TEST PLAN FOR
<<LEAGUE OF TOWERS>>

ChangeLog

Version	Change Date	By	Description
version number	Date of Change	Name of person who made changes	Description of the changes made
1.0	02/26/2020	Christian, Huy, Nurida	Initialized version 1.0 of test plan

1 INTRODUCTION 2

1.1 SCOPE 2

1.1.1 In-Scope 2

1.1.2 Out-of-Scope 3

1.2 QUALITY OBJECTIVE 3

1.3 ROLES AND RESPONSIBILITIES 3

2 TEST METHODOLOGY 5

2.1 OVERVIEW 5

2.2 TEST LEVELS 5

2.3 BUG TRIAGE 5

2.4 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS 5

2.5 TEST COMPLETENESS 6

3 TEST DELIVERABLES 7

4 RESOURCE & ENVIRONMENT NEEDS 8

4.1 TESTING TOOLS 8

4.2 TEST ENVIRONMENT 8

5 TERMS/ACRONYMS 9

1 Introduction

Testing Plan

This Test Plan will be describing the scope, methodology approach, deliverables, and resources for testing activities related to the project, League of Towers. The test plan includes the following components:

- Unit tests to test all game functionalities and logic
- Acceptance tests that will test user interaction with game features/logic and front end
- A CI workflow that will build the project and run the implemented unit tests after merging through PR
- A github branch specifically used for testing new changes. New functionalities or changes made will be merged here then tested manually. Once testing is complete and approved, it will then be merged to the main/master branch.
- A development practice to create unit tests after implementation of major game functions/features

1.1 Scope

1.1.1 In-Scope

We will do the test cases on Core Features: Co-op Mode, and Level Up System first. Beside the Core Features, we will do tests on the base game functionality.

For more details, see the list of features below:

- Base game features should work well:
  - Number of enemies can be spawned for each wave
  - Enemy can move from spawn location to player's base location
  - Enemy can be killed when their HP reach 0
  - Base's lives will be reduced if the enemy reach the base
  - Base will be destroyed if its lives is 0
  - Towers can be picked from menu correctly by the players
  - Towers can be correctly placed on the map.
  - Towers can deal damage on the enemy.
- Coop features:
  - Players can connect to each other by:
    - Join a game
    - Create a game
  - At least 2 players can play at the same time in a game.
  - Players can place their own towers.
- Level Up Features:
  - The player can earn EXP by defeating enemies
  - The player can level up by earning enough EXP
  - The player will be added point after level up

    - The player can spend points on upgrading towers.

1.1.2 Out-of-Scope

Features that will not be tested in sprint 3 are: Authorization and Special Class System. Besides those features will be assets and their animation.

For more detail, see the list of features below:

- Authorization features:
    - Sign up account
    - Log in
- Special Class System features:
    - At least 2 different classes are available for players to pick.
    - Each class has at least 2 unique towers.
    - Choosing classes at the start.
- Non-functional features:
    - Assets: Sprite for map, tower, enemy, base.
    - Theme for Menu or other UI

1.2 Quality Objective

Our objectives for testing project:

- Base game's features work well with.
    - Coop Mode will be available.
    - Level Up Features will be available.
- All the features listed above passed the test requirement.
- At least 90% of the tests are success.
- All non-functional work well
    - The UI is displayed properly
- Bugs/issues are identified and fixed before going live.
- Ensure the game will meet the expectation of clients.

1.3 Roles and Responsibilities

Roles:

- Developer: Implements most of the game features.
    - Members: Juhee Kim, Nurida Karimbaeva, Christian Hintay, Hoang Huy Pham
- Test Manager: Setting up the test cases for each in-scope feature.
    - Members: Nurida Karimbaeva, Hoang Huy Pham, Christian Hintay
- Installation Manager: Setting up the required applications or 3rd party software that support our game.
    - Members: Juhee Kim, Hoang Huy Pham

Member's information:

Name	Net ID	GitHub username	Role
Juhee Kim	7864289	juliek1217	Developer/ Installation Manager
Christian Hintay	7791441	hintaycl	Developer/Test Manager
Nurida Karimbaeva	7851221	nuridak	Developer/Test Manager
Hoang Huy Pham	7824688	phamhhuy	Developer/Test Manager/Installation Manager

