Test Summary Report

Xavier Beatrice, Dale Morris, Louis Wiley

Purpose

This document explains the various activities and functions performed as part of the testing for the Risk Bot website and Game application.

Application Overview

The Risk Bot Website and accompanying Risk Game are a set of dotnet applications that allow students and programmers to create risk 'bots'—programs that can act as autonomous units within the Risk Game. The website will distribute the Risk game and allow people to distribute their Risk bots by uploading and downloading to and from the website.

Testing Scope

a) **In Scope**

- i) Internal queries, connections, and software
- ii) Resource Usage
- iii) API, Security, and UX

b) **Out of Scope**

- i) Outside libraries, testing frameworks, and build programs
  - 1) Docker, PostgreSQL, etc
- ii) Hardware related defects or issues
- iii) Operating system bugs
- iv) Non-intended API usage

Metrics

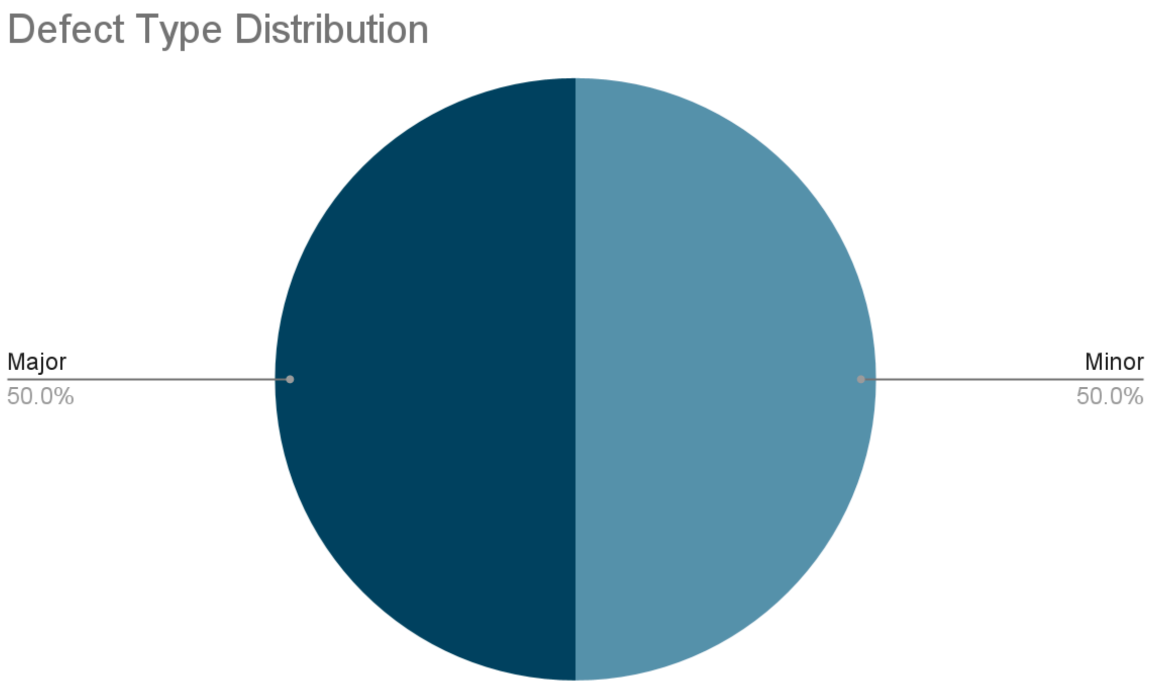
a) **No. of test cases planned vs executed**

b) **No. of test cases passed/failed**

Test Cases Planned	Test Cases Executed	Test Cases Passed	Test Cases Failed
13	**11**	**3**	**8**

