Software Requirements Specification for GenreGuru: subtitle describing software

Team 8 – Rhythm Rangers

Ansel Chen
Muhammad Jawad
Mohamad-Hassan Bahsoun
Matthew Baleanu
Ahmed Al-Hayali

October 11, 2024

Contents

Revision History

Date	Version	Notes
2024-10-11	0.0	Revision 0

1 Purpose of the Project

1.1 User Business

~~Experimentation in music production is a process driven by intuition, i.e., lacking a core systematic structure, limited by a producer's experience and exposure to complex tools and techniques. *GenreGuru* strives to greatly reduce the effort involved in *methodically* attaining exposure to the use of tools and techniques in other songs. *GenreGuru*, by extension, democratizes access to experimentation in music production to less experienced producers, hobbyist musicians, and novices in music production.~~

Discovering new music that aligns with an individual's unique taste can be overwhelming, especially in todays vast music industry. GenreGuru simplifies this process by leveraging a content-based recommendation system. Rather than relying on user ratings or external popularity metrics, GenreGuru analyzes the intrinsic features of each song, such as spectral characteristics, tempo, and key, to identify and suggest tracks that closely match a user's current favorite. This approach empowers casual listeners, hobbyists, and music enthusiasts to effortlessly explore new music that resonates with their preferences, transforming the music discovery into a more engaging experience.

1.2 Goals of the Project

- ~~*featurize* – produce tabular features corresponding to characteristics of input songs;~~
- ~~*recommend* – produce a collection of songs similar to input songs;~~
- ~~*generate* – produce an audio artifact similar to input reference songs.~~

The goal of the project is to develop a fast, accurate, and intuitive music recommendation system that leverages detailed audio features to suggest tracks closely aligned with a listener's unique taste. By analyzing intrinsic song characteristics such as spectral properties, tempo, and key, GenreGuru delivers personalized recommendations through a user-friendly interface that requires minimal learning effort. If successful, this system will enhance personalized music discovery for a broad range of users, fostering deeper engagement with their favorite genres.

2 Stakeholders

2.1 Client

The project is academic in nature, hence has no formal clients beyond the supervisor, who will be consulted periodically to direct project effort.

2.2 Customer

Please refer to 2.3 and 2.4 for the current characterization of candidate customers. Section 2.5 will more succinctly specify archetypal customers after candidate customer interviews are carried out.

2.3 Other Stakeholders

- ~~Subject Matter Experts (SMEs) – *To protect the privacy of our stakeholders, SMEs are comp*~~
  - ~~Music Producers & Sound Engineers (subsequently "producers")~~
    - ~~*Target subject matter knowledge:* description of current process and/or approach used to guide recording artists to explore or experiment with a new sound. For example, "*while recording, if I get an idea, I play*~~
    - ~~*Extent of project involvement:* minimal, i.e., no more than three interviews per producer.~~
    - ~~*Influence on project:* moderate-low – technology-keen producers may be more likely to already have a process into which *GenreGuru* can be integrated, i.e., song featurization can quickly, and in large volumes, summarize music that the producer's target audience listens to, allowing the producer to better tailor their music output. Such a producer's insights can inform and guide development, but at the discretion of the development team.~~
  - ~~Musicians~~
    - ~~*Target subject matter knowledge:* description of current process and/or approach used to generate novel ideas for unrecorded songs or experimenting with different ideas for already recorded songs. *We must be cautious so as to only consider the experimentation component*~~

    * ~~*Extent of project involvement:* minimal, i.e., no more than three interviews per musician.~~
    * ~~*Influence on project:* moderate – like technology-keen producers, we suspect musicians may already have a process into which *GenreGuru* can be integrated, i.e., song recommendation can quickly, and in large volumes, expose the musician to songs with desirable features as they explore how to create their own song. The musician's insights can inform and guide development, but again, at the discretion of the development team.~~

- ~~Music Theorists~~
    * ~~*Target subject matter knowledge:* description of current process and/or approach used to generate novel ideas for composing new songs or experimenting with different ideas for arranging existing songs. *Yet again, we must be cautious so as to only consider the experimer*~~
    * ~~*Extent of project involvement:* minimal, i.e., no more than three interviews per theorist – *though, we currently do not have any candidate musi*~~
    * ~~*Influence on project:* moderate-low – music theorists may already have a process into which *GenreGuru* can be integrated, similar to producers, i.e., song featurization can quickly, and in large volumes, summarize music from a catalogue of songs of interest to identify similarities and differences in their sound properties based on their composition and arrangement. At the discretion of the development team, the music theorist's insights can inform and guide development geared for very musically literate users.~~

- ~~Music Educators~~
    * ~~*Target subject matter knowledge:* description of current process and/or approach used to introduce students to novel music concepts through experimentation or experimenting with different ideas for previously-learned (composite) concepts. *We must be cautious so as to onl*~~
    * ~~*Extent of project involvement:* minimal, i.e., no more than three interviews per teacher – *though, we currently do not have any candidate music*~~
    * ~~*Influence on project:* low – music teachers may already have a process into which *GenreGuru* can be integrated, i.e., song generation can (relatively) quickly, and in (relatively) large~~

~~volumes, produce sound artifacts that introduce novel music concepts or demonstrate alternative use of one or more previously-learned concepts. Like other stakeholders, at the discretion of the development team, the music teacher's insights can inform and guide development geared for *shared* music experimentation settings.~~

- ~~Affiliated corporation staff – *out of scope*~~
  - ~~Label staff – *publishers, marketers, lawyers, & executives*~~
  - ~~Production studio staff – *studio managers, instrument maintainers, & sound designers*~~
- ~~Development team – *exclusively involves team members, so out of scope.*~~
- ~~Maintenance team – *exclusively involves team members, so out of scope.*~~
- ~~Music regulators – *song licensing laws to abide by when acquiring training data is the only ap*~~

Subject Matter Experts (SMEs)
Experts in music and audio analysis who can offer insights into effective recommendation practices while ensuring stakeholder privacy. For a commercial project, compliance with data protection regulations (such as PIPEDA) would be required.

Music Producers & Sound Engineers
These stakeholders can use recommendations to quickly identify songs with similar production qualities or sonic characteristics, aiding in benchmarking and creative decision-making. Their involvement will be minimal (e.g., up to three interviews) but valuable for informing system integration into professional workflows.

Musicians
Musicians can leverage the recommendation system to explore music that aligns with their style or inspires new creative directions. As with producers, their feedback (via a few interviews) will help refine the system's usability and relevance.

Music Theorists
Although their direct involvement is expected to be minimal (e.g., up to three

interviews), theorists can provide insights into how the system might reveal relationships among songs based on shared musical structures or characteristics.

Music Educators
Educators can utilize the recommendation system to expose students to a broader range of music with similar attributes, enhancing learning and creative exploration. Their feedback will help tailor the system for educational purposes.

Affiliated Corporate and Label Staff
These groups (including publishers, marketers, and studio managers) are acknowledged as potential external stakeholders but are considered out of scope for the current academic project.

Development and Maintenance Teams
Both teams consist exclusively of project members and are considered internal; therefore, they are not detailed further in this section.

Music Regulators
Regulatory bodies concerned with song licensing and data usage provide context for compliance when acquiring training data, though these aspects are managed via API documentation and metadata dictionaries.

2.4 Hands-On Users of the Project

The first four stakeholders of section 2.3 are the users of concern. To maximize project reach, we do not distinguish between separate user groups with regards to some characteristics, i.e., experience level in the subject matter or technology, attitude toward technology, and physical location. A user can be any combination of: a beginner, novice, intermediate, advanced, or expert in the subject matter or technology, they may be timid to use technology or a technology fanatic, and they can be located anywhere that is within reach of our service area. What varies between user groups are their relevant responsibilities, outlined below.

- ~~Music Producers & Sound Engineers — Edit, mix, and master live~~

~~& recorded audio; facilitate experimentation with instruments, audio effects, and lyrics.~~

- ~~Musicians – Play instruments and/or sing in live & recorded settings; experiment with instruments and vocals.~~
- ~~Music Theorists – Compose new pieces of music; arrange existing music compositions.~~
- ~~Music Educators – Conduct personal and group instruction sessions to present novel music concepts; reintroduce previously-learned music concepts used in a novel setting; present combinations of previously-learned music concepts.~~
- **Music Producers & Sound Engineers:** Professionals who leverage the recommendation system to quickly identify and integrate similar production techniques and sonic characteristics into their work.
- **Musicians:** Artists who utilize the system as a creative tool to discover new music that resonates with their style and inspires fresh ideas.
- **Music Theorists:** Scholars and analysts who explore musical relationships by examining shared structures and patterns between songs.
- **Music Educators:** Instructors who incorporate the recommendation system to expose students to a diverse range of music, thereby fostering creative learning and experimentation.

2.5 Personas

- Music Producers & Sound Engineers
  - Fictitious name – *Brianna Barboza*
  - Fictitious age – *31*
  - Relevant job – *Accountant*
  - Relevant hobbies – *Disc jockeying*
  - Relevant music genres – *pop & hip-hop*
  - ~~Relevant likes/dislikes – TBD after interviews~~

  - Relevant likes/dislikes – Like tracks that are energetic, well-produced, and dynamic, but dislikes music that is unpolished.
  - Technology attitude – *comfortable using digital tools, but prefers analog when it comes to audio.*

- Musicians
  - Fictitious name – *Luis Braga*
  - Fictitious age – *24*
  - Relevant job – *N/A, studying for a MSc in Chemistry and Biochemistry from UWaterloo*
  - Relevant hobbies – *Breakdancing*
  - Relevant music genres – *Samba & Bossa Nova*
  - ~~Relevant likes/dislikes – TBD after interviews~~
  - Relevant likes/dislikes – Enjoys creative, inspiring tunes with a unique vibe, and avoids repetitive or generic music.
  - Technology attitude – *very proficient, he grew up spending his free time in an internet café before starting university.*

- Music Theorists
  - Fictitious name – *Goran Kodeski*
  - Fictitious age – *31*
  - Relevant job – *Consulting*
  - Relevant hobbies – *Collecting LP vinyl records*
  - Relevant music genres – *Folk & Jazz*
  - ~~Relevant likes/dislikes – TBD after interviews~~
  - Relevant likes/dislikes – Appreciates music with complex structures that challenge my thinking. Stays away from overly simplistic, commercial tracks.
  - Technology attitude – *vehemently anti-digital, owns a flip-phone without a SIM card, and only uses VoIP.*

- Music Educators

- Fictitious name – *Tumanako "Tui" Teka*
- Fictitious age – *44*
- Relevant job – *Music teacher*
- Relevant hobbies – *Swimming in Lake Waikaremoana*
- Relevant music genres – *Pūoro Māori*
- ~~Relevant likes/dislikes – TBD after interviews~~
- Relevant likes/dislikes – Likes culturally rich music, avoids anything that offers little in terms of teaching opportunities.
- Technology attitude – *complete beginner, and he only ever goes to the studio to record something he's performed a few times prior.*

2.6 Priorities Assigned to Users

This section builds on 2.4, appointing *music producers* & *musicians* key users, then music theorists & music educators secondary users. ~~These priorities may change as interviews are conducted and different user groups become more concrete.~~

2.7 User Participation

Further extending 2.4, all users will be notified that they will be involved in no more than 3 interviews as mentioned in the extent of project involvement in 2.3. Should a user be willing to further contribute to the project after three interviews, they will be contacted as sparingly or generously as they outline. Asynchronous communication via e-mails and text are unrestricted, but expected to be within reason and not to cause a disturbance to its recipient.

