Development Plan
ScoreGen

Team #7, Tune Goons
Emily Perica
Ian Algenio
Jackson Lippert
Mark Kogan

Table 1: Revision History

Date	Developer(s)	Change
2024/09/24	Mark, Ian, Emily, Jackson	Initial Revision
...	...	...

The development plan contains our full overview of our goals and the subsequent methods we've chosen to accomplish them. It serves as a guide which will be referenced by each developer throughout the product cycle to create a standard of expected protocol, behavior, goals, and development practices. In this document, we have outlined how we will conduct team meetings, as well as the meeting frequency. Further outlined is communication guidelines, team member roles, development protocols, expected tech, deliverable timelines and our individual and group reflections. Overall this document shows how the group plans to develop the product through an initial proof of concept, followed by a minimal viable product and continuous development to improve the result until it satisfies our intended goals.

1 Confidential Information?

There is no confidential information needing protection for our project.

2 IP to Protect

There is no IP to protect for our project.

3 Copyright License

Our team will be adopting the MIT License, which is standard for allowing usage without limitation to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software. This is due to the nature of our project, which is focused on learning and not for commercial use.

4 Team Meeting Plan

4.1 General Team Meetings

Meeting Chair: Jackson Lippert.
Frequency: Weekly.
Format: In-person.
Location: McMaster's main campus.

- Particular location may vary, will be discussed prior to the meeting time.

Time (primary): Monday, 2:30pm-4:20pm (the designated tutorial time slot for SFWRENG 4G06A/B).
Time (secondary): Monday, 12:30pm-1:30pm.

- The secondary time slot will be used if the primary slot is occupied by synchronous tutorials or at the discretion of team members.

Structure:

AGENDA ITEMS (GENERAL MEETING)
1. Discuss/review team member updates. (a) Recent changes made. (b) Problems encountered. (c) Status of previous meeting action items. 2. Go over new business. (a) Discuss future deliverables/tasks to be done. (b) Define individual and/or group next steps. 3. Assign action items.

4.2 Supervisor/Industry Advisor Meetings

Frequency: Biweekly.
Format: In-person, virtual if necessary.
Location: Dr. Martin v. Mohrenschildt's campus office.
Time: By appointment.
Structure:

AGENDA ITEMS (SUPERVISOR MEETING)
1. Discuss/review team updates. (a) Project milestones reached. (b) Status of previous meeting action items. 2. Troubleshoot recent issues that require supervisor-specific knowledge. 3. Determine high-level plan for project progression. (a) Discuss problems the team expects to encounter in the near future. (b) Identify goals the team wishes to achieve by the next meeting with the supervisor. 4. Assign action items (if applicable). 5. Discuss details of next supervisor meeting. (a) Date & time. (b) Location.

Note: All fields are subject to change depending on the supervisor's availability or the team's need to consult the supervisor.

5 Team Communication Plan

5.1 General Purpose Communication

The primary mode of communication for general purposes will be through Apple iMessage SMS/MMS in the team group chat. In the event that a team member does not respond via iMessage, the secondary mode of communication will be used. This will be through an Instagram direct messages group chat. The primary and secondary communication channels were chosen due to their practicality, team members will almost certainly have their smartphone on hand, which they can use for both iMessage and Instagram direct messages. A tertiary form of communication through Microsoft Teams direct messaging and/or video calls. This will be used in the event a team member does not respond within a reasonable time frame to primary and secondary communication methods. It will also be the main application through which virtual meetings will take place (both general and supervisor meetings) as it has screen sharing capabilities and is familiar to all team members.

