



# Final Chat App - dragonBall

11/2/2019

# Team members



Jiangguo Zhang  
(team lead)



Yifan Wang (tech  
lead)



Sibo Wang (doc  
lead)



Shunda Huang (dev)



Can Sun (dev)



Enze Zhong (dev)



Tingting Zhou (dev)

# Contents



## Design Decisions

UML Diagram  
Design Patterns



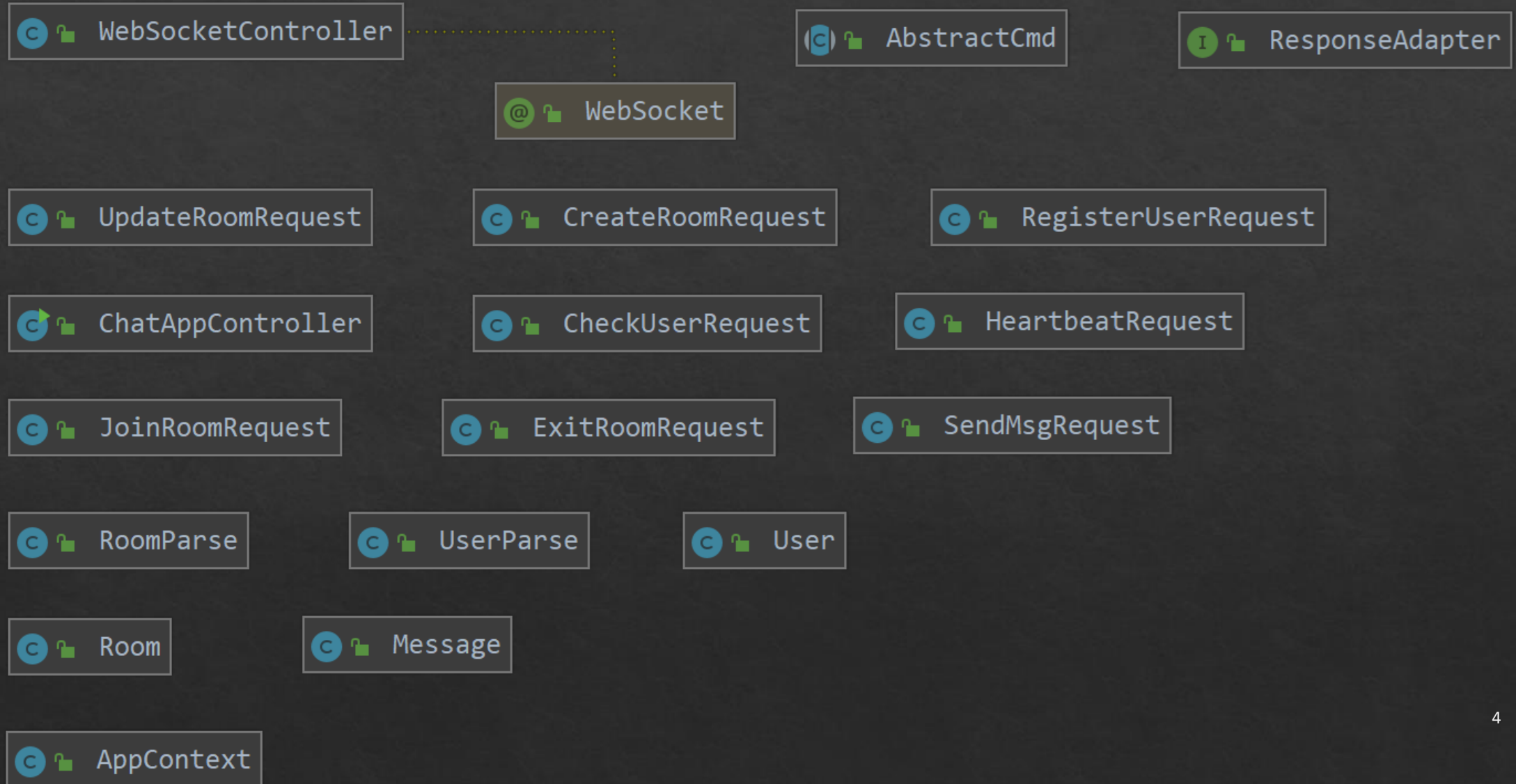
## API

Login/Register  
Create/Join/Edit a room  
Send a message  
Leave a room



## Demo

# UML Diagram



# Design Patterns



MVC Design Pattern



Command Design Pattern



Union Design Pattern



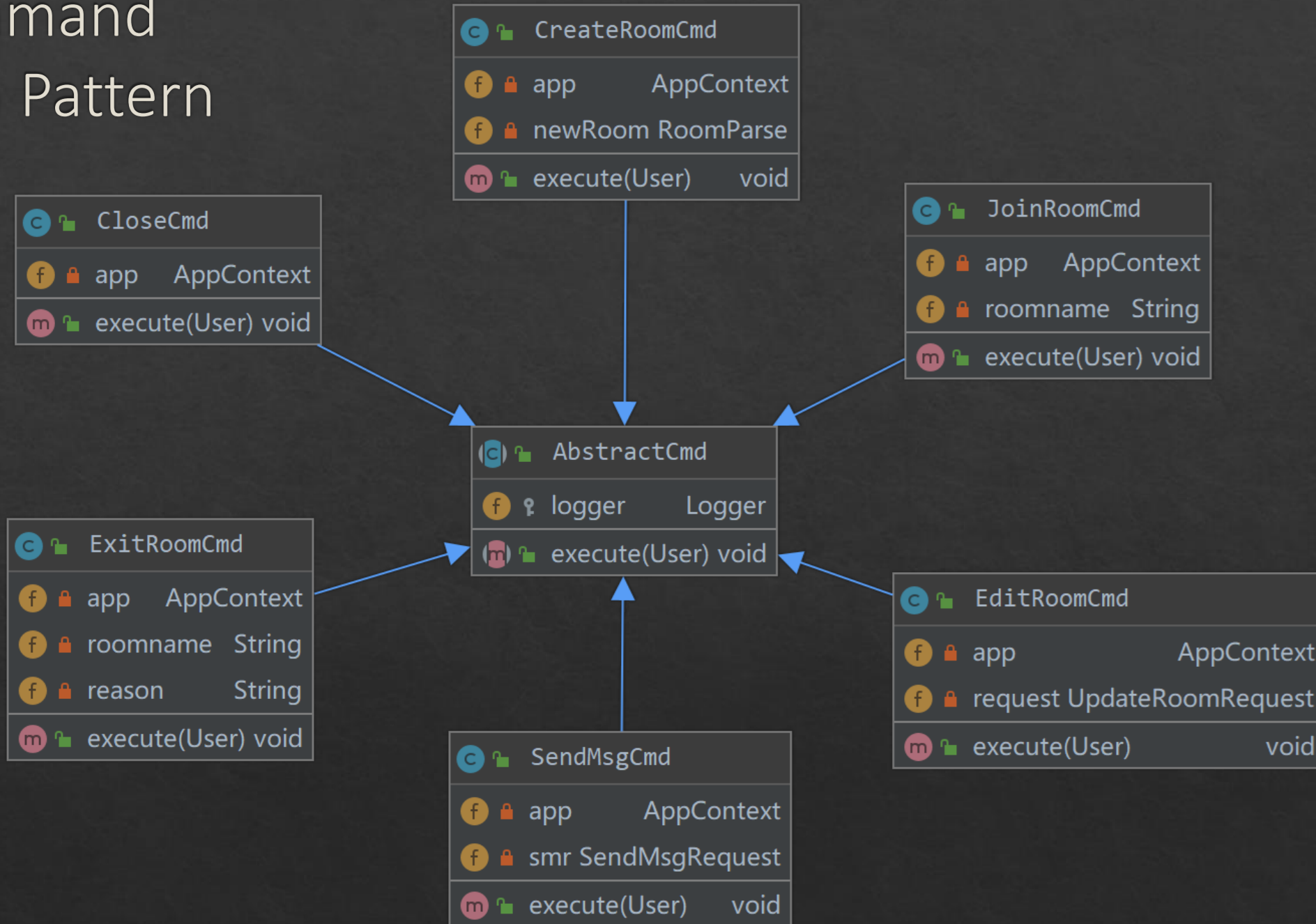
Singleton Design Pattern



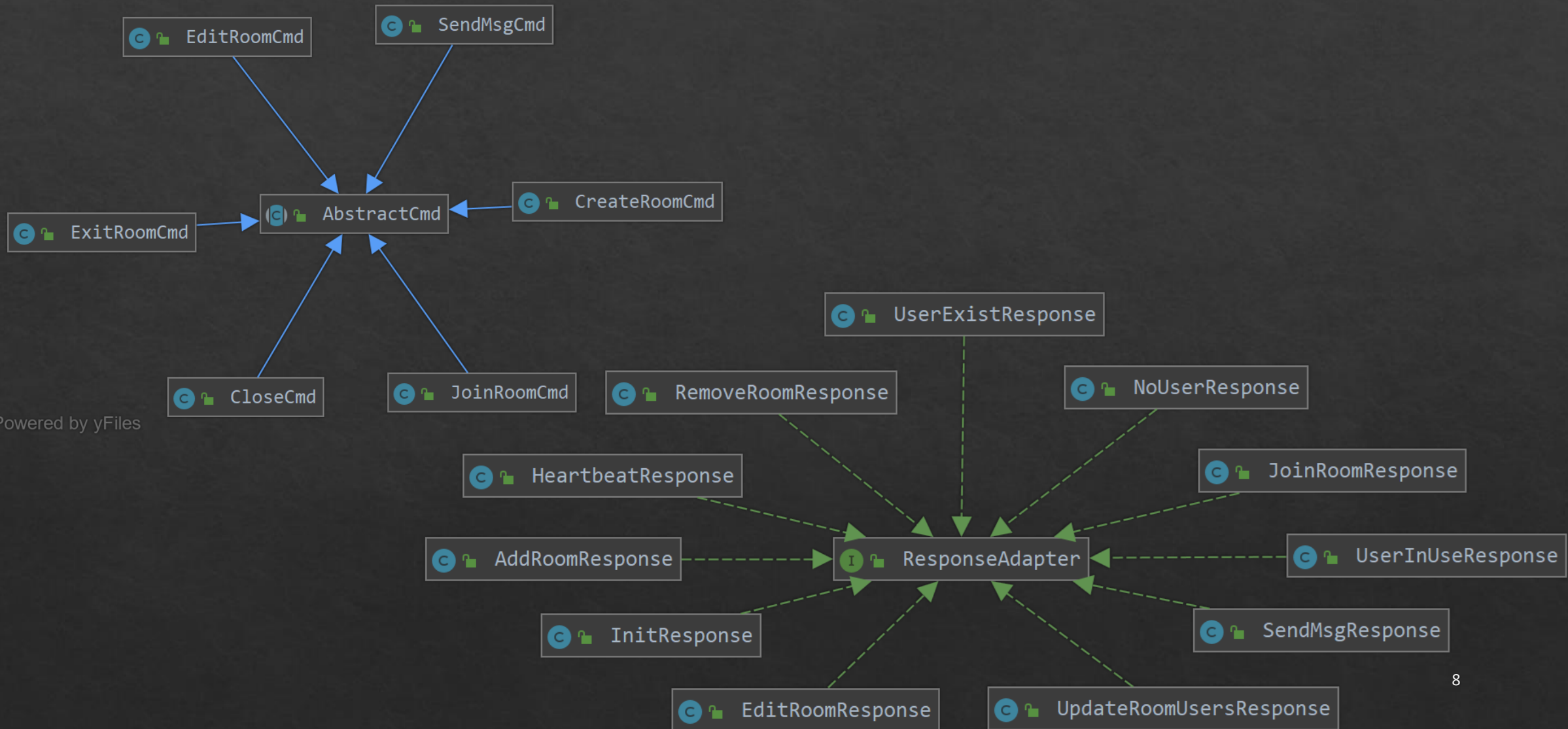
