



CHATAPP PROJECT

Technical Document

Abstract

This project is intended to be read for continuous development. It is a raw project developed by a single person, Owen Lianggara, over the course of one month.

Owen Lianggara
Email: owen@gmail.com

Abstraction

The purpose of this application development is to explore new insights into how other major chat applications, such as WhatsApp, Telegram, and others, operate with certain architectures, algorithms, and data structures. Additionally, ChatApp is available to the public for learning about chat application development components, and its source code is open-source for further development by other parties. This document contains API documentation, system architecture, code structure, and configuration and installation guides. In addition, this project ignores front-end development and focuses on back-end development.

Introduction

From the Author

My name is Owen Lianggara, and this is my first project. After one month working on this project, I see that this project is not even close to perfect, so I openly accept criticism and suggestion. Please send the email to us via the email address that has been listed on the cover. I hope this project can help others who want to learn something new about application development. Thank you.

Project Background

This project was created out of a strong desire to learn practical activities that will be useful when entering the world of informatics work, such as application development, server maintenance and development, and even Artificial Intelligence (A.I.) development. This is the first step, and the ChatApp has been chosen as the first project. This project is not only for us; it is open source, allowing everyone to view, learn from, and contribute to the development of our project for educational purposes.

In this documentation, we will discuss:

- Back-end development
- APIs used in this project
- System structure
- Code structure
- Configuration and installation guides

We will not cover:

- Front-end development
- Data design
- Program design

Development Environment

During the development of this application, we use several tools to assist in building it.

1. IDE: Apache NetBeans IDE 23
2. API: Swing
3. Java: 17.0.12 (OpenJDK 64-Bit Server VM 17.0.12+7)

System Overview

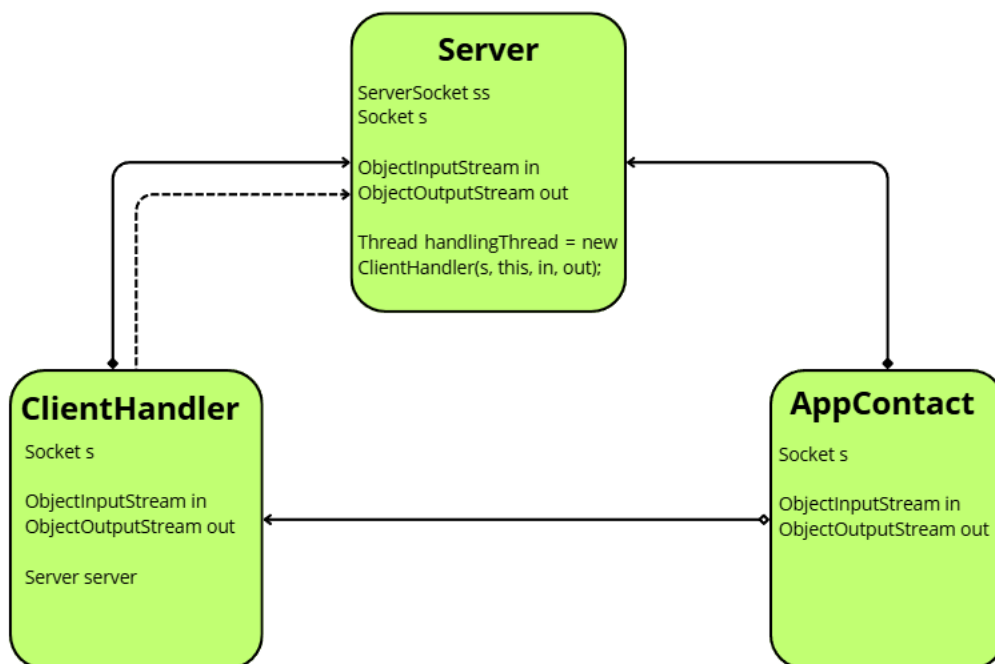
ChatApp is a chatting application designed for educational purposes to demonstrate system integration and how it operates on a Client-Server architecture. ChatApp focuses on several main features, namely:

