Voice Recorder

Screens:

1. Home Screen
    - Displays a large microphone button to start/stop recordings.
    - May include a live audio visualizer.
2. Recordings List Screen
    - Shows saved audio recordings.
    - Allows playback and deletion of files.

3. Audio Recording Screen
    - Dots appear while Speaking into the mic.
    - Timing starts from 0:00. Lasts for 10 seconds.

4. Viewing Past Recording
    - Dynamic Bar Moving s to replicate the voice modulation. Randomly generated.
    - Can be paused and continued.

Developing a Sound Recorder Application in Flutter

This report explores the detailed process of developing Rapid Note, a simple yet powerful sound recorder application built using the Flutter framework. Inspired by both practical needs and modern design principles, Rapid Note aims to provide an intuitive and aesthetically pleasing user experience while demonstrating robust technical foundations. This document outlines the architecture, implementation details, key packages, and challenges involved in building a fully functional mobile audio recording solution.

From managing runtime permissions and file I/O to implementing responsive state management and interactive visual feedback (audio visualizer), this application showcases how Flutter can be leveraged to create performance-efficient, cross-platform mobile utilities. Throughout the report, we highlight the separation of business logic via BLoC/Cubit, the use of third-party packages for audio recording and playback, and the UI design considerations that shape the user experience.

Understanding the Core Concept: Audio Recording in Mobile Apps

Voice recorder applications are crucial tools that allow users to capture, store, and playback audio through their smartphones. These tools are used in diverse scenarios — from recording lectures and meetings to capturing spontaneous ideas or audio memos.

The core technology behind this involves:

- Capturing analog audio through the device's microphone.
- Converting analog signals into digital format using the device's Analog-to-Digital Converter (ADC).
- Encoding the result into a playable audio format like .wav or .m4a.
- Saving the file in persistent storage for future access.

Developing such an app, especially with Flutter, presents specific technical challenges:

- Permission Management: Microphone and storage access must be explicitly requested and handled gracefully.
- Audio Recording and Playback: Requires reliable APIs to manage device-specific audio processing.
- Storage Management: Audio files must be saved, named uniquely, retrieved, and deleted properly.
- State Synchronization: Recording states must be cleanly managed to update the UI reactively.
- User Interface: The interface must clearly communicate current recording status and playback options.

Voice Recorder addresses each of these areas systematically using a lightweight but scalable architecture.

Application Architecture and Plan

The application consists of two major screens, each serving a distinct purpose:

Home Screen:

- Central interface for starting/stopping audio recordings.
- Features a large mic button for recording control.
- Displays a live amplitude visualizer during recording using a custom widget.

Recordings List Screen:

- Displays saved recordings using a scrollable list.
- Allows playback and deletion of each file.
- Organized chronologically using timestamped filenames.

The app adopts state management using Cubit (from the flutter_bloc package), with separate cubits for recording (RecordCubit) and managing saved files (FilesCubit). The clear division of UI and logic improves maintainability and scalability.

Key Flutter Packages Used

Package	Purpose
record	Core recording package; used to capture audio and monitor amplitude streams.
just_audio	Plays back audio files using intuitive and customizable controls.
flutter_bloc	Manages dynamic app state through Cubits, such as recording and file list states.
permission_handler	Handles microphone and storage permission checks across Android/iOS.
path_provider	Locates and accesses platform-specific directories for file storage.

These packages abstract platform differences and streamline core functionality, letting developers focus on UI and UX.

Recording Flow & RecordCubit Logic

The RecordCubit orchestrates the recording logic in response to user interactions. It manages state transitions, permission requests, and audio capture operations. Here's how the process works:

startRecording() Method – Step-by-Step:

Permission Handling:

- Uses permission_handler to check microphone and storage access.
- If denied, prompts user with a native dialog.
- If granted, proceeds to initialize the recording.

Recorder Initialization:

- Calls record.start() with parameters like format (.m4a), bit rate, and sample rate.
- Begins streaming amplitude data, which is visualized in the UI via a custom AudioVisualizer widget.

File Naming Strategy:

- Filenames use a timestamp format: e.g., 1709210344000.rn (milliseconds since epoch).
- Ensures uniqueness and sortability, simplifying file organization.

