**5.0 Requirements Specification**

**5.1 Introduction**

**5.1.1 Identification**

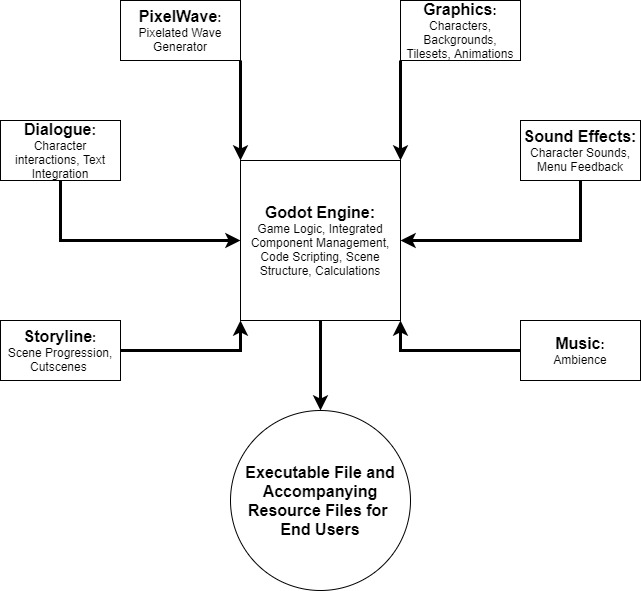
This Software Requirements Specification (SRS) documents the requirements for the envisioned Godot Engine game, *Cosmonarium*.

**5.1.2 Project Overview**

*Cosmonarium* is a turn-based RPG developed for desktop and laptop computers in the Godot Engine (ver. 3.2.2). The game aims to emulate the style of Super Nintendo JRPGs of the 1990s, and takes heavy inspiration from games such as Earthbound, Final Fantasy VI, and OneShot. The project integrates a written storyline, dialogue, pixel graphics, music, and sound effects into the engine via a multitude of programs. *Cosmonarium* will be programmed using Godot’s proprietary programming language, GDScript, and will use many of the built-in features that the Godot Engine provides to aid developers. This project also includes programmatically generated battle backgrounds. Pixelated waveforms will be generated by PixelWave, an auxiliary program that will be developed using Python. Following the incorporation of the components to be listed in this document, the executable file and the supporting files comprising the finished game will be exported from the Godot Engine and made accessible to interested parties who are using Windows, Mac, and Linux operating systems. Users will be able to interact with the game using either pure keyboard inputs or by connecting a compatible external video game controller via USB or Bluetooth.

**5.1.3 High-Level System of Components**

The following diagram briefly illustrates at a high-level the components necessary to fully implement the project (placeholder diagram):

****

**5.1.4 Document Overview**

The remainder of this document delves deeper into the project’s structure of subsystems and the relationships between individual components. Section 5.2 breaks down the functional requirements for the project by CSC. Section 5.3 provides meticulous details on how the subsystems of each CSC should act so that the project functions properly. Section 5.4 outlines how the subsystems of each CSC must act so that the project operates properly. Section 5.5 describes the software and hardware elements necessary for the development and execution of the project.

**5.2 CSCI Component Breakdown**

CSCI *Cosmonarium* is composed of the following CSCs:

5.2.1 Godot Engine / Game Management CSC

5.2.1.1 Title Screen CSU

5.2.1.2 Game Mode Manager CSU

5.2.1.3 Input Manager CSU

5.2.1.4 Scene Transition Manager CSU

5.2.1.5 Music Sequence Manager CSU

5.2.1.6 Menu Manager CSU

5.2.2 PixelWave CSC

5.2.3 Graphics / Graphical Interfaces CSC

5.2.3.1 Player Character Sprites CSU

5.2.3.2 Enemy Character Sprites CSU

5.2.3.3 Non-Player Character Sprites CSU

5.2.3.4 Background Elements CSU

5.2.3.5 Menu Interface CSU

5.2.3.6 Battle System Interface CSU

5.2.4 Story CSC

5.2.4.1 Story Elements CSU

5.2.5 Dialogue CSC

5.2.5.1 Dialogue Display Interface CSU

5.2.6 Music CSC

5.2.6.1 Musical Elements CSU

5.2.7 Sound Effects CSC

5.2.7.1 Sound Effect Elements CSU

**5.3 Functional Requirements**

The completed *Cosmonarium* game will offer a unique gaming experience for users to enjoy. Alongside the opportunities to explore a virtual landscape and fight against challenging AI opponents, the game will provide a distinctive storyline, witty banter, intriguing visuals, a complete soundtrack, and fitting sound effects.

