**🛠 Full Project Setup: SkillBridge (MERN + AI)**

**📂 Recommended Folder Structure**

/server

/controllers

/models

/routes

/middleware

/utils

/config

/services (for AI interactions)

app.js

server.js

/client

/public

/src

/api

/components

/pages

/contexts

/hooks

/styles

/utils

App.jsx

main.jsx

.env

.env.example

README.md

**🛡 Environment Variables Example (.env)**

# MongoDB

MONGODB\_URI=mongodb+srv://yourUser:yourPassword@cluster.mongodb.net/skillbridge

# JWT

JWT\_SECRET=yourSuperSecretKey

JWT\_EXPIRE=7d

# OpenAI

OPENAI\_API\_KEY=your-openai-api-key

# Cloudinary

CLOUDINARY\_CLOUD\_NAME=yourCloudName

CLOUDINARY\_API\_KEY=yourApiKey

CLOUDINARY\_API\_SECRET=yourApiSecret

# NodeMailer

EMAIL\_SERVICE=gmail

EMAIL\_USER=yourEmail@gmail.com

EMAIL\_PASS=yourEmailPassword

**🛢 Database Schema Suggestions (Mongoose)**

**User Model**

const UserSchema = new mongoose.Schema({

name: String,

email: { type: String, unique: true },

password: String,

skills: [String],

interests: [String],

learningPath: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Course' }],

resumeUrl: String,

resumeFeedback: String,

mockInterviewRecords: [{

jobRole: String,

questions: [{ question: String, userAnswer: String, feedback: String }],

date: Date,

}],

notifications: [{

message: String,

read: { type: Boolean, default: false },

createdAt: { type: Date, default: Date.now }

}],

role: { type: String, enum: ['user', 'admin'], default: 'user' }

}, { timestamps: true });

**Course Model**

const CourseSchema = new mongoose.Schema({

title: String,

description: String,

topics: [String],

link: String,

level: { type: String, enum: ['Beginner', 'Intermediate', 'Advanced'] }

});

**Job Model**

const JobSchema = new mongoose.Schema({

title: String,

company: String,

skillsRequired: [String],

description: String,

applyLink: String,

});

**Resume Upload and Feedback Storage**

* Upload the PDF to **Cloudinary**.
* Extract text using pdf-parse.
* Analyze text with **OpenAI API** and store feedback.

**✨ Key Backend Services**

* **Auth**: JWT-based authentication middleware.
* **Resume Service**: Extract + Analyze.
* **Learning Path Generator**: Query OpenAI with user skills.
* **Mock Interview Engine**: Dynamic prompt generation.
* **Job Recommender**: Match skills with jobs collection.

**🤖 Managing AI Prompt Templates (Best Practice)**

**Create a promptTemplates.js file** inside /utils:

exports.learningPathPrompt = (skills) => `

Given the following skills: ${skills.join(", ")},

suggest a personalized, progressive tech learning path.

`;

exports.resumeFeedbackPrompt = (resumeText) => `

Here is the resume text: "${resumeText}".

Provide a critical review on formatting, missing skills, and improvement areas.

`;

exports.mockInterviewPrompt = (role) => `

Generate 5 technical interview questions for a ${role} position.

`;

✅ Centralized and scalable.

**🖥 Frontend Pages (React + Tailwind)**

* **LandingPage.jsx**
* **LoginPage.jsx** / **SignupPage.jsx**
* **DashboardPage.jsx**
* **CoursesPage.jsx**
* **LearningPathPage.jsx**
* **ResumeAnalyzerPage.jsx**
* **MockInterviewPage.jsx**
* **NotificationsPage.jsx**
* **AdminPanelPage.jsx** (Bonus)

Use **Axios** to hit the API, use **Context API** or **Redux** for auth state management.

**📢 Bonus Features (Optional but Recommended)**

* **Dark Mode Toggle**: Use Tailwind’s dark mode class strategy.
* **Admin Panel**: Add/edit Courses and Jobs (role-based access).
* **Email Notifications**: Send reminders before interview schedules.

**📋 Security Best Practices**

* Use strong password hashing with **bcryptjs**.
* Validate JWT tokens with secure middleware.
* Sanitize input to prevent NoSQL Injection (express-mongo-sanitize).
* Set proper CORS settings.
* Store secrets in .env only and never commit them.

**🏗 API Structure (Backend Example)**

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/auth/register | POST | User signup |
| /api/auth/login | POST | User login |
| /api/courses | GET | List courses |
| /api/learning-path | POST | Generate learning path |
| /api/resume/upload | POST | Upload resume |
| /api/resume/feedback | GET | Get feedback |
| /api/mock-interview/start | POST | Start mock interview |
| /api/jobs | GET | Get job recommendations |
| /api/notifications | GET | Get notifications |

**🔥 Tech Stack Summary:**

| **Layer** | **Stack** |
| --- | --- |
| Frontend | React.js + Tailwind CSS + Axios |
| Backend | Node.js + Express.js + Mongoose |
| Database | MongoDB |
| Authentication | JWT |
| AI | OpenAI API (GPT-4 Turbo recommended) |
| Resume Parser | pdf-parse npm |
| File Storage | Cloudinary |
| Mailing | NodeMailer |

**🚀 Development Plan Suggestion**

1. Set up basic backend (Auth, MongoDB connection).
2. Set up frontend login/signup pages.
3. Build Courses CRUD (Admin).
4. Build Learning Path AI service.
5. Build Resume Upload + Analyzer.
6. Build Mock Interview Chatbot.
7. Build Job Recommendation engine.
8. Build Notification system + Emails.
9. Polish UI/UX (Dark Mode, Admin Panel).
10. Deploy (Vercel for frontend, Render/Heroku for backend, MongoDB Atlas).