Project Synopsis

NoteCrafts.ai - An AI-Powered Learning Assistant

Reference: Uploaded synopsis provided by the user. (See source.)

Project Objective:
The primary objective of this project is to develop NoteCrafts.ai, an intelligent web-based learning assistant designed to help students efficiently process and understand information from PDFs, websites, and YouTube videos.
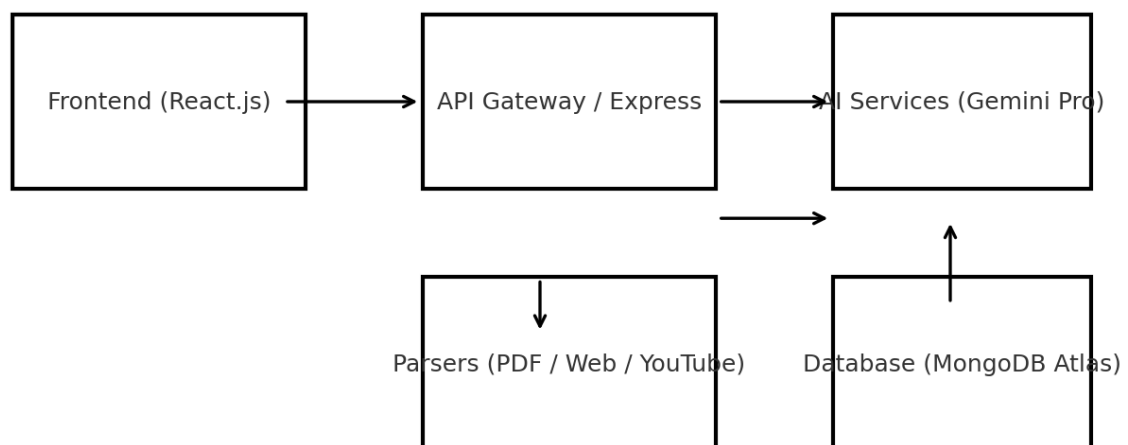
Core Features:
- Automatic Source Summary
- Dynamic Flashcard Generation
- Automated Quiz Generation
- Interactive Mind Map Visualization
- Concept Simplifier (Analogy Generator)
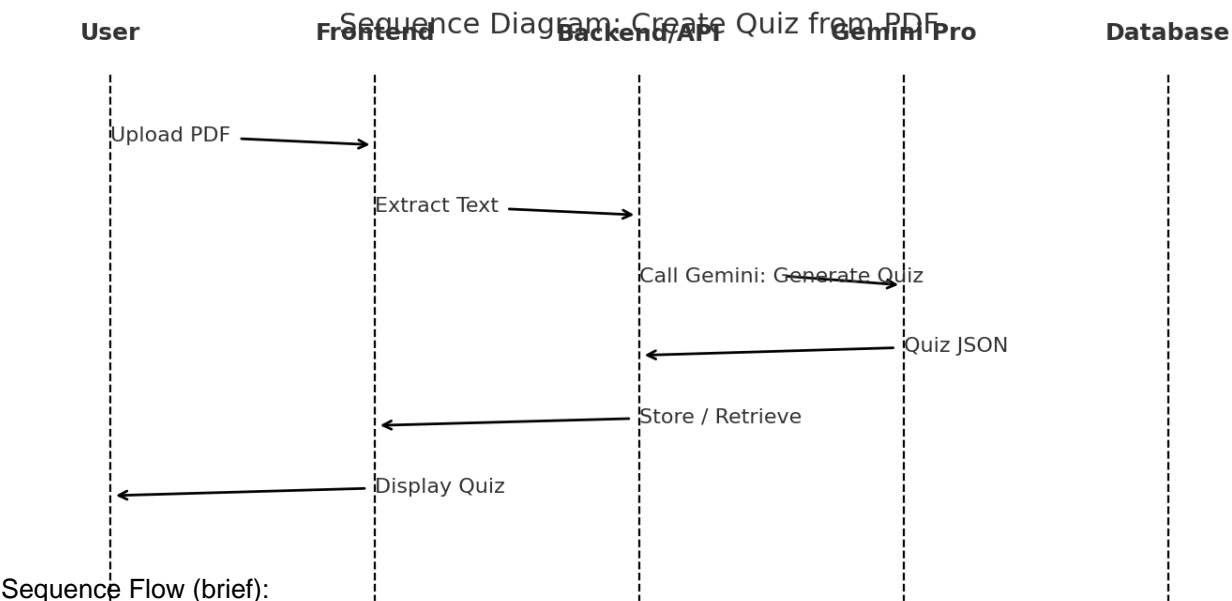- Potential Questions Bank

