A Mini Project Report on

"Automatic Music Generation"

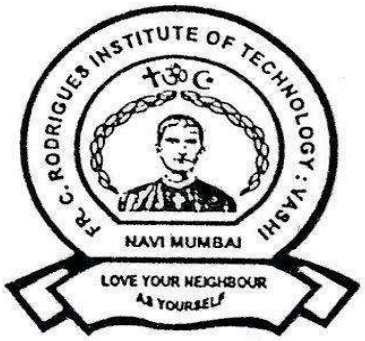
Submitted in partial fulfilment of the requirement for
Degree in T.E. (VI-Semester) in Computer Engineering

By

Benjamin Joseph (101806)
Fernando Edwin Hipson (101817)
Pinto Aldric Ivan (101847)
Verma Varun Vijaykumar (101864)

Guided By

Dr. Lata Ragha



Department of Computer Engineering
Fr. Conceicao Rodrigues Institute of Technology,
Sector 9A, Vashi, Navi Mumbai - 400703
University of Mumbai

2020-2021

CERTIFICATE

This is to certify that the Mini project entitled

Automatic Music Generation

Submitted by

Benjamin Joseph
Fernando Edwin Hipson
Pinto Aldric Ivan
Verma Varun Vijaykumar

In partial fulfilment of degree of T.E. (VI-Semester) in Computer Engineering for term work of the Mini Project is approved.

External Examiner **Internal Examiner**

External Guide **Internal Guide**

Head of Department **Principal**

Date: - **College Seal**

ACKNOWLEDGEMENT

Success of a Mini project like this involving high technical expertise, patience and massive support of guides, is possible when team members work together. We take this opportunity to express our gratitude to those who have been instrumental in the successful completion of this project. We would like to show our appreciation to **Dr. Lata Ragha** for their tremendous support and help, without them this project would have reached nowhere. We would also like to thank our project coordinator **Ms. Shweta Tripathi** for providing us with regular inputs about documentation and project timeline. A big thanks to our HOD **Dr. Lata Ragha** for all the encouragement given to our team. We would also like to thank our principal, **Dr. S. M. Khot**, and our college, Fr. C. Rodrigues Institute of Technology, Vashi, for giving us the opportunity and the environment to learn and grow.

ABSTRACT

In this topic titled ‘Automatic music generation’ we will be focussing on generating small pieces of music as the output. Our devised model will be trained using small music files (MIDI). MIDI (Musical Instrument Digital interface) is the type of music files used as they not only contain the actual audio but also some instructions. We then will be using the python library ‘music21’ which will break down these music files into chords and notes and separate them based on their frequency. We will be doing this to get the understanding of what type of frequency notes are combined together to form the actual music. We will be using the LSTM network to ensure the model does not lack structure and long-term dependencies. Being trained on this, now our trained model will take notes and chords as inputs and combine them to try to predict some meaningful music.

TABLE OF CONTENTS

ABSTRACT **4**
CHAPTER 1: INTRODUCTION 6
CHAPTER 2: LITERATURE SURVEY 8
CHAPTER 3: DESIGN 11
CHAPTER 4: TIMELINE CHART 15
CHAPTER 5: IMPLEMENTATION 16
CHAPTER 6: RESULTS 32
CONCLUSION 36
REFERENCES 37

Chapter 1

Introduction

The development of computer technologies and smart devices brought a turning point in music creation. Many individual and independent developers are in need of their own music for their own apps, and increasing people try to express their own personalities with their own style of ring tones. Therefore, the individual users are now trying to create their own music. However, the composition of music requires deep knowledge about music which cannot be easily obtained. Ordinary people have unfortunately little time to learn music composition techniques. Therefore, easy and automated composition of music for individual purposes will become more and more important. This can be achieved using Automatic Music Generation. Professional musicians also extensively use Deep Learning for their music generation. Deep learning has become an indispensable tool in the field of automated music generation. Numerous deep learning architectures have been studies to perform music generation tasks
Automatic Music Generation is a process of composing a short piece of music using minimum human intervention. Two main Deep Learning architectures are extensively used for Automatic Music Composition that are Wavenet and LSTM (Long Short Term Memory) architectures.

1. WaveNet
   WaveNet is a Deep Learning based generative model for raw audio developed by Google. The way it works is that it generates new samples from the original distribution of the given data.Wavenet takes the chunk of raw audio as input and then tries to predict the successive amplitude value according to the past pattern.
   So if given a sequence of samples , it tries to predict the next sample

2. LSTM
   Long Short Term Memory Model popularly known as LSTM is a variant of Recurrent Neural Network that is capable of capturing the long term dependencies in the input sequence. LSTM provides an added advantage i.e. to be able to capture medium-scale melodic structure in music fairly well, while the WaveNet component interprets and expands upon the generated melodic structure.

In our modelling architecture we use four different architecture which have been explained in brief below:
LSTM layers is a Recurrent Neural Net layer that takes a sequence as an input and can return either sequences (return_sequences=True) or a matrix.

Dropout layers are a regularisation technique that consists of setting a fraction of input units to 0 at each update during the training to prevent overfitting. The fraction is determined by the parameter used with the layer.
Dense layers or fully connected layers is a fully connected neural network layer where each input node is connected to each output node.
The Activation layer determines what activation function our neural network will use to calculate the output of a node.

Chapter 2

Literature Survey

Introduction

The development of computer technologies and smart devices brought a turning point in music creation. Many individual and independent developers are in need of their own music for their own apps, and increasing people try to express their own personalities with their own style of ring tones. Therefore, the individual users are now trying to create their own music. The music generated by softwares is nowadays mostly used for commercial purposes.

MIDI file format

Our Project automatic music generation uses MIDI files which are used to store message instructions which contain notes pitches, their velocity, when they begin and when they end etc.Using python library 'music21' these music files will be broken down into chords and notes and separated based on their frequency. A midi file contains two things, Header chunks and Track Chunks. Header chunk describing the file format and the number of track chunks. Each track chunk has one header and can contain as many midi commands as possible. Following the header are midi events.

Literature Review

The automation of music composition has been implemented in several different ways over the past decade. The earliest form of automatic music composition is to compose the fragments of melodies in accordance with randomly generated numbers. Some of the oldest methods used were rule-based systems (grammars) and algorithms that utilize randomness like the 'riffology' algorithm. The most common approach to the automated music composition is to utilize mathematical models or some simple rules known in music literature. Puente[1] used musical grammar to procedurally generate music. Boenn [2] also used musical grammars such as harmonics to implement 'Anton', an automated composition system. The system actually produced plausible music. However, the variety of the obtained music was not satisfactory enough.

The focus for music composition and music generation in more recent times has shifted towards machine learning and artificial intelligence techniques. This is due to the capability of NNs and deep learning to produce less predictable and more melodically complex melodies than their random algorithm and grammar systems counterparts. The majority of music generators in the past few years have been created using LSTM networks.

Eck and Schmidhuber in (2002) were the first to switch from regular RNNs to LSTMs to generate blues music. Blues is a genre of music similar to jazz containing a lot of guitar solo note sequences known as riffs. The network was successful in generating music pieces with a proper structure resembling normal blues music.Their LSTM network proved to be capable of learning chord sequences and recognizing blues chord structure.

With the recent success of speech synthesis models such as WaveNet (van den Oord et al. 2016), there has been increased interest in utilizing raw audio models for music generation. Raw audio models operate directly on the audio waveforms, both at training and generation times. WaveNet, for instance, is a model that attempts to predict the next sample of audio. The waveforms are typically sampled at 16 kHz, resulting in a computationally expensive model that requires many predictions per second of audio.
More recently, Skuli (2017) created a music generator with a dataset consisting of Final Fantasy (the video game) MIDI tracks. Skuli used MIDI's to train an LSTM NN and then generate his own MIDIs. They used Keras, which is a high-level deep learning Python API that runs on top of Tensorflow which simplifies interactions with the machine-learning library.

