

**MACAU UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**School of Computer Science and Engineering**

**Faculty of Innovation Engineering**

**<<Software Project for Course Software Engineering>>**

Homework ID : Task2-User Requirements Specifications

Report Title : xxx

Student Name : xxx

Student No. : xxx

Date : xxx

**Amendment History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Modified By** | **Reviewed By** | **Amendment Details** |
| 1.0 |  |  |  | Initial version |
|  |  |  |  |  |

Table of Contents

[1. Introduction 4](#_Toc178688185)

[1.1. Purpose 4](#_Toc178688186)

[1.2. Notation 4](#_Toc178688187)

[1.3. Scope 4](#_Toc178688188)

[1.4. Context Diagram 4](#_Toc178688189)

[1.5. Definitions and Acronyms 4](#_Toc178688190)

[1.6. References 4](#_Toc178688191)

[1.7. Overview 4](#_Toc178688192)

[2. General Description 4](#_Toc178688193)

[2.1. System Functions 4](#_Toc178688194)

[2.2. User Characteristics 5](#_Toc178688195)

[3. General Constraints 5](#_Toc178688196)

[3.1. Software Constraints 5](#_Toc178688197)

[3.2. Hardware Constraints 5](#_Toc178688198)

[4. Assumptions and Dependencies 5](#_Toc178688199)

[5. Functional Requirements Master List 5](#_Toc178688200)

[6. Functional Requirement REQ-1.1 6](#_Toc178688201)

[6.1. Description 6](#_Toc178688202)

[6.2. System Input 6](#_Toc178688203)

[6.3. Display 6](#_Toc178688204)

[6.4. System Processing 6](#_Toc178688205)

[6.5. System Output 6](#_Toc178688206)

[6.6. Other 6](#_Toc178688207)

[6.7. Constraints 6](#_Toc178688208)

[6.8. Data Handling 6](#_Toc178688209)

[6.9. Error Handling 6](#_Toc178688210)

[7. External Interface Requirement 7](#_Toc178688211)

[7.1. Data Interfaces 7](#_Toc178688212)

[7.2. User Interfaces 7](#_Toc178688213)

[7.3. Other Interfaces 7](#_Toc178688214)

[8. Non-Functional Requirements 7](#_Toc178688215)

[8.1. System Performance 7](#_Toc178688216)

[8.2. Information Security 8](#_Toc178688217)

[8.3. Availability 8](#_Toc178688218)

[8.4. Capacity 9](#_Toc178688219)

# Introduction

## Purpose

This document is a detailed system requirements specification for a defined service.

This document will build on the Business Requirements Specification (BRS) and must have a corresponding High-Level Design (HLD) document. Any required changes to the Business Requirements Specification or the High-Level Design that arise in developing this System Requirements Specification (SRS) must be updated in those documents.

Current and valid System Requirements Specification (SRS), High-Level Design (HLD), and Low-Level Design (LLD) are required as part of the operational support documentation set for any application.

## Notation

This specification and its supporting documents and diagrams use the Unified Modelling Language notation.

## Scope

This section describes the scope of the system.

## Context Diagram

A context diagram is included to demonstrate how the scope of this system integrates with related systems.

## Definitions and Acronyms

This section describes any terms used.

## References

This section lists any references and includes pre-requisite documentation and any other supporting documentation such as diagrams.

## Overview

This section describes the system requirements and explains how the references listed above can help support that overview.

# General Description

## System Functions

This section describes the high level purpose and functions of the system.

## User Characteristics

This section describes the various user types (roles) and their characteristics. The type of user interface that is required (data entry, self-service, trained user, global, organisation). This information will inform the Low Level Design. This includes information on predicted user platforms (devices, operating systems and browsers).

# General Constraints

## Software Constraints

This section lists the software constraints. This may cover licensing, technical standards, preferred products, and the impact of given decisions.

## Hardware Constraints

This section lists the hardware constraints. This may cover licensing, technical standards, preferred products, and the impact of given decisions.

# Assumptions and Dependencies

This section describes the assumptions and dependencies made in forming this system requirement.

