Vibe Player Milestone #1 Requirements

This document describes the requirements for the first development milestone **(Milestone #1)** of the **Vibe Player** app.

The goal of this milestone is to build the initial version of the offline music player with proper scanning, playback, and adaptive layout support.

The mockups define the app's appearance and behavior. Feel free to decide how you implement specific things (e.g., how you display a specific loading progress or how you specify error messages).

You can find the Vibe Player **mockups here**:

> https://www.figma.com/design/yge7H3tImtXHQZXCG9hUVr/VibePlayer?node-id=4-18822

Milestone #1 Goal

- Implement the **first-launch flow** with a Splash Screen and Permission Screen.
- Create the **Main Screen** with scanning, empty, and track list states.
- Develop the **Scan Music Screen** for manual rescan and filtering.
- Add the **Now Playing Screen** with basic playback controls (Play/Pause, Previous, Next).

Icons

You may use Material Design icons where appropriate. If a suitable Material icon is not available, use the custom icons provided in the Figma mockups.

In Figma, any icon or image can be **exported** by selecting the element and clicking "Export" in the **right-hand panel**. In this panel, you can also choose the desired format (PNG, SVG, etc.).

Adaptive Layouts

The app must support two breakpoints:

- **Up to 840 dp** → mobile layout.
- **From 840 dp and above** → wide-screen layout.

> 📌 Since there's already plenty to do in this challenge, full wide-screen adaptation is **optional**. Only **portrait orientation** is required, but still, check the 840dp+ mockups to understand how elements should look and align on larger screens.

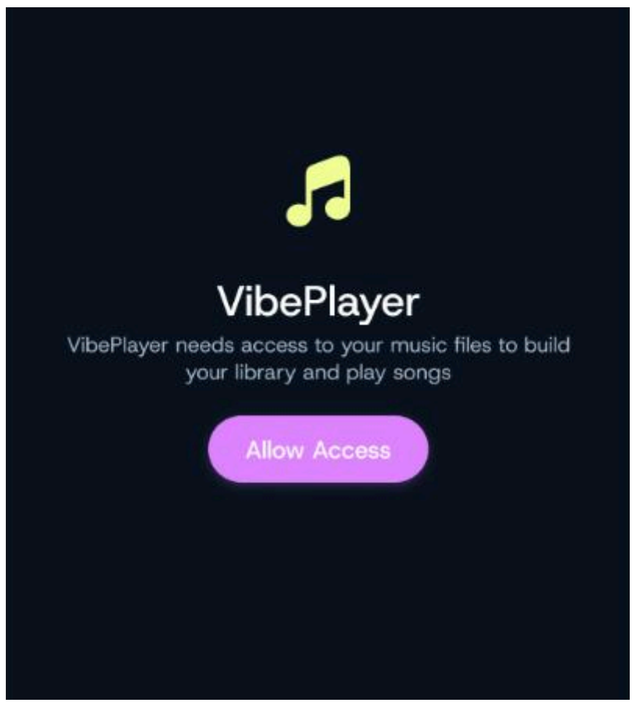
Technical Requirements

🚀 Splash Screen

- Background: **solid color without gradients**.
- Center: **music note icon** (app logo).

🔐 Permission Screen (First Launch)

- This screen appears **before the system permission dialog** to explain why access is required.
- The main purpose is to make it clear that **the app cannot function without this permission**, and the user cannot proceed to the main screen.



UI elements:

- App logo and name VibePlayer.
- Short description:“VibePlayer needs access to your music files to build your library and play songs.”
- Allow Access button that triggers the system permission request for media files.

> 💡 After tapping **Allow Access**, the standard Android permission dialog appears:
>
> - **Android 13 (API 33+)** → uses READ_MEDIA_AUDIO, introduced as a more granular and secure permission that allows reading only audio files from the media library.
> - **Android 12 (API 32-)** → uses READ_EXTERNAL_STORAGE, an **older general-purpose** permission providing access to the entire external storage, required for older versions where no media-type separation existed.