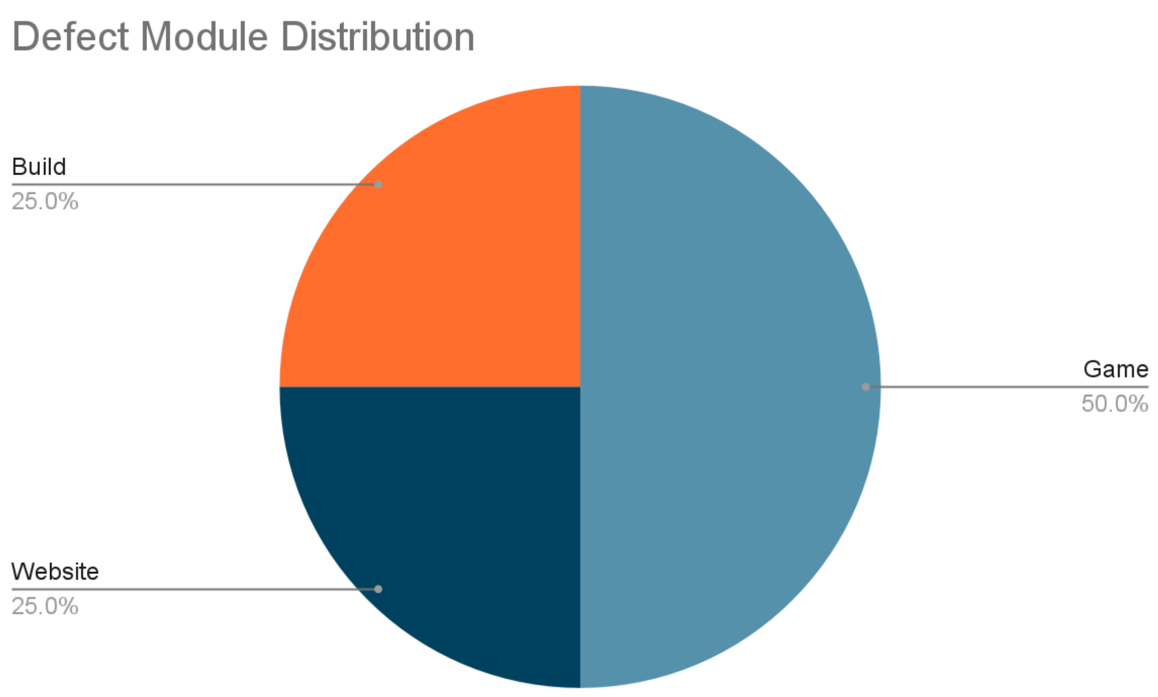
c) No. of defects identified and their Status & Severity

	Minor	Major
Closed	**0**	**0**
Open	**4**	**4**



d) Defects distribution - module wise

Game	Website	Build
4	**2**	**2**



Types of testing performed

a) Build Related

  i) For testing revolving around our build systems that are used to compile our source code.

b) Web Related

  i) For the website part of the project. Related to all aspects of the website from usability to buttons.

c) Game Related

i) For the game part of the project.

Test Environment & Tools

The system used for testing involved an 8-core AMD zen 4 CPU with 64 GB of RAM. Additionally, .NET 8 and .NET 9 were used on an Ubuntu-based Linux operating system.

Lessons Learnt

S. No.	Issues Faced	Solution
1.	All testing has to be done manually by testers.	Integrate more unit-testing within the
2.	Different .NET versions used by website and game services.	Move non-aligned services to the LTS version 8.0
3.	Bug fixes in development weren't ready in-time for testing.	Streamline PR time and reduce it so in-development work is ready and testable faster.

Recommendations

- Align back-end libraries and versioning so that programs can run easily on multiple machines and ship well in the final product.
- Automate testing processes further so human test time can be used on more intricate testing.

Best Practices

- The creation of testing templates has been instrumental in the quick but well-organized creation and oversight of test cases and testing.

- Testing should be done from a clean-slate to ensure that additional libraries weren't forgotten or to reduce occurrences of "it worked on my machine".

Exit Criteria

a) All test cases should be executed – **Yes**

b) All defects in Critical, Major, Medium severity should be verified and

the appropriate individual informed – **Yes.**

c) Any open defects in trivial severity should be placed upon the kanban board – **Yes.**

Conclusion/Sign Off

While all Exit Criteria have been met, the testing team cannot green-light the actual production use of the software until all major and minor flaws have been resolved and adequate functionality returned to the program. They will be reconsidered after the next revision of the software is available for testing.

Definitions, Acronyms, and Abbreviations

- **API**: Application Programming Interface is a set of rules that allows bots created by users to interact with the game program. It acts as a protocol for communication.
- **LTS**: Long-term support its often used to describe mature software versions that will be supported longer than normal versions.
- **.NET**: An open-source computer software framework developed by Microsoft that acts as a runtime for multiple programming languages.
- **Kanban**: A board that often has different areas for pinning activities or items of a project so they can be put onto a timeline and looked at by a project manager, usually to help keep track of what's done and not done along with what stage of the development process things are in.
- **PR**: Pull Request, often used to refer to separate branches of a program that have to be *pulled* into the main branch before they're released. It acts as a final check-in stage to ensure that all of the changes are adequate.

- **UX**: User experience defines how an interface or program treats a user. It’s important to consider UX as otherwise users may get lost or confused while using the program.
- **Build**: The act of creating an instance of the program so that it can be run.
- **Back-end**: The section of a program that isn’t directly interacted with by the user, usually used to do calculations or database work.