COMP 3059 – Capstone Project I

Software Requirements Analysis and Design Assignment

This assignment is an overview to gather the software needs with requirements analysis and help to proceed with the design.

The requirements analysis helps to break down functional and non-functional requirements to a basic design view to provide a clear system development process framework. It involves various entities, including business, stakeholders and technology requirements.

The design is the activity following requirements specification and before programming. Software design usually involves problem solving and planning a software solution.

To work on this assignment you could use the references and a sample template given below. The sample template can be customised to suit the nature of your project.

Reference Readings/Example:
http://www.uacg.bg/filebank/acadstaff/userfiles/publ_bg_397_SDP_activities_and_steps.pdf

www.cse.msu.edu/~chengb/RE-491/Papers/SRSExample-webapp.doc

Reference template:
www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc

- **1.0 Introduction**

The Introduction section provides an overview of the system using software requirements analysis and design for the scope of the system.

1.1 Purpose

The motivation of this document is to supply a transparent layout of our project and to pinpoint the necessary conditions for the success of the project. To elaborate, this document characterizes all the functions, features, and interfaces of the system by signaling clearly what the system supports and how the user can gain from that. Also, this document will contain insight about the underlying layers of the system. In addition, this document can boost our group in constructing a mature developing path that group members can pursue and therefore accomplish our goals.

1.2 Scope

Our application is a creation of a contemporary mobile application with an online streaming service.

The mobile application will be running on mobile platforms such as Android and IOS which includes offline functionalities (local playback) and is the client side for the streaming service.

The streaming system will be the backend for the online functionalities where it will provide the songs which are streamed to the client. The system will also convert songs uploaded into multiple formats (320 vs 128kbps) for quality selection and store these files. The streaming service is where users can register and gain online functionalities of the music player application which include streaming music and uploading music to the remote system.

The streaming service also provides recommendations to the user based on their preferences. The relevant goal is that it aims to provide an avenue for registered users to host their content online, enabling independent content creators to advertise their products (music/podcasts). With taking in consideration of the flaws of our competition such as features that are both new and niche, we will seek to gain users that are not within their normal target demographic.

2.0 System Overview

The System Overview section introduces the system context and design.

2.1 Project Perspective

The outcome of this project addresses the business case by delivering completely new systems called the Music Streaming System (MSS) with the Mobile Player Application (MPA) which are not intended to be replacements nor extensions of existing systems. While the systems produced (MSS and MPA) are technically separate systems, they are intended to interact with one another.

2.2 System Context

The outcomes of this project although not intended for monetization will create a base platform which can be expanded upon in later projects. These products are created in response to the flaws within the competition’s products

such as Spotify, Soundcloud, and Tidal. The platforms of the competitors lack features such as quality options for people who appreciate Hi-Fi audio or limit the outreach of independent artists such as Spotify's TuneCore program or Soundcloud's 3-hour upload limit. We can implement features to address these issues in our products to gain a competitive edge, and in doing so we can potentially gain an untapped audience of independent artists.

2.3 General Constraints

There are several constraints of the project that influences the overall product. In terms of business constraints there are time, cost, and possibly legal constraints. The time constraints are caused by the strict schedule for the project (Sep 2021 – Apr 2022) meaning that the project is inflexible and unable to change once development has begun, cost constraints are caused by the project being funded directly by the development team so any software costs or resources for servers must be greatly evaluated beforehand and finally legal constraints may be caused by DMCA holders issuing copyright claims over content uploaded by users of the platform. To handle DMCA issues we have created a solution for disclaimers claiming that we are not responsible for what users upload and any content claimed will be taken down.

In terms of system constraints there are bandwidth and performance constraints. Both constraints may be caused by outsourcing the hosted servers remotely and these servers being unable to handle the demand created by users.

2.4 Assumptions and Dependencies

Team members will have the resources and software development tools needed to complete the project.

The scope of the project will not change as the schedule is incredibly strict.

Software costs will be low or non-existent as development tools may be provided by the school or used in other classes.

The project is dependent on outsourced servers, where if the servers are down the products will be unavailable.

