Software Requirements and Specifications for Dungeon Adventure

Version 1.0 approved

Prepared by Alexander Boudreaux, Joshua Barbee, and Tinh Diep

UW Tacoma School of Engineering and Technology

TCSS 360 A Fall Quarter — Team 3

16 December 2022

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to provide a detailed description of the functional and non-functional requirements of the project in order to aid the Team in the design and development of the application. This document is also intended to communicate the goals and details of the project to the Course's professor and any other interested parties.

1.2. Scope

This software will create and allow players to move through and interact with 2D digital dungeons with multiple floors. The application will include interactive gameplay with exploration of the dungeon and with turn-based combat with various monsters, which should be defined by a table in an SQLite database.

1.3. Glossary

Term	**Definition**
Course	TCSS 360 A Fall Quarter
Team	TCSS 360 A Fall Quarter — Team 3 (Alexander Boudreaux, Joshua Barbee, and Tinh Diep)
Game	Program containing information about a dungeon and managing the interactions of the dungeon's contents
Player	User interacting with the UI to play the Game
Adventurer	Player character
Monster	Any character other than Adventurer; fights the Adventurer
Game Instance	A particular dungeon and collection of characters that can be run by the Game

1.4. Document Conventions

The priorities of different functional requirements are specified in their headings with one to three asterisks as follows:

- *** - Highest priority
- ** - Medium priority

- *	- Low priority

Higher-priority functional requirements should function and be tested (though not necessarily in their final form) before the Team begins work on any lower-priority functional requirements.

1.5. References

Naming and Style Guidelines for Java (https://canvas.uw.edu/courses/1589305/files/95799003)

This document is organized based on the following documents from the Course's Canvas page (in https://canvas.uw.edu/courses/1589305/files/95798939):

- SRSExample-webapp.pdf
- srs_example_2010_group2.pdf
- COS_SRS.pdf
- srs_template.doc

2. Overall Description

2.1. Product Perspective

The system includes a user interface, a controller, the Game, and a database for Adventurer, Monster, and Trap details.

The user interface will be console-based.

The user interface will interact with the Game through a controller to allow the Player to interact with the Game, and the Game will update its internal state according to the commands provided by the controller and provide information to the controller that can be formatted and displayed in the user interface (in accordance with the Model View Controller design pattern). Data describing the classes of Adventurers, Monsters, and Traps that the Player may see will be provided to the Game by the database when the Game starts.

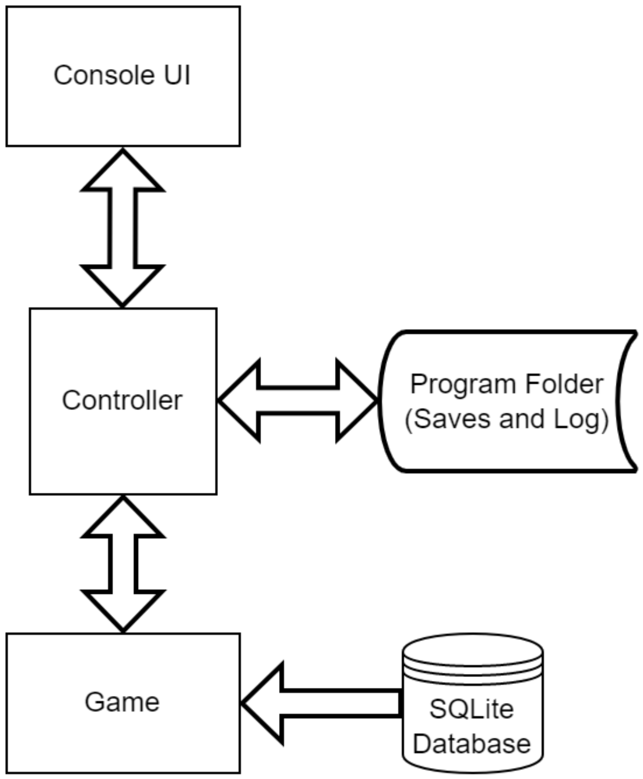

Figure 1. *System Diagram*

2.2. Product Features

The application will allow the Player to choose an Adventurer class and explore and fight their way through a dungeon maze with the aid of a map that is updated as the Adventurer explores, the Adventurer's stats and special skill, potions and other items collected from the rooms and Monsters of the dungeon, and an inventory to view and use those items. The Player must guide the Adventurer through new rooms and Monster fights to collect all four Pillars of OO (Abstraction, Encapsulation, Inheritance, and Polymorphism) and reach the exit alive.

