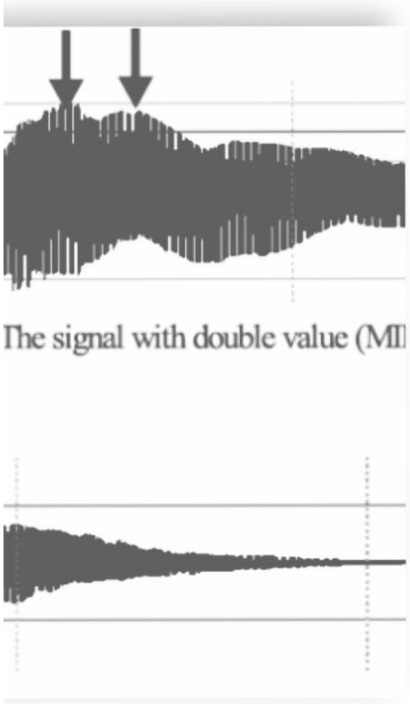
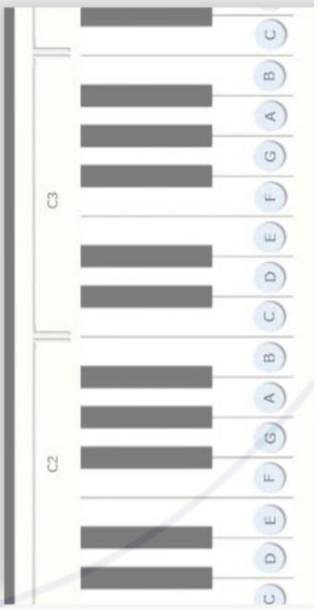
High-Level Design Report

Group 20





pianissimo

Team Members:
Melisa Kamacı
Tamer Alkım Tokuç
Nazem Yıldırım
Begüm Hatipoğlu

Supervisor: Yücel Çimtay
Jury Members: Emin Kuğu and Venera Adonava

Contents

High-Level Design Report

1. Introduction

This paper discusses the architectural concepts of decomposed subsystems in order to provide a high-level explanation of the architecture that will serve as the basis for Pianissimo's development process. The strategies for building the system are clearly specified, together with generating a creative solution to the functional design problem, in order to provide the essential elements to the current project description to represent an acceptable model for development.

1.1 Purpose of the System

Pianissimo's mission is to make it simple and accessible for musicians to get accurate, top-notch sheet music for solo piano music. The system's key characteristics include the following capabilities:

- Upload audio files of solo piano songs
- Transcribe the audio into sheet music using machine learning algorithms and signal processing techniques with user input
- Display the transcribed sheet music on the application
- Download the sheet music in various formats such as PDF or MIDI
- Manage the user account
- Customize and edit the generated sheet music, including adjusting the layout and formatting

The Pianissimo system is designed to be a comprehensive solo piano sheet music resource that provides a practical replacement for conventional music notation software. It is intended to be simple to use and open to musicians of all levels, with a focus on creating high-quality sheet music with the least amount of user work. To further improve its usability and accessibility for users, the Pianissimo system may be created for usage on the WEB and as a mobile application in the future. However, the project's first emphasis is on creating a computer application.

1.2 Design Goals

The following list of objectives serves as a guide for the Pianissimo system's design.

Accuracy: The generated sheet music should be as accurate as possible, capturing the nuances and complexities of the original audio with a high degree of fidelity. This will be accomplished by the system analyzing the audio and transcribing it into sheet music using sophisticated machine-learning algorithms and methodologies. The technology will also include tools for user input and revision to further assure the precision of the sheet music it generates.

User-friendliness: The system should be simple to use, with an intuitive user interface that requires little to no user interaction while transcription is being done. To do this, every step of the transcription process will be clearly explained to users in the system's design, which will also be simple and uncomplicated.

Customization: Users should have the option to edit and modify the generated sheet music to meet their own requirements and preferences. This should include the ability to change the style and formatting. The system will offer a variety of customizable and intuitive editing capabilities to make this possible.

Accessibility: The system should handle a wide range of audio file types and output options for the created sheet music. It should also be usable by musicians of all experience levels. The system will support a range of input and output formats and be flexible and adaptable in order to accomplish this.

Performance: The system should be capable of efficiently handling and processing big audio files with a respectable turnaround time for transcription. The system will be built with scalability and efficiency in mind, and will use optimized algorithms and data structures whenever possible, to ensure good performance.

The system will prioritize security, scalability, and maintainability in addition to these specific design objectives in order to guarantee that it can be used reliably and effectively by a large number of users. This includes techniques for managing and updating the system over time as well as safeguards to protect user data and prevent illegal access.

