T.C.

FATİH SULTAN MEHMET VAKIF UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

BLM19304 – Computer Network

Project Team Supporting Multiple Customers Communication Server Project

Name-Surname:

Yusuf Semih Kurt

Number:

2021221043

Instructors:

Arş. Gör. Samet KAYA

Prof. Dr. Ali Yılmaz ÇAMURCU

İstanbul, [Mayıs 2024]

İçindekiler

1- Summary

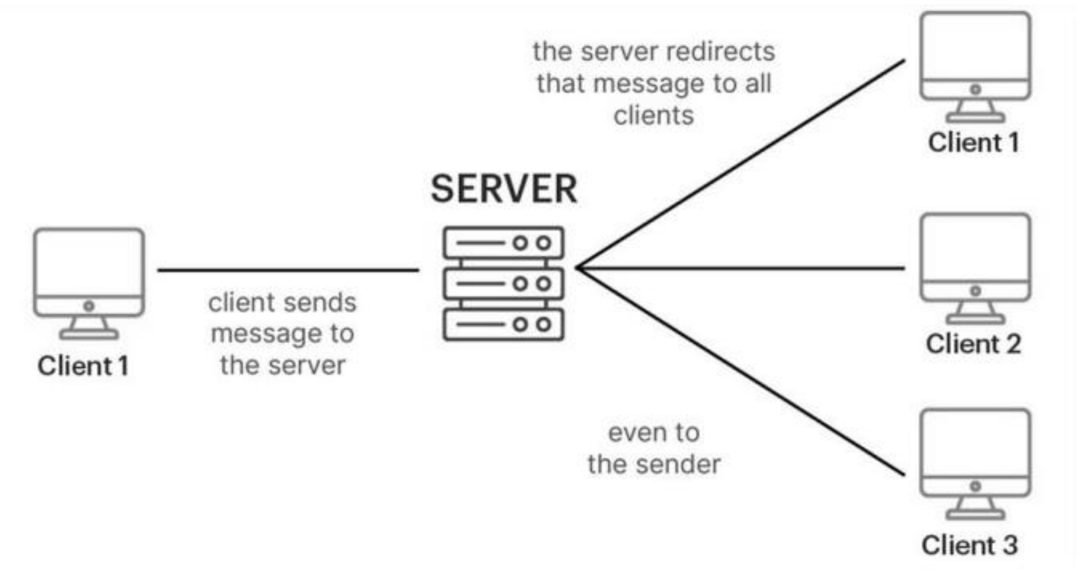
Multiclient refers to the capability of a system or software to serve multiple clients simultaneously. A client is a computer or device that connects to a network and accesses a server that provides services. Multiclient systems are typically utilized in network-based applications or server-based services. In such systems, the server accepts and processes requests from multiple clients. By connecting to the server, clients can exchange data, access resources, or use services.

In summary, multiclient architectures are commonly employed in systems and software designed to serve multiple users concurrently.

Socket Programming is a programming approach used to facilitate communication over computer networks. Sockets are communication interfaces used to send and receive data between computers on the network. Socket programming generally operates on the TCP/IP protocol suite. In socket programming, there are server and client sides.

The client is an entity that requests services in socket programs. It connects to the server, sends a request, receives the response, and can disconnect whenever it desires.

The server, on the other hand, provides services in socket programs. For a client to connect, the server must be continuously running and monitoring the port to which the client will connect.


the server redirects that message to all clients
Client 1
SERVER
Client 1
client sends message to the server
Client 2
even to the sender
Client 3

