UNIVERSITAT POLITÈCNICA DE CATALUNYA

INTELLIGENT SYSTEM PROJECT

MIDTERM DOCUMENT

Master in Artificial Intelligence

Authors:

GERARD CARAVACA IBÁÑEZ
BENJAMÍ PARELLADA CALDERER
ARMANDO RODRIGUEZ RAMOS

Supervisor:

MIQUEL SÀNCHEZ MARRÈ

Fall Term 2023/2024

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Contents

1 Introduction

The Wave2Page system introduces an innovative approach to Automatic Music Transcription (AMT) within the field of digital music processing, focusing on the conversion of audio recordings into digital sheet music. This project integrates techniques from Signal Processing, Deep Learning, and Music Theory to address the challenges inherent in accurately transcribing polyphonic music. Its primary function is to convert audio files into MIDI format, subsequently translating them into high-quality sheet music. Through this process, Wave2Page aims to effectively link the auditory experience of music with its visual representation in written form.

Music transcription, particularly of polyphonic compositions, is a task of considerable complexity and nuanced difficulty. It involves the meticulous decoding of layered musical elements, from individual notes and rhythms to the subtleties of dynamics and articulation. This process is further complicated when dealing with polyphony, where multiple voices or instruments intertwine, each contributing to the rich tapestry of the overall piece. Accurately capturing this complexity in a written format poses significant challenges [1, 2], requiring not only advanced technological solutions but also a deep understanding of musical structure and theory. The Wave2Page system is engineered to meet these challenges, employing State-of-the-Art research to render musical performances into precise, readable sheet music.

While the traditional process of music composition often involves a composer creating sheet music as a primary step, the reverse process, transcription, holds substantial value in various domains. This inverse transcription is crucial for preserving historical performances where no written score exists, as exemplified by Allegri's 'Miserere mei, Deus.' This piece, famously transcribed by Wolfgang Amadeus Mozart, demonstrates how transcription can capture and immortalize otherwise ephemeral musical masterpieces. Furthermore, transcription opens up new avenues for music education, enabling students to study and learn from contemporary pieces and performances that may not have corresponding sheet music. Moreover, it assists musicians and researchers in analyzing and understanding the stylistic nuances of different artists and genres. For composers and arrangers, it provides a means to gain inspiration and insight from existing works, facilitating the creation of new compositions or adaptations. Additionally, in the realm of digital music archiving and library sciences, accurate transcription systems like Wave2Page contribute significantly to the preservation and accessibility of musical works. By bridging these gaps, the Wave2Page system not only aids in the transcription of music but also enriches the cultural and educational aspects of music, broadening the reach and understanding of this art form in the digital age.

This report provides a comprehensive examination of the Wave2Page system. In Sections 2 and 3, we detail the system's specifications and functionalities, highlighting the technical and musical requirements that guide our approach. Section 4 offers an overview of the current system architecture, explaining the processes involved in audio file handling, instrumental track separation, MIDI conversion, and sheet music generation. In Section 5, we discuss the proposed design solutions, exploring various alternatives and potential enhancements to optimize the system's performance. Finally, Section 6 addresses the project management aspects, including task delegation, scheduling, and risk management strategies.

2 Specifications of the Problem to be Solved

The primary goal of the project is to develop the Wave2Page system, a robust and efficient solution designed to convert audio recordings into digital sheet music. This system aims to address the complex process of audio-to-notation conversion, utilizing advanced audio processing and music transcription techniques.

Audio Format Conversion and Processing: The system is engineered to support a wide range of common audio formats, including MP3, WAV, and FLAC. These inputs are seamlessly converted into a unified internal format, ensuring consistent processing across various file types. This capability is central to meeting diverse user needs without limiting their options.

Instrumental Separation: At its core, the system adeptly separates complex polyphonic audio files into discrete instrument tracks. It primarily targets instruments prevalent in pop music and rock, such as piano, guitar, bass, drums, and vocals. The focus is on achieving high-fidelity separation for as many instruments as possible, with the scope currently constrained by technological limitations in instrument detection and isolation.

Internal Representation: Following the instrument separation phase, the system translates the individual tracks into a detailed internal format. This step is crucial as it involves capturing essential musical elements including note pitch, velocity, and rhythm with high accuracy. MIDI serves as a prime example of such a format, renowned for its precision in reflecting the intricacies of a performance, thus ensuring a faithful representation of the original music.

Postprocessing and Conversion to Sheet Music: This module involves refining the internal musical representation, paying particular attention to aspects such as musical ornaments that require nuanced treatment. The system then converts this refined data into sheet music, compatible with mainstream notation software. This conversion process is carefully executed to preserve the musical integrity of the original piece, while simplifying it into notation.

System Outputs: The primary output is a sheet music file, tailored to user preferences. Options include ensemble sheet music encompassing all instruments or selected subsets, and access to intermediate outputs like individual instrument tracks or the internal MIDI representation, offering users a range of possibilities for exploration and use.

User Interface: Designed for accessibility and ease of use, the interface caters to users of varying technical backgrounds, with features that enable the generation of complete sheet music, selection of specific instrument parts, and access to intermediate outputs.

System Flexibility and Modularity: The architecture of the Wave2Page system is designed to be both modular and flexible, allowing for straightforward updates or replacement of its components. This adaptable design positions the system to evolve alongside technological advancements and shifting user requirements.

System Responsibilities: The system must ensure user anonymity and data protection by not storing any information about processed music pieces. Robust security protocols must be implemented for user privacy and data integrity. To ensure this, at the end of processing one request, the input data and intermediate steps should be all deleted. Additionally, the system is designed to deliver results within a practical timeframe, with a focus on usability and scalability.

It should be capable of handling a high volume of transcription requests from numerous users simultaneously. Furthermore, it must adhere to the licensing terms of all utilized open-source libraries, ensuring legal compliance and ethical usage of resources.

3 Requirement Analysis of the System

In this section, we will explain the functional and non-functional requirements that we consider important for this project. Even though they provide a comprehensive framework for planning and executing an Automatic Music Transcription tool, these requirements may vary throughout the project development. On the one hand, functional requirements outline the system's intended operations under normal conditions, detailing the essential end functions. On the other hand, non-functional requirements, while not directly addressing the system's operational aspects, are crucial from the outset. They play a significant role in shaping the project's objectives and must be considered throughout the entire development process.

3.1 Functional

The functional requirements are the following:

- **Multiple Instruments and tracks**: Support the conversion of audio for various musical instruments and voices, accommodating different pitch ranges, timbres, and instruments like piano, guitar, violin, etc. into separate music sheets for each instrument or voice.
- **Machine Learning and AI**: Incorporate machine learning and AI algorithms to achieve effortless correct conversions, hoping we obtain better results than previous tools.
- **Format Compatibility**: Ensure compatibility with standard music notation formats (e.g. MusicXML), making it easy for musicians to work with and edit the generated music sheets.
- **Feedback Mechanism**: Implement a feedback mechanism to collect user feedback and improve the system based on user experiences.
- **User Profile Management**: Allow users to create and manage their profiles, enabling them to save their preferences, track their usage history, and access their converted music sheets.
- **Documentation and Support**: Provide comprehensive documentation, tutorials, and customer support to assist users in effectively utilizing the tool.

3.2 Non-Functional

The non-functional requirements are the following:

- **Accuracy**: Ensure that the converted music sheets are as accurate as possible when compared to the original audio. This includes accurately transcribing notes, rhythms, dynamics, and other musical nuances.