1.3 Definitions, Acronyms, and Abbreviations

- **Sheet music:** A transcription or printing of a musical composition that includes the lyrics, harmony, and melody (if applicable). Standard notation, which uses symbols and notation to express various pitches, rhythms, and other musical elements, is generally used to write sheet music.
- **Solo Piano:** A musical composition or performance that only features a piano player. Without any additional instruments or voices, solo piano music mainly consists of melodies and harmonies played on the piano.
- **Transcription:** The process of converting a piece of music from one form (such as audio) into another form (such as sheet music). Transcription is typically done manually by a musician or using specialized software.
- **Machine Learning:** A subfield of artificial intelligence that involves the use of algorithms and statistical models to enable computers to learn and improve their performance on a specific task without being explicitly programmed.
- **MIDI (Musical Instrument Digital Interface):** A standardized protocol for electronic musical instruments, computers, and other devices to communicate and exchange information. MIDI files are digital representations of music that can be played back using a MIDI-compatible device.
- **PDF (Portable Document Format):** A file format for representing documents in a platform-independent manner. PDF files can be viewed and printed on a wide range of devices and software applications.
- **Signal Processing:** The field of electrical engineering that involves the analysis and manipulation of signals, such as audio signals or images. Signal processing techniques are often used in the analysis and processing of audio files.
- **Audio File:** A digital representation of an audio recording, such as a song or piece of music. Audio files can be in various formats, including MP3, WAV, and AIFF, and can be played back using specialized software or hardware.
- **Note:** In music notation, a symbol representing a specific pitch and duration that is played or sung as part of a musical composition. Notes are typically represented on

sheet music using standard notation, with different symbols and notation used to represent different pitches, durations, and other musical elements.

1.4 Overview

Pianissimo is a computer program that converts solo piano audio recordings into sheet music. The main goal of Pianissimo is to make it easy for musicians to access accurate and professional-quality sheet music for their favorite piano pieces, without requiring extensive knowledge of music notation or the use of specialized software.

To create the sheet music, Pianissimo uses advanced machine learning algorithms and techniques to analyze the audio and identify musical features such as pitches, rhythms, and phrasing. The program then converts this information into a format that can be displayed as sheet music, using standard notation and symbols to represent different musical elements. In order to ensure the accuracy of the generated sheet music, Pianissimo also provides mechanisms for user input and correction.

After the sheet music has been generated, it can be viewed within the program and downloaded in various formats such as PDF or MIDI. Pianissimo also includes tools for users to customize and edit the sheet music, including adjusting its layout and formatting. This allows musicians to tailor the sheet music to their specific needs and preferences.

In addition to its core functionality, Pianissimo also includes features for user account management, including sign up, login, logout and password recovery. Users can create their own accounts and customize their preferences within the app, and their account information will be stored securely using a database system. This allows users to access their sheet music and preferences from any device, and makes it easy to keep track of multiple sheets and projects.

Overall, Pianissimo is designed to be a comprehensive and user-friendly resource for solo piano sheet music, providing an alternative to traditional music notation software. The program is intended to be accessible to musicians of all skill levels, and is designed to be efficient and effective in generating high-quality sheet music. In the future, Pianissimo may also be made available on the WEB and as a mobile app to increase its accessibility.

2. Current Software Architecture (If Any)

Since the current software architecture is not available, the planned software architecture is explained in the proposed system part.

3. Proposed Software Architecture

Software architecture is merely how a system is set up. This structure comprises all of the parts, how they work together, the setting in which they operate, and the design concepts that guided the development of the software. In many circumstances, it may also refer to how software will develop in the future. The software architecture of Pianissimo is described in every sub-title on this topic.

In addition, this section provides a detailed explanation of the system's subsystem decomposition and software architecture. First, classes and diagrams are used to describe how the system is divided into multiple layers. The system's hardware/software mapping is then provided, along with a component diagram that illustrates how the various components of our system will interact with the various hardware components. Following that, a section on persistent data management is provided to illustrate our database systems and objects. The section on access control and security will describe the restrictions and security methods we'll use in our system. The section on global software control discusses the overall process and how the software is managed. The boundary conditions section concludes by discussing our system's startup, terminalization, and failure situations.

3.1 Overview

The software architecture of Pianissimo is designed to be modular and flexible, allowing for easy maintenance and extension in the future as changes can be made to individual components without affecting the overall functioning of the system. The main components of the system are the user interface, audio processing, transcription, sheet music display, and user account management. These components work together to provide the core functionality of the Pianissimo system, which is to convert solo piano audio recordings into sheet music and provide a platform for users to manage their own accounts.

In addition to these main components, the Pianissimo system may also include additional features and functionality, such as support for different audio file formats, integration with other music notation software, and advanced tools for music analysis and editing. These features will be implemented as additional modules or subsystems within the overall software architecture, in order to maintain a clear separation of responsibilities and a modular design.

Overall, Pianissimo's architecture is intended to provide a robust and user-friendly platform for solo piano sheet music, combining advanced machine-learning techniques with an intuitive and customizable user interface.

3.2 Subsystem Decomposition

Pianosimmo's large-scale architecture adopts the **MVC (Model-View-Controller Architecture)** model. It will be in the form of a computer application interacting with a single user and will allow the user to use notes in PDF or MIDI format whose voice is converted into notes using machine learning algorithms and user input and audio processing techniques. It includes most of the data processing required to convert uploaded audio files to PDF, such as machine learning/AI recognition and signal detection of notes.

