

# Software Requirements Specification for Software Engineering: subtitle describing software

Team 8 – Rhythm Rangers

Ansel Chen

Muhammad Jawad

Mohamad-Hassan Bahsoun

Matthew Baleanu

Ahmed Al-Hayali

October 11, 2024

# Contents

<b>1</b>	<b>Purpose of the Project</b>	<b>vi</b>
1.1	User Business . . . . .	vi
1.2	Goals of the Project . . . . .	vi
<b>2</b>	<b>Stakeholders</b>	<b>vi</b>
2.1	Client . . . . .	vi
2.2	Customer . . . . .	vi
2.3	Other Stakeholders . . . . .	vi
2.4	Hands-On Users of the Project . . . . .	vi
2.5	Personas . . . . .	vi
2.6	Priorities Assigned to Users . . . . .	vi
2.7	User Participation . . . . .	vii
2.8	Maintenance Users and Service Technicians . . . . .	vii
<b>3</b>	<b>Mandated Constraints</b>	<b>vii</b>
3.1	Solution Constraints . . . . .	vii
3.2	Implementation Environment of the Current System . . . . .	viii
3.3	Partner or Collaborative Applications . . . . .	ix
3.4	Off-the-Shelf Software . . . . .	ix
3.5	Anticipated Workplace Environment . . . . .	x
3.6	Schedule Constraints . . . . .	xi
3.7	Budget Constraints . . . . .	xi
3.8	Enterprise Constraints . . . . .	xii
<b>4</b>	<b>Naming Conventions and Terminology</b>	<b>xii</b>
4.1	Glossary of All Terms, Including Acronyms, Used by Stakeholders involved in the Project . . . . .	xii
<b>5</b>	<b>Relevant Facts And Assumptions</b>	<b>xii</b>
5.1	Relevant Facts . . . . .	xii
5.2	Business Rules . . . . .	xii
5.3	Assumptions . . . . .	xiii
<b>6</b>	<b>The Scope of the Work</b>	<b>xiii</b>
6.1	The Current Situation . . . . .	xiii
6.2	The Context of the Work . . . . .	xiv
6.3	Work Partitioning . . . . .	xiv

6.4	Specifying a Business Use Case (BUC)	xiv
<b>7</b>	<b>Business Data Model and Data Dictionary</b>	<b>xiv</b>
7.1	Business Data Model	xiv
7.2	Data Dictionary	xiv
<b>8</b>	<b>The Scope of the Product</b>	<b>xv</b>
8.1	Product Boundary	xv
8.2	Product Use Case Table	xv
8.3	Individual Product Use Cases (PUC's)	xv
<b>9</b>	<b>Functional Requirements</b>	<b>xxvi</b>
9.1	Functional Requirements	xxvi
<b>10</b>	<b>Look and Feel Requirements</b>	<b>xxix</b>
10.1	Appearance Requirements	xxix
10.2	Style Requirements	xxix
<b>11</b>	<b>Usability and Humanity Requirements</b>	<b>xxix</b>
11.1	Ease of Use Requirements	xxix
11.2	Personalization and Internationalization Requirements	xxix
11.3	Learning Requirements	xxx
11.4	Understandability and Politeness Requirements	xxx
11.5	Accessibility Requirements	xxx
<b>12</b>	<b>Performance Requirements</b>	<b>xxx</b>
12.1	Speed and Latency Requirements	xxx
12.2	Safety-Critical Requirements	xxx
12.3	Precision or Accuracy Requirements	xxx
12.4	Robustness or Fault-Tolerance Requirements	xxx
12.5	Capacity Requirements	xxx
12.6	Scalability or Extensibility Requirements	xxx
12.7	Longevity Requirements	xxxi
<b>13</b>	<b>Operational and Environmental Requirements</b>	<b>xxxi</b>
13.1	Expected Physical Environment	xxxi
13.2	Wider Environment Requirements	xxxi
13.3	Requirements for Interfacing with Adjacent Systems	xxxi
13.4	Productization Requirements	xxxi

13.5 Release Requirements . . . . .	xxxix
<b>14 Maintainability and Support Requirements</b>	<b>xxxix</b>
14.1 Maintenance Requirements . . . . .	xxxix
14.2 Supportability Requirements . . . . .	xxxix
14.3 Adaptability Requirements . . . . .	xxxix
<b>15 Security Requirements</b>	<b>xxxix</b>
15.1 Access Requirements . . . . .	xxxix
15.2 Integrity Requirements . . . . .	xxxix
15.3 Privacy Requirements . . . . .	xxxix
15.4 Audit Requirements . . . . .	xxxix
15.5 Immunity Requirements . . . . .	xxxix
<b>16 Cultural Requirements</b>	<b>xxxix</b>
16.1 Cultural Requirements . . . . .	xxxix
<b>17 Compliance Requirements</b>	<b>xxxix</b>
17.1 Legal Requirements . . . . .	xxxix
17.2 Standards Compliance Requirements . . . . .	xxxix
<b>18 Open Issues</b>	<b>xxxix</b>
<b>19 Off-the-Shelf Solutions</b>	<b>xxxix</b>
19.1 Ready-Made Products . . . . .	xxxix
19.2 Reusable Components . . . . .	xxxix
19.3 Products That Can Be Copied . . . . .	xxxix
<b>20 New Problems</b>	<b>xxxix</b>
20.1 Effects on the Current Environment . . . . .	xxxix
20.2 Effects on the Installed Systems . . . . .	xxxix
20.3 Potential User Problems . . . . .	xxxix
20.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product . . . . .	xxxix
20.5 Follow-Up Problems . . . . .	xxxix
20.6 Follow-Up Problems . . . . .	xxxix

<b>21 Tasks</b>	<b>xxxv</b>
21.1 Project Planning . . . . .	xxxv
21.2 Planning of the Development Phases . . . . .	xxxvi
<b>22 Migration to the New Product</b>	<b>xxxvi</b>
22.1 Requirements for Migration to the New Product . . . . .	xxxvi
22.2 Data That Has to be Modified or Translated for the New System	xxxvi
<b>23 Costs</b>	<b>xxxvi</b>
<b>24 User Documentation and Training</b>	<b>xxxvi</b>
24.1 User Documentation Requirements . . . . .	xxxvi
24.2 Training Requirements . . . . .	xxxvii
<b>25 Waiting Room</b>	<b>xxxviii</b>
<b>26 Ideas for Solution</b>	<b>xxxix</b>

## Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

# **1 Purpose of the Project**

## **1.1 User Business**

*Insert your content here.*

## **1.2 Goals of the Project**

*Insert your content here.*

# **2 Stakeholders**

## **2.1 Client**

*Insert your content here.*

## **2.2 Customer**

*Insert your content here.*

## **2.3 Other Stakeholders**

*Insert your content here.*

## **2.4 Hands-On Users of the Project**

*Insert your content here.*

## **2.5 Personas**

*Insert your content here.*

## **2.6 Priorities Assigned to Users**

*Insert your content here.*

## 2.7 User Participation

*Insert your content here.*

## 2.8 Maintenance Users and Service Technicians

*Insert your content here.*

# 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 individual



track(s) that can be used as features for the machine learning components of the service.