Custom File Format:

- Files are saved with the extension .rn (Rapid Note) as a branding choice.
- Technically still encoded in compatible formats like AAC or M4A, which ensures playback via just_audio.
- The use of .rn avoids cluttering the user's default audio folders with unintentional playbacks.

State Emission:

Once initialized, the Cubit emits RecordOnState.

- UI responds by animating the mic button and activating the visualizer.
- Upon stopping, RecordStoppedState is emitted, triggering file save and UI reset.
- This modular approach ensures a clean lifecycle for recordings and reduces bugs related to state mismatch.

UI/UX Design Philosophy

The app's design is minimalistic, focusing on usability and clarity. UI highlights include:

- Microphone Button: Large and centered for immediate access.
- Amplitude Visualizer: Custom waveform bars animate in real time.
- Dark background and soft shadows: Gives a sleek and modern feel, inspired by Neumorphism.
- Recording List: Each item displays the timestamp and includes playback controls.
- Flutter's rich widget ecosystem enables clean layout, smooth animations, and responsive design.

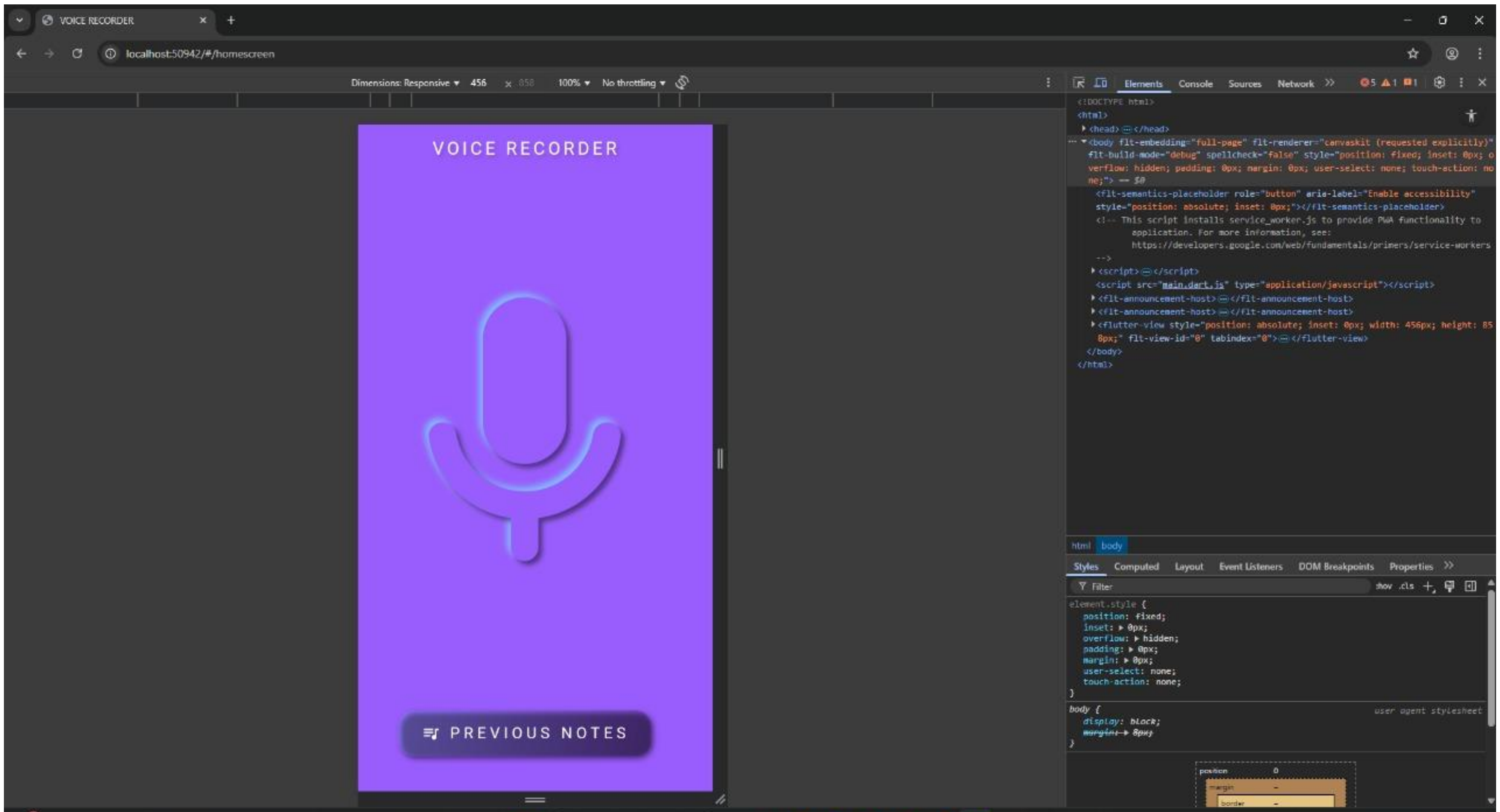
Potential Enhancements

- Add rename/share functionality to recordings.
- Visualize recording duration while capturing.
- Implement audio trimming or editing tools.
- Cloud sync support (Google Drive or Firebase).
- Switchable light/dark themes via user settings.

This state change notifies the UI that recording is now active. The UI can then update accordingly, perhaps by displaying a recording indicator, changing the microphone button's appearance, or activating the audio visualizer.

