



Software Requirements Specification

for

Dungeon Adventure 2.0

Initial Draft Document

Prepared by Halt Catch Fire

University of Washington – Tacoma Campus

January 25, 2022

1. Introduction.....	1
1.1 Purpose	1
1.2 Document Conventions	1
1.3 Intended Audience and Reading Suggestions.....	1
1.4 Project Scope	1
1.5 References.....	2
2. Overall Description	2
2.1 Product Perspective	2
2.2 Product Features	3
2.3 User Classes and Characteristics	3
2.4 Operating Environment	4
2.5 Design and Implementation Constraints.....	4
2.6 User Documentation	4
2.7 Assumptions and Dependencies	4
3. System Features	5
3.1 Start Game	5
3.2 New Game	5
3.3 Monsters	6
3.4 Heroes	6
3.5 Combat.....	7
3.6 Save Game	7
3.7 Load Game.....	7
3.8 Win Game.....	8
3.9 Lose Game	8
3.10 Restart Game	8
4. External Interface Requirements	9
4.1 User Interfaces	9
4.2 Hardware Interfaces.....	9
4.3 Software Interfaces	9
4.4 Communications Interfaces	10
5. Other Nonfunctional Requirements.....	10
5.1 Performance Requirements.....	10
5.2 Safety Requirements.....	10
5.3 Security Requirements.....	10
5.4 Software Quality Attributes	11
6. Other Requirements	11
6.1 Appendix A: Glossary	11
6.2 Appendix B: Analysis Models.....	11
6.3 Appendix C: Issues List.....	11

Revision History

Name	Date	Reason For Changes	Version
Kevin Chung	1/25/22	Initial Draft Software Requirements and Specifications (SRS)	1.00
Stephanie Liu	1/27/22	See above. 1.4, 2.1, 2.3, 2.7, 3 through appendix adjusted.	1.01
Kevin Chung	1/28/22	Added 3.8-3.10	1.02
Stephanie Liu	1/29/22	Adjusted 3.8-3.10 since 3.9 & 3.10 were duplicates.	1.03

1. Introduction

1.1 Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

This SRS describes the software functional and nonfunctional requirements for release 1.0 of Dungeon Adventure – The Spooky Bard Returns. This document is intended to be used by the members of the Halt Catch Fire team that will implement and verify the correct functioning of the software. Unless otherwise noted, all requirements specified here is a first draft.

1.2 Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

Player: User of the software.

Developer: Software engineering student.

1.3 Intended Audience and Reading Suggestions

<Describe the several types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections, and proceeding through the sections that are most pertinent to each reader type.>

This SRS document is intended for software engineering students and faculty enrolled in the Graduate Certificate program at University of Washington – Tacoma. The contents of this SRS contain implementation assessments for project deliverables. Read this document sequentially, unless otherwise noted. Previous course lectures may be referenced throughout this document, noted with course name and lecture title.

1.4 Project Scope

<Provide a brief description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. An SRS that specifies the next release of an evolving product should contain its own scope statement as a subset of the long-term strategic product vision.>

Our goal for this project is to extend our previous course project, Dungeon Adventure – The Spooky Bard.

Proposed features:

- New dungeon classes:
 - Hero
 - Monster
- Game Play Mechanics
 - Combat
 - Monsters must be defeated throughout the dungeon
 - Stronger monsters are in rooms with artifacts (bosses)
 - Each hero and monster class have different abilities
 - Ability to save and load existing game states
- Graphical User Interface (GUI)
 - Tkinter
- SQLite Database Support
 - Monster Database (name and statistics)
 - Hero Database
 - Used during the creation of the dungeon
- Additional features as deemed by Halt Catch Fire
 - Flee from an encounter (Kevin)
 - Critical failure / critical success (Steph)

1.5 References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader can access a copy of each reference, including title, author, version number, date, and source or location.>

OneDrive/Halt Catch Fire/TCSS 504/
dungeon_adventure_project_details.docx
dungeon_adventure_srs_initial.docx

GitHub/The Spooky Bard/
Original code base – tcss504 branch

Toggl time tracker
Pivotal project tracker

2. Overall Description

2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

This project is a continuation of the previous end of the quarter project from TCSS 502 called Dungeon Adventure – The Spooky Bard. The new project should be considered a replacement for the previous iteration.

The software will use Tkinter for a graphical user interface (GUI).

The software will use SQLite databases that will store Hero/Monster creation data.

The complexity of this new iteration will be significant.

