Vibe Player Milestone #3 Requirements

This document describes the requirements for the second development milestone (**Milestone #3**) of the **Vibe Player** app.

The goal of this milestone is to expand the core player functionality by adding **playlist management**, including viewing, creating, and displaying both the default **Favourites** playlist and user-created playlists.

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=4-6262

Milestone #3 Goal

- Implement two tabs on the main screen: **Songs** and **Playlist**.
- Add the ability to **create new playlists** through a bottom sheet.
- Display both the system **Favourites** playlist and user-created playlists.
- Ensure **dynamic interface updates** after creating or deleting playlists.
- Prepare the structure for further playlist management features (adding, renaming, or deleting songs).

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:

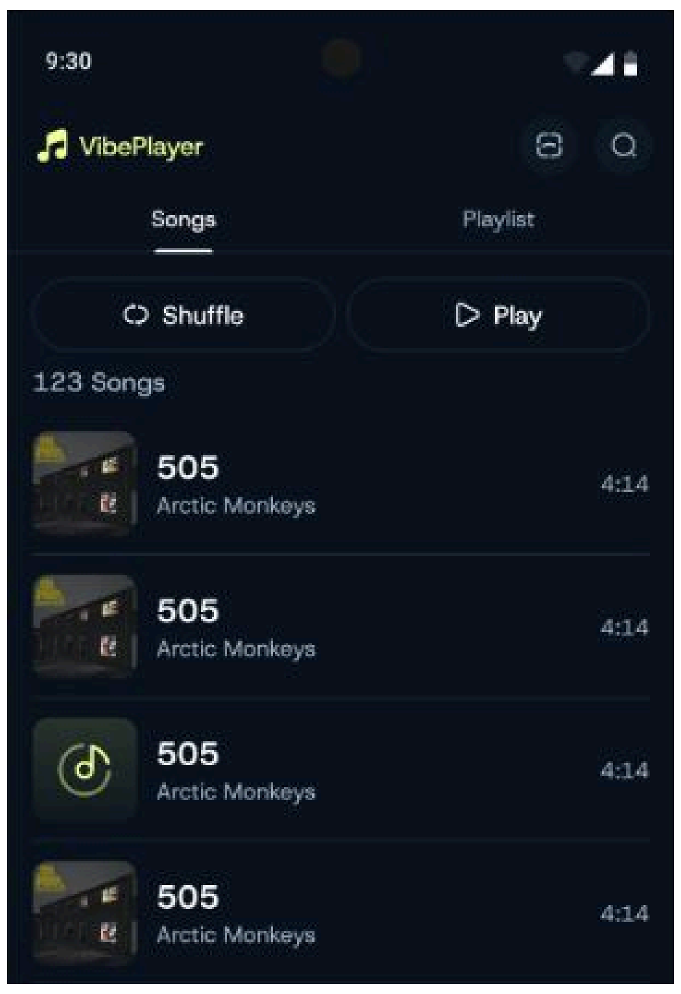
- **Up to 840 dp** → mobile layout.
- **From 840 dp and above** → wide-screen layout.

📌 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

🏠 Tabs on the Main Screen

- The main screen now contains two tabs:
  - **Songs** — displays all songs found during the device scan.
  - **Playlist** — displays the Favourites playlist and user-created playlists.
- The user can switch between the tabs via the top tab bar.
- When scrolling through content (for example, the song list), the **tab bar remains sticky at the top** — only the content below it scrolls.
- In the **Songs** tab, the song list scrolls; in the **Playlist** tab, only the user playlists section scrolls if needed.