1. Personal chat
2. Adding friends
3. Group chat

System Architecture

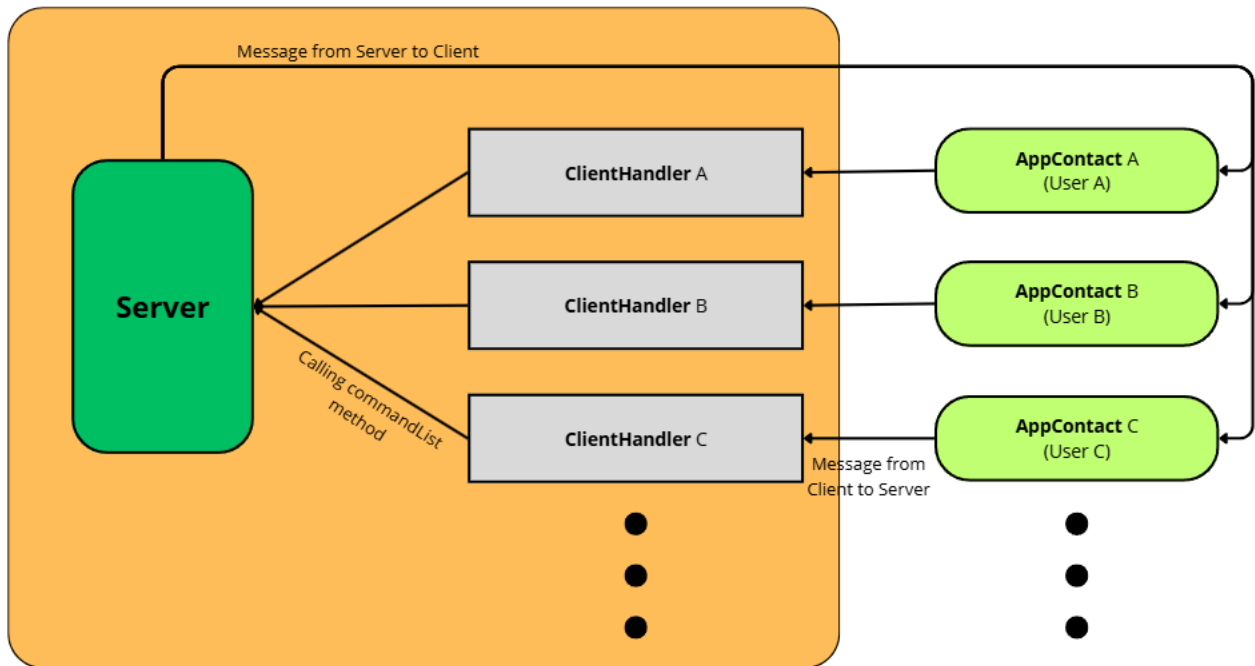
1. Client-Server Explanation

There are three main classes that keep the application running and enable communication for users: Server, ClientHandler, and AppContact (the client), as illustrated in the picture below.



Firstly, the Server needs to be executed while the ClientHandler waits for the AppContact to connect. Once the AppContact is connected, the ClientHandler reads messages from the AppContact and executes the corresponding commands using methods from the Server. During this process, some methods need to interact with the database, specifically UserData and GroupData, on the Server. After all commands are executed, if necessary, the Server will send messages to the AppContact using the `getOut()` method at the User class.

The illustration of the workflow for this system is shown below.



The orange block indicates that ClientHandler can be part of the Server. So, if we put the ClientHandler code inside the Server (meaning that the ClientHandler class does not exist), the system can still function normally as a Client-Server architecture.

Code Structure

1. User

To store the user data listed below, a User class is created.

- **Attributes**

Variable's Name	Access Modifier	Data Type	Key	Value	Description
Username	private	String	-	-	Serves as a unique identity to distinguishes each user.
Password	private	String	-	-	Used for login authentication and security purposes.
online	private	Boolean	-	-	Used by server to Indicates whether the current user is online or offline.
friendRequest	private	TreeSet	String	-	Stores all friend requests received by the current user.
friends	private	HashMap	String	String	Stores the list of friends mutually added by the user.
groups	private	HashMap	String	String	Stores all groups the current user has joined.

out	private	ObjectOutputStream	-	-	Stream to write a message from the Server to the User.
-----	---------	--------------------	---	---	--

- **Methods**

For an explanation of all the methods, each variable listed above has a getter and setter to fulfill the principles of OOP (Object Oriented Programming) principle.

2. Group

The Group class is used to store group information, but it does not include chat history, as that is maintained in the User class.

- **Attributes**

Variable's Name	Access Modifier	Data Type	Key	Description
leader	private	String	-	For some security reasons, user as a group leader manages information about the group.
groupName	private	String	-	To distinguish a group with other groups, the group name feature must be implemented for user convenience.
Members	private	TreeSet	String	Users who join a certain group will be listed in the Members variable in sorted order to display group members in the group information.

- **Methods**

All variables have getter methods to fulfill the principles of OOP (Object Oriented Programming) principle.

3. ClientHandler

The ClientHandler class is used to maintain communication in a concurrent way, ensuring that each user has a separate object stream so it does not interfere with other users' streams.

- **Attributes**

Variable's Name	Access Modifier	Data Type	Description
-----------------	-----------------	-----------	-------------

s	private	Socket	s is marked as an endpoint of communication between the client and the server. In this class, s will represent the endpoint of the server. When User gets offline, the socket will be closed as s is a reference object.
server	private	Server	The server's function is to call the commandList() method after the ClientHandler receives a message from AppContact.
in	private	ObjectInputStream	ClientHandler will wait for a message from AppContact (or User) to interact with the server's system. When a User gets offline, the in will be closed as s is a reference object.
out	private	ObjectOutputStream	When User gets offline, the out will be closed as s is a reference object.

- **Methods**

getOut() method is called when the server wants to write a message to a certain user AppContact to execute a method in it.

4. Server

This serves as a storage location for data in place of a database and the data will be used for application and user interaction needs.

