Gabriel Lee | Eunwon Moon | Oscar Kwan | Krystle Bulalakaw

glokkbewm | <https://github.com/glokkbewm/commaudio>

CommAudio

Test Document

COMP4985 – Final Project

Table of Contents

[Introduction 2](#_Toc448381725)

[Test Plan 2](#_Toc448381726)

[Test Coverage 2](#_Toc448381727)

[Test Strategy 2](#_Toc448381728)

[Test Schedule 2](#_Toc448381729)

[Test Cases 2](#_Toc448381730)

[Index 7](#_Toc448381731)

# Introduction

This document explains the testing strategy for the CommAudio project. It covers all the main elements (coverage, strategy, methods), as well as the list of test cases that the project will be tested against.

The focus of the testing will be the core functionalities of the project via manual testing. The testing will be performed by each individual member responsible for their own sections of the project. In addition, the application as a whole will be tested after the development stage is completed.

# Test Plan

This section covers the details of the testing that will be performed.

## Test Coverage

The testing will cover all aspects of the application: GUI, audio streaming, voice P2P, and file transfer. The intended coverage is the core functionalities and the predominately used areas such as: communication using TCP/IP suite and media playback.

## Test Strategy

All the testing will be performed manually. Due to the straightforwardness of the project, unit testing and automated testing will only be “more trouble than its worth”.

## Test Schedule

All testing is to be performed at the end of the development stage. However, each sections of the code are to be tested before being committed to GitHub.

