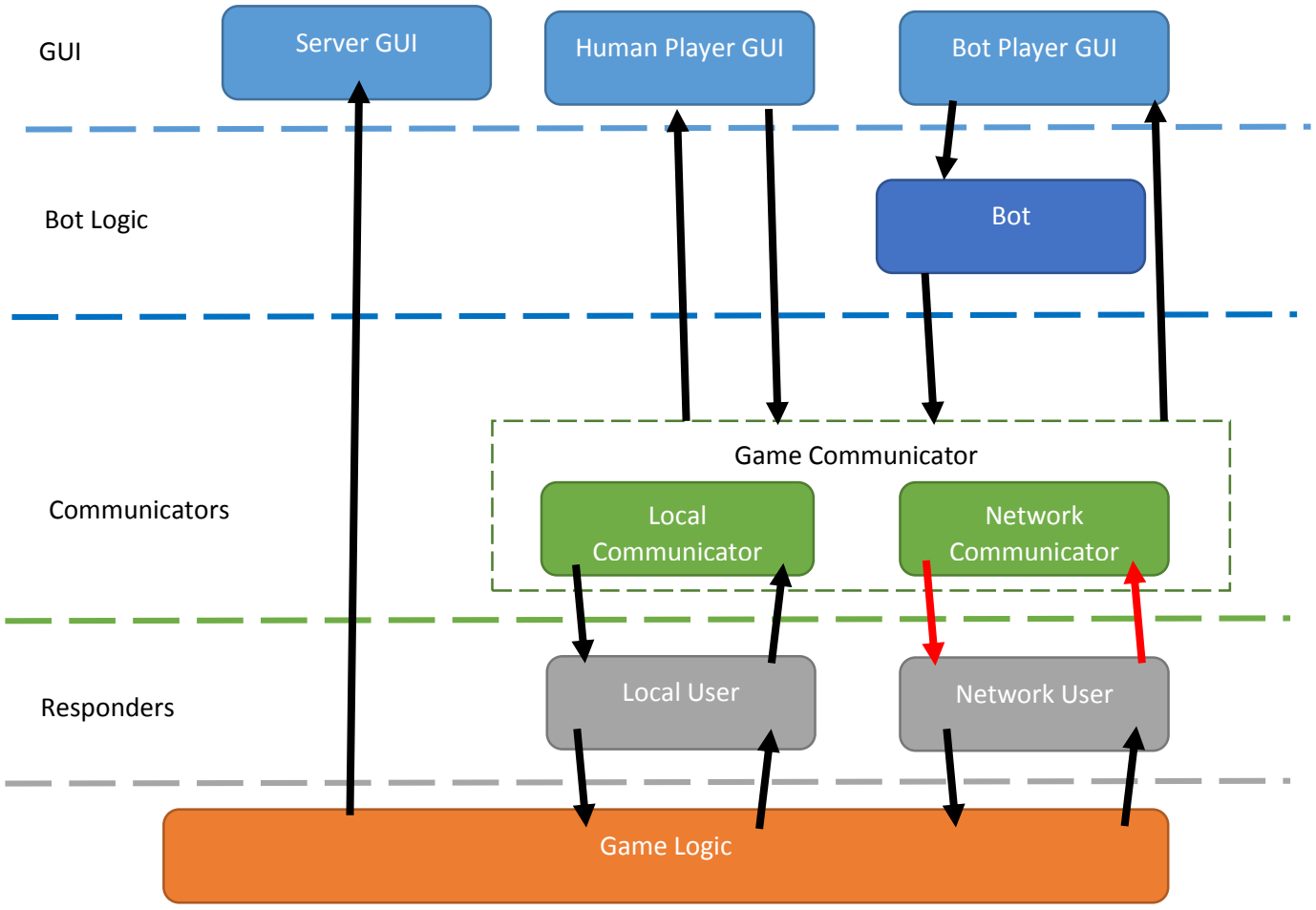
In general the revisions to my initial spec are small, there were some changes most notably the removal of non-user bots which helped give structure to my code better. Instead players must choose to play as a bot and can't launch things straight off. To counter the lack of functionality I introduced multiple types of bots. GUI has also changed slightly.

