Fall 2024

COMP 302

gameName

R1M1

Group: groupName

Enes Ak – 80090

Muhammed Babelli - 84342

Yusuf Cemâl Karataş - 83639

İbrahim Cebecioğlu - 79906

Caner Kösem - 80246

Cemal Nişan – 79158

Table of Contents

Use Case Diagram

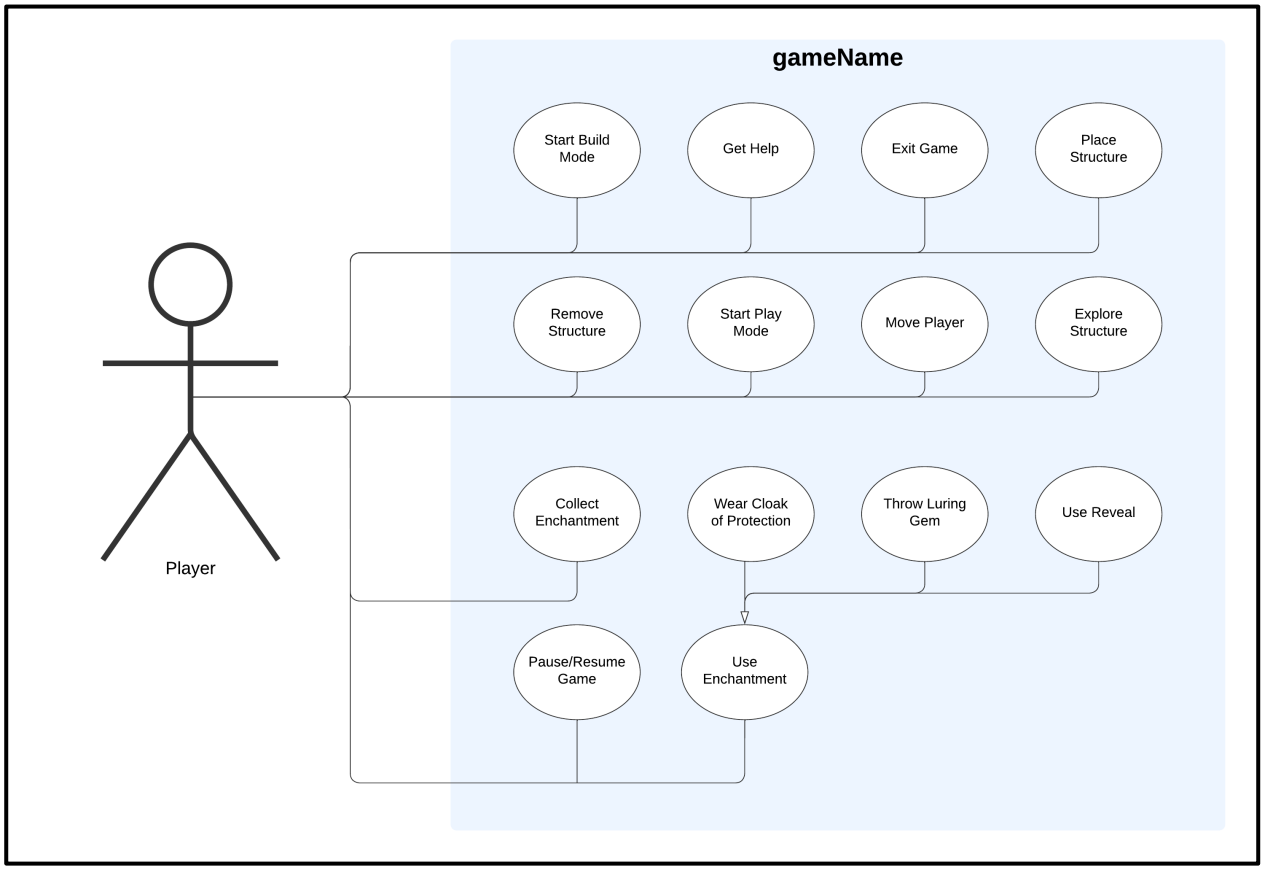

Figure 1: Use Case Diagram

Use Case Narratives

UCN 1: Move

Use Case Name	Move
Scope	Play Mode
Primary Actor	Player
Description	This use case describes the process by which the player attempts to move their character to a new location on the game map.
Pre-Condition/s	The game must be in play mode, not in build mode.
Post-Condition/s	The player's character is now standing at the location they intended to move to.
Main Success Scenario (Basic Flow)	**1.** The Player wants their character to move right, left, up, or down. **2.** The intended location is set as the player's character's new position.
Extensions (Alternate Flow/s)	**2.** **a)** Occupied location **1.** The game checks if the intended location is available. **2.** If the location is occupied, the player's character's position remains unchanged.