Different Approaches:

Recurrent neural network (RNN): RNN is a generalization of a feed-forward neural network that has an internal memory. It Performs the same function for every input of data while the output of the current input depends on the past one computation. RNN are not good at capturing long term dependencies and the problem of vanishing gradients may occur in RNN. As in every hidden stage it will calculate previous output, as the length of the sequence grows, the training time will also grow.
LSTM: In order to solve the vanishing gradient problem of RNN, aLong Short-Term Memory networks (LSTM) was proposed. The differences between RNN and LSTM are not fundamentally, the optimization is using different functions to compute the hidden stage. Memory in LSTM is called cells, it decides what to keep in memory. With more functions thatLSTM can add or remove information to the cell state. Those Functions are called gates. LSTM has three gates to preserve and discard information at cell state. Forget gate, Input gate,and Output gate.
GRU: GRU was proposed by Cho et al to make each recurrent unit to adaptively capture dependencies of different time scales. Similar to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cell.

Challenges:

For any artificial intelligence project, the biggest challenge faced is that the accuracy is never 100%. Since they are mere predictions that are formulated, which may or may not result in a true output. therefore the outputs generated are not always true and correct. Secondly Lot of data that is available today is raw and thus needs a lot of preprocessing before we can use it. Lack of computational power also is treated as a limitation to do any kind of machine learning project.

There exist no common rules for music so that discords sometimes sound fresh and nice. As a consequence, the automatic composition of music eventually fails when the musical grammar is overstressed. The well-defined grammars can be apredefined and limited set of musical pieces.

Previous works / Existing systems:

1. MuseNet:

A deep neural network that can generate 4-minute musical compositions with 10 different instruments, and can combine styles from country to Mozart to the Beatles. MuseNet was not explicitly programmed with the understanding of music, but instead discovered patterns of harmony, rhythm, and style by learning to predict the next token in hundreds of thousands of MIDI files.

2. Popgun:

Using deep learning, neural networks are trained on thousands of songs, varying across multiple genres. This training method allows the networks to interpret the style of a given musical composition, and 'play along' in a similar beat or pattern intended to complement or complete a melody played by a human user.

3. AIVA:

AIVA is an AI-powered online tool that lets you create soundtrack and theme music. There's a decent number of features provided to the users and the output is actually very decent. Their preset algorithms allow composing music in predefined styles like Cinematic, pop, rock, etc.

4. Humtap:

Humtap is a mobile app that lets anyone create music at runtime on the phone. You can just hum or tap to create a melody or beat in runtime and in different styles and instruments.

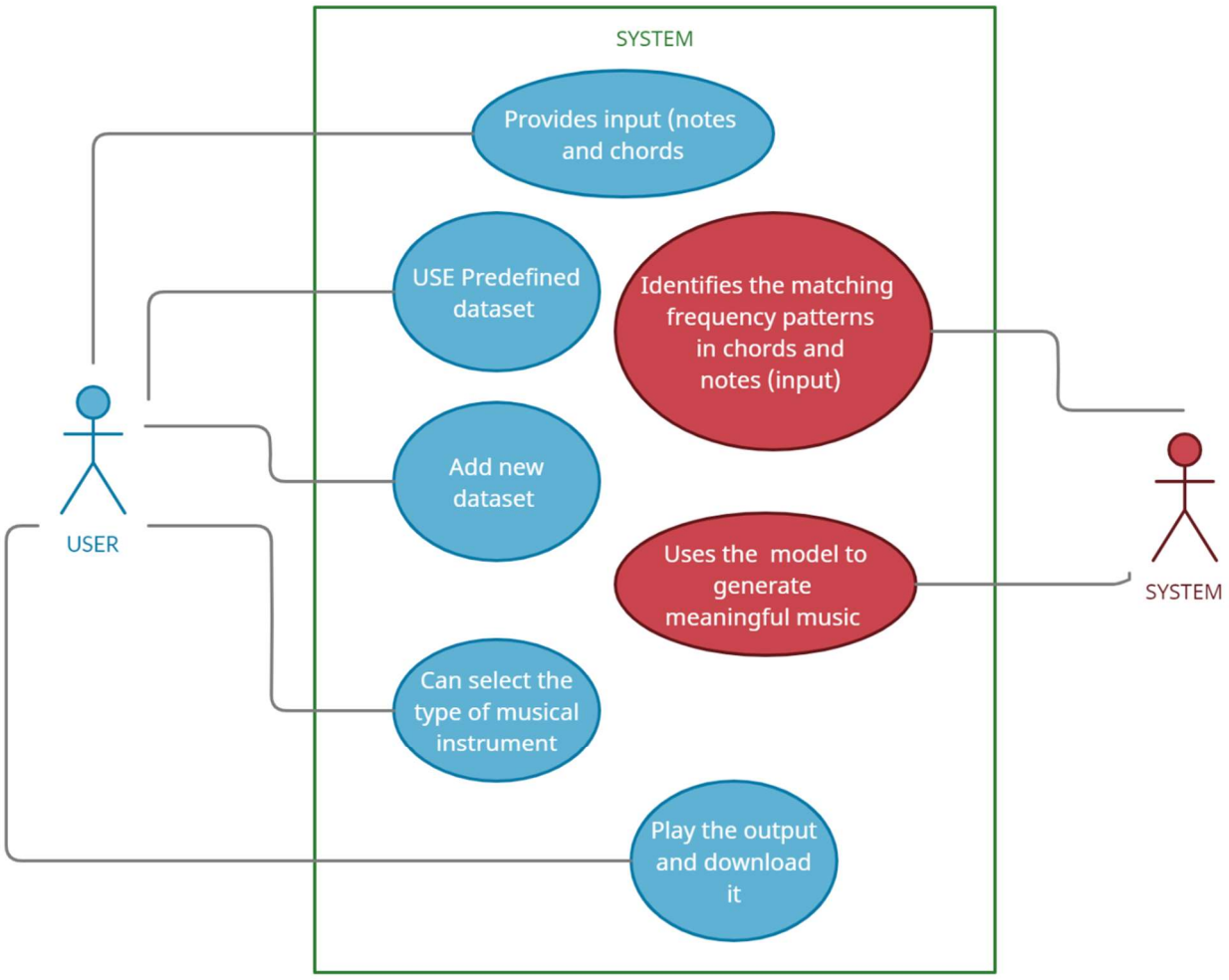
5. LANDR:

This one is a little different. Rather than create music, this tool masters the tracks to sound like a pro and uses AI/ machine learning for exactly that. So, the tool would look at the production elements of the composer, match it with a large number of album grade music it is fed and then create output filters that are musically pro.

