|  |
| --- |
| University of Pretoria |
| Software Documentation for Stream2Me |
| COS 301 Team Zeon – Version 2 |

|  |
| --- |
| Zenadia Groenewald, Bernhard Muller, Lecton Ramasila  5-18-2014 |

Change log / History

|  |  |  |
| --- | --- | --- |
| Date | Version | Description |
| 12/07/2014 | v0.1 | Added limitations |
| 13/07/2014 | v0.2 | Overview added |
| 13/07/2014 | v0.3 | Vision and Scope added |
| 13/07/2014 | v0.4 | Architecture Requirements added |
| 13/07/2014 | v0.5 | Integration Requirements added |
| 13/07/2014 | v0.6 | Functional Requirements added |
| 17/07/2014 | v0.7 | Architecture Requirements changed |
| 17/07/2014 | v0.8 | Limitations changed |
| 20/07/2014 | v0.9 | Hardware constraints changed |
| 02/08/2014 | v0.10 | Functional requirements changed |
| 15/08/2014 | v0.11 | Architecture constraints changed |
| 16/08/2014 | v0.12 | Integration Requirements |

1 Overview

The following document serves to provide specific guidelines for the documentation and development of the Stream2Me project. This document also serves to fully describe and outline the software and development details to be implemented, as well as provide a description and elaboration of the system’s details. The associated documentation will be managed and updated continuously throughout the project’s undertaking to ensure proper version control.

* 1. Assumptions

The guidelines for the documentation follow on the assumption that the development process of the project follows a hybrid process consisting of:

1. An up-front software architecture and design phase.
2. An iterative development process where functionality is determined and developed.
3. A variation of the Scrum development methodology in which flexibility and project agility are a high priority, specifically to allow for ad-hoc changes and adjustments as requested by the clients, or as deemed necessary by the developers.
   1. Overview of documents and when they are created

Within the aforementioned process, the first details specified include the vision and scope of the system to be implemented; following this, one specifies the architectural requirements and thusly defines an initial version of the software architecture specification. From this point onward, each use case for the system is developed iteratively and for each of these, functional requirements, application design, implementation and functional unit testing is performed.

The purpose of these continuous updates and evaluations is to keep the information pertaining to the system as up-to-date and relevant as possible, so as to ensure that the functional requirements, application design, functional testing and user manual documents contain the most relevant system information.

First, the vision and scope specifies what the client wishes to achieve with the software system and the high-level scope of the software system itself.

Second, the architectural requirements documentation specifies the system access and integration requirements, the quality requirements, as well as a set of architectural constraints given by the client.

When the software/design specification is defined, application development can be achieved with various software methodologies. For every change made, the functional or application requirements are added to the functional requirements specification document and the application design is added to the application design document, the functional testing is added to the functional testing document and the instructions on the use of system functionality is added to the user’s manual.

Finally, the product is shipped to the clients with the above documentation included as well as additional documents for the deployment guidelines and the non-functional testing guidelines.

2 Vision and Scope

The Stream2Me media streaming system aims to provide users with the ability to stream various media (including audio, video, images and screen captures) between devices using the Internet, without the use of e-mail or similar tools.

This system will allow students as well as other users to transfer data remotely; a common inconvenience faced is the need to share content at a moment’s notice and requiring to either swivel a screen to display it to the person in question, lend earphones, or even send the data via e-mail and hope the recipient notices its presence soon. The aim of this system is to allow users to transfer information and/or media to others without the need to use the aforementioned methods.

2.1 Scope and Limitations/Exclusions

The possible limitations of the proposed software solution include the following:

* The need to stream potentially sensitive data requires that the system itself be secure and requires that the data being transmitted not be interceptable or corruptible in its transmission. This can render some of the system functionality slow in some cases, as the data will have to pass various security and validity checks.
* Communication is achieved through TCP transfer, so data packets are able to arrive in their appropriate order, at the cost of speed and potentially transmission capacity.
* Connection speed can cause the application’s video streaming to appear pixelated, to appear as though the frames have frozen, or to appear to be of poor quality; this is not always desirable for video or image transferral regardless of speed and ease of transfer.
* Certain clients may be working on restricted bandwidth, so it will be necessary to include functionality to allow them to decline a pending stream.
* The possible need to include a built-in media player, which may occupy more space.
* The streaming of audio and video separately may cause them to appear out of sync if either stream encountered problems while being sent.

3 Architecture requirements

3.1 Access channel requirements

Senders – Streaming information

* Access to an interface which allows the user to select the medium of data to transfer (i.e. video, audio, etc.).
* Access to an interface which allows the user to select whether or not to broadcast the data to multiple recipients in a group or to a single recipient; and should the user choose to broadcast to many recipients, the system should allow the user to select to whom they wish to broadcast the information.

