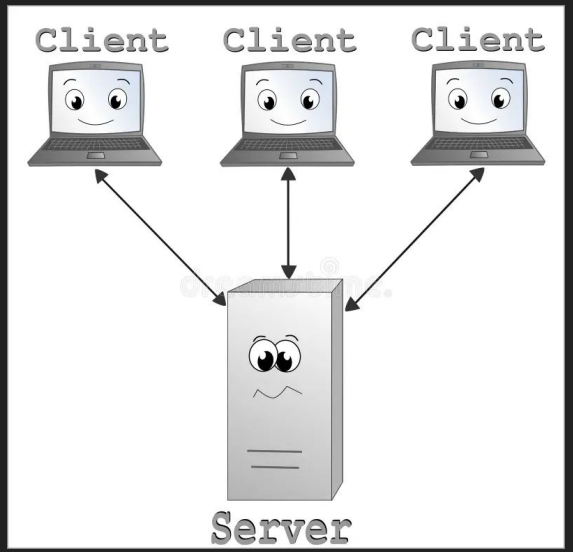
Group 7
Communication_System.phase1()

String Names [ ] = {“Isaiah”, “Joseph”, “Marcos”, “Roman”, “Salvador”}

Requirements 1 & 2

1. The server must be able to accept and process connections from multiple clients.
2. The client and server must maintain both an inbound and outbound connection with each other.



Requirements 3 & 4

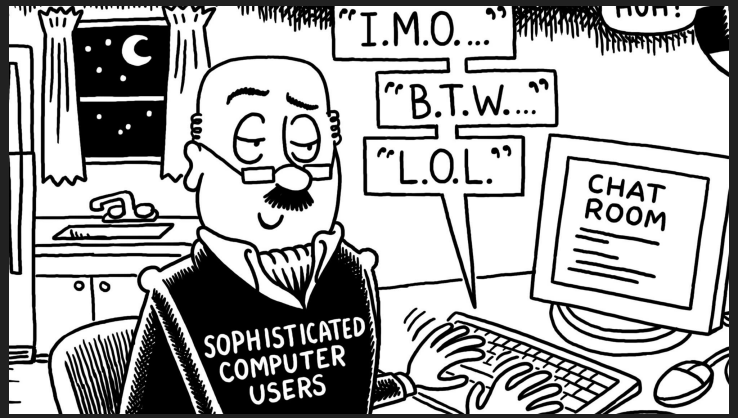
3. Users can send text-based messages to each other synchronously such that all users receive the messages in order and not at different times.
4. Users can send text-based messages to each other asynchronously such that offline users can receive and read messages they are involved in the next time they log into the system.

Requirements 5 & 6

5. Every message sent in the system is to be logged and timestamped in a structured format.
6. The system should be able to store and retrieve the account information for all users of the system.

Requirements 7 & 8

7. Administrators of the system must be able to access and read all stored message logs.
8. Users should be able to create their own chat rooms and invite other users to join.

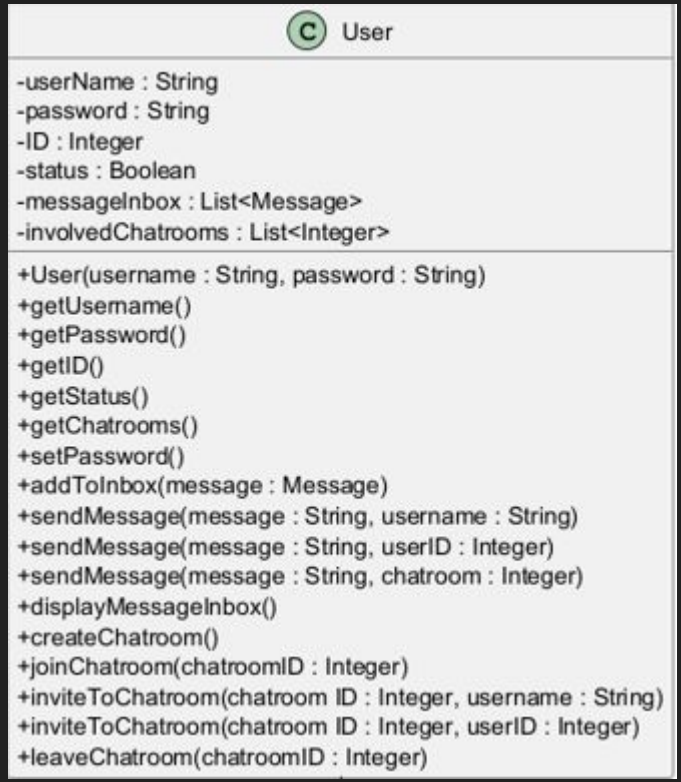


Requirements 9 & 10

9. Users should have to authenticate themselves with the system before being able to use any functionality.
10. The system should be accessible via a GUI interface such that first-time users should report minimal confusion navigating the system.

