Bilkent University
Department of Computer Engineering

Senior Design Project
T2314
Rhythmus

Analysis and Requirement Report

21802067, Berk Baltacı, berk.baltaci@ug.bilkent.edu.tr
21802632, Cihan Can Kılıç, cihan.kilic@ug.bilkent.edu.tr
21600591, Azar Hasanaliyev, azer.hasanaliyev@ug.bilkent.edu.tr
21802977, Emir Yavuz, emiryavuz@ug.bilkent.edu.tr
21903708, Edip Kerem Tayhan,ekeremt@gmail.com
Fazlı Can
Erhan Dolak,Tağmaç Topal

<14/11/2022>

Contents

1 Introduction **4**

2 Proposed System **4**

2.1 Overview 4

2.2 Functional Requirements 5

2.2.1 Navigation Through User Interface 5

2.2.2 Music List 6

2.2.3 Parameters 6

2.2.4 Pattern Detection 6

2.2.5 Generating Similar Music 6

2.3 Non-functional Requirements 6

2.3.1 User Interface 6

2.3.2 Documentation 7

2.3.3 Performance Characteristics 7

2.3.4 Error Handling and Extreme Conditions 7

2.3.5 Quality Issues 7

2.4 Pseudo Requirements 7

2.5 System Models 8

2.5.1 Scenarios 8

2.5.2 Use-Case Model 9

2.5.3 Object and Class Model 12

2.5.4 Dynamic Models 14

2.5.5 User Interface 14

3 Other Analysis Elements **23**

3.1 Consideration of Various Factors in Engineering Design 23

3.2 Risks and Alternatives 24

3.3 Project Plan 24

3.4 Ensuring Proper Teamwork 28

3.5 Ethics and Professional Responsibilities 29

3.6 Planning for New Knowledge and Learning Strategies 29

4 Glossary 29

5 References 29

Analysis and Requirement Report

Project Short-Name: Rhythmus

1 Introduction

As the world starts to recognize the horsepower behind A.I. and machine learning, a way to integrate it with music has been on the mind of many programmers and musicians. Musicians and music enthusiasts can use our tool Rhythmus to create melodies and variations. This report aims to detail our project and the ways we will aim to achieve our goal.

Rhythmus will be fed large quantities of MIDI data of certain artists and thus offer a way to recreate variations of the input data while allowing the manipulation of certain attributes of the desired piece such as the key of the piece, the used scales, the tone of the piece, etc.

2 Proposed System

2.1 Overview

Rhythmus has a very convenient user interface that users can easily adapt to. Functionalities are self-explanatory with the help of graphical content such as icons. There is a help button for those who want to quickly learn about Rhythmus and ask their questions if any. Rhythmus is accessible from any internet browser including the ones on mobile devices. This is a great advantage of our project since it provides a chance to reach billions of people.

The project aims to help people who want to compose music for both professional and personal purposes. Many people get bored of listening to the same songs and wonder if there is a way to make songs just like their favorite ones. This is where we come to play. Our project makes it possible to determine patterns followed in a song and make a new song that contains similar patterns. Rhythmus can also inspire professional composers by giving them ideas about composing new music when they have a certain type of song in their minds.

As soon as the user signs up and logs in, he or she sees the home page of Rhythmus. There are two options in the home page for composing. The first one is to create a new project and the second one is to work on an existing project. When creating a project, the user needs to specify the parameters of the project such as musical instruments that are going to be used or beats per minute. All of these parameters can be changed later.

There can be several songs that the user wants to utilize in the composition of a new song. For this purpose, there is a music list in Rhythmus. The user can upload several songs for making another one. These songs are kept as long as the user desires. When the user logs out and logs in from another device, the list will still be available. The user chooses a song from the list and Rhythmus detects melodic patterns in the song along with the musical instruments played in it. Then, Rhythmus generates similar patterns and the user chooses one of them for composing his or her own song.

2.2 Functional Requirements

Begin with, the users must be able to use Rhythmus on their Android, iOS, Windows, or macOS devices.

2.2.1 Navigation Through User Interface

- The user interface must be fully navigable by left click, scroll up and scroll down actions.
- After the login page, the home page must occur.
- From the home page, there must be buttons for opening the start from scratch page and the open existing project page.
- When creating a new project, there must be the parameters page.
- After parameters are set, the website must open the layer page, where the new song will be composed

2.2.2 Music List

- The user must be able to create a list of music that he or she uploaded and keep it as long as he or she wants.
- There must be an option for choosing a music from the list to be used for making another music.
- The application must let the user delete any music from the list.

2.2.3 Parameters

- In the parameters section, there must be options for setting the following properties of the project:
    - musical instruments
    - beats per minute
    - scale
    - key
    - time signature

2.2.4 Pattern Detection

- The melodic patterns in the music chosen by the user must be detected.
- The musical instrument played in the chosen music must be detected separately.

2.2.5 Generating Similar Music

- Rhythmus must generate several similar versions of melodic patterns in the chosen song.
- There must be an option for choosing the instrument of the pattern generated.
- It must be possible to cut certain parts of the generated music to be used.

2.3 Non-functional Requirements

2.3.1 User Interface

- There must be a nice UI design that does not make the user get bored.
- All graphical components must be fully visible for everyone including the elderly.
- All functionalities must be self explanatory.

- The help page must have instructions as clear as possible.

2.3.2 Documentation

- Documentation for explaining functionalities to the users must be available.
- There must be source code documentation for future developers.

2.3.3 Performance Characteristics

- The program must perform following operations in no more than 3 seconds.
    - navigate between pages
    - upload a song to the list
    - delete a song from the list
    - edit the song(e.g. cut some part of it)
    - set parameters of the project
- The program must perform following operations in no more than 60 seconds.
    - detect melodic patterns in the music
    - detect musical instrument played in the music
    - generate similar patterns

2.3.4 Error Handling and Extreme Conditions

- The crash rate of the program must be less than %0.5.
- In case of a crash, the data of the user must not be lost.
- In case of a crash, the user must be informed via a notification.

2.3.5 Quality Issues

- The program must be available 7/24.
- Security of the users' data must be provided.
- The program must be error-free.
- In case of a failure, the application must be restartable in less than 5 seconds.
- Independent of the device, the program must work properly.

2.4 Pseudo Requirements

- Git/Github must be used for version control.
- Planning of the project must be done.
- Discussions with the supervisor must be done regularly.

2.5 System Models

2.5.1 Scenarios

- **A non-signed up user listens the demos**

  After the development of the project is completed we are going to create some pieces using different composers and parameters and upload them into a database. Then through pressing a button on the entry page users will be redirected to a page where the demos are listed anyone will be able to listen to them and have an idea of what the application is doing. Users don't have to sign-up or log-in in order to listen to these demos.

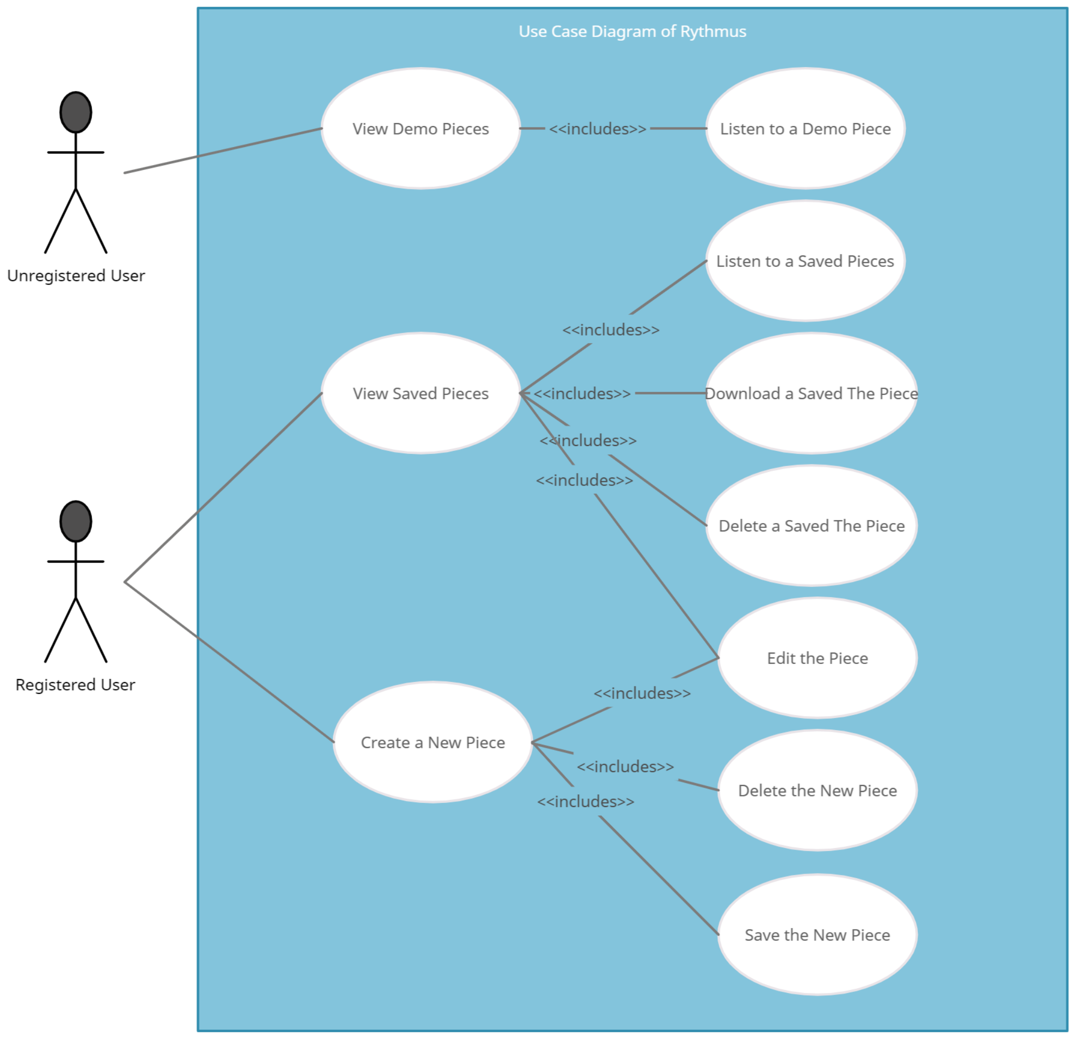
- **User listens to or modifies pieces they created earlier**