- **Attributes**

Variable's Name	Access Modifier	Data Type	Key	Value	Description
s	private	Socket	-	-	s is marked as an endpoint of communication between the client and the server. In this

					class, s will represent the endpoint of the server.
ss	private	ServerSocket	-	-	The server socket will accept a client connection.
UserData	private	HashMap	String	User [Object]	UserData saves the username of all users and the User object.
GroupData	private	HashMap	String	Group[Object]	GroupData saves the name of all group and the Group object

- Methods**

Method's Name	Access Modifier	Data Type	Code	Description
commandList	public	void	-	This method executes the AppContact command triggered by the interaction between the user and the GUI.
UserRegistration	public	void	RE	This method adds a new user to the UserData variable in the Server class when the user sends his/her username and password to the server.
UserLogin	public	void	LI	With the condition that the user has registered, this method processes the username and password submitted by user to allow access to the application.
AddFriend	public	void	AF	The server searches for the username submitted by a user to send a friend request if the other

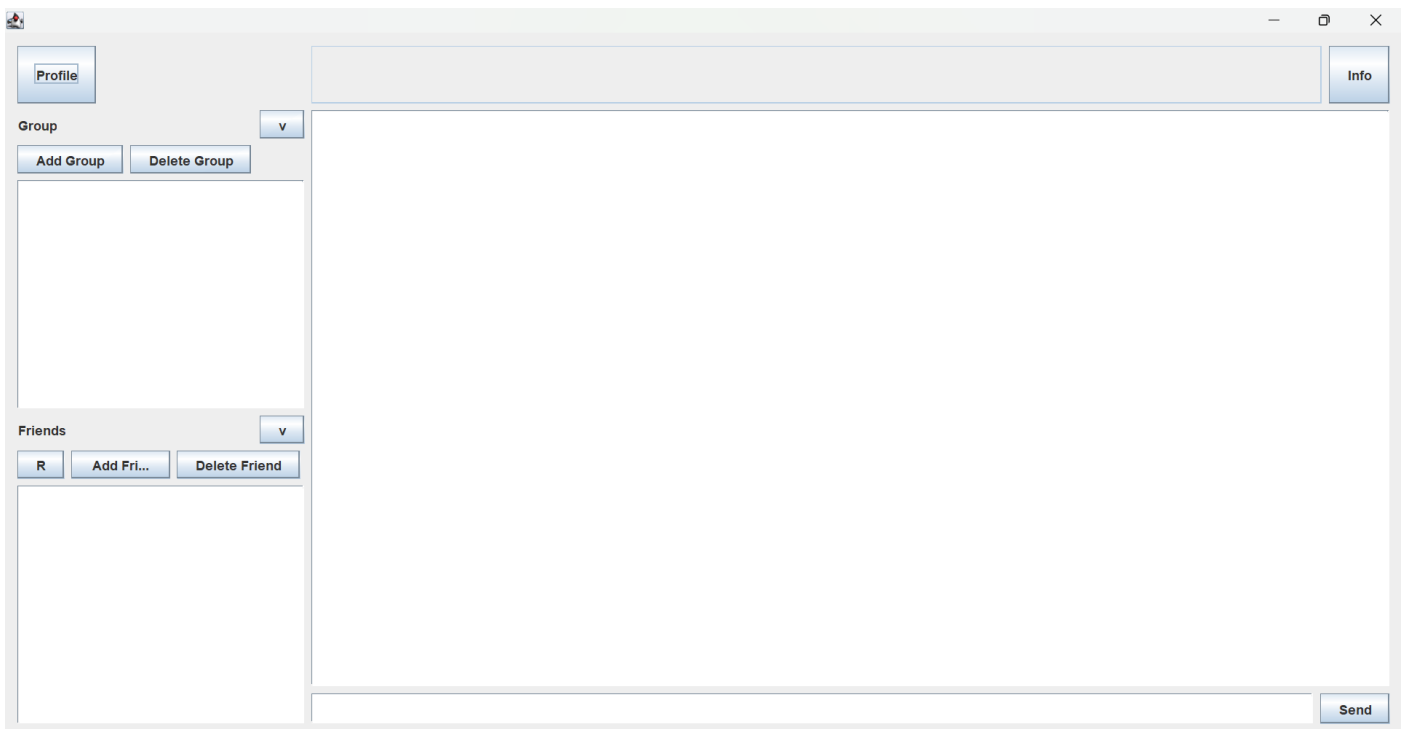
				user has not already sent a request, then the request is stored in the friendRequest variable of the User class. Or, If the user accepts the request, it is moved to the friends variable of the User class and the request is removed since they have become friends.
DeleteFriend	public	void	DF	When a user unfriends someone in their contacts (the friends variable in the User class), the user will be removed from that person's contacts (their friends variable in the User class).
MessageFriend	public	void	MF	The server receives a private message from one user to another and saves the chat history as a value in the friends variable of the User class.
AddGroup	public	void	AG	When a user creates a new group, the server adds it to the groups variable in the User class, which is stored in UserData
GroupMember	public	void	GM	Provide all group members to the user who requests to view the group information.
AddMemberToTheGroup	public	void	AM	A member is added to the group when the leader invites them.
MessageGroup	public	void	MG	The server receives a group message from one user and saves the chat history as a value in the groups variable of the User class.