○ 3.0 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

- **3.1.1 <Functional Requirement or Feature #1>**
- **<Registration>**

- **Introduction** - the registration function is used to enable the user to register their personal account into the application database
- **Inputs** - the inputs will be the user email address and password.
- **Processing -** the inputs will be read and the application will send an email to the user for email verification confirmation
- **Outputs** - once the user confirms the email verification in their mailbox, the user is registered into the application database

- **3.1.2 <Functional Requirement or Feature #2>**
- **<Login>**

- **Introduction** - the login function is used to enable the user to login to the application so that they may login to access and gain online funcitonaties of the music player application
- **Inputs** - the inputs used to login will be the user email address and password.
- **Processing -** the inputs will be read and the application will confirm that there is an account registered with those details that have just been read
- **Outputs** - user successfully logs into the application

- **3.1.3 <Functional Requirement or Feature #3>**
- **<User Upload Content>**

- **Introduction** - the user upload content function is used to allow the registered users to transfer their content onto the application
- **Inputs** - the inputs taken will be file types typically used for music (e.g. mp3, aac, flac, wav, and possibly more) and podcasts (mp3 and m4a)
- **Processing -** the inputs will be read and the application will determine if the the files types are valid and if they are allows them to be uploaded
- **Outputs** - user content is uploaded and saved in the database

- **3.1.4 <Functional Requirement or Feature #4>**
- **<User Delete Content>**

- **Introduction** - the user delete content function allows registered users to remove their content on the application registered to their account
- **Inputs** - the inputs taken will be the user touching the remove button below their specific content for deletion of specific content and then a confirmation alert to ask again if they're certain they want to delete
- **Processing -** the inputs will be read and the application will understand the user wishes to remove this specific content
- **Outputs** - application deletes a specific content of the user

- **3.1.5 <Functional Requirement or Feature #5>**
- **<Convert Audio Quality>**

- **Introduction** - enables registered users to change the audio quality of uploaded song files into multiple formats (320 vs 128kbps) for quality selection and store these files.

- **Inputs** - user uploads the desired file with an appropriate file type for audio (e.g. mp3, aac, flac, wav, and possibly more)
- **Processing -** the inputs will be read and the application will understand the user wishes to remove this specific content
- **Outputs** - application deletes a specific content of the user

    - **3.1.5 <Functional Requirement or Feature #6>**
    - **<Stream Music/Podcast>**

- **Introduction** - enables registered users to change the audio quality of uploaded song files into multiple formats (320 vs 128kbps) for quality selection and store these files.
- **Inputs** - user uploads the desired file with an appropriate file type for audio (e.g. mp3, aac, flac, wav, and possibly more) and podcasts (m4a, mp3)
- **Processing -** the inputs will be read and the application will understand the user wishes to remove this specific content
- **Outputs** - application deletes a specific content of the user