  The pieces users created will be saved on a database. After a user logs-in to his/her account they will be directed to a page where there will be two choices which are listening to their previously created pieces and creating a new piece. If they choose to listen to the previously created pieces they will be redirected to a page where their pieces are listed. Users can select any piece and listen to it or modify it through the layers interface. The layers interface is going to be a common music editing page where the user will be able to play around with the different layers and parts of the piece.

- **User creates a new piece**

  After logging-in if the user taps the "create a new piece" button they will be redirected to the creation page. This page will be designed to work in a progressive flow where the user won't be overwhelmed by seeing all of the options on one page. However, they will still be able to go back to previous steps if they want to change some preferences. First of all the composer choice screen will be opened. The users will choose the composers that they want their piece to be based on. The next step is piece selection where the user will choose pieces of the previously selected composers. These two steps will determine what the newly created piece will inspire from. Then the user moves onto the parameter selection page. In this page users will choose their preferences from the previously determined parameter types and after this step the application will create a new piece that is similar to the pieces chosen in terms of the musical theory and obeying the parameters given. Final step is the layer interface where the user will be able to manipulate and play around the piece manually. User has two options from this point, saving and deleting. Deleting will return the user to the main page. Saving will have three options; download the piece, save on the application, both.

2.5.2 Use-Case Model

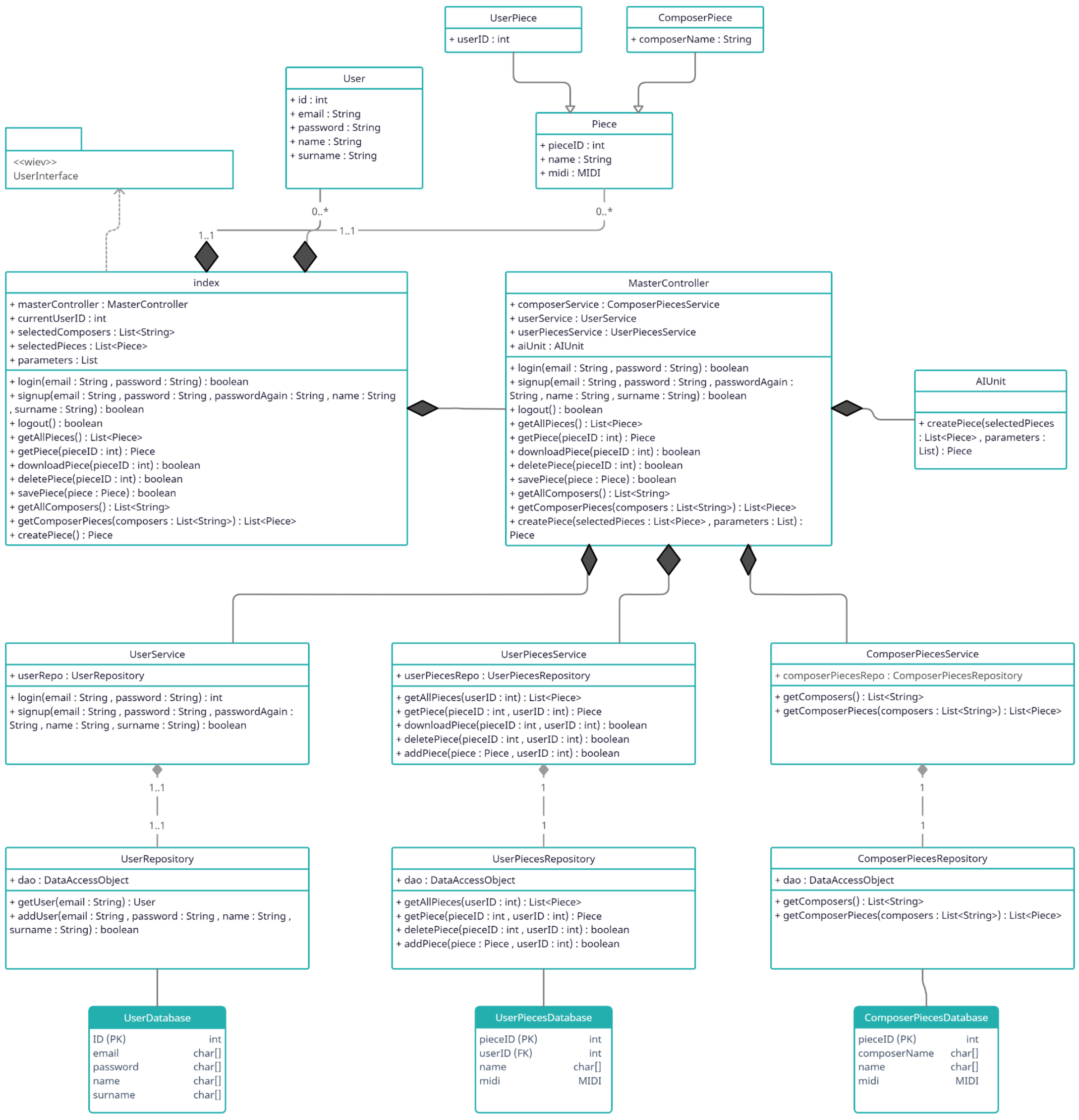


- **View Demo Pieces**
  - **Participating Actors:** Unregistered User.

    - **Flow of Events:**
        - A user clicks on the “Listen to Demo Pieces” button without logging in.
        - System redirects the user to a new page where demo pieces are listed.
        - Users can click on one and listen to it.
        - By using the back button the user can return to the home page.
    - **Entry Condition:** User clicks on the “Listen to Demo Pieces” button.
    - **Exit Condition:** User clicks on the back button.
    - **Quality Requirements:** Demo pieces should be well adjusted in order to get the user to like it and use the application.
- **View Saved Pieces**
    - **Participating Actors:** Registered User.
    - **Flow of Events:**
        - User clicks on the “Saved Pieces” button after logging in.
        - System redirects the user to a new page where saved pieces are listed.
        - Users can listen to a saved piece.
        - Users can download a piece by clicking on download.
        - Users can delete a piece by clicking on delete.
        - Users can edit a piece by clicking the edit button.
        - Users can return to the “User Page” with the back button.
    - **Entry Condition:** Uses clicks on the “Saved Pieces” button after logging in.
    - **Exit Condition:** User clicks on the back button.
    - **Quality Requirements:** All the saved pieces should be shown without any mistakes and bugs otherwise the user might lose a saved piece forever. Also the page should be well designed for the ease of use and for the user to find the desired piece easily.
- **Create a New Piece**
    - **Participating Actors:** Registered User.
    - **Flow of Events:**
        - User clicks on the “Create a New Piece” button after logging in.
        - System redirects the user to the composer selection screen.

    - After selecting the desired composers, the user clicks on the next button and gets redirected to the piece selection screen or the user can click the back button and return to the “User Page”.
    - After selecting the pieces of the previously chosen composers user can click; the next button to get redirected to the parameter choice screen, the back button to get redirected to the composer selection screen or the “Return to The Main Page” button to get redirected to the “User Page”.
    - After the user chooses the parameters they can use the back button, “Return to The Main Page” button or the next button to get redirected to the layers interface.
    - When the system redirects the user to the layers interface, the user can listen to, edit, save, download or delete the newly created piece.
- **Entry Condition:** User clicks on the “Create a New Piece” button after logging in.
- **Exit Condition:** User clicks on “Return to The Main Page” button on the pages it exists, the back button on the composer selection screen or the delete button on the layers page.
- **Quality Requirements:** As this is the main use case of the application everything should work flawlessly. The interface should be easy to use and shouldn’t exhaust the user. Users should be able to switch between steps fluently. The end product should be well and the save, download buttons should never create any bugs as if they do the created peace might be lost forever.