The Game will allow the Player to create new Game Instances, save multiple Game Instances and multiple copies of the same original Game Instance at different points in time, and load Game Instances to play.

2.3. User Characteristics

The Player is expected to be able to use a keyboard for the console-based UI.

The Player will create or load a Game Instance from the user interface, provide input to navigate their Adventurer through a dungeon and direct the Adventurer in combat, and choose to save or not save the current Game Instance's state at any time (with an explicit prompt to do so when loading another Game Instance or exiting the Game).

2.4. Operating Environment

OE-1: The program operates on Java 17 (and can run on any OS supporting Java 17).
OE-2: The program accesses an internal SQLite database.

2.5. Design and Development Constraints

CO-1: All code conforms to Naming and Style Guidelines for Java.
CO-2: All internal code (other than database queries) is written in Java.
CO-3: Database queries are written in SQL.
CO-4: The program is developed in IntelliJ IDEA.

2.6. Assumptions and Dependencies

AS-1: User device will have Java 17 (or newer).
AS-2: User device will have a keyboard.
AS-3: User will be able to save files to and read files from their device for saving and loading progress.
DE-1: Details of Adventurers, Monsters, and Traps in the Game depend on data retrieved from a local SQLite database.

3. Requirements

3.1. Functional Requirements

3.1.1. Create Game Instance ***

Description: The Player should be able to create and run a new Game Instance in the Game.

Stimulus/Response Sequences

1. Stimulus: The Player enters the character or character sequence associated with creating a new Game Instance into the title screen menu.
2. Stimulus: The Player enters the name or number of an Adventurer class.
   Response: If an invalid Adventurer class is specified, the UI reports this and prompts for another attempt.
   Response: Otherwise, the Game waits for the Player to provide the rest of the input for creating a Game Instance.
3. Stimulus: The Player enters the name or number of a difficulty level.
   Response: If an invalid difficulty is specified, the UI reports this prompts for another attempt.
   Response: Otherwise, the Game waits for the Player to provide the rest of the input for creating a Game Instance
4. Stimulus: The Player enters the Adventurer's name.
   Response: If an empty name is entered, a randomly generated name is used.
   Response: If an invalid length or character is provided, the UI indicates this and prompts for input again.
   Response: The UI passes the entered information to the controller, which creates a new Game Instance, randomly generates a dungeon, and begins running the new Game Instance.
5. Stimulus: The Player enters the back option before entering all the required information.
   Response: The UI returns to the title screen.

3.1.2. Save Game Instance **

Description: The Player should be able to save a file representing a Game Instance to the application’s dedicated save folder.

Stimulus/Response Sequences

1. Stimulus: The Player enters the character or character sequence associated with saving the Game Instance.

   Response: The UI opens a menu for saving and prompts the user to choose a save file from a list to overwrite, enter the name for a new save file, or save to the most recently opened/saved file.
2. Stimulus: When prompted to select a save file to overwrite or enter a name for a new save file, the user selects a save file, enters a name, or chooses the option for the most recent file.

   Response: If a name is submitted and is empty or otherwise invalid, the UI indicates this and repeats the prompt.

   Response: Otherwise, the Game serializes the data of the current Game Instance, stores it to the specified file in the application's dedicated save folder, and displays a message indicating that the save was completed.

3.1.3. Load Game Instance **

Description: The Player should be able to load a file representing a Game Instance from the application's dedicated save folder and run that Game Instance.

Stimulus/Response Sequences

1. Stimulus: The Player enters the character or character sequence associated with loading a Game Instance from the title or exploration screen.

   Response: If the Game Instance has unsaved changes, the UI informs the Player of this and asks the Player to save.

   Response: Otherwise, the UI opens a menu for loading and prompts the user to enter the name or number of one of the listed save files.
2. Stimulus: When prompted to enter the name or number of an existing save file, the user enters a name or number.

   Response: If a name is entered and is empty or otherwise invalid, the UI indicates this and repeats the prompt.

   Response: Otherwise, the Game deserializes the data of the specified Game Instance from the application's dedicated save folder and begins running that Game Instance, and the UI displays a message indicating that the load was completed.
3. Stimulus: When prompted to save unsaved changes, the Player selects

the option to save unsaved changes.

