

The University of Birmingham



Software Workshop Project

20 March 2017

Members

Fozia Mehboob
Jacob Smith
Jonathan Keslake
Vishnu Raman
Ian Kirkman

Statement of Contributions

All members of the group contributed equally to the project.

Fozia Mehboob 1732130

FXM630@cs.bham.ac.uk

Jonathan Keslake 1322858

JPK358@cs.bham.ac.uk

Jacob Smith 1712130

JGS630@cs.bham.ac.uk

Vishnu Raman

@cs.bham.ac.uk

Ian Kirkman 1705598

IDK698@cs.bham.ac.uk

Table of Contents

Introduction	4
System design	6
Functional requirements:.....	6
Non-functional requirements:	8
Architecture	10
Whole systems	11
Class interaction	18
Database	18
Client	18
Mafia.....	19
Server	19
GUI design.....	20
Design decisions	20
Database	24
Server	26
Client	27
Mafia (game)	28
Test plan.....	31
Testing mafia class	31
Testing GUIs	32
GUI extreme conditions testing	32
Testing client and server interaction	33
Team organisation report.....	33
Evaluation	34
Appendix	35
Project diary (meeting minutes)	57
Bibliographic references.....	63

Introduction

The Nice team has evaluated a variety of project ideas and has concluded to design a program that combines messaging whilst playing a game. We plan to create a Mafia Chat game. The aim of the messaging app will essentially be to allow users to message one another as well as participate in the game.

A brief description of the game

In a game of Mafia, a third of the players (rounded to nearest whole number) will be randomly assigned to the role of 'mafia' with the remaining players be given the role of 'innocent'. There will be two stages; day and night.

During the day stage there will be a discussion between all players, the innocent players have a discussion that will result in a player accusing another of being a "Mafioso" and prompt others to vote to eliminate them. If more than half the players vote to do so, then the accused player will be eliminated from the game. If, during the day stage, there are insufficient votes for an accused player to be eliminated then the game will move to the night stage with no elimination.

During the 'night' stage the mafia players will have a vote, hidden from the innocent players, on who they want to eliminate. The mafia players will only be able to vote for innocents to be killed and, if they can't decide on which innocent by the end of the timer, the server will choose the innocent with the most votes or, in case of a tie, choose randomly between the innocents with the most votes. If no innocents have been voted out by the end of the timer, then the game will move back to the day stage without any innocents being killed.

The game will be won, by the Mafia team, once they have eliminated enough innocents so that the number of Mafia members is greater than the remaining number of innocents.

Users' registration to the game

The game will be accessible to users of all ages and abilities and sign up will be fast. A set of rules will be provided and will be viewable for the user (in the form of a pop up resulting from clicking on the 'Rules' JButton) to quickly enable them to get up to speed with how to play Mafia. Since the

users' information will be stored in a database, they will be free to leave and/or join games at their own convenience.

The game will allow multiple users to sign up with a unique username and password to play together. There will be a function by which if any user forgets their login password, they will be able to retrieve it by creating a security question and relevant answer at their time of signing up. The app will prompt the user to input the answer to the security question once they have identified that they have forgotten their log in details.

Chatroom messaging within the game

The game will make use of multi-threading since multiple users will all be playing at the same time. Messages will be constantly sent and received by the client and server in order to facilitate the demands of each player whether it's voting to bring the game to the next stage or just casual discussion amongst the players.

Sockets will be used since they will be the primary connection between the client and the server.

Packets of data will be sent and received in order for the two to communicate. The address and port of each player will determine where the information is being sent and received.

There will be a database in which the users' information will be stored. Utilising JDBC, information that is entered from the client, will be communicated to the server, which will then send the command through a Manager class that will store the users' information in the database. The same protocol will be used to enable security information to be called from the database as well as by the user.

Our project is the game Mafia being played using an online chat service. Multiple users can sign up with a unique username and password to play together. If any user forgets their login password, they can retrieve it by creating a security question with an answer at their time of sign up. The game is playable by anybody, new or experienced, and sign up is fast. Since the user's information will be stored in a database, they can leave or join games at their own convenience.

The game makes use of multi-threading since users will all be playing at the same time. Messages will be constantly sent and received by the client and server in order to facilitate the demands of each player whether it's voting to bring the game to the next stage or just casual discussion amongst the players.

The report outlines the system design which includes whole system, database, server, and client. UML diagrams have been utilised to convey further information regarding the project. Tests, team organization, meeting minutes and evaluation provided in the report contributes towards the thorough completeness of the project.

System design

Functional requirements:

1. Log In

1.1. The System must allow users to Log In.

1.1.1. The system shall authenticate Login credentials, the username and password, using the Database.

1.1.2. The system will only sign in users with both username and password matching.

1.1.3. The system will not allow unauthenticated users to sign in.

1.1.4. Upon successful LogIn, the system shall manage the user's session.

1.1.4.1. The system shall redirect these users to the 'Game' screen.

2. Registering

2.1. The system must allow registry of new users using specific data.

2.1.1. The system must allow entry of a unique 'username'.

2.1.2. The system must allow entry for a password.

2.1.2.1. The system must hide the password using a single character.

2.1.3. The system must allow entry for a security question.

2.1.4. The system must allow entry for a security question answer.

2.2. The system must allow the submission of all registered data into the connected Database.

3. Messaging

- 3.1. The system must allow entry of text from users.
- 3.2. The system must allow users to enter multiple lines of text.
- 3.3. The system must allow users to 'send' their messages.
- 3.4. The system must display all messages from all users.
 - 3.4.1. The system must display the username of the sender of each message.
- 3.5. The system must allow the server to send messages directly to each user.
- 3.6. The system must show all online users.

4. Rules

- 4.1 The system includes a game.
 - 4.1.1 The game element should have a maximum and minimum user limit.
 - 4.1.1.1 The game should start with a minimum of 6 users.
 - 4.1.1.2 The game should have a maximum user limit of 16.
 - 4.2 The game element must assign the user game roles
 - 4.2.1 The system must select a $\frac{1}{3}$ of users as the mafia.
 - 4.3 The game element must be able to send out private and public messages.
 - 4.3.1 The game element must inform each user their role within the game in a private message.
 - 4.4 The game element should allow the game to change phases.
 - 4.4.1. The game element should have two set phases; day and night.
 - 4.5 The system must allow for voting to take place as a part of the game.
 - 4.5.1. The game element must allow any user to initiate the voting process.
 - 4.5.2 The game element should mute players who have received the majority vote.
 - 4.5.2.1 The game element should allow users to vote for who they believe to be the Mafia (set by the server) during the day
 - 4.5.2.2 The game element should allow users who are the mafia to vote to kill off users who are innocents during the night.
 - 4.5.3 The game element must ensure votes can only be started by active users.
 - 4.5.4 The same element stem should allow users who do not receive majority vote to continue in the game.
 - 4.5.5 The game element must not allow users to make duplicate votes in a voting session.
 - 4.5.6 The game element should allow users to vote for any active user.
 - 4.5.6.1 The game element should allow users to vote for different users per vote session.

- 4.5.7 The game element must complete one vote session before another is initiated.
- 4.6 The game element should have no restriction on the number of elimination votes that can occur during the day (for all players or any player in particular).
- 4.7 The game element must not allow eliminated players to continue to participate in the game (either chat or take part in the voting).
 - 4.7.1 The game element should allow muted users to still observe the game.
 - 4.7.2 The game element must not let muted users post messages.
- 4.8 The game element must allow players to add their votes to either change the game to night or to keep it as day.
- 4.9 The game element must be able to change the game phase between day and night.
 - 4.9.1 The game element must only allow a phases change to night based on majority votes from users.
- 4.10 The game element should mute all conversations during the night phases.
 - 4.10.1 The game element should only allow game commands to be run during the night phases.
- 4.11 The game element should inform each user when a vote has been made.
 - 4.11.1 The game element must not mention what each user has voted
- 4.12 The system must only change phases from day to night either after all mafia users have voted or a set time has passed
- 4.13 The game element should start a timer when any of the votes above are started.
 - 4.13.1 The system must ignore all votes cast after the timer is complete.
- 4.14 The game element must only eliminate a user if all of the votes by the mafia during the night agree.
- 4.15 The game element must allow the game to continue until the victory conditions are met.
 - 4.15.1 For the innocents this is when they have eliminated all of the mafia and for the mafia this is when they are equal in number to them.
- 4.16 The game element should reveal who the mafia users were at the end of the game. This should be the only time the system reveals the role of a user to all other users.

Non-functional requirements:

1. Availability

- 1.1. The system must be operational for 95% of the time

2. Response time

- 2.1. The system should provide response to user requests within a time that is deemed to be no greater than 3 seconds.
- 2.2. The system must complete all tasks requested by the user in a timely and efficient manner
 - 2.2.1. All system tasks must be completed within 2 seconds after the task has been initiated by the user.

3. Interface and usability

- 3.1. The system must have user friendly interface design.
 - 3.1.1. The system must have all functionality tools clearly visible
 - 3.1.2. The system must be easy to navigate for the user
 - 3.1.3. The system should maintain a common appearance or theme throughout.

4. Privacy

- 4.1. The system must store user data.
- 4.2. The system must not pass on any of the users to any third parties.

5. Security

- 5.1 The system must store user personal data correctly and in an ethical manner.

6. Delivery

- 6.1 The system must be available for use Tuesday 22nd March 2017.

7. Implementation Constraints (programming style)

- 7.1 The systems programming code must adhere to Java code conventions which are stated in java Oracle documentation
- 7.2 The systems code must make use of Javadoc commenting in the code.
- 7.3 The systems code must make use of indentation
- 7.4 The systems code must make use of commenting within each class.
- 7.5 The systems code may make use of imported Java libraries
- 7.6 The systems code must make use of object oriented programming techniques

Architecture

Database

- Should store usernames, these should be unique per users.
- Should store passwords.
- Should store security details (Secure question and answer)

Server

- The server must be active before any clients become active.
- One server must run on one host only; no further hosts should run on that host.
- The server must receive all incoming chats and relay them to the clients.
- The server must relay user credentials being passed through via database this response is sent back to the client
- The server must relay all commands to the clients.
- The server must also provide functions for the game to mute/unmute players, activate/deactivate chat and send public / private messages.