2.2 Product Features

<Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Details will be provided in Section 3, so only a high-level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top-level data flow diagram or a class diagram, is often effective.>

Proposed features:

- New dungeon classes:
 - Hero
 - Warrior
 - Priestess
 - Thief
 - Monster
 - Ogre
 - Gremlin
 - Skeleton
- Game Play Mechanics
 - Combat
 - Attack speed
 - Damage inflicted
 - Battle rounds
 - Chance to hit
 - Attack/Defense mechanics
 - Monsters must be defeated throughout the dungeon
 - Stronger monsters are in rooms with artifacts
 - Each hero and monster class have different abilities
 - Ability to save and load existing game states
- SQLite Database Support
 - Monster Database (name and statistics)
 - Used during the creation of the dungeon
- Additional features as deemed by Halt Catch Fire
 - Flee from an encounter (Kevin)
 - Luck mechanic? (Kevin)
 - Critical success / critical failure (Steph)

2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.>

Player user class:

A player will want to play a game and save the current state of progress within the game. A save/reload feature will be required for the desired behavior.

A player will want to win the game.

Tester user class:

Like the player user class but provides access to cheat codes to view map immediately, have unlimited potions, instantly beat a monster, lose, have more health, have more vision potions, etc...

Developer would use this to test functionality throughout the development process.

2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

Requirements:

- Windows, Mac/Linux operating systems
- PyCharm Community Edition 2021.3.1+ or VSCode 1.63+
- TKinter
- Python 3
- SQLite3

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

Time constraint as most peers will be able to work on this project for at most 6 hours a week

Limited knowledge of working with a team to build a complex program

Size constraint – submitted through Canvas

Limited knowledge of SQL and TKinter

2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

Docstring comments will be adherent to PEP8 standards. Sphinx preferred.

User documentation on how to use the software may be found within the Help menu.

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues

around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

We are dependent on external services listed in 2.4 and should any of those be unavailable for any reason or pose any unknown threats. Similarly, vulnerabilities in SQLite leave us more vulnerable as our code will be able to be accessed remotely. Any potential security issues on whatever service we deploy our project on can also impact us as well.

3. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

3.1 Start Game

3.1.1 Description and Priority

<Provide a brief description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

Open game from exe. **High priority.**

3.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

User opens executable file (.exe). Start screen shows.

3.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate, when necessary, information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1: Executable file compiled and opens.

REQ-2: Must display start screen to player.

3.2 New Game

3.3.1 Description and Priority

Able to start a new game. Second highest of priorities.

3.2.2 Stimulus/Response Sequence:

Player clicks "New Game." View asks for difficulty, hero name, and class. Controller sends difficulty setting to Model, which generates a dungeon containing monsters to match the current difficulty level.

3.2.3 Functional Requirements:

REQ-1: View implemented so user can click on New Game which takes to new view to ask data.

REQ-2: View able to take input for difficulty, hero name, class.

REQ-3: View able to send that information to controller on button press.

REQ-4: Controller able to modify info into Model friendly language and send it difficulty level and hero information to generate the game.

REQ-5: Controller can take generated information from Model for hero and dungeon and provide it to view.

REQ-6: View can display that information.

3.3 Monsters

3.3.1 Description and Priority

Monsters with different abilities exist within the game and are in the dungeon.

3.3.2 Stimulus / Response Sequence

When loaded, a dungeon is filled with monsters in available slots. Monsters display in dungeon room when player enters room with monsters. Combat initiated?

3.3.3 Functional Requirements

REQ-1: Monsters have the following attributes:

- Names

- Stats (hp, mp, defense, attack, initiative?)

- Special abilities

REQ-2: Monsters require way to be displayed in view.

REQ-3: Differentiation for boss monsters in pillar rooms.

REQ-4: Differentiation for difficulty setting?

REQ-5: Dungeon needs a way to know a monster is in the room.

REQ-6: Monsters need to be created when difficulty is set (builder pattern in dungeon builder?)

3.4 Heroes

3.4.1 Description and Priority

Heroes with different classes (and matching abilities) exist within the game and are in the dungeon. Identify location of hero.

3.4.2 Stimulus / Response Sequence

Created, hero name and class are selected by player. Players see their hero, and when engaging in combat, have a specific skill for their hero.

3.4.3 Functional Requirements

REQ-1: Heroes have the following attributes:

- Names

- Stats (hp, mp? defense, attack, initiative?)

- Special abilities

- Inventory?

REQ-2: Heroes require way to be displayed in combat.

REQ-3: Dungeon needs a way to display hero on a map.

REQ-4: Heroes have inventory if individual?

3.5 Combat

3.5.1 Description and priority

Combat occurs when a hero enters a room with a monster. It ends when one of them reaches 0 HP or flees.

3.5.2 Stimulus / Response Sequence:

Hero enters room with monster. Combat initiated. Based on attack speed, player selects skill or attack, does so, and then monster goes or monster goes first. May have multiple attacks in a single round. Once all have gone, starts over until one has died / drop to 0 HP. Once they have, exit combat either to game over screen or resuming from map.

