# Chat App API Design

Team I: Sammy The Owl

Xiao Xin, Zhijian Yao, Weiwei Zhou, Xinru Xiao, Xuyang Xiao, Wenlong Yan

# 1. Summary

Our ChatApp will have the following features (We specify more details in blue based on the homework requirements).

- A user must initiate the creation of a chat room as private or public. The one who creates the room will be the admin automatically.
- For a private room, the admin needs to set the interests requirements. A public room has no requirements.
- An unblocked user can send direct messages to all users online.
- The banned user will be removed from the current room and will be banned from joining other rooms. He can still create new rooms and chat in the rooms created by him. There are two ways to trigger “Ban”:
    - The admin approves the report request.
    - The user tries to either broadcast or send a private message which includes “hate” / “HATE” for the second time. (The user will be warned for the first time).
- An unblocked user should be notified that their message has been received.
- A user may be in multiple chat rooms (public and private).
- A user can choose to exit one chat room or all chat rooms.
- At least one of the users in the chat must be the admin.
- Admin broadcast approved requests of qualified users. Admin monitor users and messages. Admin can ban users and delete messages.
    - Age
    - School (e.g. Rice, Harvard, Duke, Nanjing, Wuhan, etc…)
    - Interests (e.g. Reading, Sports, Traveling, etc...)
    - A user has a profile that includes their age, school, and interests
- A user can join a public chat room (no special interests) if they are qualified to join.
- Admin invite users to join private rooms regardless of their interests.
- A user will be warned once and eventually forcibly banned from joining all other chat rooms if the user uses the word “hate” in a message for the second time.
- An user can broadcast messages to all users in the same chat room.
- An admin can remove users in the room.
- A user can determine what chat rooms they have joined and what public rooms they can join.
- When user1 sends a message to user2, user1's message should only appear on user1 and user2's screen. It should not appear on user3's screen.

- Messages can contain text, emojis, and/or links. Messages can be edited, recalled, deleted.
- If message msg2 is sent after message msg1, msg2 should appear on the screen below msg1
- If a user leaves a chat room, it should be clear to the other users why the user left (banned, connection closed, forced to leave).
- Users can report any other users (except for the admin and himself) in the same room. The reporter reports to the admin and the admin decides whether to ban the reportee.
- If the admin is banned or left from the current room. The room will be dismissed immediately.
- No JavaScript alerts should be used in the ChatApp since they will pause the app.

# 2. Use Case

a.

| Use Case UC1 | Enter ChatApp website |
|---|---|
| Primary Actor | User |
| Secondary Actor | Server |
| Pre condition | The user must have a good internet connection. |
| Post condition | The user accesses the Heroku ChatApp website successfully. |
| Main Flow | 1. The user enters the url of our ChatApp in the browser. |
| Second Flow | N/A |
| Exceptions | N/A |

b.

| Use Case UC2 | Log into ChatApp |
|---|---|
| Primary Actor | User |
| Secondary Actor | Server |
| Pre condition | 1. The user accesses the ChatApp website successfully. (UC1) |

| | |
|---|---|
| Post condition | 1. The server creates a profile for the user.<br>2. The user enters the main page. |
| Main Flow | 1. The user enters his/her name. (Username consists of alphanumeric characters (a-zA-Z0-9) or underscores. The number of characters must be between 5 to 20).<br>2. The user enters his/her age.<br>3. The user enters his/her school (e.g. Rice, Harvard, Duke, Nanjing, Wuhan, etc…).<br>4. The user chooses multiple interests (e.g. Reading, Sports, Traveling, etc...).<br>5. The user clicks the “Enter” button. |
| Second Flow | N/A |
| Exceptions | 1a. If the name is blank. Ask the user to enter the name.<br>1b. If the name is duplicated. Ask the user to choose a different name.<br>2a. If the age is empty. Ask the user to enter.<br>3a. If the school is empty. Ask the user to enter the school.<br>4a. If the interest. Ask the user to choose at least one interest. |

c.

| | |
|---|---|
| Use Case UC3 | Create a chat room with restrictions |
| Primary Actor | User |
| Secondary Actor | Server |
| Pre condition | 1. The user has accessed the ChatApp successfully. (UC1)<br>2. The user has created a profile. (UC2) |
| Post condition | 1. The chat room is created successfully.<br>2. The room name is added to the all room list.<br>3. The user becomes the admin of this room and joins this room automatically. |
| Main Flow | 1. The user enters the room name.<br>2. The user sets the room as public. |
| Second Flow | 1. The user enters the room name.<br>2. The user sets the room as private. |

| | |
|---|---|
| | 3. The user sets “interests requirements” for restriction. |
| Exceptions | N/A |

d.