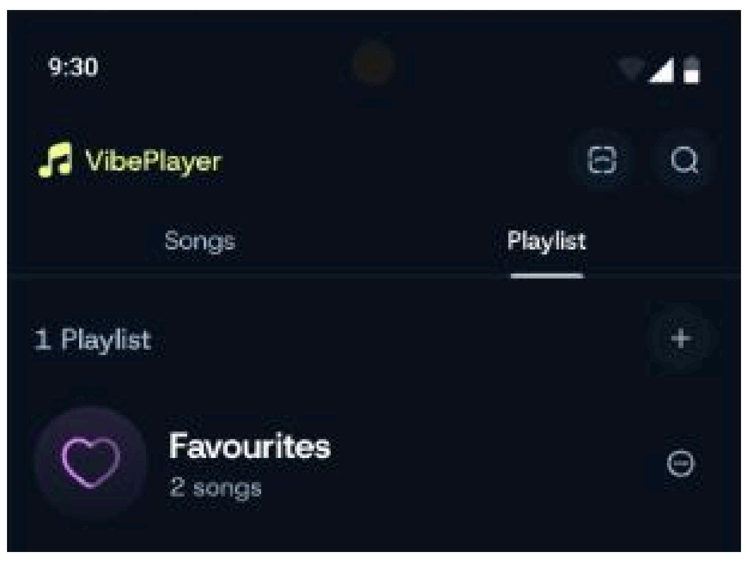
📁 Playlist Tab

Layout and Structure

- The **Playlist** tab is located next to the **Songs** tab.
- It contains the system **Favourites** playlist and a **My Playlists** section with user-created playlists.
- By default, only one playlist exists — **Favourites**, which **cannot be deleted or renamed**.

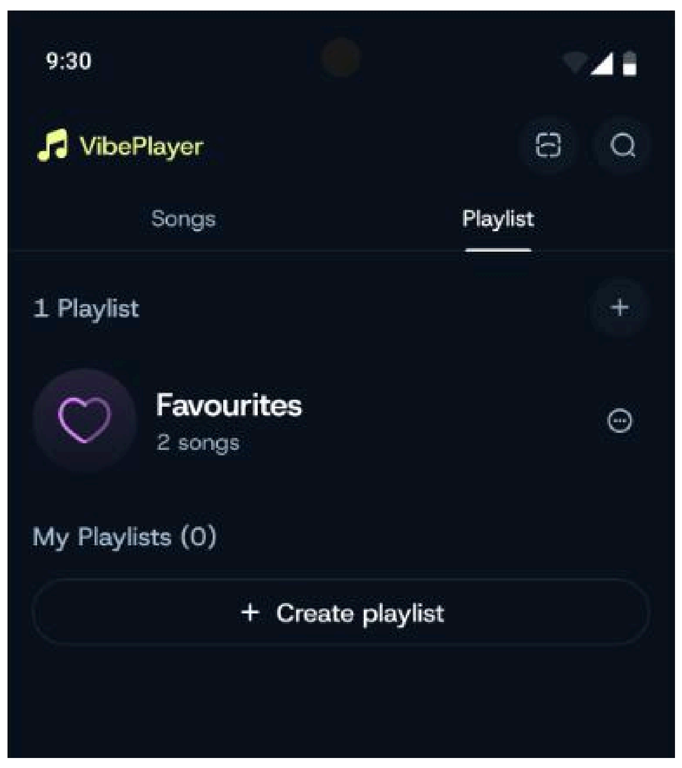
Top Section

- The top section displays a row showing the **total number of playlists**.
  - By default, it shows *“1 Playlist”*, since Favourites is included in the count.
  - When new playlists are created, the number automatically updates (for example, *“3 Playlists”*).
- On the right, a **plus (+)** icon opens the **Create New Playlist** bottom sheet.



