<Company Name>

<Ted's Quest>
<Master> Test Plan

Version <1.0>

Revision History

Date	Version	Description	Author
<07/May/20>	<1.0>	<initial description>	<Michaela Fleig>
<27/Jun/20>	<1.1>	<update information>	<Michaela Fleig>

Table of Contents

<Iteration/ Master> Test Plan

1. Introduction

1.1 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 <Ted's Quest> 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.

1.2 Scope

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

1.3 Intended Audience

This document is meant for internal use primarily.

1.4 Document Terminology and Acronyms

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

1.5 References

SAD
Function Points
UCs

1.6 Document Structure

n/a

2. Evaluation Mission and Test Motivation

2.1 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.

2.2 Evaluation Mission

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

2.3 Test Motivators

Out testing is motivated by:

- technical risks
- quality risks
- use cases
- functional requirements

3. 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

4. Outline of Planned Tests

4.1 Outline of Test Inclusions

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

4.2 Outline of Other Candidates for Potential Inclusion

Stress testing the application and its device storage.

4.3 Outline of Test Exclusions

Testing parts that are provided by the Unity Engine.

5. Test Approach

5.1 Initial Test-Idea Catalogs and Other Reference Sources

The first way to test the Unity application is by gathering reviews from test users. These reviews can be achieved by questioning them after playing. Another way to learn about the intuitive of the game is to let the testers record their screen while playing the game of the first time.

5.2 Testing Techniques and Types

5.2.1 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.

5.2.2 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.

5.2.3 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.

5.2.4 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:	-

5.2.5 Business Cycle Testing

n/a

5.2.6 User Interface Testing

n/a

5.2.7 Performance Profiling

n/a

5.2.8 Load Testing

n/a

5.2.9 Stress Testing

n/a

Volume Testing

n/a

Security and Access Control Testing

n/a

5.2.10 Failover and Recovery Testing

n/a

5.2.11 Configuration Testing

n/a

5.2.12 Installation Testing

The mainly used method for testing the application is to let testers play the game and review it afterwards.

6. Entry and Exit Criteria

6.1 Test Plan

6.1.1 Test Plan Entry Criteria

The entry criteria will be when the application is started by the tester.

6.1.2 Test Plan Exit Criteria

No further benefit will be, when the user has passed through all use cases and has filled the review sheet. Then the test plan is completed and can be finished. On the other hand, the testing can also be finished if the user does not like the application or does not get along with the game at all.

6.1.3 Suspension and Resumption Criteria

The complete testing is suspended, once the representative sample (about five people) is achieved but at least after one week.

6.2 Test Cycles

6.2.1 Test Cycle Entry Criteria

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

6.2.2 Test Cycle Exit Criteria

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

6.2.3 Test Cycle Abnormal Termination

The test is aborted once there are not enough players or the review sheet is not used correctly by testers or people who did not play the game but give feedback.

7. Deliverables

7.1 Test Evaluation Summaries

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

7.2 Reporting on Test Coverage

The screenshots from the website summary will be added to the GitHub documentation folder.

7.3 Perceived Quality Reports

Perceived quality will be reported by the test users and are visible on the screenshots.

7.4 Incident Logs and Change Requests

Change requests can be added by the tester through the review sheet.

7.5 Smoke Test Suite and Supporting Test Scripts

There are no additional suites or scripts. The user is testing the appearing of the game.

7.6 Additional Work Products

Additional products help testing the code. The Website Surveymonkey makes fast online questions possible and is used by Ted's Entertainment.

7.6.1 Detailed Test Results

A general result will be provided by the website that hosts the review sheet.

7.6.2 Additional Automated Functional Test Scripts

Since the Unity Engine does not support any code testing, not even to test the self-written code, there can only be added the .feature files that were generated for a first testing experimentation.

7.6.3 Test Guidelines

The test user is free to decide if he wants to answer all questions. He can also add further information, tipps and implementation wishes for the game.

7.6.4 Traceability Matrices

n/a

8. Testing Workflow

Developers should execute tests locally before pushing source code. The tester has to download the executable as compressed version and just run the executable.

9. Environmental Needs

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

9.1 Base System Hardware

The application does not depend on hardware.

9.2 Base Software Elements in the Test Environment

The application does not depend on either additional software nor the operating system.

9.3 Productivity and Support Tools

n/a

9.4 Test Environment Configurations

There are no configurations to be done by the user.

10. Responsibilities, Staffing, and Training Needs

10.1 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

Human Resources		
Role	**Minimum Resources Recommended** **(number of full-time roles allocated)**	**Specific Responsibilities or Comments**
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

10.2 Staffing and Training Needs

Since the aim for testing with real users is also to see how easy the game is to be used by others than the developers.

11. Iteration Milestones

Since the testing method is to ask users to test the game and review it, it is not possible to define iteration milestones.

12. 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.

13. Management Process and Procedures

13.1 Measuring and Assessing the Extent of Testing

There are not be more than 20 questions to keep the tester motivated to fill the entire document. But at least five questions.

13.2 Assessing the Deliverables of this Test Plan

The results can be good to rethink about the complexity of the application and the needed functions in the game.

13.3 Problem Reporting, Escalation, and Issue Resolution

Found issues can be fixed and updated easily and fast. Therefore, it is required to build and upload a new executable file to the GitHub server.

13.4 Managing Test Cycles

Test cycles will be automatically summarized by the used web-tool and can be reviewed at each step of the testing cycles.

13.5 Traceability Strategies

The web-tool allows to see how many different user have played the game and gives a time-structured progress.

13.6 Approval and Signoff

The results are reviewed by the entire team of Ted’s Entertainment in order to approve or decline the requests and get the feedback.