Software Requirements Specification for ScoreGen: An audio-to-sheet-music generator

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

April 4, 2025

Contents

Revision History

Date	Version	Notes
11/10/2024	0	Initial Revision
30/01/2025	0.1	Issue #96
30/01/2025	0.2	Issue #92
30/01/2025	0.3	Issue #128
03/04/2025	1.0	Issue #309
04/04/2025	1.0	Issue #139

1 Purpose of the Project

1.1 User Business

Music theory is challenging and intimidating to learn for beginners, however, the act of playing an instrument does not inherently require extensive theoretical knowledge. This project aims to bridge the gap between deep theoretical understanding and playing instruments. It shall aid musicians with limited music theory knowledge by translating analog instrument audio into sheet music and other applicable formats that users can use to visualize and document their music-making. The project also aims to foster ease of collaboration between musical artists for efficient communication.

1.2 Goals of the Project

The goal of the project is to develop a fast, accurate sheet music generator paired with an intuitive, user-friendly interface that requires minimal effort to learn. If successful, it will encourage the proliferation of musical creativity among various audiences. This goal lends itself to several metrics, such as computational performance compared to existing products or its reach to certain demographics. Ultimately, the benefit of the project will be measured by the number of original pieces created.

2 Stakeholders

2.1 Client

The client is the faculty supervisor of this project, Dr. Martin Von Mohrenschildt. With direct research interests in signal processing and a personal interest in how it relates to music, he is investing his own time in the development of the product.

2.2 Customer

The archetypal customer is a musician of any skill level, with emphasis based on musicians who have little-to-no music theory knowledge. The customer will utilize the application to quickly produce sheet music for either original compositions or existing songs.

2.3 Other Stakeholders

- **Software developers:** Provide the technical knowledge and skill to design and build the system.
- **Composers:** As SMEs, provide the music theory knowledge needed to make the system useful for the customer.
- **Users of music-annotating software:** As SMEs, provide opinions on the features they like or dislike about existing music-annotating softwares. This will help inform the developers on which features of the product to prioritize.
- **Music teachers:** Provide similar support to students that may be replaced by usage of this product. They will have a vested interest in ensuring this product does not encourage bad composing habits in users of the product.
- **Music students:** Similar to the Composers, they will provide the music theory knowledge needed to make the system useful for the customer. However, their input will be from the same learning point of view as the expected user, and thus will have a higher degree of influence on the product.
- **IP lawyers:** Will ensure this product follows fair use and does not infringe on copyrights.
- **Dr. Spencer Smith:** As the course instructor, the performance and outcome of this product will be a direct reflection of his role in the capstone course.

2.4 Hands-On Users of the Project

User Category 1: Beginner Musicians

User Role: Use this product to create sheet music, either as a learning aid or as a quick alternative to learning music notation.
Subject matter experience*: Novice - Little experience with music theory, but likely have a moderate knowledge of music in practicality (i.e., playing an instrument or listening to music).

Technological experience*: Novice - May or may not have limited experience with other music-annotating softwares.
User characteristics:

- Little to no music theory knowledge
- Passionate about music
- All ages (i.e., both children and adults)
- New to playing instruments
- Not comfortable with complex software
- Eager to learn
- Less willing to spend money

User Category 2: Composers

User Role: As masters of music notation and composition, this group will use the product as a practice aid or to quickly make note of composition ideas.
Subject matter experience*: Master - have spent years learning and practicing music theory and composition.
Technological experience*: Journeyman to Master - Have experience with music-annotating software but haven't necessarily used them extensively.
User characteristics:

- Advanced music theory knowledge
- Passionate about music
- 25+ years old
- Have a hard time learning new technologies
- Willing to spend money

**User experience level may be rated as one of novice, journeyman, or master.*

2.5 Personas

Harold Style is a software engineering student at McMaster University with a deep interest in music, and loves playing the guitar. Despite his lack of a formal music education, he has perfect pitch and thus learns songs by listening to them rather than learn how to read sheet music. However, he recently joined a local rock band called The Beetles who expect him to contribute during their songwriting sessions. He needs a quick way to produce sheet music for his own compositions, as well as learn to read the scores written by his bandmates. As a future software engineer he enjoys learning new technologies that can make his life easier, and he dislikes reading of any sort.

2.6 Priorities Assigned to Users

Key users: Beginner musicians
Secondary users: Composers

2.7 User Participation

Beginner musicians will have a high level of participation in the creation of system requirements. Their feedback will be especially essential during the early to middle development phases to ensure that the system meets their various needs for simplicity, intuitive design/usage, and music theory assistance. Composers will have a moderate level of participation in the creation of system requiremens, slightly less than that of the Beginner Musician group. Their subject matter expertise will be useful in creating relevant requirements to ensure precision of audio transcription and its overall score editing capabilities.

2.8 Maintenance Users and Service Technicians

This product will be maintained by a software development team, consisting of software developers, UI/UX designers, test engineers, and a project manager. Users will receive product support directly from the customer support team, and IP lawyers will be engaged as necessary to address matters of intellectual property.

3 Mandated Constraints

3.1 Solution Constraints

Technology Stack

Description: The app shall be developed using a fast for the core signal processing to handle real-time calculations and a robust, secure, and easy-to-configure language for backend logic.
Rationale: Signal processing needs to be done quickly and effieciently to be done in real time, and the backend must be secure while handling user information.
Fit Criterion: The final product shall process signals quickly and all sensitive user information is secured and nothing gets exposed.

Open-Source Libraries

Description: The app shall use open-source libraries and frameworks where applicable to save both time and money.
Rationale: The use of open-source libraries ensures cost-effectiveness and allows for community-driven improvements and support.
Fit Criterion: All third-party libraries used in the project must be licensed under open-source agreements such as MIT, GPL, or Apache licenses which allow for free use.

3.2 Implementation Environment of the Current System

Hardware Specifications

Description: The app shall run on personal computers with a minimum of 8GB RAM, a dual-core processor, and 256GB of available storage space, which is the minimum according to general concensus [1].
Rationale: These hardware specifications are typical of the devices used by musicians and producers, ensuring the app performs efficiently.
Fit Criterion: The app must pass performance tests on machines meeting the minimum hardware requirements, with no more than a 5% reduction in performance under heavy load.

Audio Input Devices

Description: The app shall support standard audio input devices, including USB microphones, and built-in microphones, with audio input via USB, or 3.5mm audio jacks.
Rationale: These input devices are commonly used by musicians to capture sound, ensuring the app is compatible with existing hardware setups.
Fit Criterion: The app must successfully capture and process audio from these devices, maintaining accurate signal-to-sheet transcription in at least 95% of test cases.

3.3 Partner or Collaborative Applications

Music Notation Software

Description: The app shall collaborate with music notation software that uses MusicXML formats to export sheet music. These can be seen as 'partner' applications since they fit very well with a use case for the application.
Rationale: Many composers rely on professional notation software to finalize and edit their sheet music, so it's essential that the app exports accurate, compatible sheet music files.
Fit Criterion: The app must support MusicXML files, for version support see section 3.4: Off-the-Shelf Software - Backward Compatability.

3.4 Off-the-Shelf Software

MusicXML Format for Sheet Music

Description: The app shall use MusicXML as the primary format for storing and exporting sheet music data.
Rationale: MusicXML is the industry-standard format for sheet music interchange between different music notation software, ensuring compatibility with tools like Sibelius, Finale, MuseScore, and other DAWs. Its widespread adoption makes it the best choice for interoperability.
Fit Criterion: The app must successfully export sheet music in MusicXML format, ensuring compatibility with the latest versions of major music notation software, without requiring manual adjustments by the user.

Backward Compatibility

Description: The app shall support both current and older versions of the MusicXML format to ensure maximum compatibility with various notation tools.
Rationale: As different users may be using different versions of music notation software, supporting older MusicXML formats ensures accessibility and broader usability.
Fit Criterion: The app must successfully import and export MusicXML files using both the latest version of MusicXML and at least one earlier version (e.g., MusicXML 2.0).

3.5 Anticipated Workplace Environment

Indoor and Studio-Based Environments

Description: The app shall be designed primarily for indoor environments, such as home studios, professional recording studios, classrooms, and offices where musicians and composers typically work.
Rationale: Most users will be working in controlled indoor environments where factors like lighting and noise levels will vary. The app must function optimally in these settings without relying on environmental conditions.
Fit Criterion: The app must be tested in various indoor settings (studios, offices, classrooms) with different lighting and noise levels to ensure usability and performance are not impacted by normal environmental variations.

Noise Considerations

Description: The app shall operate effectively in environments with moderate background noise, such as music rehearsal rooms or live performance spaces, without relying on audible notifications.
Rationale: Musicians often work in noisy environments. The app should rely on visual feedback rather than any sound-based notifications and should retain its signal-processing ability (see requirement OE-EP1).
Fit Criterion: The app must be tested in environments with background noise levels approaching a signal to noise ratio (SNR) of 2:1, indicating its effectiveness with moderate background noise.

Portable Workspaces

Description: The app shall be designed to accommodate musicians working in mobile or temporary workspaces, such as cafes or on-the-go setups using laptops.
Rationale: Musicians often work on the move or in shared spaces where setting up large equipment is impractical. The app must be optimized for laptops with limited screen space and varying internet connectivity.
Fit Criterion: The app must be tested for usability on laptops with screen sizes as small as 13 inches.

3.6 Schedule Constraints

Schedule constraints are provided by the capstone course itself, for a detailed schedule refer to the development plan document, section 8.

3.7 Budget Constraints

A maximum budget for this project of $750 has been identified by the capstone course professors. This amount will likely be fine for our project but additional spending will be approved on a case-by-case basis.

3.8 Enterprise Constraints

Since this project is being completed for the McMaster University Capstone course, we must adhere to all constraints provided by the Course Outline Document.

4 Naming Conventions and Terminology

4.1 Glossary of All Terms, Including Acronyms, Used by Stakeholders involved in the Project

Term	Definition
Accidental	A symbol used to raise or lower the pitch of a note from its natural state, such as a sharp, flat, or natural.

