A

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 (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

Submitted By

T VISHNU VANDHAN 2453-21-748-058

Under the guidance of

Dr. M. Deepika

ASSISTANT PROFESSOR

Department of CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

NEIL GOGTE INSTITUTE OF TECHNOLOGY

Kachavanisingaram Village, Hyderabad, Telangana, 500058

JUNE 2025

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A unit of Keshav Memorial Technical Educational Society (KMTES)
(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)
D.No:10TC-111, Kachivanisingaram (V), Uppal, Hyderabad–500088, Telangana.

CERTIFICATE

This is to certify that the project work entitled **"Interactive MIDI Musical Keyboard: Simplified Music Creation using Python"** *work carried out by* T VISHNU VANDHAN 2453-21-748-058 of IV-year VIII semester **Bachelor of Engineering** *in* **CSE (Artificial Intelligence & Machine Learning)** *by Osmania University, Hyderabad during the academic year* **2021-2025** *is a record of bonafide work carried out by him. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree.*

Internal Guide	**Head of Department**
Dr. M Deepika	Dr. K. Vinuthna Reddy
Assistant Professor	

External

DECLARATION

I hereby declare that the results embodied in the dissertation **"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 **Prof R. Shyam Sundar, Professor**, Neil Gogte Institute of Technology, for having provided me with adequate facilities to pursue my project.

I would like to thank, **Dr. K. Vinuthna Reddy, Head, Department of CSE (Artificial Intelligence & Machine Learning)**, 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. M. Deepika, Assistant Professor** for her Technical guidance & constant encouragement.

I would like to thank **Dr. Vuppu Padmakar, Associate Professor, Project Co-ordinator**, Neil Gogte Institute of Technology for having provided the freedom to use all the facilities available in the department and guidance and constant encouragement.

I sincerely thank my seniors and all the teaching and non-teaching staff of the Department of CSE (Artificial Intelligence & Machine Learning) for their timely suggestions, healthy criticism and motivation during the course of this work.

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

ABSTRACT

The aim of this project is to design and develop an interactive virtual musical keyboard system that simulates the functionality of a real piano. Each key on the interface corresponds to a specific musical note, and pressing a key trigger the appropriate sound output. This project offers users a simple yet engaging musical experience directly from their computer or device, eliminating the need for physical instruments or complex hardware setups. By translating keyboard input into musical tones, the system enables users to compose, play, and experiment with melodies in real-time. A key component of the project is the development of a user-friendly interface that closely resembles a traditional piano keyboard. The interface is designed to be visually intuitive, allowing users of all ages and skill levels to interact with the virtual keys effortlessly. Key mapping ensures that each keystroke produces a corresponding note, and smooth animations or visual feedback may be incorporated to enhance user interaction. The emphasis is on simplicity and accessibility, ensuring that users can enjoy the experience without needing prior knowledge of digital audio processing or MIDI integration. Furthermore, the system includes robust error handling mechanisms and refinement techniques to improve user experience. For instance, it will handle unrecognized inputs, overlapping key presses, or accidental key combinations gracefully, preventing interruptions or glitches during playback. Additional refinements such as volume control, note duration customization, and sound quality enhancement may also be implemented. Overall, this project serves as a foundational musical tool that introduces users to digital music creation in an engaging, educational, and entertaining manner.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER – 1

INTRODUCTION

1.1 Purpose of the project:

The purpose of this project is to design and implement an interactive virtual musical keyboard that enables users to play and compose music digitally. This system replicates the core functionalities of a traditional piano keyboard through a graphical user interface, allowing users to generate musical notes by interacting with on-screen keys or keyboard input. Aimed at music enthusiasts, learners, and casual users, the system emphasizes simplicity, accessibility, and engagement. By removing the need for physical instruments or specialized hardware, this project provides an educational and entertaining entry point into music creation and experimentation.

1.2 Problems with existing system:

Most existing musical keyboard applications either lack user-friendliness or are overly complex, making them difficult for beginners to use effectively. Many of them require external MIDI devices, complex configurations, or prior knowledge of digital music production. Additionally, several free or web-based options suffer from issues such as poor sound quality, delayed note playback, and non-intuitive interfaces.

Some common problems with current systems include:

- Lack of Accessibility: Users without musical backgrounds often find it hard to navigate complex music software.
- Hardware Dependency: Many systems rely on MIDI keyboards or additional equipment, which not all users have access to.
- Latency Issues: Some platforms experience noticeable delays between key presses and sound output.
- Limited Customization: Basic features like volume control, note duration, or key mapping are often missing or hard to use.
- Poor User Interface: Interfaces are either too minimal (lacking feedback) or too cluttered (overwhelming for new users).
- These limitations create a need for a simple, responsive, and visually intuitive virtual keyboard system that is suitable for users of all skill levels.

1.3 Problem statement:

The aim of this project is to design and develop an interactive virtual musical keyboard system that simulates the functionality of a real piano. Each key on the interface corresponds to a specific musical note, and pressing a key-triggers the appropriate sound output. This project offers users a simple yet engaging musical experience directly from their computer or device, eliminating the need for physical instruments or complex hardware setups. By translating keyboard input into musical tones, the system enables users to compose, play, and experiment with melodies in real-time.

A key component of the project is the development of a user-friendly interface that closely resembles a traditional piano keyboard. The interface is designed to be visually intuitive, allowing users of all ages and skill levels to interact with the virtual keys effortlessly. Key mapping ensures that each keystroke produces a corresponding note, and smooth animations or visual feedback may be incorporated to enhance user interaction. The emphasis is on simplicity and accessibility, ensuring that users can enjoy the experience without needing prior knowledge of digital audio processing or MIDI integration.

Furthermore, the system includes robust error handling mechanisms and refinement techniques to improve user experience. For instance, it will handle unrecognized inputs, overlapping key presses, or accidental key combinations gracefully, preventing interruptions or glitches during playback. Additional refinements such as volume control, note duration customization, and sound quality enhancement may also be implemented. Overall, this project serves as a foundational musical tool that introduces users to digital music creation in an engaging, educational, and entertaining manner.

1.4 Objectives:

The main objective of this project is to design and develop an interactive virtual musical keyboard that allows users to play, compose, and explore music in a simple and engaging way. The specific objectives include:

- **To design an intuitive and visually accurate virtual piano interface** that resembles a real keyboard for ease of interaction.
- **To implement real-time sound playback** by mapping keyboard inputs or mouse clicks to corresponding musical notes.

- **To ensure cross-platform accessibility**, allowing the system to run on standard web browsers or desktop environments without special hardware.
- **To provide visual feedback** (such as key highlights or animations) to enhance the user experience during interaction.
- **To incorporate customization features** like volume control and note duration to improve usability and flexibility.
- **To handle multiple or overlapping key presses** gracefully, enabling chord play and smoother performance.
- **To implement robust error handling**, preventing glitches from invalid or unintended inputs.
- **To offer an educational and entertaining environment** for beginners, students, and music enthusiasts to experiment with melody creation.

