

Team Chat App : Deliverable 4 – Project Phase 2

Requirements

Phase 2 Implementation Requirements:

Server Management :

1. Customize your server: Implemented server customization which is a core functionality while dealing with the app. Where the admin gets to change the name of the server and the image of the server. This feature is only done by admin and he can do this by clicking on the server settings on his server drop down list.

2. Delete server : Deleting the server is one more additional functionality provided to the admin of the server, where he can delete the server which he created, and when deleting the server all the channels and members who have joined to the server would be removed from the server and even in the database along with deletion of server. Deleting a server will only result in deleting the specific server which is deleted out of all the servers present in his dashboard.

3. Leave server : A guest or a moderator has the option to leave the server when ever they want, this option is present in their drop down list when they click on their server drop down. We need to note here that the admin of the server would not get a chance to leave the server he only gets the functionality of deleting the server. User leaving from the server will remove the access to that user from the server, like no one in the server can access them to make direct communication and they will even be removed from the channels list out of all the channels they are present in. And whatever the direct communication happened with the user who left the server that all chat will be removed from the server. And even all the message he has posted in the group channels would be removed if he gets to leave the server.

Search functionality :

4. Search functionality : Search feature is crucially a important feature to have in our app. It helps the user to navigate to the channels and server members quickly by searching their name in the search bar provided in the app. It displays all the channels/members present in the server initially, and need to show correct search results with respect to the entered text, after entering name, and navigate to specific channel/member after clicking on the channel/member.

Manage member :

5. Admin Member kicking : Kickout out the members of the server is a very prominent and must needed feature when dealing with app with more members who are not fit or abusive or who misbehaves in the servers of the app. This feature is only provided to admin of the server. So basically the user needs to get kicked out, and all the members of the channel should not be seeing the kicked out member and even that person needs to be removed from the channels and all that chats related to that person will be removed from the server.

6. Admin Role assigning : Assigning roles in the server helps distribute power and responsibilities of the server. The admin of the server can change the roles of the members present in the server to either "Guest" or "Moderator" by accessing the member management option in the server drop down menu. Given are the detailed specification of each role of the app :

"Guest" : He is allowed to message with emojis or file share in channels/direct message, and he can delete or edit his own messages, and he gets option to leave the server after click on his server drop down menu. And he has access to his manage his account by deleting, editing his account and he has even access to change the UI Theme. He can even search the server members/channels.

"Moderator" : He has the additional functionality to delete any message in group channels when compared to "Guest".

"Admin" : Additionally he is allowed to create/delete server, invite friends to the server, manage members by assigning roles or kicking them out of his server. Server setting by editing the name and image of the server while comparing to the "Moderator".

Chat message/ File Sharing deleting :

7. Admin & Moderator chat message/file delete functionality : Deleting any message / file sharing from the server should be possible when you have the role of either "Admin" or "Moderator". The message or the file sharing will be deleted when clicked on delete button. This can be only possible in the group channels, even the admin has no access to the private direct communication chatting of members present in the server to read or delete them so as the moderator can only delete from group channels.

Phase 2 development deliverable plan has been changed a bit :

1. Removed Tasks : We are planning to not include shared calendars and task management features to support team coordination and project management with the chat application, **Reason :** as we think it is not relevant to the team chat application, and this app is purely for the purpose of chatting through text, audio or video channels, or through direct one on one communication or group channel communication.

2. Features Added : We have added the features of Delete / Leave server according to the specific role in the server. And have added the feature to customize the server by editing the name and image of the server. We have added the search feature in the server so as to navigate server members and channels quickly. We have additionally added the role assignment in the app like admin assigning user with either a role of moderator or guest so as to diversify the power and responsibilities of the user in the app. And we have even added the chat message/shared file message deleting according to the role of the user in the app, which will give additional power of the users with specific role. **Reason :** All these features which are added adds more value to the app with chatting and communication capability.

3. Post phone features : we are planning to include Video / Audio calls channel in the phase 3 of the development cycle, **Reason :** as our schedule is a bit busy for phase 2 development cycle.

Modified set of requirements for other phases:

Phase 1 Updated Implemented Requirements (Done):

1. Real-time Messaging & File Sharing: Implemented a core messaging system supporting text and file messages across different chat settings for one on one and inside a general group setting, utilizing Socket.io for seamless, real-time communication.

2. User Management with user signup, signout, delete and signin the user account via Clerk:

- Integrated Clerk Service API which gives the built in functionalities for all streamlined user authentication and authorization, enabling users to authenticate and register using their Google accounts, thus prioritizing security and user convenience.

3. UI Customization and Themes:

- Introduced customizable UI themes, specifically implementing dark, light and system default mode, enhancing user experience by allowing personalization according to user preferences using next js ui components.

4. Invite Code Functionality for Group Channels for friends invite:

- Developed an invite code system allowing admin to invite his friends, focusing on easy expansion and collaboration within group settings where we can invite friends through the invite code system to the friend list channels.

5. Create server:

- Implemented create server functionality for making the app functional with group channel, one on one communication, invite friends and file sharing. Where the admin has the capability to create his server and invite his friends to communicate.