**5.3.1 Godot Engine / Game Management**

The Godot Engine exists as the focal point of the entire project. Besides coalescing the external assets into a final product, the engine manages all the necessary behind-the-scenes processes for running the actual game. The engine accepts and interprets user inputs in order to facilitate player movement and battle command decisions, and provides a wide array of tools to aid developers who wish to spend less time jumping through technical hurdles and more time implementing their envisioned game logic.

**5.3.1.1 Title Screen**

5.3.1.1.1 The Title Screen shall always be the very first portion of the game displayed to the user.

5.3.1.1.2 The Title Screen shall be displayed to the user in a new window upon launching the project’s main executable file.

5.3.1.1.3 The Title Screen shall display options that the user can select from in order to begin gameplay.

5.3.1.1.3.1 The Title Screen shall display a button in a fixed screen position for starting a new game file.

5.3.1.1.3.2 The Title Screen shall display a button in a fixed screen position for loading a previously saved game file.

5.3.1.1.3.3 The Title Screen shall display a button in a fixed screen position for Configuration

Settings.

5.3.1.1.3.4 The Title Screen shall react to mapped keyboard inputs or controller commands on the aforementioned selectable options.

5.3.1.1.4 The Title Screen shall display the game’s main logo in a graphical layer behind the previously described buttons.

5.3.1.1.5 The Title Screen shall play the game’s main theme music in the background.

**5.3.1.2 Game Mode Manager**

5.3.1.2.1 The Game Mode Manager shall always begin regular gameplay in exploration mode, regardless of whether a new game is selected or a previously saved file is loaded.

5.3.1.2.2 The Game Mode Manager shall switch from exploration to combat when signaled by the currently loaded scene.

The switch to combat should occur when player and enemy characters collide.

The switch to combat will also be made during some cutscene scenarios.

5.3.1.2.3 The Game Mode Manager shall switch from combat to exploration when signaled by the currently loaded scene.

The switch to exploration should occur following the successful completion of combat.

The switch to exploration will also be made following some cutscene scenarios.

**5.3.1.3 Input Manager**

5.3.1.3.1 The Input Manager shall convert mapped keyboard keystrokes or controller button presses to executable actions during gameplay.

5.3.1.3.1.1 The Input Manager shall initially accept directional keys or directional pad presses as inputs for movement.

The accepted keys or button presses for movement should be configurable in the game’s settings.

5.3.1.3.1.2 The Input Manager shall initially accept directional keys or directional pad presses as inputs for scrolling through menus.

The accepted keys or button presses for menu traversal should be configurable in the game’s settings.

5.3.1.3.1.3 The Input Manager shall initially accept identified keys or button presses as inputs for

accepting choices, rejecting choices, and opening menu panels.

The accepted keys or button presses for the actions listed above should be configurable in the game’s settings.

5.3.1.3.2 The Input Manager shall prevent user inputs from affecting character movements during cutscenes.

5.3.1.3.3 The Input Manager shall ensure that only one object can accept input at a time, and if

multiple objects in the scene can accept input, the priority input receiver will be favored.

**5.3.1.4 Scene Transition Manager**

5.3.1.4.1 The Scene Transition Manager shall swap scenes as directed by the underlying game scene tree.

5.3.1.4.2 The Scene Transition Manager shall ensure smooth transitions from gameplay to

predetermined cutscenes.

5.3.1.4.3 The Scene Transition Manager shall ensure smooth transitions from predetermined

cutscenes back to gameplay, such as preventing movement while the screen is fading to

and from black.

**5.3.1.5 Music Sequence Manager**

5.3.1.5.1 The Music Sequence Handler shall facilitate the changing of music tracks between different relevant game scenes.

