Software Requirements and Specifications(SRS)

for

Dungeon Adventure

Version 2.0

Prepared by

James Godwin

Maddy Whitney

Jannine G. D. MacGormain

Group 3 - Team Awesome

March 08, 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Group 3 – Team Awesome	02/03/2025	Initial Draft	Version 1
Group 3 – Team Awesome	03/08/2025	Final Draft	Version 2

1. Introduction

1.1 Purpose

This SRS outlines the continuity and enhancement of the Dungeon Adventure game project from TCSS 502. It is intended for the project team members responsible for implementing and verifying the correct functioning of version 2.0 of the Dungeon Adventure game.

The implementation will adhere to object-oriented programming principles in Python. The project aims to improve gameplay dynamics, apply the MVC (Model-View-Controller) and Factory design patterns, and incorporate an inheritance hierarchy for characters and items. It will utilize database management and pickling for data handling, implement version control using Git and GitHub, and plan tasks using YouTrack Project Management—all as part of the TCSS 504 course project.

1.2 Document Conventions

This document adheres to standard Software Requirements and Specifications (SRS) conventions, including:

- Headings: Used to organize content clearly and hierarchically.
- Bullet Points: Employed for lists to enhance readability.
- Tables: Used to present information in an organized manner.
- Figures: Illustrate concepts or designs visually.
- Terminology: Specific terminology is consistently applied for clarity and precision.

Additionally, priorities for requirements are indicated in the functional requirements sections to help stakeholders understand the importance and urgency of each requirement.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers, project managers, testers, and documentation writers. It is suggested that readers begin with the Introduction, followed by the Overall Description, and then delve into the System Features, External Interface Requirements, Other Nonfunctional Requirements, and Other Requirements as presented in Figure 1.

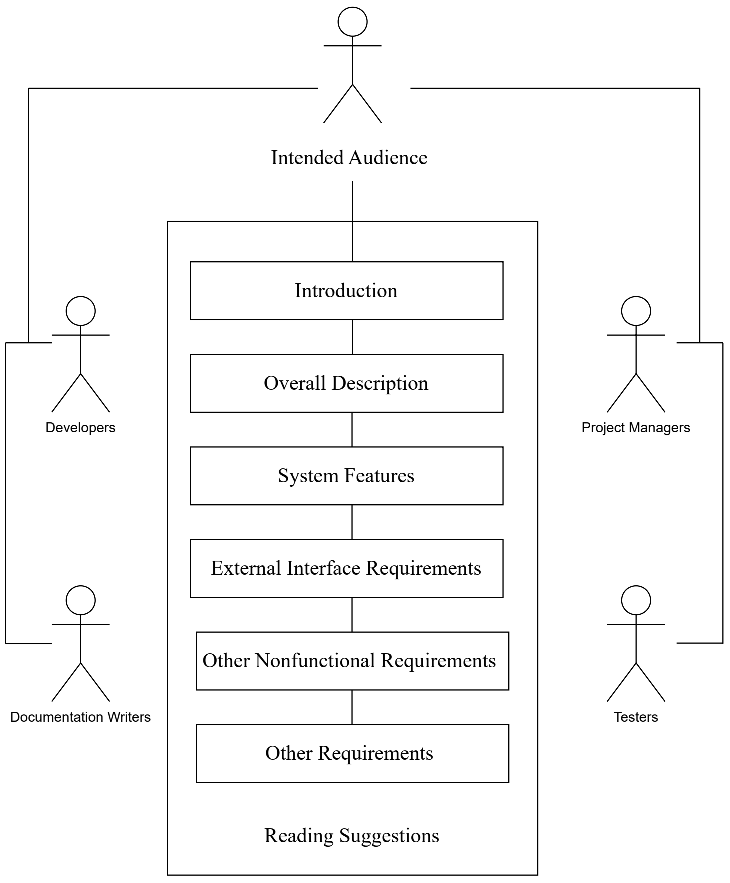

Figure 1 - Intended Audience and Reading Suggestions

This sequence will help readers gain a comprehensive understanding of the project and its requirements.