KickMember	public	void	KM	When a member is kicked from a group, they are removed from the Members variable in the Group class, which means they are no longer in the members list.
DeleteGroup	public	void	DG	When the group is deleted from the GroupData variable, all of its members, including the leader, are removed.
LogOutUser	public	void	LO	The ObjectInputStream, ObjectOutputStream, and the Socket are closed during a user logout.

5. AppContact

For user and database interaction, a simple interface is needed. Therefore, AppContact was created to serve as both the client and the GUI.

- **Frame**



***Note:** For button names that are not clear, the description of the button is displayed below.

- [R]: Friend Requests.
- [Add Fri...]: Add Friend.
- [v]: Friend/Group List Visibility Settings.

- Attributes

Variable's Name	Access Modifier	Data Type	Key	Description
s	private	Socket	-	s is marked as an endpoint of communication between the client and the server. In this class, s will represent the endpoint of the client.
in	private	ObjectInputStream	-	AppContact will wait for a message from the server (or User) to interact with the GUI. When a User gets offline, the in will be closed as s is a reference object.
out	private	ObjectOutputStream	-	When User gets offline, the out will be closed as s is a reference object. This variable is the place where AppContact writes message to the server.
CurrentUser	private	User	-	When user logs in, the CurrentUser fills with that user's data.
CurrentFriendRequestListToJList	private	DefaultListModel	String	This list displays friend requests sent by other users that want to add the user as a friend.
CurrentFriendListToJList	private	DefaultListModel	String	This list displays the user's friends

				and can be used to start a private chat or invite them to a specific group.
CurrentGroupListToJList	private	DefaultListModel	String	
MemberListToJList	private	DefaultListModel	String	
EditContact	private	EditContact (Object)	-	
GroupInfoJDialog	private	GroupInfoJDialog (Object)	-	
FriendRequestJDialog	private	FriendRequestJDialog (Object)	-	

- **Methods**

Method's Name	Access Modifier	Data Type	Code	Description
InfoButtonActionPerformed	private	void	-	When the user presses this button, it displays either a friend's information or a group's information, depending on the selected value in the friends or group list.
ProfileButtonActionPerformed	private	void	-	Provide the user profile.
GroupListDropButtonActionPerformed	private	void	-	Set the visibility of the group list.
FriendListDropButtonActionPerformed	private	void	-	Set the visibility of the friend list
AddFriendButtonActionPerformed	private	void	-	The user inputs the username of the friend they want to request.
DeleteFriendButtonActionPerformed	private	void	-	After selecting a friend from the user's friend list, the friend is removed from both the list and the database on the server.
SendMessageButtonActionPerformed	private	void	-	The message is sent either to a user's friend (private chat) or to a group chat,

				depending on the selected value.
FriendListValueChanged	private	void	-	When the selection changes from the group list to the friend list, the previously selected item in the group list is disabled.
FriendRequestButtonActionPerformed	private	void	-	This method displays all users who have sent friend requests.
DeleteGroupButtonActionPerformed	private	void	-	The selected group will be deleted and it will no longer appear in the group lists of all its members.
AddGroupButtonActionPerformed	private	void	-	The user inputs the group name, and the system adds it to the group list, with the user set as the leader of the group.
GroupListValueChanged	private	void	-	When the selection changes from the friend list to the group list, the previously selected item in the friend list is disabled.
commandList	public	void	-	Provide all messages that interact with the server.
DeleteFriendAfterBeDeleted	public	void	DF	After a user is deleted by another user, both users are automatically removed from each other's friend lists.
EditContactSetting	public	void	EC	Disposes the EditContact frame for the user.