If permission is denied

If the user denies access, show a dialog: *“VibePlayer needs access to your music files to function properly. Without this permission, the app cannot build your music library or play songs.”* ***(OK / Try Again)***

- **Try Again** → reopens the system permission dialog.
- **OK** → closes the dialog and keeps the user on the Permission Screen. The **Allow Access** button remains available so the user can retry at any time.

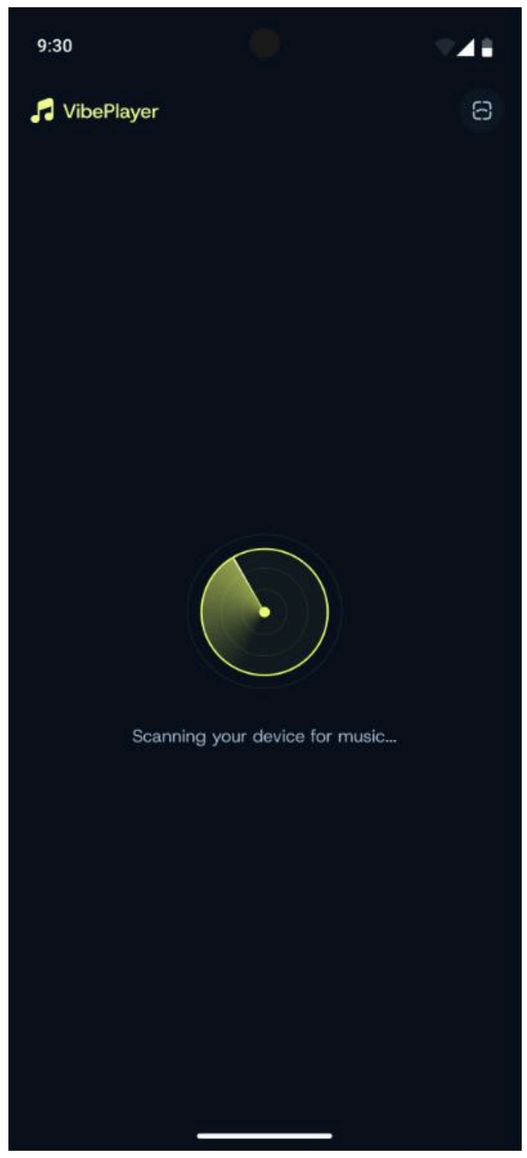
🏠 Main Screen

The main screen of the application, which can appear in several visual states:

- **Scanning State** — when scanning the device for audio files.
- **Empty State** — when no tracks are found.
- **Track List State** — when the scan is successfully completed and the music library is displayed.

Scanning State

- This is the main screen shown after permission is granted.
- **Top Bar:**
  - Left — logo and app name VibePlayer.
  - Right — Scan icon that opens the Scan Music screen.
- **Main Content:**
  - Displays the **scanning process** of local music files.
  - The scan icon **rotates smoothly around its axis**, completing one full rotation every 2 seconds.
  - Below the animation: “Scanning your device for music...”

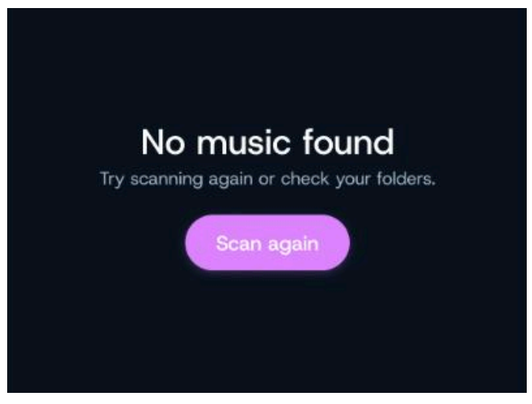


After the scan:

- If tracks are found → show the track list (described later).
- If no tracks are found → show the **Empty State**.

Empty State (No Music Found)

- Displayed when no audio files are found.
- Includes:
  - Text: *“No music found”*
  - Subtext: *“Try scanning again or check your folders.”*
  - Scan again button to restart scanning.



Optimized Library Sync on App Launch

Each time the app is launched, it should verify that all songs stored in the local music library still exist on the device.

Running a full rescan of the entire storage on every launch would significantly slow down startup performance and negatively impact user experience.

