BigFoot's API spec doc

Use Case Design

Use case: create the chat room

- Users should set a unique chat room's name, select the room type to be private or public and set the maximum size of the chat room.
- When the private room is created, the admin can send invitations to other users by adding their names to the invite list. When the room size meets the maximum size, the admin cannot add users and the system will give an alert.

Use case: join the chat room

- For Joining Public/Private Room: Users can find a list of public/private chat rooms with name, current number of users, the chat room size and the join button on the top left of the main page. Users can join the chat room by clicking the join button. When the room is full, the user cannot join the room and the system will give an alert message.
- When the user joins the chat room, no chat history appears in the room and the user can find a list of other joined users and one admin.

Use case: leave the chat room

- Users can click the leave button in the chat window to leave.
- For Current User: The chat window will be cleared.
- For Other Users Remaining in the Chat Room: They will receive one leaving message from the system in the chat room. The message shows the leaving user's name and reason. There are three reasons for leaving the chat: voluntarily leaving, connection closed and forced to leave.
- When everyone leaves the chat room, there are no users inside, then the system will delete the chat room.
- When the admin leaves the chat room, the next user on the user lists will be chosen to be the admin.

Use case: switch the chat room

- Users are able to click the chatRoom button, then it will show a list of chat rooms which have already been joined by themselves. Users can choose one room to enter.
- The chat history will be displayed by chronological order, which contains all the users' and system's activities from the joined time to now.

Use case: send message

- Message Content: Users can send text, images (emojis), and/or links in the chat room by adding contents then clicking the send button. Each message starts with the information that “from system/A To everyone/B”.
- Manage Message:
    - Admin has the access to delete others’ messages and ban other users in this chat room. Ban one user means that user is forced to leave the chat room.
    - Both admin and users have the ability to edit and delete their own message after sending.
- Find Hate Speech:
    - Sent messages will be detected and warned by the system. The system will alert the user for the first two times. For the third time, the user will be banned in all the current joined chat rooms. This user cannot join any other room or create a new room.
    - The hate speech will exist until the admin deletes it in the chat room.
- Message Status: In chat rooms, each sent message will be followed by one status information. There are two states, “sending” and “delivered”.
- Message Direction: User has one selection button to choose the sending direction, which can be one specific user or everybody in the chat room.

Use case: registration & login & logout

- A new user needs to register to create an account. The user should create a unique name and enter the password, age, school and interest to register in this application. This information will be stored.
- For the next time, the user only needs to enter the username with the correct password. The login page will be redirected to the main chat page without any login information.
- Users can click the logout button to leave the application. All the chat history will be cleared and the page will be redirected to the login page.

API Endpoints

Login / registration

- POST /register register a new user with age, school, interests and password
- POST /login log in an existing user

Join / leave / connect / invite in chatroom

- POST /chatroom/createChatroom create a chatroom with admin's name, specific room name, type and size
- POST /chatroom/joinChatroom join the chatroom with a unique chatroom name
- POST /chatroom/connectToChatroom connect to chat room with a unique chatroom name
- POST /chatroom/leaveChatroom leave a specific chat room
- POST /chatroom/leaveAllChatroom leave all the chat room
- POST /chatroom/inviteToChatroom invite a user to join the chat room (admin invite!)

Views in chatroom

- GET /chatroom/getChatroomList get a list of chat rooms of that user
- GET /chatroom/getInvitedRoomList get a list of invited rooms of that user
- GET /chatroom/getPublicRoomList get a list of public rooms of that user
- GET /chatroom/userList get all the users from one specific chatroom
- GET /chatroom/getAdmin get the admin of one specific chatroom

Message in chatroom

- GET /chatroom/getMessages get filtered messages for a user in a chatroom.
- POST /chatroom/sendMessage send the message to either everyone or a specific user in the chat room
- POST /chatroom/editMessage edit the message with a unique message id from a chat room
- POST /chatroom/deleteMessage delete the message with a unique message id from a chat room
- POST /chatroom/warnUser warn user because of hateful speech
- POST /chatroom/removeUser delete a user from one chat room or all the chat rooms
- POST /chatroom/report report a user from one chat room

Abstract classes

AUser

This class represents a single user within the app.

