# JS Apps with Supabase – Exercises

**Exercises** and **homework** assignments for “Software Technologies with AI” course at SoftUni AI.

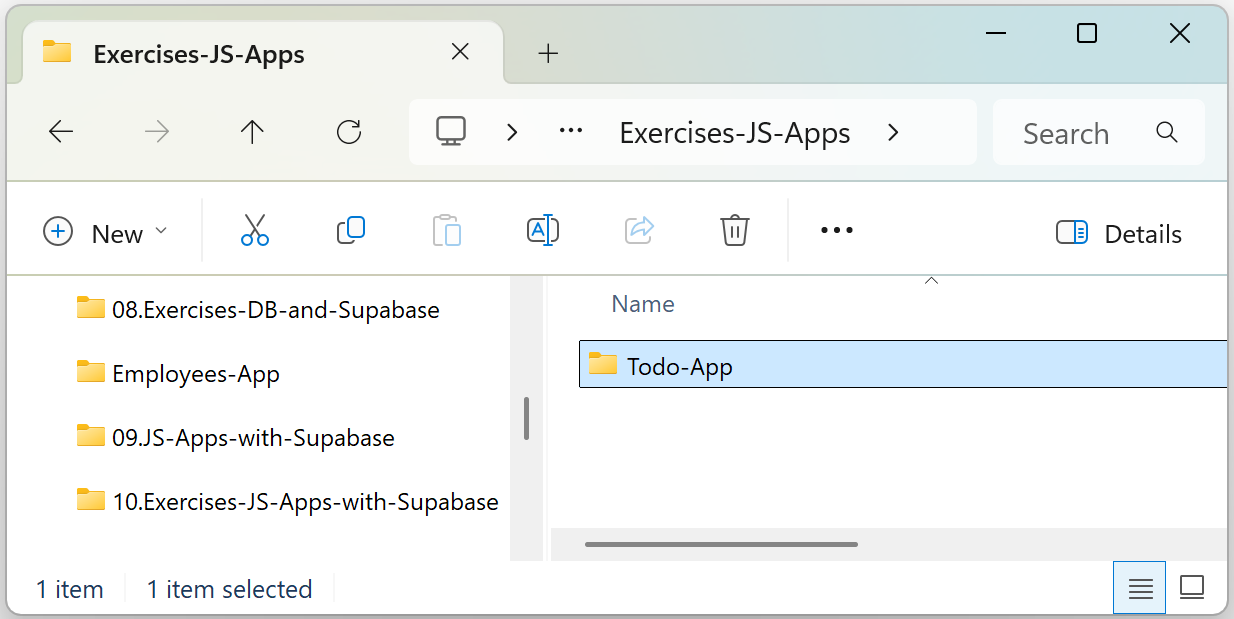
## TODO List with Users

In this exercise we shall **build** a very simple **to-do list app** with Supabase: users register, login and create, list, view, search, add, edit, delete their TODO’s.

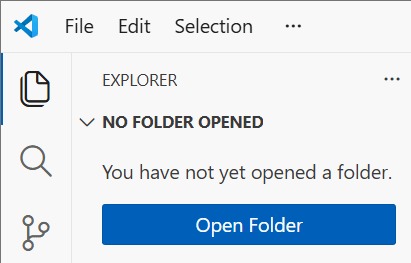
### New VS Code Workspace

Create a **new Workspace** in VS Code for your app, e.g. `Todo-App`.

Create a new **sub-folder** in the working folder for this exercise:



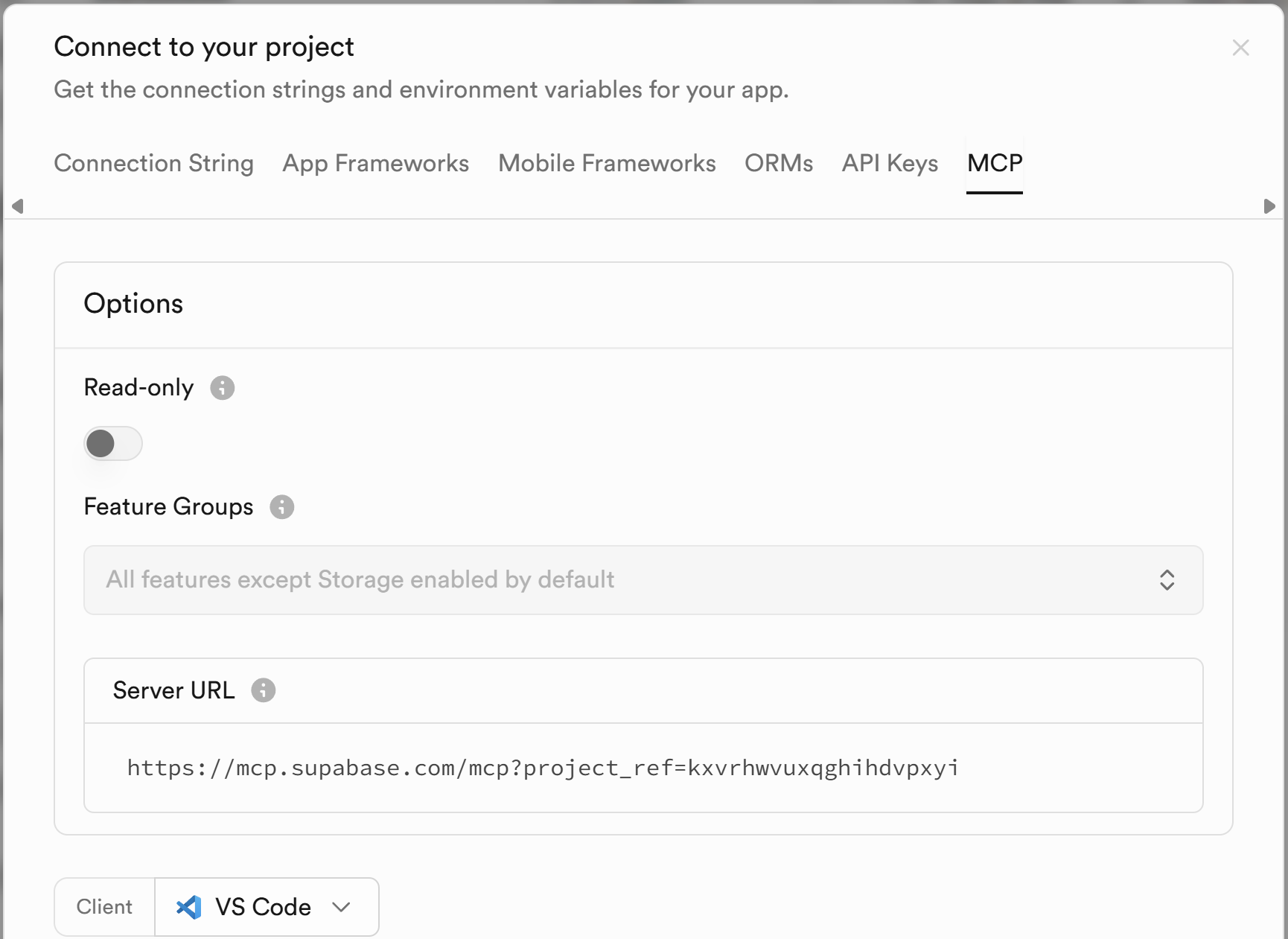
Open it in Visual Studio Code:

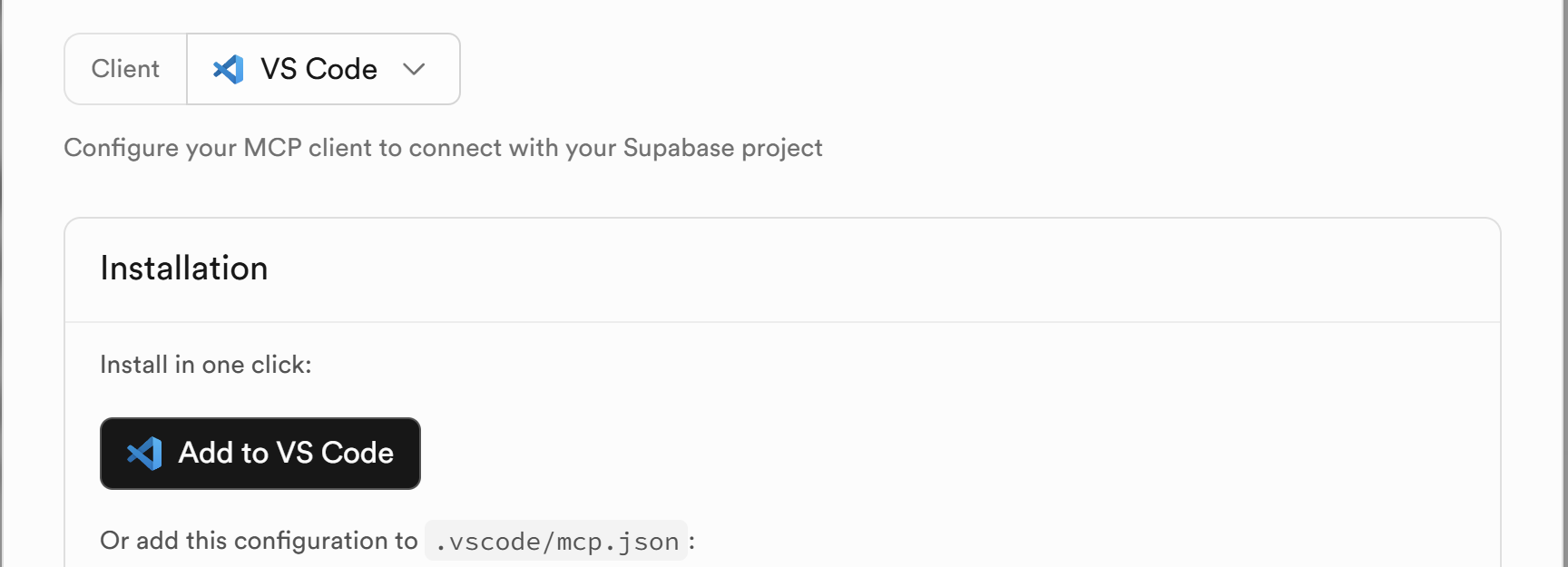
### New Supabase Project

Create a **new** **project in Supabase** for your app, e.g. `TodoDB`.

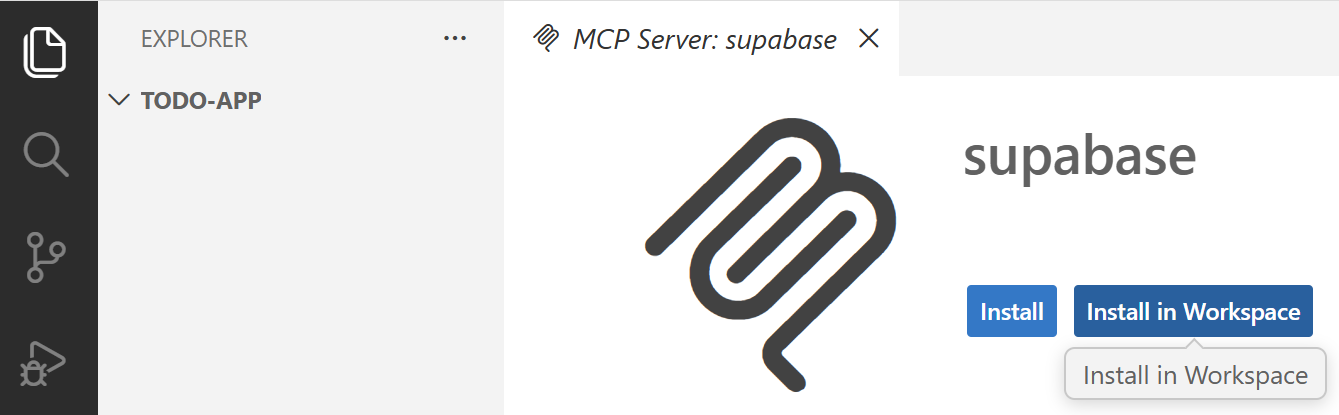
Install the **Supabase MCP**: **Supabase Dashboard** 🡪 **Connect** 🡪 **MCP**.



Select [**Add to VS Code**]:

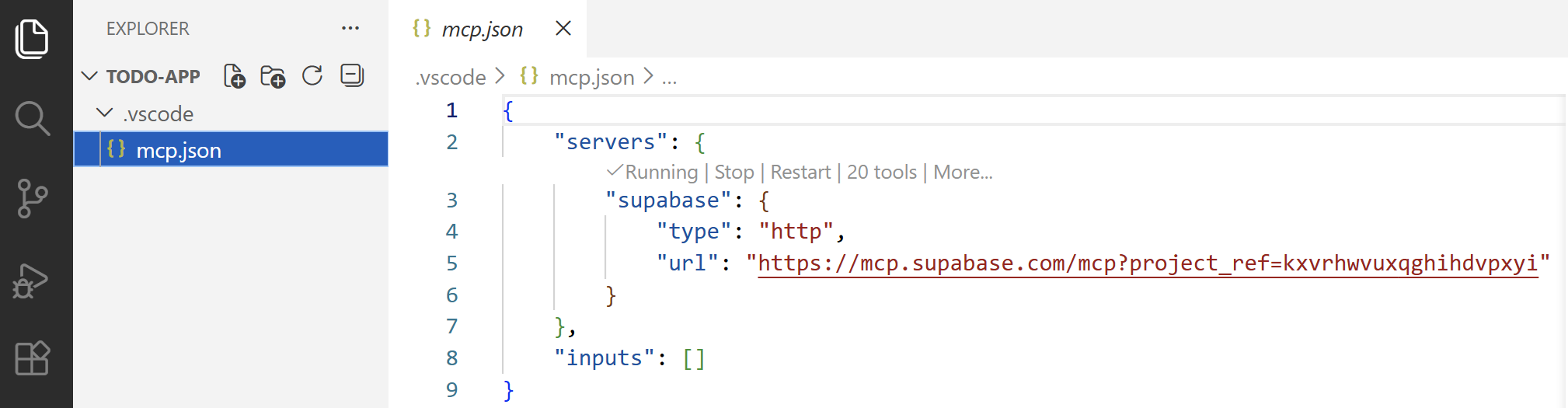


Press[**Install in Workspace**]:



**Authenticate** with **Supabase.com**.

You can check whether the **installation has been successful** in the new file. The MCP **status** should be "**Running**":



### Create the TODO App

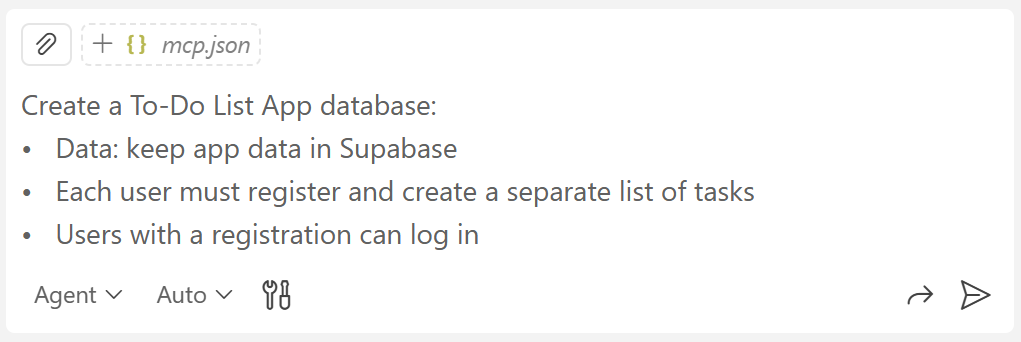
**Implement a To-Do List App**:

* **Data**: keep app data in **Supabase**
* Each user must **register**, and creates its own **list of tasks**
* Users with a registration can **log in**
* **App functionalities**:
  + users register
  + login
  + list
  + view
  + search
  + add
  + edit
  + delete TODOs
* List TODO tasks in a **table**, use **popups** for add / edit / delete confirm
* **Technologies**: use HTML, CSS, JavaScript and Bootstrap

#### The AI Prompts

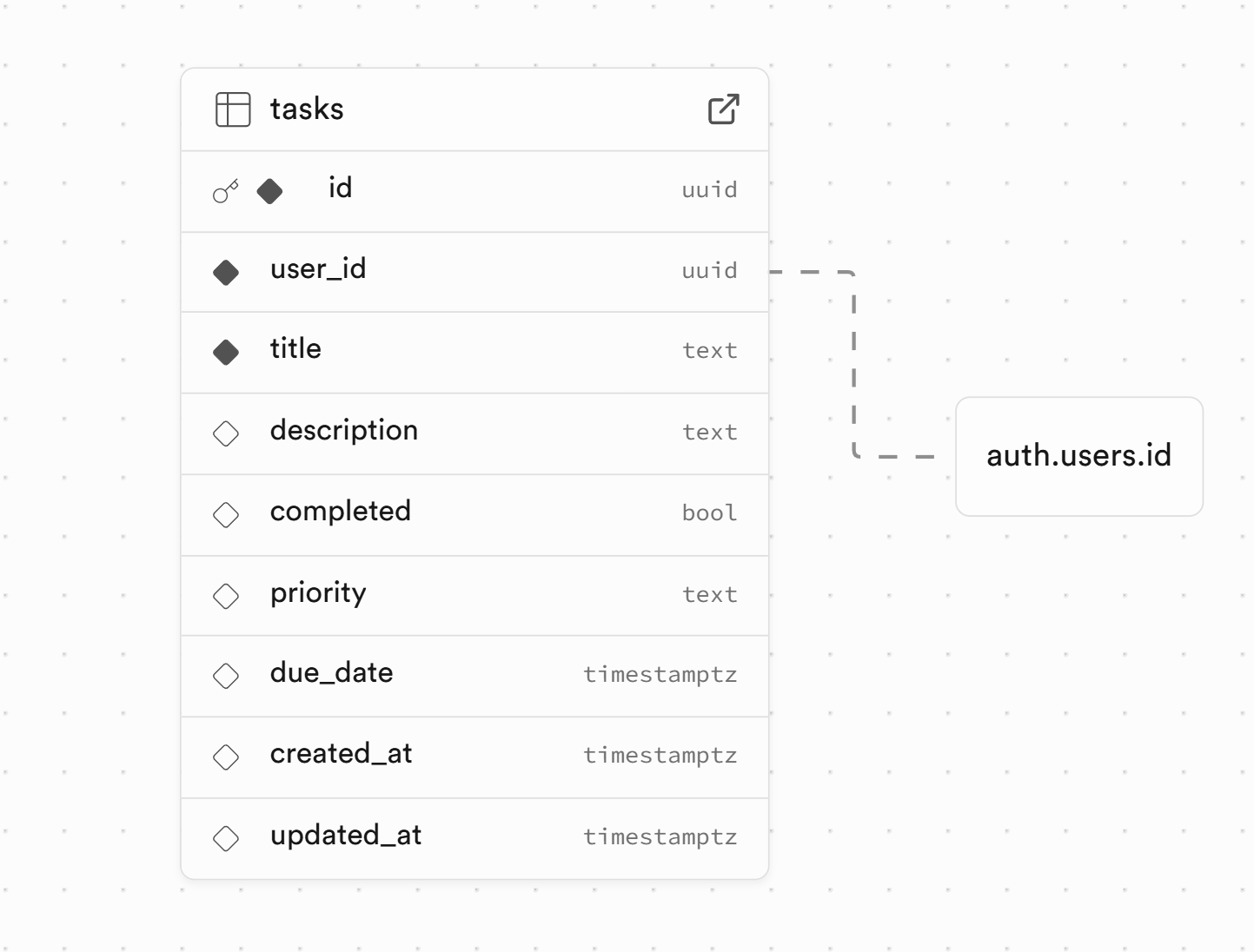
Let’s **divide the task into 3 stages**:

##### Stage 1: Create DB Tables



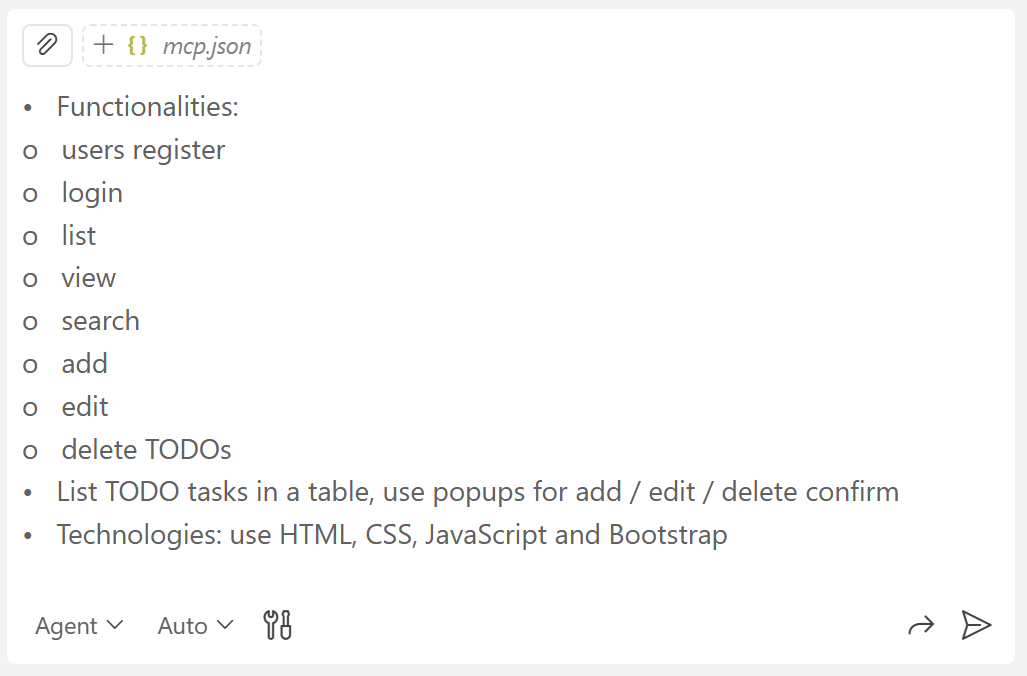
Check if the database has been created in Supabase.

There will be two **DB tables** connected to each other. Each **user** has **many tasks** with a description, completion state, priority level, due date, creation date and last update date:



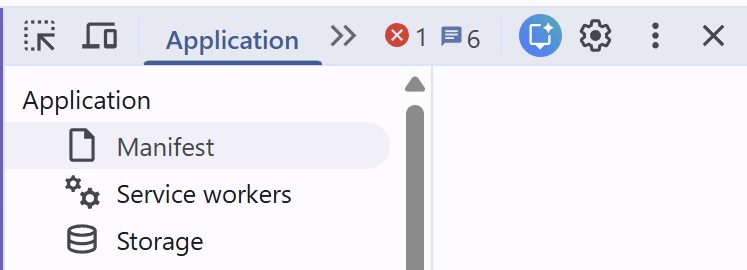
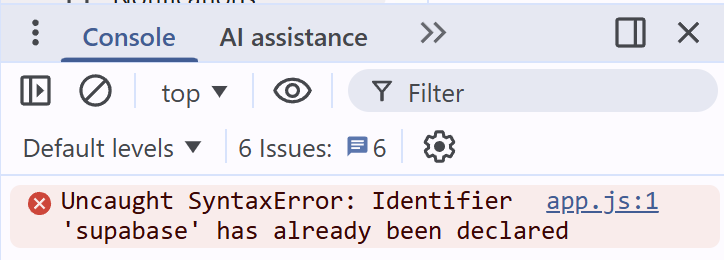
##### Stage 2: JS App Functionality

Create the application with appropriate **HTML, CSS, and JavaScript** use:



#### Bug Fixing

**Run the app** and check the browser console in [**Dev Tools**] for **errors**:

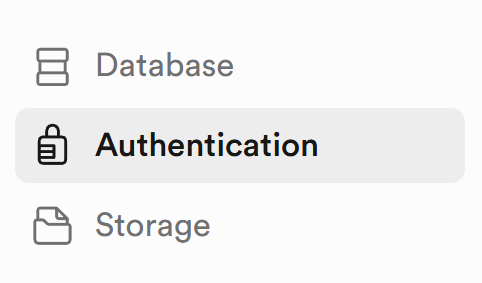
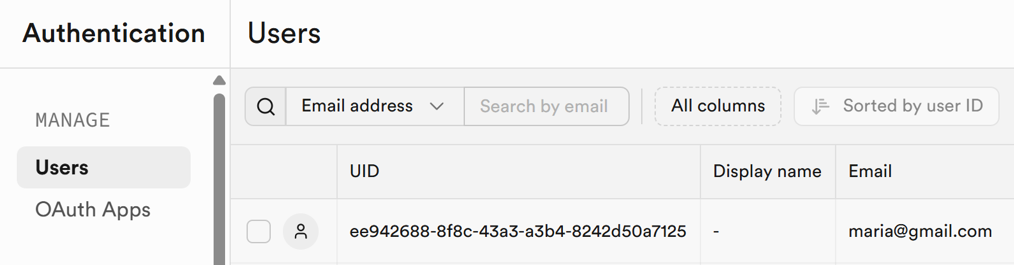
Ask GitHub Copilot to **fix the problem**: rename the duplicated variable `supabase`.

#### Register Users

After the bugs are fixed, you will be able to **register a new user in Supabase** from the app:

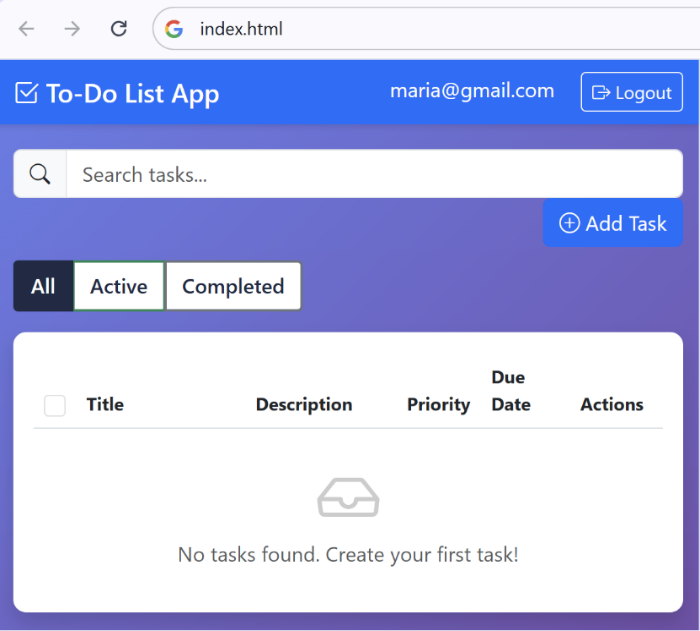


Look for the **registered** users in **Authentication** 🡪 **Users**:

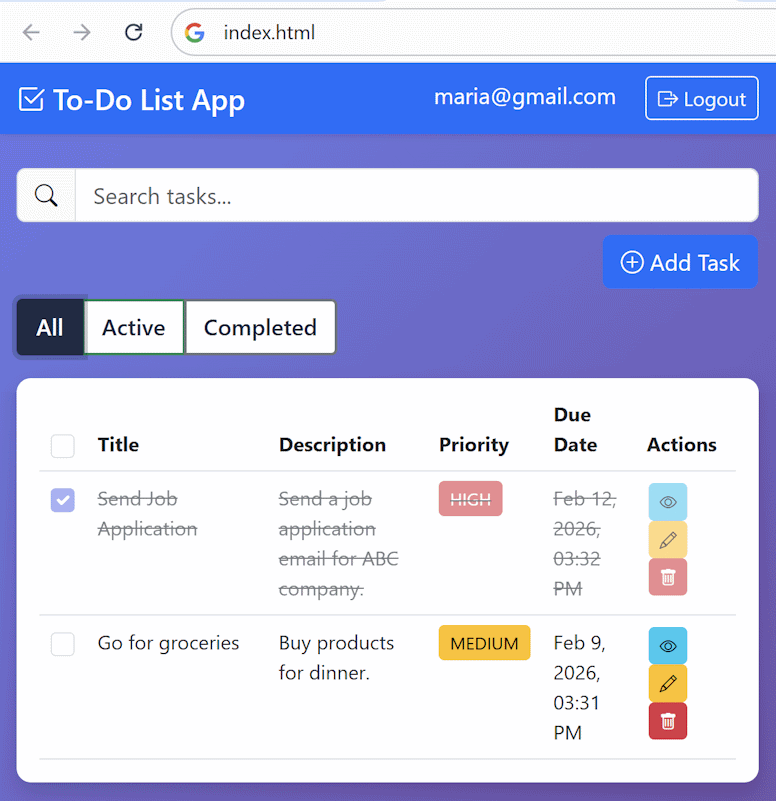
#### The App After Login

After login, you should see the **TODO list** app screen:



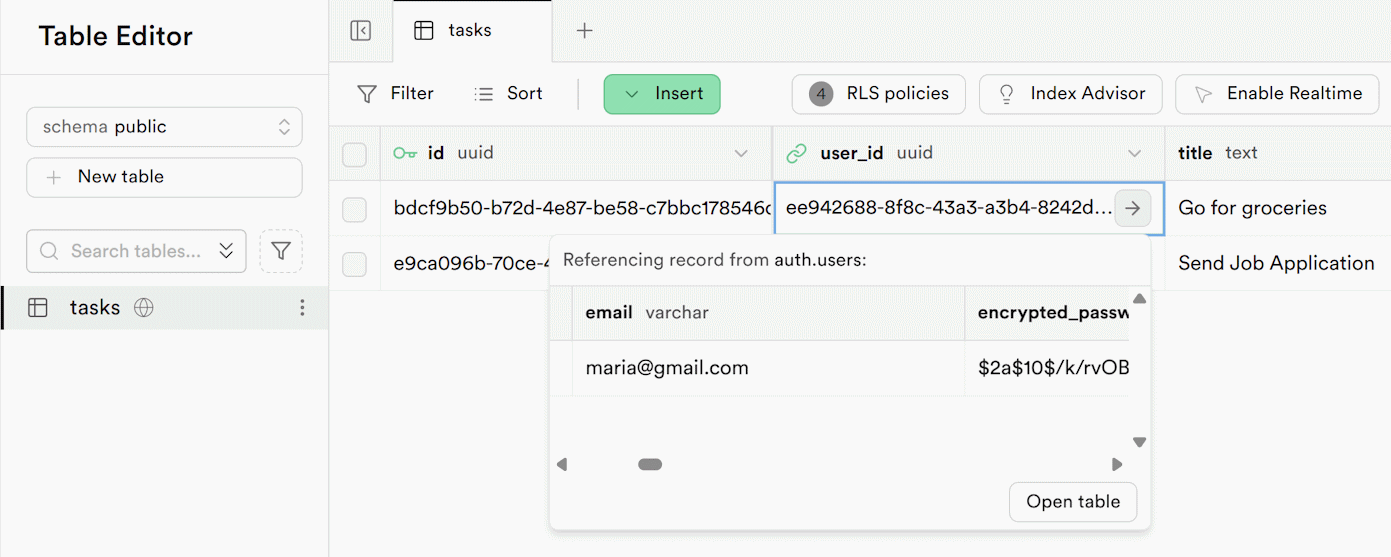
#### Create Tasks

Test if **creating tasks** works correctly:

#### Review Tasks in Supabase

**Open Table Editor** in Supabase and check if the **tasks have been inserted** successfully in the DB for the correct user.



#### Security Check

**Register another user**, log in and see if the new user can see the tasks of the first user.

## Birthday Cards Generator

In this exercise we shall **build a Birthday Cards Generator App**: users register, login, generate and share birthday cards (by template image + greetings text). **Template images** will be **static resources** in the app.

### New VS Code Workspace

Create a new **workspace in VS Code** for your app, e.g. `birthday-cards`.

### New Supabase Project

Create a new **project in Supabase** for your app, e.g. `CardsDB`.

### Install Supabase MCP

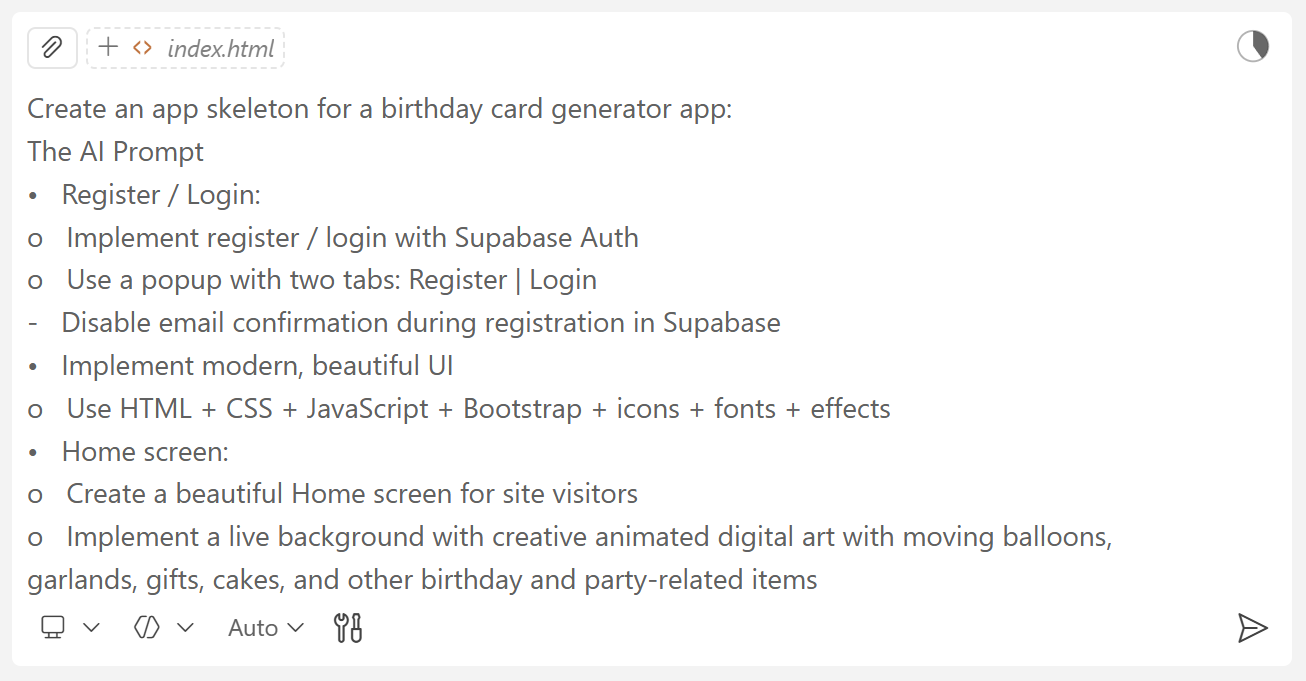
Install the **Supabase MCP**: **Supabase Dashboard** -> **Connect** -> **MCP** -> **VS Code** -> **Install in workspace** -> **Authenticate** with **Supabase.com**.

### Create App Skeleton with Auth

**Create an app skeleton** for а birthday card generator app with register / login / logout:

* **Register** / **Login**:
  + Implement register / login with Supabase Auth
  + Use a popup with two tabs: Register | Login
  + Disable email confirmation during registration in Supabase
* Implement **modern, beautiful UI**:
  + Use HTML + CSS + JavaScript + Bootstrap + icons + fonts + effects
* **Home screen**:
  + Create a beautiful Home screen for site visitors
  + Implement a live background with creative animated digital art with moving balloons, garlands, gifts, cakes, and other birthday and party-related items
  + Insert a big call to action: "Login to create a birthday card"
* **Generate Card screen**:
  + After login show the "Generate Card" screen
  + Keep it empty for now
* You can specify the files to be generated to improve structure and clarity. For example:
  + index.html
  + styles.css
  + config.js
  + auth.js
  + app.js

#### The AI Prompt



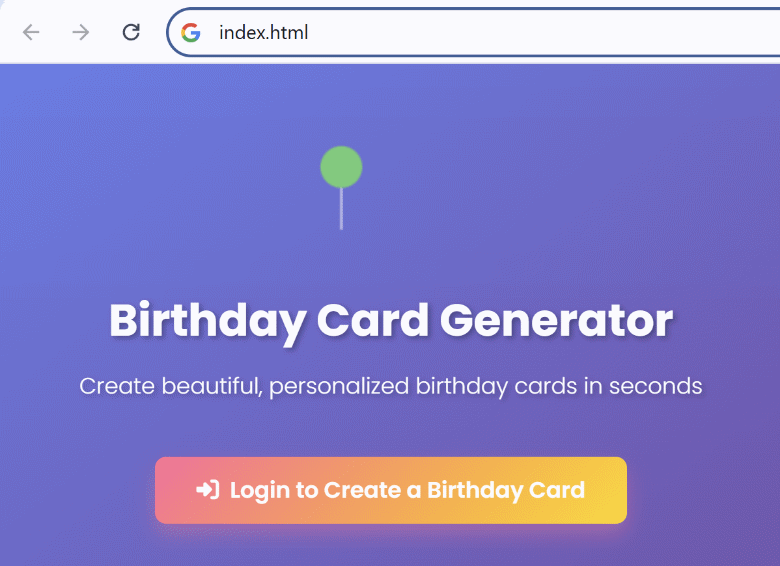
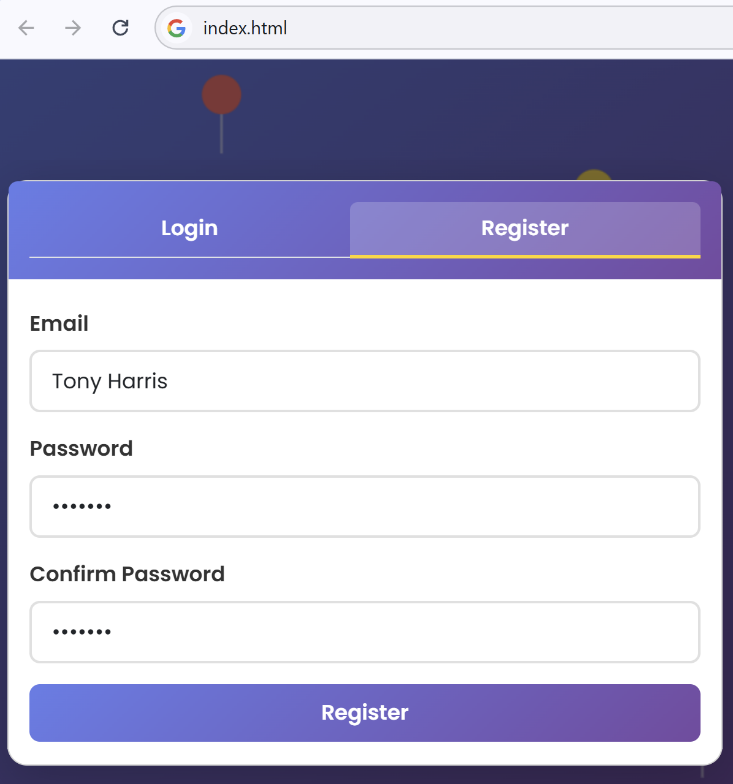
#### Fix Bugs

**Open Supabase** and check if there **are tables created properly**. In case there aren’t, ask Copilot to fix this or help you fix the issue by yourself.

Open the App in the Browser and see if there are any errors in [Dev Tools]. Ask the chat to fix them.

#### The App in the Browser

**Open the app** in the Web browser and **test register** / **login** / **logout**. After registering a new user check if the user is registered in the **Supabase Auth** system.

### Implement Card Generation

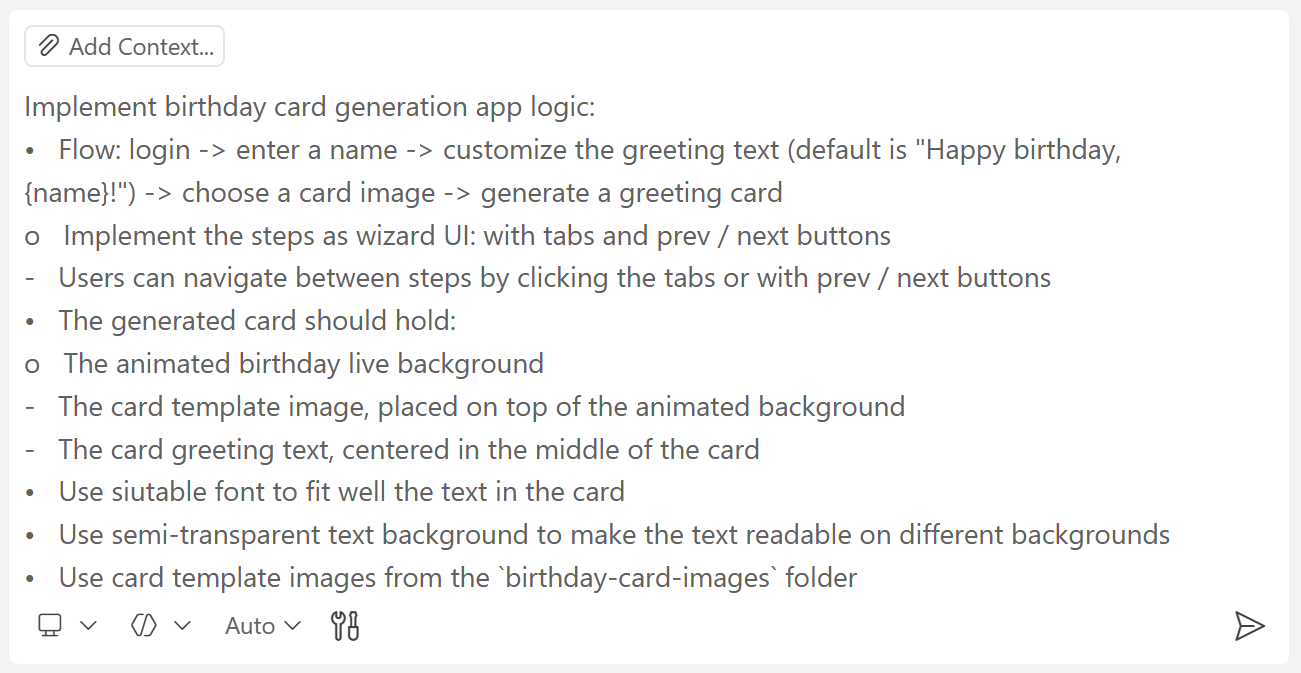
Implement **birthday card generation app logic**:

* **Flow**: **login** -> enter a **name** -> customize the **greeting text** (default is "Happy birthday, {name}!") -> choose a **card image** -> **generate** a greeting card
  + Implement the steps as **wizard UI**: with tabs and [Prev] / [Next] buttons
  + Users can navigate between steps by clicking the **tabs** or with the [Prev] / [Next] buttons
* The **generated card** should hold:
  + The animated birthday **live background**
  + The card template **image**, placed on top of the animated background
  + The card **greeting text**, centered in the middle of the card
* Card formatting:
  + Use suitable **font** to fit well the text in middle of the card
  + Use semi-transparent **text background** to make the text readable on different backgrounds
* Use card **template images** from the `birthday-card-images` folder

#### Provide Images

Copy the images provided in the Resources folder for this exercise or download your own from the Internet. Put the folder with the images in the app workspace in VS Code. Name the folder properly, for example `birthday-card-images`.

#### The AI Prompt

****

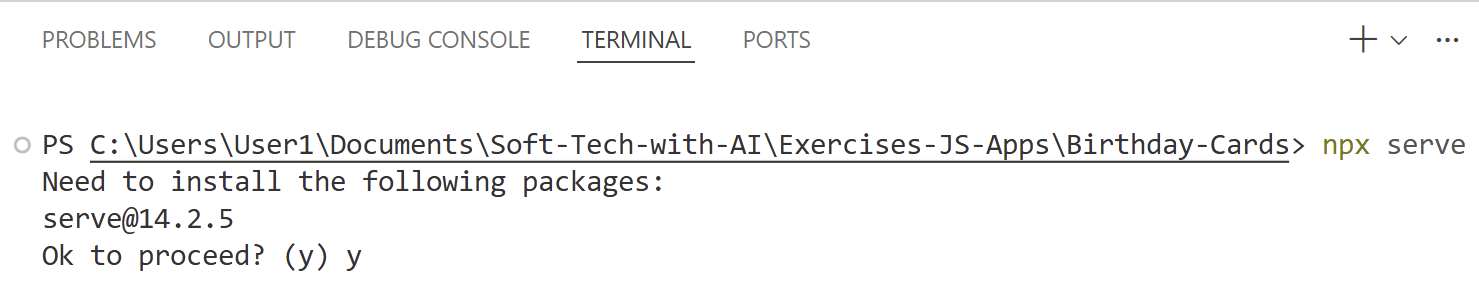
#### Note: Run a Web Server

You might need to **run the app through a Web server**. Otherwise, app images might be unavailable, because dynamic image loading with JavaScript does not work from local HTML files.

The recommended approach is to **run this command** in the **Terminal** windows in VS Code:

|  |
| --- |
| npx serve |

Run the above command in the **app code folder**, where the file index.html is located. When the command is run for the first time, it will install some **Node.js packages**.



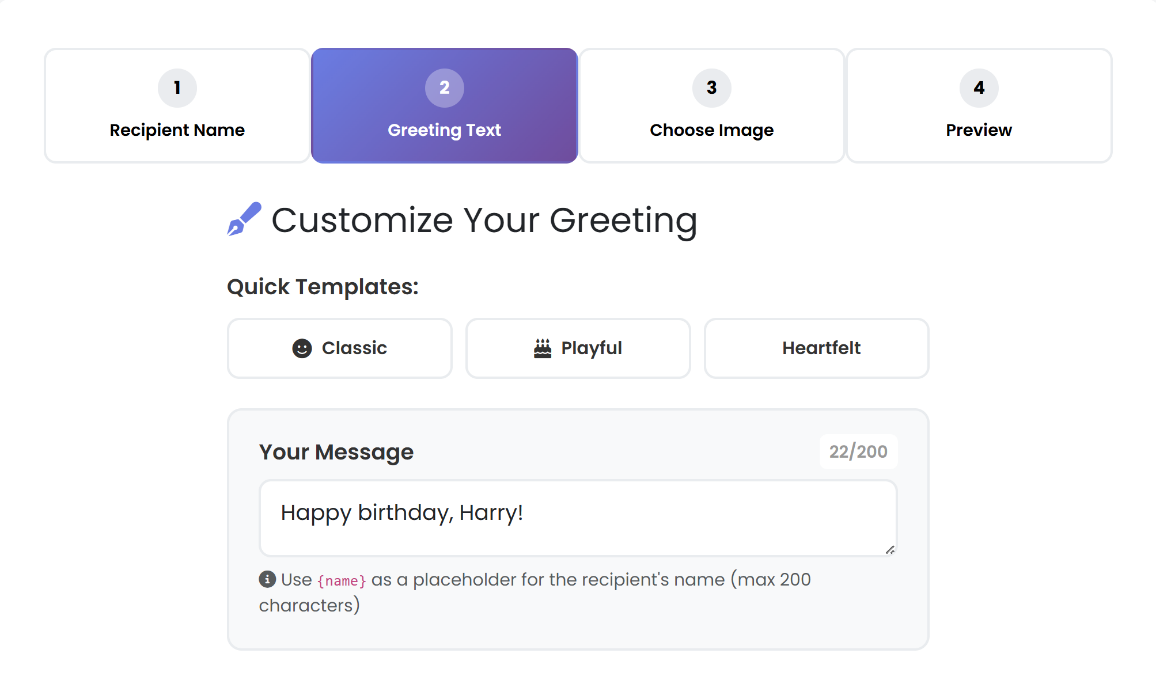
After that, the **local Web server** will start, typically at port 3000. Open the app through the local URL, visible from the Terminal: [http://localhost:3000](http://localhost:3000/)