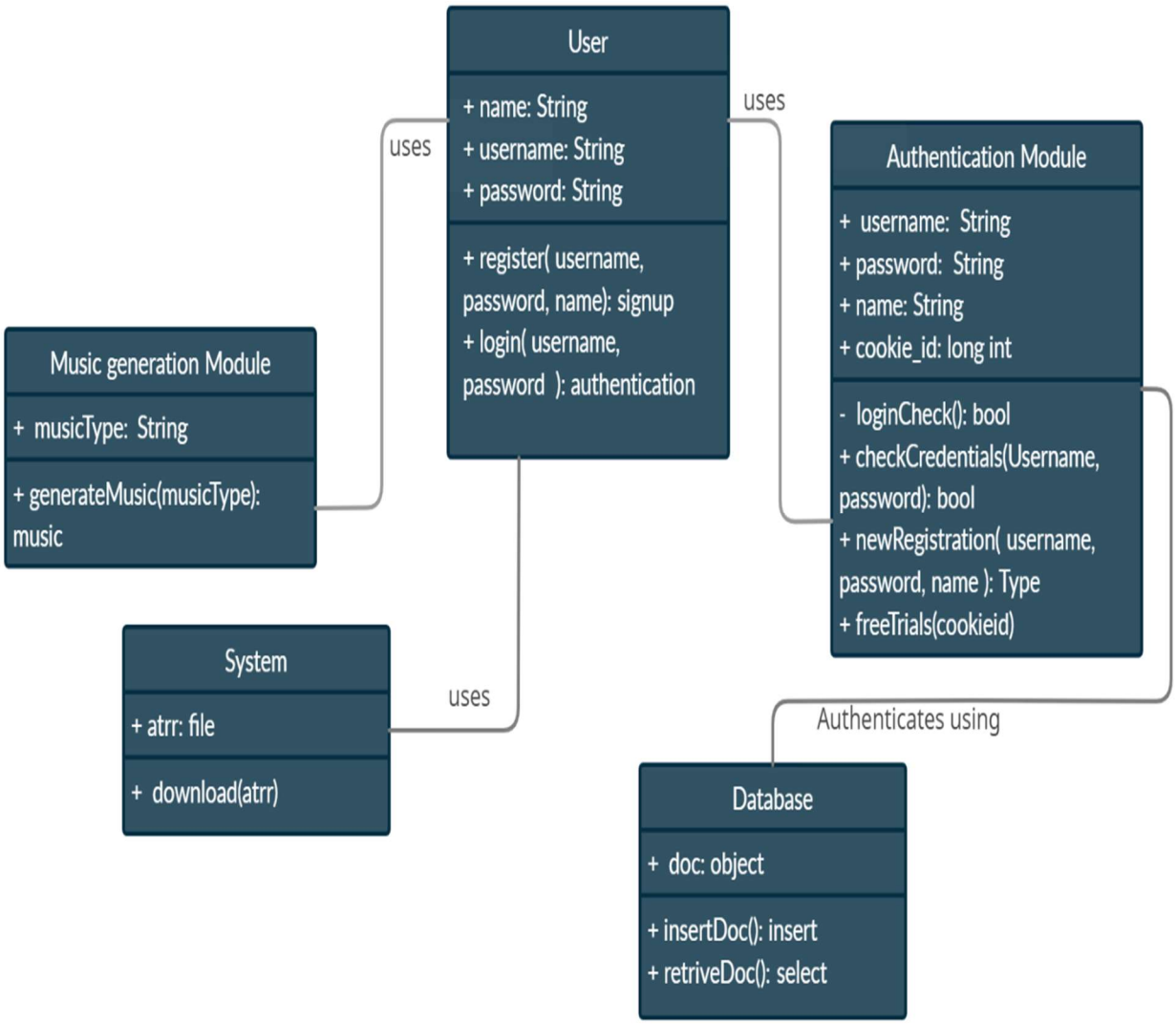
Chapter 3

Design

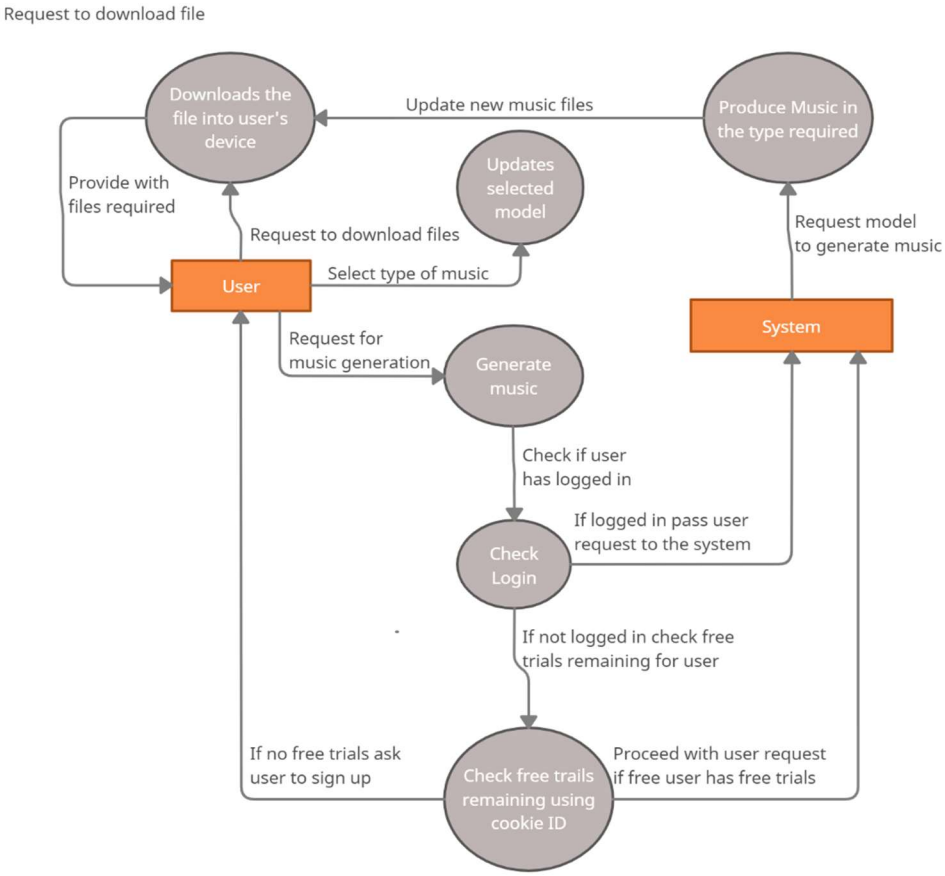
UML use case diagram:



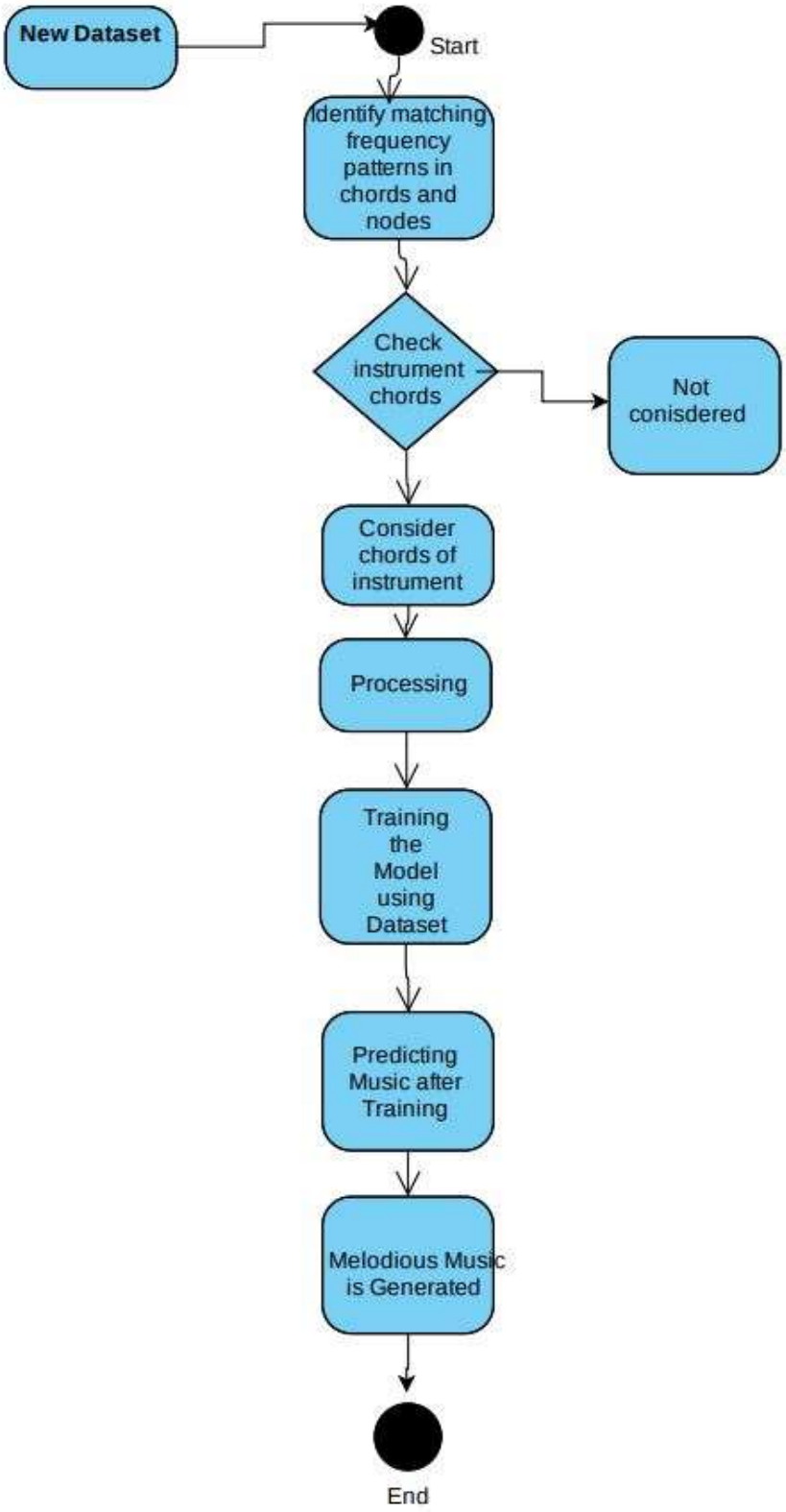
UML class diagram:



Data Flow Diagram:

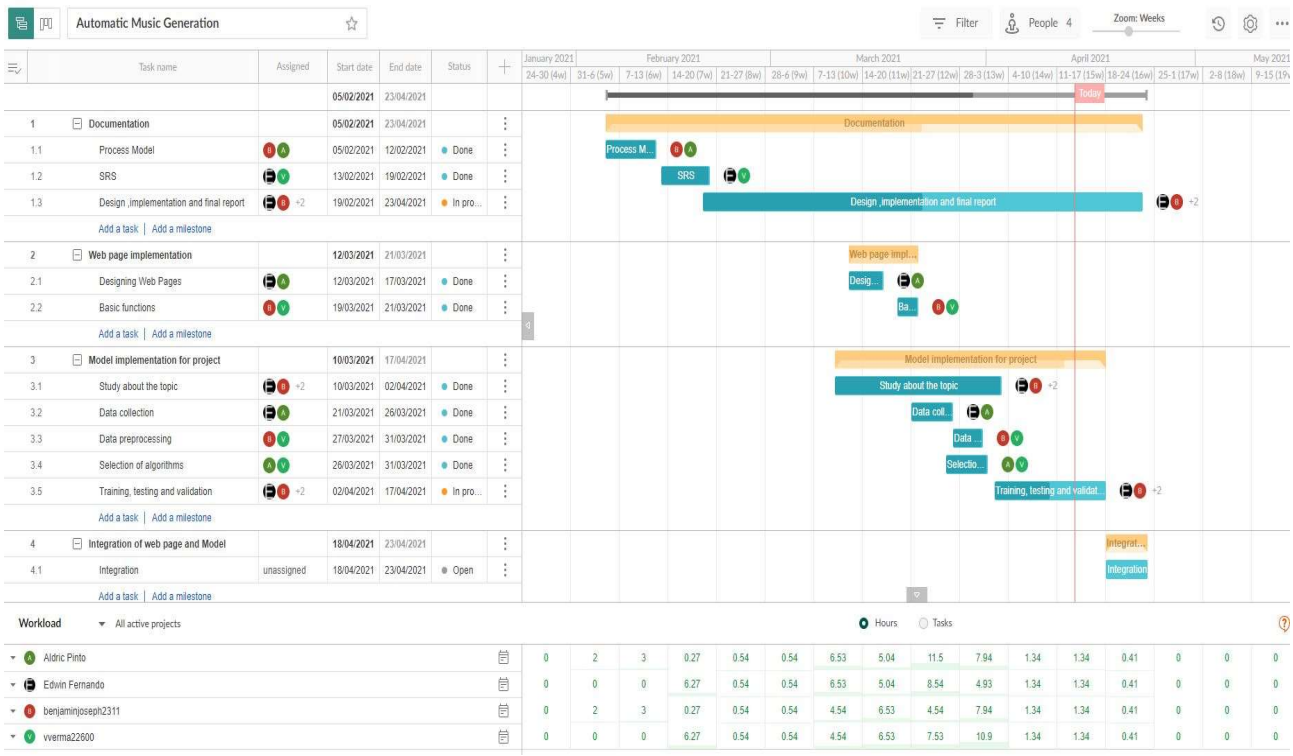


Activity Diagram:



Chapter 4

Timeline Chart



Chapter 5

Implementation

Main code (Python) files:

AMGtrain.ipynb [file for model training]

```
#library for understanding music
from music21 import *

#defining function to read MIDI files
def read_midi(file):

    print("Loading Music File:",file)

    notes=[]
    notes_to_parse = None

    #parsing a midi file
    midi = converter.parse(file)

    #grouping based on different instruments
    s2 = instrument.partitionByInstrument(midi)

    #My code
    try:

        #Looping over all the instruments
        for part in s2.parts:

            #select elements of only piano
            #if 'Piano' in str(part):

            notes_to_parse = part.recurse()

                #finding whether a particular element is note or a chord
            for element in notes_to_parse:

                #note
                if isinstance(element, note.Note):
                    notes.append(str(element.pitch))

                #chord
```

```
                elif isinstance(element, chord.Chord):
                    notes.append('.'.join(str(n) for n in element.normalOrder))

        return np.array(notes)

    except:
        print("skiping file")

#for listing down the file names
import os

#Array Processing
import numpy as np

#specify the path
path='/content/drive/MyDrive/ML_project/AMG/instrumental/'

#read all the filenames
files = [i for i in os.listdir(path) if i.endswith(".mid")]

for i in files:

    midi = converter.parse(path+i)

    #grouping based on different instruments
    s2 = instrument.partitionByInstrument(midi)
    try:
        if s2.parts:
            print("considering")
    except:
        files.remove(i)
        print("discarding")

#reading each midi file
notes_array = np.array([read_midi(path+i) for i in files])

#converting 2D array into 1D array
notes_ = [element for note_ in notes_array for element in note_]

#No. of unique notes
unique_notes = list(set(notes_))
print(len(unique_notes))

#importing library
from collections import Counter
```

```
#computing frequency of each note
freq = dict(Counter(notes_))

#library for visualiation
import matplotlib.pyplot as plt

#consider only the frequencies
no=[count for _,count in freq.items()]

#set the figure size
plt.figure(figsize=(5,5))

#plot
plt.hist(no)

frequent_notes = [note_ for note_, count in freq.items() if count>=40]
print(len(frequent_notes))

new_music=[]

for notes in notes_array:
    temp=[]
    for note_ in notes:
        if note_ in frequent_notes:
            temp.append(note_)
    new_music.append(temp)

new_music = np.array(new_music)

no_of_timesteps = 32
x = []
y = []

for note_ in new_music:
    for i in range(0, len(note_) - no_of_timesteps, 1):

        #preparing input and output sequences
        input_ = note_[i:i + no_of_timesteps]
        output = note_[i + no_of_timesteps]

        x.append(input_)
        y.append(output)

x=np.array(x)
y=np.array(y)
```

```
unique_x = list(set(x.ravel()))
file2 = open("/content/drive/MyDrive/ML_project/AMG/unique_x_1.txt", 'w')
file2.write(str(unique_x))
file2.close()
x_note_to_int = dict((note_, number) for number, note_ in enumerate(unique_x))

#preparing input sequences
x_seq=[]
for i in x:
    temp=[]
    for j in i:
        #assigning unique integer to every note
        temp.append(x_note_to_int[j])
    x_seq.append(temp)

x_seq = np.array(x_seq)

unique_y = list(set(y))
y_note_to_int = dict((note_, number) for number, note_ in enumerate(unique_y))
y_seq=np.array([y_note_to_int[i] for i in y])

from sklearn.model_selection import train_test_split
x_tr, x_val, y_tr, y_val = train_test_split(x_seq,y_seq,test_size=0.2,random_state=
0)

print(len(x_val[0]))
file1 = open("/content/drive/MyDrive/ML_project/AMG/xval_1.txt", 'a')
for i in x_val:
    print(i)
    file1.write(str(i)+"\n")
file1.close()

from keras.layers import *
from keras.models import *
from keras.callbacks import *
import keras.backend as K

def lstm():
  model = Sequential()
  model.add(LSTM(128,return_sequences=True))
  model.add(LSTM(128))
  model.add(Dense(256))
  model.add(Activation('relu'))
  model.add(Dense(len(unique_x)))
  model.add(Activation('softmax'))
  model.compile(loss='sparse_categorical_crossentropy', optimizer='adam')
```

```
  model.build((unique_x))
  model.summary()
  return model
model1 = lstm()

from keras.layers import *
from keras.models import *
from keras.callbacks import *
import keras.backend as K

K.clear_session()
model = Sequential()

#embedding layer
model.add(Embedding(len(unique_x), 100, input_length=32,trainable=True))

model.add(Conv1D(64,3, padding='causal',activation='relu'))
model.add(Dropout(0.2))
model.add(MaxPool1D(2))

model.add(Conv1D(128,3,activation='relu',dilation_rate=2,padding='causal'))
model.add(Dropout(0.2))
model.add(MaxPool1D(2))

model.add(Conv1D(256,3,activation='relu',dilation_rate=4,padding='causal'))
model.add(Dropout(0.2))
model.add(MaxPool1D(2))

#model.add(Conv1D(256,5,activation='relu'))
model.add(GlobalMaxPool1D())

model.add(Dense(256, activation='relu'))
model.add(Dense(len(unique_y), activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam')

model.summary()

mc=ModelCheckpoint('/content/drive/MyDrive/ML_project/AMG/Models/instrumental.h5',
monitor='val_loss', mode='min', save_best_only=True,verbose=1)

history = model.fit(np.array(x_tr),np.array(y_tr),batch_size=64,epochs=400, validat
ion_data=(np.array(x_val),np.array(y_val)),verbose=1, callbacks=[mc])
```