# Functional Requirements Master List

|  |  |  |
| --- | --- | --- |
| **Req. ID** | **Requirement Name** | **Requirement Description** |
| REQ-1.1 | Character Movement and Control | Players can control character movement across the game environment using keyboard or controller inputs. |
| REQ-1.2 | Map generation and optimization | All the Maps and Props and generate Randomly and Continuously, and in order to reduce the System overhead, using the function to remove the items when invisible. |
| REQ-1.3 | Implementation of character basic attacks | Implement a system allowing players to perform basic attacks against enemies. |
| REQ-1.4 | Spawning the Monsters | Implement an enemy spawn system to introduce waves of monsters that pursue and attack the player. |
| REQ-1.5 | Game achievement system | Implement an achievement tracking system to recognize and reward player accomplishments. |
| REQ-1.6 | Game Menu and Setting | Implement a game menu with options for starting the game, loading saved games, accessing settings, and exiting. |
|  |  |  |

# Functional Requirement

# REQ-1.1

## Description

Players can control character movement across the game environment using keyboard or controller inputs. This includes directional navigation, sprinting (if applicable), and any character-specific movement abilities.

## System Input

**Keyboard Controls:** W, A, S, D keys or arrow keys for movement.

**Controller Support:** Left joystick or D-pad for directional movement.

**Additional Inputs (optional):** Shift for sprinting, spacebar for jumping (if applicable).

## Display

**In-game HUD:** Display visual indicators such as stamina (if movement includes sprinting) or a minimap for directional guidance.

**Character Feedback:** Show animations corresponding to each movement (e.g., walking, running) for visual feedback.

**Control Prompts:** Option to display control instructions for keyboard/controller in the settings or tutorial screens.

## System Processing

**Movement Physics:** Calculate the character's position and velocity based on input and apply movement smoothly within the game environment.

**Collision Detection:** Ensure characters don’t pass through walls or other objects. Handle interactions with obstacles and terrain (e.g., slopes).

**Sprint and Stamina Management:** If sprinting is enabled, deplete stamina during sprints and regenerate stamina over time when not sprinting.

**Animation Triggers:** Based on movement speed and direction, trigger the appropriate animations (e.g., walk, run, idle).

## System Output

**Character Position Update:** Continuously update the character’s location within the game world.

**Display Feedback:** Output character movement animations and adjust the camera view to follow the character.

**Status Indicators**: Update and display any movement-related status (e.g., stamina bar if sprinting)..

## Other

**Camera Control**: Optionally, the camera follows or adjusts based on the character’s movement direction for a better player experience.

## Constraints

**Platform Compatibility:** Ensure movement controls work smoothly across all intended platforms (PC, consoles).

**Hardware Performance:** Movement mechanics must perform consistently across different hardware configurations, maintaining at least 30 FPS.

## Data Handling

**Input Validation:** Only registered inputs (e.g., W, A, S, D, joystick directions) are processed for movement to prevent unintended actions.

**Data Format:** Input values are processed in standard ASCII for keyboard controls or controller signal encoding.

## Error Handling

**Input Loss:** If input is lost (e.g., disconnected controller), pause the character’s movement and notify the player.

**Invalid Input:** Ignore any unintended or unassigned keys.

**Collision Errors:** If a collision error occurs, prevent the character from moving through solid objects and reset position if necessary.

# REQ-1.2

## Description

The game map should be generated randomly and continuously to create a dynamic and varied environment. To optimize system performance, map items and props (e.g., trees, rocks) should be removed from memory once they move out of the player’s view System Input.

## System Input

**Seed/Randomization Input: Seed value or random function call to initialize map generation.**

**Player Position**: Continuously track the player’s position to determine the visible map area.

## Display

**Map Visibility**: Display map elements (terrain, props) only within the visible range around the player

**Seamless Transitions**: Ensure map tiles load and unload seamlessly as the player moves, with no visual lags or stuttering.

## System Processing

**Random Map Generation:** Generate map tiles on-the-fly based on a random seed or procedural algorithm. This generation should cover terrain, props, and any interactable items.

**Culling for Optimization:** Implement a culling function to continuously detect which props and map elements are outside of the player’s field of view.