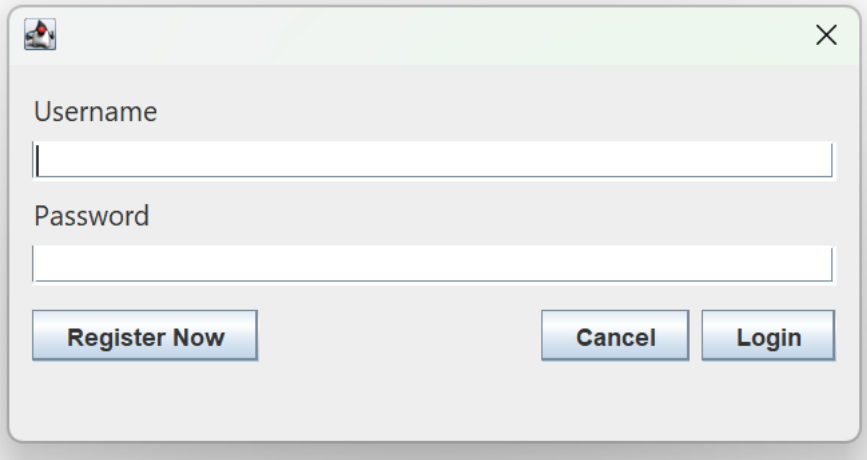
FriendRequestMessageAdd	public	void	FRA	After accepting a friend request from another user, both become friends and can start a private chat or be invited to a group.
FriendRequestMessage	public	void	FR	Send a friend request to another user after entering their username.
DeleteMemberOnTheJDialogList	public	void	DM	After a user kicks a member, the member is removed from the group information displayed on the user's GUI.
NotLeaderKickWarning	public	void	W	When a member who is not the leader of the group attempts to kick another user, a warning JDialog appears for that member
GetKicked	public	void	KM	When a member is kicked or the group is removed by the group leader, they are removed from the group on the server and from their group list.
ClearMember	public	void	CM	Before displaying changes in the group member list, the MemberListToJList variable needs to be cleared.
BeNewMemberOnGroup	public	void	NM	When a user is invited to a group, the new group is added to both the group list and the user's groups variable.
MemberListSetToJDialog	public	void	GM	Display the current members

				of the group in the group information when the InfoButton is pressed.
addFriendToList	public	void	AF	The new friend is added to the friend list in alphabetical order.
addGroupToList	public	void	AG	The new group is added to the group list in alphabetical order.
MessageMeFromFriend	public	void	MM	Messages sent by a user's friend are displayed after the friend is selected from the friend list.
MessageMeFromGroup	public	void	MG	Messages sent by users in a group are displayed after the group is selected from the group list.

6. LoginJDialog

Login is the process of identifying and verification users to a system or application for security and identification purposes.

- Frame**



- Attributes**

Variable's Name	Access Modifier	Data Type	Description
CurrentUser	private	User (Object)	This variable will retrieve the correct user data from the

			server after the login process.
in	private	ObjectInputStream	Receive the message from the server.
out	private	ObjectOutputStream	Send a message to the server.

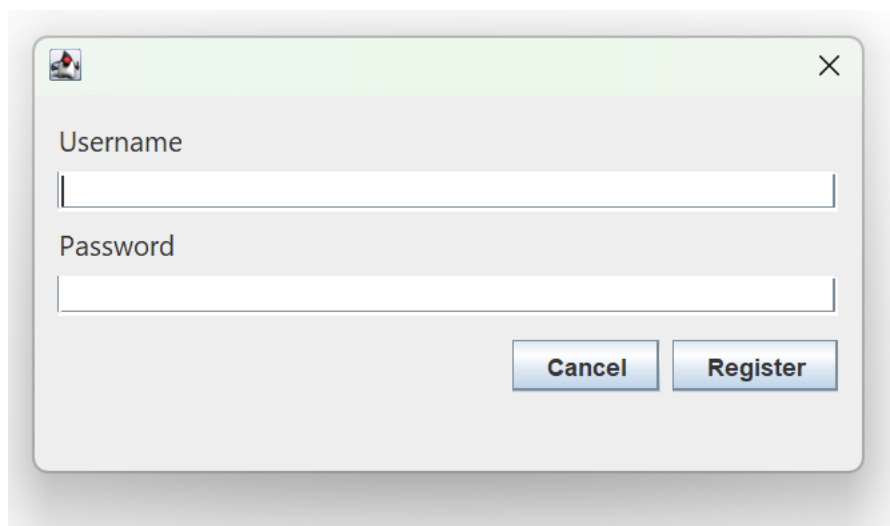
- **Methods**

Method's Name	Access Modifier	Data Type	Description
getCurrentUser	public	User	Getter for the CurrentUser variable.
LoginButtonActionPerformed	private	void	Verify whether the user is allowed to access the application.
CancelButtonActionPerformed	private	void	Cancel the login process and exit from the application.
RegisterButtonActionPerformed	private	void	Show the Registration Frame (RegisterJDialog).

7. RegisterJDialog

Handles the registration process for a new user, regardless of whether they already have an account. The account will be used to access the application through the login process.

- **Frame**



- **Attributes**

Variable's Name	Access Modifier	Data Type	Description
out	private	ObjectOutputStream	Send a message to the server.

Username	private	String	Username of the new user.
Password	private	String	Password of the new user.

