



**ÇANKAYA UNIVERSITY**  
**FACULTY OF ENGINEERING**  
**COMPUTER ENGINEERING DEPARTMENT**

**CENG 407**

Innovative System Design and Development I

Project Report

**Team ID: 202408**

**Rule and Munus / AI Based Game Development Project**

Dorukhan YILDIZ - 202011003

Sezer Can EKİZ - 202011034

Ömer YURTALAN - 202011050

Semih OĞUZ – 201911033

Advisors:

Doç. Dr. Gül TOKDEMİR

Dr. Öğr. Üyesi Talha KARADENİZ

Arş. Gör. Sezer UĞUZ

## Contents

COMPUTER ENGINEERING DEPARTMENT .....	1
Introduction .....	5
Project Plan.....	6
Literature Review .....	6
1. Abstract .....	7
2. Introduction .....	7
3. Evaluation of Game Development .....	7
4. AI in Game Development .....	8
5. What is a Game Engine .....	14
6. Statistical Analysis of Preferred Game Genres and Tools.....	20
7. Inspired and Similar Projects .....	26
8. Our Project Goals .....	27
9. Conclusion.....	28
Software Requirements Specifications.....	29
1. Introduction .....	29
1.1 Purpose of This Document .....	29
1.2 Problem Definition .....	29
1.3 Scope of This Project.....	29
1.4 Glossary .....	30
1.5 Overview .....	30
2. Overall Description .....	31
2.1 Product Perspective .....	31
2.1.1 Hardware Interfaces.....	31
2.1.2 Software Interfaces .....	32
2.1.3 User Interfaces.....	32
2.2 Product Functions.....	33
2.3 User Characteristics.....	33

2.4 Assumptions and Dependencies .....	33
3. Specific Requirements.....	34
3.1 External Interface Requirements .....	34
3.2 Functional Requirements.....	34
3.2.1 Peace and War Phase .....	34
3.2.2 Town Management .....	34
3.2.3 NPC Interactions .....	35
3.2.4 Character Attribute Points .....	35
3.2.5 AI Behaviours .....	35
3.2.6 Gathering and Loot.....	35
3.2.7 Roll a Dice.....	35
3.2.8 Combat Mechanics .....	36
3.2.9 Go to Coliseum.....	36
3.4 Performance Requirements .....	36
3.5 Design Constraints.....	37
3.6 Software System Attributes .....	37
3.6.1 Reliability .....	37
3.6.2 Availability .....	37
3.6.3 Security.....	38
3.6.4 Maintainability .....	38
3.6.5 Portability .....	38
3.6.6 Usability .....	38
3.6.7 Scalability .....	38
3.7 Safety Requirements.....	38
4. UML and Use Case Descriptions .....	39
4.1 Use Case Diagram .....	39
4.2 Use Case Descriptions .....	40
5. Planning.....	60

5.1 Team Structure.....	60
5.2 Estimated Schedule .....	60
5.3 Process Model .....	61
6. Conclusion.....	61
Software Design Description.....	62
1. Introduction .....	62
1.2 Purpose of this Document .....	62
1.3 Scope of the Project.....	62
2. Team Information .....	62
3. Glossary.....	63
4. Scope of the Project.....	63
4.1 Inspirations .....	63
4.2 Theme/Genre/Concept.....	64
4.3 Targeted Platforms and Audience .....	64
4.4 Monetization Model .....	64
5. Game Design .....	65
5.1 Story .....	65
5.2 Game Structure.....	65
5.3 Character.....	65
5.3.1 Player Definitions:.....	65
5.3.2 Player Properties.....	65
5.4 Actions .....	66
5.5 Objective.....	67
5.6 Gameplay.....	67
6. Software Design .....	67
6.1 Component-Based Architecture.....	67
6.2 Decomposition.....	67
6.2.1 Building Construction .....	67

6.2.2 Character Attributes .....	68
6.2.3 Combat Actions .....	68
6.2.4 Resource Management .....	68
6.2.5 Artificial Intelligence (AI) .....	68
6.3 Class Diagram .....	69
6.4 Activity Diagram .....	69
7. User Interface Design (UI) .....	70
7.1 Main Menu .....	70
7.2 Play .....	70
7.3 Credits.....	71
7.4 Settings .....	71
7.5 New Game .....	72
7.6 Peace Phase .....	72
7.7 War Phase .....	73
7.8 War Phase (Gathering).....	73
7.9 Combat (Interrupted Gathering) .....	74
7.10 Coliseum.....	74
7.11 Combat (Coliseum).....	75
References .....	75

## Introduction

This document is a combination of Literature Review (LR), Software Requirements Specifications (SRS) and Software Design Document (SDD). We reviewed literature based on the concept of our project and examined existing solutions at the LR. SRS part is a guideline of the overall specifications and functional requirements. The last part SDD provides information of the architectural design and conceptual overview of the project. By combining all documents, this report defines both design and implementation requirements.

## Project Plan

Start Date 30/09/2024	Current Status	WEEK 1	WEEK 2	WEEK3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14	WEEK 15	WEEK 16
Team Setup	Completed																
Project Proposal Form	Completed																
Project Selection Form	Completed																
GitHub Repository	Completed																
Project Work Plan	Completed																
Literature Review	Completed																
Software Requirements Specification	Completed																
Project Webpage	Completed																
Software Design Description	Completed																
Project Report / Project Tracking Form	In progress																
Presentation	Not Started																

## Literature Review

## **1. Abstract**

Our research work is in understanding and defining artificial intelligence in game development, with a core focus on AI techniques, machine learning, and reinforcement learning, that can improve the experience of players in gaming. We review the evolution of AI in gaming, including tools, game genres, and our core features. We discuss choosing Unity as the core game engine due to its advanced AI integration capabilities, cross-platform support, and scalability for complex AI systems. We also go into some industry statistics regarding popular game genres and development tools, such as TBS games, tactical RPGs, and roguelikes, to support strategic design choices. This review gives a basis for our AI game development project by reviewing the current projects in place with their features, pointing toward our objectives in innovation, placing in the market, and differentiation in the gaming industry.

## **2. Introduction**

In the last period, artificial intelligence blew on new winds into game development, upgrading traditional gameplay and innovative methods of holding the players' interest with adaptive gameplay, procedural content generation, and smart NPC behavior. This review intends to discuss several approaches, techniques, tools, and technologies at a modern game developer's disposal concerning the use of AI. We start by overviewing the existing AI-driven projects, discussing their goals, technologies, and common challenges. Another important focus of is the evaluation of various game development tools and game engines, comparing popular platforms such as Unity, Unreal Engine, CryEngine, and others, with a particular emphasis on Unity's AI capabilities. The analysis of preferred genres, tools, and design strategies is achieved through statistical data; hence, the understanding of market trends shall guide us in developing our project. This literature review further looks at those projects which acted as our inspirations and highlights areas we plan to improve. The end goal is to contribute to AI in gaming by working on a project that will integrate AI capabilities with an accessible and engaging user experience in gaming, especially for Steam users.

## **3. Evaluation of Game Development**

Game development has evolved through advances in design paradigms, tool development, and strategies for engaging players, resulting in a wide range of genres and mechanics. Most reviews today assess how technical complexity balances out the player's experience in genres such as turn-based strategies, roguelikes, and tactical RPGs, which rely on strategic depth and replayability. Iterative development, AI implementation to offer dynamic challenges, and user-oriented design are highly emphasized in research with respect to long-term engagement.

### **3.1. Aim of Existing Projects**

Most modern game projects try to engage players more and more with interactive storylines and decision-making mechanics. Games like "Loop Hero" focus on town-building and resource management, where progression is integral. In contrast, "Sid Meier's Civilization VI" is all about long-term strategy, enabling players to shape entire civilizations. Projects also try to include innovative AI, which is a challenge, as in the case of "Divinity 2: Original Sin 2," where NPCs and enemies adjust to player actions, thus enriching the game.

## 3.2 General Information on Tools, Genres, and Features

- **Tools:** Popular game engine Unity lets developers create complex mechanics and great visuals with wide scripting support; usually, C# is used in projects made with Unity.
- **Genres:** In turn-based strategy games and tactical RPGs, the key elements are usually player decision-making and resource management, often along with roguelike elements like randomization and permadeath to ensure replayability.
- **Features:** Key features include AI-driven combat, town-building mechanics, inventory management, and character customization. In "Rule and Munus," town development and strategic combat with dice rolls define player progression, mirroring features from games like "Swords and Sandals" and "Dungeons and Dragons."

## 4. AI in Game Development

### 4.1 What is AI?

AI is a technology area that outlays computers and machines to perform tasks that typically require human intelligence, including learning, comprehension, problem-solving, decision-making, creativity, and autonomy. [1] Not only does AI perform complex tasks, but it also updates itself, an evolutionary process of learning in nature. Nowadays, it is increasingly well-known to open new possibilities, especially in game design, character development, and player behavior analysis.

### 4.2 Machine Learning and Its Working Principle

Machine learning is a subset of artificial intelligence and computer science that imparts the abilities of human learning to machines using data and algorithms. A generic ML algorithm basically comprises three components: a decision process for pattern recognition, an error function to evaluate the accuracy of prediction, and a model optimization process that keeps adjusting weights to enhance the accuracy of the model. The algorithm iteratively updates itself based on training data to make more accurate predictions over time. [2]

#### 4.2.1 Reinforcement machine learning



The reinforcement machine learning model is similar to the supervised learning technique, except that the algorithm isn't trained using sample data. It learns while it goes by a method of trial and error. A series of rewards will be reinforced to build up the best recommendation or policy for a given problem. [2] Reinforcement learning is commonly used in games to train AI agents to make decisions in dynamic environments. By interacting with the game, the AI learns from the outcomes of its actions—receiving rewards for successful moves and penalties for mistakes. This learning process allows the AI to improve its strategies over time, making it adaptable enough to compete with human players. Video games are already using reinforcement learning, in which the AI continuously updates its behavior looking for the optimal performance.

### **4.3 What is AI for gaming?**

AI in gaming means the integration of techniques and technologies from artificial intelligence into video games to make gameplay more dynamic, responsive, and immersive. [3] AI in games is intended to provide players with a more interactive and realistic experience. AI controls the behavior of characters in the game; it responds appropriately and in real time to the actions of the player. This is more apparent in smart enemies and richer, more natural interactions with game worlds. At the same time, AI equips game designers with the capability to analyze game player behavior, and to enhance game content based on this. The process that enables adjusting game difficulty by a player's skills and to deliver personalized content, creating thus a unique game experience for the player.

### **4.4 The Evolution of AI in Games**

Early video games, such as Pac-Man (1980) and Space Invaders (1978), had utilized simple AI in generating usually totally predictable patterns that the enemies would make. These were basically the founding of AI in games, whereby the simple algorithms ran enemy movements that the AI responded to, using player inputs in a pattern-based reaction. In Pac-Man, for example, the four ghosts had different movement algorithms, each posing a different kind of challenge to the player. Because of the limitation of hardware at that time, the complexity of AI could only focus on creating challenging but predictable behaviors for enemies. [4]



**Figure 1:** Pac-Man Gameplay [5]

The 1990s began with the release of Civilization in 1991 and Starcraft: Brood War in 1998, both milestones in the development of AI within video games; each had different strategies and ways to approach game play. Civilization is a Turn-Based Strategy where the AI acts through scripted patterns but adjusts itself to the player's empire, reacting to military and tactical changes. On the other hand, Starcraft is a Real-Time Strategy game where the focus is on rapid decision-making and unit management, with the AI adjusting its tactics based on player actions but not learning or evolving through gameplay. [6]



**Figure 2:** Civilization game board [7]



**Figure 3:** Starcraft: Brood War [8]

In the 2000s, Forza Motorsport and World of Warcraft furthered the development of game AI in 2005 by using the same approach: behavior predefined. Forza's bots follow defined paths, and once hit, they will fix their trajectories. This is still not an example of adaptive behavior. In World of Warcraft, boss AI follows specific patterns and timers: reacting to the players but without learning or adapting for optimization of strategies. These games demonstrate scripted AI programmed to deliver challenging yet predictable experiences. [4]

More recently, reinforcement learning has been used to inject some dynamism into the way AI works in games, including recent titles like Red Dead Redemption 2 (2018) and Halo Infinite (2021). In the case of Red Dead Redemption 2, NPCs use RL in order to have their behavior evolve due to players' actions, offering a very realistic ever-changing world. Similarly, AI enemies in Halo Infinite learn from combat interactions and change their strategies to make the experience even more unpredictable and challenging. These games really drive home the point of how much RL is influencing the mold of adaptability and realism in modern gaming AI.



**Figure 5:** Red Dead Redemption 2 [9]





**Figure 6:** World of Warcraft [\[4\]](#)



**Figure 7:** Halo Infinite [\[10\]](#)

The evolution of artificial intelligence in games has progressed significantly alongside advancements in technology and game design. In the past, AI was primarily limited to simple commands and scripted behaviors, which resulted in predictable and static interactions within the game. Early AI systems could only execute basic tasks, such as following predefined paths and providing limited responses to player actions. These systems lacked the ability to adapt or offer dynamic gameplay experience. However, with the development of more sophisticated algorithms and increased computational power, AI in games has become much more complex. Today, AI can learn from player behavior, adapt to different situations, and provide a more immersive, dynamic, and personalized gaming experience.

#### **4.4 What Will We Use?**

In our game, we decided to use reinforcement learning techniques. That is why, over some period, the AI will continue developing depending on how the player acts, chooses, or generally develops his or her game. Precisely, that could be most applicable for the King of Gladiators, the final boss. For instance, the King of the Gladiators might initially be way stronger than the player; as the player advances and becomes strong, the AI evolves through observing what moves the player makes and thus adjusts his strategy inductively. Integration of reinforcement learning into our game: The AI will evolve with each progressing stage by the player because, as the player proceeds and wins levels, so will the King of Gladiators. For example, the AI might start out being 5 times stronger than the player, but in later stages of the game, it will evolve to be stronger and more strategic, matching the advancement of the player. Later stages could range from 10x to 3x multiplier, depending on the actions taken by both the player and the AI. This allows the AI to learn from what moves the player makes and adapt accordingly.

**Dynamic Decision Making:** The behavior of the King of the Gladiators will not be fixed; instead, the AI will reassess every fight and encounter in light of previous player actions. The AI strategies will change as the player overcomes different challenges, making each new battle harder and less predictable. This will create a more dynamic experience, much like in games such as Red Dead Redemption 2 and Halo Infinite, where AI adapts and responds to the actions of players. **Adaptation for Player Progression:** The AI will also learn from the items and resources that the player gains at each stage. As such, while the player gains new weapons or equipment, King of Gladiators learns how to counter the usage of such new tools. This will make every level more personalized and challenging. Finally, reinforcement learning will have AI improve at every stage, making for an opponent who is increasingly hard to play and adapt to. All this will be guaranteed to the players for a challenging realistic and thrilling experience.

## 5. What is a Game Engine






A game engine is a software framework that is utilized for creating video games; it contains necessary components, such as rendering, simulation of physics, processing of audio, artificial intelligence, and management of user interface. These components enable game developers to build interactive and visually appealing gaming experiences. Game engines simplify complex coding processes, which enables faster development. For instance, famous game engines include Unity, Unreal Engine, Godot. They are highly in demand due to their cross-platform abilities and well-functional tools for developing games [\[11\]](#) Game engines typically include these components:

- **Rendering System:** The rendering system is tasked with rendering the visual information within a game. This system converts the 2D or 3D models into displayable images and takes care of lighting, shading, and textures in producing realistic graphics.
- **Physics Engine:** The physics engine emulates the real-world physics that occur in a game, like gravity, collisions, and object interactions. It makes sure objects in the game move and respond to the game world in a natural, realistic way.
- **Animation System:** The animation system governs character and object movements; it allows the seamless transition between one pose or action and another. It's crucial for implementing realistic character behavior and interaction.
- **Audio System:** Game sounds, such as background music, sound effects, and voices over, are all part of the audio system. It will create a much more immersive audio environment in which the player can re-engage.
- **Artificial Intelligence (AI):** AI allows NPCs to think intelligently and react to events. It comprises various algorithms defining the actions, pathfinding, and decision-making by an NPC.
- **Scripting System:** The scripting system allows developers to program game mechanics and interactions using languages such as C# in Unity or C++/Blueprints in Unreal Engine.
- **UI System:** The UI system handles on-screen elements, such as menus, health bars, and scores, which give visual feedback to the player and allow him or her to navigate the game.
- **Networking System:** The networking system allows for online multiplayer, taking care of connections, data synchronization, and player interactions across a network.
- **Database and Asset Management:** Assets of the game, such as textures, models, and sounds, are handled along with the data of the game; it ensures proper loading and arrangement of resources.
- **Shader System:** The shader system grants developers the opportunity to achieve particular visual effects with the help of small programs running on the GPU.

- **Lighting System:** The lighting system provides different types of lighting, such as ambient, directional, and point lights, to create realistic environments with shadows and highlights.
- **Particle System:** The particle system models small, dynamic objects like smoke, fire, explosions, or snow, creating visual effects that enhance the atmosphere of your game.
- **Navigation System:** The navigation system allows NPCs to move around the game world using pathfinding algorithms that find the most optimal paths around obstacles.
- **Cinematics and Cutscene Tools:** These tools are responsible for developing story sequences or cutscenes that add more narrative depth, where the camera angle and character movements are controlled by the developer.
- **Post-Processing Effects:** It involves the addition of camera effects such as blurring, color grading, and contrast to make the game look more cinematic.
- **Input System:** This is the management of data from keyboards, mice, controllers, and other devices and mapping them to specific in-game commands.
- **Profiling and Optimization Tools:** These tools allow developers to analyze game performance, identify resource-intensive components and improving game efficiency.
- **Asset Pipeline:** The asset pipeline allows developers to import content from external sources (like 3D modeling software) into the game engine and manage these resources efficiently.
- **Platform Support:** Platform support allows developers to compile and optimize the game for various platforms (PC, consoles, mobile), ensuring it performs well on each.

Gregory, J. (2014). Game Engine Architecture. A K Peters/CRC Press.