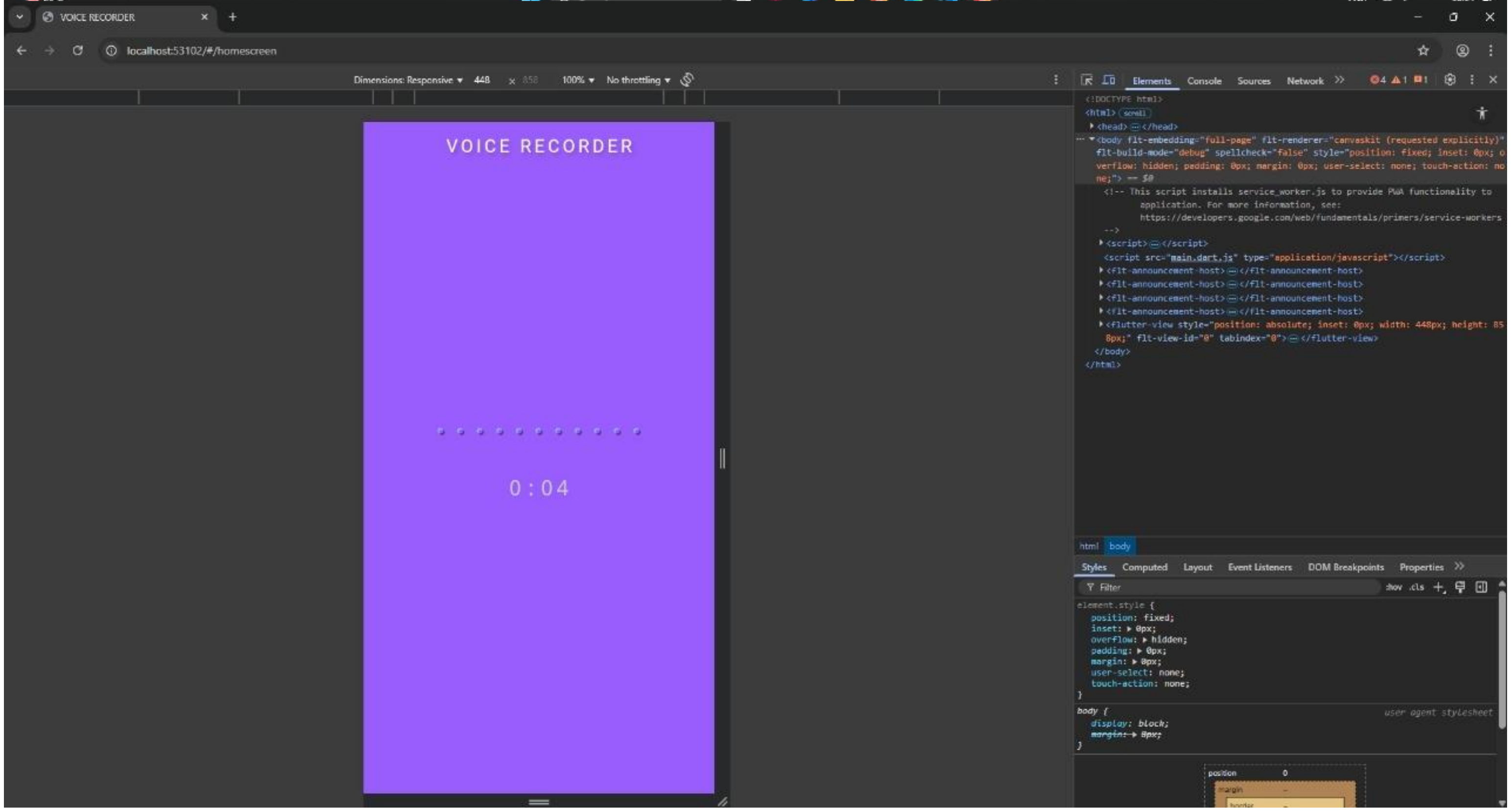
This systematic approach within startRecording() ensures that the recording process is initiated correctly and robustly, handling necessary prerequisites and providing clear feedback to the UI.

```
MaterialApp
└── BlocProvider<RecordCubit>
    └── BlocProvider<FilesCubit>
        └── HomeScreen (StatefulWidget)
            ├── Scaffold
            │   ├── AppBar
            │   │   ├── Title: Text('Rapid Note')
            │   │   └── Actions (optional icons or info)
            │   │
            │   ├── Body: Column
            │   │   ├── Spacer
            │   │   ├── AudioVisualizer (Custom Widget)
            │   │   ├── SizedBox (spacing)
            │   │   ├── RecordButton (Mic)
            │   │   │   └── BlocBuilder<RecordCubit, RecordState>
            │   │   │       ├── Icon: Start / Stop based on state
            │   │   │       └── onPressed: toggle recording
            │   │   └── Spacer
            │   │
            │   └── BottomNavigationBar
            │       ├── Home Icon
            │       └── Recordings Icon
            │
            └── Navigator.push: RecordingsListScreen
                └── Scaffold
                    ├── AppBar
                    │   └── Title: Text('My Recordings')
                    │
                    ├── Body: BlocBuilder<FilesCubit, FilesState>
                    │   └── ListView.builder
                    │       └── RecordingTile (per file)
                    │           ├── Icon (Play / Pause)
                    │           ├── Text (Filename or Timestamp)
                    │           ├── Text (Duration / CreatedAt)
                                └── Delete Button
```