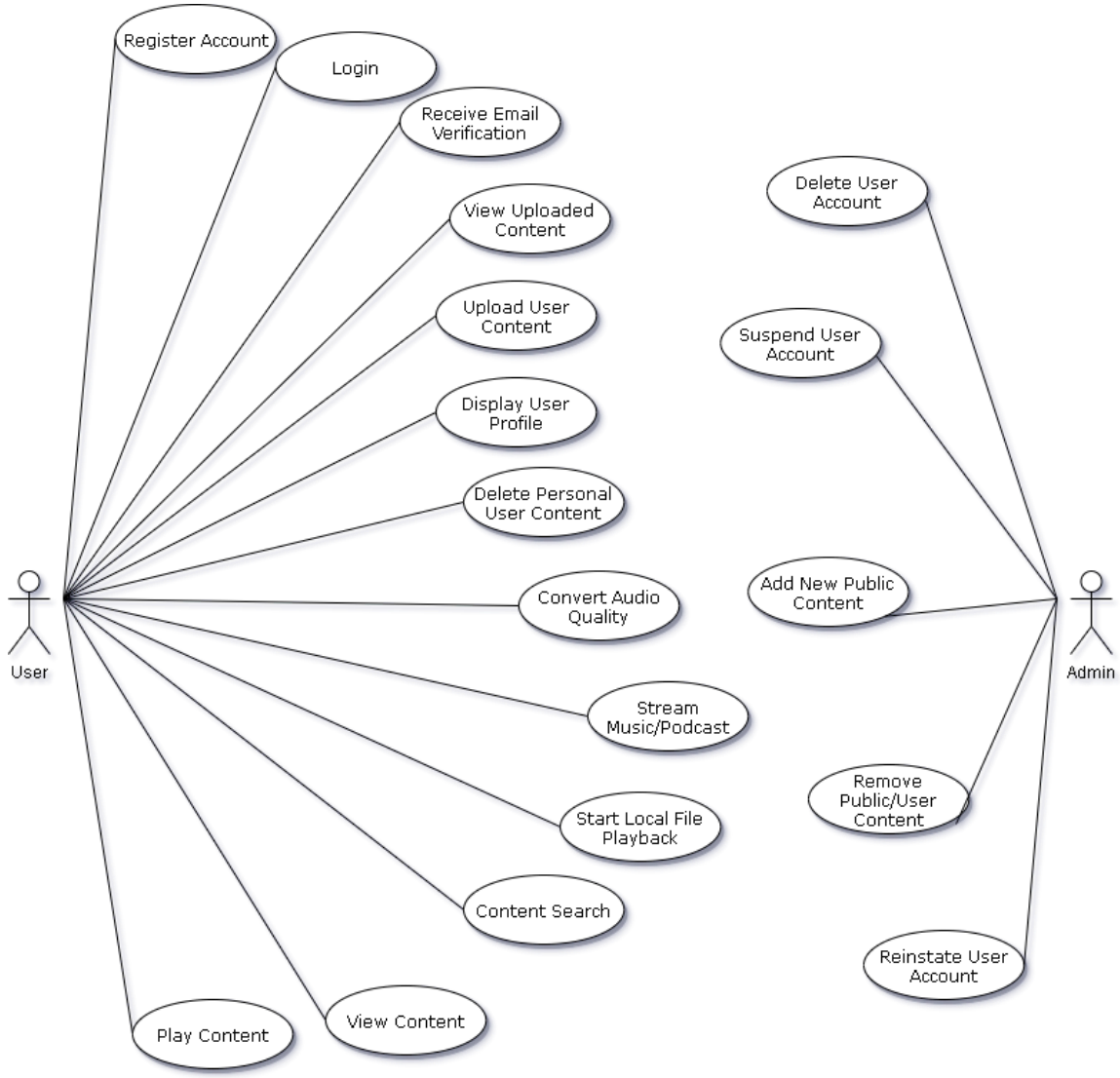
    - **<Start Local File Playback>**

- **Introduction** - enables registered users to play their personal local files
- **Inputs** - user taps selected song or podcast file
- **Processing -** the inputs will be read and the application will understand the user wishes to play the tapped file
- **Outputs** - application starts to play local audio or podcast file

    - **3.1.5 <Functional Requirement or Feature #7>**
    - **<Content Search>**

- **Introduction** - enables registered users to use a keyword to find matches within the database and to display it
- **Inputs** - user inputs keyword they wish to find
- **Processing -** the inputs will be read and the application will find matches of the keyword input
- **Outputs** - application displays results that match user input keyword

