FINAL REPORT

for

“Sanctified Retribution”

by

Metehan TÜTER

Elif Aysu KÜRŞAD

Murat BEDİZ

Erkan SANCAK

Supervisor: Fırat AKBA

Jury Members: Gökçe Nur YILMAZ, Eren ULU

Date: 06.06.2024

Links: Website GitHub

Table of Contents

1. Introduction

The project "Sanctified Retribution" embarked on a journey to create an immersive and captivating gaming experience with an interesting storyline. With a team dedicated to excellence, our aim was to develop a game that not only entertains but also challenges players, immersing them in a richly detailed world of adventure and excitement. Final report serves as a comprehensive documentation of our development process, highlighting the key aspects of our journey from conceptualization to implementation. This document also elaborates on the various stages of development, including the meticulous planning, rigorous testing, and iterative improvements that were integral to the project's success.

2. Requirement Details

2.1 Functional Requirements

In "Sanctified Retribution," functional requirements include:

2.1.1 **In Game Display:** Ensuring accessibility to the in-game menu and inventory, allowing players to interact with items and view relevant information such as health and progress. This includes seamless transitions between game states and an intuitive layout that enhances the user experience.

2.1.2 **Enemy AI:** Implementing intelligent enemy behavior that responds dynamically to player actions, engaging in combat and exhibiting diverse tactics. This includes adaptive AI that changes tactics based on player skill level and actions, providing a challenging and engaging experience.

2.1.3 **Game Mechanics:** Developing intuitive and responsive game mechanics for combat, movement, and interaction with objects. These mechanics are designed to be easy to learn but difficult to master, ensuring long-term engagement.

2.1.4 **Level Design:** Crafting well-balanced and challenging game levels that encourage exploration and offer diverse experiences. Levels are designed to be both visually appealing and strategically complex.

2.1.5 **Beta Testing:** Releasing the game for real-world testing for our testers and feedback.

2.2 Non-Functional Requirements

Non-functional requirements include for our project:

2.2.1 **Performance:** Ensuring smooth gameplay, minimal loading times, and optimal resource utilization. This involves extensive performance testing and optimization to provide a seamless experience.

2.2.2 **Scalability:** Designing the game architecture to accommodate future projects. Scalability ensures that the game can reach a large audience of players and additional features without compromising performance.

2.2.3 **Security:** Developing Implementing robust security measures to protect player data and prevent unauthorized access. Fortunately, “Sanctified Retribution" does not require online access to play, so it does not need encryption, secure login protocols, and regular security audits.

2.2.4 **Usability:** Creating an intuitive user interface and gameplay experience accessible to players of all skill levels. Usability testing is conducted to ensure the game is easy to navigate and play.

3. Final Architecture and Design Details

This The architecture of "Sanctified Retribution" is designed to facilitate client-side components. The game follows a client-server model, where the server manages game state, and data storage, while the client handles rendering, user input, and gameplay mechanics. This architecture ensures a responsive and engaging player experience while maintaining data integrity.

3.1 Server Side

The server-side architecture consists of a scalable infrastructure designed to handle player connections, game logic, and database management. This setup supports offline gameplay, ensuring that all game functions operate smoothly without an internet connection. Additionally, it guarantees secure storage of player data, maintaining data integrity and privacy while providing a seamless offline gaming experience.

3.2 Application Architecture & Design Choices

The application architecture of "Sanctified Retribution" is built on the Unity game engine, leveraging its flexibility and cross-platform capabilities. The game utilizes object-oriented design principles, modular components, and design patterns such as to ensure maintainability and extensibility. Unity's robust ecosystem and extensive support for various platforms make it an ideal choice for our game's development needs.

