|  |
| --- |
| University of Pretoria |
| Testing Information |
| COS301 - Zeon |

|  |
| --- |
| Zenadia Groenewald, Bernhard Muller, Lecton Ramasila  10-17-2014 |

Introduction

This document serves as a record and proof of the tests that took place for the Stream2Me project. Herein the reader may find the purpose of the various tests as well as the results thereof, along with links to the repository where many of the tests are stored. This document further serves to explain how tests are carried out, how they are developed and the extent to which they cover the project’s functionality.

Unit Testing

The unit testing of the Stream2Me project was performed in different phases, each pertaining to the testing of a separate component or module. Firstly, communication, the front-end features, then the back-end features, and finally the security modules. The following modules were the primary focus of the unit tests:

* Message handling: Accepting only valid Stream2Me message object, identifying them correctly and receiving them in a uncompromised state.
* Combined streaming: the streaming from client to client was the primary focus of the project as it would provide the basis for the streaming of various media once the project’s functionality expanded. The streaming functionality was tested primarily by comparing the overall efficiency and performance when the streaming of different messages was tested; this includes text, audio and video/image messages.
* Audio: audio streaming had to be performed in such a way that the audio data sent/received suffered minimal delay during transmission and was also in sync with any possible video data that might have been sent in conjunction. This would hereby allow users to transmit video or image messages while simultaneously streaming audio recordings or microphone recordings. Unit tests were performed by streaming an audio file over the network to the audio player.
* Video: video and/or image streaming had to be done in a way that allowed the screenshots (which the video files comprised of) to be transmitted with minimal delay and in a way that the delay, if any, was indiscernible. As such, screenshots had to be sent from one point to another in quick succession to allow for “smoothness” in the video’s transmission. Video frame tests were performed by streaming image files over the network to the video player, similarly to the audio unit tests.
* Android (audio & video): in much the same way as that of the desktop application, the audio and video/image streaming of the Android application had to be carried out in such a way as to ensure that the delay between messages or message segments is minimal.
* Database: the database formed a significant portion of the back-end, as it would contain all the information associated with the various users who are registered to use the program’s services; the database had to be secure, efficient in querying and it had to be able to handle fairly large quantities of data.

Integration Testing

The purpose of integration testing is to ensure that all system modules and components work as intended, when integrated into the system as a whole. This process involves extensive unit tests to ensure that the units themselves are working properly before they are included in the main project; as well as ensuring that the modules are fully functional once integrated.

The system is fully tested and functional, as per the client’s specifications.

Non-functional Testing

Several factors form part of the non-functional scope, these include factors such as security, performance, and user-friendliness.

The tests for these factors were carried out and successful. Messages sent and from clients are transmitted securely; the system registers users with a unique user identification and a password of their choosing; messages are encrypted and it is assured that no two encrypted pieces of data result in the same ciphertext, regardless of message or password similarity.

User-friendliness was a complicated test in that it required the subjective opinions of both the clients as well as several users. For this very reason, a questionnaire was compiled and several persons were asked to test the usability of the system, and the user-friendliness of the interface.

Usability Testing

A group of 20 volunteers were gathered to partake in the usability testing of the application anonymously. Testing happened in groups of 3 and users provided feedback on any bugs and errors found during their usage of the application. Users were given a set of 10 questions to rate the application on from 1 to 6 after the testing. Questions are below, with the testing results in the graph below.

Questions

|  |  |
| --- | --- |
| 1 | I think that I would like to use this system. |
| 2 | I found the system unnecessarily complex. |
| 3 | I thought the system was easy to use. |
| 4 | I thought that I would need the support of a technical person |
| 5 | I found the various functions of the system were well integrated. |
| 6 | I found there was too much inconsistency in the system. |
| 7 | I would imagine that most people would learn to use this system very quickly. |
| 8 | I found the system very cumbersome to use. |
| 9 | I felt very confident using the system. |
| 10 | I needed to learn a lot of things before I could get going with this system. |

Graph Analysis