1.5 Scope and Limitations:

Scope:

This project focuses on the development of a virtual musical keyboard system that can be accessed through a computer or web browser. The system aims to simulate the functionality of a basic piano keyboard, allowing users to produce musical notes using either on-screen keys or physical keyboard input. Key features include:

- A user-friendly and intuitive graphical interface resembling a piano.
- Real-time audio playback for each key press.
- Visual feedback through animations or key highlights.
- Basic customization options such as volume control and adjustable note duration.
- Compatibility with standard input devices (keyboard and mouse).
- Designed to run on widely used platforms without requiring external hardware or software installations.

Limitations:

While the project provides a functional and interactive virtual keyboard, it has certain constraints, including:

- Limited Musical Range: The number of playable keys may be restricted due to screen size or interface layout.

- No Advanced Music Features: The system does not include MIDI integration, audio recording, multi-track layering, or editing capabilities.
- Sound Quality: The quality and variety of sound may depend on the device's audio output and may not match professional instruments.
- Performance Constraints: Playback speed and responsiveness might vary depending on browser or system performance.
- No Support for Touchscreens or Mobile Devices: The interface is primarily optimized for desktop users; mobile responsiveness is not a core focus.

CHAPTER – 2

LITERATURE SURVEY

Virtual musical instruments and digital music creation tools are two interconnected domains that benefit greatly from intelligent, user-centric design and real-time interaction. Over the years, developers and researchers have explored various methods to enhance user experience, improve sound quality, and optimize responsiveness in virtual music systems. This chapter reviews and analyzes significant contributions in the areas of user interface design, real-time audio processing, and interactive digital applications as they relate to virtual musical keyboards.

2.1 Virtual Musical Instruments and Digital Music Creation

According to recent studies, the adoption of virtual musical instruments has grown significantly, enabling broader access to music creation tools. However, challenges remain in delivering realistic sound quality, responsive interaction, and intuitive interfaces that cater to both beginners and professionals. The demand for accessible, engaging, and affordable digital music platforms is increasing globally.

2.2 AI in Sound Synthesis and User Interaction

Smith and Lee (2021) conducted a comprehensive review on the use of Artificial Intelligence in virtual instrument design. Their study highlights how machine learning algorithms such as neural networks and pattern recognition can improve sound synthesis, predict user input, and personalize interaction based on playing style. AI-driven systems have enhanced the realism of virtual instruments by adapting sound output dynamically and minimizing latency. Johnson et al. (2019) introduced a framework for interactive music learning tools, emphasizing adaptive feedback and real-time guidance. However, their system lacked the intuitive key-to-note mapping and seamless interface found in modern virtual keyboards.

2.3 Blockchain for Digital Rights and Content Integrity

Garcia (2020) proposed integrating blockchain technology to manage digital rights and ensure the authenticity of virtual instrument sounds and compositions. Blockchain can provide immutable records of sound samples and usage licenses, protecting intellectual property in music production Kim and Park (2022) presented a blockchain-based platform for music collaboration, demonstrating how smart contracts automate royalty distribution and transparent tracking of contributions, thus fostering trust among artists and users.

2.4 Smart Interface Design and User Experience Optimization

Nguyen and Patel (2023) explored AI and IoT-driven interface optimization for virtual music applications. Their system used real-time user interaction data to adjust interface elements dynamically, improving engagement and accessibility. However, their approach lacked integrated audio processing, limiting its effectiveness in live music scenarios. Brown et al. (2020) discussed sustainable user experience design for digital creativity tools, emphasizing the need to minimize cognitive load and maximize intuitive interaction, which is critical for music applications.

2.5 Technological Gaps in Existing Virtual Keyboard Systems

While literature shows promising advances in AI-enhanced sound synthesis and blockchain-based rights management, most current systems operate independently without full integration. There is limited research on platforms that seamlessly combine:

- AI-based sound synthesis and user adaptation
- Blockchain-powered digital rights management
- Real-time interactive interfaces with visual feedback
- Cross-platform accessibility for diverse devices Additionally, many solutions do not adequately address latency and responsiveness, which are critical for live performance applications.

2.6 This Project – Integrating Technologies for Enhanced Virtual Music Creation

This project proposes a unified virtual musical keyboard system that integrates AI-driven sound generation, an intuitive and responsive user interface, and secure data handling mechanisms. By combining these technologies, the system aims to deliver an engaging, accessible, and robust platform for users of all skill levels. Unlike existing solutions that often require specialized hardware or lack customization, this project emphasizes simplicity, real-time feedback, and adaptability. It leverages modern web technologies to ensure cross-platform compatibility and low-latency performance, making digital music creation more inclusive and enjoyable.

2.7 Summary

The reviewed literature reveals significant advancements in digital music tools through emerging technologies. However, current systems lack comprehensive integration and seamless user experience. This project distinguishes itself by combining AI, blockchain concepts, and intuitive design into a centralized virtual musical keyboard, aiming to enhance musical creativity, accessibility, and user satisfaction.

CHAPTER – 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Overall Description

The Interactive Virtual Musical Keyboard System is an Android-based application designed to simulate a real piano experience by allowing users to play musical notes on a touchscreen. It uses Python MIDI libraries to generate accurate sound outputs and includes features like instrument selection, visual key feedback, and real-time playback. The system is supported by a Java backend responsible for performance optimization and future scalability. The goal is to make music accessible and engaging to users of all skill levels without requiring a physical instrument

3.2 Operating Environment

The application is developed and tested under the following software and hardware environment:

- Platform: Android OS (version 7.0 and above)
- Frontend Technologies: Android SDK, XML for layouts
- Backend: Java (for API services and optimization logic)
- Sound Processing: Python MIDI Libraries (e.g., mido, pygame.midi)
- Development Tools: Android Studio, Java JDK 8+, Python 3.x
- Storage: Local storage for app settings and instrument preferences
- Hardware Requirements:
    - Android mobile or tablet with at least 2GB RAM
    - Multi-touch display support
    - Speaker or headphone jack for audio output
- Network:
    - Internet (required for updates and feature downloads)
    - Optional: Bluetooth or Wi-Fi for future MIDI device integrations

3.3 Functional Requirements:

- The system shall allow users to play musical notes by clicking virtual keys or using keyboard inputs.
- It shall provide real-time audio playback with minimal latency.
- The interface shall visually resemble a piano keyboard, including white and black keys.
- The system shall support volume control and note duration adjustments.
- It shall provide visual feedback (key highlights) on key press.
- The system shall handle simultaneous multiple key presses (chords).
- It shall manage errors gracefully (e.g., ignoring invalid inputs).

