CENG 216

COMPUTER NETWORKS

SPRING 2024

SEMESTER PROJECT

Important Notes:

- The project will be carried out in groups of students and each group will consist of a maximum of **6 students**.
- Students in the group should share the project modules.
- The project report should be created in ".pdf" format.
- Important code structures used in coding can be shared in the project report. *** Do not share all codes in the report.
- Create all coding as a project as an archive (.zip) file and all group members should do the coding on Github.
- Each group should share their work on their Github account **with evrimguler@bartin.edu.tr**.
- Github repository names should be defined as **Spring2024_CENG216_GrupNo**.
- Create a video recording of a short simulation and introduction of the project, maximum 10 minutes.
- Submit the project report, coding and video recording as a single archive file using the project submission form.
- Project Group Members Notification and Project Topic Determination & Participation Form submission deadline: ***March 14, 2024 23:59***
- A single submission from each group is sufficient for Project Reports Submission. ***May 17, 2024 23:59*** is the deadline.

PROJECT
Real-time Instant Messaging Infrastructure

Instant Messaging (IM) is one of the core elements of the web and has been around almost since its inception, starting with simple text-based programs like talk and IRC, and progressing to today's GUI-based IM clients like Google, Yahoo, Microsoft, AOL, etc. In this project, you will design and implement an IM system, including both client and server.

The following features restrict the design space of an IM system:

In real-time communication, an IM conversation takes place in real-time: one person types a text, presses "enter" and the other person (almost) immediately sees the text.

An IM conversation can take place between two or more people. Some systems allow only two people to communicate, others allow more than two people. Most systems allow one person to be involved in more than one conversation at the same time. The main mode of communication takes place via text as opposed to audio or video and is connected over a network. The parties involved in the communication can be physically far away and are connected via the internet.

Your task will be to design an instant messaging system with the above features as well as additional features that you will incorporate into your design. This system will include a server component that handles the transfer of messages and other data, and a client component with a graphical user interface.

Objective:
The goal of this project is two-fold. First, you will use a variety of Java or familiar technologies, including networking (to support connectivity over a network), sockets and I/O (to support real-time, text-based communication), and threads (to support two or more people communicating simultaneously). State machines can be useful for specifying certain aspects of the system's behavior.

Secondly, you will need to think about the best way to present your chat system, which will require a graphical user interface.

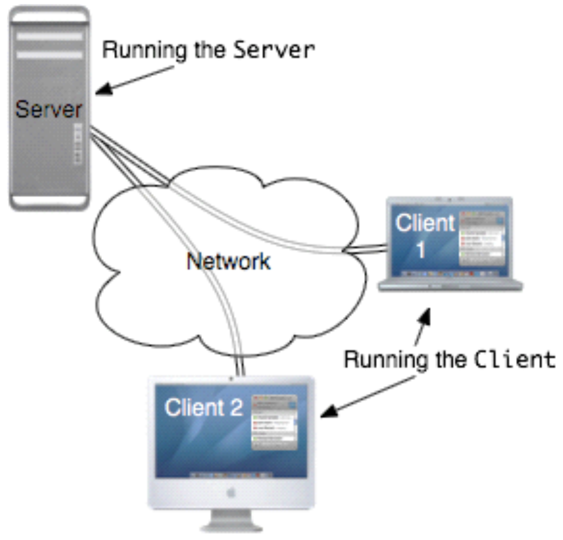
Here is what you will do:

- You will be familiar with infrastructures that have a graphical user interface (GUI) toolkit for Java or any other infrastructure you know, which is similar to many other such toolkits;
- You will be able to use important GUI programming concepts, including the concept of view hierarchy and the model-view-controller design pattern;
- The idea is to use the listening design pattern in various ways, not only in your GUI, but also in the more general publish-subscribe sense.
- Throughout the project, you will need to design and implement mutable data types, paying particular attention to concurrency issues and how they interact with each other depending on the characteristics of the structures created.

Specifications:
Implement an IM system in Java or another framework you know with the following features:

Client. A client is a program that opens a network connection to the IM server from a specified IP address and port number. The client must have a way to specify the server IP, port and user name. Once the connection is opened, the client program presents a graphical user interface to perform the interactions listed below.

Server A server is a program that accepts connections from clients. A server must be able to hold a large number of open client connections (limited only by the number of free ports) and clients must be able to connect and disconnect at will. The server must also verify that client usernames are unique and handle collisions gracefully.



The server is responsible for managing the state of both clients and conversations.

Conversations. A conversation is an interactive text exchange session between a certain number of clients and is the ultimate goal of the IM system. The exact nature of a conversation is not specified, except to say that it allows clients to send text messages to each other. Messaging in a conversation should be instantaneous, i.e. incoming messages should be displayed immediately, not held until the recipient requests them. You should visually separate messages of different conversations (e.g. into different windows, tabs, panes, etc.).

Client/server interaction. A client and server interact by exchanging messages in a protocol that you design. Using this protocol, the user interface presented by the client must

- Provide a facility to see which users are currently logged in;
- Create conversations with an opportunity to join and leave conversations;
- Allow the user to participate in multiple conversations at the same time;
- Provide a history of all messages in a conversation as long as the client is in that conversation.

Authentication. In a product system, logging in as a client will require some form of password authentication. For simplicity, this IM system will not use authentication, meaning anyone can log in as a client and claim any username they choose.

<u>Tasks of the Project:</u>
- Team preparation. Meet your team and write a team agreement.
- ***Conversation design.*** Define a precise conversation concept in your IM system. Look for tips on how to do this. In particular, name the Java or other platform you will create to implement conversations, name

their classes, give the properties of their public methods and give a brief description of how they will interact. Include a snapshot diagram of a conversation in action.

- ***Client/server protocol.*** Design a set of commands that clients and the server will use to communicate, allowing clients to perform the actions stipulated in the specification. Create a specification of the client/server protocol as a grammar. Also think about the state of the server and the state of the client (if any).
- ***Usability design.*** Draw your user interface and its various screens and dialogs. Use these drawings to quickly explore alternatives and plan the structure and flow of your interface. It is recommended to draw on paper. Submit the drawings you decide to use for preliminary preparation, together with any comments needed to explain parts that are not clear.
- ***Synchronicity strategy.*** You should argue that your design is free from constraint conditions and deadlocks. Be specific about which data structures or design patterns you will use to ensure thread-safe behavior.
- ***Test strategy.*** Develop a strategy for testing your IM system. Describe which automated tests you will use and which manual tests you will perform. Documenting your strategy is especially important since UI front-end testing is usually easiest done manually. As you think about how you will test your program, you will probably want to rethink your code design (for example, it may be necessary to make a cleaner API to allow unit testing independent of the GUI).
- ***Implementation.*** As always, your code should be clear, well organized and usefully documented.
- ***Testing*** Implement your testing strategy by using Junit (optional) and performing manual tests of the GUI. Document the results of your manual tests in your report.
- ***Outcome.*** Each team member will write a short comment explaining what you learned from this experience and write a paragraph about each of them:
    - Product. What was easy? What was difficult? What was unexpected? If you were to do it again, what would you do differently when designing the chat system?
    - Team. How did you feel the group worked? How did your team work? How was the coding? How did you divide the work?
    - Individual. How do you think you personally did? What did you do in the project? How do you feel about it?