not.
- Check if the system directs the user to the chat page and application successfully or not once, he authenticated and logged in.
- Check if the user still able to send attachments or not.

Regression Testing:

- Testing if we have directed the Authenticated user process to the chat page successfully and the user still able to login successfully or not.
- Testing if the user still has to chat with other individuals successfully once he Created Group chat or not.
- Testing if the User still able to login to system once he logged out successfully.
- Testing if the user still able to send messages even once he sent attachment.
- Testing if the user still able to send messages successfully once we added the end-to-end Encryption.
- Testing of User still able to login successfully once he updated his Profile with his new updates.

- Testing if the user still able to create new groups successfully even if he leaved groups.