3.5.3 Functional Requirements

REQ-1: Initiative / attack speed compared

REQ-2: Whoever has higher attack speed or initiative goes first.

REQ-3: Need interface for player to select what they want to do then have that selection sent to controller which processes the outcome and updates local model. View will be updated by controller with updated stats to display for both.

REQ-4: Need combat interface and menu.

REQ-5: Need to display outcomes of each round as they happen or a log to show monster actions.

REQ-6: Display status of both parties.

REQ-7: Enable potion usage during battle?

REQ-8: Option to flee?

3.6 Save Game

3.6.1 Description and priority

Ability for players to save games.

3.6.2 Stimulus / Response Sequence

Player clicks save or reaches a save point (should we add these?) and saves the game so it can be reloaded later.

3.6.3 Functional Requirements

REQ-1: View option to save the game

REQ-2: When save is clicked, generates a file, or saves data to a DB to store current game state.

Game state data:

Hero (hp, name, mp, class)

Hero location

Current inventory

Current viewed map

Whole dungeon (including pillar, monster location with reference IDs)

3.7 Load Game

3.7.1 Description and priority

Ability for players to load game from a saved game data.

3.7.2 Stimulus / Response Sequence

At the main menu, the player clicks load game. Game gives menu of saved games. Player selects from saved games and restarts from previous game state.

3.6.3 Functional Requirements

REQ-1: View option to load the game.

REQ-2: View displays saved games when load is clicked.

REQ-3: When load is clicked, then loads up the saved game to continue.

REQ-4: Need to be able to process saved game state into active running game.

3.8 Win Game

3.8.1 Description and priority

Ability for users to achieve a victory state in the game:

Escape the dungeon after collecting all the pillars and defeating the final boss.

3.8.2 Stimulus / Response Sequence

Once the player reaches the exit of the dungeon, a trivia sphinx will be the final boss. The player must answer the question correctly to win the game.

After the correct answer is selected, a final screen will be presented with information about the dungeon session and a congratulatory message upon completion.

3.8.3 Functional Requirements

REQ-0: Check on game state to see if victory conditions have been met

REQ-1: Congratulatory message at the end

REQ-2: Provides options to restart, new game, load game or quit program

3.9 Lose Game

3.9.1 Description and priority

Ability for users to lose the game.

Fail to escape the dungeon by not collecting all the pillars.

Users can be defeated by monsters that overpower the hero character.

3.9.2 Stimulus / Response Sequence

Once a player is defeated by monsters by going below 0 hit points the game should end.

Additional losing criteria may be added later.

3.9.3 Functional Requirements

REQ-0: Check on game state to see if lose conditions have been met

REQ-1: Game over message at the end

REQ-2: Provides options to restart, new game, load game or quit program

3.10 Restart Game

3.10.1 Description and priority

After a lose or victory, user can select to restart the same game already played with same maze and monsters.

3.10.2 Stimulus / Response Sequence

Once a player reaches a lose/win game state. The software will provide an option to restart from start of game state. On click, game will load played to the start of the game at starting health / location / all monsters and items back at starting position in the map.

3.10.3 Functional Requirements

REQ-1: Need to store game from initial state prior to game play and changes being made as game is played.

REQ-2: Will load hero at start with same name, class, etc selected.

4. External Interface Requirements

4.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons, and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

The software will require the following views:

View	Description
Start screen	Start a new game or continue previous game and load from saved game.
Dungeon crawler	Move character around map, indicates any item found in text to the side as a log. Menu to save game, quit, load, help, check inventory and character status.
Inventory menu	Shows character status and player inventory. Allows item use.
Battle	Log of battle actions taken by monsters and by players
Tutorial / Help	Explain how to play the game, key commands.

Figure Dungeon Crawler

Example of dungeon crawler view with dots representing heroes or monsters, dark grey representing available traversal, dots representing objects or monsters or heroes, and user clicking their desired location.

4.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

Supports computer screen 10" or larger.

4.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial

components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

TBD

4.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

Run on a local machine with Python 3 interpreter. Saved on local and loaded as well.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

TBD

5.2 Safety Requirements

<Specify those requirements that are concerned with loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

None required at this time.

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

Software will be using a SQL database; therefore, connections must be secured. Serialization of game states will be, therefore use serialized objects for secure/trusted sources only.

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

Unit testing will be vital to verify the individual modules other components depend on are comprehensible to their users. Modular design will also be vital due to how much larger this program will be.

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Avoid copyright infringement.

6.1 Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

6.2 Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>



6.3 Appendix C: Issues List

< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>

Fleeing combat?
Save data to a file or save to DB?
User interface design
Combat system – Initiative? Speed?