Figure 1: Gameplay

Figure 2: Platform Mechanics

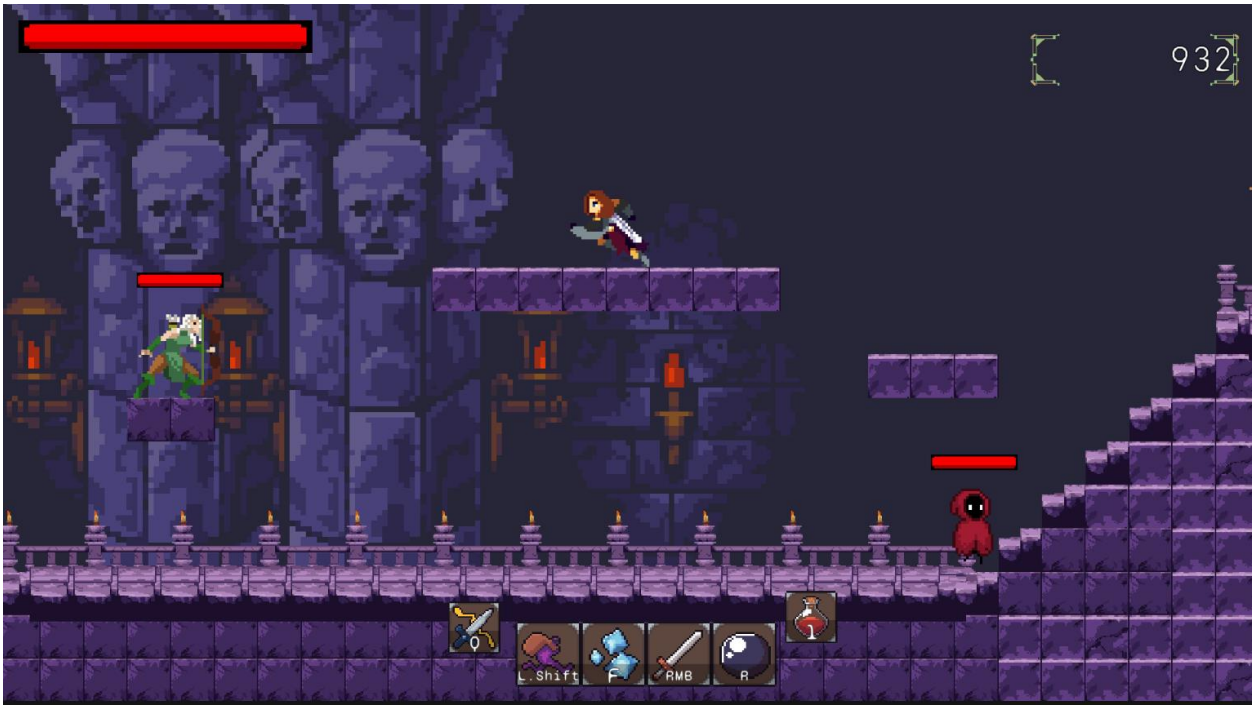

Figure 3: Different Levels of Game

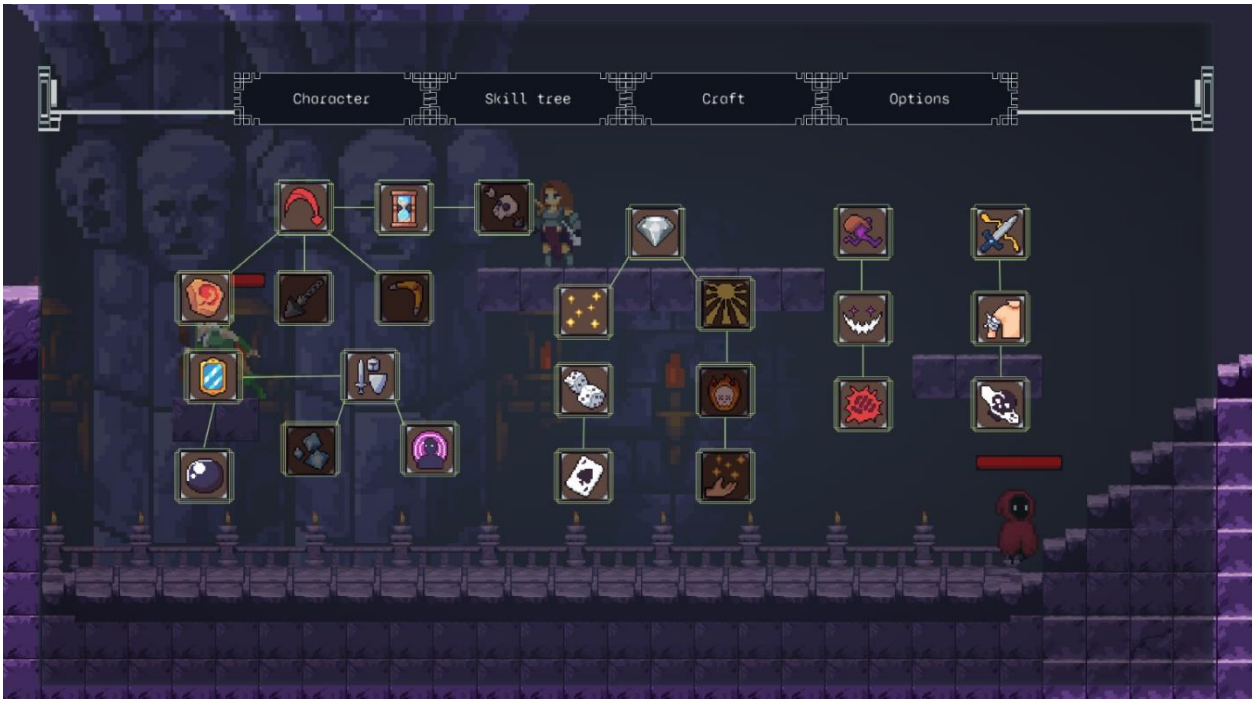

Figure 4: UI of game (Skill Tree in picture)