2.8 Maintenance Users and Service Technicians

The maintenance team exclusively involves the team members, thus is considered out of scope and will not be explored in detail.

3 Mandated Constraints

3.1 Solution Constraints

- ~~The service uses a music dataset~~
  ~~**Rationale:** A dataset for an AI project is necessary as some form of training data must be used in order to train the AI generative, analysis and recommendation systems.~~
  ~~**Fit Criterion:** The machine learning algorithms use a music dataset as the training set.~~

- ~~The service uses a machine learning algorithm in order to generate song recommendations and snippets.~~
  ~~**Rationale:** The general gist of this project is a leveraging of different signal processing and machine learning algorithms in order to provide an end user with a better experience for consuming music. We believe that using a machine learning algorithm for these ends would both be interesting in terms of implementing and training a model, but also practically useful for the end user to provide better recommendations by leveraging training over a very large dataset in order to produce results.~~
  ~~**Fit Criterion:** The recommendation and generation components utilize trained machine learning algorithm.~~

- ~~The service features integration with an existing music streaming provider's platform.~~
  ~~**Rationale:** A music service provider (such as Spotify) would allow the service to bypass the need to have music inputted, instead being able to use references to a track. In addition, through API calls, providers such as spotify already have a large amount of useful labels for invidiual track(s) that can be used as features for the machine learning components of the service.~~
  ~~**Fit Criterion:** The service has components that make uses of features (such as API calls) that belong to a music service provider's platform.~~

- ~~The service uses a server to process the user requests separately from the front end.~~
  ~~**Rationale:** Ideally, our service would use an on-premise deployed~~

~~ubuntu server in order to process the user analysis, recommendation, and generation components of the project, as this would allow a more flexible front end design (such as through a web application).~~
~~**Fit Criterion:** An on-premise server is deployed for the purposes of handling the analysis, generation and recommendations systems separately from the front end.~~

Technology Stack

Description: The system shall be developed using a high-level programming language for efficient core processing and a robust, secure language for backend logic, while the frontend will utilize modern web technologies (e.g., React and CSS) for a responsive, intuitive interface.
Rationale: A high-level language on the backend enables rapid development and efficient handling of critical processing tasks, and modern web technologies on the frontend ensure the interface is user-friendly, maintainable, and secure.
Fit Criterion: The final product shall meet established performance benchmarks for processing tasks and deliver a responsive, secure user interface across supported devices.

Open-Source Libraries

Description: The app shall use open-source libraries and frameworks where applicable to save both time and money.
Rationale: Utilizing open-source libraries is cost-effective and leverages community-driven improvements and support.
Fit Criterion: All third-party libraries used in the project must be licensed under recognized open-source agreements (e.g., MIT, GPL, or Apache) to ensure they are free to use.

Integration with a Music Streaming Provider's Platform

Description: The service shall integrate with an existing music streaming provider's platform.
Rationale: By integrating with a provider (such as Spotify, or Deezer),

the service can use track references and leverage existing metadata—eliminating the need to manually input music—and take advantage of the rich labels that the provider supplies.
Fit Criterion: The service must include components that successfully make API calls to a music streaming provider's platform to retrieve track features.

Server-Based Processing of User Requests

Description: The service shall utilize a dedicated server to process user requests separately from the front end.
Rationale: Deploying an on-premise Ubuntu server for backend processing allows for flexible front-end design (e.g., via a web application) while efficiently handling analysis, recommendations, and other processing tasks.
Fit Criterion: An on-premise server must be deployed to handle the backend processing, ensuring that recommendation functions operate independently from the user interface.

3.2 Implementation Environment of the Current System

Hardware Specifications

Description: The service shall run on a dedicated server with a minimum configuration of 8GB of RAM, a quad-core processor, and 256GB of available storage. The front-end interface will be accessible on standard personal computers or laptops (which include a keyboard, mouse, and speakers).
Rationale: These specifications reflect the typical hardware available to our target users and ensure that both the backend processing and the user interface perform efficiently under normal operating conditions.
Fit Criterion: The system must pass performance tests on devices meeting these specifications.

Software Environment

Description: The backend will be developed using a high-level programming language (e.g., Python) along with open-source libraries for efficient data processing. The frontend will be built with modern web technologies (e.g., React and CSS) to provide a responsive and intuitive user experience.
Rationale: Using a proven, open-source software stack ensures rapid development, ease of maintenance, and robust integration between backend and frontend components while meeting industry standards for security and performance.
Fit Criterion: The final product shall demonstrate seamless integration between backend and frontend components and meet established performance benchmarks in processing speed and user interface responsiveness..

Network and Integration

Description: The service shall operate over a stable internet connection to support real-time API communications with an external music streaming provider's platform for up-to-date track metadata.
Rationale: Reliable network connectivity is essential for maintaining real-time interactions and ensuring the system can access necessary external data sources to support its recommendation functionality.
Fit Criterion: The system must achieve a network uptime of at least 99% during continuous operation and successfully retrieve external data via API calls as verified through integration testing.

3.3 Partner or Collaborative Applications

Interface	Partner	Integration Type	Rationale
Music Streaming Service API	Youtube, Apple Music, Spotify, Last.fm, etc	API	Some form of a music streaming service API would be hugely useful because it would allow users to interact with the service by pointing to references of a song instead of uploading the entire song file.
Music Generation AI	Jukebox (OpenAI), MusicLM (Google), MusicGen (Meta), Suno AI (Suno.inc)	ML/AI Model	Some form of existing AI model that is tweaked would allow the service to be implemented more efficiently.
Payment API	Stripe, Paypal, etc	API	If the service is to be monetized these options need to be considered.
Analytics Services	Google	API	Tracking user analytics to feedback into the system or to improve user experience.
Cloud Services	Google/AWS, etc	API	Cloud services for remote AI computation, database management, user records, etc.

3.4 Off-the-Shelf Software

There are several existing solutions that could serve as part of the music generation and recommendation system. These include:

- **Spotify API**: Provides access to a vast library of music, including song previews and metadata, which can be leveraged for generating recommendations.
- **Librosa Library**: An open-source Python package for analyzing and processing music files, suitable for extracting features from songs and facilitating generative components.
- **TensorFlow and PyTorch Pre-trained Models**: Both frameworks offer pre-trained models that could be adapted for music generation tasks. These solutions provide a basis for deep learning models without having to build and train from scratch.
- ~~**OpenAI Jukebox**: A generative model that is capable of producing music, which could potentially be adapted and integrated into our system.~~

These off-the-shelf software solutions provide a foundation upon which we can build our custom features, significantly reducing the development time and leveraging existing technologies to enhance the functionality of our platform.

3.5 Anticipated Workplace Environment

Home Usage

Rationale: This environment is more appropriate for casual music listeners. This means users will most likely be using it in a form of a casual environment. This can include such as during a public commute on the train, where the user might try to interact with the service while there is an unreliable internet connection on a mobile device. This means the front end of the service must be easy to use and access on all types of devices, and preferrably, not a large amount of data is streamed between the user and the service. This means that preferrably, the input would have references to pieces of music instead of having say, full .MP3 files as the input. These users are also more likely interested in the recommendation system to attempt to find new pieces of music to listen to. ~~, and might be more willing to use the generation system for "fun", so they might be satifisfied even with a generation component that produces odd results.~~

Studio Usage

Rationale: This environment is more professional and is most likely what creative professionals will be using. The expectation is that they will be using this on their computers, thus they are more likely to have a reliable internet connection in addition to being willing to input a large volume of music into the service. This means that the service needs to facilitate entering a large amount of music, and be able to ~~generate results in a rapid and efficient manner~~ produce accurate recommendations quickly. Because the users of this type of environment would likely be individuals engaged in creative endeavors, they will generally have higher standards of what the service provides, and might be more interested in the ~~analysis component, such as being able to find new labels for their existing work to get new perspectives~~ recommendation functionality that efficiently surfaces new musical inspirations. This also means that they most likely expect the ~~music~~

generation component to give them something more "listenable", something they could directly use as a sample or inspiration recommendations that are both relevant and innovative, providing inspiration for creative projects.

3.6 Schedule Constraints

This project was started in the 2024 Fall Term and is expected to be completed by the 2025 winter term at McMaster university. Some of the key deadlines are:

- Proof of Concept Demonstration Plan, November 11-22
  This deadline accounts for 5% of the mark. The group should have identified the most significant risks involving the project and come up with plans on how to migitate said risks. If this is not possible, then the project can be redefined. The group also needs to predict and note other concerns about potential problems or difficulties that could arise during development, such as testing difficulties or software portability. If this deadline is not met, then it means that our group does not fully understand the potential pitfalls of our project and we need to redefine certain aspects of the project.

- Final Demonstration, March 24-30
  This deadline accounts for 20% of the mark. This deadline involves a demonstration of a finalized version of the project to supervisors before the public EXPO happening at a TBD date. By this deadline, the service should be in a completed state ready for use and demonstration. If this deadline is not met then the project can be considered a "failure".

- Final Documentation, April 2
  This deadline accounts for 30% of the mark. It involves the finalized documentation of plans pertaining to the project and the actual working program/service. Any final revisions and reflections pertaining to the project should have been completed by this deadline as no further changes to the project will be possible. Whatever documentation or code that is not completed by this deadline would be considered permanently unfinished.

Schedule constraints are provided by the capstone course itself, for a detailed schedule refer to the Development Plan Document, section 8.

3.7 Budget Constraints

The budget limit as stated by the capstone outline is 750$ CAD. Potential additional costs might include API calls, software liscensing, account fees for cloud services. For the purposes of the demonstration they should not exceed the 750$ limit.

3.8 Enterprise Constraints

There are no specific enterprise constraints as we do not have outside investors for this project.

4 Naming Conventions and Terminology

4.1 Glossary of All Terms, Including Acronyms, Used by Stakeholders involved in the Project

- ~~Track is full song, i.e., a completed audio file published by a musician.~~
- ~~A snippet is a fragment of a track, i.e., an incomplete audio file. Such audio files can be in a *lossy* compressed format, e.g., `.mp3`, a *lossless* compressed format, e.g., `.FLAC`, or an uncompressed format, e.g., `.WAV`. While the file formats of audio files are known, the source of a song or song snippet can be anything. A concrete example is a 30-second snippet from Spotify's API being an MP3 file delivered through the Spotify content delivery network, `scdn`, e.g., Jack Harlow's "First Class".~~
- ~~Model: Statistical Model (Not software module)~~
- ~~Module: Software Component~~
- ~~AI: Artificial Intelligence~~
- ~~ML: Machine Learning Model~~
- ~~Music Metadata: labels obtained through streaming music API (eg spotify) that are like features (eg genre, 'danceability')~~

Table 1: Glossary of All Terms, Including Acronyms

Term	Definition
Track	A full song, i.e., a completed audio file published by a musician.
Snippet	A fragment of a track, i.e., an incomplete audio file. Such audio files can be in a lossy compressed format (e.g., .mp3), a lossless compressed format (e.g., .FLAC), or an uncompressed format (e.g., .WAV). The source can be anything (e.g., a 30-second snippet from Spotify's API; Jack Harlow's "First Class").
Model	Refers to a statistical model (not a software module).
Module	A software component (e.g., a self-contained piece of the system's codebase).
AI	Artificial Intelligence.
ML	Machine Learning model.
Music Metadata	Labels obtained through a music streaming API (e.g., Spotify) that act as features (e.g., genre, "danceability").

