COSC 310 -101

M2: Project Description and Requirements

Team Accord

- Hoc Nguyen
- Colin Lefter
- Toby Nguyen
- Bao Pham
- Immanuel Wiessler

Table Of Contents

Introduction

Accord is an engaging social platform where users can leverage our feature-rich instant messaging interface to chat and hang out with friends and communities. We are building a full-stack web application that uses React and TypeScript for the front-end and Python for the back-end, along with a MySQL database. In addition, we are using Next.js as the React framework for our front-end and Django as our Python-based web framework for our back-end.

Our front-end tech stack aligns with the increasing popularity of Next.js as the top choice for full-stack web development while leveraging the strict type-checking properties of TypeScript that improve our development workflow by simplifying the process of test-driven development. Our choice to use Django for our back-end enables us to develop the encryption-oriented peer-to-peer direct messaging feature that is central to our application. Furthermore, it allows us to store our site-wide database on the same server.

High-level description

Our platform will support peer-to-peer direct messaging as well as group conversations. Discussions take place through group chats or discussion servers dedicated to context-driven conversations via roles and threads. We believe everyone has the right to privacy, and offering end-to-end encryption across all our messaging services allows us to accomplish our ambition. Users will have access to a range of privacy controls that allow for flexible user interactions—from specifying the duration of message history to deleting all chat history after every discussion through *private mode*, user-driven data privacy is our priority.

Communities are central to our platform, and we intend to put users in control of how they engage with communities through server roles. Owners of discussion servers can assign roles to participating users to drive user engagement. Users design these servers with purpose by organizing discussions through text channels that act as individual conversation threads.

Project Requirement

1. User, Functional and Non-Functional Requirements

I. Allow users to create an account on the platform

a. The system should present an interface of a login page where the user can enter their username and password
b. Users should have the option of creating an account from both the home page and the login page
c. The account creation page should have a place for the user to fill in the necessary information for account creation such as username, email and password
    i. The system will use client-side form validation to notify the user of accepted inputs and specific input requirements as the user fills out the form. Likewise, the user will see error messages as a result of client-side validation if there are errors present at the time they click the “Submit” button as well. The user will have the opportunity to correct these messages as otherwise the system will not allow them to submit the form.
    ii. The system will output a confirmation notification when a user successfully created an account and will be directed to the login page to sign into their new accounts
d. Add the account credentials to the “accounts” table within the site-wide database

II. Allow the user to log in to the platform with an existing account

a. The system should present an interface of a login page where the user can enter their username and password
b. The system should have a log-in button near the input form
c. The system should check the input information with the account database if the user clicks on the login button or press enter and do one of the following:

i. Direct the user to the Accord chat page when the input information matches the account database's data
ii. Show the user the error in their input (invalid username, wrong password) when the user inputs information that does not match the database, as performed through client-side validation that is also present in the account creation page

III. Allow the user to log out of their account

a. The accord chat page should have a logout button
b. The system should direct the user to the Accord login page when the user clicks on the logout button

IV. Allow the user to edit their account (username, password, email, etc.)

a. The Accord chat page must show an avatar of the user as a representation of their profile
b. The system should direct the user to their profile page when they click on their avatar
c. The profile page should contain current user information and an edit button
d. The system should allow the user to edit their information (username, password, email, etc.) when the user clicks on the edit button and a save button will appear if the user information was edited
e. When the user clicks the save button, the new account information will be checked:
    i. The system will output a confirmation notification when a user successfully created an account
    ii. The system will use client-side validation for form inputs to guide the user on the accepted form inputs to avoid account creation failure and to give them the opportunity to change their inputs as a preventative measure of account creation failure (wrong email format, weak password, etc.)
f. The data for this user’s account in the account database is updated

V. Allow the user to send text messages

a. The Accord chat page should allow the user to chat with other users by sending direct messages

b. Accord should notify the recipient about any messages through in-app notifications, internally keeping track of the sender and recipient of each message

VI. Allow user to receive text messages

a. Keep the user notified about new messages, and if the messages have been opened through in-app notifications
b. Tracking the state of each message (where the message is located, whether it is read or unread, and who sent it)

VII. Allow users to search for other users on the platform

a. Via a search bar in the application, users have the ability to search up other users by their username
b. Via a server member dropdown list available on each server that the user is part of

VIII. Allow the user to add another user as a friend

a. By clicking on a user's name-avatar pair accessed through the main application search bar or through a server's user member dropdown list, the user can click on an "Add friend" button that sends a friend request to that user

