# TEST PLAN REPORT

# for

# “Sanctified Retribution”

# by


# Metehan TÜTER

# Elif Aysu KÜRŞAD

# Murat BEDİZ

# Erkan SANCAK


# Date: 05.05.2024

# Table of Contents

# 1.Introduction

The testing process for "Sanctified Retribution" is a crucial component of the overall development lifecycle, ensuring that the final product meets the highest standards of quality, reliability, and functionality. As the project progresses towards its completion, it becomes increasingly important to establish a comprehensive test plan that outlines the scope, approach, resources, and schedule of testing activities.

## 1.1 Purpose

This test plan serves as a roadmap for the testing phase, providing a clear understanding of the objectives, methodologies, and responsibilities involved in testing the game. By systematically identifying the items to be tested, the features to be evaluated, and the testing tasks to be performed, this plan aims to streamline the testing process and facilitate effective collaboration among team members.

## 1.2 Design

The goal of the test plan is twofold: to ensure that "Sanctified Retribution" meets the specified requirements and to identify and mitigate any risks associated with the testing process. By adhering to best practices and industry standards, we aim to deliver a high-quality product that exceeds the expectations of our target audience.

Throughout this test plan, we will detail the various testing methodologies to be employed, including unit testing, integration testing, system testing, performance testing, user acceptance testing, and beta testing. Additionally, we will outline the test environment, schedule, control procedures, roles and responsibilities, and potential risks associated with the testing process.

## 1.3 Document Overview and Structure

Throughout this test plan, we will detail the various testing methodologies to be employed, including unit testing, integration testing, system testing, performance testing, user acceptance testing, and beta testing. Additionally, we will outline the test environment, schedule, control procedures, roles and responsibilities, and potential risks associated with the testing process.

The remainder of this document is structured to provide a comprehensive overview of the testing plan for "Sanctified Retribution." It is divided into several sections, each focusing on specific aspects of the testing process. These sections include:

2. **Scope and Purpose:** Describes the scope of testing activities and outlines the purpose of the test plan.
3. **Features to be Tested:** Identifies the key features of the game that will be subjected to testing.
4. **Testing Methodology and Test Environment:** Outlines the various testing methodologies to be employed, including unit testing, system and integration testing,

performance testing, user acceptance testing, and beta testing. Details the test environment setup, including hardware, software, and any other necessary tools or resources.
5. **Test Schedule:** Provides a timeline for the testing activities.
6. **Control Procedures:** Describes the procedures for controlling and managing the testing process, including version control, issue tracking, and communication protocols.
7. **Roles and Responsibilities:** Assigns roles and responsibilities to team members involved in the testing process.
8. **Risks:** Identifies potential risks associated with the testing plan and outlines mitigation strategies.
9. **Reference and Glossary:** Provides a list of external sources consulted or cited in the report (References) and defines key terms or concepts used in the document (Glossary) to aid reader understanding.

# 2. Scope and Purpose

## 2.1 Scope

The testing scope for "Sanctified Retribution" encompasses a thorough examination of the game's functionality, performance, usability, security, and compatibility. This includes testing various aspects of the game, from menu navigation to in-game mechanics, to ensure a polished and enjoyable player experience. The scope also extends to verifying adherence to design constraints, performance requirements, and overall project specifications. It includes but is not limited to:

- **Unit Testing:** Evaluation of individual functions, modules, and components to ensure they perform as expected.
- **Integration Testing:** Verification of the integration between various components and systems to ensure they work together seamlessly.
- **System Testing:** Validation of the entire system to ensure it meets the specified requirements and functions correctly.
- **Performance Testing:** Assessment of the game's performance under various conditions, including load testing and stress testing.
- **User Acceptance Testing (UAT):** Validation of the game's features and usability by internal testers representing the target audience to ensure it meets their expectations.

## 2.2 Purpose

The purpose of the testing activities outlined in this test plan is to ensure the quality, reliability, and functionality of "Sanctified Retribution" throughout its development lifecycle. The objectives include:

- **Identify Defects:** Detecting and addressing any defects, bugs, or issues within the game to ensure a smooth and enjoyable player experience.

- **Verify Functionality:** Ensuring that all features and mechanics function as intended and meet the specified requirements.
- **Evaluate Performance:** Assessing the game's performance to ensure it meets performance targets and provides a seamless gaming experience.
- **Validate Usability:** Gathering results from testing to evaluate the game's usability and user interface.
- **Enhance Quality:** Improving the overall quality of the game by addressing any identified issues, implementing enhancements, and optimizing performance.