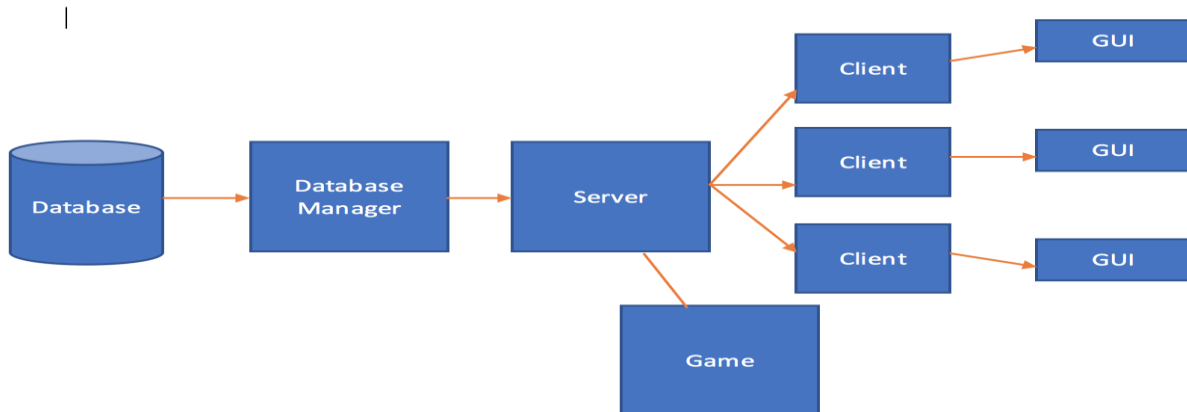
Client

- Connecting to GUI.
- Passing message and commands to server.
- Receive / relay server messages.

Whole systems

Figure 1a

Communication system diagram



This diagram shows a basic overview of the communication between entities within our system.

The idea behind the architecture is to allow multiple users to register, login and communicate with one another, as well as having the option to play a game. Having a clear picture of the architecture of the system helped team member understand the logic behind the system.

Figure 1b
Use case diagram



Text

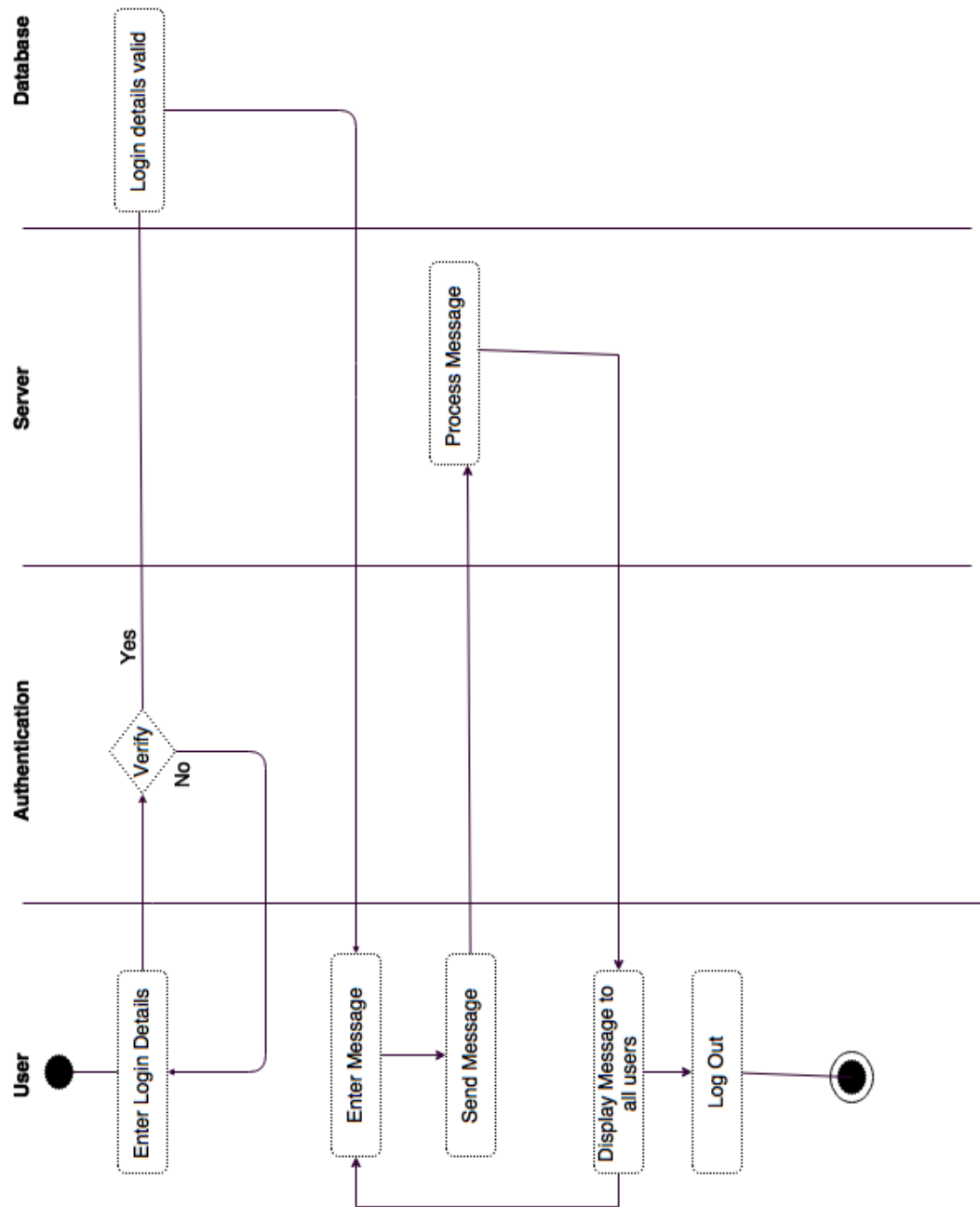
The use case diagram shows how the user interacts with the system. The user can log in if they are a registered user, if not then the user has the option to register a new account. To register there are some mandatory fields they must complete to register. Once this is done, the user has to return to the login screen and complete the login process. The login details are verified using the server who relays this to the database.

Once the user is logged in they enter a chat room. Inside the chatroom they have a number of possibilities, they can stay in the chat room send and receive messages, view the rules, or log out or they could start a game. When in game mode, there can vote for players and mute them based on majority vote.

The server can send private messages to the system and all the game logic is within the game actor.

Figure 1c

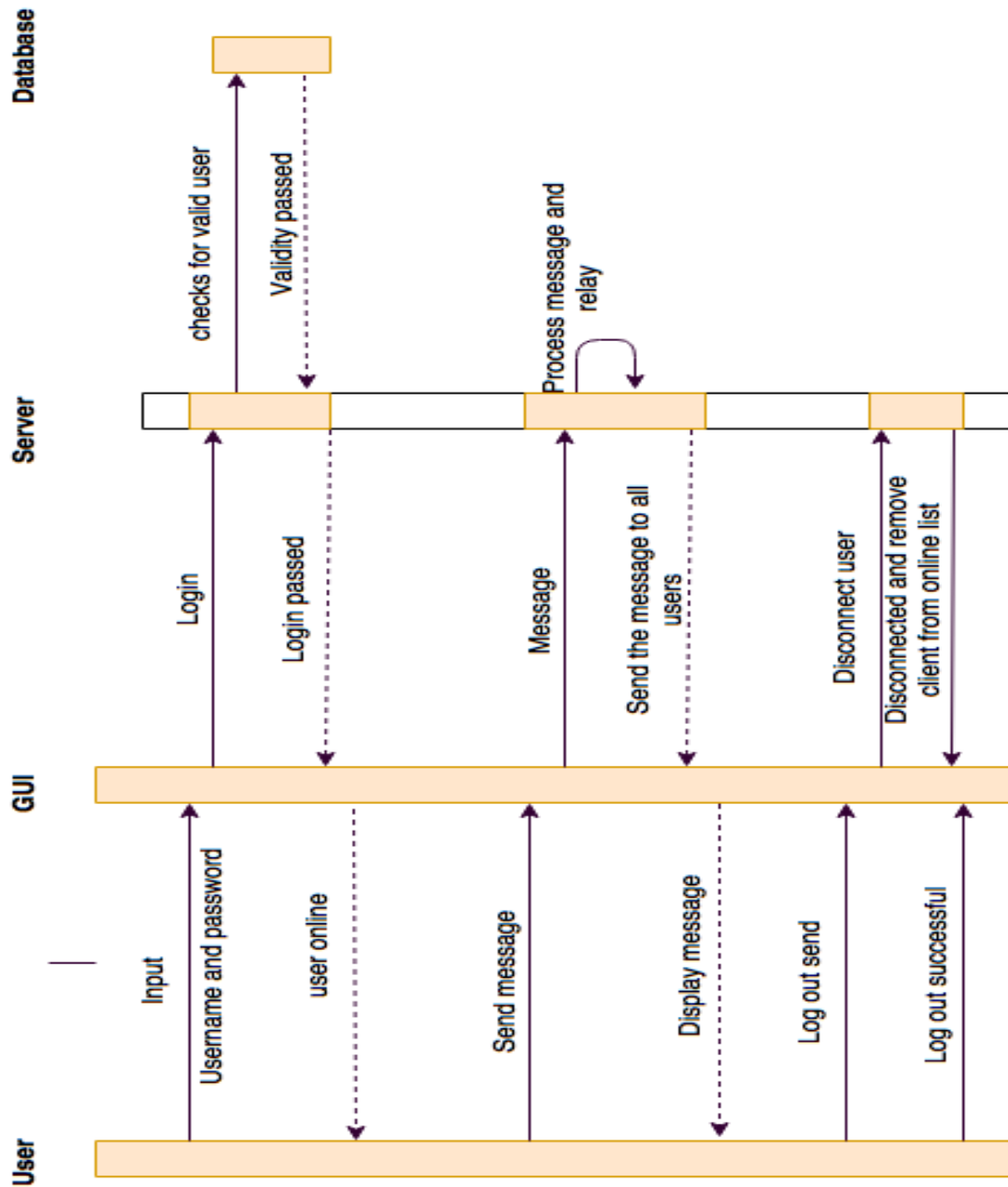
Activity diagram for sending message with login



The activity diagram shows the process of a user logging in and sending a message.