UCN 2: Place Structure

Use Case Name	Place Structure

Scope	Build Mode
Primary Actor	Player
Description	This use case describes the process by which the player places structures in the hall during build mode to prepare the game environment.
Pre-Condition/s	The game must be in build mode.
Post-Condition/s	The selected structure object is now placed at the chosen location in the hall grid.
Main Success Scenario (Basic Flow)	1. The player selects a structure from the Structure Storage. 2. The player chooses a tile in the hall. 3. Structure is placed at the chosen tile.
Extensions (Alternate Flow/s)	2. **a) Tile occupied:** 1. The game checks if the chosen grid cell is already occupied. 2. If occupied, the structure is not placed, and an error sound or visual notification is shown.

UCN 3: Remove Structure

Use Case Name	Remove Structure
Scope	Build Mode
Primary Actor	Player
Description	This use case describes the process by which the player removes a structure from the hall during build mode.

Pre-Condition/s	The game must be in build mode.
Post-Condition/s	The selected structure is removed from the hall grid.
Main Success Scenario (Basic Flow)	1. The player interacts with structure. 2. The game detects the structure at the chosen tile. 3. The game removes the structure from the tile.
Extensions (Alternate Flow/s)	None

UCN 4: Start Play Mode

Use Case Name	Start Play Mode
Scope	Build mode
Primary Actor	Player
Description	This use case describes the process by which the player transitions from Build Mode to Play Mode.
Pre-Condition/s	• The game is in build mode. • The player satisfies all the requirements of the build mode.
Post-Condition/s	The game is now in the Play Mode
Main Success Scenario (Basic Flow)	1. The game verifies that all build mode requirements (e.g., minimum number of structures in each hall) are satisfied. 2. The game transitions from Build Mode to Play Mode.

	3. The game initializes the first hall (Hall of Earth), placing the hero at a random starting location and preparing monsters, runes, and enchantments. 4. The game displays the Play Mode interface, including the hall name, remaining time, lives, and the hero's inventory.
Extensions (Alternate Flow/s)	3a) **Requirements not satisfied:** • The game checks if the build mode requirements are met. • If the requirements are not satisfied (e.g., a hall has fewer than the minimum required structures), the game prevents the transition to Play Mode. • The player is shown an error message indicating which requirements are not met.

UCN 5: Explore Structure

Use Case Name	Explore Structure
Scope	Play Mode
Primary Actor	Player
Description	This use case describes the process by which the player interacts with a structure in the hall to determine if it contains a rune.
Pre-Condition/s	• The game must be in play mode.
Post-Condition/s	If the structure contains a rune, it is revealed, and the door to the next hall is unlocked.
Main Success Scenario (Basic Flow)	1. The player navigates the hero to a tile adjacent to a structure in the hall. 2. The player interacts with the structure by clicking on it.

	3. The game verifies that the player is adjacent to the structure. 4. The game checks if the structure contains the rune. 5. If the structure contains the rune, it is revealed, and the door to the next hall is unlocked.
Extensions (Alternate Flow/s)	**3a) Player is not adjacent to the structure:** • The game prevents interaction and notifies the user. **4a) The structure does not contain the rune:** • The structure reveals no rune.

UCN 6: Use Enchantment

Use Case Name	Use Enchantment
Scope	Play Mode
Primary Actor	Player
Description	This use case describes the process by which the player uses a collected enchantment to gain an advantage or overcome a challenge in the game.
Pre-Condition/s	• The game must be in play mode. • The desired enchantment must be present in the player's inventory.
Post-Condition/s	The selected enchantment is activated, and its effect is applied to the hero or the game environment.
Main Success Scenario (Basic Flow)	1. The player activates an enchantment from the player's inventory. 2. It is verified that the enchantment is present in the inventory. 3. The game applies the enchantment's effect.

Extensions (Alternate Flow/s)	**2. a) If the enchantment is not present in the inventory** • Nothing is going to happen.