Refined version of GUI to Game communication

As before we had several classification of classes we had **Communication Classes, Responder Classes, GUI Classes** and the **Game Classes** the latter of which was mainly provided for me with only some additional functionality to allow multiple players, attacking, gifting and also the format of outputs was changed to aid the GUI in formatting (eg. Different characters to represent the state of a player so the player image can be properly drawn).

Here is the general class layout, note black arrows mean flow of data while red arrows mean flow of data via network sockets.



Let me just explain what's happening here. The GUI simply displays things for the user. The controller for the Human Player is built into the GUI while the Bot Player contains a generic bot class to decide on the next action. The server GUI simply receives messages from Game Logic.

The commands are then sent to the Game Communicator class. The two classes here are interchangeable hence the dotted box. Local Communicator simply passes data to Local User. Network Communicator uses a socket then server side network user then responds to this socket,

the Server Socket is made in the Server class which then creates Networks Users for each client (Server is not present in the diagram as after instantiation there is little to no communication).

The Responder classes interact directly with the single Game Logic class, which gives an output that the User classes interpret and relay back up the chain.

Although the seams complicated it allows code to be reused without rewriting it. As the classes take over a small responsibility they end up being relatively small.

Bots

In the diagram above bot is used to represent a generic bot. As stated in the introduction of this document I am implementing different bots which are as follows:

- **Random Bot**
  - This bot will pick up gold if it needs it and pick up the lantern if it doesn't already own it but it's movements are random except it will avoid walking into walls a players
- **Objective Bot**
  - This bot will seek out gold until it has enough to win and will then seek out the exit
  - If it cannot see its target it may try to pick up the lantern in to aid it in finding its destination
- **Aggressive Bot**
  - Has some similar functionality to the Objective Bot but will prefer to kill other players over getting gold
  - It will only attack players it sees and will always attack them.
  - It will also try to pick up a sword if it sees one and will also pick up gold and a lantern if there is no player around
- **Friendly bot**
  - This bot does not try to outright win the game
  - It prefers to help other players and bots over winning
  - It will pick up gold and attempt to give it to players
  - If there is no players in sight it will act objectively until it finds a player.

These bots have require different tools to help them some of which are common because of this it was useful to form a hierarchy of bots all of which inherit the abstract **Bot** which allows Bot Player GUI to treat the class generically. Here are the classes involved:

- **Bot**
  - Generic abstract bot that extends thread this allows a feedback ability which does not cause excessive memory depth, it also provides some randomised functions that all of the bots will use
- **Random Bot**
  - Extends Bot and Represents the Random Bot's decisions
- **Path Finding Bot**
  - An abstract class that extends bot, it provides the rather complex path finding algorithm that the all but the random bot uses
- **Objective Bot**
  - Extends Path Finding Bot and Represents the Objective Bot's decisions
- **Player Finding Bot**

    - An abstract class that extends Path Finding Bot, it extends the functionality of the path finding algorithm to allow path finding to a space next to a player
- **Aggressive Bot**
    - Extends Player Finding Bot and Represents the Aggressive Bot's decisions
- **Friendly Bot**
    - Extends Player Finding Bot and Represents the Friendly Bot's decisions

Again this promotes high code reuse with these classes having small amounts of code (except maybe the path finding bot due to the complex algorithm).

