Study Fetch Tutor

Introduction

StudyFetch AI Tutor is an advanced educational platform that combines document analysis with artificial intelligence to create an interactive learning experience. The application allows users to upload study materials (PDFs) and engage with an AI tutor that can answer questions, provide explanations, and guide users through complex topics based on the uploaded content.

The platform serves as a personalized study assistant, helping students better understand their course materials through natural language conversations. By leveraging AI to analyze documents and respond to queries, StudyFetch bridges the gap between static reading materials and interactive learning.

Technology Stack

Frontend

- Framework: Next.js 14 (React)
- Styling: Tailwind CSS
- State Management: React Hooks, Zustand and Context API
- PDF Processing: Custom PDF viewer components
- UI Components: Custom components with Tailwind

Backend

- API Routes: Next.js API routes
- Authentication: NextAuth.js
- Database: PostgreSQL with Prisma ORM
- File Storage: Vercel Blob Storage

AI Integration

- AI Models: Anthropic Claude
- AI SDK: Vercel AI SDK

Development Tools

- Language: TypeScript
- Package Manager: npm / yarn
- Version Control: Git
- Deployment: Vercel

Getting Started

Prerequisites

Before installing StudyFetch AI Tutor, ensure you have the following prerequisites:

Node.js and npm
- Node.js version 18.x or later
- npm version 8.x or later or yarn version 1.22.x or later

Database
- PostgreSQL (version 13 or later)
- Or a PostgreSQL-compatible service like Supabase or Neon
- Blob storage (using vercel blob for storing the files)

API Keys
- Anthropic API key (for Claude model)
- Database API Key
- Blob API Key (for storing PDF files and all)
- NextAuth secret key

Development Environment
- Git
- Code editor *(VS Code recommended)*
- Terminal/Command Prompt

Installation

Follow these steps to install and set up the StudyFetch AI Tutor application:

Clone the Repository

```
git clone https://github.com/your-username/studyfetch-ai-tutor.git
cd studyfetch-ai-tutor
```

Install Dependencies

```
npm install
```

Set Up Environment Variables

Create a `.env` file in the root directory with the following variables:

```
DATABASE_URL="postgresql://username:password@localhost:5432/xx"
AUTH_SECRET="your-auth-secret-key"
BLOB_READ_WRITE_TOKEN="your-blob-storage-token"
ANTHROPIC_API_KEY="your-anthropic-ai-api-key"
ALLOW_USER_REGISTRATION="true"
```

Initialize the Database

```
npx prisma migrate dev --name init
```

Database Configuration

The application uses Prisma ORM to interact with the database. The schema is defined in:

```
prisma/schema.prisma
```

Key models include:

- User: User accounts and authentication
- Document: Uploaded PDF documents and metadata
- ChatSession: Conversation sessions linked to documents
- Message: Individual messages within chat sessions

You can customize this schema to add new fields or relationships. And follow the command

```
npx prisma migrate dev --name init
```

Authentication Configuration

Authentication is handled by NextAuth.js the default is email/password. and configured in:

```
auth.ts
```

Running the Application

Development Mode

```
npm run dev
```

Production Build

```
npm run build
```

Core Components

Authentication System

The authentication system in StudyFetch AI Tutor is built on NextAuth.js, providing secure user management and session handling. It implements a credential-based authentication flow with email and password. Features like verify email, password change and OTP based login are not yet implemented.

Key Features:

- User Registration: New users can create accounts with email and password
- Login/Logout: Secure authentication with session management
- Session Persistence: Maintains user sessions across page reloads.
- Protected Routes: Restricts access to authenticated users only

Document Processing

The document processing component handles PDF uploads, storage, and text extraction. It forms the foundation for AI tutoring capabilities by making document content available for analysis. It uses a combination of client-side and server-side components. The upload process is handled by API routes, while text extraction uses PDF.js on the client side. Right now using the PDF.js but we can leverage the anthropic AI itself to parse the pdf content and make into embedded document.

PDF Viewer

The PDF viewer component (ParsedPDFViewer) is a custom-built interactive viewer that displays the processed document content with advanced features for navigation, highlighting, and annotation.

Key Features:

- Page Navigation: Move between document pages
- Text Highlighting: Highlight important text with different styles
- Search Functionality: Find and navigate to specific text
- Annotations: Add notes to specific sections by AI
- Zoom Controls: Adjust the document view size
- Interactive Elements: Respond to AI commands for navigation

```
// Example PDF viewer interface
export interface ParsedPDFViewerHandle {
  gotoPage: (page: number, options?: { blink?: boolean }) => void;
  highlight: (term: string, page?: number, style?: HighlightStyle) => void;
  scrollToPosition: (page: number, term: string, annotation?: string) => void;
  processNewAnnotations: (annotations: Annotation[]) => void;
}
```

Chat Interface

The chat interface provides the primary interaction point between users and the AI tutor. It displays conversation history, handles message input, and processes AI responses with special formatting and commands.

Key Features:

- Message Display: Shows user and AI messages with appropriate styling
- Message Input: Allows users to type questions and commands
- Markdown Rendering: Formats AI responses with proper styling
- Command Processing: Interprets special commands in AI responses
- Auto-scrolling: Keeps the latest messages visible

AI Integration

The AI integration component connects the application to AI models like Google's Gemini or Anthropic's Claude, handling prompt construction, response processing, and command generation.

Key Features:

- Model Selection: Configurable AI model choice
- Context Management: Provides document content and conversation history
- Prompt Engineering: Constructs effective prompts for the AI
- Response Processing: Parses and formats AI responses
- Command Generation: Creates navigation and highlighting commands
- Error Handling: Gracefully handles API failures and limits