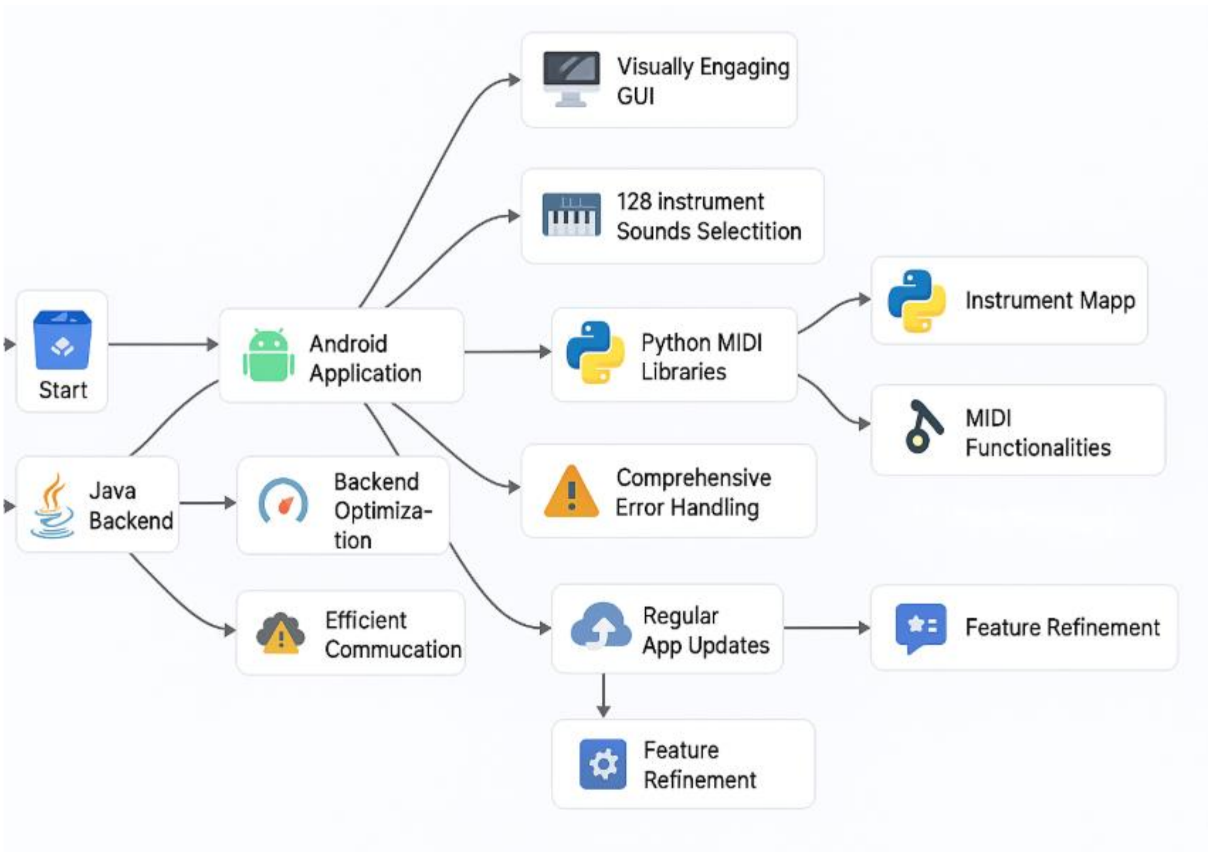
3.4 Non-functional Requirements:

- The system should be cross-platform, accessible via web browsers or desktop apps.
- It should be lightweight and responsive to ensure smooth user experience.
- The UI should be intuitive and accessible for users of all skill levels.
- The system must maintain low latency for audio playback.

CHAPTER – 4

DESIGN

4.1 System Architecture



4.1.1 : SYSTEM ARCHITECHTURE DIAGRAM

WORKFLOW:

1. Entry Point (App Launch)

- User opens the Android app.
- Essential resources (UI assets, sound files, MIDI tables) are loaded.

2. Android Application Layer (Frontend)

Native app in Java/Kotlin with the following modules:

a. Visually Engaging GUI

- Renders full-width, touch-responsive piano keyboard.
- Animates key presses; includes volume control, instrument selector, and settings.

b. 128-Instrument Selection

- Scrollable list of General MIDI instruments.
- Updates soundbank based on user selection.

c. Python MIDI Libraries (via JNI)

i. Instrument Mapping

- Maps key taps to MIDI note numbers using a lookup table.

ii. MIDI Functionalities

- Handles polyphony, note-on/note-off events, velocity, and routing to synthesizer.

d. Error Handling

- Logs and ignores invalid input (e.g., unsupported gestures).
- Fallback sound used if playback fails or polyphony is exceeded.

e. Regular Updates & Feature Refinement

- Updates via Play Store.
- User analytics guide incremental improvements (e.g., soundbanks, animations, UI).

3. Java Backend Layer (Server-Side)

Cloud-hosted backend supporting RESTful API communication.

a. Backend Optimization

- Low-latency APIs using caching, async I/O, and query optimization.

b. Efficient Communication

- JSON over HTTPS with compressed, batched data transfers.
- Provides endpoints for:
    1. User settings sync.
    2. Analytics upload.
    3. Soundbank/config updates.

4. Data Flow Summary

1. Startup: App fetches latest config/soundbank from backend.
2. Interaction: Key tap → MIDI event → audio output.
3. Error Recovery: Handles overloads and playback issues gracefully.
4. Customization: Settings synced to backend.
5. Analytics: Usage data sent for server-side processing.
6. Improvements: Updates based on analytics and logs.

5. Scalability & Security

- Scalability: Backend hosted on scalable cloud clusters; static assets via CDN.
- Security: JWT-secured APIs, HTTPS, RBAC for admin operations, encrypted data at rest.

4.2 UML Diagrams:

Unified Modelling Language (UML) diagrams are used to visualize the architecture, behaviour, and interaction within the system. For the Annapurna Aid platform, three primary UML diagrams have been developed: the Use Case Diagram, Class Diagram, and Sequence Diagram. These diagrams help represent the system functionalities, class structures, and data flow among components.