Term	Definition
Barline	A vertical line on the staff that divides the music into measures or bars.
Bar	A segment of time defined by a given number of beats, typically indicated by vertical lines on a staff.
Beat	The basic unit of time in music, often associated with a pulse.
Chord	A combination of three or more notes played simultaneously.
Chord Chart	A visual representation of the chords used in a song, often used by guitarists and other instrumentalists.
Clef	A symbol placed at the beginning of the staff that indicates the pitch of the notes on the staff, such as treble clef (G clef) or bass clef (F clef).
Common Time	A musical pattern in which there are four crotchets in a bar.
DAW	Digital Audio Workstation.
Dynamics	How quietly or loudly a piece of music is played.
Final Barline	A double barline indicating the end of a piece of music.
HCD	Human-Centered Design.
Key Signature	In Western musical notation, a key signature is a set of sharp, flat, or rarely, natural symbols placed on the staff at the beginning of a section of music.
Major	A musical scale characterized by a specific pattern of whole and half steps, typically producing a bright or happy sound.
Minor	A musical scale that differs from the major scale by having a lowered third degree, typically producing a darker or sadder sound.
MIDI	Musical Instrument Digital Interface.
Monophonic	Using only one line of music, unaccompanied by any other voices or instruments.
MP3	MPEG-1 Audio Layer 3.
MusicXML	Standard open format for exchanging digital sheet music.
MVP	Minimum Viable Product.
Notation	The system of writing music to represent pitches, rhythms, and dynamics.
Note	A symbol representing a musical sound, characterized by its pitch and duration.
Rest	A symbol indicating a period of silence in a piece of music.
PII	Personal Identifiable Information.

Term	Definition
Polyphonic	The simultaneous combination of two or more tones or melodic lines.
Rhythm	The pattern of sounds and silences in music.
SNR	Signal to Noise Ratio.
Staff	A set of five horizontal lines and four spaces that represent different pitches for musical notation.
Time Signature	Definition 3.
UI	User Interface.
WAV	Waveform Audio File Format.

5 Relevant Facts And Assumptions

5.1 Relevant Facts

- This project is under irregular time constraints, given that it is for an undergraduate-level software engineering course.
- Digitizing and transcribing polyphonic audio from multiple instruments is a difficult task and may be unachievable due to the aforementioned time constraints.
- Many products are currently available as alternatives to what this project hopes to accomplish. The team will use these products and their functionalities as metrics for success.

5.2 Business Rules

1. Instrument Support
    - The application only supports a predefined list of instruments.
    - No instruments may be added by the user to the predefined list.
2. Music Theory Compliance
    - The application must generate score sheets that adhere to basic music theory rules (e.g., correct note placement, no overlapping notes, etc.).

- The application must generate a score sheet within a reasonable amount of time to ensure a smooth user experience.

3. Ownership

    - The application must clearly identify and display the creator/owner of the score.

5.3 Assumptions

Assumptions are seperated into two categories based on the effect their falsehood will have on the application:

1. Should the following assumptions be false, the application will be unusable:

    - Users of the application have access to the hardware required, such as a music interface or quality microphone.
    - Users have access to a desktop or portable computer to use the application.

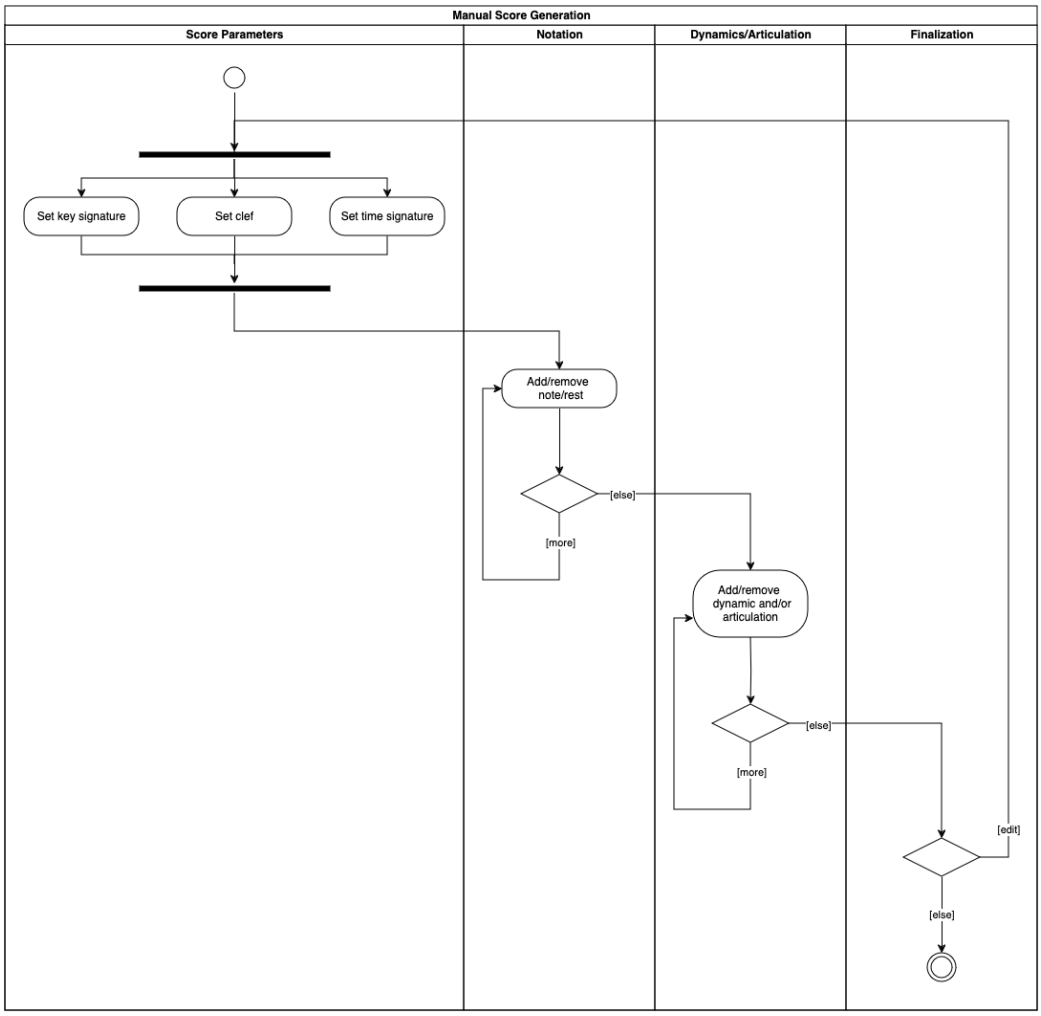
2. Should the following assumptions be false, the application will perform at sub-optimally:

    - The operational environment has low levels of background noise.
    - The human voice is not considered an instrument that the application can handle input from.
    - Users have a superficial level of understanding of music theory in order to understand what the application does.

6 The Scope of the Work

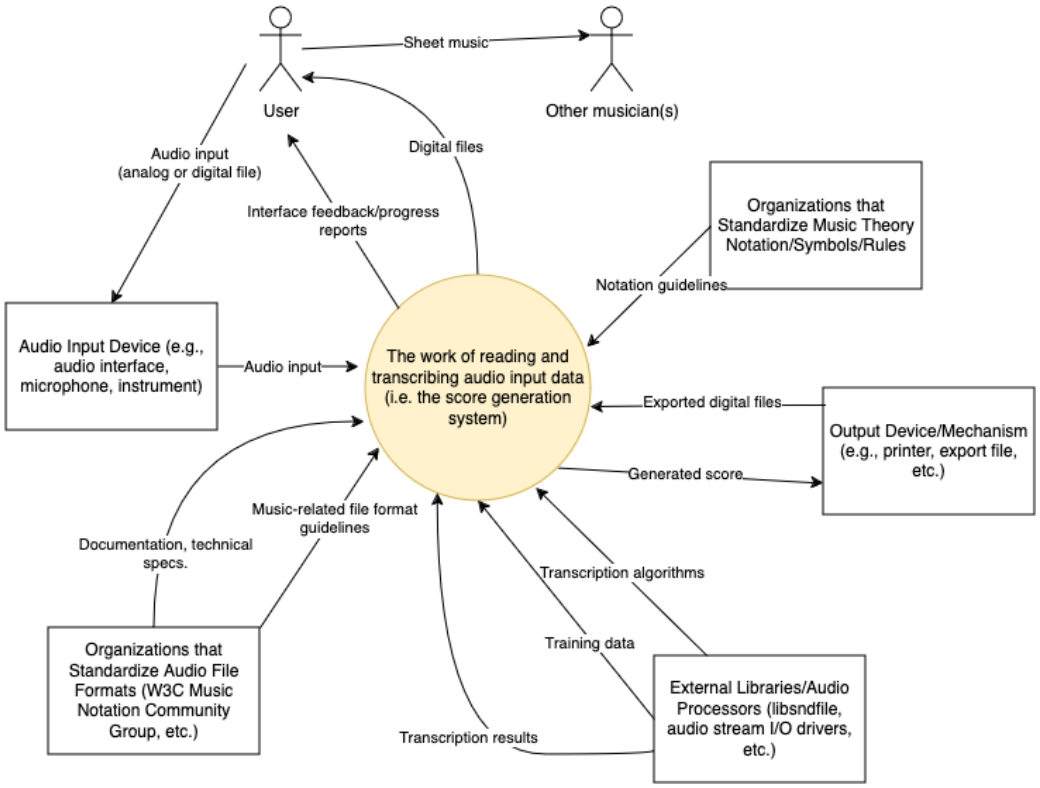
6.1 The Current Situation

The following activity diagram illustrates the workflow of manual score sheet generation, the process which the application intends to improve upon.



6.2 The Context of the Work

The diagram below shows the context in which the development of the application is taking place. It shows adjacent systems and their associated inputs and outputs as it relates to the application and it's work.