2.5.3 Object and Class Model


UserPiece
+ userID : int
ComposerPiece
+ composerName : String
User
+ id : int
+ email : String
+ password : String
+ name : String
+ surname : String
Piece
+ pieceID : int
+ name : String
+ midi : MIDI
<<wiev>>
UserInterface
0..*
1..1
1..1
0..*
index
+ masterController : MasterController
+ currentUserID : int
+ selectedComposers : List<String>
+ selectedPieces : List<Piece>
+ parameters : List
+ login(email : String , password : String) : boolean
+ signup(email : String , password : String , passwordAgain : String , name : String , surname : String) : boolean
+ logout() : boolean
+ getAllPieces() : List<Piece>
+ getPiece(pieceID : int) : Piece
+ downloadPiece(pieceID : int) : boolean
+ deletePiece(pieceID : int) : boolean
+ savePiece(piece : Piece) : boolean
+ getAllComposers() : List<String>
+ getComposerPieces(composers : List<String>) : List<Piece>
+ createPiece() : Piece
MasterController
+ composerService : ComposerPiecesService
+ userService : UserService
+ userPiecesService : UserPiecesService
+ aiUnit : AIUnit
+ login(email : String , password : String) : boolean
+ signup(email : String , password : String , passwordAgain : String , name : String , surname : String) : boolean
+ logout() : boolean
+ getAllPieces() : List<Piece>
+ getPiece(pieceID : int) : Piece
+ downloadPiece(pieceID : int) : boolean
+ deletePiece(pieceID : int) : boolean
+ savePiece(piece : Piece) : boolean
+ getAllComposers() : List<String>
+ getComposerPieces(composers : List<String>) : List<Piece>
+ createPiece(selectedPieces : List<Piece> , parameters : List) : Piece
AIUnit
+ createPiece(selectedPieces : List<Piece> , parameters : List) : Piece
UserService
+ userRepo : UserRepository
+ login(email : String , password : String) : int
+ signup(email : String , password : String , passwordAgain : String , name : String , surname : String) : boolean
UserPiecesService
+ userPiecesRepo : UserPiecesRepository
+ getAllPieces(userID : int) : List<Piece>
+ getPiece(pieceID : int , userID : int) : Piece
+ downloadPiece(pieceID : int , userID : int) : boolean
+ deletePiece(pieceID : int , userID : int) : boolean
+ addPiece(piece : Piece , userID : int) : boolean
ComposerPiecesService
+ composerPiecesRepo : ComposerPiecesRepository
+ getComposers() : List<String>
+ getComposerPieces(composers : List<String>) : List<Piece>
1..1
1..1
1
1
1
1
UserRepository
+ dao : DataAccessObject
+ getUser(email : String) : User
+ addUser(email : String , password : String , name : String , surname : String) : boolean
UserPiecesRepository
+ dao : DataAccessObject
+ getAllPieces(userID : int) : List<Piece>
+ getPiece(pieceID : int , userID : int) : Piece
+ deletePiece(pieceID : int , userID : int) : boolean
+ addPiece(piece : Piece , userID : int) : boolean
ComposerPiecesRepository
+ dao : DataAccessObject
+ getComposers() : List<String>
+ getComposerPieces(composers : List<String>) : List<Piece>
UserDatabase
ID (PK) int
email char[]
password char[]
name char[]
surname char[]
UserPiecesDatabase
pieceID (PK) int
userID (FK) int
name char[]
midi MIDI
ComposerPiecesDatabase
pieceID (PK) int
composerName char[]
name char[]
midi MIDI

- **Piece**

  This is an entity object which stores the required information for musical pieces which are the piece's id, name and MIDI file. It has two child classes UserPiece and ComposerPiece.

  - **UserPiece**

    UserPiece stores userID in addition to the Piece class.

  - **ComposerPiece**

ComposerPiece stores composerName in addition to the Piece class.

- **User**

  This is an entity object as well. It stores the required information for a system user which are id, email, password, name and surname.

- **Index**

  Index class is the entry point of the application. All of the user commands that came through the interface will be sent to and handled by this class. Index will use an instance of the MasterController class to forward the requests of the user to the lower layers of the system and to pass the information coming from the lower layers to the user.

- **MasterController**

  This class will only be used by the Index class. MasterController will be working as the handler of all services of the system. It has an instance of ComposerPiecesService, UserService, UserPiecesService and AIUnit. When MasterController receives a request from the index it is going to pass this request to the relevant service and forward the output to the index after doing some operations on the data if needed.

- **UserService**

  UserService will receive login or signup requests from the MasterController and forward it to the UserRepository through an instance of it. While logging in credential validations are going to be done here with the raw data received from the UserRepository. So the services will be the center of data operations, manipulations and validations. Most of the work will be done here in terms of user communicating with the database.

- **UserPiecesService**

  UserPiecesService will be the operation point between querying the UserPiecesDatabase through an instance of the UserPiecesRepository and forwarding the data to the MasterController. As pointed out above, services are the operation centers.

- **ComposerPiecesService**

  ComposerPiecesService will be the operation center between MasterController and ComposerPiecesRepository. It is going to use an instance of the ComposerPiecesRepository.

- **AIUnit**

  AIUnit will be the class that generates the piece with respect to the chosen composer pieces and the parameters.

- **UserRepository**

  This class will use a data access object in order to query the UserDatabase with the specifications received from the UserService and send the raw data to the UserService.

- **UserPiecesRepository**

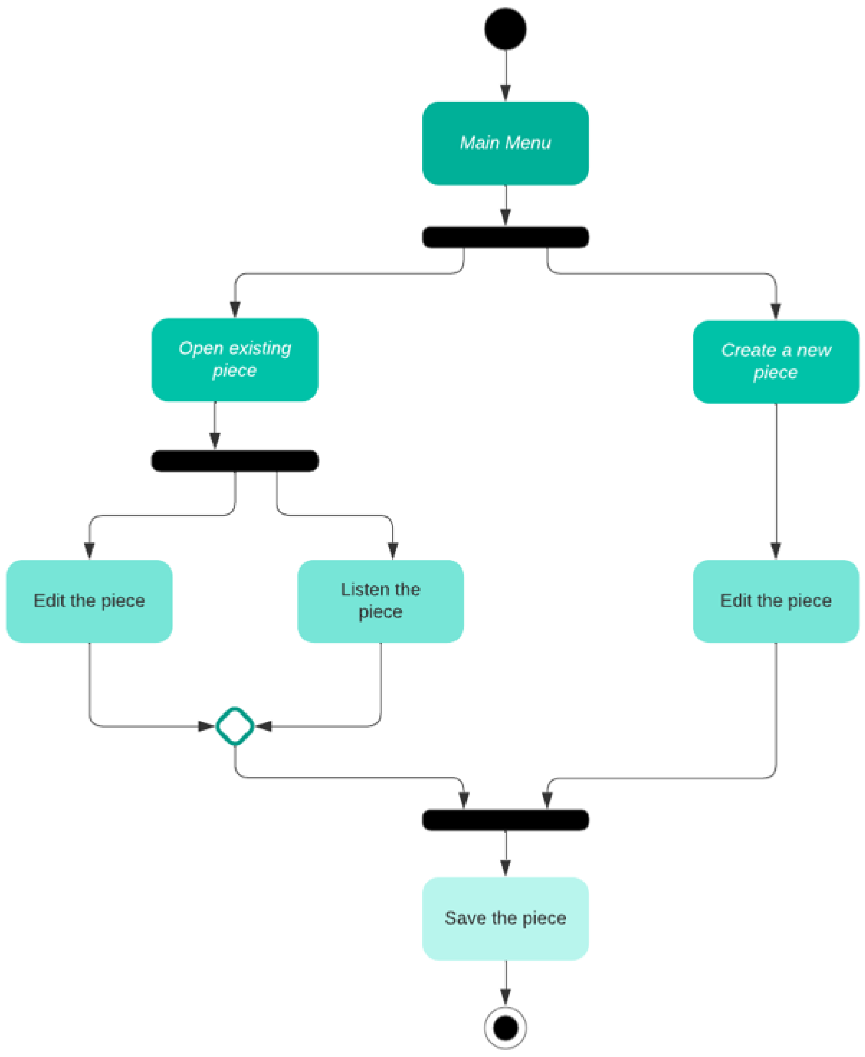
  This class will use a data access object in order to query the UserPiecesDatabase with the specifications received from the UserPiecesService and send the raw data to the UserPiecesService.

