MUD DEVELOPMENT PLAN

By March 28th

- Create Basic Objects/ class representation of each object
- Create Basic Server that can
    - Allow clients to connect,
    - Send messages to other clients
- Create/Setup database using SQLite
    - Create dummy entries for testing
    - Connect to the server for storing user info.

By April 11th

- Add functionality for users to create accounts and join rooms
- Server should recognize basic commands(movement, look, say)
- Add combat feature
    - User can challenge another user in same room
    - Accept challenge
    - Add combat commands to server (hit, load, block)

By April 20th

- Intensive testing for all features
    - Give unexpected user input
    - Find volunteers to test our application and provide feedback
- Analyze users' feedback and improve on application where necessary
- Add extra commands
    - Help command
    - Select weapons
    - Improve character abilities(e.g collect weapons and treasure)
- Setup easy way of adding rooms to the map

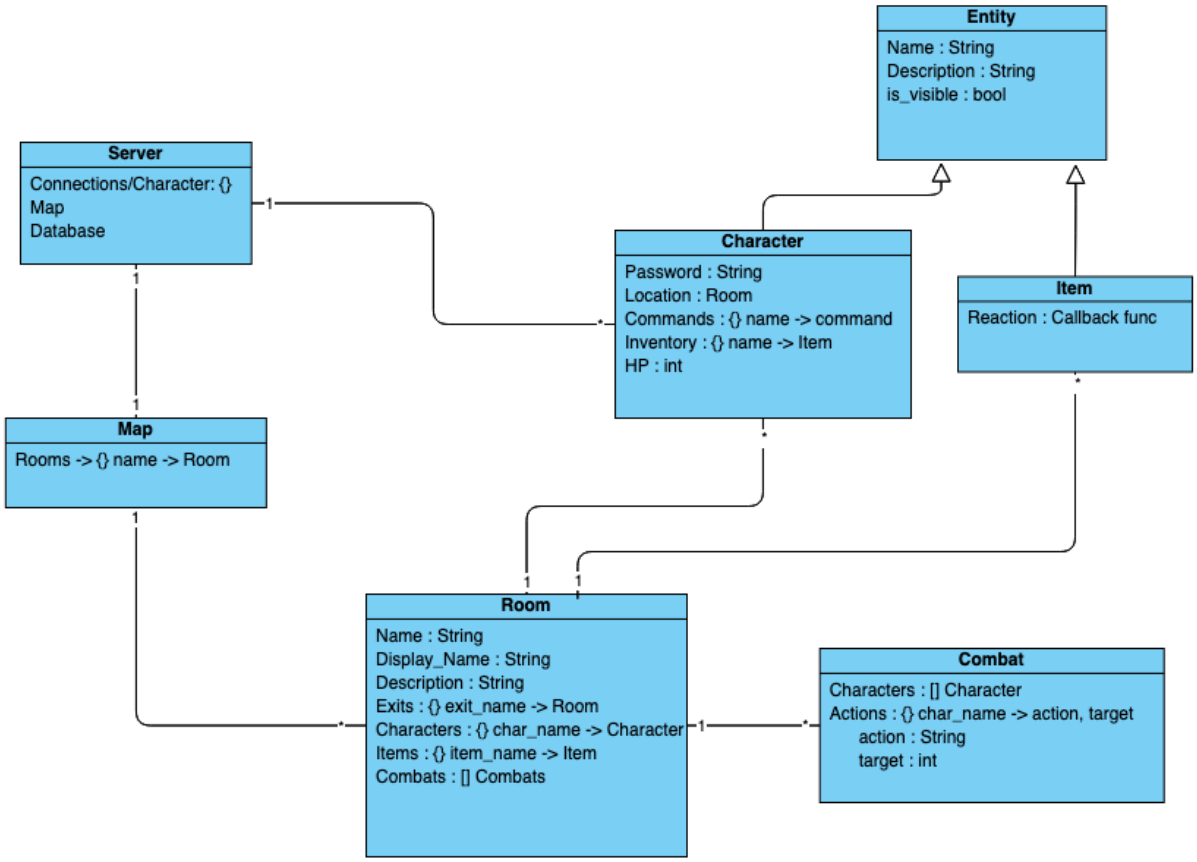
By April 23rd

- Have project presentation
- Make Application available for other users to test

By April 27th

- Finalize project final report

Class Diagram



Command

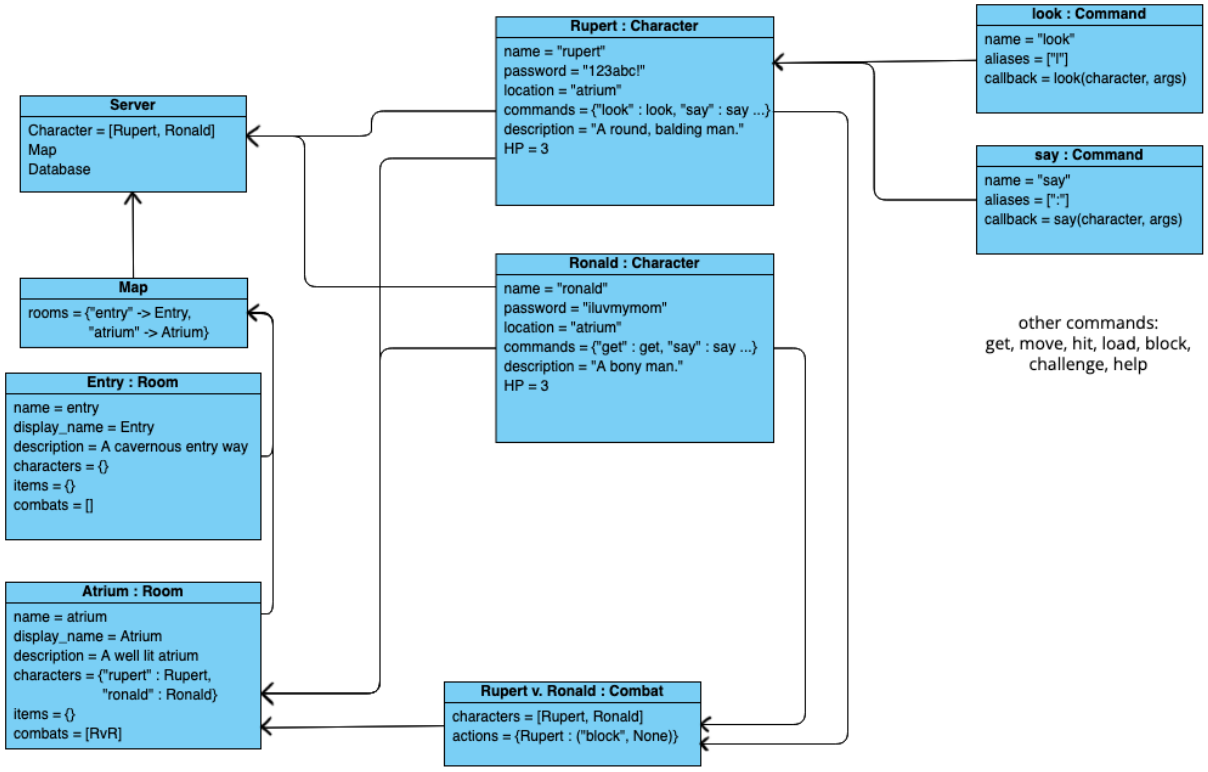
Not shown above is the command class which would be defined as follows:

Command
Name : String Aliases : String [] Callback : function(character, arguments)

Combat

In our game combat will work in timed rounds. In each round, all players involved will submit an action. At the end of the round the actions will be evaluated based on the rules for the game double-O-7: *Block* blocks an attack, *Load* loads an attack, you can only *attack* once you have loaded.

Object Diagram



The above diagram represents the state of the game with a simple map, and two connected characters. This snapshot is at the moment that the two characters are fighting.

To describe how we got to this state, we will follow Rupert's user.

Rupert's input	*Results*
Connects to the server	Prompt Rupert for their character name
> Rupert	Verify that the character exists, prompt Rupert for the password
> 123abc!	Verify that the password is correct, prompt Rupert for a starting room.
> Entry	Place Rupert in the appropriate room, add them to the list of characters connected to the server.
> north	Check that an exit north exists, remove Rupert from the Entry room, and place them in the Atrium room.

> look	Print the display name, description, and contents of Rupert's current room (atrium) to their client.
> Challenge ronald	Asks Ronald if they accept the challenge, Ronald accepts, a combat object is created and added to the atrium.