Phase 3 Updated: Testing, Final Review, and Handover (Apr 9 - Apr 29, 2024)

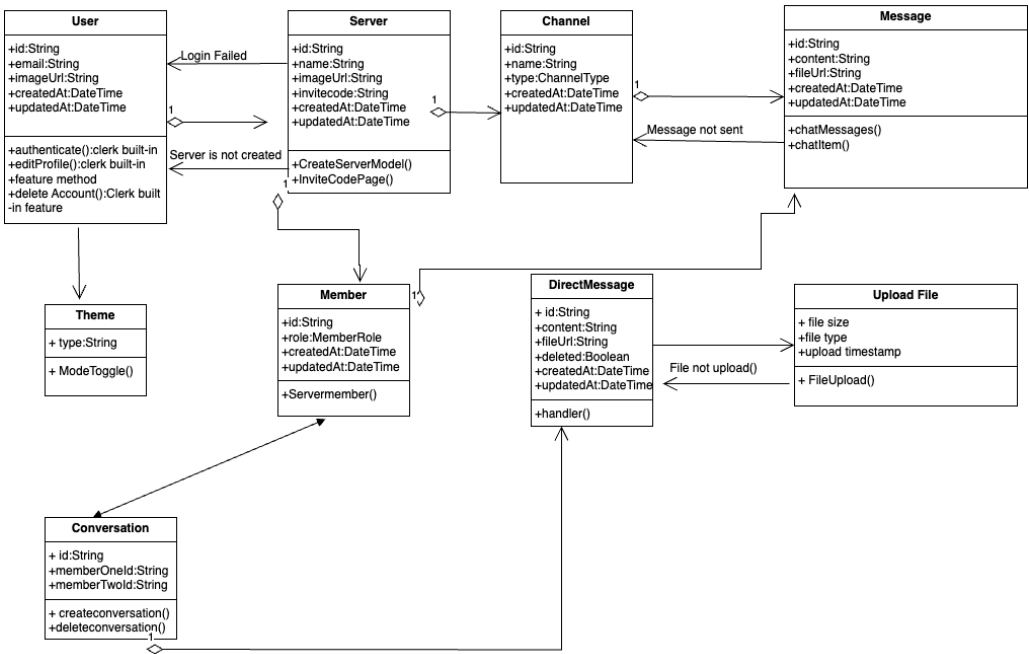
1.Comprehensive Testing: Execute a series of usability testing, unit tests and other tests to ensure each component functions as expected and the system as a whole operates seamlessly.

2.Security Audits: Conduct security assessments to identify vulnerabilities, applying fixes and enhancements to ensure robust protection of user data and interactions.

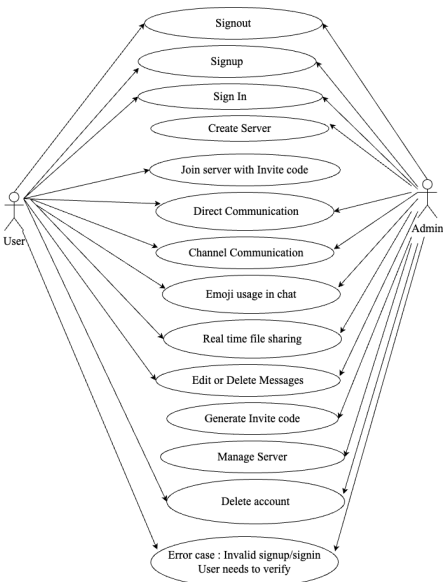
3.Video and Audio Calls: Leverage WebRTC for implementing direct peer-to-peer communication channels, offering users enhanced interaction options beyond text.

UML design

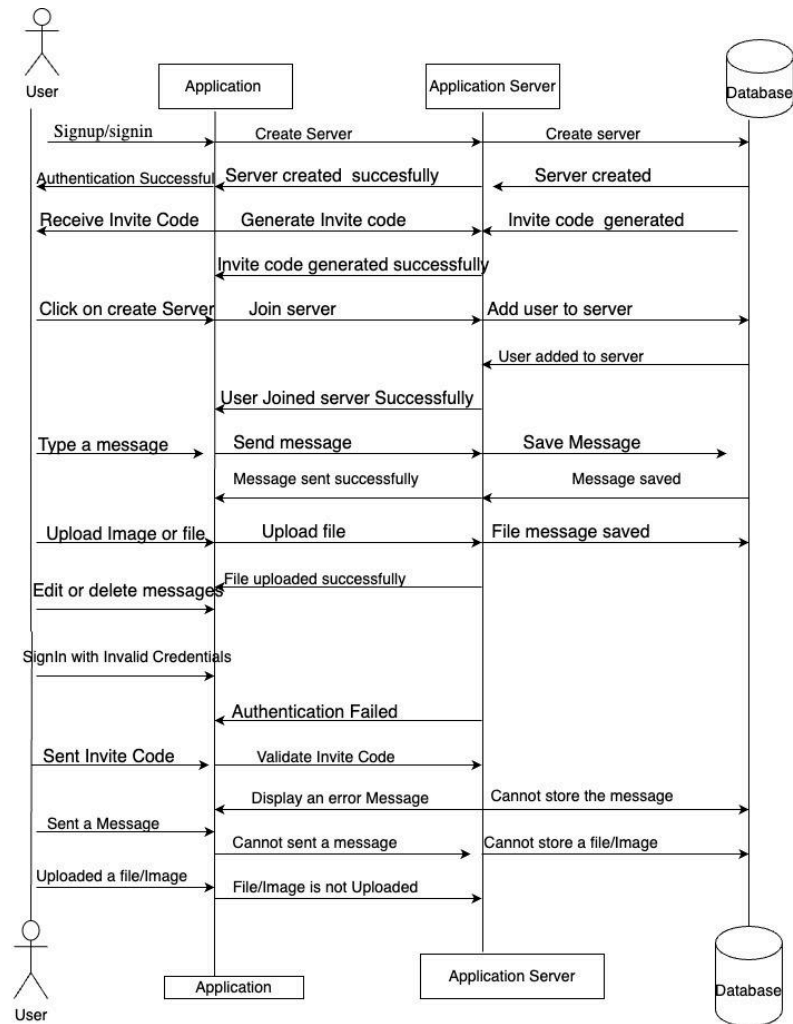
The following are the various UML diagrams for phase 1: **Class diagram**



Use case diagram



Sequence diagram



Test Cases

User Registration, authentication, signout and deletion via Clerk:

1. Test User signup through clerk authentication service:

Description: This test verifies if a user can successfully sign up with his google account.