- **ComposerPiecesRepository**

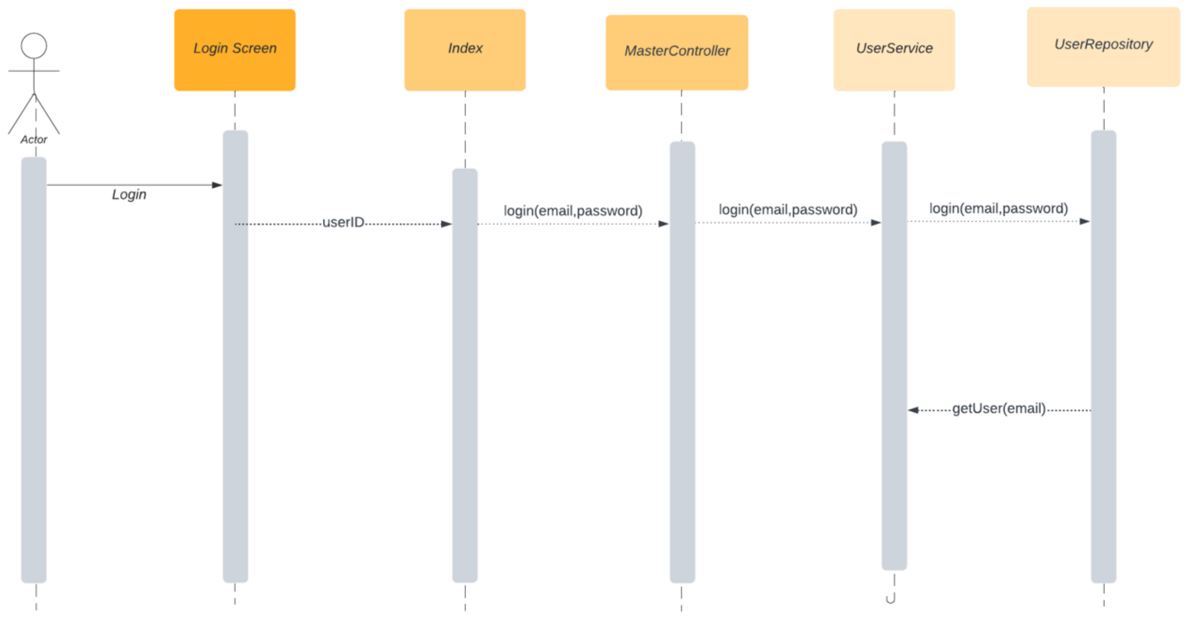
  This class will use a data access object in order to query the ComposerPiecesDatabase with the specifications received from the ComposerPiecesService and send the raw data to the ComposerPiecesService.

2.5.4 Dynamic Models

Activity Diagram

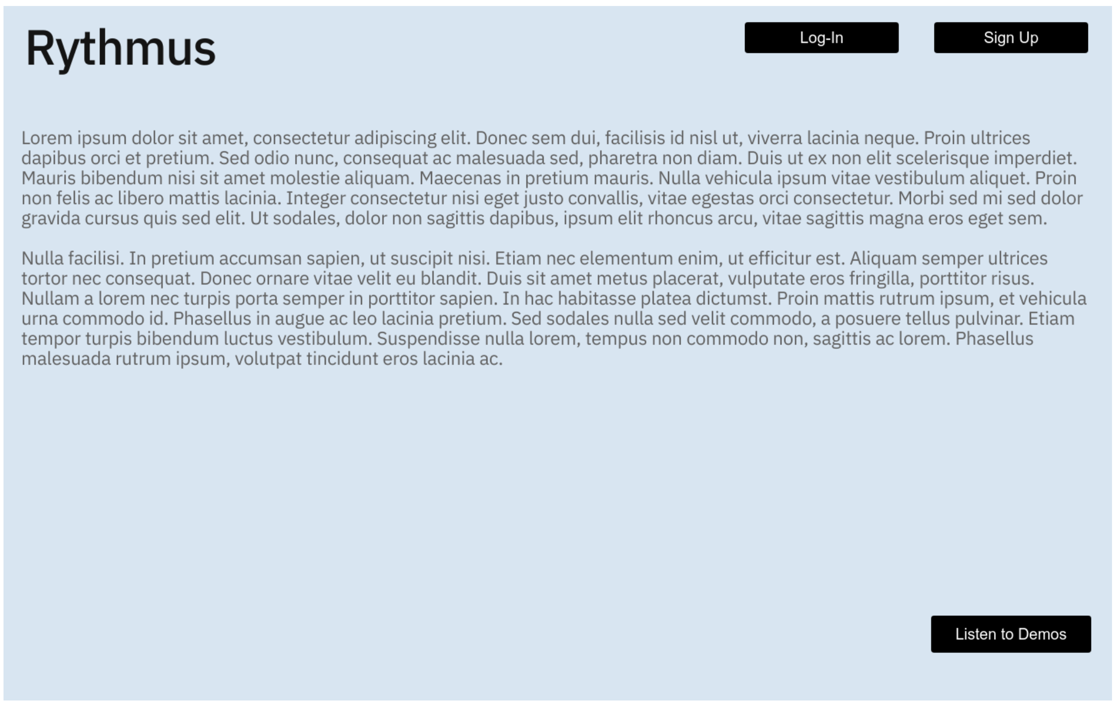


Sequence Diagram



2.5.5 User Interface

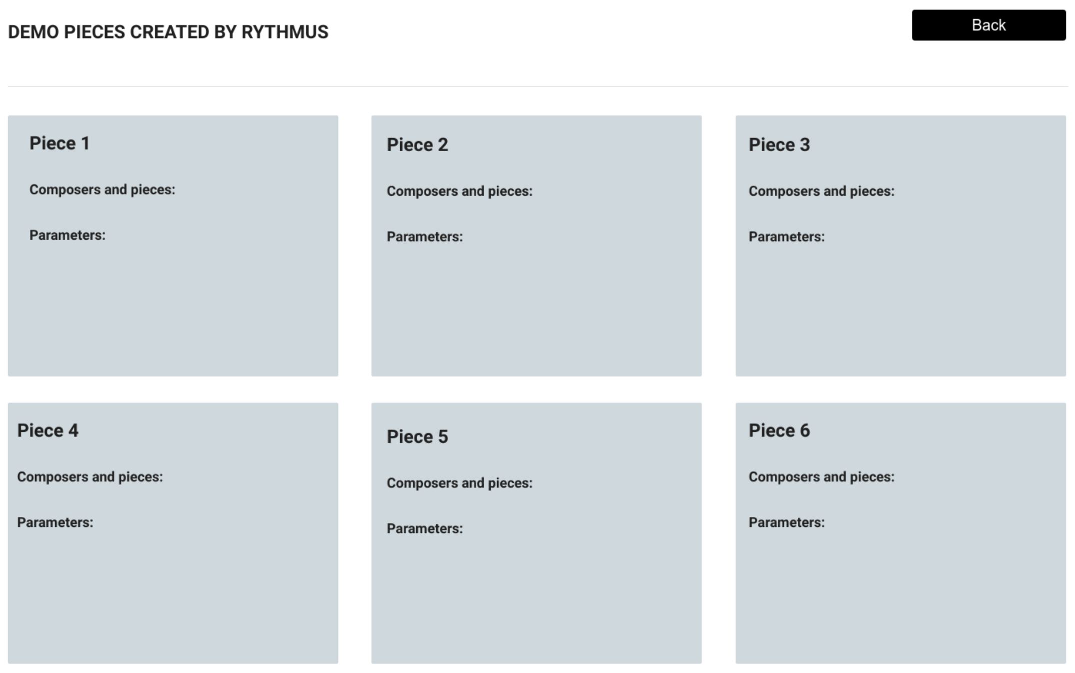
Despite the fact that Rhythmus is an application that uses music theory in order to create pieces that sound nice and familiar with the chosen composers and pieces, we don't want to overwhelm the users with complex technicalities. In order to accomplish that, we are trying to keep the interface as simple as possible. The main goal while designing the interface was to make it possible for someone who doesn't know anything about music theory or composing, to use Rhythmus and create their original A.I. generated pieces.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec sem dui, facilisis id nisl ut, viverra lacinia neque. Proin ultrices dapibus orci et pretium. Sed odio nunc, consequat ac malesuada sed, pharetra non diam. Duis ut ex non elit scelerisque imperdiet. Mauris bibendum nisi sit amet molestie aliquam. Maecenas in pretium mauris. Nulla vehicula ipsum vitae vestibulum aliquet. Proin non felis ac libero mattis lacinia. Integer consectetur nisi eget justo convallis, vitae egestas orci consectetur. Morbi sed mi sed dolor gravida cursus quis sed elit. Ut sodales, dolor non sagittis dapibus, ipsum elit rhoncus arcu, vitae sagittis magna eros eget sem.

Nulla facilisi. In pretium accumsan sapien, ut suscipit nisi. Etiam nec elementum enim, ut efficitur est. Aliquam semper ultrices tortor nec consequat. Donec ornare vitae velit eu blandit. Duis sit amet metus placerat, vulputate eros fringilla, porttitor risus. Nullam a lorem nec turpis porta semper in porttitor sapien. In hac habitasse platea dictumst. Proin mattis rutrum ipsum, et vehicula urna commodo id. Phasellus in augue ac leo lacinia pretium. Sed sodales nulla sed velit commodo, a posuere tellus pulvinar. Etiam tempor turpis bibendum luctus vestibulum. Suspendisse nulla lorem, tempus non commodo non, sagittis ac lorem. Phasellus malesuada rutrum ipsum, volutpat tincidunt eros lacinia ac.