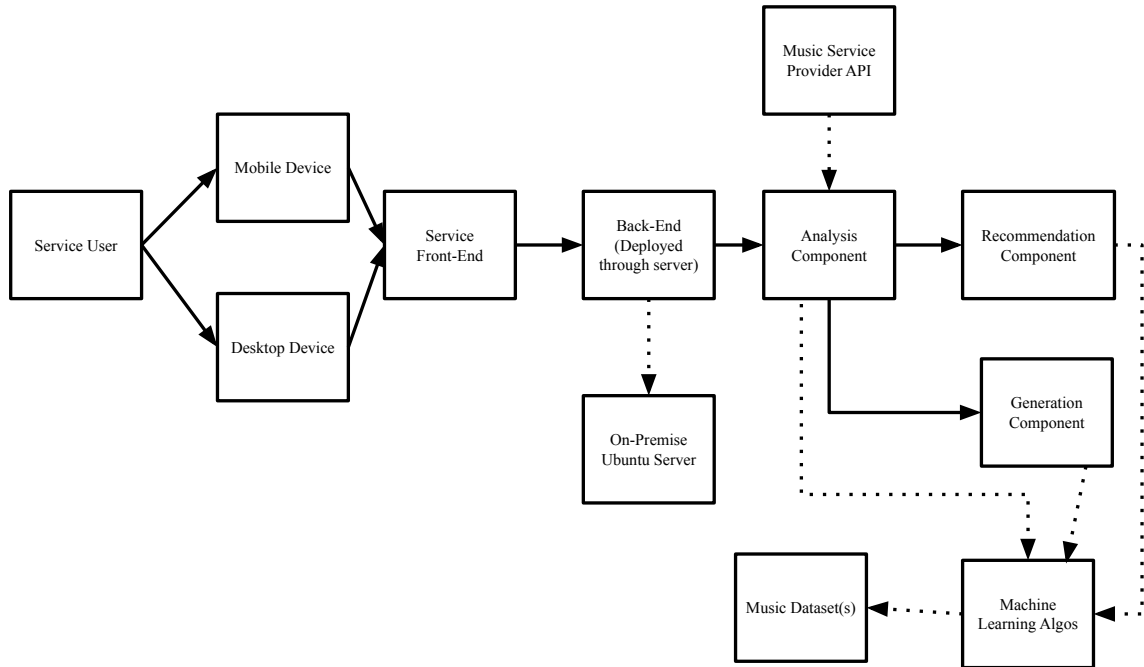
**Fit Criterion:** The service has components that make uses of features (such as API calls) that belong to a music service provider’s platform.

- 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.

### 3.2 Implementation Environment of the Current System



### 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 preferably, not a large amount of data is streamed between the user and the service. This means that preferably, 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 satisfied even with a generation component that produces odd results.

- Studio Usage

**Rationale:** This environment is more professional and what 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 have a reliable internet connection in addition to being willing to input a lot more music to 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. Because the users of this type of environment would likely be individuals engaged in some form of 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. 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.

### 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 mitigate 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.

### 3.7 Budget Constraints

The budget limit as stated by the capstone outline is 750\$ CAD. Potential additional costs might include API calls, software licensing, 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**

*Insert your content here.*

## **5 Relevant Facts And Assumptions**

### **5.1 Relevant Facts**

- Music contains the following core features
  - Tempo
  - Key signature
  - Time signature
  - Pitch
  - Timbre
- Song files have metadata that contains information such as:
  - Song title
  - Artist
  - Release date
- 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
- the user should be able to interact with the system without any external installation

### 5.3 Assumptions

- Users will have at least some familiarity of music theory
- The analysis and recommendation systems 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 systems
- 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 systems will be completed by the Revision 0 date