| | |
|---|---|
| Use Case UC4 | Send a private message |
| Primary Actor | User |
| Secondary Actor | User |
| Pre condition | 1. Both senders and receivers are online. |
| Post condition | 1. A new message shows up as a notification in the receivers’ window.<br>2. The senders receive notification. |
| Main Flow | 1. The user chooses an user from the all userlist panel to chat privately.<br>2. The user enters the message.<br>3. The user clicks on the “Send” button. |
| Second Flow | N/A |
| Exception | 2a. If the content includes “hate”. The user will be warned and banned from joining **all other chat rooms**.<br>3a. If the message is not sent successfully. The sender will receive an alert.<br>3b. If the sender is blocked by the receiver. The sender will receive an alert. |

e.

| | |
|---|---|
| Use Case UC5 | Receive a private message |
| Primary Actor | User (Receiver) |
| Secondary Actor | User (Sender) |
| Pre condition | 1. The sender is not blocked by the receiver.<br>2. The sender has sent a message to the receiver. (UC4) |
| Post condition | 1. The senders receive the reply message. |

| | |
|---|---|
| Main Flow | 1. The receivers receive the message.<br>2. The receivers enter the reply message and send it back. |
| Second Flow | 1. The receivers ignore the message. |
| Exception | (MF)1a. If the content includes “hate”. The user will be warned and banned from joining **all other chat rooms**. |

f.

| | |
|---|---|
| Use Case UC6 | Broadcast a message |
| Primary Actor | User |
| Secondary Actor | User |
| Pre condition | 1. The senders and the receivers are in the same room. |
| Post condition | 1. A new message appears on the dashboard of the receivers.<br>2. The senders are notified if the message is received or not. |
| Main Flow | 1. The user enters a message and clicks the “send” button. |
| Second Flow | N/A |
| Exception | 1a. If the content includes “hate”. The user will be warned and banned from all chat rooms. |

g.

| | |
|---|---|
| Use Case UC7 | Join a public chat room |
| Primary Actor | User |
| Secondary Actor | Server |
| Pre condition | 1. The user has logged into ChatApp. (UC2)<br>2. At least one room exists. |
| Post condition | 1. The name of the user appears on the participant list of the chat room he just joined. |
| Main Flow | 1. The user chooses a room and clicks on the |

| | |
|---|---|
| | room name. |
| Second Flow | N/A |
| Exception | 1a. If the user was banned before. Refuse and notify the user. |

h.

| Use Case UC8 | Join a private chat room |
|---|---|
| Primary Actor | User |
| Secondary Actor | Server |
| Pre condition | 1. The user has logged into ChatApp. (UC2)<br>2. At least one room exists. |
| Post condition | 1. The name of the user appears on the participant list of the chat room he just joined. |
| Main Flow | 1. The user chooses a room and clicks on the room name.<br>2. The user meets all the requirements of the room. |
| Second Flow | N/A |
| Exception | 2a. If the user fails to meet all the requirements. Refuse and notify the user. |

i.

| Use Case UC9 | Exit a chat room |
|---|---|
| Primary Actor | User |
| Secondary Actor | User |
| Pre condition | 1. The user is in a chat room. |
| Post condition | 1. The current user name is removed from the participant list of the current room.<br>2. All the other users in the chat room receive a notification. (UC10)<br>3. The chat room is removed from the joined room list of the user.<br>4. The name of the user is removed from the participant list of the chat room he just |

| | |
|---|---|
| | exited. |
| Main Flow | 1. The user is not the admin. The user clicks the “exit” button of one chat room. |
| Second Flow | N/A |
| Exception | 1a. If the room becomes empty or the admin leaves the room. Dismiss the chat room. |

j.

| | |
|---|---|
| Use Case UC10 | Receive notification -- “A user just left the room” |
| Primary Actor | User |
| Secondary Actor | User |
| Pre condition | 1. The user is in the same chat room as the user who just left. |
| Post condition | 1. A notification appears on the dashboard of every user. |
| Main Flow | 1. All the users receive a notification with a reason included. |
| Second Flow | N/A |
| Exception | N/A |

k.

| | |
|---|---|
| Use Case UC11 | Invite a user to a private room |
| Primary Actor | Admin |
| Secondary Actor | User |
| Pre condition | 1. The admin is in a private room.<br>2. The user to be invited is not in the private room as the admin. |
| Post condition | 1. The user joins the private room regardless of his interests. |
| Main Flow | 1. Admin clicks the “invite” button and input the username of the user to be invited. |
| Second Flow | N/A |

| | |
|---|---|
| Exception | 1a. If the user to be invited does not exist. Notify the admin.<br>2a. If the user has been banned to enter the room before. Notify the user. |

l.

