

COMP 4770 – Requirements Document Specification

The first milestone for COMP 4770 is to write a Requirements Document for the system you will be creating. Since this system is a game, this Requirements Document will be a hybrid of a traditional software requirements document, and a game design document. These documents will outline the game you intend to make, as well as the software requirements to make it. Please note that this is not an architecture document, so you do not need to specify exact classes, diagrams, or low-level architecture decisions. Wherever possible, give examples of gameplay with sketches that illustrate the design decision. **This document will be created / submitted on your GitHub project Wiki page.**

It is expected that your final game will deviate from this document, but try to make it as accurate as possible.

Game Design Document

1. Game Overview

- 1.1. Game Concept / Genre
- 1.2. Game Flow Summary – At a high level, how does the game progress?
- 1.3. Look and Feel – What is the basic look and feel of the game? What is the visual style?

2. Gameplay and Mechanics

- 2.1. Gameplay
 - 2.1.1. Game Start / Login (How does the user start playing the game)
 - 2.1.2. Game Progression
 - 2.1.3. Mission/challenge Structure
 - 2.1.4. Puzzle Structure
 - 2.1.5. Objectives – What are the objectives of the game?
 - 2.1.6. Play Flow – How does the game flow for the game player
 - 2.1.7. Save Progress – Save Points? Quick Save? Checkpoints? Explain why
- 2.2. Mechanics – What are the rules to the game? This is the model of the universe that the game works under. Think of it as a simulation of a world, how do all the pieces interact?
 - 2.2.1. Physics – How does the physical universe work? (ex: gravity, collisions)
 - 2.2.2. Movement in the game (limits/types of movement, controls)
 - 2.2.3. Objects – How to pick them up and move them (ex: walk over, press button, etc.)
 - 2.2.4. Actions – Switches / buttons / object interaction
 - 2.2.5. Combat – How does the player do combat with enemies?
 - 2.2.6. Economy – What is the economy of the game? How does it work?
 - 2.2.7. Screen Flow -- A graphical description of how each screen is related to every other and a description of the purpose of each screen. (example: mega man, sometimes screen flows smoothly to the right, sometimes it transitions a full screen at a time)
- 2.3. Game Options – What are the options and how do they affect game play and mechanics?
- 2.4. Cheats and Easter Eggs – (Including cheat codes? What are the input method and their effects?)

3. Story, Setting and Character

- 3.1. Story and Narrative – Our game is not heavily story driven, but include some sort of back story / motivation for why the player should care to complete the game.
- 3.2. Game World
 - 3.2.1. General look and feel of world (overworld vs. individual levels)
 - 3.2.2. Areas, including the general description and physical characteristics as well as how it relates to the rest of the world (what levels use it, how it connects to other areas)
- 3.3. Characters. The game should have a few characters (including the main player character, bosses, small story)

4. Levels

- 4.1. Overall structure of levels. How are they accessed, how are they completed?
- 4.2. Level Design - Description of level design. This document does not need to include a description of all levels that will be in the game (we are making a maker after all), however, describe characteristics of a level so that a reader can get an idea of how one looks / plays.

5. Interface

- 5.1. Visual System. If you have a HUD, what is on it? What menus are you displaying? What is the camera model?
- 5.2. Control System – How does the game player control the game? What are the specific commands?
- 5.3. Audio, music, sound effects – When is music playing? When do sound effects play?

6. Artificial Intelligence

- 6.1. Enemy AI – Describe some behaviors that enemy objects and characters will have. (ex: move toward player, shoot at player, run away, etc.)

7. Game Art

- 7.1. Key assets, how they are being developed. Intended style.

8. Level Editor

- 8.1. Overview of level editor (scope, purpose)
- 8.2. Interface – How will the user interact with the editor?
- 8.3. Menu System – What will the menu system look like?
- 8.4. Transition – How will you test the levels made in the editor?

9. Player Account

- 9.1. Player Login / Profile – Stores / displays save games, levels, scores, achievements, etc.
- 9.2. Save Game – Quick Save vs. Save Points vs. Checkpoints
- 9.3. High Scores – Saving high scores / fastest times?
- 9.4. Achievements – What type of achievements to store?

Software Requirements Document

Note: Refer to specific section numbers of the Game Design Document wherever possible

1. Project Description

- 1.1. Who are the team members, what are their roles within the project?
- 1.2. What are the main 4 modules of the project?
- 1.3. What is your timeline to accomplish the major tasks in the project? Give a text description, as well as a Gantt chart showing each module's progress. Give a rationale behind the timeline (ex: database must be working before the system is able to store saved games).

2. Product Perspective

- 2.1. User Interfaces - Describe the main user interface between the system and the users. List configuration characteristics / requirements (screen resolution, display type, input devices)
- 2.2. Hardware Interfaces - Specify the characteristics between the product and the hardware elements of the system. What are the hardware requirements to set up / run the front and back of the system?
- 2.3. Software Interfaces - Specify the required software products to use the system and their purpose / why they are required. (example: OS, libraries, browser, databases, etc.)
- 2.4. Communication Interfaces – Specify any networking / communications / protocols used in the product
- 2.5. Memory Constraints – What are the requirements on primary / secondary memory for the system

3. Use Cases

Write Use Cases that cover interactions between the user and the system, involving as much of the functionality of the system as possible. A use case flow should be a simple description of the task, written in numbered steps. You do not need to write Each use case should contain the following information:

- Title [goal] (ex: "Launch level editor")
- Description (English description of the events in the use case)
- Actor(s)
- Triggers (action which causes system change in this case)
- Preconditions & Postconditions (system state before and after use case)
- Normal Flow (when nothing goes wrong)
- Alternative Flow (something went wrong)

Wherever possible, list the section number of the Game Design document that is related. Use cases should include (but are not limited to) topics such as:

- User login screen (create account, forgot password, ...)
- User game menu (selecting options, game modes, ...)
- Using level editor (placing tiles/objects, load level, save level, etc.)
- User playing a level (start a level, playing a level, saving game, etc.)