## 6 The Scope of the Work

### 6.1 The Current Situation

*Insert your content here.*

## **6.2 The Context of the Work**

*Insert your content here.*

## **6.3 Work Partitioning**

*Insert your content here.*

## **6.4 Specifying a Business Use Case (BUC)**

*Insert your content here.*

# **7 Business Data Model and Data Dictionary**

## **7.1 Business Data Model**

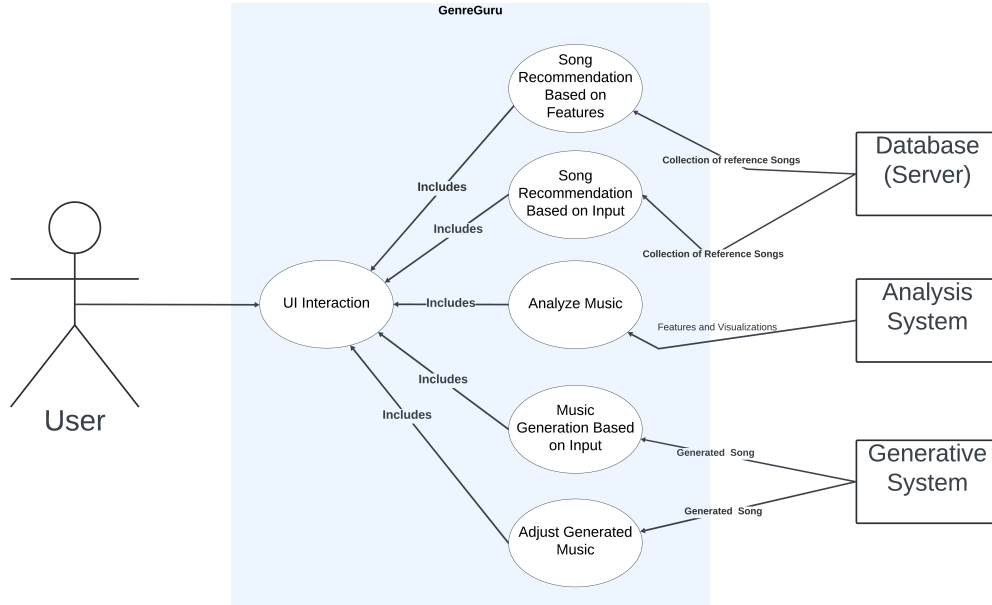
*Insert your content here.*

## **7.2 Data Dictionary**

*Insert your content here.*

## 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	UI Interaction	User	User Actions (click, swipe, drag) (in)      System Response (out)
2	Song Recommendation Based on Features	User	User's desired features (in)      Collection of reference songs (out)
3	Music Generation Based on Input	User	Reference song(s) and/or song snippet(s) (in)      Generated song or song snippet (out)
4	Analyze Music	User	Reference song or song snippet (in)      Collection of features and visualizations (out)
5	Song Recommendation Based on Input	User	Reference song(s) (in)      Collection of reference songs (out)
6	Server Interaction for Music Generation	Server	Reference song(s) and/or song snippet(s) (in)      Generated song or song snippet (out)
7	Server Interaction for Song Recommendation	Server	User's desired features or reference song(s) and/or snippet(s) (in)Collection of reference songs (out)
8	Server Interaction for Music Analysis	Server	Reference song or song snippet (in)      Collection of features and visualizations (out)

