AI EduTutor - Project Documentation

This document provides a comprehensive overview of the AI EduTutor project, including its features, technology stack, project structure, and instructions for local setup and deployment.

Live Demo

The application is currently deployed and accessible at:
 https://v0-edututuorai-clone.vercel.app/

Table of Contents

1. Project Overview

AI EduTutor is a full-stack web application that empowers users to generate high-quality educational content using Artificial Intelligence. The app allows users to specify parameters such as topic, word count, target audience, and difficulty level to generate personalized content including articles, lesson plans, or summaries.

It supports user authentication, content history tracking, and a credit-based usage system.

2. Key Features

- User Authentication
  Secure sign-in/sign-up via Clerk, including social logins like Google.
- AI Content Generation
  Generate educational content based on:
  - Topic/Title
  - Number of Words
  - Target Audience (Student, Professional, etc.)
  - Difficulty Level (Easy, Medium, Hard)
- Rich Text Editor
  View and copy generated content in a user-friendly editor.
- Content History
  Sidebar with chronological access to previously generated content.
- Usage Tracking
  Word-based credit system to monitor user activity with a live "Words Left" indicator.
- Responsive Design
  Fully optimized for desktop and mobile devices.

3. Technology Stack

Layer	Tool/Library
Framework	Next.js 14 (App Router)
Language	TypeScript
Styling	Tailwind CSS
UI Components	Shadcn/UI + Radix UI + Lucide React
Auth	Clerk
Database	Neon (Serverless PostgreSQL)
ORM	Drizzle ORM

AI Integration	Vercel AI SDK + Google Gemini Pro API

Deployment Layer	Vercel Tool/Library

4. Project Structure

```
/
├── app/
│   ├── (auth)/                # Sign-in / Sign-up routes
│   ├── (dashboard)/           # Authenticated dashboard views
│   │   ├── layout.js          # Dashboard layout (sidebar, header)
│   │   └── page.js            # Main dashboard content
│   ├── api/
│   │   └── ai-content/        # AI content generation logic
│   └── layout.js              # Root layout file
│
├── components/                # Reusable UI components
│   ├── Header.jsx
│   ├── HistorySection.jsx
│   └── SearchSection.jsx
│
├── drizzle/                   # Migration files
├── lib/                       # Database and schema logic
│   ├── db.js
│   └── schema.js
├── utils/                     # AI model interaction
│   └── ai-model.js
├── public/                    # Static files
├── .env.local.example         # Environment variable template
├── drizzle.config.js          # ORM config
├── next.config.mjs            # Next.js config
└── package.json               # Dependencies and scripts
```

5. Setup and Installation (Local Development)

✅ Prerequisites

- Node.js (v18+)
- Git
- npm / yarn / pnpm

Clone the Repository

```
git clone <your-repository-url>
cd <repository-name>
```

Install Dependencies

```
npm install
```

Set Up Environment Variables

Create a `.env.local` file:

```
cp .env.local.example .env.local
```

Fill in values from your dashboards:

```
# Clerk Authentication
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=pk_test_...
CLERK_SECRET_KEY=sk_test_...

# Neon PostgreSQL
DATABASE_URL="postgresql://user:password@host/dbname?sslmode=require"

# Google Gemini API
GEMINI_API_KEY=AIzaSy...
```

Push Database Schema

```
npx drizzle-kit push
```

Run the Development Server

```
npm run dev
```

Visit: http://localhost:3000

6. Core Workflows

Authentication Flow

- Clerk middleware protects all `/dashboard` routes.
- Unauthenticated users are redirected to `/sign-in`.
- Clerk provides `<SignIn />` and `<SignUp />` components.
- Session data is accessible via Clerk hooks across the app.

Content Generation Flow

1. User Input: Form filled in `SearchSection.jsx`.
2. API Call: POST request sent to `/api/ai-content`.
3. Server Logic:
    - Request processed and prompt structured in `ai-model.js`.
    - Sent to Google Gemini API via SDK.
4. Streaming Response:
    - Vercel AI SDK streams response in real-time to client.
5. Database Update:
    - Drizzle ORM updates content history and word usage in Neon DB.

7. Deployment

Vercel Setup

1. Push to GitHub
   Ensure your project is under version control.
2. Import on Vercel
   Connect your GitHub repo to Vercel.
3. Configure Environment Variables
   Add the `.env.local` values to your Vercel project settings.
4. Deploy
   Automatically `main` branch trigger automatic builds and redeployment.
   Commits to the