## 5.1. Popular Game Engines:

				
Features	Unity	Unreal Engine	CryEngine	Godot
Languages	C#	C++	Lua, C++, C#	GScript, C++, C#
VPL	Unity ECS	Blueprints	Flow Graph	Visual Script
Community	123k+	42,6k	10,2k+	24,7k+
Source Modify	Yes - Paid	Yes - Royalties	Yes - Royalties	Yes - Free
Platforms	PC, GC, WEB, MD, XR	PC, GC, WEB, MD, XR	PC, GC, WEB, MD, XR	PC, GC, WEB, MD, XR
Assets Store	AssetStore	MarketPlace	MarketPlace	Asset Library
License	EULA	EULA	EULA	MIT
Developer's	Unity Technologies	Epic Games	Crytek	SFC & PLC
Year	08.06.2005	28.04.1997	02.05.2002	14.01.2014

PC - Personal Computer's / GC - Game Console's / WEB - Browser's / MD - Mobile Device's / XR - Extended Reality

[29]

### 5.1.1. Unity

Description: This is one of the most accessible and intuitive game engines; it is well-suited for both 2D and 3D game development. It features cross-platform support that can be used to create games on PC, console, mobile, and VR/AR platforms. It is programmable in C#. Unity has a huge asset store and an active community. Key Features:

- Cross-platform support
- Real-time rendering
- Asset store
- Extensive documentation and tutorials [12]

### 5.1.2. Unreal Engine:

Description: Unreal Engine by Epic Games is known for its great-looking graphics and complex physics. It supports programming languages like C++, but it also has a visual scripting system called Blueprint. It's utilized in AAA games and projects that demand cinematic quality.



Key Features:

- High-quality visuals and rendering (e.g., Unreal Engine 5 with Nanite and Lumen)
- Blueprint visual scripting
- Advanced physics and AI
- Open-source and free for use with royalty system on commercial games [\[13\]](#)

### **5.1.3. Godot:**

Description: Godot is a free, open-source game engine that is most powerful for 2D game development but also supports 3D games. Godot uses its own Python-like scripting language called GDScript and supports C#. It is lightweight and fast, hence more popular among indie developers.

Key Feature:

- Open-source and free
- Lightweight and easy to use
- Supports 2D and 3D game development
- GDScript, C#, and visual scripting options [\[14\]](#)

### **5.1.4. CryEngine:**

Description: CryEngine is a game engine known for its remarkable visuals and visual effects; it is primarily used for open-world and large-scale 3D games. It is written in C++, but languages such as Lua and Python can also be used on top of it. Key Features:

- Realistic graphics and rendering
- Open-world capabilities
- Built-in AI and physics systems
- Free to use with royalty payments for commercial use [\[15\]](#)

### **5.1.5. Cocos2d:**

Description: Cocos2d is an open-source game engine widely used for developing mobile games. It is focused on 2D games, but 3D games can also be developed. It is used widely by indie developers, especially those who develop games for Android and iOS.

Key Features:

- Free and open source
- Optimized for mobile platforms
- Supports 2D and limited 3D development
- Cross-platform support [\[16\]](#)

#### **5.1.6. GameMaker Studio 2:**

Description: GameMaker Studio 2 is an impressive game engine but user-friendly for 2D game development. It combines both a visual programming approach and a script-based approach. Ideal for smaller projects and indie game developers.

Key Features:

- Drag-and-drop visual programming
- Built-in scripting language (GameMaker Language)
- Cross-platform export
- Easy learning curve [\[17\]](#)

### **5.2. Why We Chose Unity for Our AI-Based PC Game**

While developing a PC-based AI-driven game, Unity is considered an ideal engine that has been chosen for the above-mentioned project based on a few key features and capabilities. Here's why we chose Unity:

#### **5.2.1. Advanced AI Integration**

Unity comes with powerful tools and libraries to implement AI in our game. The engine, from pathfinding using NavMesh to decision trees and state machines, supports a variety of AI techniques that will make it quite easy to implement intelligent behaviors for NPCs and enemies. Moreover, Unity's support for the integration of machine learning makes it possible to experiment with different AI models with a view to enhancing gameplay and NPC interactions.

#### **5.2.2. Real-Time Rendering and Graphics**

Most AI games have great visuals to make their worlds feel immersive and real. Real-time rendering within Unity enables us to build visually pleasant environments so that the AI behaviors become part of the visual experience. By leveraging Unity's powerful rendering tools, you can build realistic, dynamic worlds that respond to the events and behaviors of AI in your game.

### **5.2.3. Cross-Platform Support**

Unity has excellent support for cross-platform development, which will be highly helpful in the development of our AI-based PC games and maybe other platforms. We can deploy our game on various operating systems like Windows, macOS, and Linux, and even consider future support for VR/AR platforms from one codebase. This flexibility allows targeting more platforms with a lot less extra work.

### **5.2.4. Ease of Prototyping and Rapid Development**

Unity's drag-and-drop functionality, together with the rich asset store, lets us prototype AI systems fast and test them in a variety of scenarios. This rapid building and iteration capability is crucial to perfecting the complex behaviors of AIs, testing their impact on gameplay, and refining every element of a game. Its visual scripting functions, such as Bolt, also come in very handy with those on the team not so into coding.

### **5.2.5. Robust Community and Asset Store**

Unity's huge and active developer community, along with its Asset Store, opens up access to a huge repository of resources, plugins, and solutions that speed up our work on game development. Specifically, it is easy to build on top of the already created AI-related assets, scripts, and common solutions for some problems, giving us more resources to concentrate on game-specific AI implementations.

### **5.2.6. Machine Learning and AI Support**

Unity's ML-Agents Toolkit makes integrating machine learning models into the game easier. For such an environment, we can start working on some advanced AI features involving reinforcement learning that may be useful for the creation of more clever NPCs or adaptive gameplay systems. This is indispensable for an AI-oriented game in case the gameplay should be done dynamically by some intelligent system.

### **5.2.7. Scalability for Complex AI Systems**

AI in games can quickly become computationally intensive, especially with complex decision-making, pathfinding, and simulations. Unity's performance optimizations and flexible architecture allow us to scale AI systems efficiently, ensuring the game remains performant even with large numbers of NPCs or advanced AI-driven events.

### **5.2.8. Extensive Documentation and Tutorials**

Unity has extensive documentation and a plethora of tutorials that cover everything from basic AI programming to advanced machine learning integration. This will be of great benefit to our team in finding solutions to problems and learning how to implement certain AI systems without having to reinvent the wheel. Unity is the best for our AI-based PC game due to its facilities, flexibility, and support necessary for effectively embedding sophisticated AI systems. Be it machine learning, advanced pathfinding, or even the creation of believable NPC behaviors, Unity has it all to materialize this AI-infused vision without losing one iota of visual quality or smoothness in gameplay.

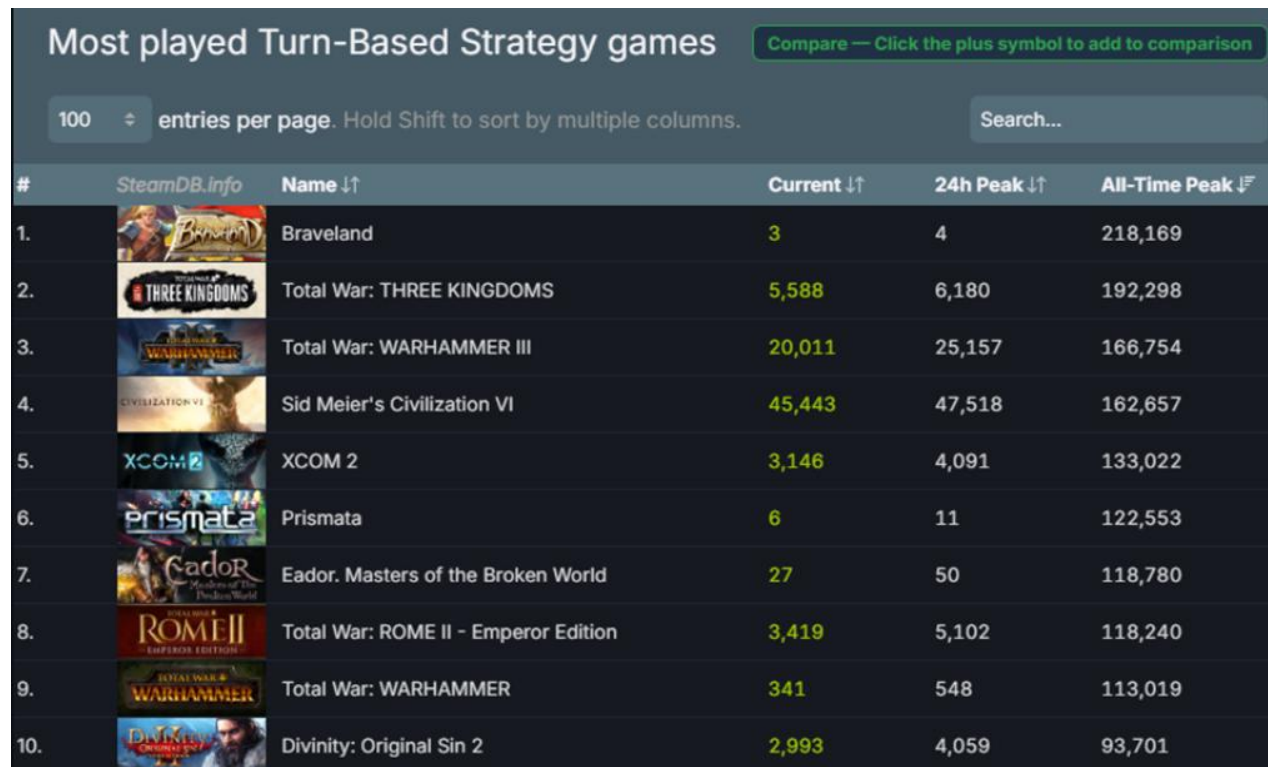
## **6. Statistical Analysis of Preferred Game Genres and Tools**











The paper now goes on to the detailed analysis of the game project developed in turn-based strategy, tactical RPG, and roguelike genres. We will analyze the tools applied for the development of the game, player demographic statistics, and favorite features. When referring to a discussion of the popularity of such genres, gameplay mechanics, and player demographics, it comes to how game design shapes player interaction. Of the genres which will be applied to our project, the turn-based strategy genre allows players to make moves one after another, hence having strategic depth, while in the tactical RPG genre, it allows story-based progress and the development of characters. Furthermore, the addition of roguelike elements in a new game ensures that it is different from any other game played, making it unique each time. This blend of genres gives the player an enriching experience where players can easily switch between playstyles and work out new strategies.

## 6.1. Preferred Genre Statistics

### 6.1.1. Turn-based strategy (TBS)

TBS games are those that usually deal with large-scale strategy and planning over a map or scenario, commanding units or factions in order to compete for territory, resources, or objectives against other players or AI. [18]



Most played Turn-Based Strategy games					
Compare — Click the plus symbol to add to comparison					
100 entries per page. Hold Shift to sort by multiple columns. Search...					
#	SteamDB Info	Name ↑↓	Current ↓↑	24h Peak ↓↑	All-Time Peak ↓↑
1.		Braveland	3	4	218,169
2.		Total War: THREE KINGDOMS	5,588	6,180	192,298
3.		Total War: WARHAMMER III	20,011	25,157	166,754
4.		Sid Meier's Civilization VI	45,443	47,518	162,657
5.		XCOM 2	3,146	4,091	133,022
6.		Prismata	6	11	122,553
7.		Eador. Masters of the Broken World	27	50	118,780
8.		Total War: ROME II - Emperor Edition	3,419	5,102	118,240
9.		Total War: WARHAMMER	341	548	113,019
10.		Divinity: Original Sin 2	2,993	4,059	93,701

**Figure 8:** Most Played Turn-based Strategy Games (14.11.2024) [19]

While some turn-based strategy games have been able to break through the 200,000-player peak milestone, as shown in figure 8, Braveland has attained this great achievement and has been a huge success in the genre. This means that the game was well-received by gamers, was deep enough, had engaging mechanics, and had the potential to keep growing in the turn-based strategy space. Such milestones are a testament to the game's widespread appeal and its place among successful titles in the industry.









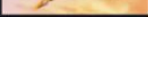



**Figure 9:** Steam Turn-based Strategy Game Releases by Month [\[20\]](#)

It can be taken from figure 9 that turn-based strategy games are released at an approximate rate of 100 a month. It also captures the uprising growth in the genre and the increasingly growing interest among players to play strategy-oriented games with deep planning and decision-making. The development focuses on providing more depth and variety in this genre by getting players to think tactically and strategically.

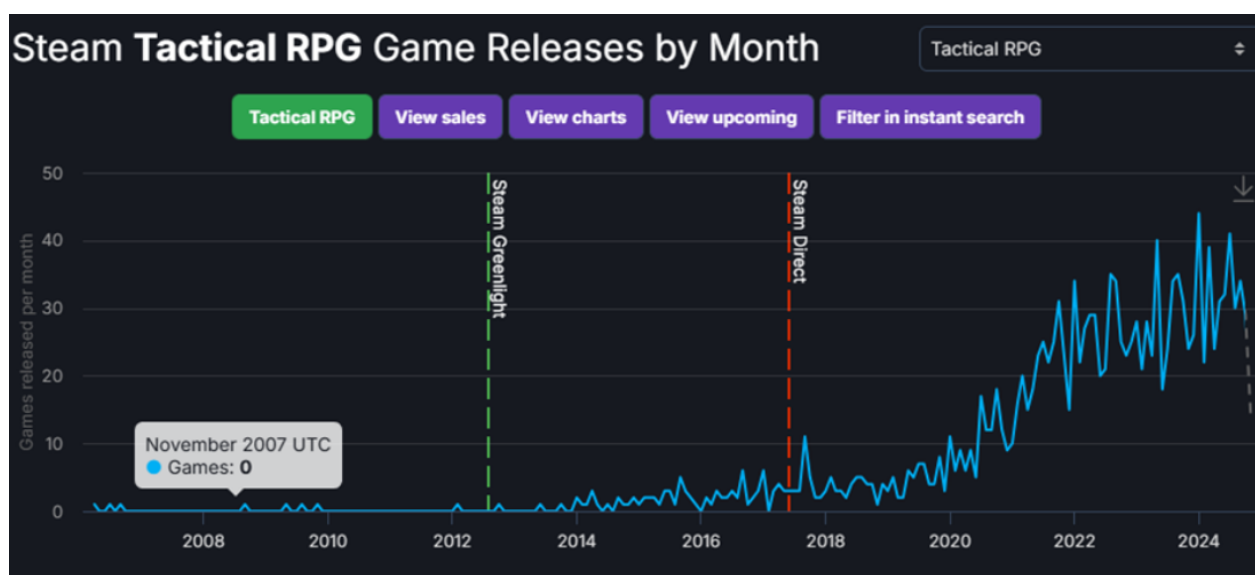
### 6.1.2 Tactical RPG

Tactical RPGs incorporate narrative and characters with strategic acting. Unlike common RPGs, most TRPGs focus on extensive strategy, team management, and tactical-based battles on grid-scaled battlefields. Players have to think over each step in a game, because positioning and terrain usage by characters can make lots of difference during battles. [\[21\]](#)

Most played Tactical RPG games						Compare
100 entries per page. Hold Shift to sort by multiple columns.		Search...				
#	SteamDB.info	Name ↓↑	Current ↓↑	24h Peak ↓↑	All-Time Peak ↓↑	
1.		Braveland	3	4	218,169	
2.		鬼谷八荒 Tale of Immortal	6,942	7,088	184,171	
3.		Divinity: Original Sin 2	3,031	4,059	93,701	
4.		RimWorld	19,972	21,399	62,257	
5.		Warhammer 40,000: Rogue Trader	3,035	5,083	45,405	
6.		Age of Wonders 4	5,773	5,940	42,877	
7.		Hero's Land	9,645	9,645	37,249	
8.		XCOM: Chimera Squad	71	104	35,955	
9.		Wartales	1,818	2,353	35,879	
10.		Jagged Alliance 3	452	722	25,366	

**Figure 10:** Most Played Tactical RPG games (14.11.2024)[\[22\]](#)

Braveland not only fits into TBS but also into the tactical RPG genre. The combination of tactical combat, character development, and story-driven gameplay combined to help it achieve over 200,000 peak players. This speaks to how wide of a net is cast for this game, within multiple genres, but it also signals a growing trend for tactical RPGs and their capability for holding onto a very large, engaged player base.



**Figure 11:** Steam Tactical RPG Game Releases by Month [\[23\]](#)

Figure 11 shows that tactical RPGs are released at an approximate rate of 30 titles a month. This proves that while the genre is picking up, it still is more selective than most. Developers are focused on releasing better content with deep strategies, character progressions, and storytelling. The relatively lower number of releases reflects the fact that such deep storytelling games take their time to make and polish.