# MVC Design Pattern

Model	AppContext, ResponseAdapter, etc.
View	GUI
Controller	ChatAppController, WebSocketController

# Command Design Pattern









# Union Design Pattern





# Singleton Design Pattern

	AppContext	
	only	AppContext
	users	ArrayList<User>
	roomModels	ArrayList<Room>
	roomMap	HashMap<String, Room>
	sessionUserMap	Map<Session, User>
	getOnly()	AppContext
	getRoomModels()	ArrayList<Room>
	addSessionUser(Session, User)	void
	closeSession(Session)	void
	findUser(Session)	User
	getUsers()	ArrayList<User>
	addUser(User)	void
	addRoom(Room)	void
	removeUser(User)	void
	removeRoom(Room)	void
	joinRoom(int, String)	Room
	sendMessage(int, String, String, String)	Room
	leaveRoom(int, String)	Room
	updateRoom(int, String, int, int, ArrayList<String>, ArrayList<String>)	Room
	createRoom(String, int, int, ArrayList<String>, ArrayList<String>)	Room
	getQualifiedUsers(int)	ArrayList<User>
	detectHate(String, Room, User)	void
	deleteRoomIfEmpty(Room)	void
	sendSystemMessage(Room, User)	void
	kickUnqualifiedUsers(Room)	void
	getRoomModel(String)	Room
	isExistedRoom(String)	boolean

# Structure



**Frontend**

react – redux



**Data**

json



**Backend**

java

# Tool Classes

@WebSocket
m inputBufferSize() int
m maxBinaryMessageSize() int
m maxIdleTime() int
m maxTextMessageSize() int
m batchMode() BatchMode