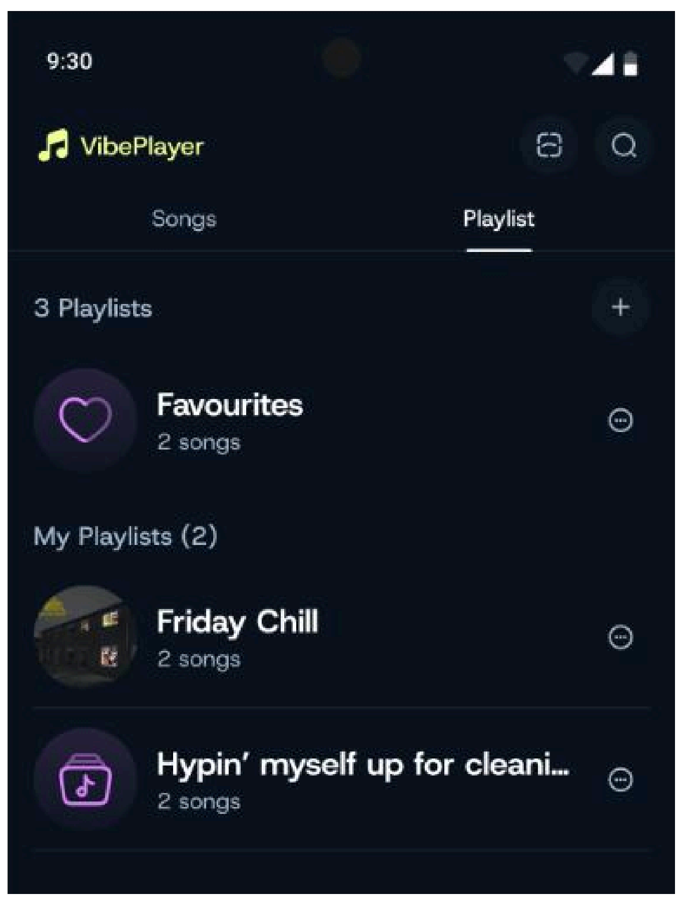
Main Content

Favourites Playlist



- Always appears at the top of the list.
- Contains:
  - A **heart icon** ❤️.
  - Name — **Favourites**.
  - The number of songs below the title (for example, *"2 songs"*).
  - A **... (three dots)** icon on the right, which opens a contextual menu (this menu will be implemented in the next milestone).

User Playlists (My Playlists)



- Below the Favourites playlist, a header displays: *"My Playlists (X)"*, where X is the number of user-created playlists.
- If no playlists have been created yet:
  - Display the text **"My Playlists (0)"**.
  - Show a **"+ Create playlist"** button below, which opens the creation bottom sheet.
- If at least one playlist exists:
  - Display a list with the **playlist name, song count**, and **album art** (or a default music note placeholder).
  - The **"+ Create playlist"** button **is hidden** in this state.
- When multiple user playlists are created, only the **user playlist section (My Playlists)** becomes scrollable, while the *Favourites* section and the top row remain fixed.

🎵 "Create New Playlist" Bottom Sheet

The bottom sheet opens after pressing the **plus (+)** icon or the **"+ Create playlist"** button