Response: The Game saves the current Game Instance as described in 3.1.2 and loads the other Game Instance as described in Stimulus/Response 2.

4. Stimulus: When prompted to save unsaved changes, the Player selects the option to continue without saving.

   Response: The Game loads the other Game Instance as described in 3.1.2.

5. Stimulus: When prompted to save unsaved changes, the Player selects the option to go back.

   Response: The UI returns to the previous menu without saving or loading any Game Instance.

3.1.4. Randomly Generate Dungeon **

Description: The application should generate dungeons of random dimensions with random door placement (while guaranteeing the maze is traversable), random item placement, random trap placement, random monster placement, random entrance and exit placement, and placement of the Adventurer at the entrance.

Stimulus/Response Sequences

1. Stimulus: The Player creates a new Game Instance.

   Response: The application randomly generates the contents of a new dungeon maze within parameters determined by the difficulty level selected for the new Game Instance.

3.1.5. Move Adventurer ***

Description: The Player should be able to choose which direction the Adventurer will move from their current position in the maze.

Stimulus/Response Sequences

1. Stimulus: The Player enters the character associated with north.

   Response: If there is not a north door in the current room, the controller reports to the UI that the action is not possible and repeats the menu.

   Response: Otherwise, the Game moves the Adventurer to the room to the north of the current room and opens the new room's menu or (if there is a Monster in that room) the combat view.

2. Stimulus: The Player enters the character associated with south.

	Response:	If there is not a south door in the current room, the controller reports to the UI that the action is not possible and repeats the menu.
	Response:	Otherwise, the Game moves the Adventurer to the room to the south of the current room and opens the new room's menu or (if there is a Monster in that room) the combat view.
3.	Stimulus:	The Player enters the character associated with west.
	Response:	If there is not a west door in the current room, the controller reports to the UI that the action is not possible and repeats the menu.
	Response:	Otherwise, the Game moves the Adventurer to the room to the west of the current room and opens the new room's menu or (if there is a Monster in that room) the combat view.
4.	Stimulus:	The Player enters the character associated with east.
	Response:	If there is not an east door in the current room, the controller reports to the UI that the action is not possible and repeats the menu.
	Response:	Otherwise, the Game moves the Adventurer to the room to the east of the current room and opens the new room's menu or (if there is a Monster in that room) the combat view.
5.	Stimulus:	The Player enters the character associated with canceling the current action.
	Response:	The UI does not send the attempt to move to the Game and opens the previous menu.
6.	Stimulus:	The Player moves onto a staircase.
	Response:	The Game moves the Adventurer to the room connected to that staircase on a different floor of the dungeon, the UI indicates that this has happened, and the UI updates to show the new Room.

3.1.6. Collect Items ***

Description: The Player should be able to add items from rooms to the Adventurer's inventory.

Stimulus/Response Sequences

1.	Stimulus:	When not in combat, the Player enters the character associated with this action.

Response: The Game adds the room's items to the inventory and removes them from the room.

3.1.7. Encounter Trap ***

Description: When entering a room with an active trap, the Player should take damage and gain debuffs or dodge the trap.

Stimulus/Response Sequences

1. Stimulus: The Adventurer enters a room that contains a trap.
   Response: If the Adventurer is affected by the trap (because the trap guarantees it or the Adventurer's stats do not meet the randomly adjusted requirement to avoid the trap), the Adventurer takes a randomly adjusted amount of damage and may gain a debuff associated with the trap.
   Response: Otherwise, the Game does not apply any damage or debuffs to the Adventurer.
   Response: If the trap is single-use, the Game will update the trap's flag to mark it as broken and prevent it from activating again.

3.1.8. Access Adventurer Inventory ***

Description: The Player should be able to open, use items from, and close the Adventurer's inventory when exploring a dungeon or in combat.

Stimulus/Response Sequences

1. Stimulus: While the inventory is closed and the Adventurer is exploring or in combat, the Player enters the character associated with this action.
   Response: The UI formats and displays information retrieved by the controller about the contents of the inventory and prompts the Player to select an item.
2. Stimulus: While the inventory is opened, the Player enters the character or number associated with an item.
   Response: If the item can be used in the current context, the item is consumed and has its consequences applied to the Adventurer, room, or map.
3. Stimulus: While the inventory is opened, the Player enters the character associated with closing the current menu.
   Response: The UI stops displaying the contents of the inventory and returns to displaying the current room or combat.

