COMP260 – Distributed programming	Assessed Worksheet 2
MUD Server	

Introduction

In this worksheet, you will analyse, design and architect a server application for a MUD based on an existing single user dungeon application (SUD) and a black box client and a bare-bones server framework. Unlike the chat service, the MUD client uses a simple protocol which will allow for a wide range of server-defined messages

Your challenges are to: 1) port the existing SUD so that it will support multiple clients whilst maintaining compatibility with the black box client, 2) host the MUD service on a remote server (a Digital Oceans server will be provided) and 3) extend the MUD application in any way you see fit

This worksheet assumes that the client will be developed in Python, though any suitable application framework can be considered. Please ask me for guidance before embarking with other frameworks.

Data communications

The client and server communicate using variable length packets (as discussed in workshop 1). Each packet consists of:

Size (2 bytes)	Payload (n bytes)

The single payload format is an array of bytes that form a utf-8 string, allowing the server to support a wide range of commands from users. The SUD currently supports the command 'go' <direction> and help, though it is assumed that you will develop new commands to add new functionality to the MUD service without needing to retrofit them to the client.

Service Design

The black-box client will send and receive packets as described in the data communications section. A single, but flexible, packet makes data communications fairly straightforward, but gives a lot of scope for communication richness, though this must be handled by the server.

The SUD component provides all the functionality for running a dungeon with a single user. Some consideration should be given to how the SUD maintains player status (position within the dungeon) and how this can be scaled to multiple players in a networked environment. One solution is to use each player's socket as key into a dictionary of player data, effectively replacing `dungeon.currentRoom`, however, functions that use `currentRoom` will need to be re-written to pass a player socket into them.

Likewise, consideration should be given to the placement of the SUD within the bare-bones server. Initially, the server will receive player commands in a `clientReceive` thread and add them to a message queue which dispatches the commands in the main thread. At some point in that process, the player message will need to be passed into the `SUD.Process` function and used to update the player within the dungeon. The outcome of that updating will need to be sent back to the player, via their client socket and to other players, if the need arises.

Tasks

1. Analysis

Analyse existing 'SUD" and 'bare-bones server' applications to determine the structure and functionality of those applications such that they can be combined and refactored to make a MUD service. UML analysis will comprise of class hierarchy and flowchart / state diagrams for both applications.

2. MUD as a Service

Port the SUD functionality into the bare-bones server such that it will meet the following use-cases:

i. Server can **support** multiple players in a dungeon
ii. Enable players to **navigate** multiple locations in a virtual dungeon
iii. Clients are **independent** of each other
iv. Players are **aware** of other players in the same location
v. Players in the same room can **communicate** with each other.
vi. Player activities do not **adversely affect** the performance of the MUD
vii. Players leaving the MUD will not result in zombie clients
viii. Players can **rename** their avatar

3.Value-add features

Develop the MUD service to do something interesting beyond the scope of part 2. Consider one of the following areas below. If you want to do something of your own design, please consult with me first.

Potential features:

i. PvP, players can damage & kill each other
ii. Room graffiti, players can leave messages in rooms for each other
iii. Player persistent, the server will maintain player state between player sessions
iv. Dungeon items, players can pick up, move and gift items to each other
v. Player dungeon building, players can dig out new rooms into the dungeon and give them names and descriptions
vi. NPC Baddies, baddies can interact with players

4.Operations

Package up your server application to run on the Digital Ocean server you have been provided with.

Final Submission

Combine the analysis and design documentation, and client source code into a folder and submit as part of 'Assignment 1 Computing Artefact' along with assessed worksheet 1 and your submission for the 'real-time gaming provision' part of the assignment.

Your work for both assessed worksheets and 'real-time gaming provision' parts of the assignment will be assessed in the viva session. Prior to the viva, please ensure that your MUD server is up to data and working on the DO box.

Marking Rubric – Assessed Worksheet 2: MUD Server

Learning Outcome Name	Learning Outcome Description	Criteria	Weighting	Refer for Resubmission	Adequate	Competent	Very Good	Excellent	Outstanding
Architect & Research	Integrate appropriate data structures and interoperating components into computing systems, with reference to their merits and flaws.	Analysis	30%	No analysis presented	Confusing or overtly simplistic UML diagrams	Rough UML diagrams	Fairly clear UML diagrams	Clear UML diagrams	Very clear UML diagrams
		MUD as a service	30%	Working MUD not presented	MUD meets few criteria mentioned in brief	MUD meets some criteria mentioned in brief to high and playable standards	MUD meets most criteria mentioned in brief to high and playable standards	MUD meets all criteria mentioned in brief to high and playable standards	MUD meets all criteria mentioned in brief to extremely high and playable standards
		Value-add features	20%	Working MUD not presented	Value add feature is very flaky.	Value add feature is somewhat flaky.	Value add feature is well-implemented.	Value add feature is well-implemented and adds some novelty or engagement to the MUD	Value add feature is well-implemented and adds novelty and engagement to the MUD
		Operations	20%	Working MUD not presented	MUD service is fairly flaky with multiple server restarts	MUD service works but has persistent issues	MUD service is fairly stable but has some issues	MUD service is generally stable and has only minor issues	MUD service is stable and has no issues