- **Speed and Efficiency**: Aim for fast processing times to convert audio to music sheets, especially for longer pieces of music.
- **Robustness**: Ensure that the system can handle a variety of audio qualities, including recordings with background noise, varying levels of instrument/vocal clarity, and different recording environments.
- **Accessibility**: Make the tool accessible to individuals with disabilities, ensuring that it can be used by people with visual or hearing impairments and the usage of a user-friendly and intuitive interface for musicians and composers to upload audio files and receive music sheets.
- **Scalability**: Design the system to handle a large volume of audio conversions, especially if the project gains popularity.
- **Security and Data Privacy**: As users will be uploading potentially original compositions and performances, ensure the security and confidentiality to protect intellectual property and personal data.
- **Multi-Platform Compatibility**: Design the tool to be compatible with various operating systems and devices, including mobile, to broaden its accessibility.
- **Localization and Language Support**: Supporting multiple languages for the tool's interface to cater to a global user base.

4 Functional Architecture of the Intelligent System

In this section, we present the architecture of the Wave2Page system, as illustrated in Figure 1. This architecture is designed to execute a three-step process, essential for transforming audio recordings into digital sheet music. The process begins with an initial stage where various music formats are converted into the uncompressed Waveform Audio File (WAV) format, setting a standard baseline for subsequent processing. Following this, the system engages in three pivotal steps: the *Splitter*, *Wav2Midi*, and *Midi2Sheet*.

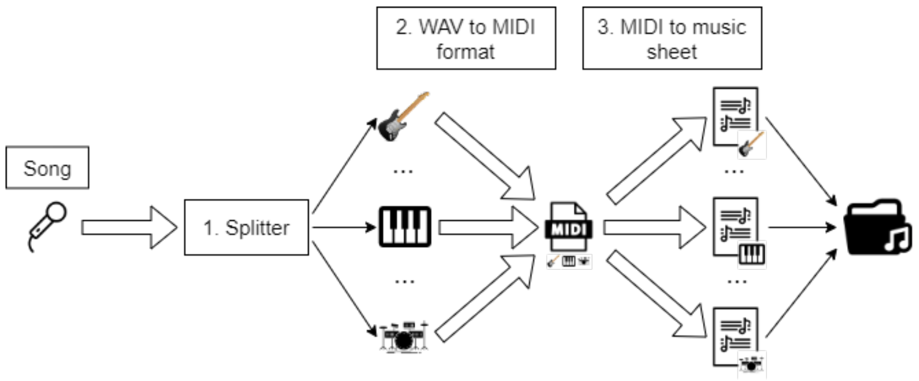


Figure 1: Functional architecture of the Wave2Page Intelligent System.

The first stage, *Splitter*, is tasked with deconstructing a WAV audio file into separate tracks for each instrument. This step is crucial for isolating the distinct musical elements within a polyphonic performance. Following this, the *Wav2Midi* stage takes over, where these individual tracks are converted into a unified MIDI file. This file effectively captures the essence of the music in a digital format, preparing it for the transcription stage, while also allowing for easy manipulation of the musical information. Finally, the *Midi2Sheet* process converts the MIDI file into detailed sheet music. Each instrument's part is transcribed onto its own score, enabling musicians and conductors to interpret and perform the music accurately.

Our system's architecture thus mirrors the logical progression of a human transcribing a musical piece. Starting from differentiating the amalgamation of sounds in the *Splitter*, followed by the harmonization and quantization of each instrument in the *Wav2Midi*, and culminating in the tangible musical notation in the *Midi2Sheet*. This well-conceived architecture ensures that the transformation from raw audio to polished musical notation is achieved with precision and clarity. Furthermore, it is designed with sufficient flexibility, allowing for enhancements and extensions to the system while maintaining the integrity of the processing pipeline.

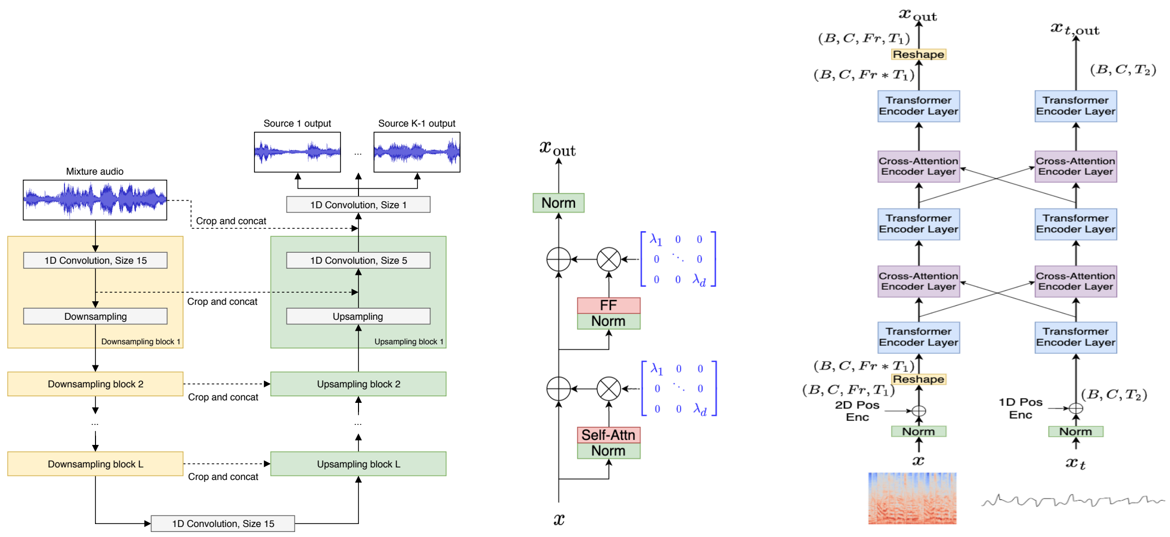
In the upcoming sections, we will delve into the intricacies of each stage in detail. It's important to note that our system also incorporates external libraries to enhance its functionality, nevertheless these are not as relevant as the ones which will be explained. For instance, we use the *librosa* [3] Python package, a tool specialized in music and audio analysis, to extract critical preliminary information like the song's tempo. This package plays a vital role in providing foundational elements necessary for the development of music information retrieval systems, further supporting the robustness of our architecture.

4.1 Splitter

The *Splitter* module occupies a pivotal position in our system, being responsible for the separation of different audio tracks from a single WAV song. This crucial process lays the foundation for accurate transcription by isolating distinct musical elements. To achieve this, we have employed the Demucs Transformer model [4, 5], a State-of-the-Art solution in music source separation.

Demucs stands out for its proficiency in separating elements like drums, bass, and vocals from a complex musical accompaniment. Initially based on a U-Net convolutional architecture inspired by Wave-U-Net [6], Demucs has evolved significantly. Its version 4, known as Hybrid Transformer Demucs, blends a hybrid spectrogram/waveform separation approach with Transformer architecture. This advanced version is built upon the foundations of Hybrid Demucs, with the crucial modification of replacing the innermost layers with a cross-domain Transformer Encoder. This Encoder employs self-attention within each domain (temporal and spectral) and cross-attention across these domains.

The original framework included two U-Nets, one handling temporal convolutions in the time domain and the other managing convolutions along the frequency axis in the spectrogram domain, as seen in Figure 2 (a). Both U-Nets comprised five encoder and decoder layers each. The Hybrid Transformer Demucs retains the outermost four layers of this structure but introduces a transformative change in the two innermost layers of both encoder and decoder, incorporating a cross-domain Transformer Encoder. This innovative Encoder processes the 2D signal from the spectral branch and the 1D signal from the waveform branch simultaneously. It operates effectively with heterogeneous data shapes, a significant enhancement over the original model, which necessitated precise parameter tuning for time and spectral representation alignment. The crux of this model lies in its cross-domain Transformer Encoder, comprising five layers that alternate between self-attention and cross-attention encoders in both spectral and waveform domains.



(a) Hybrid Transformer Demucs

(b) Transformer Encoder Layer

(c) Cross-domain Transformer Encoder of depth 5

Figure 2: Details of the Hybrid Transformer Demucs architecture. (a): Wave-U-Net with K sources and L layers. In the output layer, the K-th source prediction is the difference between the mixture and the sum of the other sources [6]. (b): the Transformer Encoder layer with self-attention and Layer Scale. (c): The Cross-domain Transformer Encoder treats spectral and temporal signals with interleaved Transformer Encoder layers and cross-attention Encoder layers [7].

