# Westify Musik: Mobile Application

A Flutter Application

Submitted by

Jashwanth E M

# Application Details

**Purpose:** Westify Musik App is a mobile application designed to allow users to browse and play music tracks stored in a Firebase database. The tracks are streamed from URLs hosted on GitHub, with support for viewing track details, playing/pausing music, and switching between English and Tamil languages.

**Key Features:**

**1) Pre-set music library:**

•Displays a list of music tracks fetched from Firebase Firestore.
•Each track includes details like title, artist, and album art (image URL).
•Users can play tracks by tapping a play button, with the currently playing track displayed in a bottom player control bar.
•Sample tracks include:

"Sample Song" (Artist: Sample Artist, Streamed from GitHub URL, Date: 2025-03-17).

# Application Details

**2) Music Playback:**

- Streams audio directly from GitHub-hosted URLs using the audioplayers package.
- Users can play, pause, or stop a track via the player controls.
- Example: Tapping "Sample Song" streams the audio from https://raw.githubusercontent.com/your-username/music-files/main/sample_song.mp3

**3) Track Filtering:**

- A search screen allows users to filter tracks by title or artist.
- Example: Searching for "Sample" shows only tracks like "Sample Song."

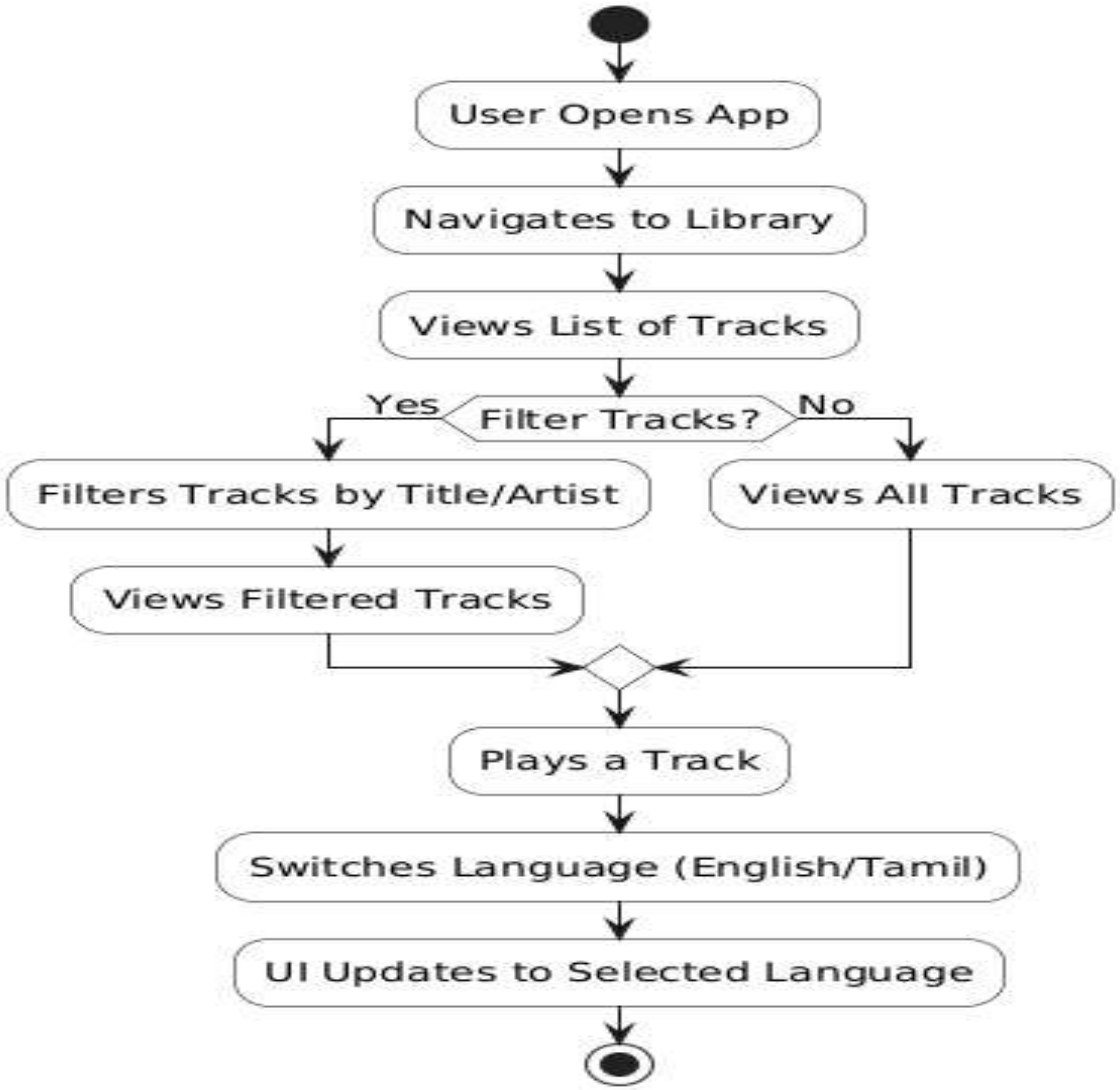
**4) Track Details:**

- Tapping a track in the library shows its details (title, artist, album art).
- Example: "Sample Song" displays its title, artist ("Sample Artist"), and album art loaded from a provided image URL.

**4) Localization:**

- Supports language switching between English and Tamil.
- Example: The app title changes from "Westify" (English) to "வெஸ்டிஃபை" (Tamil), and UI elements like "Home" change to "முகப்பு."

# Workflow Diagram



# Workflow Diagram

**User Initiation:**
- User opens the Westify Musik App, initiating the music streaming experience.
- Automatically navigates to the Home screen, displaying a grid of artists (placeholder).

