ChatApp API Specification Document

Team: Bullet(Team J)

Member:
Siyu Xie(team Lead)
Huijia Zhu (doc lead)
Pengyu Zhang (dev)
Su Zhang (dev)
Suyue Zhang (dev)

Table of Contents

API Javadoc

We have hosted our API Javadoc on surge:
bullet-chatapp-javadoc.surge.sh

Design Decision

1. Following is the tech Stack we use in this project:
   - Frontend: vanilla JavaScript
   - Backend: Java Spark
   - Communication protocol: Websocket
2. Following are the design patterns we use in our implementation:
   - Command design pattern
   - Singleton design pattern
   - Factory method design pattern
3. We called websocket messages sent from frontend to backend as request, websocket message sent from backend to frontend as response. The format of websocket messages is:

```
{
    "request": "...",
    "data":
        {
            ......
        }
}
```

```
{
    "response": "...",
```

```
    "data":
        {
            ......
        }
}
```

Use Case

Use Case Name	Registration
Participating Actors	User
Flow of Events	1. User loads registration page 2. User fills in Username, Age, School, Interests and clicks "Register" button 3. Session between User and Server has been established. 4. Server creates an instance of User class for the user. 5. Server sends a list of public rooms he/she can join and a list of rooms he/she already joined back to the user. 6. UI updates to Chat Room Main Window.
Alternative Flows 1	/
Entry Conditions	User must fill in Username, Age, School, Interests

Use Case Name	Get Own Profile
Participating Actors	User
Flow of Events	1. User clicks "Profile"on the Main Window. 2. Server sends back the user's profile to the user. 3. UI updates.
Alternative Flows 1	/
Entry Conditions	/

Use Case Name	Create Chat Room
Participating Actors	User
Flow of Events	1. User clicks "Create Room" button, fills in ChatRoomName, select whether this room is private or public and clicks "Confirm" button

	2. Server creates an instance of class ChatRoom, sets the user as chat room's admin. 3. Server sends an updated list of rooms he/she already joined back to the user. 4. UI updates.
Alternative Flows 1	1. If the room is public, Server sends an updated list of public rooms back to all users in the system. 2. UI updates.
Entry Conditions	User must fills in Username, Age, School, Interests

Use Case Name	Send Message
Participating Actors	User
Flow of Events	1. User inputs message content, selects a message receiver and clicks the "Send" button. 2. Server creates an instance of class Message, adds it to the current chat room's message list. 3. Server sends the message to the particular receiver(s). 4. Server notifies the sender that receiver(s) has received the message. 5. UI updates.
Alternative Flows 1	1. If the user is banned, the message will not be sent to receivers. Server sends error message back to sender. 2. UI updates.
Alternative Flows 2	1. If the message contains the "hate" keyword and it is the first time the user sends "hate", the message will not be sent to receivers. Server sends back a warning message to the sender. 2. UI updates.
Alternative Flows 3	1. If the message contains the "hate" keyword and it is the second time the user sends "hate", the message will not be sent to receivers. 2. Server sends back an error message to sender, indicating he/she is banned by all chat rooms he/she has joined. 3. Server sends a system message to users in all the chat rooms the sender joins, indicating that he is banned. 3. Server sends updated user lists to users in all the chatrooms the sender joins(indicating the sender is banned now). 2. UI updates.
Entry Conditions	Users must fill in message content.

Use Case Name	Edit Message

Participating Actors	User
Flow of Events	1. User right clicks a message, edits content and clicks the “Send” button. 2. Server checks the identification of the user. If User is the owner of the message: 3. Server finds the message,changes its content. 4. Server sends the message to the particular receiver(s). 5. Server notifies the sender that the receiver has received the message. 6. UI updates.
Alternative Flows 1	1. If the user is not allowed to recall the message, Server sends back an error message to the user.
Alternative Flows 2	1. If the user is banned, messages will not be edited. Server sends back error message to sender. 2. UI updates.
Alternative Flows 3	1. If the message contains the “hate” keyword and it is the first time the user sends “hate”, the message will not be sent to receivers. Server sends back a warning message to the sender. 2. UI updates.
Alternative Flows 4	1. If the message contains the “hate” keyword and it is the second time the user sends “hate”, the message will not be sent to receivers. 2. Server sends back an error message to the sender, indicating he is banned by all chat rooms he has joined. 3. Server sends a system message to users in all the chat rooms the sender joins, indicating that he is banned. 3. Server sends updated user lists to users in all the chatrooms the sender joins(indicating the sender’s state is banned now). 2. UI updates.
Entry Conditions	Users must fill in message content.

Use Case Name	Recall Message
Participating Actors	User(Admin)
Flow of Events	1. User right clicks a message. A list of operations will appear. User then clicks the “recall” option. 2. Server checks the identification of the user. 3. If User is the owner of the message or admin. Server sends the message recall response to the particular receiver(s). 4. UI updates. That message will disappear.
Alternative Flows 1	1. If the user is not allowed to recall the message, Server sends back an error message to the user. 2. UI updates.
Entry Conditions	User must right click the message and then click “recall” option.

