# Project Goals:

**An IRC client library:** for connecting to and interacting with IRC servers, to be used in building IRC clients and IRC bots.

**An IRC server library:** to manage connecting to other IRC servers and clients, to be used in building IRC servers.

**Decisions to be made:**

* Full RFC compliance vs more modern IRC implementation
  + Research required.
* Should the server and client library be combined into a single library?
  + If not we may need a 3rd library with code common to both libraries

**Decisions already made:**

* All source code will be licensed under the MIT license.
  + The source code may not be re-licensed for commercial use without the agreement all of contributors.
* Projects will be developed using .NET 5
* Libraries will be developed using .NET Standard 2.1

**Multiple IRC Clients**:

* **A console-based IRC client** with the following features:
  + Ability to connect to an IRC server.
  + Ability to join multiple channels.
    - A way to switch between channels and private messages.
  + Ability to send and receive channel messages.
  + Ability to send and receive private messages.
* **A WinForms based IRC client** with the following features:
  + Ability to connect to an IRC server.
  + Ability to join multiple channels.
    - A way to switch between channels and private messages.
      * Possibly multiple WinForms
  + Ability to send and receive channel messages.
  + Ability to send and receive private messages.

**An IRC Server:**

* IRC server daemon, additional planning to be completed after IRC client phase.

# IRC Client Library Design Decisions:

This to consider:

* Connection to server is persistent:
  + Encapsulate stream object, exposing the read and write methods
  + Listen for events such as stream disconnection
* We need to decide what methods we will expose to the end user (the person using the library) and how.
* We need to determine what TCP events should be listen to and how to do so.
* We need to decide on how to do out abstractions and implement in an OOP way
  + See Discord.Net for examples.

**Config** Class: IRC Client configuration

* GetLogger(string name): Get a logger object

**IrcClient** Class: Represents an IRC client

Methods:

* Connect(): Connect to IRC server
* Disconnect(): Disconnect from IRC server
* SendMessageToServer() – May need to re-think this

**CommandType** Enum: The type of the command/message from the IRC server

* Unknown: default
* ChannelMessage: a message to a channel
* PrivateMessage: A private message to us
* NumericReply: Numeric command Response (See RFC 2812 section 5)

**NumericResponse** Enum: All server response codes, in numeric order, with an example message

* See RFC 2812 section 5

**ServerMessage** Class: Represents a parsed message from the server

* MessageType Type: the type of message
* List<string> Parameters: any parameters to the command
* string Message: The unparsed message from the server
* string Nick: The nick name associated with the message
* string Command: The command
* string Channel: The channel associated with the command (Make this a Channel class?)
* String trailing: The final part of the command, normally the text of a message
* NumericReply ResponseCode: The numeric reply (See RFC 2812 section 5)

**Channel** Class: Represents an IRC channel

* String Name: Channel name
* SendMessage(string message): Send a chat to a channel
* MessageReceived() Event handler: message receive on channel

**User** Class: Represent and IRC user

* String NickName: Nick of user
* String RealName: “Real” name of user
* SendMessage(string message): Send a private message to the user
* MessageReceived() Event handler – Private message receive from user

**Server** Class: Represents and IRC server

* string ServerAddress: Address of server
* int Port: server port

**MessageParser** Class: Responsible for parsing the messages from the IRC server

* ParseMessage() returns a ServerMessage object, the parsed IRC command from the server

# Current Issues / Roadblocks:

The IRC message parser should be re-written after additional research and experience on the project.

* However, I believe to be fully functional at the moment.

The connection process should be async, and the ListenThread is behaving oddly.

* Additional research on threading/Tasks and TcpClient class required.

# Resources:

<https://en.wikipedia.org/wiki/Internet_Relay_Chat>

<https://en.wikipedia.org/wiki/IRCd>

<https://en.wikipedia.org/wiki/List_of_Internet_Relay_Chat_commands>

RFCs (IRC Protocol Explanation)

<https://tools.ietf.org/html/rfc2810>

<https://tools.ietf.org/html/rfc2811>

<https://tools.ietf.org/html/rfc2812>

<https://tools.ietf.org/html/rfc2813>

<https://tools.ietf.org/html/rfc1459>

<https://github.com/unrealircd/unrealircd/tree/unreal50/src> (IRC daemon written in C)

<https://stackoverflow.com/questions/930700/python-parsing-irc-messages> (Parsing in Python)

<https://github.com/pdxmholmes/ircmessagesharp> (IRC message parser in C#)

<https://ircv3.net/> (Modern IRC)

Dependency Injection:

<https://www.stevejgordon.co.uk/using-generic-host-in-dotnet-core-console-based-microservices>

<https://pradeeploganathan.com/dotnet/dependency-injection-in-net-core-console-application/>

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/host/generic-host?view=aspnetcore-5.0>