

# Chat App Design Doc

Team: Bullet(Team J)

Member:

Siyu Xie(team Lead)

He Wang (tech lead)

Huijia Zhu (doc lead)

Pengyu Zhang (dev)

Su Zhang (dev)

Suyue Zhang (dev)





# Agenda

- Use case
- Design Decision
- Interface/Class Description
- GUI Introduction



## Use case

- User inputs basic information for registration.
- User checks his/her profile.
- User creates a chat room.
- User applies for joining a chat room.
- Admin approves user application.
- Admin invite a user to join his chat room.
- User sends, edit, recall, delete message to particular receiver(s).
- User reports other users.
- Admin deletes message.
- Admin bans user.
- User exits a chat room.
- User closes connection.





































## Design Decision





























1. In this project, We use the following design patterns in our implementation: command, singleton.
2. We used websockets to communicate between the frontend and backend.
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":  
    {  
      .....  
    }  
}
```












## Interface/Class Description









  ChatRoom	
  chatRoomName	String
  bannedUserList	List<User>
  ChatRoom(String, Boolean, Integer)	
  addUser(Integer)	void
  deleteUser(Integer)	void
  addMsg(Message)	void
  removeMsg(Integer)	void
  getMsg(Integer)	Message
  hasUser(Integer)	Boolean
  banUser(User)	void
  isBanned(Integer)	Boolean
  isPrivate	Boolean
  chatRoomID	Integer
  userList	List<User>
  msgList	List<Message>
  admin	User

  Message	
  msgID	Integer
  type	String
  timestamp	Timestamp
  senderID	Integer
  receiverID	Integer
  content	String
  Message(String, Timestamp, Integer, Integer)	
  editContent(String)	void
  msgID	Integer
  type	String
  senderID	Integer
  timestamp	Timestamp
  receiverID	Integer

  User	
  username	String
  User(String, Integer, String, String, Session)	
  addChatRoom(ChatRoom)	void
  sentHate()	void
  userID	Integer
  age	Integer
  numOfHate	Integer
  interest	String
  session	Session
  school	String
  jointChatRooms	List<ChatRoom>

## Interface/Class Description

	userManager	
	◦ userList	List<User>
	◦ bannedUserList	List<User>
	ONLY	userManager
	userManager()	
	getOnly()	userManager
	addUser(User, Session)	void
	deleteUser(Integer)	void
	getUser(Integer)	User
	banUser(User)	void
	isBanned(Integer)	Boolean

	chatRoomManager	
	chatRoomList	List<ChatRoom>
	ONLY	chatRoomManager
	chatRoomManager()	
	getOnly()	chatRoomManager
	addChatRoom(ChatRoom)	void
	deleteRoom(ChatRoom)	void
	getChatRoom(Integer)	ChatRoom




## Interface/Class Description

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.

# GUI Introduction

Chat Room / [Back to Login](#) / [Back to Register](#)



**User Name**

School

Age

Interest

**Your Room**

**Chatting Room 1**

Hello, Are you there?

**Chatting Room 2**

Hello, Are you there?

**Chatting Room 3**

Hello, Are you there?

**Public Room**

**Chatting Room 1**

Hello, Are you there?

[Join](#)

**Chatting Room 2**

Hello, Are you there?

[Join](#)

**Chatting Room 3**

Hello, Are you there?

[Join](#)

name

myTest

**Pengyu has joined chatting room.**

user1 6:00 10.30 2020

myTest

user1 6:00 10.30 2020

myTest

**Pengyu has joined chatting room.**

user1 6:00 10.30 2020

myTestmyTestmyTestmyTestmyTestmyTestmyTestmyTestmyTestmyTestmyTestmyTest

**Group Member**

Pengyu

Siyu

Pengyu1

Pengyu2

Pengyu3

send to: All emoji

[Send Message](#)

Chat Room / [Register](#) / [Login](#)

## Register

### User Name

### Age

### Interest

### School

[Register](#)

[Login](#)