Our system is built on a Model-View-Controller architectural model. We have divided our system into three subsystems; PianissimoModel (Model), PianosimmoView (View), PianosimmoController (Controller). The PianosimmoModel subsystem is responsible for maintaining the domain information and the content of the entire Pianissimo view. The PianosimmoController subsystem is responsible for running controls, creating the application layout, controlling the rules, and executing the processing throughout the execution of pianissimo and sound transform to (PDF/MIDI) . It depends on the PianissimoModel. The PianosimmoView subsystem is responsible for the execution interface and for managing the order of interactions with the user. It is up to the Pianissimo Controller to provide information to continue updating the app's view.

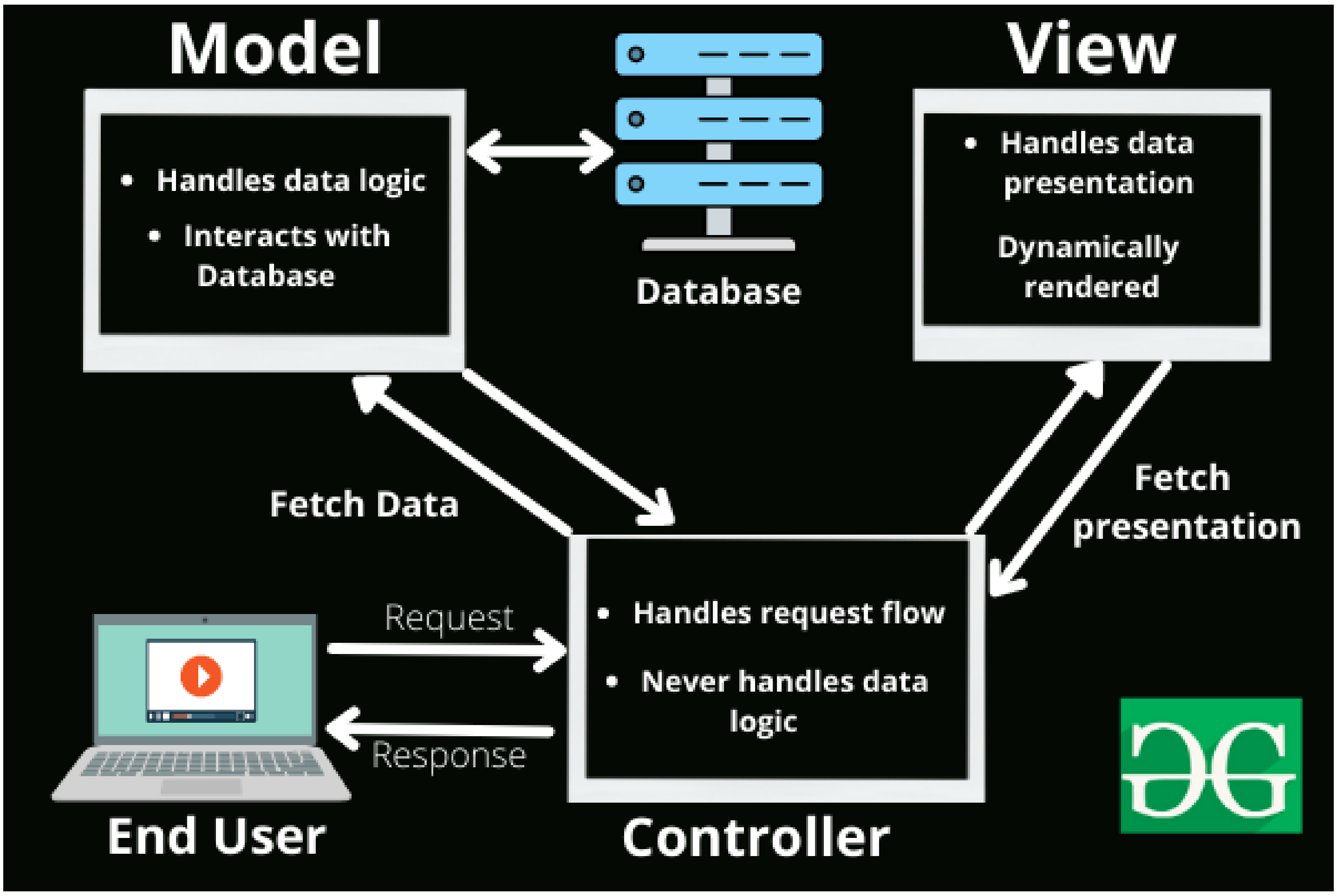


Figure: Comprehensive visual of MVC Architecture.

3.3 Hardware/Software Mapping

We can build the project's hardware and software mapping on the communication between the user's device and the input/output devices for the Pianissimo computer application. As seen in the illustration, the user device's input controller is in charge of handling each form of input that the computer application accepts and sends to the keyboard or mouse on the IO Device. The user can mark with ease thanks to the UI in the app.