2 Test Methodology

2.1 Overview

As the course is iterative with milestones, as well as some of our features or game functionality having a possibility of undergoing change, we will be following the Agile methodology approach and do incremental testing. Since incremental testing is used in the agile development, we will test thoroughly every release/major iteration of the project. This will guarantee that any bugs encountered through development or testing are squashed and fixed before the project's next iteration/release. This approach will also allow us to modify or apply new changes to the project at any given moment that still conforms or complies with the course project requirements. Furthermore, by doing rapid or agile testing, we can lessen the chance of encountering bugs in the development cycle and release. However, the disadvantage of this testing approach is time overhead to the time constraint of the project as it requires constant communication and interaction between stakeholders specifically communication between developers, testers and the clients.

2.2 Test Levels

This project will include the following test levels:

- Unit Tests
- Acceptance Tests
- Travis Continuous Integration workflow

2.3 Bug Triage

The goal of the triage is to

- To define the type of resolution for each bug
- To prioritize bugs and determine a schedule for all "To Be Fixed Bugs'.

TBD(skip this part, it should be indicated in GitHub)

2.4 Suspension Criteria and Resumption Requirements

Suspension criteria define the criteria to be used to suspend all or part of the testing procedure while Resumption criteria determine when testing can resume after it has been suspended

TBD (not required for the course)

2.5 Test Completeness

The testing is complete when:

- The test coverage is over 90% which includes the functions and classes involved in game features/functionalities such as the co-op feature and leveling/upgrade system
- The user profile stub data is unaffected when unrelated game features are used
- All manual acceptance tests & automated test cases are executed and passes before merging from testing branch to main branch
- Connection to other player remains stable and unaffected when unrelated features or components are made or changed
- All major bugs are fixed in this current iteration's product demo/release
- Every bug PR is thoroughly reviewed and approved by at least 2 reviewers

3 Test Deliverables

Name	Description	Phase in development	Example
Test Plan	Document which contains the plan for all the testing activities to be done to deliver a quality product.	Test Planning	https://www.softwaretestingmaterial.com/test-plan-template/
Test Scenarios	Test Scenario gives the idea of what we have to test. Test Scenario is like a high-level test case.	Test Case Development	https://www.softwaretestingmaterial.com/test-scenario-vs-test-case/
Test Case	A list of cases that describe what exactly needs to be tested in detail.	Test Case Development	https://www.softwaretestinghelp.com/test-deliverables-in-software-testing/
Incident Report	Document which contains all the incidents such as resolved or unresolved incidents which are found while testing the software.	Test Execution	https://www.softwaretestingmaterial.com/test-deliverables/
Test Summary Report	A summary of all the test activities done and the test results in one document.	Test Closure	https://www.softwaretestingmaterial.com/test-deliverables/

.

4 Resource & Environment Needs

4.1 Testing Tools

- Requirements Tracking Tool
  - Unity Test Framework package
- Bug Tracking Tool
  - Visual Studio/ JetBrains Rider
  - Unity Test Runner
- Automation Tools
  - Travis-CI

4.2 Test Environment

Minimum hardware requirements: (*)

- For Windows System:
  - OS: Windows 7 (SP1+) and Windows 10, 64-bit versions only.
  - CPU: X64 architecture with SSE2 instruction set support
  - Graphics API: DX10, DX11, and DX12-capable GPUs
  - Additional requirements: Hardware vendor officially supported drivers
- For macOS:
  - OS: Windows 7 (SP1+) and Windows 10, 64-bit versions only.
  - CPU: X64 architecture with SSE2 instruction set support
  - Graphics API: DX10, DX11, and DX12-capable GPUs
  - Additional requirements: Hardware vendor officially supported drivers
- Linux:
  - OS: Ubuntu 20.4, Ubuntu 18.04, and CentOS 7
  - CPU: X64 architecture with SSE2 instruction set support
  - Graphics API: OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
  - Additional Requirements: Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)

Required Software:

- Windows 7 and above
- Unity version 2020.2.3f1
- Unity Test Framework version 1.1.22
- Editors support C#: Visual Studio, JetBrains Rider, ...
- Travis-CI

(*) Based on Unity's requirement since our tests use Unity's Test Frame.

5 Terms/Acronyms

Make a mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
CI	Continuous Integration
PR	Pull Request
QA	Quality Assurance
UI	User Interface