Figure 1d

Sequence diagram for logging in and sending messages



The sequence diagram shows the process of a user logging in and sending a message, showing how long each element is active.

Figure 1e

Class diagram – server design

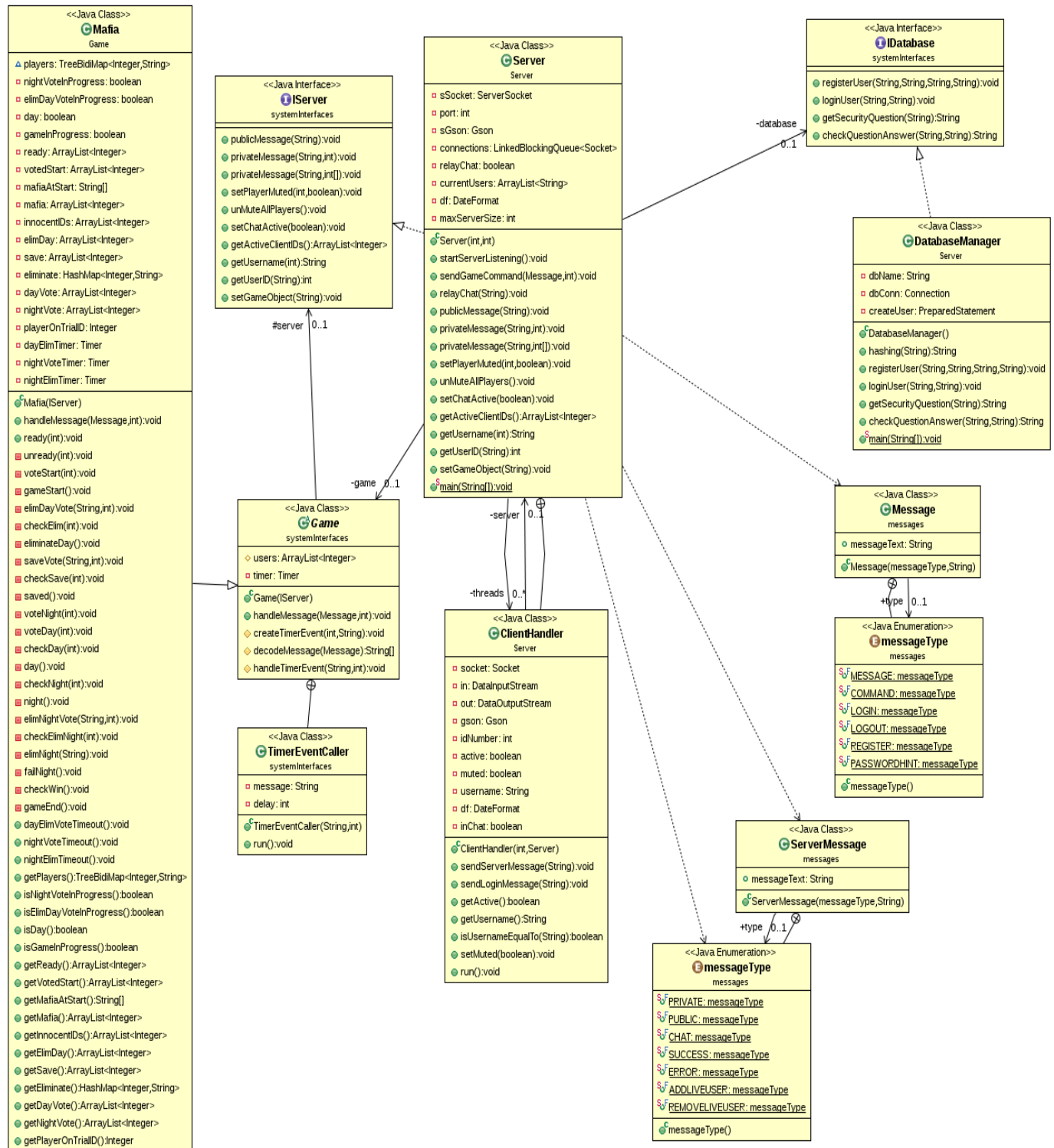
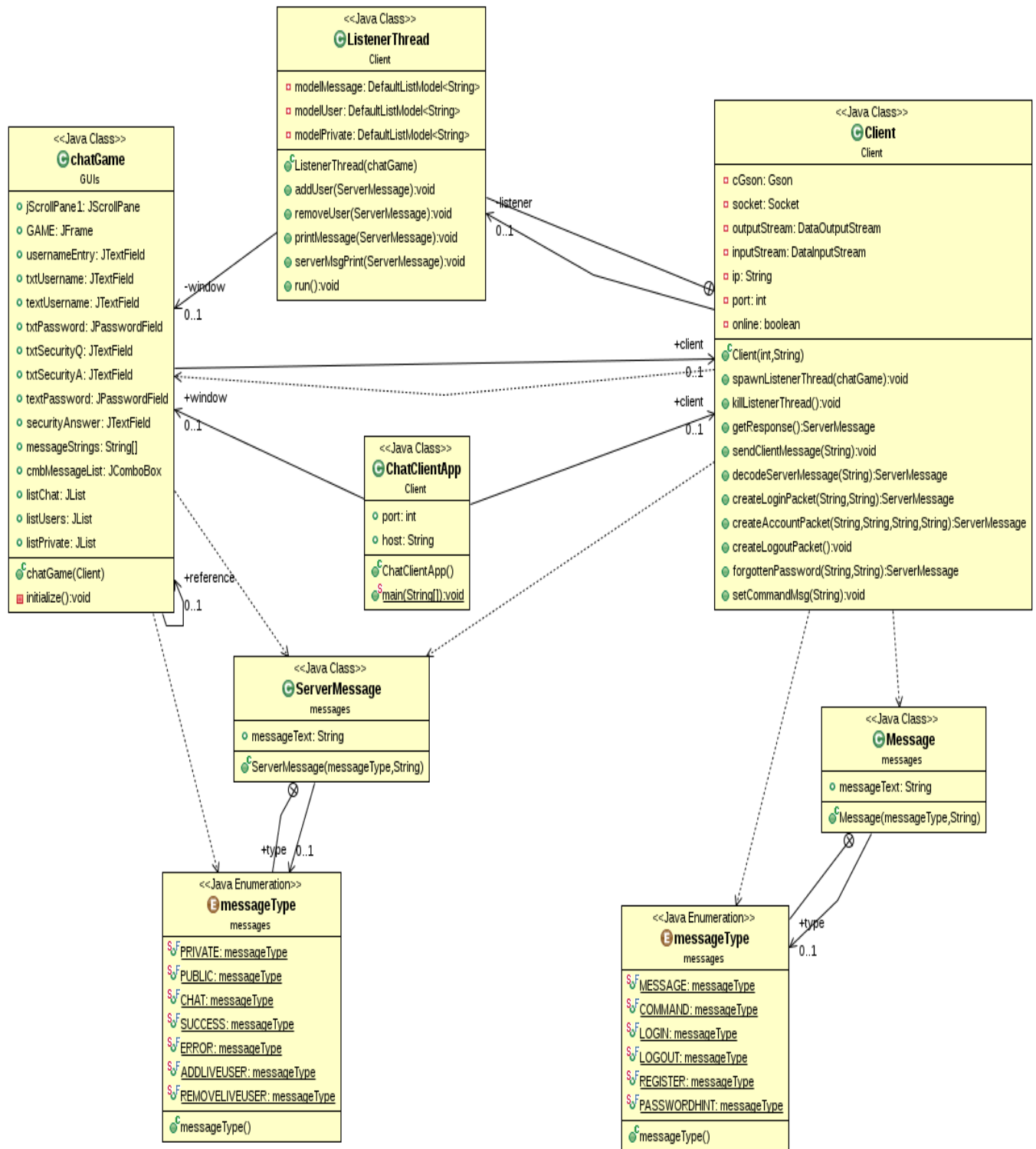


Figure 1f

Client design diagram



Class interaction

Database

The DatabaseManager class deals with interactions from the GUI through the client-server by implementing the IDatabase interface. It acts store usernames, passwords, and security questions with answers that the user enters in at the time of registration. It can also check for duplicate or existing usernames through the methods called in this class.

The user is brought to a new window when they interact with the GUI so that they can try answer their security question when they've forgotten their password. When the user guesses right then they are awarded with a popup that display the password. The methods used in this class also throw exception for invalid user and invalid information.

Client

The client class deals with dealing interactions with the server. It has a subclass called ListenerThread which constantly reads input from the server. The client class just relays the chat and send server messages. The ChatClientApp class creates an instance of the Client and the GUI (chatGame) and initialises the application. When the user opens the application, the user can log in or sign up to the game. Once signed up and logged in, the chat room window pops up and a welcome message is received by the user. The user can then chat with everyone who are online or send commands to the server to handle. The Client receives server messages defined in the ServerMessage class and sends out messages or commands defined in the Message class

Figure 2

Client-server diagram



ChatClientApp

This class executes the chat application. An instance of the GUI and the Client are created. there are two other variables, an integer port and a string host, which are the port and the IP address of the host respectively. The class only has a main method which initialises the Client and the chatGame (GUI) with the port and IP address of the host (Server).

chatGame

There are several GUIs screens used within this one class. Each screen will be made up of a JPanel. One for the login screen, the registration screen, the forgotten password screen and the chatroom game itself. The GUI uses a swing model, consisting of the frame. Anything displayed to the user is handled in by the action listeners and anything done because of user input is handled in by the client.

Mafia

This class contains all the methods to handle the commands related to the Mafia game. When a user sends a command message (identified by starting with /), the server then invokes the handleMessage method in the Mafia class which carries out the rest. The valid commands for the game depend on the state of the game (whether the game has started and whether the game is in day or night phase). When a vote is started in the game, a timer is set so that if the vote hasn't finished by the end of the set time (20 seconds) the missing votes are ignored. For the vote to eliminate a player and the vote to change the game tonight, if the vote is timed out, the vote is successful if the votes for the motion are in the majority of the votes that were cast. For the mafia vote at night, if the vote is timed out then it fails.