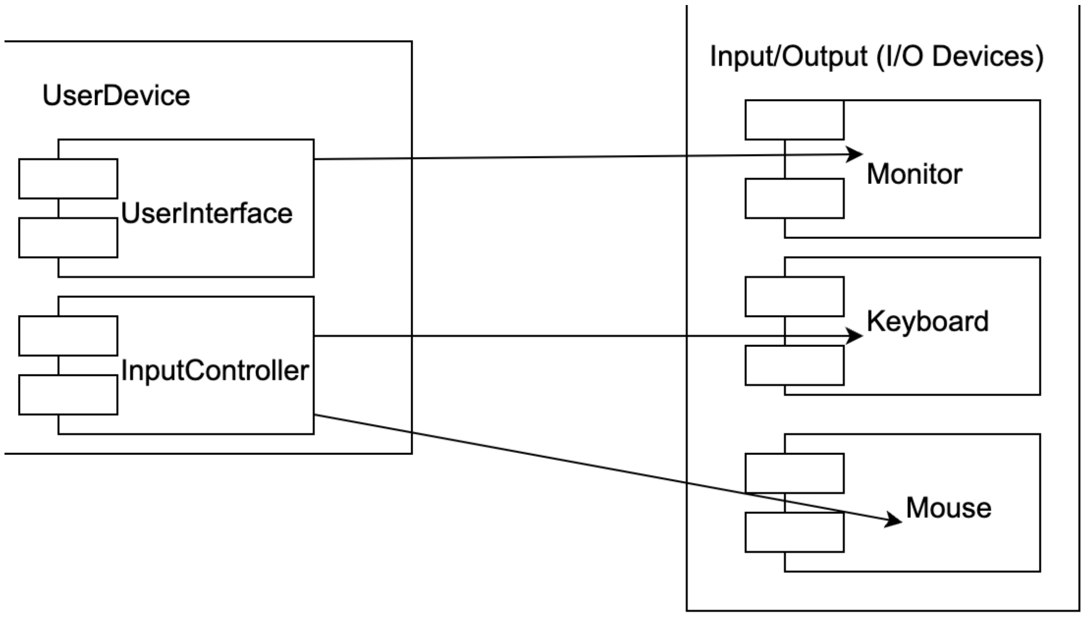


Figure: Diagram showing the components used in Hardware/Software Mapping communications between subsystems.

3.4 Persistent Data Management

Persistent data includes both user-taken notes (in PDF or MIDI format) and information held in the database. When a user saves an audio file, the note is saved locally in the database remotely if the recorded audio file was uploaded via the page. Audio files are saved in PDF or MIDI format. Notes can remain in the database as long as their total size does not exceed the limit provided by the PianissimoModel (database). As stated, a maximum of 50 projects will be stored in the system and a maximum of 100 users will be able to log in at the same time. Also, information about users and notes will be kept in an SQLite database. This information is also used for services such as authentication. Out-of-the-box trained models are also kept on the database side and used whenever a module that uses them is called. Notes will be built so that it can use local storage to store data about user data like Ubuntu 18.10. A database system is used to store notes, authentication data, and other data about users. SQLite, a very reliable SQL database engine, was chosen as the database system.

3.5 Access Control and Security

Users must create an account to use Pianissimo. For authentication services, each user must choose a username and password and email address. The user can register with Pianissimo to create a profile. The user can log in to view their profile and previously converted notes, if any. A user can log out directly after logging into the pianissimo application page. Users can upload notes before logging out.

The user can download the output of the notation file or print it without downloading it. The user can retrieve and download an existing music file from the user's database. The user can remove a selected note file from their profile that they have converted before. Users will be asked to create strong passwords when creating their account. In addition, users will have accepted the terms and conditions to be established in accordance with local regulations such as general data protection regulation KVKK. Additionally, data security is a crucial concern that needs to be rigorously maintained. We created our program to comply with the General Data Privacy Regulation (GDPR).

3.6 Global Software Control

Pianissimo has a software control with ***MVC Architecture.*** Applications are managed through event driven software control in response to user input. User selections in Pianissimo pass requests to the application while creating a transcript (PDF/MIDI) from the audio file. When the application processes the requests and returns output, the user is given the rendered result. Some of the data provided is not used by users. For example, if a user enters a valid email address and password, the application will receive an authentication token that will be used by the program to open the main menu screen. As a result, this event-based checking is also performed in other areas, including authentication, where a login request is made using the username and password that the user enters, and an authentication token is returned if the credentials are correct. After successful authentication, a request is made asking the user to list their notes. Another example would be a response to an information update request whenever a user changes a preference.

3.7 Boundary Conditions

Plantify will have three boundary conditions such as initialization, termination and failure.