5.2 Implementation and Task Assignment

GitHub Projects will be the primary tool used for project management related communication. A kanban board will be collectively maintained by the team for task assignment, progress status, meeting bookkeeping, and item reviews. GitHub Issues will be used in tandem with the kanban board. As such, every opened issue will be required to also appear on the kanban board. In addition to project tasks and to-dos, issues will also be opened for every team meeting and every SFWRENG 4G06A/B lecture to track attendance and team contribution. There are five categories of issues that can be opened by the team, each having their own template:

- Lecture
  - Date and topic
  - Attendance tracker
  - Key points discussed
  - Action items for team
- Meeting
  - General team meetings
    - Date, time, location
    - Attendees
    - Agenda items
    - Discussion points
    - Assigned action items
  - Supervisor meetings
    - Date, time, location
    - Progress update
    - Technical questions
    - Next steps
  - TA meetings
    - Date, time
    - Current concerns
    - Feedback received
    - Required changes
- Peer Review
  - PR number and description
  - Code quality checklist
  - Testing verification

  - Documentation review
  - Suggested changes

The use of these templates standardize issue creation and streamlines project progression.

6 Team Member Roles

All listed team member roles are tenative and subject to change at the discretion of the team. Should role changes occur team members should follow the process outlined below:

1. Contact all team members that are relevant to the role change.
2. Discuss proposed changes and ensure each team member understands their new responsibilies.
3. Notify uncontacted team members of the change and/or the overtaking of responsibilities.

6.1 Project Manager

Assignee: Emily Perica
The project manager directs the team towards successful completion of the project. They will be responsible for coordinating the progression by creating meeting agendas for the meeting chairperson to follow. They will also maintain the Kanban board and create and assign GitHub issues that concern the entire team.

6.2 Meeting Chair

Assignee(s): Jackson Lippert
The meeting chair is responsible for driving team meetings towards productivity and efficiency. This will be done by guiding and directing meetings to minimize off-topic conversation. They are required to ensure all meeting agenda items are addressed in a timely manner. Finally, they will facilitate constructive, collaborative discussions during meetings.

6.3 Meeting Minutes Recorder*

Assignee(s): Ian Algenio
The primary responsibility of this role is to document and record team meetings. They will also track team member attendance and address associated issues. They will record meeting discussions, issues, topics, action items, etc.

6.4 Subject Matter Expert*

Assignee(s): Mark Kogan
The subject matter expert is given the responsibility of maintaining a neat and organized record of source materials used during the lifetime of the project. This may include academic articles, open-source repositories, external libraries used, etc. They will also create any necessary citations for project documentation elements.

6.5 Software Developer*

Assignee(s): ALL MEMBERS
All team members will take on the responsibilites of this role. As developers they are responsible for designing, testing, coding, and maintaining the software components of the project.

6.6 Quality Assurance and Testing Specialist*

Assignee(s): ALL MEMBERS
This role ensures the software meets stakeholder needs and quality standards. They will develop and execute test plans and test cases, measure code coverage, and report defects in the software. Additionally, they will verify the software meets functional and non-functional requirements such as performance, security, usability, etc.

6.7 Music Domain Expert

Assignee(s): Emily Perica
As the expert in the music theory domain, this person will address any confusion regarding general music theory such as terminology. They will share their prior knowledge and experience with music whether that's playing an instrument or clarifying details about the domain.

Note: Any team member roles marked with an asterisk (*) are able to be assigned to multiple people, have overlapping responsibilities with other roles, and/or have responsibilities are likely to be shared amongst the enrire team.

7 Workflow Plan

Issue Management Workflows
When creating an issue in the Capstone Kanban Board, there are two possible workflows:

1. Meeting issue is created (using the relevant template) prior to a scheduled meeting.

   (a) Issue is given the 'meeting' label and assigned the 'Meeting Minutes' status.
   (b) The notetaker (see section 6) will add a comment to the issue, outlining the meeting minutes and action items of each attendee.
   (c) Issue is closed once all action items have been addressed.