| | |
|---|---|
| Use Case UC12 | Edit a message |
| Primary Actor | User |
| Secondary Actor | Message |
| Pre condition | 1. The user is in a chat room.<br>2. The message to be edited was sent by the user. |
| Post condition | 1. The content of the message is updated on all dashboards of all the receivers.<br>2. All receivers receive a notification -- “The message has been recalled”. |
| Main Flow | 1. The user clicks the “edit” button following a message.<br>2. The user edits the message and clicks the “done” button. |
| Second Flow | N/A |
| Exception | 2a. If the content includes “hate”. The user will be warned and banned from all chat rooms. |

m.

| | |
|---|---|
| Use Case UC13 | Delete a message |
| Primary Actor | User |
| Secondary Actor | Message |
| Pre condition | 1. The user is in a chat room.<br>2. The message to be deleted was sent by the user. |
| Post condition | 1. The message is removed from the chat window of the current user but still exists in the chat windows of other users. |
| Main Flow | 1. The user clicks the “delete” button of the message. |

| | |
|---|---|
| Second Flow | N/A |
| Exception | N/A |

n.

| | |
|---|---|
| Use Case UC14 | Recall a message |
| Primary Actor | User |
| Secondary Actor | Message |
| Pre condition | 1. The user is in a chat room.<br>2. The message to be recalled was sent by the user. |
| Post condition | 1. The message is overwritten as “The message was recalled”. |
| Main Flow | 1. The user clicks the “recall” button of the message. |
| Second Flow | N/A |
| Exception | N/A |

o.

| | |
|---|---|
| Use Case UC15 | Admin forces a user to leave |
| Primary Actor | Admin |
| Secondary Actor | User |
| Pre condition | 1. The admin and the user are in the same chat room. |
| Post condition | 1. The user’s name is removed from the participant list of the chat room.<br>2. All other users receive a notification. (UC10) |
| Main Flow | 1. The admin chooses one user and clicks the “remove” button. |
| Second Flow | N/A |
| Exception | N/A |

p.

| | |
|---|---|
| Use Case UC16 | Users report a user |

| | |
|---|---|
| Primary Actor | User |
| Secondary Actor | User |
| Pre condition | 1. The reporter and the reportee are in the same room.<br>2. The reporter has sent at least one message in the chat room. |
| Post condition | 1. The admin receives a reporting message. (UC17) |
| Main Flow | 1. The reporter chooses one message and inputs his/her reason.<br>2. The reporter clicks the “report” button. |
| Second Flow | N/A |
| Exception | N/A |

q.

| | |
|---|---|
| Use Case UC17 | Receive a reporting message |
| Primary Actor | Admin |
| Secondary Actor | Reporter (User) |
| Pre condition | 1. A user reports another user. (UC16) |
| Post condition | 1. The reporting message disappears.<br>2. The reportee is removed from that chat room and is banned from joining all other rooms if the admin approves. |
| Main Flow | 1. The admin clicks the “approve” button. |
| Second Flow | 1. The admin clicks the “disapprove” button. |
| Exception | N/A |

r.

| | |
|---|---|
| Use Case UC18 | Exit all chat rooms |
| Primary Actor | User |
| Secondary Actor | Server |
| Pre condition | 1. The user has joined at least one chat room. |

| | |
|---|---|
| Post condition | 1. The user exits all chat rooms.<br>2. All the other users in the chat room receive a notification. (UC10)<br>3. All chat rooms are removed from the joined room list.<br>4. The name of the user is removed from all the participant list of the chat rooms he just exited. |
| Main Flow | 1. The user clicks the “exit all” button. |
| Second Flow | N/A |
| Exception | 1a. If the user has not joined any rooms. The server does nothing. |

s.

| | |
|---|---|
| Use Case UC19 | Block a user |
| Primary Actor | User |
| Secondary Actor | User |
| Pre condition | 1. The two users are in the same chat room. |
| Post condition | 1. The user is added to the blocked list of the other user. |
| Main Flow | 1. The user chooses to block the other user and clicks the “block” button. |
| Second Flow | N/A |
| Exception | 1a. If the to-be-blocked user has been blocked before. Notify the user. |

t.