Use Case Name	Delete Message
Participating Actors	User
Flow of Events	1. User right clicks message. A list of operations will appear. User then clicks the “delete” option. 2. Front end will delete the message from cache.
Alternative Flows 1	/
Entry Conditions	User must right click the message and then click “delete” option.

Use Case Name	Exit Chat Room
Participating Actors	User(Admin)
Flow of Events	1. User clicks the “exit” button on the chat room in his/her own chat room list. 2. Server sends a system message to users in the chat room indicating that he dropped out. 3. Server sends updated user lists to users in the chatroom(indicating the sender left now). 4. Server sends updated own chat room lists back to the exit user. 5. UI updates.
Alternative Flows 1	If the user is chat room’s admin and there are more than one people in chat room: 1. Server sends a system message to users in the chat room, indicating that he left. 2. Server sends a system message to users in the chat room, another user has been chosen as admin. 2. Server sends updated user lists to users in the chatroom(indicating the sender left now). 3. Server sends updated own chat room lists back to the user. 4. UI updates.
Alternative Flows 2	If the user is chat room’s admin and there only one people in chat room: 1. Server sends updated own chat room lists back to the user. 2. If the room is public, Server sends an updated list of public rooms back to all users in the system. 3. UI updates.
Entry Conditions	User clicks the exit button on the chat room in his/her own chat room list.

Use Case Name	Join Chat Room
Participating Actors	User
Flow of Events	1. User chooses any public room from the public room list, clicks the “Join” button. 2. Server will send a ask_approve response to the admin in this chat room.
Alternative Flows 1	/
Entry Conditions	Users must click the “Join” button of the chat room.

Use Case Name	Approve Join Request
Participating Actors	Admin
Flow of Events	1. There is a notification for the admin that someone wants to join his chat room. If admin click “Approve” on the notification: 2. Server sends a system message to users in the chat room that another user has been approved to join the chat room. 3. Server sends updated user lists to users in the chatroom(indicating a new user has joined now). 4. Server sends updated own chat room lists back to the user who asked for approval.
Alternative Flows 1	If admin click “Deny” on the notification: 1. Server sends back an error message to the user who asked for approval.
Entry Conditions	/

Use Case Name	Ban User
Participating Actors	Admin(User)
Flow of Events	1. Admin(user) clicks the “Ban” button on a user in the user list of a chat room. 2. Server checks the identification of the user. If User is the admin: 3. Server sends back an error message to the person being banned, indicating he is banned by the chat room’s admin. 4. Server sends a system message to users in that chat room, indicating the user is banned. 5. Server sends an updated user list to users in that chatroom(indicating the user is banned now). 6. UI updates.
Alternative Flows 1	If admin click “Deny” on the notification:

	If User is not the admin: 1. Server sends back an error message to the user.
Entry Conditions	Admin(user) clicks “Ban” button on a user in the user list of a chat room

Use Case Name	Invite User
Participating Actors	Admin(User)
Flow of Events	1. Admin(user) fills in user id and clicks the “Invite” button on a chat room. 2. Server checks the identification of the user. If User is the admin: 3. sends updated own chat room lists back to the user being invited. 4. Server sends a system message to users in that chat room, indicating the user has joined. 5. Server sends an updated user list to users in that chatroom(indicating the user has joined now). 6. UI updates.
Alternative Flows 1	If User is not the admin: 1. Server sends back an error message to the user.
Alternative Flows 2	If User been invited is not exists: 1. Server sends back an error message to the user.
Alternative Flows 3	If User been invited is already in the chat room: 1. Server sends back an error message to the user.
Entry Conditions	Admin(user) fills in user id and clicks the “Invite” button on a chat room.

Use Case Name	Report
Participating Actors	User
Flow of Events	1. User chooses any user from the user list, clicks the “Report” button. 2. Server will send a report response to the admin in this chat room.
Alternative Flows 1	/
Entry Conditions	User chooses any user from user list, clicks “Report” button

Use Case Name	Exit System

Participating Actors	User(Admin)
Flow of Events	1. User closes the website. 2. For each chat room the user has joined: If user is not the chat room's admin: 3. Server sends a system message to users in all the chat rooms the sender joined, indicating that he left because the connection closed. 4. Server sends updated user lists to users in all the chatrooms the sender joins(indicating the sender left now). 5. UI updates. If the user is a chat room's admin and there are more than one people in chat room: 3. Server sends a system message to users in that chat room the sender joined as admin, indicating that he left. 4. Server sends a system message to users in that chat room that another user has been chosen as admin. 5. Server sends an updated user list to users in that chatroom(indicating the sender exits now). 6. UI updates. If the user is chat room's admin and there only one people in chat room: 3. If the room is public, Server sends an updated list of public rooms back to all users in the system(delete this room). 4. UI updates.
Entry Conditions	User clicks the exit button of the chat room in his/her own chat room list.

Interface and Class Description

Interface

1. ICommand
   Interface Command works as the interface of the concrete commands. It requires a simple function to execute the necessary functions.

Method	Description
void execute()	Execute the required command.