Class: User

- Contains the information for each client. This is like their log in credentials, their message inbox, the list of chatrooms they are involved in.
- Has the methods for the client to interact with the server (send messages, create chatrooms, etc).
- There are 2 users: User & Admin



Class: Server

- Handles requests from the client
- The client sends requests to the server, the server processes each request by handing it off to the appropriate handler, then responds to each client

Class: UserManager

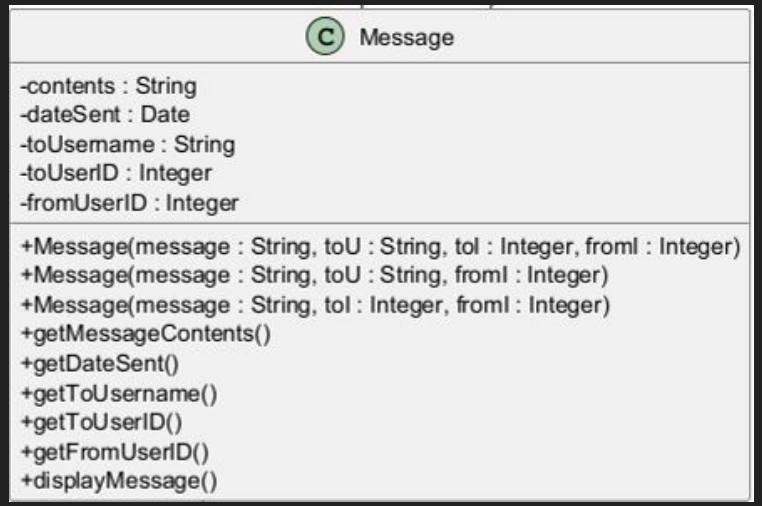
- Handles all the actions that involve the user object.
- Retrieving & storing user data, passing a message to a user's inbox, and authenticating the user

