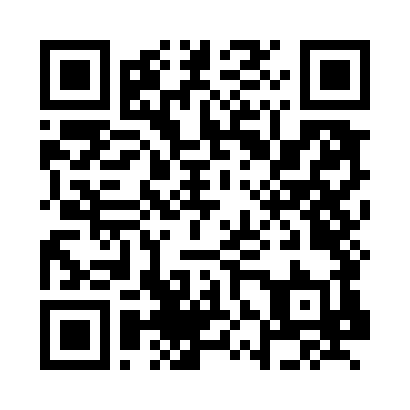
**TextGen-AI (Artificial-intelligence)**

**Just Scan To Get The Source Code****.**

**Abstract**

TextGen-AI is a full-stack MERN application that combines a modern React/Tailwind frontend with an Express/MongoDB backend to deliver an AI-powered text generation and summarization service. The system provides user authentication, contact form/email delivery, server-side summarization using Google’s Generative AI (Gemini), and an SSE streaming endpoint to deliver long or real-time model responses. Designed for interactive conversational workflows, the app emphasizes UX polish (particles, 3D globe, responsive auth pages) and pragmatic backend patterns (Mongoose models, Nodemailer, streaming abort control). TextGen-AI serves as a practical prototype for integrating cloud generative models into web platforms; recommended next steps include securing and documenting required environment variables (API keys, SMTP creds), adding tests and CI, and hardening error handling and rate-limit/usage controls before production deployment.

**What is TextGen-AI**

TextGen-AI is a web-based application that combines traditional web application features (user authentication, contact forms) with advanced AI conversational capabilities. The system provides two distinct chat interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Chat Type | Location | Features | Limitations |
| Main Chat | /Chat/index.html | Full history, import/export, code viewing, unlimited messages | Requires authentication |
| Trial Chat | /Temp/chat.html | Basic chat functionality | Limited to 5 messages, no authentication required |

The application demonstrates modern full-stack development practices including JWT-based authentication, real-time streaming with Server-Sent Events (SSE), and responsive UI design with Tailwind CSS.

**Core Capabilities**

The system provides four primary capability groups:

|  |  |  |
| --- | --- | --- |
| Capability | Implementation | Key Components |
| User Authentication | JWT-based stateless auth with bcrypt password hashing | **routes/auth.js**, **models/User.js**, localStorage tokens |
| Contact Management | Email-based contact form with SMTP delivery | Inline route in **server.js**, Nodemailer transporter |
| AI Chat | Streaming chat with Gemini 2.5 Flash, code extraction, history management | **EnhancedGeminiChat** class, **/api/stream** endpoint, SSE |
| Public Content | React SPA with static pages and standalone chat interfaces | **HomePage**, **DeveloperPage**, **AuthPage** components |

**Prerequisites**

Before beginning installation, ensure the following are installed on your system:

|  |  |  |
| --- | --- | --- |
| Requirement | Minimum Version | Purpose |
| Node.js | **18.0.0+** | **Runtime for both frontend build tools and backend server** |
| npm | **6.0.0+** | **Package manager for dependency installation** |
| MongoDB | **4.0+** | **Database for user account persistence** |
| Git | **Any recent** | **Version control and repository cloning** |

**Technology Stack**

**Frontend Stack**

The frontend is built with React and Vite, utilizing modern tooling for fast development and optimized production builds:

|  |  |  |
| --- | --- | --- |
| Technology | Purpose | Key Files |
| React 18.3 | UI component framework | **frontend/src/main.jsx**, **frontend/src/App.jsx** |
| Vite 5.4 | Build tool and dev server | **frontend/vite.config.js** |
| React Router 7.1 | Client-side routing | **frontend/src/App.jsx** (route definitions) |
| Tailwind CSS 3.4 | Utility-first styling | **frontend/tailwind.config.js**, **frontend/src/index.css** |
| Axios 1.7 | HTTP client for API calls | Used throughout components for backend communication |
| React Globe GL | 3D globe visualization | **frontend/src/components/HomePage/HeroSection.jsx** |
| tsParticles | Particle animation effects | Homepage background animations |

