Project Title: Connect Four

Team: “Tic-Tac-Oh-No”

- Dureke Sanchez II

Project Objective: The objective of this project is to, at minimum, use a server and two client applications to run a connect four game between networks using Python and sockets. The game will be, at minimum within a terminal as a CLI, but should time permit, a full GUI.

Scope:

- Inclusions:
    - Server Python file that will handle server operations
    - Client Python file that each player will use to connect to the shared server
    - Connect Four Game that two players of internet connection can utilize
    - Client can use the following arguments:
        - -h (help show the player how to connect, and play)
        - -i ip address of the server (required argument)
        - -p listening port of the server
        - -n DNS name of the server
    - Server can use the following arguments:
        - -h (help show the user how to run the server)
        - -i host-ip
        - -p port
- Exclusions:

Deliverables:

- Github Wiki page describing documentation and development
- UML Diagram explaining classes
- Working Python script to play a full game of Connect Four
- Client-Server architecture to ensure effective communication between server and clients
- A clear messaging protocol between server and client
- Server can handle multiple clients connecting and playing simultaneously, as well as disconnections
- Server can handle unexpected errors easily and informs the user an error has occurred, with detailed explanation
- Code will be clean, commented, efficient, modular, and factored without duplicating code
- A README.md file that explains how to run the game, its features and limitations, as well as any known issues and/or bugs
- An easy to use UI, whether it be a GUI or CLI
- An accurate implementation of the Connect Four rules, as described by Hasbro

Timeline:

- Key Milestones:

    - Sprint 0: Form teams, Setup Tools, Submit SOW [Template] (Sept 08-Sept 22)
    - Sprint 1: Socket Programming, TCP Client Server (Sept 22-Oct 06)
    - Sprint 2: Develop Game Message Protocol, Manage Client connections (Oct 06-Oct 20)
    - Sprint 3: Multi-player functionality, Synchronize state across clients. (Oct 20-Nov 03)
    - Sprint 4: Game play, Game State (Nov 03-Nov 17)
    - Sprint 5: Implement Error Handling and Testing (Nov 17-Dec 6)
- Task Breakdown:
  Tasks are broken down into 3 epics, each handing a domain of the project. There is a fourth domain that is responsible for testing and ensuring everything is working as intended. Tasks featuring a star (*) are not a priority but are desired features if time permits.
    - Server Management:
        - errorHandling: handles network failures, invalid input, or unexpected game states
        - serverEncryption*: Symmetric or asymmetric encryptions for the server
        - saveGame: saves the current state of the game to an iterable file name using TBD. Stores previous moves, current game board, current player's turn
        - loadGame: call from client to load a previously saved game
        - gameInstance: call from client to begin a new game
        - userDatabase (*): keeps track of unique user IDs
    - Client Management:
        - startGame: client call to start a new game using a game ID, which another client can connect to. Game settings can be modified. Game can also be a loaded saved game
        - userInterface (*): A GUI that clients and interact with and easily understand functions of Connect 4
        - downloadGame: Allows the user to download a game state onto the client device
        - ClientEncryption (*): Client portion of encryptions. Symmetric or asymmetric TBD
        - userLogin (*): Allows the user to reuse unique user ID to track game history and use for invites
    - Game Management:
        - gameRules: Governs the rules of Connect Four, defines the win condition, delivers who begins their turn first
        - gameSettings: Allows for timed games, change of colors, modification of game rules
        - matchInfo: handles the display and recording of games in progress. Informs client of match scores, current player's turn
        - gameDatabase: database of previous matches and users
    - *Testing*

        - Tests invalid moves, three or more connecting to a single game, network failures, invalid inputs, saves to server without the space, saves with same file name, loading a game, ensuring game ends on win condition

Technical Requirements:

- Hardware:
    - Internet connectivity
    - Keyboard
    - Server: At least 1 MB of space (optional save)
- Software:
    - Python 3.9 or higher
    - Terminal Access

Assumptions:

- Server response will be within 1 second of client sending an action
- Server has room for a game file to be saved of at least 1 MB
- Client firewall does not block the connection established and communicated with the server

Roles and Responsibilities:

Since this is a solo project, I will be responsible for weekly updates to documentation, project planning, code review, and general coding.

Communication Plan:

Project should be updated to Github regularly on a weekly basis with a new version released every 2 weeks. Documentation and changes to the project will be updated to wiki at the end of the week (Sunday).

Additional Notes: