Software Requirements Specification for ScoreGen: subtitle describing software

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

October 11, 2024

Contents

Revision History

Date	Version	Notes
09/10/2024	1.0	Initial Revision

1 Purpose of the Project

1.1 User Business

Insert your content here.

1.2 Goals of the Project

Insert your content here.

2 Stakeholders

2.1 Client

Insert your content here.

2.2 Customer

Insert your content here.

2.3 Other Stakeholders

Insert your content here.

2.4 Hands-On Users of the Project

Insert your content here.

2.5 Personas

Insert your content here.

2.6 Priorities Assigned to Users

Insert your content here.

2.7 User Participation

Insert your content here.

2.8 Maintenance Users and Service Technicians

Insert your content here.

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 this source.
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.

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

- **DAW**: Digital Audio Workstation
- **SNR**: Signal to Noise Ratio

5 Relevant Facts And Assumptions

5.1 Relevant Facts

Insert your content here.

5.2 Business Rules

Insert your content here.

5.3 Assumptions

Insert your content here.

6 The Scope of the Work

6.1 The Current Situation

Insert your content here.

6.2 The Context of the Work

Insert your content here.

6.3 Work Partitioning

Insert your content here.

6.4 Specifying a Business Use Case (BUC)

Insert your content here.

7 Business Data Model and Data Dictionary

7.1 Business Data Model

Insert your content here.

7.2 Data Dictionary

Insert your content here.

8 The Scope of the Product

8.1 Product Boundary

Insert your content here.

8.2 Product Use Case Table

Insert your content here.

8.3 Individual Product Use Cases (PUC's)

Insert your content here.

9 Functional Requirements

9.1 Functional Requirements

Insert your content here.

10 Look and Feel Requirements

10.1 Appearance Requirements

Insert your content here.

10.2 Style Requirements

Insert your content here.

11 Usability and Humanity Requirements

11.1 Ease of Use Requirements

Insert your content here.

11.2 Personalization and Internationalization Requirements

Insert your content here.

11.3 Learning Requirements

Insert your content here.

11.4 Understandability and Politeness Requirements

Insert your content here.

11.5 Accessibility Requirements

Insert your content here.

12 Performance Requirements

12.1 Speed and Latency Requirements

Insert your content here.

12.2 Safety-Critical Requirements

Insert your content here.

12.3 Precision or Accuracy Requirements

Insert your content here.

12.4 Robustness or Fault-Tolerance Requirements

Insert your content here.

12.5 Capacity Requirements

Insert your content here.

12.6 Scalability or Extensibility Requirements

Insert your content here.

12.7 Longevity Requirements

Insert your content here.

13 Operational and Environmental Requirements

13.1 Expected Physical Environment

Insert your content here.

13.2 Wider Environment Requirements

Insert your content here.

13.3 Requirements for Interfacing with Adjacent Systems

Insert your content here.

13.4 Productization Requirements

Insert your content here.

13.5 Release Requirements

Insert your content here.

14 Maintainability and Support Requirements

14.1 Maintenance Requirements

Insert your content here.

14.2 Supportability Requirements

Insert your content here.

14.3 Adaptability Requirements

Insert your content here.

15 Security Requirements

15.1 Access Requirements

Insert your content here.

15.2 Integrity Requirements

Insert your content here.

15.3 Privacy Requirements

Insert your content here.

15.4 Audit Requirements

Insert your content here.

15.5 Immunity Requirements

Insert your content here.

16 Cultural Requirements

16.1 Cultural Requirements

Insert your content here.

17 Compliance Requirements

17.1 Legal Requirements

Insert your content here.

17.2 Standards Compliance Requirements

Insert your content here.

18 Open Issues

Insert your content here.

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**
  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**
  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**
  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!**
  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**
  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 which could be extremely useful listed below.

PortAudio (for Audio Input/Output)

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

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)

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

Insert your content here.

20.2 Effects on the Installed Systems

Insert your content here.

20.3 Potential User Problems

Insert your content here.

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

Insert your content here.

20.5 Follow-Up Problems

Insert your content here.

21 Tasks

21.1 Project Planning

Insert your content here.

21.2 Planning of the Development Phases

Insert your content here.

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 requirements 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 tran-

sitioning 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

Insert your content here.

24 User Documentation and Training

24.1 User Documentation Requirements

Insert your content here.

24.2 Training Requirements

Insert your content here.

25 Waiting Room

Insert your content here.

26 Ideas for Solution

Insert your content here.

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. 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.

2. 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?