To avoid this, **VibePlayer should store the results of the last scan locally** — including the list of all detected tracks and their file paths.

On subsequent launches, the app performs a **lightweight validation pass** over this list to ensure that each file still exists (for example, using a simple *File.exists()* check).

If a file is no longer available, it should be automatically removed from the local database and hidden from the library view.

Users can still perform a manual full rescan at any time through the Scan Music screen, ensuring the library can be refreshed on demand.

This approach keeps the music library **accurate and up to date** while ensuring **fast and smooth startup performance** without unnecessary rescanning.

Track List State

- This state appears after the user has granted storage access and the scanning process successfully finishes, showing all detected tracks.

Content:

- Displays a vertical list of songs (LazyColumn).
- Each list item includes:
  - **Album Art** — the cover image retrieved from the file metadata.
    - If metadata contains embedded album art, it is displayed automatically (using **MediaMetadataRetriever**, the official Android API for reading audio file metadata).
    - If no cover is available, a default **placeholder** icon (music note) is shown instead.
  - **Song Title** — extracted from metadata (METADATA_KEY_TITLE).
  - **Artist Name** — (METADATA_KEY_ARTIST).
  - **Track Duration** — (METADATA_KEY_DURATION), displayed on the right in the mm:ss format.

💡 **MediaMetadataRetriever** allows extracting key information from audio files such as title, artist, duration, and album art. It is recommended as the primary API for building the song list in VibePlayer.

Behavior:

- Tapping a track opens the **Now Playing** screen.
- When the user scrolls **more than 10 items down**, a **scroll-to-top** button (FAB with upward arrow) appears in the bottom-right corner.
- Tapping the button smoothly scrolls the list back to the top.
- The button disappears once the user returns to the top of the list.

📡 Scan Music Screen

This screen opens after tapping the **Scan icon** in the top bar of the main screen. Its purpose is to let users rescan their device and refresh the local music library.

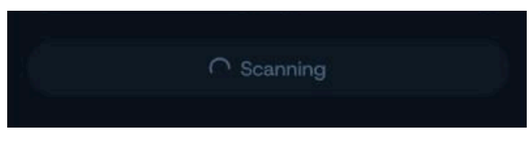
Top Bar

- Left — Back button.
- Center — Title: *Scan Music*.
- Pressing the Back button returns the user to the Main Screen and **removes the Scan Music screen from the back stack** (it should not persist in navigation history).

Content

- A static radar icon is displayed in the center.
  - The radar is not **animated by default**.
  - The rotation animation starts only during an active scan.
- Below the radar, two groups of parameters are displayed:
  a. **Ignore duration less than:**
    - Defines the minimum track length to be included in the scan results.
    - Options: **30s** or **60s** (radio buttons).
    - Helps to skip short sound clips or notifications.
  b. **Ignore size less than:**
    - Defines the minimum file size to be included in the scan results.
    - Options: **100KB** or **500KB** (radio buttons).
    - Filters out small non-music audio files (e.g., voice messages).
- At the bottom is the main **Scan button**, which starts the scanning process.

Scan Button Behavior



- When the Scan button is pressed:
  - The button text changes to *“Scanning...”*.
  - The button becomes **disabled**.
  - A small **progress indicator (spinner)** appears next to the text, rotating while scanning is in progress.
  - The radar icon in the center begins to **rotate smoothly around its axis** (one full rotation every 2 seconds).
- After scanning is complete:
  - The animation and indicator stop.
  - A short **SnackBar message** appears: *“Scan complete — 128 songs found.”*
  - The user is automatically returned to the **Main Screen (Track List State)**, where the song list is updated with the newly found tracks.