5 Relevant Facts And Assumptions

5.1 Relevant Facts

Music contains the following core features:

- Tempo
- Key signature
- Time signature
- Pitch
- Timbre

Song files from streaming services (e.g., Spotify, Deezer) include metadata such as:

- Track ID

- ISRC
- Album name
- Artist name
- Track name

Most songs can be classified into multiple genres.

5.2 Business Rules

- ~~The user should be able to generate their own music~~
- ~~The user should be able to figure out what musical features a song contains~~
- The user should be able to ask for similar songs based on features
- the user should be able to interact with the system without any external installation
- The system shall deliver recommendations in real time, ensuring that the user's interaction feels imeediate and responsive

5.3 Assumptions

- Users will have at least some familiarity of music theory
- The ~~analysis and~~ recommendation system~~s~~ will use as many well-established musical features as possible
- All API inputs will be easily accessible and reliable enough to support the recommendation ~~and analysis~~ system~~s~~
- The system will be written in a language that all developers are familiar with
- The system will use a local server to handle the processing of the machine learning model and large datasets

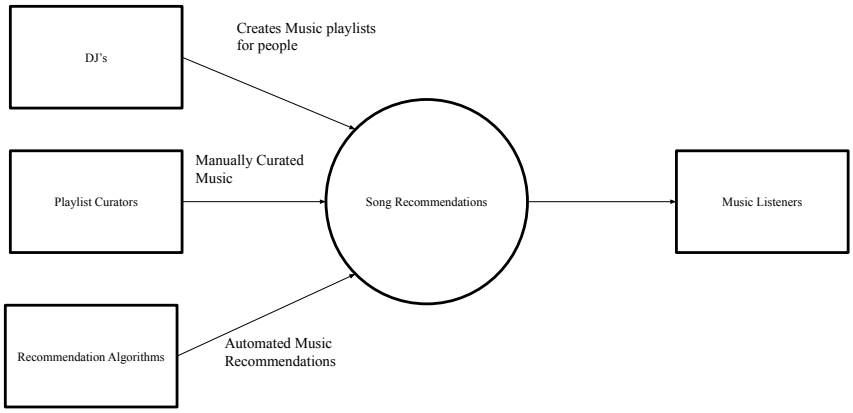
- ~~Handling of niche features and cover art are designed to enhance the user experience, but these will not be a part of the core functionality of the system~~
- ~~The generative system will be completed by the POC demo date~~
- The recommendation ~~and analysis~~ system~~s~~ will be completed by the Revision 0 date

6 The Scope of the Work

6.1 The Current Situation

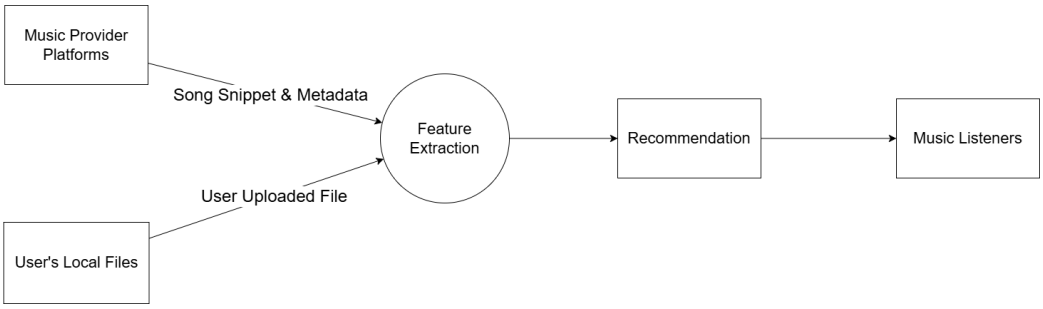
~~**Music Analysis Component**~~
~~**Manual Process**: Currently, this is usually manually done by the artist self-labelling their work or third parties labelling the music. This can end up being time consuming.~~
~~**Automated Process**: Most automation on this regard (retrieving song information from a music provider's api) is mostly useful for retreiving already labelled information, there is not a lot of automation that is be done for labelling, but a lot that is possible retreiving said labels.~~
~~What we wish to do with this component of the service is to gather and present said information that can be gleamed from the music service provider's API to the user in a more presentable manner (such as audio visualizations) but also featurize an individual song for the other components in our service. Our service should also generate new labels using ML algorithms, which allows for faster automation of song labelling and more features for our other components.~~
Music Recommendation Component
Manual Process: Generally, this is mostly done through music curators should as DJs, playlist makers, etc, where curators manually select songs and make recommendations based on popular trends, new artists releases, or shifts in the genre. The main issue here is that it requires a large amount of manual work, as curators need to listen to new music before adding them to some form of playlist or recommending them to other people, which greatly limits their potential recommendations.
Automated Process: Most music service providers such as spotify, apple music or youtube already offer some form of music recommendation service

use algorithms to recommend users new songs based on their listening history. These generally already work decently well, but they end up usually being heavily weighted towards more popular songs (for example, sabrina carpenter's espresso being automatically recommended through smart shuffle no matter what your prior listening history was like), as record labels often have deals with these music service platforms so it is not unusual to get recommendations that do not necessarily suit the user's music tastes.
The main benefit our service should provide is the ability to recommend songs that are more niche, more accurate based on the user input while still using an automated system powered by machine learning, as we would have more features due to the analysis component for the ML algorithm to process, in addition to not being heavily weighted towards more popular, trendy songs or some form of likely record label deals meddling.



Music Feature Extraction Component
Manual Process: Currently, musical features are often determined manually by artists or third parties labeling the music. This process is time-consuming and can be inconsistent.
Automated Process: Most existing solutions rely on retrieving pre-labeled metadata from music service providers' APIs. However, there is limited automation in extracting detailed musical features directly from the audio.

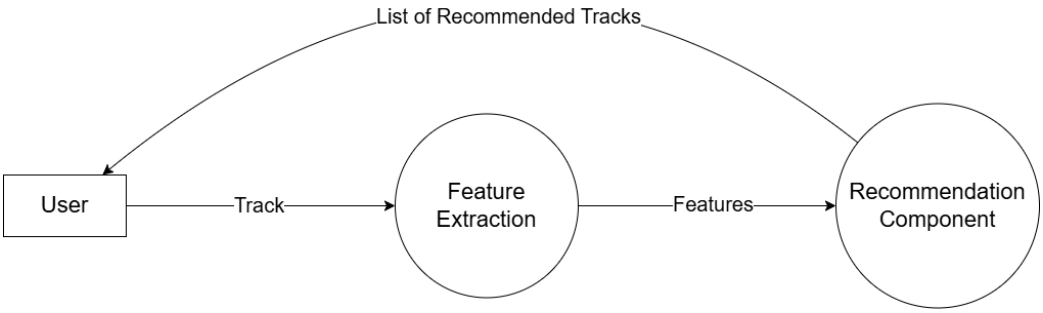
Our service will automate the extraction or featurization of key musical features. By doing so, we will provide users with clear and comprehensive feature data for each song. This feature extraction will serve as the foundation for generating accurate and personalized song recommendations.



~~**Music Generation Component**~~

~~**Manual Process**: Currently, generating music usually requires years of dedication to the craft and involves the labour of musicians, producers, and composers. This means that there is a large amount of manpower that is necessary in order to create new unique pieces. We wish to have our service provide a new avenue to generate new types of music. We do not envision this service as something that replaces the traditional music creation process, but rather as something that can potentially enhance it.~~

6.2 The Context of the Work



6.3 Work Partitioning

Table 2: Event List for Feature Extraction and Recommendation

Event Name	Input/Output	Summary
User Requests Recommendation	Track [input]	The system receives user input track for generating a recommendation.
Features Extracted From Song for Recommendations	Analyzed Track [input]	The system extracts relevant features from the user's input track to inform the recommendation process.
Recommendations Returned to User	List of recommended tracks [output]	The system returns a curated list of recommended tracks based on the extracted features.

6.4 Specifying a Business Use Case (BUC)

- **User Requests Track Recommendations**
  Primary Actor: App User
  Trigger: User intiates a request with the service to ~~generate~~ recommend some new songs to listen to.
  Preconditions:
  - User has a valid reference track
  - ~~User has valid references to tracks for the service to accept~~
  - ~~ML generation components are currently active~~

  Main Success Scenario:
  - ~~user inputs track(s) to the service~~
  - ~~user submits request for track recommendation~~
  - ~~the service's analyzation component analyzes the tracks~~
  - ~~analyzation component passes on tracks & analysis to recommendation component~~
  - user inputs a track to the system
  - the user submits a request for track recommendation
  - the system then extracts features and passes these features to the recommendation component

- recommendation component generates tracks
- user receives new tracks to listen to

- ~~**User Requests Track Analysis**~~
  ~~Primary Actor: App User~~
  ~~Trigger: User intiates a request with the service to generate some new songs to listen to.~~
  ~~Preconditions:~~
  - ~~User has valid references to tracks for the service to accept~~
  - ~~ML generation components are currently active~~

  ~~Main Success Scenario:~~
  - ~~user submits request for track recommendation~~
  - ~~user inputs track(s) to the service~~
  - ~~the service's analyzation component analyzes the tracks~~
  - ~~analyzation component generates presentations and visualizations based on the inputted track(s)~~
  - ~~the service returns generated presentations and visualizations to the user~~