Concrete Classes

1. Command classes

Command Name	Description
ApproveCmd	Approve a user's request to join a chat room by the admin.
BanUserCmd	Ban a user from a chat room by the admin.
CreateRoomCmd	Create a chat room.
EditMsgCmd	Edit the content of a sent message.
InviteUserCmd	Invite a user to join a chat room by the admin.
JoinRoomCmd	Send out a user's request to join a chat room.
RecallMsgCmd	Recall a message from the chat room.
SendMsgCmd	Send a message to the chat room or a specific user.

2. User
   A user can join or exit chat rooms and send messages in chat rooms.

Field	Description
private String username	Username of the user.
private Integer userID	The userID assigned to the user by the system.
private Integer age	Age of the user.
private String school	School of the user.
private String interest	Interest of the user.
private Integer numOfHate	Number of times the user has sent "hate".
private List<ChatRoom> joinedChatRooms	The list of joined chat rooms.
private Session session	The websocket session associated with the user.

Method	Description
public User(String username, Integer age, String school, String interest, Session session)	The constructor.
public Integer getUserID()	Get the userID of the user.

public Integer getAge()	Get the age of the user.
public Integer getNumOfHate()	Get the number of times the user has used “hate” in their messages.
public List<ChatRoom> getJoinedChatRooms()	Get the list of chat rooms the user has joined.
public String getSchool()	Get the school of the user.
public String getInterest()	Get the interest of the user.
public void addChatRoom	The user has joined a chat room.
public void sentHate()	The user has sent a message that contains “hate”.
public Session getSession()	Get the websocket session of the user.

3. ChatRoom
   A chat room is where users have conversations.

Field	Description
private Integer chatRoomID	The unique chat room ID that the system has generated for the chat room.
private String chatRoomName	The name of the chat room.
private Boolean isPrivate	The attribute that tells if the chat room is private or public.
private List<User> userList	The list of users that have joined the chat room.
private List<Message> msgList	The list of messages that have been sent in the chat room.
private User admin	The admin of the chat room.
private List<User> bannedUserList	The list of users that have been banned from the chat room.

Method	Description
public User(String username, Integer age, String school, String interest, Session session)	The constructor.
public Integer getChatRoomID()	Get the chat room ID.
public Boolean getIsPrivate()	Tell if the chat room is private or public.

public List<User> getUserList()	Get the user list of the room.
public List<Message> getMsgList()	Get the message list of the room.
public User getAdmin()	Get the admin of the chat room.
public void addUser(Integer userID)	Add a new user to the room.
public void deleteUser(Integer userID)	Delete a user from the room.
public void addMsg(Message newMsg)	Add a new message into the message log.
public void removeMsg(Integer msgID)	Remove a message from the chat room.
public Message getMsg(Integer msgID)	Get a specific message according to the message ID.
public Boolean hasUser(Integer userID)	Tell if a user is in the chat room.
public void banUser(User user)	Ban a certain user from the chat room.
public Boolean isBanned(Integer ID)	Tell if the user is banned.

4. Message
   A message is sent by users in chat rooms.

Field	Description
private Integer msgID	The unique message ID that the system generated for the message object.
private String type	Type of the message(regular/system/admin).
private Timestamp timestamp	The timestamp of the message.
private Integer senderID	The user ID of the sender.
private Integer receiverID	The user ID of the receiver of this message.
private String content	The content of the message.

Method	Description
public Message(String type, Timestamp timestamp, Integer senderID, Integer receiverID)	The constructor.
public Integer getMsgID()	Get the message ID.
public String getType()	Get the type of the message.
public Timestamp getTimestamp()	Get the timestamp of the message.

public Integer getSenderID()	Get the sender's user ID.
public Integer getReceiverID()	Get the receiver's user ID.
public void editContent(String content)	Edit the content of the message.

5. UserManager
   The UserManager is a singleton class that manages all the users online.

Field	Description
List<User> userList	The list of users online.
private static UserManager ONLY	The singleton object.
List<User> bannedUserList	The list of users that has been banned.

Method	Description
private UserManager()	Private constructor.
public static UserManager getOnly()	Get the singleton object of this class.
public void addUser(User newUser, Session session)	Add a user into the list.
public void deleteUser(Integer userID)	Delete a user from the list.
public User getUser(Integer userID)	Get a specific user with userID.
public void banUser(User user)	Ban a certain user from all the chat rooms.
public Boolean isBanned(Integer ID)	Tell if the user is banned.

6. ChatRoomManager
   The ChatRoomManager is a singleton class that manages all the users online.

Field	Description
private List<ChatRoom> chatRoomList	The chat room list.
private static ChatRoomManager ONLY	The singleton object.

Method	Description
private ChatRoomManager()	The private constructor.
public static ChatRoomManager getOnly()	Get the singleton object of this class.

public void addChatRoom(ChatRoom chatRoom)	Add a chat room into the list.
public void deleteRoom(ChatRoom chatRoom)	Delete a chat room from the list.
public ChatRoom getChatRoom(Integer chatRoomID)	Get the chat room object according to the chat room ID.