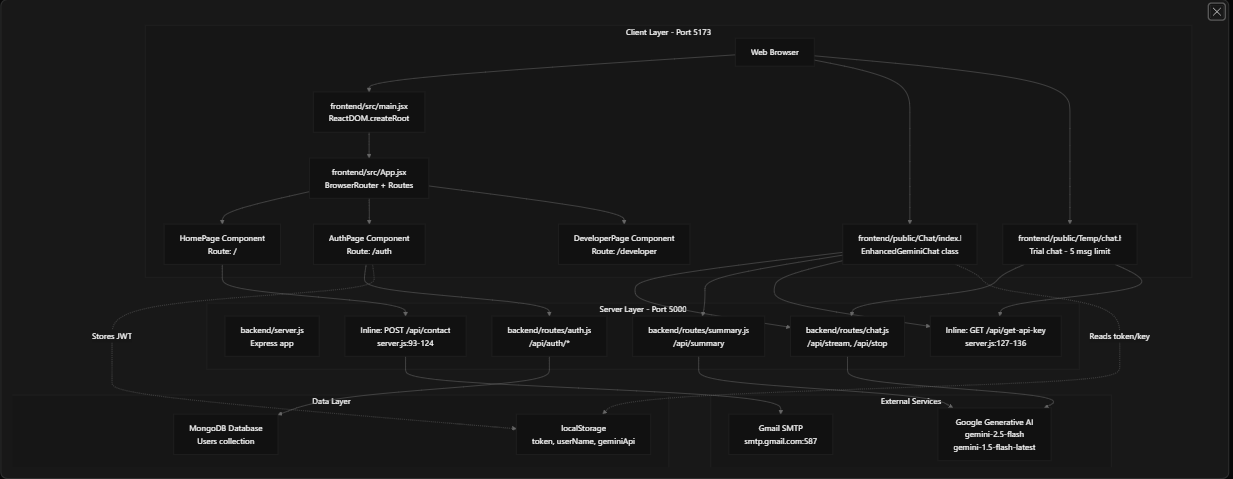
**Backend Stack**

The backend implements a RESTful API server with Express and MongoDB for data persistence:

|  |  |  |
| --- | --- | --- |
| Technology | Purpose | Key Files |
| Node.js | JavaScript runtime | **backend/server.js** (entry point) |
| Express 4.21 | Web framework and routing | **backend/server.js**, **backend/routes/\*** |
| MongoDB | NoSQL database | Connection in **backend/server.js:14-23**() |
| Mongoose 8.8 | MongoDB ODM | **backend/models/User.js** |
| JWT (jsonwebtoken) | Token-based authentication | **backend/routes/auth.js:17-23,48-54**() |
| Bcrypt.js 2.4 | Password hashing | **backend/models/User.js:17-24**() (pre-save hook) |
| Nodemailer 6.9 | Email delivery | **backend/server.js:33-41**() (Gmail SMTP) |
| @google/generative-ai 0.21 | Gemini API SDK | **backend/routes/chat.js**, **backend/routes/summary.js** |

**System Architecture Overview**

The following diagram maps the three-tier architecture to actual code entities and deployment components:



**Architecture Characteristics:**

* **Hybrid Frontend**: Mix of React Router SPA (**/**, **/auth**, **/developer**) and standalone HTML pages (**/Chat/index.html**, **/Temp/chat.html**)
* **Modular Backend**: Authentication logic separated into **routes/auth.js**, while utility endpoints (contact, API key) are inline in **server.js**
* **Stateless Authentication**: JWT tokens stored in **localStorage**, validated on backend via **jsonwebtoken.verify()**
* **Fail-Fast Initialization**: Server exits immediately if **MONGO\_URI** is missing or database connection fails.

**System Entry Points**

The application has distinct entry points for frontend and backend systems:



**Startup Sequence:**

1. **Backend Startup** (**npm start** in **backend/** directory):
   * Load environment variables from **.env.**
   * Connect to MongoDB using **MONGO\_URI.**
   * Initialize Nodemailer transporter with email credentials.
   * Register route handlers.
   * Start Express server on port 5000.
2. **Frontend Startup** (**npm run dev** in **frontend/** directory):
   * Vite dev server starts on port 5173.
   * Serves **index.html** as entry point.
   * React renders via **main.jsx** mounting to **#root**.
   * React Router initializes and renders initial route.
3. **Chat Access**:
   * Main chat: Direct file access at **/Chat/index.html** (checks for auth token on load)
   * Trial chat: Direct file access at **/Temp/chat.html** (no auth required)

**Installation**

**Step 1: Clone Repository**

**git clone https://github.com/AlwaysDhruv/TextGen-AI-Node.js.git**

**cd TextGen-AI-Node.js**

**Step 2: Backend Installation**

**Navigate to the backend directory and install dependencies:**

**cd backend**

**npm install**

**This installs the following critical dependencies as defined in backend/package.json :**

|  |  |  |
| --- | --- | --- |
| Package | Version | Purpose |
| express | **^5.1.0** | **Web server framework** |
| mongoose | **^8.18.1** | **MongoDB ODM for User model** |
| jsonwebtoken | **^9.0.2** | **JWT token generation and verification** |
| bcryptjs | **^3.0.2** | **Password hashing (pre-save hooks)** |
| nodemailer | **^7.0.6** | **Email delivery for contact form and OTP** |
| @google/generative-ai | **^0.24.1** | **Gemini API SDK for chat functionality** |
| dotenv | **^17.2.3** | **Environment variable loading** |
| cors | **^2.8.5** | **Cross-origin request handling** |
| dompurify | **^3.2.7** | **XSS sanitization on backend** |