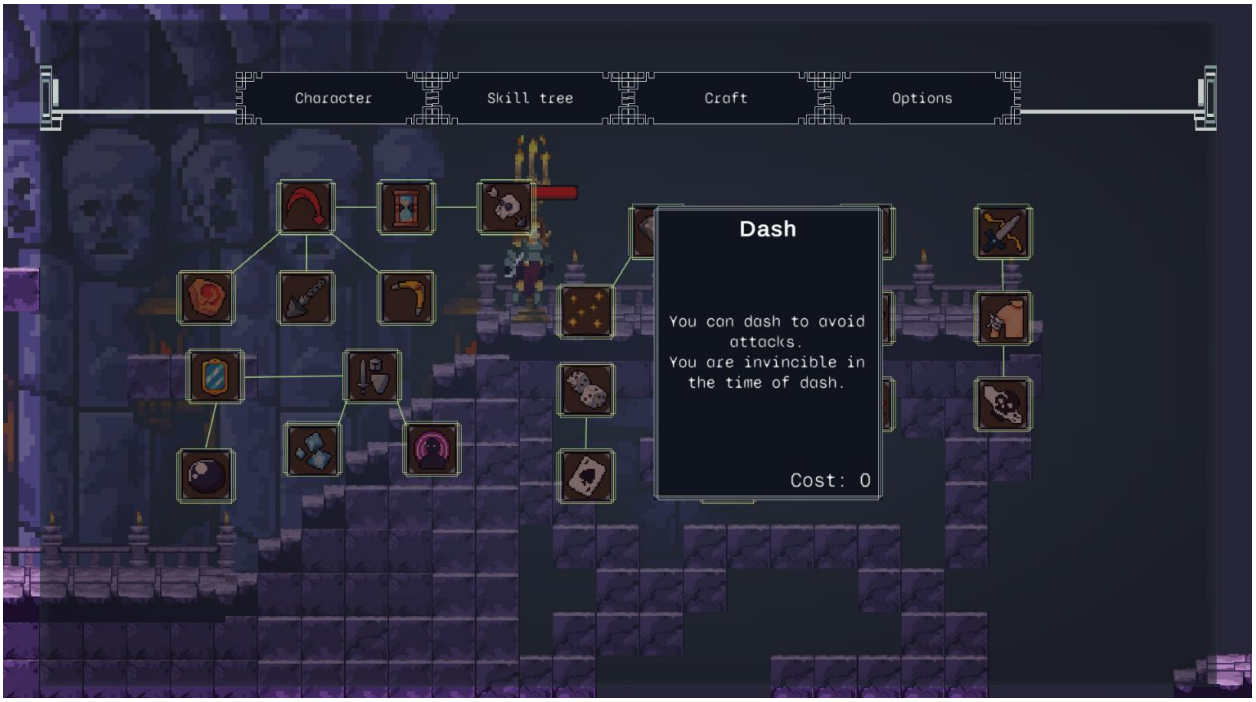

Figure 5: Guide for players (User Manuel)

4. Development and Implementation Details

4.1 Backend

Backend development involves creating server-side APIs, managing databases, and implementing security features. The backend is implemented using technologies such as Node.js, Express.js, and MongoDB, ensuring scalability, performance, and security. These technologies provide a strong foundation for handling complex game logic, user data, and real-time interactions.

4.2 Frontend

Frontend development focuses on creating an immersive user interface and engaging gameplay experience. The frontend is developed using Unity and C#, with an emphasis on responsive design, smooth animations, and intuitive controls. The use of Unity allows for high-quality graphics and responsive gameplay mechanics, enhancing the overall player experience.

5. Testing Details

- **Unit Testing:** During this phase, individual functions, modules, and components of the game will be tested to ensure they perform as expected.
- **System and Integration Testing:** Integration between various components and systems will be verified to ensure they work together seamlessly, and the entire system will be validated to meet specified requirements.
- **Performance Testing:** The performance of the game under various conditions, including load testing and stress testing, will be assessed to ensure optimal performance.
- **User Acceptance Testing (UAT):** Internal testers will validate the game's features and usability to ensure it meets their expectations and provides an enjoyable gaming experience.
- **Beta Testing:** The game will be released for real-world testing and feedback, allowing for further refinement, identifying any issues or bugs and optimization.