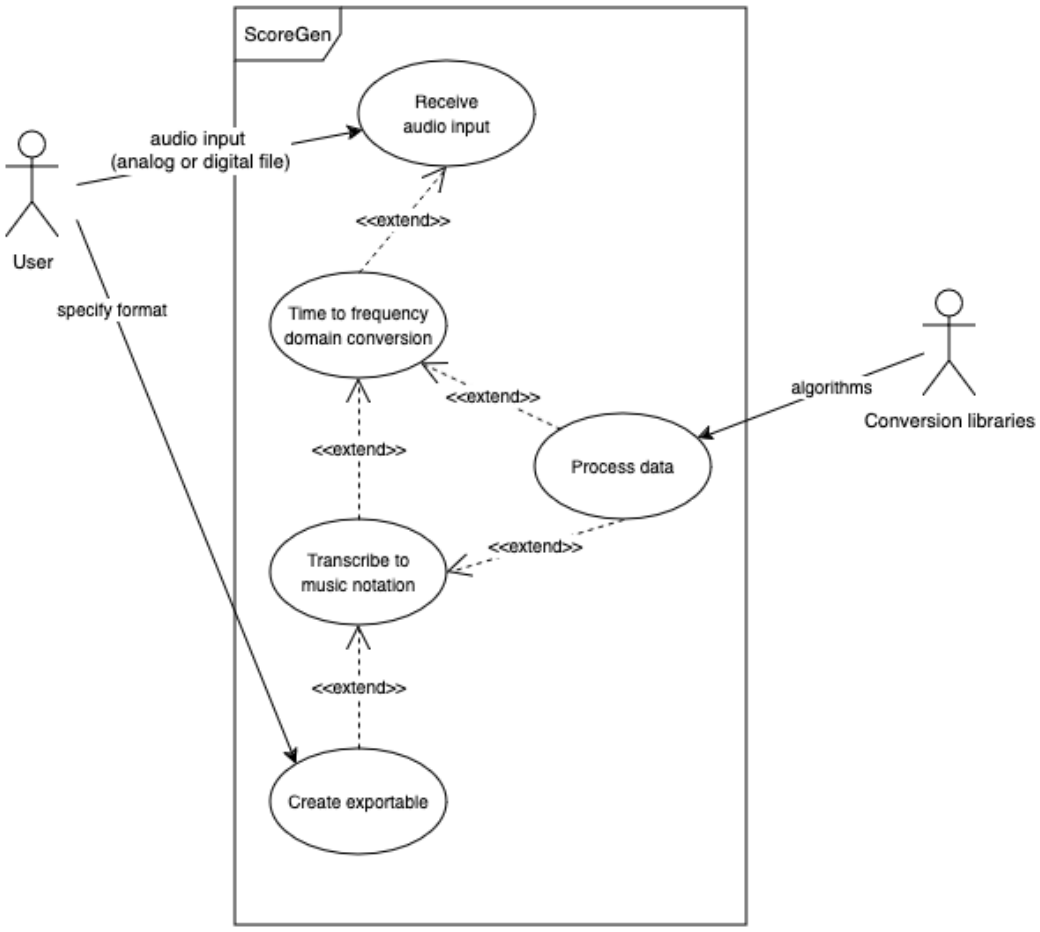
6.3 Work Partitioning

The table below summarizes the business events that might be triggered by adjacent organizations or application users themselves.

EVENT NAME	INPUT/OUTPUT	SUMMARY OF BUC
BE1. User provides audio input.	Audio input (IN)	Process the input audio, digitize if necessary.
BE2. Audio transcription.	Transcription results (IN)	Record the notes transcribed.
BE3. Score completion.	Generated score (OUT)	Finalize the transcription results into proper music notation/symbols.
BE4. Export files.	Exported digital files (IN)	Convert the generated score into a digital file format for export (see BE6.).
BE5. Progress reports.	Interface feedback/progress reports (OUT)	Provide visual indicators to user of transcription status.
BE6. Share sheet music.	Digital files (OUT)	Output the exportable file format (see BE4.).

6.4 Specifying a Business Use Case (BUC)

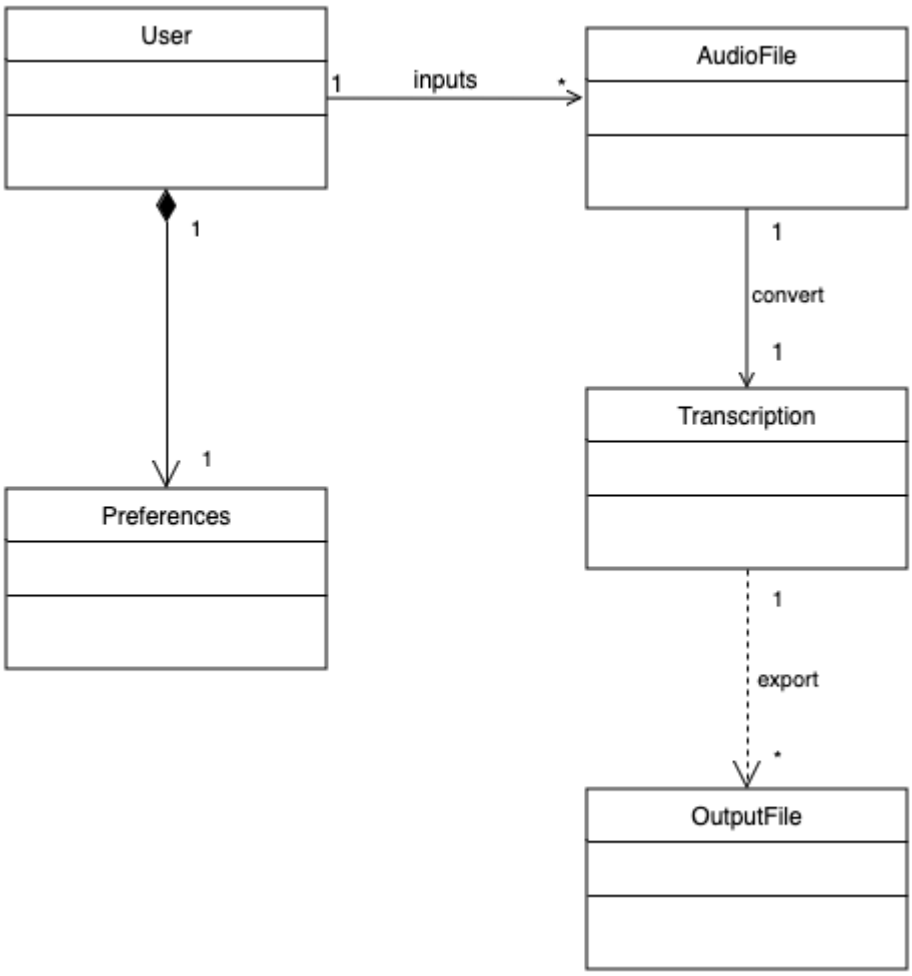
The use case diagram below has been used to specify the details of the BUC associated with BE1.



7 Business Data Model and Data Dictionary

7.1 Business Data Model

A first-cut UML class diagram has been used to model the data entities relevant to the scope of the work. This abstractly describes the relationship between these entities. For definitions and attributes see section 7.2 Data Dictionary.



7.2 Data Dictionary

NAME	CONTENT	TYPE
User	userID + name + preferences	Class
Preferences	prefID + userID + instrument + preferredInput	Class
AudioFile	audioID + inFileName + format + userID + uploadDate	Class

Transcription	transcID + audioID + outputFormat + creationDate + outFileName	Class
OutputFile	outputID + outFileName + fileType	Class
*ID	Unique identifier for distinguishing between entities	Attribute/Element
name	Name of the user, for ownership purposes	Attribute/Element
preferences	Data related to user interface preferences	Attribute/Element
instrument	The musical instrument the user has selected to use for input	Attribute/Element
preferredInput	Type of input the user will use (e.g., digital files, analog input)	Attribute/Element
inFileName	Name given to the file that stores input data	Attribute/Element
format	The format of the input file (e.g., WAV, MP3, etc.)	Attribute/Element
uploadDate	*DD/MM/YYYY* For file management and ownership purposes	Attribute/Element
outputFormat	The format that the exportable output file will use	Attribute/Element
creationDate	*DD/MM/YYYY* For ownership purposes	Attribute/Element
outFileName	Name given to the file that stores output data	Attribute/Element
fileType	The format that the exportable output file will use	Attribute/Element

8 The Scope of the Product

8.1 Product Boundary

Input: ScoreGen will be designed to take in and process music data which can be converted to a MIDI or musicXML file representation. The audio should not have excessive noise, or other degradation of quality which could impact the ability to clearly distinguish notes. The input will also be intended to be mono or polyphonic with a singular instrument. The input volume should remain consistent and sufficiently loud, and the user microphone should be correctly calibrated to ensure accuracy by avoiding pitch drift.
Output: The resulting music file will aim to capture the pitch and length of individual notes or chords, but will not capture accents, dynamics, or tempo markings, as well as other advanced elements. If the system detects that a note cannot be reliably transcribed (due to unclear input), it will leave that note blank on the sheet and prompt the user to review the section manually.

8.2 Product Use Case

UC1: Record and Transcribe

Description: Musician plays an instrument, and the system transcribes it into sheet music in real-time.
Actors: Musician, Desktop App
Preconditions: Musician has a working microphone and is playing a supported instrument.
Basic Flow:

1. Musician plays the instrument.
2. The system listens through the mic.
3. Sheet music is generated.

Alternate Flow:

- If the note is unclear, the system skips it and warns the user.

Postconditions: Sheet music is generated and displayed to the user.

UC2: Edit Transcription

Description: Musician reviews and edits the generated sheet music if there are missing or incorrect notes.
Actors: Musician, Desktop App
Preconditions: Sheet music has already been generated from a recording.
Basic Flow:

1. Musician views the transcription.
2. Edits or corrects notes as needed.
3. Saves the updated sheet.

Alternate Flow:

- Musician manually inserts missed or unclear notes.
- Musician deletes incorrect notes.

Postconditions: Musician has an edited, correct version of the sheet music.

UC3: Save and Export

Description: Save the sheet music in a chosen format (e.g., PDF, MIDI, musicXML).
Actors: Musician, Desktop App
Preconditions: Sheet music is completed and reviewed.
Basic Flow:

1. Musician chooses the save option.
2. Selects a file format (PDF, MIDI, musicXML).
3. System exports the file.

Alternate Flow:

- If export fails, the system provides an error message and suggests retrying.

Postconditions: The sheet music file is saved in the selected format.

UC4: Instrument Setup

Description: Select instrument type before recording (e.g., piano, guitar).
Actors: Musician, Desktop App
Preconditions: Desktop app is open, and the user is ready to begin recording.
Basic Flow:

1. Musician selects the instrument.
2. System optimizes transcription settings for the chosen instrument.

Alternate Flow:

- If no instrument is selected, default settings for piano are used.

Postconditions: System is ready to accurately transcribe the chosen instrument.

UC5: Error Warning

Description: System warns if it cannot hear or recognize a note during the recording process.
Actors: Musician, Desktop App
Preconditions: Musician is actively recording.
Basic Flow:

1. System detects a missing or unclear note.
2. A warning is displayed to the user.

Alternate Flow:

- The system may suggest trying again or inserting the note manually post-recording.