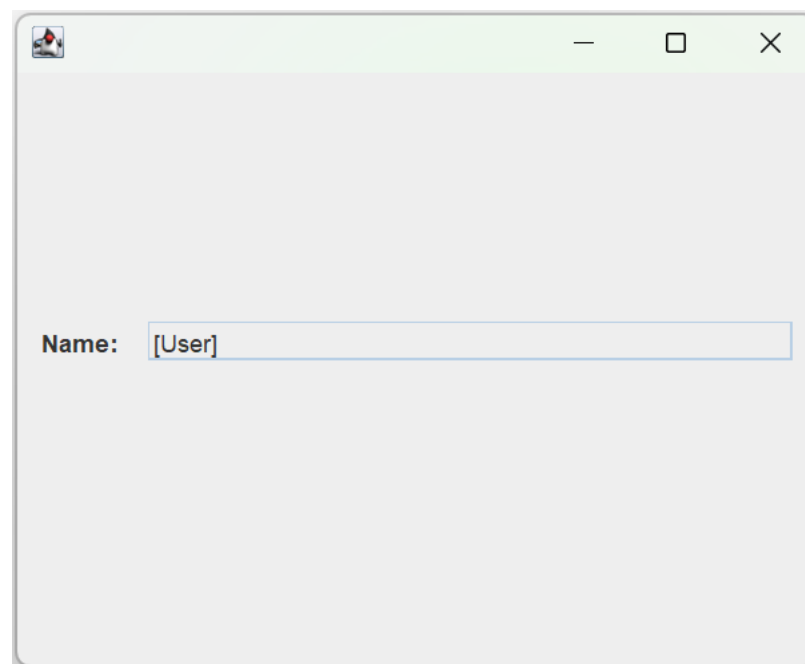
- **Methods**

Method's Name	Access Modifier	Data Type	Description
RegisterButtonActionPerformed	private	void	Register the new user. The message is sent from this frame to the server.
CancelButtonActionPerformed	private	void	Cancel the registration process and return to the login frame (LoginJDialog).

8. ProfileFrame

Displays the user's information (currently, only the username is shown).

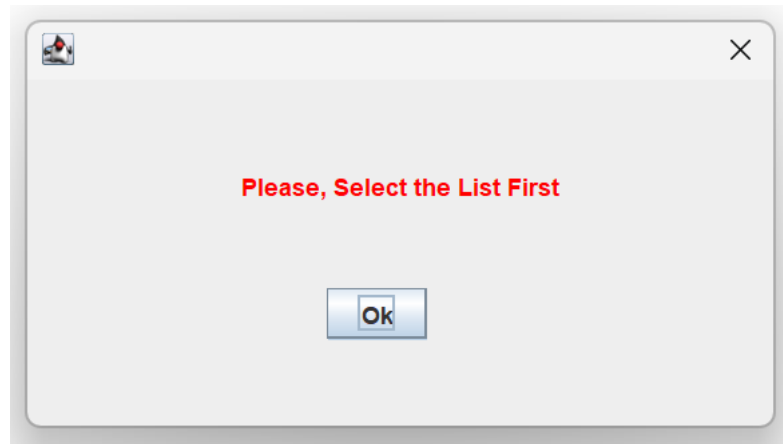
- **Frame**



9. WarningJDialog

When a user enters incorrect data, either in the application or to the server, the warning frame (WarningJDialog class) will appear to notify the user.

- **Frame**



***Note:** The red text above can be any sentence according to the warning received

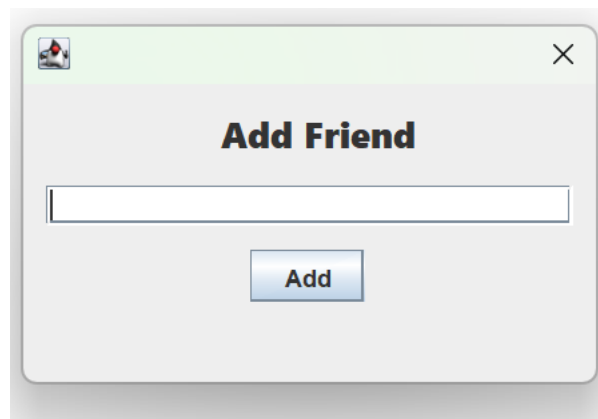
- **Methods**

Method's Name	Access Modifier	Data Type	Description
getWarningLabel	public	JLabel	Getter for the WarningLabel variable.
OkButtonActionPerformed	private	void	Closes the warning frame (WarningJDialog).

10. EditContact

To communicate with other users privately or in a group, the user needs to send a friend request.

- **Frame**



- **Attributes**

Variable's Name	Access Modifier	Data Type	Description
out	private	ObjectOutputStream	Send a message to the server.
Username	private	String	The username of the current user whom this user can add as a friend.