Server

The Server manages an array of ClientHandler threads which listen for Message instances from the clients and acts upon them. The server also provides a suite of functions allowing the game class to send private and public messages to the clients, as well as muting/unmuting certain players and turning the chat on or off.

ClientHandler

A ClientHandler instance waits for commands from one client and distributes them depending on the message type, relaying messages to other clients, sending commands to the game class, sending login, register and passwordhint messages to the game class.

GUI design

For the prototyping stage, two set of prototypes were designed. Initially, these were designed using paper and pencil, in a very rough manner, designs were seen to be unclear not containing the correct amount of detail needed. For this reason, these prototypes were then re-created in a neater fashion to enable a clear evaluation of each screen.

Before creating the prototypes, careful consideration was paid to the user requirements document (above) to incorporate the design features that the users had specially identified within their sequence diagram and use case diagram, which helped showcase user interaction with the system.

Design decisions

Initial design ideas are to keep the look and feel of the GUI very clean and simple. Each player should be able to select a character image to represent themselves within the game.

The GUI will have a variety of different features some of which are. The GUI should be centred around chat conversations of the players. When the game elements start, the system must randomly assign a third of the players to be Mafia. This should be done using a private message from the server to the player. The private message from the server to each of the Mafia players telling them that they are a member of the Mafia and who their Mafia allies are. The system should have a way for the Mafia players to be reminded of who the other mafia players are. This could be done through the Client displaying the names of the Mafia players (this must only be for the clients of the Mafia players). The system should place the players that have been eliminated on mute, whereby they can see the messages and but not have any features to communicate with other clients. The game continues until all the mafia have been eliminated or until the number of Mafia equals the number of innocents. Each user must register an account and be logged in to play the game this must be shown in on the GUI.

The GUI is the interface that allows users to interact with the entire program. It is the front end of the system and often in business experts in HCI are assigned the tasks of investigating what would make the best system for the end user. GUI's should be structured in a manner that makes navigating around a system seem seamless and natural, it should not request too much thought from the user, as studies show user quickly become annoyed and give up. Bearing all this information, the GUI was initially designed on paper, before being mocked up on MockPlus. This software was chosen to its simplicity and ease of use.

Below is a brief description of our GUI design.

Figure 3a

Welcome Screen

Log

In

Screen

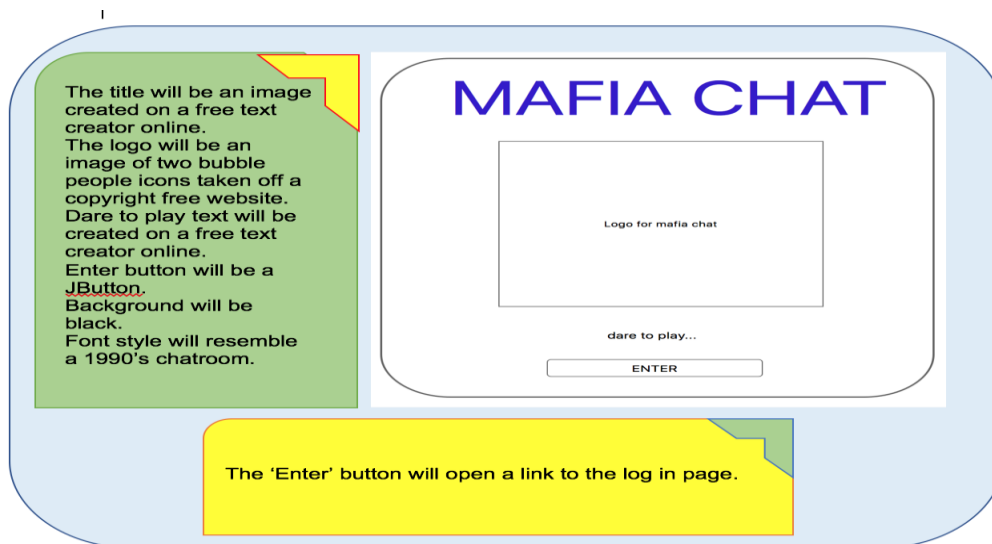



Figure 3b
Registering screen

The title will be an image created on a free text creator online.
 The logos will be images of two bubble people icons taken off a copyright free website.
 'Enter Username' and 'Enter Password' will be **JLabels**. These will each have a white bar in which users can enter the required fields.
 'Log In', 'Sign Up' and 'Forgotten Password' buttons will be **JButtons**.
 Background will be black.
 Font style will resemble a 1990's chatroom.



The mockup shows a window titled "MAFIA CHAT" with a black background. It contains two input fields labeled "Enter Username" and "Enter Password". Below these are three buttons: "Log In", "Sign Up", and "Forgotten Password". There are also two placeholder boxes for "Image of mafia logo" on either side of the "Forgotten Password" button.

The title will be an image created on a free text creator online.
 The logos will be images of two bubble people icons taken off a copyright free website.
 'Enter Username', 'Enter Password', 'Set security question' and 'Set your answer' will all be **JLabels**. These will each have a white bar in which users can enter the required fields.
 'Register' and 'Back' buttons will be **JButtons**.
 Background will be black.
 Font style will resemble a 1990's chatroom.



This mockup is similar to the first one but includes additional fields for security questions. It has input fields for "Enter Username", "Enter Password", "Set security question", and "Set your answer". Below these are two buttons: "Register" and "Back". There are also two placeholder boxes for "Image of mafia logo" on either side of the "Register" button.

The 'Registration' button will open up a pop up box informing the user that their registration was successful.
 The 'Back' button will open a link back to the title screen.

Figure 3c

Forgotten Password Screen

The title will be an image created on a free text creator online.

The logos will be images of two bubble people icons taken off a copyright free website.

'Enter Username' and 'Enter the set security answer' will be **JLabels**.

These will each have a white bar in which users can enter the required fields.

'Get Question', 'Submit Security Answer' and 'Return to login screen' buttons will be **JButtons**.

Background will be black.

Font style will resemble a 1990's chatroom.

The 'Get Question' button will open up a pop up box displaying the security question.

The 'Submit Security Answer' button will open a link to the chatroom screen.

The 'Return to Login Screen' button will open a link back to the log in screen.

Figure 3d

Game Screen

The title will be an image created on a free text creator online.

The global chat box will be a white box in which users' chat messages will appear along with their name and the time that the message was sent.

The names of online users will appear in the relevant box on the right hand side.

'Online Users' and 'Server Messages' will be **JLabels**.

Users will be able to type their messages in the 'type text here' box before sending.

Send, rules and log out buttons will be **JButtons**.

Background will be black.

Font style will resemble a 1990's chatroom.

The 'Rules' button will open up a pop up box informing the user of the rules of the chatroom.

The 'Log Out' button will open a link back to the log in screen.

The 'Send' button will submit the message and it will appear in the global chat box.

Database

The database has one schema, 'users,' which was created using PostgreSQL. All functions of the database are called from IDatabase interface. Players will be able to interact with the GUI to register:

- Username
- Password
- Security question
- Security answer

A unique serial ID 'userid' is associated with each registered user as a primary key. This is generated automatically by the database when a player registers. There is built in functionality to check if a username exists or not in the database so that duplicate usernames cannot exist. We hope to create additional schemas for storing chat history and high scores for the player. The primary key will act as a link between the tables so that they may communicate and transmit stored data.

Figure 3e
Entity table

Entity: users				
Attributes	Description	Data Type	Primary Key	Comments
userid	Unique identifier for each user	serial	Yes	UNIQUE NOT NULL
username	Name submitted by the user	varchar (255)	No	UNIQUE NOT NULL
password	Password submitted for user login	varchar (255)	No	NOT NULL
question	Security question to be provided from user	varchar (255)	No	NOT NULL
answer	Security answer to be provided by user	varchar (255)	No	NOT NULL

Figure 3f

Schema

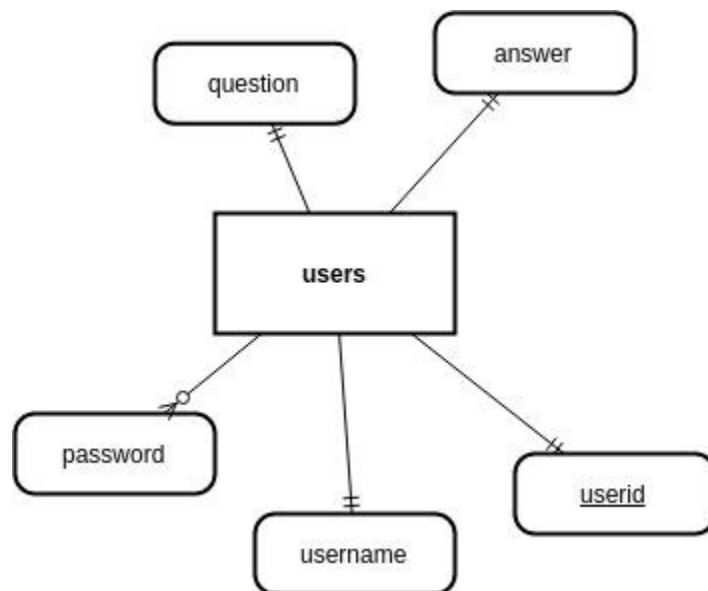
SQL code to create the table:

```
CREATE TABLE users (  
    userid      serial UNIQUE NOT NULL,  
    username    varchar(255) UNIQUE NOT NULL,  
    password    varchar(255) NOT NULL,  
    question    varchar(255) NOT NULL,  
    answer      varchar(255) NOT NULL,  
    PRIMARY KEY (userid)  
);
```

Implemented Queries from our DatabaseManager using prepared statements:

- "SELECT username FROM users WHERE username LIKE ?"
- "INSERT INTO users (username, password, question, answer) VALUES (?, ?, ?, ?)"
- "SELECT password FROM users WHERE username = ?"
- "SELECT question FROM users WHERE username = ?"

Figure 3g
Entity relationship diagram



There is one table to be diagrammed. The users table stores registered user information and has five attributes. The userid is stored as the primary key and automatically generated by the database. This ensures each user has a unique id that can be referenced in other tables. The username field is also unique as we don't want duplicate usernames to be registered. The password, question, and answer fields all must have data entered. These fields will accept duplicate information in the table since it is private information it would be a security flaw if any user was notified that their sensitive information replicated another user's.