UCN 7: Collect Enchantment

Use Case Name	Collect Enchantment
Scope	Play Mode
Primary Actor	Player
Description	This use case describes the process by which the player collects an enchantment from the hall tiles to use it immediately or store it for later use.
Pre-Condition/s	• The game must be in play mode. • The enchantment must be present on the tile.
Post-Condition/s	The enchantment is collected, and its effect is either applied immediately or stored in the player's inventory for later use.
Main Success Scenario (Basic Flow)	1. The player clicks on the enchantment. 2. The game verifies the clicked tile contains an enchantment. 3. The game removes the enchantment from the grid. 4. The game applies the enchantment effect immediately (for instant effects) or stores it in the hero's inventory for later use.
Extensions (Alternate Flow/s)	**3a) The clicked tile does not contain an enchantment:** • No action is performed. **4a) Enchantment disappears before collection:** • The game checks if the enchantment has expired.

	• If expired, the enchantment is no longer collectable, and is removed from the tile.

Domain Model

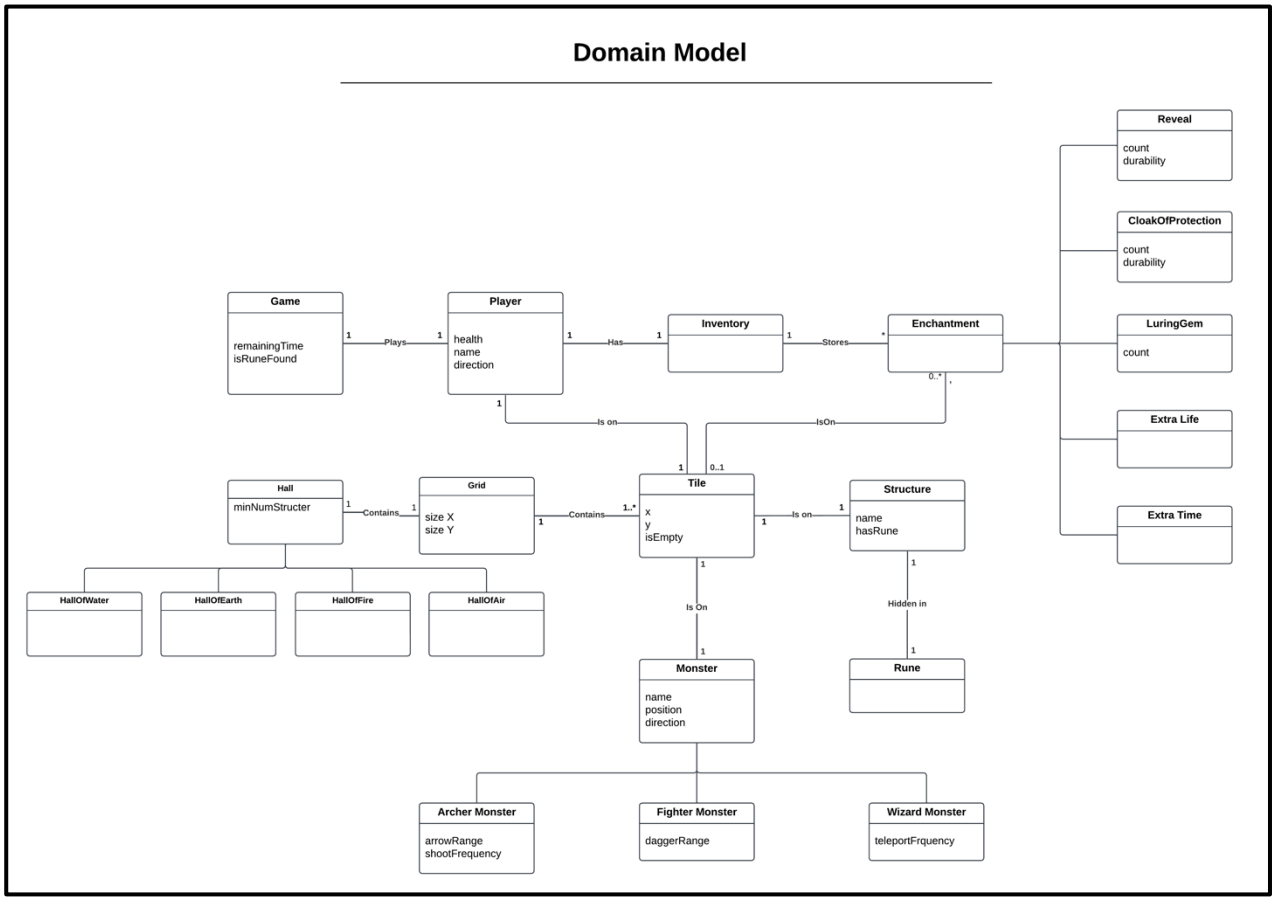