Input: User directly sign up with his existing google account.

Expected Output: User is sign up successfully using clerk authentication service.

2. Test User Authentication sign in and sign out:

Description: This test verifies if a registered user can sign in securely and sign out after signing in to the app.

Input: User signup using google account and signs out using signout option.

Expected Output: User is authenticated successfully and granted access to the app if he signs in and sign out when he clicks on sign out after signing in.

3. Test Invalid User Authentication:

Description: This test checks the system's response to invalid login credentials.

Input: User logs in with unregistered google account.

Expected Output: User receives an authentication error message and is not granted access.

4. Test user Delete action his profile via clerk services:

Description: This test checks the app response to delete user from the app.

Input: User clicks on delete account option provided by clerk.

Expected Output: User account gets deleted.

Create Server:

5. Test if the create server is working fine:

Description: This test checks if a server can be created within which channels can be created in the future for communication and one on one communication.

Input: User clicks on create server options providing server name and image.

Expected Output: New server that is specific to the user gets created where the created user is now the admin, and he can invite his friends for one on one communication with general channel being common.

Invite Code functionality for friends:

6. Test Invite friends code url generation:

Description: This test checks if the admin of the server can generate invite code to invite his friends to his server.

Input: Clicks on the invite people button where the invite url will be generated or we can even generate new code by clicking generate a new link button in invite modal.

Expected Output: Generates invite friends url.

7. Test if the friends can join the server using the invite code:

Description: This test checks if the friend is able to join the server using the generated invite code by using it in the browser.

Input: Friend pastes the invite code url in his browser and registers his account using clerk.

Expected Output: user should be able to join the server of this friend through invite code and see his name in the left column below the general channel.

Real time messaging:

8. Test communication in general channel and in one on one communication where general is set as default for a server:

Description: This test checks if the messages can be sent in the general channel and direct communication through socket.

Input: Type a message in the chat input of the general channel or select a user in the user column and enter a chat message from chat input.

Expected Output: Receives successful message from both general channels and on direct message in one on one communication.

9. Chat with emojis:

Description: This test checks if we can send emoji as messages.

Input: User clicks on the emoji option on the chat input and sends a message.

Expected Output: Successful can communicate with emojis.

10. Edit and delete messages:

Description: This test checks we can edit and delete the sent messages.

Input: User clicks on edit and delete option for the messages he sent.

Expected Output: Successful can edit and delete his messages.

Real time file sharing:

11. Test functionality of file sharing:

Description: This test checks if the images and pdf are sent..

Input: Upload image and pdf from the chat.

Expected Output: Receives successful message from both general channels and on direct one on one communication where upon clicking them will opens up in the new tab in the browser.

UI customization and themes :

12. Test functionality of UI themes:

Description: This test checks if the user can switch between dark, light and system default modes.

Input: Click on the button to toggle between dark, light and system default modes.

Expected Output: The UI toggles between dark, light and system default modes.

PHASE 2 Feature Additions

Server Management :

13. Customize your server:

Description: This test checks if the admin can customize the server by changing the server by its name and by its image

Input: Clicks on the Server Settings button from drop down menu of the server, where a modal appears to edit and save server name and image, where after changing them the admin can click on save button on that modal to save his changes.

Expected Output: The name of the server and its image needs to change accordingly with the changes made by admin.

14. Delete server

Description: This test checks if the admin can delete the server which he created.

Input: Click on the delete server drop down option from the drop down menu of the server

Expected Output: The server which is previously created by the admin should be deleted.

15. Leave server

Description: This test checks if the guest or the moderator can leave the server

Input: For this user needs to be either guest or moderator of the server, and note the admin can only delete his server and not leave.

Expected Output: The user will leave the server and he will no longer be accessible inside the server in channels / direct messages.

Search functionality :

16. Search functionality

Description: This test checks if the user can search and navigate the channels and members and who are present in the server.

Input: Click on the search bar and type the channel/member name who the user likes to search for. And navigate to a specific channel/member when the user clicks on the search result.

Expected Output: Displays all the channels/members present in the server initially, and need to show correct search results with respect to the entered text, after entering name, and navigate to specific channel/member after clicking on the channel/member.

Manage member :

17. Admin Member Kicking

Description: This test checks if the admin can kick (remove) any member from the server.

Input: Click on manage member button from server drop down menu, and after the display of all the members in the channel, click on more options button for the specific member you want to kick, and then select kick option to kick them out of the server.

Expected Output: The user needs to get kicked out, and all the members of the channel should not be seeing the kicked out member and even that person needs to be removed from the channels.

