Idea: App that takes your location and makes a Spotify playlist based on the top songs in your area

For our project, we propose the idea of creating an app that uses the user's location and creates a Spotify playlist based on the most popular songs in the user's area. This app will be helpful in community building and helping the user get to know what is popular in their area.

Database: The user will store their location. They will also create a music profile with their name and other simple characteristics.

APIs: We will use the Spotify + Apple Music and Location (Google Maps(?)) APIs.

Third-party authentication: The user will sign in using their email accounts that is linked to the spotify account

This application will have a **decoupled architecture.**

The Cooking Challenge App is a culinary platform designed for users to have cooking competitions. Users will create their own cooking groups with friends or join pre-existing cooking groups. The app will give suggested ingredients and users will cook for a certain time period. When the time goes off user has 1 minute or less to submit a photo of their dish. After this is submitted users will vote/comment//like the uploaded image of their dish.

Implement third-party authentication for user registration and login (e.g., using Google, Facebook, or email). This application will use a database to store user profiles, challenge data, ingredient lists, and user submissions.

- API's
    - Recipe and Ingredient APIs
        - Using (cooking API) (chat gpt)
        - Edaman API
        - Spoonacular API
        - https://www.postman.com/spoonacular-api/workspace/spoonacular-api/collection/7431899-ef0368a7-643c-4c87-975c-68399d4f0c12?action=share&creator=30157286
        - TheCocktailDB API
        -
    - Image API's
        - Unsplash API
    - Nutrition API's
        - Nutritionix API
    - Randomizer API
        - Random.org API
    - Weather API (if you want to add a weather-related challenge)
        - OpenWeatherMap API

Use (select from choices for each)

- Front end
    - HTML, CSS, JS
    - React
    - Vue.js
- Back End
    - Express for Node.js
    - Python with Flask
- Database
    - MongoDB
    - SQLite
- User Authentication
    - Passport.js
    - Firebase Authentication
- Deployment

- Heroku
- Netlify

Chat GPT version:

Creating a full-fledged app like the one you've described is a complex and time-consuming task that goes beyond the scope of what I can do here. However, I can guide you through a more detailed outline of each step, including some code snippets and key considerations, to help you get started on building it yourself.

1. **Planning and Design**
- Identify key features: recipe generation, video diary, user profiles.
- Design wireframes for each screen (use tools like Figma or Sketch).
- Plan the user journey through the app.

2. **Frontend Development**
- **React Native** for a mobile app:
  - Set up your project environment (`npx react-native init YourAppName`).
  - Create screens: Home, Recipe Display, Video Upload, User Profile.
  - Implement navigation (use React Navigation).

- **Example Code (React Native):**
```javascript
// Home screen with ingredient input
import React, { useState } from 'react';
import { View, TextInput, Button } from 'react-native';

const HomeScreen = ({ navigation }) => {
  const [ingredients, setIngredients] = useState('');

  return (
    <View>
      <TextInput
        placeholder="Enter ingredients"
        onChangeText={setIngredients}
        value={ingredients}
      />
      <Button
        title="Find Recipes"
        onPress={() => navigation.navigate('Recipe', { ingredients })}
      />
    </View>
  );
};
```

```
export default HomeScreen;
```

3. **Backend Development**
- **Node.js with Express**:
  - Set up a new Node project (`npm init`).
  - Install necessary packages like express, axios, mongoose (`npm install express axios mongoose`).
  - Write API endpoints to handle recipe searches and user interactions.

- **Example Code (Express):**
```javascript
// Server setup
const express = require('express');
const app = express();
app.use(express.json());

// Endpoint to fetch recipes
app.post('/api/recipes', async (req, res) => {
  const { ingredients } = req.body;
  // Fetch recipes using ingredients
  // ...
  res.json({ recipes: fetchedRecipes });
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

4. **Database Design**
- **MongoDB**:
  - Define schemas for Recipes, Users, and Videos.
  - Implement models in your Node.js application.

- **Example Code (Mongoose Schema):**
```javascript
const mongoose = require('mongoose');
const UserSchema = new mongoose.Schema({
  username: String,
  // other fields
});
const User = mongoose.model('User', UserSchema);
```

5. **Connecting Frontend with Backend**
  - Use Axios in your React Native app to make HTTP requests to your Express server.
  - Handle responses and update the state accordingly.

6. **Testing**
  - Write unit tests for your backend logic.
  - Use Jest for testing React Native components.
  - Perform end-to-end testing.

7. **Deployment**
  - Use services like Heroku for the backend.
  - Use Expo for deploying the React Native app.

8. **Maintenance and Updates**
  - Regularly update the app based on user feedback.
  - Monitor performance and fix any emerging issues.

Additional Considerations
- **Security:** Implement authentication and ensure data security, especially for user-related data.
- **Performance:** Optimize both frontend and backend for speed and efficiency.
- **Scalability:** Design your app's architecture to handle growth.

Learning Resources
- **React Native Documentation:** For frontend development.
- **Express.js Documentation:** For backend setup.
- **MongoDB Documentation:** For database management.
- **Heroku and Expo Documentation:** For deployment guidance.