A

MINI PROJECT REPORT

ON

Interactive MIDI Musical Keyboard: Simplified Music Creation using Python

Submitted in partial fulfillment of the requirements

For the award of Degree of

BACHELOR OF ENGINEERING

IN

CSE (AI ML)

T VISHNU VANDHAN 2453-21-748-058

Under the guidance

Of

Dr. B. Suvarnamukhi

ASSISTANT PROFESSOR

Department of CSE(AIML)

NEIL GOGTE INSTITUTE OF TECHNOLOGY

Kachavanisingaram Village, Hyderabad, Telangana, 500058.

FEBRUARY 2024

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

CERTIFICATE

This is to certify that the Mini project work entitled "**Interactive MIDI Musical Keyboard: Simplified Music Creation using Python"** *is a bonafide work carried out by* **T. Vishnu Vandhan (2453-21-748-058)** of III-year V semester **Bachelor of Engineering** *in CSE(***AIML)** *by Osmania University, Hyderabad during the academic year* **2023-2024** *is a record of bonafide work carried out by me. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree.*

Internal Guide

Dr. B. Suvarnamukhi

Assistant Professor

Head of Department

Dr. T. Prem Chander

Associate Professor

External

NEIL GOGTE INSTITUTE OF TECHNOLOGY
A Unit of Keshav Memorial Technical Education (KMTES)
Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

DECLARATION

I hereby declare that the Mini Project Report entitled, "**Interactive MIDI Musical Keyboard: Simplified Music Creation using Python**" submitted for the B.E degree is entirely my work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

Date:

T VISHNU VANDHAN (2453-21-748-058)

ACKNOWLEDGEMENT

I am happy to express my deep sense of gratitude to the principal of the college **Dr. R. Shyam Sunder, Professor**, Neil Gogte Institute of Technology, for having provided me with adequate facilities to pursue my project.

I would like to thank**, Dr. K. T. Prem Chander**, **Head of the Department,** CSE(AIML), Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

I would also like to thank my internal guide **Dr. B. Suvarnamukhi, Assistant Professor** for her technical guidance & constant encouragement.

I sincerely thank my seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering for their timely suggestions, healthy criticism and motivation during this work.

Finally, I express my immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

ABSTRACT

The objective is to develop a musical keyboard system where each key corresponds to a specific musical note. When a key is pressed, it triggers the corresponding musical sound. The design focuses on creating a user interface that resembles a piano, allowing users to interact with virtual keys and produce music effortlessly. The key emphasis is on simplicity and interactivity, ensuring that users can enjoy playing music without the complexities of raw audio signals or advanced technologies. The project will also incorporate error handling and refinement techniques to provide users with a seamless experience while playing music. The goal is to offer a straightforward yet engaging platform for users to explore and enjoy the world of music through their keyboards.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	ACKNOWLEDGEMENT	I
	ABSTRACT	II
	LIST OF FIGURES	III
	LIST OF TABLES	IV
1.	INTRODUCTION:	
	1.1. PROBLEM STATEMENT	1
	1.2. MOTIVATION	1
	1.3. SCOPE	1
	1.4. OUTLINE	2
2.	LITERATURE SURVEY:	
	2.1. EXISTING SYSTEM	3
	2.2. PROPOSED SYSTEM	3 - 4
3.	SOFTWARE REQUIREMENT SPECIFICATION:	
	3.1. OVERALL DESCRIPTION	5
	3.2. OPERATING ENVIRONMENT	5
	3.3. FUNCTIONAL REQUIREMENTS	5 - 6
	3.4.NON-FUNCTIONAL REQUIREMENTS	6 - 8

4.	**SYSTEM DESIGN:**	
	4.1. USE – CASE DIAGRAM	**9**
	4.2. CLASS DIAGRAM	**10**
	4.3. SEQUENCE DIAGRAM	**10**
	4.4. ACITVITY DIAGRAM	**11**
5.	**IMPLEMENTATION:**	
	5.1. SAMPLE CODE	**12 - 20**
6.	**TESTING:**	
	6.1. TEST CASES	**21 - 22**
7.	**SCREENSHOTS**	**23**
8.	**CONCLUSION AND FUTURE SCOPE**	**24**
	BIBLIOGRAPHY	**25 - 26**
	APPENDIX: TOOLS AND TECHNOLOGY	**27**

CHAPTER – 1
INTRODUCTION

1.1 PROBLEM STATEMENT

The problem tackled in this project is the inadequacies of current musical keyboard systems that map keyboard keys to musical notes. These systems, used in music production and education, face issues like limited customization, complexity for beginners, and insufficient error handling. The proposed solution, a Python-based musical keyboard, utilizes MIDI and sound synthesis libraries to overcome these challenges. The aim is to offer an improved system with better error handling, increased customizability, and enhanced integration potential, providing users with a more user-friendly and versatile musical experience.