- AUser(String *username,* String *pwd,* int *age,* String *school,* String[] *interests*): Constructor, where *username* is the unique username, *pwd* is the password of user, *age* is the age of user, *school* is the school of user and *interests* is the list of interests of user.
- getUsername(): Return the unique username of the user.
- getPwd(): Return the password of the user.
- getAge(): Return the age of the user.
- getSchool(): Return the school of the user.
- getInterests(): Return the list of interests.
- addChatRoom(String *chatroomName*): Add a chatroom to the user's chatroom list.
- getMyChatRooms(): Get the list of chatrooms of the user.
- removeChatRoom(String *chatroomName*): Remove a chatroom from the user's chatroom list by *chatroomName*.
- getStatus(): Get status of the user (normal, warned, banned).
- setStatus(int *status*): Set *status* of the user (normal, warned, banned).
- getNumOfHateSpeech(): Get the number of hate speech. (Users will be banned from all rooms if this number reaches 3).
- setNumOfHateSpeech(int *numOfHateSpeech*): Update the number of hate speeches.
- isOnline(): Return whether the user is online (i.e. has an active session that connects to the server).
- setOnline(boolean *online*): Toggle the online status of the user.
- getOpenChatroom(): Get which chatroom user has opened.
- setOpenChatroom(String chatroom): Set which chatroom user has opened.

AMessage

This class represents a single message within the app.

- AMessage(int *messageID*, String *chatRoomName*, String *content*, String *sender*, String *type*):
  Constructor, where *messageID* is a unique message ID, *chatRoomName* shows the name of chatroom to which this message belongs, *content* is content of the message, *sender* is a sender of the message and *type* is type of the message (normal or direct).
- getMessageID(): Return the ID of the message.
- getType(): Return the type of the message (normal or direct).
- getTimestamp(): Return the timestamp of the message (sent or last edited).
- getChatRoomName(): Return the chatroom name from which this message comes.
- getSender(): Return the sender of the message.
- getContent(): Return the content of the message.
- setContent(String *content*): Reset the content of the message.

AChatroom

- This class represents the chatrooms within the app.
- AChatroom(String *name*, String *type*, int *size*):
  Constructor, where *name* is unique name of chat room, *type* is type of chat room (private vs. public) and *size* is maximum size of the chat room.
- getRoomName(): Get the unique name of chatroom.
- getSize(): Get the size of chatroom.
- getType(): Get the type of chatroom.
- getAdmin(): Get the unique admin of chatroom.
- setAdmin(String *admin*): Set the unique admin of chatroom.
- getNumberOfUsers(): Return current number of users in the room. When this number becomes 0, the room will be destroyed.
- setNumberOfUsers(int *numberOfUsers*): Set the number of users in the chatroom.
- getUsers(): Get the list of users in the chatroom.
- addUser(String *username*): Add a user to the chatroom.
- removeUser(String *username*): Remove a user from the chatroom.
- getBans(): Get the list of banned users in the chatroom.
- addBans(String *username*): Add a user to the banned user list.
- removeBans(String *username*): Remove a user from the ban list.
- getMessages(String *username*): Get the message for a specific user.
- addMessage(): Add the message to the chatroom.
- deleteMessage(int *messageID*): Delete the message from the chatroom. There are two cases inside, one is user delete by himself, another is did by Admin.
- editMessage(int *messageID*, String *newContent*): Edit the message. *messageID* is ID of the message that will be edited and *newContent* is the new content of the message.

Interfaces

IUserFactory
A factory that will make messages.
- makeUser(String *username*, String *pwd*, int *age*, String *school*, String *interests*):
  Make a user object, where *username* is the unique username, *pwd* is the password of user, *age* is the age of user, *school* is the school of user and *interests* is the list of interests of user.