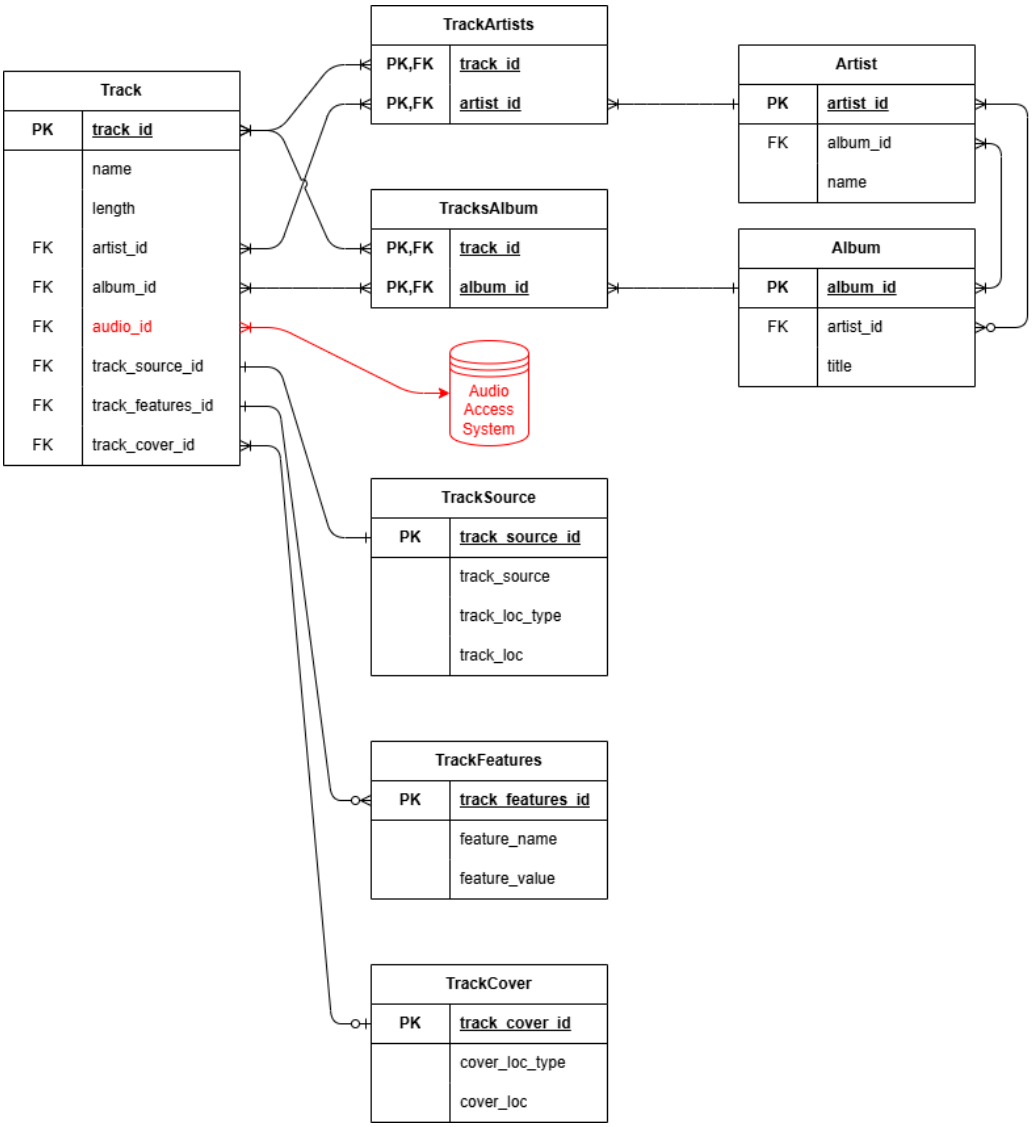
- ~~**User Requests Snippet Generation**~~
  ~~Primary Actor: App User~~
  ~~Trigger: User intiates a request with the service to generate some new songs to listen to.~~
  ~~Preconditions:~~
  - ~~User has valid references to tracks for the service to accept~~
  - ~~ML generation components are currently active~~

  ~~Main Success Scenario:~~
  - ~~user submits request for snippet generation~~
  - ~~user inputs track(s) to the service~~
  - ~~the service's analyzation component analyzes the tracks~~

- ~~analyzation component passes on tracks & analysis to generation component~~
- ~~generation component generates tracks~~
- ~~user receives a snippet to listen to~~

7 Business Data Model and Data Dictionary

7.1 Business Data Model



7.2 Data Dictionary

7.2.1 Relations

Please refer to table 7.2.1.

Relation Name	Relation Description
Track	Primary relation containing track name, track length, & references to other relations.
TrackArtists	Intermediate relation to retain information about which artists appear on which tracks.
TracksAlbum	Intermediate relation to retain information about which tracks appear on which albums, if applicable.
TrackSource	Secondary relation containing information about the source of a track, e.g., Spotify, Deezer, Amazon music, BandCamp, or local.
TrackFeatures	Secondary relation contianing information about the features of a track, following the entity-attribute-value (EAV) model.
TrackCover	Abstract relation containing information about the cover art of the track.
Artist	Secondary relation containing information about artists.
Album	Secondary relation containing information about albums.
Audio Access System	Abstract storage system containing information about the track audio.

Table 3: Relations Data Dictionary

7.2.2 Attributes

Please refer to table 7.2.2.

Attribute Name	Attribute Description
track_id	Unique track identifier.
artist_id	Unique artist identifier.
album_id	Unique album identifier.
audio_id	Unique track audio reference identifier.
track_source_id	Unique track source reference identifier.
track_features_id	Unique track features reference identifier.
track_cover_id	Unique track cover art reference identifier.
Track.name	Track name.
Track.length	Track length.
Artist.name	Artist name.
Album.title	Album title
TrackSource.track_source	Track source name.
TrackSource.track_loc_type	Track source location type, e.g., "URL" or "path".
TrackSource.track_loc	Track location, i.e., a URL or path.
TrackFeatures.feature_name	Track feature name, e.g., popularity.
TrackFeatures.feature_value	Track feature corresponding value, e.g., 88.
TrackCover.cover_loc_type	Track cover art location type, e.g., "URL" or "path".
TrackCover.cover_loc	Track cover art location, i.e., a URL or path.

Table 4: Attributes Data Dictionary

7.2.3 Relationships

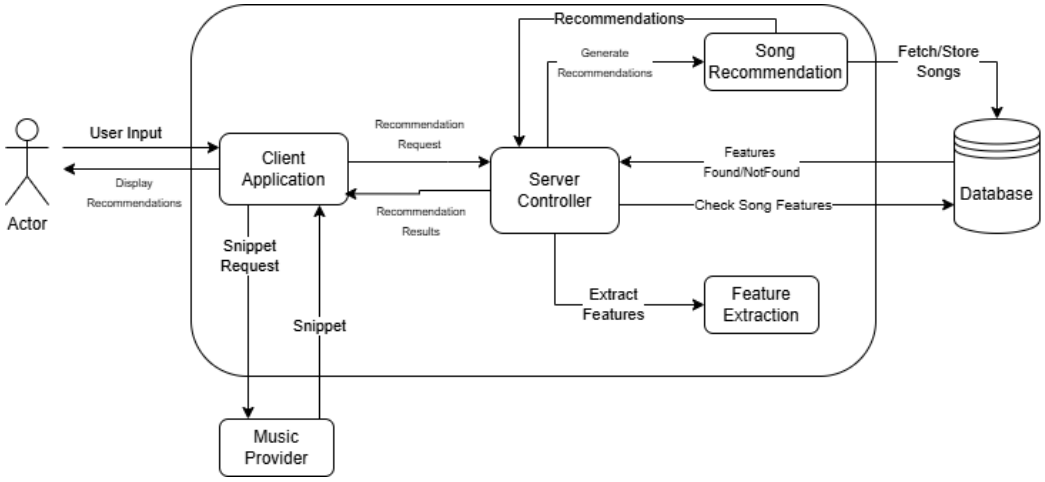
Please refer to table 7.2.3. Please consult Entity Relationship Model - Crow's Foot Notation as a notation reference if necessary - it is mostly useful for the connector multiplicities.

Relation 1	Relation 2	Relationship Description
Track	*TrackArtists*	A track can correspond to multiple artists, and vice versa.
Track	*TracksAlbum*	A track can correspond to multiple albums (a single and a long-playing (LP) album), and vice versa.
Track	Audio Access System	Multiple tracks can correspond to a single instance of an audio file, i.e., a single and an LP release of a song point to the same audio file.
Track	*TrackSource*	A track should only belong to one and only one source to avoid duplication.
Track	*TrackFeatures*	A track can correspond to 0 or more features, i.e., a song with no features has not been featurized and a song with multiple features has. This allows modular and evolvable data storage.
Track	*TrackCover*	A track can correspond to at most one cover, i.e., it can have no cover art. The cover art of an album can also apply to its constitutent tracks.
TrackArtists	*Artist*	A mapping relationship, where an artist can exist on multiple tracks.
TracksAlbum	*Album*	A mapping relationship, where an album can contain multiple tracks.
Artist	*Album*	An artist can have 0 or more albums, but an album must have at least one artist.

Table 5: Relationships Data Dictionary

8 The Scope of the Product

8.1 Product Boundary



8.2 Product Use Case Table

PUC No.	PUC Name	Actor(s)	Input & Output
1	Submit Song Query	User	**Input:** Song name, or .wav entered by the user. **Output:** Query sent from the Client Application to initiate the recommendation process.
2	Retrieve Song Snippet	Client Application, Music Provider	**Input:** Song query from the Client Application. **Output:** Song snippet (e.g., metadata or audio preview) returned from Spotify/Deezer to the client.
3	Process Recommendation Request	Server Controller, Song Recommendation, Feature Extraction	**Input:** Song snippet and query received from the Client Application. **Internal Steps:** - Check if song features exist in the Database. - If not, invoke Feature Extraction (which is handled by the Server Controller). - Call the Recommendation Module (which fetches/stores songs as needed). **Output:** A list of recommended songs.
4	Display Recommendations	Client Application, User	**Input:** Recommended songs from the Server Controller. **Output:** Recommendations displayed to the user.

8.3 Individual Product Use Cases (PUC's)

~~**1. Product Use Case Name:** UI Interaction~~
~~**Trigger:** User commits some action (e.g. clicking, swiping, dragging)~~
~~**Preconditions:** User has successfully accessed GenreGuru, or is already in GenreGuru~~
~~**Interested Stakeholders:** All~~
~~**Actor/s:** User~~
~~**Activity Diagram:**~~
~~**Outcome:** The user will commit an action like swiping or pressing and the~~

system will react depending on the action.

2. Product Use Case Name: Song Recommendation Based on Features
Trigger: User picks features, and indicates they want to search for recommendations
Preconditions: User must have GenreGuru open, the user has selected features to search for
Interested Stakeholders: Casual Music Listeners, Hobbyist Musicians
Actor/s: User
Activity Diagram:
Outcome: The user will select or manually enter features they are looking for in a song, and the system will first check to see if the features they selected/entered are valid, and the system will return a collection of reference songs that match those features.

3. Product Use Case Name: Music Generation Based on Input
Trigger: User inputs reference song(s) and/or song snippet(s), and indicates they want to generate a song
Preconditions: User must have GenreGuru open, and the user has provided a valid input(s)
Interested Stakeholders: Music producers, Hobbyist Musicians
Actor/s: User
Activity Diagram:
Outcome: The user will enter song(s) and/or song snippet(s) and indicate to the system that they want to generate music, the system will check that these inputs are valid (correct format) and then will generate a song and return it to the user.

4. Product Use Case Name: Analyze Music
Trigger: User inputs a reference song or song snippet and indicates they want to analyze the music
Preconditions: User must have GenreGuru open, and the user has provided a valid input
Interested Stakeholders: Music Producers, Audio Engineers, Music Educators
Actor/s: User
Activity Diagram:

Outcome: The user will input a reference song or song snippet and indicate they want to analyze the song, the system will validate the input and return a set of features and visualizations.

5. Product Use Case Name: Song Recommendation Based on Input
Trigger: User inputs reference song(s) and/or snippet(s), and indicates they want to search for recommendations
Preconditions: User must have GenreGuru open, and the user has provided a valid input(s)
Interested Stakeholders: Casual Music Listeners, Hobbyist Musicians
Actor/s: User
Activity Diagram:
Outcome: The users will input reference song(s) and/or snippet(s), the system will first check to see if the inputs are valid. Then the system will return a collection of reference songs.