Figure 2: Domain Model

System Sequence Diagrams

SSD 1: getHelp

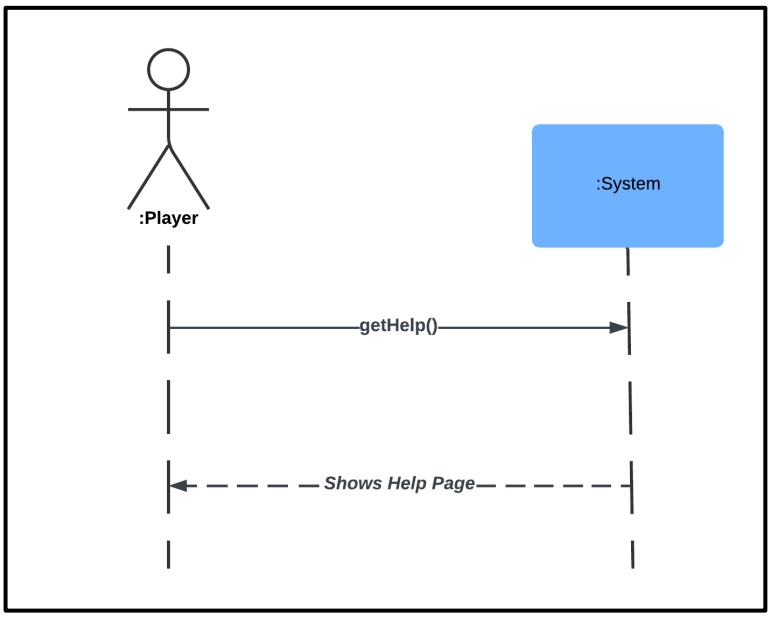

Figure 3: SSD 1 getHelp

SSD 2: collectEnchantment

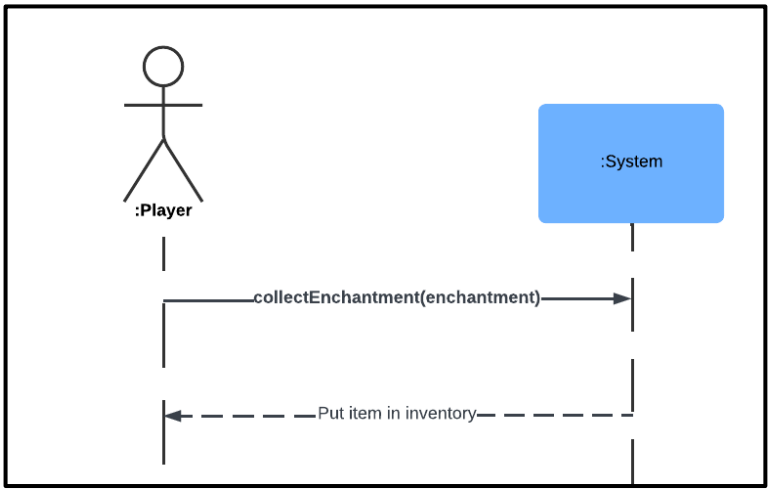

Figure 4: SSD 2 collectEnchantment

SSD 3: exploreStructure

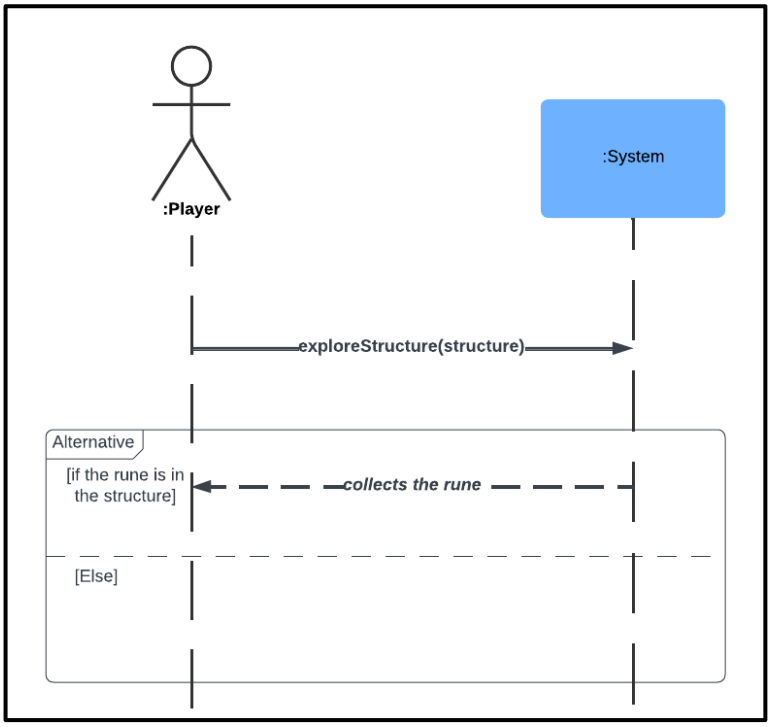

Figure 5: SSD 3 exploreStructure

SSD 4: pauseGame

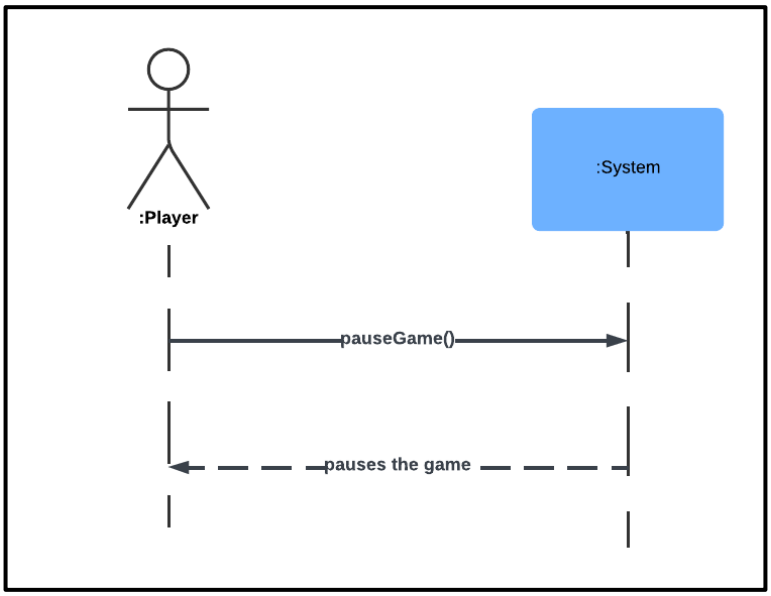

Figure 6: SSD 4 pauseGame

SSD 5: useEnchantment

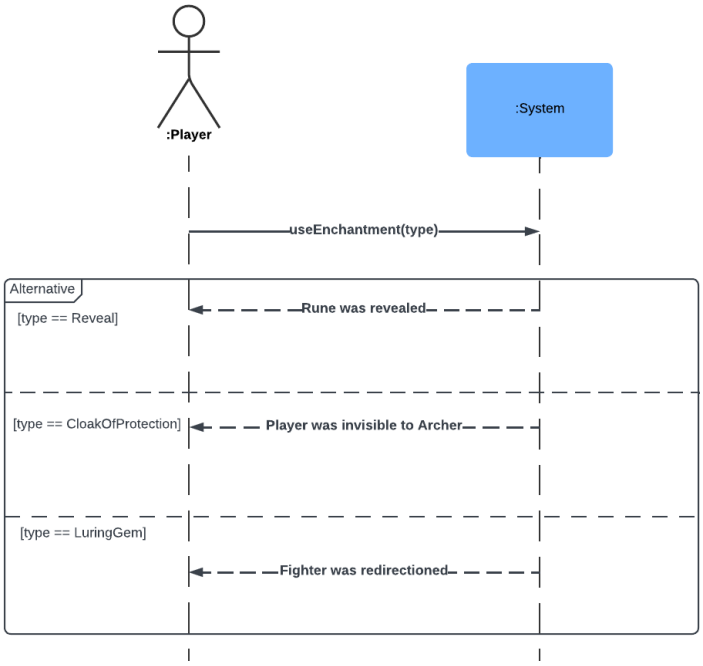

Figure 7: SSD 5 useEnchantment

SSD 6: placeStructure

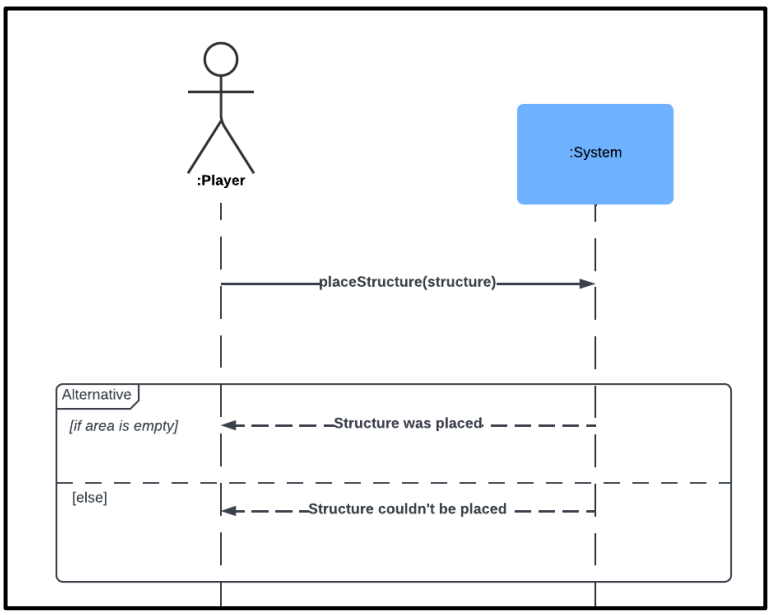