Postconditions: Musician is informed about missing notes and is directed on how to take corrective action, which may involve editing within the app or using a third-party tool depending on available features.

UC6: Playback of Transcription

Description: Musician can play back the recorded notes in the desktop app to review transcription accuracy.
Actors: Musician, Desktop App
Preconditions: Transcription has been generated from a recording.
Basic Flow:

1. Musician selects the playback option.
2. System plays the transcribed notes.

Alternate Flow:

- Musician can pause or stop playback to make corrections.

Postconditions: Musician reviews the transcription by listening to the played-back notes.

8.3 Individual Product Use Cases (PUC's)

PUC1: Record Instrument and Transcribe Notes

Description:
The user records a live performance on an instrument using their microphone, and the system transcribes the notes into sheet music.

Primary Actor:
Musician

Trigger:
The user presses the "Record" button.

Preconditions:
- User has selected an instrument (e.g., piano, guitar).
- User has a working microphone connected to the desktop app.

Main Success Scenario:
1. User begins playing the instrument.
2. System listens to the instrument through the microphone.
3. Sheet music is displayed on the screen after the user finishes playing.

Exceptions:
- If the system cannot detect a note, it skips the note and warns the user with a notification.

PUC2: Save and Export Sheet Music

Description:
The user saves the transcribed sheet music and exports it in the desired format (PDF, MIDI, musicXML).

Primary Actor:
Musician

Trigger:
The user clicks the "Save" or "Export" button.

Preconditions:
- The transcription is completed, and the user has reviewed it.

Main Success Scenario:
1. The user selects a file format (PDF, MID, musicXMLI).
2. The system generates a file in the selected format.
3. The file is saved to the user's chosen directory.

Exceptions:
- If the export fails due to file system issues, the user is prompted to retry or select another directory.

PUC3: Receive Error Warnings for Missing Notes

Description:
The user is notified if the system cannot detect or accurately transcribe certain notes during recording.

Primary Actor:
Musician

Trigger:
The system fails to detect a note.

Preconditions:
- User is actively recording their instrument.

Main Success Scenario:
1. The system detects a missing note during recording.
2. A warning message is displayed to the user regarding the missing note.

Exceptions:
- The system suggests the user retry playing the note or to manually insert it later, either within the app or using a third-party tool depending on available features.

9 Functional Requirements

9.1 Audio Input Handling

FR-AI1 Audio Formats

Requirement: The system shall accept audio input from users in various formats (e.g., WAV, MP3, AAC).
Fit Criterion: The system confirms a succesful upload.

FR-AI2 Audio Recording

Requirement: The system shall provide a user interface for recording audio directly through a microphone connected to the user's device.
Fit Criterion: The system displays a confirmation of reception during real-time recording.

FR-AI3 Audio Uploads

Requirement: The system shall allow users to upload pre-recorded audio files.
Fit Criterion: The system confirms reception of the pre-recorded audio file.

FR-AI4 User Input

Requirement: The user shall give their instrument (i.e., piano or guitar) to the system as input.
Fit Criterion: The interface and concert pitch of the generated sheet music reflect the user's choice.

9.2 Signal Processing

FR-SP1 Noise Reduction

Requirement: The system shall perform noise reduction on the audio input to enhance quality.
Fit Criterion: The system generates identical sheet music for input with no background noise and input with a 10% increase in background noise.

FR-SP2 Note Identification

Requirement: The system shall analyze the audio to identify pitch and rhythm.
Fit Criterion: The generated sheet music visually matches expected notes from a reference audio source and corresponding sheet music.

FR-SP3 Key and Time Signature Identification

Requirement: The system shall analyze the audio to identify the key and time signatures.
Fit Criterion: The system correctly identifies key and time signatures or their equivalencies (e.g. a minor to C major, common time to cut time) from a reference audio source and corresponding sheet music.

FR-SP4 Processing Limitations

Requirement: The system gracefully handles overly complex audio input (i.e., it is unable to perform accurate signal processing).
Fit Criterion: The system notifies the user that their input cannot be processed.

FR-SP5 Polyphonic Audio

Requirement: The system shall handle polyphonic audio, allowing for multiple notes to be detected simultaneously.
Fit Criterion: The generated sheet music contains chords/simultaneous melody.

FR-SP6 Monophonic Audio

Requirement: The system shall handle monophonic audio.
Fit Criterion: The generated sheet music contains single pitch notes.

9.3 Sheet Music Generation

FR-SMG1 Notation

Requirement: The system shall generate sheet music in standard music notation based on the parsed audio data.
Fit Criterion: The generated sheet music is visually comparable to a reference piece of sheet music.

FR-SMG2 Concert Pitch

Requirement: The system shall generate sheet music in the concert pitch of the instrument played in the audio input.
Fit Criterion: The generated sheet music visually matches expected notes from a reference audio source and corresponding sheet music.

FR-SMG3 Post-Processing Edits

Requirement: The system shall allow users to view and edit the generated sheet music within the application.
Fit Criterion: Changes made by the user are reflected in the output files produced during sheet music generation.

9.4 User Interface

FR-UI1 System Feedback

Requirement: The interface shall display processing feedback during audio input.
Fit Criterion: Feedback from system is displayed to the user within 2 seconds of their input.

FR-UI2 System Documentation

Requirement: The system shall provide help documentation and user guides for each feature.
Fit Criterion: At least 90% of users during user testing find the interface easy to navigate.

FR-UI3 User Feedback

Requirement: The system shall provide a feedback mechanism for users to report issues or suggest features.
Fit Criterion: The development team successfully receives user feedback via the feedback mechanism 100% of the time.

FR-UFI4 Customer Support

Requirement: The system shall include customer support contact information for troubleshooting.
Fit Criterion: At least 90% of users during user testing report successfull interactions with customer support.

9.5 Save and Load

FR-SL1 Save

Requirement: The system shall allow users to save their work, including both audio files and generated sheet music.
Fit Criterion: 100% of the time when the system reports a succesfull save, the user is succesfully able to locate the audio files and generated sheet music.

FR-SL2 Load

Requirement: The system shall support loading previously saved audio files and generated sheet music for further editing.
Fit Criterion: The system sucesfully retains user progress such that the user may return to their unfinished work seamlessly.

10 Look and Feel Requirements

10.1 Appearance Requirements

LF-A1 Discoverability

Requirement: Main functionalities of the application interface must have elements that are easily identifiable.
Fit Criterion: Upon viewing the product for the first time, 75% of potential users shall be able to locate the interactive element that initiates the score generation process.

LF-A2 Colour Cohesion

Requirement: The application interface must use a consistent color scheme for uniformity.
Fit Criterion: The product shall have color guidelines that specify at least three colors to define a palette (i.e. primary, secondary, and accent colors).

LF-A3 Resolution

Requirement: The application must use high definition graphics.
Fit Criterion: The application will only display images that are of a resolution of 720p or higher.

10.2 Style Requirements

LF-S1 Authority and Trust

Requirement: The application must appear authoritative and trustworthy, users are more likely to engage and remain loyal to an application that is percieved as reliable.
Fit Criterion: After one encounter with the application, 70% of users will agree that they can trust the application and how it handles input data.

11 Usability and Humanity Requirements

11.1 Ease of Use Requirements

UH-EOU1 Human Centered Design (HCD)

Requirement: The app interface must adhere to human-centered design principles to be simple and intuitive to use.
Fit Criterion: The user interface must incorporate all four fundamental HCD principles.

11.2 Personalization and Internationalization Requirements

UH-PI1 Language

Requirement: The application must provide support for Canadian English (en-CA).
Fit Criterion: All text, labels, and instructions within the application shall be presented in Canadian English, with no grammatical or spelling errors.

11.3 Learning Requirements

UH-L1 Music Theory Familiarity

Requirement: The application must be usable by users without a formal music theory background.
Fit Criterion: After having 10 minutes to familiarize themselves with the app, 95% of users without a formal background will be able to generate a score sheet.

11.4 Understandability and Politeness Requirements

UH-UP1 Icon Identification

Requirement: The app must use symbols and icons that are naturally understandable and/or self-explanatory.
Fit Criterion: Users shall be able to easily identify the function of at least 70% of interactive elements that have unlabeled icons.

UH-UP2 Information Hiding

Requirement: The app must hide audio processing and transcription details from the user.
Fit Criterion: No user-accessible components of the interface or application shall reveal implementation-specific or algorithmic solutions used.

11.5 Accessibility Requirements

UH-A1 Web Content Accessibility Guidelines

Requirement: The app must comply with the standardized Web Content Accessibility Guidelines (WCAG) [2].
Fit Criterion: The app must meet or exceed WCAG 2.1 Level AA compliance. This shall be verified through an accessibility audit using automated tools (e.g., WAVE, Axe) and manual reviews, ensuring adherence to guidelines such as text alternatives, text readability, etc.

12 Performance Requirements

12.1 Speed and Latency Requirements

PR-SL1 User Interface Response Time

Requirement: The interface between the user and the sheet music generator app shall respond within a reasonable time when processing input or selecting menu options.
Fit Criterion: The product shall respond in less than 2 seconds for 90% of user interactions. No response shall take longer than 5 seconds.

PR-SL2 File Import and Export Speed

Requirement: The app shall import and export music files within a reasonable time for files up to 100MB in size.
Fit Criterion: For files up to 100MB, the app shall import and export them within a reasonable time frame for their system for more than 95% of operations.

12.2 Safety-Critical Requirements

PR-SC1 Epilepsy Safety

Requirement: The app shall not display rapid flashing lights, animations, or any visual effects that could trigger epileptic seizures in users.
Fit Criterion: The app's graphical interface must comply with WCAG 2.1 (Web Content Accessibility Guidelines) [3] for flashing content, limiting visual effects to fewer than 3 flashes per second. The interface will be tested with accessibility tools to ensure compliance with these standards.

PR-SC2 Instrument Input Setup

Requirement: The app shall guide users through the correct setup of instrument input to prevent audio misconfiguration, damage to the user/instrument, and damage to the application's device.
Fit Criterion: The app will include an interactive, step-by-step tutorial for configuring instrument inputs (via external devices or microphones). The setup process should be completed with a 95% success rate based on user testing, ensuring correct audio input detection on the first attempt.