# 3. Features to be Tested

This section identifies the key features of "Sanctified Retribution" that will be subjected to testing. Testing these features ensures that they function as intended and meet the specified requirements. The features to be tested include:

- **Menu Display:**
  - Ensuring that the menu display is user-friendly and easy to navigate.
  - Testing functionality of options such as "New Game," "Load Game," "Options," and "Exit."
  - Verifying that instructions and tooltips are provided where necessary to guide the player.
- **In-Game Display:**
  - Testing the functionality of accessing the in-game menu and inventory.
  - Verifying that players can access and use items found within the game.
  - Ensuring that the in-game display provides relevant information such as player health, available items, and progress.
- **Map Display:**
  - Testing the functionality of accessing the map in-game.
  - Verifying that players can view available parts of the map and that undiscovered areas are appropriately shaded.
  - Ensuring that the map provides useful information for navigation and exploration.
- **Enemy AI:**
  - Testing the behavior of enemy AI to ensure it responds appropriately to player actions.
  - Verifying that enemies react to the player's presence and engage in combat when necessary.
  - Ensuring that different types of enemies exhibit varied behaviors and tactics.
- **Game Mechanics:**
  - Testing various game mechanics such as combat, movement, and interaction with objects.
  - Verifying that game mechanics are intuitive and responsive to player input.
  - Ensuring that mechanics contribute to an engaging and challenging gameplay experience.
- **Story Events:**
  - Testing scripted events and dialogues related to the game's storyline.

    - Verifying that story events trigger correctly and progress the narrative as intended.
    - Ensuring that dialogue options have the expected outcomes.
- **Level Design:**
    - Testing the layout and structure of game levels to ensure they are well-designed and balanced.
    - Verifing that levels provide a variety of challenges and opportunities for exploration.
    - Ensuring that level design supports the overall gameplay experience.
- **Beta Testing:**
    - Releasing of the game for real-world testing and feedback.

# 4. Testing Methodology and Test Environment

This section outlines the methodologies to be employed for testing "Sanctified Retribution" and describes the test environment setup.

## 4.1 Testing Methodology

- **Unit Testing:** Individual functions, modules, and components will be isolated and subjected to testing to ensure they perform as expected. Automated unit tests will be developed using frameworks such as UTF and/or NUnit to verify the correctness of code at the smallest level.
- **Integration Testing:** The integration between various components and systems will be tested to ensure they work together seamlessly. Test cases will be designed to validate data flow, communication, and interactions between different modules.
- **System Testing:** The entire system will undergo comprehensive testing to validate its compliance with specified requirements and its overall functionality. Test scenarios will cover various user interactions, edge cases, and system-wide behaviors to find any defects or inconsistencies.
- **Performance Testing:** The performance of game will be evaluated under different conditions, including load testing to assess its scalability and stress testing to determine its stability under different conditions. Metrics such as response times, throughput, and resource utilization will be monitored to identify potential bottlenecks or performance issues.
- **User Acceptance Testing (UAT):** Sessions to evaluate the game's features, usability, and overall user experience. Results gathered during UAT will be used to validate that the game meets user expectations and to identify areas for improvement

## 4.2 Test Environment

### 4.2.1 Hardware:

- **Specification:**
    - Processor: Dual-core 2.0 GHz or higher
    - Memory: 2 GB RAM or higher.

    - Graphics Card: Integrated graphics card or dedicated graphics card with at least 512 MB VRAM
    - DirectX: Version 9.0c or higher.
- **Devices:**
    - Keyboard and Mouse HID

### 4.2.2 Software:

- **Operating System:**
    - Windows 10 or higher.
- **Development Tools:**
    - Integrated Development Environment (IDE): Unity Editor
    - Scripting Language: C#
- **Testing Tools:**
    - Unity Test Framework and NUnit: Unit testing framework for .NET applications, utilized for writing automated tests within the Unity environment.

### 4.2.3 Network Environment:

- **Internet Connectivity:**
    - No internet connection required since "Sanctified Retribution" is based of offline single-player game.

### 4.2.4 Other Resources:

- **Documentation:**
    - Unity Test Framework Manual
    - NUnit Documentation

# 5 . Test Schedule

- **Unit Testing:** Week 1
    - During this phase, individual functions, modules, and components of the game will be tested to ensure they perform as expected.
- **System and Integration Testing:** Weeks 1-2
    - 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:** Weeks 2-3
    - 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):** Weeks 3-4
    - Internal testers will validate the game's features and usability to ensure it meets their expectations and provides an enjoyable gaming experience.
- **Beta Testing:** Weeks 4-5

- The game will be released for real-world testing and feedback, allowing for further refinement, identifying any issues or bugs and optimization.