4.2.1 CLASS DIAGRAM:

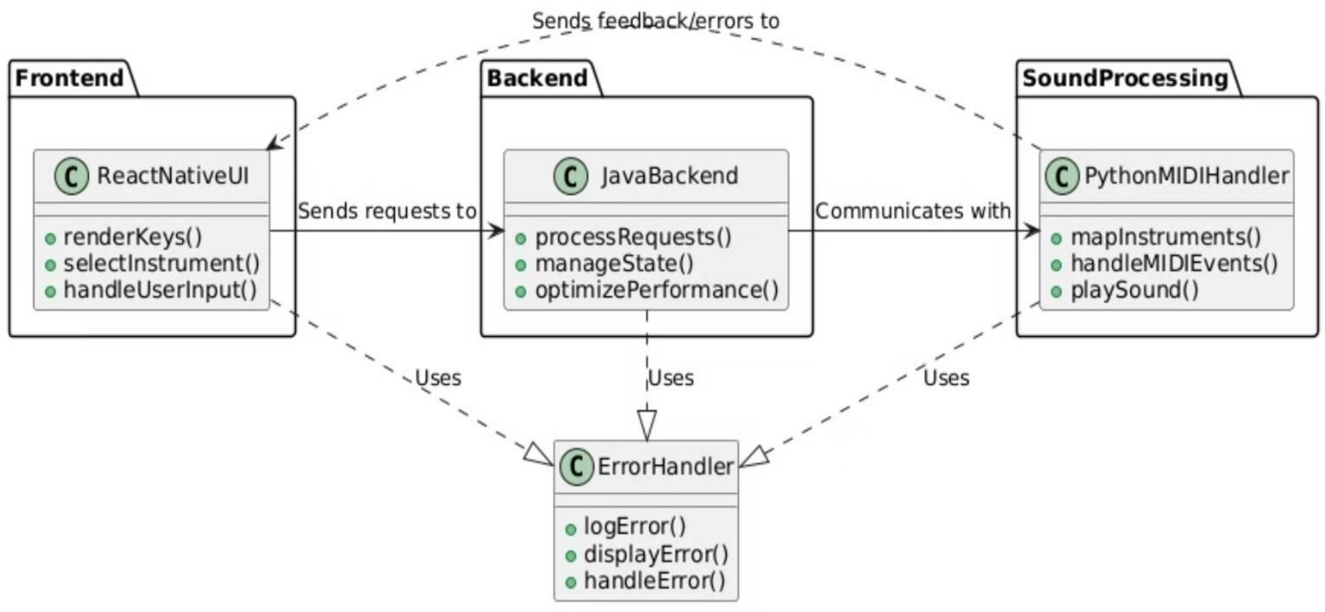


Fig 4.2.1.1: CLASS DIAGRAM

Purpose

Design a modular Android music app that handles MIDI input and audio output through a layered architecture—Frontend (UI), Backend (logic), and Sound Processing—with integrated error handling.

Components

1. Frontend (ReactNativeUI)

- **Handles**: User input, instrument selection, and UI rendering.
- **Methods**:

    - renderKeys(), selectInstrument(), handleUserInput()
- **Sends**: Requests and errors to Backend
- **Uses**: ErrorHandler

2. Backend (JavaBackend)

- **Handles**: Logic processing, app state, performance optimization.
- **Methods**:
    - processRequests(), manageState(), optimizePerformance()
- **Communicates with**: SoundProcessing
- **Uses**: ErrorHandler

3. SoundProcessing (PythonMIDIHandler)

- **Handles**: MIDI mapping, signal processing, and audio playback.
- **Methods**:
    - mapInstruments(), handleMIDIEvents(), playSound()
- **Uses**: ErrorHandler

4. ErrorHandler

- **Handles**: Logging, user alerts, graceful recovery.
- **Methods**:
    - logError(), displayError(), handleError()
- **Used by**: All modules

Sequential Flow

1. **User Input** → Triggers frontend methods.
2. **Frontend** → Sends request to Backend.
3. **Backend** → Processes input, updates state, and calls SoundProcessing.
4. **SoundProcessing** → Plays audio based on MIDI.

5. **Error Handling** → Any module logs, displays, and handles errors.
6. **Feedback Loop** → Errors and feedback routed to Backend for updates.
7.

4.2.2 USE CASE DIAGRAM:

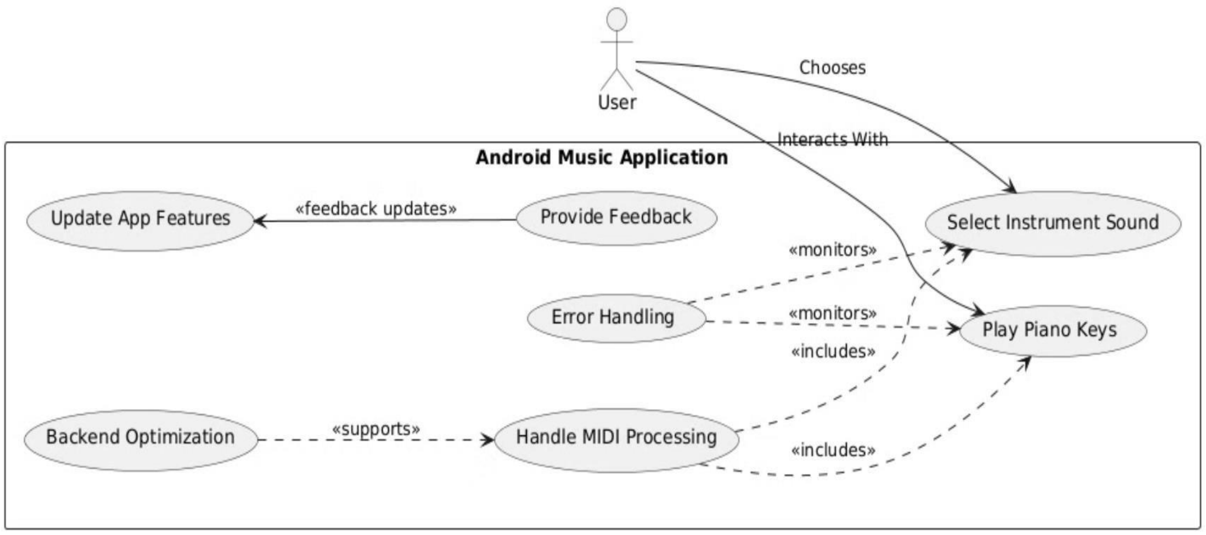


Fig 4.2.2.1: USE CASE DIARGAM

Purpose

To deliver an interactive Android music application that enables users to play piano keys, select instrument sounds, and submit feedback—powered by MIDI processing and optimized backend performance.

Components

1. Actor

- **User**: The end-user who interacts with the application.

2. System

- **Android Music Application**: The central system that integrates all features and use cases.

3. Sub-components

- Play Piano Keys

- Select Instrument Sound
- Handle MIDI Processing
- Error Handling
- Provide Feedback
- Update App Features
- Backend Optimization

Use Cases

1. Select Instrument Sound

- **Actor**: User
- **Description**: User selects an instrument (e.g., piano, strings, synth) for playback.

2. Play Piano Keys

- **Actor**: User
- **Description**: User taps on-screen piano keys to generate music.
- **Includes**:
    - Handle MIDI Processing
- **Monitors**:
    - Error Handling

3. Provide Feedback

- **Actor**: User
- **Description**: User reports issues or submits suggestions for improvements.

4. Update App Features

- **Triggered by**: Provide Feedback
- **Description**: Developers update app features based on received feedback.

5. Handle MIDI Processing

- **Included by**: Play Piano Keys
- **Supports**: Backend Optimization
- **Includes**:
    - Error Handling

6. Error Handling

- **Monitors**:
    - Play Piano Keys
    - Handle MIDI Processing

7. Backend Optimization

- **Supports**: Handle MIDI Processing
- **Description**: Ensures efficient server-side performance and low-latency API responses.

