Hazard Analysis
ScoreGen

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

Table 1: Revision History

Date	**Developer(s)**	**Change**
25-10-2024	Jackson, Emily, Mark, Ian	Revision 0
28-10-2024	Jackson	Issue 123: peer review

Contents

1 Introduction **1**

2 Scope and Purpose of Hazard Analysis **1**

3 System Boundaries and Components **1**

3.1 The Application 1

3.1.1 User Interface (UI) 1

3.1.2 Back-end Software 2

3.2 User Device 2

3.3 User External Input Devices 2

4 Critical Assumptions **2**

5 Failure Mode and Effect Analysis **3**

6 Safety and Security Requirements **4**

7 Roadmap **4**

7.1 Safety-Critical Requirements 4

7.2 Privacy Requirements 5

7.3 Access Requirements 5

7.4 Audit and Immunity Requirements 5

List of Tables

1 Revision History i
2 FMEA 3

1 Introduction

Hazards within a software system are not physically dangerous in the same way other engineering systems can be, but nonetheless there are still inherent properties of this system that when combined with its environment have the potential to cause harm. In the case of ScoreGen, a hazard is any situation or condition that may produce an undesirable outcome for any of the system's stakeholders. The hazardous outcomes of this system are not likely to cause physical harm, but may instead provide harm to a user's mental wellbeing, reputation, and other intangible ill-effects. There is a possibility of physical hazards that arise from interfacing with external systems, which may cause bodily harm in addition to the non-physical effects.

2 Scope and Purpose of Hazard Analysis

Hazards within the system have potential to incur losses on behalf of the developers, users, and other stakeholders. Performing a hazard analysis on the ScoreGen system can reduce the scope of potential losses and ensure the correct mitigation strategies are in place. Risks and liabilities of the system are likely to reduce user satisfaction with the system, pushing them away from our product and losing their trust. Accumulation of such occurrences will damage the system developers' reputation, making it difficult to compete with other software systems that provide similar functionalities.

This hazard analysis will focus on potential hazards arising from the software system being designed (ScoreGen), but hazard scope is extended to include various external interfacing systems as well. These systems include an Audio Interfacer (a device needed to process continuous signals from a cable audio input), the user's instrument of choice, and the device being used to run the application.

3 System Boundaries and Components

Outlined below are the key components that define the system and its interactions. These include both software and hardware elements.

3.1 The Application

3.1.1 User Interface (UI)

A graphical user interface (GUI) that allows the users to interact with the application. The user will use the GUI to upload audio files, export files, and view system feedback.

3.1.2 Back-end Software

The core of the application that processes user input, it can be further broken down into the following:

- **Raw audio data processing:** Read in audio signals and convert to symbolic music notation (e.g., .WAV to MIDI).
- **Transcription:** Convert the processed data to music layout formats for rendering into sheet music.
- **File management:** Export symbolic music data or rendered sheet music (MIDI, musicXML, PDF).

3.2 User Device

The device that the user will install and run the application on. The device provides the existing hardware and software that the application will interact with, such as the operating system (OS) and relevant system libraries that perform I/O operations (e.g., file management, audio device access, etc.).

3.3 User External Input Devices

Other devices or hardware the user may have, such as microphones, audio interfaces, or instruments.

4 Critical Assumptions

Critical Assumptions About Users

User Environment

Users will operate in a reasonably quiet environment when recording to minimize background noise and ensure clear audio capture.

User Knowledge and Skills

Users possess a basic understanding of how to connect and use audio equipment (e.g., microphones) and navigate the application interface effectively.

Technical Proficiency

Users have a general level of comfort and familiarity with technology, enabling them to troubleshoot minor issues independently without significant frustration.

User Intent

Users are genuinely interested in creating music and will not intentionally misuse the product or engage in harmful behaviors that could disrupt functionality.

Hardware Availability

Users have access to compatible devices and equipment necessary for the application to function properly (e.g., computers, microphones).

