Presentation Project Guideline

Having mastered a great deal of algorithmic and programming skills, the goal of the project is to challenge you further. You will be asked to work in a team of two and complete two topics. One is the compulsory topic and the other is the selective topic. They should be implemented on top of your completed distributed chat system. Ideally, your work on the selective topic should be integrated with the GUI.

**Compulsory topic:**

- [GUI](#_bv82a6fc1744): add an interface to your chat system

**Selective topics:** (You may choose one from the followings)

- [Secure messaging](#_1ncdxn8rwkq2): guard against eavesdropping

- [Reliable messaging](#_d1h0ategwo81): protection against a bad communication channel

- [Online gaming:](#_jr3z836bdyeu) 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).

You will submit **a video record** in which you present your project including the demo of your app in Week14.

## 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 there is no need to type or read from the terminal window when a client connects 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. Other interesting features to try out are logging in with passwords, file transferring, voice/video chatting, etc.

Besides, you should try to integrate your work on the selective topic with the GUI. For example, if your selective topic is Secure Messaging, you can use the 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, by itself, stands as a double-edged sword. As a user, you have the right to protect your transactions against eavesdropping.

But the question is: *how?*

The intuitive idea is, of course, to scramble your message, a process that we call encryption. You want your friend to read the clear message, so you will need decryption as well. This takes the form of an agreement between you and your friend before you send your scrambled message.

On the Internet, everything said is in the open. So there is no hideaway in which you and your friend can meet. So 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 with each other securely, meaning that the server who is passing the messages in between has no idea what they are talking about. You will need to implement a shared key protocol as well as a simple encryption/decryption algorithm for the client-side, and make this work!

Reference: MacCormick, 9 algorithms, **Chapter 4**.

## Reliable Messaging

Going over your homework with your classmate in a noisy location is challenging. It has a lot more to do with the noise than what you drink.

Interestingly, this is one area in which a computer seems to do slightly better than a human. The idea, as covered in the 9-algorithm book, is to tell the other party just a little bit extra. These extra bits carry enough information to recover what the environment might have corrupted from your message.

If you are interested, you should read more on information theory, a discipline that giants such as Claude Shannon helped build. But here, we just want to get a taste of it.

In the chat system, the server is responsible for passing a message from A to B (two chat clients).

**The goal:** Your server should emulate the "noisy location" by flipping bits of the messages randomly. Your next task is to implement a checksum on the client-side so that despite a bad server, the clients read each other’s messages without a problem.

You could try this algorithm with your friend in the noisy bar, and start to appreciate why computers do better than you!

Reference: MacCormick, 9 algorithms, **Chapter 5**.

## Online Gaming

Online gaming is one of the fastest-growing industries in the world today. It is a huge, 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, and 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 model of communication 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). Two clients playing a game can send their moves to the server, which will evaluate them, maintain a consistent game state, and relay the changes in game state to the clients as appropriate. Incidentally, this model is conceptually similar to how many popular and successful online games actually work.

## 

## Grading Policy

The work will be evaluated on the following criteria:

* Compulsory topic: 50%
  + Baseline (40%): the GUI can take and display messages for users
  + Innovative features (10%): the GUI contains one (or more) ingenious functions, e.g., login password, emoji, etc.
* Selective topic: 20%
  + Completeness (10%): the code works properly, and the goals of the topic are achieved.
  + Integrated into the GUI (10%): the functions work with the GUI
* Video Presentation: 30%
  + Introduction(6%): Tell the audience what your project is; what is your motivation?
  + Demo of the app(6%): A demo of your work, showing what your app looks like and how it works
  + Discussion (6%): Tell the audience what you did in the development, e.g., what packages you use, and/or, the significant modification in the original chat system
  + Analysis(6%): Your reflections on the development, e.g., is the organization of your code reasonable? What classes do you define? Or, the possible improvements
  + Impression(6%): This is about the impression of your presentation and your workload, being ranked with regard to others