1.2 MOTIVATION

The motivation behind this project is to enhance the user experience in music creation by addressing the limitations found in existing musical keyboard systems. Current systems often lack customizability, pose complexity for beginners, and suffer from inadequate error handling. The project aims to provide a user-friendly solution with a focus on simplicity and interactivity. By creating a musical keyboard system in Python, leveraging MIDI and sound synthesis libraries, the goal is to offer users a seamless and engaging platform for music production. The incorporation of error handling and refinement techniques further contributes to a smooth and enjoyable music-playing experience. Overall, the project aspires to provide a professional yet accessible tool for users to explore and enjoy the world of music through their keyboards.

1.3 SCOPE

The project aims to create a user-friendly musical keyboard system using Python, incorporating MIDI and sound synthesis libraries for enhanced musical experiences. The development phases include pre-processing, model training, and classification, leveraging a VGG16 CNN model for bird species identification. Technical implementation involves Python, MIDI libraries like mido or python-rtmidi, and sound synthesis libraries like FluidSynth or pygame. The user interface mimics a piano, offering an interactive experience with virtual keys lighting up upon pressing physical keys. Error handling ensures a smooth user experience, and user feedback refines the interface. Advantages over existing systems include improved error handling, enhanced customizability, open-source accessibility, and increased integration potential, providing a professional yet accessible platform for music enthusiasts.

1.4 OUTLINE:

The project seeks to improve current musical keyboard systems used in music production and education. These systems face issues like limited customization, complexity for beginners, and poor error handling. To address this, the proposed solution is a Python-based musical keyboard utilizing MIDI and sound synthesis libraries. The main goals are to enhance error handling, increase customizability, and improve integration, providing users with a more user-friendly and versatile musical experience. The project's structure includes an introduction to existing issues, a detailed exploration of problems, an overview of the proposed solution, and a focus on comprehensive resolution to the inadequacies in current musical keyboard systems.

CHAPTER – 2

LITERATURE SURVEY

EXISTING SYSTEM:

There are existing systems and software applications based on the concept of mapping keyboard keys to musical notes. These systems are often used in music production, live performances, and educational settings. Some popular examples include:

DAWs (Digital Audio Workstations): Many professional music production software like Ableton Live, FL Studio, and Logic Pro allow users to map computer keyboard keys to musical notes. This feature enables musicians and producers to play virtual instruments using their computer keyboards.

Online Virtual Pianos: Several websites and applications offer virtual pianos where users can play piano notes using their computer keyboards.

These platforms map keys to corresponding piano keys, allowing users to create music online without the need for physical instruments.

Educational Software: Educational software designed for teaching music theory and piano often includes virtual keyboards that can be played using computer keyboards. These tools are valuable for beginners learning to play musical instruments.

Gaming Keyboards with Music Modes: Some gaming keyboards come with customizable keys and modes that allow users to map keys to musical notes or functions. These keyboards can be configured to create music or trigger sound effects during gaming sessions. The limitations of the existing system are as follows:

1. Limited Customizability
2. Complexity for Beginners
3. Lack of Seamless Error Handling
4. Dependency on Specific Software
5. Limited Integration Potential

PROPOSED SYSTEM:

The proposed system involves creating a musical keyboard system using Python. It utilizes MIDI libraries like mido or python-rtmidi to generate MIDI signals corresponding to different

musical notes. Sound synthesis libraries such as pygame are employed to produce audio signal directly from MIDI notes without complex processing. Each key on the mechanical keyboard is mapped to a specific MIDI note, triggering the corresponding musical sound. The user interacts with a piano-like interface displayed on the screen using a Web App with HTML, CSS, JavaScript. Pressing a key on the physical keyboard lights up the corresponding virtual key on the GUI, providing an interactive experience. When a key is pressed, the mapped MIDI signal is sent to the sound synthesis library, generating the musical note. Error handling is implemented to manage rapid or simultaneous key presses, and user feedback is utilized to refine the user interface and enhance the overall experience. The proposed system is advantageous against the existing systems in many ways:

- Error Handling for Seamless Experience
- Customizability and Sound Libraries
- Open-Source and Community Collaboration
- Integration Potential

CHAPTER - 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Overall Description:

This SRS is an overview of the whole project scenario. This document is to present a detailed description of the course management system. It will explain the purpose and features of the system, the interfaces of the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both stakeholders and developers of the system.