Server

The Server receives messages from the Client, and distributes them to the classes which deal with the message types. The Server sends messages to be handled by either a Database class or a Game class. The Database is referred to through the IDatabase interface and the Game is referred to through the abstract parent class Game, allowing different Games to be swapped in and played on the system. The destination of the message is determined by the value stored in the type variable within the Message instance, which is a member of the messageType enum in the Message class. The Messages are distributed as follows:

- MESSAGE – Sent to every other client currently in chat.
- COMMAND – Sent to the Game instance stored within the game variable.
- LOGIN – Sent to the Database instance stored within the database variable.
- LOGOUT – Sent to the Database instance stored within the database variable.
- REGISTER – Sent to the Database instance stored within the database variable.
- PASSWORDHINT – Sent to the Database instance stored within the database variable.

In order to allow the Game class to send messages to the client, the server also provides a suite of functions to distribute messages and create scenarios desirable for the game, such as muting/unmuting certain players or retrieving all active players within the chat upon starting the game. To establish this contract and simplify testing through the creation of stubs, the Server implements the IServer interface, which lists all these functions.

The Server maintains an array of threads which are fed connections through a LinkedBlockingQueue, the number of threads within the pool is determined at run-time, and represents the maximum number of concurrent users the system can take. Each thread is an instance of the ClientHandler inner class which extends Thread.

Client

The client class has 8 private variables. There are cGson (Gson object), socket (Socket), outputStream (DataOutputStream), inputStream(DataInputStream), ip (String), port(Integer), online (boolean), and listener (ListenerThread). The class contains an inner class, ListenerThread which constantly reads input from server.

The ListenerThread class has four variables, an object of the chatGame class (window), list models for messages (modelMessage), users online (modelUser), and private messages (modelPrivate). The inner class has five void methods. The addUser method adds the users who are online shown on the application. the removeUser methods removes anyone from the list who have disconnected from the server. The printMessage method prints messages on the message window. The serverMsgPrint method prints private messages on a separate window. the run method initialises the input stream from the server upon connecting and differentiates the input from the server based on the types defined in the ServerMessage class.

The constructor of the Client class creates an instance of the Client. The GsonBuilder is instantiated and an Gson object is created. The sockets are defined together with the input stream and the output stream. The spawnListenerThread method initialises the ListenerThread inner class. the killListenerThread method interrupts the listener thread if input is paused. The getResponse method reads in the input from the server and decodes it from Json text to Java string (decodeServerMessage method is called). The sendClientMessage method sends messages from the server. The decodeServerMessage method decodes messages from Json text to Java String.

The createLoginPacket method sends out a login message (defined in the Message class) to the server and returns the response (server message defined in the ServerMessage class) if or not the login was successful. createAccountPacket method sends a sign up request and returns a response if the user manages to create an account successfully. The createLogoutPacket sends a logout message request and closes the socket. The forgottenPassword method sends a message type of password hint and returns the question if successful. Then the user has to enter the answer and the response is the password. The setCommandMsg method sets the messages sent from the user, whether it is a command (starts with "/") or a normal message.

ChatClientApp

This class has four variables, an object of the Client, an integer port, a string host, and an object of chatGame. The constructor creates an instance of the Client and chatGame. The main method executes the application.

Mafia (game)

Players send the message "/ready" to indicate that they are ready to play a game of Mafia. Every time a new player is ready, the server sends a public message mentioning the new player along with the number of players that are currently set to ready. If the player sends "/ready" again, they are informed by a private message from the server that they are already set to ready and doing this again has no effect.

Once the number of players that are ready is at least 6, the server sends a private message to the ready players telling them that they can vote to start the game by sending the message "/start". The game only starts when all the ready players have voted to start. Every time a player votes to start the game the server sends a public message to all players on the server mentioning the player

that wants to start as well as the number of players that have voted for this to occur. If a player who isn't set to ready vote to start they are informed that they cannot do this while they are not set to ready. Until all the ready players have voted to start more players can be set to ready, where they also need to vote to start. If the amount of players set to ready reaches 16 players, then any more players trying to set themselves to ready are informed by private message from the server that the game is ready full.

A player who is set to ready can (as long as the game hasn't started) send the message "unready" to remove themselves from the ready list. If a player who is not ready sends the "unready" message, they are informed via private message from the server that this command has no effect when they are not set to ready to begin with. If the number of players who are set to ready goes below 6 (minimum needed to play Mafia), the vote to start is cancelled. When the number of ready players reaches 6 again, the vote to start is reinitialised, where each player needs to make their vote to start the game again.

Once all the ready players have voted to start the game, the system starts the game by assigning the role of mafia to a third of the players (rounded to nearest whole number). The system sends each of the players a private message through the server telling them their role, and in the case of the mafia, who the other mafia players are. The game starts in the day phase which is the main phase of the game.

When the game is in session, whenever a player issues a command the system first checks whether the game is in the day or night phase and then compares the command message to the known commands. If this matches, then it invokes the relevant method.

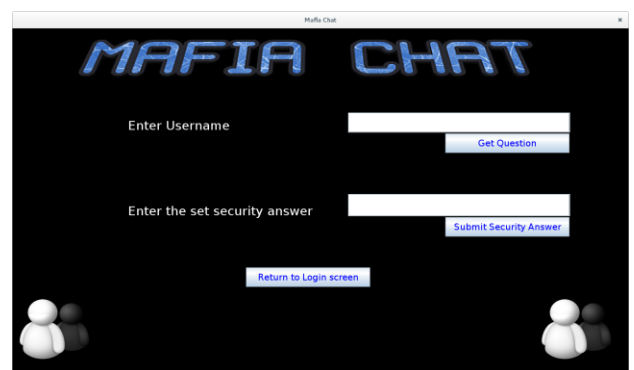
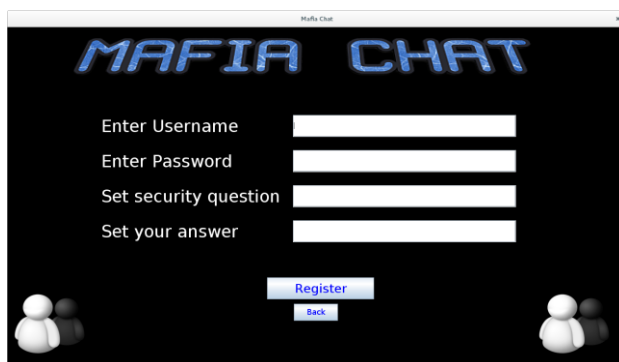
During the day a vote can be started to eliminate a player (started by sending "/elim [player name]" or a vote started to change the game to night (started by sending "/night").

Client

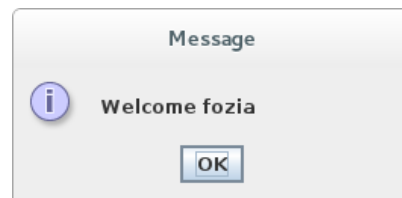
Working System Implementation

Figures 3h, i, j, k, l, m, n

Screenshots of our working Graphical User Interface.



The GUI was initially created much smaller however when tested out on the Linux machine, due to resolution issues this was changed. The design is implemented and was kept simple, the dark background helped created smart image and the font style used help to resemble a 90's style chatroom. JButtons we used for each button, JLabels were used for all text which is being displayed. This was simple to code and initially done using window builder however later changed and done in source view only.



The print screens above show a user logging in and sending a message.

Test plan

Testing mafia class

The tests for the Mafia class are designed to check if the user commands work as intended, as well as making sure that the game rules are implemented correctly. For the commands, users should only be able to use /ready, /unready and /start when there is not a game in progress. When the game is in progress and it is the day phase, the valid commands are /elim [playername], /save [playername], /night and /day. While the game is in progress and it is the night phase, the valid command is /elim, where only the Mafia can use this command. While the game is in progress, the valid commands can only be used by the users that are players in the game. For the /elim and /save commands, the [playername] has to be the name of a user in the game. For the /elim and /night votes, there needs to be a majority in favour for the

vote to be successful. For elimination vote at night all the mafia users need to use /elim for the same innocent player in order for it to be successful. The above have been tested with JUnit tests, a summary of what was expected of each test as well as their actual outcome is shown in the accompanying table (appendix 1).

When a vote starts during a game, a timer is set so that if the voting takes too long the timeout result occurs. The amount of time before the timeout for each of the types of votes is the same length of time (20 seconds). For the votes during the day, the first instance of a valid /night or /elim vote starts the timer for that vote. If the timer reaches the timeout duration, the aforementioned types of votes are successful if the majority of the votes are for the motion. As for the timer for the elimination vote by the mafia during the night, if the timer reaches the timeout duration, the vote fails and no-one is killed during the night.

Testing GUIs

Testing the GUI cannot be done as simply as JUnits test. Therefore, functional tests have been created to ensure all areas of the systems interface have been tested. At the start of the process of constructing the GUI tests were done throughout to check for layout and design. The GUI was resized to resolve issues of resolution.

Once the system was fully functional the GUI was opened in a variety of different situations to test the GUI at extreme working levels as well as under 'normal' running conditions.

Problems which were encountered:

1. The logout button was not working.
2. The Rules button was not working.
3. The Scroll bar for the Global Text Box would not allow automatically move down to show latest messages.
4. There were many other teething issues that were often quickly tested out whilst creating code and fixed straight away.

GUI extreme conditions testing

The GUI was tested under abnormal conditions to see what effect this would have on the overall system (see Table 3 in Appendix).

Testing client and server interaction

As the client and Server were developed in parallel, they were tested together throughout the development of the system. The tests used stubs for the Database (DatabaseStub) and for the Game class (GameStub), which were both independently tested and verified to be working correctly with their own JUnit tests (For evidence see SVN). As there were difficulties involved in JUnit testing a server with multiple clients, these tests were carried out from command-lines or through the GUI with the response from the system recorded here. The tests are carried out from the client side and record the response from the server (see Table 4 in Appendix).

Team organisation report

As Team Nice we worked very well together, we decided not to appoint a team leader instead decided each team member would take responsibility to manage their own workload.