AMG.ipynb (File for running trained model)

```
#loading best model
from keras.models import load_model
import datetime
import numpy as np
from music21 import *
import random
import sys

model_given = sys.argv[1]
print(model_given)

if model_given == 'blues':
    model = load_model('D:/Study Related/SEM 6/Mini Project/Code/Models/blues.h5')
    file1 = open("D:/Study Related/SEM 6/Mini Project/Code/xval_blues.txt", 'r')
    file2 = open("D:/Study Related/SEM 6/Mini Project/Code/unique_x_blues.txt", 'r'
)
elif model_given == 'instrumental':
    model = load_model('D:/Study Related/SEM 6/Mini Project/Code/Models/instrumenta
l.h5')
    file1 = open("D:/Study Related/SEM 6/Mini Project/Code/xval_instrumental.txt",
'r')
    file2 = open("D:/Study Related/SEM 6/Mini Project/Code/unique_x_instrumental.tx
t", 'r')
elif model == 'piano':
    model = load_model('D:/Study Related/SEM 6/Mini Project/Code/Models/piano.h5')
    file1 = open("D:/Study Related/SEM 6/Mini Project/Code/xval_piano.txt", 'r')
    file2 = open("D:/Study Related/SEM 6/Mini Project/Code/unique_x_piano.txt", 'r'
)

lines = file1.readlines()
array_string = ""
for line in lines:
    array_string += str(line)
file1.close()
array_string = array_string.replace("[","").replace("]","")
x_val = np.fromstring(array_string, dtype=int, sep=' ')
x_val = x_val.reshape(-1,32)

ind = np.random.randint(0,len(x_val)-1)

random_music = x_val[ind]
no_of_timesteps = 32
```

```
predictions=[]
for i in range(100):

    random_music = random_music.reshape(1,no_of_timesteps)

    prob  = model.predict(random_music)[0]
    y_pred= np.argmax(prob,axis=0)
    predictions.append(y_pred)

    random_music = np.insert(random_music[0],len(random_music[0]),y_pred)
    random_music = random_music[1:]

print("\n"*4)
print(predictions)

unique_x = file2.readline()
file2.close()

unique_x = unique_x.replace("'","").replace(" ","").replace("[","").replace("]","")
.split(",")
x_int_to_note = dict((number, note_) for number, note_ in enumerate(unique_x))
predicted_notes = [x_int_to_note[i] for i in predictions]

def convert_to_midi(prediction_output):

    offset = 0
    output_notes = []

    # create note and chord objects based on the values generated by the model
    for pattern in prediction_output:

        # pattern is a chord
        if ('.' in pattern) or pattern.isdigit():
            notes_in_chord = pattern.split('.')
            notes = []
            for current_note in notes_in_chord:

                cn=int(current_note)
                new_note = note.Note(cn)
                new_note.storedInstrument = instrument.Piano()
                notes.append(new_note)

            new_chord = chord.Chord(notes)
            new_chord.offset = offset
            output_notes.append(new_chord)
```

```
        # pattern is a note
        else:

            new_note = note.Note(pattern)
            new_note.offset = offset
            new_note.storedInstrument = instrument.Piano()
            output_notes.append(new_note)

        # increase offset each iteration so that notes do not stack
        offset += 1
    midi_stream = stream.Stream(output_notes)
    midi_stream.write('midi', fp='D:/Study Related/SEM 6/Mini Project/AMG/static/mu
sic/music.mid')

convert_to_midi(predicted_notes)
```

Website files:

Index.html:

```
{% load static %}

<html>
    <head>
        <title>Automatic music generation</title>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.
js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/po
pper.min.js"></script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.m
in.js"></script>
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.
2/css/bootstrap.min.css">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
        <link rel="stylesheet" type="text/css" href="{% static 'css/index.css' %}">
    </head>

    <body>
        <!--alert messages for signup-->
        {% if redundant %}
            <script>
                window.alert('An account with this email Id already exists');
                window.history.go(-1);
```

```
            </script>
        {% endif %}

        {% if notredundant %}
            <script>
                window.alert('Registered successfully, login to continue');
                window.history.go(-1);
            </script>
        {% endif %}

        <!--alert messages for login-->{% if invalid %}
            <script>
                window.alert('Invalid Email or password');
                window.history.go(-1);
            </script>
        {% endif %}

        <div id="main_box" class="container">
            <h1 style="font-family: 'Georgia'; padding-
top: 0.3cm;"><b>Automatic Music Generation</b></h1>
            <div style="padding-top: 0.5cm;">
                <div align="right">
                    <a href="" class="btn btn-default btn-rounded mb-4" data-
toggle="modal" data-target="#SignupForm"><i class="fa fa-
user"></i> <b>Sign up</b></a>     
                    <a href="" class="btn btn-default btn-rounded mb-4" data-
toggle="modal" data-target="#LoginForm"><i class="fa fa-sign-
in"></i> <b>Login</b></a>
                </div>
                <div align="left">
                    <a href="{% static 'Automatic Music Generation Report.docx' %}"
 download><button class="btn"><i class="fa fa-
download"></i> <b>Report</b></button></a>     
                    <button align="left" class="btn" onclick="window.alert('Develop
ers won\'t reveal their identity for now')"><i class="fa fa-
gears"></i> <b>Know about the developers</b></button>     
                    <button class="btn" onclick="window.alert('Contact us at: amg.f
crit2022@gmail.com')"><i class="fa fa-
send"></i> <b>Contact Us</b></button>     
                </div>
            </div>
            <div class="row" style="padding-top: 0.3cm; padding-bottom: 0.5cm;">
                <div class="col-lg-6 col-md-6 col-sm-12">
                    <div class="mybox">
                        <p style="padding-left: 0.2cm; padding-
right: 0.2cm;" align="justify">
```

```
                        The development of computer technologies and smart devices
brought a turning point in music creation.
                        Many individual and independent developers are in need of t
heir own music for their own apps,
                        and increasing people try to express their own personalitie
s with their own style of ring tones.
                        Therefore, the individual users are now trying to create th
eir own music.
                        However, the composition of music requires deep knowledge a
bout music which cannot be easily obtained.
                        Ordinary people have unfortunately little time to learn mus
ic composition techniques.
                        Therefore, easy and automated composition of music for indi
vidual purposes will become more and more important.
                        This can be achieved using Automatic Music Generation.
                        Professional musicians also extensively use Deep Learning f
or their music generation.
                        Deep learning has become an indispensable tool in the field
 of automated music generation.
                        Numerous deep learning architectures have been studies to p
erform music generation tasks
                        Automatic Music Generation is a process of composing a shor
t piece of music using minimum human intervention.
                        </p>
                    </div><br>
                    <div align="center"><button class="glow" onclick="window.alert(
'Please signup and login to continue with this feature')"><i class="fa fa-
music"></i> <b>Generate Music</b></button></div>
                </div>

                <div class="col-lg-6 col-md-6 col-sm-12">
                    <img src="{% static 'images/pic1.png' %}" style="height:auto; w
idth:100%">
                </div>
        </div>

        <!--Modal for Signup Form-->
        <div class="modal fade" id="SignupForm" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel" aria-hidden="true">
            <div class="modal-dialog" role="document">
                <div class="modal-content" style="background-
color: rgb(43, 44, 44); color: white;">
                    <div class="modal-header text-center">
                        <h4 class="modal-title w-100 font-weight-bold">Sign up</h4>
                        <button style="color: white;" type="button" class="close" d
ata-dismiss="modal" aria-label="Close">
```

```
                        <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <form action="/signup" method="POST">
                        {% csrf_token %}
                        <div class="modal-body mx-3">
                            <div class="md-form mb-5">
                                <label data-error="wrong" data-
success="right"><i class="fa fa-user"> </i>Enter your Name</label>
                                <input type="text" id="name" name="name" class="for
m-control validate" required>
                            </div>

                            <div class="md-form mb-5">
                                <label data-error="wrong" data-
success="right" for="defaultForm-email"><i class="fa fa-
user"> </i>Enter your email</label>
                                <input type="email" id="emailid" name="emailid" cla
ss="form-control validate" required>
                            </div>

                            <div class="md-form mb-4">
                                <label data-error="wrong" data-
success="right" for="defaultForm-pass"><i class="fa fa-
key"> </i>Enter your password</label>
                                <input type="password" id="pwd" name="pwd" class="f
orm-control validate" required>
                            </div>
                        </div>
                        <div class="modal-footer d-flex justify-content-center">
                            <button type="submit" class="btn btn-
default"><b>Signup</b></button>
                        </div>
                    </form>
                </div>
            </div>
        </div>

        <!--Modal for Login Form-->
        <div class="modal fade" id="LoginForm" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel" aria-hidden="true">
            <div class="modal-dialog" role="document">
                <div class="modal-content" style="background-
color: rgb(43, 44, 44); color: white;">
                    <div class="modal-header text-center">
                        <h4 class="modal-title w-100 font-weight-bold">Login</h4>
```

```
                        <button style="color: white;" type="button" class="close" d
ata-dismiss="modal" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <form action="/login" method="POST">
                        {% csrf_token %}
                        <div class="modal-body mx-3">
                            <div class="md-form mb-5">
                                <label data-error="wrong" data-
success="right" for="defaultForm-email"><i class="fa fa-
user"> </i>Your email</label>
                                <input type="email" id="emailid" name="emailid" cla
ss="form-control validate" required>
                                </div>

                                <div class="md-form mb-4">
                                <label data-error="wrong" data-
success="right" for="defaultForm-pass"><i class="fa fa-
key"> </i>Your password</label>
                                <input type="password" id="pwd" name="pwd" class="f
orm-control validate" required>
                                </div>
                        </div>
                        <div class="modal-footer d-flex justify-content-center">
                            <button type="submit" class="btn btn-
default"><b>Login</b></button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </body>
</html>
```

AMG.html:

```
{% load static %}

<html>
    <head>
        <title>Automatic music generation</title>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.
2/css/bootstrap.min.css">
```

```
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
        <script src='https://kit.fontawesome.com/a076d05399.js' crossorigin='anonym
ous'></script>
        <link rel="stylesheet" type="text/css" href="{% static 'css/AMG.css' %}">
    </head>

    <body>
        <!--Go back-->
        {% if generated %}
        <script>
            window.history.go(-1);
        </script>
        {% endif %}
        <div id="main_box" class="container">
            <h1 style="font-family: 'Georgia'; padding-
top: 0.3cm;"><b>Automatic Music Generation</b></h1>
            <!--<div style="padding-top: 0.5cm;">
                <div align="right">
                    <a href="Automatic music generation.docx" download><button clas
s="btn"><i class="fa fa-
download"></i> <b>Report</b></button></a>     
                    <button align="left" class="btn" onclick="window.alert('Develop
ers won\'t reveal their identity for now')"><i class="fa fa-
gears"></i> <b>Know about the developers</b></button>     
                    <button class="btn" onclick="window.alert('You cannot contact u
s')"><i class="fa fa-send"></i> <b>Contact Us</b></button>
                </div>
            </div>-->
            <div class="row" style="padding-top: 0.3cm; padding-bottom: 0.5cm;">
                <div class="col-lg-5 col-md-5 col-sm-12">
                    <div class="mybox">
                        <form action="/script" method="POST">
                            {% csrf_token %}
                            <div>
                                <br><br>
                                <h3 style="font-
family: Tahoma;">Select the type of music you want:</h3>
                                <select class="form-
control" id="instrument" name="instrument">
                                    <option value="blues">Blues</option>
                                    <option value="instrumental">Instrumental</opti
on>
                                    <option value="piano">Piano</option>
                                </select>
                            </div><br><br>
```

```
                            <div align="center">
                                <button type="submit" class="gen"><i class="fa fa-
music"></i> <b>Generate Music</b></button><br><br>
                                <h3 style="font-
family: Tahoma;">Click on generate music to start generating random music</h3><br>

                            </div>
                        </form>
                        <div align="center">
                            <a href="{% static 'music/music.mid' %}" download><butt
on class="dnld"><i class="fa fa-
download"></i> <b>Download Music</b></button></a>
                        </div>
                    </div><br>
                </div>

                <div class="col-lg-7 col-md-7 col-sm-12">
                    <img src="{% static 'images/musicgif.gif' %}" style="height:aut
o; width:100%">
                </div>
        </div>

    </body>
</html>
```

Django Files:

views.py:

```
from django.shortcuts import render
from django.http import HttpResponse
import requests
from subprocess import run,PIPE
import sys

def index_page(request):
    return render(request, 'index.html')

def amg(request):
    return render(request, 'AMG.html')

def script(request):
    inp = request.POST.get('instrument')
    print(inp)
```

```
    out = run([sys.executable, 'D://Study Related//SEM 6//Mini Project//Code//AMGnew.py',inp],shell=False,stdout=PIPE)
    return render(request, 'AMG.html', {'generated':True})

def signup(request):
    import sqlite3
    con=sqlite3.connect('D://Study Related//SEM 6//Mini Project//AMG//amg.db')
    cursorobj=con.cursor()

    #creating table if not exists
    cursorobj.execute('CREATE TABLE IF NOT EXISTS members (Name varchar(50), email varchar(200) PRIMARY KEY, pass varchar(100))')

    name = request.POST.get('name')
    emailid = request.POST.get('emailid')
    pwd = request.POST.get('pwd')
    #print(name, emailid, pwd)

    data = cursorobj.execute('SELECT name FROM members WHERE email = ?',(emailid,))
    result = data.fetchall()
    if len(result) == 0:
        cursorobj.execute('INSERT INTO members VALUES (?, ?, ?)',(name, emailid, pwd))
        con.commit()
        return render(request, 'index.html',{'notredundant':True})
    else:
        return render(request, 'index.html', {'redundant':True})

def login(request):
    import sqlite3
    con=sqlite3.connect('D://Study Related//SEM 6//Mini Project//AMG//amg.db')
    cursorobj=con.cursor()

    emailid = request.POST.get('emailid')
    pwd = request.POST.get('pwd')

    data = cursorobj.execute('SELECT name FROM members WHERE email = ? AND pass = ?',(emailid, pwd))
    result = data.fetchall()
    if len(result) == 0:
        return render(request, 'index.html',{'invalid':True})
    else:
        return render(request, 'index2.html', {'valid':True, 'user':result[0][0]})
```