1.4 Project Scope

This project will expand the existing Dungeon Adventure game from last quarter's course, TCSS 502, by incorporating an inheritance hierarchy for dungeon characters and items, implementing Model-View-Controller (MVC) and Factory design patterns, and enhancing character data management using the SQLite database engine.

Key Features Include as Illustrated in Table 1:

MVC Design Pattern Compliance	
Model - Contains business logic and data.	
Class:	
Room	Handles the storage of items with respect to their symbols implemented in the string representation ["i", "e", "H", "V", "O", "X", "M", "A", "E", "I", "P"]. Each symbol has its own description. The Room class imports the data from the items' respective module file class.
Dungeon	Dungeon 5x5 Grid: Generates multiple rooms in a maze layout.
DungeonCharacter	Character Creation: Utilizes the "DungeonCharacter" abstract class.

MVC Design Pattern Compliance	
MODEL - contains business logic and data.	
Class:	
Hero and Monster	Inheritance Hierarchy: Derived from the Hero and Monster abstract classes to their respective subclasses. Characters Include: Hero: Warrior, Priestess, Thief Monster: Ogre, Gremlin, Skeleton, Mind Leech, Final Boss
Item	Item Creation: Utilizes the “Item” abstract class.
Pillar and Potion	Items Categorization: Pillar Class: A - Abstraction, E - Encapsulation, I - Inheritance, P – Polymorphism Potion Class: H - Healing, V - Vision, A - Agility, M - Medicine, P - Poison

MVC Design Pattern Compliance	
VIEW: Renders the MODEL into a form suitable for interaction, typically a user interface element. Multiple views can exist for a single model for different purposes.	
Class:	
GameView	It does not hold business logic; it only collects and displays data using Pythonic input function syntax.
DungeonAdventureGUI	DungeonAdventureGUI Class: (Standby and one of the pending decisions) The GUI is optimal for viewing as it contains windows, panels, frames, and buttons for real-time player graphical interaction.
Monster Images	
MVC Design Pattern Compliance	
CONTROLLER: Passes user input from VIEW to MODEL as necessary – model changes its state and notifies the view. Responsible for calling input methods from the GameView Class and processing the results, maintaining the separation of concerns.	
Class:	
GameController	Controls the game logic and flow.
MockBattle	Simulates a battle between a hero and a monster.
Battle	Handles the battle logic between the hero and the monster.
Fake Controller	

Factory Design Pattern Compliance Separates the instantiation logic from the objects themselves.	
1. Pillar Factory 2. Monster Factory 3. Potion Factory 4. Room Contents Factory	
SQLite – Standard Query Language Lite - A DATABASE engine recognized as the number one relational database. - Widely used for storing and managing data.	
Class:	
DatabaseManager	Manages the database connection for the game. Supporting files include: 1. db_test.py 2. dungeon_game.sql
Pickling (A.K.A. Serialization) The ability to save information about an object in memory to a file.	
Class:	
Pickler	Handles saving and loading the game state using pickle.

Table 1 – Key Features

1.5 References

- Course Project guidelines and outline prepared by Professor Tom Capaul.

- Team in-person discussions and discussions via Discord, Zoom and Google meet and collaborative task progress tracking through the GitHub repository and YouTrack project management.

- TCSS 502 Assignment 5: Putting it All Together - Dungeon Adventure!

- TCSS 504 Course Project: Trivia Maze, Dungeon Adventure 2.0, or File Watcher.

- [SRS Templates] TCSS 504 Software Requirements and Specifications (SRS) Assignment.

- Zoom lecture recordings and class modules prepared by Professor Tom Capaul.

- Git and GitHub Documentation.

- YouTrack Documentation.

- Class modules by Mr. Kevin Anderson.

- Class modules by Professor Varik Hoang.

- Class modules by Professor Robert Cordingly.

- Getting Started with Python by Fabrizio Romano et al.

- [Python Enhancement Proposal (PEP) process. GitHub public domain.] https://github.com/python/peps/tree/main

- https://app.diagrams.net

- https://emojicombos.com/dot-art-generator

2. Overall Description

2.1 Product Perspective

The Dungeon Adventure 2.0 game will be a standalone program developed as a course project.

It will feature the MVC and Factory Design Patterns. The GameController class will call user inputs from the GameView class for player interaction and will be implemented using object-oriented programming principles.