18. Admin Role assigning

Description: This test checks if the admin can change the roles of the users in his server to either "Guest" or "Moderator"

Input: Click on manage member button from server drop down menu, and after the display of all the members in the channel, click on more options button for the specific member you want to change the role to “Guest” or “Moderator”

Expected Output: The roles and their specific permission of the users needs to be updated when changed to :

“Guest” : He is allowed to message or file share in channels/direct message, and he can delete or edit his own messages, and he gets option to leave the server after click on his server drop down menu. And he has access to his manage his account by deleting, editing his account and he has even access to change the UI Theme.

“Moderator” : He has the additional functionality to delete any message in group channels when compared to “Guest”.

Chat message/ File Sharing deleting :

19. Admin & Moderator chat message/file delete functionality

Description: This test checks if moderator or admin can delete the shared messages or file in group channels or in direct messages.

Input: Click on the delete button after the end of message or shared file like image or pdf field box, by any user.

Expected Output: The message or the file sharing will be deleted when clicked on delete button.

Compilation Instructions

Installing required libraries and Running the application

Prerequisites :

0. A computer with at least 8GB Ram.
1. Need to have node.js and npm run installed
2. A text editor like VS code
3. Git (Optional) for cloning the repo
4. Web Browser (Chrome or Safari)

After prerequisites :

1. Navigate to the root directory of the app (team-chat-app).
2. Install all the packages listed on package.json using npm: **npm install**
3. Make sure to set your **env** file with “**NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY**”, “**CLERK_SECRET_KEY**” which are clerk authentication service api keys for user authentication, “**NEXT_PUBLIC_CLERK_SIGN_IN_URL**”, “**NEXT_PUBLIC_CLERK_SIGN_UP_URL**” for handling both sign up and signin services, “**NEXT_PUBLIC_CLERK_AFTER_SIGN_IN_URL**”, “**NEXT_PUBLIC_CLERK_AFTER_SIGN_UP_URL**” for handling after sign up and signin services. “**UPLOADTHING_SECRET**”, “**UPLOADTHING_APP_ID**” for using UploadThing service which is for file sharing of images and pdfs, and “**DATABASE_URL**” for using the prisma DB for this go to aiven website to create the DB.

Note : .env file needs to be present in the root directory of the project

Check these website docs for additional queries :

Prisma DB : <https://console.aiven.io/>

Clerk : <https://dashboard.clerk.com/>

UploadThing : <https://uploadthing.com/>

As for now, i have updated all the **env** file parameters with the values that i have created, you can use them to run the app:

Use this snippet to add your .env file in the top directory inside “team-chat-app” folder as “.env” to run the app successfully.

```
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=pk_test_cGx1YXNlZC1ibHVlZ21sbC03Ny5jbGVyay5hY2NvdW50cy5kZXlYk
CLERK_SECRET_KEY=sk_test_0M8YmK3QM3C1MNPRfdybv128XnWRkj7CfePe5MPctE
NEXT_PUBLIC_CLERK_SIGN_IN_URL=/sign-in
NEXT_PUBLIC_CLERK_SIGN_UP_URL=/sign-up
NEXT_PUBLIC_CLERK_AFTER_SIGN_IN_URL=/
NEXT_PUBLIC_CLERK_AFTER_SIGN_UP_URL=/

UPLOADTHING_SECRET=sk_live_3d1e2ac22dd353d9d4107a36bece3751024386c780dd1e3f453c69a514c4f6a6
UPLOADTHING_APP_ID=5m3h3jfb5c

LIVEKIT_API_KEY=APIAJLtUCZ5ZWv8
LIVEKIT_API_SECRET=vtFzyPnYvp5p5RaCsxuKHZVqDes2poS7Fkmo3c8bgcH
NEXT_PUBLIC_LIVEKIT_URL=wss://discord-wdgh6yvk.livekit.cloud

DATABASE_URL='mysql://avnadmin:AVNS_ZdB912GZjcKyCUfG71S@mysql-32092d6d-team-chat.a.aivencloud.com:21748/defaultdb?ssl-mode=REQUIRED'
```

4. Build the next js app: **npm run build**

5. Start the app locally: **npm run dev**

6. Visit port <http://localhost:3000/> on successfully building and running the application.

Troubleshooting while compile and run (Typical Problems and Solutions) :

Problem: Problem with any of the dependency or next js app

Solution: Try to delete node_modules directory and install the dependencies again using “npm install” without changing anything in package.json.

Problem: error during any functionality breaking in the app.

Solution: Re-build the app using “npm run build” and run it again using “npm run dev”.

To learn more about Next.js, take a look at the following resources:

<https://nextjs.org/docs> - learn about Next.js features and API.

Testing the Application

The procedures below should be followed to compile and execute the test cases for the team chat web application:

1. Make sure the project is already setup by building the next.js app.
2. Run the app using "npm run dev"
3. Visit 127.0.0.1:3000 to explore the websites and related features.
4. Make sure to use your own env file for testing your own app with your updated parameters..
5. Make sure to create your account and create a server to test various test cases.

User Manual

Before you can use the application, please make sure your system meets the following requirements:

Operating System: Windows, macOS, or Linux

Web Browser: Chrome, Firefox, Safari, or Edge

Installation

Downloading the Application

1. Request the latest version of the application from our team member or repository or TA.
2. Extract the downloaded archive to your preferred location.

