NETWORK PROGRAMMING FINAL PROJECT REPORT
Project Name: Chess multiplayer

Lecturer: Prof. Trương Thị Diệu Linh

Group: 1

Members:

Student Name	Student ID
Nguyễn Trần Minh Tuấn	20194877
Phạm Đức Minh	20194803
Phan Nguyên Anh	20194727

Table of Contents

1) Application introduction

Python Client and Java Server

Used with MySQL database to store user info

Chess Multiplayer

Screens:

- Login/ Register
    - User enter (username, password) to login the game
    - User enter (username, password, ingame) to register
- Home
    - Navigation bar to Profile, Friend and Play
    - Logout
- Profile
    - Display match history
    - Show profile
- Friend
    - Add friend
    - Display online friends in Friend list
- Play
    - Create Custom Room (1v1)
    - Join Custom Room
- Custom Room
    - Start button
    - Chat
- Chess Game
    - Normal chess game and rules
    - Quit

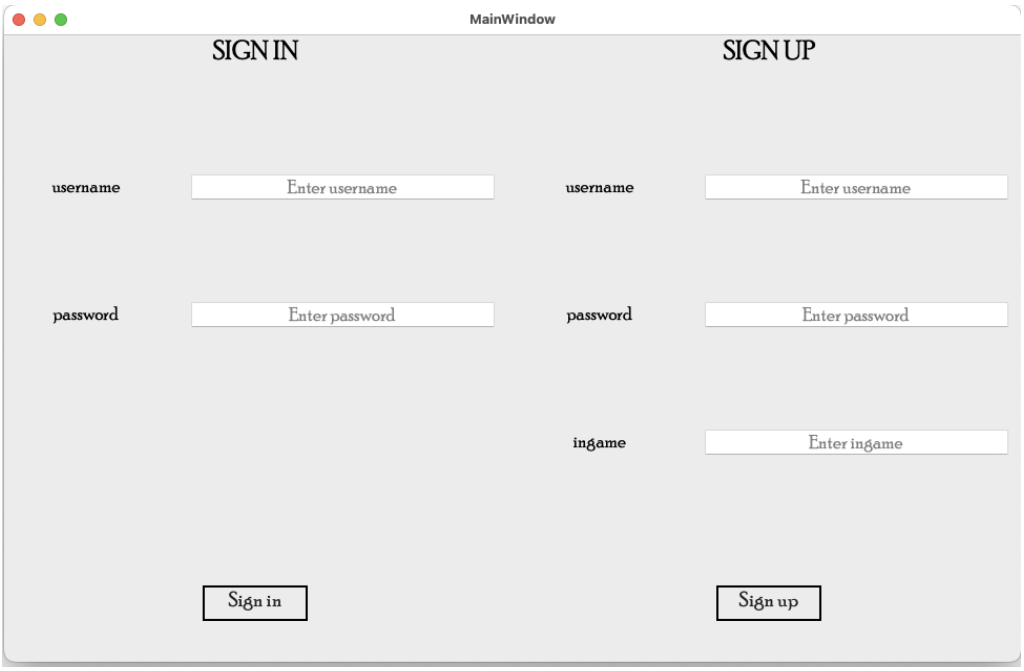

Figure 1. Sign in/ Sign up window

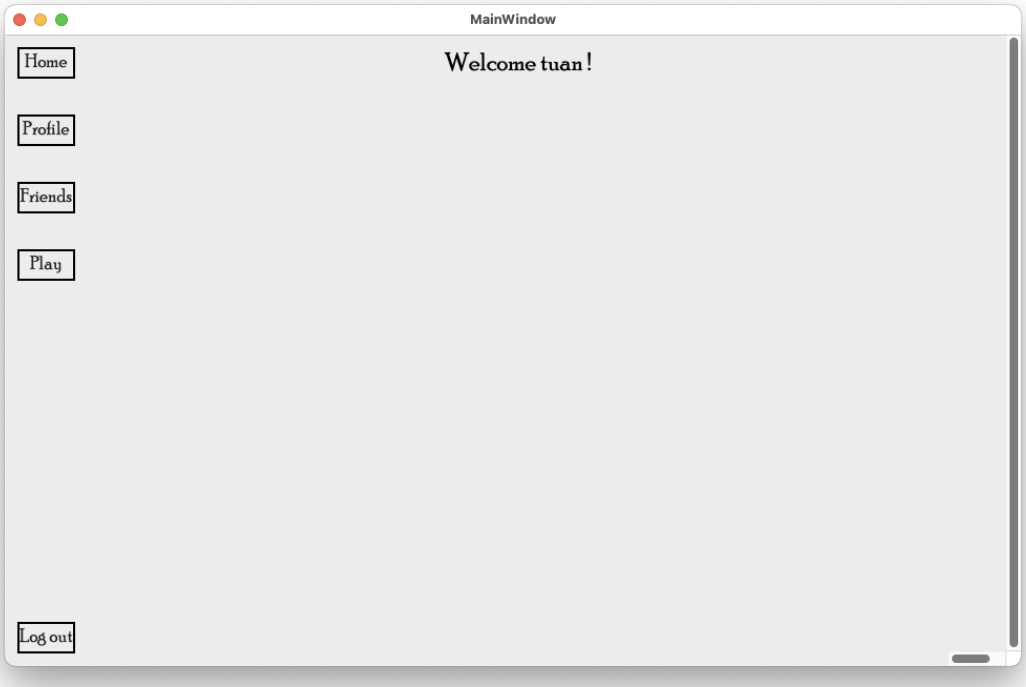

Figure 2. Home Page

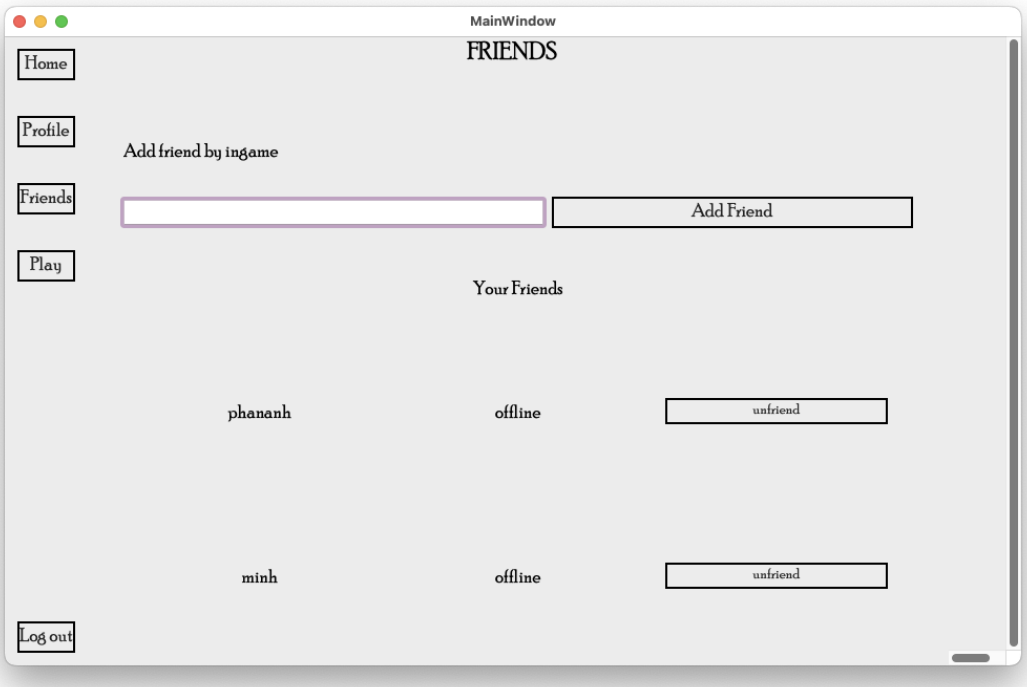

Figure 3. Friends Page

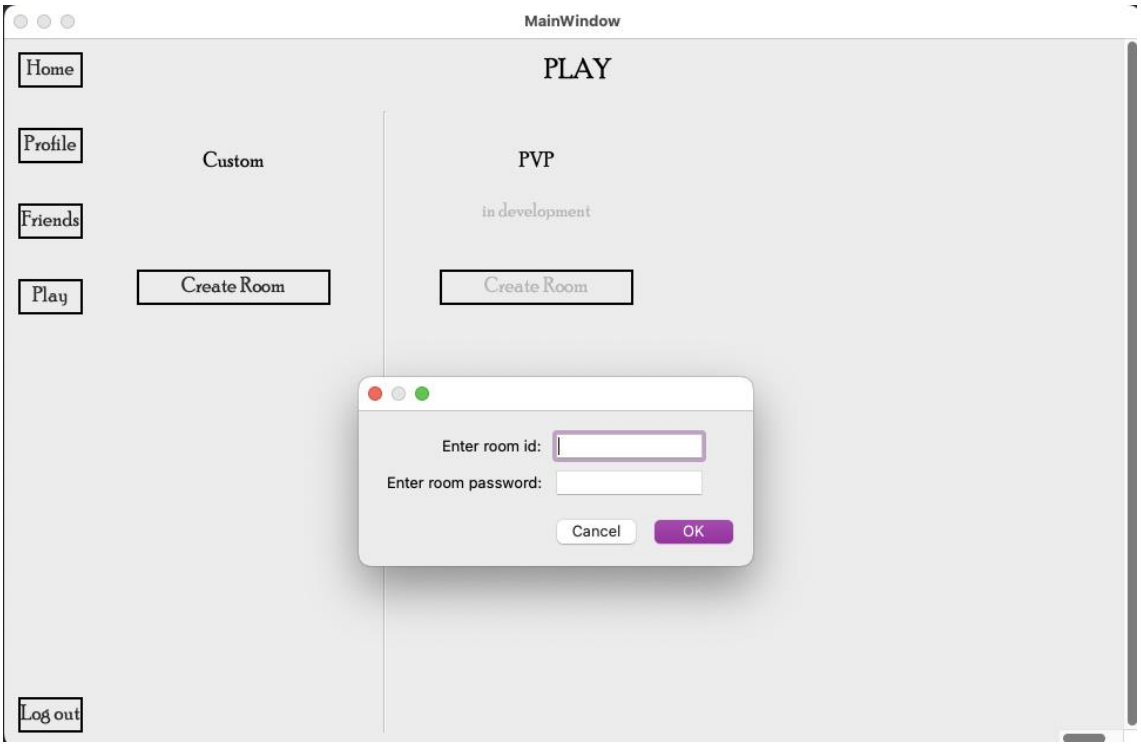

Figure 4. Create a custom room

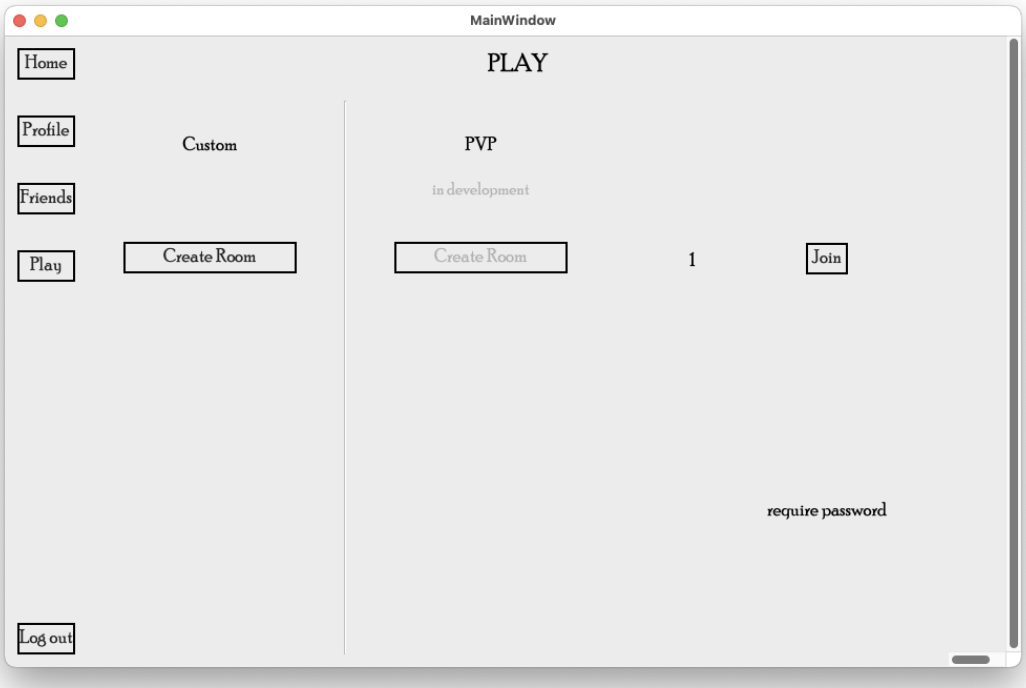

Figure 5. Current room list is always updated

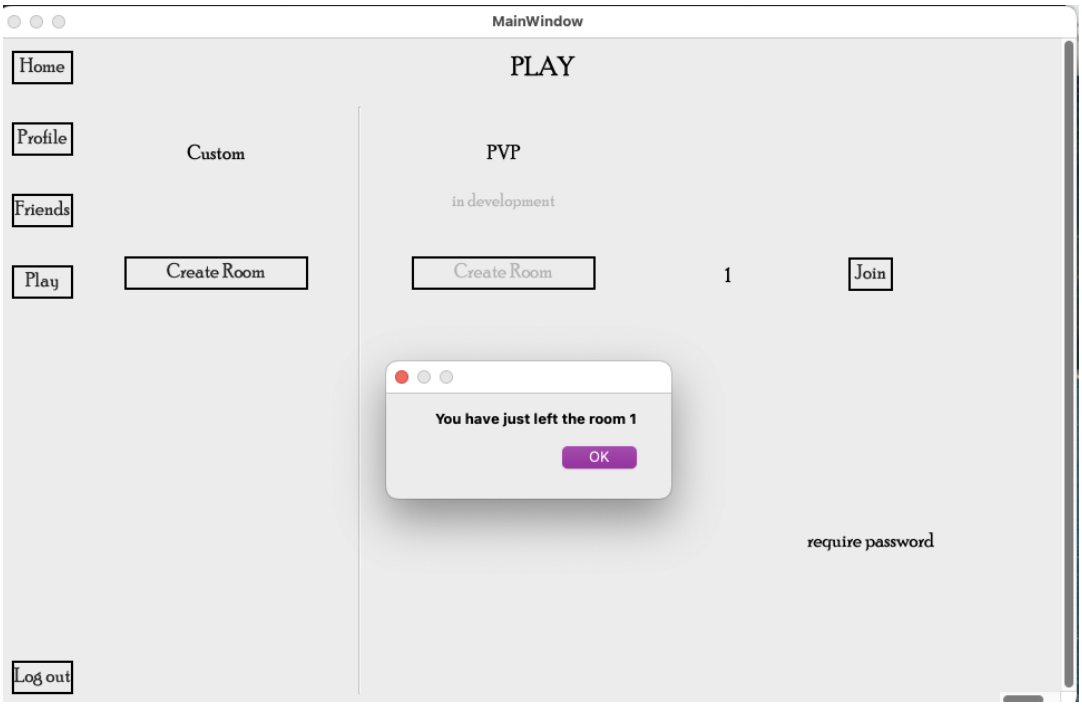

Figure 6. Notification when user leave the room

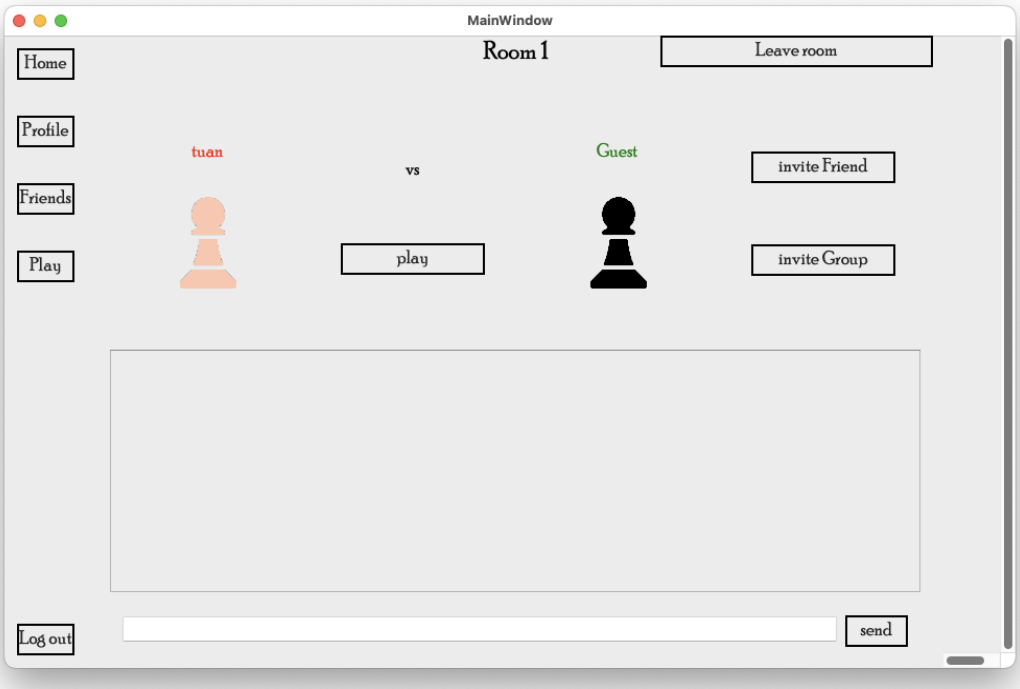

Figure 7. Custom room page

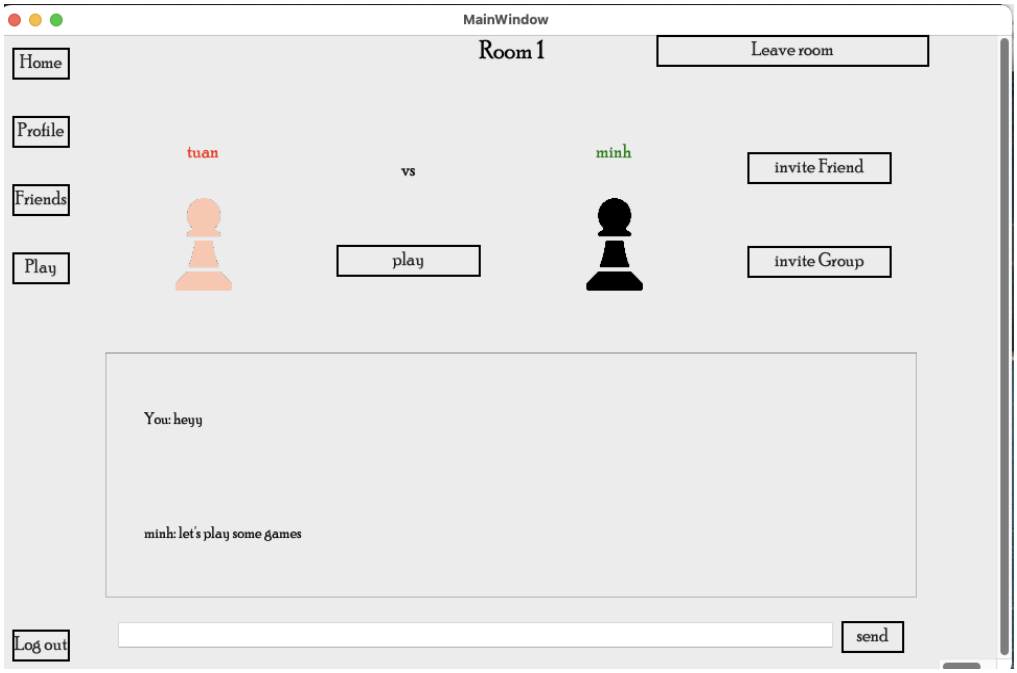

Figure 8. Custom room chat

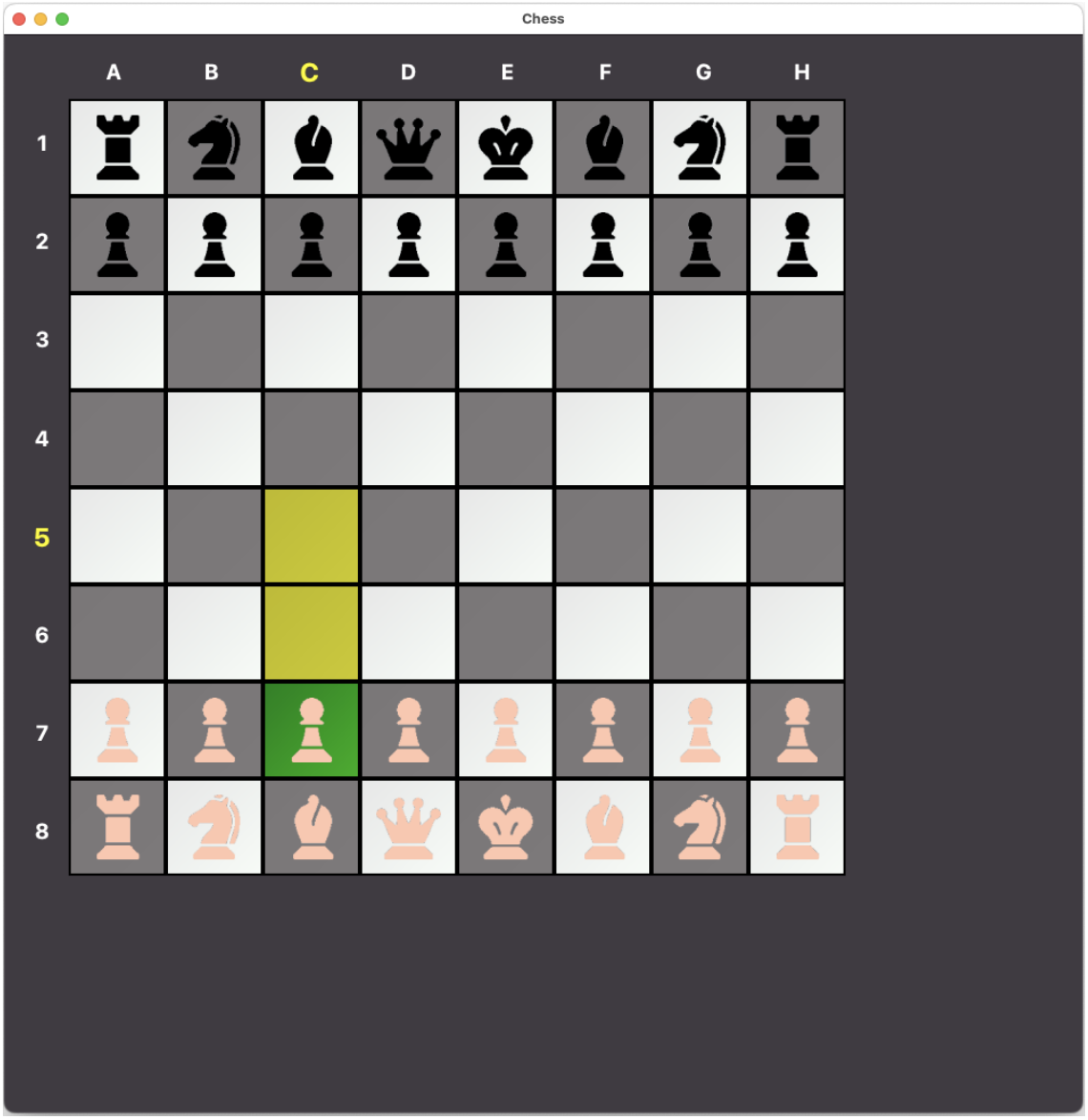

Figure 9. Move a chess piece

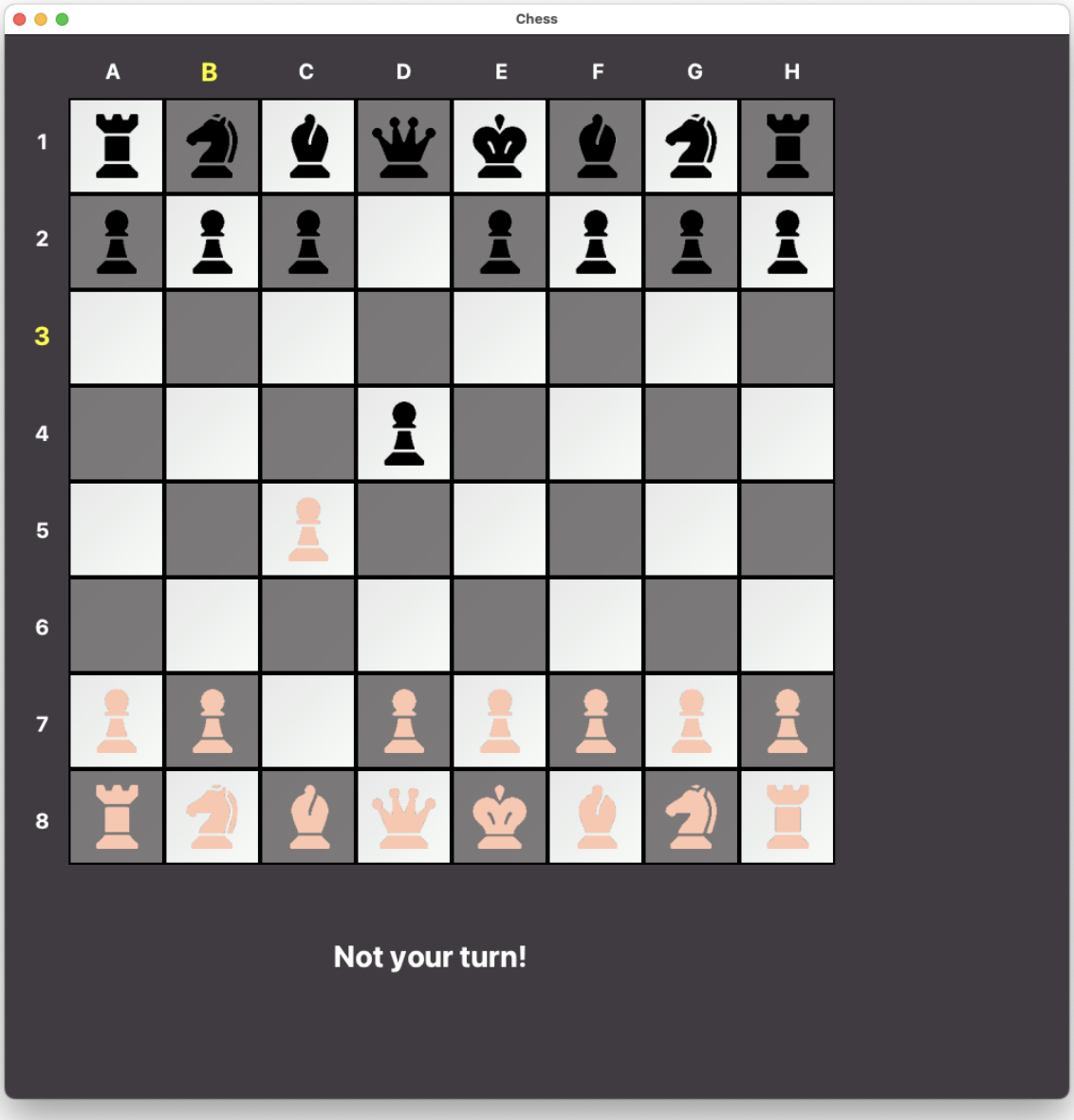