### 6.1.3 Roguelike

The roguelike is a class of role-playing video game that traditionally features a dungeon crawl through procedurally generated levels, turn-based gameplay, grid-based movement, and permanent death of the player character. [24]



#	SteamDB.info	Name ↓↑	Current ↓↑	24h Peak ↓↑	All-Time Peak ↓↑
1.		Don't Starve Together	55,522	57,094	115,925
2.		Hades II	3,687	5,133	103,567
3.		Vampire Survivors	10,638	12,018	77,061
4.		Risk of Rain 2	6,489	8,297	75,406
5.		太吾绘卷 The Scroll Of Taiwu	1,077	1,238	72,533
6.		The Binding of Isaac: Rebirth	25,590	28,597	70,701
7.		Hades	5,466	6,723	54,240
8.		Loop Hero	337	434	51,156
9.		Risk of Rain Returns	186	314	45,141
10.		暖雪 Warm Snow	1,547	1,795	44,702

**Figure 12:** Most Played Roguelike games (14.11.2024)[25]

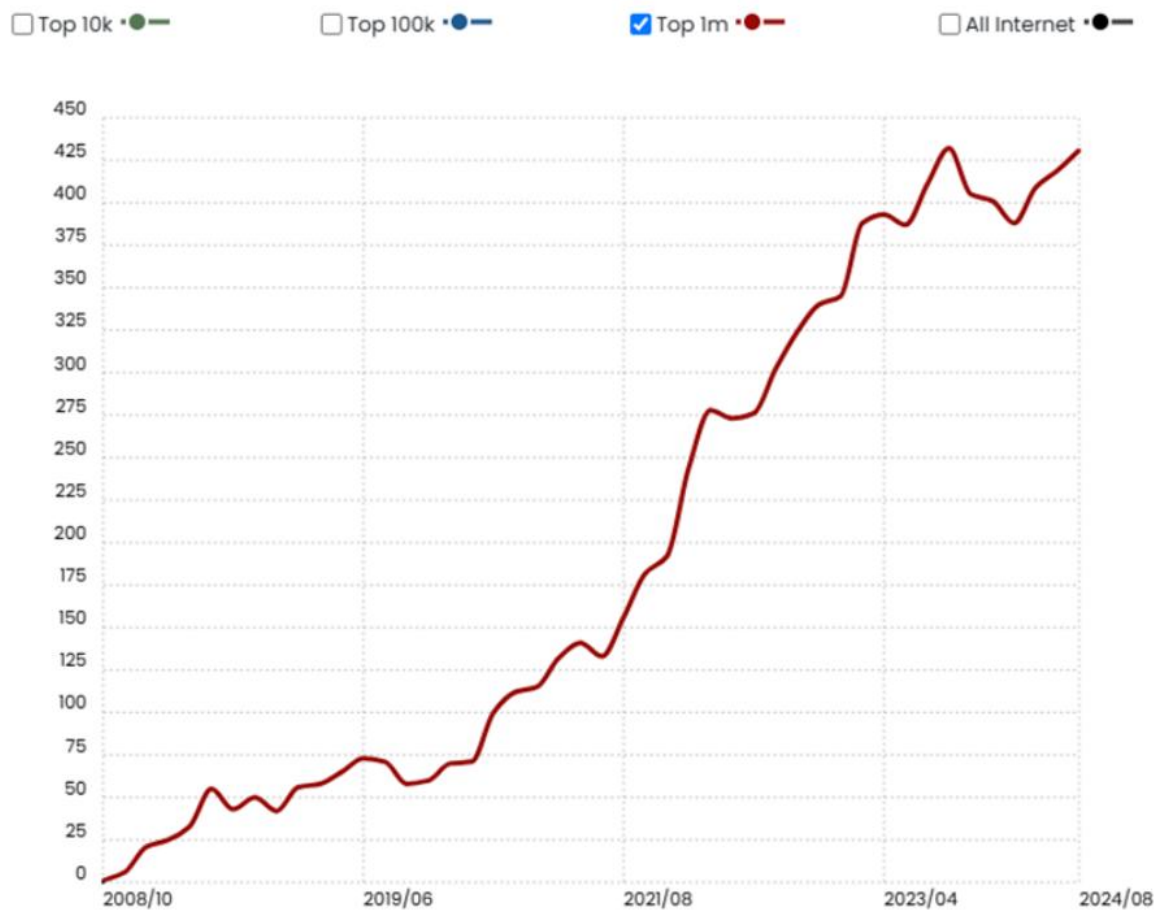
As illustrated in Figure 12, the rogue-like game Don't Starve Together has reached over 100,000 peak players. That goes to show its mass appeal by marrying survival elements with the unpredictability of roguelike gameplay. Its frequent updates, cooperative features, and tough mechanics have contributed to its success and ensured its status as a big roguelike title. This milestone shows the capability of this game to consistently attract a massive player base.



## 6.2. Preferred Tools Statistics

### 6.2.1. Unity Statistics

#### Unity Usage Statistics

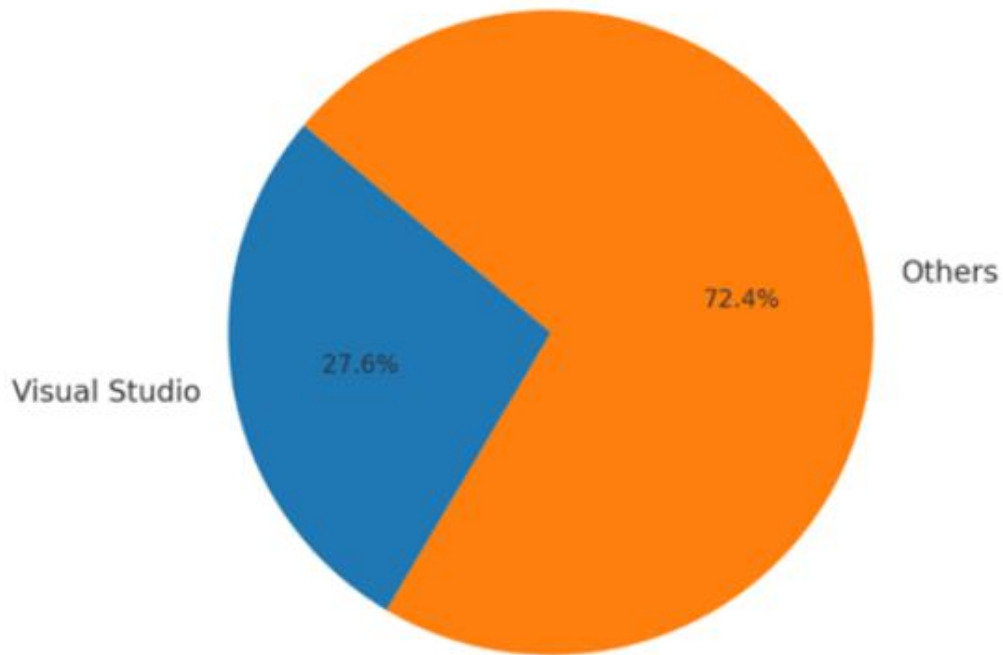


**Figure 13:** Unity Usage Statistics [\[27\]](#)

Figure 13 shows the high growth of Unity since it was released in the year 2008. It suddenly began to rise and, particularly in game development, became a vital tool in which many developers from around the world plunged by 2023. In 2024, Unity started getting used not only in game development but also for educational purposes, simulations, and architectural visualization. Such sudden growth was the result of its strong features combined with accessible platforms that can work on a number of systems easily.

### 6.2.2 Visual Studio Statistics

## IDE Popularity Share: Visual Studio vs Others



**Figure 14:** Visual Studio Statistics [\[28\]](#)

From figure 14, the market share of Visual Studio stands at 27.6%, indicating that it is among the most widely used IDEs among developers. The "Others" are 72.4%, meaning that a lot of developers still use other IDEs and tools. That means even though Visual Studio is one of the leading choices, the landscape of development tools is so varied, with different tools fitting different needs and preferences.

## 7. Inspired and Similar Projects

Looking through examples of other, similar works gives valuable insight during refinement and in innovating new concepts within a game. The most immediate inspirations are "Rule and Munus", which relates with turn-based strategies and RPGs: "Loop Hero", "Civilization VI", "Divinity: Original Sin 2", and "Swords and Sandals" did their parts for inspiration with mechanics and in-game events. Each of these games has set standards in town-building, character development, and AI-driven gameplay and thus provided both inspiration and points of contrast for the unique aspects of "Rule and Munus."

## **7.1. How We Are Different**

"Rule and Munus" is unique because it combines the mechanics of a roguelike game with a strong town-building feature. It therefore offers a gameplay loop wherein players will be concerned with combat but also with the strategic growth of their town. Unlike turn-based games, this project puts the emphasis on decision-making with consequences throughout on town resources, character growth, and into subsequent rounds. Again, a dice-roll-based system for combat actions brings that layer of randomness more familiar to the tabletop RPG but is developed in its execution for a digital, replayable game.

## **7.2. What We Improved and Can Improve**

We improved existing models by refining the balance between chance-namely, dice rolls-and strategy. Whereas in "Swords and Sandals," the fights start to get a little repetitive, "Rule and Munus" offers randomized events in town-building that force one to strategize for the long run rather than just preparing for a single encounter. Further development in NPC intelligence and decision-making for adapting challenges is possible. Increasing the number of unique character skill trees and options in resource management may further enhance strategic depth in town-building and combat.

## **7.3. Inspired Features**

Key features inspired by other games include: Dice-Roll Combat: from "Dungeons and Dragons," adding excitement and unpredictability. Town-Building and Resource Management: influenced by "Loop Hero," allowing players to shape their home base, which directly impacts character progression. Strategic Depth and Replayability: inspired by "Civilization VI," encouraging a cycle of growth, conflict, and expansion. Integrating these elements into "Rule and Munus" allows us to blend familiar mechanics with novel interactions, positioning the game as a fresh take within the turn-based, strategy, and RPG genres.

# **8. Our Project Goals**

The goal of this game is to develop an AI-based strategy RPG. In our game, the AI will be able to adapt and evolve based on the player's in-game behavior. It would analyze the player's strategies and create a far more dynamic and challenging in-game experience. The player will play the role of a gladiator who develops himself and his village. The goal of the gladiator is to defeat the king, who, through AI, develops and strengthens himself constantly. This process will give more strategic depth to the game and simultaneously provide players with an exciting and challenging experience. The game tries to keep the player in it with its rich storyline, high levels of competitiveness, great visuals, and atmospheric music.

### **8.1. Market Goals (On Steam Platform):**

- Achieve over 85% positive reviews: We aim to offer a high-quality gaming experience that will be well-received by players and gather positive feedback.
- Reach 15,000+ players: We plan to introduce the game to a large player base, building a solid player community on the Steam platform.
- Maintain 100+ concurrent players: The game should have a constant presence of active players, ensuring a dynamic community.
- Offer support for 10 different languages: Our goal is to provide multilingual support to cater to a broader global audience.
- Become a competitive game within its genre: We aim to create a strong competitive environment within AI-based games and achieve a position where we can compete with other games in the same category.
- Build a player community: We aim to create a community where players can share their in-game achievements and experiences. By bringing players together on social media platforms, we will encourage them to share their experiences and successes. This will be crucial for enhancing community interaction and fostering player engagement.

## **9. Conclusion**

The evolution of AI and machine learning in game development offers developers various possibilities to create intelligent, dynamic, and immersive gaming experiences. We identified key trends in game design and market preferences that align with our project's goals, including the appeal of strategy-driven games like turn-based strategies and roguelikes. As we develop the project, our primary objectives are to differentiate our product in a competitive market and deliver a scalable, user-focused experience

# Software Requirements Specifications

## 1. Introduction

### 1.1 Purpose of This Document

The purpose of this document is to show the functional requirements and non-functional requirements of our game called Rule and Munus. In addition, this document is a guide prepared to show the goals, mechanics and technical features of our game. The main purpose of the document is to provide a roadmap during the development phase to ensure that all functions are compatible with design and user expectations.

### 1.2 Problem Definition

In modern games, difficult situations may arise when seamlessly combining elements such as combat, resource management, strategy and artificial intelligence integration. This limits the replayability of the game for players. It can also lead to predictable experiences. Rule and Munus aims to eliminate this problem by offering dynamic turn-based combat and an artificial intelligence that constantly improves itself with the player. The game features unique dice-based mechanics and the integration of features inspired by well-known games, creating an impressive and unpredictable game. The main problem we are trying to solve with this game is the lack of an artificial intelligence that reflects player actions and changes in complex situations and adapts to the conditions it finds itself in. It aims to provide interesting experiences to the player by providing competitive and balanced combat.

### 1.3 Scope of This Project

Rule and Munus is a single-player, turn-based strategy game that combines the tactical RPG and roguelike genres. The game, set in an isometric world created with rich details, consists of two main stages:

1. **Peace Phase:** Peace Phase: Players manage a town; build, upgrades buildings, crafts/sells/buys items, and improves your characters with stat distribution.
2. **War Phase:** Players roll dice when they want to gather supplies, fight, or participate in gladiatorial tournaments. Combat is strategic, turn-based, and dice rolls can influence battle strategies

The highlight of the game is the dynamic AI opponent, the King of Gladiators, who plays the game with the player, improving himself over time and serving as the main opponent. Project limits include:

- **Platform:** PC
- **Language:** C#
- **Graphics:** 2D isometric view
- **Target audience:** Those who love tactical RPG and turn-based strategy games

By combining components such as resource management, dice mechanics and adaptive artificial intelligence, this project aims to challenge players' strategic skills. The other goal is to provide a unique and replayable experience.

## 1.4 Glossary

Term	Description
AI (Artificial Intelligence)	AI (Artificial Intelligence) refers to a computer system created to make decisions akin to a human being. In this game, an AI governs the King of Gladiators, emulating human thought processes and adapting to the player's moves.
Action Points (AP)	Action Points (AP) serve as a resource for players or AI to execute various actions in combat, including attacking, defending, or utilizing abilities.
Colosseum	Colosseum A mode in which players take part in gladiator tournaments, confronting adversaries in orchestrated fights to progress the narrative and gain rewards.
Combat Interface	Combat Interface A strategic grid framework that allows players to engage in turn-based combat against their opponents.
Dice Mechanics	Dice Mechanics refer to a system in which the results of dice rolls influence the success of various actions, the caliber of crafted items, and the results of events. For instance, a d20 may be used for gathering materials, while a d100 is utilized to assess the quality of item crafting.
Encounter	Encounter A chance occurrence during the War Phase, in which the player either confronts an adversary or collects resources determined by the results of a dice roll.
Heuristic-based Decision-making	Heuristic-based Decision-making A strategy employed by the game's AI to reach decisions through straightforward rules instead of intricate computations, ensuring a balance between efficiency and flexibility.
Isometric View	Heuristic-based Decision-making A strategy employed by the game's AI to reach decisions through straightforward rules instead of intricate computations, ensuring a balance between efficiency and flexibility.
Peace Phase	The Peace Phase is the game section focused on managing the city, creating characters, and improving these created characters.
Roguelike	A sub-genre of games characterized by procedurally generated challenges, randomness, and a high degree of replayability.
Stat Points	Roguelike refers to a sub-category of games known for their procedurally generated challenges, an element of randomness, and significant replay value.
Turn-based Combat	Turn-Based Combat is when players and opponents take turns performing actions. Resources such as action points are used. The Battle Phase is the part of the game where players collect supplies, participate in fights, or take part in the Coliseum.
War Phase	The gameplay phase where players gather materials, engage in combat, or participate in the Colosseum.
Unity Engine	Unity Engine is a game development platform used in game production and handles graphics rendering, physics and AI behavior.
User Interface (UI)	User Interface (UI) is the visual and interactive elements that allow players to perform actions, organize inventory, and navigate game menus.
XP (Experience Points)	XP are points that players earn through activities such as combat and crafting, which help them level up their characters

## 1.5 Overview

This archive is organized to supply a clear understanding of the amusement, client profiles, framework necessities, utilitarian highlights, client interface plans, and UML models. Begins with a clarification and proceeds with the outline, framework necessities, UI plans and utilize case portrayals.

## 2. Overall Description

### 2.1 Product Perspective

This archive is organized to supply a clear understanding of the amusement, client profiles, framework necessities, utilitarian highlights, client interface plans, and UML models. Begins with a clarification and proceeds with the outline, framework necessities, UI plans and utilize case portrayals.

- **Standalone Operation:** The game itself is autonomous and does not depend on other systems or services. A player interacts with the game via an intuitive interface on a PC; it also does not depend on third-party platforms for gameplay.
- **Relation to Existing Systems:**
  - Inspired by Swords and Sandals, this game adds new strategic skin to city building and AI-powered opponent interactions. It also preserves the basic combat mechanics.
  - Borrowing from Dungeons and Dragons added depth to gameplay by inspiring the use of dice-based mechanics to provide randomness and decision-making.
  - The turn-based structure of Sid Meier's Civilization VI has been adapted to merge strategic peace and war phases.
  - It borrows town development and crafting mechanics from Loop Hero, adapted to offer bespoke player development.
- **Innovative Features:** It features a unique, realistically human-like AI opponent known as the King of the Gladiators, dynamic in nature and changing upon the player's every action. This AI is reflective of the player's strategy, of growing strength, and is designed for persistence throughout the game to offer unique, adaptive gameplay.
- **Gameplay Integration:** The game goes a long way to making resource management, character progression, and turn-based combat a well-balanced and engaging experience. Players will switch between Peace and War Phases, making strategic decisions that directly influence their progress and success. From this point of view, Rule and Munus are an innovative project in the genres of turn-based strategy and tactical RPG; they introduce new aspects to existing systems and patch gaps in adaptive AI and replayability.

#### 2.1.1 Hardware Interfaces

Users can play the game with a general PC device that runs a compatible operating system. Necessary hardware requirements are.

1. **Game Console:** User can interact with the system by corresponding buttons of Game Console to take actions.
2. **Keyboard:** User can interact with the system by corresponding keys such as selecting an action from the action bar in combat by numbers.
3. **Mouse:** Mouse can do user interactions like clicking on an action-on-action bar or clicking on any building to display respective set of actions

## 2.1.2 Software Interfaces

### Asset Management:

- It states about the handling mechanism that would be applied on to all game textures, sounds and models. The process is provided using the calls made on OS which facilitates game interaction with FS for dynamically loading or saving different sets of assets throughout the course of runtime, **Unity Requirement:**
- Unity is highly needed for the construction and implementation of the functional needs of the game. This will provide a complete set of tools and structure for building the features of the gameplay, UI elements, and the overall mechanics of the game.
- **Windows and macOS Compatibility:**
  - The game of Rule and Munus is compatible with Windows and macOS computers. This makes the game accessible for a wide number of players using these operating systems.
- **Physics Computations:**
  - Calculations such as gravity, collision and application of forces will be carried out through the Unity3D game engine. This will enable more realistic interactions to be implemented in the game world.
- **Object Collisions and Interactions:**
  - This would be handled by Unity3D. For example, when collisions occur between characters or objects, the physics in Unity will compute the result of the collision, and the interactions would be smooth.
- **Animations:**
  - Unity3D has to deal with character and object animations, thus enabling transitions, movements, and multiple other visual effects. Character walking, jumping, or any object-specific animations like rotation or glow.

## 2.1.3 User Interfaces

Key interface components include:

- **Main Menu:** This shall contain the buttons of play, settings, and credits.
- **Town Interface:** An isometric elaborated view of the town. The players will have the option of clicking on the buildings to build, upgrade, and engage the right NPC accordingly.
- **Character Menu:** character stats, inventory, equipped items
- **Combat Interface:** turn-based tactical grid system with action points, dice roll results, combat options: attack, defend, use skill, flee.
- **War Phase Interface:** allow user to choose either gather or go to coliseum. This interface is designed to be used both with mouse-and-keyboard or game console inputs. Users can easily navigate around and control with ease.