# 6. Control Procedures

The following control procedures are implemented for the testing of "Sanctified Retribution"

- **Version Control and Issue Tracking:** The game's source code and assets will be managed using a system that has a version control system capabilities, such as GitHub. GitHub offers features such as issue tracking, wikis, and project management tools in addition to its version control capabilities. GitHub Project
- **Communication Protocols:** Regular team meetings are scheduled to discuss testing progress, share updates, and address any challenges or concerns. Additionally, communication channels, such as Zoom or Discord, utilized for real-time communication and collaboration among team members.
- **Quality Criterias:** Quality criterias are established at key milestones in the testing process to assess the readiness of the game for further development or release.
- **Change Management:** Any changes to the game's requirements, design, or functionality identified during testing are documented, evaluated, and managed. This process will ensure that changes are properly reviewed, approved, and implemented to maintain project integrity and alignment real-use expectations.

# 7. Roles and Responsibilities

In this section, we define the roles and responsibilities of our team. Each role is crucial for ensuring the thoroughness and effectiveness of the testing activities. The following are the key roles and their corresponding responsibilities:

- **Project Manager / Lead Tester:** Responsible for overseeing the testing process and coordinating activities.
- **Testers:** Responsible for executing test cases and reporting defects.
- **Developers:** Responsible for addressing defects and collaborating with testers.
- **Documentation Specialist:** Responsible for maintaining testing documentation.

# 8. Risks

Identifying potential risks and developing strategies to mitigate them is crucial for the successful execution of the testing plan for "Sanctified Retribution." Below are some of the risks associated with the testing process along with their mitigation strategies.

1. **Resource Constraints:**
    - **Risk:** Limited availability of testing resources such as time, personnel, and equipment may impede the testing process.

    - **Mitigation:** Prioritizing testing activities based on the importance and allocate resources efficiently. Utilize automation tools to streamline testing tasks and maximize resource utilization.
2. **Technical Challenges:**
    - **Risk:** Technical issues or limitations may arise during testing, hindering the identification of defects and affecting the overall quality of the game.
    - **Mitigation:** Implementing robust debugging and troubleshooting procedures to resolve issues promptly.
3. **Scope Creep:**
    - **Risk:** Expansion of the project scope beyond the initial requirements may lead to increased testing complexity and resource requirements.
    - **Mitigation:** Regularly reviewing and prioritizing testing objectives to ensure alignment with project goals. Communicating changes in scope effectively in team to manage expectations.
4. **Communication Breakdown:**
    - **Risk:** Inadequate communication among team members may result in misunderstandings, delays, and errors.
    - **Mitigation:** Establishing clear channels of communication and protocols for reporting progress, issues, and updates. Conducting regular meetings, status updates, and reviews to ensure transparency among all members.
5. **External Dependencies:**
    - **Risk:** Dependencies on external factors such as third-party libraries, APIs, or services may introduce vulnerabilities, compatibility issues, or performance bottlenecks.
    - **Mitigation:** Identifying and documenting all external dependencies and assess their impact on testing activities.

# 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

1. **Unit Testing**: Testing individual functions, modules, or components of the software to ensure they perform as expected in isolation.
2. **Integration Testing**: Testing the integration between various components and systems to ensure they work together seamlessly.
3. **System Testing**: Comprehensive testing of the entire system to validate its compliance with specified requirements and its overall functionality.
4. **Performance Testing**: Evaluation of the software's performance under different conditions, including load testing and stress testing.
5. **User Acceptance Testing (UAT)**: Sessions conducted to evaluate the software's features, usability, and overall user experience by internal testers representing the target audience.
6. **Beta Testing**: Real-world testing and feedback phase where the software is released to users for further refinement and identification of issues or bugs.
7. **Test Environment**: The hardware, software, and network setup used for testing purposes.
8. **Version Control**: A system that manages changes to documents, computer programs, large web sites, or other collections of information.
9. **Issue Tracking**: The process of documenting, managing, and resolving reported issues or bugs during software development.
10. **Communication Protocols**: Established rules and conventions for exchanging information between team members, ensuring effective collaboration and coordination.
11. **Change Management**: The process of managing changes to software, including documenting, evaluating, and implementing changes to maintain project integrity.
12. **Roles and Responsibilities**: Defined roles within the team and their corresponding responsibilities for ensuring the effectiveness of testing activities.
13. **Risks**: Potential threats or challenges that may affect the testing process, along with strategies to mitigate them.