Cool Cyber Games – Interactive Platform for Teaching Cybersecurity

Test Plan Document

Matthew Goembel, Anthony Clayton, Ludendorf Brice, Ben Allerton
Team Members

Sneha Sudhakaran
Faculty Advisor

College of Science and Engineering
Florida Institute of Technology
Melbourne, United States

February 2025

Index

- 4.5.3 Test Case: Game Data Storage
- 4.5.4 Test Case: Database Backup and Recovery
- 4.5.5 Test Case: Database Performance

4.6 Server Testing (Hosted on Render)

- 4.6.1 Test Case: Server Uptime
- 4.6.2 Test Case: API Endpoint Availability
- 4.6.3 Test Case: Server Response Time
- 4.6.4 Test Case: Server Scalability
- 4.6.5 Test Case: Server Security

5. Conclusion

1. Introduction

1.1 Project Overview

Project Name: Cool Cyber Games – Interactive Platform for Teaching Cybersecurity
Team Members: Matthew Goembel, Anthony Clayton, Ludendorf Brice, Ben Allerton
Faculty Advisor: Sneha Sudhakaran
Client: Sneha Sudhakaran, College of Engineering and Science

1.2 Purpose

The purpose of this document is to outline the testing strategy for Cool Cyber Games to ensure that all functionalities, user interactions, and security features work as intended. The plan defines test cases, expected outcomes, and verification methods to validate the system.

1.3 Scope

This test plan covers:

- Functional testing of core features
- Performance testing under different loads
- Security testing to ensure data protection
- Usability testing for user experience
- Compatibility testing across devices and browsers

1.4 References

- IEEE Std 829-1998, Standard for Software Test Documentation
- Project Plan
- System Requirements Specification

---

2. Test Items

The following items will be tested:

- User Authentication (Login, Logout, Validation)
- Interactive Tutorials and Quizzes
- Gamification and Cybersecurity Challenges (Phishing Simulation, Malware Defense Game)

- Progress Tracking and Certifications
- Performance under heavy load
- Security features (SQL Injection, XSS Protection)
- Cross-browser compatibility

---

3. Definitions

This section defines key terms used in the test plan to ensure clarity and consistency across testing activities.

- **Test Case:** A specific set of conditions under which a tester determines whether a system or software application is working correctly.
- **OAuth 2.0:** An industry-standard protocol for authentication, enabling secure login without exposing user credentials.
- **MongoDB:** A NoSQL database used to store user and game data. Two separate databases are maintained: **User DB** for user-related information and **Game DB** for game-related data.
- **Godot:** An open-source game engine used for developing the gamification and cybersecurity challenges in Cool Cyber Games.
- **XSS (Cross-Site Scripting):** A security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users.
- **SQL Injection:** A code injection technique used to attack data-driven applications by inserting malicious SQL statements into a database query.
- **Render Hosting:** A cloud-based service used to deploy and host the Cool Cyber Games application.

---

4. Test Cases

4.1 User Authentication (OAuth 2.0 with Google)

4.1.1 Test Case: Google OAuth 2.0 Login

- **Description:** Verify that users can log in to the platform using their Google account via OAuth 2.0.
- **Expected Result:** The user is redirected to the platform's dashboard after successful authentication. The user's profile information (name, email, avatar) is retrieved and stored in the **User DB**.

4.1.2 Test Case: Login with Invalid Google Account

- **Description:** Validate system response when logging in with an invalid or unauthorized Google account.
- **Expected Result:** The system displays an error message and denies access.

4.1.3 Test Case: Revoke OAuth Access

- **Description:** Ensure that users can revoke access to the platform from their Google account settings.
- **Expected Result:** The user is prompted to re-authenticate and grant permissions again.

4.1.4 Test Case: Token Expiry and Refresh

- **Description:** Validate that the system handles OAuth 2.0 token expiry and refresh correctly.
- **Expected Result:** The system automatically refreshes the access token using the refresh token.

4.1.5 Test Case: Multiple Google Accounts

- **Description:** Verify that users can switch between multiple Google accounts.
- **Expected Result:** The system logs in the user with the newly selected Google account.

4.1.6 Test Case: Logout Functionality

- **Description:** Ensure that users can log out and terminate their session.
- **Expected Result:** The user is logged out, and the session is terminated.

4.2 Interactive Tutorials and Quizzes

4.2.1 Test Case: Access Tutorials

- **Description:** Verify users can open and complete tutorials.
- **Expected Result:** Tutorial content is displayed, and progress is tracked in the **User DB**.

4.2.2 Test Case: Quiz Functionality

- **Description:** Ensure quizzes function correctly.
- **Expected Result:** Score is displayed with correct/incorrect answers, and results are stored in the **User DB**.

4.3 Gamification and Cybersecurity Challenges

4.3.1 Test Case: Phishing Simulation

- **Description:** Verify users can engage in phishing attack simulations developed using **Godot**.
- **Expected Result:** Correct responses increase score; incorrect responses provide feedback. Game data is stored in the **Game DB**.

4.3.2 Test Case: Malware Defense Game

- **Description:** Ensure malware defense game, developed using **Godot**, functions properly.

- **Expected Result:** Game mechanics work, and progress is saved in the **Game DB**.

4.4 Progress Tracking and Certifications

4.4.1 Test Case: Progress Tracking

- **Description:** Verify users' progress is saved and displayed correctly.
- **Expected Result:** Progress is updated correctly in the **User DB**.

4.4.2 Test Case: Certificate Generation

- **Description:** Ensure users receive certificates upon completion.
- **Expected Result:** A downloadable certificate is generated, and the record is stored in the **User DB**.

4.5 Database Testing (MongoDB)

4.5.1 Test Case: Database Connection

- **Description:** Verify application connects to the MongoDB databases (**User DB** and **Game DB**).
- **Expected Result:** No connection errors are logged.

4.5.2 Test Case: User Data Storage

- **Description:** Ensure user data is correctly stored in the **User DB**.
- **Expected Result:** User data is accurately stored and retrievable.

4.5.3 Test Case: Game Data Storage

- **Description:** Verify game-related data storage in the **Game DB**.
- **Expected Result:** Game data is stored accurately and retrievable.

4.5.4 Test Case: Database Backup and Recovery

- **Description:** Validate that database backups can be restored in case of failure for both **User DB** and **Game DB**.
- **Expected Result:** No data loss or corruption is observed.

4.5.5 Test Case: Database Performance

- **Description:** Ensure that the databases handle high read/write operations efficiently.
- **Expected Result:** Response times remain under 500ms for both **User DB** and **Game DB**.

4.6 Server Testing (Hosted on Render)

4.6.1 Test Case: Server Uptime

- **Description:** Verify that the server hosted on Render is running.
- **Expected Result:** The website is accessible.

4.6.2 Test Case: API Endpoint Availability

- **Description:** Ensure API endpoints function correctly.
- **Expected Result:** All API endpoints return expected responses.

4.6.3 Test Case: Server Response Time

- **Description:** Validate server response within acceptable time limits.
- **Expected Result:** Server responds within 2 seconds.

4.6.4 Test Case: Server Scalability

- **Description:** Verify that the server can scale under increased traffic.
- **Expected Result:** No downtime or performance degradation.

4.6.5 Test Case: Server Security

- **Description:** Ensure that the server is secure.
- **Expected Result:** No critical security vulnerabilities found.

---

5. Conclusion

This test plan ensures Cool Cyber Games meets its functional, performance, security, and compatibility requirements. Comprehensive testing will identify and address potential issues before deployment.