Testing Strategy

1. Introduction

This document delineates the testing strategy for the CSGOEmpire roulette feature, merging the allure of Counter-Strike: Global Offensive (CS:GO) with traditional roulette gambling. It enables users to wager with CS:GO skins, coins, or other virtual assets, offering a distinctive gambling experience within the CS:GO ecosystem.

2. Testing Objectives

The primary testing objectives for the CSGOEmpire roulette feature are:

Functionality: Ensuring all betting processes, from selecting a bet to payout transactions, are operational as designed.
Performance: Maintaining responsiveness and stability under varied loads to ensure real-time updates for all users.
User Interface: Providing an engaging and intuitive interface that offers a seamless experience across devices and platforms.
Security: Securing the platform against unauthorized access to protect user data and financial transactions.

3. Areas of Testing

Key areas for testing include:

Betting Mechanics: Validation of bet placement, outcome determination, and payout calculation.
Spin Result Animation: Ensuring accurate display of visual and sound effects for the roulette spin and its outcomes.
User Interaction: Assessing ease of use, including bet amount selection and social feature interactions.
Compatibility: Ensuring functionality across major web browsers and devices, including various screen sizes and resolutions.
Performance: Testing application responsiveness during peak usage and under heavy user load.
Security: Identifying and mitigating vulnerabilities, such as SQL injection and XSS attacks, and verifying robust authentication and data encryption.

4. Test Strategy

1. Functional Testing: Test cases include placing bets, verifying payouts, and interacting with social features.
2. Compatibility Testing: Cross-browser and device/screen size compatibility tests on both desktop and mobile platforms.
3. Performance Testing: Load testing with multiple concurrent users and stress testing to identify the system's breaking point.
4. Security Testing: Penetration tests for SQL injection, XSS attacks, and brute force login attempts.

5. Test Environment

A staged environment closely replicating production will be used, including similar hardware, network configurations, and a variety of test accounts for betting.

6. Tools and Resources

Automation Tools: Selenium for functional and compatibility testing, JMeter for performance testing.
Security Testing Tools: OWASP ZAP and Burp Suite.
Project Management: JIRA for tracking test cases, defects, and progress.

7. Test Schedule

Testing activities will align with the software development lifecycle, with milestones for functional, compatibility, performance, and security testing.

8. Risk Management

Key risks include performance bottlenecks, security vulnerabilities, and compatibility issues, with contingency plans involving optimization, immediate vulnerability resolution, and fallback UI/UX solutions for older browsers or devices.