2- Topic Of Project

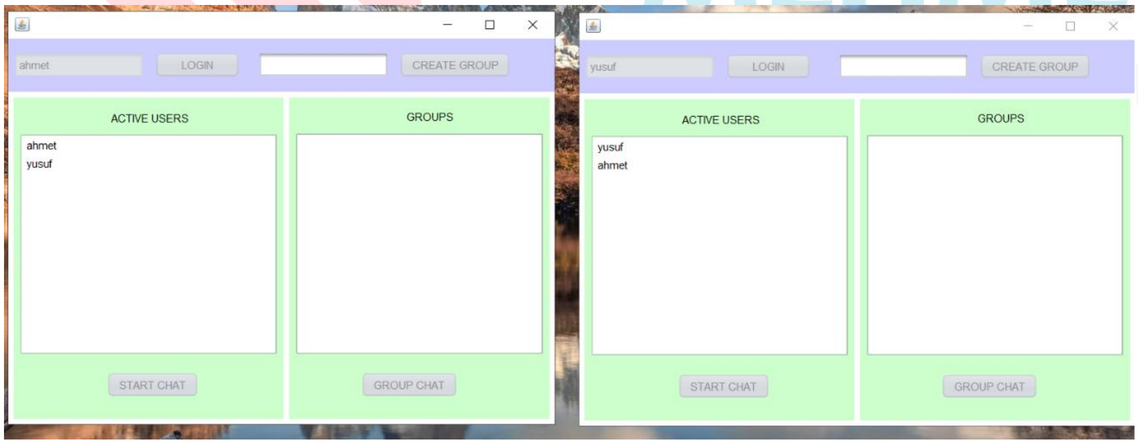
Multiclient chat application is a software that allows multiple users to communicate at the same time. This type of application provides a platform where different users can send and receive text-based messages with each other. Multiclient chat application is usually built on a server-based architecture. The server is a central point where all users connect and can receive messages sent by users, forward them and forward them to other users. Users can communicate with each other by connecting to this server from various devices (computer, smartphone, tablet, etc.). In this type of chat application, when a user sends a message, the server uses appropriate protocols to forward that message to other users. Messages are usually delivered in real time, meaning they reach target users as soon as they are sent. Thus, a message written by a user can be quickly seen and replied to by other users.

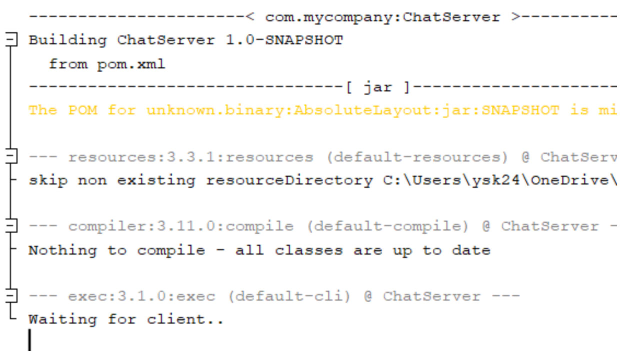
- Users log into the system with their username and password.
- A user can create a project with a unique name.
- When a project is created, a unique key is generated and sent to the project owner.
- Projects that a user is involved in are listed when they log into the system.
- When a user enters a project, if they are the project owner, they can see the project's key.
- Other users can join a project if they know the project key.
- Members involved in a project can communicate in a common area.
- Users in the same project can send messages to each other.
- A project member can see the team members who are currently online.
- Clients can send different types of files (txt, jpg, pdf, etc.) to each other.
- Providing emoji support in the messaging section will earn an extra +10 points.
- The server must be deployed on a remote server platform like Amazon and made operational there.
- Design the system and provide a solution using socket programming.

