DOC-Tech (Project)

1. Technology Stack

Frontend (Patient and Doctor Dashboard)

- **React**: For building the user interface.
- **Redux or Context API**: For state management (if needed).
- **Material-UI or Tailwind CSS**: For responsive and modern UI design.
- **WebRTC/MediaRecorder API**: For capturing audio in the browser.
- **WebSockets**: For real-time communication (e.g., between patient and doctor).
- **React Router**: For routing between different pages (patient data, history, etc.).

Backend

- **Node.js & Express**: For handling server-side logic, APIs, and routes.
- **MongoDB**: As your database to store patient medical history, doctor notes, etc.
- **Mongoose**: For object data modeling (ODM) to interact with MongoDB.
- **Firebase or AWS SDKs (S3)**: For media storage, like recorded audio files.
- **Socket.io**: To enable real-time features (e.g., live status when the patient enters a room).
- **Nodemailer**: For sending emails automatically with medical records to the medical representative.
- **Twilio**: For SMS notifications (if required).

Speech-to-Text API

- **Google Cloud Speech API or IBM Watson Speech to Text**: To convert recorded audio to text.
- **Web Speech API**: For browser-based speech recognition (if server-based APIs are not required).

Authentication

- **JWT (JSON Web Tokens)**: For secure authentication and authorization.
- **Firebase Authentication or Passport.js**: For user authentication (doctor, patient, and admin).

DevOps & Hosting

- **Heroku or DigitalOcean**: For backend deployment.
- **Netlify or Vercel**: For frontend deployment.
- **MongoDB Atlas**: Cloud-hosted MongoDB database.
- **Docker**: For containerization (optional).

Other Tools

- **Postman**: For API testing.
- **Jest or Mocha/Chai**: For testing the backend APIs.

2. Outline of the Project

- **User Roles**:
    1. **Doctor**: Can view and update patient records, view medical history, and prescribe medicines.
    2. **Patient**: Can view their medical history.
    3. **Medical Representative**: Receives a copy of the patient's medical data via email.
- **Key Features**:
    1. **User Authentication**: Doctors, patients, and medical representatives sign in via JWT/Firebase.
    2. **Patient History Management**: Doctors can update patient records and medical history.
    3. **Speech-to-Text Recorder**: The recorder will turn on when the patient enters the room and store medical information automatically.
    4. **Automatic Email**: After the doctor's consultation, a report will be sent to the medical representative via email.

3. Flow of the Project

1. **User Authentication**:
    - A doctor or patient logs in to access their respective dashboards.
2. **Patient Entry**:
    - When the patient walks in (triggered by a specific action, like scanning a QR code or facial recognition), the voice recorder is automatically turned on.
    - **Flow**: WebRTC/MediaRecorder API starts recording audio > Audio is streamed > Speech-to-text API processes the input.
3. **Doctor's Consultation**:
    - The doctor's suggestions, symptoms, and prescribed medicines are entered automatically through the speech-to-text converter.
    - This data is stored in MongoDB.
    - **Flow**: Audio converted to text > Data is saved in the database (via the backend API).
4. **Report Generation and Email Delivery**:
    - Once the consultation is complete, the system generates a detailed report of the visit.
    - This report is automatically emailed to the medical representative at the pharmacy.

   - **Flow**: Final report is created > PDF/HTML format is generated > Nodemailer sends the email.

5. **Patient and Doctor Dashboards**:
   - The patient can access their medical history through their dashboard.
   - The doctor can review previous visits and update medical records.
   - **Flow**: Data is fetched from MongoDB and rendered on the front end using React.

6. Additional Features:

1. AI-Powered Diagnosis Support

- **Feature: Integrate AI to provide diagnosis suggestions based on symptoms.**
- **How: Use a machine learning model trained on medical data (such as symptoms, diseases, and treatments) to assist doctors by suggesting possible diagnoses.**
- **Tools: TensorFlow.js for client-side models or use pre-trained models hosted via an API (e.g., Google Cloud AI or IBM Watson).**

2. Multi-Language Support

- **Feature: Support for multiple languages for both speech-to-text recognition and UI content. This would make the system more accessible to non-English speaking patients.**
- **How: Use language options in Google Cloud Speech API or IBM Watson Speech to Text, which supports multiple languages.**
- **Tools: Google Cloud Translation API for translating between languages.**

3. Video Consultation

- **Feature: Allow video consultations between doctors and patients.**
- **How: Integrate WebRTC for real-time video calls.**
- **Use case: Helpful in telemedicine where doctors can provide remote consultations**

4. Appointment Scheduling and Management

- **Feature: A patient can book appointments directly through the platform, and doctors can manage their availability.**
- **How: Build a calendar interface where patients can book slots and doctors can view/manage them.**
- **Tools: Use a library like FullCalendar or React Big Calendar.**