Play-list Sharing Application Requirements Document

CSC 445 Group Project

May 2, 2018

v1.0

Contents

1 Introduction

1.1 Objective

The purpose of this document serves to document and identify all aspects of the Play-list Sharing Application. This document will serve as a guide to all of the functional, quality, and constraints of this project.

1.2 Scope

The purpose of this project is to create an application to allow multiple clients to join a room created by another user in order to create a play-list of YouTube URLs that will be synchronized to play in parallel to each user.

1.3 Definitions, acronyms and Abbreviations

- **Room**- A group of users all contributing to the same play-list.
- **GUI**- Graphical User Interface, The interface that will allow the user to interact with the server.
- **sync-algorithm**- The algorithm the server will use to sync all of the clients' videos by calculating the rtt between the client and video provider, as well as the rtt between the client and the server. **This needs to be developed further.**

2 Functional Requirements

2.1 Client Side

The following describes the functionality of the client side of the project:

2.1.1 Rooms

The Following describes rooms in the application.

-A user will open a room, by clicking on a "Create New Room" button.

-The user that created the room will become the room owner.

-The room owner will have control of the seeking functionality of the video.

-Other users can join in the room with a limit of [we need to decide this] users.

-The room will include a chat area, where users can chat about the play-list or whatever they would like.

2.1.2 GUI

The following describes the GUI for the client side of the application.

-Users will have an option to view either an embedded player or have an embedded player hidden from their view.

-If a user chooses to have the embedded player shown the embedded player will be displayed in the upper left corner of the GUI

-The clients will be shown a list of the most current up-next queue. This will be shown on the right hand side of the GUI from the top to the bottom

-Directly underneath the embedded player will be a **clearly marked** text field where the user can submit a video URL, or Youtube id to be added to the end of the queue.

-Beneath the url/id text field, will be a clearly marked chat area for the users.

-The area will consist of a large are which will show a collection of the last few messages sent and who they were sent by.

-Below the large area will be a smaller text area where users can type messages they would like to send, with a send button next to it.

2.1.3 Client Engine

The following describes the back end functionality of the client.

-The client will connect to the server with it's host-name and port number it would like to communicate on.

-Once a connection is established the user can either choose to create a new room, or choose to join a room presented by the server.

-If the user chooses to create a new room, they will be allowed to add [#] videos to the queue, to keep the queue small enough for incoming clients.

-If the user chooses to join a room, they will be allowed to add videos to the queue immediately.

-Once a new user joins a room a notification message will be presented in the chat area so that all users already in the room know a new user has connected.

-A client joining a new room will join the playback once the next video in the play-list begins. They will not view or hear the video currently in progress.

-A client joining in the middle of a video will be shown a message that displays the amount of time until the next video will start.

-A Time stamp will be sent to the server every half second from each client in a room to ensure that the video is synced for all users in the room.

-Each client will send a ready acknowledgment when the current video ends so that the server knows when to issue the start next video command.

2.2 Server Side

2.2.1 Server UI

The following describes the front end functionality for the server

-The server will not have a GUI, it will be operated purely from the command line.

2.2.2 Server Engine

The following describes the back end functionality of the server.

-The Server will be multi-threaded, with one thread listening for incoming connection requests.

-Once a connection request is found, a separate thread will be started which will handle all requests from the connected client.

-Once a client tries to connect with the server, the server will acknowledge the connection by presenting the client with a list of available rooms the client can connect to.

-If the client responds with a request to create a new room, the server will create a new room for that user.

-If the client responds with a request to join a room, the server will try to enter the client into the room depending on its availability by using a Compare-AndSet method.

-If the room is unavailable the server will respond with a room is full message to the client and display the current available rooms

-Once a room opens the client's thread will begin listening for incoming URLs or Youtube ids.

-As soon as the server receives a URL, it will try to add the url to a priority queue using a CompareAndSet method.

-If the CompareAndSet fails, the server will retry until the CompareAndSet is successful.

-Anytime the priority queue is changed, the titles of the first 10 entries will be sent to all clients connected to the room.

-When the server receives a time stamp it will use its time-syncing algorithm to send the correct sync time to the clients.

-If a server fails to receive an expected response from a client within a set amount of time, the thread allocated for the client will send a final disconnected packet and the client will be removed from the room.

2.2.3 Server Engine

The following describes the back end functionality of the server.

-The server will loop listening on a set port for incoming connections.

-The clients will be set up to access the server on the aforementioned port.

-When a client requests to open a new room, a thread to handle the client will be initialized, the thread will have access to all of the tools in each room.

-Each client's thread will send and receive data packets to/from the client and handle the packets accordingly based on the Opcode in the packet header.