3- Project Workflow Diagram

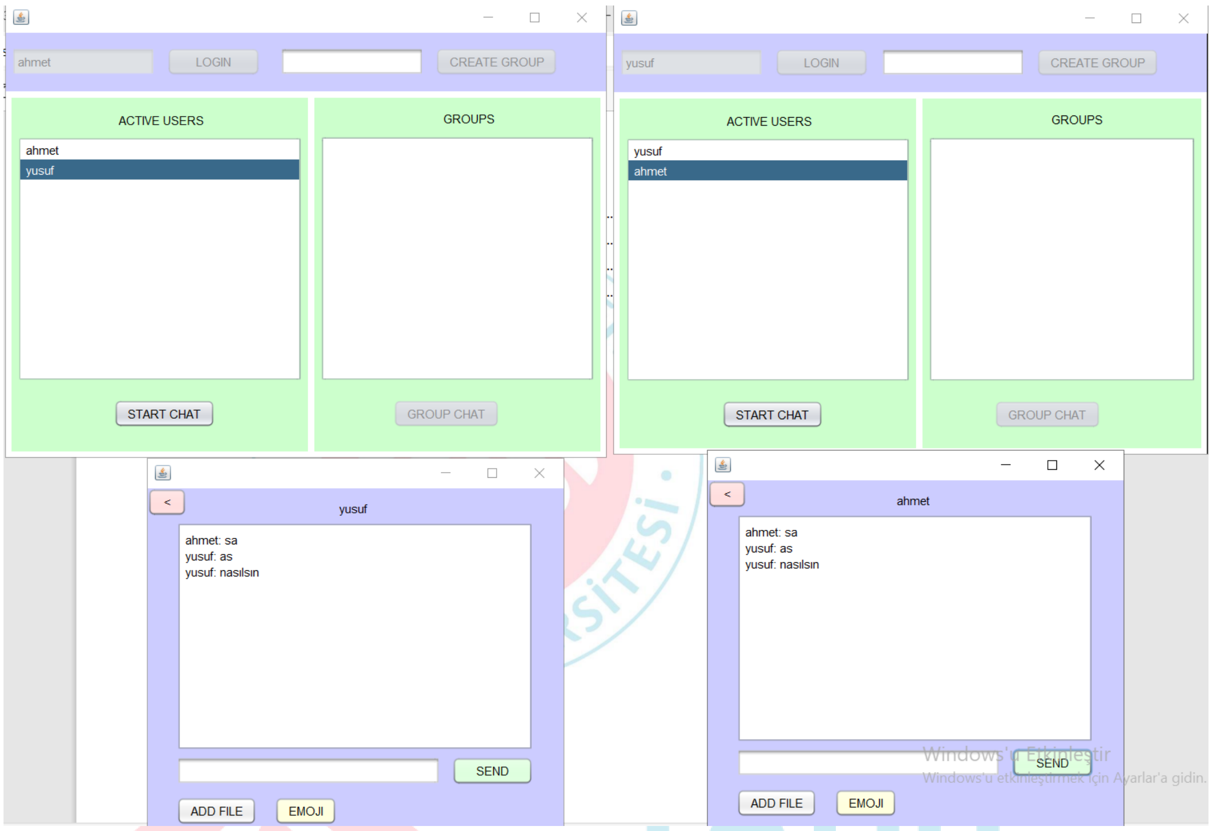
İşin Tanımı	2008												2009					
	O	Ş	M	N	M	H	T	A	E	E	K	A	O	Ş	M	N	M	H
1. Literatür Taraması	🟥	🟥																
2 Altbirim Tasarımları			🟥	🟥	🟥													
3 Teorik olarak Analizi						🟥	🟥											
4 Simulasyon Test ve Doğrulama						🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥				
5. Sistem Entegrasyonu ve Test Aşaması														🟥	🟥	🟥		
6. Testlerinin Gerçekleştirilmesi, Ürün Ticarileştirme																🟥	🟥	🟥

