Chanelle Glasgow - 816008723 | Blog Page

Kael Wason - 816009413 | Blog Page

Asa Yee - 815009011 | Blog Page

Markus Hackshaw - 816008387 | Blog Page

Miguel Forde - 816007269 | Blog Page

Table of Contents

Introduction

Problem Statement

There are currently no popular platforms for a reliable way to listen to and watch media synchronously. Activities that can be done together online are mostly limited to video games and communication and most platforms like Twitch and YouTube offer the ability to stream but this is not the same as it is usually delayed for all but the host depending on their respective latencies.

Scope

SyncByte is a desktop application that promises to provide a reliable way for it's users to stream and watch local and online videos synchronously, as well as listen to music synchronously with their friends and family.

Purpose

This document was created to serve as a resource for the SyncByte project to guide stakeholders and developers toward building the software.

Product Description

Product Functions

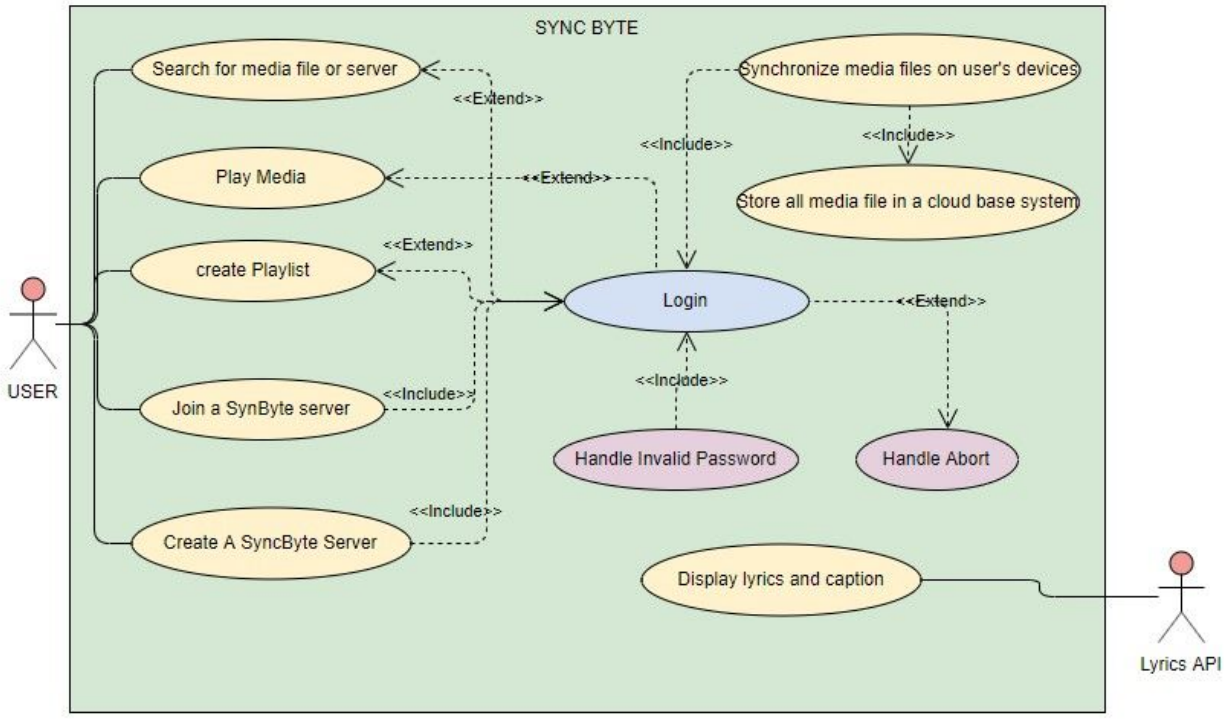
The product will primarily serve as a synchronous media streaming platform. This means that users will be able to open their own video, audio, photo, or other media stream and share it across their own devices or any other device that they allow a connection to. It will also be able to function as a media server across both local and external networks, which enables the sharing of files across devices.

User Stories

1. User plays a song
    a. User navigates to their library
    b. User selects a file to play from their library
    c. System loads a song and plays it
2. User creates a channel
    a. User selects the "create channel" option
    b. System requests a channel name
    c. User inputs a name for the channel
    d. System creates a new channel with the specified name
3. User streams content to another device (via channel)
    a. User creates a new channel on device #1 (see "User creates a channel")
    b. User plays a song on the channel, from device #1
    c. User opens the application on device #2 and enters the newly created channel
    d. System prompts device #2 to select which device to play the song on
    e. User selects "Playback on device #2" option
    f. System stops playback on device #1
    g. System resumes playback on device #2 from where device #1 left off
4. System automatically synchronizes files (after adding a new file)
    a. User selects a new file to add to their library
    b. System adds the file to the local database
    c. System pushes changes to the master database
5. User shares a file
    a. User selects a file's options
    b. User selects "share file"
    c. A list of the user's friends appears
    d. User selects friend to share to
    e. System sends the file to a temporary storage area in the selected user's files
6. User creates a playlist
    a. User selects "create playlist"
    b. System shows the user's library of media

    c. User selects the media they would like to add to the playlist, sequentially
    d. User selects "done"
    e. System creates a new playlist with the selected media
7. System automatically displays lyrics
    a. User plays a song
    b. System detects song
    c. System generates lyrics for the song
    d. System displays lyrics to the user