## System Output

**Generated Map Layout:** The map layout and all generated elements are displayed based on player location and visibility.

**Active Elements**: Only elements within the player’s visible area are rendered and kept in memory, reducing overall system load.

## Constraints

**Platform Compatibility:** Ensure movement controls work smoothly across all intended platforms (PC, consoles).

**Hardware Performance:** Movement mechanics must perform consistently across different hardware configurations, maintaining at least 30 FPS.

## Data Handling

**Reusability:** When tiles go out of view, retain their procedural data for fast reloading if they become visible again.

## Error Handling

**Map Loading Failures:** If map generation fails, the system should retry with a default or backup tile layout.

**Unloading Issues:** If props or tiles do not unload properly, log these items for debugging and continue to render only visible elements.

# External Interface Requirement

|  |  |  |
| --- | --- | --- |
| **Req. ID** | **Requirement Name** | **Requirement Description** |
| REQ-1.1 | Implementation of the game music. | Use software to upload and download music, create database to store music file; user could also change the different music any time in the game. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# REQ-1.1

## Data Interfaces

**Purpose**: Store all game music files, including background tracks, action cues, and victory/defeat sounds.

**Interaction**: The game retrieves specific music tracks based on in-game events.

**Data Format**: Music files store in a compressed format.

## User Interfaces

**Purpose**: Enable players to adjust game music volume or mute/unmute.

**Interaction**: Available in the settings menu, allowing players to configure their preferred audio experience.

**Display**: Show volume controls as sliders and options to toggle music on or off.

## Other Interfaces

**Interface Type**: Software interface to an audio engine, we use the Unity.

**Requirements**: Support for looping, cross-fading, and event-based triggers like starting battle music upon enemy encounter

**Error Handling**: If an audio file is missing or corrupted, the system should skip playback for that file and log an Error.

# Non-Functional Requirements

## System Performance

This section describes the system performance requirements. Example response times are suggested though different systems requirements may apply.

|  |  |  |
| --- | --- | --- |
| **Req. Id** | **Description** | **Response Time** |
| PERF-1.1 | In between levels, to maintain a seamless user experience. | 2 second |
| PERF-1.2 | Time for opening a new screen in the window. | 1 seconds |
| PERF-1.3 | Time for saving the current game data. | 0.5 seconds |
| PERF-1.4 | Maintain a minimum frame rate of 30 FPS to ensure smooth gameplay, with an ideal target of 60 FPS | / |
| PERF-1.5 |  |  |

## Information Security

This section describes the information security requirements.

|  |  |
| --- | --- |
| **Req. Id** | **Description** |
| SEC-1.1 | Player data (such as progress and settings) stored securely and encrypted. |
| SEC-1.2 | Data covered by the Data Protection Act shall not be held in environments that have not passed full system testing. |
| SEC-1.3 | Any personal data collected was complied with privacy standards, ensuring no unauthorized access. |
| SEC-1.4 |  |

## Availability

This section describes the availability requirements in terms of days and permissible planned and unplanned unavailability.

|  |  |
| --- | --- |
| **Req. Id** | **Description** |
| AVA-1.1 | The game’s services (emultiplayer servers, if applicable) should be available 99% of the time. |
| AVA-1.2 | Scheduled maintenance windows should be communicated to users in advance. |
| AVA-1.3 |  |
| AVA-1.4 |  |

## Capacity

|  |  |
| --- | --- |
| **Req. Id** | **Description** |
| CAP-1.1 | The system must be able to manage the following data volumes in year ONE.   * 5000 |
| CAP-1.2 | The system must be able to manage the following data volumes in year TWO.   * 25000 |
| CAP-1.3 | The system must be able to manage the following data volumes in year THREE.   * 100000 |
| CAP-2.1 | The system must be able to manage the following concurrent user operation volumes in year ONE.   * 800 |
| CAP-2.2 | The system must be able to manage the following concurrent user operation volumes in year TWO.   * 2500 |
| CAP-2.3 | The system must be able to manage the following concurrent user operation volumes in year THREE.   * 10000 |
| CAP-3.1 |  |