💡 **Optional UX Tip:**

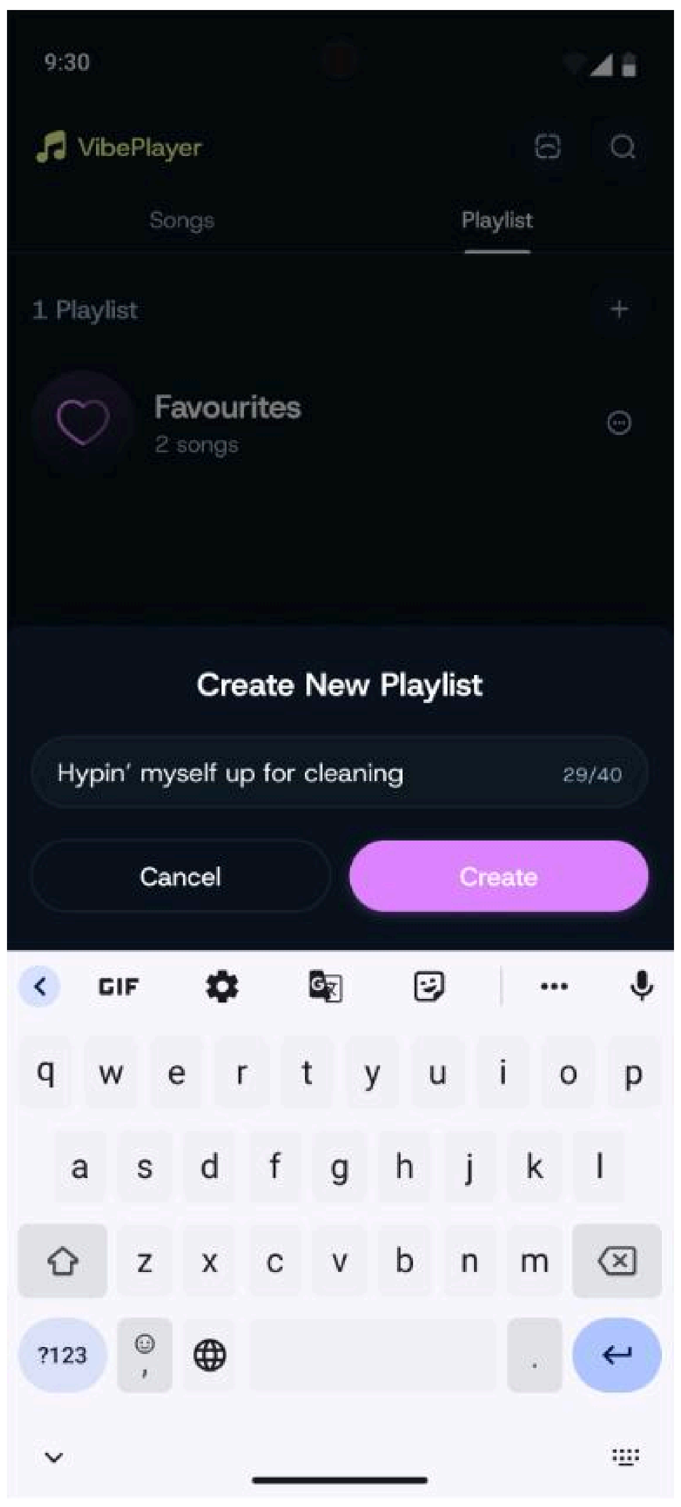
When the keyboard opens, ensure that the bottom sheet content automatically shifts upward so the input field and action buttons remain visible.

It is recommended to use the Android parameter `windowSoftInputMode="adjustResize"` to handle this behavior.

Structure

- Title: Create New Playlist.
- Input field with placeholder "Enter playlist name".
- A character counter in the upper-right corner of the field **(0/40)**, indicating that the playlist name can contain a maximum of **40 characters**.

- Once the user reaches the 40-character limit, no additional input is accepted.
- Action buttons:
  - **Cancel** — closes the sheet without saving.
  - **Create** — creates the playlist (active only when text is entered).