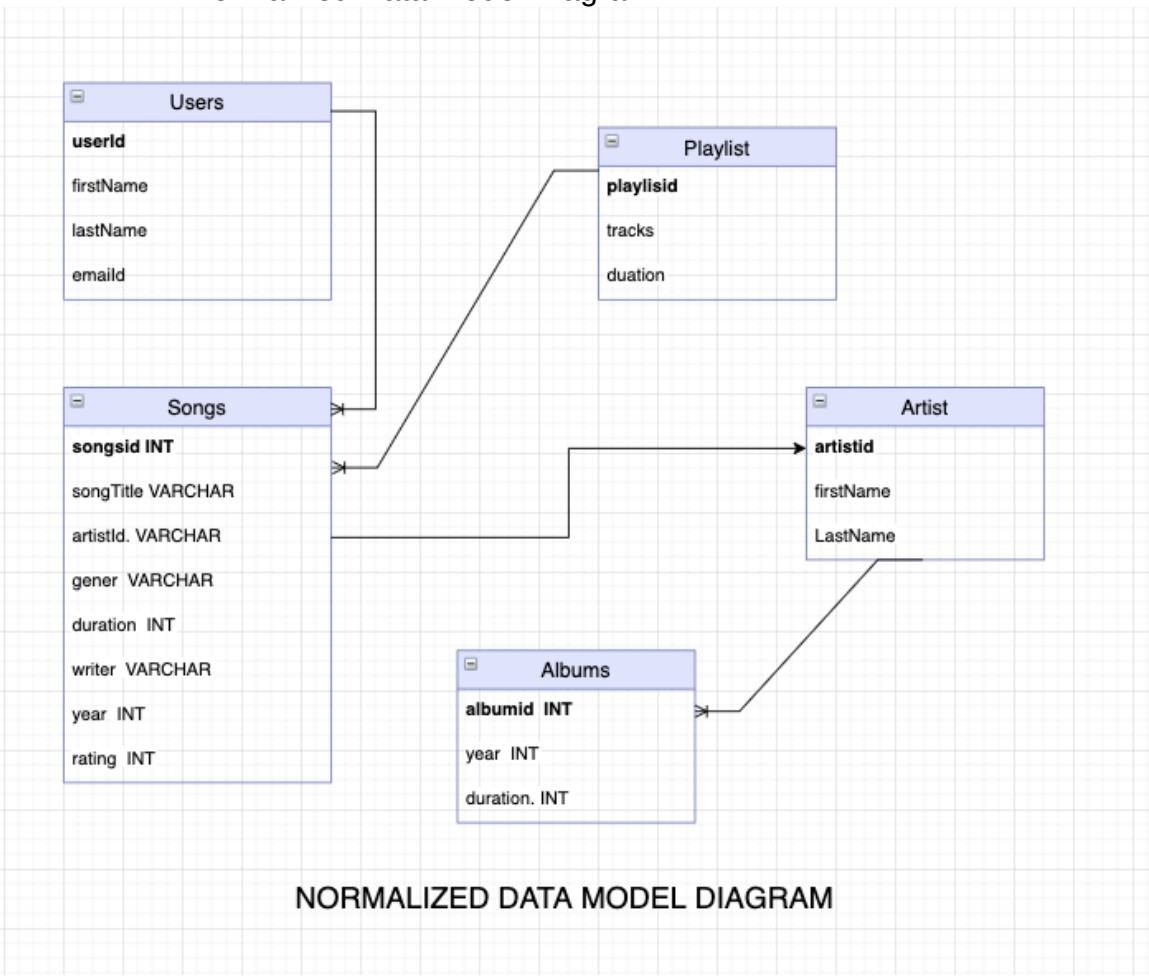
- **3.2 Use Cases**


Register Account
Login
Receive Email Verification
View Uploaded Content
Upload User Content
Display User Profile
Delete Personal User Content
Convert Audio Quality
Stream Music/Podcast
Start Local File Playback
Content Search
Play Content
View Content
Delete User Account
Suspend User Account
Add New Public Content
Remove Public/User Content
Reinstate User Account
User
Admin

