Software Design Specification

Group 7

Github Link:
https://github.com/yoshiyahoo/Communication-App

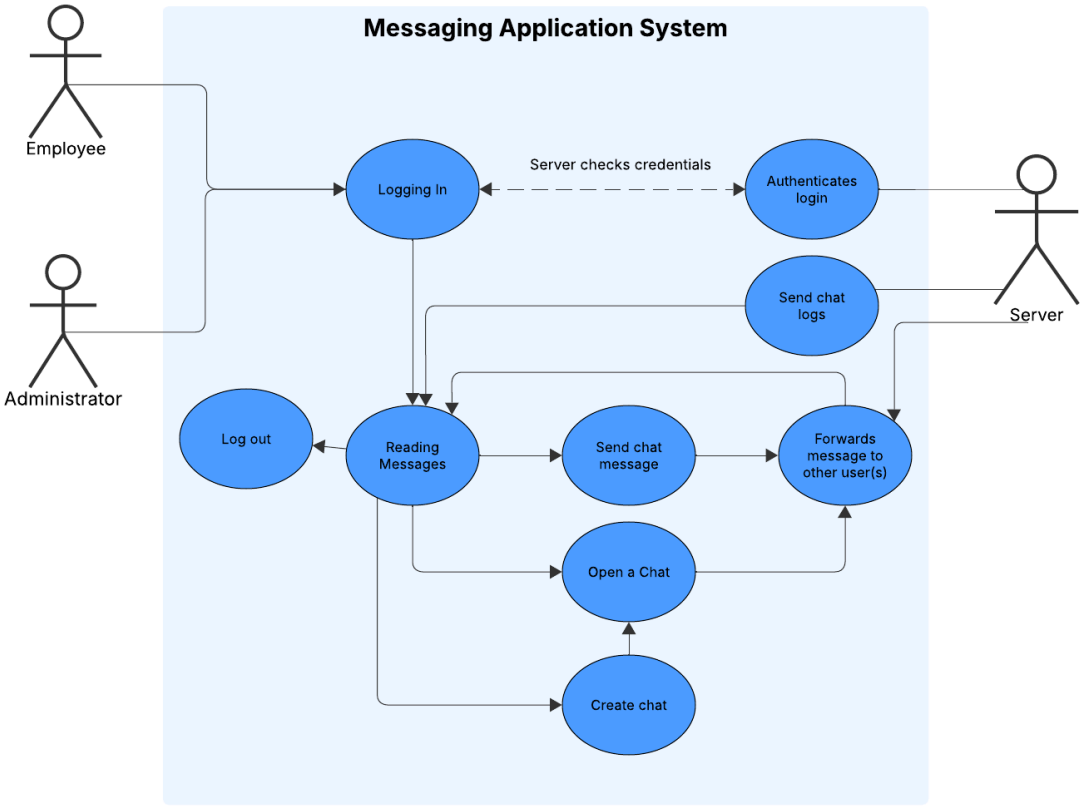
Revision History

Date	Revision	Description	Author
3/13/2025	1.0	Initial Version	Josiah
3/18/2025	1.1	Imported unformatted use cases	Jacob
3/18/2025	1.2	Made a list of new classes/attributes/methods to put in the new class diagram	Adam
3/20/2025	1.3	Added Figma Design	Kamari
3/20/2025	1.3.1	Formatted Use Cases, Table of Contents, and Revision History	Josiah
3/24/2025	1.4	Made and added Class Diagram V2	Adam
4/04/2025	1.5	Made and added V3 Class Diagrams	Adam
4/05/2025	1.6	Added and edited some use cases	Anna
4/08/2025	1.6.1	Further edited Use Cases	Josiah
4/08/2025	1.7	Add an overview for V4 of class diagrams	Adam
4/08/2025	1.8	Imported use case diagram	Anna
4/08/2025	1.9	Added V5 class diagrams	Adam
4/09/2025	1.9.1	Added Use Case 3 and Slightly Changed GUI Concept	Kamari
4/09/2025	1.9.2	Added Sequence Diagram for Use Case 5	Adam
4/10/2025	1.9.3	Added Sequence Diagram for Use Case 1 and formatted document.	Josiah
4/10/2025	1.9.4	Added Sequence Diagram for Use Case 2 and updated Use Case Diagram	Anna
4/10/2025	1.9.5	Formatted the Class Diagrams Section	Adam
4/10/2025	1.9.6	Added Gantt Chart	Josiah
4/10/2025	1.9.7	Added Github Link	Josiah

Table Of Contents

Revision History **1**
Table Of Contents **3**
UML Case Diagram **4**
Use Case Specification **5**
Use Case 1: User Logs In 5
Use Case 2: User Sends a Message 6
Use Case 3: User Opens a Chat With Other Users 7
Use Case 4: User Creates Chat With Other Users 8
Use Case 5: User Logs Out 9
UML Class Diagrams **10**
Overview: 10
Core Client-Server-Database: 11
Client-Server: 12
Database: 13
Client: 14
UML Sequence Diagrams **15**
Use Case 1: User Logs In 15
Use Case 2: Sending a Message 16
Use Case 3: Opening New Chat 17
Use Case 4: User Creates Chat with Other Users 18
Use Case 5: User Logs Out 19
GUI Concept Diagram **20**
Gantt Chart **21**

UML Case Diagram



Use Case Specification

Use Case 1: User Logs In

Relevant Requirements:
3.1.2.3 - 3.1.2.5, 3.1.3.1 - 3.1.3.2, 3.2.1, 3.3.2, 4.1, 4.2.1, 4.3.3 - 4.3.4

Primary Actor: User

PreCondition: User is not logged in
PostCondition: User is logged in and can start seeing chats

Basic Flow of Main Scenario:
1. User enters username
2. User enters password
3. User clicks login button
4. Client sends login request to server
5. Server checks database for login information
6. Server finds valid login information
7. Server sends login confirmation to client
8. Client window updated to message board

Alternative Flow: User Fails to Log In
1. User enters name
2. User enters password
3. Client sends login request to server
4. Server checks database for login information
5. Server can't find matching login information
6. Server sends login invalid confirmation to client
7. Client informs user login unsuccessful
8. Client can try to login again

Exceptions:
- Login Credentials Invalid
- Request timed out

Related Use Cases:
2, 3, 4, 5

Use Case 2: User Sends a Message

Relevant Requirements:
3.1.1.2, 3.1.2.2, 3.1.3.1, 3.1.3.3. - 3.1.3.6, 3.1.4.2 - 3.1.4.3, 3.2.2, 3.3.1, 3.3.4, 4.1.4, 4.2.1, 4.3.1. - 4.3.2.

