Test Plan

Group 2
CIS 411
Keith, Martin, Cassidy, Michael

Document prepared by: Keith Reis

Edge of Extinction

February 12th, 2023

Overview

This document was created in order to help provide a detailed overview regarding test strategies, objectives, schedule, time-estimations/deadlines, and resources required in order to efficiently and thoroughly complete testing.

Components

1. **Scope:** *Provide objectives for our game - specifically user scenarios of issues that players may run into whilst playing the game.*
2. **Schedule:** *Start dates and deadlines for testers to deliver detailed and well-documented results.*
3. **Resource Allocation:** *Which testers will work on what test objectives.*
4. **Environment:** *What testing interface will we use for*
5. **Tools:** *What tools will be used for testing, bug reporting, and any other relevant activities.*
6. **Defect Management:** *Details how bugs should be reported and documented. For example, should we include screenshots, text logs, or videos of the bug occurrence within the code?*
7. **Risk Management:** *A detailed description of what risks may occur both in testing, and possible risks if the project is released without sufficient testing.*
8. **Exit Parameters:** *When testing activities must stop.*

Project Analysis

Our project is a game based on the education card-game titled 'Edge of Extinction'. The focus of the game is to provide players with a fun yet strategically challenging experience. While also being an educational platform that teaches players about different types of animals, plant types, and other facets of nature and wildlife. Therefore, our game's target audience is centered around being used in an educational setting by teachers and students.

We want to make the game as easily accessible to download and play as possible. As well as having a clean UI that is intuitive enough to quickly learn and begin playing. Our goal is for teachers to be able to have their students choose between two modes: Assisted Mode or Standard Mode. With assisted mode being focused on younger or inexperienced players who may need extra help/tips given by the UI. While standard mode will focus on older or experienced players that should be able to play the game without any external help from the program. This way, we can ensure that all types of players can find enjoyment from our game without getting frustrated, confused, or feeling overwhelmed.

As this project has already been previously worked on and attempted by 2 groups before us, we will continue development of it in the Unity Engine. The editor version we'll be using is *2019.3.12f1*. Upon completion, we'll export the game to an executable program that will be hosted on the card-game's home website. We also plan to continue following and adhering to the past group's coding architecture and design. Mainly their use of OOP(Object-Oriented Programming) methodology. As we also believe the use of objects and inheritance is the best and most efficient means of coding and implementing the necessary components for the game.

Our client - Jason Strohm, is very passionate about the game and seeing it come to fruition in a virtual environment. His two daughters are the ones who developed the card game when they were younger. As Strohm's dedicated development team, it is our goal to ensure that we meet all of his requested requirements for the game.

Test Strategy

Scope:

We will be testing the code found within the previous group's project folder that was submitted titled '*Edge_Of_Extinction_Working*'. All testing will be done within the Unity environment. Our focus for now lies specifically within the '*Scripts*' subdirectory that is contained in the project's '*Assets*' main directory. The scripts subdirectory contains all C# code files, with each file being separated into a distinct class. In total, there are 28 files in the Scripts directory that need to each be thoroughly reviewed and tested during gameplay.

Type of Testing:

Instead of using an external automated testing software, human testing will be our approach to discovering and eliminating any and all bugs. Due to the fact that we're creating a game played by humans, it's ideal to use the human testing approach. As we want to account for ways in which a human may approach certain aspects of our game and unintentionally cause errors or bugs to arise during gameplay.

We'll use two specific types of software testing methods that best suit our current resources and timeframe.

- **Unit Testing** - When we find a line(s) of code that we know is responsible for a given bug; we'll test those individual components. Rather than spending the time necessary to test the entire module that contains said code.
- **Regressive Testing** - As we make changes to the project's code, we'll want to ensure that our current versions of the game works as intended/expected. This method of testing will ensure that we don't '*regress*' back to versions of our project that suffer from bugs or issues we've previously run into. Instead, the project should always be moving forward - with each fix providing a more stable version than the last.

Risks and Issues:

- **Maintaining an equal balance of focus on testing - as well as development.**
    - Deadlines of other tasks must still be met while also dedicating time to meeting testing deadlines.
- **Fixing one bug may cause a waterfall effect of other unexpected errors occurring.**
    - Ensure that each bug found is well-documented, and a backup copy of the class file is kept. That way, we can revert to the previous version if necessary to figure out a different approach to fixing the bug if need be.
- **We may face regression of the project at some point**
    - As a group - we each must actively keep old versions of the code to compare against our current ones. As well as a detailed bug tracking document of issues we have resolved. That way, we can compare the newest/current version against both of these factors to avoid the project regressing and wasting resources.
- **Task responsibilities and other work may spread our resources thin when it comes to actively testing.**
    - Time management is a key factor that needs to be maintained as strictly as possible to ensure testing is happening regularly, as well as bug documentation. This will hopefully avoid the issue of our resources being spread too thinly across all of our different project deadlines.

Logistics:

List of testers

- **Martin**
- **Michael**
- **Keith**
- **Cassidy**

Test Objectives

- **Main Screen:** Ensure the main screen of the game is adequately tested and working as intended. The selection options on the main screen can be considered suitable for launch if they do the following without errors:
    - **New Game -** Begin execution of all necessary game files and properly populate the game environment with the necessary objects/GUI needed for gameplay.
    - **Learn To Play -** Bring up a tutorial screen that has a brief description on how to play the game. The exit 'X' button should take the player back to the main screen.

Test Exit Criteria

1. If at least **50%** of **low priority** bugs are resolved - the game can be considered suitable for public use.

2. If **100%** of **medium/high priority** bugs are resolved - the game can be considered suitable for public use.

3. If the client is content with the state of the game when development has finished, and feels that all avenues of gameplay have been tested thoroughly enough - the game can be considered suitable for public use