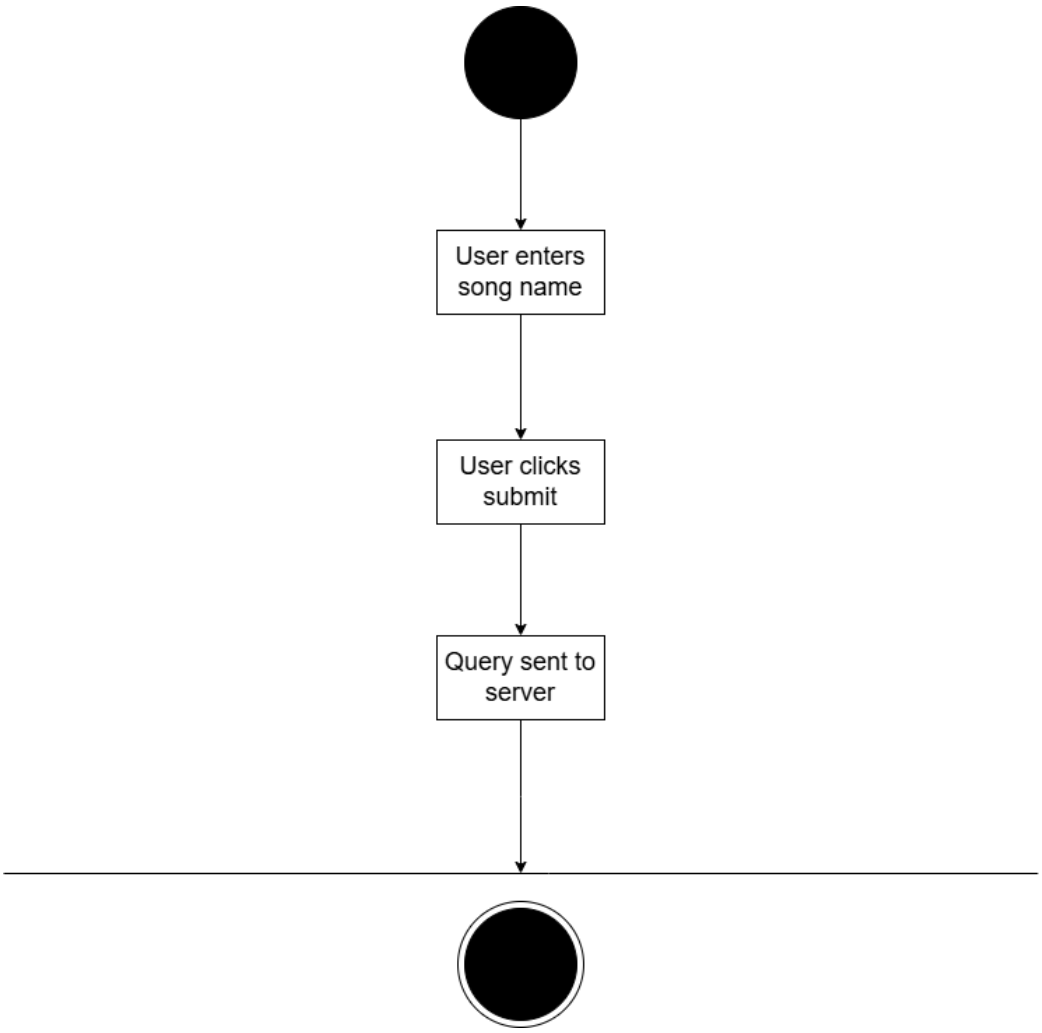
6. Product Use Case Name: Server Interaction for Music Generation
Trigger: User submits a reference song and/or snippet and requests music generation
Preconditions: User has provided a valid input through, and the server is operational
Interested Stakeholders: Music Producers, Hobbyist Musicians
Actor/s: Server
Activity Diagram:
Outcome: The server processes the input, generates music, and returns the generated song to the user

7. Product Use Case Name: Server Interaction for Song Recommendation
Trigger: User submits desired features or reference songs/snippets and requests song recommendations
Preconditions: User has provided valid input, and the server is available
Interested Stakeholders: Casual Music Listeners, Hobbyist Musicians
Actor/s: Server
Activity Diagram:
Outcome: The server processes the input and returns a collection of recommended

~~songs based on the input features or reference songs/snippets.~~

~~**8. Product Use Case Name:** Server Interaction for Music Analysis~~
~~**Trigger:** User submits a reference song or snippet and requests music analysis~~
~~**Preconditions:** User has provided a valid input, and the server is ready to analyze~~
~~**Interested Stakeholders:** Music Producers, Audio Engineers, Music Educators~~
~~**Actor/s:** Server **Activity Diagram:**~~
~~**Outcome:** The server analyzes the song or snippet and returns a collection of features and visualizations to the user.~~

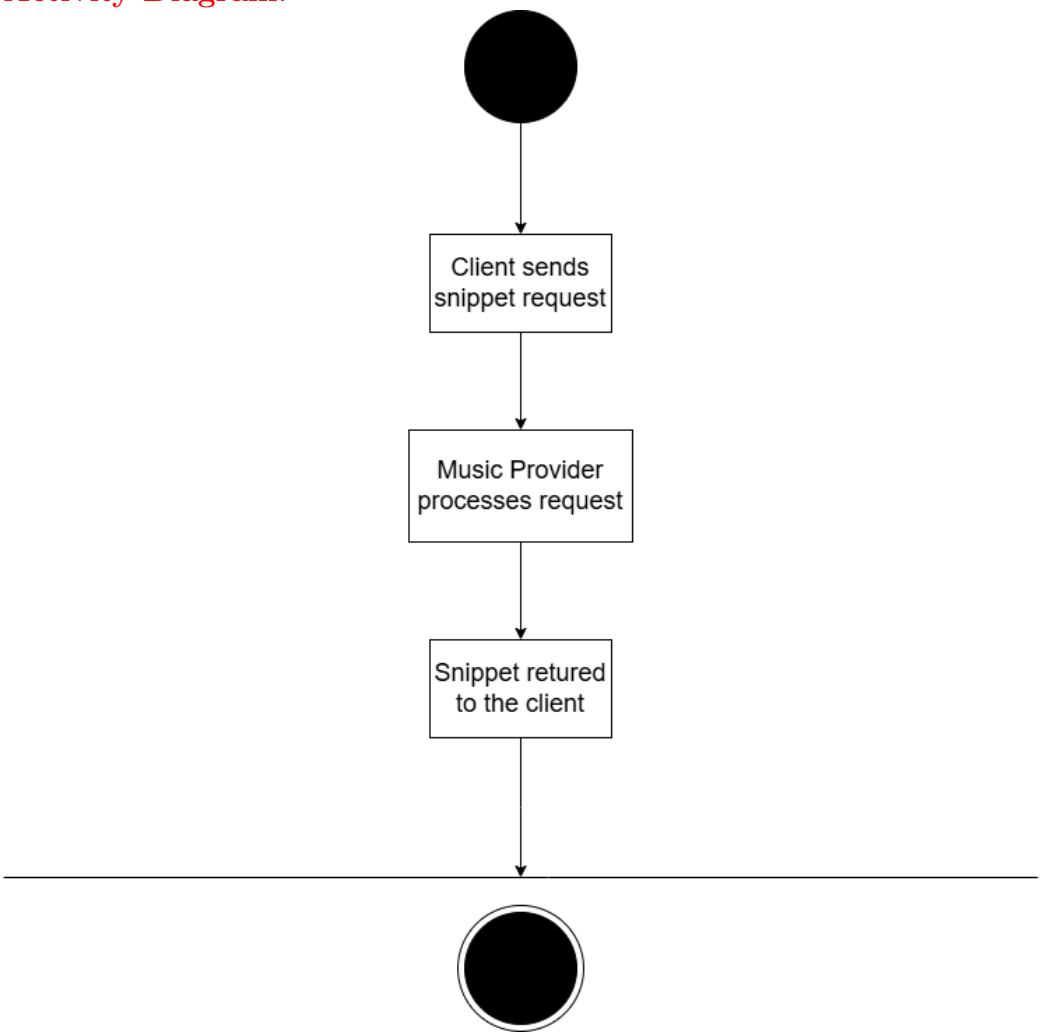
1. Product Use Case Name: Submit Song Query
Trigger: The user enters a song name and clicks the submit button.
Preconditions: The user is logged into GenreGuru and on the query submission screen.
Interested Stakeholders: Client, Customer, Subject Matter Experts, Musicians, Music Educators
Actor/s: User
Activity Diagram:



Outcome: The query is transmitted from the client to the server, initiating the recommendation process.

2. Product Use Case Name: Retrieve Song Snippet
Trigger: The client application sends a snippet request upon receiving a song query.
Preconditions: A valid song query has been submitted, and the client has access to the music provider API.
Interested Stakeholders: Client, Customer, Subject Matter Experts, Mu-

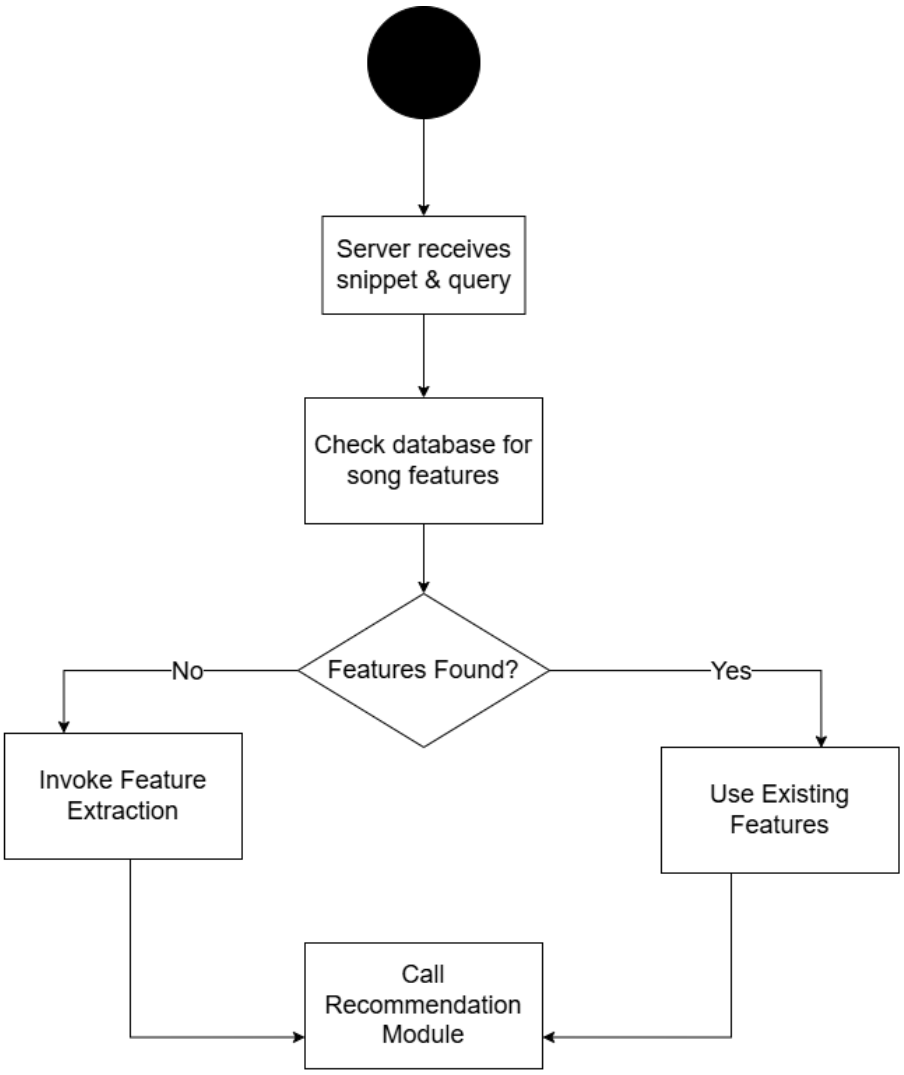
sic Producers & Sound Engineers, Music Regulators
Actor/s: Client Application, Music Provider
Activity Diagram:



Outcome: A song snippet (metadata or audio preview) is received from the music provider and passed to the client.