VOICE RECORDER
localhost:50942/#/homescreen
Dimensions: Responsive ▾ 456 × 858 100% ▾ No throttling ▾
VOICE RECORDER
PREVIOUS NOTES
Elements Console Sources Network
html body
Styles Computed Layout Event Listeners DOM Breakpoints Properties

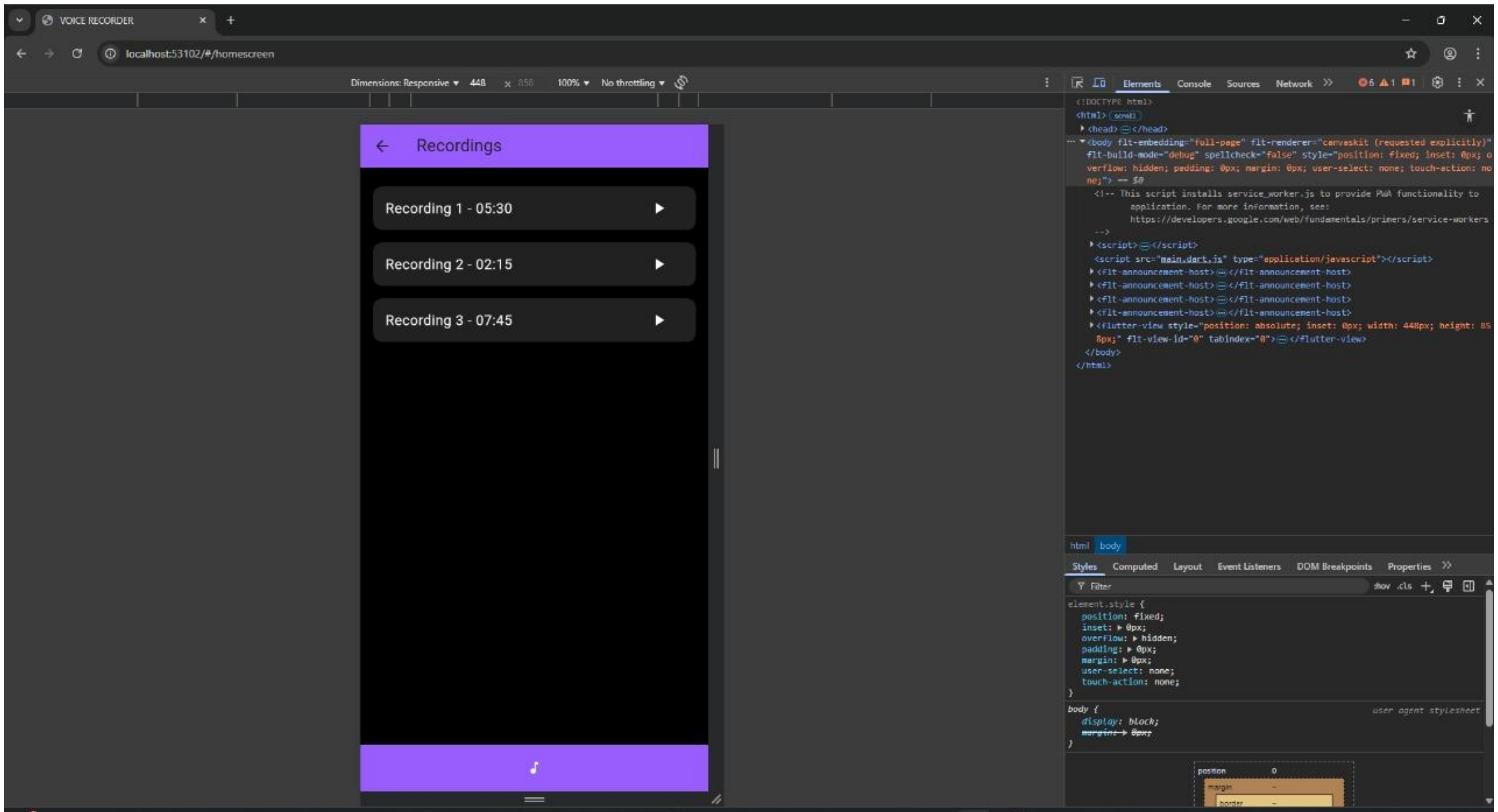
```
<!DOCTYPE html>
<html>
  ▸ <head> ⋯ </head>
··· ▾ <body flt-embedding="full-page" flt-renderer="canvaskit (requested explicitly)"
    flt-build-mode="debug" spellcheck="false" style="position: fixed; inset: 0px; o
    verflow: hidden; padding: 0px; margin: 0px; user-select: none; touch-action: no
    ne;"> == $0
      <flt-semantics-placeholder role="button" aria-label="Enable accessibility"
      style="position: absolute; inset: 0px;"></flt-semantics-placeholder>
      <!-- This script installs service_worker.js to provide PWA functionality to
            application. For more information, see:
            https://developers.google.com/web/fundamentals/primers/service-workers
      -->
    ▸ <script> ⋯ </script>
      <script src="main.dart.js" type="application/javascript"></script>
    ▸ <flt-announcement-host> ⋯ </flt-announcement-host>
    ▸ <flt-announcement-host> ⋯ </flt-announcement-host>
    ▸ <flutter-view style="position: absolute; inset: 0px; width: 456px; height: 85
      8px;" flt-view-id="0" tabindex="0"> ⋯ </flutter-view>
    </body>
</html>
```

```
element.style {
  position: fixed;
  inset: ▸ 0px;
  overflow: ▸ hidden;
  padding: ▸ 0px;
  margin: ▸ 0px;
  user-select: none;
  touch-action: none;
}
body {                                   user agent stylesheet
  display: block;
  ~~margin: ▸ 8px;~~
}
```