Table 1: Product Use Case Table

### 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

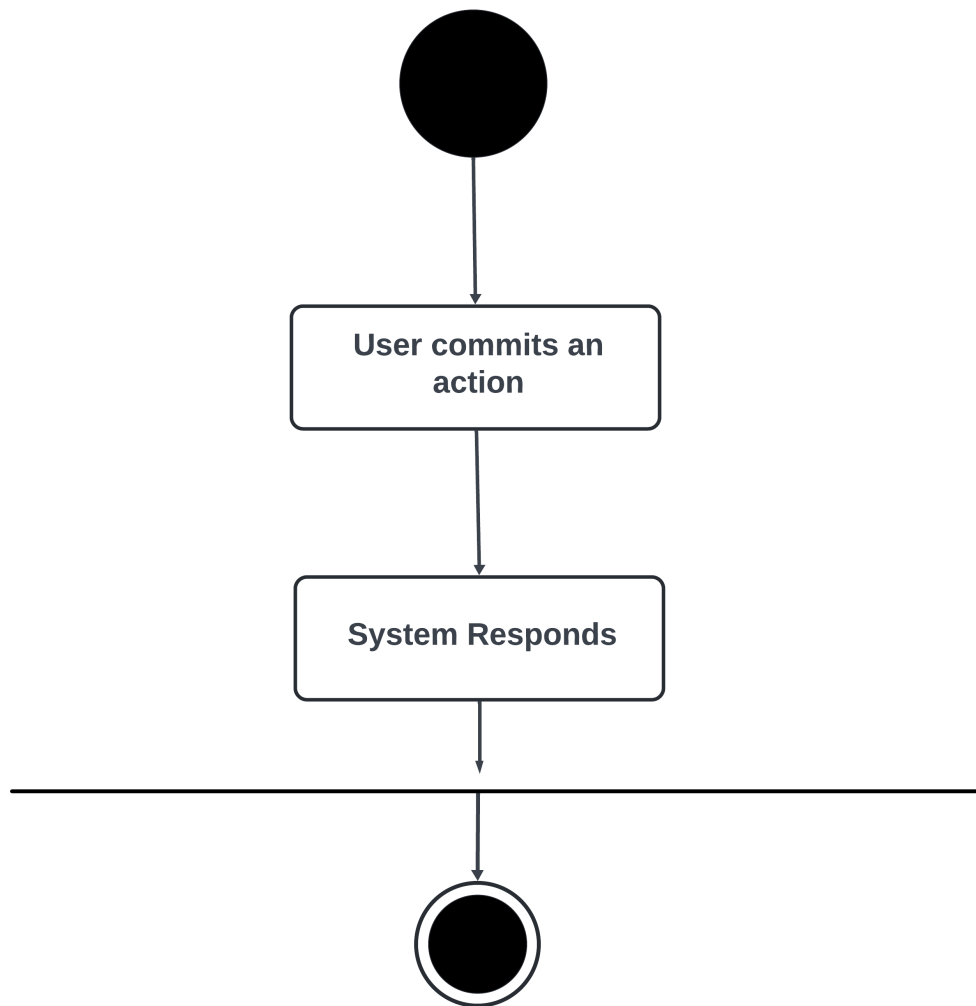


GenreGuru

**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

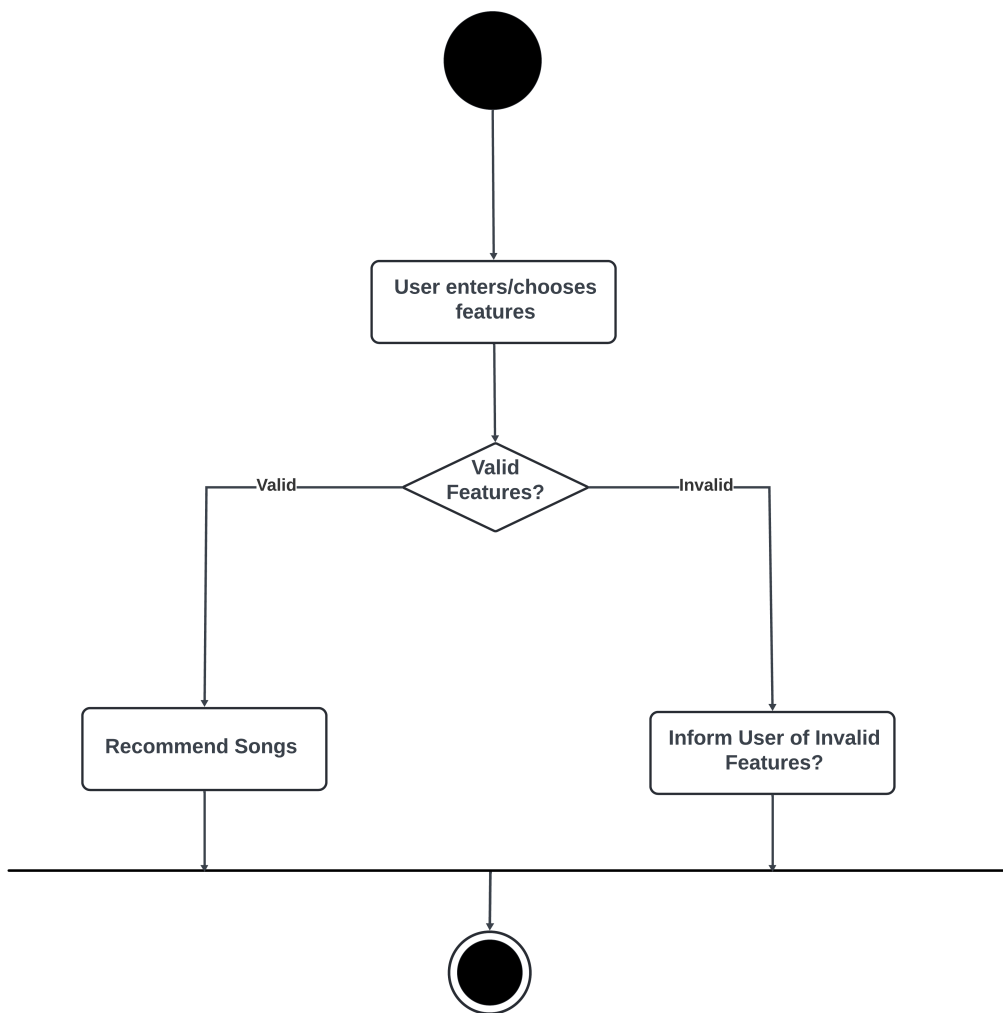
**Trigger:** User picks features, and indicates they want to search for recommendations

**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

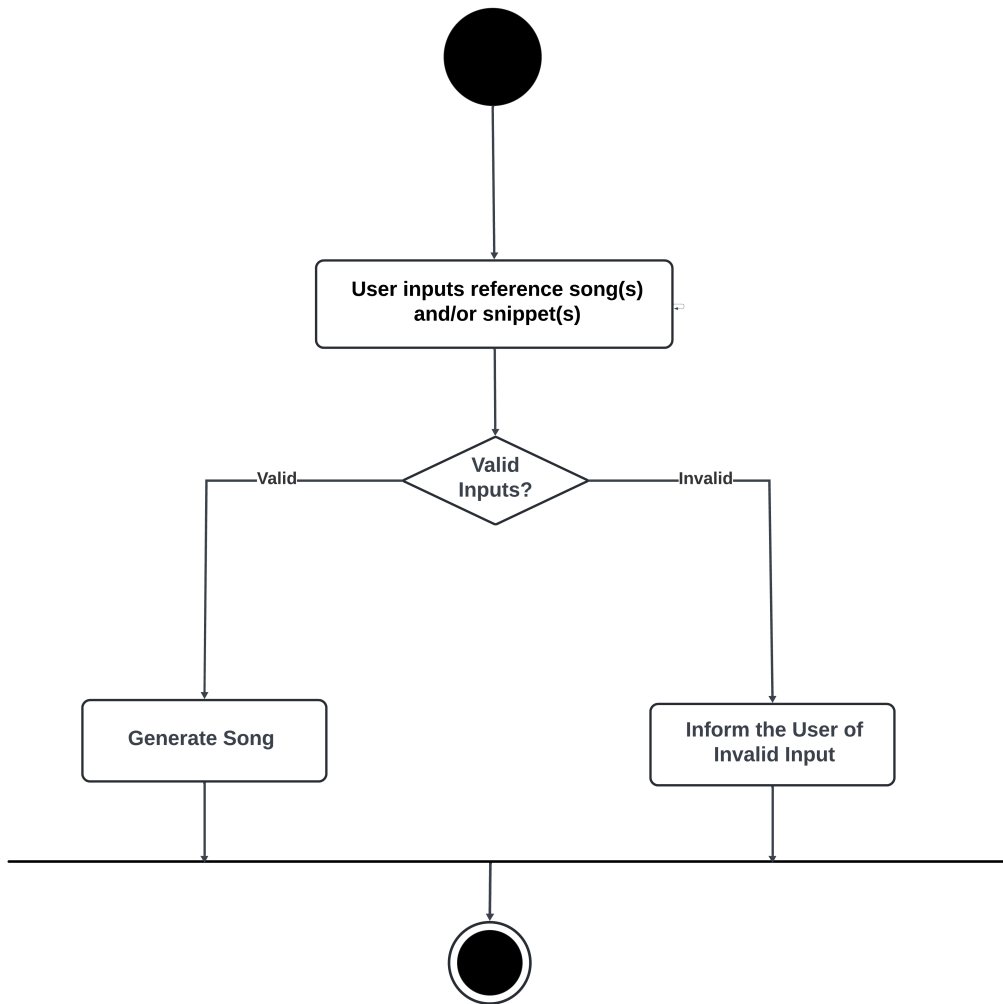
**Trigger:** User inputs reference song(s) and/or song snippet(s), and indicates they want to generate a song