3.7.1 Initialization

Since Pianissimo will be a computer application based application, users must have a device to download and use the application. That is, users need an internet connection and a computer to use the Pianissimo computer application. The application can be accessed via the application interface; (Control provided by Pianissimo Model). The following procedures are the same for both WEB application and Android/IOS (if android and WEB programming format target is accomplished later). If the user has an account registered in the system, the user can log in by entering the email and password. The data provided must match a registered combination in the database. If there is no match, the user cannot access the application and receives a message that their email address or password is incorrect. If the user does not have an account, a record as a user must be created (registered in the system). Account creation can be started with the corresponding button on the login screen. The user is reminded of the required username and password format (eg minimum password length) during both login and registration. If the login fails, the user returns to the login screen. Internet connection is not required during the authentication phase. When the user logs in to Pianissimo, he is presented with the conversion screens. It is shown in the appendix. In addition, with the forgot password option, it provides the opportunity to reset the password for users who forget their passwords.

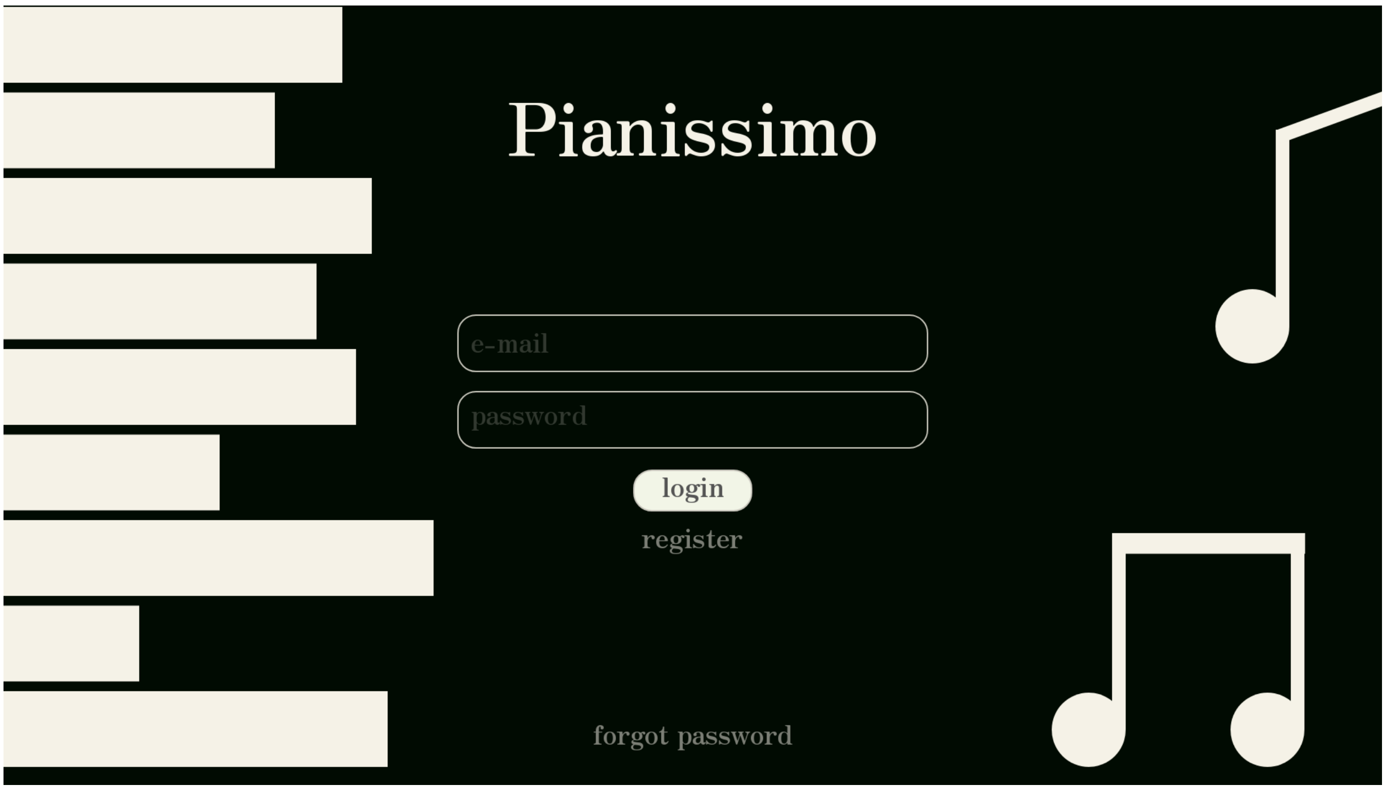


Figure: Login Page

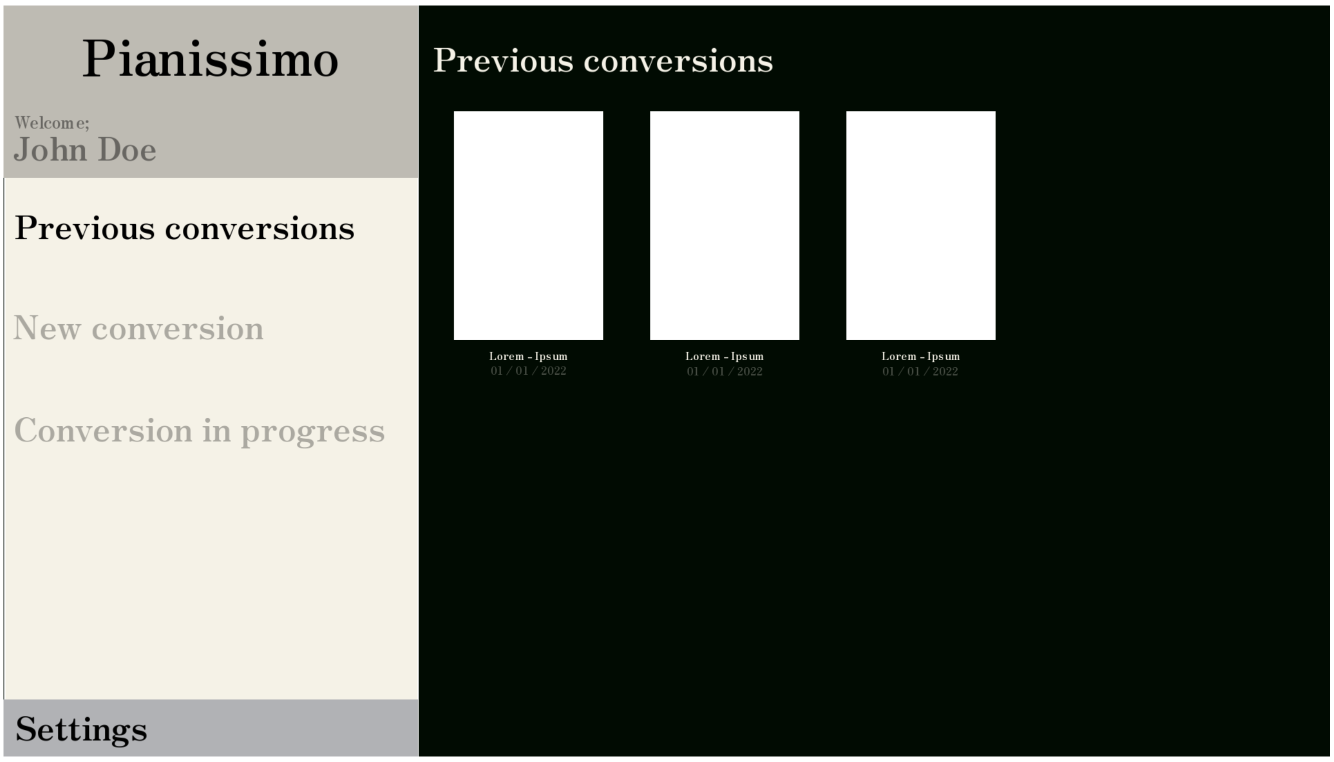


Figure: Main interface with previous conversions tab active

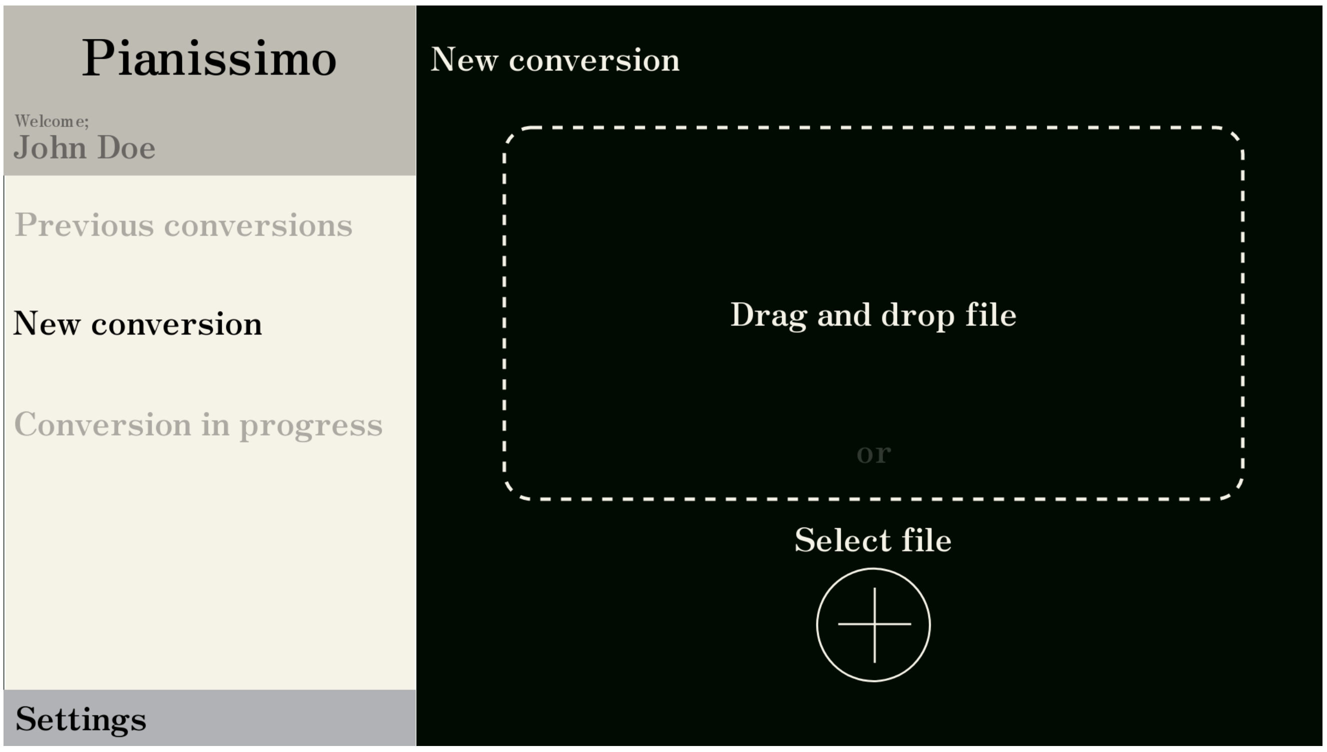


Figure: Main interface with new conversion tab active

3.7.2 Termination

Logging out (log out) in Pianissimo is simple. Users can exit the application at any time by closing the application with the buttons provided by the application page or the exit button on the home screen (from the settings tab at the bottom left). The user can log out of Pianissimo at any time, except waiting for the result of an operation on a note file in the application.

3.7.3 Failure