Tasks were allocated based on each team member's choice and strength, as shown below.

- Jonathan Keslake – Server
- Ian Kirkman – Game
- Jacob Smith – Database
- Vishnu Raman – Client
- Fozia Mehboob – GUI's
- Jacob was elected as the group secretary and
- Vishnu as the point of contact for David.

The approach we took when tackling the project was we arranged two meetings per week, initially these were on Tuesdays and Wednesdays however quite early in the project we came to the understanding that between each meeting we needed time to complete tasks set out, therefore the Wednesday meeting was changed to Thursday. Tuesday we would meet without tutor and showcase progress had made. This was extremely useful as ensure we had set work completed each week to show.

In addition to the weekly meetings we also communicated regularly on a Facebook chat group. We often asked one another questions and gave support to each other. Fozia and Vishnu also decided to hold additional meetings on weekend at University, this is because the GUI and Client classes need

to synchronize and pass in and out information. Jon and Jacob also held extra meeting for their Database and Server communication. Towards the end of the project we often spend whole afternoons in the computer labs working on code together.

Overall the project was a success based on great communication and teamwork.

Evaluation

The project goal was to achieve a multi-threaded Mafia game in a chat client. This was done by having users register a username, password, and security question with answer. Once the user logged into an account, the user would be brought to a game screen where they could view other users to chat, initiate a game with the other users, and close the client as needed. For this to be accomplished a: server, client, database, game logic, and GUI were needed to be constructed. Several interfaces tied each of the classes together through the different functions and methods being called upon. Thorough testing was required in order to prove the functionality of the game.

The features currently work as intended. The server provided overall functionality for communication amongst the GUI, game, chat client, and database:

- The GUI facilitates signing up, logging in, and communicating through the chat client to play the game including private messages
- The database stores the user's information so that they can log into the server in order to chat or play.
- There is functionality for a whole new game to be played, Resistance, through use of the interface method calling.

However, there were several ideas that were hard to manage based on the length and scope of the project. Code refactoring would have been useful in terms of organising and simplifying the classes. Also, the GUI would have been nicer to have been built in separate classes for each window rather than in just one class. The GUI could have been flashier with resizable windows in order to cater to user's personal preferences. While a second game was incorporated, more instances of games would have been a welcome addition with the ability to share joinable games in private messages. Some such features that we would have liked to add were:

- GUI windows designed for players that were either Mafia or Innocent
- Storage for chat history and high scores
- Multiple roles for Mafia (Doctor, Detective, etc.)
- Multiple lobbies
- Configuration file before game starts for each game, allowing a game designer to adjust the look of the chat room being used to play their game

Overall, we have successfully completed what we had set out to do and then some. A fully functional client server has been integrated with a GUI, database, and game. Players get to register, chat with friends, and enjoy a relaxing game of Mafia or Resistance at their leisure.

Appendix

Table 1

Test No.	Test name	Expectation	Outcome
1	testActiveClients	ArrayList containing active client ids is not empty	As expected
2	testReady	User in ready list after using /ready command	As expected
3	testReady2	User only appears once in ready list after using /ready command multiple times	As expected
4	testReady3	After 16 users are in the ready list no more users can be added	As expected
5	testUnready	User is removed from ready list after using /unready command	As expected
6	testUnready2	User that uses /unready command is the one removed from the ready list	As expected

7	testUnready3	Ready list still empty after user uses /unready command	As expected
8	testUnready4	User in the ready list is not removed from the list if a user, not in the ready list, uses /unready command	As expected
9	testUnready5	User that uses /unready command after using /start command is removed from both the votedStart list and the ready list.	As expected
10	testUnready6	User that uses /unready command after using /start command does not remove other users from the votedStart list.	As expected
11	testUnready7	User that uses /unready command will only affect the votedStart if that user is in the votedStart list as well as the ready list.	As expected
12	testStart	User using /start command will not add the user to the votedStart list if the size of the ready list is less than 6 users	As expected
13	testStart2	User using /start command will add user to the votedStart list when size of ready list is at least 6 users	As expected
14	testStart3	All users in ready list (where list's size is at least 6) using /start command will start the game. Game is starts in day phase with a third (rounded to the nearest whole number) of the players as mafia.	As expected
15	testStart4	User in ready list but not in votedStart list will stop the game from starting even if the votedStart list's size is at least 6.	As expected

16	testElimDayVote	UserA using /elim for a player causes that player to be on trial and UserA is added to the elimDayVote list.	As expected
17	testElimDayVote 2	User using /elim multiple times only adds the user to elimDayVote list once.	As expected
18	testElimDayVote 3	User using /elim for userB is not counted when /elim has already been used for userA	As expected
19	testElimDayVote 4	Exactly half of the users using /elim for a player will not eliminate the player as a majority hasn't been reached	As expected
20	testElimDayVote 5	Majority of the users using /elim for a player will eliminate the player. The eliminated player is muted.	As expected
21	testElimDayVote 6	User using /elim for a player when a nightVote is in progress will not be counted	As expected
22	testElimDayVote 7	User using /elim for a non-player is not counted.	As expected
23	testElimDayVote 8	User using /elim for themselves is not counted	As expected
24	testSaveVote	UserA using /save for userB when userB is on trial will add the userA to the save list.	As expected
25	testSaveVote2	UserA using /save for a user after using /elim for that user will remove UserA from the elimDayVote list and add them to save list. This does not cancel the dayElimVote even if UserA started it.	As expected

26	testSaveVote4	User using /save command for a user will have no effect when playerOnTrialID is null.	As expected
27	testSaveVote5	Exactly half of the users (if number of players is even) using /save for a player will save that player.	As expected
28	testSaveVote6	Majority of users using /save command for a player, when number of players is odd, will save that player.	As expected
29	testSaveVote7	User using /save command for a user that isn't in the game will have no effect.	As expected
30	testSaveVote8	For an even number of players when half of users use /elim for a player and the other half then uses /save, the overall result is the player is saved.	As expected
31	testNightVote	User using /night commands adds the user to the nightVote list and sets nightVoteInProgress to true.	As expected
32	testNightVote2	User using /night command twice only adds the user to the nightVote list once.	As expected
33	testNightVote3	Exactly half of the users using /elim for a player will change the game to night as a majority hasn't been reached	As expected
34	testNightVote4	Majority of the users using /night will change the game to night and end the vote.	As expected
35	testNightVote5	User using /night when there is a vote to eliminate a player will have no effect.	As expected
36	testDayVote	User using /day command when there is a nightVoteInProgress will add that user to the	As expected

		dayVote list.	
37	testDayVote2	User using /day for a user after using /night will remove user from the nightVote list and add them to dayVote list. This does not cancel the dayElimVote even if UserA started the dayElimVote.	As expected
38	testDayVote3	User using /day command will have no effect when a nightVoteInProgress is false.	As expected
39	testDayVote4	User using /day after using /night will remove that user from the nightVote list and add them to dayVote list. This does not cancel the nightVote even if User started it.	As expected
40	testDayVote5	Exactly half of the users (if number of players is even) using /day will keep the game as day.	As expected
41	testDayVote6	Majority of users using /save command for a player, when number of players is odd, will save that player.	As expected
42	testElimNightVote	All mafia users use /elim in the night for the same innocent leads to that player being eliminated and game being switched back to day.	As expected
43	testElimNightVote2	Mafia user using /elim for themselves has no effect.	As expected
44	testElimNightVote3	Mafia users using /elim for another mafia user has no effect.	As expected
45	testElimNightVote4	Mafia users using /elim for different innocent players causes no one to be eliminated during the night.	As expected

46	testElimNightVote5	Mafia users using /elim for different innocent players causes no one to be eliminated during the night even if there was one of the innocents had a majority of the votes.	As expected
47	testElimNightVote6	Innocent user using /elim during the night has no effect.	As expected
48	gameEnd	After win condition for innocents is reached the game ends correctly. Value of gameInProgress is false and all players that were muted are now unmuted.	As expected
49	gameEnd2	After win condition for mafia is reached the game ends correctly. Value of gameInProgress is false and all players that were muted are now unmuted.	As expected
50	testMafiaNum	When game starts the number of mafia is equal to a third of the number of players (rounded to nearest whole number).	As expected
51	testMafiaRandom	When game starts the mafia are assigned randomly.	As expected
52	gameStartedReady	Once the game starts the /ready command is not a valid command.	As expected
53	playerUnready	Once the game starts the /unready command is not a valid command (even for a player).	As expected
54	playerVoteStart	Once the game starts the /start command is not a valid command.	As expected
55	nonPlayerVoteElim	User using the /elim command during the day, is not counted when that user isn't a player.	As expected

56	nonPlayerVoteSave	User using the /save command during the day, is not counted when that user isn't a player.	As expected
57	nonPlayerVoteNight	User using the /night command is not counted when that user isn't a player.	As expected
58	nonPlayerVoteDay	User using the /day command is not counted when that user isn't a player.	As expected
59	nonPlayerElimNight	User using the /elim command during the night, is not counted when that user isn't a player.	As expected