3.2. Operating Environment:

Software Requirements:

Operating System	:	Windows 7 (Min)
Front End	:	HTML, CSS, JavaScript
Back End	:	Python, Flask

Hardware Requirements:

Processor	:	Intel Pentium®Dual CoreProcessor(Min)
Speed	:	2.9 GHz (Min)
RAM	:	2 GB (Min)
Hard Disk	:	2 GB (Min)

3.3 Functional Requirements:

User Functionality:

- The user will be able to generate music according to the requirements he/she needs.
- The user can dynamically change the instrument to their needs.

Admin Functionality:

- The admin manages the website.

- The admin can make changes to the website such as modifying the UI and making it more interactive than earlier.
- The admin can implement a better algorithm if at all a better algorithm is created in future.

3.4 Non-Functional Requirements:

3.4.1 Performance Requirements:

Performance requirements refer to static numerical requirements placed on the interaction betIen the users and the software.

Response Time:

Average response time shall be less than 5 sec.

Recovery Time:

In case of system failure, the redundant system shall resume operations within 30 secs. Average repair time shall be less than 45 minutes.

Start-Up/Shutdown Time:

The system shall be operational within 1 minute of starting up.

Capacity:

The system accommodates 1000 Concurrent Users.

Utilization of Resources:

The system shall store in the database no more than 450 different species with room for improvement.

3.4.2 Safety Requirements:

-NA-

3.4.3 Security Requirements:

The model will be running on a secure website i.e., an HTTPS website and on a secure browser such as Google Chrome, Brave, etc.

3.4.4 Software Quality Attributes:

Reliability:

The system shall be reliable i.e., in case the webpage crashes, progress will be saved.

Availability:

The website will be available to all its users round the clock i.e., they can access the website at any time.

Security:

The model will be running on a secure website i.e., an HTTPS website and on a secure browser such as Google Chrome, Brave, etc.

Maintainability:

The maintainability of the proposed musical keyboard system is a priority for ease of future management. The use of Python, a readable and well-supported language, ensures straightforward maintenance. The chosen MIDI and sound synthesis libraries are widely accepted, adding stability to the system. With a modular design and distinct phases, updates or changes can be made efficiently. The inclusion of error handling enhances resilience, simplifying the identification and resolution of potential issues over time. Additionally, the open-source nature encourages community collaboration, allowing ongoing improvements and ensuring the long-term maintainability and development of the musical keyboard system.

Usability:

The interfaces of the system will be user friendly enough that every user will be able to use it easily.

Scalability:

The proposed Python-based musical keyboard system aims to address limitations in current systems by utilizing MIDI and sound synthesis libraries. It offers a user-friendly interface through a web app, allowing users to play virtual keys that correspond to different musical notes on a physical keyboard. The system emphasizes error handling, responsive user feedback, and integration potential. Its scalability is contingent on hardware compatibility, adaptability of the user interface, flexibility in library integration, robust error handling, open-source community collaboration, and performance scalability. The success of the project in these aspects will determine its scalability and broader applicability over time.

CHAPTER-4

SYSTEM DESIGN

USE CASE DIAGRAM:

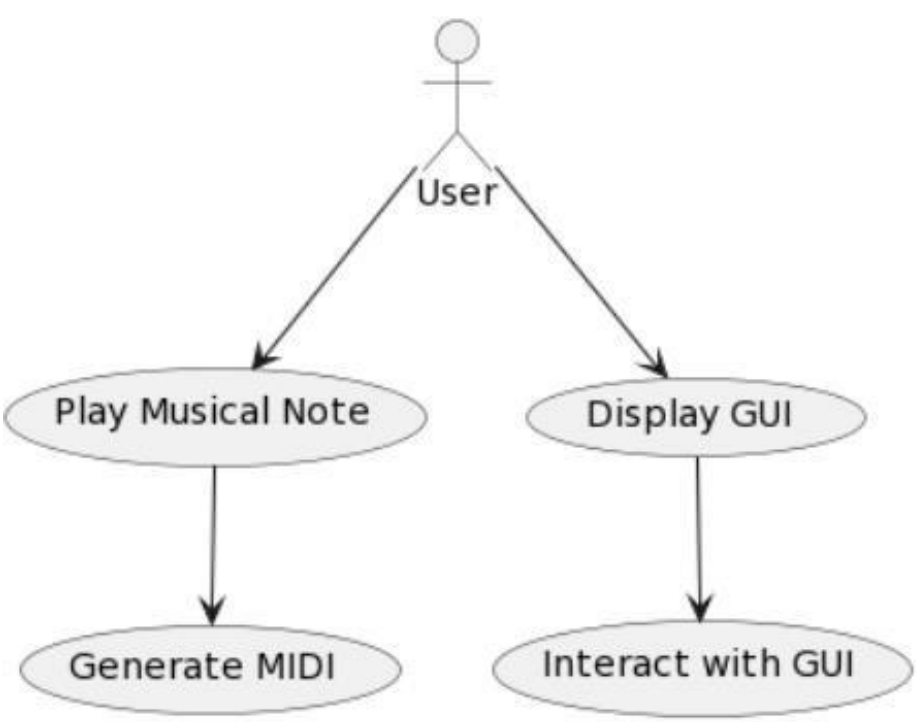


Fig 4.1: USE CASE DIARGAM FOR USER

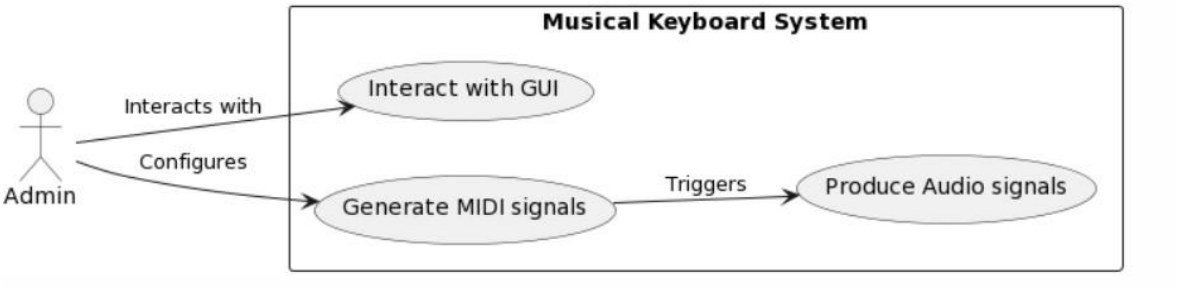


Fig 4.2: USE CASE DIAGRAM FOR ADMIN

CLASS DIAGRAM:

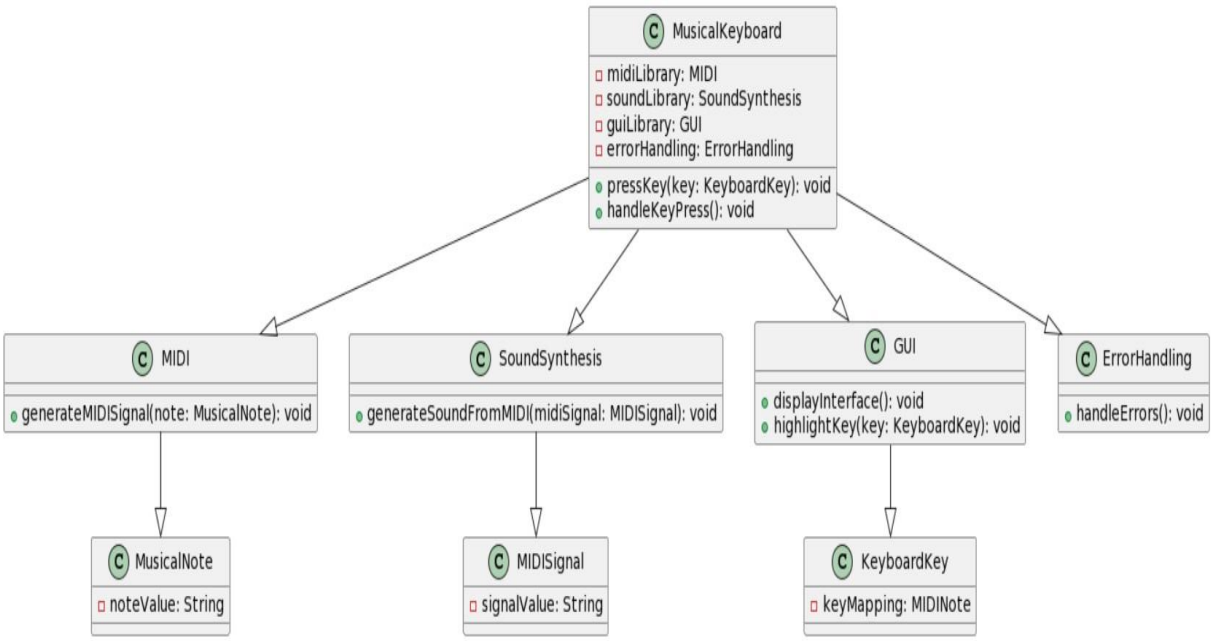


Fig 4.3: CLASS DIAGRAM FOR MUSIC IDENTIFICATION

SEQUENCE DIAGRAM:

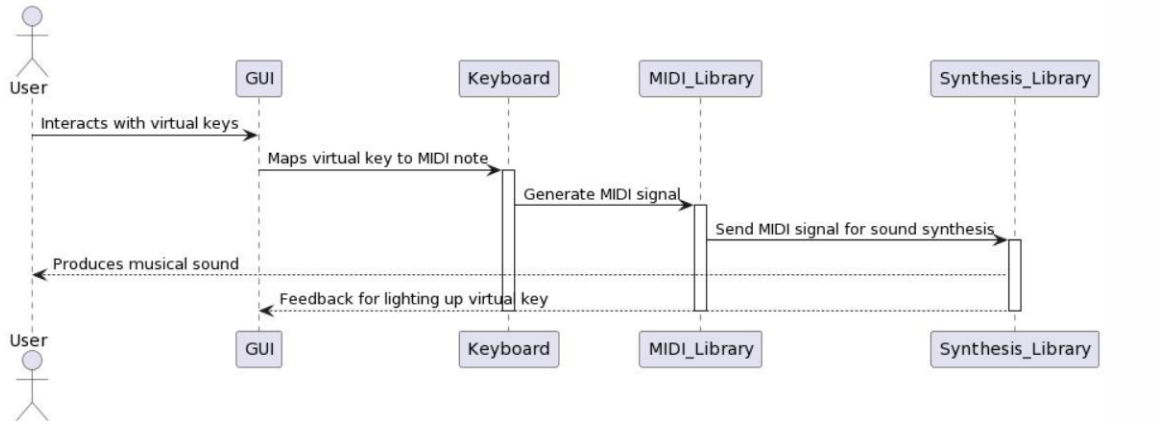


Fig 4.4: SEQUENCE DIAGRAM FOR MUSIC GENERATION

ACTIVITY DIAGRAM:

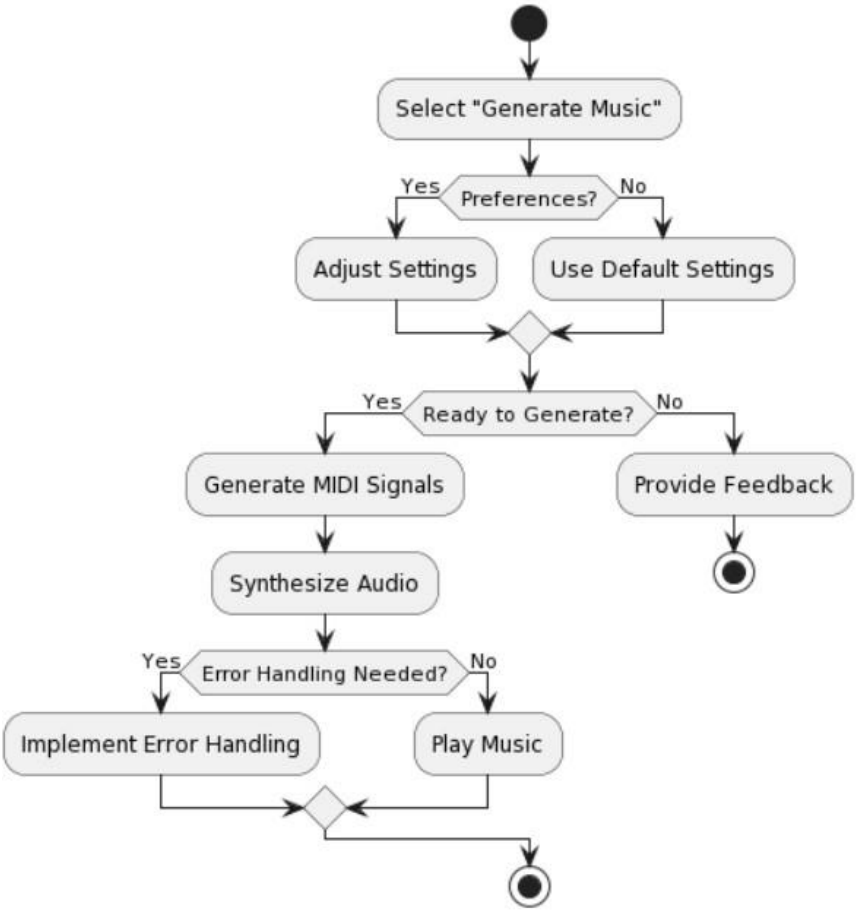


Fig 4.5: ACTIVITY DIAGRAM FOR MUSIC GENERATION

CHAPTER – 5

IMPLEMENTATION

5.1 SAMPLE CODE

```
import mido
from pynput import keyboard


# Initialize MIDI
mido.set_backend('mido.backends.rtmidi')
output_port = mido.open_output()


# Mapping of keys to MIDI notes for Sa Re Ga Ma classical music
key_to_note = {
    'a': 60,  # Sa
    's': 62,  # Re
    'd': 64,  # Ga
    'f': 65,  # Ma
    'g': 67,  # Pa
    'h': 69,  # Dha
    'j': 71, # Ni
    'k': 72,  # Sa (Higher Octave)
```

```
    'l': 74,
    'z': 76,
}

gm_instruments = [
   "Acoustic Grand Piano",
   "Bright Acoustic Piano",
   "Electric Grand Piano",
   "Honky-tonk Piano",
    "Electric Piano 1 (Rhodes Piano)",
    "Electric Piano 2 (Chorused Rhodes)",
    "Harpsichord",
    "Clavinet",
    "Celesta",
    "Glockenspiel",
    "Music Box",
    "Vibraphone",
    "Marimba",
    "Xylophone",
    "Tubular Bells",
    "Dulcimer",
    "Drawbar Organ",
    "Percussive Organ",
    "Rock Organ",
    "Church Organ",
```

"Reed Organ",

"Accordion",

"Harmonica",

"Tango Accordion",

"Acoustic Guitar (nylon)",

"Acoustic Guitar (steel)",

"Electric Guitar (jazz)",

"Electric Guitar (clean)",

"Electric Guitar (muted)",

"Overdriven Guitar",

"Distortion Guitar",

"Guitar Harmonics",

"Acoustic Bass",

"Electric Bass (finger)",

"Electric Bass (pick)",

"Fretless Bass",

"Slap Bass 1",

"Slap Bass 2",

"Synth Bass 1",

"Synth Bass 2",

"Violin",

"Viola",

"Cello",

"Contrabass",

"Tremolo Strings",

"Pizzicato Strings",

"Orchestral Harp",

"Timpani",

"String Ensemble 1",

"String Ensemble 2",

"SynthStrings 1",

"SynthStrings 2",

"Choir Aahs",

"Voice Oohs",

"Synth Voice",

"Orchestra Hit",

"Trumpet",

"Trombone",

"Tuba",

"Muted Trumpet",

"French Horn",

"Brass Section",

"SynthBrass 1",

"SynthBrass 2",

"Soprano Sax",

"Alto Sax",

"Tenor Sax",

"Baritone Sax",

"Oboe",

"English Horn",

"Bassoon",

"Clarinet",

"Piccolo",

"Flute",

"Recorder",

"Pan Flute",

"Blown Bottle",

"Shakuhachi",

"Whistle",

"Ocarina",

"Lead 1 (square)",

"Lead 2 (sawtooth)",

"Lead 3 (calliope)",

"Lead 4 (chiff)",

"Lead 5 (charang)",

"Lead 6 (voice)",

"Lead 7 (fifths)",

"Lead 8 (bass + lead)",

"Pad 1 (new age)",

"Pad 2 (warm)",

"Pad 3 (polysynth)",

"Pad 4 (choir)",

"Pad 5 (bowed)",

"Pad 6 (metallic)",

"Pad 7 (halo)",

"Pad 8 (sweep)",

"FX 1 (rain)",

"FX 2 (soundtrack)",

"FX 3 (crystal)",

"FX 4 (atmosphere)",

"FX 5 (brightness)",

"FX 6 (goblins)",

"FX 7 (echoes)",

"FX 8 (sci-fi)",

"Sitar",

"Banjo",

"Shamisen",

"Koto",

"Kalimba",

"Bag pipe",

"Fiddle",

"Shanai",

"Tinkle Bell",

"Agogo",

"Steel Drums",

"Woodblock",

"Taiko Drum",

"Melodic Tom",

"Synth Drum",

"Reverse Cymbal",

```
    "Guitar Fret Noise",
    "Breath Noise",
    "Seashore",
    "Bird Tweet",
    "Telephone Ring",
    "Helicopter",
    "Applause",
    "Gunshot"

]

# Function to get a subset of instruments
def get_instrument_subset(start):
    return {str(i % 10): (start + i) for i in range(10)}

# Initialize the current program and subset
current_program = 0
instrument_subset_start = 0

# Function to play a MIDI note with the specified program
def play_midi_note(note, program):
    msg_note_on = mido.Message('note_on', note=note, velocity=64, channel=0)
    msg_program_change = mido.Message('program_change',
program=program, channel=0)
    output_port.send(msg_program_change)
```

```
    output_port.send(msg_note_on)


# Function to handle key press
def on_key_press(key):
    global current_program, instrument_subset_start
    try:
        char_key = key.char.lower()
        if char_key in key_to_note:
            note = key_to_note[char_key]
            play_midi_note(note, current_program)
        elif char_key.isdigit():
            current_program                                                    =
get_instrument_subset(instrument_subset_start)[char_key]
        elif char_key == 'n': # Switch to the next subset of instruments
            instrument_subset_start = (instrument_subset_start + 10) % 128
        elif char_key == 'p':  # Switch to the previous subset of instruments
            instrument_subset_start = (instrument_subset_start - 10) % 128
    except AttributeError:
        pass


# Function to stop playing a MIDI note
def stop_midi_note(note):
    msg_note_off = mido.Message('note_off', note=note, velocity=64, channel=0)
    output_port.send(msg_note_off)
```

```
# Function to handle key release
def on_key_release(key):
    try:
        char_key = key.char.lower()
        if char_key in key_to_note:
            note = key_to_note[char_key]
            stop_midi_note(note)
    except AttributeError:
        pass

# Set up keyboard listener
with keyboard.Listener(on_press=on_key_press, on_release=on_key_release) as
listener:
    try:
        listener.join()
    except KeyboardInterrupt:
        # Close the output port when the program exits
        output_port.close()
```