Primary Actor: User

PreCondition: User hasn't sent their message yet
PostCondition: User sends their message for all to see

Basic Flow of Main Scenario:
1. User types in message
2. User clicks send button
3. Client sends message to server
4. Server receives the message
5. Server updates the chat log with new message
6. Server sends new chat log back to client
7. Client receives chat log
8. Client displays chat log for user

Alternate Flow: Network Failure
1. User types in message
2. User clicks send button
3. Client fails to connect to server
4. Message is marked as "failed to send"
5. User can try to resend message

Exceptions:
- Server takes too long to process message

Related Use Cases:
1, 3, 4, 5

Use Case 3: User Opens a Chat With Other Users

Relevant Requirements:
3.1.2.1, 3.1.3.1, 3.1.3.3, 3.1.3.6, 3.1.4.2 - 3.1.4.3, 3.2.2, 3.3.4,

Primary Actor: User

PreCondition: User is about to open the chat and is in a different chat
PostCondition: Chat is opened and user is ready to type

Basic Flow of Main Scenario:

1. User clicks chat icon
2. Client updates UI with up to date clicked on chat

Exceptions:

- None

Related Use Cases:
1, 2, 4, 5

Use Case 4: User Creates Chat With Other Users

Relevant Requirements: 3.1.1, 3.1.2.5, 3.1.2.2, 3.1.2.1, 3.1.3.5, 3.1.3.4, 3.1.3.3, 3.1.3.1, 3.2.2.1, 3.3.4, 3.3.3, 3.3.1.

Primary Actor: User

PreCondition: User hasn't created group chat
PostCondition: User has created group chat and users can start chatting

Basic Flow of Main Scenario:

1. User Clicks create new group chat
2. User searches for new users with search feature
3. User adds new users into group chat
4. User clicks enter
5. Client finds that this chat does not exist
6. Client sends open chat request with these users
7. Server creates new log file
8. Server sends back new log file to client
9. Client updates UI with new window

Alternate Flow:

1. User Clicks create new group chat
2. User searches for new users with search feature
3. User adds new users into group chat
4. User clicks enter
5. Client finds that chat already exists
6. Client informs user that chat exists
7. Client updates UI with new window

Exceptions:

- Server times out
- Chat already exists

Related Use Cases:
1, 2, 3, 5

Use Case 5: User Logs Out

Relevant Requirements:
3.1.2.6, 3.1.3.1, 3.2.2.2, 4.1.2, 4.1.4, 4.2.1, 4.3.4

Primary Actor: User

PreCondition: User is logged in
PostCondition: User is logged out

Basic Flow of Main Scenario:

1. User presses logout button
2. Client sends logout request to the server
3. Server receives request
4. Server logs client out
5. Server sends success message back to client
6. Client is sent back to login screen

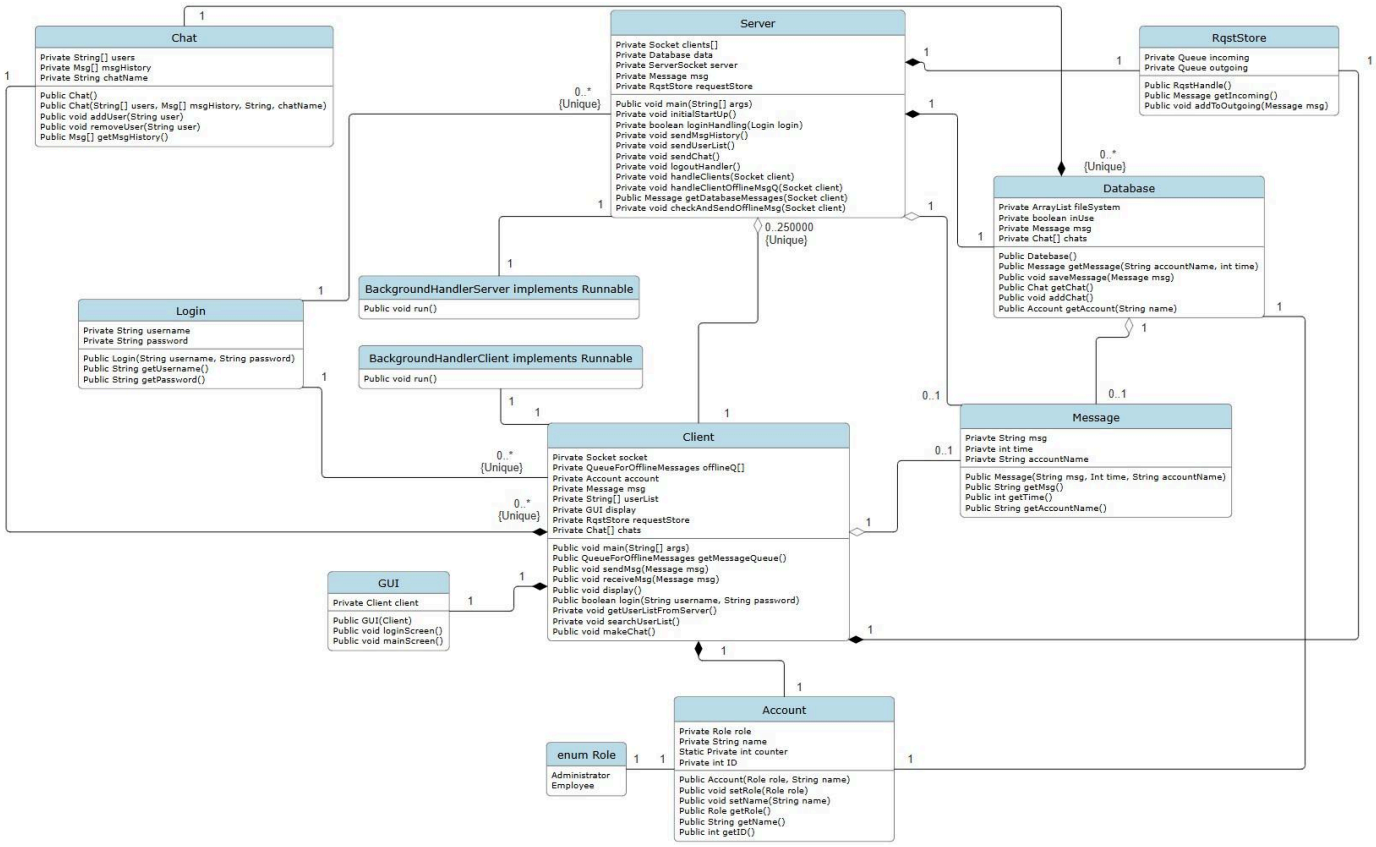
Exceptions:

- Request fails to send
- Server times out

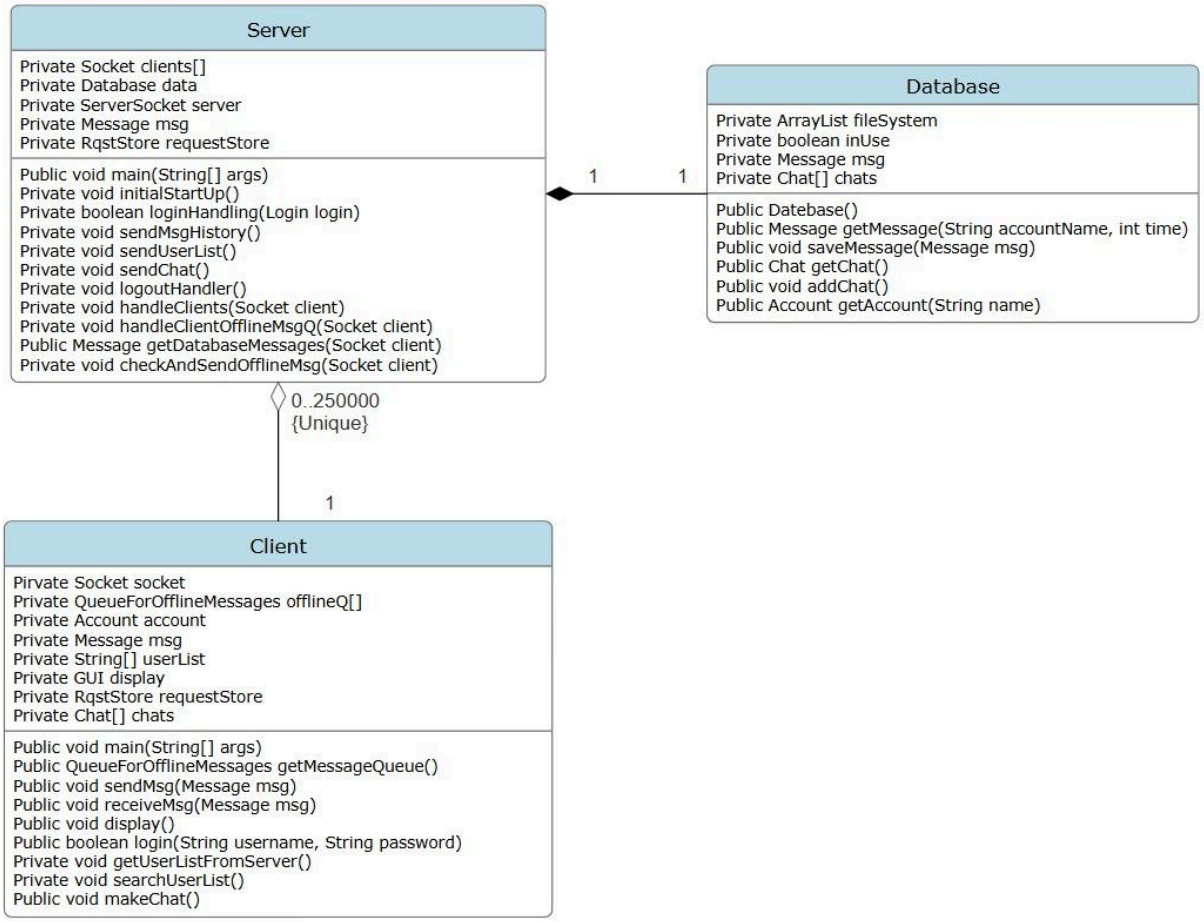
Related Use Cases:
1, 2, 3, 4

UML Class Diagrams

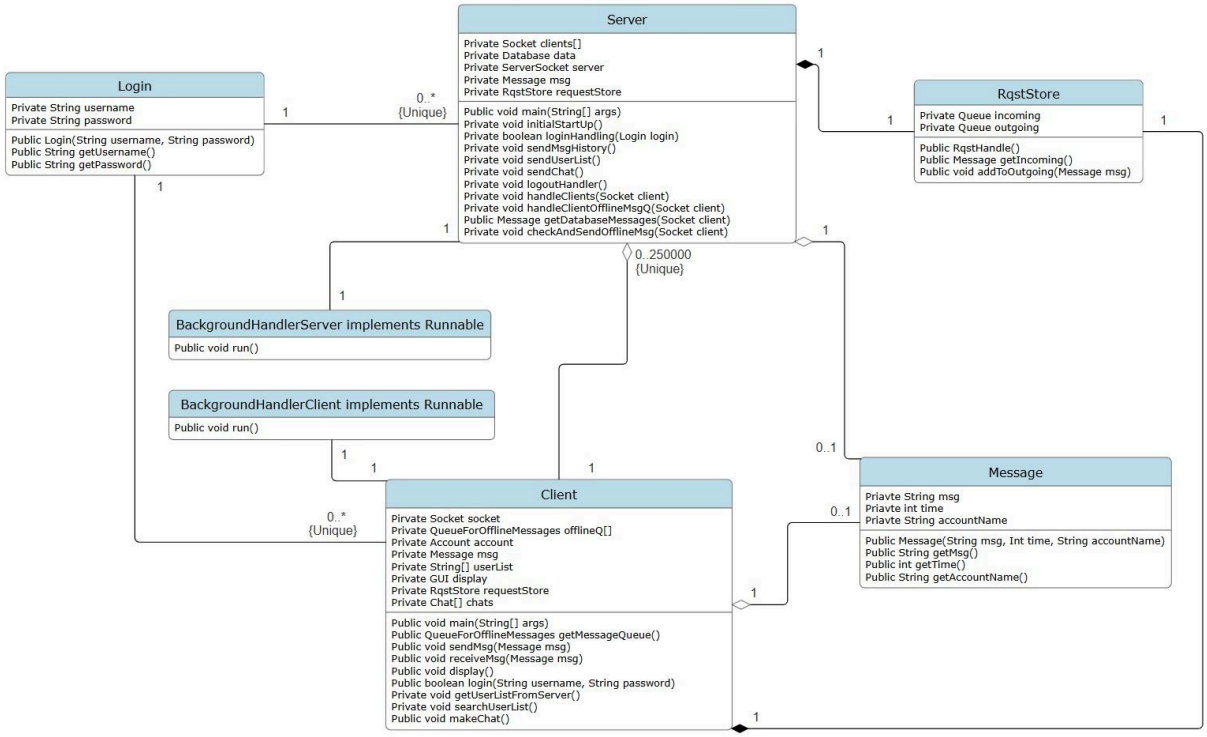
Overview:



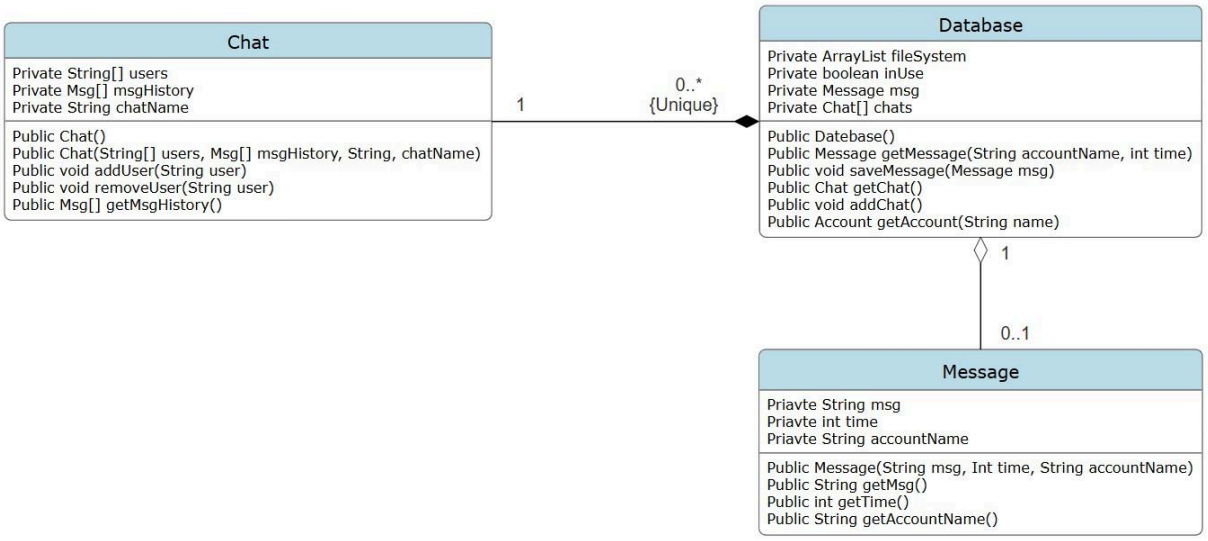
Core Client-Server-Database:



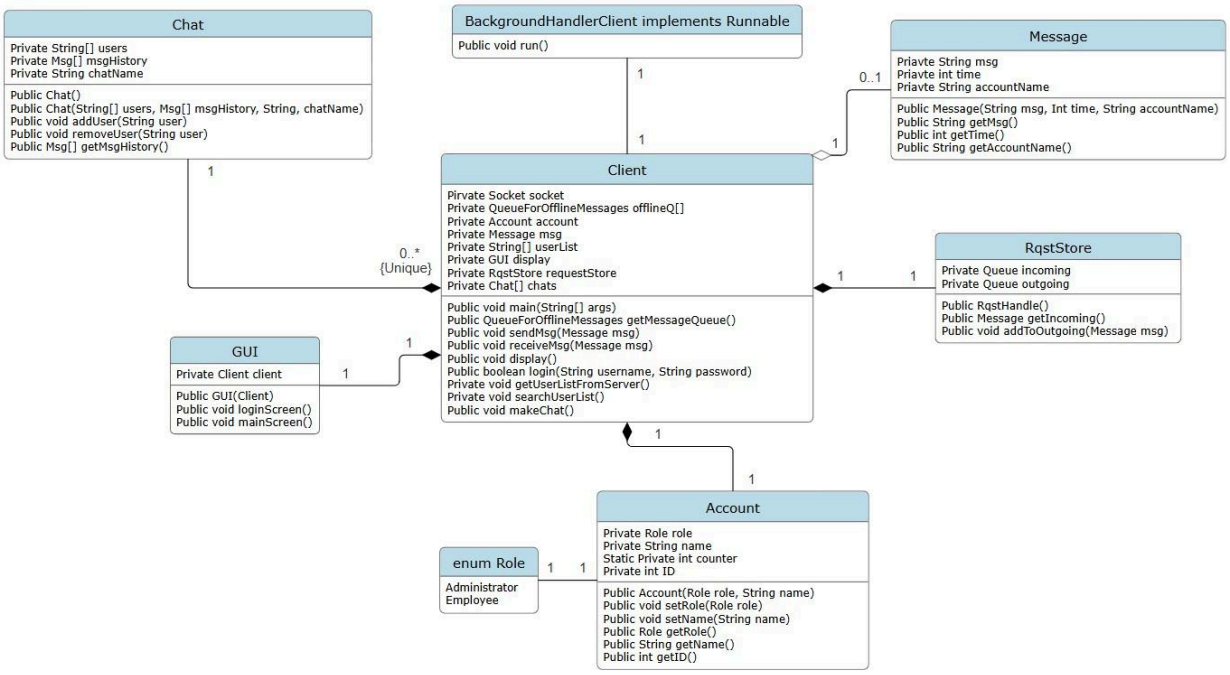
Client-Server:



Database:

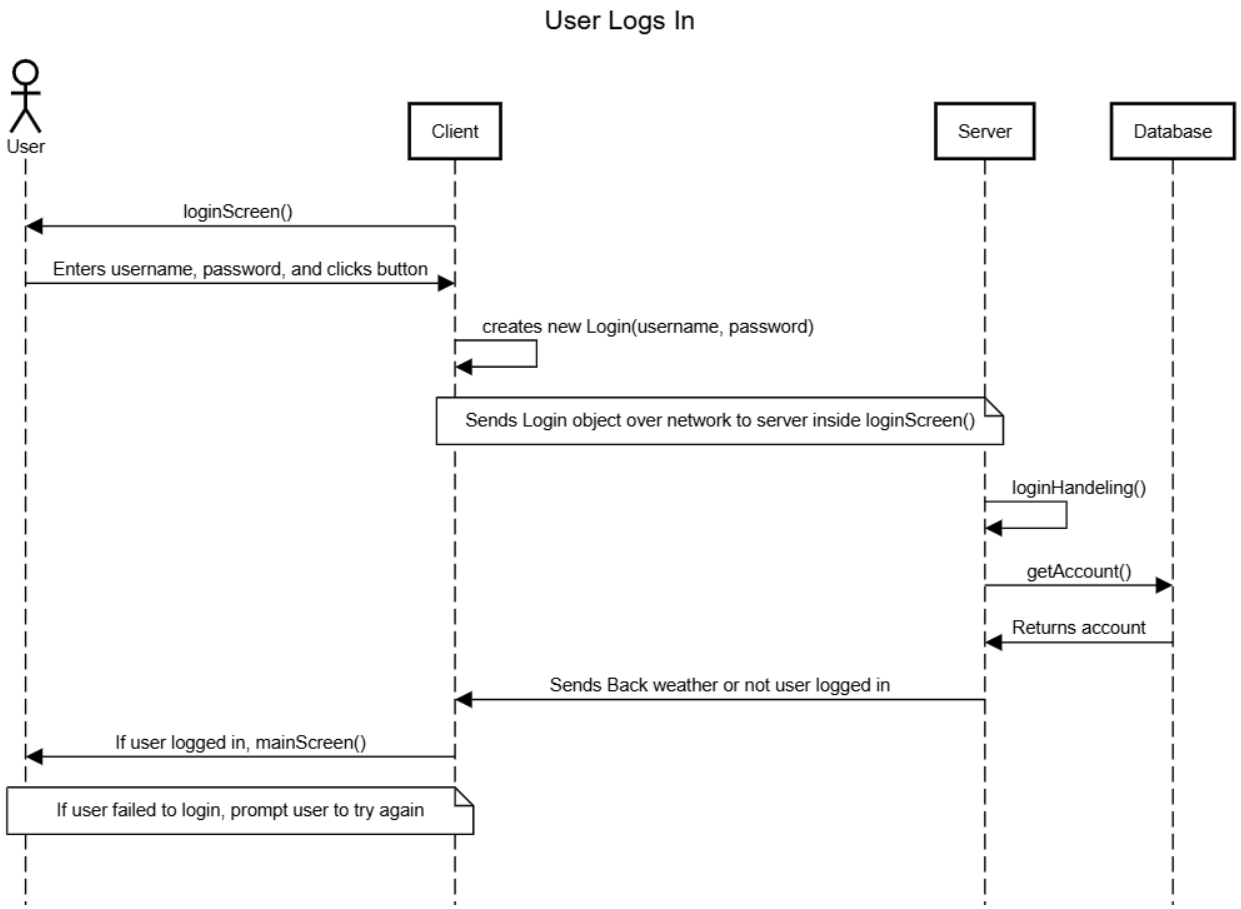

Chat
Private String[] users
Private Msg[] msgHistory
Private String chatName
Public Chat()
Public Chat(String[] users, Msg[] msgHistory, String, chatName)
Public void addUser(String user)
Public void removeUser(String user)
Public Msg[] getMsgHistory()
1
0..*
{Unique}
Database
Private ArrayList fileSystem
Private boolean inUse
Private Message msg
Private Chat[] chats
Public Datebase()
Public Message getMessage(String accountName, int time)
Public void saveMessage(Message msg)
Public Chat getChat()
Public void addChat()
Public Account getAccount(String name)
1
0..1
Message
Priavte String msg
Priavte int time
Priavte String accountName
Public Message(String msg, Int time, String accountName)
Public String getMsg()
Public int getTime()
Public String getAccountName()

Client:

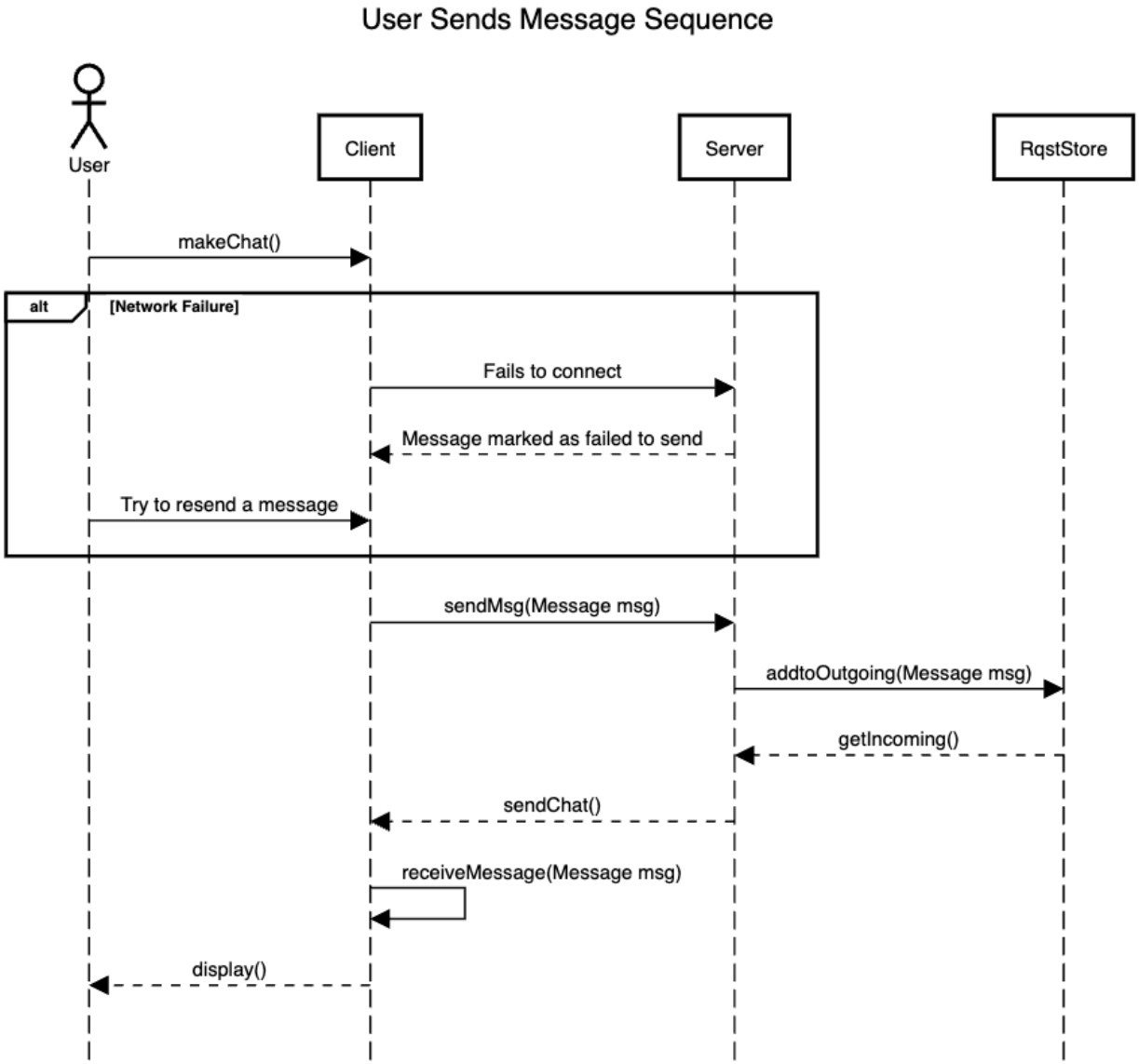

Chat
Private String[] users
Private Msg[] msgHistory
Private String chatName
Public Chat()
Public Chat(String[] users, Msg[] msgHistory, String, chatName)
Public void addUser(String user)
Public void removeUser(String user)
Public Msg[] getMsgHistory()
BackgroundHandlerClient implements Runnable
Public void run()
Message
Priavte String msg
Priavte int time
Priavte String accountName
Public Message(String msg, Int time, String accountName)
Public String getMsg()
Public int getTime()
Public String getAccountName()
Client
Pirvate Socket socket
Private QueueForOfflineMessages offlineQ[]
Private Account account
Private Message msg
Private String[] userList
Private GUI display
Private RqstStore requestStore
Private Chat[] chats
Public void main(String[] args)
Public QueueForOfflineMessages getMessageQueue()
Public void sendMsg(Message msg)
Public void receiveMsg(Message msg)
Public void display()
Public boolean login(String username, String password)
Private void getUserListFromServer()
Private void searchUserList()
Public void makeChat()
RqstStore
Private Queue incoming
Private Queue outgoing
Public RqstHandle()
Public Message getIncoming()
Public void addToOutgoing(Message msg)
GUI
Private Client client
Public GUI(Client)
Public void loginScreen()
Public void mainScreen()
Account
Private Role role
Private String name
Static Private int counter
Private int ID
Public Account(Role role, String name)
Public void setRole(Role role)
Public void setName(String name)
Public Role getRole()
Public String getName()
Public int getID()
enum Role
Administrator
Employee
1
0..1
0..*
{Unique}