> block	In the first round of combat Rupert uses their block action. The action is added to the dictionary of actions for the round.

Design Decision

One design decision we considered was using one thread per character/user versus using one thread for multiple characters. A potential limitation of using one thread per character is a potential impact on performance due to too many threads and running out of available memory on the heap. On the other hand, using one thread for multiple characters can quickly get complicated keeping data separate and contained to each user. We ultimately decided to use one thread per user because the small scale of our MUD game and low number of users will accommodate this well. We also found a nice framework called socketserver that will help us accomplish this as well.

MUD Project Proposal

1. **What is your team name?**
   MUD

2. **Who's in the group?**
   Elise, Percy, Mico

3. **What's the project?**
   A small MUD (multi-user dungeon) style text based rpg that multiple players can connect to and play together in real time.

4. **What's the minimum/maximum deliverable?**

Minimum deliverable:

The smallest version of our project would implement only the core features of the game:

- Users should be able to connect to the game server, and log in as their character.
- There should be a map composed of connected rooms that players can walk around in.
- Players should be able to interact with objects by picking them up, looking at them, and dropping them.

- Players should be able to speak to other users in the same room as them.

Maximum deliverable:

Once the basic features are implemented, we would want to start building more mechanics, and elements on top of it. One important one we would likely want to add first is a basic combat system where players can kill monsters (or objects). We would also create some way to generate ascii maps of the world to help players navigate.

Ideally, we would like to turn this project into a code base that could be used to create a variety of different games. To do this we would need to include some in-game utilities for extending the world: commands to create and link new rooms, and define new items.

5. **What's your first step?**

The first steps will mostly be familiarizing ourselves with existing MUD games and the format/user experience they offer. After we've become comfortable as the client, we'll look at existing architectures and designs to better understand essential game functions.

6. **What's the biggest problem you foresee or question you need to answer to get started?**

Question: Evaluate resources available for Erlang vs. Python.

Networking: we need to understand telnet protocol and how it works. We also need to decide whether we use it or use a chat server instead.