VOICE RECORDER
localhost:53102/#/homescreen
Dimensions: Responsive ▾ 448 × 858 100% ▾ No throttling ▾
VOICE RECORDER
0:04
Elements Console Sources Network
html body
Styles Computed Layout Event Listeners DOM Breakpoints Properties

```
<!DOCTYPE html>
<html> scroll
  ▸ <head> ⋯ </head>
··· ▾ <body flt-embedding="full-page" flt-renderer="canvaskit (requested explicitly)"
    flt-build-mode="debug" spellcheck="false" style="position: fixed; inset: 0px; o
    verflow: hidden; padding: 0px; margin: 0px; user-select: none; touch-action: no
    ne;"> == $0
      <!-- This script installs service_worker.js to provide PWA functionality to
            application. For more information, see:
            https://developers.google.com/web/fundamentals/primers/service-workers
      -->
    ▸ <script> ⋯ </script>
      <script src="main.dart.js" type="application/javascript"></script>
    ▸ <flt-announcement-host> ⋯ </flt-announcement-host>
    ▸ <flt-announcement-host> ⋯ </flt-announcement-host>
    ▸ <flt-announcement-host> ⋯ </flt-announcement-host>
    ▸ <flt-announcement-host> ⋯ </flt-announcement-host>
    ▸ <flutter-view style="position: absolute; inset: 0px; width: 448px; height: 85
      8px;" flt-view-id="0" tabindex="0"> ⋯ </flutter-view>
    </body>
</html>
```

```
element.style {
  position: fixed;
  inset: ▸ 0px;
  overflow: ▸ hidden;
  padding: ▸ 0px;
  margin: ▸ 0px;