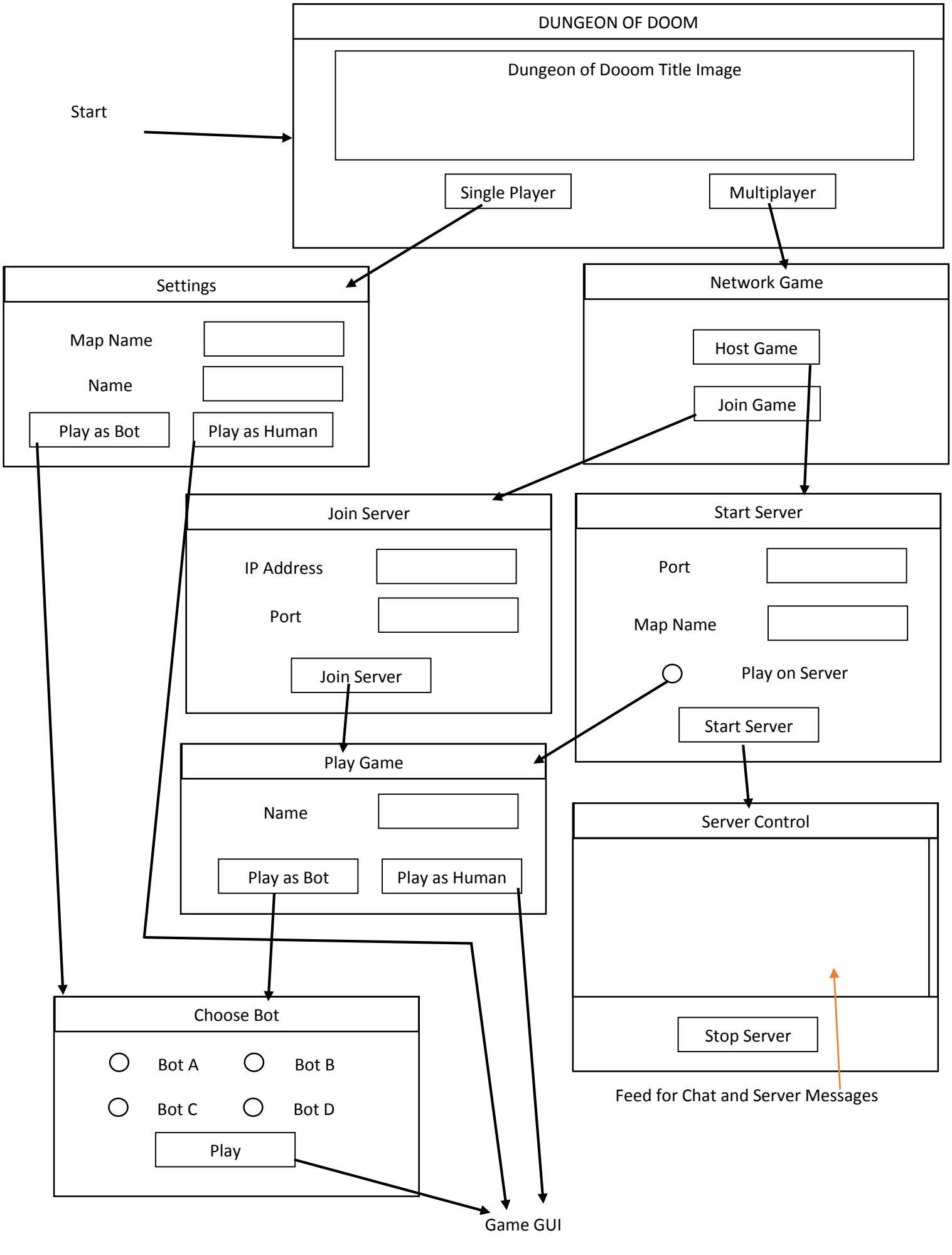
Change in the GUI

There were some changes in the GUI most of them reflecting some change in functionality. The first the mode menu has changed to include a title page to appear more visually appealing. The new GUI designs are present on the next page.

Feature Table

Here is my feature table

Intended Features	Points	Self-Evaluation	Check Evaluation	Special Instructions
Basic Client GUI	15	Yes		Bot and Human GUIs Look different so please check both
Graphical Pane A)	10	Yes		None
Respond to Server Events B)	10	Yes		None
More types of Interactions Ei)	5	Yes, I believe so		Gifting and Attacking are the interactions
Multiple Bots Eii)	5	Yes		There are four types of named bots (Baldrick, Wes, Ledeon and Alison) all but Baldrick are rather complex. Select the one to use from the bot selection menu.


DUNGEON OF DOOM
Dungeon of Dooom Title Image
Start
Single Player
Multiplayer
Settings
Map Name
Name
Play as Bot
Play as Human
Network Game
Host Game
Join Game
Join Server
IP Address
Port
Join Server
Start Server
Port
Map Name
Play on Server
Start Server
Play Game
Name
Play as Bot
Play as Human
Server Control
Stop Server
Feed for Chat and Server Messages
Choose Bot
Bot A
Bot B
Bot C
Bot D
Play
Game GUI