# Test Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test Description | Prerequisite | Test Data | Test Procedure | Expected Results | Pass/Fail |
| Client | | | | | | |
| GUI | | | | | | |
| CG1 | The application loads normally as intended. | N/A | N/A | 1. Run the executable to start the program. | The application loads properly without errors.  Figure 3  Figure 9 |  |
| CG2 | The application looks as intended without graphical issues. | CG1 | N/A | 1. Run the executable to start the program. | The application has no graphical glitches or issues. |  |
| CG3 | The application window scales based on the user input. | CG1 | Resize application window. | 1. Click-and-hold bottom right corner of the window. 2. Drag-and-release the mouse. | The application window is able to resize a needed. The elements of the window scale with the window. |  |
| CG4 | Selecting on specific tab opens the desired tab. | CG1 | N/A | 1. Click any one of the tabs (Connect, Playlist, Voice, Settings). 2. Click another tab. | The window should display the user requested tab. |  |
| CG5 | The Connect tab provides proper connection functionality. | CG1 | 127.0.0.1  7000 | 1. Click on the ‘Connect’ tab. 2. Input ‘Socket’. 3. Input ‘IP Address’. | The application properly handles user input. Allows the user to adjust values as needed. |  |
| CG6 | The ‘Connect’ button in the Connect tab performs intended action. | CG1 | N/A | 1. Click on the ‘Connect’ tab. 2. Click ‘Connect’ button. | If no data is not given, the button does nothing. Otherwise, it performs connect logic.  Figure 6 |  |
| CG7 | The Playlist tab displays all the available songs. | CG1 | N/A | 1. Click on the ‘Playlist’ tab. | The list widget should be populated with the available songs.  Figure 8 |  |
| CG8 | The ‘Download’ button functions as described. | CG1  CG7 | N/A | 1. Select the desired song. 2. Click ‘Download’ button. | The application will start the logic for downloading audio (.wav) files from the server. |  |
| CG9 | The “Volume” slider in Settings tab adjusts the playback volume accordingly. | CG1 | N/A | 1. Click-and-drag the slider pointer. | The volume of the playback is adjusted based on the position of the pointer.  Figure 5 |  |
| CG10 | The song title changes based on the current song being played. | CG1 | N/A | 1. Play a song. | The title changes to the title of the current song being played.  Figure 7 |  |
| Audio Streaming | | | | | | |
| CA1 | The application is able to connect to the server without errors. | CG5  CG6 | 127.0.0.1  7000 | 1. Click on the ‘Connect’ tab. 2. Input ‘Socket’. 3. Input ‘IP Address’. 4. Click ‘Connect’ button. | The application connects to the server and immediately begins to stream music. |  |
| CA2 | Change the streaming song to a different song. | CA1 | N/A | 1. Double-click on a song in the list. | The server should stop the current stream and stream the new desired song. |  |
| CA3 | The title of the song changes when a new song is being played. | CA2 | N/A | 1. Select a new song to play. | The client updates the title of the song to the current song title. |  |
| CA4 | The audio playback has little to no delay. Latency and jitter is to a dismissible minimum. | CA1 | N/A | 1. Play a song. | The client is able to play the streamed song ‘flawlessly’. |  |
| Voice P2P | | | | | | |
| CV1 | The client can communicate with other clients via P2P. | CG1 | N/A | 1. Select a client from the user list. 2. Click “Chat” to start sending audio. | The user is able to select other clients and communicate between them via P2P.  Figure 4 | Fail |
| CV2 | The client can receive audio data from other clients via P2P. | CG1 | N/A | 1. Run the application. | The user is able to receive audio data from other clients via P2P and chat with them.  Figure 4 | Fail |
| File Transfer | | | | | | |
| CF1 | The client is able to request a song from the server for download. | CA1 | N/A | 1. Select a desired song. 2. Click ‘Download’ button. | The client will request a file for download from the server. |  |
| CF2 | The request file is downloaded without corruption or other issues. | CF1 | N/A | 1. Run the downloaded file via an existing .wav file player. | The downloaded file should be exactly like the original. |  |
| CF3 | The requested file is the correct file that was requested. | CF1 | N/A | 1. Run the downloaded file via an existing .wav file player. | The downloaded file should be the file that the client requested. |  |
| Server | | | | | | |
| GUI | | | | | | |
| SG1 | The application loads normally as intended. | N/A | N/A | 1. Run the executable to start the program. | The application loads properly without errors.  Figure 3  Figure 9 |  |
| SG2 | The application looks as intended without graphical issues. | CG1 | N/A | 1. Run the executable to start the program. | The application has no graphical glitches or issues. |  |
| SG3 | The application window scales based on the user input. | CG1 | Resize application window. | 1. Click-and-hold bottom right corner of the window. 2. Drag-and-release the mouse. | The application window is able to resize a needed. The elements of the window scale with the window. |  |
| SG4 | Selecting on specific tab opens the desired tab. | CG1 | N/A | 1. Click any one of the tabs (Connect, Playlist, Voice, Settings). 2. Click another tab. | The window should display the user requested tab. |  |
| SG5 | The Connect tab provides proper connection functionality. | SG1 | 127.0.0.1  7000 | 1. Click on the ‘Connect’ tab. 2. Input ‘Socket’. 3. Input ‘IP Address’. | The application properly handles user input. Allows the user to adjust values as needed. |  |
| SG6 | The ‘Connect’ button in the Connect tab performs intended action. | SG1 | N/A | 1. Click on the ‘Connect’ tab. 2. Click ‘Connect’ button. | If no data is not given, the button does nothing. Otherwise, it performs connect logic.  Figure 6 |  |
| SG7 | The Playlist tab displays all the available songs. | CG1 | N/A | 1. Click on the ‘Playlist’ tab. | The list widget should be populated with the available songs.  Figure 8 |  |
| Audio Streaming | | | | | | |
| SA1 | The application is able to start listening for clients. | SG1 | 127.0.0.1  7000 | 1. Click on the ‘Connect’ tab. 2. Input ‘Socket’. 3. Input ‘IP Address’.   Click ‘Connect’ button. | The application starts listening for clients to send song data. |  |
| SA2 | The application handles new song requests. | SA1 | N/A | 1. The client requests a new song. | The server stops the current song and begins playing (transmitting) the new, requested song. |  |
| SA3 | The title of the song changes when a new song is being played. | SA2 | N/A | 1. The client requests a new song. | The server updates the title of the song to the current song title. All clients are notified.  Figure 7 |  |
| SA4 | The audio playback has little to no delay. Latency and jitter is to a dismissible minimum. | SA1 | N/A | 1. Play a song. | The server is able to play the streamed song ‘flawlessly’. |  |
| File Transfer | | | | | | |
| SF1 | The server is able to handle song requests from the client for download. | SA1 | N/A | 1. The client requests a song for download. | The server will handle the request for a file for download to the client. |  |
| SF2 | The requested file is sent without corruption or other issues. | SF1 | N/A | 1. Run the downloaded file via an existing .wav file player on the client side. | The downloaded file should be exactly like the original. |  |
| SF3 | The requested file is the correct file that was requested. | CF1 | N/A | 1. Run the downloaded file via an existing .wav file player on the client side. | The downloaded file should be the file that the client requested. |  |