4- Project Outputs and Success Criterias



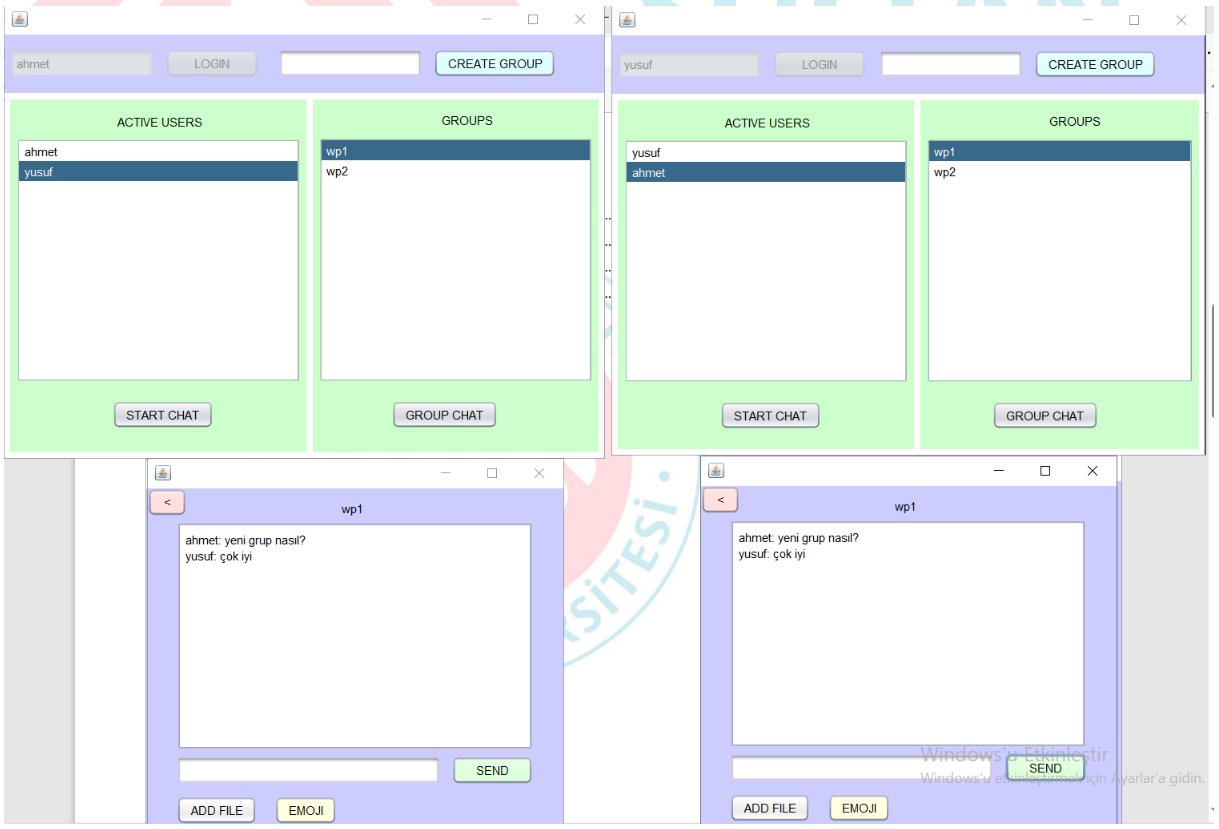


```
---------------------< com.mycompany:ChatServer >----------
Building ChatServer 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------
The POM for unknown.binary:AbsoluteLayout:jar:SNAPSHOT is mi

--- resources:3.3.1:resources (default-resources) @ ChatServ
skip non existing resourceDirectory C:\Users\ysk24\OneDrive\

--- compiler:3.11.0:compile (default-compile) @ ChatServer -
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ ChatServer ---
Waiting for client..
```