Figure 8: SSD 6 placeStructure

SSD 7: removeStructure

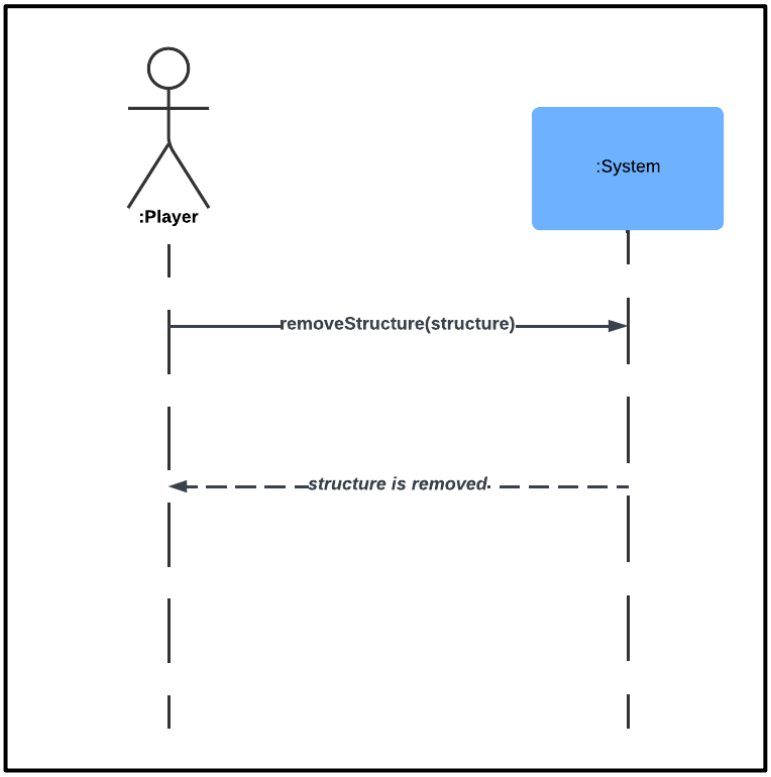

Figure 9: SSD 7 removeStructure

SSD 8: resumeGame

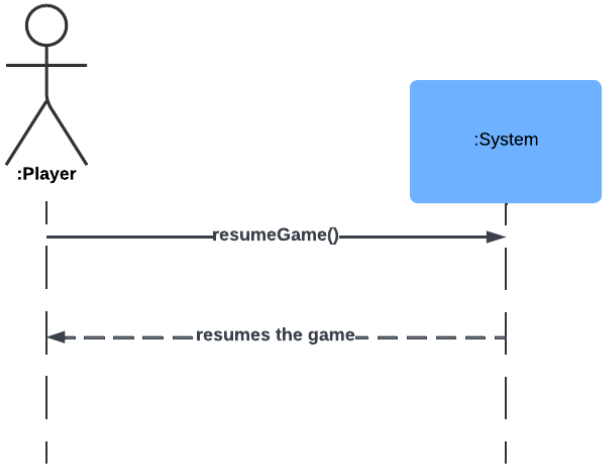

Figure 10: SSD 8 resumeGame

Operation Contracts

OC 1: collectEnchantment

Operation	collectEnchantment(location: Tile)
Cross References	Use Cases: Collect Enchantment
Preconditions	• The Game is in the Play Mode. • There is an Enchantment in *location* in question.
Postconditions	• The Enchantment was added to inventory or directly used. • The Enchantment was removed from the *location*.