Figure 10. A player try to move the chess piece of the other

Figure 11. Game end notification

2) Architecture of the application: Client-Server

- Client Server
- Design goals
    - The communication in game should be fast, but the client & server communication should be reliable and sequential.
    - Account authentication should be secure.
    - The transferred data doesn't need to be confidential.
- Everyone can communicate with each other through the chat room, people can add friends and chat with friends and groups.
- There should be limitations to bandwidth or connection availability since the server runs on only 1 computer.
- We only need moderate error handling.
- We should maintain the communication channels.
- When Server receive a connection, it will create as thread to handle that connection
- When a user login, they will be added to active client list, when a room/ match is created, they will also be added to active room list or active matches
- The Client has a main thread to handle the GUI, other thread can be create when client need (Polling thread, calling to server …)
- When a message is received from client, it will be added to the received response array, client will check if the message just send have any response in array -> can be async when need

3) Functionalities:

- Login / Signup / Logout
- View profile, matches
- Manage friends, add friend, see online/ offline friends
- Create and join game rooms
    - Chat in room
    - View the game history
- Play chess

4) Application protocol design

- Message design
    - Message format: text-oriented protocol
    - Types of messages

Code	Description
200	OK
201	Not accept
404	Not found
405	Missing data
406	Incorrect format
407	Ingame existed
408	Username existed
202	Correct move
409	Wrong move
500	Unsuccessful

    - Message structures
        - REPL <Message_Type *code body* (Receive message)
        - <Message_Type> arguments (send message)

- Communication rules: Sequences of message
    - A message is a sequence of one or more lines
    - The start of the first line of the message is typically a word that represents the message type.
    - The rest of the first line and successive lines contain the data.

- Polling
    - When we open friend console, the server will continuously send message named FRND
    - The Server will then receive the friends list with online/ offline status and update to the screen

