End-Term Project Report

Mobile Application Development



Faculty name: Ms. Aarti Sangwan Student name: Aniket Kumar Yadav (21csu487)

Lokesh Yadav (21csu504)

Ravi (21csu525)

Nadeem (21csu515)

Mayank (21csu506)

Semester: 6th

Group: FS-B

**Department of Computer Science and Engineering**

**The NorthCap University, Gurugram- 122001, India**

**Session 2023-24**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No** |  | **Page No.** |
| **1.** | **Project Description** | **2** |
| **2.** | **Problem Statement** | **2** |
| **3.** | **Analysis**  **3.1 Hardware Requirements**  **3.2 Software Requirements** | **2** |
| **4.** | **Design**  **4.1 Data/Input Output Description:**  **4.2 Algorithmic Approach / Algorithm / DFD / ER diagram/Program Steps** | **3** |
| **5.** | **Implementation and Testing (stage/module wise)** | **4** |
| **6.** | **Output (Screenshots)** | **5** |
| **7.** | **Conclusion and Future Scope** | **8** |

1. **Project Description**

Melody Maker is a cutting-edge music production application developed using Flutter, aimed at empowering users to create, edit, and produce their own music tracks seamlessly. The app caters to both amateur musicians and seasoned producers by providing a user-friendly interface coupled with powerful audio processing capabilities.

1. **Problem Statement**

In the current digital era, music production has become more accessible, yet there remains a gap in providing a comprehensive, user-friendly mobile application that caters to both beginners and professionals. Existing solutions often fall short in offering a seamless integration of recording, editing, and producing music, especially on mobile platforms.

**3. Analysis**

**3.1 Hardware Requirements**

* Processor: Intel Core i5 or higher
* RAM: 8 GB or higher
* Storage: 50 GB of free space
* Smartphone: Android 6.0 or higher / iOS 11.0 or higher
* Internet Connection: Stable broadband connection

**3.2 Software Requirements**

* Operating System: Windows 10 or higher / macOS / Linux
* Flutter SDK
* Dart Programming Language
* Integrated Development Environment (IDE): Android Studio / Visual Studio Code
* Firebase (for backend services)
* API Services: RESTful APIs for inventory management
* Version Control: Git

**4. Design**

**4.1 Data/Input Output Description:**

* **Product Information & Attributes:** Takes Audio as input and give Processed audio as output. Recording new Track, Applying Effect and exporting effects and play music.

**4.2 Algorithmic Approach / Algorithm / DFD / ER diagram/Program Steps**

Developing Melody Maker involves several key algorithmic approaches to handle audio processing, state management, user interaction, and integration with external libraries. Here is a detailed breakdown of the main algorithmic strategies used in the app:

**Audio Capture Algorithm:**

* **Initialize Microphone Input:** Set up the microphone to capture audio input with specified sample rates and bit depth.

**Audio Effects Application:**

* **Effect Chain Processing:** Create a chain of audio effects that can be applied sequentially to the audio signal.
* **Real-time Processing:** Implement real-time processing algorithms to ensure that effects are applied instantaneously without noticeable delay.

**State Management Architecture:**

* **Provider/Riverpod:** Use state management libraries like Provider to manage the app's state efficiently.

**Responsive UI:**

* **Adaptive Layouts:** Design adaptive layouts that adjust to different screen sizes and orientations.

**5. Implementation and Testing (stage/module wise)**

#### 1. Setting Up the Development Environment

* **Install Flutter SDK:** Ensure the latest version of Flutter is installed.
* **Set Up IDE:** Configure an Integrated Development Environment (IDE) like Android Studio or Visual Studio Code.
* **Dependencies:** Install necessary packages for audio processing, state management, and UI components (e.g., **audioplayers**, **provider**, **flutter\_audio\_recorder**).

#### 2. Core Functionalities Implementation

* **Audio Recording Module:**
  + Initialize microphone input.
  + Implement audio buffering and storage.
  + Develop compression algorithms if necessary.
* **Audio Effects Module:**
  + Implement a chain processing system for applying multiple effects.
  + Develop real-time audio effects such as reverb, delay, and equalization.
  + Create dynamic parameter control for adjusting effects.
* **Looping and Samples Module:**
  + Implement tempo matching and seamless looping algorithms.
  + Develop time-stretching and pitch-shifting functionalities.

#### 3. User Interface Development

* **Design Layout:**
  + Develop adaptive layouts to support different screen sizes and orientations.
  + Implement intuitive touch controls for recording, editing, and mixing.
* **Interactive Elements:**
  + Implement gesture handling for dragging, zooming, and adjusting controls.
  + Develop visual elements like waveforms, timelines, and effect visualizations.