**Navigation to Music Library:**
- User navigates to the Library screen via the bottom navigation bar, the main page displaying a list of tracks.

**Viewing Tracks:**
- User selects the "Library" tab to explore the music library.
- Views the list of tracks with details like title, artist, and album art.
- Decides whether to filter tracks by searching for a specific title or artist.

**Filtering Tracks:**
- If the user chooses to filter, they enter a search query in the "Search" tab (e.g., "Sample").
- Views filtered tracks matching the query (e.g., "Sample Song").
- If no filter is applied, the user views all tracks in the library.

# Workflow Diagram

**Interacting with Tracks:**

- User plays a track by tapping the play icon, streaming the audio from the GitHub URL.
- Views the currently playing track in the player controls at the bottom.
- Pauses or stops the track using the player controls.

**Switching Language**:

- User selects the "Switches Language" option from the AppBar to toggle between English and Tamil.
- Switches the language, updating the UI to the selected language (e.g., "Library" to "நூலகம்").
- The entire app UI, including labels and buttons, reflects the chosen language.

**Workflow Termination**:

- The process concludes after the user completes their chosen action, with the workflow converging to a stop state.
- Future enhancements could include user authentication and playlist creation.

# Flutter Concepts Used

1. **Widgets**:

- **StatelessWidget**: Used for static UI components like the track cards in the library (e.g., displaying title, artist, album art).
- **StatefulWidget**: Used for dynamic components like the player controls (to handle play/pause state) and the Home screen (to manage navigation).
- **Scaffold**: Provides the app's basic structure with an AppBar (e.g., "Westify" title) and body (track list).
- **AppBar**: Displays the app title and language switcher dropdown.
- **ListView**: Renders the list of tracks in the library dynamically.
- **Card**: Wraps each track in the library for a clean, elevated look.
- **TextField**: Used in the Search screen for filtering tracks by title or artist.
- **DropdownButton**: Implements the language switcher with options (English, Tamil).
- **IconButton**: Used for actions like play/pause in the track tiles and player controls.

**2. State Management:**

•**Provider:** Manages the audio playback state across the app using **AudioProvider.**

**Example:** final audioProvider = Provider.of<AudioProvider>(context, listen: false);
audioProvider.play(song);

- **setState:** Manages state changes, such as updating the UI when switching languages.

**Example:**
```
setState(() {
_locale = newLocale;
});
```

# Flutter Concepts Used

**3. Navigation:**

- **Navigator:** Handles navigation between screens**.**
  **Example:** Navigating to the Search or Library screen via the bottom navigation bar.
- **BottomNavigationBar**: Implements tab navigation (Home, Search, Library).

**4. Localization**:

- **flutter_localizations Package**: Enables localization support for English and Tamil.
- **LocalizationsDelegate**: Defines supported languages (Locale('en'), Locale('ta')).
- **intl Package**: Manages translations for UI elements.
  Example: AppLocalizations.of(context)!.home displays "Home" (English) or "முகப்பு" (Tamil).
- **Language Switcher**: A dropdown in the AppBar to toggle between languages, updating the UI by rebuilding the app with the new locale.

**5. Network and Storage:**

- **Firebase (Firestore)**: Stores track data (title, artist, audio URL, image URL)
- **http Package (Indirect)**: Streams audio from GitHub URLs using audioplayers.

# Flutter Concepts Used

**Example:**

```
await _firestore.collection('songs').add({
    'title': 'Sample Song',
    'artist': 'Sample Artist',
    'audioUrl':'https://raw.githubusercontent.com/your-username/music-
    files/main/sample_song.mp3',
    'imageUrl': 'https://example.com/sample_image.jpg',
    });
```

**6. Audio Playback**:

- **audioplayers Package**: Streams audio from GitHub-hosted URLs.
- **Example:** await _audioPlayer.play(UrlSource(song.audioUrl));

**7. UI Design**:

- **Material Design**: Follows Flutter's Material Design guidelines for a consistent look (e.g., Card, AppBar, IconButton).
- **Custom Styling**: Uses ThemeData to define colors (e.g., Spotify green 0xFF1DB954 for buttons) and typography.
- **Image Loading**: Displays album art using Image.network with the provided image URL.

# Flutter Concepts Used

**9. File Structure**:

- **lib/main.dart:** Entry point of the app, setting up Firebase, localization, and providers.
- **lib/screens/:** Contains screens like HomeScreen, SearchScreen, LibraryScreen.
- **lib/models/:** Defines models like Song (e.g., class Song { String title; String artist; String audioUrl; String imageUrl; ... }).
- **lib/providers/:** Contains AudioProvider for managing audio playback state.
- **lib/services/:** Contains FirebaseService for interacting with Firestore.
- **lib/widgets/:** Contains reusable widgets like SongTile and PlayerControls.
- **lib/generated/:** Contains localization files (e.g., l10n.dart, generated from app_en.arb and app_ta.arb).

**10. Future Enhancements with Flutter Concepts**

- **User Authentication**: Add Firebase Authentication for user login and personalized playlists.
- **Playlist Creation**: Allow users to create and manage playlists stored in Firestore.
- **Search Enhancement**: Improve search functionality with a SearchDelegate for more advanced filtering.
- **Push Notifications**: Notify users of new tracks using Firebase Cloud Messaging.
- **More Languages**: Extend localization to support additional languages like Hindi or Spanish.
- **Offline Playback**: Cache audio files locally using path_provider for offline access.
- **Advanced State Management**: Use Bloc or Riverpod for more complex state management (e.g., managing playlists).