12.3 Precision or Accuracy Requirements

PR-PA1 Pitch Detection Accuracy

Requirement: This app shall correctly detect musical pitch from instruments' input to ensure that all notes are transcribed correctly.
Fit Criterion: The app must maintain this pitch accuracy over all test cases involving diverse instruments and note ranges, with no more than 1% error. Also must satisfy requirement OE-EP1 for noise tolerance.

PR-PA2 Timing Accuracy

Requirement: The app shall accurately capture note durations and rhythms within a reasonable tolerance to ensure correct rhythmic transcription.
Fit Criterion: The app must maintain timing accuracy within the 100ms across all test cases.

12.4 Robustness or Fault-Tolerance Requirements

PR-RFT1 Reliability (Time Between Failures)

Requirement: The app shall operate continuously for at least 24 hours between failures or system crashes, ensuring that users experience minimal interruptions during usage.
Fit Criterion: The app must pass reliability testing by operating under typical usage conditions for 24 hours without a failure in at least 95% of test cases.

PR-RFT2 Availability (Uptime)

Requirement: The app shall achieve 99.5% uptime, ensuring that it is available for use at nearly all times, excluding brief periods for necessary updates or maintenance.
Fit Criterion: The app must demonstrate an average of 99.5% uptime over a reasonable time.

PR-RFT3 Crash Recovery

Requirement: In an unexpected crash, the app shall automatically save the user's current session, including any partially transcribed sheet music, and allow the user to recover their work upon restart.
Fit Criterion: During testing, the app must successfully recover user data in 98% of cases where a crash occurs, with no data loss.

PR-RFT4 Performance Under Load

Requirement: The app shall maintain reliable performance under conditions of high user activity (e.g., processing complex polyphonic inputs or large files) without significant slowdowns or crashes.
Fit Criterion: The app must maintain stable performance, with no more than a 30% decrease in processing speed, when tested under maximum expected load conditions.

PR-RFT5 Handling Signal Interruptions

Requirement: The app shall continue to operate and retain the audio signals received up to the point of interruption, even if the instrument input is

temporarily lost (e.g., due to cable disconnection or microphone failure).
Fit Criterion: The app must store buffered audio data for up to 2 minutes during signal interruptions and automatically resume transcription once the input is restored, with 0% data loss in at least 95% of test cases.

PR-RFT6 Graceful Degradation

Requirement: In the event of a performance slowdown (e.g., due to excessive input data or limited system resources), the app shall degrade its services gracefully by notifying the user of delays without crashing.
Fit Criterion: During stress testing under heavy load (e.g., multiple instruments or large file sizes), the app must maintain operational status and notify the user of delays or lowered performance within 5 seconds of detection, with no system crashes in 99% of test cases.

PR-RFT7 Automatic Recovery from Software Glitches

Requirement: If the app encounters a minor software error (e.g., unexpected input format or corrupted file), it shall log the error, notify the user, and continue functioning by skipping the problematic section rather than halting it.
Fit Criterion: The app must pass testing by handling minor software errors in at least 90% of cases without crashing and will log the issue for future debugging.

12.5 Capacity Requirements

PR-C1 Audio Input Capacity

Requirement: The app shall process audio input of up to 90 minutes of continuous recording in a single session without performance degradation.
Fit Criterion: The app must be tested with audio recordings of varying lengths (up to 90 minutes) and successfully process and transcribe the entire duration with no more than a 5% slowdown in transcription speed or accuracy.

PR-C2 Simultaneous User Sessions

Requirement: The app shall support at least 10 simultaneous user sessions (local or cloud-based) without degradation in performance or delays in processing.
Fit Criterion: During testing with 10 simultaneous active user sessions, the app must maintain performance levels, including stable response times and transcription accuracy, with no noticeable slowdown in 95% of test cases.

12.6 Scalability or Extensibility Requirements

PR-SE1 User Base Growth

Requirement: The app shall be capable of scaling to support an increase in the user base to 1000 users without requiring major architectural changes to support a change from personal to commercial use.
Fit Criterion: The app's infrastructure must be designed to scale up by a factor of 10 and should undergo stress testing to ensure it can handle 1000 simultaneous user sessions with no significant performance degradation.

PR-SE2 Feature Extensibility

Requirement: The app shall be designed with a modular architecture that allows for future feature additions, such as multi-instrument detection or cloud-based collaboration tools, without requiring significant refactoring.
Fit Criterion: The app must pass code reviews and architecture validation to ensure it can integrate additional modules or features within 6 months of the product's initial release, with minimal impact on core functionalities.

PR-SE3 Processing Power for Complex Music Compositions

Requirement: The app shall be built to be able to scale its processing capabilities to handle future complex compositions involving multiple simultaneous instrument tracks supporting growth for polyphony.
Fit Criterion: During testing, the app must demonstrate the ability to process increasingly complex musical arrangements.

12.7 Longevity Requirements

PR-L1 Expected Lifetime

Requirement: The app shall be designed to operate effectively with regular software updates and minor maintenance for a minimum of five years without requiring a major rewrite or overhaul.
Fit Criterion: The app's development roadmap must include planned updates and feature expansions to maintain functionality and relevance for at least five years after launch.

13 Operational and Environmental Requirements

13.1 Expected Physical Environment

OE-EP1 Operating Environment

Requirement: The app shall be used on personal computers or laptops in typical indoor environments, such as home studios, classrooms, or offices. It should perform well under varying levels of ambient noise.
Fit Criterion: The app must be tested in multiple indoor settings and perform well without requiring any environment-specific adjustments.

OE-EP2 Noise and Audio Input Quality

Requirement: The app shall be able to capture clear audio input in environments with moderate ambient noise, such as home studios or cafes, without significant degradation in transcription accuracy.
Fit Criterion: The app must be tested in environments with background noise to ensure that it can still accurately capture and process audio inputs without more than a 5% drop in accuracy.

OE-EP3 Workspace Flexibility

Requirement: The app shall be usable on laptops with limited screen space, such as 13-inch displays, and must adjust its layout to accommodate various screen resolutions and sizes.
Fit Criterion: The app must be tested on screens ranging from 13 inches

to 27 inches, with functionality and UI usability maintained across all screen sizes and resolutions.

OE-EP4 Portable Setup Compatibility

Requirement: The app shall be compatible with portable setups, such as laptops used in temporary workspaces like cafes or live performance venues, ensuring that it can be used effectively in different environments without special hardware.
Fit Criterion: The app must function smoothly on portable setups with wireless internet and standard laptop hardware, and be tested for usability in transient environments where quick setup and breakdown are required.

13.2 Wider Environment Requirements

OE-WE1 General Hardware

Requirement: The app shall operate effectively on devices with varying interface specifications.
Fit Criterion: The app must be tested on screens ranging from 13 inches to 27 inches, with functionality and UI usability maintained across all typical screen sizes and resolutions.

13.3 Requirements for Interfacing with Adjacent Systems

OE-IA1 Audio Input Devices

Requirement: The app shall interface with standard audio input devices, including USB microphones, instrument pickups, and built-in computer microphones.
Fit Criterion: The app must support audio input via devices adhering to USB Audio Class 1.0 and higher, and be tested with common setups (e.g., USB microphones, line-in jacks).

OE-IA2 Audio File Import and Export

Requirement: The app shall interface with common audio file formats for both import and export, including WAV and MP3.
Fit Criterion: The app must successfully import and export audio files in WAV (PCM), and MP3 (MPEG-1 Layer III).

OE-IA3 Music Notation Software Integration

Requirement: The app shall export sheet music in formats compatible with popular music notation software, including MusicXML and MIDI files.
Fit Criterion: The app must generate MusicXML and MIDI files compatible with the software.

13.4 Productization Requirements

OE-P1 Distribution

Requirement: The app shall be distributed as a downloadable executable file.
Fit Criterion: The app must be packaged in an installer that can be downloaded from a website or software repository, with no additional dependencies required for installation.

OE-P2 Installation Process

Requirement: The product shall be capable of being installed by an untrained user with minimal technical knowledge, requiring minimal steps from download to full installation.
Fit Criterion: The installation wizard must guide the user through the process, providing automatic configuration options and requiring minimal user input. Instructions must be included within the installer interface.

OE-P3 Size and Compatibility

Requirement: The app's installation file size shall not exceed 500MB, ensuring that the product can be easily downloaded even with moderate inter-

net connections.
Fit Criterion: The app must be easily installed on a variety of systems.

OE-P4 Post-Installation Configuration

Requirement: The product shall configure its default settings upon installation, but allow users to modify these settings (e.g., input source, output format) during or after the installation process.
Fit Criterion: A settings configuration panel must be accessible from the app's user interface, allowing users to adjust settings without needing to edit configuration files manually.

13.5 Release Requirements

OE-R1 Initial Release

Requirement: The first stable version of the app shall be released within the timeline of this capstone course after completing the testing and POC phase. This release will include core functionality such as converting instrument signals to sheet music and basic editing capabilities.
Fit Criterion: The initial release must undergo testing and pass all quality assurance checks. Core features must be fully operational, and the product should be ready for general use without major bugs.

14 Maintainability and Support Requirements

14.1 Maintenance Requirements

MS-M1 Version Releases

Requirement: The system must notify the user of new releases of the system within 1 hour of the release.
Fit Criterion: Following a new version release of the system, the next time the user opens the application with an internet connection they will be prompted to install the new release.

MS-M2 System Crash

Requirement: The application must self-boot within 1 minute of an unexpected shutdown of the application.
Fit Criterion: A non-user prompted shutdown of the system shall result in the application reopening without losing data from the user's previous session.

14.2 Supportability Requirements

MS-S1 Software Bugs

Requirement: The system must allow users to report bugs to the development team.
Fit Criterion: The development team shall be notified within 1 hour of a user submitting a report via the GUI.

MS-S2 Operating System

Requirement: The system must operate on Windows 10 and 11.
Fit Criterion: The system shall be compatible with Windows 10 and later version such that it performs all expected functionalities

14.3 Adaptability Requirements

MS-A1 Internet Connection

Requirement: The system must function with or without an internet connection.
Fit Criterion: User experience of the system will be consistent during usage on an online device as well as an offline device.

MS-A2 GUI Navigation

Requirement: The GUI must be navigable using keyboard shortcuts without requiring a mouse.
Fit Criterion: The system shall remain functional and usable when a mouse is not connected to the device running the system.

15 Security Requirements

15.1 Access Requirements

S-A1 User Authentication

Requirement: The system must not require a login or password to be accessed.
Fit Criterion: The user successfully accesses the system without performing any authentication activities.