UML Sequence Diagrams

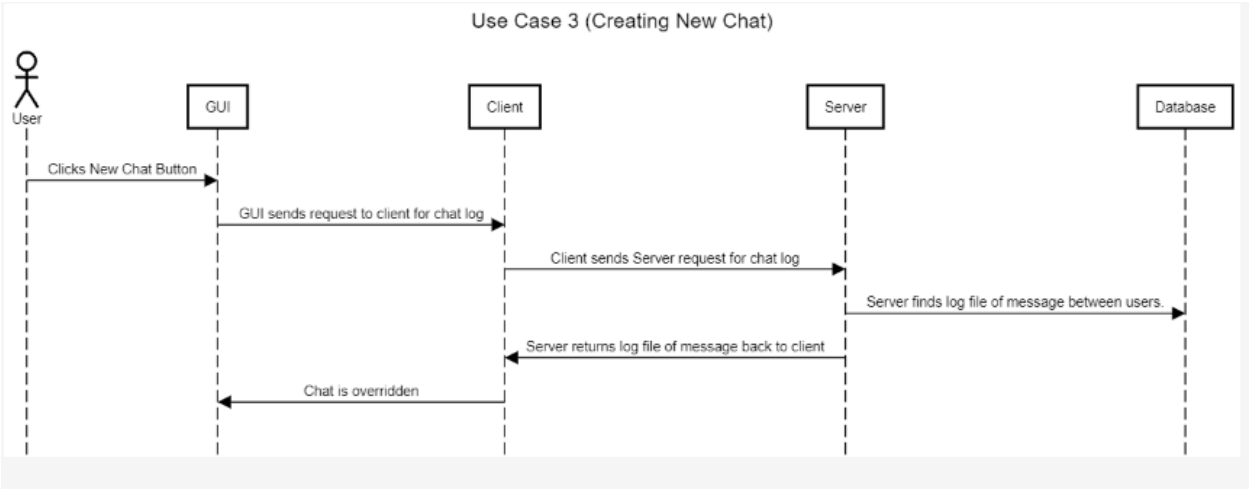
Use Case 1: User Logs In



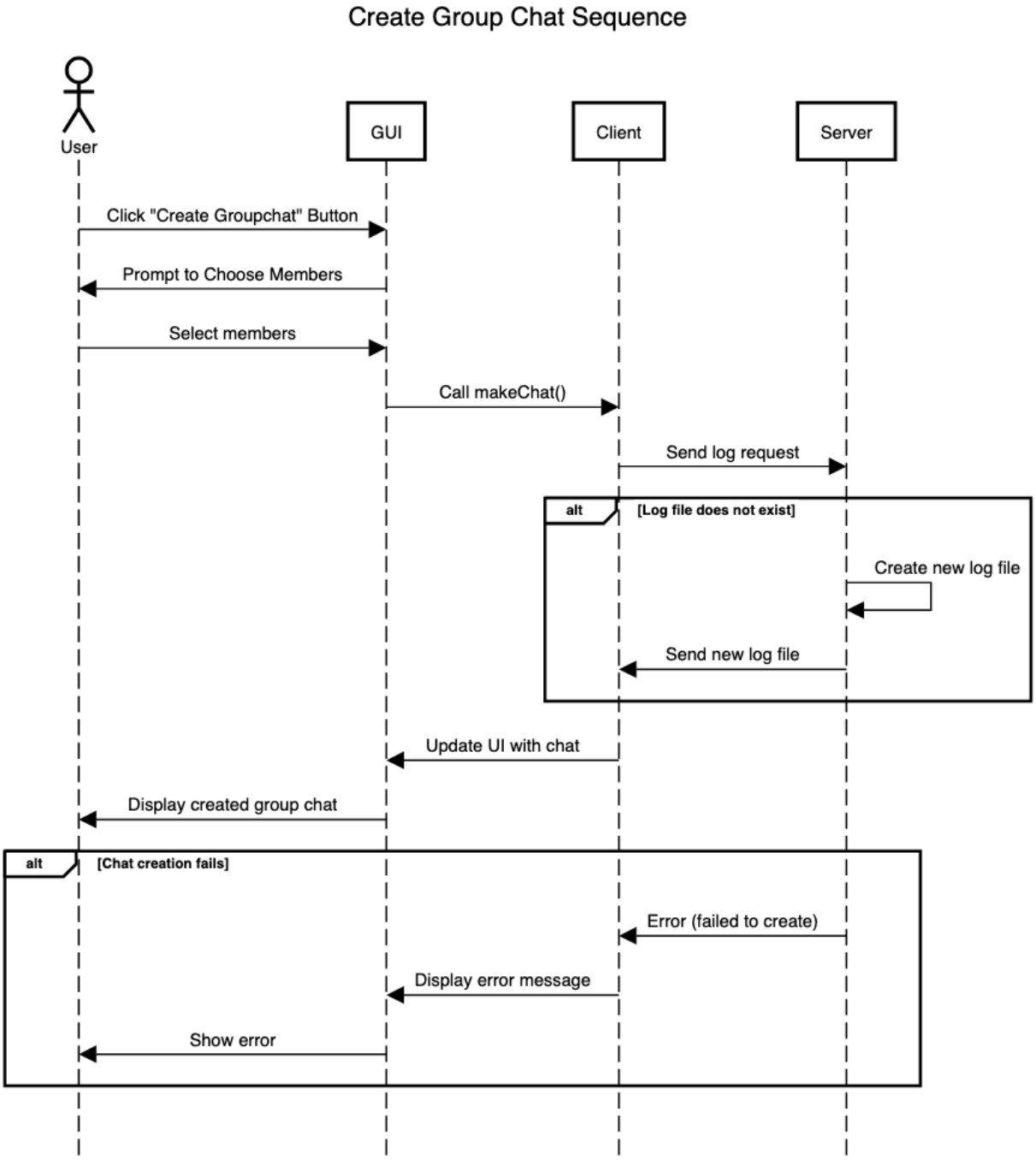
Use Case 2: Sending a Message



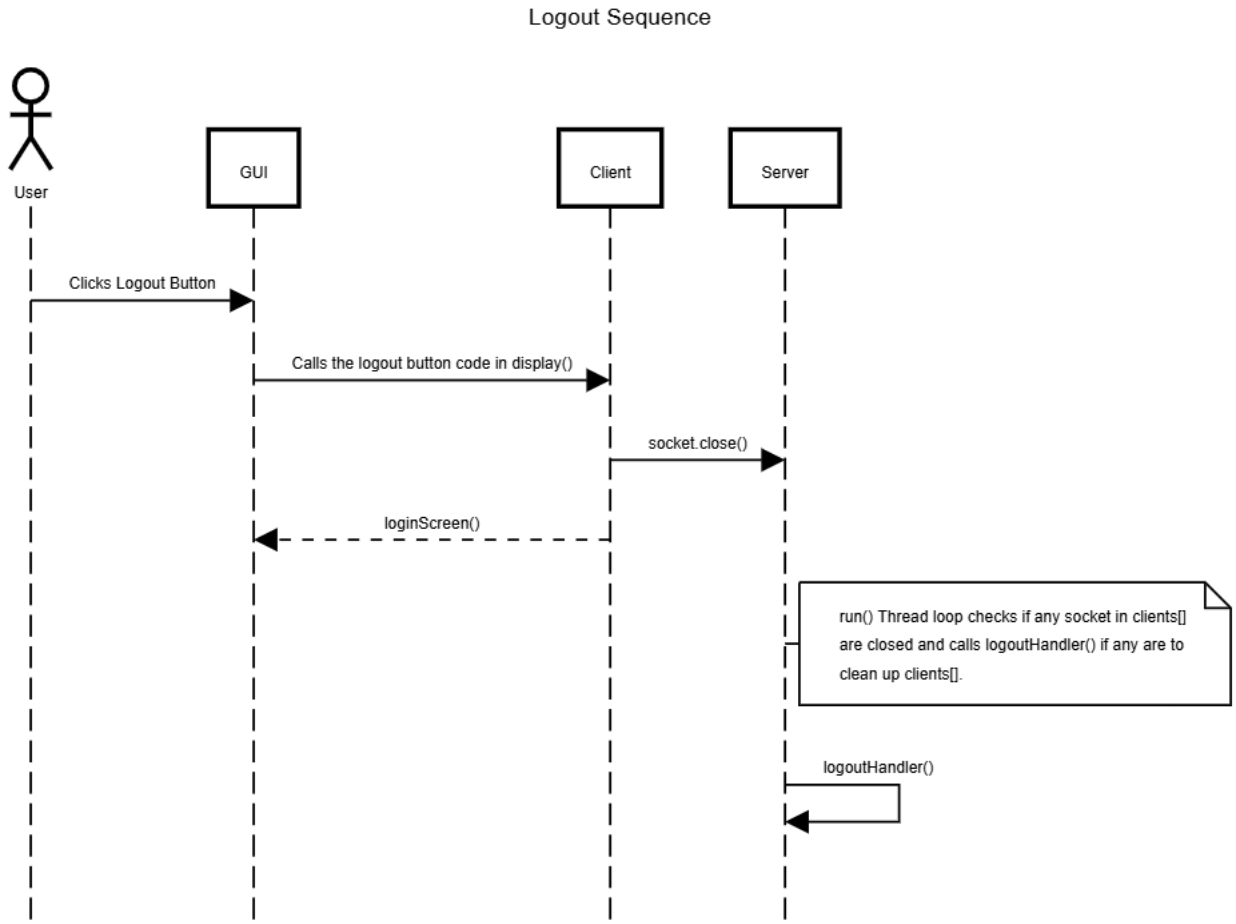
Use Case 3: Opening New Chat



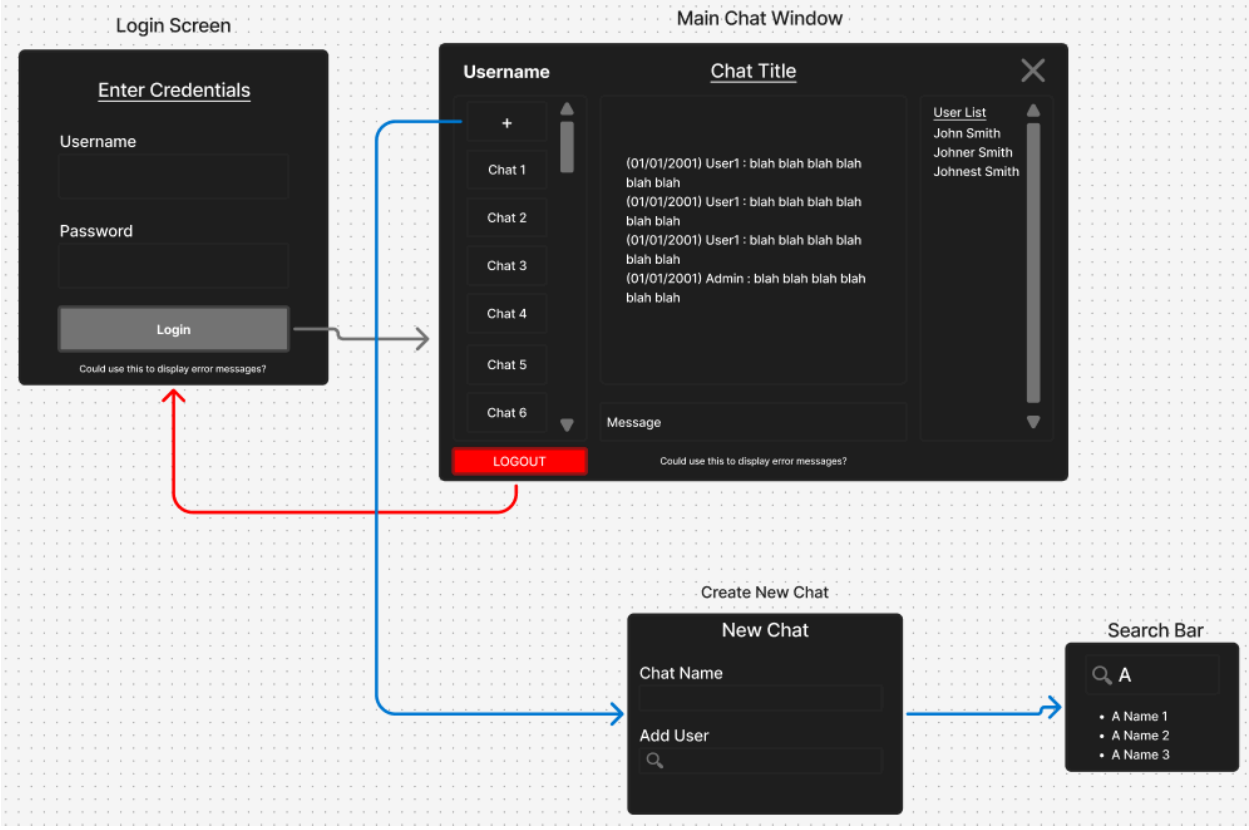
Use Case 4: User Creates Chat with Other Users



Use Case 5: User Logs Out

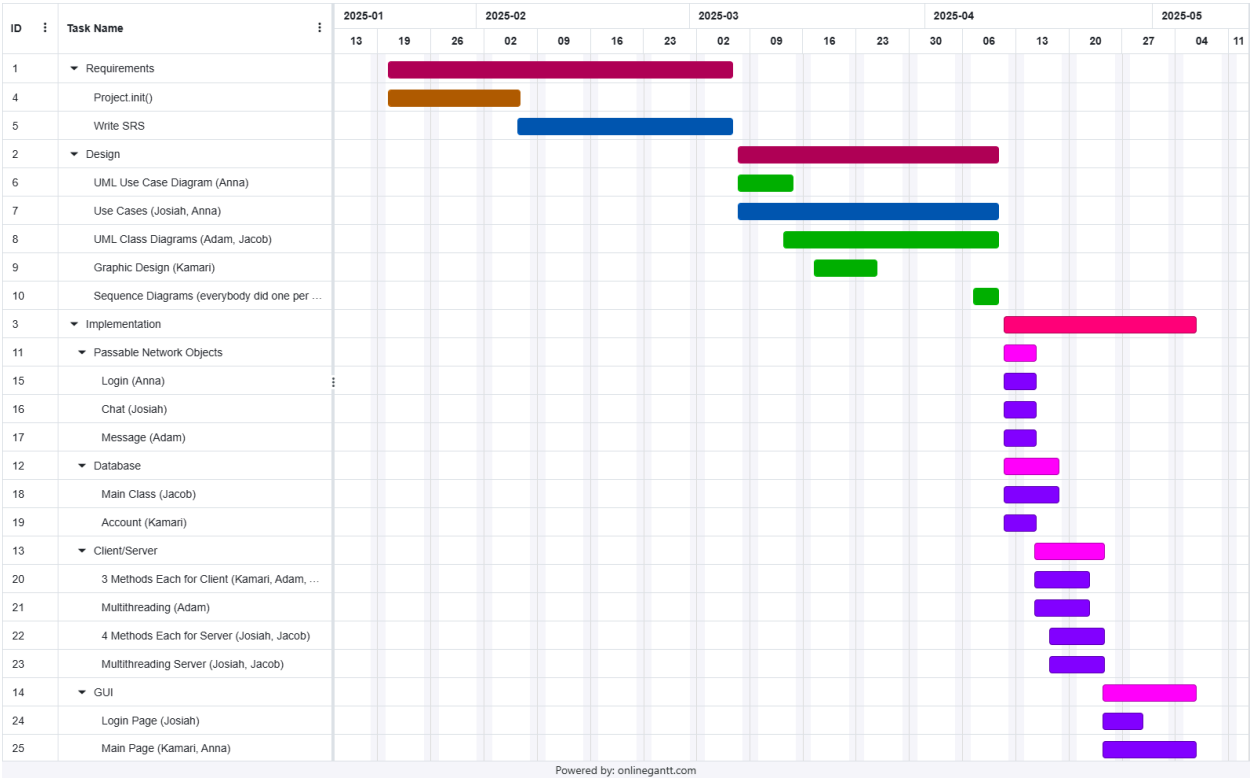


GUI Concept Diagram



Figma Link:
https://www.figma.com/board/9q77PtpLaQqwhea6qjYmXv/GUI-Concept?node-id=0-1&t=MMfCgX20RItdWPO8-1

Gantt Chart


ID
Task Name
2025-01
2025-02
2025-03
2025-04
2025-05
13 19 26 02 09 16 23 02 09 16 23 30 06 13 20 27 04 11
1 Requirements
4 Project.init()
5 Write SRS
2 Design
6 UML Use Case Diagram (Anna)
7 Use Cases (Josiah, Anna)
8 UML Class Diagrams (Adam, Jacob)
9 Graphic Design (Kamari)
10 Sequence Diagrams (everybody did one per ...
3 Implementation
11 Passable Network Objects
15 Login (Anna)
16 Chat (Josiah)
17 Message (Adam)
12 Database
18 Main Class (Jacob)
19 Account (Kamari)
13 Client/Server
20 3 Methods Each for Client (Kamari, Adam, ...
21 Multithreading (Adam)
22 4 Methods Each for Server (Josiah, Jacob)
23 Multithreading Server (Josiah, Jacob)
14 GUI
24 Login Page (Josiah)
25 Main Page (Kamari, Anna)
Powered by: onlinegantt.com