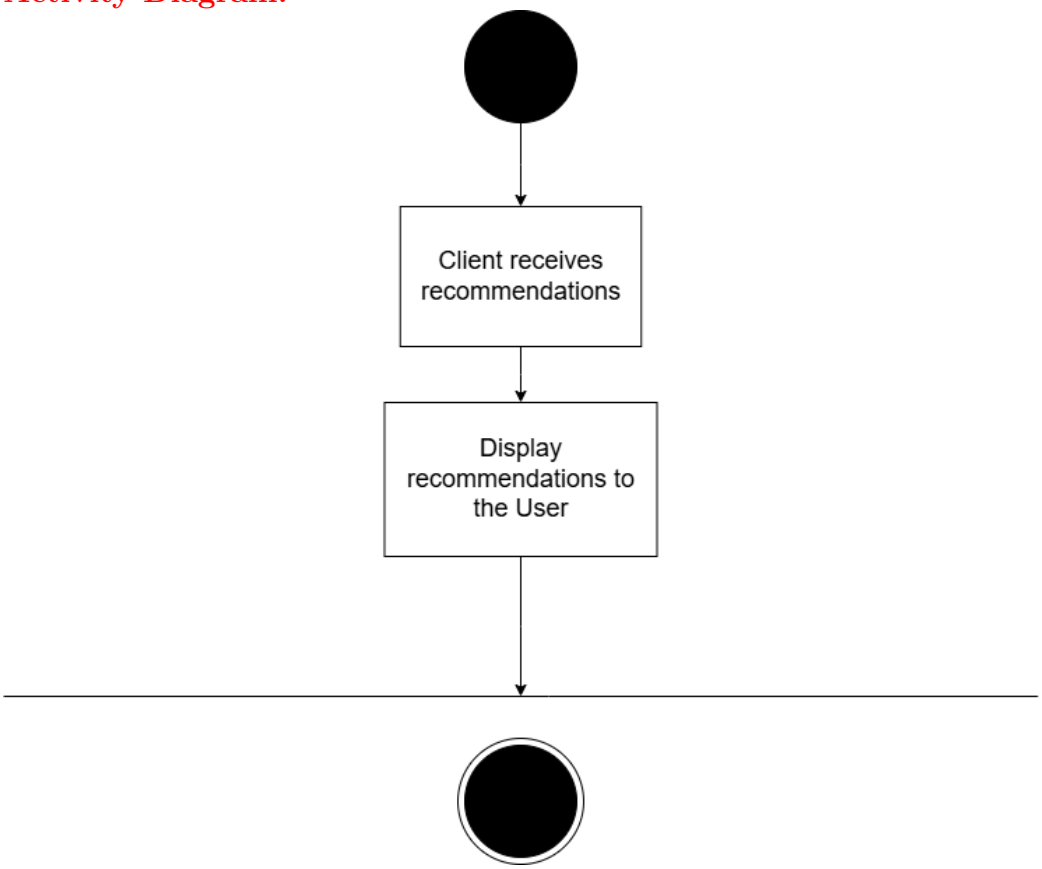
3. Product Use Case Name: Process Recommendation Request
Trigger: The server receives a song snippet and query, triggering the recommendation process.

Preconditions: The client has forwarded the song snippet; the database and recommendation module are accessible.
Interested Stakeholders: Client, Customer, Subject Matter Experts, Music Producers & Sound Engineers, Music Theorists, Music Educators, Music Regulators
Actor/s: Server Controller, Recommendation Module, Feature Extraction
Activity Diagram:



Outcome: The system checks for existing song features, performs feature extraction if needed, and generates a list of recommended songs.

4. Product Use Case Name: Display Recommendations
Trigger: The recommendation module sends a list of recommended songs to the client.
Preconditions: A valid recommendation list has been generated.
Interested Stakeholders: Client, Customer, Subject Matter Experts, Musicians, Music Educators
Actor/s: Client Application, User
Activity Diagram:



Outcome: The list of recommended songs is displayed to the user in an intuitive format.

9 Functional Requirements

9.1 Functional Requirements

~~**Requirement # 1**~~
~~**Requirement Type:** 9~~
~~**Event/Use Case #:** (need to figure this out later)~~
~~**Description:** The client application should allow the user to provide a reference link to a song to initiate a request from the system~~
~~**Rationale:** This provides a straightforward way for users to interact with the system and start a process of their choosing~~
~~**Originator:** Requirement Analyst~~
~~**Fit Criterion:** The user can paste a link, then make a request of the system~~
~~**Customer Satisfaction:** 5~~
~~**Customer Dissatisfaction:** 0~~
~~**Priority:** High~~
~~**Conflicts:** None~~
~~**Supporting Material:** None~~
~~**History:** Created November 1, 2024~~

~~**Requirement # 2**~~
~~**Requirement Type:** 9~~
~~**Event/Use Case #:** (need to figure this out later)~~
~~**Description:** The system should establish a connection between the client application, server, and external APIs to handle data requests and responses~~
~~**Rationale:** Want to ensure data flow across the system, from client inputs to external API requests~~
~~**Originator:** Requirement Analyst~~
~~**Fit Criterion:** The client's request will trigger network communication with various components of the system, which will be confirmed by successful data retrieval and submission.~~
~~**Customer Satisfaction:** 5~~
~~**Customer Dissatisfaction:** 0~~
~~**Priority:** High~~
~~**Conflicts:** None~~
~~**Supporting Material:** None~~
~~**History:** Created November 1, 2024~~

~~**Requirement # 3**~~
~~**Requirement Type:** 9~~
~~**Event/Use Case #:** (need to figure this out later)~~
~~**Description:** The system should process feature extraction requests~~
~~**Rationale:** Want to provide efficient and reliable processing~~
~~**Originator:** Requirement Analyst~~
~~**Fit Criterion:** For each valid request, the server completes feature extraction within a specified time~~
~~**Customer Satisfaction:** 5~~
~~**Customer Dissatisfaction:** 0~~
~~**Priority:** High~~
~~**Conflicts:** None~~
~~**Supporting Material:** None~~
~~**History:** Created November 1, 2024~~

~~**Requirement # 4**~~
~~**Requirement Type:** 9~~
~~**Event/Use Case #:** (need to figure this out later)~~
~~**Description:** When a reference link is provided, the system shall use External Service APIs to retrieve song data~~
~~**Rationale:** Using external APIs to retrieve song data will reduce input requirements from the user~~
~~**Originator:** Requirement Analyst~~
~~**Fit Criterion:** A valid reference link triggers data retrieval from the external API~~
~~**Customer Satisfaction:** 5~~
~~**Customer Dissatisfaction:** 0~~
~~**Priority:** High~~
~~**Conflicts:** None~~
~~**Supporting Material:** None~~
~~**History:** Created November 1, 2024~~

~~**Requirement # 5**~~
~~**Requirement Type:** 9~~
~~**Event/Use Case #:** (need to figure this out later)~~
~~**Description:** The system should analyze the provided reference song and extract a set of features (e.g., tempo, pitch, rhythm, genre, etc.)~~
~~**Rationale:** Users want a way to break down songs for a more detailed analysis~~
~~**Originator:** Requirement Analyst~~
~~**Fit Criterion:** With every request, the system will return a consistent set of features~~
~~**Customer Satisfaction:** 5~~
~~**Customer Dissatisfaction:** 0~~
~~**Priority:** High~~
~~**Conflicts:** None~~
~~**Supporting Material:** None~~
~~**History:** Created November 1, 2024~~

~~**Requirement # 6**~~
~~**Requirement Type:** 9~~
~~**Event/Use Case #:** (need to figure this out later)~~
~~**Description:** The system should provide error feedback for each component~~
~~**Rationale:** Clear error messaging supports debugging and ensures users receive actionable feedback~~
~~**Originator:** Requirement Analyst~~
~~**Fit Criterion:** Each component generates specific error messages that guide users and/or developers~~
~~**Customer Satisfaction:** 5~~
~~**Customer Dissatisfaction:** 0~~
~~**Priority:** High~~
~~**Conflicts:** None~~
~~**Supporting Material:** None~~
~~**History:** Created November 1, 2024~~

Requirement # 7
Requirement Type: 9
Event/Use Case #: (need to figure this out later)
Description: The client application should display the output of system components
Rationale: To provide a way of interpreting/visualizing the output to the user
Originator: Requirement Analyst
Fit Criterion: The client application displays the output for the user to see
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created November 1, 2024

Requirement # 8
Requirement Type: 9
Event/Use Case #: (need to figure this out later)
Description: The system will validate user inputs to ensure they are correct
Rationale: Prevents errors and ensures the system processes valid data.
Originator: Requirement Analyst
Fit Criterion: The system will display an error message if the input is invalid, or will let the user proceed if the input is valid.
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created November 1, 2024

~~**Requirement # 9**~~
~~**Requirement Type:** 9~~
~~**Event/Use Case #:** (need to figure this out later)~~
~~**Description:** The system should feature a client application for user interaction~~
~~**Rationale:** To allow users to interact with the system efficiently and intuitively~~
~~**Originator:** Requirement Analyst~~
~~**Fit Criterion:** The client application allows users to initiate interactions such as submitting song links, requesting analysis, etc.~~
~~**Customer Satisfaction:** 5~~
~~**Customer Dissatisfaction:** 0~~
~~**Priority:** High~~
~~**Conflicts:** None~~
~~**Supporting Material:** None~~
~~**History:** Created November 1, 2024~~