## 2.2 Product Functions

Key Features and Functions of Rule and Munus:

1. **Peace Phase:**
  - Town-building and upgrading.
  - Craft items with gathered materials.
  - Sell and buy items in the market.
  - Assign stat points for character development.
2. **War Phase:**
  - Gather materials by rolling dice and events in turns.
  - Combat will give options to attack, run, use skill, and defend
  - Take part in the gladiator tournaments for rewards and story progress.
3. **Human-like AI:**
  - Dynamic evolution shared with the player's actions.
  - The main opponent-the King of Gladiators-controlled by AI, strategically modifies himself and grows stronger with time.
4. **Dynamic Gameplay Elements:**
  - Randomized events and loot via dice rolls.
  - Strategic decision-making influences story progression and AI behavior.

## 2.3 User Characteristics

- **Primary Audience:**
  - Gamers interested in turn-based strategy, tactical RPGs, and rouge-like games.
- **Goals:**
  - To experience strategic decision-making in improving character and combat actions.
  - To challenge a dynamic AI opponent.
  - To progress through a basic story while improving their character.
- **Skills:**
  - Basic familiarity with PC gaming controls (keyboard and mouse or game console).

## 2.4 Assumptions and Dependencies

- **Assumptions:**
  - Players will be equipped with PCs that meet at least the minimum hardware requirements.
  - No external server or online connection is required for core gameplay.
- **Dependencies:**
  - Development relies on the Unity Engine and C# programming language for implementation.
  - Graphics and UI assets rely on third-party design tools such as Adobe Photoshop or Blender.
  - AI components rely on machine learning frameworks integrated into Unity.

## 3. Specific Requirements

### 3.1 External Interface Requirements

#### Input:

- Player input via mouse and keyboard to select actions, manage inventory, and interact with in-game menus.
- Dice rolling mechanism triggered by player input; the result of the roll is displayed on screen. Output: **Output:**
- Isometric representation of town, character and battle.
- Step-by-step representation of dice rolls, actions taken, and AI decisions.
- Notification system for resource gain/loss, loot and battle results. **Behavior:**
- Smooth transitions between Peace and War Phases based on player actions.
- Artificial intelligence will dynamically adapt to players' strategies in battle and city development.
- Player decisions are kept clear and consistently fed back.

### 3.2 Functional Requirements

#### 3.2.1 Peace and War Phase

The flow of the game is divided into the Peace and War Phases, each with a different way of gameplay. During the Peace Phase, the player will manage the town, arrange his inventory, review the materials, and build new constructions. For instance, the Blacksmith allows for the creation of items, and generally, at each leveling up, stat points can be spent on improving one's character attributes. This War Phase is action-packed; players can gather material, fight, or take part in the gladiator tournament. In these phases, the transition from one phase to the next maintains a balance between a strategic preparatory phase and an active gameplay phase. It lets players fine-tune their strategy in the Peace Phase, followed by taking to the actual War Phase.

#### 3.2.2 Town Management

The player builds and upgrades various buildings throughout the game. Simple buildings like the Market are first constructed, enabling the player to trade and perform simple upkeep. While upgrading buildings, new features are gradually made available. The Blacksmith and Wizard buildings can be upgraded for a higher-tier item craft to have more rare or powerful gear, for instance. Similarly, with the upgrading of buildings, gathering materials becomes much better and diverse in their type, so actions can also be performed accordingly to make better, strong progress in the game. This, therefore, means that with each building upgraded, qualities of items and resources become available to the players for even better benefits in their quest.

### **3.2.3 NPC Interactions**

Through the process of building and upgrading buildings in their town, the player unlocks the ability to interact with NPCs to trade and craft items. The market already exists and enables the trade of basic items and materials from the very beginning. As the town develops and buildings are upgraded, new NPCs appear that sell more valuable and powerful items. In this way, the player can collect resources from these interactions. Similarly, crafting NPCs like the Blacksmith allows players to create new items using the materials they collect. The manufacturing of items starts with rather simple ones, but, as with the rest of the game content, upgrading town buildings unlocks the possibility to craft more advanced and powerful items. Quality depends on a d100 roll, which increases the possibility of getting rare and powerful items. NPCs guide players on crafting recipes and required materials. Building and upgrading the structures gives power to players to take a stronger management role in the town, where interacting with the NPCs enables trade, crafting, and making strategic decisions—a process where players get an opportunity to strengthen themselves and generate resources.

### **3.2.4 Character Attribute Points**

Character attributes have a direct relation to success and performance in combat, among other aspects of the game. Strength amplifies the damage dealt with equipped items and is crucial for all actions requiring physical power. Intelligence amplifies the character's maximum mana and enhances spell effectiveness, therefore making the abilities of magic much stronger. Constitution increases the character's health points, which help in survival. Dexterity reduces the amount of damage taken while on defense and also increases mobility, thus making a character more agile. Charisma is very important in trading, as it allows the player to buy items at a lower price and sell them for more value. These attributes can be distributed strategically to fit different playstyles and provide advantages in various challenges throughout the game.

### **3.2.5 AI Behaviours**

AI behaviors are shaped by the game structure and the player's interactions. The AI adapts to the player's chosen role and the progression of the game. It analyzes the player's movements and preferences to devise strategies. In combat, the AI analyzes the player's defense tactics and creates counter-attack plans. Also, the action and decisions taken by AI vary with different stages of the game. That too requires it to adapt to peaceful and combat situations. The AI reacts to the player's role and further polishes those reactions continuously with the dynamics of the game. This makes the game dynamic in experience because the player keeps facing new challenges throughout the game.

### **3.2.6 Gathering and Loot**

The Gathering phase generates the type and quantity of the enemies the player will face through a dice system. This impacts the level of challenge and the variations of enemies encountered. After the combat is over, the player goes through the Loot system to gain loot. The type and quantity of loot depends on the outcome of the fight and is awarded to the player by adding it to their inventory.

### **3.2.7 Roll a Dice**

Gathering and looting are done with the help of a dice mechanism. The dice rolls determine the amount and type of enemies, or the percentage chance of successful attack or defense if combat takes place. That makes the game more dynamic, not turning into a routine or some sort of drill for players.

### 3.2.8 Combat Mechanics

If the player uses turn-based combat, the game loads a battle scene. In the battle scene, the player has a few options: to attack with an equipped weapon, to use scrolls or mana for powerful skills, to move left or right for repositioning, to wait and see what the enemy does next, or to defend and reduce the damage taken. These various combat options offer a bunch of strategic possibilities to players, enabling them to adapt to every situation and the behavior of their enemy.

### 3.2.9 Go to Coliseum

Three of the options that the player can choose from in the Colosseum are as follows: Encounter, Gladiator or King. Based on his selection, he enters the arena and can fight at a certain level of difficulty. Each of these options presents a different type of challenge and hence compels the player to decide strategically.

## 3.4 Performance Requirements

The following performance indicators will be focused on to ensure that the player's experience is smooth and entertaining:

#### 1. Response Time:

- User Input to Action Execution: Less than 100 milliseconds for town management actions, inventory management, and combat commands.
- Dice Roll Feedback: Results to be displayed within 1 second from when the roll was triggered.
- Phase Transitions: Peace-to-War or War-to-Peace transitions should be completed in less than 2 seconds.

#### 2. Throughput:

- Up to 100 active AI decision processes per second simulate the King of the Gladiators' human-like behavior.

#### 3. Scalability:

- The game should run consistently at a set framerate on both minimum and recommended hardware specifications.
- Towns with up to 10 buildings, inventories holding 100 items, and combat involving up to 2 opponents simultaneously should function without framerate drops below 60 FPS on recommended specifications and 30 FPS on minimum specifications. The minimum and recommended hardware specifications are as follows:
- **Minimum Requirements:**
  - Processor: Dual-core CPU (2.0 GHz or faster)
  - RAM: 4 GB
  - Graphics Card: Integrated GPU supporting DirectX 10 or OpenGL 3.3
  - Storage: 2 GB free space
  - Operating System: Windows 10 or newer

These specifications are selected to balance performance and accessibility, ensuring smooth gameplay and high-quality visuals for most players.

## 3.5 Design Constraints

### 1. Regulatory Compliance:

- The game will have to follow all copyright laws and licensing agreements, especially in inspirations taken from other games and systems.
- The game has to respect the data protection policy, which means any save-game files or user profiles should be saved securely and locally.

### 2. Environmental Constraints:

- The game shall be strictly designed for PC-based platforms only and shall not work on mobile devices or consoles.
- The game is supposed to work offline and not depend on any form of internet connectivity.

### 3. Technical Limitations:

- The AI system relies on heuristic-based decision-making rather than advanced neural networks to reduce computational complexity and meet performance requirements on minimum hardware.
- Graphics are limited to 2D isometric rendering to ensure compatibility with lower-end systems while maintaining visual clarity and style.
- The dice mechanics and their associated randomization functions must be lightweight, deterministic, and optimized for fast execution without impacting other game systems.

### 4. Development Constraints:

- The game is developed in C# using the Unity Engine, restricting certain features to those supported by the engine.
- A fixed development timeline requires prioritization of core gameplay mechanics over extended features such as multiplayer or extensive cutscenes.

## 3.6 Software System Attributes

### 3.6.1 Reliability

- The system shall handle all the expected inputs and it shall response w/o crash. In case of any failure, the mean time to repair shall be between 1 to 24 hours.
- Save and load shall be robust - no data corruption in normal use.
- Core gameplay mechanics (including but not limited to combat, crafting, AI behavior) shall function correctly in all the defined scenarios with appropriate error handling for invalid input or states.

### 3.6.2 Availability

- The game should be playable in offline mode and not depend on any network.
- It should be ready to play directly after installing the build, not depending on services or updates

### 3.6.3 Security

- Debug utilities/administrative functionality shall be unavailable to non-developers in a production build

### 3.6.4 Maintainability

- Code will be modular with clear boundaries separating game logic from AI routines and also from UI.
- Each major system will have accompanying documentation to help when updating or fixing bugs.
- Error logs will be produced upon crashes or strange behavior to ease debugging.

### 3.6.5 Portability

- The game is being developed on the Windows operating system. Due to using the Unity Engine and C#, this allows compatibility with macOS or Linux should additional platforms be desired in the future.
- System configurations include resolutions and graphical settings that can be changed for different hardware capabilities

### 3.6.6 Usability

- User interface shall guide players through complex game mechanics like crafting, combat, and town management.
- Tutorials will introduce core functionality and dice mechanics in step-by-step parts to ease into the learning curve.
- Visual and audio confirmation of player interactions shall be communicated clearly for a variety of situations that occur in a game, such as rolling dice, combat outcomes, and crafting completion.

### 3.6.7 Scalability

- The game will be able to support expansions-such as additional skills, AI behaviors, or new item sets-without heavy reworking of the core systems.
- AI computation scales proportionally with player actions to maintain consistent challenge without affecting performance.

---

## 3.7 Safety Requirements

Rule and Munus does not involve any form of real-world physical threat but does need to address safety-critical software aspects in order to protect the user's experience:

- **Data Safety:**
  - Backup mechanisms will be in place to ensure that save files are not corrupted, even in the case of unexpected power loss or system failure.

- Autosave functionality will work at key gameplay moments, preventing significant progress loss.
- **System Integrity:**
  - The game will not perform unauthorized operations on the player's device, ensuring no interference with other applications or system files.
  - Resource management in the game engine will prevent memory leaks or overuse of system resources, ensuring smooth operation without crashes.
- **Player Well-being:**
  - The game will include options to adjust visual settings to reduce strain, such as brightness control and a mode for colorblind accessibility.
  - The random nature of dice rolls, and AI evolution will be balanced to ensure fairness.

## 4. UML and Use Case Diagrams

### 4.1 Use Case Diagram

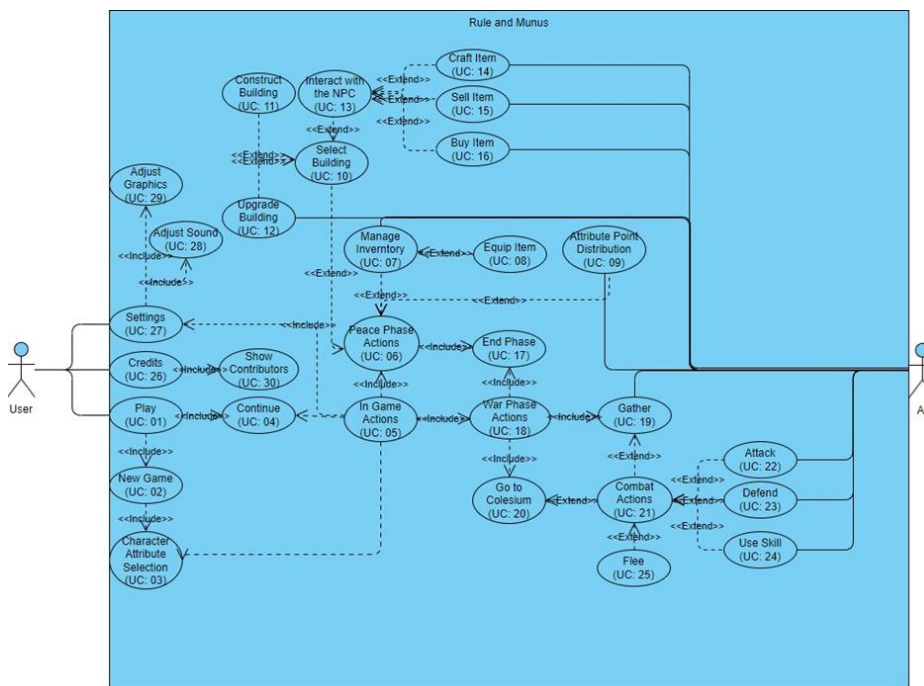


Figure 1: Use case diagram.



## 4.2 Use Case Descriptions

### Play Use Case Description

Elements	Details
ID and Name	UC-01, Play
Primary Actor	User
Secondary Actor	System
Description	The player selects the Play option to continue a previous game, start a new game, or return to the main menu.
Pre-Condition	* The player should be at the main menu screen.
Main Flow	<ul style="list-style-type: none"><li>* The player clicks the "Play" button from the main menu.</li><li>* The system will display the options: Continue, New Game, and Return.</li><li>* Player selects one of the following:<ul style="list-style-type: none"><li>* Continue: Resumes the last saved game.</li><li>* New Game: Begins the character creation process.</li><li>* Return: Returns to the main menu.</li></ul></li></ul>
Post-Condition	* The game continues from the saved state, begins a new game, or returns to the main menu.
Alternative Flow	~.
Special Requirements	Smooth transition between game states and visual feedback for selected options.
Related Use Cases	UC-02, UC-04

### New Game Use Case Description

Elements	Details
ID and Name	UC-02, New Game
Primary Actor	User
Secondary Actor	System
Description	The player selects the New Game option to create a character by selecting a class/race and distributing attribute points.
Pre-Condition	* Player selects New Game from the Play menu.
Main Flow	<ul style="list-style-type: none"><li>* Player selects New Game.</li><li>* The system displays class and race options that affect attributes and gameplay.</li><li>* Player selects a class and race, and the system shows an Attribute Point Distribution screen.</li><li>* Player allocates attribute points and confirms choices.</li><li>* System shows a character preview screen.</li><li>* Player either starts the game or returns to the main menu.</li></ul>
Post-Condition	* The character is created, and the game begins, or the player returns to the main menu.
Alternative Flow	The player may return to the main menu without starting a new game.
Special Requirements	Real-time effects for attribute point allocation and visual character preview.
Related Use Cases	UC-01, UC-03



#### Character Attribute Selection Use Case Description

Elements	Details
ID and Name	UC-03, Character Attribute Selection
Primary Actor	User
Secondary Actor	System
Description	The player allocates points to character attributes such as Strength, Dexterity, and Intelligence.
Pre-Condition	* The player is on the Attribute Point Distribution screen.
Main Flow	<ul style="list-style-type: none"> <li>* System shows available attributes and point pool.</li> <li>* Player allocates points to attributes.</li> <li>* System shows real-time updates based on allocations.</li> <li>* Player confirms or resets the allocation.</li> </ul>
Post-Condition	* The character attributes are updated based on the allocation.
Alternative Flow	The player resets the allocation and redistributes points.
Special Requirements	Real-time feedback and visual updates for attribute allocation.
Related Use Cases	UC-01, UC-05

#### Continue Use Case Description

Elements	Details
ID and Name	UC-04, Continue
Primary Actor	User
Secondary Actor	System
Description	The player continues gameplay from the last saved state.
Pre-Condition	* A saved game exists.
Main Flow	<ul style="list-style-type: none"> <li>* System checks for saved game data.</li> <li>* The saved game is loaded.</li> </ul>
Post-Condition	* Gameplay resumes from the saved state.
Alternative Flow	If no saved game exists, the system notifies the player.
Special Requirements	Quick and smooth game loading.
Related Use Cases	UC-02, UC-05

#### In-Game Actions Use Case Description

Elements	Details
ID and Name	UC-05, In-Game Actions
Primary Actor	User
Secondary Actor	System
Description	The player performs actions such as movement, combat, and interaction during gameplay.
Pre-Condition	* The player has an active game session.
Main Flow	* Player inputs a command or action. * The system processes the action and updates the game state accordingly.
Post-Condition	* The game state is updated based on the performed action.
Alternative Flow	The player may cancel or undo certain actions if allowed.
Special Requirements	Responsive input processing and minimal latency.
Related Use Cases	UC-03, UC-04, UC-06, UC-18, UC-27

#### Peace Phase Actions Use Case Description

Elements	Details
ID and Name	UC-06, Peace Phase Actions
Primary Actor	User
Secondary Actor	System
Description	This use case describes the actions and activities available to the player during the Peace Phase of the game. During this phase, the player can manage resources, trade, and perform other peaceful activities without engaging in combat.
Pre-Condition	* The game must be in the Peace Phase, and there should be no active combat or conflict.
Main Flow	* The game transitions into the Peace Phase. * The player accesses the peace-related activities menu (e.g., manage resources, trade). * The player selects an action (e.g., trading, managing resources). * The system processes the selected action (e.g., updating resource counts, completing a trade). * The player performs the selected peaceful activities (e.g., trades, upgrades, manages buildings). * The system confirms the successful completion of actions or activities.
Post-Condition	* The player's resources are updated based on the actions taken during the Peace Phase (e.g., resources are gained or spent, buildings are upgraded, trades are completed).
Alternative Flow	If the player does not have enough resources or is unable to perform a selected action, the system will notify the player and cancel the action.
Special Requirements	The player must not be in any active combat situation, and the game must be in the Peace Phase for these actions to be available.
Related Use Cases	UC-05, UC-07, UC-10, UC-09, UC-17