**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

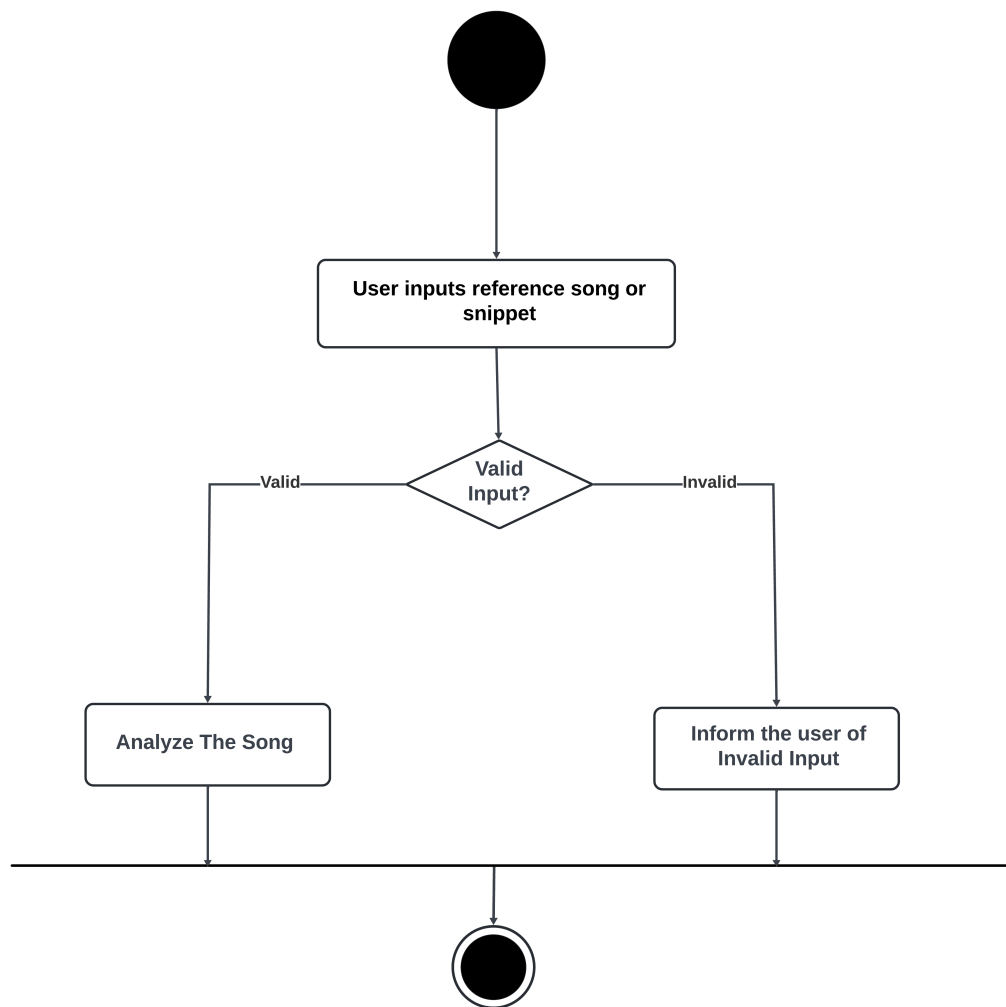
**Trigger:** User inputs a reference song or song snippet and indicates they want to analyze the music

**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

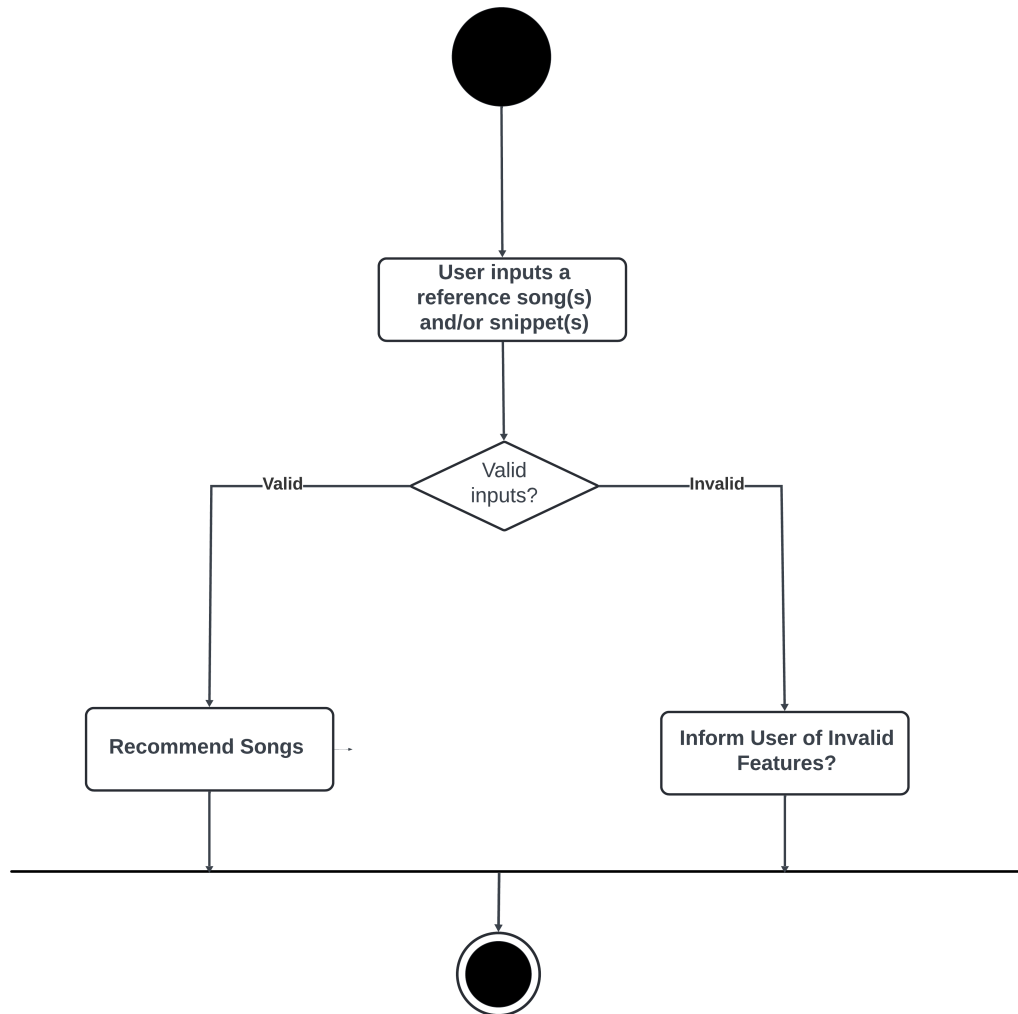
**Trigger:** User inputs reference song(s) and/or snippet(s), and indicates they want to search for recommendations