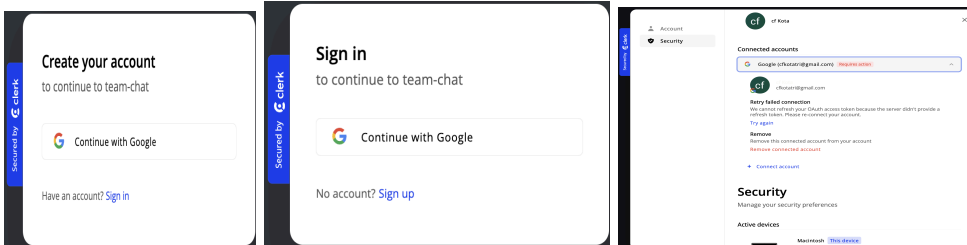
Installing required libraries and Running the application

1. Navigate to the root directory of the app.
2. Install all the packages listed on package.json using npm: npm i
3. Build the next.js app: npm run build
4. Start the app locally: npm run dev
5. Visit port <http://localhost:3000/> on successfully building and running the application.

Using the Application

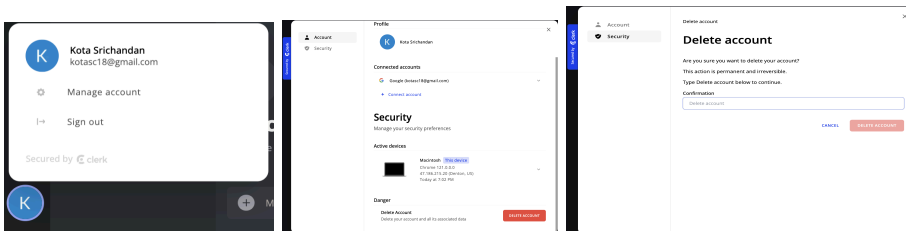
User Registration, sign in

1. Create your account using Clerk by sign up and sign in using your google account.
2. Sign In using your registered google account.
3. Try sign up and sign in again if you did not do the sign in or registration properly, the app indicates "requires action" on your account if you haven't set up your account properly.



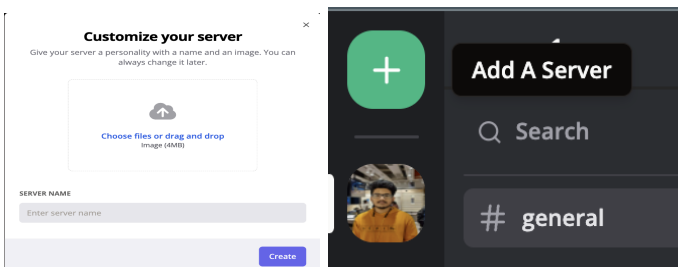
User Sign out and Delete account:

1. Click on the sign out button after signing to the app.
2. Click on delete account upon clicking on Manage Account to delete your account and confirm your delete account after typing "Delete account" in confirm delete screen



Create server:

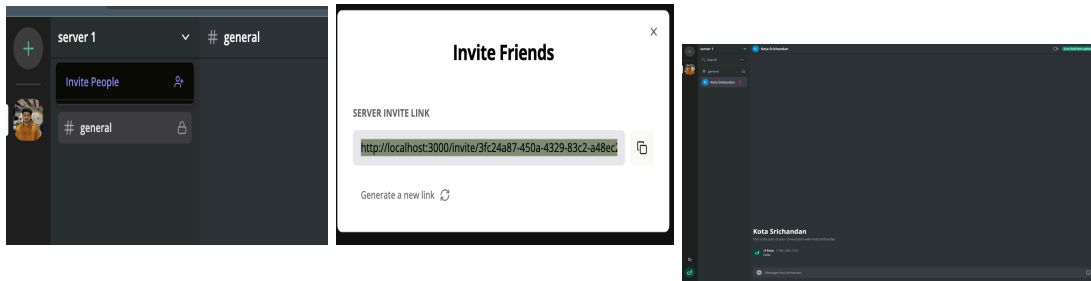
1. Create your own server by providing the name and image of your server.
2. And create any additional servers by clicking on the add server button (+) with your custom name and image.



Invite friends:

1.Generate your invite code url to invite your friends to your server, by clicking on the invite people button on your server, this generates a new invite code link, if it expires try to generate a new one using the “generate a link” button.

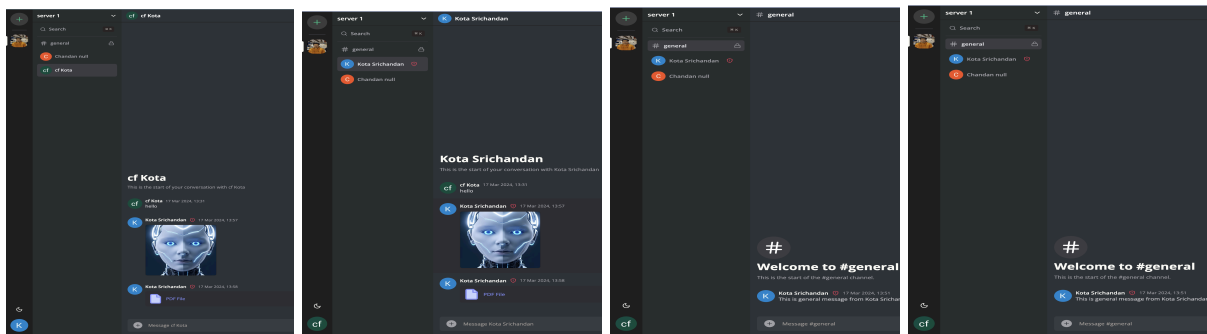
2.After successful invite code link generation, invite your friends by sharing your invite link, and after your friend pastes this link in his browser he will be added to your server after he does the registration and signing process, and he will be shown with his name as mentioned in his google account name.