5 Failure Mode and Effect Analysis

Table 2: FMEA

Failure Mode and Effects Analysis **System**: Audio to sheet music generator **Phase/Mode**: System Requirements						
Design Function	**Failure Modes**	**Effects of Failure**	**Causes of Failure**	**Recommended Action**	**SR**	**Ref.**
Generate sheet music	Sheet music flashes on the screen	Potential trigger for users with epilepsy.	Incorrect rendering or scrolling through the sheet music	Limit screen effects that could potentially flicker	PR-SC1	
	A bad microphone causes notes to be off due to pitch drift	Embarassment for the user	The user microphone has pitch drift (mechanical issue)	Suggest that the User acquire a new/higher quality microphone	N/A	
Process audio	Audio improperly recorded	Loss of musical performance	Microphone error	Notify the user that there is a microphone issue	N/A	
			Signal processing error	Notify the user that a signal processing error that has occurred	FR-SP4	
File functions	Files outside scope of the application are deleted	Hardware failure if BIOS and system critical files are deleted	Improper file I/O actions	Limit system access	S-P1	

	Files outside scope of the application are modified	Hardware failure if BIOS and system critical files are modified	Improper file I/O actions	Limit system access	S-P1	
Output audio	Overly loud audio emitted	Hearing damage	Volume settings improperly calibrated	Advise user to check their device's volume settings	N/A	
		User is surprised and has a heart attack	Volume settings improperly calibrated	Advise user to check their device's volume settings	N/A	
Application runtime	App crashes unexpectedly	User loses progress	Power outage, OOM error, device overheats, malware	Regular automated saving, if allowed by user	PR-RFT3	

6 Safety and Security Requirements

Currently, there are no new safety or security requirements, but any new requirements will be added to this document as part of our iterative development process. Newly discovered requirements will also be evaluated and incorporated into the SRS to ensure alignment with evolving project needs and insights gained during development.

7 Roadmap

We will aim to fulfill every one of the security and safety-critical requirements within the timeline of the capstone course, and we will be loosely following the compliance of the requirements based on the priority assigned below:

7.1 Safety-Critical Requirements

- **PR-SC1 Epilepsy Safety**: *High priority*, as it ensures the app's visual safety for users from the start. Compliance with WCAG 2.1 should be confirmed early on to prevent any risks during development.
- **PR-SC2 Instrument Input Setup**: *High priority* for both user safety and accurate functionality, ensuring users can set up their instruments without misconfiguration. This interactive guide can be iteratively refined through user testing early in development.

7.2 Privacy Requirements

- **S-P2 PII**: *High priority*, as ensuring the app does not collect any Personal Identifiable Information or session logs aligns with the app's privacy-first design. This requirement should be validated as soon as any data processing is introduced.
- **S-P3 Input Data**: *Low priority*, focusing on ensuring that temporary files are properly cleared after processing is complete. Implementing this is less critical since we will not be storing any data outside of the user's system.
- **S-P1 Data Storage**: *Medium priority*, ensuring that user-generated data is stored in local, user-chosen locations. This feature should be verified once output functions are introduced.

7.3 Access Requirements

- **S-A1 User Authentication**: *High priority*, by removing the need for login functionalities, we will speed up initial development. It should be verified after the core interface is complete.

7.4 Audit and Immunity Requirements

- **Audit Requirements (N/A)**: *Low priority*, as the system is offline and does not require developer responsibility over user data.
- **Immunity Requirements (N/A)**: *Low priority*, as no data storage minimizes exposure to security risks.

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?
   Mark This deliverable went very smoothly. It demonstrated that the group is on the same page regarding project understanding as there was minimal conflict, and the progress of the task was relatively consistent throughout its duration.

   Ian Outlining the system boundaries and components further solidified my understanding of the project and helped me to neatly segment it in my mind. Thinking about and contributing to the analysis also allowed me to recognize that software still has relevant hazards despite the fact that it is non-physical. Without this deliverable, I probably would have given little thought to potential software hazards in other projects or at the very least, undermined its importance.