#### 4. State Management

* **Implement State Management Architecture:**
  + Use Provider or Riverpod for managing global and local states.
  + Ensure efficient state updates and reactivity for real-time audio processing.

#### 5. Integration with External Libraries

* **API Integration:**
  + Implement RESTful API calls to fetch audio samples and effects from external libraries.
  + Develop caching mechanisms to store frequently used data locally.

### Testing Stage

#### 1. Unit Testing

* **Core Functionality Tests:**
  + Write unit tests for audio recording, processing, and effects application.
  + Ensure each component functions correctly in isolation.
* **State Management Tests:**
  + Test state transitions and ensure the state is managed efficiently.
  + Verify undo/redo functionality.

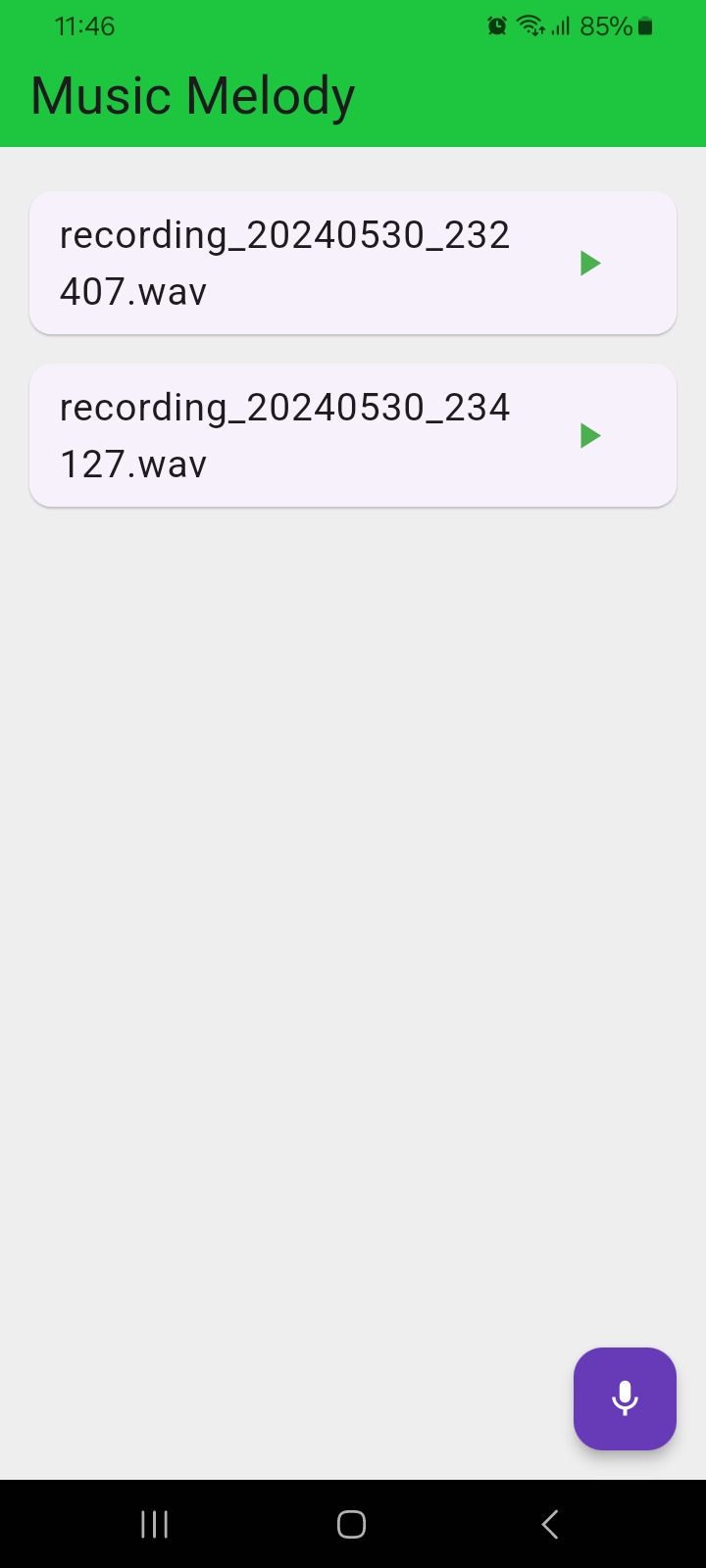
#### 2. Integration Testing

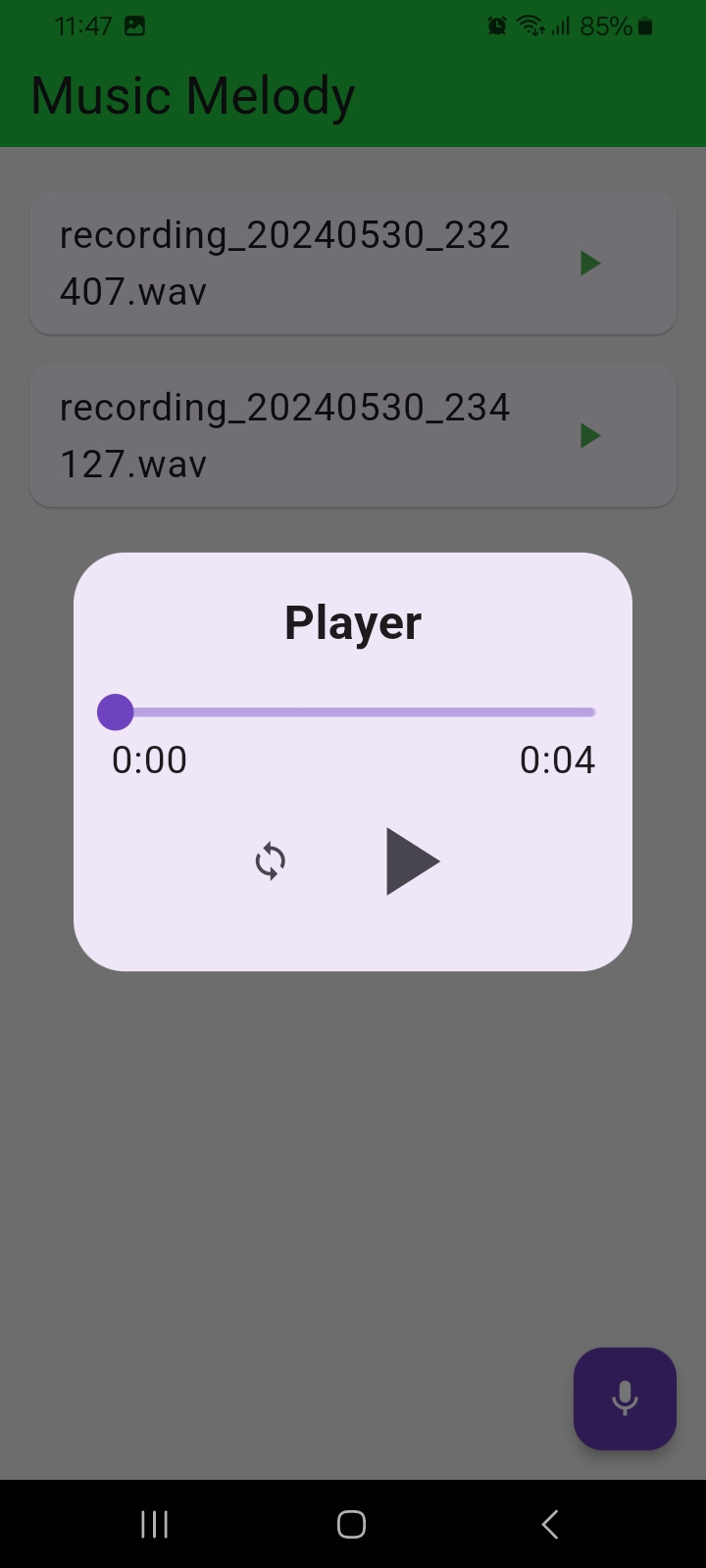
* **Module Interaction:**
  + Test the interaction between different modules (e.g., recording and effects processing).
  + Ensure seamless integration of external libraries and API responses.
* **User Interaction:**
  + Simulate user interactions to test UI responsiveness.
  + Verify touch controls and gesture handling.

#### System Testing

* **End-to-End Testing:**
  + Test complete workflows from recording to exporting a project.
  + Ensure all functionalities work together as expected.
* **Performance Testing:**
  + Test the app’s performance under various conditions (e.g., multiple tracks, long recording sessions).
  + Ensure real-time processing capabilities without noticeable latency.

**6. Output (Screenshots)**





**A screenshot of a phone

Description automatically generated**

**7. Conclusion and Future Scope**

Melody Maker represents a significant advancement in the realm of mobile music production applications. By leveraging the robust capabilities of Flutter, this app successfully bridges the gap between complexity and usability, offering a comprehensive solution for both beginners and experienced music producers. The seamless integration of audio recording, effects processing, and sample looping within a user-friendly interface ensures that users can focus on creativity without being hindered by technical challenges.

By continually evolving and incorporating these advanced features, Melody Maker can remain at the forefront of mobile music production technology, catering to the ever-changing needs of musicians and producers. The future scope of the app envisions a dynamic, collaborative, and intelligent music creation platform that empowers users to bring their musical ideas to life with ease and precision.

**GitHub Link:-** [**https://github.com/ravi-daas/Music\_melody**](https://github.com/ravi-daas/Music_melody)