For our project, we have selected the Hybrid Transformer Demucs 6s model. This experimental model is particularly proficient in segmenting drums, bass, vocals, guitar, and piano, although its performance with piano is currently lacking. In pursuit of isolating and extracting a broader range of instruments, we are considering several alternatives. This includes exploring other pre-trained models and configurations from diverse Python libraries, which may offer different strengths in terms of instrument recognition and separation.

Moreover, there lies a significant opportunity in developing and fine-tuning custom artificial neural network models tailored for source separation. By venturing into this domain, we can create models that not only excel in accurately segregating audio tracks but also maintain a high standard of audio quality. Such an endeavor is not only an exciting avenue for innovation but also a step towards pushing the boundaries of music source separation. These custom models, specifically designed to meet our unique requirements, could substantially enhance the effectiveness and precision of the Splitter module, thereby elevating the overall performance of our system in the complex task of music transcription.

4.2 Wav2Midi

The *Wav2Midi* module plays a pivotal role in our system, tasked with converting individual instrument WAV files into a unified MIDI file that captures the entirety of a song, neatly organized into distinct tracks. As detailed in the architecture, this phase is crucial for categorizing notes, greatly simplifying the transformation of a musical piece into a coherent and legible sheet of music. In enhancing the richness of our MIDI files, we incorporate the song's tempo, adding essential musical context that aids in producing visually informative scores, while additionally helping the models performance. For this vital conversion process, we employ *basic-pitch* [8], a State-of-the-Art convolutional neural network designed for audio-to-MIDI AMT.

The efficacy of *basic-pitch* stems from its innovative architecture and multipitch support, enabling it to adeptly handle a variety of musical elements and instruments with a relatively small model size. This makes *basic-pitch* particularly advantageous for our purposes, as it combines high note accuracy with efficiency, rivaling the performance of larger, more resource-intensive AMT systems. Our workflow primarily uses the model's inference function to generate MIDI files from the WAV inputs. These MIDI outputs, representing the distinct tracks of each instrument, are then merged into a complete musical composition. This merging process is facilitated by the *Mido* library [9], selected for its ease of use and versatility in handling MIDI data.

The architecture of *basic-pitch*, illustrated in Figure 3, is a fully convolutional model that takes audio as input and produces three posteriorgram outputs. A posteriorgram is a time-frequency matrix encoding probabilities of musical events such as note onsets, active notes, and pitches. These outputs have the same number of time frames as the input Constant-Q Transform (CQT), a transformation that provides a time-frequency representation of the audio signal. Moreover, the model employs a Harmonic CQT (HCQT) to align harmonically-related frequencies, facilitating the capture of related information through small convolutional kernels. This process involves shifting the CQT vertically by frequency bins corresponding to each harmonic, using up to 7 harmonics and 1 sub-harmonic. The model's architecture is tailored to exploit the differing properties of its three outputs: Y_p for multipitch information, Y_n for note event information, and Y_o for note onsets. Notably, the model eschews the need for full-frequency receptive fields or complex data augmentations, thanks to its efficient design. Post-processing of these posteriorgrams involves creating note events defined by start and end times and pitches, utilizing Y_o and Y_n. This process includes peak picking, likelihood assessments, and tracking through Y_n to form note events, ensuring that only the most significant musical elements are retained.

As we continue to refine our approach, the exploration of alternative tools like *music21* [10] offers us additional flexibility and functionality. This exploration is pivotal in ensuring that our system meets a wide range of musical needs with the highest quality and accuracy in the MIDI outputs. Additionally, we anticipate further fine-tuning and potentially developing custom artificial neural network models for the task, aiming to eliminate sound artifacts and enhance the overall quality of the music sheet results. Finally, we want to underscore that this component is subject to continuous enhancement, particularly in the realm of post-processing. Future work will focus on meticulously curating the generated MIDI files to ensure they are optimally formatted for the subsequent translation into sheet music.

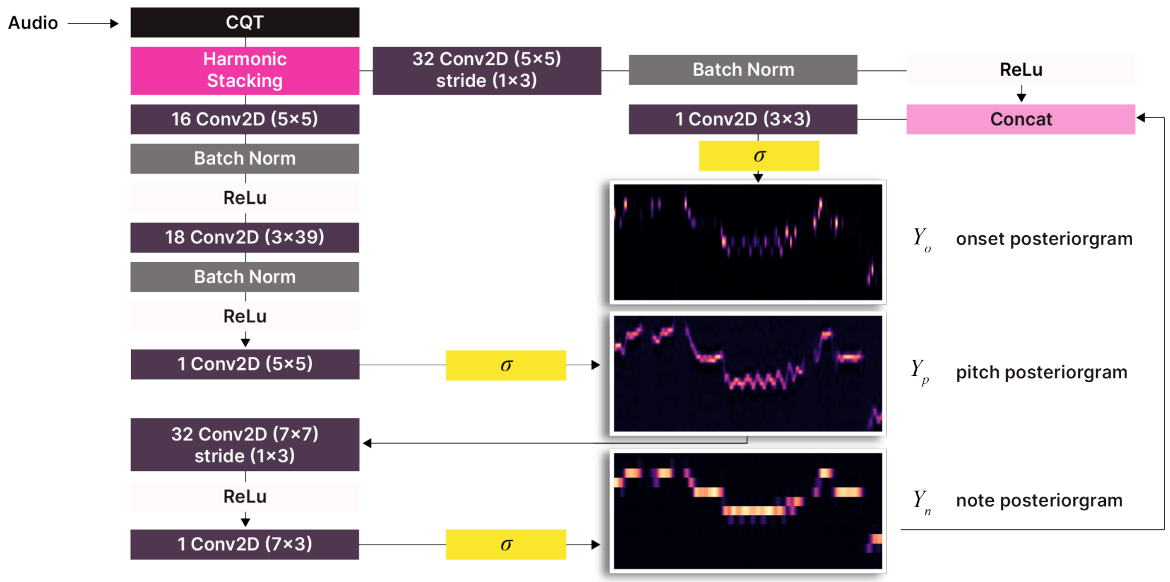


Figure 3: Basic-Pitch architecture [8]. The matrix posteriorgram outputs Y_o, Y_p, and Y_n are outlined in green. σ indicates a sigmoid activation.

4.3 Midi2Sheet

The final and decisive stage of our workflow is the *Midi2Sheet* module. This stage is responsible for transforming the MIDI file, produced in the previous module, into a set of detailed music sheets. These sheets are meticulously designed to accurately reflect the music for each instrument in the ensemble, culminating in a well-rounded and coherent musical score.

To achieve this, we utilize MuseScore4 [11], a renowned and multifaceted software known for its proficiency in music notation. MuseScore4 excels in converting MIDI files into music sheets that are both expressive and visually appealing. Its broad range of features and exceptional output quality make it the tool of choice for our Midi2Sheet conversion process.

We produce not just one, but multiple music sheets tailored to meet diverse needs. We begin by creating a master music sheet that encapsulates all individual tracks of the composition. This serves as a valuable resource, particularly for the music conductor, allowing them to grasp the intricate interaction between the various instruments, as understanding the interplay of instruments is key to effective orchestration and musical direction.

Additionally, we produce individual music sheets for each instrument. These sheets offer a focused depiction of what each instrument contributes to the overall composition. By doing so, we significantly ease the musicians' task of accessing and studying their specific parts. This segmented method of presenting the music not only augments the musicians' understanding and performance but also streamlines the rehearsal process, making it more targeted and efficient.

In essence, the *Midi2Sheet* process, powered by MuseScore4, plays a pivotal role in bridging the gap between digital audio and physical sheet music. It presents a robust, adaptable solution that caters to the intricate needs of both the music director and performers, ensuring that every participant in the musical endeavor has access to high-quality, accurate music sheets.