| | |
|---|---|
| Use Case UC20 | Log out |
| Primary Actor | User |
| Secondary Actor | Server |
| Pre condition | 1. The user has entered the chat app. |
| Post condition | 1. The user is redirected to the login page. |

| | |
|---|---|
| Main Flow | 1. The user clicks the “Logout” button or refreshes the page. |
| Second Flow | N/A |
| Exception | 1a. If the user has joined other chat rooms before he logs out. Notify all the other users. (UC 10) |

u.

| | |
|---|---|
| Use Case UC21 | Include “hate” in the message |
| Primary Actor | User |
| Secondary Actor | Server |
| Pre condition | 1. The user is in a chat room / is sending a private message. |
| Post condition | 1. The user receives a warning after trying to send a hate message for the first time / The user is banned from joining other rooms after trying to send a hate message for the second time.<br>2. The message is not sent. |
| Main Flow | 1. Send a message which Includes “hate” / “HATE”. |
| Second Flow | N/A |
| Exception | N/A |

# 3. API Design

For this assignment, we use the **Model-View-Controller design pattern**,**Singleton design pattern, Factory design pattern and Command design pattern**. All data is in JSON-format. The data is only transferred via WebSocket protocol.

## 3.1 View

We use React.js and AntUI Design on the front-end. The front-end mainly consists of two parts -- a login form and a chatting page. After logging into our chatApp successfully, the user will be redirected to our main chatting page.

The login page has a login form which includes the following parts:

1) Name
2) Age
3) School
4) Interests

The main chatting page is divided into several parts:

1) A list of joined rooms.
2) A sublist of participants for each joined room. Each participant name is followed by a report button and a block button.
3) A list of all rooms, which includes both private and public rooms.
4) A list of all online users. A user can chat with another user privately.
5) A chat window where the user can receive and send messages.
6) An “invite” and an “exit” button in each chat room.
7) A button for creating a new chat room with restrictions.
8) A button for exiting all rooms.
9) Each message sent by the owner has a delete button, a recall button and an edit button.
10) A logout button.
11) A profile sider.

## 3.2 Model

The model mainly consists of a User Class, a ChatRoom Class, a Message Class, a dispatcher adapter and commands.

### 3.2.1 User

Class User has four fields -- age, name, school, interest. It also includes all getter and setter functions, as well as a constructor method.

| Modifier and Type | Field | Description |
|---|---|---|
| Private Int | age | The age of the user. |
| Private String | name | The name of the user. |
| Private String | school | The school of the user. |
| Private List<String> | interest | The interest of the user. |
| Boolean | isWarned | It Indicates if the user has been warned before. |

### 3.2.2 ChatRoom

Class ChatRoom represents the chat room. It contains 3 fields. It also includes all getter and setter functions, as well as a constructor method.

| Modifier and Type | Field | Description |
|---|---|---|
| Private String | name | The chat room name. |
| Private String | owner | The owner of the room |
| Private List<String> | interestRequirement | The requirements of the chatRoom. Only the user whose interests match the requirements can join this room. |
| Private Boolean | isPublic | Indicates if the room is public. |

According to the use cases we mentioned above, we use field -- *interestsRequirement* to mark a chat room as private or public. If the field interestsRequirement is null, the chat room is private. Otherwise, the room is public.

### 3.2.3 DispatchAdapter

The adapter interfaces with the models and the controllers. It uses a **Singleton design pattern** in order to guarantee the consistency of the data. It contains 7 concurrent maps to store all room, user and message data and a concurrent list to store all the banned users. It is responsible for handing off requests to corresponding commands for a given user session.

| Modifier and Type | Field | Description |
|---|---|---|
| Private Static DispatchAdapter | singleton | Singleton object. |
| Public Static Map<Session, User> | session2user | Map a session to a user. |
| Public Static Map<String, ChatRoom> | name2ChatRoom | Map a room name to a chatRoom. |

| | | |
|---|---|---|
| Public Static Map<ChatRoom, List<User>> | chatRoom2listUser | Map a room to its corresponding list of users. |
| Public Static Map<List<User>, ChatRoom> | listUser2chatRoom | Map a list of users to its chatRoom. |
| Public Static Map<User, Session> | user2session | Map a user to a session. |
| Public Static Map<User, List<User>> | user2blockList | Map a user to his/her blocked user lists. |
| Public Static List<User> | chatRoomBanList | A list of users banned from accessing all chatRooms. |

As shown in Figure 1., the data structures in DispathAdapter are dependent with each other.