IX. Allow users to remove another user from their friend list

a. From the name-avatar pairl, there also exists a remove friend button that displays a confirmation modal when clicked
b. The removed user should not be notified that a user has removed them as a friend

X. Allow the user to create a group chat

a. Being able to create a group chat, where the messages are only visible to the members of the group chat
b. Simulate a mini-server that only has a simple text-messaging interface without the elements of a typical server (text channels, roles, etc.)
    a) Supports real-time texting through network requests handled in Django
    b) Supports end-to-end message encryption through generated encryption keys that are specific to the current messaging

session and the people within the discussion through our Django back-end server

c) Supports a privacy option that does not store any message history once all users leave the chat

d) Supports a standard message-saving feature where all user conversations are logged within the database in real-time and conversation history is loaded each time the user clicks on that chat

XI. Allow group chat owners to manage a group chat

a. Provide the group chat creator to delete messages and remove members

XII. Allow server owners to create a text channel

a. The server owner sees a "+" button next to the text channels drop down, enabling them to create a new text channel
b. When the server owner clicks on the "+", the system displays a modal that prompts the owner to type in name of the text channel to be created
c. A "Create server" at the bottom of the modal
d. A new text channel is created and the server database is updated once all form inputs are validated with client-side validation

XIII. Allow server owners to edit text channels

a. In a server UI, the system should show a "..." button to the server owner
b. When a server owner/moderator clicks on the "..." button, the system displays a dropdown containing two buttons: "Delete server" and "Edit server profile"
    i. When the user clicks on the "Delete server" button, a modal prompts the user to confirm that they want to delete the server
        1. If the user clicks on the "Confirm" button, the system will delete the text channel from the server database. Otherwise, the user can click on a "Back" button to dismiss the modal

ii. When the user clicks on the “Edit server profile button”, a modal prompts the user to update the existing server name
    1. If the user clicks on the “Confirm” button, the system will update the server name from the server database. Otherwise, the user can click on a “Back” button to dismiss the modal

XIV. Allow users to add other users to the server

a. In a server UI, the system should show an add user button to the server owner
b. When the server owner clicks on the “Add user” button, the system should show a list of addable users with an “Add” button next to the users’ name-avatar pair and a search bar for the server owner to search for the user's name
c. When the server owner clicks on the “Add user” button, the user is added to the server, and the server is added to that user’s server list
d. The added user should be able to see the new server
e. The database of the user and the server is updated

XV. Allow server owners to remove a user from a server

a. The owner should have a full view of the member list and should have full control of the removal of the user
b. The user should not be able to see, receive, communicate or benefit from the server that they have been removed from