**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

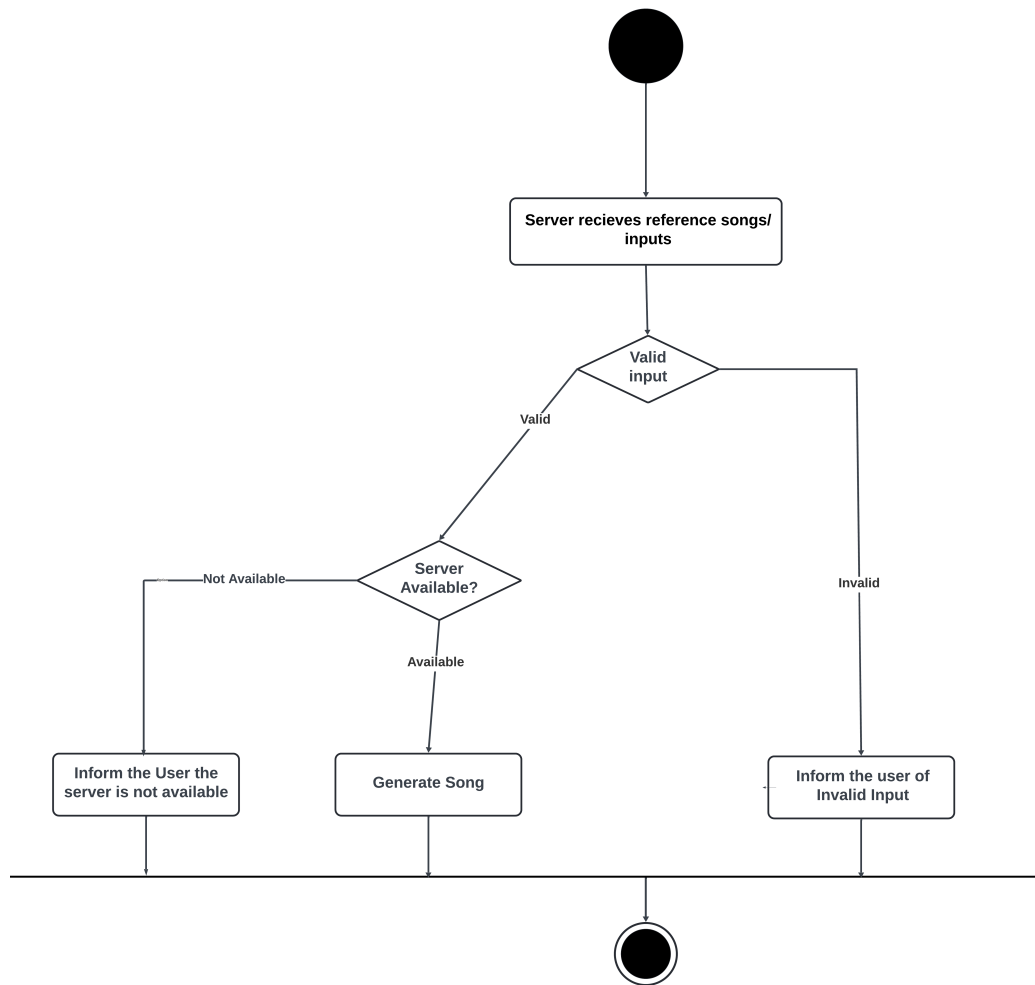
generation

**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

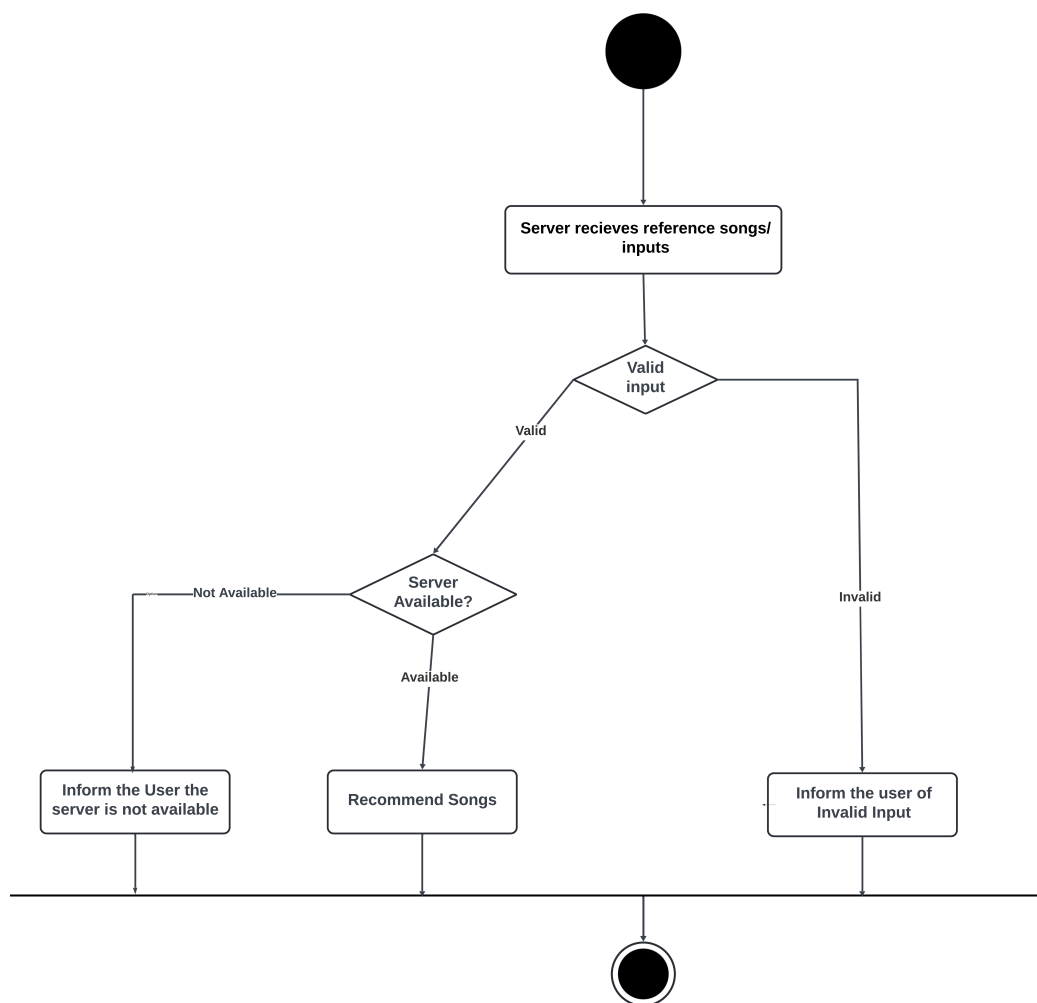
**Trigger:** User submits desired features or reference songs/snippets and requests song recommendations