It's worth noting that prior to the *Midi2Sheet* stage, considerable effort is put into pre-processing the MIDI files to ensure their readiness for sheet music conversion. This pre-processing includes refining note sequences, adjusting timings, and eliminating any artifacts that may have arisen during the earlier stages of our workflow. As a result, the MIDI files fed into MuseScore4 are already optimized to a significant degree. However, it remains an open question whether further improvements will be necessary in the *Midi2Sheet* process beyond meticulous formatting and presentation of the sheets. As we continue to evaluate and refine our system, we remain open to the possibility of enhancing this phase to better suit the nuanced requirements of sheet music production, ensuring that the final output not only meets but exceeds the expectations of musicians and directors alike.

5 Proposed Solution Design

5.1 Initial Task Analysis

In the forthcoming section, we will thoroughly examine the foundational aspects of our project by splitting it into tasks. This analysis is pivotal as it lays the groundwork for all subsequent stages of project development. We will explore the various stages of the project, specifying each task in detail to provide clarity and direction. Additionally, this section will outline the key milestones, presenting a roadmap for the project's progress and success. We will also discuss the methods and alternatives considered for accomplishing these tasks, highlighting our strategic approach and decision-making process. This comprehensive analysis is crucial for ensuring a well-structured and effective project execution.

5.1.1 Project Stages

To bring clarity and structure to our project planning, we have categorized the tasks into distinct phases. This systematic approach allows us to organize and define each task more effectively. The phases we have identified are as follows:

- **Project Management**: This phase encompasses all tasks related to defining, organizing, and managing the the project's progress.
- **Preliminary Study**: This phase is dedicated to building a foundational understanding of the key concepts in digital music production, providing a solid theoretical base for the project.
- **Viability Analysis**: This phase involves assessing the economic and social feasibility of the project, including a comprehensive risk analysis.
- **Documentation**: Focused on the creation and maintenance of project documentation, this area covers the preparation of technical documents, user manuals, progress reports, and other essential materials for ensuring project transparency and facilitating knowledge dissemination.
- **Application Design**: This phase is centered around the design of the application's user interface and user experience, aiming to craft an intuitive and aesthetically pleasing interface that aligns with the project's goals and user requirements.
- **Implementation**: This phase is where the project plan is put into action, implementing each component incrementally, starting from a basic prototype and progressively enhancing its functionality.

- **Testing and Evaluation**: In this stage, the project is subjected to thorough testing to verify its performance and functionality, ensuring it meets the intended objectives and specifications.

5.1.2 Task definition

The following section outlines the various tasks that constitute our project, organized into distinct blocks corresponding to each phase of the project. These tasks have been defined according to our goals from the first document, and task dependencies have been formed.

Project Management

- Problem Definition (M1) - Clarify the specific challenge or issue that the project aims to address.
- Study Requirements (M2) - Analyze and document the essential features and constraints of the project.
- Define Goals (M3) - Set clear, measurable objectives for the project's outcome.
- Develop a Plan (M4) - Create a comprehensive strategy, including timelines, resources, and milestones.
- Conduct Weekly Meetings (M5) - Hold regular sessions to discuss progress, obstacles, and plan the next steps.

In the Project Management phase, the initial step is Problem Definition (M1), which is essential to set the project's scope and focus before other tasks can begin. Following this, the Study Requirements (M2) phase takes place to define the project's needs based on the identified problems. Defining Goals (M3) can occur concurrently with M2 or immediately afterward to establish clear project objectives. The Development of a Plan (M4) relies on a clear understanding of the problem, requirements, and goals and thus depends on the completion of M1, M2, and M3. Conducting Weekly Meetings (M5) is a continuous task that begins after M1 and carries on throughout the project.

Preliminary Study

- Study Music Source Separation (S1) - Research techniques and methods to distinguish individual sounds within a music piece.
- Study MIDI Transcription (S2) - Investigate approaches to convert audio signals into MIDI format.
- Study Sheet Music Generation (S3) - Explore tools and algorithms for transforming audio or MIDI into notated music.
- Analyze Prominent Studies (S4) - Review leading research and breakthroughs related to the project's focus.

In the Preliminary Study, tasks such as studying Music Source Separation (S1), MIDI Transcription (S2), Sheet Music Generation (S3), and analyzing Prominent Studies (S4) are generally independent activities. However, they should ideally commence after completing M1 to align with the project's objectives.

Viability Analysis

- Identify and Analyze Risks (V1) - Detect potential pitfalls or challenges and evaluate their impact.
- Conduct Sustainability Analysis (V2) - Assess the long-term feasibility and ecological responsibility of the project.
- Perform Economic Analysis (V3) - Examine the financial implications, including costs, benefits, and ROI.

In the Viability Analysis phase, Identifying and Analyzing Risks (V1) is ideally initiated after completing the Study Requirements to understand potential challenges. Conducting Sustainability Analysis (V2) and Performing Economic Analysis (V3) can start concurrently with V1.

Documentation

- Develop "Definition of the Project Document" (D1) - Gather initial findings, plans, and research from the early phase.
- Develop "Midterm IS Project Document" (D2) - Deliver a detailed report on the project's mid-stage achievements and challenges.
- Prepare Midterm Presentation (D3) - Develop slides for the midterm showcase.
- Develop "Final IS Project Document" (D4) - Detail the entire project, from inception to completion.
- Create a User Guide (D5) - Produce a comprehensive manual for end-users detailing functionality and usage.
- Prepare Final Presentation (D6) - Develop slides or materials for the concluding showcase.

The Documentation phase involves developing the "Definition of the Project Document" (D1) after M1, M2, and M3 are completed. The "Midterm IS Project Document" (D2) and the preparation of the Midterm Presentation (D3) should start following significant progress in the Implementation phase (I1, I2, I3). Towards the project's end, the focus shifts to developing the "Final IS Project Document" (D4), creating a User Guide (D5), and preparing the Final Presentation (D6), after most implementation (I4 to I7) and testing tasks (T1, T2, T3) are completed.

Solution Design

- Design the Architecture (DS1) - Structure the system's high-level components and their interactions.
- Define Interaction Design (DS2) - Outline how users will engage with the system.
- Develop User Interface Design (DS3) - Craft visual and functional elements for user interaction.

In the Solution Design phase, the Design of the Architecture (DS1) begins post M2 to ensure alignment of the system architecture with the project requirements. This is followed by the Definition of Interaction Design (DS2) and the Development of User Interface Design (DS3).

Implementation

- Implement Initial *Splitter* (I1) - Develop the first version of the audio separation tool.
- Implement Initial MIDI Transcription (I2) - Build the preliminary system for converting audio to MIDI.
- Implement Initial Sheet Music Generation (I3) - Produce the first iteration of the music notation generator.
- Postprocess *Splitter* (I4) - Refine and enhance the audio separation tool based on feedback and testing.
- Improve MIDI Transcription (I5) - Continuously enhance the accuracy and efficiency of the MIDI transcription process.
- Customize Sheet Music Generation (I6) - Optimize the conversion process from audio or MIDI to sheet music, adding features to refine the output.
- Implement User Interface (I7) - Construct interactive front-end components for user navigation and utilization.

The Implementation phase starts with the initial implementation tasks (I1, I2, I3) after the architecture design (DS1). This is followed by subsequent improvements and post-processing tasks (I4, I5, I6). The Implementation of the User Interface (I7) commences after completing the User Interface Design (DS3).

Testing and Evaluation

- Test Individual Components (T1) - Evaluate each module or element separately to ensure functionality.
- Test the System (T2) - Examine the integrated system for overall performance and reliability.
- Conduct User Testing (T3) - Gather feedback from real or potential users to identify usability issues and improvement areas.

Lastly, in the Testing and Evaluation phase, Testing of Individual Components (T1) begins after the initial implementation tasks (I1, I2, I3). System Testing (T2) is conducted after completing all implementation tasks (I1 to I7). User Testing (T3) is most effective after T2, ensuring the system is fully integrated and functional.