CHAPTER – 6

TESTING

6.1 TEST CASES

Test Case to check whether the required Software is installed on the systems

Test Case ID:	1
Test Case Name:	Required Software Testing
Purpose:	To check whether the required Software is installed on the systems
Input:	Enter python command
Expected Result:	Should Display the version number for the python
Actual Result:	Displays python version
Failure	If the python environment is not installed, then the Deployment fails

Table 6.1.1 python Installation verification

Test Case to check Program Integration Testing

Test Case ID:	2
Test Case Name:	Programs Integration Testing
Purpose:	To ensure that all the modules work together
Input:	All the modules should be accessed.
Expected Result:	All the modules should be functioning properly.
Actual Result:	All the modules should be functioning properly.
Failure	If any module fails to function properly, the implementation fails.

Table 6.1.2 python Programs Integration Testing

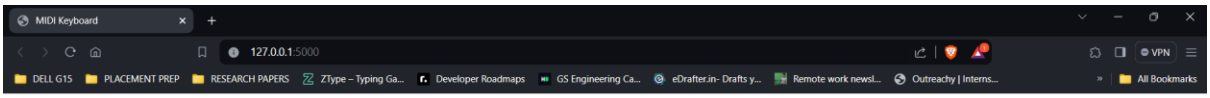
Test Case to check whether the music is generated

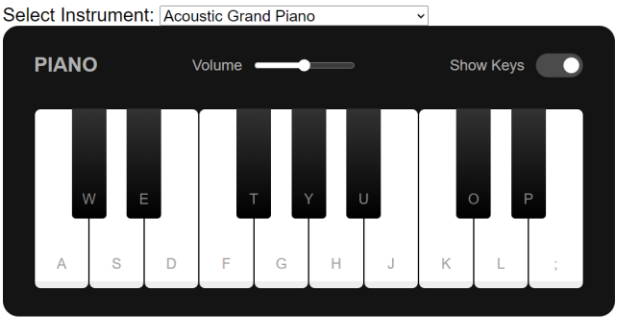
Test Case ID:	3
Test Case Name:	Music Generation
Purpose:	Music Generation using MIDI
Input:	Provide Instrument using ‘i’ input and click the expected keys
Expected Result:	After Evaluation I get the music generated
Actual Result:	After Evaluation I get the music generated
Failure	If the data is not Evaluated, it does not play the music

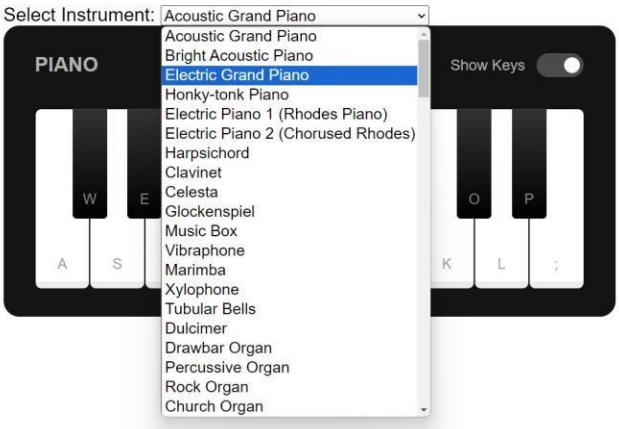
Table 6.1.4 Music Recognition

CHAPTER – 7

SCREENSHOTS







CHAPTER - 8

CONCLUSION AND FUTURE SCOPE

The development of a Python-based musical keyboard system leveraging MIDI and sound synthesis libraries, coupled with a user-friendly GUI, brings forth a seamless and interactive musical experience. The system's robust error handling ensures smooth operation even during rapid or simultaneous key presses. The incorporation of open-source libraries fosters customizability and a diverse range of sound options. The engagement of user feedback further refines the interface, contributing to an enhanced overall user experience.