15.2 Integrity Requirements

N/A: This system does not deal with storage of PII, session logs, usage logs, or output files (i.e., generated sheet music). It is the user's responsibility to maintain long-term storage of output files.

15.3 Privacy Requirements

S-P1 Data Storage

Requirement: The system must store user data (i.e., output files) in a local user-chosen location.
Fit Criterion: All output files appear in the user-inputted location such that the user has read and write capabilities over the files.

S-P2 PII

Requirement: The system must not collect Personal Identifiable Information (PII) from the user.
Fit Criterion: The user is never prompted to supply information to the system other than audio input files.

S-P3 Input Data

Requirement: The system must not store audio data once it has been processed.
Fit Criterion: All caches and temporary files are cleared from the system once signal processing is completed.

15.4 Audit Requirements

N/A: This system is for personal and offline use, and as such the developers do not hold responsibility over the usage of the system by future customers.

15.5 Immunity Requirements

N/A: This system does not store data and thus will not be susceptible to maleveolent actors.

16 Cultural Requirements

16.1 Cultural Requirements

CR-CR1 Musical Convention

Requirement: The application will focus on Western music notation, commonly used in the United States and other English-speaking regions. **Fit Criterion:** The application must accurately represent Western music notation standards, allowing users to create and view sheet music that adheres to these conventions.

CR-CR2 Expected music theory level of users

Requirement: It is designed for users with a basic to intermediate understanding of Western music theory, who can read standard sheet music, though advanced expertise is not required. **Fit Criterion:** The user interface and functionality of the application must be intuitive for individuals with basic to intermediate music theory knowledge, providing tutorials or guides for those who may need additional assistance.

17 Compliance Requirements

17.1 Legal Requirements

CR-LR1 Copyright issues

Requirement: The app's legality is primarily concerned with copyright issues surrounding the transcription of copyrighted music. It will generate

sheet music based on audio input, and the focus will be on transcription of musical elements, which is generally allowed under copyright law, as it involves interpretation rather than direct reproduction of the original work. Users will be advised to ensure their audio inputs do not violate copyright restrictions. **Fit Criterion:** The application must include a disclaimer regarding copyright laws and provide users with clear instructions on ensuring their audio inputs comply with copyright restrictions.

17.2 Standards Compliance Requirements

CR-SCR1 Technological Standards

Requirement: The application will follow established music technology standards, including MusicXML for exporting sheet music. Any third-party libraries used will be reviewed to ensure compliance with their licensing terms. **Fit Criterion:** The application must successfully export sheet music in MusicXML format without errors, and all third-party libraries must be documented for compliance verification.

18 Open Issues

Handling Complex Rhythms and Non-Monophonic Instruments: Scoregen may struggle to accurately transcribe complex rhythms (e.g., syncopation, irregular time signatures) or multiple instruments played simultaneously.
Real-Time Processing Latency: Potential issue if some near real-time note generation and display system is implemented. There could be a delay between when the instrument is played and when the notes appear on the screen, disrupting the real-time transcription experience.
Inconsistent Detection of Chords: The system may have difficulty accurately detecting and transcribing complex chords or overlapping harmonic frequencies, leading to errors.
Difficulty with Non-Standard Tuning or Instruments: Musicians using non-standard tuning or less common instruments may experience inaccurate note detection or transcription errors.
Microphone and Equipment Dependency: The quality and type of microphone used can significantly affect transcription accuracy, leading to more errors for users with lower-quality equipment.

Legal and Licensing Concerns: There may be intellectual property issues if users transcribe and share sheet music for copyrighted music without permission.

19 Off-the-Shelf Solutions

19.1 Ready-Made Products

Content: In exploring potential off-the-shelf (OTS) solutions that could fulfill part or all of our project requirements, we have reviewed several existing software products related to music transcription and signal processing. While our primary focus is on learning and building the product from scratch as a capstone project, these products were considered for their applicability and potential use in enhancing our understanding of the field.
Motivation: The goal of this investigation was to consider whether any existing solutions could meet our project's goals or provide insight into how to address specific technical challenges. Given that our focus is on educational value rather than commercial competition, we explored these solutions to learn from their features and limitations.

Products Investigated:

- **Sibelius** [4]
  Description: Sibelius is a professional music notation software that supports transcription, editing, and playback of sheet music.
  Applicability: While Sibelius provides a robust notation environment, it does not directly fulfill the core requirement of real-time audio-to-sheet-music transcription. However, its export capabilities (MusicXML and MIDI) align with our needs for cross-compatibility.
  Conclusion: Sibelius is not suitable as a full solution for our project but offers useful insights into notation and export formats.

- **AnthemScore** [5]
  Description: AnthemScore is an automated music transcription software that converts audio recordings into sheet music.
  Applicability: AnthemScore is the most closely aligned with our project's goal of audio-to-sheet-music transcription. It uses machine

learning to process audio, which could provide valuable lessons on handling real-time audio input and accuracy.
Conclusion: Although AnthemScore is a ready-made solution for audio transcription, integrating it into our project would limit our learning opportunities. However, understanding its machine learning approach offers valuable technical insights.

- **MuseScore** [6]
  Description: MuseScore is an open-source music notation software that supports MusicXML and MIDI input/output.
  Applicability: MuseScore offers advanced notation tools and is widely used for music transcription. While it does not handle real-time audio transcription, it could serve as a valuable resource for understanding notation standards and file format handling (MusicXML, MIDI).
  Conclusion: MuseScore provides a strong learning resource for notation and file management, but it is not directly applicable for our audio signal processing goals.

- **Transcribe!** [7]
  Description: Transcribe! is a software focused on assisting musicians in transcribing audio files into written music manually.
  Applicability: Although Transcribe! focuses more on aiding manual transcription, its use of time-stretching and pitch-shifting techniques for audio processing could offer valuable learning insights for real-time transcription in our project.
  Conclusion: Transcribe! is not a direct solution but may offer useful ideas for how to manage and process audio signals for manual transcription.

- **Melody Scanner by Klangio** [8]
  Description: Melody Scanner is an online tool that converts audio files, such as recordings of melodies, into sheet music automatically which is the general idea of this project. It focuses on providing a quick and user-friendly way to transcribe melodies for musicians.
  Applicability: Melody Scanner aligns closely with our goal of audio-to-sheet-music transcription. However, its focus is primarily on melodic transcription rather than handling complex polyphonic music or real-time input. Despite this, it offers a straightforward user interface and backend processes that could provide useful insights for our project,

particularly in terms of simplicity and user experience.
Conclusion: While Melody Scanner is not a comprehensive solution for our capstone project's real-time transcription needs, it offers valuable lessons in streamlining the user interface and simplifying the transcription process for ease of use.

19.2 Reusable Components

The primary motivation for this capstone project is to enhance our technical skills by building key components ourselves, rather than relying heavily on pre-existing solutions. By developing our own algorithms for real-time transcription, signal processing, and music notation, we will gain a deeper understanding of the challenges and intricacies of these areas. However, there are some components and methods, which could be extremely useful, listed below.

PortAudio (for Audio Input/Output) [9]

Description: PortAudio is a widely used open-source library for handling audio input and output across multiple platforms.
Rationale for Limited Use: While audio capture and output are critical components, reinventing this from scratch would require extensive low-level work that would detract from our focus on learning signal processing and transcription. Therefore, we plan to use PortAudio for audio input/output while building the higher-level processing systems ourselves.

MusicXML Format [10]

Description: MusicXML is an open standard for music notation interchange, commonly used in music software.
Rationale for Limited Use: Rebuilding a file format from scratch would not significantly contribute to our learning, so we will use MusicXML to ensure compatibility with existing music notation software. However, we will focus our learning efforts on generating and manipulating the MusicXML data from the transcription process.

19.3 Products That Can Be Copied

Although copying or modifying parts of existing products could save time and effort, our primary goal for this capstone project is to deepen our technical knowledge by solving the challenges ourselves. This section acknowledges the potential solutions but explains why we prefer not to use them, except where strictly necessary. There is one example of an open-source product available to copy from:

LilyPond (Music Notation Rendering) [11]

Description: LilyPond is an open-source music engraving program that renders high-quality sheet music. Its notation rendering capabilities could be adapted for our needs.
Rationale for Avoiding Copying: While using LilyPond for notation rendering would simplify output formatting, we aim to build our own system for translating audio to notation and rendering it. This will give us valuable insights into how music notation software operates, which we would miss by relying on LilyPond.
Approximate Time Savings: Copying LilyPond's functionality could reduce our output formatting time by 30–40%, but we would miss out on understanding the process of rendering notation from scratch.

20 New Problems

20.1 Effects on the Current Environment

1. Data Privacy: The application shall not access or make use of any personal data without explicit consent (if necessary).

2. Compatibility: The application will be developed to ensure that it does not harm or interfere with any of the existing applications on the user's device.

3. Performance Efficiency: The application will not excessively consume device resources such that it significantly degrades the performance of the user's device.

20.2 Effects on the Installed Systems

1. Application Programming Interfaces (APIs)
    - Use of any OS-specific APIs to perform tasks like file management and user interface rendering.
2. Existing User Interfaces (UIs)
    - The application's graphical user interface (GUI) may use existing frameworks to handle user input and display information.
    - Creates application shortcuts.
3. File System
    - Interactions with the device's file system to read and write files using system calls and other external libraries.
    - Saves transcribed sheet music to the device's disk in various formats.
4. Network
    - Although the application is intended to be standalone, network sockets might be used for communication purposes (e.g., for updates/patches).
5. Hardware Components
    - The application captures input signals through the existing device's microphone or audio interfaces.
    - Could use the analog-to-digital Converter (ADC) built into the device's sound card or external audio interface.

20.3 Potential User Problems

1. Access Control and Elevated Permissions
    - The application may require changes to the access control settings on the target device. This might be unexpected for some users. These changes could be necessary for audio input processing or to interact with other specific system components.

- More specifically, the application might need elevated permissions to handle audio hardware, system resources, etc. Users may be required to grant administrative access during installation.

2. Resource Consumption

   - The application might consume high amounts of resources, especially during complex audio conversions or real-time transcriptions. This could adversely affect the target system's performance, especially on devices with limited processing power or memory.
   - The battery life of portable devices may also be significantly drained as a result of the resource consumption.