This is the main page where all users first see. With the buttons on the top right corner, labeled “Log-In '' and “Sign-Up”, users can navigate to log-in and sign-up pages. By pressing the button on the bottom right corner, labeled “Listen to Demos' ', a non-signed-up user can navigate to a page where some previously generated pieces are demonstrated. So any user without signing-up can go to this page and listen to some of the A.I. generated pieces and get a better understanding of what Rhythmus is doing and what it's capable of. Also on this page users will see some information about the application itself and the developers in the body paragraphs filled with “Lorem İpsum '' text.


DEMO PIECES CREATED BY RYTHMUS

Back

Piece 1

Composers and pieces:

Parameters:

Piece 2

Composers and pieces:

Parameters:

Piece 3

Composers and pieces:

Parameters:

Piece 4

Composers and pieces:

Parameters:

Piece 5

Composers and pieces:

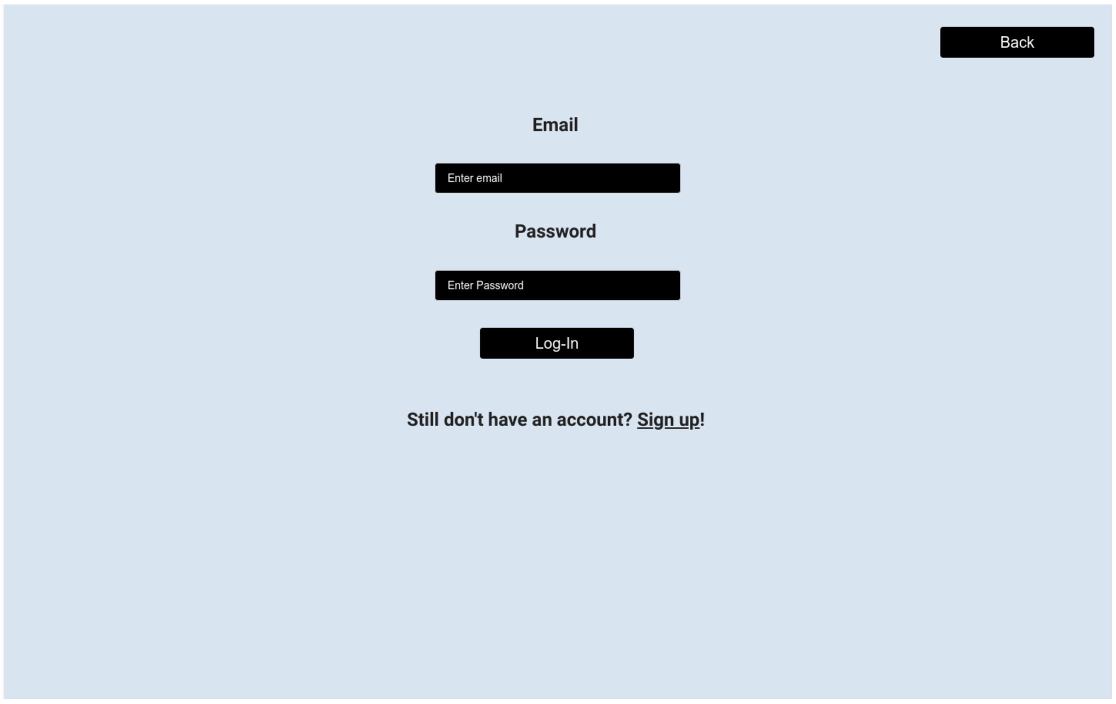
Parameters:

Piece 6

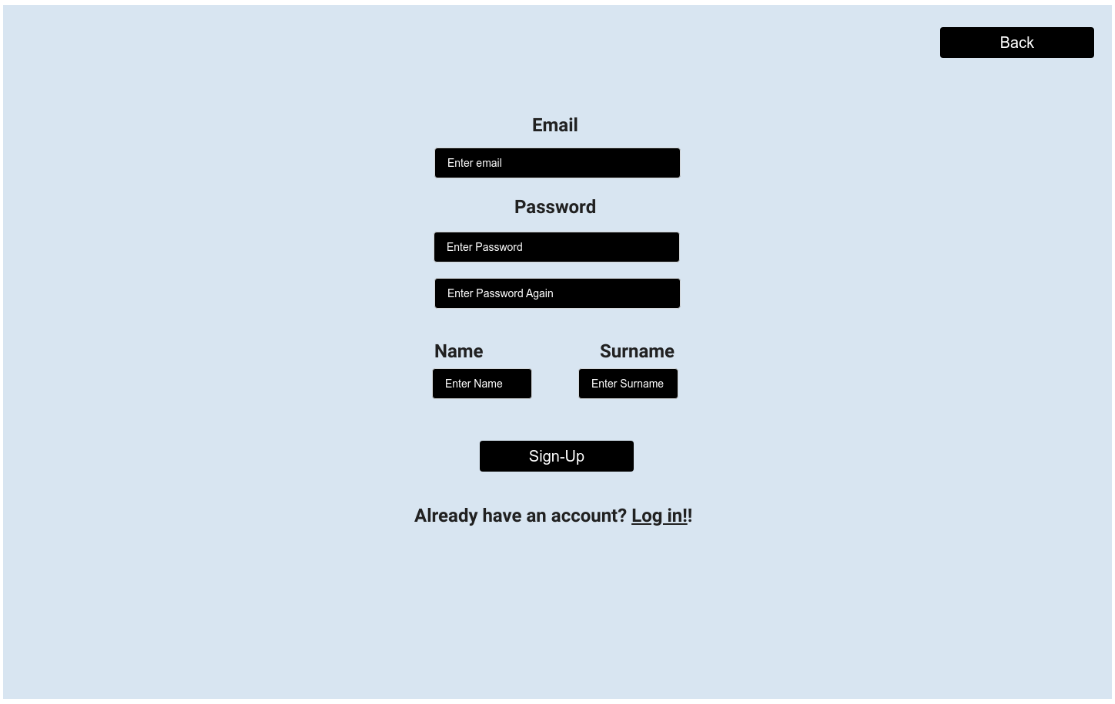
Composers and pieces:

Parameters:

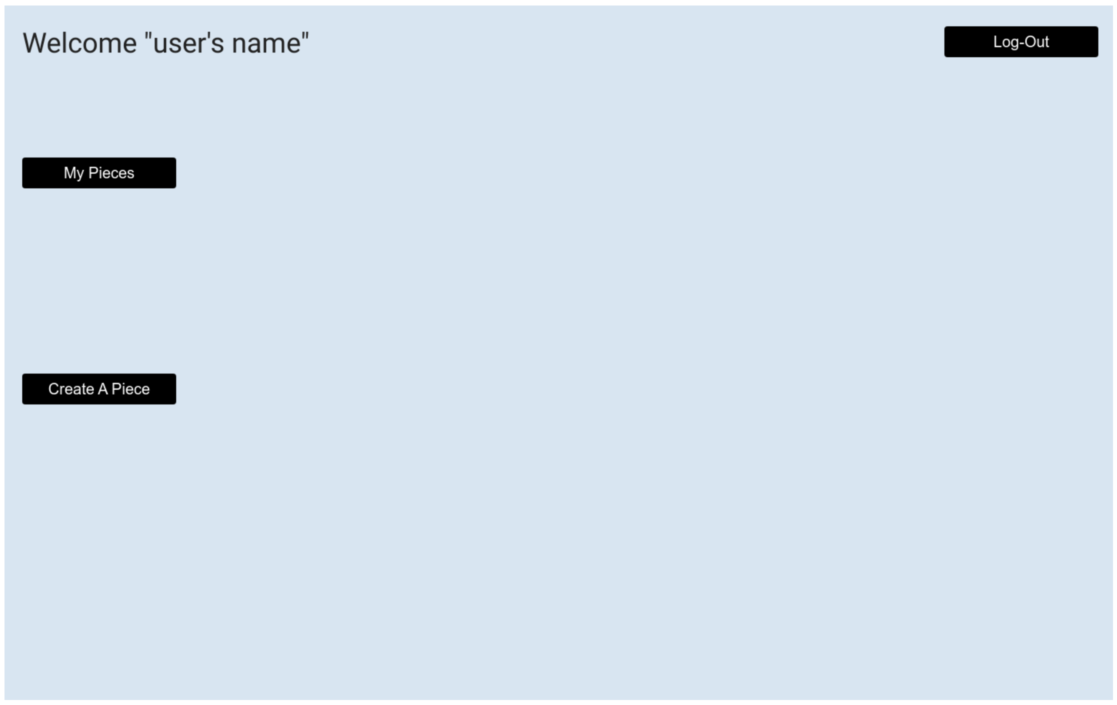
In this page user sees the previously generated example pieces listed. Also they can see which composers, songs and parameters were selected while generating these pieces. With the button on the top right corner, the user can navigate back to the “Main Screen”.