Since some features of Pianissimo require an internet connection, no internet connection can cause malfunctions (sometimes). Also, in some error situations, the user may need to close and reopen the application. In the event of a problem with Pianissimo, an attempt is made to save the relevant system data before it is terminated. Since the application has been closed, users must log in again. Since all completed transactions are recorded in the database, there will be no data loss if the application fails while the user is performing the operation. This incomplete action will not be saved, so the action needs to be done again after the user reopens the app. A stable internet connection is important when using the application, as it provides the connection between the Internet and the computer application. Pianissimo may fail if the user device loses the internet. Additionally, Major Errors in Pianissimo will be capped at a maximum of 0.25 errors/KLOC, including system failures, security breaches, and failure to produce output. It is stated that critical errors such as users with excessive or insufficient access will be limited to a maximum of 1 error/KLOC.

4. Subsystem Services

A subsystem is a service provider that performs one function or many functions, but does nothing until it is requested.

4.1 Authentication and Authorization

The authentication and authorization subsystem of the application is intended to guard the system from malicious or unauthorized behavior. User authentication needs the user to provide credentials, namely a username and a password. Authentication and authorization is included to make sure that only authorized users have access to the profiles that have been created. The system shall implement authentication and authorization methods in order to enable safe access to user profiles and private data.

4.2 Model-Data Access

Data access is an important subsystem of any software which enables users to get, update, create and delete data, as well as to store and retrieve data. Data access can be provided through a database, a file system, or through a web service. The data access subsystem in our application is mainly used to store and retrieve individual users' profile information. Data access should be secure and controlled to guarantee that it can only be accessed safely and securely by authorized individuals and that it cannot be compromised to other parties. Since accessing large amounts of data can take time, data access should also be efficient.

4.3 View-Presentation Layer