2. Project work issue is created using a blank issue.
   (a) Issue begins in 'To Do'
   (b) One or more appropriate labels are assigned - ‘documentation’, ‘deliverable’, ‘enhancement’, ‘bug’, 'DevOps',and potentially others as the need arises.
   (c) The tags will be used to filter issues on the Kanban board, and help ensure even distribution of workload in a more precise way than number of issues. Labels will also help inform task priority.
   (d) If the issue does not have the ‘deliverable’ label AND is not time sensitive, it will be moved to ‘Backlog’.
      i. The issue will move out of 'Backlog' and into 'To Do' if the issue becomes time sensitive or the team has the time/resources to take it on.
   (e) The issue moves to ‘In Progress’ once it is assigned and being worked on. It may be assigned to more than one person, depending on the scope of the issue.
      i. At this point, an issue pertaining to the codebase will branch to the Git Workflow (below).
   (f) The issue moves to ‘In Review’ once a PR has been made and approval is requested from the team.
   (g) The issue moves to ‘Done’ and can be closed once all associated tasks have been completed, including all related PRs being merged/closed.

In addition to the above usage of the Kanban board, all course deliverables are documented as milestones in the repository and will have issues assigned as necessary. Each milestone has a due date and a description of the tasks to be completed.

The Project Manager (see section 6 for team member roles) is responsible for making issues and keeping the project on track. As well, they will run monthly backlog grooming sessions with the team to ensure any stale issues are taken care of.

Git/GitHub Workflow

A new developer on the project will begin at step 1; otherwise step 1 can be skipped:

1. Developer may choose to fork or clone the repo. A fork is recommended so that contributors can display the project directly in their GitHub profile.

2. Main is a protected branch - any commits must be made in a separate branch before being pushed to a PR. There is no specific naming scheme for branches.
3. Commit and PR names will both start with 'Issue #: '. This will be enforced through a GitHub Action.
    (a) Each PR will only address a single issue, but an issue may be spread out across multiple PRs.
    (b) Squash commits are enforced to ensure there is a single commit per PR.
4. At least one reviewer must approve the PR before it can be merged into main.

CI/CD Workflow
Actions on Push/Pull Request:

- LaTeX to PDF converter
- Code packaged as an executable
- Linting
- Unit tests
- PR naming convention enforcer

Note that a PR may not be merged if any failures are present in the pipeline.

8 Project Decomposition and Scheduling

Link to our GitHub Project - Capstone Kanban Board.

Due Date	**Deliverable**
Sept 24/24	Problem Statement, POC Plan, Development Plan
Oct 9/24	Requirements Document Revision 0
Oct 23/24	Hazard Analysis
Nov 1/24	V&V Plan Revision 0
Nov 11/24*	Proof of Concept Demonstration
Jan 15/25	Design Document Revision 0
Feb 3/25*	Revision 0 Demonstration
Mar 7/25	V&V Report Revision 0
Mar 24/25*	Final Demonstration (Revision 1)
Apr ##/25 - TBD	Expo Demonstration
Apr 2/25	Final Documentation (Revision 1)

*The demonstration should be prepared by this date, but this will not necessarily be the date of the demo itself.

Ideally, progress on a deliverable will begin at least 3 days before the previous deliverable is due (i.e., D2 work should begin on or before Sept 21). The above schedule will be enforced through the use of Milestones in the project repository, such that each deliverable (including its due date and all relevant information) is its own Milestone. The only exception to this is the Expo Demonstration, which is not expected to require any specific issues.

Each deliverable will be split into issues based on dependencies and relevant tasks. For example, when writing this document a single issue consisted of sections 7 and 8. They both cover project management-related plans, so completing one section provides the writer with the needed knowledge to write the other section. An alternative proposed method was to create an issue for each section of the document, but the granularity of this strategy may make the document feel more disjointed to the reader.

9 Proof of Concept Demonstration Plan

The Proof of Concept will be a demonstration of the team's ability to develop the most crucial aspect of the final product, which is to take in a single note and output the note pitch and length. If successfully performed, it will clearly show that a viable product can be produced given enough time.

The main risk associated with the success of the project is potential nuance introduced by human performance on instruments. The product will have to adjust its notation based on parameters such as key signature and detected tempo to create most likely reasonable interpretations of music. It will become necessary to test and benchmark accuracy during audio edge cases.

