Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <07/May/20> | <1.0> | <initial description> | <Michaela Fleig> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 5

1.1 Purpose 5

1.2 Scope 5

1.3 Intended Audience 5

1.4 Document Terminology and Acronyms 5

1.5 References 5

1.6 Document Structure 5

2. Evaluation Mission and Test Motivation 6

2.1 Background 6

2.2 Evaluation Mission 6

2.3 Test Motivators 6

3. Target Test Items 6

4. Outline of Planned Tests 6

4.1 Outline of Test Inclusions 6

4.2 Outline of Other Candidates for Potential Inclusion 7

4.3 Outline of Test Exclusions 7

5. Test Approach 7

5.1 Initial Test-Idea Catalogs and Other Reference Sources 7

5.2 Testing Techniques and Types 7

5.2.1 Data and Database Integrity Testing 7

5.2.2 Function Testing 8

5.2.3 Business Cycle Testing 10

5.2.4 User Interface Testing 11

5.2.5 Performance Profiling 11

5.2.6 Load Testing 13

5.2.7 Stress Testing 14

5.2.8 Volume Testing 16

5.2.9 Security and Access Control Testing 17

5.2.10 Failover and Recovery Testing 18

5.2.11 Configuration Testing 20

5.2.12 Installation Testing 21

6. Entry and Exit Criteria 22

6.1 Test Plan 22

6.1.1 Test Plan Entry Criteria 22

6.1.2 Test Plan Exit Criteria 22

6.1.3 Suspension and Resumption Criteria 22

6.2 Test Cycles 22

6.2.1 Test Cycle Entry Criteria 22

6.2.2 Test Cycle Exit Criteria 22

6.2.3 Test Cycle Abnormal Termination 22

7. Deliverables 22

7.1 Test Evaluation Summaries 22

7.2 Reporting on Test Coverage 22

7.3 Perceived Quality Reports 23

7.4 Incident Logs and Change Requests 23

7.5 Smoke Test Suite and Supporting Test Scripts 23

7.6 Additional Work Products 23

7.6.1 Detailed Test Results 23

7.6.2 Additional Automated Functional Test Scripts 23

7.6.3 Test Guidelines 23

7.6.4 Traceability Matrices 23

8. Testing Workflow 23

9. Environmental Needs 24

9.1 Base System Hardware 24

9.2 Base Software Elements in the Test Environment 25

9.3 Productivity and Support Tools 25

9.4 Test Environment Configurations 25

10. Responsibilities, Staffing, and Training Needs 26

10.1 People and Roles 26

10.2 Staffing and Training Needs 28

11. Iteration Milestones 28

12. Risks, Dependencies, Assumptions, and Constraints 29

13. Management Process and Procedures 30

13.1 Measuring and Assessing the Extent of Testing 30

13.2 Assessing the Deliverables of this Test Plan 30

13.3 Problem Reporting, Escalation, and Issue Resolution 30

13.4 Managing Test Cycles 30

13.5 Traceability Strategies 30

13.6 Approval and Signoff 30

# Introduction

## Purpose

The purpose of the Iteration Test Plan is to gather all of the information necessary to plan and control the test effort for a given iteration. It describes the approach to testing the software, and is the top-level plan generated and used by managers to direct the test effort.

This *Test Plan* for the supports the following objectives:

* Identifies the items that should be targeted by the tests.
* Identifies the motivation for and ideas behind the test areas to be covered.
* Outlines the testing approach that will be used.
* Identifies the required resources and provides an estimate of the test effort.

## Scope

This document describes the used tests, as they are unit tests and functionality testing.

## Intended Audience

This document is meant for internal use primarily.

## Document Terminology and Acronyms

* **SRS** Software Requirements Specification
* **n/a** not available

## References

[SAD](https://github.com/GabrielBalzer/hypercasualgame/blob/master/SAD.docx)

[Function Points](https://github.com/GabrielBalzer/hypercasualgame/tree/master/documentation/FunctionPoints)

[UCs](https://github.com/GabrielBalzer/hypercasualgame/tree/master/documentation/Use%20Cases)

## Document Structure

n/a

# Evaluation Mission and Test Motivation

## Background

By testing source code, we ensure our application to run smoothly. The goal is to make sure, that our application does not run into any unexpected errors.

## Evaluation Mission

The mission of this test plan is to prevent errors and ensure an outstanding software quality.

## Test Motivators

Out testing is motivated by:

* technical risks
* quality risks
* use cases
* functional requirements

# Target Test Items

The listing below identifies those test items⎯software, hardware, and supporting product elements ⎯that have been identified as targets for testing. This list represents what items will be tested.

Items for testing:

* API testing (does the API react to the user input)
* Player behavior testing (does the player react appropriate)
* Obstacles and background (behavior testing)
* Game settings

# Outline of Planned Tests

## Outline of Test Inclusions

* Unit testing our methods as described by each use cases.
* Functional testing of player, obstacles, background and user input.

## Outline of Other Candidates for Potential Inclusion

Stress testing the application and its device storage.