- Async

    - When user join room, he/she will receive instantly a reply message if joined room success.
    - The room owner will then receive the message from server without polling (checking if anyone joined room)

- Sync
    - When user login, server will check conditions and then create reply message

5) Flow

**********REGISTER**********
Desc: Register a user with username, password and ingame (name in the game)
REGT {"username": "tuan", "password": "123", "ingame": "tuan"}
REGT {"username": "phananh", "password": "123", "ingame": "phananh"}
REPL REGT 200 OK

Error:
REPL REGT 406 {"message":"User name exist"}

**********LOGIN**********
Desc: User login with credentials

LOGN {"username": "tuan", "password": "123"}
LOGN {"username": "phananh", "password": "123"}
REPL LOGN 200 {"id":1,"username":"tuan","password":"123","inGame":"tuan"}

Error:
Wrong password or username: REPL LOGN 406 {"message":"Bad Credential"}

**********LOGOUT**********
Desc: User logout with his ingame

LOUT {"ingame": "tuan"}
LOUT {"ingame": "phananh"}

REPL LOUT 200 OK

Error:
Wrong ingame: REPL LOUT 400 {"message":"Wrong user session"}

**********FRIEND**********
FRND {"ingame": "tuan"}
REPL FRND 200 [{"isOnline":true,"ingame":"phananh"}]
REPL FRND 200 [{"isOnline":false,"ingame":"phananh"}]

**********ADD FRIEND**********
ADFR {"ingame": "tuan"} // ingame cua nguoi muon add

phananh: REPL ADFR 200 OK
tuan: FRRQ {"ingame":"phananh"}

Error:
REPL FRND 400 {"message":"Must login first"}
REPL ADFR 422 {"message":"User is not online"}