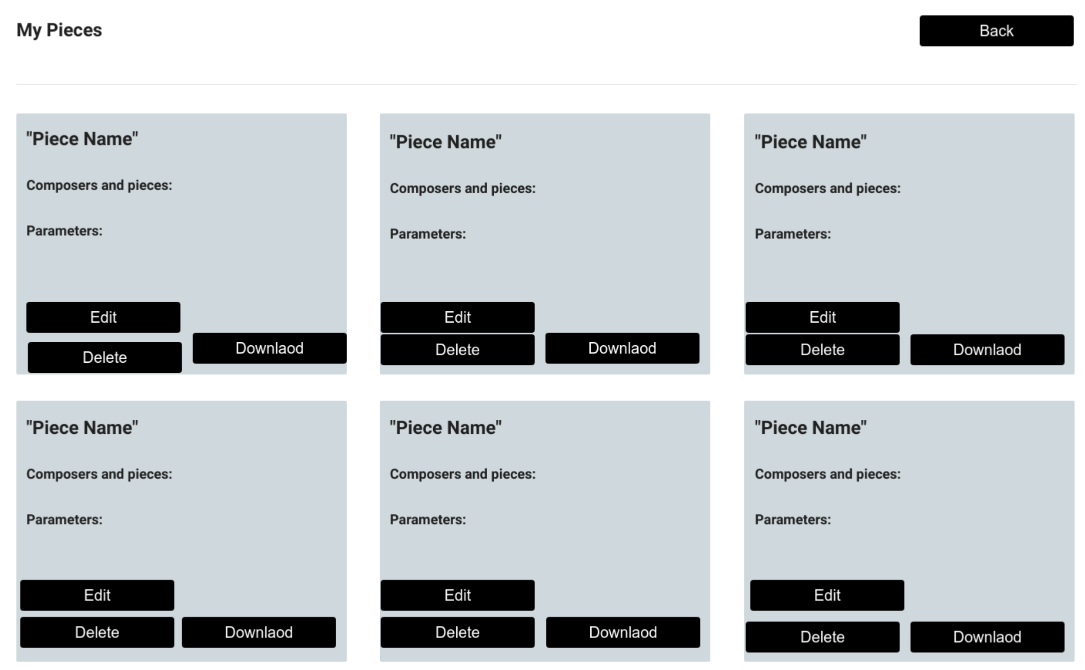
This is the log-in page. Users can perform the login process by writing their email and password in the text fields labeled as “Enter email” and “Enter Password” and clicking the “Log-In” button. Also if a user enters this page without having an account they navigate to the “Sign-up Page” by clicking on the underlined “Sign up” text. “Back” button navigates the user to the ”Main Page”.



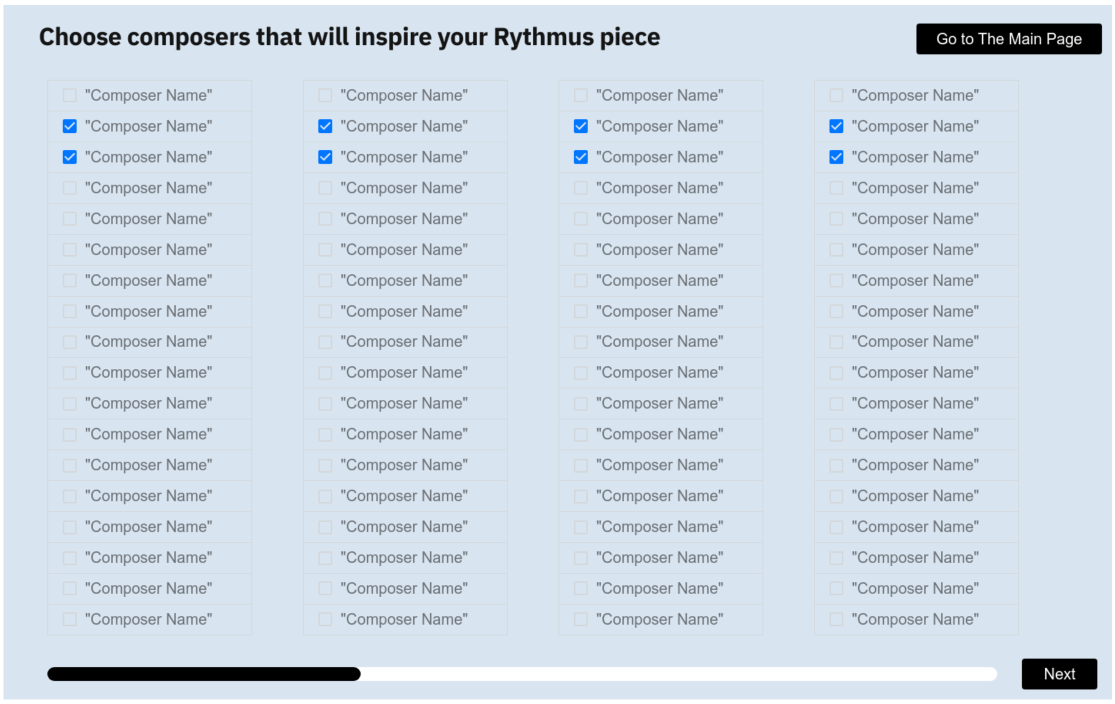
This page is for signing-up to the application. Users enter their email and password to the text fields labeled as “Enter email”, “Enter Password” and “Enter Password Again”. Then they are going to fill in the name and surname field and click on the “Sign-Up” button. “Back” button navigates the user to the ”Main Page”.



This is the page that the users are navigated to after successfully logging-in. On the top left corner with the “Log-Out” button a user can log-out of his/her account and be navigated to the “Main Page”. With the button labeled “My Pieces” users can go to the “My Pieces Screen” and with the “Create A Piece” button below that user can start the process of creating a new piece.



In this page users can see the pieces that they generated and saved before. Every item on the list will include piece name information, names of the composers and pieces and the parameters chosen while generating the piece. There are three buttons on each list item labeled “Delete”, “Download” and “Edit”. Delete Button will delete an item from the My Pieces section, download Button will download the piece to the computer and the edit button will redirect the user to the layers page to edit the piece. With the button on the top right corner labeled “Back” users can navigate to the “User Page”.



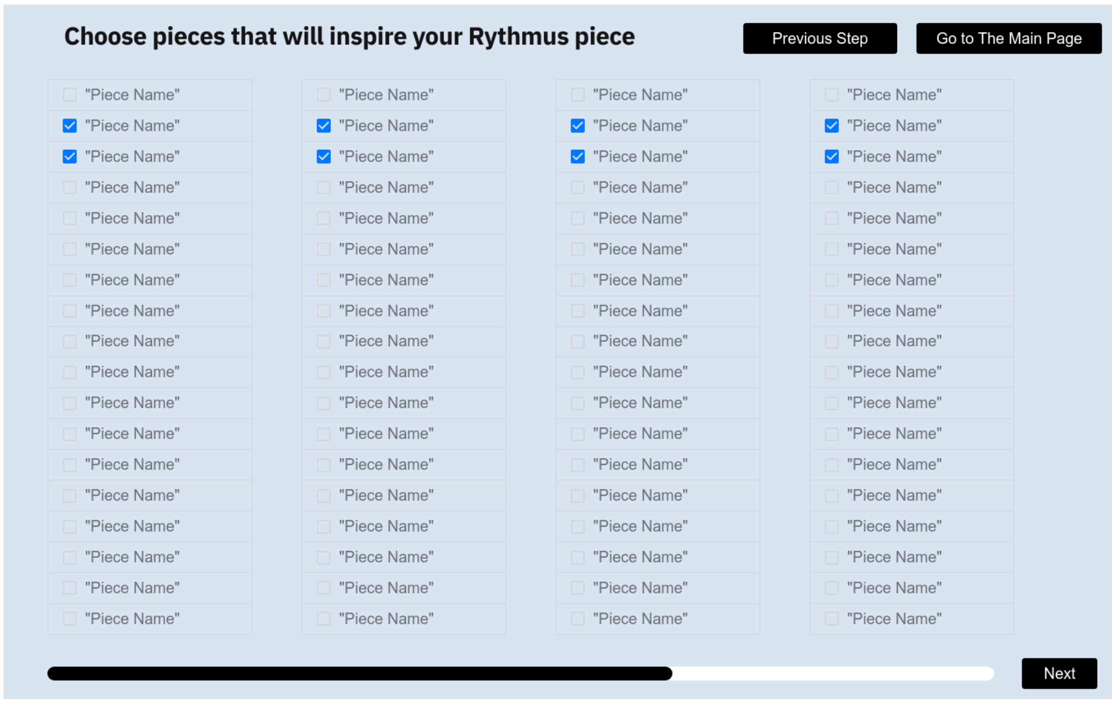
Choose composers that will inspire your Rythmus piece

