Project Overview

This project will allow users to **record or upload audio**, convert it to text using a **Speech-to-Text API (Google Speech-to-Text, OpenAI Whisper, or Mozilla DeepSpeech)**, and store the transcriptions in a **database**.

Week 1: Project Setup & Core Functionality

Day 1: Understanding the Project & Initial Setup

- Explain the **MERN stack** and how **Speech-to-Text APIs** work.
- Choose a **Speech-to-Text API** (Google, OpenAI Whisper, Mozilla).
- Set up a **React app using Vite**:
- Install **Tailwind CSS**:
- Initialize **Git repository** for version control.

Day 2: Backend Setup (Node.js & Express.js)

- Set up an **Express.js server**.
- Install dependencies:
- Create an API route to **handle file uploads** using multer.

Day 3: Database Setup (Supabase or MongoDB)

- If using **Supabase**:
    - Create a **Supabase project**.
    - Set up a **table for storing audio files and transcriptions**.
    - Install Supabase SDK:
- If using **MongoDB**:
    - Set up **MongoDB with Mongoose**.
    - Create a schema for **storing uploaded audio and transcriptions**.

Day 4: Implement Speech-to-Text API Integration

- Set up API calls to a **Speech-to-Text provider**:
    - **Google Speech-to-Text API**
    - **OpenAI Whisper API**
    - **Mozilla DeepSpeech**
- Example Google Speech-to-Text API integration:

Day 5: Frontend UI for File Upload & Recording

- Create a **React UI** with:
    - A file **upload button**.
    - A **record audio** button using MediaRecorder.
    - A section to **display transcriptions**.
- Use **Tailwind CSS** for styling.

Day 6: Connecting Frontend to Backend

- Use **Axios or Fetch API** to send audio files from React to Express.
- Show **loading states** while the transcription is being generated.
- Display the **transcription result on the frontend**.

Day 7: Storing Transcriptions in the Database

- Modify backend to **save transcriptions in Supabase/MongoDB**.
- Fetch **previous transcriptions from the database** and display them on the frontend.

Week 2: Optimization, Deployment & Testing

Day 8: Enhancing UI with Tailwind CSS

- Improve UI with **better typography, button designs, and animations**.
- Display **history of transcriptions in a card format**.

Day 9: Implementing Error Handling & Validation

- Handle **errors such as invalid file types and API failures**.
- Show **proper error messages**.

Day 10: Authentication & User Sessions (Optional)

- If needed, add **user authentication with Supabase Auth**.
- Allow users to **save and retrieve their transcriptions**.

Day 11: Deploying the Backend

- Deploy **Express.js backend on Render/Vercel**.
- Ensure the **database is accessible from the deployed backend**.

Day 12: Deploying the Frontend & Backend

- Deploy **React app on Netlify/Vercel**.
- Ensure frontend and backend work seamlessly together.

Day 13: Final Testing & Debugging

- Test the project for **UI bugs, API errors, and database issues**.
- Fix **any remaining problems** before final submission.

Day 14: Documentation & Project Submission

- Write a **README.md** explaining the project setup, API usage, and deployment steps.
- Clean up **unnecessary console logs and improve code structure**.

- Submit or present the project.

This **structured 2-week plan** ensures the student builds a **fully functional Speech-to-Text project** while learning **MERN, Supabase, and Tailwind CSS**. Let me know if you need any modifications! 🚀

Resources:
https://deepgram.com/learn/best-speech-to-text-apis

https://www.assemblyai.com/blog/the-top-free-speech-to-text-apis-and-open-source-engines

https://supabase.com/docs/reference/javascript/initializing

Tech Stack

MERN STACK

MongoDB, Express js, React js and Node js

Tailwind CSS

MongoDB/ Supabase