🎧 Now Playing Screen

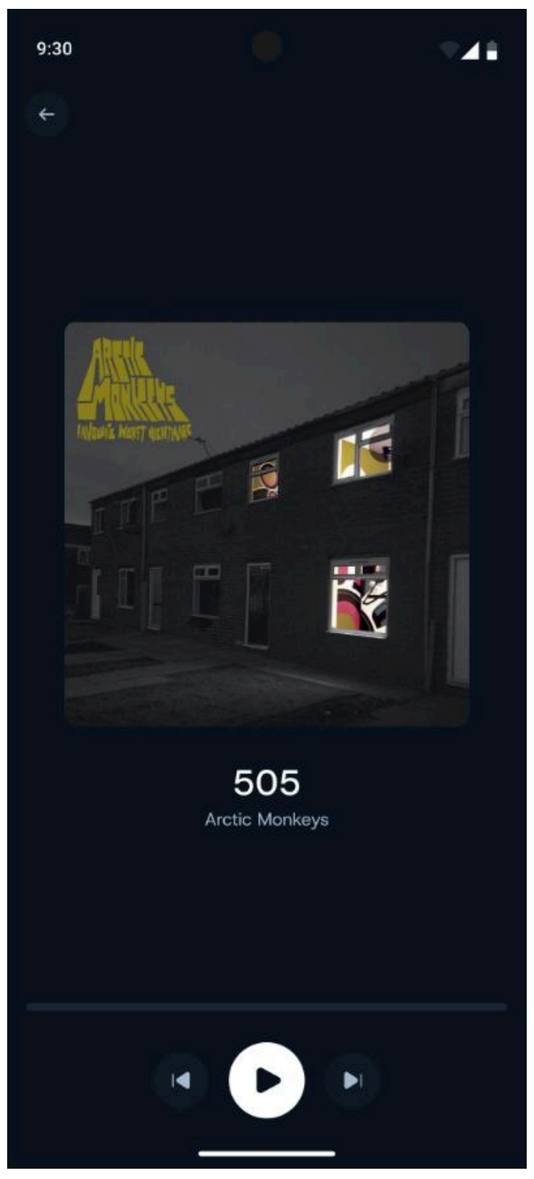
This screen opens after the user taps any song from the list on the main screen. In this milestone, only the **basic player** is implemented — without background playback.

Top Bar

- Left — Back button.
- Tapping the button returns the user to the **Main Screen** and **stops playback**.
- When returning to the main screen, the player completely stops the track — **no background playback** is implemented in this version.

Content

- **Album Art** is centered on the screen.
  - During playback, **ExoPlayer** automatically retrieves and updates the embedded album artwork as tracks change.
  - For the initial music scan and building the local song list, **MediaMetadataRetriever** is used to extract metadata such as titles, artists, durations, and album art directly from audio files.
  - This combination ensures that artwork is correctly displayed both in the song list and during live playback transitions.
  - If no embedded album art is found, a default **placeholder** with a music note icon is shown instead.
- **Song Title** and **Artist Name** are displayed below the album art:
  - Song title (METADATA_KEY_TITLE) — large font.
  - Artist name (METADATA_KEY_ARTIST) — smaller, secondary text.



Progress Bar

- Placed below the track information block.
- Shows the current playback progress in real time.
- In this milestone, the progress bar is not interactive — the user cannot change the track position.

Playback Controls

Arranged horizontally below the progress bar, centered on the screen.

1. ⏮ **Previous Track**

- Moves to the previous track in the list.
- If the current song is the first in the list — the button **does nothing**.

2. ▶️ / ⏸️ **Play / Pause**

- **Play** — starts playing the selected track.
- **Pause** — pauses playback while keeping the current position.
- The icon changes according to the playback state.

3. ⏭️ **Next Track**

- Moves to the next track in the list.
- If the current song is the last in the list — the button **does nothing**.

Exit Behavior

- Pressing the **Back button** stops playback.
- Track progress is **reset**.
- Upon returning to the main screen, the song list view is displayed.

🔗 Useful Links for This Challenge

- Create a Splash Screen
- UX With Material3
- Full Guide to Material3 Theming
- Request runtime permissions
- Introduction to Jetpack Media3
- Create a basic media player app using Media3 ExoPlayer
- Media3 ExoPlayer
- MediaMetadataRetriever
- Access media files from shared storage
- Save data in a local database using Room
- Animations in Compose
- The Full Jetpack Compose Responsive UI Crash Course
- How to Save & Restore the Scroll Position of a LazyColumn Persistently
- Stateful vs. Stateless Composables
- State Hoisting in Compose
- Managing State in Jetpack Compose (Codelab)