The subsystem in software that displays data to the user is the presentation layer. It is the topmost layer in the software architecture and is in charge of enabling user interactions as well as the graphical representation of data. The user interface, which includes data presentation, user input, and processing of user interactions, is handled in this layer. The presentation layer, or the graphical user interface (GUI) is in charge of creating an intuitive and engaging user experience. It must be developed to be efficient, responsive, and to provide a simple and reliable user experience.

4.4 Controllers

Following the MVC pattern, the general purposes of the controllers that will be created include responding to user requests, including input from and communications with the system from users, analyzing the information provided by the user and sending it to the model for analysis, obtaining the model output and sending it to the view so it can be seen, monitoring the system's performance and changing the model's parameters as needed and implementing the algorithms for signal processing and machine learning. The proposed controllers would generally implement the following functionalities:

4.4.1 Data Input Controller

Responsible for gathering data from the input sources and formatting the data for use in the machine learning models.

4.4.2 Model Controller

Responsible for developing the machine learning models and assessing their effectiveness.

4.4.3 Signal Processing Controller

Responsible for applying various signal processing techniques, such as filtering and Fourier Transform, to the data before it is fed into the machine learning models.

4.4.4 Conversion Controller

Responsible for the conversions from the processed signals to files containing sheet music which has been transcribed according to the universally accepted musical notations.

4.5 Security Layer

The security layer of a system provides defense against malicious attacks and data loss. Generally, security layers are divided into four separate tiers. The application layer is the top layer and it consists of security measures like encryption and authentication. System layer consists of system-level security features like firewalls, anti-virus software and similar security measures. The third layer is the network layer and it protects the network and any data that passes through it. Finally, the fourth layer is the data layer and it consists of methods like data loss prevention, encryption, and other data security procedures. In the making of the Pianissimo application, different security measures across all four of these layers shall be considered and implemented.