Use Case Diagram



System Requirements

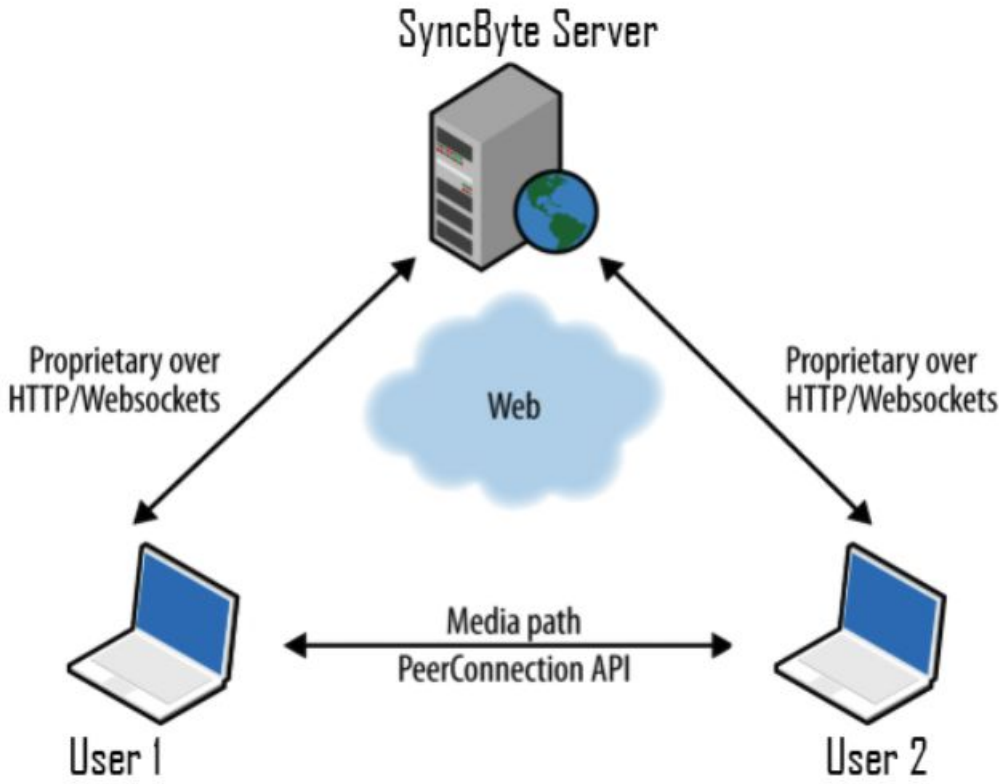
- The user should be able to playback songs.
- Every song should be accessed on any user's device using a shared folder.
- The user should be provided with the ability to host songs with other users over the internet.
- The user should be able to create playlists and host those playlists with other users.
- Every song should have its appropriate lyrics.

User Requirements

1) Users must be able to access, upload, delete and manipulate stored music and files from any supported device.
2) Users should be able to sync music or videos from any device.
3) Users should be able to connect to another user's server over the internet and listen to or watch the media being hosted if given permission.
4) Users should be able to create accounts to access main features.
5) Users should be able to log into accounts they have made.
6) The system must be able to playback media the user selects.
7) Users should be able to search for songs and playlists by name.
8) The application must support files of the Video encoding Format and Audio Format found on https://www.encoding.com/formats/ .
9) The system must allow the user to control the volume through the use of a volume slider.
10) The system must allow for the creation, deletion, syncing, viewing and playback of playlists.
11) The user must be able to add songs to an already existing playlist.
12) The system must allow the user to fast forward, rewind, skip or pause a song.
13) The system should provide a lyric integration function to display auto-generated lyrics for a song.

Architecture

The application will utilize a web architecture that involves a peer connection api similar to that of discord.



Since the application is built using Electron as it's front end. The framework WebRTC can easily be implemented into the chromium system.

General Log of Assigned Tasks

Task	Person	Date	Misc
Develop Introduction	Bruce	03/10/2019	
Create user stories	Asa	03/10/2019	
Design use cases	Asa	03/10/2019	
Document System Requirements	Miguel	03/10/2019	
Document user requirements	Marcus	03/10/2019	
Design Logo	Bruce	03/10/2019	
Review work started on 03/10/2019	Bruce + Chanelle	04/10/2019	
Design UML Diagram	Chanelle	05/10/2019	The UML Diagram was designed based on the User Stories and Use Cases