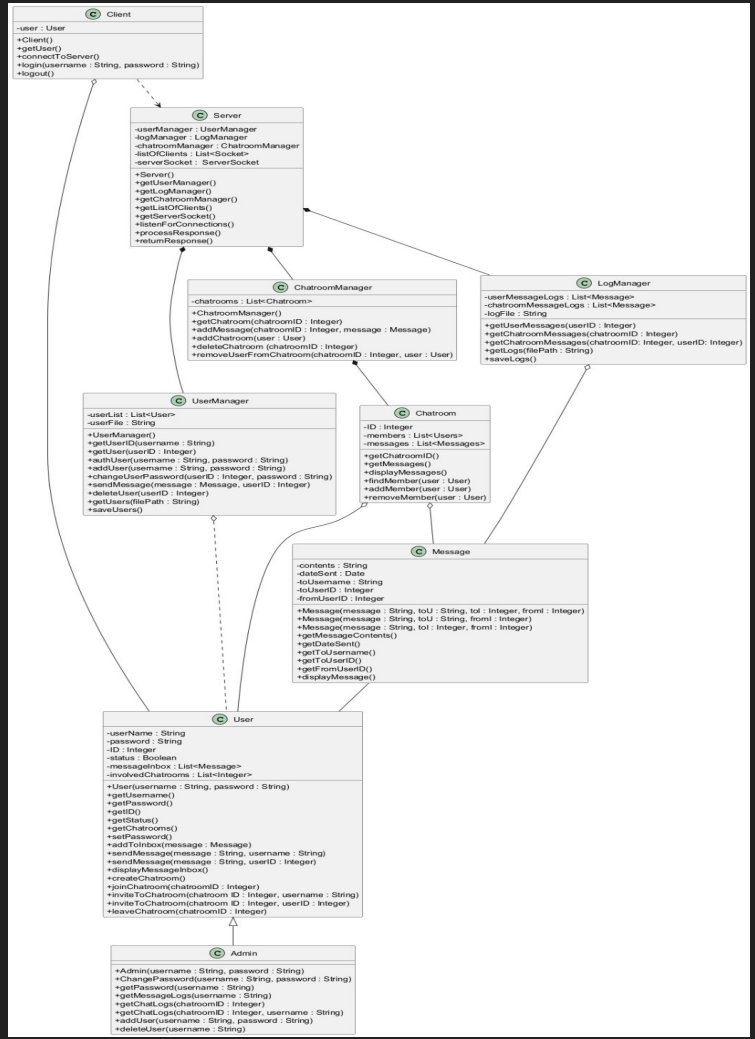
Class: LogManager

- Handles all logging functionalities needed in the project.
- Everytime a message is sent it is forwarded to the LogManager for logging. The LogManager holds and organizes these lists of logs for easy retrieval.

Class: Message

When the user sends a message we would need to keep track of the recipient, the sender, the contents, and the date sent, so we encapsulated that into its own object for better management, flexibility, and maintainability




Client
-user : User
+Client()
+getUser()
+connectToServer()
+login(username : String, password : String)
+logout()
Server
-userManager : UserManager
-logManager : LogManager
-chatroomManager : ChatroomManager
-listOfClients : List<Socket>
-serverSocket : ServerSocket
+Server()
+getUserManager()
+getLogManager()
+getChatroomManager()
+getListOfClients()
+getServerSocket()
+listenForConnections()
+processResponse()
+returnResponse()
ChatroomManager
-chatrooms : List<Chatroom>
+ChatroomManager()
+getChatroom(chatroomID : Integer)
+addMessage(chatroomID : Integer, message : Message)
+addChatroom(user : User)
+deleteChatroom (chatroomID : Integer)
+removeUserFromChatroom(chatroomID : Integer, user : User)
LogManager
-userMessageLogs : List<Message>
-chatroomMessageLogs : List<Message>
-logFile : String
+getUserMessages(userID : Integer)
+getChatroomMessages(chatroomID : Integer)
+getChatroomMessages(chatroomID: Integer, userID: Integer)
+getLogs(filePath : String)
+saveLogs()
UserManager
-userList : List<User>
-userFile : String
+UserManager()
+getUserID(username : String)
+getUser(userID : Integer)
+authUser(username : String, password : String)
+addUser(username : String, password : String)
+changeUserPassword(userID : Integer, password : String)
+sendMessage(message : Message, userID : Integer)
+deleteUser(userID : Integer)
+getUsers(filePath : String)
+saveUsers()
Chatroom
-ID : Integer
-members : List<Users>
-messages : List<Messages>
+getChatroomID()
+getMessages()
+displayMessages()
+findMember(user : User)
+addMember(user : User)
+removeMember(user : User)
Message
-contents : String
-dateSent : Date
-toUsername : String
-toUserID : Integer
-fromUserID : Integer
+Message(message : String, toU : String, toI : Integer, fromI : Integer)
+Message(message : String, toU : String, fromI : Integer)
+Message(message : String, toI : Integer, fromI : Integer)
+getMessageContents()
+getDateSent()
+getToUsername()
+getToUserID()
+getFromUserID()
+displayMessage()
User
-userName : String
-password : String
-ID : Integer
-status : Boolean
-messageInbox : List<Message>
-involvedChatrooms : List<Integer>
+User(username : String, password : String)
+getUsername()
+getPassword()
+getID()
+getStatus()
+getChatrooms()
+setPassword()
+addToInbox(message : Message)
+sendMessage(message : String, username : String)
+sendMessage(message : String, userID : Integer)
+displayMessageInbox()
+createChatroom()
+joinChatroom(chatroomID : Integer)
+inviteToChatroom(chatroom ID : Integer, username : String)
+inviteToChatroom(chatroom ID : Integer, userID : Integer)
+leaveChatroom(chatroomID : Integer)
Admin
+Admin(username : String, password : String)
+ChangePassword(username : String, password : String)
+getPassword(username : String)
+getMessageLogs(username : String)
+getChatLogs(chatroomID : Integer)
+getChatLogs(chatroomID : Integer, username : String)
+addUser(username : String, password : String)
+deleteUser(username : String)