Go to The Main Page

☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☑ "Composer Name"	☑ "Composer Name"	☑ "Composer Name"	☑ "Composer Name"
☑ "Composer Name"	☑ "Composer Name"	☑ "Composer Name"	☑ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"
☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"	☐ "Composer Name"

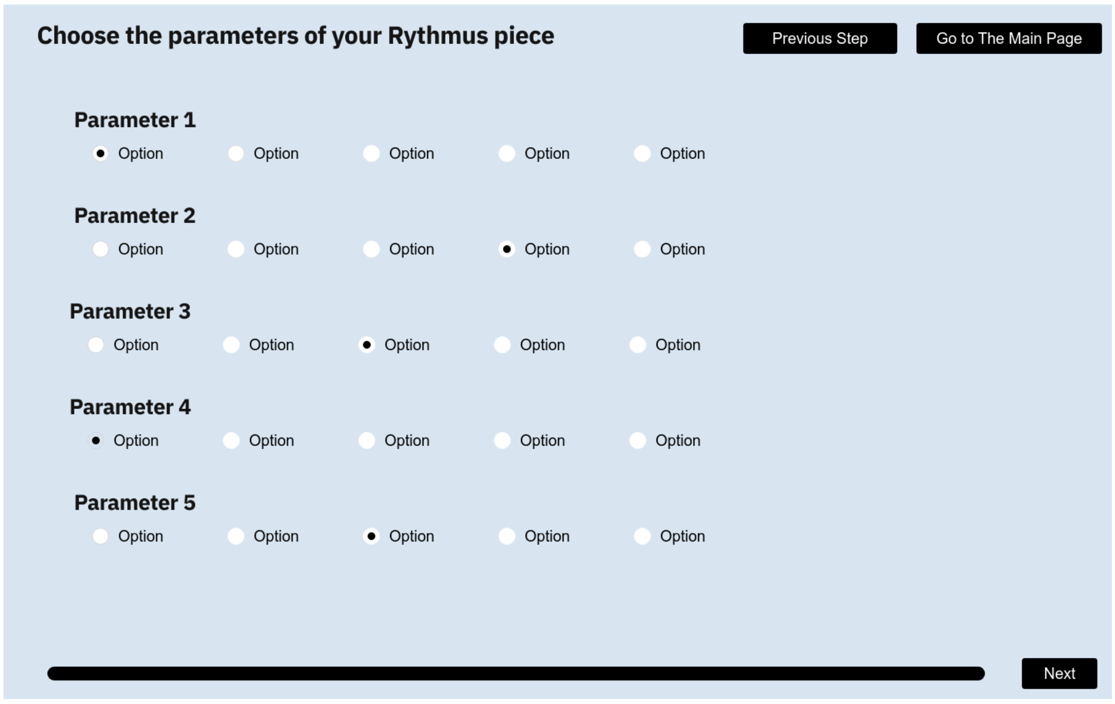
Next

This is the page that the user sees after pressing the “Create A Piece” button on the “User Page”. This is the first step of creating a piece where the user ticks the desired composers on the list. After they are done with selecting the composers they can go to the next step which is choosing pieces with the button on the bottom right corner labeled “Next”. If they want to return to the “User Page” they can do this by clicking on the button that is on the top right corner labeled “Go to The Main Page”. Also the progress bar on the bottom shows how much of the steps are done which will progress on the next steps.



☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
✓ "Piece Name"	✓ "Piece Name"	✓ "Piece Name"	✓ "Piece Name"
✓ "Piece Name"	✓ "Piece Name"	✓ "Piece Name"	✓ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"
☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"	☐ "Piece Name"

This page is the second step of generating a piece. Pieces of the previously selected composers are displayed on the screen and the user can choose any of these pieces by clicking on the box next to the name. If the user wants to change some preferences on the previous step they can navigate to the “Choose Composer” page by clicking on the “Previous Step” button on the top right corner. Also the user can abort the whole process and return to the “User Page” by clicking the button on the top right corner labeled “Go to The Main Page”. After the user is satisfied with his/her choices of pieces they can progress by clicking on the “Next” button on the bottom right corner.



Choose the parameters of your Rythmus piece

Parameter 1
[✓ Option] Option Option Option Option

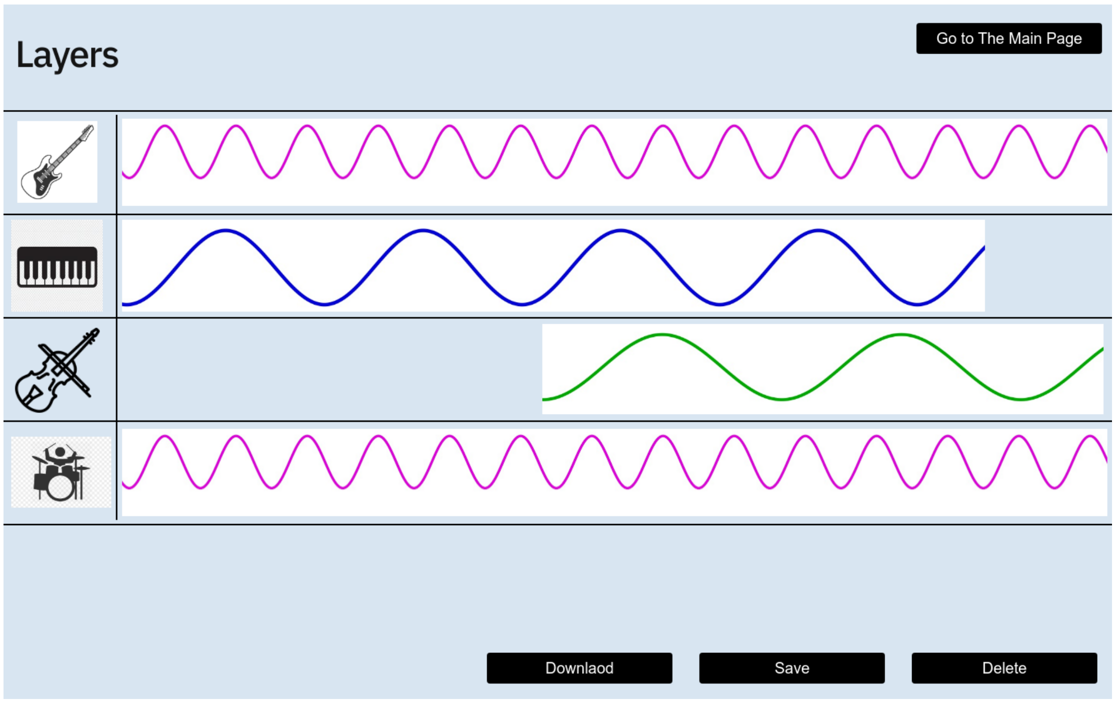
Parameter 2
Option Option Option [✓ Option] Option

Parameter 3
Option Option [✓ Option] Option Option

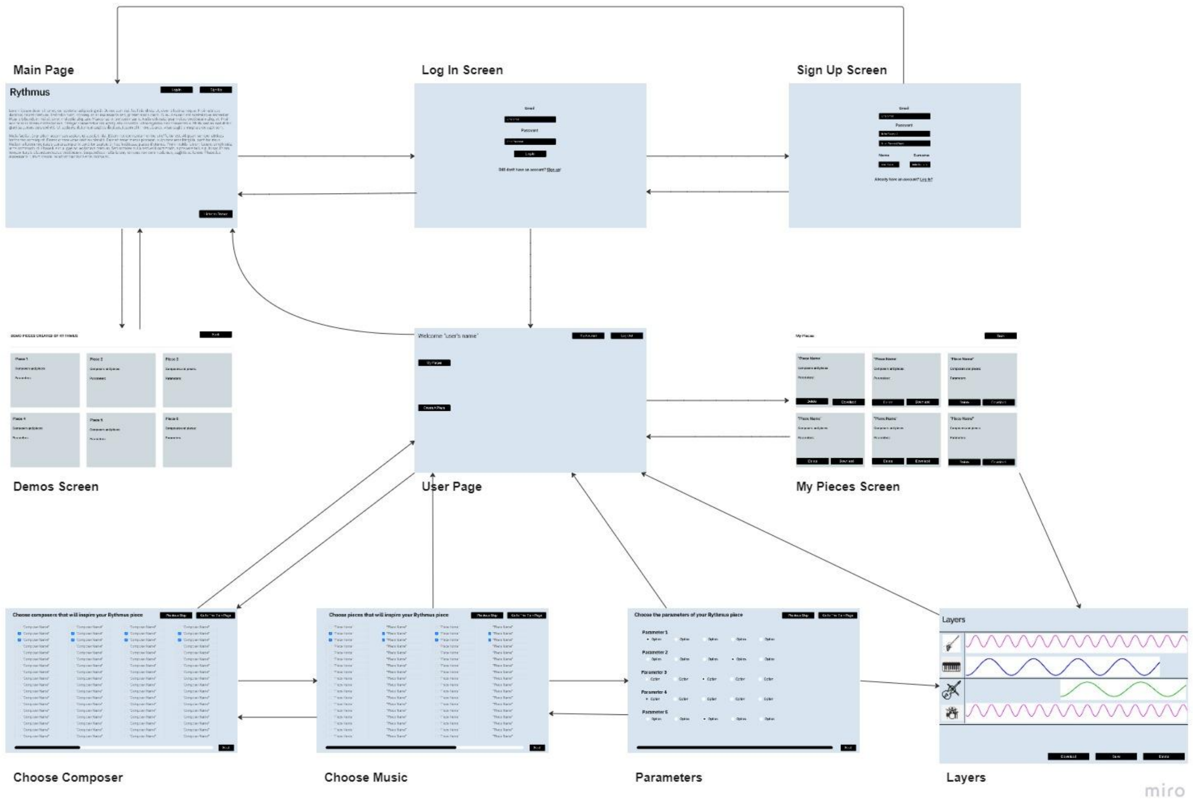
Parameter 4
[✓ Option] Option Option Option Option