Requirement #1
Requirement Type: 9
Event/Use Case #: 1
Description: The system shall allow users to input song queries either by typing in a text field or by uploading an audio file.
Rationale: Provides the primary mechanism for users to initiate interactions with GenreGuru.
Originator: Requirement Analyst
Fit Criterion: The GUI displays clearly labeled input fields for text entry and file upload
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Requirement #2
Requirement Type: 9
Event/Use Case #: 2
Description: The system GUI shall display search results—including song title, artist, and album cover—for text-based queries.
Rationale: Enables users to easily identify and select the desired track.
Originator: Requirement Analyst
Fit Criterion: Search results are presented within 10 seconds of query submission in a clear, user-friendly format.
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Requirement #3
Requirement Type: 9
Event/Use Case #: 2
Description: The system shall allow users to play a 30-second preview of each search result.
Rationale: Helps users confirm their song selection before proceeding further.
Originator: Requirement Analyst
Fit Criterion: Clicking the "Preview" button plays a 30-second audio clip within 3 seconds.
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Requirement #4
Requirement Type: 9
Event/Use Case #: 1
Description: The system shall validate uploaded audio files to ensure they are in supported formats (e.g., WAV, MP3).
Rationale: Prevents processing errors from unsupported file formats.
Originator: Requirement Analyst
Fit Criterion: Unsupported files are rejected with a clear error message; only validated formats proceed to processing.
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Requirement #5
Requirement Type: 9
Event/Use Case #: 1
Description: The system shall convert non-WAV audio files to WAV format before processing.
Rationale: Standardizes input formats for the feature extraction module.
Originator: Requirement Analyst
Fit Criterion: Any uploaded file not in WAV format is automatically converted to WAV with no user intervention.
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Requirement #6
Requirement Type: 9
Event/Use Case #: 2
Description: The system shall construct a query based on user input and interface with a music provider to retrieve a list of matching songs.
Rationale: Facilitates song discovery via text-based search by leveraging external music databases.
Originator: Requirement Analyst
Fit Criterion: A correctly formatted query string is generated and transmitted to the music provider, returning a valid list of matching songs within 10 seconds.
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Requirement #7
Requirement Type: 9
Event/Use Case #: 3
Description: The system shall extract essential audio features (e.g., spectral, rhythmic, timbral characteristics) from input songs required for generating recommendations.
Rationale: Audio features serve as the basis for comparing songs and generating meaningful recommendations.
Originator: Requirement Analyst
Fit Criterion: The feature extraction module shall successfully extract all required audio features from with consitent results.
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Requirement #8
Requirement Type: 9
Event/Use Case #: 3
Description: The system shall support feature extraction for all ten predetermined audio features defined in the project scope.
Rationale: Ensures comprehensive analysis of song attributes for robust recommendation generation.
Originator: Requirement Analyst
Fit Criterion: Each feature extraction module outputs valid data for supported audio files as verified by testing with representative samples.
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Requirement #9
Requirement Type: 9
Event/Use Case #: 3
Description: The system shall generate a ranked list of song recommendations based on the extracted audio features.
Rationale: Enables users to discover new songs aligned with the characteristics of their input without exposing internal algorithm details.
Originator: Requirement Analyst
Fit Criterion: Recommendations are generated and displayed within 30 seconds of feature extraction; internal algorithm specifics remain abstract in the SRS.
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Requirement #10
Requirement Type: 9
Event/Use Case #: 1
Description: The system shall reliably transmit user-selected inputs (text queries or audio files) from the client to the server for processing.
Rationale: Ensures efficient and accurate data exchange between the client and server.
Originator: Requirement Analyst
Fit Criterion: Data is transmitted with minimal latency (under 5 seconds)
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Requirement #11
Requirement Type: 9
Event/Use Case #: 4
Description: The system shall receive processed recommendation results from the server and update the GUI accordingly.
Rationale: Displays the final output to the user, completing the interaction cycle.
Originator: Requirement Analyst
Fit Criterion: The GUI updates with the recommendation list within 5 seconds of receiving the processed results.
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Requirement #12
Requirement Type: 9
Event/Use Case #: 3
Description: The system shall store and retrieve song metadata, user queries, and extracted features in a secure and efficient database.
Rationale: A robust database is critical for caching data, minimizing redundant processing, and enabling quick retrieval.
Originator: Requirement Analyst
Fit Criterion: Database queries execute with an average latency of under 2 seconds and maintain data integrity during transactions.
Customer Satisfaction: 5
Customer Dissatisfaction: 0
Priority: High
Conflicts: None
Supporting Material: None
History: Created April 3, 2025

Traceability Matrix

	PUC1	PUC2	PUC3	PUC4
FR1	X			
FR2		X		
FR3		X		
FR4	X			
FR5	X			
FR6		X		
FR7			X	
FR8			X	
FR9			X	
FR10	X			
FR11				X
FR12			X	

Table 6: Traceability Matrix Linking Functional Requirements to Product Use Cases

10 Look and Feel Requirements

10.1 Appearance Requirements

APR1. The system shall present usable options without clutter.
APR2. The system shall be visually accessible by using highly contrasting elements.
APR3. The system shall use an easily navigable, professional design.

10.2 Style Requirements

STR1. The system shall be responsive to various screen sizes.

11 Usability and Humanity Requirements

11.1 Ease of Use Requirements

EUR1. The system shall show tooltips pop-ups upon users hovering on an interactable element.

11.2 Personalization and Internationalization Requirements

PIR1. The system shall support different display colour themes.

11.3 Learning Requirements

LR1. The system shall provide a first-time user guide to use features upon first launch.
LR2. The user shall be able to complete the tutorial within *10* minutes.

11.4 Understandability and Politeness Requirements

UPR1. The system shall not generate vulgar content.
UPR2. The system shall only use standardized, inoffensive iconography wherever icons are needed.

11.5 Accessibility Requirements

ACR1. The system shall only use accessible fonts.
ACR2. The system shall offer a color-blind mode.

12 Performance Requirements

12.1 Speed and Latency Requirements

N/A – while there is a server and networking in the system, such requirements are considered out of scope.

12.2 Safety-Critical Requirements

N/A – there are no safety-affiliated components affiliated with the project.

12.3 Precision or Accuracy Requirements

PAR1. Query request times are displayed to within 1 minute of precision.
PAR2. Displayed percentages will be rounded to the nearest two decimals.

12.4 Robustness or Fault-Tolerance Requirements

RAR1. The system shall maintain 97.5% uptime, i.e., an average downtime of 18 hours every month.

12.5 Capacity Requirements

CR1. The system shall handle concurrent usage by up to four users.
CR2. The system shall store a cached record of songs and corresponding information.

12.6 Scalability or Extensibility Requirements

N/A – while there is a server in the system, scaling is not in scope.

12.7 Longevity Requirements

N/A – maintenance and longevity of the project is not in scope, at least not until after the POC demonstration.

13 Operational and Environmental Requirements

13.1 Expected Physical Environment

EPER1. The system server shall run on a modified Dell OptiPlex 3050.

13.2 Wider Environment Requirements

N/A

13.3 Requirements for Interfacing with Adjacent Systems

N/A

13.4 Productization Requirements

PRR1. The system front-end shall be accessible from a browser across different devices.

13.5 Release Requirements

NA – the project is academic in nature, so a release cycle is unnecessary.

14 Maintainability and Support Requirements

14.1 Maintenance Requirements

MR1. The system shall facilitate switching between music providers overnight.

14.2 Supportability Requirements

NA – the project is academic in nature, and its lifespan is unlikely to extend such that it needs support.

14.3 Adaptability Requirements

N/A – a web-app is the most portable format for the specified target users.

15 Security Requirements

15.1 Access Requirements

ACR1. The user shall only be able to access songs they upload or ones that are licensed for their use.

15.2 Integrity Requirements

IR1. *The server shall perform a weekly backup of its database to prevent data loss in the event of a catastrophic failure.*
IR2. *The server shall implement deduplication measures to guarantee no data redundancy.*

15.3 Privacy Requirements

PR1. The system shall only expose query requests to the user that made them.

15.4 Audit Requirements

AUR1. The system shall maintain a history of user query requests.

15.5 Immunity Requirements

N/A – again, the project is academic in nature, and its purpose is not to be robust for attacks on a commercial software.

16 Cultural Requirements

16.1 Cultural Requirements

CUR1. The system shall not contain any (potentially) culturally offensive iconography or language.

17 Compliance Requirements

17.1 Legal Requirements

LGR1. The system shall not make use of any copyrighted material without express permission.
LGR2. The data collection process must obey all potential API developer rules.

17.2 Standards Compliance Requirements

N/A – out of project scope.

18 Open Issues

N/A.

- ~~Generated output use regulation, #1~~
  Summary: ~~The service EULA agreement might need to consider how and where the generated snippet by the service is allowed to be used.~~

  ~~This is important because the training data or inputted track(s) into the service might have copyright laws & rules regulating commerical use associated with them, thus we need to consider whether those rules & regulations also apply with the generated snippet.~~

  ~~For example, the EULA might need to state that generated snippets cannot be used for commercial purposes if the input contains song from a specific artist belonging to some specific record label. This means that a creative professional's ability to actually use this portion of the service could be greatly limited depending on what tracks they are inputting, and/or what the machine learning algorithm was trained on.~~

- ~~Dataset Bias, #2~~
  ~~**Summary:** We are currently not exactly certain about what dataset we will be using for the algorithm training. For example, if we are using some form of non-copyrighted music, there's a likely chance it is heavily biased towards electronic dance music as a genre. This could skew the algorithm's ability to interact with music that is not of that genre, thus the service might not be able to properly process requests users who perhaps listen to more niche genres of music. This means we might need to train the machine learning algorithms on multiple different datasets, or we need to manually modify an existing dataset to integrate more niche genres of music as the training source.~~

19 Off-the-Shelf Solutions

19.1 Ready-Made Products

N/A.

19.2 Reusable Components

N/A.

19.3 Products That Can Be Copied

N/A.

20 New Problems

20.1 Effects on the Current Environment

~~The platform's high computational requirements, especially during the training and usage of generative models, may put a strain on existing computational resources. This could lead to slower performance for other applications that share these resources. Additionally, increased data transmission due to the integration with APIs such as Spotify and Deezer could lead to higher network bandwidth utilization, potentially affecting the performance of other network-dependent systems. Mitigating these effects may require infrastructure~~

~~scaling, such as increasing server capacity or optimizing data processing, to ensure that the existing systems are not disrupted.~~

With the revised scope, the system's computational load is significantly reduced because it no longer involves intensive generative modeling or in-depth audio analysis. However, relying heavily on external music provider APIs (e.g., Spotify and Deezer) for song data and snippets may increase network dependency. This could lead to performance bottlenecks if the external services experience latency or downtime. Additionally, although computational requirements are lower, the system must still handle potentially high volumes of recommendation requests, which may require improved load balancing on the server.

20.2 Effects on the Installed Systems

~~The new platform will interact with various installed systems, leading to the introduction of new dependencies. For example, using third-party APIs like Spotify or integrating machine learning frameworks such as TensorFlow may require changes to the current software stack. These dependencies could introduce compatibility issues, particularly if third-party providers update their APIs or if new versions of required libraries are released. Such updates might necessitate timely system changes to maintain functionality and avoid disruptions. Ensuring that installed systems remain compatible requires diligent version management, comprehensive testing, and a structured process for managing library and API updates.~~

The new recommendation system now integrates with external APIs and an internal database, rather than relying on custom-built generative models. This change reduces stress on local processing resources but introduces new dependencies on third-party API stability and compatibility. As external providers update their APIs or change data formats, our system might require timely adjustments to maintain functionality. Furthermore, integrating with the existing client interface and database now centers on efficient retrieval and display of recommendations rather than processing raw audio data.

20.3 Potential User Problems

~~The platform is designed to offer advanced features, which may present usability challenges for some users. For example, users with limited technical skills may struggle to understand the process of modifying musical features like key, rhythm, or tempo, which are essential for customizing music generation. Furthermore, the quality of generated outputs is inherently subjective, and users may feel frustrated if the system-generated music does not meet their expectations. Another potential problem is system latency, especially when resource-intensive operations like model inference are being performed. Long waiting times could negatively impact user experience, making the platform feel less responsive, which may deter regular use.~~

Since the system now only provides recommendations, user problems may shift toward issues of relevance and clarity. For example, if the recommendation engine fails to consider a user's context adequately, users might receive suggestions that do not align with their preferences. In addition, users might be confused if the recommendations are not clearly explained or if the interface does not provide intuitive feedback regarding how recommendations are derived. Although the computational complexity is reduced, latency in fetching data from external APIs could still impact the overall user experience, particularly if network conditions are poor.