20.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

1. Hardware

   - Built-in microphones that do not meet average audio quality standards may hinder the accuracy of music transcription. External audio interfaces or microphones may be necessary for the application's intended performance.
   - The application may adversly affect the battery life of the user's portable device. This could be caused by high processing demands during intense transcription sessions.

2. Operating System (OS)

   - The application's performance may vary due to the user's operating system version. Older versions may not have required libraries or optimizations that the application depends on.
   - Some operation systems might limit real-time task programming and processing, especially on systems with restrictive scheduling or low hardware capabilities.

3. Background Processes

- Resource intense background processes may lower the performance of the application especially for devices with lower hardware capabilities.

20.5 Follow-Up Problems

There exist some problems that the application may not be able to cope with. These include, but are not limited to:

- Extremely low-quality, noisy, or distorted audio input.
- Highly complex audio.
- High resource demand.
- Insufficient hardware.
- Limitations of pitch detection algorithms.

21 Tasks

21.1 Project Planning

The following table displays the tasks and associated deadlines that make up the lifecycle of the project.

21.2 Planning of the Development Phases

Phase 1: Proof of Concept

Benefit to user: This phase will result in an MVP to present to relevant stakeholders, ensuring that the development team remains aware of potential users and their related needs.
Required operational date: Nov 11/24
System components included: Backend subsytem.
Related functional requirements: FR-AI1, FR-AI3, FR-SP6
Related non-functional requirements: OE-IA1, OE-IA2, CR-CR1, CR-CR2, CR-SCR1

Due Date	Task
Sept 24/24	Problem Statement, POC Plan, Development Plan
Oct 11/24	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 8/25	Expo Demonstration
Apr 2/25	Final Documentation (Revision 1)

Table 3: Project Plan

Phase 2: Backend Development

Benefit to user: The core functionality of the system will be developed in this phase. This is projected to occur at the same time as Phase 3.
Required operational date: Jan 15/25
System components included: Backend subsytem containt the main functionalities of the system, namely signal processing of audio input and generation of output files (i.e., sheet music). This phase will also include necessary hardware components for signal processing and testing, such as a keyboard/guitar, microphone, audio interfacer, amplifiers, audio cables, and other audio equipment as necessary.
Related functional requirements: FR-SP1, FR-SP2, FR-SP3, FR-SP5, FR-SMG1, FR-SL1
Related non-functional requirements: PR-SL2, PR-PA1, PR-RFT5, PR-C1, PR-SE3, OE-EP1, OE-EP2, OE-IA3, MS-S2, MS-A1, S-A1, S-P3

Phase 3: User Interface

Benefit to user: This phase will make the system usable by the intended customer, so that technical knowledge is not necessary to operate the system. This is projected to occur at the same time as Phase 2.
Required operational date: Jan 15/25

System components included: Frontend subsytem.
Related functional requirements: FR-AI2, FR-SP4, FR-UI2, FR-UI3, FR-UI4
Related non-functional requirements: LF-A1, LF-A2, LF-A3, LF-S1, UH-EOU1, LF-S1, UH-PI1, UH-L1, UH-UP1, UH-A1, PR-SC2, OE-EP3, OE-EP4, OE-WE1, MS-A2, S-P2

Phase 4: Integration of Subsystems and Deployment

Benefit to user: This phase will result in the completion of revision 0, such that all core functionalities are present in the system and all implementation details are abstracted from user view.
Required operational date: Feb 3/25
System components included: Backend subsystem, frontend subsytem.
Related functional requirements: FR-AI4, FR-UI1, FR-SL2
Related non-functional requirements: UH-UP2, PR-SL1, PR-RFT1, PR-RFT1, PR-RFT3, PR-RFT4, PR-RFT7, PR-C2, PR-L1, OE-P1, OE-P2, OE-P3, OE-P4, MS-M1, MS-S1, S-P1

Phase 5: Refinements

Benefit to user: This phase will focus on implementing stretch goals and waiting room requirements into the system.
Required operational date: Mar 24/25
System components included:
Related functional requirements: FR-SMG2, FR-SMG3
Related non-functional requirements: PR-SE1, PR-SE2,OE-R1, MS-M2, CR-LR1

22 Migration to the New Product

22.1 Requirements for Migration to the New Product

Since our sheet music generator app is being developed from scratch and does not replace any existing system, there are no traditional migration re-

quirements from an old system to a new one. However, as users may want to transition from using existing software solutions to our product, the following minimal migration efforts are anticipated:

- **File Compatibility**: Users should be able to import existing sheet music files (MusicXML) from other music notation or production software such as Sibelius, Finale, and MuseScore. The app must support these file types and ensure accurate translation into the app's native system.
- **User Familiarization**: Since the app is new, existing users of other music transcription or notation software may need time to familiarize themselves with the workflow and interface. Providing clear tutorials and guidance within the app will be important to ensure a smooth transition for users.
- **System Requirements**: The app should clearly communicate its system requirements (e.g., minimum RAM, processor speed) to users transitioning from older or less efficient music notation tools, ensuring their hardware is compatible with the new system.

22.2 Data That Has to be Modified or Translated for the New System

Given that this project is being developed from the ground up, no legacy data exists that needs to be migrated. However, users may bring data from other platforms. The following types of data may need to be translated or modified to work within the new system:

- **MusicXML and MIDI Files**: The app must allow seamless import of MusicXML files created on other platforms. Any discrepancies or unsupported elements should be flagged, and users should have the ability to adjust elements that do not translate perfectly.
- **Audio Files**: Users may import existing audio recordings (WAV, MP3, FLAC) for transcription. The app should accommodate different audio formats and sample rates, translating these into the internal format used for transcription without losing accuracy or quality.

- **Notation Adjustments**: Imported sheet music may not always align perfectly with the app's internal notation system. In such cases, the app should automatically translate formatting elements and provide users with editing tools to make manual adjustments where necessary.

23 Costs

Mid-Range Microphones: $100 - $500:
High quality Microphones are essential to capture audio by converting sound waves into electrical signals. If necessary, it may be beneficial for the project to acquire higher quality microphones for testing.

Storage for Large Data Files:
Cloud or local storage could be an additional potential cost, as features like account management, stored sheet music, or other data could require subscription of a cloud database server ($100 - $500 per year), or purchasing more memory ($50 - $400).

24 User Documentation and Training

24.1 User Documentation Requirements

Installation and Setup Guide:
This documentation should provide clear instructions for downloading, installing, and configuring the software, including system requirements and initial setup steps to ensure users can easily get the application running on their devices.
Quick Start Guide:
This section should offer a streamlined overview of the essential features and workflow of the application, allowing users to quickly record their first piece of music and generate sheet music with minimal effort.
Tutorials and Walkthroughs:
Developers should document step-by-step guides for various user scenarios, from basic recording to advanced features, ensuring users can understand how to effectively use all functionalities of the application.
Version History and Updates:

This documentation should detail changes made in each version of the software, including new features, improvements, and bug fixes, helping users stay informed about updates and how they might impact their use of the application.

Feedback and Support Channels:

Clear documentation should outline how users can provide feedback or seek support, including contact information for customer service or a support form, enabling users to get assistance with any issues they encounter.

Glossary of Terms:

This section should define key technical and musical terminology used throughout the software, helping non-expert users understand the language and concepts related to music transcription and audio processing.

24.2 Training Requirements

To ensure sufficient resources for product use, the development team will provide:

Tutorial Videos:

Develop a series of engaging tutorial videos that visually demonstrate how to use the software, covering basic and advanced features, to cater to various learning styles and help users quickly grasp the application's functionalities.

Feedback Mechanism:

Establish a system for collecting user feedback on the training materials and sessions, enabling continuous improvement of the training content and methods based on user experiences and suggestions.

25 Waiting Room

- The system must support operation on macOS, Linux, and Windows operating systems. Specifically, the supported versions include:
  - macOS Sonoma 14.0 (or later),
  - Ubuntu 24.04 LTS (or later),
  - Windows 11 (or later).
- Real-time signal processing should be conducted using Real-Time Operating Systems (RTOS) as advised by our supervisor. A proof-of-

concept must be conducted to evaluate the feasibility of using an RTOS for this purpose.

- The system must offer basic MIDI-to-audio conversion for audible feedback, enabling users to determine the correctness of the transcription through a playback feature.
- The system must support continuous pitch handling, specifically for instruments and vocals that do not have discrete notes, such as singing and violin performances.
- The system should automatically download and install version updates when connected to the internet to ensure the user is always working with the latest software features and bug fixes.
- The system must generate optimized guitar tabs that suggest the easiest finger positioning for each note or chord based on the context of the score.
- The system must provide a feature to allow users to edit the transcribed sheet music. For example, users should be able to adjust the key signature if they find it incorrect or wish to make changes.

26 Ideas for Solution

- Real time signal processing using C/C++ for speed
- Use the libsndfile open source library for signal processing
- Deploy as a .exe file
- Use some modern framework for front-end development (e.g. React Native, Vue.js)
- The user manually identifies the tempo and key to be used in the generated sheet-music
- A sheet music editing feature compatible with the musicXML format

References

[1] C. Edney, "Everything you need to build your own pc for music production," 2024, accessed: 2024-10-05.

[2] "Web content accessibility guidelines," https://www.w3.org/TR/WCAG21/, 2024, accessed: 2024-10-11.

[3] "Seizures and physical reactions," https://www.w3.org/WAI/WCAG21/Understanding/seizures-and-physical-reactions.html, 2024, accessed: 2024-10-11.

[4] "Sibelius," https://www.sibelius.com/download/index.html, 2024, accessed: 2024-10-11.

[5] "Anthemscore," https://www.lunaverus.com/, 2024, accessed: 2024-10-11.

[6] "Musescore," https://musescore.org/en, 2024, accessed: 2024-10-11.

[7] "Transcribe!" https://transcribe.com/, 2024, accessed: 2024-10-11.

[8] "Melody scanner by klangio," https://melodyscanner.com/, 2024, accessed: 2024-10-11.

[9] "Portaudio," https://www.portaudio.com/, 2024, accessed: 2024-10-11.

[10] "Musicxml," https://www.musicxml.com/, 2024, accessed: 2024-10-11.

[11] "Lilypond," https://lilypond.org/, 2024, accessed: 2024-10-11.

[12] L. Rabiner and R. W. Schafer, "On the use of autocorrelation analysis for pitch detection," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 1, pp. 24–33, 1977.