5.3.1.5.2 The Music Sequence Handler shall maintain the playback of the same music track during similar relevant game scenes.

5.3.1.5.3 The Music Sequence Handler shall facilitate the changing of music tracks within cutscenes as needed.

5.3.1.5.4 The Music Sequence Handler shall smoothly loop music tracks within scenes as needed to maintain the proper mood for each scene.

**5.3.1.6 Menu Manager**

5.3.1.6.1 The Menu Manager shall produce in-game context menus as needed for user interactions.

The Menu Manager should work in tandem with the input manager to produce menus on demand.

The Menu Manager should automatically produce menus during and after battle phases.

The Menu Manager should produce menus relevant to interacted in-game objects.

5.3.1.6.2 The Menu Manager shall facilitate the changing of features to the settings of the user's preference.

The features should be organized by sensory interaction.

The settings should be persistent.

5.3.1.6.3 The Menu Manager shall facilitate the player's access to in-game property.

The property should be organized by categories.

The property should be interactable.

5.3.1.6.4 The Menu Manager shall include a Debug Menu to facilitate play testing.

The Debug Menu should be hideable at a production level.

The Debug Menu should not be accessible to non-testers.

**5.3.2 PixelWave**

PixelWave acts as an auxiliary mechanism for generating battle backgrounds from music files. Taking in a music file as input, PixelWave will process the data and output a low-res live spectral waveform in a series of image files or as an animation. All the waveform files generated will be utilized as moving graphical backgrounds during the battles in-game while the corresponding original music files are played.

5.3.2.1 The PixelWave application shall produce live waveforms for use in the game’s battle sequences.

The waveforms should be as low resolution as possible to ease integration into the game.

The waveforms should be output into an animated file of some kind, e.g. a .gif or .bmp file, or alternatively output a series of .png images.

5.3.2.2 The PixelWave application shall accept a music file as a parameter for processing.

5.3.2.3 The PixelWave application itself shall be kept separate from Godot programming processes.

PixelWave will likely be built as a web application or discreet IO program on the side.

**5.3.3 Graphics / Graphical Interfaces**

The graphical subsystem of the project presents all the visual data being supported by the underlying game engine. The main items being displayed graphically include the main characters, any enemy characters, non-playable characters, props, tile maps, and backgrounds. In-game menus are also a key feature that the graphical subsystem provides visual access to, as is the battle system interface when the gameplay switches to combat mode.

**5.3.3.1 Player Character Sprites**

5.3.3.1.1 The Player Character Sprites shall be visible to the user in the overworld of the game during exploration mode.

5.3.3.1.2 The Player Character Sprites shall provide the user with a point of reference for his or her movements within the overworld.

5.3.3.1.3 The Player Character Sprites shall consist of the main controlled character as well as two other party members.

5.3.3.1.4 The main controlled character of the Player Character Sprites shall move around the overworld as the user inputs directional keyboard or button commands.

5.3.3.1.5 The two non-controlled party members of the Player Character Sprites shall follow the main controlled character by mimicking his or her movements on a timed delay.

5.3.3.1.6 Each member of the Player Character Sprites shall be animated to accurately depict movement when the main controlled character is commanded to move by the user.

5.3.3.1.7 The Player Character Sprites shall be able to collide with Enemy Character Sprites to initiate combat mode.

5.3.3.1.8 The Player Character Sprites shall be able to collide with designated Background Elements in order to restrict user movement to certain locations.

**5.3.3.2 Enemy Character Sprites**

5.3.3.2.1 The Enemy Character Sprites shall be visible to the user in the overworld of the game during exploration mode.

5.3.3.2.2 The Enemy Character Sprites shall have set movement behaviors when unaware of Player Character Sprites within the overworld.

Some Enemy Character Sprites should remain stationary in the overworld.

Some Enemy Character Sprites should patrol overworld corridors.

5.3.3.2.3 The Enemy Character Sprites shall chase the Player Character Sprites when the two entities come within a set distance of each other.

5.3.3.2.4 The Enemy Character Sprites shall be able to collide with the Player Character Sprites to initiate combat mode.

