Development Plan
Software Engineering

Team 8 – Rhythm Rangers

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

Table 1: Revision History

Date	Developer(s)	Change
2024-09-24	All members	Complete Revision 0

1 Team Meeting Plan

Team meetings will be scheduled in a relatively ad-hoc fashion. Members have shared their daily schedules throughout the working week with each other, and weekly availability notes are accounted for when scheduling meetings. Availability notes are to be shared before the start of the working week so a week's meeting plan can be drafted and voted on by the end of Monday. Team meetings can occur for:

- task delegation for upcoming deliverables & discussing deliverable progress

  we hope to make these meetings brief and infrequent in the future by using GitHub Projects and asynchronous communication instead;

- work sessions to collaborate and discuss ideas (deliverable-related) synchronously, *preferably in-person*;

- pair programming;

- conducting deliverable reviews;

- conducting deliverable retrospectives, i.e., reflecting on successful and unsucessful practices used in the most recent deliverable after its completion.

2 Team Communication Plan

All team communication is done through a discord server. The discord has three text channels:

- General (anything not project-related is posted here)
- Important Updates (anything related to weekly meeting availability goes here)
- Locked In (anything related to project work goes here)

The discord also has 2 voice channels:

- Weekly Meeting (administrative meetings happen here)
- Locked In (collaboration of project development happens here)

3 Team Member Roles

- Ahmed: Meeting manager and scheduler, developer
- Ansel: Team liaison, developer
- Matthew: Developer
- Mohammed-Hassan: Developer
- Muhammad: Developer

4 Workflow Plan

- Git standards
  - Development branches will follow the "dev/*name*/*description*" naming convention
  - Documentation branches will follow the "docs/*name*/*description*" naming convention
  - A feature branch will be created for every major component of the project
  - Developers will base their branches off the feature branch they are contributing towards
  - Developers will create pull requests that concisely explain what the pull request is doing, with any useful information being written in the PR description
- Issue standards

- Issues are created during usability testing

  - Issues must contain a comparison between what is expected of the code and what is actually seen by the user

  - If possible, issue creators should attach any log files they collect to the issue

  - The issue must be assigned to the developer who originally pushed the code that is causing the bug

5 Proof of Concept Demonstration Plan

There are two main risks to the project — the song data collection and song generation.

5.1 Risks regarding song collection

- License acquisition may be necessary for some songs;

  - Acquisition of the song in general may require a license from the artist, label, publisher, or platform, e.g., Spotify;

  - Platform providing songs may have limited API access.

- Songs may be only partially accessible, e.g., song snippets from Spotify;

- Songs may be available but we are prohibited from using them to train a machine learning model.

These risks can be dismissed if the project is to use different, less strict, song providers, or tailor the project to only use non-copyrighted songs.

5.2 Risks regarding song generation

- The generative mechanism will inherently be a machine learning model, which entails issues,

  - The model may hallucinate and produce unexpected outputs, i.e., music of the wrong genre (particularly of concern if training data is unbalanced), or just uncomfortable nonsensical sounds. *This could be a result of too little data to train a complex model (resulting in high variance), or too simplistic of a model (resulting in high bias)*;

- The model will be challenging to formulate, e.g., establishing architecture, objective function, and optimizer;

- The model will be so complex, i.e., will contain many parameters, such that it requires tremendous quantities of data and training time to converge to sensical results.

These risks cannot be entirely dismissed, but can be remedied greatly by considering the work of previous similar works and following their process, i.e., reusing architecture, data, or training mechanism. Nonetheless, this project is doable, as a parallel of it was completed in 2017.

6   Expected Technology

The technologies and tools expected for this project include:

- **Programming Language:** Python, due to its vast libraries in machine learning and audio processing, such as `librosa` and `pydub`.

- **Libraries:**
    - `librosa`: For music and audio analysis.
    - `pydub`: For audio processing and manipulation.
    - `scikit-learn` and `TensorFlow`: For building machine learning models to classify and generate music.

- **Frameworks:**
    - `Flask` or `Django`: For the web-based interface, allowing users to interact with the system.
    - `PyTorch` or `TensorFlow`: For implementing deep learning models to generate and classify music.

- **External APIs:** Spotify API will be used for fetching song previews, features, and other metadata for recommendation purposes.

- **Pre-trained Models:** We will leverage some pre-trained models for audio generation, such as OpenAI's Jukebox or similar publicly available models, while customizing them to fit our needs.

[What programming language or languages do you expect to use? What external libraries? What frameworks? What technologies. Are there major components of the implementation that you expect you will implement, despite the existence of libraries that provide the required functionality. For projects with machine learning, will you use pre-trained models, or be training your own model? —SS]

[The implementation decisions can, and likely will, change over the course of the project. The initial documentation should be written in an abstract way; it should be agnostic of the implementation choices, unless the implementation choices are project constraints. However, recording our initial thoughts on implementation helps understand the challenge level and feasibility of a project. It may also help with early identification of areas where project members will need to augment their training. —SS]

Topics to discuss include the following:

- Specific programming language

- Specific libraries

- Pre-trained models

- Specific linter tool (if appropriate)

- Specific unit testing framework

- Investigation of code coverage measuring tools

- Specific plans for Continuous Integration (CI), or an explanation that CI is not being done

- Specific performance measuring tools (like Valgrind), if appropriate

- Tools you will likely be using?

7 Coding Standard

8 Project Scheduling

[How will the project be scheduled? —SS]

Appendix — Reflection

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. Why is it important to create a development plan prior to starting the project?
   - Creating a development plan is crucial for a few reasons:

     - It gives the project a clear direction and scope since all goals are outlined
     - It sets boundaries on the project to prevent unplanned expansions
     - It outlines anticipated challenges and contingency plans
     - It provides transparent communication of the project expectations to the stakeholders

2. In your opinion, what are the advantages and disadvantages of using CI/CD?

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?