5.1.3 Milestones

The project is structured around a series of four significant milestones, each with associated deliverables and targeted deadlines.

MS1: Definition of the Project. The first milestone focuses on the initial phase, involving the clear articulation of the project's scope and objectives. The associated deliverable for this phase is the "Definition of the Project Document" (MS1-D1), which was expected to be completed and submitted by *September 28th, 2023*. This deliverable was successfully completed and submitted by the designated deadline.

MS2: Midterm Reporting. As the project progresses to its mid-point, a comprehensive report capturing the more in depth the project definition, achievements, and challenges encountered is required. This is encapsulated in the "Midterm IS Project Document" (MS2-D2) and "Midterm Presentation" (MS2-D3). The deadline for this deliverable is set for *November 16th, 2023*. This deliverable has been successfully completed and submitted in the designated deadline.

MS3: Final Software and Documentation. This decisive milestone marks the culmination of the project's primary objectives, where the final product is ready for delivery. Three key deliverables are associated with this phase. They are the "Final IS Project Document" (MS3-D4), "Create User Guide" (MS3-D5), and the "User Interface" (MS3-I7), which culminates the end of the software development. All these deliverables share the same deadline, which is *January 14th, 2024*.

MS4: Public Project Exposition and Defense. The project's journey culminates with a public presentation, where the team will present and defend the project's achievements, challenges, and learnings. The preparation for this defense includes the creation of "Final Presentation" (MS4-D6). This document will serve as the main tool for the public defense of the project, and it is due on *January 15th, 2024*.

5.2 Methods and alternatives

Having established the tasks to be undertaken, the implementation phase is recognized as crucial for the success of the project. Each task in the implementation phase is integral, encompassing a range of essential building blocks from the initial audio separation to crafting the final user interface. Given the complexity and multifaceted nature of these tasks, it is crucial to understand not only the primary methods by which they might be achieved but also possible alternative approaches. Such an approach ensures a robust and adaptable design process, capable of overcoming potential challenges.

In the upcoming section, we will delve into the specific methods and alternative strategies for the *implementation* (I) tasks, as the other tasks are pretty straightforward. Our exploration will cover established techniques, innovative approaches, and contingency plans, providing a thorough framework for the successful completion of the project.

5.2.1 Splitter Implementation (I1, I4)

***Main Method**: Using Demucs API*

DEep MUSic Source Separation (Demucs) [7] developed by Meta AI stands out as the State-of-the-Art approach to music source separation. Harnessing the power of transformers, Demucs has been designed to efficiently and accurately isolate individual instruments and vocals from mixed audio tracks. By leveraging the Demucs API, we can tap into this robust technology, making the integration straightforward and seamless.

Alternatives:

Spectral Clustering [12], rooted in the theory of Spectral Graph Theory, presents an innovative approach to the task of audio source separation. By dissecting a similarity graph into discrete, non-overlapping clusters, it aims to distinctly segregate various audio sources. A key advantage of spectral clustering is its reliance on traditional techniques rather than deep learning approaches. Additionally, it can be particularly beneficial for audio sources with pronounced spectral differences. However, it is important to acknowledge that while spectral clustering holds significant potential, it may require precise tuning and might not always achieve the same level of accuracy as deep learning methods, particularly in complex audio compositions. Nevertheless, it can be useful for removing background noise.

Alternatively, another option could be to train a **Wave-U-Net model** [6]. This architecture is a variation of the conventional U-Net designed specifically for processing 1D waveforms, and has garnered attention for its capabilities in the realm of audio separation. Beginning with the baseline model provides an opportunity to familiarize ourselves with its fundamental operations, enabling us to tailor it to our specific project requirements. However, it's crucial to recognize that developing a model, whether from scratch or from a pretrained stage, demands considerable investment in terms of time and data resources. This approach could be particularly useful in addressing certain limitations identified in the Demucs model. For instance, the Wave-U-Net could be fine-tuned to enhance piano sound extraction, and then using the Demucs separation for the rest of the instruments.

5.2.2 MIDI Transcription Implementation (I2, I5)

***Main Method**: Leveraging the Basic Pitch Library*

Basic Pitch [8] library emerges as our primary choice for MIDI transcription in the early stages of development. Developed by the talented team at Spotify's Audio Intelligence Lab, Basic Pitch offers a nimble and remarkably swift audio-to-MIDI conversion. Its standout feature is the ability to detect pitch bends, and it demonstrates impressive versatility, functioning effectively with recordings from a vast array of instruments, including vocal tracks.

***Alternatives**:*

CREPE [13], a specialized deep learning model tailored for pitch tracking within audio signals, excels at precisely estimating pitch and translating it into MIDI notes. Its specialized focus on pitch tracking addresses a pivotal aspect of MIDI transcription, presenting a notable advantage. While integrating CREPE into the project might require fine-tuning to adapt it to specific musical genres or styles, it is a viable alternative to test.

Additionally, the **Differentiable Digital Signal Processing (DDSP)** [14] library presents an innovative approach to audio-to-MIDI conversion. DDSP integrates classic signal processing elements with deep learning techniques, focusing on audio synthesis. It offers high-fidelity generation by leveraging strong domain knowledge in signal processing and perception, while still maintaining the expressive power of neural networks. DDSP's modular nature allows for manipulations like independent control of pitch and loudness, making it a versatile tool for tasks such as dereverberation and timbre transformation. This approach provides a balance between interpretability and the benefits of deep learning, making DDSP an intriguing alternative for our audio-to-MIDI transcription needs. As discussed earlier, all deep learning models, including CREPE and DDSP, have the potential to be fine-tuned to our needs.

5.2.3 Sheet Music Generation Implementation (I3, I6)

***Main method**: MuseScore4*

MuseScore 4 is the latest version of MuseScore [11], a popular free and open-source music notation software. It allows musicians to compose, edit, and print sheet music. This version is known for its enhanced user interface, improved playback capabilities, and various new features designed to streamline the music composition process. MuseScore 4 supports a wide range of musical notations and can be used for everything from simple compositions to complex scores, catering to both amateurs and professionals.

***Alternatives*:**

Another option could be to apply the **ABC notation** [15]. Opting for the ABC style presents an alternative method for representing musical information. The standout advantage of ABC notation is its simplicity. On the downside, the very simplicity that makes it appealing can also be its limitation. For compositions with complex musical elements or intricate notations, ABC might fall short in capturing all the nuances.

5.2.4 User Interface implementation (I7)

***Main method**: Flask web application*

Flask [16] is a lightweight web application framework written in Python. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. Flask offers simplicity, flexibility, and fine-grained control. It does not require particular tools or libraries and does not have a database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

***Alternatives*:**

Tkinter is a standard GUI (Graphical User Interface) toolkit for Python. It's not a direct alternative to Flask, as Flask is used for web applications while Tkinter is used for creating desktop applications. The learning curve for this technology is relatively low, yet we have determined that a web-based format would be more appropriate for user interaction.

Another alternative is **Streamlit** [17], a Python library designed to simplify and accelerate the creation of web applications for data science and machine learning projects. While it shares similarities with Flask, this technology is more oriented towards the data science community. However, Flask offers a broader range of tools and is generally regarded as a more versatile and comprehensive solution.

