Problem Statement and Goals
Software Engineering

Team 8, Rhythm Rangers Ansel Chen Muhammad Jawad Mohamad-Hassan Bahsoun Matth

Table 1: Revision History

Date	Developer(s)	Change
2024-09-21	Muhammad Jawad	Updated Problem section with Problem Definition and Problem Importance subsections.
Date2	Name(s)	Description of changes
...	...	...

1 Problem Statement

[You should check your problem statement with the problem statement checklist. —SS]

[You can change the section headings, as long as you include the required information. —SS]

1.1 Problem

1.1.1 Problem Definition

Current music recommendation systems, such as Spotify, often lack personalization and depth, leading to suboptimal suggestions. Furthermore, the ability to create music typically requires extensive knowledge of instruments, production software, or technical tools, limiting accessibility. This project aims to bridge that gap by offering a platform that allows users to receive tailored music recommendations and generate music without needing specialized training. By simplifying the creation and recommendation process, this platform democratizes music production for both professionals and casual users.

1.1.2 Problem Importance

This project holds significant value for a wide range of users. Music recommendation systems have millions of daily users, and improving the quality of

recommendations can greatly enhance user satisfaction. Additionally, by providing tools that make music creation more accessible, the project opens creative opportunities for individuals who might otherwise be excluded due to the technical barriers involved in music production. The platform is poised to impact both the music industry and hobbyists, facilitating innovation and creativity in music generation and exploration.

1.2 Inputs and Outputs

[Characterize the problem in terms of "high level" inputs and outputs. Use abstraction so that you can avoid details. —SS]

There are two main components of this project. They are both generative systems, one being a recommendation system, the other being a pseudo-music generator. There is also a song analysis system, necessary in order to get the recommendation system and generative systems working as they will most likely operate on the data that the song analysis system provides.

Inputs:
A "track" is defined as a full song. A snippet is defined as a fragment of a track. These are objects that contain a sound file and some labels describing the song.
High-Level: For system 1 (recommendation): a list of track(s).
For system 2 (music generation): a reference track or snippet.
For system 3 (analysis): a singular reference track.

Explanation: The main idea behind the inputs is essentially to make them all related to each other as they are fundamentally connected within the system. This means that roughly, all the files are either lists containing the track as an object and a snippet is functionally under the hood a track, simply labelled differently for categorization purposes. This allows the system to share behaviors along the sub systems for faster implementation and design. As for format specifically, we can pull these from the spotify API song previews, API song features.

Outputs:
For System 1 (recommendation): A list of track(s), indexed with a similarity score.
For System 2 (music generation): a snippet.
For System 3 (analysis): a breakdown of the main aspects of the track.

Explanation: for system 1, it should output a list of recommended tracks and some form of a similarity score as a way to justify itself. This score could be derived on factors such as genre, sound signature, artist, etc.

For system 2, the snippest is an object containing the sound file.

For system 3, a score that categorizes the main features of the song, such as the musical range, genre, rythm, etc.

1.3 Stakeholders

The stakeholders for this project include:

- **Music Producers**: Professionals who are looking to generate new ideas, experiment with different genres, and augment their existing works.
- **Hobbyist Musicians**: Individuals interested in exploring and tinkering with familiar sounds and experimenting with new, creative compositions.
- **Music Theorists**: Users who seek to study and analyze different musical elements (e.g., pitch, rhythm) as a part of their research or hobby.
- **Audio Engineers**: Audio experts who can use the system to better understand sound characteristics and possibly optimize their workflows.
- **Independent Artists and Musicians**: Artists looking for accessible and affordable tools to experiment with and create music.
- **Music Educators**: Teachers seeking innovative ways to introduce students to music theory and composition.
- **Content Creators and Streamers**: Creators in need of unique music or soundtracks for their content.
- **Sound Designers**: Professionals working in film, games, or other industries where custom soundscapes and music are needed.
- **DJs**: DJs who want to generate new tracks and remixes based on existing music for performances.
- **Music App Developers**: Developers looking to integrate music recommendation and generation into their own apps.
- **Music Therapy Practitioners**: Therapists using music to help clients and generate tailored soundscapes for treatment.
- **Casual Music Listeners**: Users who want to discover and generate music for personal enjoyment.
- **Record Labels**: Companies interested in discovering new trends and helping artists explore experimental genres.

1.4 Environment

We strive to launch an on-premise server operating with a version of Ubuntu server, likely the most recent version, 24.04.1. The server shall respond to and process requests from a web-application front-end, making the service accessible to many different devices, but requiring a network.

2 Goals

Goals	Importance
The system shall adequately process and respond to requests involving widely-published music genres, e.g., pop, hip-hop, and rock.	Widely-published music genres have the largest corpus of data that can be used to train the featurization and generation mechanisms of the system, i.e., the system must perform favourably in tasks that it is well-trained on.
The system shall generate tabular features that correspond to characteristics of the input song (snippet), akin to those of Spotify, e.g., danceability, instrumentalness, and energy.	Structured tabular data can be rapidly process, making the task of song recommendation more efficient and song generation more explainable.
The system shall produce a list of songs that are similar to a single song provided or a collecetion of songs provided by the user.	This is a core feature of the system. Its inclusion should facilitate users to explore a music genre or "sound" of interest.