2.2 Product Features

The product will include the following key features:

- Character management
- Dungeon exploration
- Item Categorization
- Collection of items and interactions with game mechanics
- Usage of items.
- Data Management
- Project Management
- Version Control

2.3 User Classes and Characteristics

Players: Individuals who interact with the game, choosing heroes and battling monsters.

Players will input their name, chooses a hero type and navigate through the dungeon to collect items and avoid dangers.

Game Administrators: Users responsible for managing game data and configurations.

2.4 Operating Environment

OE-1: The software will run on Windows, macOS, and Linux operating systems with a minimum requirement of 4 GB RAM.

OE-2: The software will run on a modern CPU.

OE-3: The software will require a graphics card capable of rendering basic 2D graphics.

2.5 Design and Implementation Constraints

CO-1: Compliance with object-oriented design principles.

CO-2: Use of the MVC design pattern.

CO-3: Use of the Factory design pattern.

CO-4: Use of database management and pickling for data handling.

CO-5: Version control using Git and GitHub.

CO-6: Project management using YouTrack.

2.6 User Documentation

UD-1: Comprehensive user documentation will be provided, including installation instructions, gameplay tutorials, and troubleshooting guides.

2.7 Assumptions and Dependencies

AS-1: Users have basic knowledge of gaming and computer operations.

DE-1: The project relies on third-party libraries for data management (SQLite).

3. System Features

3.1 Game play

3.1.1 Description

The gameplay experience in the Dungeon Adventure 2.0 game includes exploring a dungeon filled with multiple rooms, each presenting unique challenges.

Each room may contain randomly generated item(s) and present threats, such as monsters and pits.

Player chooses a hero type for a battle advantage, as each hero comes with unique features (e.g., special skills, damage, etc.).

Player can navigate the dungeon maze using directional inputs (N, E, W, S) to collect various items.

Player uses the Healing Potion to restore hit points.

Player uses the Vision Potion to display the dungeon map, revealing Player's current location and surroundings.

Only one of the other potions, with effects for Agility, Medicine, and Poison, is randomly present in the room.

Player must be prepared to face dangers such as pits (X), where Player may fall and take damage, and monsters (M) that must be defeated.

Player must defeat a monster before collecting each pillar (A, E, I, P).

Finally, Player faces the boss monster to exit the dungeon. Player's objective is to collect all four pillars of Object-Oriented Programming (Abstraction, Encapsulation, Inheritance, and Polymorphism), find the exit, and defeat the boss monster guarding it to win the game.

If Player's hit points reach 0, it will result in Game Over!

3.1.2 Game Mechanics

Game Introduction:	The player will be welcomed with a game introduction display, explaining the objectives and game mechanics.
Name Entry:	The player enters a name.
Hero Selection:	The player chooses a hero type for a battle advantage. Each hero has unique features (e.g., special skills, damage, etc.).
Player Exploration:	The player will explore a dungeon with multiple rooms. Each room may contain randomly generated item(s) and present threats, such as monsters and pits.
Player Movement:	The player can move around the dungeon maze using directional inputs (N, E, W, S).
UI updates:	After any Player action, the UI updates to reflect the new room's details and the Player's status.
Item Collection:	
Pillar:	The player collects pillars (i.e., A, E, I, P).
Healing Potion:	The player collects healing potions.
Vision Potion:	The player collects vision potions.
Item Usage:	
Healing Potion:	Restores Player's hit points.
Vision Potion:	Displays the dungeon map and reveals the player's current location and surroundings.
Agility:	The player picks up an Agility potion and dodges an attack from a monster.

Item Usage:	
Medicine:	The player uses Medicine to restore HP from poison inflicted by a monster.
Poison:	A monster inflicts poison on the player, causing damage.
Dangers:	
Pit:	The player falls into a pit and takes damage.
Monster:	The player fights a monster and a boss monster. Each monster has unique features (i.e., damage, chance to heal, etc.).
Combat:	The player must defeat a monster before collecting each pillar found (i.e., A, E, I, P). The player faces the boss monster and must defeat it to exit the dungeon.
Save Game:	The player can choose to start a new game or continue the player’s saved game from where the player left off.
Quit Game:	The player can quit the game at any time.
Winning Conditions:	The player must collect all four pillars of Object-Oriented Programming (Abstraction, Encapsulation, Inheritance, and Polymorphism), find the exit, and defeat the boss monster to win the game.
Losing Conditions:	If the player's hit points reach 0, it will result in Game Over!