3.1.9. View Dungeon Map **

Description: The Player should be able to open a map to view their current location in a dungeon and the contents of any rooms they have explored or seen with a vision potion.

Stimulus/Response Sequences

1. Stimulus: While exploring, the Player enters the character associated with this action.
   Response: The UI displays the layout of the dungeon, known rooms' contents, and the Adventurer's current location retrieved by the controller.
2. Stimulus: With the map open, the Player presses enter.
   Response: The UI stops displaying the map and returns to viewing the current room.

3.1.10. Enter Combat ***

Description: The Player should automatically enter a fight with a Monster when entering a room that contains a Monster.

Stimulus/Response Sequences

1. Stimulus: The Player enters a room containing a Monster.
   Response: The UI displays the Monster and their name and stats (including current and maximum health) and buffs/debuffs, the Adventurer and their name and stats and buffs/debuffs, and (when it is the Adventurer's turn) a menu to choose from the combat actions (open inventory to use item, basic attack, special skill, and attempt to flee).

3.1.11. Flee Combat ***

Description: The Player should be able to attempt (and succeed some percentage of the time) to flee while in combat.

Stimulus/Response Sequences

1. Stimulus: When the combat menu is open, the Player enters the character associated with fleeing.
   Response: The UI prompts the Player to choose a direction to move in and receives input as described in 3.1.5.
2. Stimulus: After selecting the option to flee, the Player selects the back option instead of choosing a direction.
   Response: The UI does not submit the attempt to flee to the Game and

returns to the main combat view.

3. Stimulus: After selecting the option to flee, the Player enters the character associated with a direction.

   Response: The UI and the Game respond as described in 3.1.5 for valid and invalid input given the submitted direction.

   Response: When a valid direction is submitted, the Game randomly tests whether the Adventurer can flee based on the Adventurer's and Monster's speeds.

   Response: If the Game calculates the Adventurer's attempt to flee to be successful, the Game ends the fight without removing the Monster from the room (and without modifying the Monster's stats from what they were at the point in the battle when the Adventurer fled), moves the Adventurer into the provided room, and opens a corresponding menu.

   Response: If the Game calculates the Adventurer's attempt to flee to be unsuccessful, the Adventurer loses that turn, advancing the fight to the next turn (which may be the Adventurer's or the Monster's).

3.1.12. Be Attacked ***

Description: The Player should be able to be attacked by a Monster in combat.

Stimulus/Response Sequences

1. Stimulus: The Player enters combat with a Monster or is already in combat and ends the Adventurer's turn.

   Response: If the previous sequence of turns (if any), the Monster's speed, and the Adventurer's speed dictate that the Monster has the next move, the Monster attacks the Adventurer with a random chance of damaging and possibly applying a debuff to the Adventurer (based on the Monster's hit chance and debuff chance and the Adventurer's block chance), possibly killing the Adventurer as described in 3.1.15.

   Response: Otherwise, the Game advances the combat to the next turn (which may be the Monster's or the Adventurer's).

3.1.13. Use Basic Attack ***

Description: The Player should be able to attack a Monster in combat.

Stimulus/Response Sequences

1. Stimulus: When the combat menu is open and it is the Adventurer's turn, the Player enters the character associated with attacking.
   Response: The Game uses the Adventurer's hit chance as a probability for dealing a randomly adjusted amount of damage to and possibly applying a debuff to the Monster.
   Response: The Game uses the Monster's and Adventurer's speeds to determine whether the Monster or Adventurer has the next turn.
   Response: If the Monster's HP drops to 0 after the attack, the Monster dies and is removed from the room by the Game, the Game places the Monster's drops (if any) in the room, and the UI switches from the combat view to the exploration view.

3.1.14. Use Special Skill **

Description: The Player should be able to use the Adventurer's special skill(s) in combat.

Stimulus/Response Sequences

1. Stimulus: When the combat menu is open and it is the Adventurer's turn, the Player enters the character associated with using a special skill.
   Response: The Game applies the effects of the special skill to the Adventurer's or (after a probability test) Monster's stats (such as damage and a debuff from an attack or applying healing to the Adventurer) and ends the Adventurer's current turn.
   Response: If the Monster dies, the Game performs the actions described for ending combat in the Stimulus/Response section of 3.1.13.

3.1.15. Die **

Description: The Adventurer should die when their HP drops to 0 in or out of combat.