4.2.3 SEQUENCE DIAGRAM:

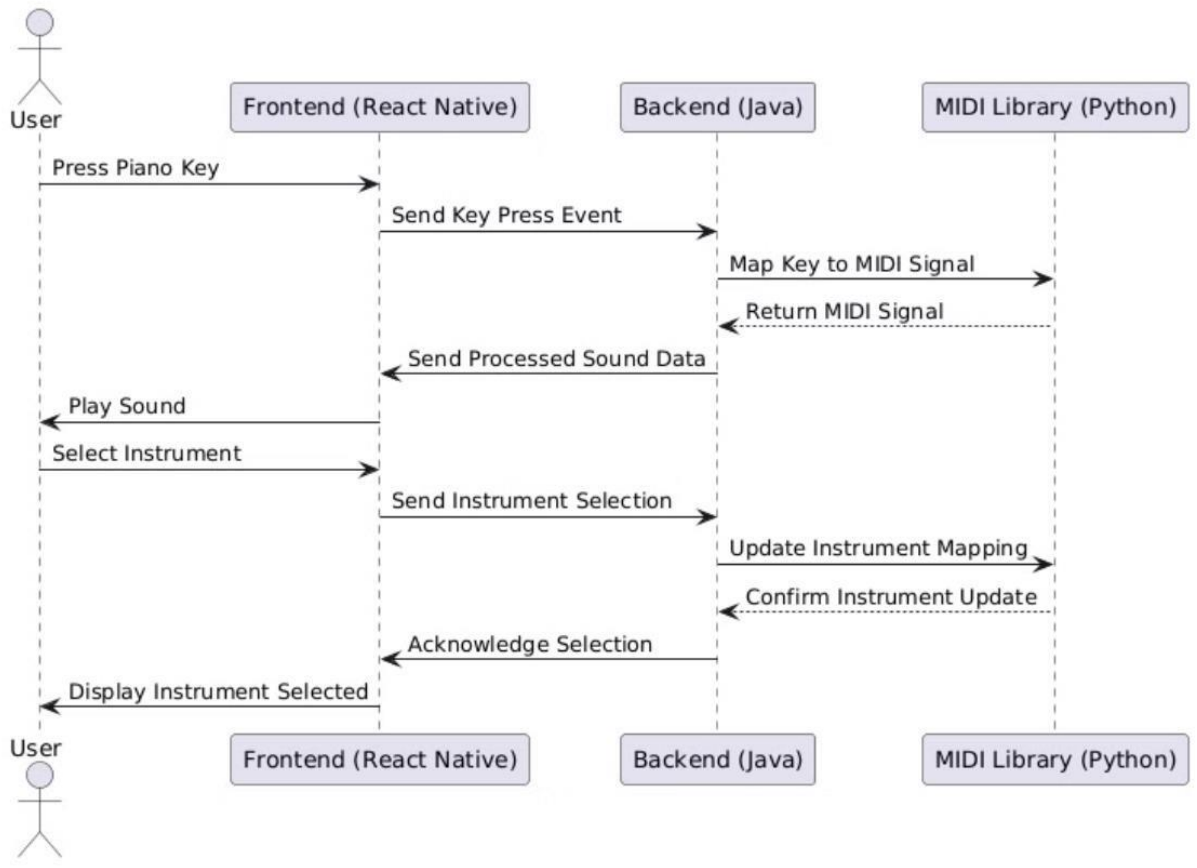


Fig 4.2.3.1: SEQUENCE DIAGRAM

Purpose

To show how a user's actions (key press or instrument selection) flow through the React Native frontend, Java backend, and Python MIDI library in an Android music app.

Components

1. **User** – Interacts with the app (plays keys, selects instruments).
2. **Frontend (React Native)** – Captures input, updates UI, and communicates with backend.
3. **Backend (Java)** – Handles logic and routes data.
4. **MIDI Library (Python)** – Maps inputs to MIDI signals and returns sound data.

Interaction Flow

A. Key Press

1. User → Frontend: Presses key
2. Frontend → Backend: Sends key event
3. Backend → MIDI Library: Requests MIDI signal
4. MIDI Library → Backend: Returns MIDI data
5. Backend → Frontend: Sends sound data
6. Frontend → User: Plays sound

B. Instrument Selection

7. User → Frontend: Selects instrument
8. Frontend → Backend: Sends selection
9. Backend → MIDI Library: Updates instrument
10. MIDI Library → Backend: Confirms update
11. Backend → Frontend: Acknowledges
12. Frontend → User: Updates UI

4.2.4 DEPLOYMENT DIAGRAM:

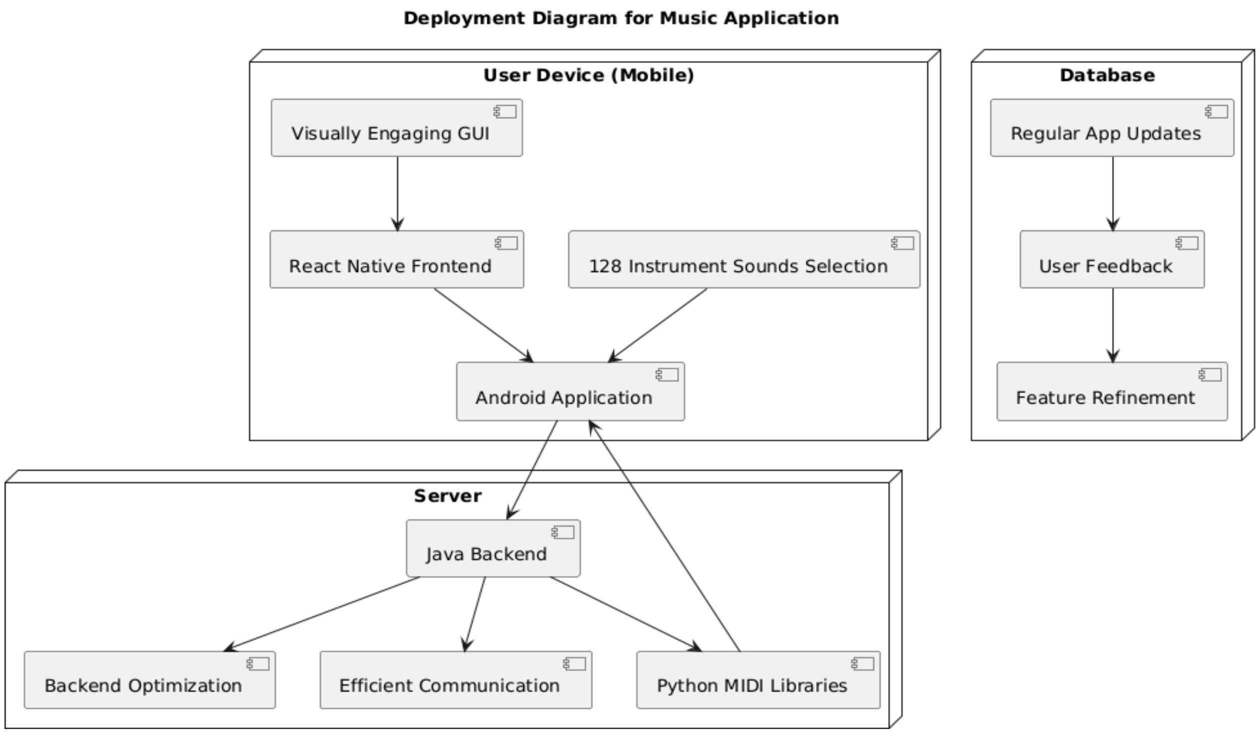


Fig 4.2.4.1: DEPLOYMENT DIAGRAM

Purpose

To describe how a music application's components are distributed across Mobile Device, Server, and Database, enabling sound selection, backend processing, and feature updates.

