Software Requirements Specification

for

Untitled Dungeon Adventure

Version 1.0 approved

Prepared by Noah Herd, Christina Blackwell, Seth Campbell

University of Washington Tacoma, School of Engineering and Technology

18 April 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The following SRS is provided to give a detailed description of the currently untitled Dungeon Adventure (UDA) game. It will outline and explain the game's requirements, purpose, implementation, primary intended market, as well as all interfaces, libraries, and platforms used in its creation and use.

1.2 Document Conventions

Throughout this document, the following terminology is used interchangeably:

- Game, platform, project, product: the software system described by this document.
- User, player, gamer: users of this software system.
- Character, entity: game characters users can play as or interact with as part of gameplay.
- Developers, students, production team, development team: the team developing this software system.

1.3 Intended Audience and Reading Suggestions

This SRS is primarily intended for use by the development team to facilitate the development of this project, but is written for anyone seeking to understand the operation and development of the platform. This document is intended to be read sequentially, and sections are organized by depth of content, starting with an overview of the project before going into specifics. It is organized into six numbered sections. The first section provides an introduction to the project and this SRS, describing the purpose of the document, terminology used throughout this document, the project's scope, and a list of references this document refers to. The second section provides a general overview of the software system, including its context and origin, important features, the types of users it is relevant to, its operating environment, design and implementation constraints, user documentation that will come with the software, and a description of factors that may influence the requirements specified by this document. Section three describes specific implementation features, including playable characters, non-playable characters, game difficulty customization, game saves, and map generation. Thereafter, section four covers external interface requirements, such as user, hardware, software, and communication interfaces. Finally, sections five and six discuss nonfunctional and miscellaneous requirements related to this software system. Following that are Appendices A, B, and C, which are dedicated to a glossary of important terms used throughout this document,

analysis models, and a log of issues with the software system that are currently unresolved, respectively.

1.4 Project Scope

This software system is being created for a class project to develop students' program design and development skills. It is a dungeon adventure game where a user chooses a playable character type and is placed at the entrance of a randomly generated dungeon. The dungeon will contain features that hinder and help a player's progress. In order to successfully complete the game, the player must find four totems, known as the four pillars of Object-Oriented Programming (OOP). They are as follows: Abstraction, Encapsulation, Inheritance, and Polymorphism.

1.5 References

- ➔ Course Project Outline
- ➔ GODOT Documentation: https://docs.godotengine.org/en/stable/index.html

2. Overall Description

2.1 Product Perspective

The untitled "Dungeon Adventure" game is a single-player, 16-bit, 2-D, top-down game modeled after the classic dungeon crawler games of the late 1990s. This game is a new, self-contained product meant for short-term play (~1-2 hours of casual play) that is aimed at bringing a lighthearted feel to the genre while incorporating design elements and software requirements detailed by the Software Development and Quality Assurance Techniques course of the UW Tacoma Computer Science Program.

2.2 Product Features

The UDA, as a single-player game, will allow a user to choose their preferred difficulty and a playable character, navigate a dungeon maze, and store and reload progress. Player characters come in several classes with different starting statistics and skills, such as hit points and attack speed, that will affect the game experience. Additionally, while navigating the dungeon, players will have to avoid traps and battle monsters that try to hinder their progress. However, players will also receive boons that aid them on their quests, such as healing and vision potions. Upon completion of the game, users will be able to start a new game.

2.3 User Classes and Characteristics

The primary, and only user class the developers anticipate, are individuals looking to play a fun game with an interesting story arc. Of this user class, there are several subclasses. These subclasses can be divided by play styles. Some players prefer relaxed gameplay that does not present too much difficulty. For these users, easy or normal difficulty will be most suitable. For players who are more skilled or prefer a greater challenge, a harder difficulty level will be available. Additionally, some users may be interested in speed running, or completing the game as fast as possible. If time permits, cheats or “Easter eggs”, which are secret features the user must discover, may be implemented to address these users’ preferences.

2.4 Operating Environment

This platform is built to function on any modern computer. The development team does not anticipate any special hardware or software requirements needed to successfully run the software system. It should be able to run on top of, and without disrupting, most general ongoing processes within the Operating System (OS).

