*PROG 280 Final Project*

The completed project will vary greatly depending on student aptitude and software design practices, the basis of the project will depend on the client/server architecture, in addition there may be some local/offline functionality. How the client server connection is established and maintained for the project is up to each individual student. Each student will should maintain a backlog of their project work. The backlog will indicate all tasks, which feature they are associated with, and the status of the task:

* Not-Started
* In-Progress
* Abandoned
* Completed

During the project creation process, marks will be awarded for maintaining a backlog, discussing the project progress, as well as the finished project. You are encouraged to talk to classmates for inspiration or help to implement functionality. Note you should choose one of the two projects below, you may switch projects but any work completed should be logged and submitted for marks.

*Basic Functionality*

The TCPIP chat functionality created in Assignment06 can serve as a starting point for the final project, or you may choose to start from scratch. It is expected that users will be able to communicate with each other, and this communication channel will be used to initiate the more advanced features of the final project. While designing the communication protocol for the final project you may consider issuing commands to the server to trigger functionality, or come up with your own design.

*Documentation*

During the project you will be required to keep a log of the work you are planning, working on, abandoned or completed. The backlog excel document should be updated and uploaded at the end of each class to show progress.

*Option 1: Multiplayer Games Project*

The server for this project should support multiple connections and allow one user to invite another user to participate in a game, but disallow a user to invite themselves. The invitee can accept or decline the game, the server application can either facilitate the game, or help the two clients establish their own connection to play the game. Consider allowing or disallowing a user to participate in multiple game at the same time.

The following is a list of features that you may choose to implement in your project, feel free to create and implement your own features:

* User Accounts
* High Scores
* Ping / Connection Delay
* Turn Timer
* How to Play / Tutorial
* Resign Game
* Artificially intelligent bots for single player mode
* Connection Lost / Rejoin Timer
* Games
* Tic Tac Toe
* Checkers
* Connect Four
* Blackjack
* Othello
* etc
* etc

*Option 2: Meeting / Remote Help Application*

The server for this application should be able to host a meeting which will immediately allow users to chat with the other participants and the host. The server should maintain a list of users, and the server host may boot any users that have joined. Alternatively, you may structure your application to only support a single user. You may also consider adding an access code so that only users that have been invited may participate in the meeting, or let the host user to allow public meetings. The application can be supported by a single window, or multiple windows. At a minimum the application must allow clients to view the host’s computer screen, if the host has allowed this feature. The host should be able to turn this feature on and off at any time. While implementing the sharing of a screen you may want to investigate the difference between UDP and TCP, although either is acceptable and we did not cover UDP in class. Some features to consider adding to the application include but are not limited to:

* Allow clients to request control
* Allow a client user to share their screen
* Remotely control the server OR client computer
* Sending mouse events
* Sending keystrokes
* User Accounts / Avatars
* Hotkey to revoke control from the user ( ESC? )
* Distribute a files to clients
* Prompt users to accept or reject the file
* Encrypted messaging
* Voice Over IP
* etc