Chatbot Web Project Documentation

**1. Project Requirements**

**Software & Accounts:**

- Node.js: JavaScript runtime environment required to run backend code and development tools on your computer.

- npm: Node.js package manager (comes with Node.js) used to install project dependencies.

- Vercel account: Required for deploying your project to the Vercel platform and for using the Vercel CLI for local development.

- OpenAI API key: Needed to access OpenAI's language models for generating chatbot responses.

- Supabase account: Required for database storage of conversations and user sessions.

**Project Files:**

- index.html — The frontend (chat UI)

- api/chat.js — The backend (Vercel serverless function for chat with Supabase integration)

- package.json — Project dependencies

- .env — (You need to create this) Contains your API keys:

OPENAI\_API\_KEY=your\_openai\_api\_key\_here

SUPABASE\_URL=your\_supabase\_project\_url

SUPABASE\_SERVICE\_ROLE\_KEY=your\_service\_role\_key

Database Setup:

- Create a table named "Conversations" in Supabase with columns:

- id (auto-increment primary key)

- created\_at (timestamp)

- conversation\_id (text)

- messages (jsonb)

------------------------------------------------------------

**2. How to Set Up and Run Locally**

Step-by-Step Setup:

1. Clone or download the project to your computer.

2. Install dependencies:

npm install

3. Install Vercel CLI globally (if not already):

npm install -g vercel

4. Add your API keys to .env file:

- Create a file named .env in your project root.

- Add these lines (replace with your actual keys):

OPENAI\_API\_KEY=sk-xxxxxxxxxxxxxxxxxxxx

SUPABASE\_URL=https://your-project.supabase.co

SUPABASE\_SERVICE\_ROLE\_KEY=your\_service\_role\_key\_here

5. Run the project locally:

vercel dev

- The first time, you'll log in and set up the project.

- After that, just run vercel dev to start the local server.

6. Open your browser to http://localhost:3000 to use the chatbot.

------------------------------------------------------------

**3. How the Web Chatbot Works**

Frontend (index.html):

- Displays a chat interface (messages, input box, send button).

- When you type a message and hit send:

1. The message is shown in the chat window.

2. The message (and a session ID) is sent to the backend at /api/chat using fetch.

3. When a response is received from the backend, it's displayed as a bot message.

**Backend (api/chat.js):**

- Receives POST requests at /api/chat with { message, sessionId }.

- Connects to Supabase database to retrieve existing conversation history.

- If no conversation exists, creates a new one with system message.

- Sends the conversation to OpenAI's API (using your API key).

- Gets the AI's response and saves the complete conversation back to Supabase.

- Returns the bot's response to the frontend.

- Includes comprehensive error handling for database operations and API calls.

**Session Handling:**

- Each user gets a unique session ID (stored in their browser's localStorage).

- Conversations are persisted in Supabase database, maintaining context across sessions.

- Users can continue conversations even after closing and reopening their browser.

**Database Operations:**

- Conversations are stored in the "Conversations" table in Supabase.

- Each conversation is identified by conversation\_id (session ID).

- The complete message history is stored in the messages JSONB column.

- New conversations are inserted, existing ones are updated.

- Error handling ensures graceful failure if database operations fail.

------------------------------------------------------------

**4. How to Deploy to Vercel**

1. Push your project to GitHub (or another git provider).

2. Connect your repo to Vercel (via the Vercel dashboard).

3. Set your environment variables in Vercel's dashboard:

- OPENAI\_API\_KEY

- SUPABASE\_URL

- SUPABASE\_SERVICE\_ROLE\_KEY

4. Deploy!

- Vercel will serve your static files and run your /api/chat.js as a serverless function.

- The backend will connect to your Supabase database for conversation storage.

**Important Deployment Notes:**

- Environment variables must be set in Vercel's dashboard (Settings → Environment Variables).

- Make sure all three variables are set for "All Environments".

- After adding environment variables, redeploy the project to ensure they are applied.

- If you see "supabaseUrl is required" errors, check that SUPABASE\_URL is properly set in Vercel.

------------------------------------------------------------

5. Summary Diagram (Text Version)

User types message and clicks Send ->

Browser (index.html) sends POST /api/chat {message, sessionId} ->

Vercel Serverless (api/chat.js) retrieves conversation from Supabase ->

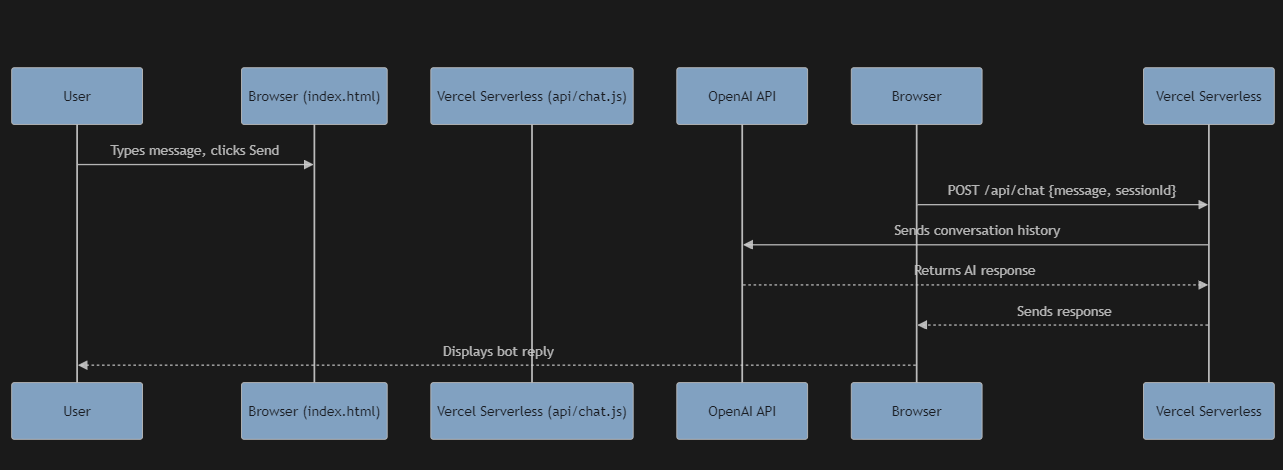
Vercel Serverless sends conversation to OpenAI API ->

OpenAI API returns AI response ->

Vercel Serverless saves updated conversation to Supabase ->

Vercel Serverless sends response to Browser ->

Browser displays bot reply to User



------------------------------------------------------------

**6. Key Points**

- All you need to run locally is Node.js, npm, Vercel CLI, OpenAI API key, and Supabase credentials.

- The frontend and backend are integrated: the frontend sends messages to the backend, which talks to OpenAI and stores conversations in Supabase.

- Vercel makes it easy to deploy and run both static frontend and serverless backend together.

- Conversations are now persistent and stored in a database, allowing for better user experience and conversation history.

- The system automatically handles database operations (create/update conversations) based on session IDs.

- Comprehensive error handling ensures the application remains stable even if database or API operations fail.

------------------------------------------------------------

**7. Dependencies**

Required npm packages:

- openai: For OpenAI API integration

- @supabase/supabase-js: For Supabase database operations

- cors: For handling cross-origin requests

- dotenv: For environment variable management

- express: For server functionality (if using traditional backend)

------------------------------------------------------------

**8. Environment Variables**

Required environment variables:

- OPENAI\_API\_KEY: Your OpenAI API key for AI responses

- SUPABASE\_URL: Your Supabase project URL

- SUPABASE\_SERVICE\_ROLE\_KEY: Your Supabase service role key for database access

These must be set both locally (.env file) and in Vercel's environment variables for deployment.

------------------------------------------------------------

9. Troubleshooting

Common Issues and Solutions:

\*\*500 Error on Vercel Deployment:\*\*

- Check that all environment variables are set in Vercel dashboard

- Ensure SUPABASE\_URL starts with "https://"

- Redeploy after adding environment variables

\*\*"supabaseUrl is required" Error:\*\*

- Verify SUPABASE\_URL is set correctly in Vercel environment variables

- Make sure the variable is set for "All Environments"

- Redeploy the project after fixing environment variables

\*\*Database Connection Issues:\*\*

- Verify your Supabase credentials are correct

- Check that the "Conversations" table exists with correct column names

- Ensure your service role key has proper permissions

\*\*Local vs Production Issues:\*\*

- Local development works with .env file

- Vercel deployment requires environment variables set in dashboard

- Always redeploy after changing environment variables

\*\*Error Handling:\*\*

- The updated api/chat.js includes comprehensive error handling

- Check Vercel logs for detailed error messages

- Database operations are wrapped in try-catch blocks

------------------------------------------------------------

10. Current Status

✅ \*\*Fully Functional Features:\*\*

- Local development with Vercel CLI

- Vercel deployment with Supabase integration

- Persistent conversation storage

- Session management across browser sessions

- Error handling and graceful failure recovery

- Environment variable management for both local and production

✅ \*\*Deployment Status:\*\*

- Frontend: Deployed and accessible via Vercel URL

- Backend: Serverless function working with Supabase

- Database: Conversations table storing all chat data

- Environment: All variables properly configured

The chatbot is now production-ready with full database persistence and error handling.