Test Plan for Software Testing Portfolio:

1. Introduction:

This test plan outlines the testing strategies and objectives for the multiplayer game project used for the software testing portfolio coursework. The approach includes both validation and verification processes to ensure functionality and performance.

2. Testing Objectives:

The goal of this test plan is to:

- Validate functional and non-functional requirements.
- Identify defects early in the development lifecycle.
- Ensure smooth gameplay by testing key subsystems, including player interactions and game state management.

3. Validation Testing Strategy:

3.1. Functional testing:

3.1.1. Game Initialization Subsystem:

- **Objective:** Ensure proper loading of game assets and environment setup.

- **Tests:**
    - Verify game assets load correctly.
    - Check proper initialization of game states and variables.

3.1.2. Player Connection and Disconnection Subsystem:

- **Objective:** Ensure accurate and efficient connection as well as a graceful disconnection.

- **Tests:**
    - Validate WebSocket connections for multiple users.
    - Validate adequate disconnection from server.

3.1.3. Gameplay Mechanics Subsystem:

- **Objective:** Validate core gameplay features such as movement, interactions and scoring.

- **Tests:**

    - Validate game state updates based on player inputs.
    - Ensure consistency across connected clients.
    - Perform boundary tests for player movement and interactions.

4. Non-Functional Testing:

4.1. Performance Testing:

- **Objective:** Evaluate the system response under high user load.

- **Tests:**

    - Measure latency under concurrent player actions.
    - Ensuring standard function across devices and browsers.

4.2. Security Testing:

- **Objective:** Ensure data security and Integrity.

- **Tests:**

    - Validate secure data transmission using HTTPS.
    - Implement authentication verification tests.

4.3. Usability Testing:

- **Objective:** Improve the player experience by ensuring intuitive control.

- **Tests:**

    - Measure the ease of use for key game interactions.

5. Verification Approach:

I will use code reviews ensuring consistent formatting, adherence to coding standards and thorough documentation. Automated testing will be implemented for unit, integration and system tests using tools like Jest and Puppeteer.

6. Testing Tools and Instrumentation:

- **Testing frameworks:** Jest for unit testing (jsdom for specific cases) and Puppeteer for system testing in the browser.
- **CI/CD Integration:** GitHub Actions will be used to create automatic testing workflows.

- **Coverage analysis:** Codecov will be used in alignment with the CI/CD integration to provide automatic coverage analysis and logging of said coverage analysis.
- **Load testing tools:** Artillery will be used to simulate a large amount of concurrent users.

7. Test Checklist:

Test Name	**Expected Time**	**Status**
Unit tests	2 hours	Completed
Integration tests	5 hours	75%
System tests	5 hours	90%
Security Tests	4 hours	Incomplete
Load stress tests	5 hours	Incomplete

8. Evaluation Criteria:

- **Code Coverage:** 70%+ statement and branch coverage.
- **Performance:** Evaluating response time under specific loads.
- **Security Compliance:** Ensuring no critical vulnerabilities remain unaddressed.