  user-select: none;
  touch-action: none;
}
body {                                   user agent stylesheet
  display: block;
  ~~margin: ▸ 8px;~~
}
```


VOICE RECORDER
localhost:53102/#/homescreen
Dimensions: Responsive 448 × 858 100% No throttling
Elements Console Sources Network
Recordings

Recording 1 - 05:30

Recording 2 - 02:15

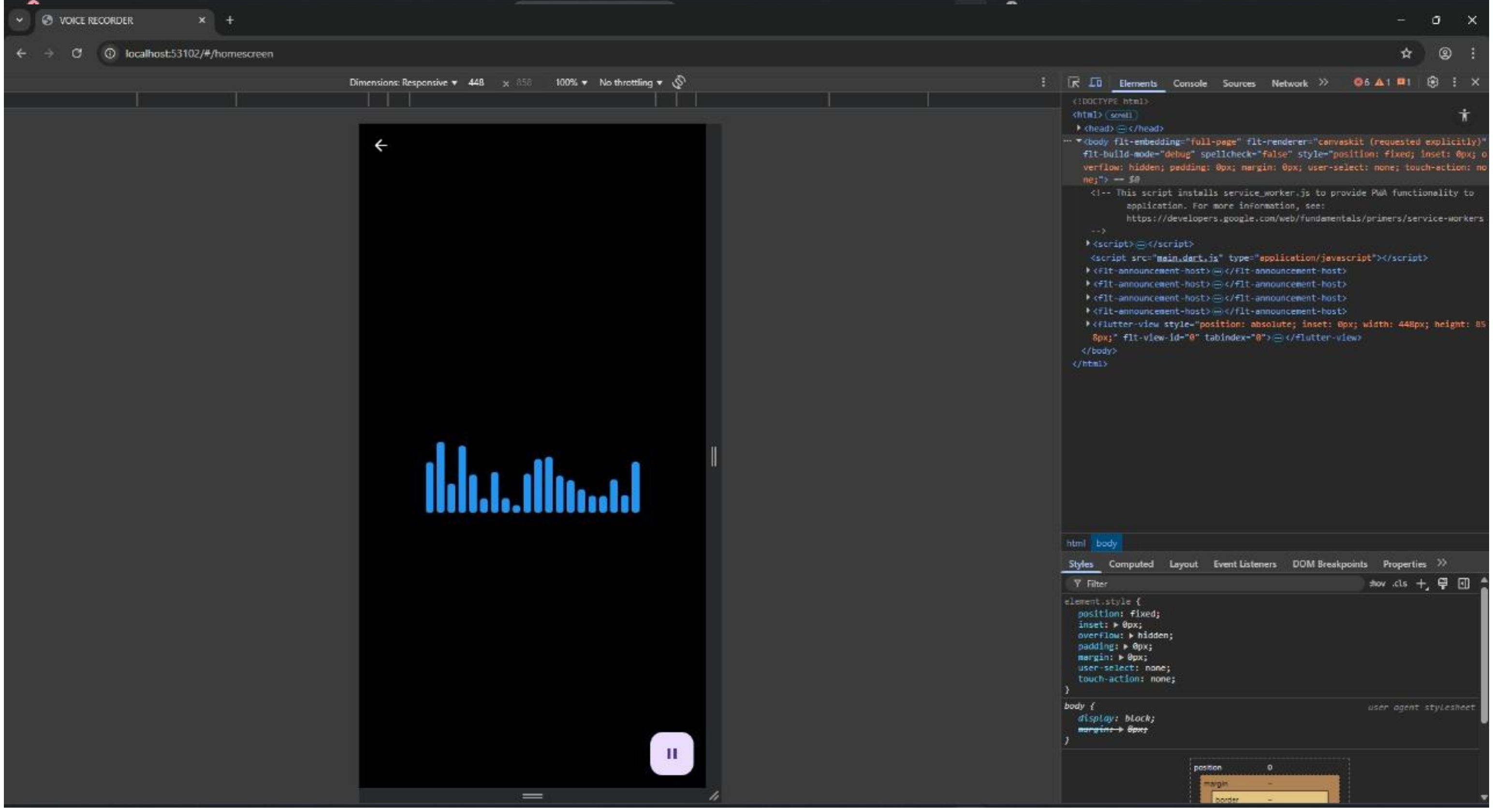
Recording 3 - 07:45

```
<!DOCTYPE html>
<html> scroll
  <head> ⋯ </head>
  <body flt-embedding="full-page" flt-renderer="canvaskit (requested explicitly)"
  flt-build-mode="debug" spellcheck="false" style="position: fixed; inset: 0px; o
  verflow: hidden; padding: 0px; margin: 0px; user-select: none; touch-action: no
  ne;"> == $0
    <!-- This script installs service_worker.js to provide PWA functionality to
         application. For more information, see:
         https://developers.google.com/web/fundamentals/primers/service-workers
    -->
    <script> ⋯ </script>
    <script src="main.dart.js" type="application/javascript"></script>
    <flt-announcement-host> ⋯ </flt-announcement-host>
    <flt-announcement-host> ⋯ </flt-announcement-host>
    <flt-announcement-host> ⋯ </flt-announcement-host>
    <flt-announcement-host> ⋯ </flt-announcement-host>
    <flutter-view style="position: absolute; inset: 0px; width: 448px; height: 85
    8px;" flt-view-id="0" tabindex="0"> ⋯ </flutter-view>
  </body>
</html>
```

