1/21/2025

Proof of Concept

PoC – Personalized Radio Station

Geethapriyan S
23CS049

Proof of Concept (PoC) Document for Personalized Radio Station

1. Project Overview

The **Personalized Radio Station** is a web-based application that enables users to explore and listen to online radio stations based on selected genres. Users can add their favourite stations to a custom playlist, providing a personalized listening experience. The app supports **dynamic station filtering, playlist management, and audio playback controls**.

The project utilizes **React.js for the frontend** and a **Node.js Express server** to fetch station data from **Radio Browser API**.

---

2. Components of the Project

Frontend

Framework: React.js
Description:

- Interactive user interface for browsing and managing radio stations.
- Allows users to **filter stations by genre** and add/remove stations from a custom playlist.
- Provides **playback controls** such as play, pause, volume, and station switching.
- Persists user **playlists** using localStorage.

Libraries Used:

- **React Router** – Handles navigation between different pages.
- **React Hooks (useState, useEffect)** – Manages state and user interactions.
- **React H5 Audio Player (react-h5-audio-player)** – Simplifies handling of radio playback.

Backend

- Built with **Node.js & Express.js** to serve as a proxy for the **Radio Browser API**.
- Fetches radio station data based on user-selected **language** and **genre filters**.
- Returns a list of **radio station URLs, metadata, and logos**.

Database

- **No database integration** in the current version.
- **Playlists and preferences** are stored in localStorage on the frontend.

- Future iterations may include **MongoDB** to store user playlists and playback history.

Hosting Platform

- **Frontend:** Vercel (Static Hosting)
- **Backend:** Render (Node.js API Deployment)

Deployment Process

1. Build the frontend using Vite.
2. Deploy the React app using vercel.
3. Deploy the Node.js backend on **Render** for handling API requests.

---

3. Frontend Components

1. **Header.jsx** – Displays the app title and navigation links for language selection.
2. **RadioBox.jsx** – Main component for browsing, filtering, and managing stations.
3. **TamilRadio.jsx** – Handles Tamil radio stations separately with similar functionalities.

---

4. Backend Components

- **API Endpoint:** /api/stations?language={lang}&tag={genre}&limit=15
- Uses RadioBrowserApi to fetch radio stations dynamically.
- Handles errors and ensures **CORS support** for frontend requests.
- Future enhancements: **User authentication**, **API integrations (Spotify, Deezer)**, and **Database storage** for playlists.

---

5. Database Components

- **Current Version:** No external database. Uses **localStorage**.
- **Future Enhancements:**
    - Store user playlists in a remote **MongoDB .**
    - Save **user preferences and playback history**.
    - Enable dynamic fetching of song metadata via **external APIs**.

---

6. Hosting & Deployment

Hosting Platforms

- **Frontend:** Vercel Pages (Static Hosting).
- **Backend:** Render (Node.js Hosting).

Deployment Steps

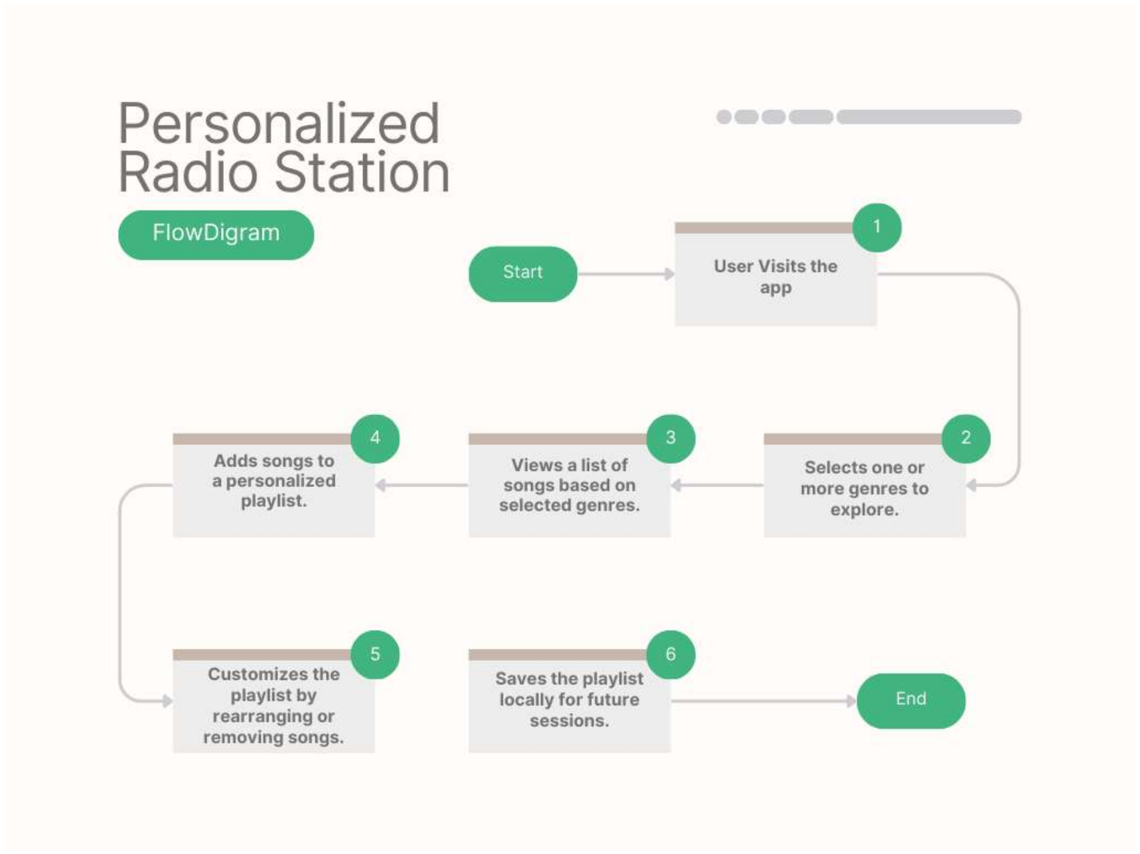
1. **Frontend:**
    - ✓ Build using Vite (npm run build).
    - ✓ Deploy using gh-pages.
2. **Backend:**
    - ✓ Deploy the Express server on **Render**.
    - ✓ Ensure API endpoints are accessible for the frontend.

---

7. Flow Diagram of the Project

User Flow:

1. User visits the application.
2. Selects a **language** (English/Tamil).
3. Filters stations by **genre** (Pop, Jazz, Rock, etc.).
4. Views a list of stations matching the selected filters.
5. Adds selected stations to a **custom playlist**.
6. Customizes the playlist (rearrange, remove stations).
7. Plays stations using **built-in audio controls**.
8. Saves the playlist for **future sessions**.


Personalized Radio Station
FlowDigram
Start
1
User Visits the app
2
Selects one or more genres to explore.
3
Views a list of songs based on selected genres.
4
Adds songs to a personalized playlist.
5
Customizes the playlist by rearranging or removing songs.
6
Saves the playlist locally for future sessions.
End

---

8. Summary

The **Personalized Radio Station** app is a **lightweight, frontend-driven** web application for browsing and managing **online radio stations**. It allows users to **filter, play, and organize radio stations** into a **custom playlist**. The project is **scalable** for future enhancements like **user authentication, API integrations, and database storage**. With its **simple deployment setup**, this PoC demonstrates an effective way to build a **personalized music experience**.