## Manage Inventory Use Case Description

Elements	Details
ID and Name	UC-07, Manage Inventory
Primary Actor	User, AI
Secondary Actor	System
Description	This use case describes how the player (or AI) manages their inventory, adds, removes, and organizes items. The player (or AI) selects, moves, or uses items in their inventory.
Pre-Condition	* The player (or AI) must have access to the inventory, and there must be at least one item in the inventory.
Main Flow	<ul style="list-style-type: none"> <li>* The player (or AI) accesses the inventory menu.</li> <li>* The system displays the player's (or AI's) current items.</li> <li>* The player (or AI) selects an item from their inventory.</li> <li>* The player (or AI) chooses an action for the selected item (e.g., use, sell).</li> <li>* The system performs the chosen action.</li> <li>* The player (or AI) updates their inventory.</li> </ul>
Post-Condition	* The player's (or AI's) inventory is updated based on the performed action, with the item either used, sold, or moved. The inventory count and item status are updated accordingly.
Alternative Flow	If there is not enough space in the inventory, the system displays a notification and informs the player (or AI) that the item cannot be added.
Special Requirements	The system must be able to correctly update the number, type, and condition of the items in the inventory.
Related Use Cases	UC-08, UC-06

### Equip Item Use Case Description

Elements	Details
ID and Name	UC-08, Equip Item
Primary Actor	User
Secondary Actor	System
Description	This use case describes how the player equips an item to their character. The player can equip items such as weapons, armor, and accessories to enhance the character's abilities or improve stats. The system checks the player's inventory and the item's requirements before allowing the player to equip it.
Pre-Condition	* The player must have an item in their inventory that can be equipped.
Main Flow	<ul style="list-style-type: none"> <li>* The player accesses their inventory.</li> <li>* The system presents the list of items in the inventory.</li> <li>* The player selects the item they wish to equip.</li> <li>* The item is equipped, and the player's stats or abilities are updated.</li> <li>* The system updates the inventory, removing the equipped item from the available inventory items.</li> <li>* The player's character's visual representation is updated.</li> </ul>
Post-Condition	* The item is equipped, and the character's stats or abilities are updated. The item is no longer available in the inventory as it is now equipped.
Alternative Flow	-
Special Requirements	The system must handle equipment that has associated visual changes.
Related Use Cases	UC-07

#### Attribute Point Distribution Use Case Description

Elements	Details
ID and Name	UC-09, Attribute Point Distribution
Primary Actor	User, AI
Secondary Actor	System
Description	This use case describes how the player (or AI) allocates attribute points to customize their character. Attribute points can be distributed across attributes such as Strength, Intelligence, Constitution, Dexterity, Charisma, and Luck, affecting various aspects of gameplay.
Pre-Condition	* The player (or AI) must have earned new attribute points and reviewed their character's current status.
Main Flow	<ul style="list-style-type: none"> <li>* The player (or AI) checks their character's current status and available attribute points.</li> <li>* The system presents an interface to allocate the attribute points.</li> <li>* The player (or AI) distributes points across the following attributes: <ul style="list-style-type: none"> <li>- Strength: Increases damage dealt and enhances physical power.</li> <li>- Intelligence: Boosts mana capacity and spell effectiveness.</li> <li>- Constitution: Improves health points and survival chances.</li> <li>- Dexterity: Enhances agility and reduces incoming damage.</li> <li>- Charisma: Affects trading efficiency, improving buying and selling prices.</li> <li>- Luck: Increases critical success chances in various gameplay aspects.</li> </ul> </li> <li>* The system saves the changes and displays the updated character status.</li> <li>* The player (or AI) proceeds with other actions.</li> </ul>
Post-Condition	* The player's (or AI's) attribute points have been allocated, and the character's updated status has been saved.
Alternative Flow	If points are left unused, the system notifies the player and encourages completing the allocation.
Special Requirements	The system should accurately calculate the impact of attributes on the player's performance.
Related Use Cases	UC-06

## Select Buildings Use Case Description

Elements	Details
ID and Name	UC-10, Select Buildings
Primary Actor	User
Secondary Actor	System
Description	This use case describes the process of selecting a building to interact with, upgrade, or use its functionalities. Actions include initiating construction, performing upgrades, or interacting with NPCs.
Pre-Condition	* The player must have access to the building interface, and the building must be visible and interactable.
Main Flow	<ul style="list-style-type: none"> <li>* The player opens the game interface displaying buildings.</li> <li>* The player clicks on a building they wish to select.</li> <li>* The system highlights the selected building and displays available actions (e.g., upgrade, NPC interaction).</li> <li>* The player selects an action.</li> <li>* The system processes the action and provides feedback.</li> <li>* The interaction concludes.</li> </ul>
Post-Condition	* The building's status is updated based on the selected action (e.g., construction begins, upgrades initiated, or NPC interaction completed).
Alternative Flow	If resources are insufficient for the action, the system notifies the player and offers alternative options.
Special Requirements	The building must be interactable, and the player must have the resources required for actions like upgrades or construction.
Related Use Cases	UC-06, UC-13

## Construct Buildings Use Case Description

Elements	Details
ID and Name	UC-11, Construct Buildings
Primary Actor	User
Secondary Actor	System
Description	This use case describes the phase where players construct buildings in their village during the peace period. Players use their available resources to select a building and place it in a suitable location. This action is only available during the peace period and is strategically important. Initially, only basic buildings can be constructed, which serve as the foundation for game progression.
Pre-Condition	* The player accesses the building menu or interface.
Main Flow	<ul style="list-style-type: none"> <li>* The player selects a specific building type to construct.</li> <li>* The game switches to building placement mode.</li> <li>* The player chooses a suitable location for placement.</li> <li>* The player clicks or taps on the selected location to initiate building placement.</li> </ul>
Post-Condition	<ul style="list-style-type: none"> <li>* The building construction is completed.</li> <li>* The required resources are deducted from the player's total resources.</li> </ul>
Alternative Flow	If the player does not have enough resources, the game checks for sufficient resources before placement. If resources are insufficient, a notification is displayed, and the process is canceled. If an invalid location is selected, the game notifies the player and prompts for a new location.
Special Requirements	The player must have sufficient space and resources. The selected building must be placed in a suitable location, and infrastructure conditions must be met.
Related Use Cases	UC-10, UC-12

### Upgrade Building Use Case Description

Elements	Details
ID and Name	UC-12, Upgrade Building
Primary Actor	User, AI
Secondary Actor	System
Description	This use case describes the process where players upgrade existing buildings to enhance their avatars, granting them new abilities or stronger traits. The upgrade process depends on the player's resources and progress. Upgraded buildings provide the player with improved features, enabling strategic game advancement.
Pre-Condition	<ul style="list-style-type: none"> <li>* The player (or AI) has selected an existing building to upgrade.</li> <li>* The player has sufficient resources and tools for the upgrade.</li> </ul>
Main Flow	<ul style="list-style-type: none"> <li>* The player (or AI) selects an existing building to upgrade.</li> <li>* The player (or AI) accesses the upgrade option for the selected building.</li> <li>* The game checks if sufficient resources and tools are available.</li> <li>* If requirements are met, the upgrade is confirmed, and the process proceeds.</li> <li>* The game completes the upgrade, and the building grants new abilities or powers.</li> </ul>
Post-Condition	* The upgrade is completed, and the building grants new abilities or powers to the player.
Alternative Flow	If the player (or AI) lacks the necessary resources or tools, a notification is displayed, and the process is canceled.
Special Requirements	The player (or AI) must have the necessary resources for the upgrade.
Related Use Cases	UC-11, UC-10



## Interact with the NPC Use Case Description

Elements	Details
ID and Name	UC-13, Interact with the NPC
Primary Actor	User
Secondary Actor	System
Description	This use case describes the process of the player interacting with NPCs. Interactions with NPCs can include trading, accepting quests, initiating dialogue, or other activities.
Pre-Condition	* The player must be in the vicinity of an NPC, and the related building must be dry to initiate interaction.
Main Flow	<ul style="list-style-type: none"> <li>* The player clicks on a building or structure to open the interaction menu.</li> <li>* The player selects the "Talk to NPC" option from the menu.</li> <li>* The system displays available interaction options (e.g., trade).</li> <li>* The player selects an interaction option.</li> <li>* The system processes the selected option.</li> <li>* The NPC interaction concludes.</li> </ul>
Post-Condition	* The interaction is completed, and relevant changes (e.g., resources deducted, quests accepted) are reflected.
Alternative Flow	If the player lacks the necessary resources for trade, the NPC rejects the interaction.
Special Requirements	The related building must be dry in order to interact with the NPC.
Related Use Cases	UC-10, UC-14, UC-15, UC-16

### Craft Item Use Case Description

Elements	Details
ID and Name	UC-14, Craft Item
Primary Actor	User, AI
Secondary Actor	System
Description	This use case describes how the player (or AI) crafts a new item using available resources. Crafting is only possible if the blacksmith building has been constructed. The player selects the desired item, and the system deducts the required resources. Once crafting is complete, the new item is added to the player's inventory.
Pre-Condition	<ul style="list-style-type: none"> <li>* The blacksmith building must be constructed.</li> <li>* The player (or AI) must have sufficient resources to craft the item.</li> </ul>
Main Flow	<ul style="list-style-type: none"> <li>* The player (or AI) accesses the crafting menu at the blacksmith building.</li> <li>* The player (or AI) selects the item to craft.</li> <li>* The game checks if the player has enough resources to craft the item.</li> <li>* If sufficient resources are available, the game proceeds with crafting and deducts the resources.</li> <li>* Upon completion, the crafted item is added to the player's (or AI's) inventory.</li> </ul>
Post-Condition	<ul style="list-style-type: none"> <li>* The crafted item is added to the player's (or AI's) inventory.</li> <li>* The required resources are deducted.</li> </ul>
Alternative Flow	If the player (or AI) does not have enough resources to craft the item, a notification is displayed, and the crafting process is canceled.
Special Requirements	<ul style="list-style-type: none"> <li>* The blacksmith building must be constructed.</li> <li>* The player (or AI) must have the necessary resources to craft the item.</li> </ul>
Related Use Cases	UC-13

#### Sell Item Use Case Description

Elements	Details
ID and Name	UC-15, Sell Item
Primary Actor	User, AI
Secondary Actor	System
Description	This use case describes the process of the player (or AI) selling an owned item through the market. The player selects an item from their inventory, and the market transaction occurs. After the sale, the player's (or AI's) resources are increased by the item's selling price.
Pre-Condition	* The player (or AI) must have the item they wish to sell in their inventory.
Main Flow	<ul style="list-style-type: none"> <li>* The player (or AI) accesses the market.</li> <li>* The player (or AI) selects the item to sell.</li> <li>* The game checks the selling price of the item and the available resources.</li> <li>* If the transaction is valid, the sale proceeds.</li> <li>* After the sale, the player's (or AI's) resources are increased, and the item is removed from the inventory.</li> </ul>
Post-Condition	<ul style="list-style-type: none"> <li>* The sale is completed.</li> <li>* The player's (or AI's) resources are increased, and the item is removed from their inventory.</li> </ul>
Alternative Flow	If the item cannot be sold, the game notifies the player (or AI) and cancels the sale.
Special Requirements	The player (or AI) must have the item in their inventory that they wish to sell.
Related Use Cases	UC-13

#### Buy Item Use Case Description

Elements	Details
ID and Name	UC-16, Buy Item
Primary Actor	User, AI
Secondary Actor	System
Description	This use case describes how the player (or AI) buys a new item from the market, which is already established at the start of the game. The player selects an item and purchases it using available resources. After a successful purchase, the item is added to the player's (or AI's) inventory.
Pre-Condition	* The player (or AI) must have sufficient resources for the purchase.
Main Flow	<ul style="list-style-type: none"> <li>* The player (or AI) accesses the market.</li> <li>* The player (or AI) selects the item to purchase.</li> <li>* The game checks if the player has enough resources.</li> <li>* If sufficient resources are available, the required amount is deducted from the total.</li> <li>* The purchased item is added to the player's (or AI's) inventory.</li> </ul>
Post-Condition	<ul style="list-style-type: none"> <li>* The item is added to the player's (or AI's) inventory.</li> <li>* The player's (or AI's) resources are updated accordingly.</li> </ul>
Alternative Flow	If the player (or AI) lacks enough resources, the game shows a notification, and the purchase is canceled.
Special Requirements	The player (or AI) must have sufficient resources for the purchase.
Related Use Cases	UC-13

#### End Phase Use Case Description

Elements	Details
ID and Name	UC-17, End Phase
Primary Actor	User
Secondary Actor	System
Description	This use case outlines the process of completing the current phase of the game and preparing for the next. Once the player completes the current phase, the game transitions to the next phase.
Pre-Condition	* The player must have completed all requirements for the current phase.
Main Flow	<ul style="list-style-type: none"> <li>* The system checks if the current phase is complete.</li> <li>* The player is notified that the phase is finished.</li> <li>* The system records the player's progress.</li> <li>* Necessary preparations for the next phase are made.</li> <li>* Once the end phase is completed, the game is ready to transition to the next phase.</li> </ul>
Post-Condition	* The player's progress is recorded, and preparations for the next phase are completed.
Alternative Flow	If the phase is incomplete or the requirements are not met, the system informs the player and provides guidance on completing the phase.
Special Requirements	The system must accurately record the player's rewards and progress.
Related Use Cases	UC-18, UC-06

## War Phase Actions Use Case Description

Elements	Details
ID and Name	UC-18, War Phase Actions
Primary Actor	User
Secondary Actor	System
Description	This use case outlines the actions the player can perform during the war phase. The player can gather materials, engage in combat, or participate in tournaments. These actions influence the outcomes of the war phase. The system processes the player's input and provides feedback.
Pre-Condition	<ul style="list-style-type: none"> <li>* The war phase must be active.</li> <li>* The player must be able to make choices based on the current phase.</li> </ul>
Main Flow	<ul style="list-style-type: none"> <li>* The system starts the war phase and presents the player with an overview of the battlefield or available options.</li> <li>* The player chooses an action (e.g., Gather Materials, Go to the Colosseum, or other strategic options).</li> <li>* The system processes the action, considering factors such as resources, environmental conditions, AI actions, and player stats.</li> <li>* If "Gather Materials" is selected, the system rolls a d20 die to determine: <ul style="list-style-type: none"> <li>- Whether an encounter occurs.</li> <li>- The multiplier for gathered materials (0.5x, 1x, 2x).</li> </ul> </li> <li>* If an encounter occurs, the player can choose to: <ul style="list-style-type: none"> <li>- Flee: A d20 die is rolled to check for success. If successful, materials are safely collected (1x).</li> <li>- Engage in combat: A turn-based combat scene is initiated.</li> </ul> </li> <li>* If "Go to the Colosseum" is selected, the player participates in a gladiator tournament, fighting sequential opponents.</li> <li>* After each action's outcome, the player evaluates the results and chooses the next step.</li> <li>* The system ends the war phase, calculates the final outcomes, and prepares for the next phase.</li> </ul>
Post-Condition	<ul style="list-style-type: none"> <li>* The war phase ends, and the player's actions and decisions are recorded.</li> <li>* The system calculates the final outcomes and prepares for the next phase.</li> </ul>
Alternative Flow	If a sudden event occurs, the system dynamically adjusts the results and informs the player.
Special Requirements	The system must generate realistic and dynamic war outcomes based on the player's stats and decisions.
Related Use Cases	UC-05, UC-17, UC-20

#### Gather Use Case Description

Elements	Details
ID and Name	UC-19, Gather
Primary Actor	User, AI
Secondary Actor	System
Description	This use case determines the type and number of enemies the player (or AI) will encounter during a gathering action. It specifies which enemies will appear and how many.
Pre-Condition	* The player (or AI) interacts with a section of the game where enemy selection occurs.
Main Flow	* The system determines the types and numbers of enemies using dice rolls. * The selected enemies and their numbers are displayed to the player (or AI).
Post-Condition	* The player (or AI) proceeds to fight the selected enemies.
Alternative Flow	If the player uses a special item or ability, it may affect the type and number of enemies. For instance, using a "hard level" item could result in stronger enemies appearing.
Special Requirements	The results of the gathering action should be communicated to the player with visual and sound effects.
Related Use Cases	UC-18, UC-21

#### Go to Colosseum Use Case Description

Elements	Details
ID and Name	UC-20, Go to Colosseum
Primary Actor	User, AI
Secondary Actor	System
Description	This use case involves the player (or AI) choosing one of three options in the Colosseum: Encounter, Gladiator, or King. By selecting one of these, the player enters the arena and engages in battle based on the chosen difficulty level. Each option offers a unique challenge, encouraging strategic decision-making.
Pre-Condition	The player (or AI) decides to enter the arena.
Main Flow	* The system presents three options: Encounter, Gladiator, or King. * The player (or AI) selects their desired challenge. * The system determines the appropriate difficulty and enemies for the chosen option and initiates the battle.
Post-Condition	The player (or AI) proceeds with the chosen strategy to succeed in the battle.
Alternative Flow	If the player (or AI) loses the battle, they can return to the arena, select a different option, or try a new strategy.
Special Requirements	Unique enemy structures and strategies must be designed for each type of combat.
Related Use Cases	UC-18, UC-21