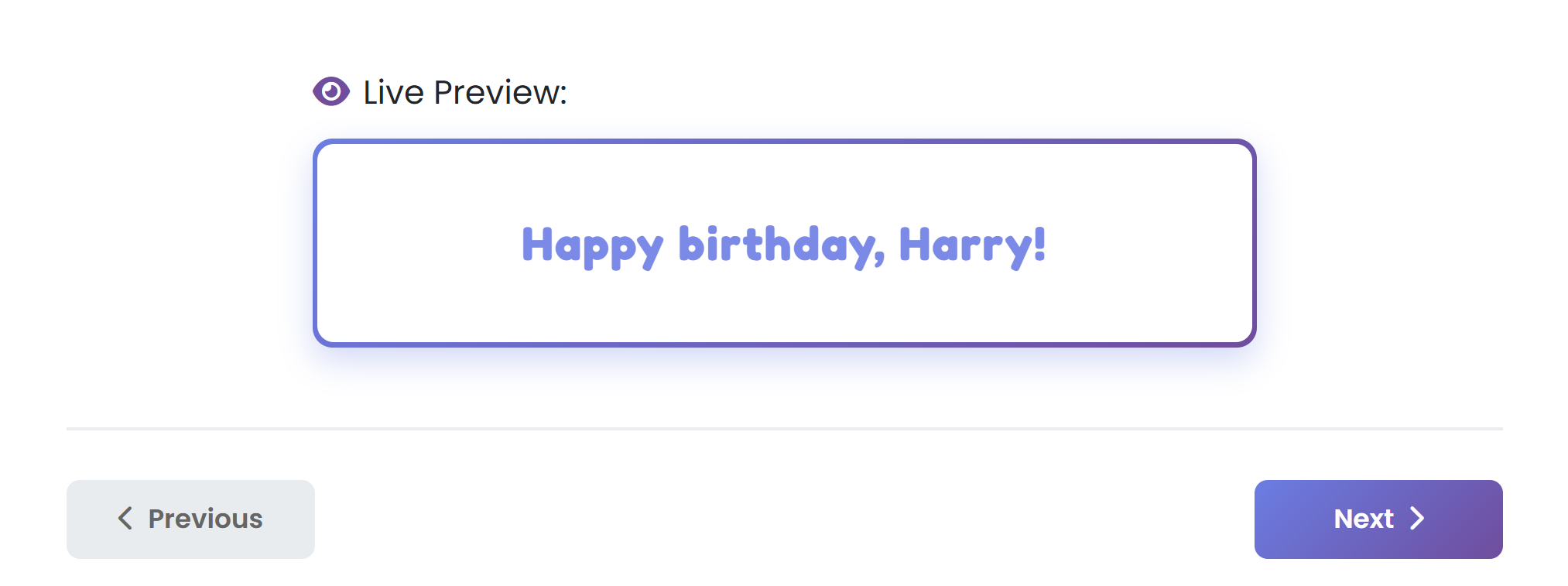
#### The App in the Browser

Below are **sample screenshots** of the card generation UI.

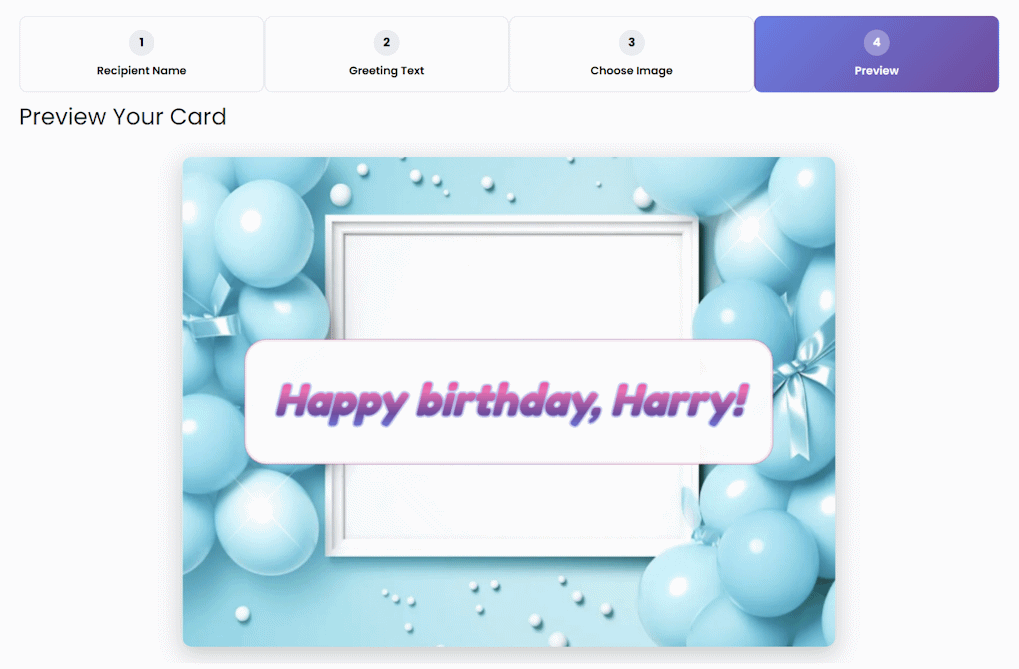
##### The Card Customization Tabs

****

##### The Navigation Buttons

****

##### The Card Preview Tab

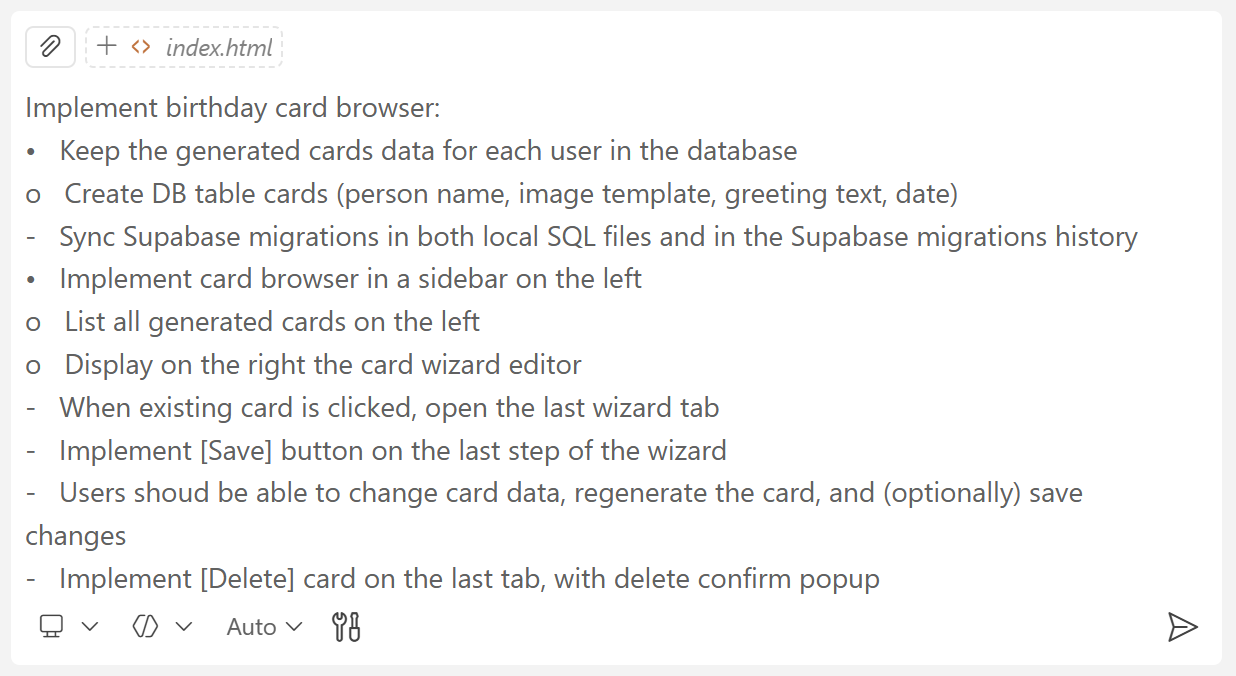


### Implement Card Browser

Implement **birthday card browser**:

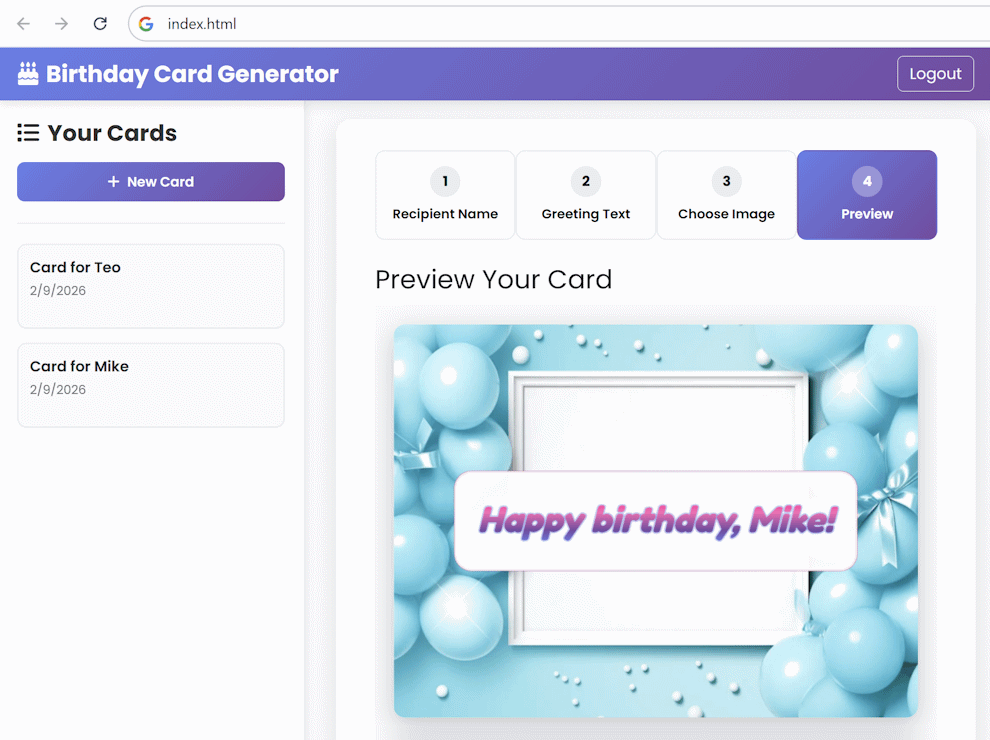
* Keep the **generated cards data** for each user in the **database**
  + Create DB table: **cards (person name, image template, greeting text, date)**
  + Sync **Supabase migrations** in both local SQL files and in the Supabase migrations history
* Implement **card browser** in a sidebar on the left
  + List all generated cards on the left
  + Display on the right the card wizard editor
  + When **existing card** is clicked, load the card and open the last wizard tab
  + Implement **[Save] button** on the last step of the wizard
  + Users should be able to **change card data**, **regenerate the card**, and (optionally) **save changes**
  + Implement [Delete] card on the last tab, with delete confirm popup