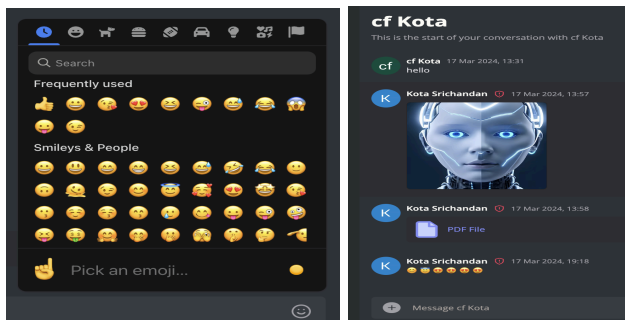
Real time messaging in direct messages and common channels like “general”:

1.Try sending out messages in the “general” channel which is a default channel provided for every server using the chat input box.

2.Try sending out messages by manually selecting the users you want to have direct one on one communication which will be below the “general” channel and use the chat input field to type your messages and press enter to send messages.



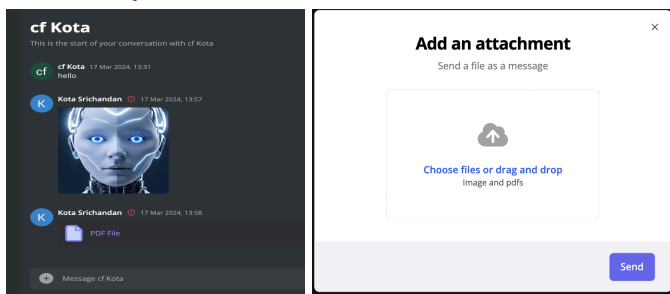
Emoji during communication: Use emoji during communication by clicking on the emoji button the chat input field, and use your desired emoji for communication.



Real time pdf and images sharing:

1. Share your pdf and images during communication on both one on one communication and channels like “general” using the upload button in the chat input field (+).

2.Then attach your attachments in the attach modal and click on send to send them.

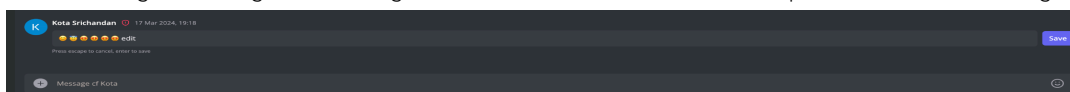


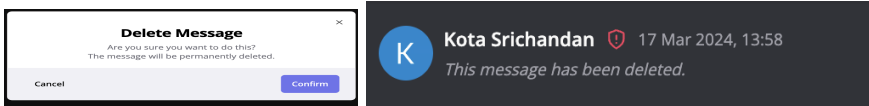
Edit and delete messages during communication:

1. Use edit and delete features for your messages during communication.

2. click on the edit and delete button on messages after sending them to edit or delete the message. For editing upon editing the message click on the save to save the edited message. For delete confirm if you really want to delete the message in delete modal.

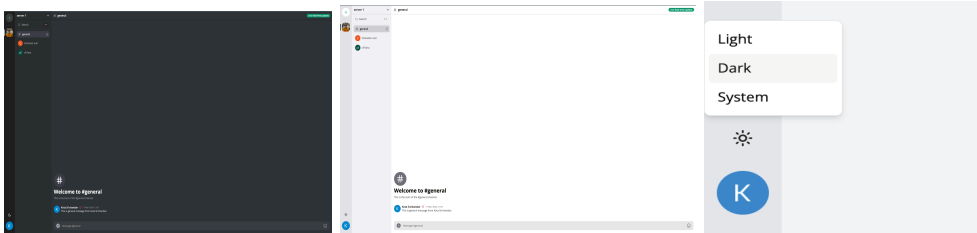
3. After deleting the message “This message has been deleted” text will be shown in place of the deleted message.





UI theme customization:

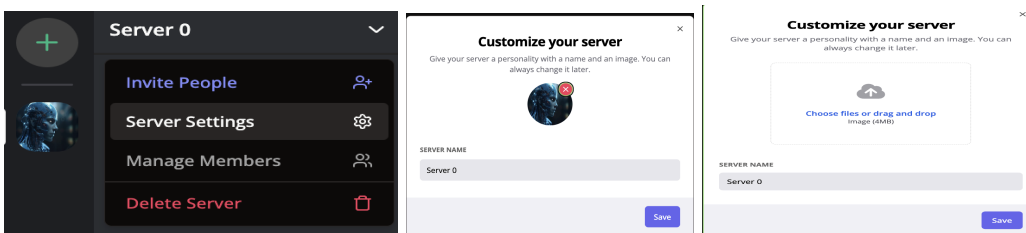
1. Click on the theme customization button to customize your theme for dark, light or keep it system default for UI customization.



Server Management :

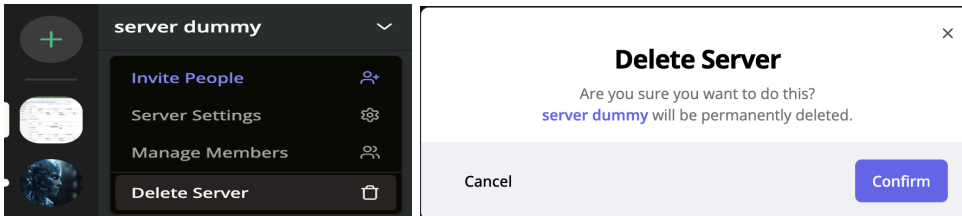
13. Customize your server:

1. Click on the server settings option from the server drop down menu, after which customize your server modal would be displayed under which you will have the option to change the server image by clicking on the red cross button to remove the existing server image and then choose files to upload your required server image then click on "Save" button to save your new changes.
2. In the server name field you can edit the existing server name and click then click on "Save" button to save your new changes.