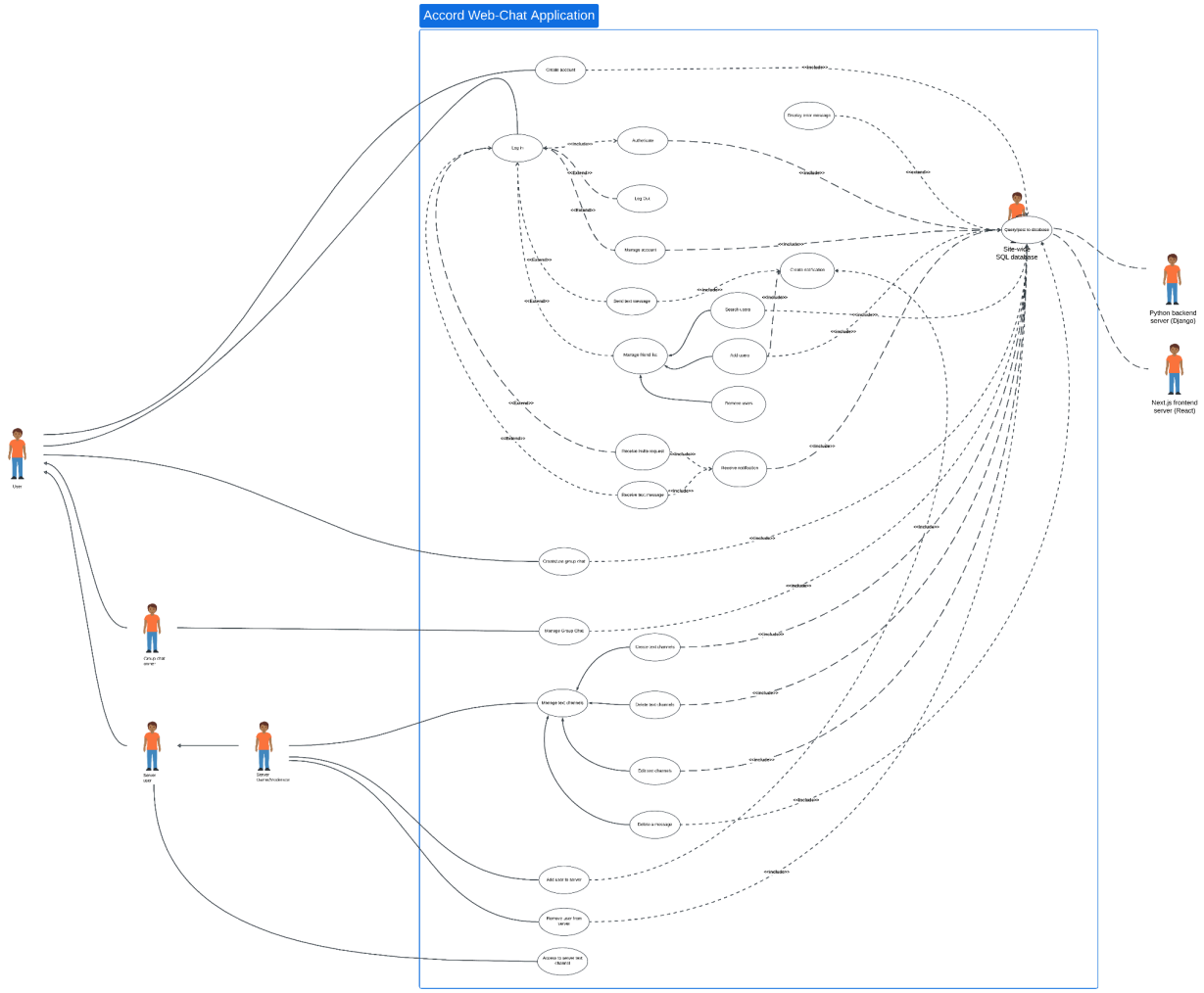
XVI. Allow server owners to delete a message from a server

a. A message can be removed by the server owner—this includes the visibility, storage, and existence of the message
b. Other messages surrounding and/or related to this message should keep their integrity (time posted, position among other messages)

XVII. Allow users to access servers

a. A user should be able to see the server list on the left side of the Accord chat page and be able to click on one of the servers in the list to go to the server page
b. The system should direct the user to the server chat if the user clicks on a server in the server list

2. Use Case Diagram [(link to PDF)]



Use Case Description

- Use Case 1. Create an account

- Primary actor: User.
- Description: Describes the process when a user creates a new account.
- Precondition: None.
- Postcondition: If successful creation, the users' accounts database is updated with the new account's details.

★ Main Scenario:

1. User clicks to create a new account.
2. The user provides their information to set up a new account.

3. If the new username has not been used, the account is created and updated in the accounts database.

★ Extensions:

2a. Error message if the user's input is in the wrong format (wrong email format, the password is too weak, etc.).

3a. The username has already existed, but account creation fails.

- Use Case 2. Log in

- Primary actor: User.
- Description: Describes the process when a user logs in.
- Precondition: None.
- Postcondition: If successful login, the user gains access to his/her account and can start using the application.

★ Main Scenario:

1. The user types in their username and password.
2. The user clicks the “login” button to log in.
3. The information then gets verified with the database.
4. The user is logged in.

★ Extensions:

3a. User’s input for either username or password or both is invalid (non-existing username, wrong password, typo in username).

- Use Case 3. Log out

- Primary actor: User.
- Description: Describes the process when a user logs out.
- Precondition: The user must have been successfully logged in.
- Postcondition: The user is now logged out.

★ Main Scenario:

1. User clicks the “logout” button to log out.

★ Extensions: None.

- Use Case 4. Manage account

- Primary actor: User.

- Description: Describes the process when a user manages this/her account's information.
- Precondition: The user must have been successfully logged in.
- Postcondition: After successfully updating the account's information, the database that contains the user's account is updated.

★ Main Scenario:

1. User clicks on their account profile to check their information.
2. Their information is displayed with an option to edit (aside from username, which is non-editable for security reasons).
3. The user gains access to edit mode for their information.
4. User edits their information accordingly.
5. The user clicks the "save" button to update their account with the new information.

★ Extensions:

5a. At least one new information input is invalid (wrong email format, new password does not meet the minimum security requirement).