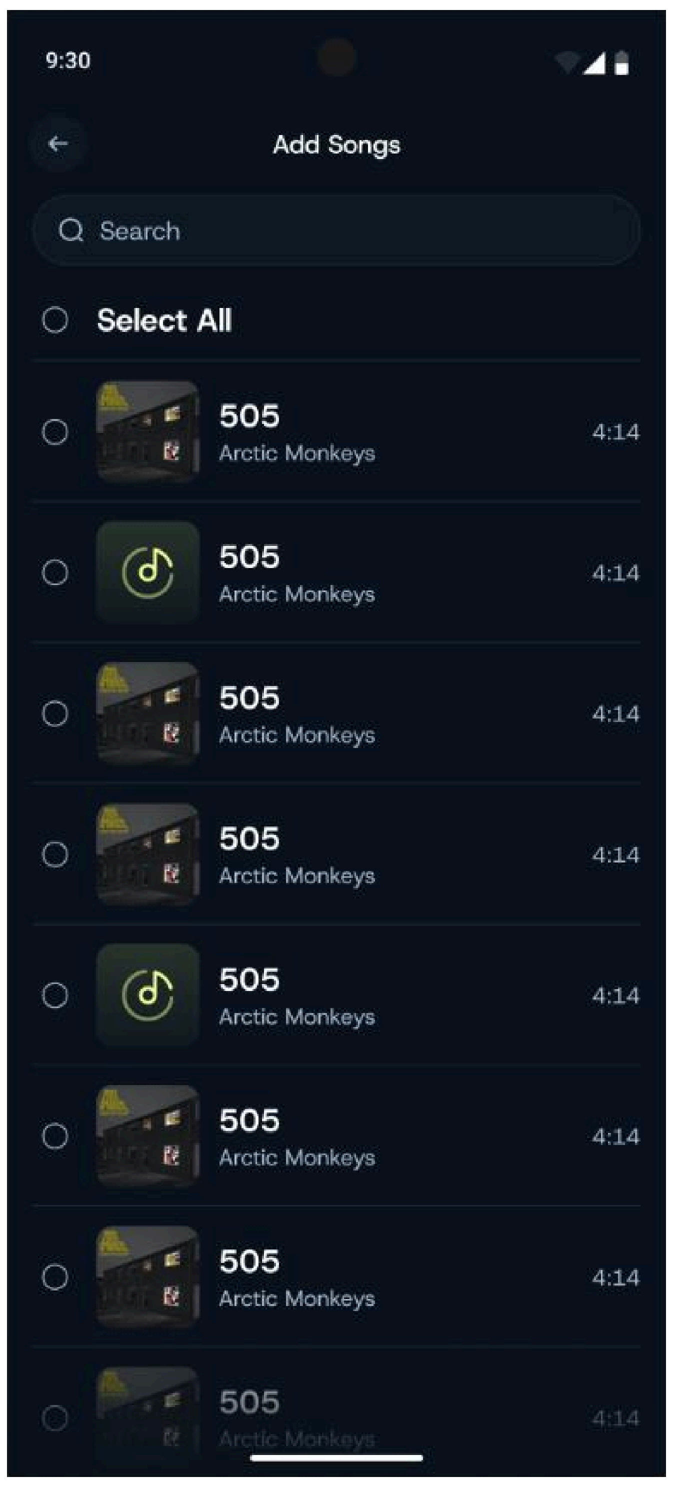


Behavior

- When **Create** is pressed:
  - The new playlist is created and added to the list.
  - The user is redirected to the **Add Songs** screen, where they can add songs to the newly created playlist.
- When **Cancel** is pressed, the sheet closes without changes.
- If the user enters a name that already exists, a **SnackBar message** appears: *"Playlist with this name already exists."*

🎼 Add Songs Screen

The Add Songs screen opens after pressing the **Create** button in the **Create New Playlist** bottom sheet. Its purpose is to allow the user to select songs to be added to the newly created playlist.



Top Bar

- The top **AppBar** includes:
  - A **Back** button (←) — returns the user to the **Playlist Tab**.
    - The created playlist **is not deleted**, even if no songs were selected.
    - In this case, it remains saved with a **placeholder icon** and displays "0 songs".
  - **Title:**
    - Initially shows **"Add Songs"**.
    - When the user selects at least one song, the title updates dynamically to **"X Selected"**, where X is the number of selected tracks.