# Index

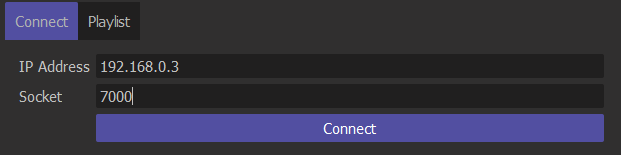


Figure 1

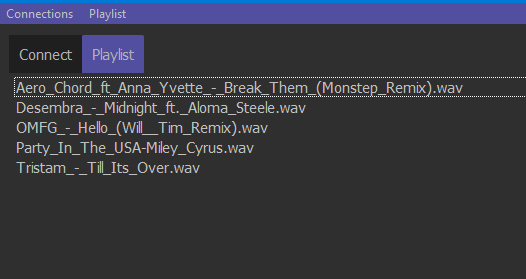


Figure 2

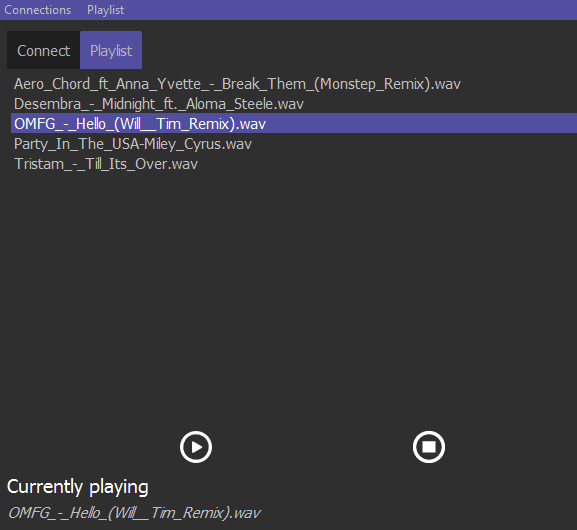


Figure 3

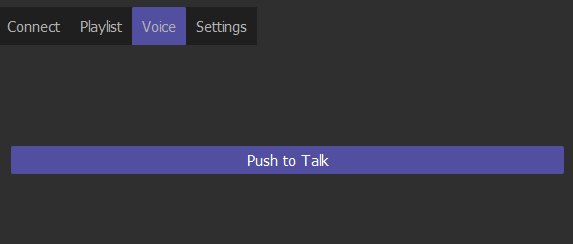


Figure 4

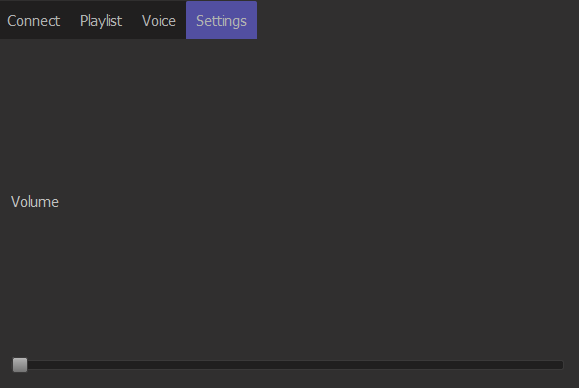


Figure 5

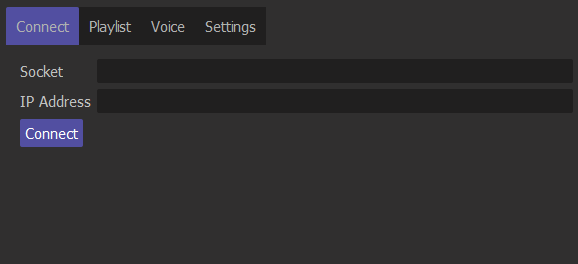


Figure 6

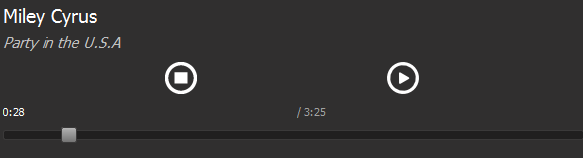


Figure 7

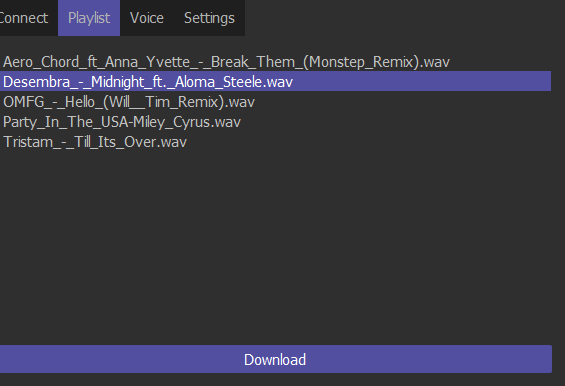


Figure 8

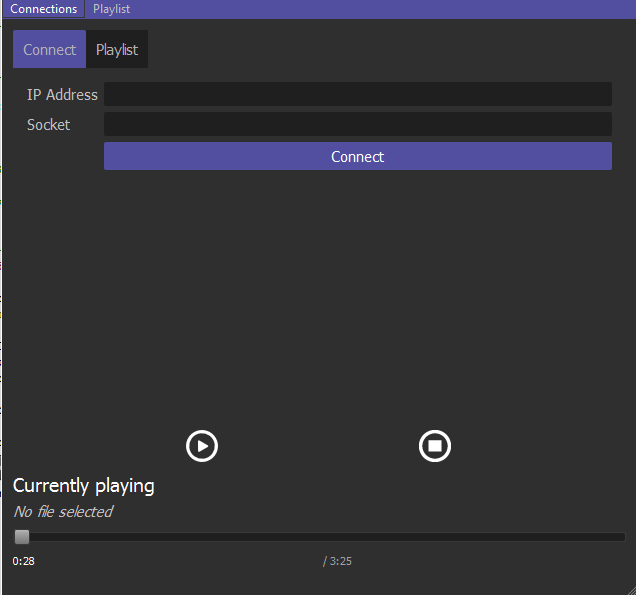


Figure 9