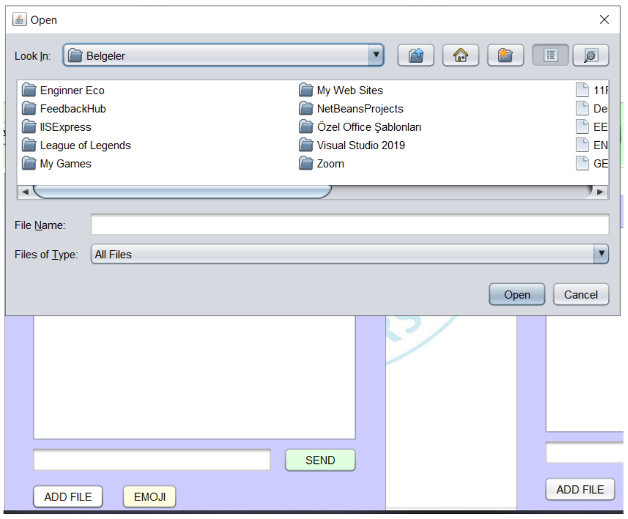
ahmet: sa
yusuf: as
yusuf: nasılsın

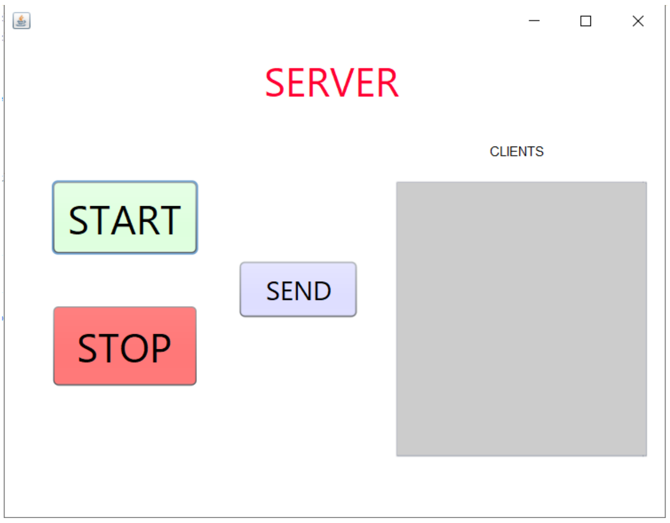
ahmet: sa
yusuf: as
yusuf: nasılsın



ahmet: yeni grup nasıl?
yusuf: çok iyi

ahmet: yeni grup nasıl?
yusuf: çok iyi





5- Project Design

Firstly, I created a server interface for the project design. This interface allows for easy connection to the server with start and stop buttons, and it can be seamlessly integrated into the AWS platform to enable opening and closing the server effortlessly. All clients connected through the interface are listed along with their IP addresses and port numbers.

In the next stage, I designed a screen displaying the names of users connecting to the server. On this screen, users can connect to the server by entering a name with a limited number of characters. Following this, a messaging area was designed. This area features a global chat screen where all users connected to the server can communicate. Messages sent by users are displayed on the screen in a list format. Additionally, a button was added to allow users to select and share files from their computers. Users can also view the list of people currently online (connected to the server) on this screen, making it possible to select individuals from the list and send one-to-one messages.

Finally, I included a chat room feature. Users can create chat rooms by naming them, and others can join these rooms to message within that specific group. This setup enhances the communication capabilities within the project by offering both global and private messaging options.

6- Actions Taken During the Project

6.1. Requirements Analysis and Planning

At the beginning of the project, a detailed requirements analysis was conducted to identify the needs of the users. Based on these requirements, a project plan was created, outlining the phases of development. The technologies to be used, system design, and task distribution were clarified.

6.2. User Registration and Login System

- A registration module was developed to allow users to register with a unique username.
- The ability to log in with a username and password was implemented.
- A database was used to store user information, enhancing data security and access speed.

6.3. Project Creation and Management

- Users were enabled to create projects with unique names.
- Upon project creation, a unique project key was generated and sent to the project owner.
- The system was set up to list the projects a user is involved in upon logging in.

6.4. Messaging System

- A group messaging feature was developed, allowing project members to communicate in a shared space.
- The system enabled users in the same project to send direct messages to each other.
- Users could see project members and which team members were online in real-time.

6.5. File Sharing

- A file-sharing module was developed to allow users to send files in various formats (txt, jpg, pdf, etc.).
- Protocols and methods were established to ensure secure and efficient file sharing.

6.6. Emoji Support

- Emoji support was added to the messaging section to enhance user experience.
- Necessary adjustments were made to allow emojis to be seamlessly added to and displayed in messages.

6.7. Deploying the Server on AWS

- The developed server was deployed on the Amazon Web Services (AWS) platform.
- Configurations and optimizations were carried out to ensure the server ran smoothly on AWS.
- The server's performance and security were enhanced using AWS's features.

6.8. Testing and Debugging

- All system modules were tested individually and in an integrated manner.

- Identified bugs were fixed, ensuring stable system operation.
- Various improvements were made to enhance user experience.

6.9. Documentation and Conclusion

- Detailed documentation was prepared for each phase of the project.
- User guides and technical documents were created.
- The project was completed and successfully delivered according to the initial requirements.

As a result of these processes, a comprehensive project communication server was developed, allowing users to communicate effectively within projects, share files, and interact with each other.

7- Additional Explanations

I succeeded in connecting to the AWS server in my project. I followed the video shared by our teacher Samet on LMS and took the necessary steps. I initially opened my project with java with ant. After watching this video, I realized that I needed to build my pom.xml file. Even though it was a little late, I converted my java project to a maven project. I had many problems while translating and lost time because I had to transfer all the dependencies completely in the dependencies section, and with the slightest deficiency, the entire project became inoperable. After converting the Maven project, I built my xml file and then uploaded the required jar file to my Amazon server, but I had a problem running this jar file.

8- References

- https://www.geeksforgeeks.org/multithreaded-servers-in-java/
- https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/java-rds.html
- https://www.geeksforgeeks.org/socket-programming-in-java/
- https://www.javatpoint.com/socket-programming
- https://stackoverflow.com/questions/67579226/how-display-label-with-emojis-using-swing-java

FATİH SULTAN MEHMET
VAKIF ÜNİVERSİTESİ