Table 2

Test Number	Test name	Expectation	Outcome
1	testEnterButton	When user presses the Enter button it should take them to the log in screen.	When I press the enter button from the title page it directs me to the log in screen.
2	testTitleDisplay	The 'Mafia Chat' title should appear at the top of each screen and stand out in blue.	The title is a vibrant blue design that is clearly displayed at the top of each page.
3	testLogoDisplay	The logo should be a white bubble person and appear in the centre of the screen below the title.	On all pages that it appears, the logo is a consistent white bubble person design. On the title page it appears in the centre of the screen.
4	testDareTextDisplay	The 'Dare to Play' text	The Dare to Play' text is

		should be the same font style as the title but smaller. It should appear below the title but above the 'Enter' button.	a similar vibrant design to the main title text. It features below the logo on the title page and the '...' evokes a curiosity.
5	testUsernameEntryBox	The Enter Username box should be a white box that enables the user to type in their unique username.	The Enter Username box is a sufficiently sized box in which I was able to type my username.
6	testPasswordEntryBox	The Enter Password box should be a white box that enables the user to type in their unique password that matches the username.	The Enter Password box is a sufficiently sized box in which I was able to type my password.
7	testLoginButton	When the user presses the Log In button a welcome message should pop up along with the user's unique username - 'Welcome (username)'	When I filled in my log in details and pressed the Log In button a message appeared saying 'Welcome' with my username.
8	testSignUpButton	When the user presses the Sign Up button it should take them to the registry screen in which they can enter a unique username, password and security question and	On my first visit to the game I pressed the Sign Up button and was redirected to a new screen that prompted me to set a username, password and security

		answer.	information.
9	testForgottenPassword	When the user presses the Forgotten Password button it should take them to a screen in which they will be prompted to enter their registered username and security information.	On an instance in which I had forgotten my password I clicked on the Forgotten Password button and it redirected me to a screen so that I could answer my security question.
10	testLogoDisplay	The logo should be displayed as two white bubble people, one in the bottom left corner and one in the bottom right corner of the screen.	On all pages that it appears, the logo is a consistent white bubble person design. On the log in and sign up page it appears twice, once in each of the bottom right and left corners.
11	testTitleTextDisplay	The 'Mafia Chat' title should appear at the top of each screen and stand out in blue.	The title is a vibrant blue design that is clearly displayed at the top of each page.
12	testEnterUsernameEntryBox	The Enter Username box should be a white box that enables the user to type in their unique username.	The Enter Username box is a sufficiently sized box in which I was able to type my username.
13	testEnterPasswordEntryBox	The Enter Password box should be a white box	The Enter Password box is a sufficiently

		that enables the user to type in their unique password that matches the username.	sized box in which I was able to type my password.
14	testGetQuestion	When the user presses the Get Question button their selected security question should be displayed on the screen.	Having forgotten my password I needed to enter my security information. When I clicked on the Get Question button, my registered security question appeared on the screen.
15	testSubmitSecurityAnswer	When the user presses the Submit Security Answer button a welcome message should pop up along with the user's unique username - 'Welcome (username)'. If the question has been answered correctly.	Having typed in the answer to my security question, I pressed the Submit Security Answer button and a message popped up on the screen welcoming me with my username.
16	testReturnLoginScreen	When the user presses the Return to Login screen button it should take them back to the Log In screen.	When I pressed the Return to Login screen button I was redirected to the Log In screen.
17	testEnterUsername	The Enter Username box should be a white box	The Enter Username box is a sufficiently

		that enables the user to type in a unique username.	sized box in which I was able to type my username.
18	testEnterPassword	The Enter Password box should be a white box that enables the user to type in a unique password.	The Enter Password box is a sufficiently sized box in which I was able to type my password.
19	testSetSecurityQuestion	The Set security question box should be a white box that enables the user to type in a selected security question.	When setting my security information I was able to type my own security question in the Set Security Question box.
20	testSetAnswer	The Set your answer box should be a white box that enables the user to type in a selected answer to the relevant security question.	When setting my security information I was able to type the answer to my security question in the Set your Answer box.
21	testRegisterButton	When the user presses the Register button a welcome message should pop up along with the user's unique username - 'Welcome (username)'.	Having filled in all my details I pressed the Register button and a welcome message popped up along with my username.
22	testBackButton	When the user presses the Back button it should	I pressed the back button and was

		take them back to the Log In screen.	redirected back to the Log In screen.
23	testPressLoginWithNoCreditials	If a user presses the Log In button without filling in the credentials then an error message should appear instructing them to enter their unique username and password.	On one occasion I pressed the Log In button before I had entered my username and password. An error message popped up alerting me that I needed to fill in my unique username and password before continuing.
24	testViewOnlineUsers	The Online Users box should be a white box that displays the usernames of all users currently online.	Whilst in the chatroom I could view who was online as their usernames were displayed in the Online Users box on the right of the screen.
25	testViewServerMessages	The Server Messages box should be a white box that displays messages received from the server.	Whilst in the chatroom I could view any server messages that had been sent as they appeared in the Server Messages box on the right of the screen.
26	testSendButton	When the user presses the Send button it should	Having typed my chat message I pressed the

		send their typed message to other users. It should also display their username and message within the Global Message box.	send button and the message appeared in the Global Message box along with my username and the time that I sent the message.
27	testRulesButton	When the user presses the Rules button a pop up should appear listing all of the rules and regulations of the Mafia Game.	When I pressed the Rules button a pop up appeared that informed me of all of the rules of the Mafia Game.
28	testLogOutButton	When a user presses the Log Out button it should take them back to the title screen.	At the end of my session I pressed the Log Out button and was redirected back to the title page.
29	testGlobalMessageDisplay	The Global Message box should be a large white box in which all sent messages can be displayed and seen. The messages should also show the time of the message and the username of the sender.	Having typed my chat message I pressed the send button and the message appeared in the Global Message box along with my username and the time that I sent the message.
30	testTextEntryBox	The Text Entry box should be a white box at the bottom of the screen	I was able to type my messages in to the Text Entry box at the bottom

		in which the user should be allowed to type their message prior to sending it.	of the chatroom screen prior to sending it.
31	testScrollBarTextEntryBox	When the user drags the scroll bar it should allow them to scroll up and down the Text Entry box to read their full message prior to sending.	When I was writing a long message the scroll bar enabled me to scroll back up so that I could re-read the whole message before pressing send.
32	testScrollBarGlobalMessageBox	When the user drags the scroll bar it should allow them to scroll up and down the Global Messages box to read previous sent messages.	When wishing to view previous messages in the Global Message box, I was able to use the scroll bar to look back through the messages.

Table 3

Test Number	Test name	Expectation	Outcome
1	testExcessiveTogglingBetweenLogInandRegistryScreen	If a user toggles between the Log In screen and the registry screen more than 30 times by excessively pressing the Sign Up and Back buttons the system should continue to work normally and move from one screen to the other	Whilst on the Log In screen I pressed the Sign Up button to take me to the registry screen. I then pressed the Back button and was taken back to the Log In screen. I repeated this over 30 times moving from one screen to the other and the system

		without any issues.	continued to work, taking me between the two screens.
2	testExcessiveTogglingBetweenLogInandSecurityQuestionScreen	If a user toggles between the Log In screen and the Security Question screen more than 30 times by excessively pressing the Forgotten Password and Return to Log In Screen buttons the system should continue to work normally and move from one screen to the other without any issues.	Whilst on the Log In screen I pressed the Forgotten Password button to take me to the security question screen. I then pressed the Return to Log In Screen button and was taken back to the Log In screen. I repeated this over 30 times moving from one screen to the other and the system continued to work, taking me between the two screens.
3	testExcessivePressingOfLogInButton	If a user fails to type in their username and password but continues to press the Log In button over 30 times the system should continue to display an error message every time the button is pressed. The error message should prompt the user to complete the required username and password fields.	Without typing in any details I pressed the Log In button and was immediately prompted to enter my username and password details by an error message. I ignored this and continued to press the Log In button over 30 times. Each time I pressed it the same error message appeared.
4	testExcessivePressingOfRegisterButton	If a user fails to type in their username, password, set security question and/or set answer fields, on the registry screen, but continues to press	Without typing in any details I pressed the Register button and was immediately prompted to enter a unique username and password and to set a security

		the Register button over 30 times the system should continue to display an error message every time the button is pressed. The error message should prompt the user to complete the required username, password, set security question and set your answer fields.	question and answer by an error message. I ignored this and continued to press the Register button over 30 times. Each time I pressed it the same error message appeared.
5	testExcessivePressingOfGetQuestionButton	<p>If a user fails to type in their username but continues to press the Get Question button more than 30 times the system should continue to display an error message prompting the user to enter their username before continuing.</p> <p>If the user has correctly entered their username but, nevertheless, continues to press the Get Question button then the system should continue to display the security question set during user registration.</p>	Without entering my username I pressed the Get Question button. An error message appeared instructing me to enter a valid username. I continued to press the Get Question button over 30 times regardless. The same error message appeared each time I pressed the button. When I then correctly entered my registered username and pressed the Get Question button, the set security question appeared. This happened regardless of how often I pressed the Get Question button. I pressed it over 30 times.
6	testExcessivePressingOfSubmitSecurityAnswerButton	If the user types in an incorrect answer to the security question (or no answer at all)	Without getting the answer to my security question correct I still pressed the Submit Security

		then an error message should appear informing the user that they have not entered the correct username. If the user continues to press the Submit Security Answer button more than 30 times the error message should continue to appear each time the button is pressed.	Answer button. I did this over 30 times and each time the same error message appeared informing me that I had not entered the correct answer to my set security question.
7	testExcessiveSendingOfBlankMessages	In the chatroom screen, if a user fails to write a message in the Text Entry box and presses the Send button then an error message should appear informing the user that they must type a message to be sent. If the user then continues to press the Send button over 30 times then the error message should continue to be displayed each time the button is pressed.	Whilst in chat I pressed the Send button without having written a message in the Text Entry box. A message appeared telling me that I had to type a message before attempting to send. Nevertheless, I continued to press the Send button over 30 times but the same error message kept appearing.
8	testExcessivePressingOfRulesButton	If a user continues to press the Rules button more than 30 times then the pop up explaining the rules of the game should continue to appear each time the button is pressed.	I pressed the Rules button repeatedly more than 30 times. Each time I pressed it the rules pop up continued to appear.