## Combat Actions Use Case Description

Elements	Details
ID and Name	UC-21, Combat Actions
Primary Actor	User, AI
Secondary Actor	System
Description	This use case occurs when the player (or AI) engages in turn-based combat. During the battle, the player has multiple tactical options, including attacking with an equipped weapon, using scrolls or mana for powerful skills, repositioning, waiting and observing the enemy's next move, or adopting a defensive stance to reduce incoming damage.
Pre-Condition	The player (or AI) decides to enter turn-based combat.
Main Flow	<ul style="list-style-type: none"> <li>* The player (or AI) enters the battle scene.</li> <li>* The system presents the tactical options to the player (or AI).</li> <li>* The player (or AI) selects an action based on the current situation (attack, defense, skill use, movement, waiting, etc.).</li> <li>* The chosen action affects the battle's outcome.</li> </ul>
Post-Condition	The player's (or AI's) selected tactic impacts the outcome of the battle.
Alternative Flow	If the player (or AI) chooses to attack and the attack is successful, but the enemy counters and returns the attack, the player may take damage, but the battle continues.
Special Requirements	<ul style="list-style-type: none"> <li>* Visual and sound effects should be customized to the action (e.g., sword attack, magic use).</li> <li>* The success of actions depends on the player's (or AI's) abilities and resources (e.g., mana, stamina).</li> <li>* Enemy reactions should dynamically respond to the player's (or AI's) actions.</li> <li>* Actions should be flexible, allowing the player (or AI) to change their strategy during combat.</li> </ul>
Related Use Cases	UC-19, UC-20, UC-22, UC-23, UC-24, UC-25

#### Attack Use Case Description

Elements	Details
ID and Name	UC-22, Attack
Primary Actor	User, AI
Secondary Actor	System
Description	In turn-based combat, the player (or AI) can attack an enemy using an equipped weapon. This action deals damage to the enemy and affects the course of the battle.
Pre-Condition	The player (or AI) has equipped a weapon for the attack.
Main Flow	<ul style="list-style-type: none"> <li>* The player (or AI) selects the "attack" option to use the weapon.</li> <li>* The system calculates the damage based on the player's (or AI's) attack power and the enemy's defense.</li> <li>* The attack is applied to the enemy, and the damage is displayed.</li> </ul>
Post-Conditions	The enemy reacts to the damage received.
Alternative Flow	If the player's (or AI's) weapon is damaged or there is not enough power to attack (e.g., low durability or lack of skill), the player cannot attack. In this case, the player must choose another action, such as defense or using a skill.
Special Requirements	<ul style="list-style-type: none"> <li>* Attack success percentage should be proportional to the player's (or AI's) attack power and the enemy's defense value.</li> <li>* Attack animations and effects should be customizable according to the attack type (e.g., sword attack, arrow shooting).</li> <li>* Sound and visual effects should align with the attack type.</li> </ul>
Related Use Cases	UC-21

#### Defend Use Case Description

Elements	Details
ID and Name	UC-23, Defend
Primary Actor	User, AI
Secondary Actor	System
Description	In turn-based combat, the player (or AI) can reduce incoming damage by taking a defensive stance. This minimizes the damage received from enemy attacks.
Pre-Conditions	The player (or AI) selects the "defend" option.
Main Flow	<ul style="list-style-type: none"> <li>* The system reduces the damage based on the player's (or AI's) defense power.</li> <li>* The enemy's attack is applied to the player (or AI) in the defensive stance.</li> </ul>
Post-Conditions	The player takes less damage as a result of the defensive stance.
Alternative Flow	If the player (or AI) does not have enough resources (e.g., mana or skills) for defense, the defense fails, and the player takes damage.
Special Requirements	<ul style="list-style-type: none"> <li>* The defense mechanism should dynamically calculate defensive strength (e.g., based on armor, skills, or temporary buffs).</li> <li>* Defensive animations and visual effects can be customized based on the type of defense (e.g., shield, positioning).</li> <li>* Appropriate feedback should be given when defense fails.</li> </ul>
Related Use Cases	UC-21



#### Use Skill Use Case Description

Elements	Details
ID and Name	UC-24, Use Skill
Primary Actor	User, AI
Secondary Actor	System
Description	The player (or AI) can use powerful skills during combat by utilizing scrolls or mana. These skills can affect the enemy or provide advantages to the player.
Pre-Condition	The player (or AI) has a skill and the necessary resource (mana or scroll).
Main Flow	<ul style="list-style-type: none"> <li>* The player selects the "use skill" option.</li> <li>* The system considers the effect of the selected skill and the required resource (mana or scroll).</li> <li>* The skill's result applies damage to the enemy, provides defense, or grants another advantage to the player (or AI).</li> </ul>
Post-Condition	After using the skill, the resource level (mana or scroll) decreases.
Alternative Flow	If an error occurs during skill use (e.g., incorrect skill selection or target), the skill fails. The player will spend the resource but the skill will have no effect.
Special Requirements	<ul style="list-style-type: none"> <li>* Visual effects and sounds during skill use should be customized based on the skill type (e.g., magic damage, armor break, speed increase).</li> <li>* The effect of skill use should vary depending on the skill type and the resource used.</li> <li>* The skill success percentage should depend on the player's (or AI's) skill level and resource amount.</li> </ul>
Related Use Cases	UC-21

#### Flee Use Case Description

Elements	Details
ID and Name	UC-25, Flee
Primary Actor	User
Secondary Actor	System
Description	The player (or AI) attempts to flee from a battle or dangerous situation to avoid defeat or damage.
Pre-Conditions	The player presses the flee button.
Main Flow	* The user is informed of the consequences if they escape.
Post-Conditions	The player either escapes successfully or continues the encounter after a failed attempt.
Alternative Flow	The player does not select the flee option and continues the combat/danger scenario.
Special Requirements	* Clear notification of success/failure and immediate transition to safe locations on success.
Related Use Cases	UC-21

### Credits Use Case Description

Elements	Details
ID and Name	UC-26, Credits
Primary Actor	User
Secondary Actor	System
Description	The player views information about the development team and contributors.
Pre-Condition	The player is on the main menu screen.
Main Flow	<ul style="list-style-type: none"> <li>* Player clicks on the "Credits" button.</li> <li>* The system displays a list of contributors.</li> <li>* Player scrolls through or exits to return to the main menu.</li> </ul>
Post-Condition	The player views the credits or returns to the main menu.
Alternative Flow	-
Special Requirements	The credits list should be easy to read and navigate.
Related Use Cases	UC-30

### Settings Use Case Description

Elements	Details
ID and Name	UC-27, Settings
Primary Actor	User
Secondary Actor	System
Description	The player adjusts game settings such as effects, music, and graphic quality.
Pre-Condition	The player is on the main menu screen.
Main Flow	<ul style="list-style-type: none"> <li>* Player clicks on the "Settings" button.</li> <li>* The system displays options: Effects, Music, Graphic Quality, and Return.</li> <li>* Player adjusts settings and confirms or selects Return to go back.</li> </ul>
Post-Condition	Changes are applied (if selected) or the player returns to the main menu.
Alternative Flow	-
Special Requirements	-
Related Use Cases	UC-29, UC-28, UC-05

#### Adjust Sound Use Case Description

Elements	Details
ID and Name	UC-28, Adjust Sound
Primary Actor	User
Secondary Actor	System
Description	The player adjusts sound settings such as effects and music volume.
Pre-Condition	The player is on the Settings screen.
Main Flow	<ul style="list-style-type: none"> <li>* System shows sound options.</li> <li>* Player adjusts sliders.</li> <li>* System applies changes in real-time or upon confirmation.</li> </ul>
Post-Condition	The sound settings are updated.
Alternative Flow	-
Special Requirements	Real-time audio feedback for sound adjustments.
Related Use Cases	UC-27, UC-29

#### Adjust Graphics Use Case Description

Elements	Details
ID and Name	UC-29, Adjust Graphics
Primary Actor	User
Secondary Actor	System
Description	The player adjusts graphic settings such as quality and resolution.
Pre-Condition	The player is on the Settings screen.
Main Flow	<ul style="list-style-type: none"> <li>* Player selects a graphical setting.</li> <li>* System applies changes in real-time or upon confirmation.</li> </ul>
Post-Condition	The graphical settings are updated.
Alternative Flow	-
Special Requirements	Real-time preview of graphical changes.
Related Use Cases	UC-27, UC-28

## Show Contributors Use Case Description

Elements	Details
ID and Name	UC-30, Show Contributors
Primary Actor	User
Secondary Actor	System
Description	The player views a list of contributors from the Credits menu.
Pre-Condition	The player is present in the main menu screen.
Main Flow	<ul style="list-style-type: none"> <li>* The player clicks on the button marked "Credits"</li> <li>* The system will show the players a list of names and their corresponding roles.</li> <li>* The player then has the option to go back and browse that list.</li> </ul>
Post Condition	Player has either seen who all are the contributors or has gone back to the main menu.
Alternative Flow	-
Special Requirements	List of contributors: The list of contributors has to be neat readable format.
Related Use Cases	UC-26

## 5. Planning

### 5.1 Team Structure

- Doç. Dr. Gül TOKDEMİR - Advisor
- Dr. Öğr. Üyesi Talha KARADENİZ - Advisor
- Arş. Gör. Sezer UĞUZ - Advisor
- Dorukhan YILDIZ – Designer, Developer
- Sezer Can EKİZ- Developer
- Ömer YURTALAN - Developer
- Semih OĞUZ- Developer

### 5.2 Estimated Schedule

Start Date 30/09/2024	Current Status	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14	WEEK 15	WEEK 16
Team Setup	Completed																
Project Proposal Form	Completed																
Project Selection Form	Completed																
GitHub Repository	Completed																
Project Work Plan	Completed																
Literature Review	Completed																
Software Requirements Specification	Completed																
Project Webpage	Completed																
Software Design Description	Completed																
Project Report / Project Tracking Form	In progress																
Presentation	Not Started																

Figure 2: Work Plan.

## 5.3 Process Model

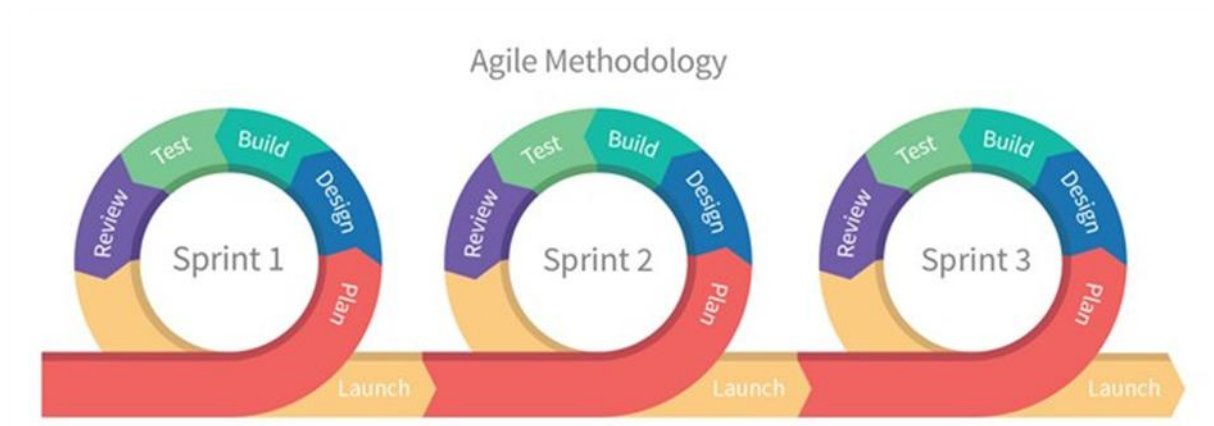


Figure 3: Agile Methodology.

## 6. Conclusion

This SRS provides the basic document that outlines the objectives, features, and technical requirements of the game. It emphasizes how the game merges new game mechanics, such as strategic actions and decisions, dice combats, and a human-like adaptive AI opponent. Each requirement outlined in this document shall guide the development team. It also ascertains that collaboration among team members and stakeholders is clearly elaborated on. This paper will provide one common understanding, scope, and a set of constraining parameters from which to refer for the continued phases of a project, that is, implementation, testing, up to deployment.

# Software Design Description

## 1. Introduction

### 1.2 Purpose of this Document

This document is a bridge between the Game Design Document (GDD) and Software Design Document (SRS). While the GDD outlines the conceptual overview such as gameplay mechanics, story, and artistic vision of "Rule and Munus," this document combine it with explaining the technical specifics like architecture, algorithms, and implementation strategies necessary to develop the game. We can divide the purpose of the document as follows:

1. **Guiding:** Providing a framework for developers, enabling the transition from design to implementation.
2. **Ensure Consistency:** Ensures arrangement between the concept of the game and the technical application by defining specifications.

### 1.3 Scope of the Project

"Rule and Munus" is a turn-based strategy game set in an isometric 2D world. It blends elements of tactical RPGs and roguelikes. The gameplay revolves around two phases: "Peace Phase" and "War Phase." The standout feature of the game is the adaptive AI system controlling the King of the Gladiators. This AI evolves in parallel with the player, analyzes their strategies, and grows in strength to provide a continuous and challenging experience. We preferred the dice-based mechanics to add an element of chance that influences most of the features from crafting outcomes to combat scenarios. The project aims to offer a replayable experience that combines strategic depth and dynamic gameplay.

## 2. Team Information

- Doç. Dr. Gül TOKDEMİR - Advisor
- Dr. Öğr. Üyesi Talha KARADENİZ - Advisor
- Arş. Gör. Sezer UĞUZ - Advisor
- Dorukhan YILDIZ – Designer, Developer
- Sezer Can EKİZ - Developer
- Ömer YURTALAN - Developer
- Semih OĞUZ - Developer

### 3. Glossary

Term	Description
AI (Artificial Intelligence)	A computational system designed to mimic human-like decision-making. In this game, the King of the Gladiators is controlled by a human-like AI that adapts to player actions.
Action Points (AP) / Stamina	A resource consumed by the player or AI during combat to perform actions such as attacking, defending, or using skills.
Dice Mechanics	A feature where dice rolls determine the success of actions, quality of crafted items, and outcomes of events. Examples include d20 for material gathering and d100 for item crafting quality.
Encounter	A random event during the War Phase, where the player faces an enemy or gathers materials based on dice roll outcomes.
Heuristic-based Decision-making	A method used by the game's AI to make decisions based on simplified rules rather than complex calculations, balancing performance and adaptability.
Isometric	A visual perspective used in the game to present the town and combat scenes, offering a pseudo-3D appearance on a 2D plane.
Peace Phase	The gameplay phase focused on town management, crafting, and character development.
Roguelike	A sub-genre of games characterized by procedurally generated challenges, randomness, and a high degree of replayability.
Stat Points	Points allocated by players to improve their character's attributes, such as Strength, Dexterity, or Intelligence, enhancing performance in combat and other activities.
Turn-based Combat	A combat system where players and opponents take turns performing actions based on available resources like action points.
War Phase	The gameplay phase where players gather materials, engage in combat, or participate in the Colosseum.
User Interface (UI)	The visual and interactive components of the game that allow players to perform actions, manage inventory, and navigate game menus.

### 4. Scope of the Project

#### 4.1 Inspirations

"Rule and Munus" get inspired by several games, integrating their notable features to create a unique experience:

##### 1. Swords and Sandals:

- Features adapted: Gladiatorial combat and character progression mechanics.
- Implementation: Participating in the gladiatorial tournaments under the name of "Munus" and a level-based progression system.

##### 2. Loop Hero:

- Features adapted: Town-building and resource management elements.
- Implementation: Players can construct and upgrade buildings, manage resources, and craft items.

### **3. Dungeons and Dragons:**

- Features adapted: Dice mechanics and fantastic themes.
- Implementation: Dice rolls influence combat, crafting, and resource gathering, adding randomness. Fantastic themes such as character attributes, classes, races and skills.

### **4. Divinity 2: Original Sin:**

- Features adapted: AI opponents and turn, and action point-based combats.
- Implementation: The Decision of the AI opponent of Divinity is strategic, and situation-based that's what we wanted from "King of the Gladiators". The combat scene has its own turn mechanic and opponents of both sides will take actions based on their stamina like action points.

### **5. Sid Meier's Civilization VI:**

- Features adapter: Turn-based strategy.
- Implementation: Peace and War Phases are our main turns the player will take actions depending on the current phase. These inspirations provide the basis for the game while allowing us to innovate on the mechanics.

## **4.2 Theme/Genre/Concept**

- **Theme:**
  - A high-fantasy world where players take on the role of an exiled hero rebuilding their life in a foreign land.
- **Genre(s):**
  - Turn-based strategy, tactical RPG, and roguelike.
- **Concept:**
  - The game emphasizes replayability through dynamic gameplay and dice-based mechanics by combining strategic decision-making and resource management with turn-based combat and a dynamically evolving AI opponent.

## **4.3 Targeted Platforms and Audience**

- **Platforms:**
  - PC (primary development focus), with potential for console support in the future.
- **Audience:**
  - Lovers of tactical RPGs, turn-based strategy games, and roguelikes.
  - Players who enjoy fantastic characters and strategic gameplay.
  - Steam users who are seeking innovative single-player experiences.

## **4.4 Monetization Model**

"Rule and Munus" will adopt a straightforward monetization model:

- The game will be published on Steam as a premium product.
- Players can purchase the game for a one-time cost, granting full access to all features and content.
- No microtransactions or additional purchases will be required, ensuring a fair and complete experience for all players.



## 5. Game Design

### 5.1 Story

Depending on the main story, the player's character is exiled from the Kingdom of Gladiators by the King, takes refuge in a town, and is elected Mayor as time passes. Players create a character based on his/her preferences (class, race, name, etc.) and the game starts in this small town that can be named in the beginning, at Peace Phase with some initial items and materials. The main questline is defeating all opponents at the coliseum, climbing stages till facing the "King of the Gladiators" and beating him to overcome exile.

### 5.2 Game Structure

"Rule and Munus" is structured around two phases:

#### 1. Peace Phase:

- Players rule the town, focusing on resource management, town-building, and character progression. Actions include constructing and upgrading buildings, crafting and trading items, and allocating stat points to improve their character's abilities and prepare their character for upcoming challenges.

#### 2. War Phase:

- Players gather materials and engage in turn-based combat with encounters ranging from regular enemies to the AI-driven opponent, the King of the Gladiators. Players can also compete in gladiatorial tournaments at the Colosseum, climbing ranks to face progressively stronger foes.

These phases are linked, with the outcomes of one directly affecting the other. This cyclical structure encourages strategic planning.

### 5.3 Character

#### 5.3.1 Player Definitions:

The player is the controller of the main character of the story. It has the following properties:

- **Inventory**
- **Stats**
- **Attributes**
- **Actions**

Basically, the player is the mayor, gatherer, fighter, and strategist.