6. Maintenance Plan and Details

6.1 Limitations of the Server

Since the game does not need any type of connection with internet, there is no limits to it since it does not use any type of servers. However, in future updates, players may record their save data to the cloud servers.

6.2 Version Checking

Version control systems such as GitHub and Unity Version Control (Previously Plastic SCM) are used to manage code changes and ensure consistency across development environments. Continuous integration and deployment pipelines are employed to automate version checking and deployment processes. These practices may help maintain code quality and streamline the development process during future updates.

6.3 Machine Learning Concepts

Machine learning components may be integrated into the game to enhance AI behavior, generate dynamic content, or personalize player experiences. Techniques such as reinforcement learning, neural networks, and natural language processing can be explored to augment gameplay dynamics. These advanced technologies can provide more intelligent and adaptive game elements, enhancing player engagement.

7. Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

Engineering design considerations include factors such as performance, scalability, security, usability, and maintainability. These factors are carefully balanced to ensure the overall success of the project. Each consideration plays a crucial role in creating a robust and enjoyable game.

7.2 Ethics and Professional Responsibilities

Ethical considerations include data privacy, fair treatment of players, and responsible use of AI and machine learning technologies. Professional responsibilities involve transparency, honesty, and accountability in all aspects of game development. These principles ensure that the game is developed and maintained with integrity and respect for players. The game helps player's data keeping safe since it does not require any online access.

7.3 Teamwork Details

Team collaboration is facilitated through regular meetings, and clear communication channels. Tasks are assigned, tracked, and reviewed to ensure timely progress and effective coordination. Effective teamwork is essential for meeting project goals and delivering a high-quality product.

7.3 Project Plan Observed and Objectives Met

The project plan outlines key milestones, deliverables, and timelines, ensuring alignment with project objectives and expectations. Progress is monitored, and adjustments are made as needed to stay on track. This systematic approach helps ensure that the project meets its goals and delivers on its promises.

7.4 Extensibility

The game architecture is designed to be modular and extensible, allowing for easy integration of new features, content updates, and expansions. APIs and SDKs may be provided to encourage community-driven development and customization. Extensibility ensures that the game can grow and evolve over time.

7.5 Reliability

Reliability is ensured through testing, error handling, and fault-tolerant design. Monitoring and alerting systems are in place to detect and respond to issues proactively, minimizing downtime and disruptions. Reliable systems are crucial for maintaining player trust and ensuring a smooth gaming experience.

7.6 Usability

Usability testing involves gathering feedback from testers to identify points, improve user interfaces, and streamline gameplay mechanics. Iterative design and user-centered approaches are employed to optimize usability and enhance player satisfaction. User feedback is invaluable for making meaningful improvements.

7.7 Accessibility

Accessibility features such as customizable controls, text-to-speech support, and color-blind modes are implemented to ensure inclusivity and diverse player needs. Compliance with accessibility standards was important throughout development. With accessibility, our team tried to ensure that the game can be enjoyed by a wide range of players.

7.8 Portability

Portability refers to the ability of the game to run seamlessly across different platforms and devices. Right now, the game does not have any access to the consoles. Because of that, cross-platform development frameworks and responsive design techniques are wanted to utilized to maximize portability and reach a wider audience in the future.

7.9 Efficiency

Efficiency encompasses performance optimization, resource management, and energy efficiency. Techniques such as code profiling, caching, and asynchronous processing are employed to maximize computational efficiency and minimize resource usage. Efficient design helps provide a smooth and responsive gaming experience.

8. Conclusion and Future Work

8.1 Conclusion

In conclusion, the development of "Sanctified Retribution" has been a journey marked by dedication, creativity, and collaboration. Through meticulous planning, rigorous testing, and relentless iteration, we tried our best to create a game that offers players a truly immersive and enjoyable experience. This project demonstrates the power of teamwork, innovation, and a player-centric approach to game development.

8.2 Future Work

Looking ahead, there are several avenues for future work and improvement:

- **Expanding Content:** There may be some additions to new levels, characters, storylines, and gameplay mechanics to enrich the gaming experience in the future. Continuous content updates can keep the game fresh and engaging for players
- **Enhancing AI:** Further refining enemy AI behavior, implementing adaptive difficulty levels, and exploring advanced machine learning techniques. Enhanced AI can provide more challenging and dynamic gameplay.