**Step 3: Frontend Installation**

**In a new terminal, navigate to the frontend directory:**

**cd frontend**

**npm install**

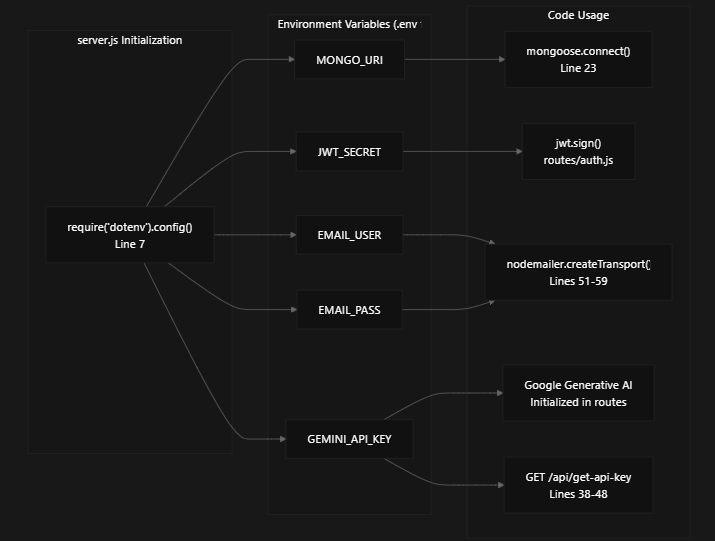
**This installs dependencies from frontend/package.json :**

|  |  |
| --- | --- |
| Package | Purpose |
| react + react-dom (^19.1.1) | **UI library** |
| react-router-dom (^7.9.1) | **Client-side routing** |
| axios (^1.12.2) | **HTTP client for API calls** |
| @tailwindcss/vite (^4.1.13) | **Styling framework integration** |
| vite (^7.1.2) | **Build tool and dev server** |
| react-globe.gl (^2.36.0) | **3D globe visualization** |
| tsparticles + @tsparticles/react | **Particle background effects** |

**Environment Configuration**

**Backend Environment Variables**

**Create a .env file in the backend/ directory. This file is excluded from version control by**



**Required Environment Variables**

**Create backend/.env with the following structure:**

**MONGO\_URI=mongodb://localhost:27017/textgen-ai**

**JWT\_SECRET=your-secret-key-here**

**EMAIL\_USER=your-gmail@gmail.com**

**EMAIL\_PASS=your-app-specific-password**

**GEMINI\_API\_KEY=your-gemini-api-key-here**

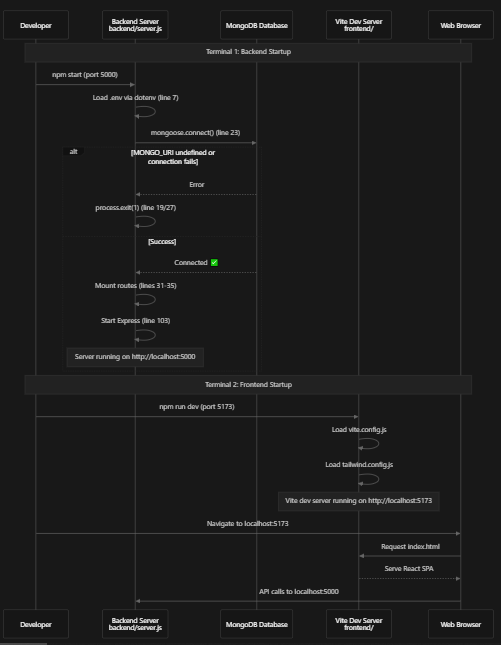
**PORT=5000**

**Detailed Configuration:**

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Usage Location | Purpose | Validation |
| MONGO\_URI | [**backend/server.js17-20**](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/backend/server.js#L17-L20)[**backend/server.js23**](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/backend/server.js#L23-L23) | **MongoDB connection string** | **Required - Server exits if undefined** |
| JWT\_SECRET | **routes/auth.js (JWT signing)** | **Secret for JWT token generation** | **Required for authentication** |
| EMAIL\_USER | [**backend/server.js56**](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/backend/server.js#L56-L56) | **Gmail SMTP username** | **Required for contact form** |
| EMAIL\_PASS | [**backend/server.js57**](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/backend/server.js#L57-L57) | **Gmail app-specific password** | **Required for email delivery** |
| GEMINI\_API\_KEY | [**backend/server.js39**](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/backend/server.js#L39-L39) | **Google Gemini API key** | **Checked at**[**backend/server.js40-46**](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/backend/server.js#L40-L46) |
| PORT | [**backend/server.js10**](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/backend/server.js#L10-L10) | **Server port (defaults to 5000)** | **Optional** |

**Starting the Development Environment**

**The application requires two concurrent processes: backend API server and frontend development server.**



**Terminal 1: Start Backend Server**

**cd backend**

**npm start**

Expected console output :

**✅ MongoDB connected**

**🚀 Server running on port: 5000**

**Troubleshooting:**

* If you see **❌ FATAL ERROR: MONGO\_URI is not defined**, verify **.env** file exists in **backend/** directory
* If MongoDB connection fails, ensure MongoDB is running: **mongod --version** or start service.