[13] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Prentice-Hall, 1978.

[14] F. J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," in *Proceedings of the IEEE*, vol. 66, 1978.

[15] S. Dixon, "Onset detection," in *Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx'06)*, 2006.

[16] R. Zhou and J. D. Reiss, "Music onset detection," in *Machine Audition: Principles, Algorithms and Systems*, W. Wang, Ed. IGI Global, 2010, pp. 297–316.

Appendix A — Formal Math Specifications

This appendix provides detailed mathematical formulations for two pitch detection methodologies that may be employed for this project's signal processing: autocorrelation-based and Fourier-based techniques. These specifications are general and intended to be used as a simplified overview of the algorithms. The actual implementation may vary based on emprical performance.

A.1 Autocorrelation-based Pitch Detection

Description: The autocorrelation method is a time-domain approach that analyzes the periodicity of a signal to determine its fundamental frequency. It does so by computing the correlation between the signal and time-shifted versions of itself over a range of lags. The lag corresponding to the highest normalized correlation is identified as the pitch period, from which the fundamental frequency is derived.
The formulation below is based on the autocorrelation technique for pitch detection as detailed in [12] and further discussed in [13].

Mathematical Formulation: Let $x[n]$ be a discrete-time signal defined for $n = 0, 1, \ldots, N-1$. The total energy of the signal is computed as:

$$r_0 = \sum_{n=0}^{N-1} x[n]^2$$

For a given lag ℓ, the autocorrelation function is defined as:

$$r(\ell) = \sum_{n=0}^{N-\ell-1} x[n]\, x[n+\ell]$$

The normalized autocorrelation is then given by:

$$R(\ell) = \frac{r(\ell)}{r_0}$$

The algorithm searches for the lag ℓ^* within a predefined range $[\ell_{\min}, \ell_{\max}]$ that maximizes $R(\ell)$:

$$\ell^* = \underset{\ell \in [\ell_{\min}, \ell_{\max}]}{\arg\max}\ R(\ell)$$

If the maximum normalized autocorrelation $R(\ell^*)$ exceeds a threshold (e.g., 0.5), the fundamental frequency f_0 is estimated by:

$$f_0 = \frac{f_s}{\ell^*}$$

where f_s is the sampling rate. If $R(\ell^*) < 0.5$, no clear pitch is detected (i.e., $f_0 = 0$).

Parameter Constraints: The search range for ℓ is derived from the desired frequency bounds:

$$\ell_{\max} = \min\left(N-1,\ \left\lfloor \frac{f_s}{f_{\min}} \right\rfloor\right), \quad \ell_{\min} = \max\left(1,\ \left\lfloor \frac{f_s}{f_{\max}} \right\rfloor\right)$$

with typical values $f_{\min} = 100\,\text{Hz}$ and $f_{\max} = 2000\,\text{Hz}$.

Relevant Details: To reduce computational load, the lag ℓ may be incremented in steps (e.g., by 2 samples). The procedure returns the fundamental frequency f_0 if a valid peak is found; otherwise, it indicates the absence of a clear pitch by returning 0.

A.2 Fourier-based Pitch Detection

Description: The Fourier-based method analyzes the frequency content of a signal using the Fast Fourier Transform (FFT). It computes the frequency spectrum of the signal, identifying the peak frequency as the pitch.

Mathematical Formulation: The following describes the three major mathematical components that are considered for this project's signal processing framework:

1. *The Input Signal:* Let $x : \mathbb{Z} \to \mathbb{R}$ be a discrete-time signal with finite energy:

   $$E = \sum_{n=-\infty}^{\infty} |x[n]|^2 < \infty$$

   This formalizes the representation of our raw audio data.

2. *The Window Function:* To localize the Fourier analysis to specific segments and reduce spectral leakage, a window function is applied [14]. For a window length $N \in \mathbb{Z}^+$ (with $N \geq 1$), we define the Hanning window $w : \{0, 1, \ldots, N-1\} \to \mathbb{R}$ as:

$$w(n) = \begin{cases} 0.5 - 0.5 \cos\left(\frac{2\pi n}{N-1}\right), & 0 \leq n \leq N-1, \\ 0, & \text{otherwise} \end{cases}$$

3. *The Short-Time Fourier Transform (STFT):* The STFT converts segments of the time-domain signal into the frequency domain, enabling the analysis of evolving spectral content [15, 16]. Given a hop size $H \in \mathbb{Z}^+$, each frame of the signal is defined as:

$$\{x[mH + n] : n = 0, 1, \ldots, N-1\}$$

where $m \in \mathbb{Z}$ is the frame index. The Fourier transform of the windowed frame is computed by:

$$X(m, k) = \sum_{n=0}^{N-1} x[mH + n]\, w(n)\, e^{-j\frac{2\pi}{N}kn}$$

with $k \in \{0, 1, \ldots, N-1\}$ representing the frequency bin index.

Appendix B — 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. What went well while writing this deliverable?
   Mark This deliverable was a good opportunity to coordinate our efforts amongst a much larger body of work compared to the first deliverable. By maintaining a continuous line of feedback and communication, our team helped avoid contradictions within the documentation, and sections such as "stretch goals", "future plans", "waiting room" etc were very helpful in having the group think forward and define exactly what was happening for sure vs what might happen provided things go well.

   Ian I think we communicated and split work well during this deliverable. In some ways this deliverable acted as the official foray into the capstone project because it was much larger than the previous deliverables. It also required that we further narrow down our vision for the project which I think excited us all.

   Emily This deliverable really helped us to gain a better understanding of the different aspects of this project we'll be taking on. It can be difficult to have a thorough understanding of a project as large as this one from the get go, and I think that we worked really well as a team to communicate and prioritise the features we'd like to see in the final product.

Jackson I thought this was great because the deliverable cemented how we will work as a team. I thought we communicated very well on each section, referencing each other's work and asking for clarification when needed. I also enjoyed our open discussions when we were discussing what we would be implementing in the future, as thinking ahead is always good for these projects

2. What pain points did you experience during this deliverable, and how did you resolve them?
   Mark A pain point of the deliverable was trying to minimise repetition in similar sections of the requirements document. It often felt as though the same thing was being said multiple times throughout the different sections, and it was difficult to maintain a cohesive flow among sections.

   Ian One of the pain points I encountered was trying to apply the Volere template (specifically business data and adjacent business analysis) to a project that is more or less standalone. The template is definitely geared towards very complex projects that involve making additions to existing systems and interacting with many other organisations. While our project does do this in some sense, it was difficult to draw parallels when using the template and its descriptions of the sections.

   Emily The big pain point I ran into was trying to differentiate what a functional vs non-functional requirement should look like. This is something I've had trouble with since I first started writing requirements, and I found myself going back and forth between which category a requirement belonged to. I ended up just doing a lot of research on how to write a good requirement, and I found that having examples to compare this project against was what helped me gain clarity in this process.

   Jackson There was one main pain point I experienced when doing this deliverable, which was dealing with non-functional vs functional requirements. This has always been a tough thing for me to rationalise in my head since sometimes they can be very close to each other. In the case of this deliverable, I was dealing with the performance non-

functional requirements which made me second guess myself since they sounded a lot like functional requirements. I resolved that by learning more about the differences between the two, and how non-functional requirements can deal with the performance of the system whereas functional requirements deal with how the system must work.

3. How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?

   The majority of requirements were derived through personal experience and understanding of necessary features for an acceptable product. However, requirements related to user proficiency in music theory were formed by communicating with potential clients with varying levels of music understanding. This helped to determine what may be assumed to be common knowledge, versus what may require additional documents for users to understand

4. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.

   Mark In terms of technical knowledge, the most valuable courses will have been 3MX3 and 3DX4, both focused on signal processing, a key aspect of the project. Additionally, experiential courses like 3XB3 and 2AA4 provided a strong foundation in building Git-based projects and collaborating on code within a team.

   Ian 3MX3 has to be the course that relates most to this project. This course provides valuable insight into the world of signal processing which is arguably the most important part of the project. Other than that, 4AA4 has given some valuable knowledge about real-time tasks and scheduling which will be useful depending on the direction we take with the goals of the project. 4HC3 also fits very well into this project as it gave me experience not only in conducting stakeholder/client analysis but also in designing interfaces that put a large focus on user experience.

Emily At this point in the course, the largest similarities I've seen have been with SFWRENG 3A04, Large System Design. This course was set up very similarly to capstone, as most of the marks received came from a term-long project with a large focus on requirements elicitation, use cases, and stakeholder definitions. Going back to my notes from 3A04 helped me a lot for this deliverable, and will likely continue to be a main source of truth moving forward.

Jackson A large part of this project will be signal processing, for which I have taken 3MX3 and 3DX4 which directly deal with the subject matter that we'll need to use to process the signals. Additionally, all of the assignment-based courses (e.g. 2aa4, 3bb4) gave me the experience to feel confident when starting these larger projects.

5. 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.

   Successful completion of this capstone project will be reliant on the team acquiring both technical and non-technical skills. Project management has already proven to be a vital part of this project, requiring leadership, an understanding of agile methodologies, and an ability to delegate work where necessary. Music theory and composition knowledge is the entire basis of our problem statement, and will be a necessary skill to ensure our system is fulfilling its purpose and providing accurate transcriptions. UI/UX design will need knowledge of graphic design, the qualities of a good user interface, and prototyping tools such as Figma. Software testing and quality analysis skills will be exercised through use of our Kanban board and ensuring we have a robust suite of regression tests in our CI pipeline. Signal processing and related controls skills will also be an important part of processing audio in-

put from the user. Given these 5 knowledge areas, each member of our group can choose one to laser focus on and ensure our breadth of knowledge sufficiently covers the topics needed in our project's success.

6. 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?

   For all of the knowledge areas mentioned above, there are two approaches that we can apply in the context of this project. The first is self-directed learning with online resources. There are plenty of free courses, tutorials, tools, and texts online that the team can take advantage of. The second approach is to develop and practise these skills with small personal projects, experiments, or even by contributing to open source projects that involve the skills. Both of these approaches are reasonable given that the team is composed of full-time students and because both cater well to self-directed and self-paced learning. Due to the breadth of knowledge and skills required by the project, the team will collaboratively pursue both of these approaches. Therefore, individually, each team member will pursue both simultaneously sharing newly gained skills with the team where possible. This choice is a result of the desire for the team to increase collaboration and to tackle the knowledge gaps we may have as efficiently as possible.