#### 5.3.2 Player Properties

##### Inventory:

- Helmet, Chest plate, Leggings, Ring, Necklace, Scrolls, Firsthand and Secondhand weapons are the equipment slots.
- Every slot is a tag. Tagged items are stored in the inventory by default. The player must equip the items.
- Materials are stored in the inventory by default just for inspection.

**Stats:**

- Strength: Increases the damage of hitting with the equipped item.
- Intelligent: Increases the maximum number of Mana Points and damage of Skills.
- Constitution: Increases the maximum number of Health Points.
- Dexterity: Increases the amount of neglected damage while defending.
- Charisma: Decreases the value of buying an item and increases the value of selling an item on the market.
- Luck: Increases the chance of hitting, looting high-quality items, crafting high-quality items, and the amount of gathered materials. Attributes:
- Health depends on the Constitution. It can be regenerated by skills. The remaining health will be transferred to the next turn at War Phase. The player can refill it by ending the War Phase.
- Mana depends on the Intelligent. The remaining mana will be transferred to the next turn at War Phase. The player can refill it by ending the War Phase
- Stamina is constant at 80. Starts with 60 points at the beginning of combat. Unspent stamina will be added on the next turn but will not exceed 80.

## 5.4 Actions

"Rule and Munus" features a variety of player actions as outlined in the GDD:

- **Town Management:**
  - Build new buildings.
  - Upgrade existing buildings.
  - Craft new items.
  - Sell and buy items at the market.
  - Allocate stat points to improve character attributes.
- **Combat Actions:**
  - Attack by equipped weapons.
  - Defend to reduce incoming damage.
  - Use skills by consuming mana or scrolls.
  - Move within the combat grid.
  - End turn after using available stamina.
- **Resource Gathering:**
  - Roll dice to determine gathering outcomes or trigger encounters.
  - Attempt to flee or engage in combat if encounters occur.
- **Dice Rolls:**
  - Use d20 rolls for material gathering and combat escape attempts.
  - Use d100 rolls to determine the quality of crafted items. These actions combine strategic decision-making with randomness, enhancing the replayability of the game.

## 5.5 Objective

The primary objectives for players are:

- **Win Condition:**
  - Defeat the King of the Gladiators, in the final Colosseum battle.
- **Lose Condition:**
  - Although there is no permanent game-over state, players face setbacks such as losing resources and progress if defeated in combat. These setbacks challenge players to rebuild their strategies and try again.

## 5.6 Gameplay

"Rule and Munus" offers a cyclical gameplay loop blending strategic planning and tactical execution:

- During the Peace Phase, the player develops their town, manages resources, and improves their character for upcoming challenges.
- In the War Phase, the player gathers materials and faces turn-based combat scenarios.
- Participation in gladiatorial tournaments provides high-risk, high-reward opportunities to advance the story and character progression. This dynamic gameplay loop, supported by adaptive AI ensures a challenging experience for the player.

## 6. Software Design

### 6.1 Component-Based Architecture

This architecture in system design refers to a methodology where software is built by assembling pre-defined, reusable components. Each component encapsulates a specific piece of functionality or behavior, with well-defined interfaces that govern how components interact with each other. This approach promotes modularity, flexibility, and reusability in software development. [\[1\]](#)

The reason we chose this architecture is to ensure that the system is modular, flexible, and sustainable. Component-Based Architecture allows components to be developed and tested independently. This enables each component to be designed more efficiently by focusing on its specific functionality. Additionally, the reusability of components speeds up the development process and reduces maintenance costs. In the long run, this approach will make system maintenance and expansion easier and more efficient.

### 6.2 Decomposition

#### 6.2.1 Building Construction

Building construction is an important gameplay mechanism that allows players to unlock new features and functions. This includes gaining access to interactions with related NPCs, such as shops, blacksmiths, or quest-giving characters. Buildings also serve as symbols of the player's progress. For example, constructing a forge allows for weapon crafting, offering the player new strategic options.

### 6.2.2 Character Attributes

Character attributes define the core abilities of the player character and have a significant impact on gameplay. These attributes include:

- **Class:** Shapes character's abilities and defines a playstyle (e.g., Warrior, Mage, Archer).
- **Race:** Offers diversity and provides special bonuses (e.g., Elves gain bonus Dexterity, Humans gain bonus Wisdom).
- **Attributes:** Strength, Intelligence, Constitution, Dexterity, Charisma, and Luck are attributes that affect a character's actions. Strength increases melee damage, Intelligence boosts mana points and the power of skills, Constitution affects maximum health points, Dexterity reduces chance of damage taken, Charisma influences the buying and selling value of items, and Luck improves attack accuracy, and the amount of materials gathered.
- 

### 6.2.3 Combat Actions

Combat actions are the mechanics of battle. The core actions include:

- **Attack:** The character initiates an attack, supported by their stats and the weapon used.
- **Defense:** This involves using shields, dodging, or reducing incoming damage through counterattacks.
- **Movement:** Getting close to the enemy or retreating.
- **Use Skill:** Using special abilities such as magic, buffs, or control effects, typically consuming resources like mana or energy.
- **End Turn:** After using available stamina, the turn ends, and any unused stamina will be carried over to the next round but will not exceed the maximum limit.

### 6.2.4 Resource Management

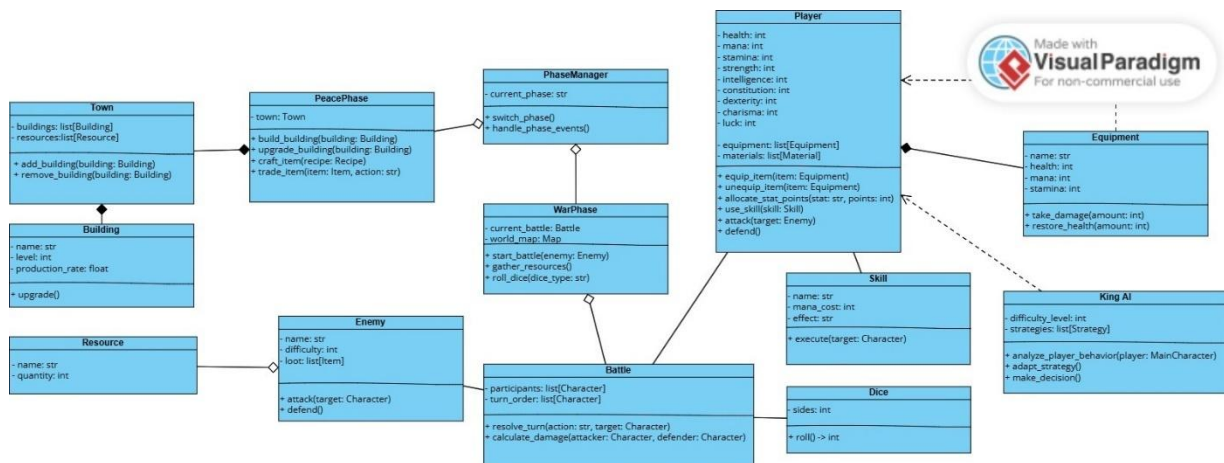
Resource management challenges players to effectively gather and allocate resources. The core elements of this system include:

- **Gathering:** Players collect materials by rolling a d20 dice. The roll determines the number of materials gathered, influenced by a multiplier (e.g., 0.5x, 1x, 2x). This mechanic shapes the quantity and variety of resources players can acquire.
- **Spending:** Players use gathered resources to build structures and craft items. Effective resource management requires balancing short-term needs with long-term objectives.

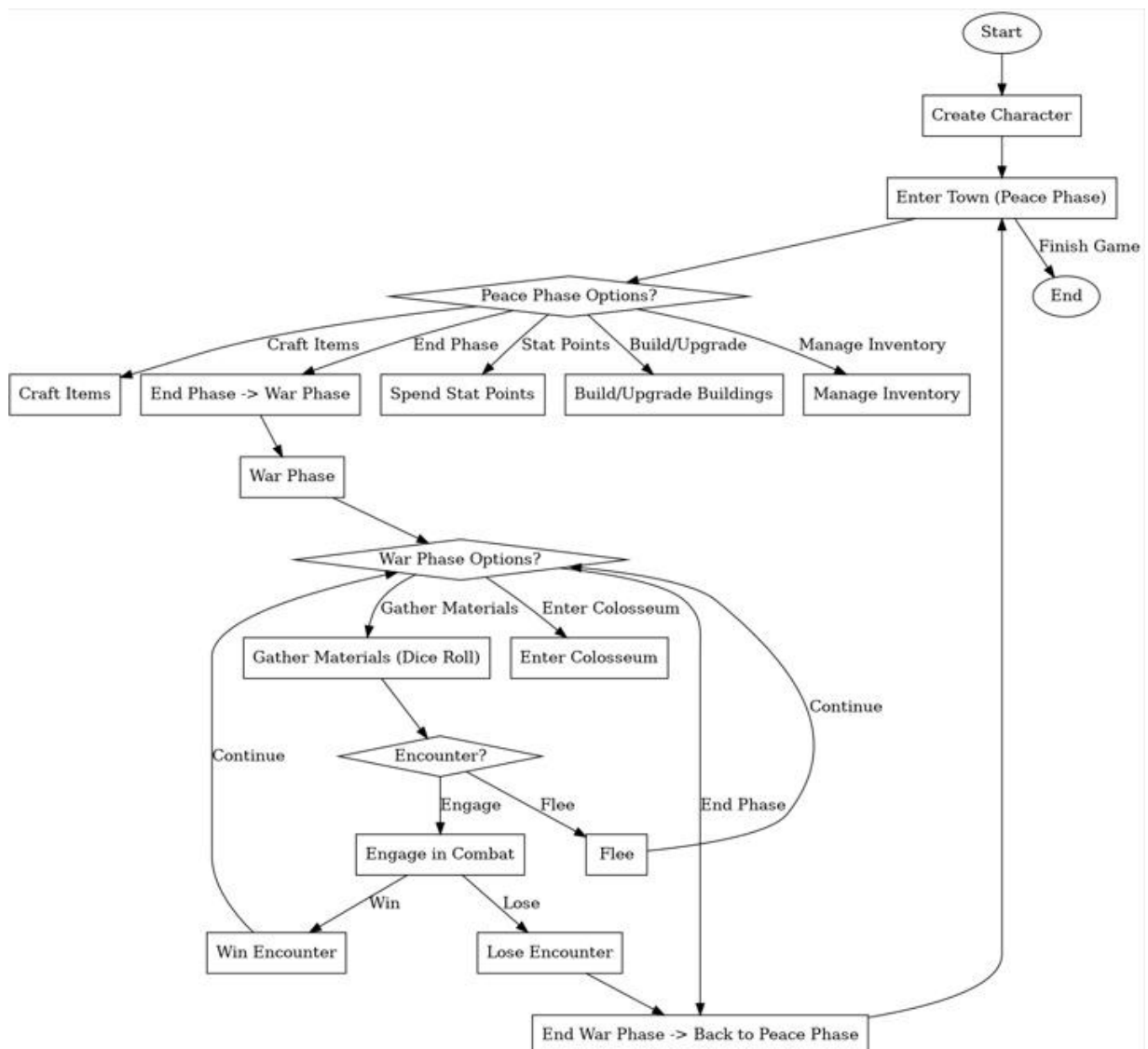
### 6.2.5 Artificial Intelligence (AI)

The AI influences the game by controlling gladiators and encounters, especially in the Colosseum tournaments. Each stage has gladiators and a boss, with the final boss, "King of the Gladiators," using a dynamic AI that grows stronger as the player progresses. The AI adapts to the player's tactics, improving its strength and kingdom based on shared actions like gathering and crafting. In combat, the AI reacts to the player's strategies, adjusting its actions to create a challenging experience. The "King of the Gladiators" represents the final challenge, with the AI becoming tougher as the game advances.

