✅ Phase 1: MVP – Core Platform

🎯 **Focus:** User Authentication, PDF Upload, Flashcard Generation

🔧 Backend (Django + DRF)

- Initialize Django project and configure Django REST Framework
- Create models: `User`, `UploadedFile`, `Flashcard`
- Implement user authentication with JWT (register, login, logout)
- Build PDF upload API
- Extract clean text from PDFs using `pdfplumber` or `PyMuPDF`
- Integrate OpenAI API to generate flashcards (Q&A pairs)
- Save generated flashcards to the database

🎨 Frontend (React + Tailwind CSS)

- Initialize React project with Tailwind CSS and `shadcn/ui`
- Implement JWT-based user authentication (Signup/Login/Logout)
- Design PDF upload interface
- Display AI-generated flashcards after upload

✅ Phase 2: Flashcard & Quiz Management

🎯 **Focus:** Flashcard interaction, Quiz generation, Exporting

🔧 Backend

- Add flashcard editing and deletion endpoints
- Implement OpenAI-powered MCQ generation (1 correct + 3 distractors)
- Store MCQs in database as a new model (`MCQ`)

- Add quiz scoring and evaluation logic
- Implement export functionality (CSV/TXT for flashcards & quizzes)

🎨 **Frontend**

- Create flashcard deck view with pagination
- Add edit/delete options for flashcards
- Build MCQ quiz UI with answer selection and feedback (score, correct answers)
- Add “Export” buttons to download flashcards/quizzes as CSV/TXT

✅ Phase 3: AI Tutor Assistant & Contextual Learning

🎯 **Focus:** Context-aware AI Q&A (Tutor Mode)

🔧 **Backend**

- Chunk large PDF content intelligently for contextual Q&A
- Use OpenAI API for tutor-style answers based on uploaded files
- (Optional) Integrate vector similarity search (e.g., Faiss, Pinecone)
- Build tutor chat API with context, message history, and fallback logic

🎨 **Frontend**

- Develop real-time chat UI for AI Tutor
- Let users ask questions about their uploaded notes
- Show answers with reference to relevant document sections

✅ Phase 4: Dashboard, Polish & Deployment

🎯 **Focus:** UX/UI Improvements, Final Testing, Deployment

🔁 **Full Stack**

- Create user dashboard showing uploaded files, flashcards, quizzes
- Add responsive design and UI improvements with `shadcn/ui`
- Implement global error handling and toast notifications
- Conduct unit and UI testing
- Optimize performance and handle edge cases

🚀 Deployment

- Deploy backend on Render, Railway, or Fly.io
- Deploy frontend on Vercel or Netlify
- Set up custom domain + SSL (HTTPS)
- Configure environment variables, rate limits, and basic security

---

🗂️ Summary by Feature (Task Quick Reference)

📍 User Auth & Setup

- Django/React boilerplate
- JWT-based register/login/logout

📍 PDF Upload & Parsing

- Upload PDF via API
- Extract and store clean text

📍 Flashcard Generator

- Integrate OpenAI for Q&A
- Save to DB and display on UI

📍 MCQ Quiz Generator

- Use OpenAI to generate MCQs
- Build interactive quiz UI

📍 AI Tutor Mode

- Use context from PDF chunks
- Real-time chat assistant

📍 Final Touches

- CSV/TXT export
- UI polish
- Full deployment