6 Project Management

6.1 Task Assignment

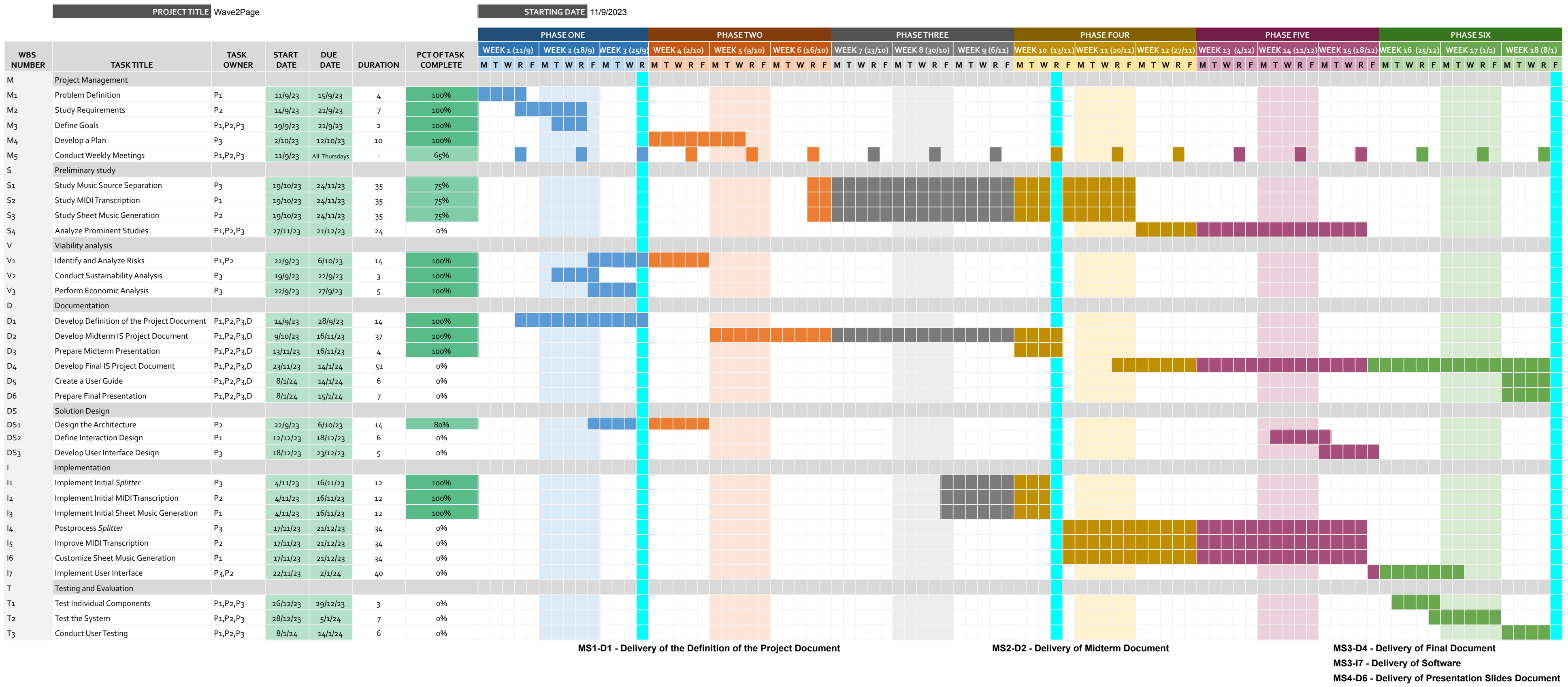
In the complex environment of a multifaceted project, clear distinction of responsibilities ensures both efficiency and accountability. This section outlines the distribution of tasks among our team, consisting of three dedicated developers (P1: Benjamí, P2: Armand, P3: Gerard) and a director (D). The director, with an overarching view of the project, is primarily involved in giving support and feedback to the developers, while the developers, handle the technical dimensions. It is worth noting that while tasks are assigned to individual members, collaboration and mutual support are the principles of our team's operations. Table 1 captures the essence of our distributed workflow, serving as both a guide and a record of our structured approach.

ID	Task	Assigned Member	Duration (h)	Dependencies
M	**Project Management**		**46**	
M1	Problem Definition	P1	2	
M2	Study Requirements	P2	3	M1
M3	Define Goals	P1, P2, P3	1	M1
M4	Develop a Plan	P3	4	M2, M3
M5	Conduct Weekly Meetings	P1, P2, P3	36	
S	**Preliminary Study**		**75**	
S1	Study Music Source Separation	P3	20	M1
S2	Study MIDI Transcription	P1	20	M1
S3	Study Sheet Music Generation	P2	20	M1
S4	Analyze Prominent Studies	P1, P2, P3	15	M1
V	**Viability Analysis**		**8**	
V1	Identify and Analyze Risks	P1, P2	3	M2, S
V2	Conduct Sustainability Analysis	P3	3	M2, S
V3	Perform Economic Analysis	P3	2	M2, S
D	**Documentation**		**119**	
D1	Develop "Definition of the Project Document"	P1, P2, P3, D	14	M1, M2, M3
D2	Develop "Midterm IS Project Document"	P1, P2, P3	37, D	I1, I2, I3, M4, S
D3	Prepare Midterm Presentation	P1, P2, P3	4	D2
D4	Develop "Final IS Project Document"	P1, P2, P3	51	I7, T1-T3
D5	Create a User Guide	P1, P2, P3	6	I7, T1-T3
D6	Prepare Final Presentation	P1, P2, P3	7	D4
DS	**Solution Design**		**10**	
DS1	Design the Architecture	P2	5	M2, M3
DS2	Define Interaction Design	P1	2	DS1
DS3	Develop User Interface Design	P3	3	DS2, I1, I2, I3
I	**Implementation**		**105**	
I1	Implement Initial *Splitter*	P3	10	DS1, DS2
I2	Implement Initial MIDI Transcription	P1	10	DS1, DS2
I3	Implement Initial Sheet Music Generation	P2	10	DS1, DS2
I4	Postprocess *Splitter*	P3	20	I1, DS3, S
I5	Improve MIDI Transcription	P1	20	I2, DS3, S
I6	Customize Sheet Music Generation	P2	20	I3, DS3, S
I7	Implement User Interface	P3, P2	15	I4-I6, DS3
T	**Testing and Evaluation**		**16**	
T1	Test Individual Components	P1, P2, P3	3	I1, I2, I3
T2	Test the System	P1	7	I1-I6
T3	Conduct User Testing	P2, P3	6	I7
Total			**379**	

Table 1: Summary of the task assignment, with their dependencies, and projected workload. Take into consideration the dependencies are sequential, so while the dependencies on M4 are M2 and M3, we need to recursively complete all dependencies of each before we can begin M4. The final workload hours sum is computed from the top level tasks, which in turn are the sum of their corresponding low level tasks. For some tasks, the Director is needed to give feedback which we will act upon. (Benjamí - P1, Armand - P2, Gerard - P3, Miquel - D).

6.2 Task Scheduling

In this section, we delve into the intricate details of how tasks are planned and organized over the course of the project. To provide a clear and comprehensive visual representation of this schedule, we will present a Gantt chart (see Figure 4). This chart is an essential tool for project management, offering a timeline view that outlines the start and end dates of various project components. It allows us to effectively illustrate the sequence and duration of tasks, dependencies between tasks, and key milestones. The Gantt chart serves not only as a planning mechanism but also as a progress-tracking tool, ensuring that all team members are aligned and informed about the project timeline and their respective responsibilities.