**********ACCEPT FRIEND**********
ACFR {"ingame": "phananh"} // ingame cua nguoi gui loi moi ket ban

REPL ACFR 200 OK

**********CREATE ROOM**********
CRRM {"id": "123", "password": "123", "type": "CUSTOM"}
CRRM {"id": "123", "type": "CUSTOM"}
REPL CRRM 200
{"id":"123","white":"tuan","isWhiteOwner":true,"password":"123","type":"CUSTOM"}
CRRM {"id": "123", "password": "123", "type": "PvP"}

**********GET ALL ROOM**********
Desc: Get all the room in the server

ROOM
REPL ROOM 200 [{"roomid":"123","isPassword":true,"total":1,"maximum":2}]
(return roomid, isPassword or not, total users in the room and maximum of the room)

**********JOIN ROOM**********
Desc: User join a room
Get a reply if he/she joined a room
Reply contains info of the room
Room is fix with white player on the left, black player on the right
Room can have type, with password

phananh: JNRM {"roomid": "123", "password": "123"}
phananh: REPL JNRM 200
{"id":"123","white":"tuan","black":"phananh","isWhiteOwner":true,"password":"123","type":"C
USTOM"}
tuan: JOIN {"ingame":"phananh"} // tuan la chu phong

Error:
REPL JNRM 400 {"message":"Room not exist"} // join vao phong khong ton tai
REPL JNRM 400 {"message":"Room is full"} // join vao phong day nguoi

**********LEAVE ROOM**********
Desc Leave a room
Opponent will receive a message to notice that other is left
If a player leave, the other one is the owner, if the room is empty then it will be destroyed
If owner leave the room, other will be owner

phananh: LVRM {"roomid": "123"}
phananh: REPL LVRM 200 OK
tuan: OLVR {"isWhiteOwner":true}

tuan: LVRM {"roomid": "123"}
tuan: REPL LVRM 200 OK
phananh: OLVR {"isWhiteOwner":false}

tuan: JNRM {"roomid": "123", "password": "123"} // tuan join lai
tuan: REPL JNRM 200
{"id":"123","white":"tuan","black":"phananh","isWhiteOwner":false,"password":"123","type":"CUSTOM"}
phananh: JOIN {"ingame":"tuan"}

**********PLAY**********
Desc: owner of the room play the game, receive the details of the match
Opponent will receive the start match with that details

tuan: PLAY {"roomid": "123"}
tuan: REPL PLAY 200
{"match":{"id":5,"white":"tuan","black":"phananh","state":"","type":"CUSTOM"},"isWhiteTurn":true}
phananh: STRT
{"match":{"id":5,"white":"tuan","black":"phananh","state":"","type":"CUSTOM"},"isWhiteTurn":true}

**********MOVE**********
Desc: Player move a step, receive the move he/she has just made and the current state of the game
If receive the message with move end with ++ means that checked and no way to unchecked -> game end

tuan: MOVE {"matchid": "5", "move": "P7A5A"}
tuan: REPL MOVE 200 {"move":"P7A5A","game_state":"P7A5A "}
phananh: OPMV {"move":"P7A5A","game_state":"P7A5A "}

phananh: MOVE {"matchid": "5", "move": "P7A5A"}
phananh: REPL MOVE 200 {"move":"P7A5A","game_state":"P7A5A P7A5A "}
tuan: OPMV {"move":"P7A5A","game_state":"P7A5A P7A5A "}

// ++ chieu tuong het co
tuan: MOVE {"matchid": "5", "move": "PA++"}
tuan: REPL MOVE 200 {"move":"PA++","game_state":"P7A5A P7A5A PA++ "}

phananh: OPMV {"move":"PA++","game_state":"P7A5A P7A5A PA++ "}

Error:
REPL MOVE 400 {"message":"Not your turn"} // neu di 2 luot lien tiep

**********CHAT ROOM**********

tuan join phong
na join phong
tuan: CHAT {"roomid": "123", "message": "ban danh hay qua"}
na: OCHT {"message": "ban danh hay qua", “opponent_ingame”: “tuan”}

**********INFO**********
Desc: Get full detail of a user include all matches that he/she had played in
tuan login
tuan: INFO
tuan: REPL INFO 200
{"user":{"id":2,"username":"ming2","password":"123456","inGame":"ming2"},"matches":[{"id":5,"white":"ming","black":"ming2","state":"P7A5A P7A5A++ ","winner":"ming","type":"CUSTOM"},{"id":6,"white":"ming","black":"ming2","state":"P7A6A D7A5A++ ","winner":"ming","type":"CUSTOM"}]}