- Use Case 5. Send text message

- Primary actor: User.
- Description: Describes the process when a user sends a text message to another user/group chat/server text channel.
- Precondition: The user must have been successfully logged in.
- Postcondition: If successful delivery of message, the message gets sent and the database for message history is updated.

★ Main Scenario:

1. User chooses the target to start chatting. This target can be another user, a group chat, or a server text channel that they are in.
2. The user types in their message and sends it.

★ Extensions: None.

- Use Case 6. Search User

- Primary actor: User.
- Description: Describes the process when a user searches for another user.
- Precondition: The user must have been successfully logged in.
- Postcondition: The searched username gets displayed

★ Main Scenario:

1. User types in the username that they want to search into the search bar.
2. The queried username is verified to see if it exists.
3. If it exists, the searched username is displayed.

★ Extensions:

1a. User searches for an invalid username.

1a1. Nothing is displayed.

- Use Case 7. Add a user

- Primary actor: User.
- Description: Describes the process when a user adds another user to their friend list.
- Precondition: The user must have been successfully logged in and the one to be added is not already on the user's friend list.
- Postcondition: If successful, the user's friend list is updated with a new friend and the database is also updated.

★ Main Scenario:

1. User search for a username from the search bar.
2. The queried username is verified to see if it exists.
3. If it exists, the searched username is displayed.
4. The user has the option to send a friend request.
5. If the friend request is accepted, the user's friend list is updated with their new friend.

★ Extensions:

1a. User searches for an invalid username.

1a1. Nothing is displayed.

- Use Case 8. Remove a user from the friend list

- Primary actor: User.
- Description: Describes the process when a user removes another user from their friend list.

- Precondition: The user must have been successfully logged in and the one that they want to remove must be on their friend list.
- Postcondition: If successful, the “friend” is removed, and the user's friend list and database are updated.

★ Main Scenario:

1. User navigates to their friend list.
2. The user searches for the “friend” that they want to remove from the search bar.
3. User clicks on the friend’s username and chooses “remove from friend list”.
4. The “friend” is removed, and the user's friend list and database are updated to reflect the change.

★ Extensions:

2a. User searches for an invalid username.

2a1. Nothing is displayed.

- Use Case 9. Create a group chat

- Primary actor: User.
- Description: Describes the process when a user creates a new group chat with at least 2 other users.
- Precondition: The user must have been successfully logged in.
- Postcondition: If successful group chat creation, the database is updated.

★ Main Scenario:

1. User clicks “create group chat”.
2. The user types in the usernames that they want to include in the group chat in the mini-search box.
3. The User adds participants to the list and clicks “Ok” to proceed with the group chat creation.
4. A new group chat is created and the database is updated.

★ Extensions:

2a. User searches for an invalid username.

2a1. Nothing is displayed.

- Use Case 10. Manage a group chat

- Primary actor: Group chat owner.
- Description: Describes the process when a group chat owner adds/removes a user from the group chat.
- Precondition: The user must have been successfully logged in and must also be the group chat owner.
- Postcondition: The group chat's list of members as well as the database are updated accordingly.

★ Main Scenario:

1. Group chat owner clicks the "manage group chat" button to start editing it.
2. Group chat owner clicks to add to or remove usernames from the list.
3. The list and database are updated accordingly after the edit.

★ Extensions:

2a. User searches for an invalid username for adding to the group chat.

2a1. Nothing is displayed.

- Use Case 11. Create a Text channel

- Primary actor: Server owner/moderator
- Description: Describes the process when a server owner/moderator creates a new text channel within the server.
- Precondition: The user must have been successfully logged in and must also be the server owner or server moderator.
- Postcondition: A new text channel is created and the database is updated.

★ Main Scenario:

1. Server owner/moderator clicks the "+" button to create a new text channel within their server.
2. Server owner/moderator types in the name for the text channel.
3. Server owner/moderator clicks "create" to create it.
4. A new text channel is created within the server and the database is updated.

★ Extensions: None.

- Use Case 12. Delete a Text channel

- Primary actor: Server owner/moderator

- Description: Describes the process when a server owner/moderator deletes a text channel within the server.
- Precondition: The user must have been successfully logged in and must also be the server owner or server moderator.
- Postcondition: A chosen text channel is deleted and the database is updated.

★ Main Scenario:

1. Server owner/moderator clicks the "-" button right next to a text channel to delete it.
2. The chosen text channel is deleted and the database is updated.