5.3.3.2.5 The Enemy Character Sprites shall be able to collide with designated Background Elements.

**5.3.3.3 Non-Player Character Sprites**

5.3.3.3.1 The Non-Player Character Sprites shall be visible to the user in the overworld of the game during exploration mode.

5.3.3.3.2 The Non-Player Character Sprites shall have set movement behaviors at all times, regardless of the locations of Player Character Sprites or Enemy Character Sprites.

Most Non-Player Character Sprites will remain stationary.

5.3.3.3.3 The Non-Player Character Sprites shall yield useful information to the player through interactions with the Player Character Sprites.

Some Non-Player Character Sprites should offer lore or story background information.

Some Non-Player Character Sprites should provide the player with hints or tips.

Some Non-Player Character Sprites should offer the user consumable or usable items for the Player Character Sprites.

**5.3.3.4 Background Elements**

5.3.3.4.1 The Background Elements shall act in conjunction with the positions of Player Character Sprites to help users orient themselves within the overworld during exploration mode.

5.3.3.4.2 The Background Elements shall mainly simulate real-world objects and scenes, such as hallways, rooms, walls, and boxes containing items.

The Background Elements will also consist of some science fiction themed objects that are inspired by but not necessarily instantiated in the real world.

5.3.3.4.3 Designated Background Elements shall restrict the Player Character Sprites from moving further in the direction of the given Background Elements upon collision.

5.3.3.4.4 Designated Background Elements shall restrict the Enemy Character Sprites from moving further in the direction of the given Background Elements upon collision.

**5.3.3.5 Menu Interface**

5.3.3.5.1 The Menu Interface shall be displayed when the user inputs the mapped keyboard key or controller button.

5.3.3.5.2 The open Menu Interface shall display selectable options in a panel that lead to further submenus.

5.3.3.5.2.1 Selecting the Overview option in the open Menu Interface shall display basic stats concerning the current Player Character Sprites’ HP and SP in a panel.

5.3.3.5.2.2 Selecting the Items option in the open Menu Interface shall allow the user to traverse through the Items submenu.

5.3.3.5.2.2.1 The open Items submenu shall display the consumable items currently held in the Player Character Sprites’ inventory in a panel.

5.3.3.5.2.2.2 The user shall be able to traverse through the Items submenu to select a specific consumable item for use on a specific Player Character Sprite, e.g. to restore HP.

5.3.3.5.2.3 Selecting the Equip option in the open Menu Interface shall allow the user to traverse through the Equip submenu.

5.3.3.5.2.3.1 The open Equip submenu shall display the equipment currently equipped by the Player Character Sprites in a panel.

5.3.3.5.2.3.2 The open Equip submenu shall display the equipment currently unequipped in the Player Character Sprites’ inventory in a panel.

5.3.3.5.2.3.3 The open Equip submenu shall additionally display information in a panel near the bottom of the screen about each piece of equipment as it is currently highlighted by the user.

5.3.3.5.2.3.4 The user shall be able to traverse through the Equip submenu to select a specific piece of equipment to equip to a specific Player Character Sprite, e.g. equipping a baseball bat as a weapon.

5.3.3.5.2.3.5 The user shall be able to traverse through the Equip submenu to select a specific piece of current equipment to unequip from a specific Player Character Sprite, e.g. removing a previously equipped cloak from a defensive armor slot.

5.3.3.5.2.4 Selecting the Status option in the open Menu Interface shall display more detailed stats in a panel, including the status afflictions affecting the Player Character Sprites and character stats beyond simply HP and SP.

5.3.3.5.2.5 Selecting the Skills option in the open Menu Interface shall allow the user to traverse through the Skills submenu.

5.3.3.5.2.5.1 The open Skills submenu shall display all the Player Character Sprites’ skills that can be equipped, both learned and unlearned, in a panel.

Learned skills will be shown in regular text.

Unlearned skills will be shown in grayed-out text.

5.3.3.5.2.5.2 The user shall be able to traverse through the Skills submenu to select a specific skill to equip on a specific Player Character Sprite, e.g. equipping a Heal skill.

5.3.3.5.2.5.3 The user shall be able to traverse through the Skills submenu to select a specific skill to unequip from a specific Player Character Sprite, e.g. removing an unlearned Scan skill.

