Requirements & Test Plan Summary

Group 22 - Aravind Achanta, Yonathon Gordon, Naga Kameswari Spoorthi Pendyala, Sai Priya Jyothula, Sumanth Reddy Pandugula, Vakkalanka V S S Dilip Raju

This document provides a summary of the requirements, overview of the test plans and testing schedule. It puts forward the approach for the development of the multiplayer game **Lithos** and serves as a rulebook for the game developers. It further provides the testers with an insight about the tests that should be performed to ascertain that the system satisfies all the specified requirements.

Requirements

The requirements for the development of this game are formulated by considering a sample persona 'John' who is a high schooler. Since the target audience mostly consists of students, the requirements are devised by taking into account the perspective of a potential user. Ideally, while developing any product, a logical classification of the requirements is based on the degree of importance of the particular requirement. Implementation of some might be of utmost importance, while the others may be negotiable. The requirements are organized in a way that best assists the development process and are prioritized for an efficient implementation. The ones that cannot be incorporated in the current release must be included in the future releases. The requirements in this document are broadly classified into two classes – Functional and Non-functional requirements. Further, the requirements documentation also includes the services offered by the system and the constraints under which the game operates.

1. **Functional Requirements:** Functional Requirements are used to judge the system operation as they specify the criteria that pertain to the functional aspect of the product. In this game, the system must display the goals of the current level and provide the player with relevant clues according to the challenges of the level. In order to assign tasks to the members of the tribe, the system must provide a means for the player to choose a team member, assign the tasks and update the task table based on the assigned tasks. The loop runs until the player assigns the tasks to all the tribe members. The AI engine should simulate the tasks and make decisions on how these tasks solve the game challenges or affect the game environment. The system should allow the player to connect to external players in the multiplayer mode and to collaborate or fight with them based on his/her preference. The system should support multiplayer game environment and must be able to accommodate the actions of multiple players. After meeting the goals at each level, the system should display an end-of-level quiz and progress to the next level if the player performs satisfactorily.

2. **Non-functional Requirements:** The Non-functional requirements are the expected characteristics of a software that enhance the overall experience of the user. A variety of non-

functional requirements can be listed for **Lithos**, such as the requirements related to security, performance, cost, interoperability, accessibility etc. For example, when it comes to online games, performance is a major issue. When the user is playing the game, the number of frames per second (FPS) displayed to the user in high resolution should be 24 FPS for 90% of the time and it should not fall below 18 FPS under any circumstances. If the bandwidth of the user falls below 1 MBPS then the graphics should be compromised to lower resolution graphics instead of lowering the number of FPS. Reliability requirements consider the unlikely event that the server crashes – the game log of users should be backed up to a secondary server, which then acts as the actual server so that the system is still available to the users. Usability features include support for people who cannot identify colors by providing closed captioning and audio cues. Availability requirements specify the up-time of the system – the game server should be available 99.9% of the time and in case of server maintenance, the server should resume functionality within 15 minutes.

Test Plans and Testing Schedule

Developing a game is a very complex task, and hence it will have to pass many various tests until it is ready for deployment. This application will require thorough testing for the single player as well as the multiplayer mode. The testing methodology for this software should start with unit testing, to enable the tester to test a function or a particular module separately from the rest of the program. Unit testing is performed on the components as and when they become available. These individual units are compiled into systems (each system performs a particular functionality). After a system has passed all previous tests, integration testing will be done prior to that system's actual integration into the game. Since the game is developed using the UNITY 3D engine, testing should be done within the UNITY framework. The test cases are developed for different modules of the game. There will be test cases for the login functionality of the game, for every level in the game, for single and multiplayer mode and for validating the download and installation of the game.

Once the software is developed using the agile methodology and tested, there has to be a build made for each of the platforms such as Windows, Mac OS X and Linux. Each build has to be tested on the platform on which it is meant to run. For the testing of the multiplayer mode, it has to be tested using different network connections with different bandwidth speeds. The performance and the latency criteria are set to industry standards. For example, if it takes 30 seconds to verify user credentials, clearly the algorithm implemented has to be refactored even if it is theoretically correct.

The test schedule specifies that the project will be given about three years to develop, and so the testing should be done once a month. There will be extensive testing done after a year within the UNITY platform. The last 2 weeks before the final release will be spent on testing the build on various platforms and hardware.