WebSocketController
m onConnect(Session) void
m onClose(Session, int, String) void
m onMessage(Session, String) void

ChatAppController
f app AppContext
m main(String[]) void
m handleRegisterUser(Session, UserParse) void
m handleCheckUser(Session, String) void
m handleClose(Session) void
m handleSendMsg(Session, SendMsgRequest) void
m handleCreateRoomRequest(Session, RoomParse) void
m handleJoinRoomRequest(Session, String) void
m handleExitRoomRequest(Session, String, String) void
m handleEditRoomRequest(Session, UpdateRoomRequest) void
m handleHeartbeatRequest(Session, long) void
m getHerokuAssignedPort() int

AppContext
f only AppContext
f users ArrayList<User>
f roomModels ArrayList<Room>
f roomMap HashMap<String, Room>
f sessionUserMap Map<Session, User>
m getOnly() AppContext
m getRoomModels() ArrayList<Room>
m addSessionUser(Session, User) void
m closeSession(Session) void
m findUser(Session) User
m getUsers() ArrayList<User>
m addUser(User) void
m addRoom(Room) void
m removeUser(User) void
m removeRoom(Room) void
m joinRoom(int, String) Room
m sendMessage(int, String, String, String) Room
m leaveRoom(int, String) Room
m updateRoom(int, String, int, int, ArrayList<String>, ArrayList<String>) Room
m createRoom(String, int, int, ArrayList<String>, ArrayList<String>) Room
m getQualifiedUsers(int) ArrayList<User>
m detectHate(String, Room, User) void
m deleteRoomIfEmpty(Room) void
m sendSystemMessage(Room, User) void
m kickUnqualifiedUsers(Room) void
m getRoomModel(String) Room
m isExistedRoom(String) boolean

Login/Register

Create/Join/Edit a room

Send a message

Leave a room

API

Login/Register

Create/Join/Edit a room

Send a message

Leave a room



# handleCheckUser

## Request

- checkUserRequest
  - Payload: username

## Logic

- Return = user in memory ? data : noUser.
- User in use.

## Response

- initResponse, noUserResponse, userInUseResponse

```
let user: {  
  username: string,  
  age: number,  
  school: string,  
  area: string  
};
```



# handleRegisterUser

Request

- registerUserRequest
- Payload: user

Logic

- Return = user in use ? userExist : data.

Response

- initResponse, userExistResponse

```
let user: {  
  username: string,  
  age: number,  
  school: string,  
  area: string  
};
```

# handleCreateRoom

Request

- createRoomRequest
- Payload: room

Logic

- Return = room exist ? none : new room.

Response

- addRoomResponse, sendMsgResponse

```
let room : {  
  owner: string,  
  roomname: string,  
  ageMin: number,  
  ageMax: number,  
  areas: string[],  
  schools: string[],  
  users: string[],  
  msgs: typeof msg[]  
};
```

# handleJoinRoom

## Request

- joinRoomRequest
  - Payload: roomname

## Logic

- Send message to user and other members in the room.
- Update user list in the room.

## Response

- sendMsgResponse, joinRoomResponse, updateRoomUsersResponse

# handleUpdateRoom

## Request

- updateRoomRequest
  - Payload: roomname, ageMin, ageMax, areas, schools

## Logic

- Update constraint and send system message.
- Kick unqualified members. Update rooms for other members.

## Response

- addRoomResponse, sendMsgResponse, removeRoomResponse

# handleSendMsg

Request

- sendMsgRequest
  - Payload: roomname, msg

Logic

- Send message to target.
- Detect “hate”.

Response

- sendMsgResponse

```
let msg: {  
  from: string,  
  to: string,  
  toAll: boolean,  
  text: string,  
  isSysMsg: boolean,  
  timestamp: number  
};
```



# handleExitRoom

## Request

- exitRoomRequest
  - Payload: roomname

## Logic

- Delete room if no member. Assign new owner if owner exit.
- Exit room, send system message and update room info.

## Response

- sendMsgResponse, removeRoomResponse, updateRoomUsersResponse



Enze Zhong

Age: 28

Area: North America

School: Rice University

My Rooms

new

Happy Family

Enter

Available Rooms

12/1/2019 8:23:15 PM Enze Zhong created room

Demo

<http://chatapp-team-dragonball.herokuapp.com>

Room Info

Room Name:

Happy Family

Age: 10 to 80

Areas:

North America

Schools:

Rice University

Users

Enze Zhong owner

Send to all



Thank you!