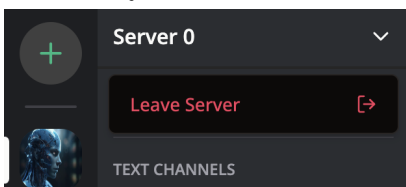
14. Delete server

You can delete your server by clicking on the delete server button from the server drop down menu, and after which the delete server modal will be popping up, in which you need to click on confirm button to delete your server.



15. Leave Server

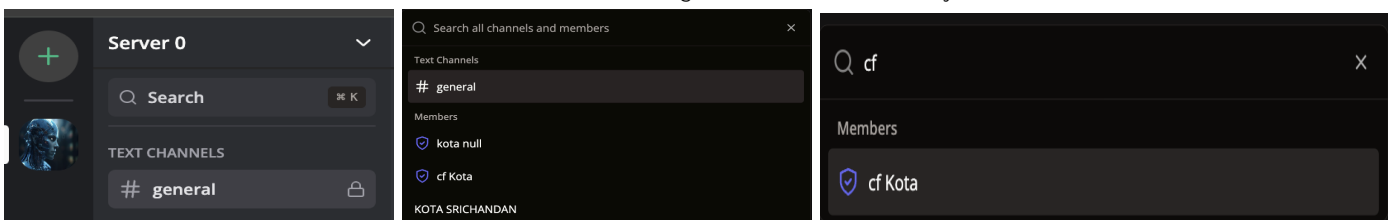
The moderators and guest can leave the server by clicking on the leave server button on their server drop down menu. Note : The admin who has created the server can only delete the server, he does not get the option to leave the server.



Search functionality :

16. Search functionality

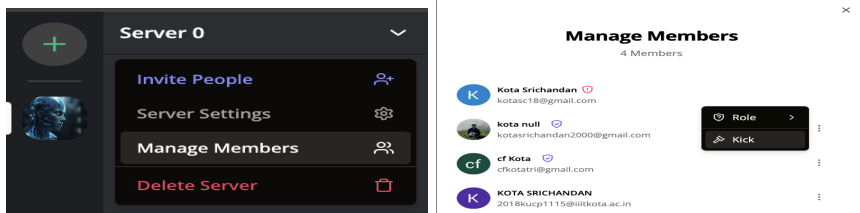
1. You can search for channel names or members in the server by clicking on the search bar under server drop down menu.
2. After which the search modal will be displayed initially displaying all the existing channels and members in the server.
3. You can enter the name of the channel or the member in the server to get search results related to your entered text.



Manage member :

17. Admin Member kicking

1. Click on the manage members button from the server drop down menu, after wich Manage members modal will be displayed.
2. Click on the kick option by clicking on the specific user after clicking on more options button which is 3 vertical drop button

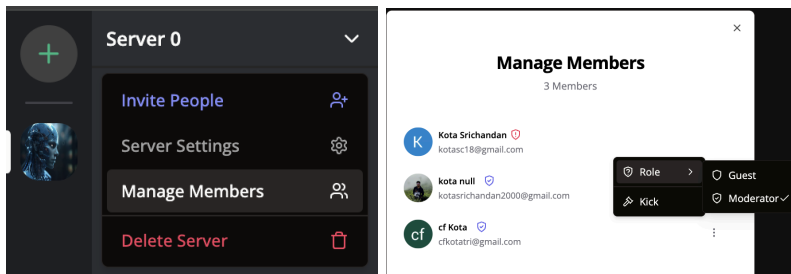


18. Admin Role assigning

1. Click on the manage members button from the server drop down menu, after wich Manage members modal will be displayed.
2. Hover on the Role button to display the existing roles to switch the current user to :

“Guest” : He is allowed to message or file share in channels/direct message, and he can delete or edit his own messages, and he gets option to leave the server after click on his server drop down menu. And he has access to his manage his account by deleting, editing his account and he has even access to change the UI Theme.

“Moderator” : He has the additional functionality to delete any message in group channels when compared to “Guest”.



Chat message/ File Sharing deleting :

19. Admin & Moderator chat message/ file delete functionality

1. For this the user role needs to be either admin or moderator. He can delete any message. He can delete any message or file shred either in group channels or in direct communication by click on the delete button on the message box.



Code Inspection Feedback and Actions

The partner SE group provided our team with insightful criticism throughout the peer review session. Here is a rundown of the main ideas raised: Feedback Received During Peer Review Session:

1. Technical Proficiency and Security Measures: Reviewers praised the technical architecture and the secure integration of authentication and authorization mechanisms through Clerk. The choice of technologies and the emphasis on security were seen as strengths of the project.
2. User Interface and Experience: Feedback highlighted the user interface's intuitiveness and the application's ease of use, especially the implementation of real-time messaging and file sharing. However, suggestions were made for further improvements to enhance accessibility and user engagement.
3. Documentation and Knowledge Sharing: The comprehensive documentation of technical implementations and architectural decisions was well-received. However, it was suggested that more detailed usage guides and API documentation could facilitate better understanding and collaboration.
4. Team Collaboration: The effective team collaboration and communication were commended. The reviewers appreciated the regular team meetings, clear role allocations, and the supportive environment that contributed to the project's progress.