**Preconditions:** User has provided valid input, and the server is available

**Interested Stakeholders:** Casual Music Listeners, Hobbyist Musicians

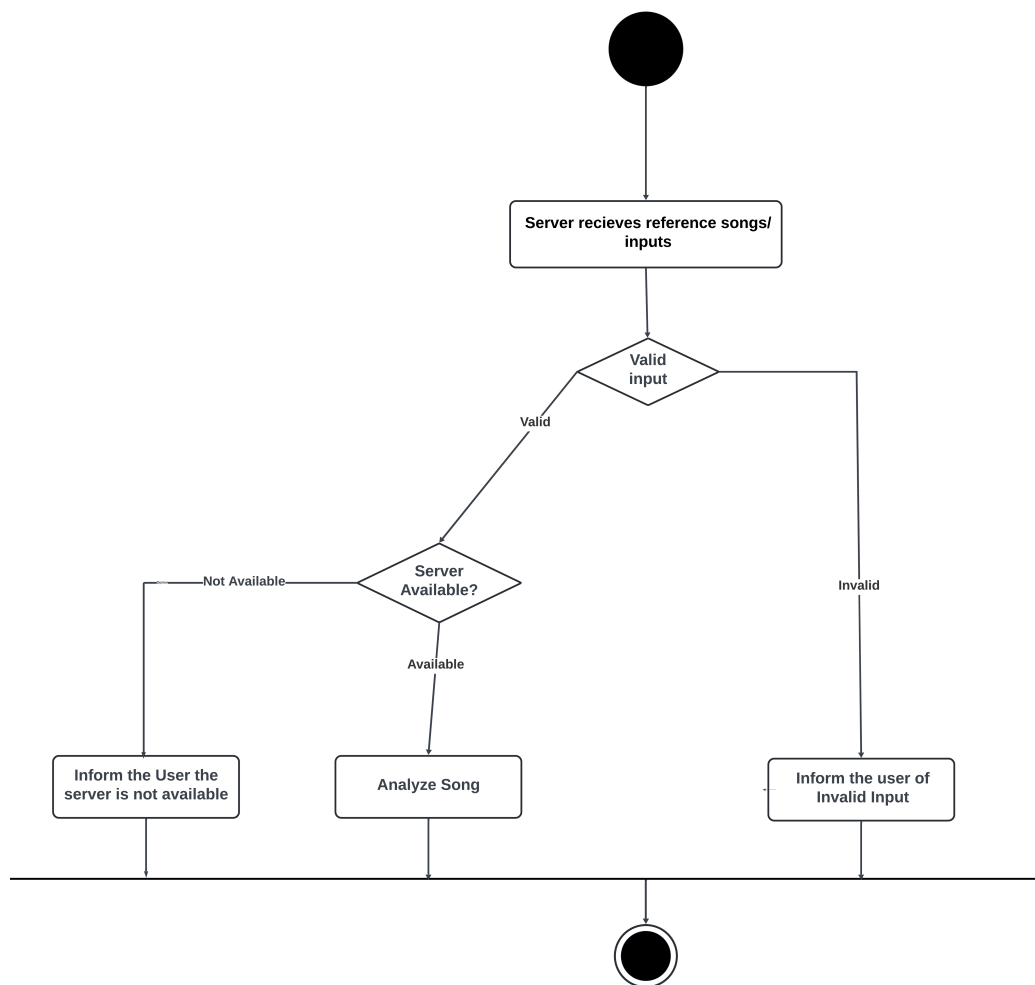
**Actor/s:** Server

**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.

## 9 Functional Requirements

### 9.1 Functional Requirements

**Requirement # 1****Requirement Type:** 9**Event/Use Case #:** 1**Description:** The system should respond to user actions (e.g. swipe, tap).**Rationale:** To allow users to interact with the system efficiently and intuitively.**Originator:** Requirement Analyst**Fit Criterion:** User performs an action (e.g. swipe, tap) and system responds within 2 seconds.**Customer Satisfaction:** 5**Customer Dissatisfaction:** 0**Priority:** High**Conflicts:** None**Supporting Material:** None**History:** Created October 6, 2024**Requirement # 2****Requirement Type:** 9**Event/Use Case #:** 2**Description:** The system allows the users to select song features (e.g. tempo, genre), and it returns recommendations.**Rationale:** User wants song recommendations based on desired features.**Originator:** Requirement Analyst**Fit Criterion:** User can select features and does receive song recommendations.**Customer Satisfaction:** 5**Customer Dissatisfaction:** 0**Priority:** High**Conflicts:** None**Supporting Material:** None**History:** Created October 6, 2024

**Requirement # 3****Requirement Type:** 9**Event/Use Case #:** 3**Description:** The system generates a song based on reference song(s) or snippet(s) received from the user as input.**Rationale:** Users need an easy way to create music, without prior knowledge, that is similar to their input songs.**Originator:** Requirement Analyst**Fit Criterion:** Music is generated from the input reference song(s) or snippet(s).**Customer Satisfaction:** 5**Customer Dissatisfaction:** 0**Priority:** High**Conflicts:** None**Supporting Material:** None**History:** Created October 6, 2024**Requirement # 4****Requirement Type:** 9**Event/Use Case #:** 4**Description:** The system will analyze a reference song or snippet, and provide its features.**Rationale:** Users, more particularly music producers and educators, need a way to break down songs for a more detailed analysis of their features.**Originator:** Requirement Analyst**Fit Criterion:** User is provided with various features and visualizations showing an accurate breakdown of their song or snippet.**Customer Satisfaction:** 5**Customer Dissatisfaction:** 0**Priority:** High**Conflicts:** None**Supporting Material:** None**History:** Created October 11, 2024