★ Extensions: None.

- Use Case 13. Edit Text channels

- Primary actor: Server owner/moderator
- Description: Describes the process when a server owner/moderator edit the name of a text channel within the server.
- Precondition: The user must have been successfully logged in and must also be the server owner or server moderator.
- Postcondition: A chosen text channel's name is changed and the database is updated.

★ Main Scenario:

1. Server owner/moderator clicks on the "edit" button right next to the text channel they want to edit.
2. Server owner/moderator then type in a new name for the text channel.
3. The server owner/moderator clicks the "Ok" button to update the change.
4. The text channel's name is changed and the database is updated accordingly.

★ Extensions: None.

- Use Case 14. Add a user to the server

- Primary actor: Server owner/moderator
- Description: Describes the process when a server owner/moderator sends an invitation to a new user to join the server.
- Precondition: The user must have been successfully logged in and must also be the server owner or server moderator.
- Postcondition: If successful, a new user is added to the server and the database is updated accordingly.

★ Main Scenario:

1. Server owner/moderator clicks the “add user” button in the server UI.
2. Server owner/moderator types in the username of that user.
3. If found, the server owner/moderator clicks the “invite” button to invite that user to the server.
4. If the user accepts the invitation, the user is then added to the server and the database is updated accordingly.

★ Extensions:

2a. Server owner/moderator searches for an invalid username for adding to the server.

2a1. Nothing is displayed.

3a. The invited user declines the invitation.

3a1. The user will not be added to the server.

- Use Case 15. Remove a user from the server

- Primary actor: Server owner/moderator.
- Description: Describes the process when a server owner/moderator removes a user from their server.
- Precondition: The user must have been successfully logged in and must also be the server owner or server moderator.
- Postcondition: If successful, the chosen user is removed from the server and the database is updated accordingly.

★ Main Scenario:

1. Server owner/moderator searches for the username that they want to remove from the server’s search box.
2. If found, the server owner/moderator clicks on that username.
3. A small pop-up with that username will show up, and the server owner/moderator will click on the “remove” button to remove the user.
4. The user is successfully removed from the server and the database is updated accordingly.

★ Extensions:

1a. Server owner/moderator searches for an invalid username to remove from the server.

1a1. Nothing is displayed.

- Use Case 16. Remove a message from the server

- Primary actor: Server owner/moderator.
- Description: Describes the process when a server owner/moderator removes a message from their server.
- Precondition: The user must have been successfully logged in and must also be the server owner or server moderator.
- Postcondition: If successful, the chosen message is removed from the server and the database is updated accordingly.

★ Main Scenario:

1. Server owner/moderator navigates to the message in a text channel in their server that they want to remove.
2. Server owner/moderator clicks the small “x” indication right next to a message to delete it.
3. The message is successfully removed from the server and the database is updated accordingly.

★ Extensions: None.

- Use Case 17. Access server

- Primary actor: Server user
- Description: Describes the process when a server user opens up the server that they’re in to see all the relevant chats.
- Precondition: The user must have been successfully logged in and must also be on that server.
- Postcondition: The server user successfully sees all the chats in the server and can participate in chatting.

★ Main Scenario:

1. The server user navigates the server list from the left-hand side and then picks the one that they want to open up.
2. The server is opened up for the server user.

3. All the messages are displayed and the server user can start chatting in any of the text channels on that server.

★ Extensions: None.

3. User Stories

Normal user

As a user, I want:

- To be able to send direct messages to my friends on the platform through a private text channel so that I can communicate with my friends directly
- The ability to join a server after being invited to the server from the server owner so that I can join my desired server through the request of the server owner
- The ability to know when my friends are talking to me through private text channels so that I can respond to their message requests as quickly as possible
- To be able to see a list of users I'm friends with so that I can interact with them in a quick and convenient manner
- To be able to create a group chat so that I can converse with a group of people
- To be able to create a server so that I can communicate and interact with a community having similar interests

Server owner

As a server owner, I want:

- To have the ability to create and delete text channels
- To be able to invite people to my server so that I can grow my Accord community of users having similar interests
- The ability to remove users who are not following my server rules so that I can keep my community clear of any unwanted users
- To be able to delete certain text messages within text channels so that I can remove inappropriate messages/content with ease

Group chat owner

As a group chat owner I want to:

- Have the ability to invite people to my group chat so that I can chat with multiple users sharing the same interest at the same time
- Be able to remove individuals from the group chat so that I can keep the group clear of any unwanted users