2.5 Design and Implementation Constraints

The software system has the following constraints:

1. Technologies: The program will be written using the Godot game engine and C#.
2. Timing: The program must be completed by June 13th, 2025.
3. Database (DB): The program must use a SQL database for storing game instance data for later use.
4. Assets: Assets will be used to handle the visual components of the game.
5. Design patterns: the model-view-controller (MVC) design pattern will be utilized to separate the graphical user interface (GUI) implementation from the game logic code.

2.6 User Documentation

Players will be given instruction and tutorial demonstration via a menu option within the application to cover basic controls, objectives, and a brief “About” section. If possible, the developers will implement a tutorial feature that guides new players through the game.

2.7 Assumptions and Dependencies

Dependency: Assets will need to be collected for use within the project. →
Assumption: Any assets collected and implemented will be under a free use license.

Should that licensing change, we would need to find new assets to replace pre-existing ones.

Dependency: Version control through git/GitHub. → **Assumption:** The version control tool is consistent and reliable. If our version control systems fail, a replacement is necessary.

Dependency: Youtrack software to organize sprints. → **Assumption:** The software will allow the team to reliably plan and run sprints. If it fails, a replacement is necessary.

Dependency: Godot, development engine. → **Assumption:** The version of the software remains stable throughout development. If the version becomes unstable, we can revert to a stable version to continue production.

Dependency: C# → **Assumption:** .Net libraries and frameworks are integrated and work well within the development engine. If there is an issue, there are multiple resources to use that can help solve dependency and integration issues for C# in Godot (See Section 1.5, References).

Dependency: SQL/DB equivalent to store game state and objects. → **Assumption:** Objects are stored within and accessed from the database easily. If there are challenges to the use of the databases within the development engine, an alternative DB must be implemented.

3. System Features

3.1 Playable Character (PC)

3.1.1 Description and Priority

A representation of a playable entity within the game. This should be a general framework that can be used to build other versions of an entity with specialized functions. This is the adventurer, or hero, of the game.

Priority: HIGH

3.1.2 Stimulus/Response Sequences

Stimulus: Movement is applied. → **Response**: The entity changes position.

Stimulus: The entity is struck or damaged. → **Response**: The entity has a chance to block the damage. If it fails, their hit points are reduced.

Stimulus: The entity is healed/restored (through either a healing potion or innate character skill). → **Response**: The entity's hit points are increased.

Stimulus: The entity encounters a pit. → **Response**: The entity's hit points are reduced.

Stimulus: The entity encounters a vision potion. → **Response**: The user will be able to see up to eight rooms surrounding the entity's current room.

Stimulus: The entity encounters an adversary. → **Response**: The entity has a chance to attack its opponent. The success of the attack, if it can attack, is determined by randomly selecting a value from a range of possible values assigned upon creation of the character. The higher the value, the more successful the attack is.

3.1.3 Functional Requirements

REQ-1: A name. Users will be prompted to enter a name for their PC when they start a new game.

REQ-2: Hit points: A representation of the PC's life. When this reaches 0, the PC is unable to continue exploring the dungeon.

REQ-3: The ability to move within the map provided and pass between rooms within the dungeon.

REQ-4: A method to change the current PC's hit points.

REQ-5: A way to store items. In other words, an inventory.

REQ-6: A method to interact with objects in the environment. The character must be able to pick up items, such as potions and pillars. They must also be able to take damage from pits and attack monsters.

REQ-7: A "class" that determines the character's special ability. The classes that are available for selection are the warrior, priestess, and thief classes.

REQ-7: The special ability of the character.

REQ-8: An attack speed attribute. Higher numbers indicate faster speed. This determines how quickly the entity can attack. For example, if the entity's attack speed is two times as fast as its opponent's, it can attack two times that round versus, while the opponent can only attack once.

REQ-9: The entity must have the ability to block attacks. The success of the block depends on the class of the entity.

REQ-10: A damage range attribute, from a minimum value to maximum value, inclusive. When attacking, a value from this range will be randomly selected, which determines how many hit points' worth of damage will be dealt to the opponent.

3.1.4 Playable Character Specifics

As described previously, playable characters are divided into classes: the warrior, the priest/priestess, and the thief. The table below details their unique characteristics.