The Proof of Concept will help dispel concerns by allowing the team to show the product's potential for accurate signal processing. For the sake of minimum viable product, more complex parameters such as key signautre and tempo will be assumed, however the task of determining note pitch and duration will be completed.

Given the ability to accurately process audio, the completion of the product will be simplified, as the focus can shift towards construction of a user interface and adding features while optimizing performance and accuracy.

10 Expected Technology

10.1 Tabular Overview

DOMAIN	TOOL/TECHNOLOGY	DESCRIPTION
Version Control	Git, GitHub	For version control and issue tracking respectively.
Project Management	GitHub Projects	Used for Kanban board, task assignment, backlogging, etc.
Back-End Development	C/C++, MATLAB	Possible specific languages, high performance and speed is required for signal processesing.
Sound File Conversion	libsndfile	A specific library that was researched and will likely be used for sampled sound file format conversion. One option is the libsndfile C library.
Code Formatting	Linter(s)	Specific tools are undetermined at this stage as it is language dependent. One potential option is clang-tidy To be used with CI workflows to format code.
Testing	MATLAB unittest/automation, CMake, Unity Test, etc.	Specific tools are undetermined, likely a unit testing framework that does not transcend languages (i.e. language-specific).
AI/ML	N/A	Will use existing libraries (pytorch, tensorflow, etc.) given the time constraint of the project, however, the team aims to generate our own datasets for training.
Front-End Development	HTML, CSS, JS, React Native	Languages and existing libraries deemed useful for web-based GUI. A framework will likely be used, in particular React Native.
UI/UX	Figma, Canva	To design and effectively prototype UI/UX elements.
Hardware	Audio Interface	An external unit for I/O of audio signals to the host device. Possible to use the host computer's built-in microphone as opposed to this.
Musical Instruments	Keyboard, guitar, ukelele, etc.	Used to generate input signals.

10.2 Continuous Integration Plans

GitHub Actions (GHA) will be used for continuous integration. They will employ automated workflows for various unit and integration tests as well as code formatting/linting. One specific workflow that will be created is .tex to .pdf format conversion to ease documentation editing. GHA will also ideally be used to test code coverage metrics. One of the major components of the project is the signal processing algorithm. Depending on the implementation details, different metrics in addition to the basic ones such as statement coverage might be used. For example, the designed algorithm might benefit from condition or path coverage for pitch detection.

11 Coding Standard

Our source code will adhere to the following widely adopted coding style guides:

- For **C++**, which will likely be our main coding language for the signal processing, we will follow Google's C++ Style Guide due to its widespread adoption.
- For **Python**, we will adhere to the PEP8 Style Guide.
- For **HTML** and **CSS**, we will implement the BEM Methodology.

All functions, classes, and modules must be well-documented using inline comments and docstrings where appropriate. Comments should provide context when necessary, rather than stating the obvious. This ensures clear communication and explanation of code between team members, making our pull requests easy to follow and the codebase more maintainable. Additionally, all pull requests must be reviewed by at least one other team member before being merged into the main branch. Reviews will focus on code quality, adherence to coding standards, and potential performance improvements.

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?
   Emily: The development plan that we've created in this document will serve as the groundwork that the rest of the project will be built off of. We've already seen what a difference a formalized plan can make compared with jumping in blindly - our first ideas for this project have already changed and been refined significantly since organizing our thoughts and anticipating our future needs. In creating this development plan we've been able to set realistic goals and discuss within the group each of our individual visions. One of the benefits of working in teams is combining different people's thought processes and background knowledge, and yet this may also provide a disadvantage when communication is lacking. By creating a solid plan ahead of writing any code, it allows us to ensure that our levels of communication are up to the standards expected from a final year capstone project.