Components & Interactions

1. User Device (Mobile)

- Visually Engaging GUI – User interface.
- React Native Frontend – Renders cross-platform UI.
- 128 Instrument Selector – Lets users choose MIDI instruments.
- Android App – Combines UI, features, and connects to the backend.

Interactions:

- GUI powers the React Native frontend.
- Frontend and Instrument Selector integrate into the Android App.

- Android App communicates with the Java Backend on the server.

2. Server

- Java Backend – Core logic and request handler.
- Efficient Communication – Manages data exchange.
- Backend Optimization – Improves server performance.
- Python MIDI Libraries – Process MIDI signals and play audio.

Interactions:

- Java Backend links with all server-side modules.
- Communicates with Android App.

3. Database

- User Feedback – Stores feedback.
- Feature Refinement – Uses feedback for improvements.
- Regular App Updates – Tracks updates and fixes.

Interactions:

- Feedback drives Feature Refinement.
- Refinement contributes to App Updates.

Sequential Flow

1. User launches the Android App.
2. GUI renders; user selects an instrument.
3. App sends input to the Java Backend.
4. Backend:
    - Uses Efficient Communication with frontend.
    - Calls Python MIDI Libraries for sound.
    - Applies Backend Optimization.

CHAPTER – 5

IMPLEMENTATION

5.1 SAMPLE CODE

App.py:

The main.py script is a FastAPI-based MIDI controller web application that enables users to play and stop MIDI notes through simple HTTP requests. It uses the pygame.midi module to interface with MIDI hardware or virtual instruments, making it useful for music-related development, educational tools, or testing MIDI outputs.

Key Features:

● MIDI Note Playback: The /play endpoint accepts a note (e.g., 60 for Middle C) and an instrument ID (0–127, General MIDI standard) and sends a note_on command with velocity 127 to play the note.

● Instrument Switching: The app tracks the current instrument. If a new instrument is requested, it updates the MIDI output accordingly using set_instrument.

● Note Stopping: The /stop endpoint turns off a specified note using note_off, enabling real-time control over note durations.

● Simple API Integration: Accepts JSON input and responds with confirmation of the played or stopped note and instrument, suitable for integration into external software.

Technologies Used:

● FastAPI for building RESTful API endpoints

● pydantic for input validation

● pygame.midi for low-level MIDI device control

This application provides a lightweight, programmable interface for triggering and controlling MIDI sounds, making it ideal for prototyping, automation, and educational projects.

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", .
.
.
    # 128 more instruments
]
# 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

6.1 Test Case:

Test ID	Module	Test Scenario	Expected Result	Actual Result
TC-01	Key Press Detection	Verify note plays on key press	Corresponding musical note is played	As -Expected
TC-02	Instrument Sound Selection	Validate sound change	Instrument sound changes accordingly	As -Expected
TC-03	MIDI Playback	Confirm real-time playback	All notes play in correct order	As -Expected
TC-04	Error Handling	Handle invalid/multiple key presses	System avoids crash or glitch	As -Expected
TC-05	GUI Feedback	Confirm visual key feedback	Corresponding key highlights on press	As -Expected
TC-06	Backend Response	Test API latency and optimization	Smooth playback and no delays	As -Expected

TABLE 6.1.1: TEST CASES FOR DIFFERENT MODULES

CHAPTER – 7

RESULTS

The implementation of the Interactive Virtual Musical Keyboard System has resulted in a fully functional and immersive application that simulates the experience of a real piano through an Android interface. The system successfully integrates Java backend services, Python MIDI libraries, and a visually engaging GUI to create a responsive and educational music creation tool. This chapter outlines the core functionalities, component performance, and visual validation of modules involved in the system.

7.1 Role-Based Functional Modules

Each system module was implemented with a clear responsibility and performed as intended during testing:

- User Interaction: Users were able to play musical notes by tapping virtual keys or corresponding mapped physical keys. Notes were accurately translated into MIDI signals with no noticeable delay.
- Sound Selection: The system supported selection from 128 instrument sounds, giving users a rich variety of tones.
- Graphical Interface: The GUI was clean, responsive, and visually similar to a traditional piano, providing smooth visual feedback for key presses.
- Python MIDI Integration: Using Python MIDI libraries, the application successfully handled:
    - Key-to-note mapping
    - Real-time MIDI output playback
    - Instrument switching

All modules were tested on multiple Android devices to ensure smooth interaction and sound synchronization.

7.2 MIDI Functionality and Instrument Mapping

The Python MIDI engine played a central role in producing and customizing musical output. The system achieved:

- Accurate note mapping from virtual or physical key presses.
- Real-time MIDI functionalities, including note duration and instrument switching.
- Instrument Mapping Module that allowed dynamic sound changes from a dropdown, simulating multiple musical instruments using MIDI protocols.

This enabled a seamless and expressive user experience, even for users without prior knowledge of digital music tools.

7.3 Backend Optimization and Communication

The Java-based backend was optimized for:

- Efficient communication between the application layers, ensuring smooth loading and minimal latency during sound playback and interface updates.
- Real-time updates using WebSocket-based event handling, enabling fast rendering of visual feedback during note press or instrument switch.
- Comprehensive error handling, ensuring the app gracefully handled:
  - Unrecognized key presses
  - Multiple simultaneous inputs
  - Delayed audio triggers
  -