#### The AI Prompt



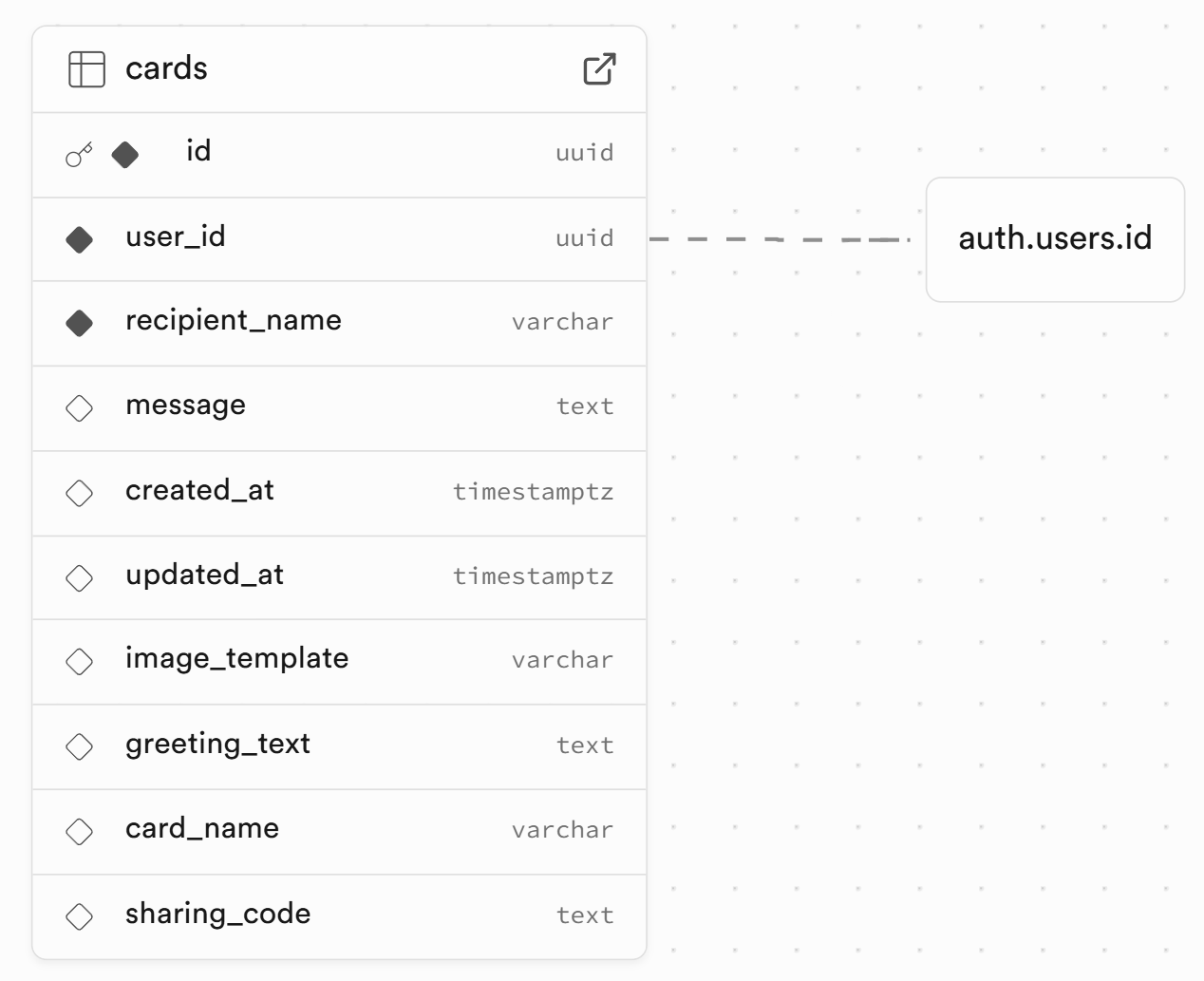
#### The App in the Browser

This is a sample of how the cards browser may look like



#### The Tables in Supabase

This is an example of how the **DB tables** may look like:



## Birthday Cards Sharing

In this exercise we shall **implement** "**card sharing**" for the card generator app.

We shall use "**sharing codes**" to **identify shared cards**.

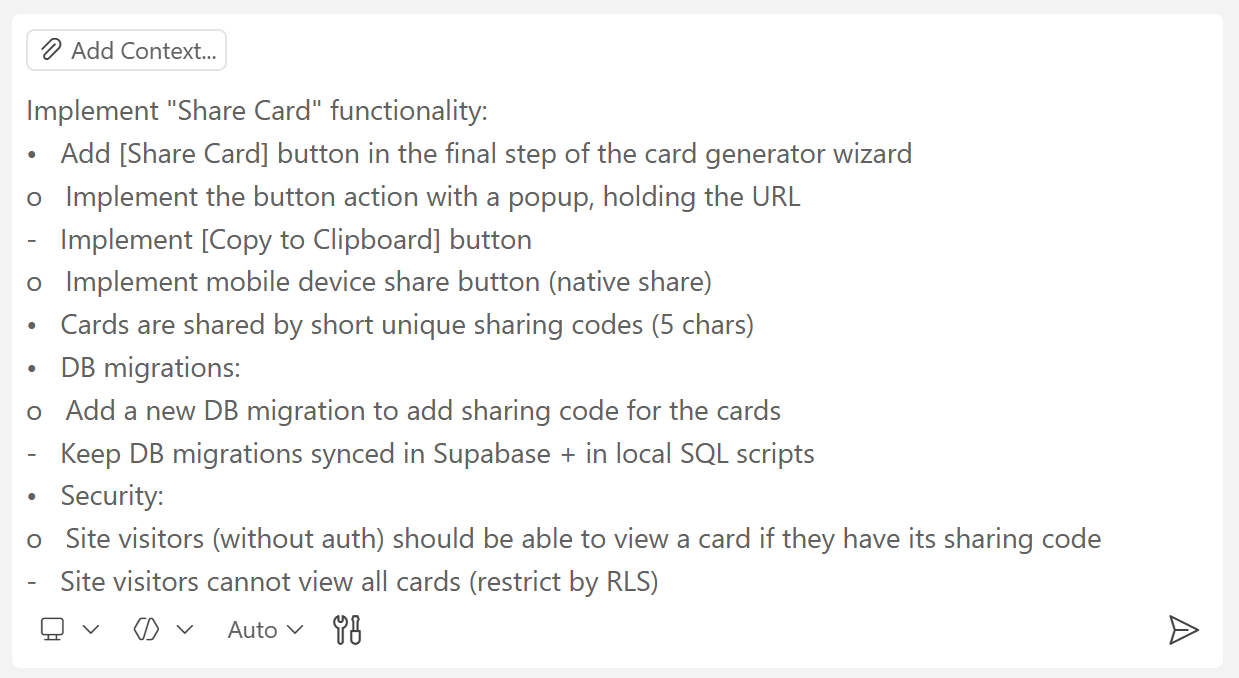
**Note**: Use a **frontier LLM** to reduce chance of wrong implementation in security-related functionality.

### Implement Card Sharing

Implement "**Share Card**" functionality:

* Add **[Share Card] button** in the final step of the card generator wizard
  + Implement the button action with a **popup**, holding the URL
  + Implement **[Copy to Clipboard] button**
  + Implement mobile device **share button** (native share)
* Cards are shared by short **unique sharing codes** (5 chars)
* **DB migration**:
  + Add a new **DB migration** to add sharing code for the cards
  + Keep DB migrations synced in Supabase + in local SQL scripts
* **Security**:
  + Site **visitors** (without auth) should be able to **view a shared card** if they have its sharing code
  + Site visitors cannot view all cards (restrict by RLS)
  + **Users can see only their own cards** after login (restrict by RLS)
  + Create a **stored function** to view card by sharing code (with public access by RLS)
* **UI**: implement a separate page `view-card.html` to view shared cards

#### The AI Prompt



#### Note

If you run your app with `npx serve`, your sharing links should be like this:

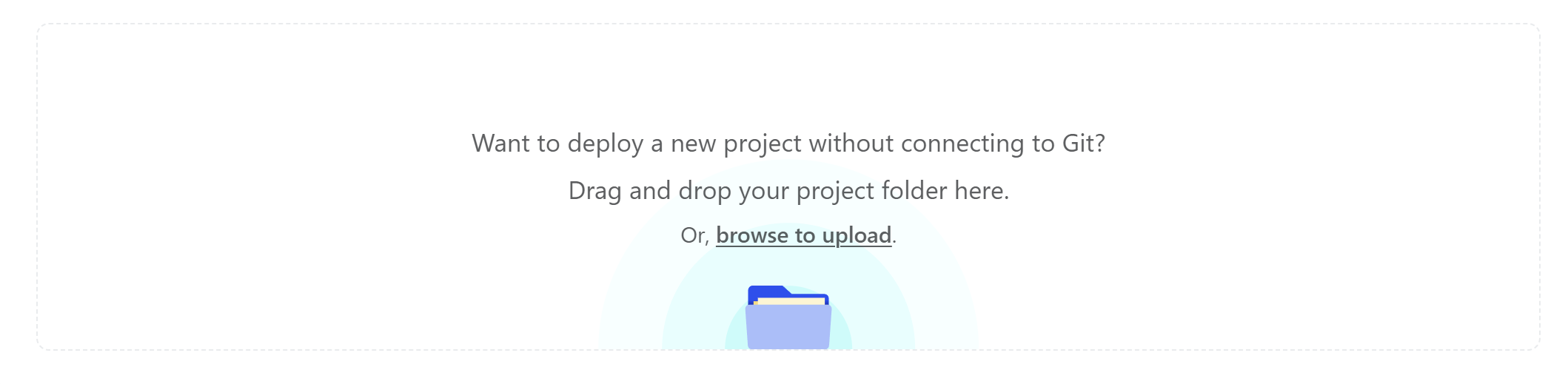
* <http://localhost:3000/view-card?code=B2CJH>