   Emily Writing the introduction section for this deliverable helped me to gain an understanding of how we can mitigate risks and avoid hazards in our project, despite our system not containing many hazards in the traditional sense. When I think 'hazard analysis' my mind goes to things like WHMIS and high voltage training - even the lab trainings I've done for software courses have only ever focused on physical hazards. That being said, there are certainly still other ways for a person to incur loss, and putting this deliverable together painted a better picture in my mind of how we can do our best to mitigate risks throughout our capstone project.

   Jackson I think this deliverable was more straightforward than others which allowed us all to have a clear picture of what is expected. As a result, we were able to accomplish the sections we designated as 'group sections' very easily as we could have conversations about hazards without any confusion. This was also helped by our group member Emily, who kindly laid out clear definitions of hazards for us to follow.

2. What pain points did you experience during this deliverable, and how did you resolve them?
   Mark The most significant challenge was revising my understanding of hazards, since my traditional approach would be to focus on physical dangers, but due to the software nature of the project, I had to consider issues regarding customer satisfaction, or developer reputation as well. Additionally, it was difficult to determine which hazards could be considered reasonable, as opposed to ones that cannot be addressed and would not be our fault. Fleshing out the critical assumptions helped resolve this pain point and create a better understanding of how to think of new hazards.

   Ian One of the pain points I experienced was the feeling that the system components section was lacking in quantity. I sometimes mistakenly associate quantity with quality. This was easily resolved just by going through everything our team has fleshed out for this project so far with a fine tooth comb and making sure no major components are missing.

   Emily The part of this deliverable that I thought went well was simultaneously the largest pain point - brainstorming the possible non-physical hazards for ScoreGen. The process overall helped me to expand my understanding of a hazard, but it took a lot of back and forth for us to settle on and come up with a viable set of potential hazards. This is the kind of pain point that can't really be resolved, but just has to be worked through instead. Talking aloud and throwing ideas at one another helped to ease the brainstorming process and finalize the FMEA.

   Jackson I think there was one main pain point that stood out in my mind, which was redundancy. This came about because much of what we are discussing in this document was already discussed at length in the form of security and safety requirements, along with use cases and risks. While we were writing some of the sections it felt very redundant to be repeating the points we already made, and in an effort to reduce redundancy, we tried our best to link things back to other sections, which proved to be difficult since the deliverables were so large in scope.

3. Which of your listed risks had your team thought of before this deliverable, and which did you think of while doing this deliverable? For the latter ones (ones you thought of while doing the Hazard Analysis), how did they come about?

   Team-level Reflection:
   Before writing this deliverable our team thought that unexpected crashes

would appear as a potential hazard. Since software systems aren't inherently 'hazardous,' we saw unexpected crashes as a more general, intuitive hazard applicable to similar applications. All of the other hazards came about through brainstorming as a team. Our approach to brainstorming proved very effective and made the process of thinking of hazards easier. We first approached this by trying to come up with hazards directly but found that identifying the design functions first was easier. After identifying these functions, we began exploring each area individually rather than approaching the whole system at once.

4. Other than the risk of physical harm (some projects may not have any appreciable risks of this form), list at least 2 other types of risk in software products. Why are they important to consider?

   Legal Risk: Software products can carry legal risks, such as copyright infringement, data privacy violations, or failing to comply with industry regulations. These risks are important to consider because they can result in lawsuits, fines, or a loss of trust in the company. Ensuring compliance with laws and obtaining necessary licenses or permissions can mitigate these risks.

   User Satisfaction Risk: A software product that doesn't meet user expectations or is difficult to use can lead to poor user satisfaction. This is crucial because low satisfaction can result in negative reviews, loss of users, and damage to our brand's reputation. Prioritizing user experience through thorough testing and feedback can help address this risk.