## 6.3 Class Diagram

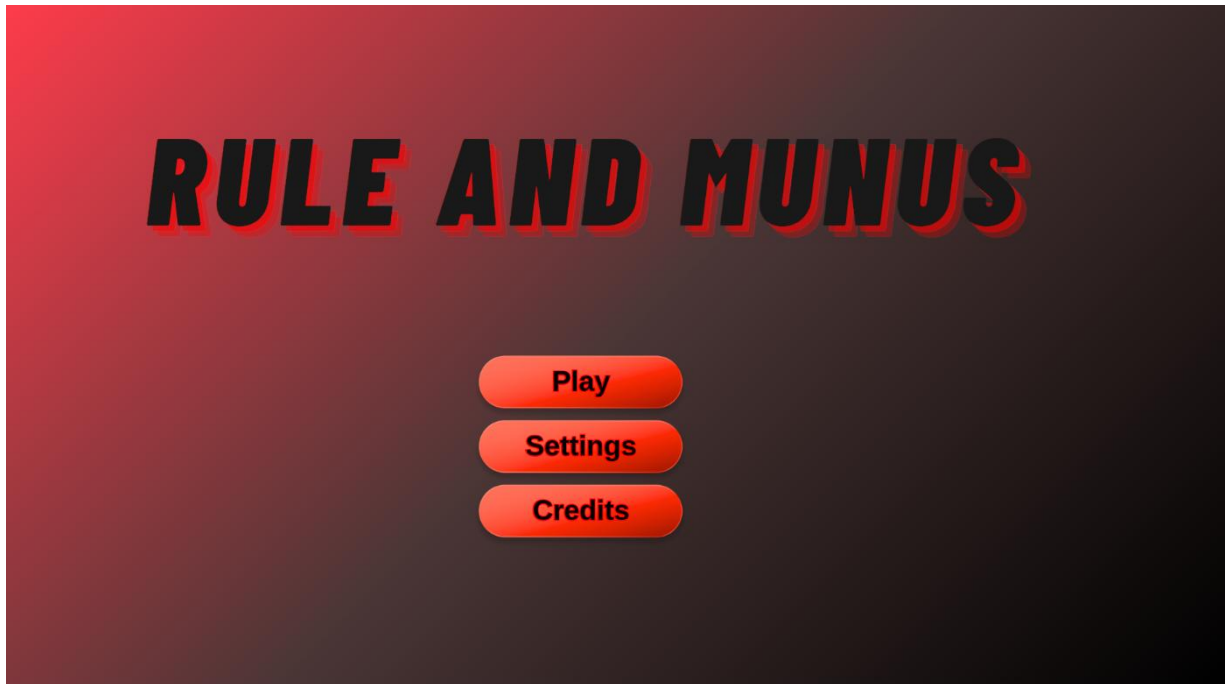


## 6.4 Activity Diagram

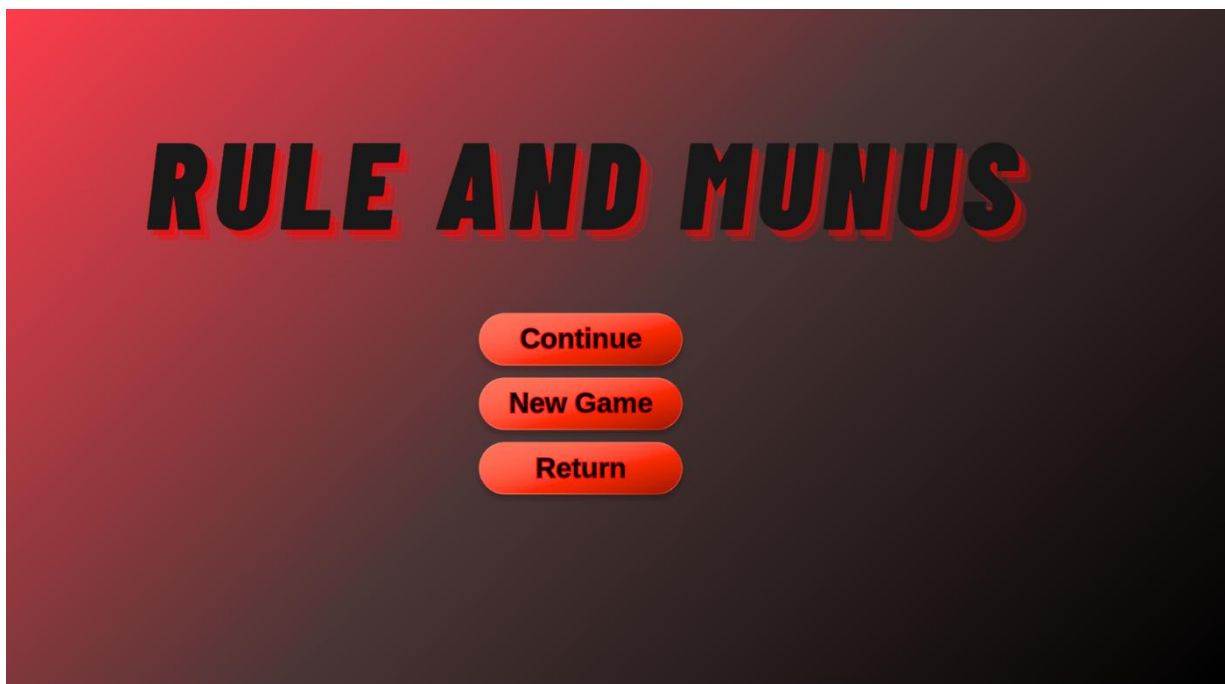


## 7. User Interface Design (UI)

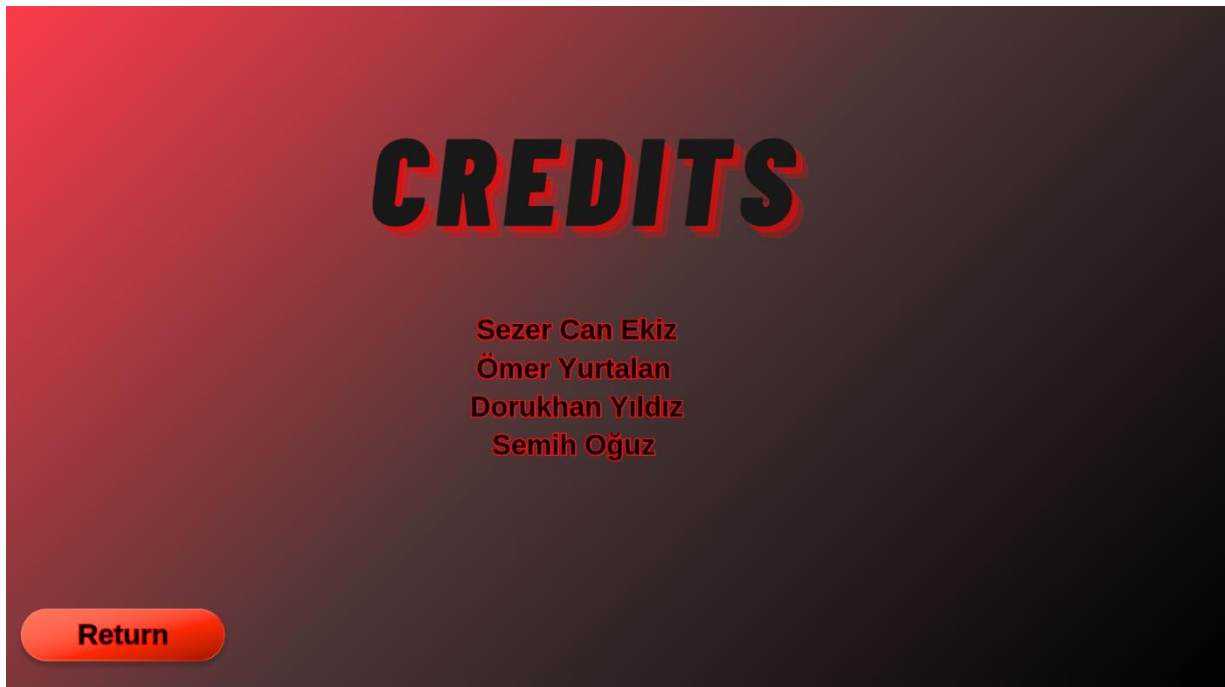
### 7.1 Main Menu



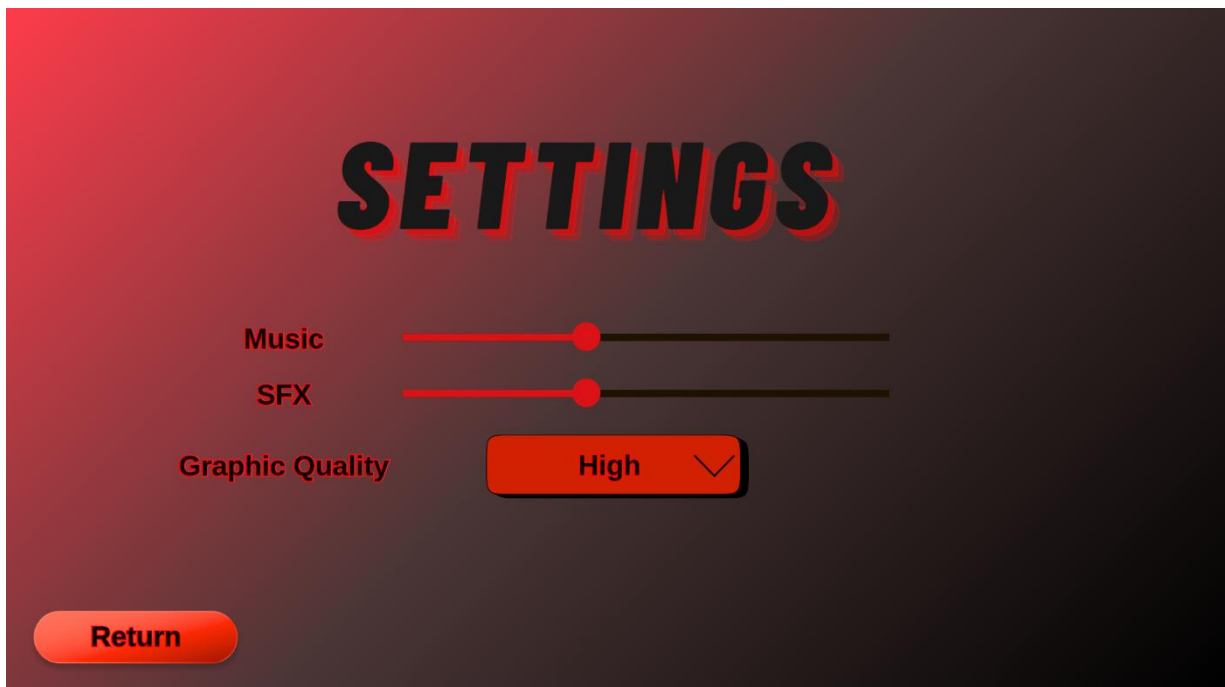
### 7.2 Play



### 7.3 Credits



### 7.4 Settings

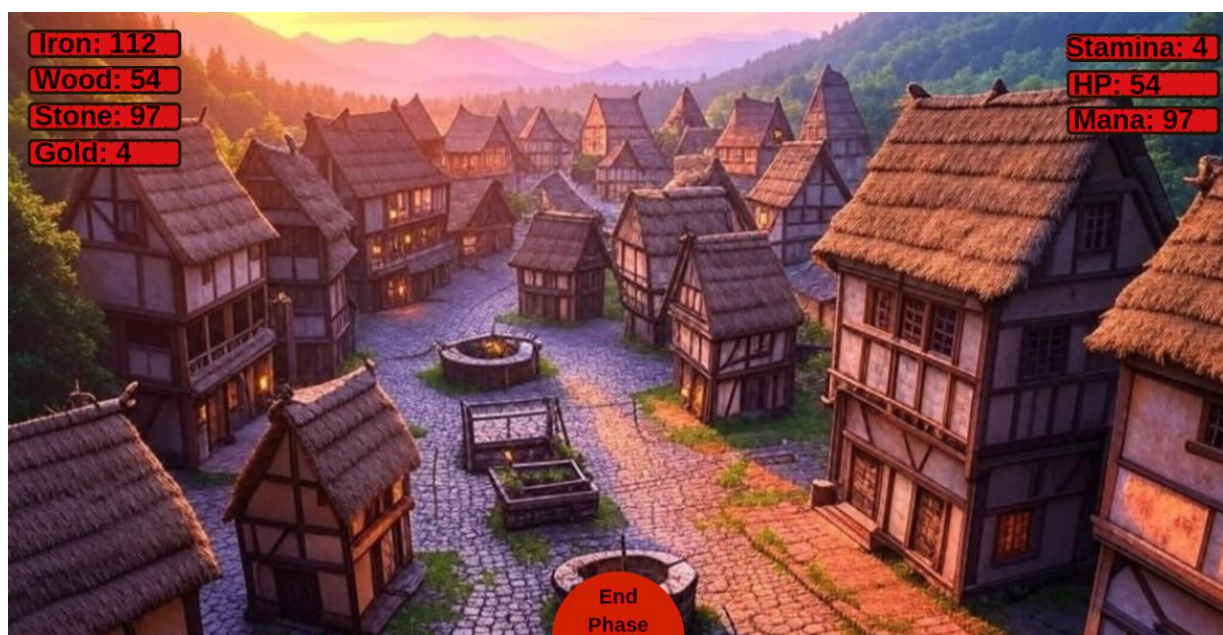




## 7.5 New Game

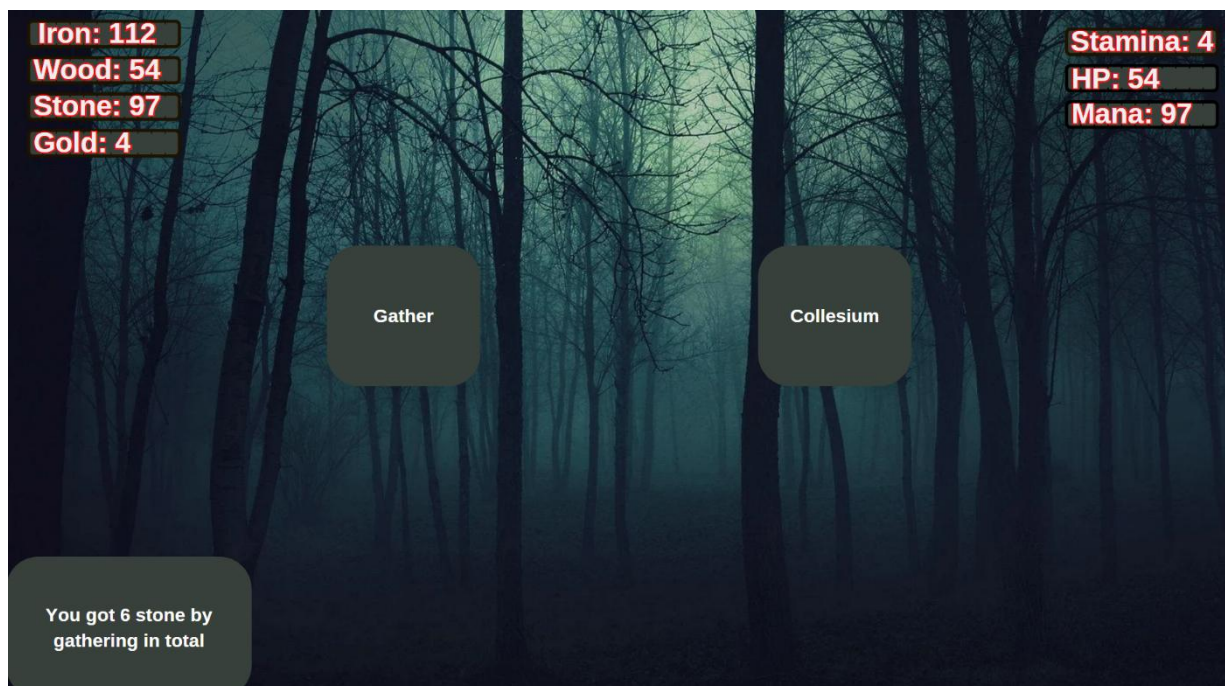


## 7.6 Peace Phase

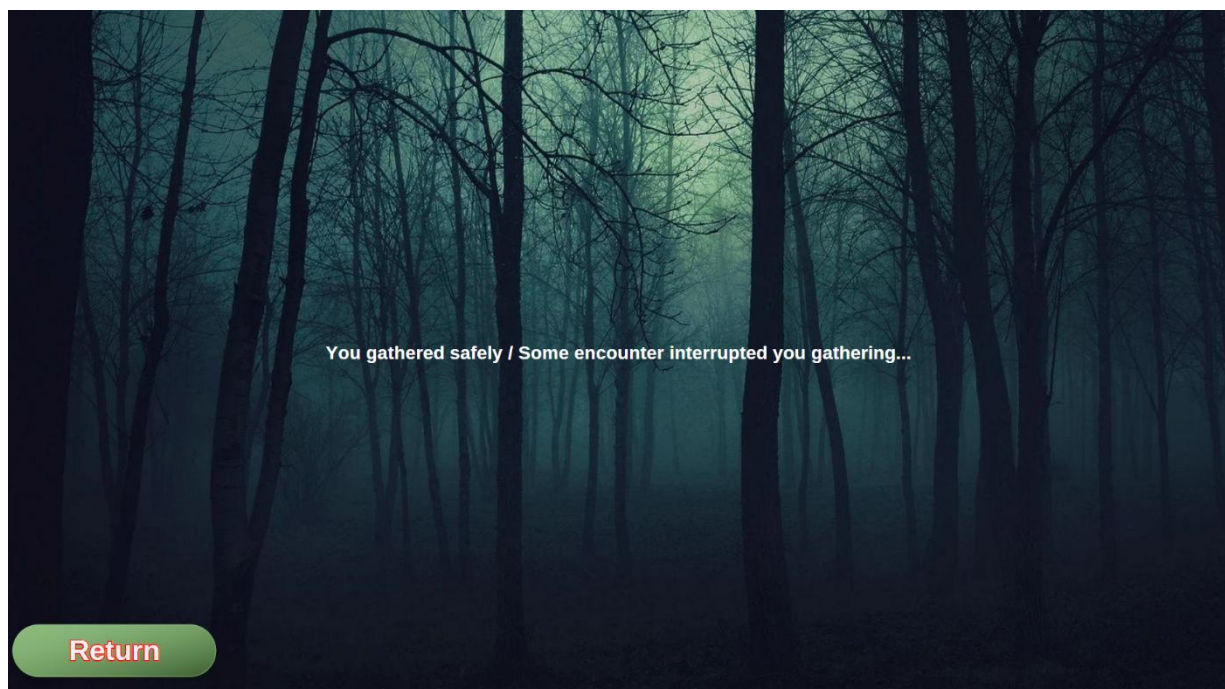




## 7.7 War Phase



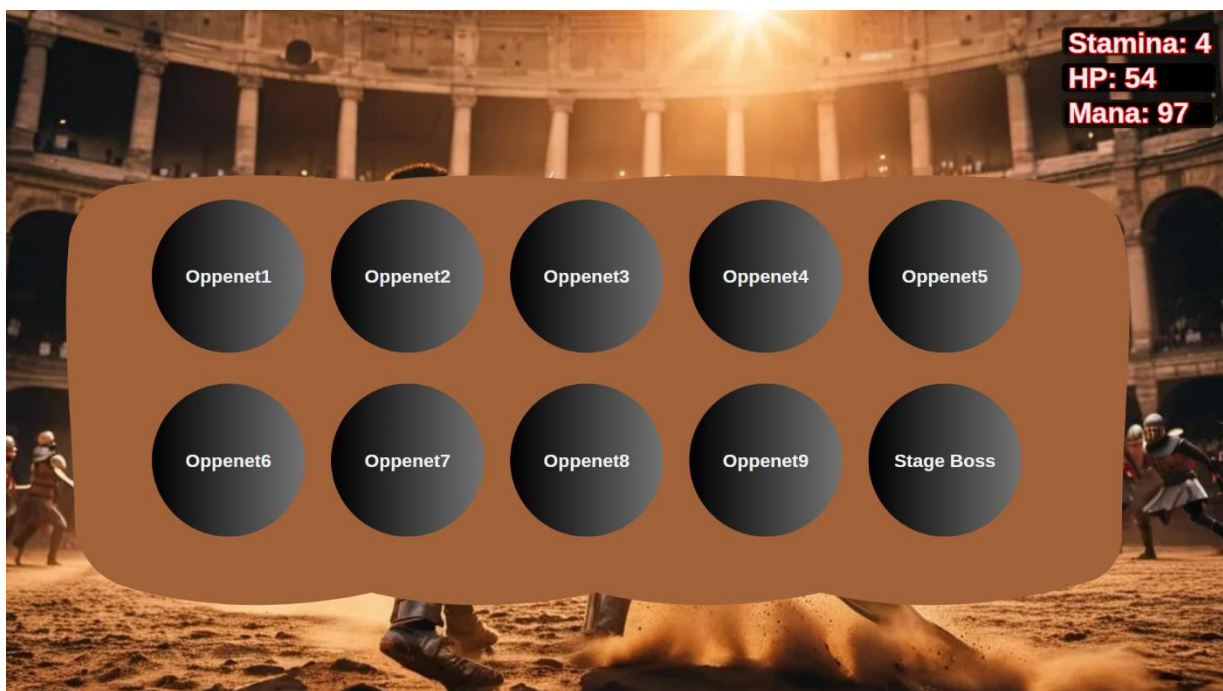
## 7.8 War Phase (Gathering)



## 7.9 Combat (Interrupted Gathering)



## 7.10 Coliseum





## 7.11 Combat (Coliseum)



## References

- [1] <https://www.ibm.com/topics/artificial-intelligence>
- [2] <https://www.ibm.com/topics/machine-learning>
- [3] <https://www.engati.com/blog/ai-for-gaming>
- [4] [https://medium.com/@DylanCa\\_/evolution-of-ai-in-video-games-from-1980-to-today-e3344acaed4c](https://medium.com/@DylanCa_/evolution-of-ai-in-video-games-from-1980-to-today-e3344acaed4c)
- [5] <https://en.wikipedia.org/wiki/Pac-Man>
- [6] [https://medium.com/@DylanCa\\_/evolution-of-ai-in-video-games-from-1980-to-today-e3344acaed4c](https://medium.com/@DylanCa_/evolution-of-ai-in-video-games-from-1980-to-today-e3344acaed4c)
- [7] <https://civilization.2k.com/civ/>
- [8] [https://en.wikipedia.org/wiki/StarCraft:\\_Brood\\_War](https://en.wikipedia.org/wiki/StarCraft:_Brood_War)
- [9] [https://store.steampowered.com/app/1174180/Red\\_Dead\\_Redemption\\_2/](https://store.steampowered.com/app/1174180/Red_Dead_Redemption_2/)
- [10] [https://store.steampowered.com/app/1240440/Halo\\_Infinite/](https://store.steampowered.com/app/1240440/Halo_Infinite/)
- [11] <https://www.taylorfrancis.com/books/mono/10.1201/9781315267845/game-engine-architecture-third-edition-jason-gregory>

- [12] <https://docs.unity.com/>
- [13] <https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-5-documentation>
- [14] <https://docs.godotengine.org/en/stable/index.html>
- [15] <https://www.cryengine.com/docs/static/engines/cryengine-5/categories/23756816>
- [16] <https://docs.cocos2d-x.org/cocos2d-x/v4/en/>
- [17] <https://manual.gamemaker.io/monthly/en/#t=Content.htm>
- [18] [https://agate.id/the-art-of-strategic-gameplay-turn-basedgames/#:~:text=Turn%2Dbased%20strategy%20\(TBS\),territory%2C%20resources%2C%20or%20objectives](https://agate.id/the-art-of-strategic-gameplay-turn-basedgames/#:~:text=Turn%2Dbased%20strategy%20(TBS),territory%2C%20resources%2C%20or%20objectives)
- [19] <https://steamdb.info/charts/?tagid=1741&sort=peak>
- [20] <https://steamdb.info/stats/releases/?tagid=1741>
- [21] <https://www.europeanstudios.com/encyclopedia/tactical-rpg-video-game-genre/>
- [22] <https://steamdb.info/charts/?tagid=21725>
- [23] <https://steamdb.info/stats/releases/?tagid=21725>
- [24] <https://en.wikipedia.org/wiki/Roguelike>
- [25] <https://steamdb.info/charts/?tagid=1716&sort=peak>
- [26] <https://steamdb.info/stats/releases/?tagid=1716>
- [27] <https://trends.builtwith.com/widgets/Unity>
- [28] <https://distantjob.com/blog/visual-studio-vs-visual-studio-code/>
- [29] <https://recepkarademir.blogspot.com/2022/07/godot-oyun-motoru-hakkinda.html>
- [30] <https://www.geeksforgeeks.org/component-based-architecture-system-design/>