Class	Special Skill	Statistics
Warrior	Crushing Blow: This move does 75 to 175 points of damage, but only has a 40% chance of working.	Hit points: 125 Attack speed: 4 Hit chance: 80% Minimum damage: 35 Maximum damage: 60 Chance to block: 20%
Priest(-ess)	Heal: This move heals some range of the entity's hit points (the amount is TBD).	Hit points: 75 Attack speed: 5 Hit chance: 70% Minimum damage: 25 Maximum damage: 45 Chance to block: 30%
Thief	Steal: This move has a 40% chance of being successful. If it is successful, the thief gets an extra attack added to the current round. If it fails, there is a 20% chance they are caught and no attack is performed, and another 40% chance that they perform their normal attack with no benefit.	Hit points: 75 Attack speed: 6 Hit chance: 80% Minimum damage: 20 Maximum damage: 40 Chance to block: 40%

3.2 Non-Playable Character (NPC)

3.2.1 Description and Priority

A representation of a variety of NPCs within the game to include common and unique entities. These entities pose danger to the PC and must be defeated to progress the game. These entities are referred to as monsters within the game.

Priority: HIGH

3.2.2 Stimulus/Response Sequences

Stimulus: Movement is applied. → **Response**: The entity changes position.

Stimulus: The entity is struck or damaged. → **Response**: The entity's hit points are reduced.

Stimulus: The entity is healed/restored. → **Response**: The entity's hit points are increased.

Stimulus: The entity encounters an adversary. → **Response**: The entity has a chance to attack its opponent. The success of the attack, if it can attack, is determined by randomly selecting a value from a range of possible values assigned upon creation of the character.

Stimulus: The entity's health is reduced to zero. → **Response:** The entity is defeated. If it has any items in its inventory, they are added to the player's inventory.

3.2.3 Functional Requirements

REQ-1: The amount of enemies that exist in an area must depend on game instance difficulty and type of enemy. NPC data must be stored in a database and used to populate the map with monsters when a game is started.

REQ-2: Hit points: A representation of the NPC's life.

REQ-3: Movement is based on NPC type.

REQ-4: A method to change the current NPC's hit points. The entity has a random chance to heal after taking damage.

REQ-5: An attack speed attribute. This determines how quickly the entity can attack. For example, if the entity's attack speed two times as fast as its opponent's, it can attack two times that round versus, while the opponent can only attack once.

REQ-6: An ability to interact with the environment - specifically, the ability to duel with playable characters.

REQ-7: Randomly assigned inventory that PCs can loot upon the entity's defeat.

REQ-8: The ability to be stunned if dealt too much damage at once. The threshold required to trigger this status effect is TBD. If they are stunned, they cannot heal or attack.

REQ-9: A damage range attribute, from a minimum value to maximum value, inclusive. When attacking, a value from this range will be randomly selected, which determines how many hit points' worth of damage will be dealt to the opponent.

3.2.4 Non-Playable Character Specifics

Much like PCs, NPCs come in several categories. Monsters that guard exits and important objects, such as the Pillars of OO, are stronger and rarer. The type and number of these entities is randomly generated. There is a chance that some rooms may contain a mob, or a group of monsters.

Class	Special Skill	Statistics
Ogre	Roar: Specifics TBD.	Hit points: 200 Attack speed: 2 Hit chance: 60% Minimum damage: 30 Maximum damage: 60 Chance to heal: 10% Minimum heal points: 30 Maximum heal points: 60
Goblin	Shank: Specifics TBD.	Hit points: 70 Attack speed: 5 Hit chance: 80% Minimum damage: 15 Maximum damage: 30 Chance to heal: 40% Minimum heal points: 20 Maximum heal points: 40
Skeleton	Shamble: Specifics TBD.	Hit points: 100 Attack speed: 3 Hit chance: 80% Minimum damage: 30 Maximum damage: 50 Chance to heal: 30% Minimum heal points: 30 Maximum heal points: 50
Tom	Educate: Specifics TBD.	Increased attack damage. Other specifics TBD.

3.3 Customized Gameplay

3.3.1 Description and Priority

This feature will provide players a way to adjust the difficulty of gameplay at the initialization of an instance of a game.

Priority: HIGH

3.3.2 Stimulus/Response Sequences

Stimulus: The user selects easy difficulty. → **Response:** Enemies have less health points, their attack damage is reduced by 15%, and all healing actions performed by the PC restore 15% more hit points.

Stimulus: The user selects normal difficulty. → **Response:** Default values are used for enemy health points, attack damage, and hit points restored by healing.

Stimulus: The user selects hard difficulty. → **Response:** Enemies have more health points, higher attack speed, and attack damage is increased by 15%. All healing actions performed by the PC restore 15% less hit points.