3.1.3 Game Initialization

Maze generation:	The game initializes a dungeon.
Character:	The “DungeonCharacter” class is instantiated with the player’s name and a reference to the dungeon.
Rooms:	The entrance and exit rooms are randomly generated, along with the placement of items, pillars, and monsters.

3.1.4 User Interface Components

Player’s Stats:	Hit points, healing potions, vision potions, and pillars found are displayed.
Rooms:	The current room's details are shown, including its contents.
Directional Inputs:	Directional inputs are available for movement (North, South, East, West).
Player Action Inputs:	Player action inputs are available for moving, battling, using potions, quitting, attacking, using special skills, and quitting battles.

3.1.5 Stimulus/Response Sequences

Players initiate actions through directional inputs (N, E, W, S), and the console responds by updating the UI and game state accordingly.

3.1.6 Functional Requirements

REQ-1: The system shall allow players to move between rooms.

REQ-2: The system shall update the UI to reflect the player's current status.

3.2 Create, View, Modify, and Delete

This feature allows game administrators to manage the in-game menu of available items and rooms. Administrators can add or modify room attributes, such as whether a room has an entrance or contains specific items.

4. External Interface Requirements

4.1 User Interfaces

The game will feature a user-friendly console interface that allows players to navigate easily and move around the dungeon maze using directional inputs (N, E, W, S).

UI-1:	Running the GameController program will prompt the player to enter their name.
UI-2:	The game introduction and welcome will be displayed, explaining the objectives and game mechanics.
UI-3:	The player will be prompted to select a hero type as well.
UI-4:	Upon entering their name and selecting the hero type, the Dungeon Adventure begins.

4.2 Hardware Interfaces

For player commands and interactions, the game will utilize standard hardware interfaces, including:

HI-1:	Monitor (Laptop or desktop)
HI-2:	Keyboard
HI-3:	Mouse

4.3 Software Interfaces

SI-1: Python 3.x or higher (preferably the latest stable version) is required.

SI-2: SQLite 3.x or higher (preferably the latest stable version) is required.

4.4 Communications Interfaces

CI-1: The game will not require external communications but will support local multiplayer functionality in future versions.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

PE-1: The game should load within 5 seconds and maintain a minimum frame rate of 30 FPS during gameplay.

5.2 Safety Requirements

The game will ensure that user data is securely stored and that the application does not crash unexpectedly.

5.3 Security Requirements

SE-1: User data will be encrypted, and the application will comply with standard security practices to prevent unauthorized access.

5.4 Software Quality Attributes

The game will focus on usability, reliability, and maintainability, ensuring a smooth gaming experience for users.

6. Other Requirements

No additional requirements have been identified at this time.

Appendix A: Glossary

Developers	Members of Group 3 - Team Awesome who are responsible for designing, coding, and implementing software applications.
Documentation Writers	Developers responsible for creating and maintaining documentation related to software products, including user manuals, technical guides, and other materials that help the intended audience understand the features and functionalities of the software.
Factory Design Pattern	Separates the instantiation logic from the objects themselves.
Game Administrators	Users responsible for managing game data and configurations.
Hero	The main character controlled by the player.
Pickling	A.K.A. Serialization The ability to save information about an object in memory to a file.
Players	Individuals who interact with the game, choosing heroes and battling monsters.
Project Managers	Professionals (often referred to as Professors) who oversee projects from inception to completion, ensuring that goals are met within the desired timelines.
SQLite	Standard Query Language Lite a DATABASE engine recognized as the number one relational database. It is widely used for storing and managing data.
Stakeholders	The developers, project managers, testers, and documentation writers who oversee the development of the project.