5.3.3.5.2.6 Selecting the Config option in the open Menu Interface shall allow the user to traverse through the Config submenu.

5.3.3.5.2.6.1 The open Config submenu shall display selectable options for editing game settings in a panel.

The user will be able to edit the mapped keys or buttons for controls.

The user will be able to change the game window resolution.

The user will also be able to return to the title screen by selecting the proper option here.

5.3.3.5.3 The selectable options in the open Menu Interface and corresponding submenus shall be traversable by pressing the mapped directional keys or buttons.

5.3.3.5.4 Each option in the open Menu Interface and corresponding submenus shall be selected by first traversing to it and subsequently pressing the mapped key or button for selection / acceptance.

**5.3.3.6 Battle System Interface**

5.3.3.6.1 The Battle System Interface shall be displayed when the game mode switches to combat.

5.3.3.6.2 The Battle System Interface shall visually focus on the enemy sprite or sprites confronting the player characters.

5.3.3.6.3 The Battle System Interface shall depict player characters as modules anchored towards the bottom of the screen that contain basic stats about each character (HP and SP).

5.3.3.6.3.1 Player character status ailments such as shock and incapacitation shall be indicated via graphical changes when applicable.

5.3.3.6.4 The Battle System Interface shall accept user inputs for character actions via menus.

5.3.3.6.4.1 The character action menus shall appear above their respective character modules when it is the player’s turn to plan battle actions.

5.3.3.6.5 The Battle System Interface shall play back the live spectrum waveforms in the background of their corresponding battle scenes.

5.3.3.6.6 In-battle notifications shall be displayed towards the top of the screen as events occur over the course of battles.

5.3.3.6.7 A game over screen shall be displayed when all player characters are incapacitated within the battle.

5.3.3.6.7.1 The game over screen shall provide options for the user to return to the title screen or reload from the last save point.

**5.3.4 Story**

The game’s plot drives the scenes and gameplay forward. *Cosmonarium’s* lore is revealed through careful searching of the environment during exploration mode and via information supplied during cutscenes. The story acts as a means to further draw users into an enjoyable gameplay experience by adding relatable characters, plot twists and interesting concepts to ponder.

**5.3.4.1 Story Elements**

5.3.4.1.1 The Story Elements shall form a narrative sequence of scenes for integration with the Godot Engine.

5.3.4.1.2 The Story Elements shall offer a semi-linear narrative structure with no opportunities for branching story paths.

5.3.4.1.3 The Story Elements shall be revealed to the user in such a way that a basic understanding of the plot is conveyed even if the user performs the minimum amount of exploration in-game.

5.3.4.1.4 The Story Elements shall hide additional optional plot information throughout the game for curious users.

5.3.4.1.4.1 Extra lore shall be discoverable in certain conversations with Non-Player Characters and the acquisition of other in-game documents.

5.3.4.1.5 The Story Elements shall reveal plot points primarily during cutscenes and exploration mode.

The Story Elements should also reveal some relevant story information during major battles occurring in combat mode.

**5.3.5 Dialogue**

Dialogue conveys story points and engages users with the fictional world that has been designed in *Cosmonarium*. The use of conversations between characters adds context to plot exposition while also helping players grow more attached to the characters they are controlling and interacting with on-screen.

**5.3.5.1 Dialogue Display Interface**

5.3.5.1.1 The Dialogue Display Interface shall facilitate player-NPC interactions.

The Dialogue Display Interface should only be visible during exploration mode.

5.3.5.1.2 The Dialogue Display Interface shall display such interactions via on-screen text.

5.3.5.1.3 The Dialogue Display Interface shall reside towards the lower half of the displayed game screen.

5.3.5.1.4 The Dialogue Display Interface shall progress through dialogue text during gameplay via button commands input by the player.

5.3.5.1.5 The Dialogue Display Interface shall progress through dialogue text during cutscenes via button commands input by the player.

**5.3.6 Music**

Music underscores all the actions of the game in both exploration and combat mode. The project’s soundtrack seeks to set the mood in every scene in order to draw the user more fully into the world of *Cosmonarium*.