PROJECT TITLE Wave2Page

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
M	Project Management					
M1	Problem Definition	P1	11/9/23	15/9/23	4	100%
M2	Study Requirements	P2	14/9/23	21/9/23	7	100%
M3	Define Goals	P1,P2,P3	19/9/23	21/9/23	2	100%
M4	Develop a Plan	P3	2/10/23	12/10/23	10	100%
M5	Conduct Weekly Meetings	P1,P2,P3	11/9/23	All Thursdays	-	65%
S	Preliminary study					
S1	Study Music Source Separation	P3	19/10/23	24/11/23	35	75%
S2	Study MIDI Transcription	P1	19/10/23	24/11/23	35	75%
S3	Study Sheet Music Generation	P2	19/10/23	24/11/23	35	75%
S4	Analyze Prominent Studies	P1,P2,P3	27/11/23	21/12/23	24	0%
V	Viability analysis					
V1	Identify and Analyze Risks	P1,P2	22/9/23	6/10/23	14	100%
V2	Conduct Sustainability Analysis	P3	19/9/23	22/9/23	3	100%
V3	Perform Economic Analysis	P3	22/9/23	27/9/23	5	100%
D	Documentation					
D1	Develop Definition of the Project Document	P1,P2,P3,D	14/9/23	28/9/23	14	100%
D2	Develop Midterm IS Project Document	P1,P2,P3,D	9/10/23	16/11/23	37	100%
D3	Prepare Midterm Presentation	P1,P2,P3,D	13/11/23	16/11/23	4	100%
D4	Develop Final IS Project Document	P1,P2,P3,D	23/11/23	14/1/24	51	0%
D5	Create a User Guide	P1,P2,P3,D	8/1/24	14/1/24	6	0%
D6	Prepare Final Presentation	P1,P2,P3,D	8/1/24	15/1/24	7	0%
DS	Solution Design					
DS1	Design the Architecture	P2	22/9/23	6/10/23	14	80%
DS2	Define Interaction Design	P1	12/12/23	18/12/23	6	0%
DS3	Develop User Interface Design	P3	18/12/23	23/12/23	5	0%
I	Implementation					
I1	Implement Initial *Splitter*	P3	4/11/23	16/11/23	12	100%
I2	Implement Initial MIDI Transcription	P2	4/11/23	16/11/23	12	100%
I3	Implement Initial Sheet Music Generation	P1	4/11/23	16/11/23	12	100%
I4	Postprocess *Splitter*	P3	17/11/23	21/12/23	34	0%
I5	Improve MIDI Transcription	P2	17/11/23	21/12/23	34	0%
I6	Customize Sheet Music Generation	P1	17/11/23	21/12/23	34	0%
I7	Implement User Interface	P3,P2	22/11/23	2/1/24	40	0%
T	Testing and Evaluation					
T1	Test Individual Components	P1,P2,P3	26/12/23	29/12/23	3	0%
T2	Test the System	P1,P2,P3	28/12/23	5/1/24	7	0%
T3	Conduct User Testing	P1,P2,P3	8/1/24	14/1/24	6	0%

Figure 4: Gantt Diagram of the tasks. The tasks of our project are systematically visualized in a Gantt Diagram. Key milestones, as detailed in Section 5.1.3, are highlighted in Arctic Blue for ease of reference. The first significant milestone is the completion of the "Definition of the Project," marking a crucial initial stage of our work. The second milestone aligns with the completion of the "Midterm Document" and "Midterm Presentation," signifying a phase where the problem specifications have been more thoroughly refined and developed. The final milestone, marked by the last blue line, represents the deadline for submitting a copy of the "Final IS Project Document," "User Guide," and "Final Presentation" in preparation for the project defense. This deadline also encompasses the submission of the developed software as a User Interface.

The Gantt Diagram is intuitively designed to be self-explanatory. Each square on the diagram approximates four hours of work, aligning with the task assignments detailed in Table 1. This arrangement not only reflects the planned workload but also allows some flexibility to accommodate any unforeseen delays that may arise during the project's progression.

6.3 Risk Management

6.3.1 Risk Identification

As in any professional project of this magnitude, unforeseen events and problems can arise. Therefore, it is always important to make a list of the main setbacks that we may encounter during the course of the work:

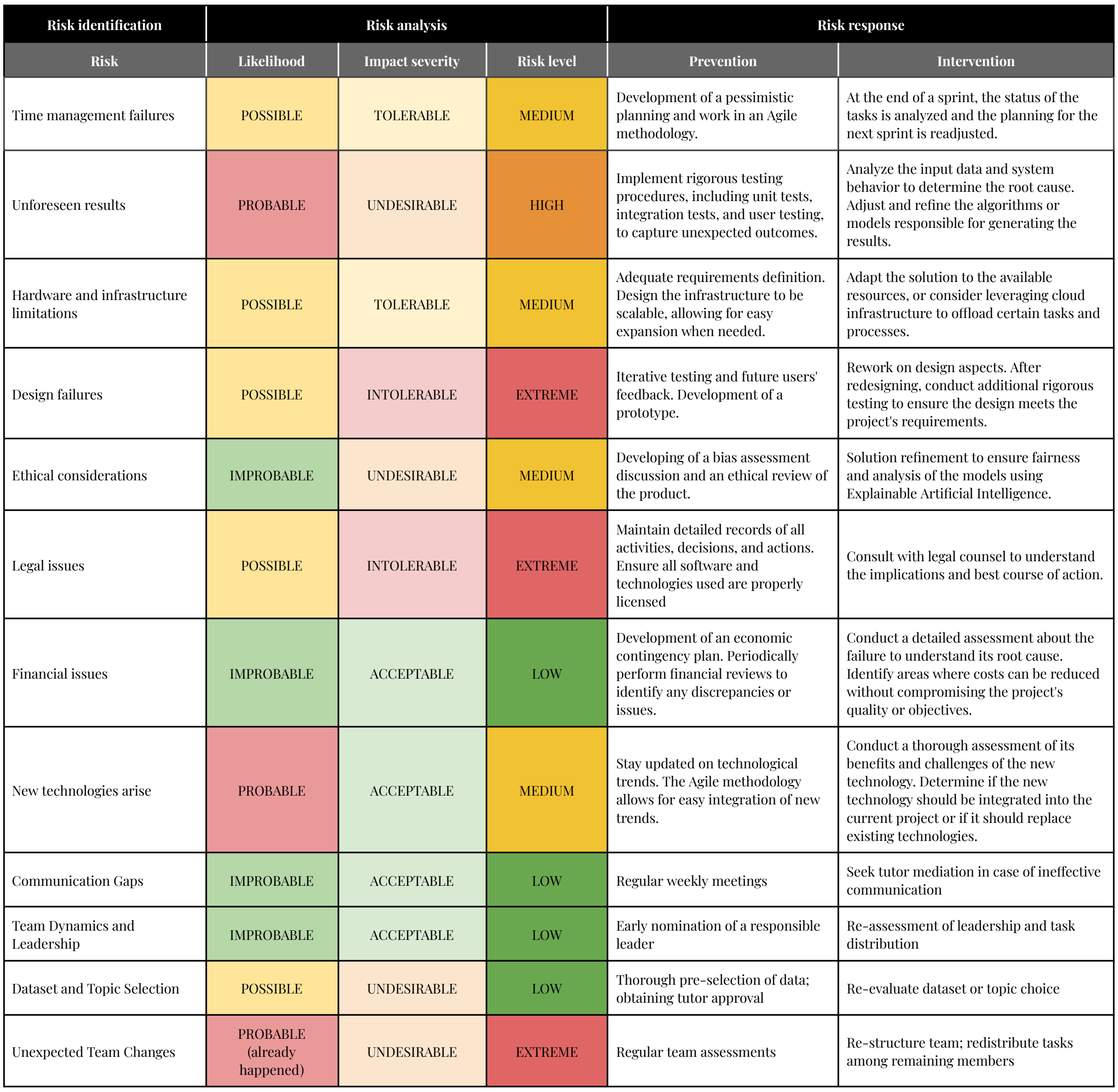
- **Time management failures:** in this case, we are talking about a poor forecast of the time it will take to complete a task. It is due to unrealistic forecasts in the planning or successions of consecutive risks.
- **Unforeseen results:** one of the inherent challenges in all kind of projects is the unpredictability of results. The model's performance might not meet the desired expectations, or it may exhibit unexpected behaviors. These issues could arise due to various factors, including inadequate training data, model architecture, or hyperparameter choices. Addressing unforeseen results may require iterative model refinement, which can impact project timelines and resource allocation.
- **Hardware and infrastructure limitations:** Deep learning often demands significant computational resources. Hardware failures, resource bottlenecks, or scalability issues could disrupt the workflow. Adequate provisioning and monitoring of hardware and infrastructure are essential to ensure the project runs smoothly.
- **Design failures:** these types of errors are due to the fact that at some point in the design phase of the project we made a mistake. Therefore, they cause us to modify the initial design, and may include errors in efficiency, scalability and data acquisition.
- **Ethical considerations:** As with any AI project, ethical concerns must be addressed. This includes issues related to bias in data or algorithms, the potential for misuse, and unintended consequences. To prevent this, an in-depth study must be conducted to verify that the model performs well regardless of the musical style or source of the input song.
- **Legal issues:** When working with data, algorithms, or technology, there's a potential for legal challenges. This could be related to intellectual property rights, data privacy regulations, or the use of copyrighted content.
- **Financial issues:** Financial setbacks can significantly derail a project. These might include unexpected costs, budget overruns, or a lack of funds to secure necessary resources. An accurate financial forecast, continuous monitoring of expenditures, and having a contingency fund can help in managing such risks.
- **New technologies arise:** The world of technology, especially in the domain of AI and deep learning, is rapidly evolving. While this is exciting, it also poses a risk. A new technology or methodology could emerge midway through the project, rendering the current approach obsolete or less efficient.
- **Communication Gaps**: While infrequent, lapses in team communication can occur, potentially leading to misunderstandings or misalignments in project objectives. Regular weekly meetings are crucial to maintain clarity, with the option of seeking mediation from a tutor if communication becomes ineffective.
- **Team Dynamics and Leadership**: The risks associated with team dynamics include the lack of a clear leader, which could lead to disorganized task distribution and prioritization.