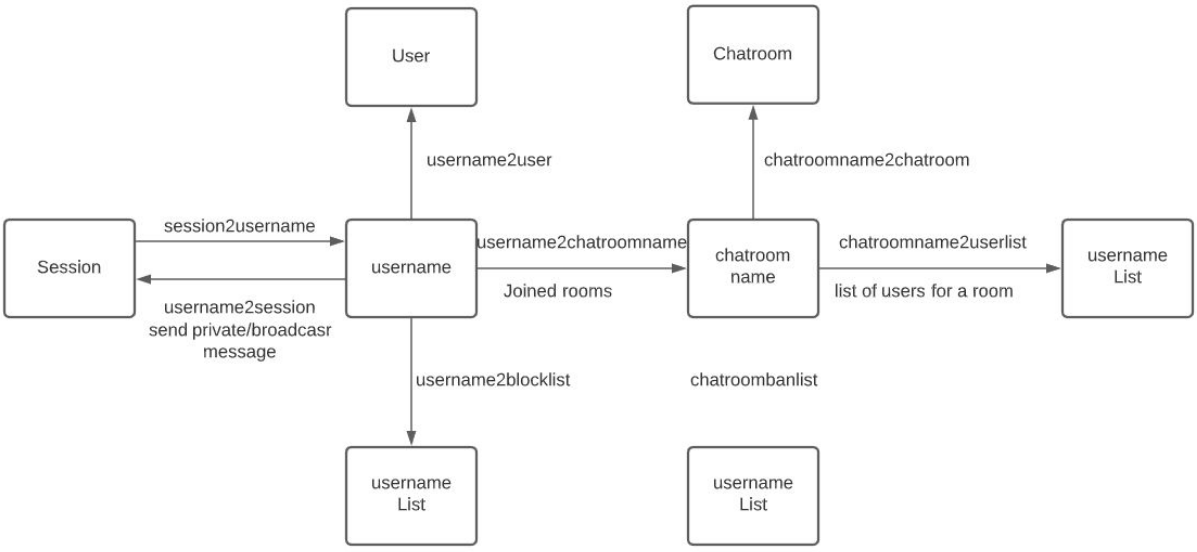


Figure 1. Variable dependency in Dispatch Adapter

### 3.2.4 AbstractCmd

We use an abstract class named Acmd which shares common logic of different sub command classes.

1. Member Fields

| Modifier and Type | Field | Description |
|---|---|---|
| Protected Gson (com.google.gson.Gson) | gson | An Gson object used to parse requests to JSON objects. |

2. Methods

| Modifier | Return Type | Name | Parameters | Description |
|---|---|---|---|---|
| Public Abstract | void | execute | - Session userSession<br>- Map<String, Object> request | Perform the execution of a command. |
| Protected | String | getUser | - Session session | Helper method to get the user by session. |
| Protected | void | sendWSMsg | - Session session<br>- String section<br>- String command<br>- String type<br>- String msg<br>- String ... body | Helper method to send different types of messages via WebSocket. |
| Protected | void | updateAllSession | N/A | Helper method to update all other sessions. |
| Protected | void | dismissChatRoom | - String chatRoomName<br>- String type | Helper method to dismiss a chat room according to the *chatRoomName* and send notification based on the *type*. |
| Protected | void | userLeftChatRoom | - Session userSession<br>- String chatRoomName<br>- String userName<br>- String type | Remove the user (*userName)* from the room (chatRoomName). Notify the removed user and others users in the same room. |

Each concrete command class will implement the abstract execute method. The commands use **Singleton Design Pattern** and **Factory Design Pattern**.

The CmdFactory Class itself uses a **Singleton Design Pattern**. Inside the class, it uses a map to map name to singleton object of each concrete class to increase the lookup efficiency.

According to the requirements of this assignment, we use the following concrete classes to handle different user actions:

1) BanCmd
2) BlockCmd
3) SendMsgCmd
4) CreateRoom
5) DeleteMsgCmd
6) RecallMsgCmd
7) EditMsgCmd
8) GetARoomCmd
9) LeaveRoomCmd
10) InviteCmd
11) LeaveAllRoomCmd
12) PrivateMsgCmd
13) LoginCmd
14) LogoutCmd
15) RemoveUserCmd
16) SendReportCmd

## 3.3 Controller

### 3.3.1 ChatAppController

The chat app controller communicates with all the clients on the web socket. It has a main function, which is the entry point of this chat app. It also includes methods for hosting the app on Heroku.

### 3.3.2 WebSocketController

The web socket controller creates a web socket for the server. It contains methods -- onConnect(), onClose() and onMessage().

# 4. UML Diagram

Since the image is too big to show in the API doc, please refer to this link below:
https://github.com/RiceGradOOCourse/chatapp-sammy-the-owl/blob/master/UML.png?raw=true