3.3.3 Functional Requirements

REQ-1: Appropriate parameters adjusted based on difficulty factor (i.e., easy, normal, hard).

REQ-2: Difficulty choice must be tied to game save instance data.

3.4 Game Save Functionality

3.4.1 Description and Priority

This functionality provides players a way to save an instance of the game to play later, and a method to load previously saved data to resume gameplay.

Priority: MEDIUM

3.4.2 Stimulus/Response Sequences

Stimulus: The user selects “Save” from the Options Menu. → **Response:** The current instance of the game is saved, allowing the player to stop the application safely and resume gameplay at a later date.

Stimulus: The user selects “Load” from the Main Menu. → **Response:** The selected instance of the game is loaded and users are able to resume the game where they left off.

3.4.3 Functional Requirements

REQ-1: The selected difficulty level must be saved.

REQ-2: PC current statistics, to include character name, class, attributes, and inventory, must be saved.

REQ-3: Upon resuming the saved game, the system must load the instance of the game with PC in the location they were in when the game was saved, with appropriate bosses and items still available.

3.5 Map generation

3.5.1 Description and Priority

A feature to randomly generate an instance of a map that contains all necessary objectives and can be successfully navigated. The map will also contain a single entrance and exit, and several dead-end routes that do not lead to the exit.

Priority: HIGH

3.5.2 Stimulus/Response Sequences

Stimulus: The user has selected a class and attempts to start the game. → **Response:** A map layout is generated and stored for use within the game.

Stimulus: A room is generated. → **Response:** A room type is chosen, and the room is generated/selected to be placed into the map layout.

Stimulus: A room is placed into the map layout. → **Response:** The room is populated with NPCs and items the player can use.

3.5.3 Functional Requirements

REQ-1: A complete map is generated and can be successfully navigated.

REQ-2: The map is populated with the necessary objective items. None of the four pillars can be generated within the same room, and no more than four pillars are generated within one instance of the game.

REQ-3: The map is populated with a random assortment of objects that will help and hinder the player.

REQ-4: Rooms are generated and appropriately placed within the map layout so they are accessible by the player. This includes the entrance and exit rooms, which are to be empty, except for a boss guarding the exit. The exit should not be available before the player has collected all the pillars of OO.

REQ-6: NPCs are appropriately placed in rooms. Rooms with important items will trigger stronger and rarer NPCs to spawn.

4. External Interface Requirements

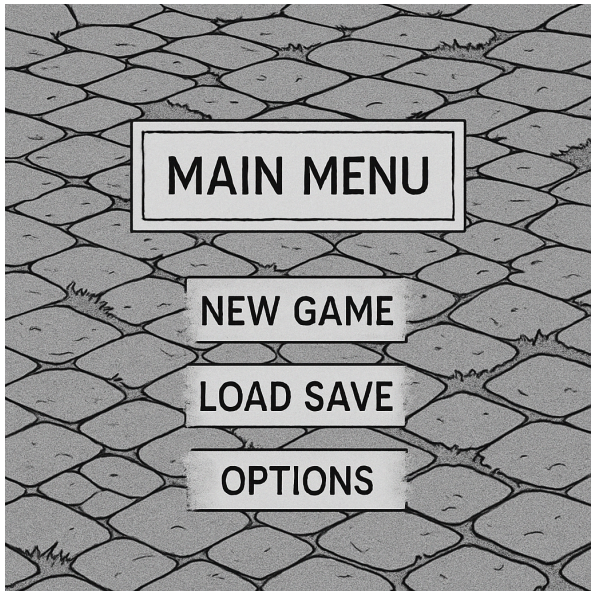
4.1 User Interfaces

This software system will provide a GUI that players can interact with. The main window will display a top-down view of the PC as they navigate through the maze. Only the current room will be displayed unless the PC encounters a vision potion. Upon successfully completing the game, the entire map will be displayed.

Upon starting a new game, the application will display a customization screen where users can name their character and select what class to play as. If the user pauses the game, they will be presented with a menu that allows them to save and quit the game. As the player navigates the dungeon and performs actions, relevant information (such as items collected or damaged taken) will be relayed through text pop-ups. Other pertinent graphical features are described below.