**Terminal 2: Start Frontend Development Server**

**cd frontend**

**npm run dev**

Expected output indicates Vite server started on **http://localhost:5173** (configurable in **vite.config.js**).

The frontend dev server:

* Serves React SPA from frontend/src/main.jsx
* Enables Hot Module Replacement (HMR) for instant updates
* Processes Tailwind CSS via frontend/tailwind.config.js
* Proxies API calls to backend (configured as needed)

**Verification & Testing**

**Backend Health Check**

Verify backend is running:

curl <http://localhost:5000>

Expected response (from [backend/server.js98-100](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/backend/server.js#L98-L100)):

✅ TextGen-AI Server is running and healthy.

**Debug Route Inspection**

List all registered routes:

curl <http://localhost:5000/api/debug-routes>

This endpoint is defined at [backend/server.js89-95](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/backend/server.js#L89-L95) and returns JSON array of registered routes.

**Frontend Access**

Navigate to **http://localhost:5173** in your browser. You should see:

1. **Particle background** - Initialized from [frontend/src/index.css38-45](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/frontend/src/index.css#L38-L45)
2. **Navigation bar** - Rendered by **NavigationBar** component
3. **Landing page** - **HomePage** component with hero section

**API Integration Test**

Test API key distribution endpoint:

curl <http://localhost:5000/api/get-api-key>

Expected response structure (from [backend/server.js38-48](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/backend/server.js#L38-L48)):

{

"apiKey": "your-gemini-api-key"

}

If API key is not configured, returns:

{

"success": false,

"message": "API key is not configured on the server."

}

**Next Steps**

After successful setup:

1. **Explore the Application:**
   * Navigate to **/auth** for user registration/login flows
   * Try the trial chat at **/Temp/chat.html** (limited to 5 messages)
   * Access full chat at **/Chat/index.html** (requires authentication)
2. **Read Architecture Documentation:**
   * [System Architecture](https://deepwiki.com/AlwaysDhruv/TextGen-AI-Node.js/3-system-architecture) - Understand the three-tier MERN stack
   * [Authentication System](https://deepwiki.com/AlwaysDhruv/TextGen-AI-Node.js/4-authentication-system) - Learn about JWT-based auth flows
   * [AI Chat System](https://deepwiki.com/AlwaysDhruv/TextGen-AI-Node.js/5-ai-chat-system) - Understand Gemini API integration and streaming
3. **Review API Endpoints:**
   * [Authentication Endpoints](https://deepwiki.com/AlwaysDhruv/TextGen-AI-Node.js/7.1-authentication-endpoints) - User signup, login, OTP reset
   * [Chat Endpoints](https://deepwiki.com/AlwaysDhruv/TextGen-AI-Node.js/7.2-chat-endpoints) - Streaming, summarization, and control
   * [Utility Endpoints](https://deepwiki.com/AlwaysDhruv/TextGen-AI-Node.js/7.3-utility-endpoints) - Contact form, API key distribution
4. **Development Workflow:**
   * [Build Configuration](https://deepwiki.com/AlwaysDhruv/TextGen-AI-Node.js/9.1-build-configuration) - Vite and Tailwind setup
   * [Dependencies](https://deepwiki.com/AlwaysDhruv/TextGen-AI-Node.js/9.2-dependencies) - Full dependency reference
   * [Development Workflow](https://deepwiki.com/AlwaysDhruv/TextGen-AI-Node.js/9.3-development-workflow) - Hot reloading, debugging, testing

**Common Issues & Solutions**

|  |  |  |
| --- | --- | --- |
| Issue | Cause | Solution |
| process.exit(1) on backend start | Missing **MONGO\_URI** in **.env** | Create **backend/.env** with all required variables |
| SMTP authentication failed | Using regular Gmail password | Generate app-specific password from Google Account settings |
| Frontend can't reach backend | CORS or wrong port | Verify backend running on port 5000, check CORS config at [backend/server.js13](https://github.com/AlwaysDhruv/TextGen-AI-Node.js/blob/c38131cf/backend/server.js#L13-L13) |
| MongoDB connection timeout | MongoDB not running | Start MongoDB service: **mongod** or via service manager |
| Gemini API errors in chat | Invalid **GEMINI\_API\_KEY** | Verify API key at <https://makersuite.google.com/app/apikey> |