We're in the process of creating a musical keyboard system using Python. We're utilizing MIDI libraries like mido or python-rtmidi to generate MIDI signals corresponding to different musical notes. To produce audio signals directly from these MIDI notes without complex processing, we've incorporated sound synthesis libraries such as pygame. Each key on the mechanical keyboard is mapped to a specific MIDI note, triggering the corresponding musical sound. Our users interact with a piano-like interface displayed on the screen using a GUI library like HTML, CSS, Javascript. When a key on the physical keyboard is pressed, the corresponding virtual key on the GUI lights up, providing an interactive experience. Upon key press, the mapped MIDI signal is sent to the sound synthesis library, generating the musical note. We've implemented robust error handling to manage rapid or simultaneous key presses, and we actively seek user feedback to refine the user interface and enhance the overall experience.

BIBLIOGRAPHY

[1] C. M. Belinda M J, M. Shyamala Devi, J. A. Pandian, R. Aruna, S. RaviKumar and K. A. Kumar, "Music Note Series Precipitation using Two Stacked Deep Long Short Term Memory Model," 2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 2022, pp. 1-5, doi: 10.1109/ICAECT54875.2022.9807884.

[2] G. A. Restrepo and J. A. G. Herrera, "Seat & Play: A virtual learning environment for harmony," 2011 6th Colombian Computing Congress (CCC), Manizales, Colombia, 2011, pp. 1-5, doi: 10.1109/COLOMCC.2011.5936335

[3] G. G. N. S and V. V. P. D, "Generating Creative Classical Music by Learning and Combining Existing Styles," 2023 4th International Conference on Communication, Computing and Industry 6.0 (C216), Bangalore, India, 2023, pp. 1-7, doi: 10.1109/C2I659362.2023.10431294.

[4] Lassauniere, G. E. Tewkesbury, D. A. Sanders, J. Marchant and A. Close, "A new music technology system to teach music," Proceedings 25th EUROMICRO Conference. Informatics: Theory and Practice for the New Millennium, Milan, Italy,

[5] Ji, "Sakura: A VR musical exploration game with MIDI keyboard in Japanese Zen environment," 2020 IEEE Conference on Games (CoG), Osaka, Japan, 2020, pp. 620-621, doi: 10.1109/CoG47356.2020.9231808.

[6] J. Díaz-Estrada and A. Camarena-Ibarrola, "Automatic evaluation of music students from MIDI data," 2016 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), Ixtapa, Mexico, 2016, pp. 1-6, doi: 10.1109/ROPEC.2016.7830597.

[7] M. Chaudhry, S. Kumar and S. Q. Ganie, "Music Recommendation System through Hand Gestures and Facial Emotions," 2023 6th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2023, pp. 1-7, doi: 10.1109/ISCON57294.2023.10112159.

[8] S. Conan, O. Derrien, M. Aramaki, S. Ystad and R. Kronland-Martinet, "A Synthesis Model With Intuitive Control Capabilities for Rolling Sounds," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 8, pp. 1260-1273, Aug. 2014, doi: 10.1109/TASLP.2014.2327297.

[9] T. Tanaka and Y. Tagami, "Automatic MIDI data making from music WAVE data performed by 2 instruments using blind signal separation," Proceedings of the 41st SICE Annual Conference. SICE 2002., Osaka, Japan, 2002, pp. 451-456 vol.1, doi: 10.1109/SICE.2002.1195442.

APPENDIX: TOOLS AND TECHNOLOGIES

i. PYTHON V3: The Python language comes with many libraries and frameworks

ii. HTML: HTML (Hyper Text Markup Language) is the standard markup language for creating web pages, defining their structure.

iii. CSS: CSS (Cascading Style Sheets) is a stylesheet language used for describing the presentation of a document written in HTML.

iv. JavaScript: JavaScript is a programming language that enables interactive web pages and is essential for web development.

v. Flask: Flask is a lightweight web application framework written in Python, suitable for building small to medium-sized web applications.

vi. MIDO PYGAME: Mido is a Python library for working with MIDI messages, and Pygame is a set of Python modules designed for writing video games.

vii. KEYBOARD: The Python library keyboard provides cross-platform support for simulating keyboard input and listening to keyboard events.

viii. PYNPUT: Pynput is a Python library that allows cross-platform control of the mouse and keyboard, enabling automation and interaction with graphical user interfaces.

ix. PYTHON-RTMIDI: Python-rtmidi is a Python binding for RtMidi, providing a high-level interface for interacting with MIDI devices in Python.

x. WINDOWS 11: Windows 11 was used as the operating system.