- **Story Events:** Implementing the story of the game and expanding it
- **Map Display:** Implementing a comprehensive map feature for navigation, shading undiscovered areas, and providing useful information for exploration.
- **Accessibility Improvements:** Continuing to prioritize accessibility features and compliance with accessibility standards to ensure inclusivity. Accessibility improvements help make the game more enjoyable for all players.
- **Performance Optimization:** Fine-tuning performance, optimizing resource usage, and addressing any bottlenecks to deliver a smooth and responsive gaming experience. Ongoing optimization ensures that the game performs well under various conditions.

8.2 User Manual

Explanations to guide players through the game's features, controls, equipment, settings, and gameplay mechanics.

9. References and Glossary

9.1 References

1. **Unity Documentation:** The official documentation for the Unity game engine provides detailed information on features, APIs, and best practices for game development.
    - Website: Unity User Manual
2. **Unity Testing and Quality Assurance Tips:** This article provides valuable insights and tips for testing Unity projects, including best practices for quality assurance, testing methodologies, and tools available within the Unity ecosystem.
    - Website: Unity Testing and Quality Assurance Tips
3. **Unity Forum:** The Unity community forum is a valuable resource for developers, offering discussions, tutorials, and troubleshooting tips related to game development and testing.
    - Website: Unity Forums
4. **Unity Test Framework Manual:** The Unity Test Framework Manual provides comprehensive documentation on the Unity Test Framework, detailing its features, usage, and best practices for writing and executing automated tests within the Unity environment.
    - Website: Unity Test Framework Manual
5. **NUnit Documentation:** NUnit is a popular unit testing framework for .NET applications, including those developed with Unity. Its documentation offers guidance on writing and executing unit tests.
    - Website: NUnit Documentation
6. **GitHub Guides:** GitHub provides extensive guides and documentation on using its platform for version control, issue tracking, and collaboration, which can be useful for managing the testing process.
    - Website: GitHub Docs

7. **Stack Overflow:** Stack Overflow is a popular platform for asking and answering technical questions, including those related to Unity development and testing.
    - Website: Stack Overflow

9.1 Glossary

- **AI (Artificial Intelligence):** The simulation of human intelligence processes by machines, especially computer systems. In gaming, AI is used to create responsive, adaptive, and intelligent behaviors in non-player characters (NPCs).
- **API (Application Programming Interface):** A set of routines, protocols, and tools for building software and applications. APIs specify how software components should interact.
- **Backend:** The server side of an application. It includes all the operations behind the scenes that manage the application's functionality, such as database interactions, server logic, and user authentication.
- **Beta Testing:** The phase of software testing in which a sampling of the intended audience tries the product out. Beta testing is typically the last step before a product launch.
- **Client-Side:** Refers to operations that are performed by the client in a client-server relationship in a network. This typically includes the user's computer and the software that runs on it.
- **Design Patterns:** General reusable solutions to common problems in software design. They are templates designed to help write code that is easier to understand and maintain.
- **Extensibility:** The quality of a design that allows for future growth. Extensible software can accommodate new features and functions without significant changes to the architecture.
- **Frontend:** The part of the application that interacts directly with the users. It includes everything the users experience, such as the graphical user interface and command line.
- **Game Mechanics:** The rules and systems that dictate how a game operates and how the players interact with it. These mechanics are the foundation of the game's design and include elements such as combat systems, movement, and inventory management.
- **Hardware:** The physical components of a computer system, such as the CPU, memory, and storage devices.
- **Modular Components:** Components designed to function independently but can be connected to form a more complex system. This approach makes it easier to manage, update, and scale software.
- **Non-Functional Requirements:** Requirements that define how a system is supposed to be. They encompass all the qualities the system should have, such as performance, usability, reliability, etc.
- **Object-Oriented Design:** A programming paradigm based on the concept of "objects," which can contain data and code to manipulate the data. It emphasizes reusable and modular code.
- **Performance Testing:** Testing conducted to evaluate the compliance of a system or component with specified performance requirements. It involves testing software applications to ensure they will perform well under their expected workload.
- **Scalability:** The capability of a system to handle a growing amount of work or its potential to accommodate growth.

- **System and Integration Testing:** Testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing.
- **Unity:** A cross-platform game engine used to develop video games for PC, consoles, mobile devices, and websites. It's known for its flexibility and robust toolset.
- **Usability:** The measure of the quality of a user's experience when interacting with a product or system. It involves ease of use, efficiency, and satisfaction.
- **User Acceptance Testing (UAT):** A phase of software development in which the software is tested in the "real world" by the intended audience to ensure it can handle required tasks in real-world scenarios according to specifications.
- **Version Control:** A system that records changes to a file or set of files over time so that specific versions can be recalled later. Examples include Git and Unity Version Control (previously Plastic SCM).