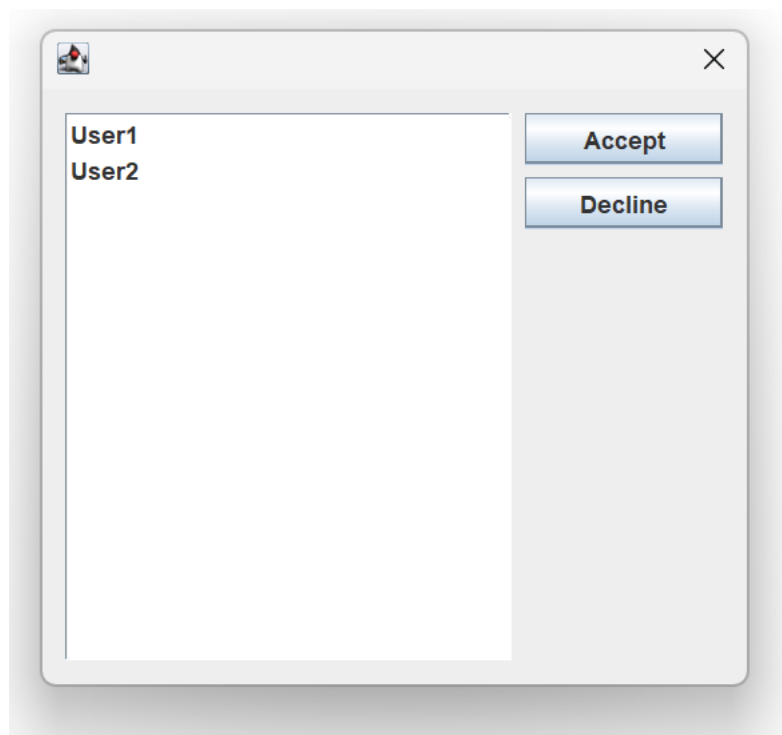
- **Methods**

Method's Name	Access Modifier	Data Type	Description
getAddButton	public	JButton	Getter for the AddButton variable.
getUsernameField	public	TextField	Getter for the UsernameField variable.
getWarningLabel	public	JLabel	Getter for the WarningLabel variable.
setWarningLabel	public	void	Setter for the WarningLabel variable.
AddButtonActionPerformed	private	void	Send the friend's username to the server to check its availability.

11. FriendRequestJDialog

After a user sends a friend request, it will be displayed to the other user's friend request frame, who is the target of the request.

- Frame**



- Attributes**

Variable's Name	Access Modifier	Data Type	Description
out	private	ObjectOutputStream	Send a message to the server.
currentUser	private	String	This variable is needed to identify

			who the friend request is sent from.
--	--	--	--------------------------------------

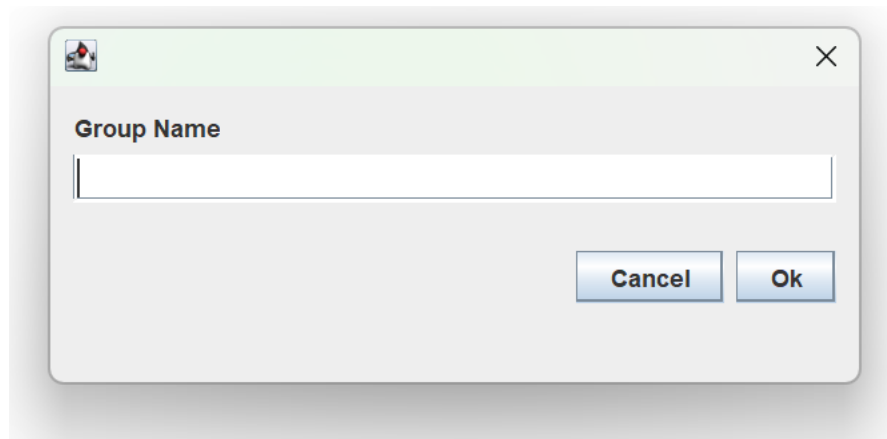
- **Methods**

Method's Name	Access Modifier	Data Type	Key	Description
getRequestFriendJList	public	JList	String	Getter for the FriendRequestJList variable.
AcceptButtonActionPerformed	private	void	-	Accept the friend request.

12. EditGroup

When a user wants to chat with multiple people without contacting each one individually, they can simply create a group chat.

- **Frame**



- **Attributes**

Variable's Name	Access Modifier	Data Type	Description
out	private	ObjectOutputStream	Send a message to the server.
GroupName	private	String	To create a new group, the group name will be sent to the server and stored in the database.
CurrentUser	private	String	When a new group is sent to the server, it must include the first member of the group as the leader.