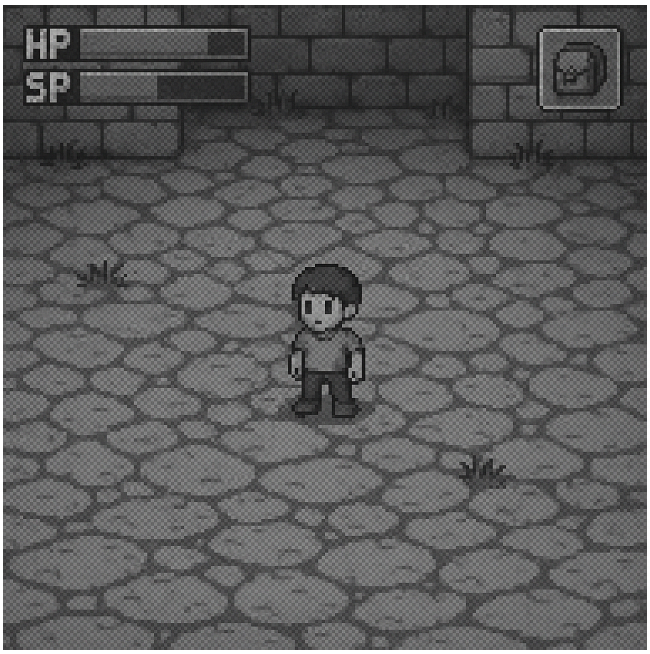
4.1.1 Main Menu

The main menu allows the player to start a game and configure the difficulty settings to alter enemy health and damage by pressing the “Options” button. The player can also load a previously saved game to continue playing from a fixed point. Additionally, a “Help” button will be implemented that provides information on how to play the game when selected, as well as an “About” button that brings up game details information (e.g., version history, developer names, copyright notices)



4.1.2 Concept of Gameplay

Players can see their health and inventory and have a field of vision that shows a space around the player. If something within a room exists outside the player's vision, it will not be rendered to the player.



4.1.3 Game Over Screen

If the player's health points are reduced to zero, a menu screen appears that allows the player to exit the game or to start a new one.

4.1.4 Inventory Management Screen

Displays the player's current inventory. Contains images/icons depicting the items within the inventory. Contains a display of the current objectives collected and the ones remaining to be collected.

4.1.5 Game Completion Screen

Displays a “thank you” message for playing the game and a short message detailing what happened after the player beat the game.

4.1.6 Style Guide

All art assets will be constrained to 16 by 16 pixels.

4.1.7 Style Genre

Currently, the developers have not decided on a style genre.

4.2 Hardware Interfaces

Users will interact with the game by using a keyboard and mouse. The keyboard will be used primarily for gameplay and utilize traditional WASD controls for movement. And surrounding keys for select actions. The mouse will be for the application's pre-game selection and setup, as well as in-game option selection.

4.3 Software Interfaces

The UDA platform will utilize a SQLite database for storing monster data. The application will pull from this database when it populates monsters within the dungeon.

4.4 Communications Interfaces

This software system does not require any communication interfaces as it is a single player game.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The game needs to run at standard and a consistent refresh rate. All in game interfaces should be reasonably responsive. Player input should have little to no input delay.

5.2 Safety Requirements

This game has violent themes, which may be unsuitable for some players.

5.3 Security Requirements

The program needs to run locally on any of the major OS without any external libraries, services, or connections.

5.4 Software Quality Attributes

The UDA should be reliable in its usage. It should be able to run successfully from start to finish, with no significant progress hindering bugs in the process.

6. Other Requirements

At this time, the development team has not identified any other requirements.