It's crucial to nominate a responsible individual early on to ensure orderly progress.

- **Dataset and Topic Selection**: Inadequate datasets or errors in topic selection could pose significant risks. Ensuring a thorough pre-selection of data and obtaining tutor approval for the project topic can minimize these risks.
- **Unexpected Team Changes**: The possibility of a team member leaving or being unable to contribute significantly requires prompt attention. In such cases, re-structuring the team and re-distributing tasks equitably among remaining members is necessary.

6.3.2 Risk Analysis and Response

While unforeseen challenges are inevitable in any complex project, this risk analysis, constant vigilance, and a dedicated team can mitigate these risks and keep the project on course. Our core strategies will involve routine risk evaluations and adaptability to the evolving circumstances. Figure 5 shows the risks identified, together with possible prevention and intervention actions.



Risk identification	Risk analysis			Risk response	
Risk	Likelihood	Impact severity	Risk level	Prevention	Intervention
Time management failures	POSSIBLE	TOLERABLE	MEDIUM	Development of a pessimistic planning and work in an Agile methodology.	At the end of a sprint, the status of the tasks is analyzed and the planning for the next sprint is readjusted.
Unforeseen results	PROBABLE	UNDESIRABLE	HIGH	Implement rigorous testing procedures, including unit tests, integration tests, and user testing, to capture unexpected outcomes.	Analyze the input data and system behavior to determine the root cause. Adjust and refine the algorithms or models responsible for generating the results.
Hardware and infrastructure limitations	POSSIBLE	TOLERABLE	MEDIUM	Adequate requirements definition. Design the infrastructure to be scalable, allowing for easy expansion when needed.	Adapt the solution to the available resources, or consider leveraging cloud infrastructure to offload certain tasks and processes.
Design failures	POSSIBLE	INTOLERABLE	EXTREME	Iterative testing and future users' feedback. Development of a prototype.	Rework on design aspects. After redesigning, conduct additional rigorous testing to ensure the design meets the project's requirements.
Ethical considerations	IMPROBABLE	UNDESIRABLE	MEDIUM	Developing of a bias assessment discussion and an ethical review of the product.	Solution refinement to ensure fairness and analysis of the models using Explainable Artificial Intelligence.
Legal issues	POSSIBLE	INTOLERABLE	EXTREME	Maintain detailed records of all activities, decisions, and actions. Ensure all software and technologies used are properly licensed	Consult with legal counsel to understand the implications and best course of action.
Financial issues	IMPROBABLE	ACCEPTABLE	LOW	Development of an economic contingency plan. Periodically perform financial reviews to identify any discrepancies or issues.	Conduct a detailed assessment about the failure to understand its root cause. Identify areas where costs can be reduced without compromising the project's quality or objectives.
New technologies arise	PROBABLE	ACCEPTABLE	MEDIUM	Stay updated on technological trends. The Agile methodology allows for easy integration of new trends.	Conduct a thorough assessment of its benefits and challenges of the new technology. Determine if the new technology should be integrated into the current project or if it should replace existing technologies.
Communication Gaps	IMPROBABLE	ACCEPTABLE	LOW	Regular weekly meetings	Seek tutor mediation in case of ineffective communication
Team Dynamics and Leadership	IMPROBABLE	ACCEPTABLE	LOW	Early nomination of a responsible leader	Re-assessment of leadership and task distribution
Dataset and Topic Selection	POSSIBLE	UNDESIRABLE	LOW	Thorough pre-selection of data; obtaining tutor approval	Re-evaluate dataset or topic choice
Unexpected Team Changes	PROBABLE (already happened)	UNDESIRABLE	EXTREME	Regular team assessments	Re-structure team; redistribute tasks among remaining members

Figure 5: Table of risk evaluation and responses.

7 References

[1] Emmanouil Benetos et al. “Automatic music transcription: challenges and future directions”. In: *Journal of Intelligent Information Systems* 41 (2013), pp. 407–434.

[2] Emmanouil Benetos et al. “Automatic music transcription: An overview”. In: *IEEE Signal Processing Magazine* 36.1 (2018), pp. 20–30.

[3] Brian McFee et al. *librosa/librosa: 0.10.1*. Version 0.10.1. Aug. 2023.

[4] Simon Rouard, Francisco Massa, and Alexandre Défossez. “Hybrid Transformers for Music Source Separation”. In: *ICASSP 23*. 2023.

[5] Alexandre Défossez. “Hybrid Spectrogram and Waveform Source Separation”. In: *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*. 2021.

[6] Daniel Stoller, Sebastian Ewert, and Simon Dixon. *Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation*. 2018. arXiv: 1806.03185 [cs.SD].

[7] Simon Rouard, Francisco Massa, and Alexandre Défossez. “Hybrid Transformers for Music Source Separation”. In: *ICASSP 23*. 2023.

[8] Rachel M. Bittner et al. “A Lightweight Instrument-Agnostic Model for Polyphonic Note Transcription and Multipitch Estimation”. In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 781–785. DOI: 10.1109/ICASSP43922.2022.9746549.

[9] Matthias Geier. *Mido: MIDI Objects for Python*. https://github.com/mido/mido/. Accessed: October 26, 2023. 2023.

[10] Michael Scott Asato Cuthbert. *music21: A Toolkit for Computer-Aided Musical Analysis and Computational Musicology*. https://github.com/cuthbertLab/music21. Accessed: November 1, 2023. 2023.

[11] *MuseScore - The world's most popular notation app*. URL: https://musescore.org/es.

[12] Andrew Ng, Michael Jordan, and Yair Weiss. “On spectral clustering: Analysis and an algorithm”. In: *Advances in neural information processing systems* 14 (2001).

[13] Jong Wook Kim et al. *CREPE: A Convolutional Representation for Pitch Estimation*. 2018. arXiv: 1802.06182 [eess.AS].

[14] Jesse Engel et al. “DDSP: Differentiable digital signal processing”. In: *arXiv preprint arXiv:2001.04643* (2020).

[15] Wikipedia contributors. *ABC notation — Wikipedia, The Free Encyclopedia*. [Online; accessed 27-October-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=ABC_notation&oldid=1177515662.

[16] Armin Ronacher and contributors. *Flask Documentation*. Accessed: 2023-11-6. Pallets Projects. https://flask.palletsprojects.com/.

[17] Streamlit Team. *Streamlit Documentation*. Online. Accessed: 2023-11-6. URL: https://docs.streamlit.io.