- **Methods**

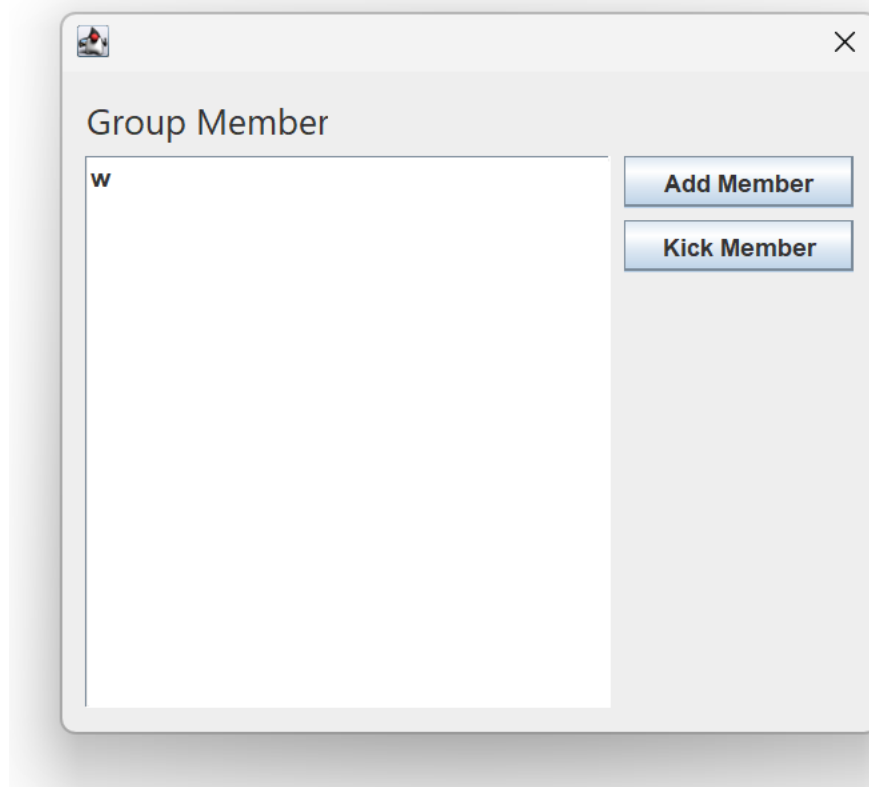
Method's Name	Access Modifier	Data Type	Description
---------------	-----------------	-----------	-------------

OkButtonActionPerformed	private	void	The group name will be sent to the server after the user presses the OK button.
CancelButtonActionPerformed	private	void	Closes the add group frame (EditGroup class).

13. GroupInfoJDialog

Information about a group member appears in this frame. The user can invite or remove members from the group.

- Frame**



- Attributes**

Variable's Name	Access Modifier	Data Type	Key	Description
out	private	ObjectOutputStream	-	Send a message to the server.
groupName	private	String	-	This variable is needed for interactions between the user and the group system, such as kicking

				or adding members.
CurrentUser	private	String	-	The username that interacts with a group.
FriendList	private	DefaultListModel	String	Displays the list of members in a group.

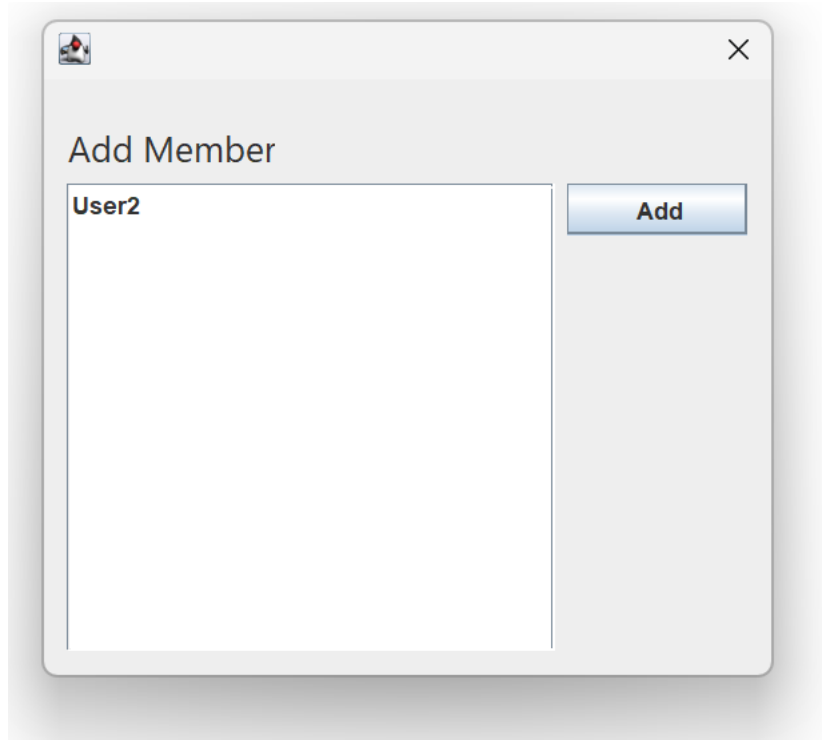
- **Methods**

Method's Name	Access Modifier	Data Type	Key	Description
AddButtonActionPerformed	private	void	-	Displays add member frame to add a member to a group.
KickButtonActionPerformed	Private	void	-	Removes the selected member from the MemberJList.
getMemberJList	public	JList	String	Getter for the MemberJList variable.

14. AddMemberJDialog

The user can add a new member to a group to start a group conversation, with the condition that the user and the new member are friends.

- **Frame**



- **Attributes**

Variable's Name	Access Modifier	Data Type	Description
out	private	ObjectOutputStream	Send a message to the server.
groupName	private	String	When a user wants to add a member, the group must be properly initialized for the server.

- **Methods**

Method's Name	Access Modifier	Data Type	Key	Description
AddButtonActionPerformed	private	void	-	This frame sends the group's name and the new member to the server for management in the database.
getFriendList	public	JList	String	Getter for the FriendList variable.

Configuration and Installation Guide

First, run Server.java, then run AppContact.java. Start interacting with the GUI, and the application will work.