IChatroomFactory
A factory that will make chatroom.
- makeUser(String *username*, String *pwd*, int *age*, String *school*, String *interests*):
  Make a user object, where *name* is the unique name of the chat room, *type* is the type of chat room (private vs. public) and *size* is the maximum size of the chat room.

IMessageFactory
A factory that will make messages.
- makeUser(String *username*, String *pwd*, int *age*, String *school*, String *interests*):
  Make a user object, where *messageID* is a unique message ID, *chatRoomName* shows the name of chatroom to which this message belongs, *content* is content of the message, *sender* is a sender of the message and *type* is type of the message (normal or direct).

IUserStore
The interface of the user store, which will manage the user list and related actions.
register(String username, String pwd, int age, String school, String[] interests):
* Register the new user.
* @param username String username
* @param pwd String password
* @param age Integer age
* @param school String school name
* @param interests String interests (combined)
* @return the registered user or null

login(String username, String pwd):
* Log in an existing user.
* @param username String username
* @param pwd String password
* @return the logged user or null

getChatRoomForUser(String username):
* Return a list of users in the chat room.
* @param username String username of the user

* @return List of chat rooms of that user

online(String username, Session userSession):
* Mark a user as online and store its unique session.
* @param username String username
* @param userSession Session userSession

offline(String username, Session userSession):
* Mark a user as offline and remove its session from the session map.
* @param username String username
* @param userSession Session userSession

addChatRoomToList(String username, String chatroomName):
* Add a chatroom to the user's chatroom list.
* @param username unique username
* @param chatroomName name of the chatroom to be added

removeChatRoomFromList(String username, String chatroomName):
* Remove a chatroom from user's chatroom list.
* @param username unique username
* @param chatroomName name of the chatroom to be removed

invitedToJoin(String chatroomName, String username):
* Invite the user to join the chatroom.
* @param username unique username
* @param chatroomName name of the chatroom that invite the user

getAllUsers():
* Return a list of all users.
* @return list of all users

IChatroomstore

The interface of the chatroom store, which will manage the chatroom list and related actions.

getChatRoom(String chatroomName):
* Get the chatroom by its name.
* @param chatroomName Unique chatroom name
* @return the chatroom

getAllPublicChatRooms():
* Get all public chat rooms on the server.
* @return list of all public chatrooms.

createChatRoom(String chatroomName, String type, String username, int size):

* Create a chatroom with specific name and size.
* @param username String username
* @param chatroomName String unique chatroomName
* @param type String type
* @param size Int size of the chatroom
* @return the chatroom or null

addUserToChatroom(String chatroomName, String username):
* Add a user to the chatroom.
* @param chatroomName unique chatroom name
* @param username unique user name

removeUserFromChatroom(String chatroomName, String username):
* Remove a user from the chatroom.
* @param chatroomName unique chatroom name
* @param username unique user name

getUserList(String chatroomName):
* Get the user list from a chatroom.
* @param chatroomName unique chatroom name
* @return list of users

getMessageForUser(String username, String chatroomName):
* Get the message for a user in a chatroom.
* @param username unique username
* @param chatroomName unique chatroom name
* @return list of messages for that specific user

getAdmin(String chatroomName):
* Get the admin of the chatroom.
* @param chatroomName unique chatroom name
* @return username of the admin

setAdmin(String chatroomName, String username):
* Set the admin of a chatroom.
* @param chatroomName unique chatroom name
* @param username username of the admin

IMessagestore

The interface of the message store, which will manage actions related to messages.
sendMessage(String type, String sender, String receiver, String content, String chatroomName):
* Send the message to either everyone in the chat room or a specific user.

* @param content String content of the message
* @param type String type of the message (direct or normal)
* @param sender String sender of the message
* @param receiver String receiver of the message, if empty means send to everyone
* @param chatroomName String chatroomName in which the message will be sent.
* @return Success or failure to deliver the message

editMessage(int messageID, String chatroomName, String content):

* Edit the message from a chat room.
* @param messageID Int message ID of message that will be edited.
* @param chatroomName String chat room name in which message resides.
* @param content String content of the new message

deleteMessage(int messageID, String chatroomName):

* Delete the message from a chat room.
* @param messageID Int message ID of message that will be deleted.
* @param chatroomName String chat room name in which message resides.