Song Analysis

- **Explanation:** GenreGurus will analyze a song or snippet and extract important musical features. The data gathered from this analysis will then be fed into the recommendation and generation systems, ensuring they work with accurate data about each song's musical features.
- **Reasoning:** By providing a detailed analysis, users can better understand the components of a given song. The analysis also ensures that the recommendations and generated music are based on actual musical data, which in turn improves the system's accuracy and output quality.

Music Generation

- **Explanation:** GenreGurus will allow users to input one or more reference songs or snippets and generate new music based on the features of these inputs. Users can adjust certain musical characteristics, and the system will produce an original track reflecting these changes.
- **Reasoning:** Users will be able to create and customize music through AI without needing extensive musical knowledge, making the platform more accessible while still appealing to expert musicians.

User Customizable Recommendations

- **Explanation:** GenreGurus will allow users to adjust the musical features of the input clip/song. Based on these adjustments, the system will update its recommendations in real-time.
- **Reasoning:** Customizable recommendations give users more control over the output, increasing user engagement and user satisfaction.

User Centric Design and Interface

- **Explanation:** GenreGurus will include a clean, intuitive interface where users can easily access the music recommendation, generation, and analysis features. The UI will be designed to require minimal understanding of how music adjustments work for a better user experience.
- **Reasoning:** An accessible and simple interface will appeal to a broader audience, from casual listeners to professionals.

Supportive of Many Music Genres

- **Explanation:** GenreGurus will include a variety of popular genres of music.
- **Reasoning:** This will allow users to customize and explore their favorite genres of music.

3 Stretch Goals

Goals	**Importance**
The system shall adequately process and respond to requests involving *not-as-widely*-published music genres, e.g., jazz, funk, and blues.	Such music genres have a smaller corpus of data that can be used for training, hence the system may not perform as favourably in tasks that it is not very well-trained on, but the inclusion of such genres would allow access to a larger user-group.
The system shall generate tabular features that correspond to characteristics of the input song's *cover art*.	Cover art tends to capture, however abstractly, the mood, energy, and intent of a song or album, thus may contain tacit information that can be accessed with image processing.

Machine Learning Cover Art Generation

- **Explanation:** GenreGurus will use AI models to generate custom art based on the features of a generated song or user preferences. The art will visually reflect the song's mood, genre, style, etc.
- **Reasoning:** Music encompasses more than just using one's auditory senses; it is also a visual and emotional experience. By generating art that matches the music, the system offers more connection for creators, appealing to both their auditory and visual senses.

Supportive of Many Music Genres

- **Explanation:** GenreGurus will explore a variety niche genres.
- **Reasoning:** This will allow users to explore genres they've never heard or experienced before. Which will engage the users more.

4 Challenge Level and Extras

The project is of a *general* challenge level.

- It requires domain knowledge about signal (audio) processing, music theory, learning models, generative models, and infrastructure setup.
- Its implementation is non-trivial, incorporating algorithm implementations, training and testing models, assessing their performance, automating the extraction- processing-storage workflow and the live-response workflow.
- The system is not particularly novel. Recommender systems are not new, but we are attempting to find and use features to create a better recommender system. The generative component has been done before with images and video, so scaling down to audio and frequency should be attainable, especially as it is a field that was researched quite deeply even before the advent of neural network-based generative techniques.

Project will include extras like user & API Documentation for ease of reference, usability testing for easy startup, and design thinking to build an intuitive user interface.

Appendix — Reflection

[Not required for CAS 741 —SS]

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

   Ansel The main disagreements we had were the project selection. To resolve this, we used a project selection matrix where we rated the proposed projects by a large amount of criteria and then discussed which projects we felt were the most interesting, feasible, or doable. We ended up eliminating some projects due to this. We then proceeded to use a strawpoll in order to sort out which remaining projects were our top choices, and which ones were projects where there was 1 team member who simply was not interested in at all. This lead us to select the GenreGuru project. The documentation allowed us to neatly organize our thoughts on each project and to compare them in the most fair and honest method possible.

   We also predicted that editing a latex document and then resolving conflict through github could be potentially an issue, which we resolved by having each member edit one section of the document and then using pull requests to review our changes before having one person handling merging all our edits into the final .tex file. This helped us stay organized and not cause headaches with merge conflicts.

   did you resolve them?

   Bahsoun During this deliverable, we anticipated challenges, so we approached the task by dividing the workload among our team. Each member volunteered to focus on a specific section while contributing thoughts during collaborative discussions. After completing our individual sections and reviewing each other's work, we convened for a call to share our feedback on what we liked and what needed adjustments. Overall, while the deliverable went smoothly, we did encounter some pain points. One critical challenge was selecting the right technologies, particularly in deter-

mining which frameworks and machine learning libraries to use. To tackle this, each member conducted research to evaluate the strengths of various options, allowing us to narrow down our choices effectively. Another challenge arose from team members' busy schedules, making it difficult to coordinate in-person meetings. To overcome this, we prioritized communication throughout the week, ensuring everyone stayed informed. Instead of requiring everyone to attend every meeting, we arranged for a few members to attend and then relay key points to those who couldn't make it. This approach helped keep the entire team aligned and engaged.

3. How did you and your team adjust the scope of your goals to ensure they are suitable for a Capstone project (not overly ambitious but also of appropriate complexity for a senior design project)?