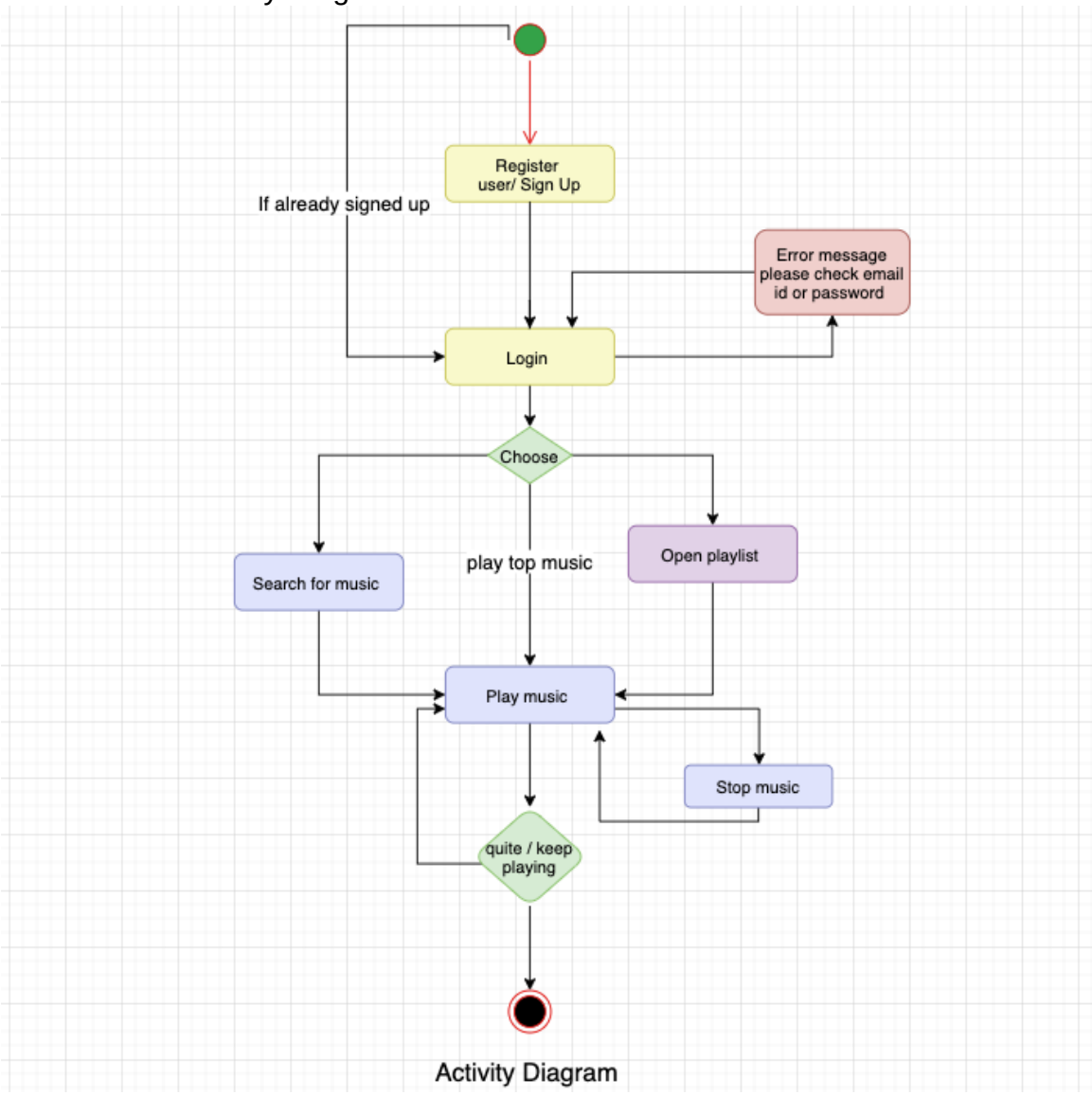
■

3.3 Data Modelling and Analysis

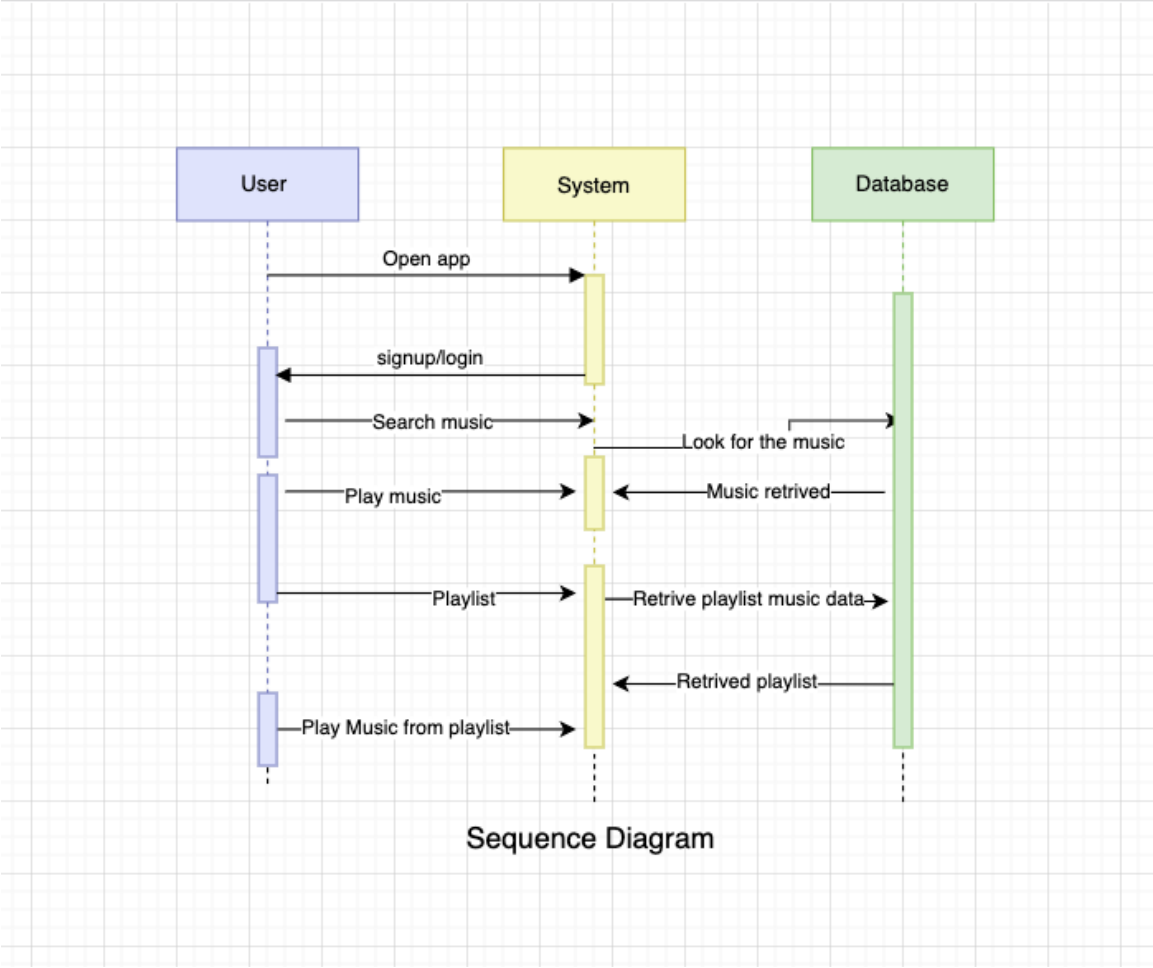
- Normalized Data Model Diagram



- Activity Diagrams

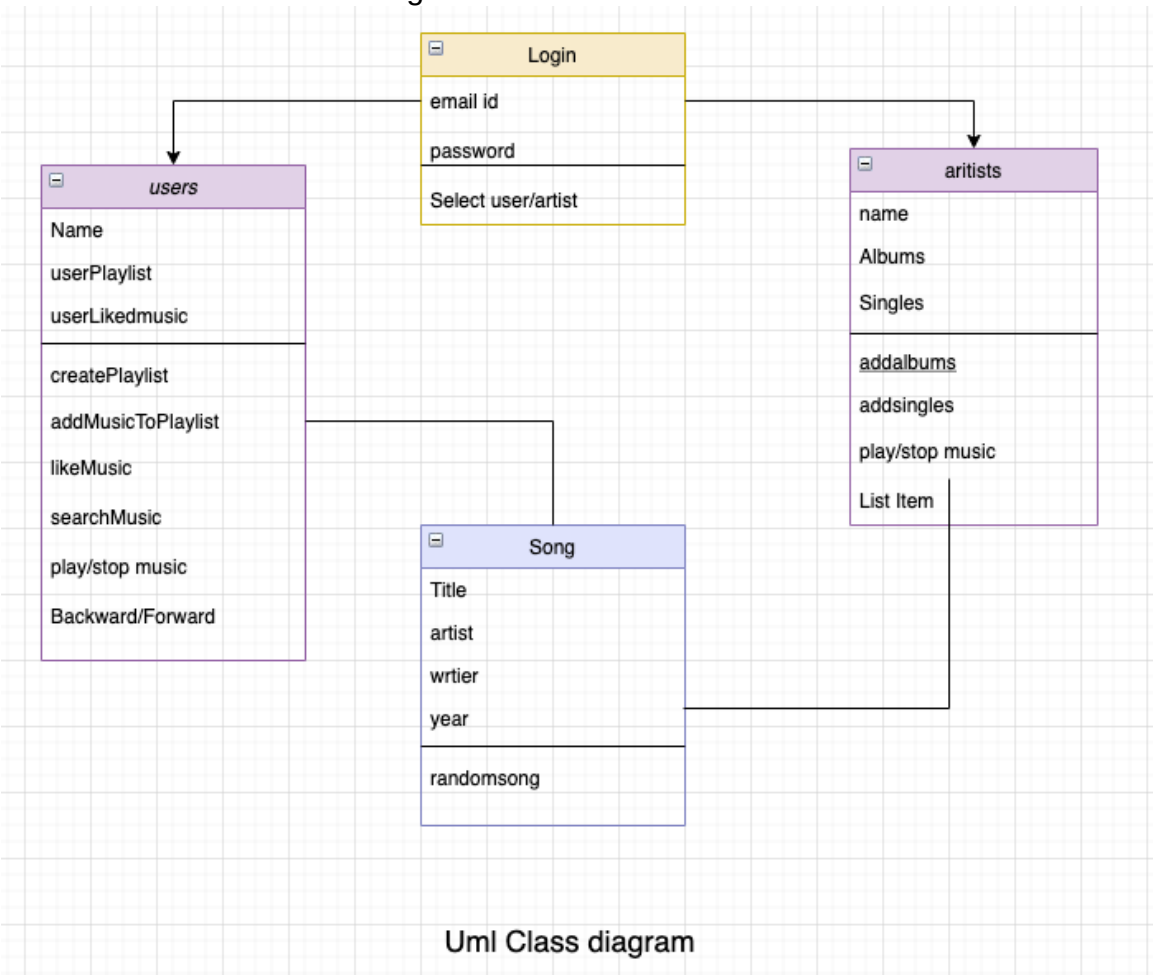


- Sequence Diagrams



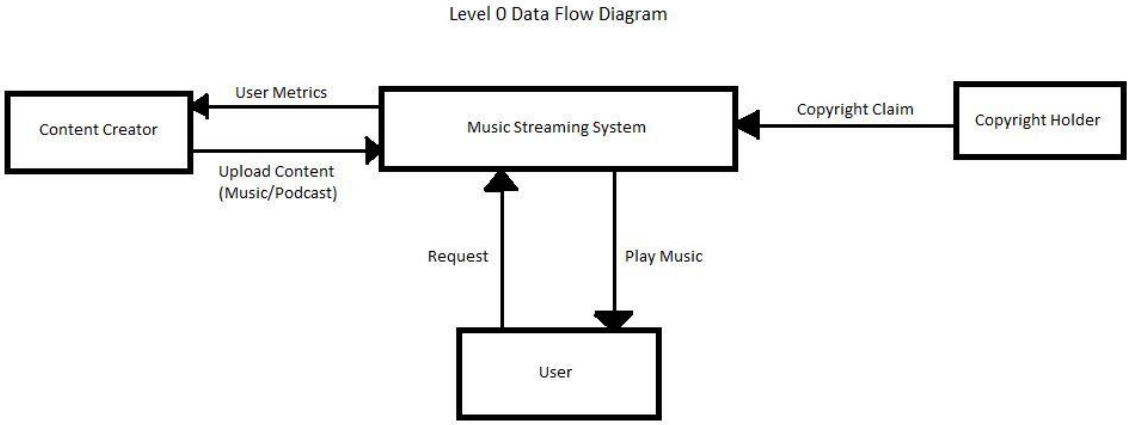
Sequence Diagram

- UML Class Diagram



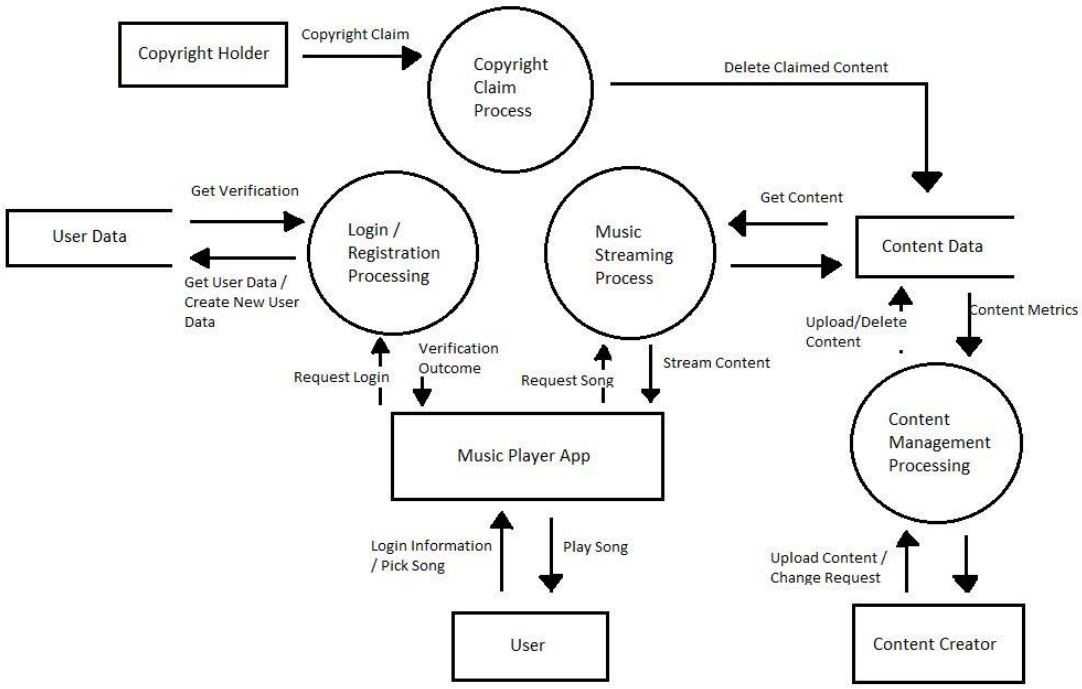
3.4 Process Modelling

- Level 0 Data Flow Diagram



- Level 1 Data Flow Diagram

Level 1 Data Flow Diagram



○ 4.0 Non-Functional Requirements

4.1 Performance

- All process should take less than a second to finish

4.2 Reliability

- App is always available to use offline.
- App aims to have online features available 24/7
- If downtime occurs it should not exceed 1 hour

4.3 Availability