IDispatcherAdapter

The class to maintain and organize the state of the chat app.

register(String *username*, String *pwd*, int *age*, String *school*, String *interests*):

* Register the new user.
* @param username String username
* @param pwd String password
* @param age Integer age
* @param school String school name
* @param interests String interests (combined)
* @return the registered user or null

login(String username, String pwd):

* Log in an existing user.
* @param username String username
* @param pwd String password
* @return the logged user or null

online(String username, Session userSession):

* Mark a user as online and store its unique session.
* @param username String username
* @param userSession Session userSession

offline(String username, Session userSession):

* Mark a user as offline and remove its session from the session map.
* @param username String username

* @param userSession Session userSession

getChatRoom(String chatroomName):
* Return the chatroom with that specific name.
* @param chatroomName The chatroom name
* @param chatroomName unique chatroom name
* @return the chatroom or null

createChatRoom(String username, String chatroomName, String type, int size):
* Create the chatroom with a specific name and size.
* @param username String username
* @param chatroomName String unique chatroomName
* @param type String type
* @param size Int size of the chatroom
* @return the chatroom or null

joinChatRoom(String username, String chatroomName):
* Join the chatroom with a unique chat room name.
* @param username String username
* @param chatroomName String chat room name
* @return the chatroom or null

inviteToJoin(String username, String chatroomName):
* Invite a user to join the chat room.
* @param username String invited username
* @param chatroomName String chat room name
* @return the chatroom or null

getChatRoomForUser(String username):
* Return a list of users in the chat room.
* @param username String username of the user
* @return List of chat rooms of that user

getAllUsers(String chatroomName):
* Return a list of all users from the chatroom.
* @return List of usernames.

getAllPublicChatRooms():
* Return a list of all public chat rooms.
* @return List of public chatrooms.

deleteUser(String username, String chatroomName):
* Delete a user from the chat room.
* @param username String username to be deleted

* @param chatroomName String chat room name
* @return Success or failure to delete the user

getAdmin(String chatroomName):
* Return the admin of the room.
* @param chatroomName String chatroom to be checked.
* @return username of the admin

setAdmin(String chatroomName, String username):
* Set the admin of the room.
* @param chatroomName String chatroom to be set.
* @param username String username of the new admin.
* @return username of the new admin.

sendMessage(String content, String type, String sender, String receiver, String chatroomName):
* Send the message to either everyone in the chat room or a specific user.
* @param content String content of the message
* @param type String type of the message (direct or normal)
* @param sender String sender of the message
* @param receiver String receiver of the message
* @param chatroomName String chatroomName in which the message will be sent.
* @return Success or failure to deliver the message

getMessageForUser(String username, String chatroomName):
* Get the messages based on a specific user in a chat room.
* @param username String username of the user.
* @param chatroomName String chat room name.

deleteMessage(int messageID, String chatroomName):
* Delete the message from a chat room.
* @param messageID Int message ID of message that will be deleted.
* @param chatroomName String chat room name in which message resides.

editMessage(int messageID, String chatroomName, String content):
* Edit the message from a chat room.
* @param messageID Int message ID of message that will be edited.
* @param chatroomName String chat room name in which message resides.
* @param content String content of the new message

leaveRoom(String username, String chatroomName):
* Leave a specific chat room.
* @param username String username of the user that will leave the room
* @param chatroomName String chat room name from which the user will leave

Design Pattern

Null object design pattern:

- NullUser class can avoid internal errors.

Factory design pattern:

- User: has NullUser and normal user concrete classes, using factory design pattern can help to create objects.
- Chatroom: has nullChatroom and normal room classes, using factory design pattern can help to create objects.
- Message: has DirectMessage, NormalMessage, NullMessage classes, using factory design pattern can help to create objects.

Singleton design pattern:

- Each store class uses make and only to create one instance to save spaces.