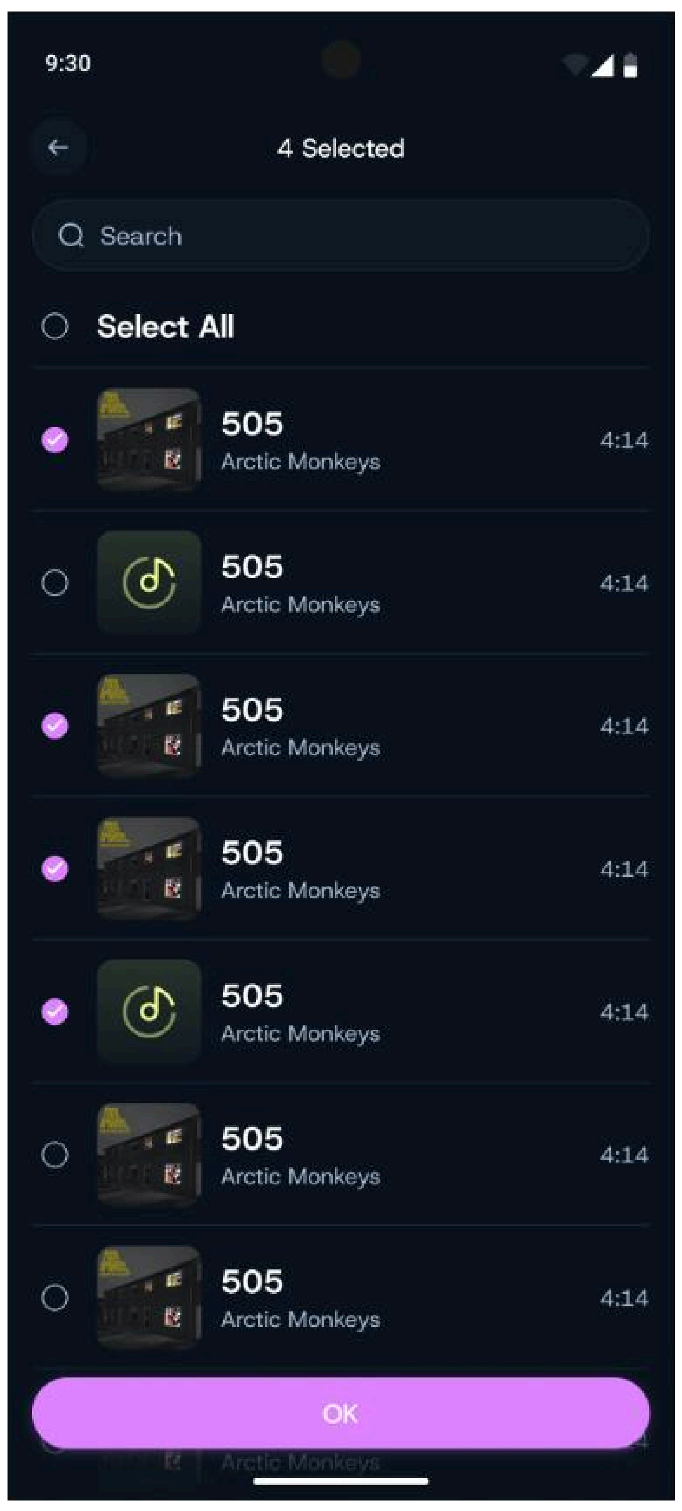
Search Field

- A **Search bar** appears below the title, featuring a 🔍 icon on the left and a *"Search"* placeholder.
- Search operates **locally**, scanning through all available songs found during the initial device scan.

- The query is matched against both:
  - **Song title**, and
  - **Artist name.**
- Search results update in **real time** with a short delay (300–500 ms).
- If no matches are found, display *“No results found.”*
- When the user clears the field (presses ❌ or deletes all text), the full song list reappears.
- The **Search bar** and **Select All** option remain sticky at the top of the screen during scrolling.

Main Content

- The main section displays the full list of locally available songs.
- Each song item includes:
  - **Checkbox** (circle) — on the left side, for selecting or deselecting a song.
  - **Album Art** — if available in metadata, otherwise a default music note placeholder.
  - **Song Title** — main text.
  - **Artist Name** — secondary text below the title.
  - **Track Duration** — right-aligned, in mm:ss format.
- At the top of the list is a **Select All** option:
  - If all songs are selected, the checkbox appears **active**.
  - If only some or none are selected, the checkbox appears **inactive**.
- The song list can be **scrolled** if the number of tracks exceeds the visible area.
- The **top bar, search field,** and **Select All** section remain **fixed (sticky)** while scrolling.