Stimulus/Response Sequences

1. Stimulus: The Player moves the Adventurer into a room and encounters a trap, takes damage from an attack in combat, or takes continuous damage from a debuff in or out of combat.

Response: If the Adventurer's HP drops to 0, the Player will lose the game.

3.1.16. Lose Game **

Description: The Player should lose the game within the current Game Instance if the Adventurer's HP is 0 or lower.

Stimulus/Response Sequences

1. Stimulus: The Player performs one of the actions described in the Stimulus/Response section of 3.1.15 and causes the Adventurer to die.
   Response: The Game ends the current Game Instance, the UI displays a message indicating that the Player has lost, and the UI returns to the title screen.

3.1.17. Win Game **

Description: The Player should win the game within the current Game Instance if the Adventurer brings all four Pillars of OO to the dungeon's exit.

Stimulus/Response Sequences

1. Stimulus: When the Adventurer is in the room with the exit and has all four Pillars of OO in their inventory, the Player enters the character associated with exiting the dungeon.
   Response: The UI displays a message indicating that the Player has won and displays a menu for continuing the current Game Instance or returning to the title screen. The player will be prompted to save if they have unsaved changes and choose to return to the title screen.

3.2. Non-Functional Requirements

3.2.1. Readability and Organization

- All class names, method names, and field names (including in testing code) should be intent-revealing.
- All classes should have consistent spacing and indentation.
- Classes should be organized into packages based on their place in the model-view-controller design pattern.

3.2.2. Testability

- Each method should perform the single operation described by its name.
- Each class should perform only the tasks associated with what it is designed to represent.
- Instances of identical or very similar multiple-operation code should usually be extracted to a method both to minimize the amount of code that needs to be tested and to avoid redundant tests.
- To aid regression testing, tests should be kept in the development code rather than removed after running.
- Testing classes and methods should be organized and kept in a single folder (possibly with subfolders).

3.2.3. Maintainability

- Most code should be written with easy addition of features in mind.

3.2.4. Reliability

- Production code for the application should not freeze, crash, or frequently lag under the conditions in which the user is expected to execute the application.

3.2.5. Portability

- The application should be able to be easily placed on a new computer as a single executable file or with an installer application.

3.3. User Interface

All menus except for the combat and exploration views will include a back option to return to the previous menu.

3.3.1. Title Screen

When the Game opens, the UI will display the Game's title and a menu to create a new Game Instance, load a Game Instance, view the play guide, or close the application.

3.3.2. Play Guide

When the Player has selected the play guide option a description of the Game's premise will appear with a menu for the Player to choose a guide to view to explain the UI interactions, symbol meanings, Adventurer interaction, exploration system, combat system, and inventory system.

3.3.3. Save Menu

When the Player has selected the save option, a menu will display the existing save files and prompt the Player to select one to overwrite, enter the name of a new file, or save to the most recently loaded or saved file.

If the Player enters an invalid file name, a message printed to the console will indicate this and prompt the Player to enter another name.

3.3.4. Load Menu

When the Player has selected the load option, a menu will display the existing save files and prompt the Player to select one to load.

If the Player enters the name or number of a nonexistent file a message will be printed to the console to indicate this and prompt the Player to enter another name or number.

3.3.5. Exploration View

When the Player has created or loaded a Game Instance and is not in combat, the Game will display the current room, its items and other contents, and the options for collecting the items in the room, moving to an adjacent room, using the stairs, opening the inventory or map, opening the play guide, opening the saving or loading menus, or returning to the title screen.

The UI will prompt the player to save or confirm that they wish to continue without saving before returning to the title screen or loading a file.

3.3.6. Combat View

When the player enters a room with a Monster, the Game will display the Monster, the Monster’s name and stats, the Adventurer, and the Adventurer’s name and stats. The combat view will also display the actions the Adventurer can perform (attack, use special skill, open inventory, flee, or open play guide) and allow the Player to choose from these actions.

3.3.7. Inventory

When the Player has selected the inventory option, a menu will display the Adventurer’s current items and allow the Player to use an item.

The inventory can be opened from the exploration view or the combat view.

3.3.8. Map

When the Player has selected the map option, a formatted String representing the layout of the dungeon will show the current items and other contents (including any living monsters) of rooms the Adventurer has explored or seen with a vision potion.

The map can only be opened from the exploration view.

3.4. Software Interface

The application runs on Java 17 or newer and communicates with an internal SQLite database with SQL queries.