Receivers – Accepting information

* Access to an interface which allows the recipient to choose whether or not they want to accept the information being broadcasted to them.
* Allow the users to select who they receive information from.
* Allow the users to be able to stop a data stream or broadcast midway through the process, or cancel it completely.

Mobile application and desktop interface

* Connects to the Internet via a hosted server.
* Connects to hosted servers via IP.

3.2 Quality requirements

Performance

* The system must be able to send, receive and stream data in as close to real time as the transmission method allows.
* The system must be able to stream data via an Android mobile application.
* The system must be largely independent of additional software or operating system versions.
* The system should be able to run sufficiently fast

Reliability

* If a device or computer goes offline or disconnects, the data streaming must be cancelled immediately but still be available for streaming once the device or computer re-establishes its connection.
* The system will not be functional unless there is a stable connection established with a server and, in turn, the Internet.

Security

* The system must be able to secure the transfer of data before and during transmission until it reaches its intended recipient.
* Data being streamed must not be corruptible or interceptable during its transmission.

Scalability

* The system must be able to scale for multiple users both on the mobile and desktop application. It must be able to do so concurrently.
* Performance should, ideally, not be affected by the number of users on the system.
* Must be able to scale to use by 400 users or more.
* The server must be mountable from the Internet as well as a local server.

Flexibility

* The system should be accessible from a mobile application interface as well as a desktop application interface.

Maintainability

* Both desktop and mobile versions of the application should be easy to maintain and not be co-dependent in any way.

Integrability

* All layers of the application must integrate with the others, without the need for regular human attention/intervention to function as intended.
* Any and all modules in the application must be able to run independently in the testing phases and must integrate fully with the rest of the system once testing is complete.

Usability

* The interface for both the Android and the desktop application must be user-friendly, intuitive, and straightforward; preferably with help functionality and/or on-screen guidance.

Cost

* The monetary cost of the mobile application and using the desktop interface must be free.
* The cost of hosting a server over the Internet may incur monetary expense and is not recommended for a free-to-use application unless funding can be accommodated.

3.3 Integration requirements

The integration channel section will cover the information relevant to the program’s interfacing with different systems; the systems involved and how the interfacing is to be achieved. The integration between the users and the server will be the most crucial part of this undertaking, and plays the biggest role in the connection between users and the transfer of data.

The integration takes place mostly between the users and some server that is able to establish a stable Internet connection. The users serve largely as a point of origin (the senders) and the endpoint (the recipients) in the majority of the use cases; whereas the server is identified largely as the middleground where the authentication and management of connections and data takes place. The server and multiple clients would communicate with the use of TCP to ensure the easy transmission of certain media types (such as video) without drastic impact on the system’s overall performance. Both the mobile and desktop applications will be required to perform real time media streaming. This implies that a strong and stable connection is required to ensure that data is not lost or corrupted in transmission.

The following protocols include all the necessary functionality to achieve stable and secure media transmission:

* TCP/IP
* Netty

3.4 Quality requirements

Performance

The server must be able to accommodate for multiple clients accessing and/or using the connection at the same time. Integration of the client and server components must be performed in such a way that the performance of the system is not compromised.

Security

The system must be secure and the transfer of media and other data must be kept as simple but effective as possible. No malevolent party or user should be able to gain access to information from another user that is not explicitly sent to him/her; nor should the information be corruptible or interceptable during transmission. Persons should also not be able to access information shared by groups they are not in.

3.5 Architecture constraints

The constraints of the system’s architecture provide boundaries and more rigid outlines for the system and allows us to specify the system design within specific limitations.

Hardware constraints

First of all, the system should be able to function on any personal computer running a Windows 7 operating system. On these types of hardware, the user will make us of the application via a user interface which allows them to perform media streaming. The system also needs to be able to run on mobile devices running Android operating systems, this includes smartphones and tablets; this must be achieved through a programmed mobile version of the desktop application, in the form of a suitable Android application.

Software constraints

The software constraints of the system are discussed more extensively as they form a much larger part of the scope. The following technologies and approaches must be considered and utilised in the system implementation:

* Android and Windows 7 are the only systems that the application needs to be designed for.
* GitHub will be used as the web-based repository to store any of the files or documentation associated with the project.
* The Java programming language will be used to implement the system.
* TCP will be used as the primary protocol by which media will be transmitted.
* SSL encryption will most likely be used in the sending of data to ensure its security.
* Netty will be used to provide an asynchronous event-driven network application framework for communication between client and server.

4 Functional requirements

4.1 Introduction

The system requires that, in order to make use of the program, the users must have downloaded the program/application or acquired it by some other means such as copying the program from another user, etc.; essentially, to use it, the user must be in possession of it.

4.2 Use case prioritization

Critical: The system must be able to recognise a new user, connect the user to the server, and allow users to stream audio and/or video data to other users through the server.

Important: The system should allow the users to be able to send text data and/or stream media to several users at once.

Nice-to-have: The system should enforce security by password authentication and protective measures such as SSL encryption.