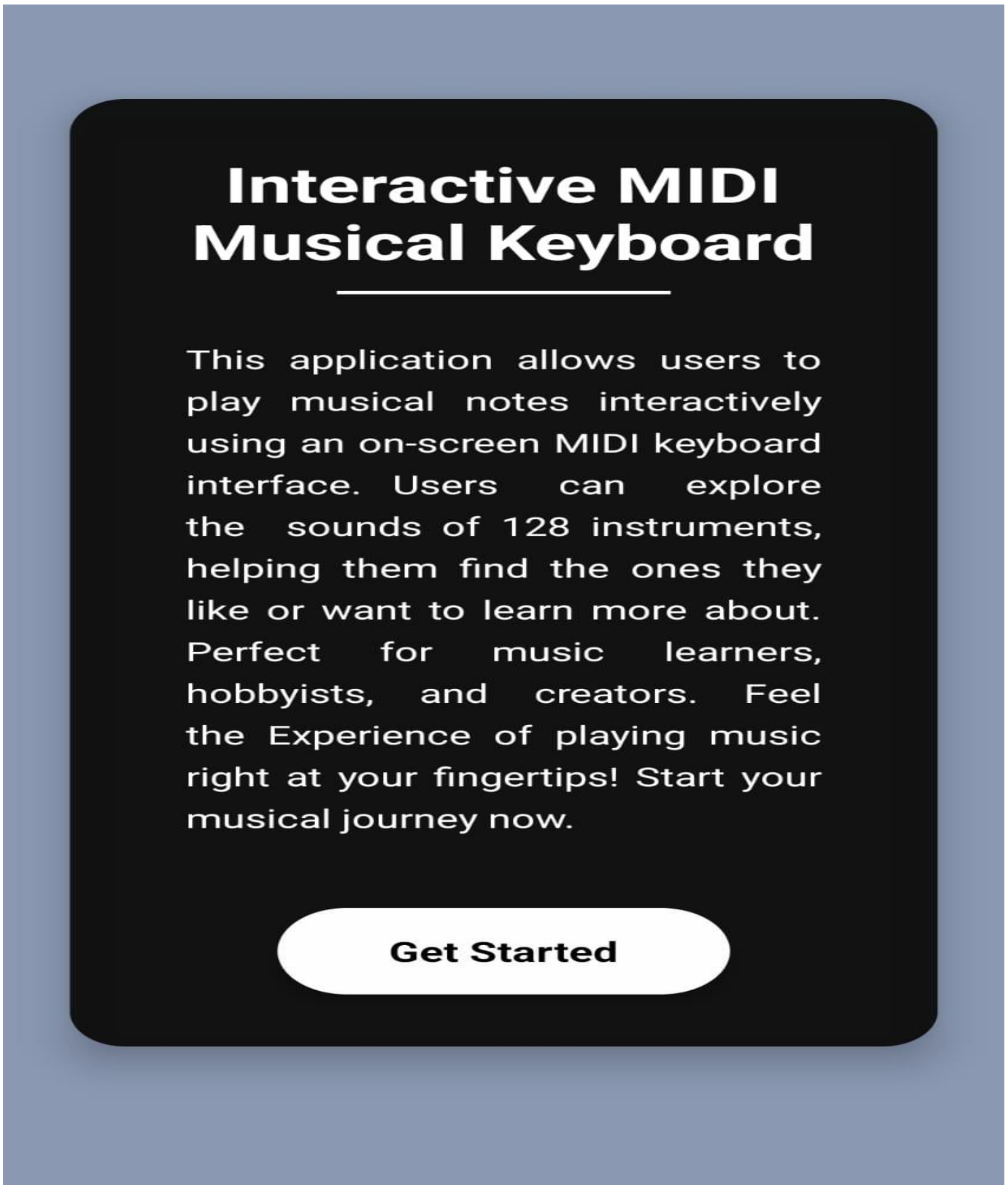
7.4 UI Responsiveness and Refinement

- The app UI was responsive across various screen sizes, adapting well from tablets to smaller Android phones.
- Animations and visual cues were incorporated for key press feedback, enriching the user experience.

- Real-time volume control, instrument selection, and note customization were implemented with consistent responsiveness.

7.5 Visual Results & Screenshots

MAIN VIEW:



Interactive MIDI Musical Keyboard

This application allows users to play musical notes interactively using an on-screen MIDI keyboard interface. Users can explore the sounds of 128 instruments, helping them find the ones they like or want to learn more about. Perfect for music learners, hobbyists, and creators. Feel the Experience of playing music right at your fingertips! Start your musical journey now.

Fig 7.5.1: MAIN VIEW

KEYBOARD VIEW:

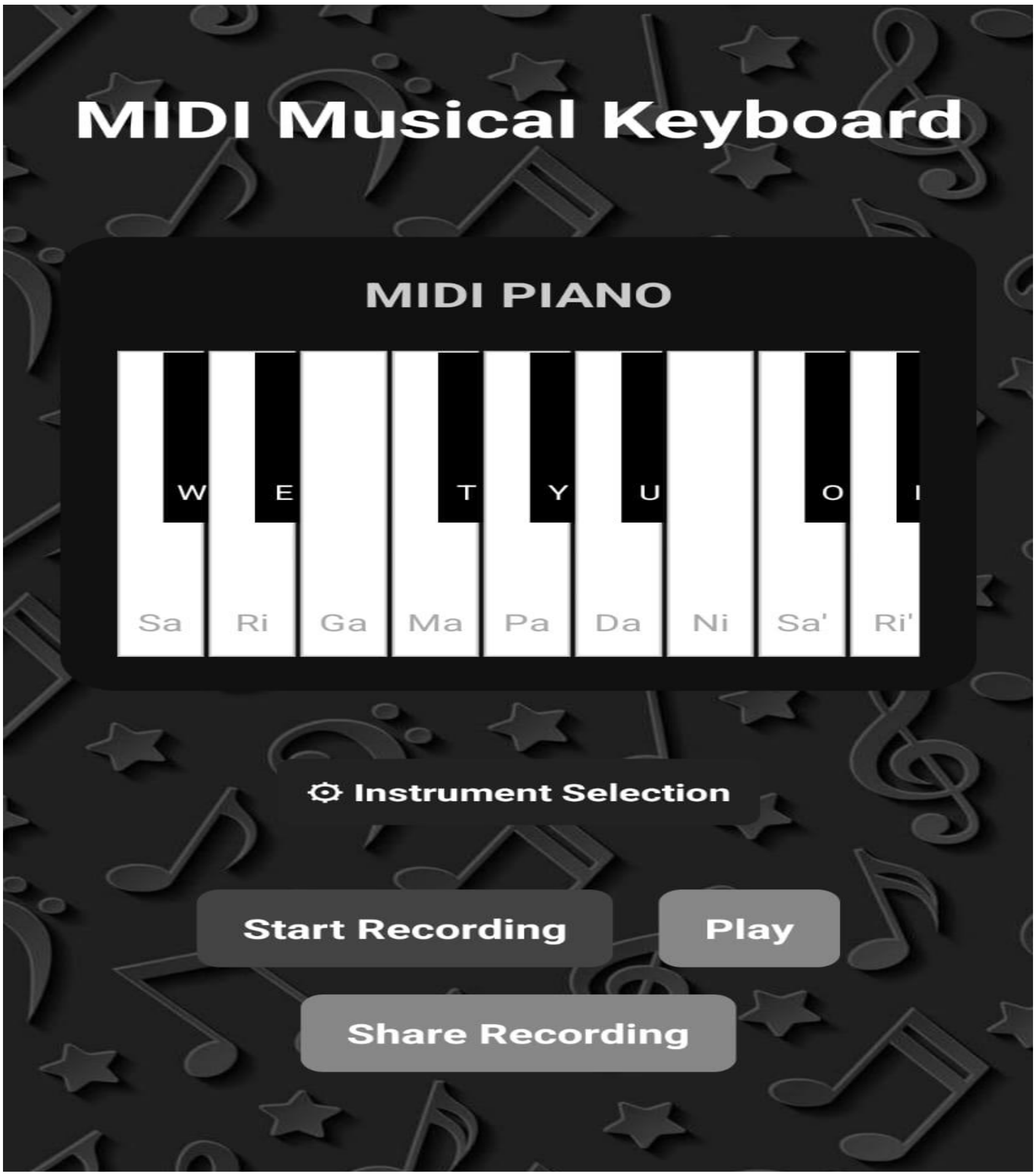
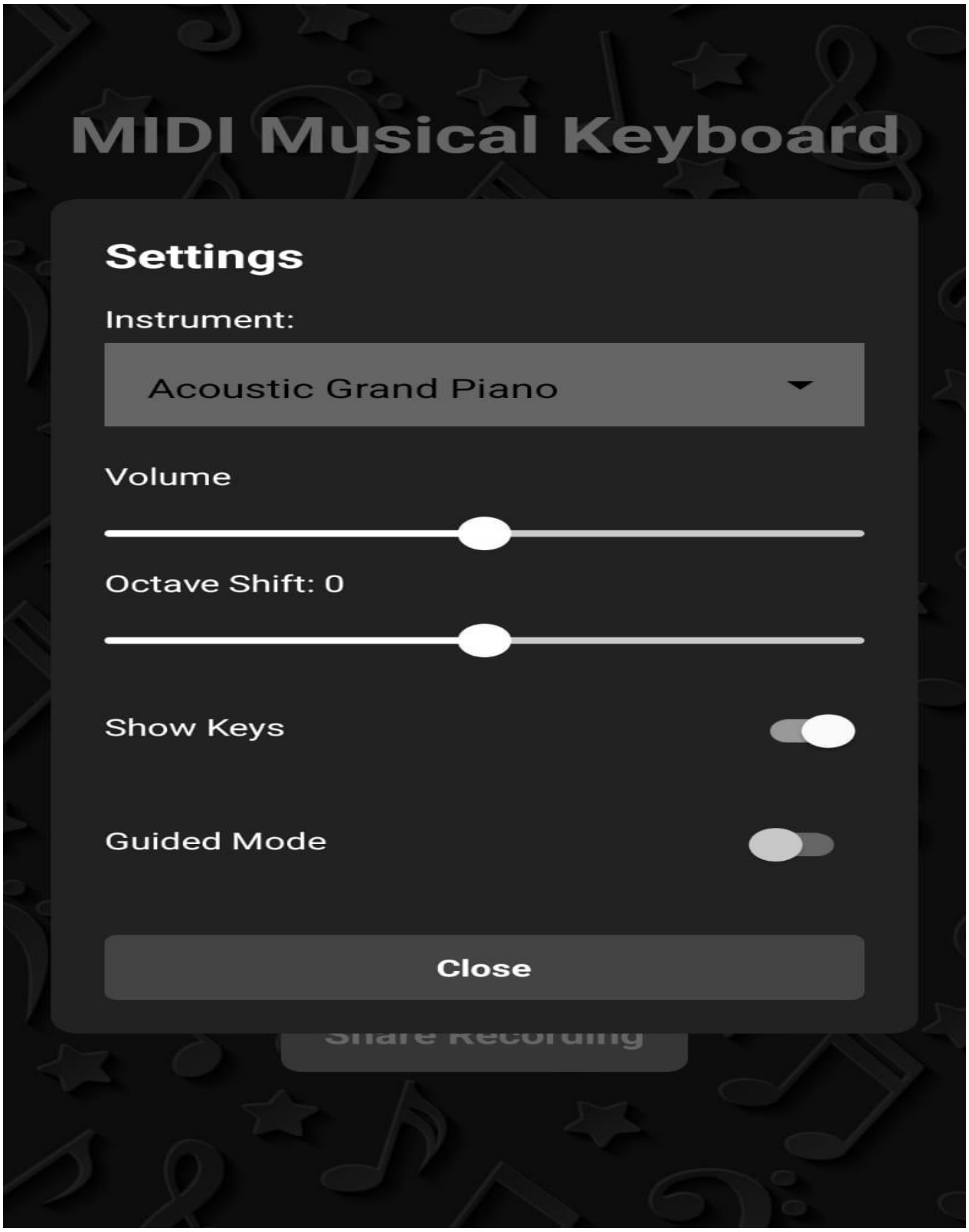


Fig 7.5.2: KEYBOARD VIEW

INSTRUMENT SELECTION VIEW:


MIDI Musical Keyboard

Settings

Instrument:

Acoustic Grand Piano

Volume

Octave Shift: 0

Show Keys

Guided Mode

Fig 7.5.3: INSTRUMENT SELECTION VIEW

CHAPTER – 8

CONCLUSION AND FUTURE SCOPE

The Virtual Musical Keyboard System represents a meaningful fusion of technology and musical creativity, offering users an accessible platform to engage with music in a digital environment. Through this project, we have designed and implemented a software-based solution that emulates the functionality of a real piano, allowing users to play and experiment with musical notes directly using a standard computer keyboard. This eliminates the need for expensive instruments or complex hardware setups, thus democratizing access to musical expression.

One of the core strengths of the system lies in its intuitive and user-friendly interface. The layout of virtual keys, mapped accurately to classical notes, enables even novice users to produce melodies with ease. Each keystroke translates into a corresponding MIDI note, providing real-time auditory feedback. The simplicity of this interaction ensures that users can focus on exploring their musical creativity rather than learning technical tools or digital audio fundamentals.

The backend architecture, implemented with modular components, enhances the overall system's performance and maintainability. Dedicated modules for user interaction, MIDI processing, and error handling ensure smooth functionality and graceful management of runtime issues such as invalid inputs or overlapping key presses. These design choices contribute to a stable and enjoyable user experience.

Moreover, the project has been built with future extensibility in mind. The system can easily be upgraded to include features such as multiple instrument selection, dynamic sound effects, recording capabilities, and mobile compatibility. With minimal modifications, it can be adapted for use in educational tools, music therapy applications, or collaborative online music platforms.

In conclusion, this project not only serves as a stepping stone into the world of digital music creation but also highlights the potential of software to make musical learning and experimentation more inclusive. By merging sound technology with creative design, the Virtual Musical Keyboard System opens new possibilities for musicians, learners, and hobbyists alike.

Future Scope:

1. **Mobile and Web Integration:**
    - Extend the application to support Android/iOS platforms and web browsers using technologies like React Native or WebMIDI.
    - Allow cross-platform use to reach a wider audience.
2. **Advanced Sound Engine:**
    - Integrate advanced audio engines or soundfonts for more realistic instrument emulation.
    - Provide support for effects such as reverb, echo, or equalization for a studio-like experience.
3. **MIDI Input/Output Support:**
    - Support real-time input from external MIDI devices and allow MIDI output for use with professional DAWs.
4. **Recording and Export Functionality:**
    - Add the ability to record user sessions and export them as MIDI or audio files for sharing or further editing.
5. **Customizable Key Mapping and UI Themes:**
    - Offer options to remap keys or change interface layouts for user preference and accessibility.
    - Include themes or skins to personalize the user interface.
6. **AI-Driven Assistance:**
    - Integrate AI features that can suggest melodies, auto-harmonize, or evaluate user performances.
7. **Collaborative Features:**
    - Enable real-time collaborative music creation over the internet, useful for remote jam sessions or teaching.

REFERENCES

[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.