“Selected” State

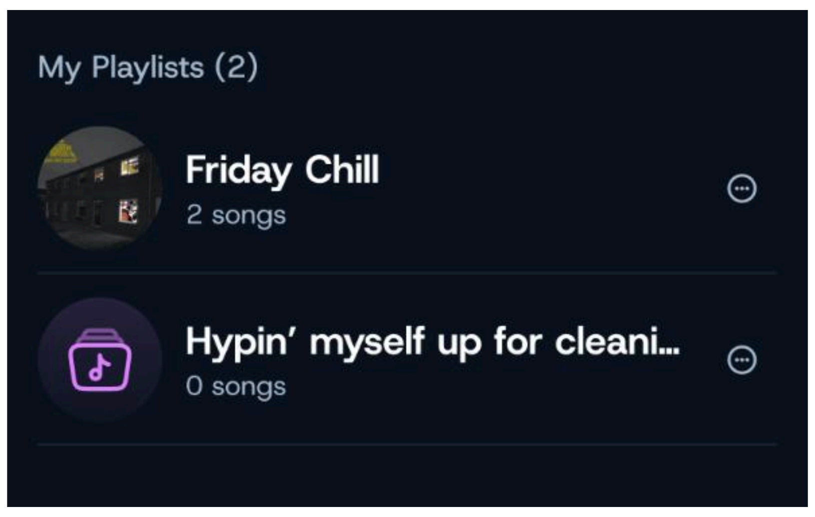
- When one or more songs are selected:
  - The title updates to **“X Selected”**.
  - A bottom **OK** button appears:
    - It uses the app’s accent color (purple background).
    - It disappears when no songs are selected.
- Tapping **OK** adds the selected songs to the current playlist and returns the user to the **Playlist Tab**.

> 💡 **UX Note:**
>
> The navigation back to the Playlist Tab should happen immediately, without noticeable delay. Adding songs to the playlist can continue in the background, ensuring smooth and responsive user experience.

OK Button Behavior

- The **OK** button is visible only when at least one song is selected.
- Upon tapping:
  - The selected tracks are saved to the playlist in the local database.
  - A short **SnackBar message** appears: *"4 songs added to playlist."*
  - The app navigates back to the **Playlist Tab**, where the updated playlist is immediately visible.
- **Double action prevention:** if the user taps OK multiple times, only the first action is processed, and subsequent taps are ignored to prevent duplicate song additions.

Returning to the Playlist Tab



- If songs were selected:
  - The selected tracks are **added to the playlist**.
  - The playlist's **icon updates** to show the **album art** of the first added song.
  - If the first song does not include album art, a **default placeholder** (folder with a music note) is used instead.
  - Below the playlist name, display the **number of added songs**, e.g. *"2 songs"*.
- If no songs were selected:
  - The playlist remains in the list with its chosen **name** and a **default placeholder icon**.
  - The text below the name displays *"0 songs"*, indicating that no tracks have been added yet.

Technical Details

- The selection state (including **Select All**) should be managed within a **ViewModel** to preserve it during configuration changes (e.g., orientation changes).
- The song list should use the same data source as the **Main Screen / Songs Tab**.
- The selected tracks are added to the relevant playlist in the **local database**.
- Each time a playlist is opened or played, the app must **verify that all tracks in that playlist still exist** on the user's device.
  - If a file is missing, it must be **automatically removed** from the playlist's local database.
  - The playlist view should update immediately to reflect the change.

⚙️ Playlist Behavior & Storage

- The number of playlists updates dynamically both in:
  - the top row (*"X Playlists"*), and
  - the *"My Playlists (X)"* section title.
- When a new playlist is created, it appears instantly in the list.
- If all user playlists are deleted, the **"+ Create playlist"** button reappears.

- The **Favourites** playlist is always present and cannot be modified.
- Playlist data must be stored in a **local database**.

💡 **Recommendation:**

Using a structured local database ensures that playlists and their songs can be efficiently managed, updated, and retrieved.

For each playlist, it is recommended to store the following parameters:

- **Unique playlist ID** — to uniquely identify each playlist;
- **Playlist name** — to display and reference it in the UI;
- **List of songs** — stored by track IDs or file paths for easy loading and playback.

🔗 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)