- Music app can run in the background with other apps. Unless the other app uses sounds in which case the other app takes priority and music is paused.
- App icon should be displayed at the top bar of the phone to indicate that it is being used.

4.4 Security

- Passwords must have a minimum length of 8. Must contain an uppercase, lowercase, and a number.
- Passwords are displayed as • unless indicated by the user.
- Passwords should be encrypted in the database
- Forgot the password, ask for the user's email and send them a link where they can reset their password.

4.5 Maintainability

- Weekly updates for bugs and fixes
- Infrequent updates for new features

4.6 Portability

- User can gain access to their data by logging in through the app

- **5.0 Logical Database Requirements**

A database would be required for the storage of music files(including music metadata), user information, playlist/album information and queue information. This database would need to support JSON for user and music information. It would also need to be able to store different kinds of music files such as WAV, MP3, AAC, FLAC, ALAC and others. In terms of data retention most data will be kept and studied to further improve the app such as looking for patterns in music choices. Users that choose to delete their account will have their data retained for up to 6 months before getting permanently deleted. However music uploaded by a deleted account will still persist unless requested to be removed. To preserve data integrity we would make sure that all inputs are valid from the user before allowing them to send data, data is again validated before getting into the database, back up data frequently to avoid permanent loss, only allow access of database to trusted colleagues, and keeping an audit trail to keep track of what is changing in the database. For data security encryption of important data such as passwords on the database in case of breach and making sure that database access is restricted only to trusted admin.

-

- **6.0 Other Requirements**

No additional requirements.

7.0 Approval

The signatures below indicate their approval of the contents of this document.

Project Role	Name	Signature	Date
Developer	James Weber De Asis	JDWA	November 10, 2021
Developer	Chi Calvin Nguyen	CCNguyen	November 10, 2021
Developer	AryanLuthra	Aluthra	November 10, 2021
Developer	Simon Ung	SimonU	November 10, 2021