html body
Styles Computed Layout Event Listeners DOM Breakpoints Properties
Filter :hov .cls

```
element.style {
  position: fixed;
  inset: ▶ 0px;
  overflow: ▶ hidden;
  padding: ▶ 0px;
  margin: ▶ 0px;
  user-select: none;
  touch-action: none;
}
body {                                    user agent stylesheet
  display: block;
  ~~margin: ▶ 8px;~~
}
```

position 0
margin –
border –


VOICE RECORDER
localhost:53102/#/homescreen
Dimensions: Responsive 448 × 858 100% No throttling
Elements Console Sources Network

```
<!DOCTYPE html>
<html> scroll
  <head> ⋯ </head>
  <body flt-embedding="full-page" flt-renderer="canvaskit (requested explicitly)"
  flt-build-mode="debug" spellcheck="false" style="position: fixed; inset: 0px; o
  verflow: hidden; padding: 0px; margin: 0px; user-select: none; touch-action: no
  ne;"> == $0
    <!-- This script installs service_worker.js to provide PWA functionality to
         application. For more information, see:
         https://developers.google.com/web/fundamentals/primers/service-workers
    -->
    <script> ⋯ </script>
    <script src="main.dart.js" type="application/javascript"></script>
    <flt-announcement-host> ⋯ </flt-announcement-host>
    <flt-announcement-host> ⋯ </flt-announcement-host>
    <flt-announcement-host> ⋯ </flt-announcement-host>
    <flt-announcement-host> ⋯ </flt-announcement-host>
    <flutter-view style="position: absolute; inset: 0px; width: 448px; height: 85
    8px;" flt-view-id="0" tabindex="0"> ⋯ </flutter-view>
  </body>
</html>
```

```
element.style {
  position: fixed;
  inset: ▶ 0px;
  overflow: ▶ hidden;
  padding: ▶ 0px;
  margin: ▶ 0px;
  user-select: none;
  touch-action: none;
}
body {                                    user agent stylesheet
  display: block;
  ~~margin: ▶ 8px;~~
}
```