9	testProlongedInactivity	If a user has been inactive for more than one hour then the system should automatically log them out.	At one point I was away from the game for over an hour and the system automatically logged me out of my account.
---	-------------------------	---	--

Table 4

Test Number	Test Description	Expectation	Outcome
1	Log-In valid user, valid password	No error thrown	No error thrown
2	Log-in invalid user	InvalidUserException thrown	InvalidUserException thrown
3	Log-in valid user, invalid password	InvalidInformationException thrown	InvalidInformationException thrown
4	Register valid user, and log that user in	No error thrown	No error thrown
5	Register already taken user	UserExistsException thrown	UserExistsException thrown
6	Get security question for a valid user	No error thrown	No error thrown
7	Get security question for invalid user	InvalidUserException thrown	InvalidUserException thrown
8	Check question answer for valid fields	No error thrown	No error thrown
9	Check question answer for invalid	InvalidUserException thrown	InvalidUserException thrown

	user		
10	Check question answer for invalid answer	InvalidInformationException thrown	InvalidInformationException thrown

Table 5

Test Number	Test Description	Expectation	Outcome
1	Send Message instance with Login messageType with valid user and password	Receive success ServerMessage instance with empty messageText	Success
2	Send Message instance with Login messageType with invalid user	Receive failure ServerMessage instance with relevant error message	Success
3	Send Message instance with Login messageType with valid user and invalid password	Receive failure ServerMessage instance with relevant error message	Success
4	Send Message instance with Login messageType empty string literals "" ""	Receive failure ServerMessage instance with relevant error message	Success
5	Send Message instance with Register messageType with valid fields	Receive success ServerMessage instance	Success
6	Send Message instance with Register messageType with taken user	Receive failure error ServerMessage instance with relevant error message	Success

7	Send Message instance with Login messageType valid user and empty password ""	Receive failure error ServerMessage instance with relevant error message	Success
8	Send register Message instance with empty literals	Receive failure error ServerMessage each time	Success
9	Send Passwordhint Message instance with only a valid username	Receive success ServerMessage instance with the security question	Success
10	Send Passwordhint Message instance with only an invalid username	Receive failure ServerMessage instance with relevant error message	Success
11	Send Passwordhint Message with an empty messageText (empty username)	Receive failure ServerMessage instance with relevant error message	Success
12	Send Passwordhint Message with a valid username and question answer combination	Receive success ServerMessage instance with relevant error message	Success
13	Send Passwordhint Message with an invalid username	Receive failure ServerMessage instance with relevant error message	Success
14	Send Passwordhint Message with a valid username and invalid question answer	Receive failure ServerMessage instance with relevant error message	Success
15	Send Passwordhint Message with empty question answer	Receive failure ServerMessage instance with relevant error message	Success

16	We logged in two users and tested if the messages are sent to the right person. For example, when user1 was in chat and user2 logs in right after. We checked if the message are sent to the right person (chat appears in user1's chat window, and welcome message appears in user2's welcome text box).	Messages sent from chat are not sent to users who are still logging in	Success
17	Multiple users created on different machines and users constantly sends messages out	The messages appear in the chat box of all users.	Success
18	Send private message (/private <user> <message>) when in regular chat mode to specific user.	The message appears in the private message box of the intended user.	Success.
19	Server sending game specific server public message to all users.	The messages appear in all users' chat window.	Success
20	Relaying chat between users in regular chat mode (check if the messages are sent chronologically)	The messages are sent chronologically	Success
21	Send message Message instance from one client	Chat ServerMessage instance is received by all clients in chat and none in the login screen	Success

22	Send /public message within command type Message instance packet to server	Receive public ServerMessage instance with the sent message only	Success
23	Send /mute testUser1 within a command type Message instance packet to server	testUser1's chats are no longer relayed	Success
24	Send /unmute testUser1 within a command type Message instance packet after the user has been muted	testUser1's chats are relayed	Success
25	Send /chat within a command type Message instance packet	Chat is no longer relayed	Success
26	Send /chat within a command type Message instance one chat is no longer being relayed	Chat is being relayed	Success
27	Click rules button on Gui	Dialog box opens with the rules attached to the parent Game object	Success
25	Clicking log out button successfully closes the connection of that user and updates list of users online	The user logged out does not appear on users list and the connection was terminated.	Success
26	Have 6 users typing simultaneously with all chats being relayed and all commands accepted	No errors seen, no messages missed and all commands correctly processed	Success
27	Send multiple commands to	User in login receives none of the	Success

	the server while a user is waiting in login	private/public messages	
28	Check if the time stamps on messages are correct and in order.	The time stamps in the chat box are accurate	Success

Project diary (meeting minutes)

Meeting 1 - February 15 - 1:00 PM

Attendants: Jacob, Fozia, Jonathan, Ian, Vishnu

Collected group members email addresses.

Ideas Considered:

- Chat Box
 - Chat history saved locally
 - Database storing emoticons
 - Pre-set chatrooms with sub chatrooms
- Shark fish eating game
 - Database can store high scores
 - Share button for social media

GitHub or Subversion? GitHub for now

Roles assigned:

- Fozia: GUI
- Jacob: Secretary, Database
- Vishnu, Ian, Jon: Client-Server
- Vishnu: Point of contact with David

Settled on a Mafia chat room game.

Preliminary notes on what we would like to do with the game:

- Admins have usernames that begin with @
- Server announcements and private announcements
- See when users are typing

- Pre-made static chatrooms
- Animations in GUI
- Timers in terms of moves to be made
- Minimum 6 players
- Players state when ready to play before the game begins
- When a person is eliminated from the game they remain in the room, but unable to participate
- Database
 - Stores the usernames and passwords (account info) of users
 - Achievements
 - Points earned
 - Chat history

Google drive will be our primary file sharing server and meetings will be every Wednesday at 1 PM.

Meeting 2 - February 22 - 1:00 PM

Attendants: Jacob, Jon, Fozia, Vishnu

Meetings will now be Thursday at 12:00PM

Google Hangouts for communication

Interactions will be decided as a group, then reports will be drafted

All MSc testing must be done

Will be using SVN instead of GitHub

Preliminary GUI design:

- Login page
- Next page is rule page
- Game page that has the chat and features for interacting with the game

Impose a character limit on text box, store local copies of chat logs on local machine not database.

Write most of the server and game rules using interfaces so that we can make changes and keep the functions without having to implement.

Tasks:

What tasks are people doing?

Jon says we'll do a test a server that will bind with the database Jacob creates, so we'll get the client to use the server to call the test database.

So this weekend we will do a test database and test server so that we can run tests the function of the databases.

How will we be designing the database, client-server, gui relationship for users to be able to register?

The only information that is required to create an account will be username and password.

So 1 table for username, password. An additional idea that would be nice is to have an integer for games won next to a username.

Meeting 3 - February 28 - 2:00 PM

Attendants: Jacob, Vishnu, Ian, Jon, Fozia

Met with David to discuss our plans going forward with the Mafia chat game. We will be meeting the following Wednesday, tomorrow, to discuss tasks that we want to accomplish before the next week.

Meeting 4 - March 1 - 1:00 PM

Attendants: Fozia, Jacob, Ian, Jonathan, Vishnu

Ideas discussed:

Jonathan has created interfaces called IMessage {message, command} and ILogin {login, logout} that would handle any strings of texts and login requests. IServerMessage would take enum {private, public, chat}. Enum allows you to write out statements for example if (message) in text to look snazzy. IGame and IServer relays messages handled. Server stores IDs and game creates array of integers that is associated with what int is and what ID ie 0, 3, 4 are all mafia. Handles public Message String Message and private Message String, int [] recipients based on the roles of the players. Server should reject anything typed into chat at night. Java Security API needs to store hashed passwords, basically encrypt then decrypt.

Public messages will be used to update votes for night time, updating players being eliminated at night or during day.

Commands can be used as dashes.

Server can attach numbers to users and the numbers will then be matched with certain roles.

Tasks:

Tuesdays: Fozia will finalize the Login screen and open a game window.

Ian: Making the game rules.

Vishnu: Will establish the communication between Fozia GUI and logging into the game.

Jacob: Finishing user registration database and communication.

Jonathan: Client Server and interfaces sending messages/commands back and forth.

Meeting 5 - March 7 - 2:20 PM

Attendants: Fozia, Jacob, Jon, Ian, Vishnu

Discussion:

Client needs to be sorted in terms of Json still.

GUI: Enter takes you to next screen, brings you to login screen, if you hover over the buttons then it tells you what the options will do, text boxes work for logins. Done in windowbuilder.

Client class is still being worked on. Think about function tests for example button testing. JUnit testing is fine with expected result and actual result. Database test for same username should provide the expected result. Game needs to handle logging out, disconnect. If a user disconnects then make a disconnect function.

Tasks:

- Need to split up the paperwork so that the paper has been dispersed.
- Can WindowBuilder scale the image when changing size of the box? Perhaps make the window much bigger.
- Database still needs to communicate between the server and client side before users can actually register into the database.

Meeting 6 - March 9 - 1:00 PM

Attendants: Fozia, Jacob, Jon, Ian, Vishnu

Discussion:

Project Paper to do list

- Intro, Architecture: Database, server, client
- System Design: Whole system, rules (Ian please write this), protocol, Sequence Diagram, Class Diagram, Database, Server, Client (client gui)
- Working System Implementation: Database, Client, Server, GUI
- Test Plan: Database, Client, Server, GUI test plans
- Team Organization Report
- Evaluation
- Meeting Minutes

We will have more meetings if necessary

Tasks:

- GUI and the client will be connected this weekend.
- Client and server should communicate by tomorrow. That will enable the database.
- JUnit testing will be done by each user's writer

Meeting 7 - March 14 - 1:00 PM

Attendants: Jacob, Fozia, Jon, Ian, Vishnu

Discussion:

GUI complete – Sign up works as intended. Forgotten password screen requires for you to enter the username, it then pops up with the security question so it's not a text box, and then you must supply the security answer for a popup to appear with the password.

Top high scores page still possible.

Possible issues with majority vote being reached. Can't vote to eliminate someone while people vote from day to night, for example. With multiple threads both methods could be called at the same time and one hasn't been locked, there could be deadlock. Timer would be implemented for voting session. JUnit testing using multiple threads; test the most important methods only.

Functional tests should be considered i.e.; for clicking the buttons too much or incorrect logins.

12:00 Monday 20th for report; presentations are on Wednesday. Mention ideas that were not implemented into your report.

Meeting 8 - March 17 - 10:00 AM

Attendants: Jacob, Fozia, Jon, Ian, Vishnu

Group met to just work as a team together in the lower ground labs. Communication between GUI-client-server-database was the main focus of our meeting.

Meeting 9 - March 17 - 2:20 PM

Attendants: Jacob, Fozia, Jon, Ian, Vishnu

Met with David at this time to show off the functionality of our game client. Everything went well with showing how the GUI interacts with the client-server and database.

The group continued to work on implementing the functionality of the Mafia game into the client-server so that it could be played through the GUI.

Bibliographic references

Class Diagrams generated using ObjectAidUml Explorer:

ObjectAid, 2016. *The ObjectAid UML Explorer for Eclipse*. [Online]

Available at: <http://www.objectaid.com/home>