OC 2: exploreStructure

Operation	exploreStructure(structure: Structure)
Cross References	Use Cases: Explore Structure
Preconditions	• The Game is in the Play Mode. • There is a Structure in *location* in question. • Player must be within the one tile of the Structure.
Postconditions	• The rune was revealed if it was in this *structure.*

OC 3: movePlayer

Operation	movePlayer(direction: Enum)
Cross References	Use Cases: Move Player
Preconditions	• The game is in the Play Mode. • Game is in Play Mode
Postconditions	• Location of the player was updated.

OC 4: useEnchantment

Operation	• useEnchantment(type: Enchantment)
Cross References	• Use Cases: Use Enchantment
Preconditions	• The game is in the Play Mode. • Player has the *enchantment* of type *type* in his inventory.
Postconditions	• Player was put under the effect of the *enchantment* of type *type*. • The enchanment *in question* was removed from the inventory.

OC 5: placeStructure

Operation	placeStructure(itemID: itemID, location: Location)
Cross References	Use Cases: Place Structure
Preconditions	• Game is in build mode
Postconditions	• The structure with the itemID was placed in the location.

OC 6: exitGame

Operation	exitGame()
Cross References	Use Cases: Exit the Game
Preconditions	• You are already in the game
Postconditions	• The game was terminated

OC 7: removeStructure

Operation	RemoveStructure(itemID: itemID, location: Location)
Cross References	Use Cases: Remove the Structure
Preconditions	• Game is in build mode • There is a structure in the *location.*
Postconditions	• Structure was removed from location • Location was made available

Vision

Introduction

We envision a next-generation, interactive, and immersive dungeon adventure game, *gameName*" designed to offer players an engaging experience of exploration, puzzle-solving, and strategy. The game combines dynamic gameplay mechanics, such as monster encounters and enchantment collection, with user-friendly customization features, enabling players to create their unique journey through customizable hall designs.

Positioning

Business Opportunity

The current landscape of 2D dungeon adventure games often lacks sufficient user control over the environment and gameplay mechanics. Existing games rarely provide players with the ability to design their own game environment or introduce strategic elements like enchantments and diverse monster behaviors. This creates a demand for a customizable game that balances user creativity with challenging gameplay. "*gameName*" addresses this gap by providing a unique dungeon adventure experience with a fully customizable hall-building mode, dynamic monster encounters, and strategic use of enchantments.

Problem Statement

Many adventure games lack modularity and fail to provide an experience designed to both casual and strategic players. This project aims to resolve these issues by integrating a grid-based environment that supports seamless game mechanics and strategic elements, while maintaining accessibility and visual appeal. Our approach targets both the entertainment of gaming by creating a scalable game that adheres to software engineering best practices.

Product Position Statement

"*gameName*" is a dungeon exploration game designed for adventure enthusiasts and gamers seeking an interactive, strategic, and customizable experience. Its standout features include:

- A hall-building mode that enables players to design game levels.

- Diverse monster interactions requiring strategic thinking.

- A range of enchantments that add depth to gameplay.

The game differentiates itself through its integration of design, exploration, and strategy in a single package.

Alternatives and Competition

Existing 2D dungeon games emphasize either aesthetics or gameplay complexity, often neglecting user-driven customization and modularity. "*gameName*" fills this gap by combining user creativity with an engaging, action-packed experience.

Stakeholder Descriptions

Players: Seek an immersive and customizable gaming experience.

Developers: Gain hands-on experience in implementing modular software systems.

Supplementary Specifications

Version	Date	Description	Author
Inception Draft	November 20, 2024.	First draft. To be refined during further development phases.	Project Team

Introduction

This document serves as the repository of all requirements for the "*gameName"* game that are not explicitly detailed in the use cases. It outlines essential functionality, usability standards, and non-functional requirements to ensure the development of a robust and engaging game. While the use cases focus on gameplay mechanics, this specification expands on the foundational elements required for a seamless user experience and scalability of the game.

Functionality

Logging and Error Handling

All errors, including invalid grid movements or overlaps, should be logged for debugging

purposes. Persistent storage should record these errors for further analysis and future improvements to the game.

Pluggable Rules

The game should support the ability to customize gameplay rules, such as modifying the time limits or adjusting difficulty settings, to enhance replayability and adapt to player preferences. These rules may be defined and executed at various points within gameplay.

Security

The game ensures that players cannot access the source code to manipulate it.

Usability

Human Factors

The user interface should be clear and easy to navigate for players of all skill levels. Game elements such as the player's inventory, remaining time, current hall, and player's lives must be prominently displayed and easily visible during gameplay.

