Criterion A: Planning

Defining the problem

My client, xx, is an active Discord (a messaging social platform) user. He has a Discord server (A Discord space for a community of people to connect) with his friends where he gets to connect with his friends in America.

Since most of his friends are Rubik's cube speed solvers (taking a "mixed up" cube and making all sides having the same color as fast as possible), they would like a program within Discord that provides scrambles (a sequence of moves that "mixes up" the cube), times solves, and gives statistics on past solves. Although cstimer (a popular speedcubing timing program) supports these features, it is not embedded in Discord. In addition, my client would like to compete against his friends, which cstimer does not have.

There are some cubing Discord bots (AI-driven tools that help automate tasks on Discord servers) right now that could generate scrambles in Discord. However, current Discord bots either are outdated (not supporting slash commands, which is a new feature that replaced text commands) or don't have a timing and competition feature.

I volunteered to develop a Discord bot for my client that would help him time his solves and allows him to compete against his friends. My CS teacher, Mr. xx, agree to be my supervisor for this project.

Rationale

I decided that a Discord bot would suit my client the best as all the commands could be performed within Discord. Also, his friends are in the same server, making Discord bots easily accessible by his friends as well. Moreover, Discord bots now supports buttons and various other GUI, allowing for rich user-bot interaction.

I chose to use python as the language because I am proficient at python. Furthermore, python has a library called nextcord that is used for developing discord bots. Nextcord was chosen other discord bot libraries, such as discord.py and pycord, because nextcord is regularly maintained and supports slash commands.

Success Criteria

1. Can provide user with instructions on how to use the application
2. Can generate scrambles for 2x2 to 7x7 cubes
3. Displays image of the cube to ensure that the user have scrambled the cube according to the scramble
4. User can time his solves
5. Has a virtual cube that the user can turn with by using cubing notation (see appendix)
6. User can challenge other users to a timed solve

7. User can add penalty to a solve
8. Displays the most recent solve times
9. Generates current and best statistics of the user's solves, which includes: average of 5 (ao5), average of 12 (ao12), and mean of 3 (mo3). Average of *x* is calculated by taking *x* consecutive solves, removing the fastest and slowest solve, and calculating the mean of the remaining. Mean of *x* is simply the mean of *x* consecutive solves
10. User can manually add times
11. User can delete unwanted times