**AI Prompt Helper Extension – Project Summary**

**Overview**

AI Prompt Helper is a **Chrome Extension** that enables users to **store, manage, and retrieve AI prompts** efficiently. It features **template-based storage, history tracking, and integration with Supabase** as a backend.

The project has migrated from **Webpack** to **Vite** for faster builds and better performance.

**📌 Technology Stack**

* **Frontend:** React, TypeScript, Vite, Material UI
* **Backend:** Supabase (PostgreSQL, Auth, Storage)
* **Build System:** Vite + Rollup (Previously Webpack)
* **Chrome Extension API:** Background scripts, Content scripts, Popup UI
* **Deployment:** Hosted via Supabase (functions, storage)

**📌 Core Features**

**1. AI Prompt Templates**

* Users can create, store, and retrieve templates.
* Templates are stored in **Supabase Database**.

**2. User Authentication**

* Supabase **Auth** (email/password, OAuth).

**3. Prompt History**

* Users can track and retrieve past prompt usage.
* Stored in **Supabase Database**.

**4. Chrome Extension Architecture**

* **Popup UI**: User interacts with prompts.
* **Background Script**: Handles events and messaging.
* **Content Script**: Injects code into web pages for AI interactions.

**5. Cloud Storage & Hosting**

* Supabase **Database**: Stores prompts, history, and user data.
* Supabase **Storage**: Stores assets (if needed).
* Supabase **Functions**: Handles backend logic.

**📌 Deployment Details**

**Frontend (Chrome Extension)**

* Developed using **Vite + React**.
* Build using:
* npm run build
* Load unpacked extension from /dist.

**Backend (Supabase)**

* Hosted on **Supabase**
* Uses **PostgreSQL** for storage.
* Uses **Supabase Auth** for authentication.

**📌 File & Component Breakdown**

**📂 Key Files & Directories**

**Frontend (Chrome Extension)**

📂 src/

├📂 popup/ # Popup UI

│ ├📄 popup.html # Main UI for extension

│ ├📄 index.tsx # Renders React app in popup

│ └📄 AuthView.tsx # Handles authentication flow

│

├📂 components/

│ └📄 TemplateEditor.tsx # Component for editing templates

│

├📂 content/

│ └📄 content.ts # Injects scripts into webpages

│

├📂 background/

│ └📄 background.ts # Handles background events

│

├📂 types/

│ └📄 template.ts # Type definitions

│

├📂 userhistory/

│ └📄 history.json # Local user history (if needed)

│

├📂 usertemplates/

│ └📄 templates.json # Local template storage (deprecated)

**Backend (Supabase)**

* **Tables:**
  + users → Stores user accounts.
  + prompts → Stores saved AI prompts.
  + history → Tracks prompt usage history.
  + templates → Stores prompt templates.
* **Functions:**
  + getPrompts() → Fetches prompts from DB.
  + savePrompt() → Saves user prompts.
  + getUserHistory() → Retrieves user history.
  + authHandler() → Manages authentication.

**📌 Build & Development**

**Building the Chrome Extension**

npm run build

* Output stored in /dist
* Load **unpacked extension** from /dist in Chrome.

**Running in Development**

npm run dev

* Starts **Vite dev server**.

**📌 Vite Configuration (vite.config.ts)**

* **Alias @/ for src/**
* **Static copy plugin** (Copies manifest & assets)
* **Compression enabled** (gzip, Brotli)
* **Rollup for optimized builds**
* **PWA support** (optional)
* **Supabase integrated for backend**

**📌 How to Restart This Chat**

To resume development **at any point**, simply refer to:

1. **Project Summary (this document)**
2. **Latest Git commit history**
3. **Supabase database schema & functions**

**📌 Next Steps**

✅ **Confirm Supabase backend connection**  
✅ **Test CRUD operations with Supabase**  
✅ **Finalize extension packaging & manifest**  
✅ **Push latest changes to GitHub**

**👉 With this summary, you can restart the chat anytime & resume development! 🚀**

Let me know if you'd like any additions.