The use of colors should be mindful of accessibility needs, such as providing a colorblind mode. The game should use alternative visual cues or textual hints for important events (e.g., finding a rune or losing a life) to ensure clarity for all players.

The gameplay experience should be smooth and intuitive, emphasizing strategic thinking and exploration. Audio and visual feedback, such as sound effects when the rune is found or time is running out, should enhance the immersive experience. Pausing, resuming, or exiting the game should be simple and immediately effective.

Performance

The game must maintain a smooth user experience throughout, ensuring that all animations and interactions are seamless. Gameplay should run without delays, even when multiple monsters, objects, or enchantments are active in the hall. Loading times between halls or transitions should be minimal to maintain player immersion.

Glossary

Player: The main character controlled by the player. The player explores the dungeon halls, searching for runes while avoiding monsters and collecting enchantments. The player begins the game with three lives, which can be replenished by collecting specific enchantments. The player is the key figure responsible for unlocking each hall and eventually escaping the dungeon in a limited time.

Rune: A magical artifact hidden within each hall. The player must locate the rune to unlock the exit and proceed to the next hall. Runes may be teleported to new locations by Wizards, adding an extra layer of challenge to the game.

Halls: The dungeon consists of four themed halls—Earth, Air, Water, and Fire. Each hall must be completed in sequence by finding its hidden rune within the given time limit. The design and object placement in each hall differ, designed by the player and also, each hall has their own requirements.

Structure: A type of object in the game that is placed on tiles within the grid. Structures serve as obstacles or strategic elements in the game. They can hide runes or influence the player's navigation, requiring players to plan their movements and actions around their placement.

Tile: The basic unit of the grid-based map in the game. Each Tile can host a structure, the player, a monster, or an enchantment. The player can move only on empty tiles. Tiles serve as the foundation for strategic movement and structure placement in the game.

Monsters: These are enemies that obstruct the player's progress and create challenges. There are three types of monsters:

1. **Archer Monster**: Shoots arrows every second if the player comes within four squares. The player can avoid detection by using the Cloak of Protection enchantment.
2. **Fighter Monster**: Attacks the player with a dagger when adjacent. It can be distracted using the Luring Gem enchantment.
3. **Wizard Monster**: Does not attack the player but teleports the rune to a random location every 5 seconds, making it harder to locate.

Enchantments: Special items that assist the player in various ways. They appear randomly in the halls and provide advantages like extra time, extra lives, or strategic abilities. Some enchantments are used immediately upon collection, while others can be stored in the player's inventory for later use.

Inventory: The player's inventory for storing collected enchantments. Stored enchantments can be used strategically during the game to overcome specific obstacles or challenges.

Grid: The dungeon map is divided into a grid of tile. Each tile can contain one object, such as the player, a monster, a wall, or an enchantment. The grid structure ensures organized and systematic movement for all game elements.

Timer: A countdown mechanism that sets a time limit for each hall. If the timer runs out before the rune is found, the game ends. Collecting the Extra Time enchantment can add seconds to the remaining time.

Build Mode: The phase where players design the interiors of the dungeon halls before starting the game. Players must place a minimum number of objects in each hall, creating opportunities for rune hiding and strategic gameplay.

Play Mode: The active phase of the game where the player explores the halls, avoids monsters, collects enchantments, and searches for the rune. The player's progress, remaining lives, and timer are displayed during this mode.

Luring Gem: A specific enchantment used to distract Fighter monsters. The player throws the gem in a particular direction, causing nearby Fighter monsters to follow it, allowing the player to move away safely.

Cloak of Protection: An enchantment that hides the player from the Archer monster's vision, enabling close proximity without losing lives. The effect lasts for a limited duration.

Extra Life: An enchantment that immediately adds one life to the player upon collection. This helps the player continue their journey if lives are lost during encounters with monsters.

Extra Time: An enchantment that adds additional seconds to the timer when collected. This effect is applied immediately upon collection, giving the player more time to search for the rune and complete the current hall.

Reveal Enchantment: A helpful enchantment that highlights a 4x4 area on the grid where the rune is hidden. The player can use it to narrow down the search area strategically.

Pause/Resume Button: A feature in the game interface that allows the player to pause the game at any time during play mode. The game can be resumed from the same state.

Main Menu: The initial screen that appears when the game is launched. It provides options to start a new game, access the help screen, or exit the game.

Help Page: A section of the game's interface that provides players with detailed information about gameplay mechanics, controls, objectives, and descriptions of game elements such as monsters, enchantments, and runes. It is designed to guide new players and assist them in understanding how to play the game effectively.