Parameter 5
Option Option [✓ Option] Option Option

This is the final step before listening to the A.I. generated music. Users should choose their preferences of the parameters by clicking on the radio buttons labeled “Option”. Only one for each parameter will be allowed to select. With the “Previous Step” button on the top right corner users can navigate to the “Choose Pieces” page and with the button right next to that users can navigate to the “User Page”. After all parameters are selected, users can click the “Next” button and wait for the piece to be generated. When this process is done the user will be navigated to the “Layers” page.



Users can see the generated piece with its layers separated on this screen. There are going to be editing tools for editing the piece but they aren’t completely determined yet so they aren’t shown in this report. Users can use the buttons on the bottom right corner to choose what they are going to do with this song. “Download” button downloads the piece to the computer. “Save” button will save the piece to our database so that he/she can access it later on the “My Pieces” page. “Delete” button will delete the piece and return to the “User Page”. Also users can access this page through going to “My Pieces” and selecting a piece from the list. “Go to The Main Page” button will navigate the user to the “User Page”.



3 Other Analysis Elements

3.1 Consideration of Various Factors in Engineering Design

The project focused on music composers. Therefore the concern of how we can extend our application user occurs. There are many platforms that enable users to listen to music. However, they cannot change the music rhythms as they wish. Therefore, we can extend the user profile to this kind of people who want to mix pieces of music and rhythms.

Copyright is the main issue for the consideration factor because music companies have a right to their products. Our main goal is to create a safe place for the user to create their music from existing songs.

This program will also help people who want to become music composers. However, they do not have knowledge of how to create music from scratch. Therefore, our consideration is to encourage the people who are eager to create their own music from the existing song and can become well-known music composers in the future or they can do it as a hobby. Therefore, it can also expand the user profile.

3.2 Risks and Alternatives

The user amount is the main risk for our program because in the country there are not a lot of people who have an interest in creating music from scratch. Therefore, the user amount can be limited to a few people and it creates a risk to get a limited number of users and limited budget gain from the user. However, if the program becomes worldwide it can achieve an adequate number of people.

Furthermore, The risk is also the copyright rules of the music and getting fined by the music companies. If the companies do not allow the program to use their music the database of the instrumental layers will decrease dramatically. The alternative to this risk can be to create your own instrumental layers and use these types of layers to create the music from scratch.

3.3 Project Plan

Table 1: Factors that can affect analysis and design.

	Effect level	Effect
Public health	4	Rhythmus can affect the user's ear if the user listens to it at a high volume.
Public safety	1	The program does not affect any safety issues
Public welfare	8	Music makes people happy and relieves stress.
Global factors	5	Music can link people
Cultural factors	8	User can create their cultural music using layers

Social factors	8	Music helps people's minds and it can improve people's social life.

Table 2: Risks

	Likelihood	Effect on the project	B Plan Summary
Copyright. Issue	Not Likely	Our project is based on pieces of music and it has copyright but we are not affected	We can create our own music databases.
AI	Likely	creates pieces of music automatically for selecting layers	It can be created by the user.
Decompo sition instrumen tal layers	Likely	The project can not have well design instrumental music	Create the instrumental based from the instruments not the

Table 3: List of work packages

WP#	Work package title	Leader	Members involved
WP1	Create music database		Berk,Cihan,Azer
WP2	User Interface		Emir, Kerem Azer
WP3	Music decomposition to the instrumental layers		Berk,Cihan,Azer
WP4	Creating new music		Azer,Berk, Cihan,Emir, Kerem

WP5	Download the new music		Azer,Berk, Cihan,Emir, Kerem
WP6	Reports		Azer,Berk, Cihan,Emir, Kerem

WP 1: *Create music database*			
Start date: *13/11/2022* **End date:** *10/11/2022*			
Leader:		**Members involved:**	*Berk Cihan Azer*
Objectives: *Create music the databases*			
Tasks: ***Task 1.1 Select the composer*** **:** Selecting the composers who are going to involve to our program ***Task 1.2 Select the musics of the composer*** **:** *Selecting the composer musics*			
Deliverables ***D1.1:*** *Spotify API* ***D1.2:*** Music database.			
WP 2: *User Interface*			
Start date: *13/11/2022* **End date:** *10/11/2022*			
Leader:		**Members involved:**	Emir, Kerem Azer
Objectives: *Implementing the user interface*			
Tasks: ***Task 2.1*** **Main Page :** Include composer selection predefined instrumental layers. ***Task 2.2*** **Composer Page :** Composers and their music list will be implemented ***Task 2.3*** **Composer Page :** UI development *…*			
Deliverables ***D2.1:*** Screen mockups and navigation paths			

WP 3: *Music Decomposition to the layers*			
Start date: *13/11/2022* **End date:** *10/11/2022*			
Leader:		**Members involved:**	*Berk Cihan Azer*
Objectives: *Partition the music to instrumental layers*			
Tasks: ***Task 3.1 Partition to layers*:** Partition the instrumental layers of the music ***Task 3.2 Adding layers*:** *Adding layers to personnel layer database*			
Deliverables ***D3.1:*** *Holding the instrument layers in the database* ***D3.2:*** Music database.			

WP 4: *Create new music*			
Start date: *13/11/2022* **End date:** *10/11/2022*			
Leader:		**Members involved:**	*Azer, Berk, Cihan, Kerem ,Emir*
Objectives: *Create new music using the layers*			
Tasks: ***Task 4.1 Adding layers*:** Selecting the layers from existing layers ***Task 4.2 Create Music* :** *Creating new music using*			
Deliverables ***D4.1:*** *Layers from the music database* ***D4.2:*** Music database which is created			

WP 5: *Download the music*			
Start date: *13/11/2022* **End date:** *10/11/2022*			
Leader:		**Members involved:**	*Azer, Berk, Cihan, Kerem ,Emir*
Objectives: *Make the music downloaded and listen to it through computer.*			
Tasks: ***Task 5.1 Download music***: Turn the music to mp3 format			
Deliverables ***D5.1:*** *Turning the instrumental layers to the mp3 format*.			

WP 6: *Reports*			
Start date: *13/11/2022* **End date:** *10/11/2022*			
Leader:		**Members involved:**	*Azer, Berk, Cihan, Kerem ,Emir*
Objectives: *Distribution of the report tasks header*			
Tasks: ***Task 6.1 Project Specification Report*** ***Task 6.2 Analysis and Requirements Report*** ***Task 6.3 Presentation and Prototype Demo***			
Deliverables ***D6.1:*** *Turning the instrumental layers to the mp3 format*.			

3.4 Ensuring Proper Teamwork

To ensure proper teamwork, we must divide the project report pieces equally. As the instructors create our group, we do not know each other. Consequently, we have to create meetings to determine who wants to do which part of the project and ensure the proper teamwork we have to make approximately equal parts. We use google drive to write the reports simultaneously, and we can check our work. There must also be synchronization between group people's ideas, so all people are on the same page. Furthermore, we have to determine deadlines early because if a problem occurs, we can eliminate this problem.

3.5 Ethics and Professional Responsibilities

Ethical responsibility of the rhythmus project is based on the copyright issue. As described before, the project uses the kinds of music, divides them into instrument layers, and creates new music.; During this process, it can violate the copyright of the music. Therefore, we must ensure that this program does not infringe the copyright.

Professional responsibility is the main issue for our group because we are the five people who need to learn to know each other when the project begins and all of us should do their work before the deadline. In addition to that, the inside group deadline should be before the actual deadline because if the responsibilities still need to be done there has to be time to compensate for that.

3.6 Planning for New Knowledge and Learning Strategies

Our project rhythmus is different from the typical software applications because it decomposes the existing music and creates new music using the instrumental layers. Therefore, our members should search for how to decompose music and how to create a music instruments layer database. However, in that part, the strategy should be to search for the partition method of the music into instruments. Furthermore, Spotify API would benefit the project, and we also have to learn how to use Spotify API reading from the documents. We have never used such an API or an application based on music; therefore, it can be challenging for groups.

4 Glossary

Glossary for any domain-specific terms you use in your report.

5 References

Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.