Technology Stack (high level): React.js (Frontend), Node.js + Express (Backend), MongoDB (Atlas), Google Gemini Pro A

Block Diagram: System Architecture

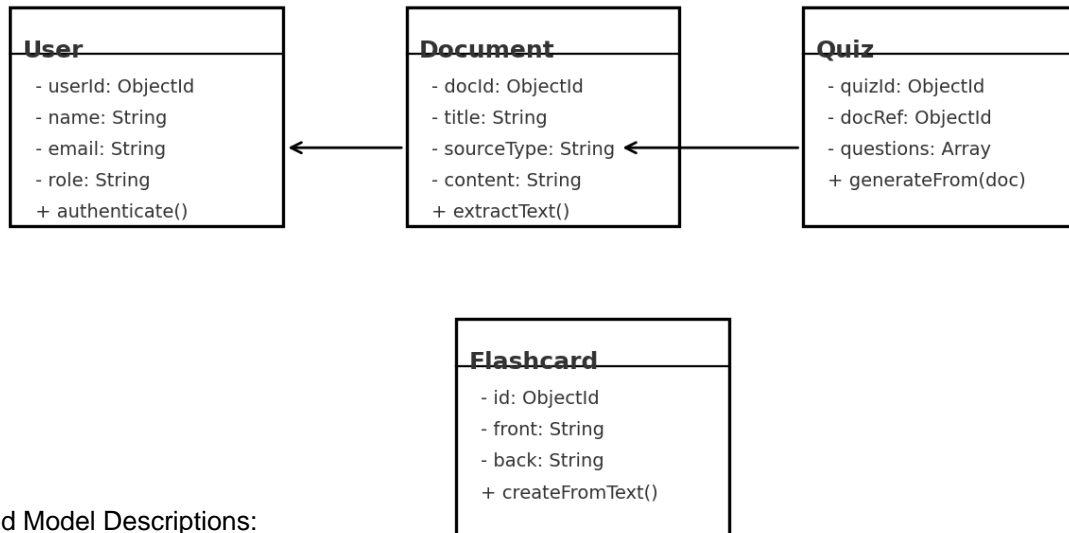Block Diagram: NoteCrafts.ai - High-level Architecture

Sequence Diagram: Create Quiz from PDF

User	Frontend	Backend/API	Gemini Pro	Database

Upload PDF →

Extract Text →

Call Gemini: Generate Quiz →

← Quiz JSON

← Store / Retrieve

← Display Quiz

Sequence Flow (brief):
1. User uploads a PDF via the React frontend.
2. Frontend sends the file to Backend/API endpoint.
3. Backend extracts text (pdf-parse) and optionally stores raw text in MongoDB.
4. Backend calls Google Gemini Pro API with a prompt to generate quiz questions.
5. Gemini returns structured quiz JSON; Backend stores and forwards it to the frontend.
6. Frontend renders the quiz for user interaction.

Class Diagram: Core Data Models

Class Diagram: Core Models (simplified)

User
- userId: ObjectId
- name: String
- email: String
- role: String
+ authenticate()

Document
- docId: ObjectId
- title: String
- sourceType: String
- content: String
+ extractText()

Quiz
- quizId: ObjectId
- docRef: ObjectId
- questions: Array
+ generateFrom(doc)

Flashcard
- id: ObjectId
- front: String
- back: String
+ createFromText()

Simplified Model Descriptions:
User: Stores authentication and authorization information. Methods include authentication helpers.
Document: Represents uploaded sources; includes metadata, raw text, and helper methods for extraction.
Quiz: Stores generated quizzes linked to source documents; contains an array of questions.
Flashcard: Represents flashcards generated from a document.

Notes on Implementation:
- Use multer and pdf-parse for server-side file handling and text extraction.
- Store only necessary metadata and processed text in MongoDB to reduce storage.
- Use prompt engineering with Gemini Pro to produce structured JSON responses for quizzes and flashcards.

Generated based on user-provided synopsis. Diagrams are simplified conceptual illustrations.