   Ian: It ensures the team understands and thinks about the project, even if the thoughts are very abstract and lacking in detail. Without a development plan laid out the parts of the project are left to interpretation for each team member, which can very significantly. One of the best examples of this was the expected technology section. Most of the specific tools and libraries we will use aren't specified, but that's exactly the point. Even though we didn't set anything in stone like a "plan" typically would, we got to discuss possibilities, share expectations, and narrow down our choices. Without this, we would be left to deal with the consequences of not discussing expectations later down the road, when deadlines are approaching quickly and stress is high. Decisions made then would not have the opportunity to be refined like they do now, which, I think, would almost certainly impact the results of the project negatively.

   Jackson: Development plans are vital to projects even outside of the realm of software projects. In my personal experience, I have been a part of group projects in school where we didn't lay out what we were going to do and how we were going to divide the work, leading to many inefficiencies and missed requirements, resulting in loss of marks. All of that could have been prevented if we just laid out a solid development plan. Additionally, when projects come down to the deadline, wasting valuable development time on discussing what we should do leads to unnecessary stress which I have experienced. A case could be made for a situation where a development plan maybe shouldn't be fully documented and discussed, such as a proof-of-concept that could be done within the scope of a day.

Mark: The development plan will be a crucial point of reference for the entire lifespan of our project. The construction of complex systems relies on extremely tight cohesion between collaborating partners, and failures in communication can easily result in messy and difficult situations. The development plan is an objective overview of the intermediate and end goals of the product, which keeps each worker on the right track. This helps prevent potentially catastrophic miscommunications, such as different goals, different timelines, or different expectations of task delegation. In a collaborative product where communication is the difference between success and abject failure, the development plan is the starting point on which all future communication can be based.

2. In your opinion, what are the advantages and disadvantages of using CI/CD?
   Emily: CI/CD acts as the last barrier before deployment, and ensures that only well-tested code makes it to production. In our case, it will help us to maintain a uniformity across our codebase and prevent us from collecting tech debt as the project matures. That being said, it also will require a significant amount of effort to get running smoothly, both from the development side and the planning/project management side. It can also be resource intensive, which may be hard for a group of students to handle and maintain.

   Ian: It's similar to an overall development plan in the sense that it can act as a general guideline for the project. Whether that's automation of low-level tasks or testing. It's a great way to save time especially for projects with a time constraint like ours. An example would be enforcing PR conventions. Standardized and proper references to issues really helps cut down the time it takes to understand what a PR is for and why. CI/CD also has the advantage of reducing the number of errors introduced into the code base, again saving time that would be lost trying to fix these errors and conflicts. One of the disadvantages I'm worried about is how easy it is to focus too much on it. It's very new to me and I'm worried I will spend too much time trying to understand the fine details and perfecting actions rather than working on progressing the project goals themselves. I tend to do this with many things, maybe as a way of distracting myself from bigger issues. But reflecting on CI/CD helps me make sure I avoid this mistake.

Jackson: In my opinion, CI/CD can be extremely valuable in certain situations. For example, in the case of this project, it is valuable to see our merged pull requests being integrated immediately without the need for our hands-on management of deployments. This is great for saving time on a project that will last a long time because the value of CI/CD goes up the longer a project will be in its development phase. Some disadvantages could be the time it takes to set up and resolve issues that may arise because of it. This could take the form of a failed deployment, for a project which has a short development lifecycle, meaning that the time taken to set up CI/CD would be greater than the time-saving it provides.

Mark: The continuous integration pipeline is a valuable but resource-intensive element of a project. While its initial construction and continuous development require a significant time investment which is not directly related to product development, The CI pipeline helps prevent the introduction of bugs, poor design choices, or non-compliant code, resulting in a cleaner product and fewer setbacks during development. However, a downside is that more complex pipelines can take significant time to run, meaning developers might not be notified of failures until several minutes or even hours after a commit. This delay can lead to a choppy and extremely frustrating development experience. To work efficiently with a CI/CD pipeline, it's best to simplify individual commits while working, and to have alternative tasks to complete while the pipeline runs. One of the key advantages of CI is its automatic enforcement—it triggers on every commit or pull request and ensures that merges don't happen unless all tests pass. This ensures an ironclad adherence to rules and practices, which becomes even more critical as development teams grow.