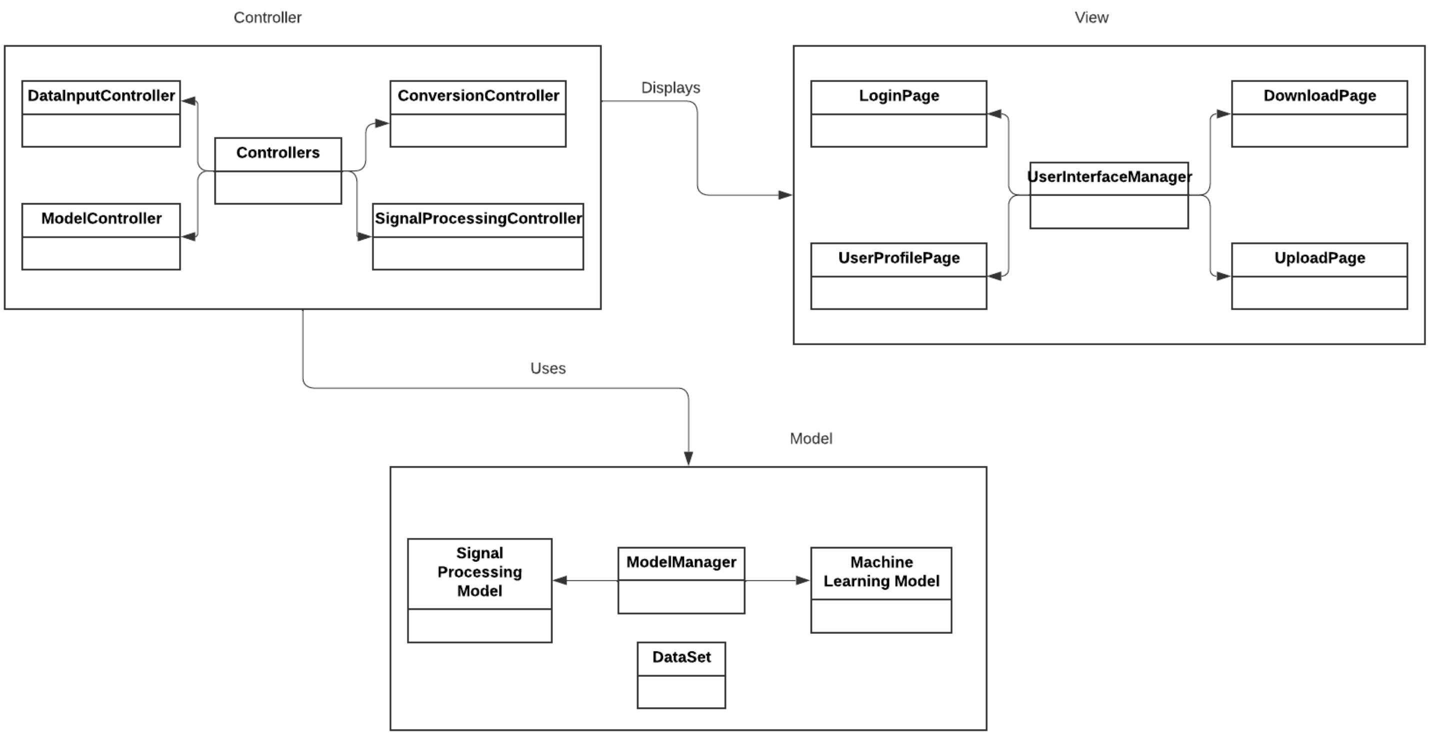


Figure: Subsystem Decomposition

5. Glossary

Pianissimo (noun): a passage marked to be performed very softly.

MIDI (Used in Section 1.1): Musical Instrument Digital Interface

PDF (Used in Section 1.1): Portable Document Format

WAV (Used in Section 1.3): Waveform Audio File Format

MP3 (Used in Section 1.3): MPEG-1 Audio Layer 3

AIFF (Used in Section 1.3): Audio Interchange File Format

UI (Used in Section 3.3): User Interface

IO (Used in Section 3.3): Input/Output

SQL (Used in Section 3.4): Structured Query Language

KLOC (Used in Section 3.7.3): Thousands (Kilos) of Lines of Code

KVKK (Used in Section 3.5): Personal Data Protection Authority

GDPR (Used in Section 3.5): General Data Privacy Regulation

6. References

[1]: Ohio State University Department Of Computer Science and Engineering, Systems Requirements Specification (SRS), 2006
https://web.cse.ohio-state.edu/~bair.41/616/ReqDoc_Outline.htm

[2]: Bilkent University Department Of Computer Engineering, High Level Design Report, 2018 https://ninoapp.github.io/4_HighLevelDesign.pdf

[3]: https://www.geeksforgeeks.org/mvc-framework-introduction/

[4]: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

[5]: https://crimsonpublishers.com/prsp/fulltext/PRSP.000505.php

[6]: https://www.ibm.com/docs/en/zos/2.3.0?topic=interface-what-is-subsystem

[7]: https://docs.osgi.org/specification/osgi.cmpn/7.0.0/service.subsystem.html

[8]:http://www.cs.fsu.edu/~myers/cop3331/notes/sysdesign.html