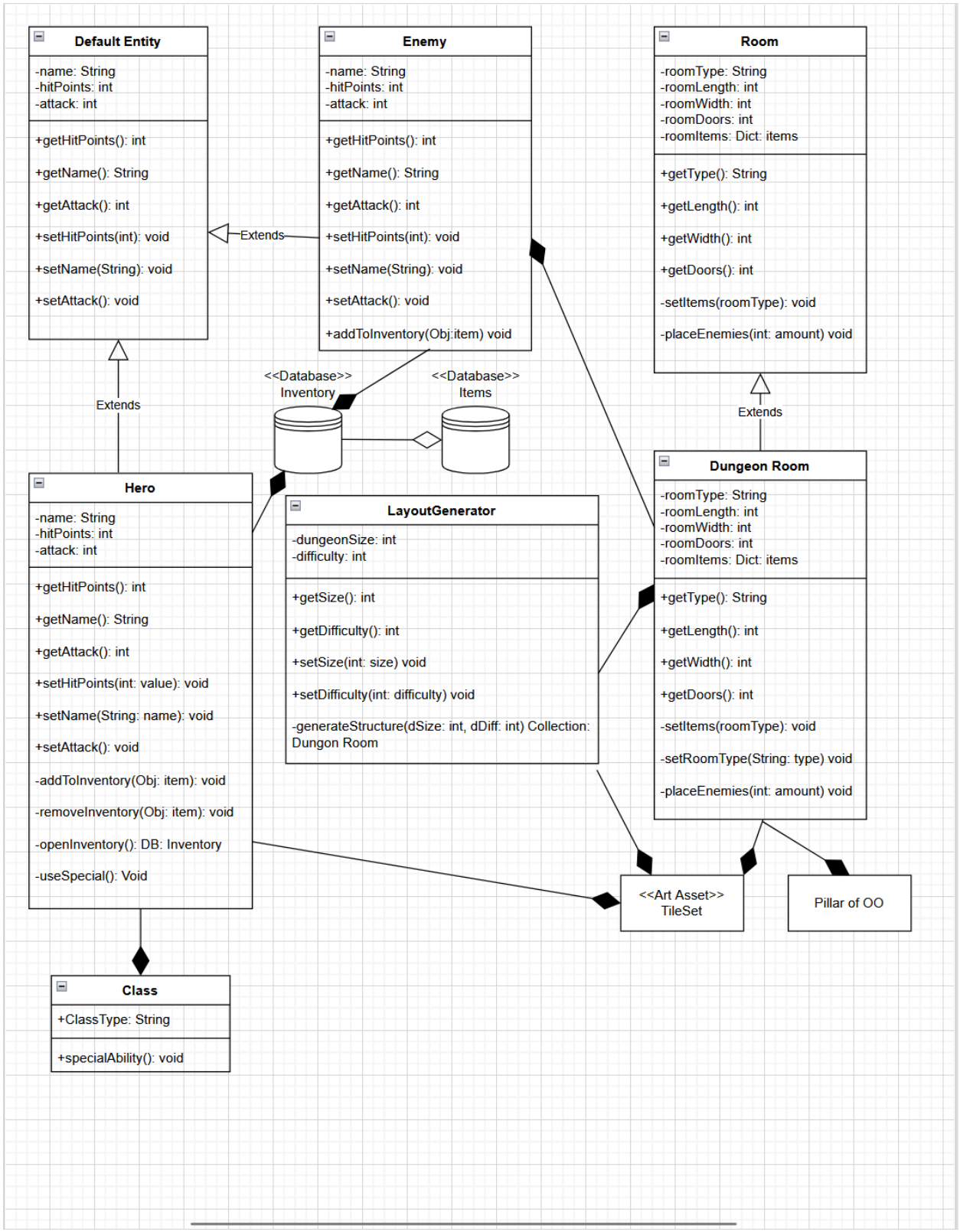
Appendix A: Glossary

Term (Acronym)	Definition
Untitled “Dungeon Adventure” (UDA)	Placeholder for the application's name until it is determined.
Entity / Entities	Encompassing term for playable and non-playable characters.
Playable Character (PC)	Character entity the user can operate.
Non-Playable Character (NPC)	All other character entities that the user cannot operate direction but can engage with.
Boss	NPC characters with single unique instances.
Teaching assistants (TA)	NPC characters with multiple instances but carry unique functions.
Database (DB)	SQL database for storing game save data/ objects.

Term (Acronym)	Definition
Operating system (OS)	The software that supports a computer’s basic functions.
GitHub(Git)	Systems that track the versions of software in development.
YouTrack	A planning software to plan work and workflow.
Godot / Development engine	A creation tool for software development.
Sprints	Time period where the dev team works to complete specified sections of the project.
Stimulus	An action performed by a user/method of the software.
Model View Controller (MVC)	Design pattern for organizing a program's logic into three connected components.

Term (Acronym)	Definition
Operating system (OS)	The software that supports a computer’s basic functions.
GitHub(Git)	Systems that track the versions of software in development.
Dungeon/Room	Space that the player and NPCs occupy and move in. Can contain traps and objects required for progression.

Appendix B: Analysis Models



Appendix C: Issues List

TBD