20.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

~~The platform is expected to be implemented on a local server, which presents certain limitations. The local server environment will need sufficient processing power to handle computationally intensive tasks, such as model training and music generation. If the server infrastructure is under-resourced, performance issues such as latency and reduced processing speed could arise. Additionally, reliance on a local server means that the platform's effectiveness is tied directly to server availability and maintenance. Any server downtime could render the platform inaccessible to users. If we decide to extend the platform's implementation onto personal devices, new limitations arise. Personal devices, especially those with limited hardware capabilities, may struggle with the computational requirements of generative music models, leading to significant~~

~~performance bottlenecks. This could make the platform inaccessible or difficult to use for some users. Moreover, the reliance on internet connectivity for accessing external APIs is another major limitation. Users in areas with unreliable internet or low bandwidth may face difficulties, particularly with real-time features like music recommendations or generating new songs based on streaming data. These limitations could restrict the user base to only those with high-performance devices and stable internet connections.~~

The reliance on external APIs introduces new limitations. Any changes in API rate limits, unexpected downtime, or shifts in available data could directly affect system performance. Furthermore, while the reduced computational load allows the system to run on less powerful hardware, it also means that network reliability becomes the primary constraint. Users in regions with low bandwidth or unreliable internet connections may experience delays in receiving recommendations. Finally, the centralized nature of the recommendation processing still requires a robust server environment to handle concurrent requests effectively.

20.5 Follow-Up Problems

~~Once the platform is deployed, several follow-up issues will need to be addressed to maintain and improve the product. One of the primary challenges will be keeping up with third-party API changes, such as modifications to Spotify or Deezer APIs, which could lead to broken features if not handled promptly. The generative models used by the platform will also require regular updates and retraining to stay relevant to emerging musical styles and user preferences. Another follow-up problem is related to usability: feedback from users might reveal unforeseen pain points or desired improvements. Addressing these concerns will require an iterative development approach, incorporating user feedback into future versions of the platform to enhance usability and performance.~~

Post-deployment, the system will need ongoing monitoring of external API changes. The team must establish a process to rapidly adapt to updates from Spotify, Deezer, or other providers to prevent broken functionality. In addition, user feedback may reveal new issues—such as a need for more personalized recommendation criteria or additional explanation of recommendation logic—that were not apparent during development. Periodic reviews of the recommendation algorithm may also be necessary to adjust parameters or

incorporate new features as user preferences evolve. Lastly, ensuring seamless integration with evolving client interfaces and maintaining data integrity within the database will remain continuous challenges.

20.6 Follow-Up Problems

N/A.

21 Tasks

- ~~Revise requirements document~~
- ~~Design the systems structure~~
- ~~Build prototype and get feedback~~
- ~~Integrate Spotify API for song data~~
- ~~Develop music recommendation module~~
- ~~Work on music generation module~~
- ~~Create song analysis moudle~~
- ~~Test the platform and gather feedback~~
- ~~Revise based on feedback~~
- ~~Write user guide and documentation~~
- ~~Finalize revisions to documentation~~

The following tasks outline our planned next steps following the successful completion of the capstone project. These tasks focus on further enhancing and scaling GenreGuru:

- **Revise Requirements and Documentation:** Update all requirements documents and supporting materials to reflect the completed recommendation system and identify future enhancements.

- **Implement Neural Network for User Preferences:** Develop a machine learning component to learn from user interactions and assign adaptive weights to different audio features, personalizing recommendations.
- **Expand Audio Feature Set:** Investigate and integrate additional nuanced audio traits (e.g., emotional tone, energy, instrumentation) to enrich the musical profile used for recommendations.
- **Refine User Experience:** Enhance the frontend interface for smoother interactions, better visual feedback, and improved responsiveness across various devices.
- **Finalize WAV Upload Integration:** Complete and productionize the WAV file upload feature, ensuring robust error handling and a seamless user experience.
- **Optimize and Scale the Recommendation Engine:** Review and enhance server resources, database queries, and concurrency handling to support increased load and minimize latency.
- **Conduct Further User Testing and Feedback Cycles:** Gather detailed feedback from diverse user groups on the new features and overall user interface to guide further improvements.
- **Iterate and Evolve:** Incorporate user feedback into continuous development, refine the neural network, and monitor external API changes to maintain system relevance.
- **Update User Documentation:** Revise and publish updated user guides and technical documentation reflecting the new features and enhancements.

22 Migration to the New Product

22.1 Requirements for Migration to the New Product

There are no migration requirements as this project is not a replacement or upgrade of a previous project

22.2 Data That Has to be Modified or Translated for the New System

Similarly, there currently is no data that needs to be modified

23 Costs

The monetary cost estimate of the project is $0 CAD. All of the necessary equipment is owned by at least one group member.

~~The total time cost estimate of the project is 8 months (September 2024 - April 2025).~~
The total time cost estimate of the project is 1,600 man-hours (5 people × 2 hours/day × 5 days/week × 32 weeks).
The function point cost estimate is 12. This is derived from the above sections, mainly the functional requirements, non-functional requirements, and business rules

24 User Documentation and Training

24.1 User Documentation Requirements

The ~~music featurization feature~~ music recommendation feature is heavily API-driven, and as such, detailed documentation will primarily be covered within the API reference section. This approach ensures that developers and advanced users can understand the feature's capabilities without needing additional user guides.

To ensure that users can effectively interact with the ~~music generation and recommendation platform~~ music recommendation platform, the following user documentation will be provided:

- **Quick Start Guide**: A concise guide aimed at helping new users get started with the basics of ~~generating and~~ recommending music.
- **API Reference and Technical Specifications**: Detailed documentation of the platform's API, including available endpoints, request/response formats, and example queries. This reference is crucial for developers

and advanced users who want to integrate the platform with other applications or automate tasks.

- **Installation Guide**: A step-by-step guide for installing the platform on local servers, including system requirements, installation commands, and troubleshooting common setup issues.
- **FAQs and Troubleshooting**: A list of frequently asked questions and troubleshooting tips to help users solve common issues independently.
- **Video Tutorials**: Step-by-step video guides that visually demonstrate key features and workflows, including setting up the platform, using the API, and ~~generating music~~ using the recommendation engine.

These documents will be designed for users of varying technical backgrounds to ensure they can fully utilize the platform's capabilities. The documentation will be created and maintained primarily by the development team, ensuring accuracy and alignment with the latest platform features. However, feedback from user groups will be actively sought to improve clarity and address any documentation gaps. Updates to the API reference and technical specifications will be managed as part of the regular software update cycle.

24.2 Training Requirements

To provide users with sufficient knowledge to operate the platform effectively, the following training resources will be developed:

- **Video Tutorials**: Developed by the development team, these tutorials will cover various aspects of the platform, including API usage, ~~generating music,~~ and using the recommendation system.
- **Online Training Modules**: If additional resources become available, online training modules could be developed to provide users with a structured learning path. However, due to current resource constraints, we do not plan to offer live training sessions.

These training requirements aim to encourage users to explore the full potential of the platform, regardless of their prior experience in music production or technology.

25 Waiting Room

- The recommendation system will be able to recommend songs from less popular music genres (jazz, blues, etc.).
- ~~The analysis system will be able to extract musical features from less popular music genres (jazz, blues, etc.).~~
- ~~The generative system will be able to generate songs from less popular music genres (jazz, blues, etc.).~~
- The recommendation system will be able to recommend songs from unpopular music genres (gnawa, Libyan funk, etc.).
- ~~The analysis system will be able to extract musical features from less popular music genres (gnawa, Libyan funk, etc.).~~
- ~~The generative system will be able to generate songs from less popular music genres (gnawa, Libyan funk, etc.).~~
- The recommendation system will be able to search for songs with cover art similar to an input song.
- ~~The generative system will be able to generate new cover art for a newly generated song, based on the user's input criteria.~~
- ~~The generative system will be able to generate new cover art for an existing song.~~
- The recommendation system will incorporate user feedback to improve song suggestion relevance over time.

26 Ideas for Solution

- **Hybrid Recommendation System**: A hybrid recommendation system combines content-based filtering and collaborative filtering techniques to provide a more personalized experience for users. Content-based filtering analyzes song features, such as genre, key, and rhythm, to suggest similar tracks. Collaborative filtering uses user preferences and historical listening patterns to suggest music. By combining these

approaches, the system can offer users personalized suggestions while also helping them discover new genres and music styles.

- ~~**Generative Music Model**: To enable the creation of new music, a generative model will be used. This model could be based on techniques such as a Generative Adversarial Network (GAN) or Recurrent Neural Network (RNN). A GAN would allow for the generation of realistic music by having the generator and discriminator work together to produce convincing compositions. An RNN, on the other hand, would be well-suited for learning the sequential nature of music, generating new melodies based on learned patterns. This solution provides users with an innovative way to create new music based on their inputs and preferences.~~

- ~~**Feature Manipulation Interface**: This interface will allow users to interact directly with song features, such as tempo, key, and rhythm, enabling them to create customized versions of existing tracks or generate entirely new compositions. By adjusting different musical parameters, users can personalize their musical experience and experiment with creative variations, providing a high level of control over the output.~~

- **Integration with Existing Platforms**: Integrating the system with existing music platforms, such as Spotify, will allow users to easily access and analyze a large library of songs. Users will be able to input their favorite tracks from these platforms and receive recommendations. This integration ensures a smooth user experience, allowing seamless interaction between existing music libraries and the recommendation engine.

- User Preference Neural Network: Develop a model that learns from user interactions to weight different audio features, thereby improving recommendation personalization.

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

   As a team, the Rhythm Rangers, we need to acquire a diverse range of knowledge and skills from various domains, including software development, music generation, and collaborative teamwork, to successfully complete our capstone project. Given the scope of this task, it is essential for each team member to focus on specific areas of expertise that align with their skills, passions, roles, and responsibilities, as well as learn new skills and gain new knowledge. Outlined below is the knowledge and skills the team will collectively need to acquire to successfully complete this capstone project:

   Music Analysis and Signal Processing: This capstone project involves developing expertise in audio signal processing to analyze sound data and extract valuable insights for music recommendation and generation systems. The team will learn to implement machine learning models for tasks such as genre classification and feature extraction. Proficiency in Python libraries for audio analysis and model training is essential. This will deepen the teams understanding of music theory and the connections between song features and genres.

   Frontend or Backend Development: The team will need to understand backend frameworks for building and managing the recommendation system's infrastructure. The will be integrating external APIs to access song previews and features. They will also gain knowledge in database management for storing and organizing song data and

user preferences. Furthermore, this involves learning how to scale and efficiently handle data for a local server-based system.

UI/UX and Design: The team will need to design user-friendly interfaces that ensure smooth interaction with the music recommendation and generation systems. UI/UX design skills will need refinement and utilizization of frontend development frameworks will be needed to craft the systems user interface. They will also learn to connect frontend components with backend APIs for real-time updates, such as delivering song recommendations.

Music Generation: An understanding of generative models to create music snippets from input tracks or references will be a huge component for this project. The team will delve into music feature engineering, transforming audio data into usable features for machine learning applications. Familiarity with music data will assist in generating new content and making recommendations.

Team Management and Infrastructure: For this project to be a success improving project management and team coordination skills will foster effective communication, sprint planning, and task assignment. We will need to learn how to establish and maintain local server infrastructure for efficient hosting and operation of the platform. Understanding security best practices to safeguard user data and ensure the system's resilience against vulnerabilities is critical. Mastery of version control and Git management will promote seamless collaboration and code review among the team.

2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?