

# 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

[Put your introductory blurb here. Often the blurb is a brief roadmap of what is contained in the report. —SS]

[Additional information on the development plan can be found in the lecture slides. —SS]

## 1 Confidential Information?

[State whether your project has confidential information from industry, or not. If there is confidential information, point to the agreement you have in place. —SS]

[For most teams this section will just state that there is no confidential information to protect. —SS]

## 2 IP to Protect

[State whether there is IP to protect. If there is, point to the agreement. All students who are working on a project that requires an IP agreement are also required to sign the “Intellectual Property Guide Acknowledgement.” —SS]

## 3 Copyright License

[What copyright license is your team adopting. Point to the license in your repo. —SS]

## 4 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 unsuccessful practices used in the most recent deliverable after its completion.

[How will the meetings be structured? There should be a chair for all meetings. There should be an agenda for all meetings. —SS]

## 5 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)

## 6 Team Member Roles

- Ansel Chen: Team liaison, developer
- Muhammad Jawad: Developer
- Mohamad-Hassan Bahsoun: Developer
- Matthew Baleanu: Developer
- Ahmed Al-Hayali: Meeting manager and scheduler, developer

## 7 Workflow Plan

### 7.1 General Workflow

- Issues are created, assigned, and attached to the project. Issues will have a template akin to those found on [stevemao/github-issue-templates](https://github.com/stevemao/github-issue-templates). These issues should pertain to deliverable sections, split into a completion assignee and a reviewer;
- The `main` branch is protected, so team members must work in independent branches. We hope to restrict branch naming to a standardized format, e.g., [this](#) or akin to [conventional commits](#) (which comes with [a linter](#)!);

- Team members' commits must follow the [conventional commits](#) standard;
- Whenever necessary, a team member can choose to merge their changes to the `main` branch with a pull request. Pull requests should have a template akin to the simple pull request found [here](#) or the more complicated [data-centric template from dbt](#). A pull request should be attached to the [GenreGuru Project on GitHub](#) with a *dedicated* reviewer and potentially a review timeline and checklist. Having only one reviewer avoids the issue of [diffusion of responsibility](#).

## 7.2 Usability Testing

- How will you be using git, including branches, pull request, etc.?
- How will you be managing issues, including template issues, issue classification, etc.?
- Use of CI/CD

## 8 Project Decomposition and Scheduling

- How will you be using GitHub projects?
- Include a link to your GitHub project

[How will the project be scheduled? This is the big picture schedule, not details. You will need to reproduce information that is in the course outline for deadlines. —SS]

## 9 Proof of Concept Demonstration Plan

**Demonstration** Our reduced-scope demonstration should illustrate the ability to acquire data programmatically, i.e., accessing MP3 files from [Deezer](#) or [Spotify](#)'s APIs, and producing a single musical feature of the song, e.g., pitch, timbre, or whatever other feature is found to be representative of songs and relatively easy to implement.

**Post-demonstration** A successful demonstration will allow us to later scale up our data ingestion to satisfy multiple users' requests automatically, and provide songs' musical features on demand, expanding the available features incrementally. If the featurization/feature engineering component works correctly, a song recommendation component can later be integrated by using song features as distance metrics.

There is one main risk to the project — the song data collection.

## 9.1 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;
- API rate limits may restrict the number of requests we can make within a specific time period, impacting scalability and usability for demonstration.

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.

## 10 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 may 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.

- **Linters:** a CI-integrable Python-specific linter, e.g., `pylint`, `ruff`, or `flake8`. There also are git-specific linters like [conventional commit's linter](#).
- **Formatter(s):** a CI-integrable formatter, e.g., `black`.
- **Type-checker(s):** a CI-integrable type-checker, e.g., `mypy`.
- **Documentation Generation:** a CI-integrable documentation generator, e.g., `sphinx`.
- **Tester(s):** a CI-integrable testing framework, e.g., `PyTest`.
- **Continuous Integration:** the 5 previous “CI-integrable” components are a good baseline, but we plan on using a local server, and deployment onto it with CI would be sweet. As much automation with CI, within reason, is desirable as it is excellent experience to be transferred to a working setting.
- **Environment Management:** environment managers for Python like `pipenv` and `poetry` are excellent for ensuring the program works the same everywhere, installation becomes straightforward, and CI becomes marginally easier because of more standardization.

[\[git, GitHub and GitHub projects should be part of your technology. —SS\]](#)

## 11 Coding Standard

The coding standards will be a series of PEPs, but importantly [PEP 8](#) for coding style and [PEP 257](#) for docstrings that will form the basis for the API and user reference. For simplicity, a [package-like folder structure](#) should be used in conjunction with the provided template.

## 12 Project Scheduling

There will be no GANTT charts - we will conduct weekly meetings as “standups” and decide what is to be worked on. Please refer to section 1, “Team Meeting Plan” for details.

## 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?

CI/CD is a wonderful tool that automates mind-numbing tasks of linting, styling, formatting, deployment, and document generation, for example, but it comes at the heavy cost of relatively complicated setup process for said CI/CD to work. Thankfully, there are many GitHub actions templates online, so we often need not to worry about starting from scratch.

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

There were few disagreements, but the frequency of meetings is a hotly-contested topic. We chose to have relatively frequent meetings during the first 3 weeks of the project, will experiment with far fewer in the future and rely on asynchronous communication. Depending on the success of it during the next few weeks, we will continue or adjust it to better suit our needs in completing future deliverables successfully.

## Appendix — Team Charter

[borrows from University of Portland Team Charter —SS]

### External Goals

[What are your team’s external goals for this project? These are not the goals related to the functionality or quality fo the project. These are the goals on what the team wishes to achieve with the project. Potential goals are to win a prize at the Capstone EXPO, or to have something to talk about in interviews, or to get an A+, etc. —SS]

### Attendance

#### Expectations

[What are your team’s expectations regarding meeting attendance (being on time, leaving early, missing meetings, etc.)? —SS]

#### Acceptable Excuse

[What constitutes an acceptable excuse for missing a meeting or a deadline? What types of excuses will not be considered acceptable? —SS]

### In Case of Emergency

[What process will team members follow if they have an emergency and cannot attend a team meeting or complete their individual work promised for a team deliverable? —SS]

### Accountability and Teamwork

#### Quality

[What are your team’s expectations regarding the quality of team members’ preparation for team meetings and the quality of the deliverables that members bring to the team? —SS]

#### Attitude

[What are your team’s expectations regarding team members’ ideas, interactions with the team, cooperation, attitudes, and anything else regarding team member contributions? Do you want to introduce a code of conduct? Do you want a conflict resolution plan? Can adopt existing codes of conduct. —SS]



**Stay on Track**

[What methods will be used to keep the team on track? How will your team ensure that members contribute as expected to the team and that the team performs as expected? How will your team reward members who do well and manage members whose performance is below expectations? What are the consequences for someone not contributing their fair share? —SS]

[You may wish to use the project management metrics collected for the TA and instructor for this. —SS]

[You can set target metrics for attendance, commits, etc. What are the consequences if someone doesn't hit their targets? Do they need to bring the coffee to the next team meeting? Does the team need to make an appointment with their TA, or the instructor? Are there incentives for reaching targets early? —SS]

**Team Building**

[How will you build team cohesion (fun time, group rituals, etc.)? —SS]

**Decision Making**

[How will you make decisions in your group? Consensus? Vote? How will you handle disagreements? —SS]