Presentation Project Guideline

Building on the algorithmic and programming skills you have already mastered, this project is designed to provide an additional challenge. You will work in a team of one, two, or three and will be required to complete one, two, or three topics, depending on the size of your team. One of these topics will be compulsory, while the others will be selective. All topics must be implemented on top of your completed distributed chat system. We recommend integrating your work on the selective topics with the GUI for optimal results.

The compulsory topic must be completed even if you are working alone. Additionally, if you are working in a team of two (resp., three), you must complete one (resp., two) selective topics.

Compulsory topic:

GUI: Add an interface to your chat system. (A sample is available, and you may refer to it. However, you are expected to extend it a bit.)

Selective topics: (You may choose one from the following)

- Secure messaging: guard against eavesdropping
- Online gaming: extend the chat system with game functionality

Note: Teams can propose different projects, but they must (1) be approved with instructors' consensus, (2) have an equivalent degree of difficulty, and (3) extend the chat system (ideally).

In Week 14, you will submit a video record in which you present your project, including a demo of your app.

Graphic User Interface (GUI)

Currently, the chat system is run from the terminal command line. All input and output are only found from the plain terminal window. You might want to add a user interface to enhance the chatting experience.

The goal: To construct an interface using Python packages like Tkinter so that clients do not need to type or read from the terminal window when connecting to the chat server. All information being exchanged will be displayed on the GUI.

The bottom line is that you need to make a GUI that can take and display messages for the user (e.g., you may add several buttons for "time", "who", "poem", and "search", respectively). Other exciting features are logging in with passwords, file transferring, voice/video chatting, etc.

Besides, you should try integrating your work on the selective topic with the GUI. For example, if your selective topic is Secure Messaging, you can use encryption algorithms to transmit the messages behind the GUI. Or, if your selective topic is Online Gaming, you may add a button in the GUI that can start the game you designed.

Secure Messaging

Turing and his friends built one of the first computers to crack the Enigma code. Every technology is a double-edged sword. As a user, you can protect your transactions against eavesdropping.

But the question is: *how?*

Of course, the intuitive idea is to scramble your message, which we call encryption. You want your friend to read the clear message, so you will need decryption. This forms an agreement between you and your friend before you send your scrambled message.

On the Internet, everything is in the open. So, there is no hideaway where you and your friend can meet. How is it possible to establish such an agreement and then communicate?

The field of computer security (and a related field, data privacy) is a vast and interesting research area. In this project, however, we only want to get a taste of it.

The goal: To get two chat clients to communicate securely, meaning that the server passing the messages in between has no idea what they are talking about. You must implement a shared key protocol and a simple encryption/decryption algorithm for the client side and make this work!
Reference: MacCormick, 9 algorithms, **Chapter 4**.

Online Gaming

Online gaming is one of the fastest-growing industries in the world today. It is a vast, interdisciplinary field that brings together artists and creative professionals (asset generation and narrative development), business executives (marketing, logistics, distribution), and, of course, talented computer science professionals (game engine design, system infrastructure, implementation, QA). This project topic is designed to give you a taste of some of the complexities behind implementing a simple game. The game shall use the chat system as its backbone.

In the chat system, the communication model for chatting across clients is to send a message to the server, have the server forward the message to a client (or set of clients), and then have the server relay responses. Essentially, our server is our “man in the

middle," which the clients rely upon for sending and receiving messages.

The goal: To adapt this model for implementing a simple game, such as Tic-Tac-Toe, which is the suggested game to implement, or you may find a game from a website such as Pygame(https://www.pygame.org/news) or Tkinter-based games from GitHub (https://github.com/, using the search bar with the keywords like "tkinter games") and adapt it to the chat system. Two clients playing a game can send their moves to the chat server, which will evaluate them, maintain a consistent game state, and relay the changes in the game state to the clients as appropriate. Incidentally, this model is conceptually similar to how many famous and successful online games actually work.

Grading Policy

The work will be evaluated on the following criteria:

- Compulsory topic: 40%
    - Baseline: the GUI can take and display messages for users
- Selective topic: 20%
    - Completeness: the code works properly, and the goals of the topic are achieved.
- Video Presentation (40%): Your video presentation should include four key components: introduction, demo of the app, discussion, and analysis.
    - Introduction (8%): Your introduction should clearly explain your project and your motivation for creating it. This will help your audience understand the context of your work and why it is important. You should strive to be clear and concise in your delivery to maximize the impact of your message.
    - Demo of the App (8%): Your demo should showcase your app's key features and functionality. This will help your audience understand how it works and what makes it unique. You should aim to create an engaging and informative demonstration highlighting your work's strengths.
    - Discussion (8%): Your discussion should provide insights into your development process. This may include discussing the packages you used, any significant modifications you made to the original chat system or other relevant details. By sharing this information, you can help your audience better understand the technical aspects of your project.
    - Analysis (8%): Your analysis should reflect on the development of your project. This may include discussing the organization of your code, the classes you defined, or possible areas for improvement. By providing this analysis, you can demonstrate your expertise and thoughtfulness as a developer.
    - Each component will be evaluated based on the following criteria (clearance, impressiveness, and relevance to the project):
        - Clearance: how easy it is to understand your presentation
        - Impressiveness: how well you showcase your skills and expertise

        - Relevance: how well your presentation aligns with the objectives of your project
- Your presentation should be 10 to 15 minutes long (8%). Going over or under this time frame may result in a loss of points. Focusing on clarity, impressiveness, and relevance can help you create a compelling video presentation that effectively showcases your work.