**Requirement # 5****Requirement Type:** 9**Event/Use Case #:** 4**Description:** The system will analyze a reference song or snippet, and provide its visualizations.**Rationale:** Users, more particularly music producers and educators, need a way to visualize the breakdown of songs for a better understanding and analysis.**Originator:** Requirement Analyst**Fit Criterion:** User is provided with various visualizations that accurately represent their song or snippet.**Customer Satisfaction:** 5**Customer Dissatisfaction:** 0**Priority:** High**Conflicts:** None**Supporting Material:** None**History:** Created October 11, 2024**Requirement # 6****Requirement Type:** 9**Event/Use Case #:** 5**Description:** Users want recommendations based on reference song(s) and/or snippet(s).**Rationale:** Users want to discover new music or want music similar to the ones they are listening to.**Originator:** Requirement Analyst**Fit Criterion:** The system returns a list of recommendations.**Customer Satisfaction:** 5**Customer Dissatisfaction:** 0**Priority:** High**Conflicts:** None**Supporting Material:** None**History:** Created October 6, 2024

**Requirement # 7**  
**Requirement Type:** 9  
**Event/Use Case #:** 2,3,4,5  
**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 October 6, 2024

## 10 Look and Feel Requirements

### 10.1 Appearance Requirements

*Insert your content here.*

### 10.2 Style Requirements

*Insert your content here.*

## 11 Usability and Humanity Requirements

### 11.1 Ease of Use Requirements

*Insert your content here.*

### 11.2 Personalization and Internationalization Requirements

*Insert your content here.*

### **11.3 Learning Requirements**

*Insert your content here.*

### **11.4 Understandability and Politeness Requirements**

*Insert your content here.*

### **11.5 Accessibility Requirements**

*Insert your content here.*

## **12 Performance Requirements**

### **12.1 Speed and Latency Requirements**

*Insert your content here.*

### **12.2 Safety-Critical Requirements**

*Insert your content here.*

### **12.3 Precision or Accuracy Requirements**

*Insert your content here.*

### **12.4 Robustness or Fault-Tolerance Requirements**

*Insert your content here.*

### **12.5 Capacity Requirements**

*Insert your content here.*

### **12.6 Scalability or Extensibility Requirements**

*Insert your content here.*

## **12.7 Longevity Requirements**

*Insert your content here.*

# **13 Operational and Environmental Requirements**

## **13.1 Expected Physical Environment**

*Insert your content here.*

## **13.2 Wider Environment Requirements**

*Insert your content here.*

## **13.3 Requirements for Interfacing with Adjacent Systems**

*Insert your content here.*

## **13.4 Productization Requirements**

*Insert your content here.*

## **13.5 Release Requirements**

*Insert your content here.*

# **14 Maintainability and Support Requirements**

## **14.1 Maintenance Requirements**

*Insert your content here.*

## **14.2 Supportability Requirements**

*Insert your content here.*



### **14.3 Adaptability Requirements**

*Insert your content here.*

## **15 Security Requirements**

### **15.1 Access Requirements**

*Insert your content here.*

### **15.2 Integrity Requirements**

*Insert your content here.*

### **15.3 Privacy Requirements**

*Insert your content here.*

### **15.4 Audit Requirements**

*Insert your content here.*

### **15.5 Immunity Requirements**

*Insert your content here.*

## **16 Cultural Requirements**

### **16.1 Cultural Requirements**

*Insert your content here.*

## **17 Compliance Requirements**

### **17.1 Legal Requirements**

*Insert your content here.*

## **17.2 Standards Compliance Requirements**

*Insert your content here.*

## **18 Open Issues**

*Insert your content here.*

## **19 Off-the-Shelf Solutions**

### **19.1 Ready-Made Products**

*Insert your content here.*

### **19.2 Reusable Components**

*Insert your content here.*

### **19.3 Products That Can Be Copied**

*Insert your content here.*

## **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.

## 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.

## 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.

## 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.

## 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.

*Insert your content here.*

## 20.6 Follow-Up Problems

*Insert your content here.*

# 21 Tasks

## 21.1 Project Planning

*Insert your content here.*

## **21.2 Planning of the Development Phases**

*Insert your content here.*

## **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 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 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, 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.

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 advanced features.

- **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

1. The recommendation system will be able to recommend songs from less popular music genres (jazz, blues, etc.)
2. The analysis system will be able to extract musical features from less popular music genres (jazz, blues, etc.)
3. The generative system will be able to generate songs from less popular music genres (jazz, blues, etc.)
4. The recommendation system will be able to recommend songs from unpopular music genres (gnawa, libyan funk, etc.)
5. The analysis system will be able to extract musical features from less popular music genres (gnawa, libyan funk, etc.)
6. The generative system will be able to generate songs from less popular music genres (gnawa, libyan funk, etc.)
7. The recommendation system will be able to search for songs with cover art similar to an input song
8. The generative system will be able to generate new cover art for a newly generated song, based on the user's input criteria
9. The generative system will be able to generate new covert art for an existing song

## 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 generate variations or receive recommendations. This integration ensures a smooth user experience, allowing seamless interaction between existing music libraries and the platform's generative capabilities.



## 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 utilization 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?

**Music Analysis and Signal Processing:** To acquire knowledge and skills in Music Analysis and Signal Processing, we can take on-line courses focused on audio signal processing and machine learning

for music, building a solid theoretical foundation. We can also engage in discussions with university professors knowledgeable in this field. Additionally, we will take a look into analyzing audio data and implementing models using libraries like Librosa.

**Frontend or Backend Development:** For knowledge and skills in Frontend or Backend Development, we will look at documentation for backend frameworks like Django or Flask to learn how to build and manage the recommendation system. We will collaborate with teammates to share knowledge and enhance our skills.

**UI/UX and Design:** To acquire knowledge and skills in UI/UX and Design, we can read books and articles on UI/UX best practices to deepen our understanding of design principles. We can conduct user testing sessions with prototypes to gather feedback and iterate on the design.

**Music Generation and AI:** To acquire knowledge and skills in Music Generation and AI, we can will read research papers on generative models in music to understand their applications. Using Python libraries, we can test various algorithms to gain practical experience.

**Team Management and Infrastructure:** For Team Management and Infrastructure, the team will read books and articles on effective team leadership and management strategies, allowing us to understand different team dynamics and communication styles. We will also draw on our past experiences with team managers from co-op terms. Additionally, studying documentation on local server management and security best practices will help to establish a strong foundation for the project's infrastructure.