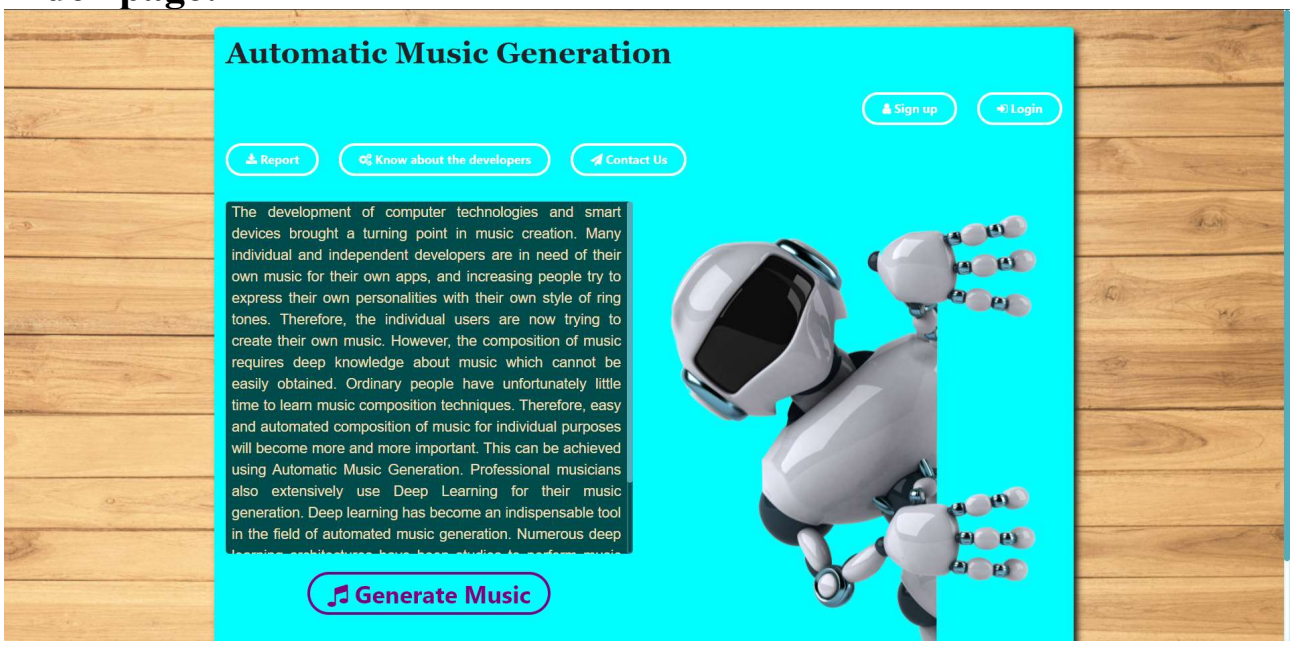
urls.py:

```
"""AMG URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index_page, name='index'),
    path('amg', views.amg, name='amg'),
    path('script', views.script, name='script'),
    #signup trial
    path('signup', views.signup, name='signup'),
    path('login', views.login, name='login'),
]
```

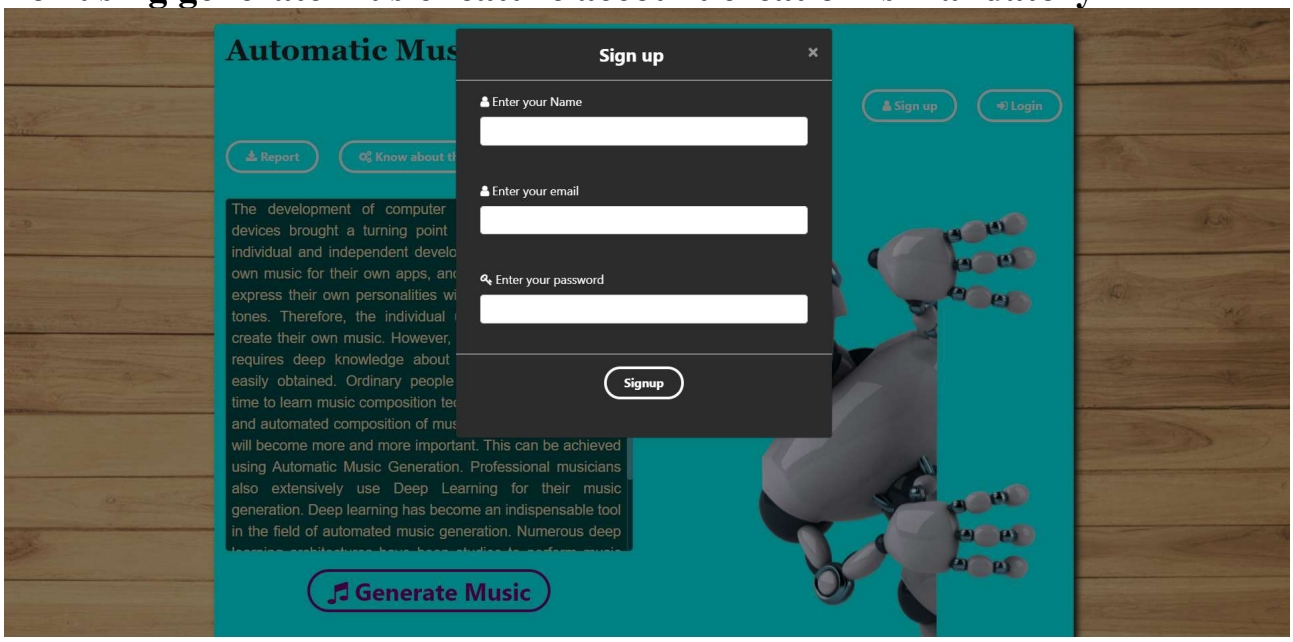
Chapter 6

Results

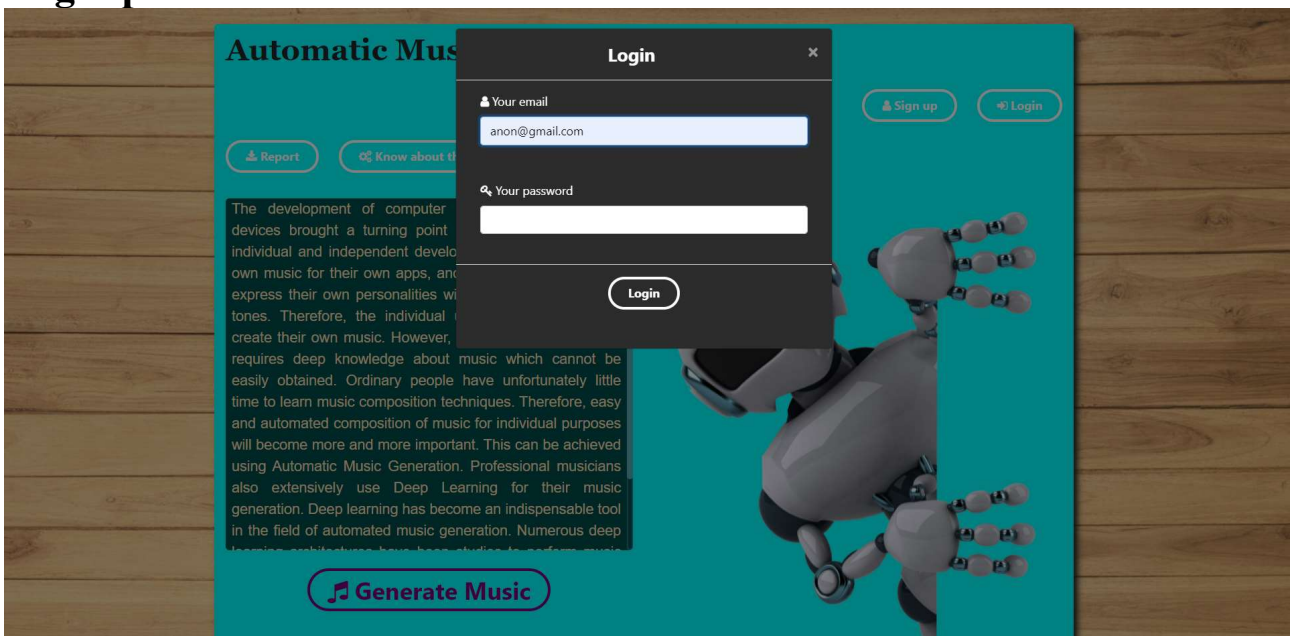
Index page:



Signup portal:
For using generate music feature account creation is mandatory



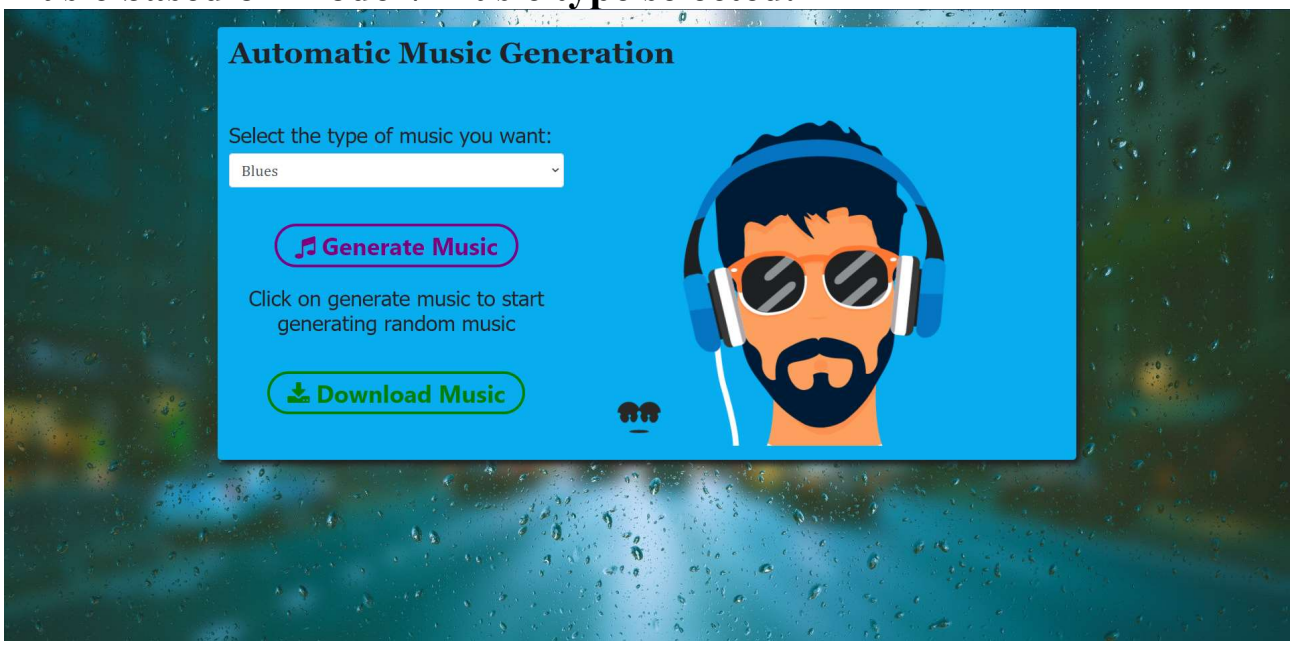
Login portal:



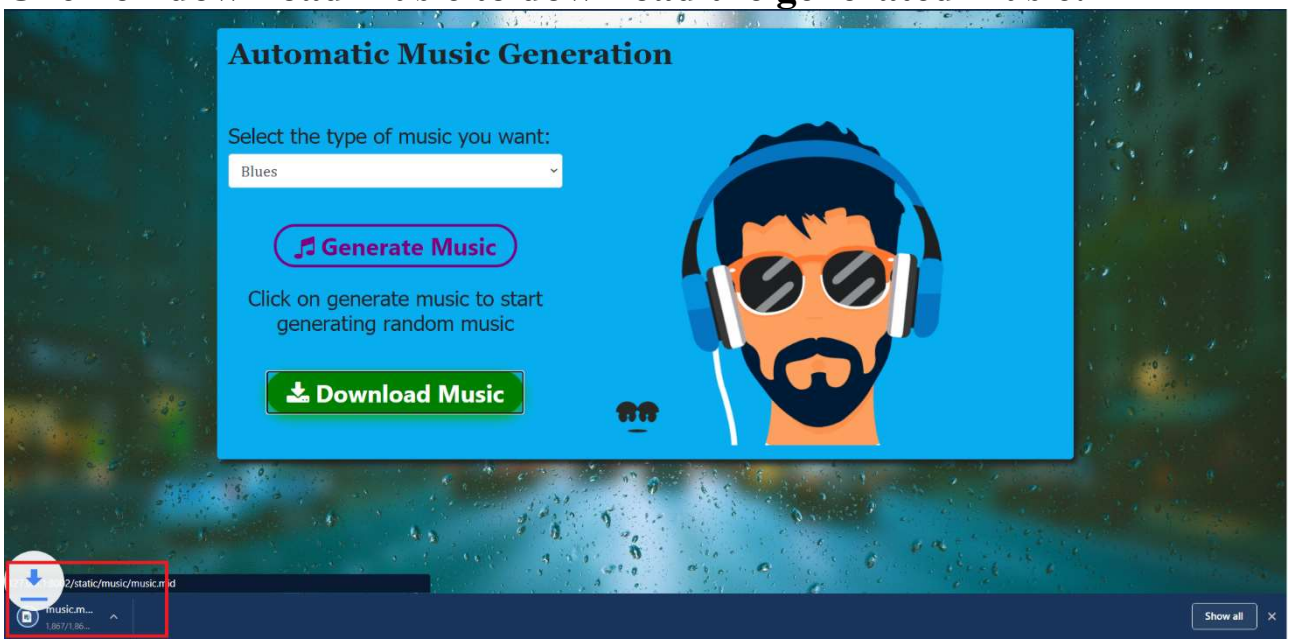
Index page after login:

Downloading report on button click:

Music Generation Page:
Click on generate music to run the external python script to generate music based on model / music type selected.



Click on download music to download the generated music:



Conclusion

Benjamin Joseph (101806)
Fernando Edwin Hipson (101817)
Pinto Aldric Ivan (101847)
Verma Varun Vijaykumar (101864)
of 6th semester
have successfully
completed the project
Automatic Music Generation
Under the guidance of
Dr. Lata Ragha

References

1. **https://www.researchgate.net/publication/302206040_Automatic_Music_Generation_and_Machine_Learning_Based_Evaluation**
2. **https://www.analyticsvidhya.com/blog/2020/01/how-to-perform-automatic-music-generation/**
3. **https://www.techrxiv.org/articles/preprint/A_Review_of_Automatic_Music_Generation_Based_on_Performance_RNN/12088980/1**
4. **https://openai.com/blog/musenet/**
5. **https://analyticsindiamag.com/7-online-artificial-intelligence-tools-to-generate-your-own-music/**
6. **Puente A.et al :Automatic composition of music by means of grammatical evolution . Proceedings of the 2002 conference on APL, pp 148-155. (2002)**
7. **Boenn G. : Anton: Answer Set Programming in the Service of Music. Proceedings of the Twelfth International Workshop on Non-Monotonic Reasoning. Sydney: University of New South Wales, pp. 85-93.. (2008).**