Testers	Quality assurance professionals and developers who evaluate software applications to identify defects and ensure that products meet specified requirements.
Users	The players and game administrators of the software.
UML class diagram	A Unified Modeling Language (UML) class diagram that represents the class hierarchy of characters and items, relationships, and interactions within the game.
MVC	Model-View-Controller (MVC) design pattern. Model - Contains business logic and data. View - renders the MODEL into a form suitable for interaction, typically a user interface element. Multiple views can exist for a single model for different purposes. Controller - passes user input from VIEW to MODEL as necessary – model changes its state and notifies the view. Responsible for calling input methods from the GameView Class and processing the results, maintaining the separation of concerns.

Table 2 – Glossary

Appendix B: Analysis Models

This section will include UML diagrams to represent the class hierarchy of characters, relationships, and interactions within the game.

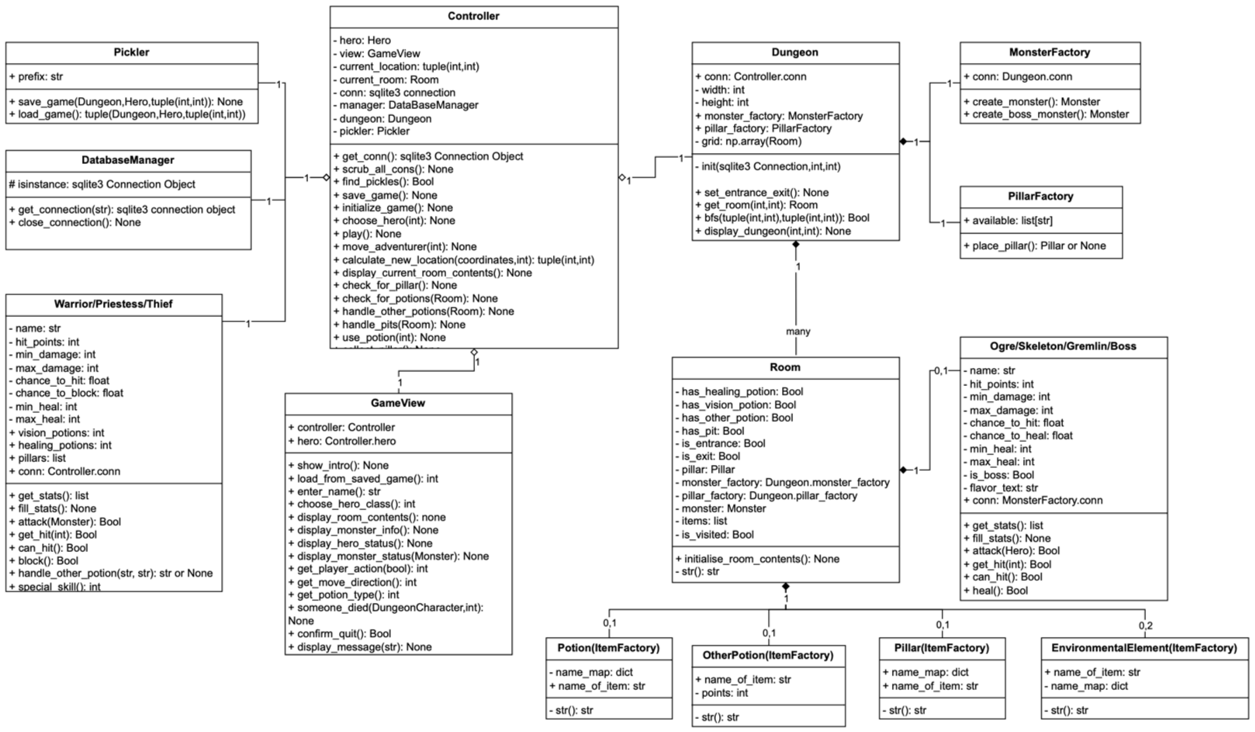

Figure 2a – UML Class Diagram

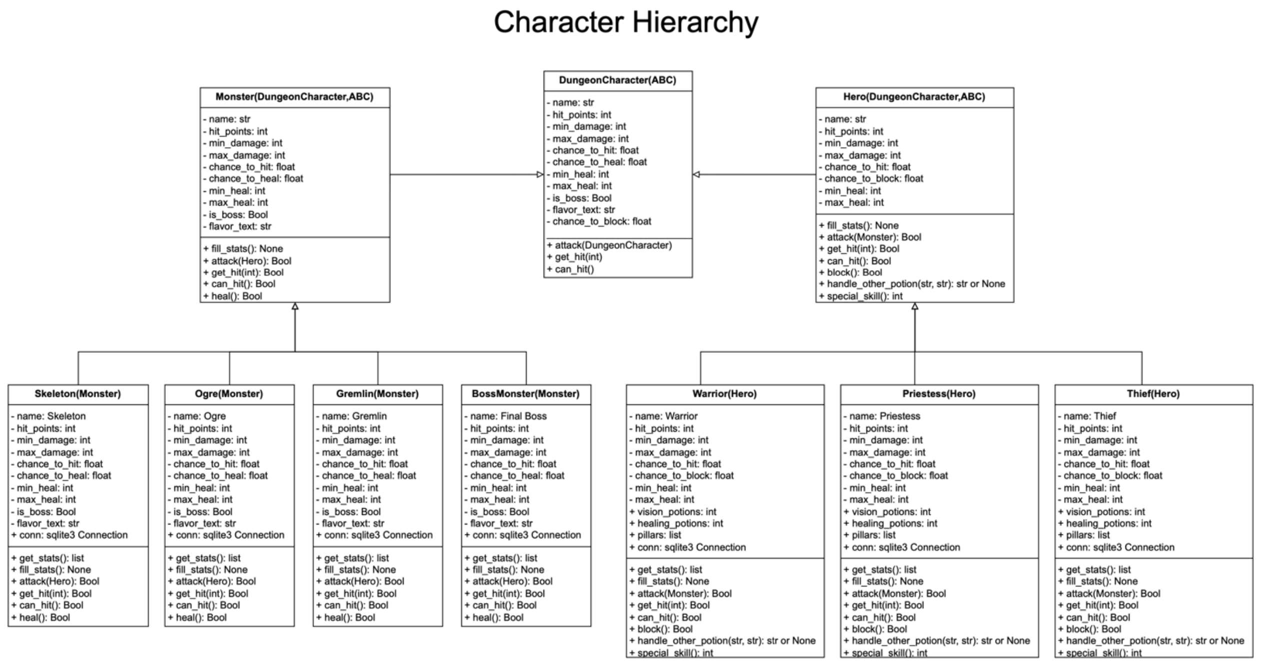

Figure 2b – UML Class Diagram

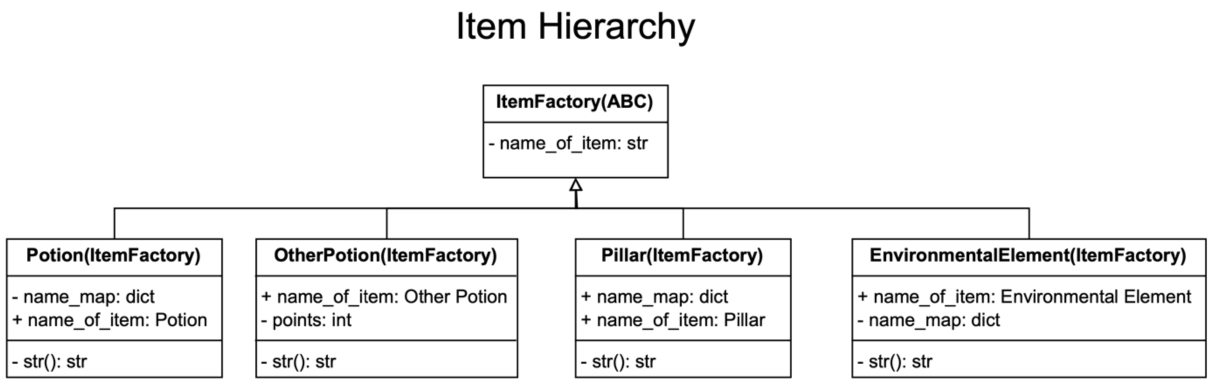

Figure 2c – UML Class Diagram

Appendix C: Issues List

Information Needed:	Clarification on the specific performance metrics desired by stakeholders.
Pending Decision:	The choice of additional third-party libraries for enhanced graphics (e.g., for a standby GUI).
	Outstanding decisions regarding the implementation of multiplayer functionality.