## Outline of Test Exclusions

n/a

# Test Approach

## Initial Test-Idea Catalogs and Other Reference Sources

n/a

## Testing Techniques and Types

### API Testing

|  |  |
| --- | --- |
| Technique Objective: | Unity works as a testing system of implemented methods. Calls its methods multiple times. Result is evaluated by the tester. |
| Technique: | Test user starts the game. Static code review. |
| Oracles: | API returns the correct output as expected. |
| Required Tools: | Visual Studio 2017 (Editor), Unity Framework 2018 (Engine) |
| Success Criteria: | Output equals input. |
| Special Considerations: | Settings in Unity Framework can block implemented methods. |

### Player Behavior Testing

|  |  |
| --- | --- |
| Technique Objective: | The behaviour of the virtual player shall be tested. Expected results are predefined. |
| Technique: | Test user plays the game. Static code review. |
| Oracles: | API returns the correct output as expected by the developers. |
| Required Tools: | Visual Studio 2017 (Editor), Unity Framework 2018 (Engine) |
| Success Criteria: | Output equals input. |
| Special Considerations: | Settings in Unity Framework can block implemented methods. |

### Obstacles and Background

|  |  |
| --- | --- |
| Technique Objective: | The behaviour and reactions of the obstacles shall be tested. Expected results are predefined. |
| Technique: | Test user starts the game. Can also be tested by code testing. Static code review. |
| Oracles: | API shows the behaviour as expected. |
| Required Tools: | Visual Studio 2017 (Editor), Unity Framework 2018 (Engine) |
| Success Criteria: | Output equals input. Obstacles can be destroyed or have to stay. |
| Special Considerations: | Settings in Unity Framework can block implemented methods. |

### Game Settings

|  |  |
| --- | --- |
| Technique Objective: | Several functions that were predefined in each use case shall be tested. |
| Technique: | Test user plays the game. Static code review. Unit tests. |
| Oracles: | Function is executed and its result is compared to the expected. |
| Required Tools: | Visual Studio 2017 (Editor), Unity Framework 2018 (Engine) |
| Success Criteria: | Output equals input. |
| Special Considerations: | - |

### Business Cycle Testing

n/a

### User Interface Testing

### n/a

### Performance Profiling

n/a

### Load Testing

n/a

### Stress Testing

n/a

Volume Testing

n/a

### Security and Access Control Testing

n/a

### Failover and Recovery Testing

n/a

### Configuration Testing

n/a

### Installation Testing

n/a

# Entry and Exit Criteria

## Test Plan

### Test Plan Entry Criteria

[Specify the criteria that will be used to determine whether the execution of the **Test Plan** can begin.]

### Test Plan Exit Criteria

[Specify the criteria that will be used to determine whether the execution of the **Test Plan** is complete or that continued execution provides no further benefit.]

### Suspension and Resumption Criteria

[Specify the criteria that will be used to determine whether testing should be prematurely suspended or ended before the plan has been completely executed, and under what criteria testing can be resumed.]

## Test Cycles

### Test Cycle Entry Criteria

The test version will be the desktop version of Ted’s Quest. Computer or operating system version does not matter.

### Test Cycle Exit Criteria

When the tests are being marked as passed. Even failed tests can pass, when the customer is fine with that.

### Test Cycle Abnormal Termination

n/a

# Deliverables

## Test Evaluation Summaries

A test run summary will be available in our repository, once they are executed and evaluated.

## Reporting on Test Coverage

n/a

## Perceived Quality Reports

n/a

## Incident Logs and Change Requests

n/a

## Smoke Test Suite and Supporting Test Scripts

n/a

## Additional Work Products

n/a

### Detailed Test Results

Detailed results will be available in our repository, once they are executed and evaluated.

### Additional Automated Functional Test Scripts

n/a

### Test Guidelines

n/a

### Traceability Matrices

n/a

# Testing Workflow

Developers should execute tests locally before pushing source code. When pushing to master, tests are executed automatically, when pipelining is implemented.

# Environmental Needs

This section presents the non-human resources required for the test plan.

## Base System Hardware

n/a

## Base Software Elements in the Test Environment

n/a

## Productivity and Support Tools

n/a

## Test Environment Configurations

n/a

# Responsibilities, Staffing, and Training Needs

## People and Roles

This table shows the staffing assumptions for the test effort.

| **Human Resources** | | |
| --- | --- | --- |
| **Role** | **Minimum Resources Recommended**  **(number of full-time roles allocated)** | **Specific Responsibilities or Comments** |
| Test Manager | 1 | Provides management oversight.  Responsibilities include:   * planning and logistics * agree mission * identify motivators * acquire appropriate resources * present management reporting * advocate the interests of test * evaluate effectiveness of test effort |
| Test Analyst | 1 | Identifies and defines the specific tests to be conducted.  Responsibilities include:   * identify test ideas * define test details * determine test results * document change requests * evaluate product quality |
| Test Designer | 1 | Defines the technical approach to the implementation of the test effort.  Responsibilities include:   * define test approach * define test automation architecture * verify test techniques * define testability elements * structure test implementation |
| Tester | 2 | Implements and executes the tests.  Responsibilities include:   * implement tests and test suites * execute test suites * log results * analyze and recover from test failures * document incidents |
| Test System Administrator | 1 | Ensures test environment and assets are managed and maintained.  Responsibilities include:   * administer test management system * install and support access to, and recovery of, test environment configurations and test labs |
| Designer | 1 | Identifies and defines the operations, attributes, and associations of the test classes.  Responsibilities include:   * defines the test classes required to support testability requirements as defined by the test team |
| Implementer | 1 | Implements and unit tests the test classes and test packages.  Responsibilities include:   * creates the test components required to support testability requirements as defined by the designer |

## Staffing and Training Needs

n/a

# Iteration Milestones

n/a

# Risks, Dependencies, Assumptions, and Constraints

| **Risk** | **Mitigation Strategy** | **Contingency (Risk is realized)** |
| --- | --- | --- |
| Unexpected failures. | Cover all important functions in tests. | * Rollback deployment to last stable version. * Fix in new stable version. |

# Management Process and Procedures

## Measuring and Assessing the Extent of Testing

n/a

## Assessing the Deliverables of this Test Plan

n/a

## Problem Reporting, Escalation, and Issue Resolution

n/a

## Managing Test Cycles

n/a

## Traceability Strategies

n/a

## Approval and Signoff

n/a