Actions Taken Based on the Feedback:

1. Enhancing UX and UI: In response to the suggestions for improving the user interface, the team plans to adopt a more user-centered design approach. This includes conducting iterative testing and incorporating user feedback to refine navigation and interactions, making the application more intuitive and accessible.
2. Improving Documentation: To address the feedback on documentation, the team has committed to developing more detailed usage guides and enhancing API documentation. This effort aims to make onboarding easier for new team members and external contributors, ensuring that knowledge is accessible and maintainable.
3. Focusing on Security and Performance Optimization: Encouraged by the positive feedback on the project's technical proficiency and security measures, the team will continue to prioritize these aspects. Future plans include exploring advanced security protocols and optimizing the application for performance and scalability.
4. Maintaining Strong Team Dynamics: The team recognizes the value of effective collaboration and communication highlighted during the review session. Moving forward, we plan to maintain our supportive environment, encourage more open feedback, and engage all team members in decision-making processes to further enhance teamwork and project outcomes.

Reflection

What Has Been Accomplished:

- 1. Implementation of Microservices Architecture: Successfully implementing a robust microservices architecture utilizing a sophisticated tech stack, including Next.js, React, Socket.io, Prisma, and MySQL. This foundational work supports the application's core functionalities like real-time messaging and file sharing.
- 2. Integration of Clerk for Secure Authentication: By integrating Clerk for authentication and authorization, we've established a secure and seamless login process. This choice underscores our commitment to safeguarding user data and ensuring a trustworthy user experience.
- 3. Efficient Team Collaboration: The project benefited significantly from strong team collaboration. Regular meetings, clear communication, and effective role distribution have been key to maintaining high levels of productivity and motivation among team members.
- 4. User Interface Development: The development of a functional and visually appealing user interface, considering both aesthetics and user experience. This achievement reflects the team's capability in frontend development and design.
- 5. Features of User Registration, authentication, signout and deletion via Clerk Create Server Invite Code functionality for friends Real time messaging Real time file sharing UI customization and themes have been done

What Went Well:

- 1. Technical Selection and Implementation: The decision-making process behind selecting our technology stack and the successful implementation of chosen technologies stood out as a major success. These technologies have proven to be well-suited for the project's needs, facilitating real-time interactions and efficient data management.
- 2. Security and User Authentication: Our focus on security, particularly through the use of Clerk for handling user authentication and authorization, was a highlight. This approach has significantly enhanced the application's security, making it reliable for users.
- 3. Team Dynamics and Support: The strong dynamics within the team, characterized by mutual support and effective collaboration, contributed greatly to the phase's success. The ability to work cohesively and support each other through challenges has been a defining strength.
- 4. Initial User Interface and Experience: The initial development of the user interface and the attention to creating a positive user experience were well-received. Efforts to make the interface intuitive and engaging have laid a good foundation for further UX improvements.

Areas for Improvement:

- 1. Documentation and Knowledge Sharing: While the project's documentation covers technical aspects well, there is room for improvement in creating more detailed usage guides and API documentation. This will facilitate easier onboarding for new team members and contribute to the project's long-term maintainability.
- 2. Feedback Loops and User Testing: Establishing more structured feedback loops and conducting extensive user testing can provide valuable insights into user needs and preferences. This will be essential for prioritizing future development efforts and ensuring the application remains relevant and user-friendly.

Member Contribution Table

Here is the member contributions table for Phase 3, including details of each team member's contributions, overall contribution percentage, and any relevant notes:

Member Name	Contribution Description	Overall Contribution (%)	Note
Srichandan Kota	- Led system design and technological selection, focusing on microservices architecture and security planning and GitHub\ - Contributed in server management	12.5%	Exhibited outstanding leadership and decision-making in system design, ensuring a robust and scalable foundation for the application and Github
Swapna Sonti	- Developed front-end components, particularly for UI customization themes and real-time messaging functionalities. - Worked on the JS part of the system to add more dynamic functionalities to the front-end	12.5%	Played a pivotal role in front-end development, enhancing the application's interactivity and visual appeal
Sandeep Chowdary Ari	- Engineered the backend for real-time communication and managed server and channel functionalities - contributed in developing search functionality	12.5%	Essential in establishing a reliable and efficient backend system, facilitating seamless communication across the application
Venkata Sai Shankar Koppula	- Directed UI/UX design efforts, focusing on usability, accessibility, and implementing dark and light modes	12.5%	Significantly contributed to the application's design, improving user experience and accessibility through thoughtful design choices.
Shivanandha Reddy Vasudevula	- Managed database architecture, ensuring data integrity and optimizing data management strategies	12.5%	His expertise in database architecture enhanced data handling efficiency, supporting the application's core functionalities.
Gana Deekshith	- Led the integration of secure authentication flows, detailing security protocols and user verification processes. - Contributed in member maangement	12.5%	Bolstered the application's security framework, ensuring user data protection and a trustworthy environment.
Kantumutchu Dinesh	- Spearheaded the file sharing implementation and contributed to the real-time messaging system using Socket.io. Enhanced data handling for efficient communication and file exchange within the application.	12.5%	His pivotal role in integrating file sharing and optimizing real-time communication channels significantly bolstered the application's functionality, ensuring seamless interactions and data exchange.
Bhanu Prasad Krishna Murthy	- Led documentation efforts and developed the application's security framework, focusing on authentication and authorization.	12.5%	His comprehensive documentation and focus on security have been vital in establishing a solid foundation for the project.