**5.3.6.1 Musical Elements**

5.3.6.1.1 The Musical Elements shall produce music tracks for integration with the Godot Engine.

5.3.6.1.2 The Musical Elements shall produce music tracks in the OGG file format.

5.3.6.1.3 The Musical Elements shall set the appropriate tone based on how the story is unfolding and which game mode is currently activated.

Faster paced tracks should be played during combat mode to get players excited.

Somewhat calmer tracks should be played during exploration mode to contrast with the tracks of combat mode.

**5.3.7 Sound Effects**

Sound effects add character and flavor to the gameplay by producing recognizable notes or noises. These effects also provide important feedback to players by indicating whether or not certain in-game actions or events have occurred, e.g. whether a menu option was selected or if a player has taken damage.

**5.3.7.1 Sound Effect Elements**

5.3.7.1.1 The Sound Effect Elements shall play appropriate sounds to provide audible feedback to the user on menu screens / panels.

5.3.7.1.1.1 The Sound Effect Elements shall play an appropriate noise when a menu option is confirmed by the user.

5.3.7.1.1.2 The Sound Effect Elements shall play an appropriate noise when a menu option is rejected by the user.

5.3.7.1.2 The Sound Effect Elements shall play appropriate sounds to provide audible feedback to the user during battle sequences in combat mode.

5.3.7.1.2.1 The Sound Effect Elements shall play an appropriate noise to indicate when a player character has taken damage.

5.3.7.1.2.2 The Sound Effect Elements shall play an appropriate noise to indicate when an enemy character has taken damage.

5.3.7.1.2.3 The Sound Effect Elements shall play an appropriate noise to indicate which types of attacks are being chosen.

**5.4 Performance Requirements**

**5.4.1 Runs at 60 FPS.** The game shall have a consistent frame rate due to the resolution and lightweight design of its pixelated graphics.

**5.4.2 Quick Scene Transitions.** The game shall be able to transition between scenes at a pace acceptable to the human attention span. Users should not be left waiting in front of an empty transitionary screen for extended periods of time.

**5.4.3** **Easy Menu Navigation**. Users shall rapidly be able to intuit and understand how menus and submenus are ordered.

**5.4.4 Game Session Crash Recovery.** The game shall successfully reload the most recent saved data regardless of whether the game session terminated via user command or unexpected errors.

**5.4.5 Accessibility for English Readers.** The game shall be accessible for users with a basic understanding of the English language.

**5.5 Project Environment Requirements**

**5.5.1 Development Environment Requirements**

**5.5.1.1 Hardware Requirements**

|  |  |
| --- | --- |
| **Category** | **Requirement (minimum)** |
| CPU | Any modern CPU - 2-4 cores |
| Hard Drive Space | 4+ GB |
| RAM | 4+ GB |
| GPU | AMD Radeon HD 7000 series  NVIDIA GeForce 8 series  Intel 3rd generation (Ivy Bridge) series |

The Godot Engine typically uses around 4 GB of RAM as a baseline for a project, though more RAM may be utilized depending on the complexity of any given project design. Since *Cosmonarium* is a 2D game, the initial 4 GB of RAM should be adequate for running the editor. The required 4+ GB of hard disk space is necessary for downloading and storing the FL Studio software for music production, as well as the other pieces of auxiliary software. The GPUs (or their equivalents) must be powerful enough to support OpenGL 3.0.

**5.5.1.2 Software Requirements**

|  |  |
| --- | --- |
| **Category** | **Requirement (minimum)** |
| Operating System | Windows 7, macOS 10.10, Linux |
| Graphics | OpenGL 3.0 support |
| Auxiliary Software | Pro Motion NG  FL Studio  Famitracker  MuseScore  Metalogue |

**5.5.2 Execution Environment Requirements**

**5.5.2.1 Hardware Requirements**

|  |  |
| --- | --- |
| **Category** | **Requirement (minimum)** |
| GPU | AMD Radeon HD 7000 series  NVIDIA GeForce 8 series  Intel 3rd generation (Ivy Bridge) series |

**5.5.2.2 Software Requirements**

|  |  |
| --- | --- |
| **Category** | **Requirement (minimum)** |
| Operating System | Windows 7, macOS 10.10, Linux |
| Graphics | OpenGL 3.0 support |