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

   We had conflicting thoughts on whether or not to mark the addition of initial content as revisions in the revision tables. This was discussed quickly through the use of requesting changes on PRs, which perfectly highlights how important a development plan is since we decided how to structure additions to our repository with PRs. We also disagreed initially on how to name the project, which didn't prove to be much of an issue as we followed standard voting protocols to find a name we were all satisfied with.

There was another disagreement on whether or not to have real-time signal processing and sheet music generation as a stretch goal. This problem arose when we were discussing the feasibility of the feature, and if it would even be worth adding it as a stretch goal. We resolved this with a group discussion that led to us choosing to pursue it as a stretch goal, even if it isn't feasible, as a way to challenge ourselves.

Appendix — Team Charter

External Goals

Our team's primary external goal for this project is to maximize learning and skill development in software engineering. We are focused on creating a technically impressive and innovative project to earn the highest possible grade. Additionally, we want to ensure our project can be showcased in interviews and added to our professional portfolios and résumés. These goals will help us stand out when pursuing future opportunities in the tech industry.

Attendance

Expectations

Team members are expected to attend all scheduled meetings on time within reason. Leaving early is acceptable if communicated in advance and if that member has completed their contributions for the meeting. Additionally, all team members may leave if they feel the meeting's agenda has been completed. Missing meetings should be a rare occurrence, and any absences must be communicated at least 24 hours in advance when possible.

Acceptable Excuse

Acceptable excuses for missing a meeting or deadline include illness, family emergencies, or unavoidable academic conflicts (exams or critical project deadlines). Unacceptable excuses include forgetfulness, oversleeping, or personal plans that were not communicated ahead of time. All excuses must be communicated as soon as possible to the rest of the group.

In Case of Emergency

If a team member has an emergency and cannot attend a meeting or complete their work, they should notify the team as soon as possible via the agreed-upon communication method. If possible, the team member should provide updates on their progress and share any work that has been completed so far, so others can step in if necessary. In case of an emergency during critical project milestones, the team will redistribute tasks to ensure deadlines are met.

Accountability and Teamwork

Quality

All deliverables should be thoroughly tested, documented, and meet the project's technical and design standards before being submitted to the team. If a team member encounters difficulties, they should raise a GitHub issue for discussion before the meeting to avoid delays and ensure early feedback. Pull requests raised to change code will be reviewed by at least one other team member, ensuring only quality code is being deployed via CI. Finally, all team members will look over and edit each others' work for documentation, ensuring everything is clear to the reader.

Attitude

We will foster a positive, supportive, and professional team environment where everyone's ideas are heard and respected. During discussions, we will prioritize constructive feedback and aim to solve problems together. In every interaction, we will emphasize respect, inclusion, and clear communication. Any conflicts will be addressed through open discussions. If unresolved, a neutral team member may mediate, or the issue will be escalated to the TA.

Stay on Track

For progress tracking we will use GitHub's issue tracking and Kanban board features to manage tasks, ensuring that all assignments are transparent and visible to the team. Each team member will update the status of their tasks by moving issues across the Kanban board into their respective swim lanes. GitHub commits should reference corresponding issues, ensuring that work is traceable. A meeting chair is appointed to keep meetings on schedule and ensure the meeting agenda is followed.

Team Building

To strengthen our teamwork, members are encouraged to engage with each other informally outside the scope of this project. As a team ritual, we will celebrate milestones by acknowledging successes and accepting every failure gracefully and as a team.

Decision Making

We will aim for consensus in decision-making. If a consensus cannot be reached within a set timeframe, we will hold a majority vote. For major disagreements, we will discuss the pros and cons of each option and if needed, consult a TA, our supervisor, or the professors of the course.