URLs like this will be redirected without the sharing code and **will not work**:

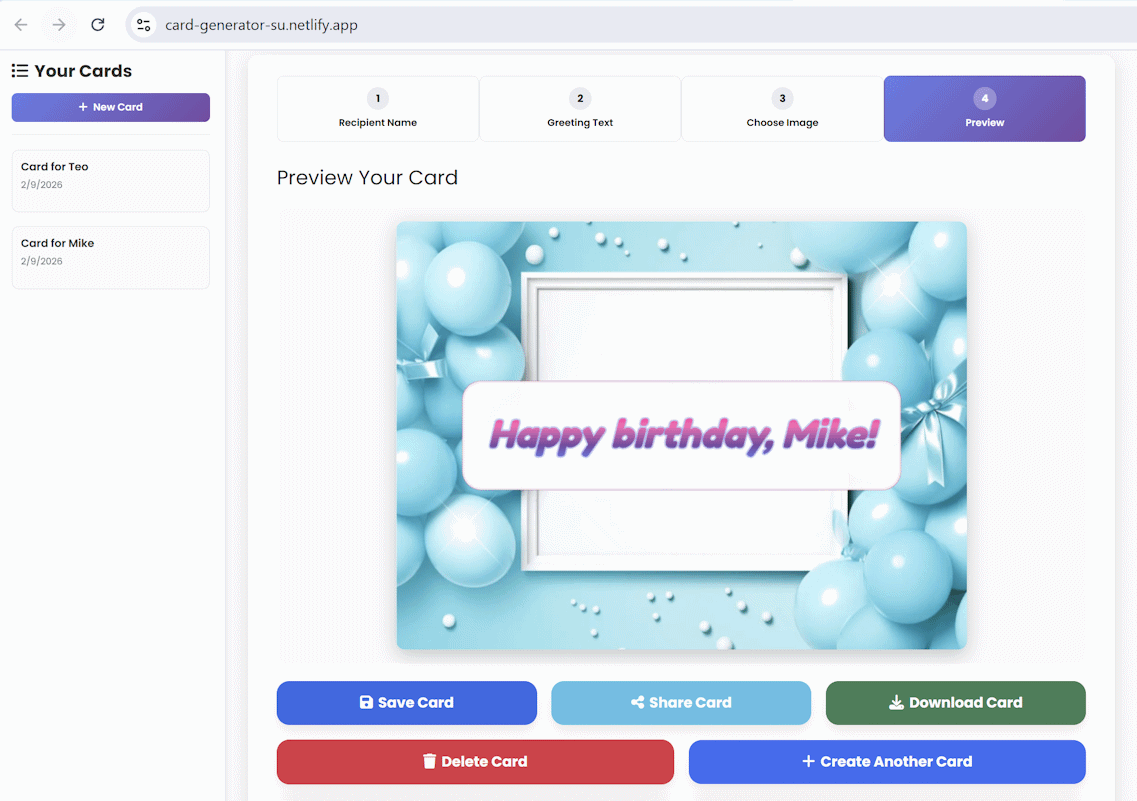
* <http://localhost:3000/view-card.html?code=B2CJH>

#### Deploy to Netlify

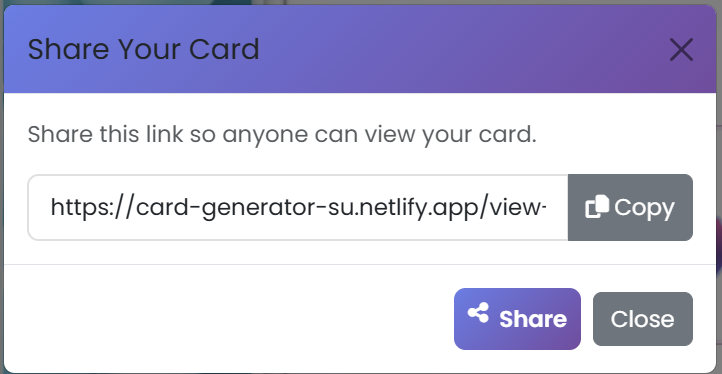
Deploy the app to **Netlify** to test it in a real-world environment. Use **the [Deploy Manually] option**:



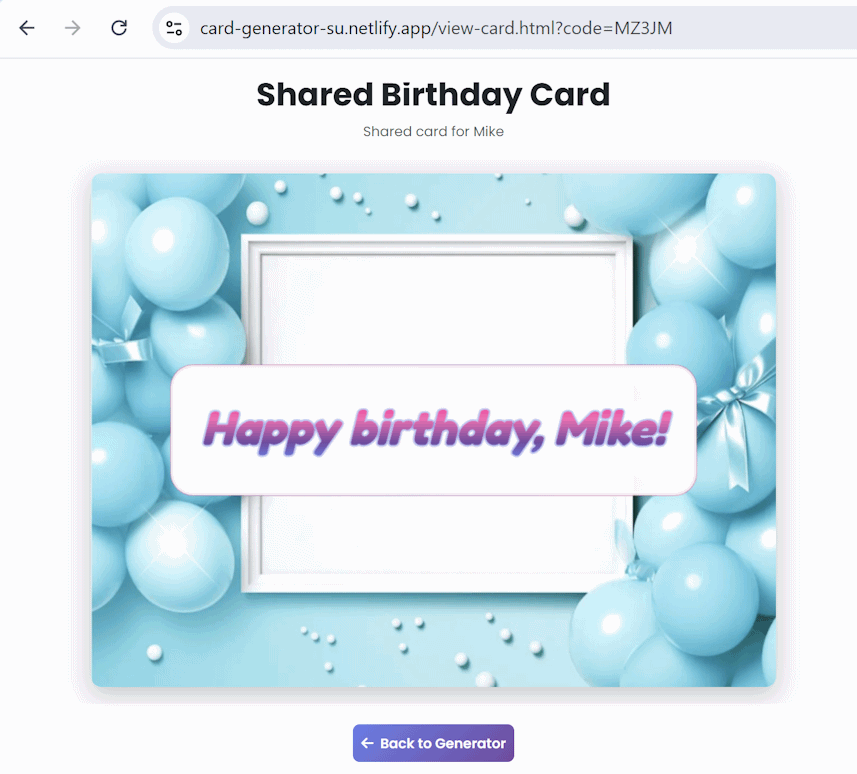
#### The App in the Browser



#### The Sharing Pop-Up Window



#### The Shared Card in the Browser



## Birthday Cards Admin

In this exercise we shall **implement** "admin" role and **Admin Panel** for the card generator app.

### User Roles in Supabase

Implement **user roles** in Supabase:

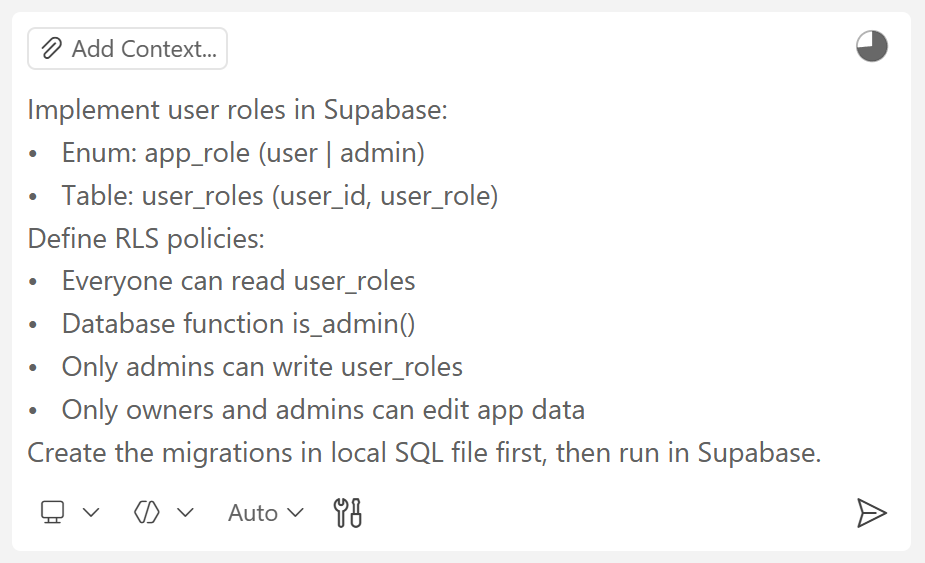
* Enum: app\_role (user | admin)
* Table: user\_roles (user\_id, user\_role)

Define **RLS policies**:

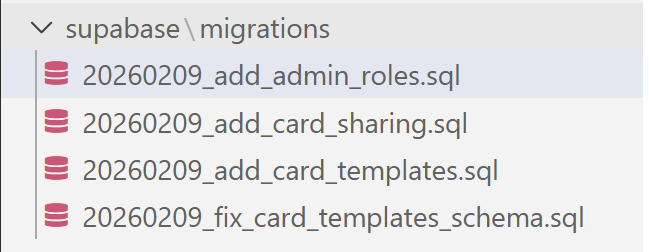
* Everyone can **read** user\_roles
* Database function is\_admin()
* Only admins can **write** user\_roles
* Only owners and admins can edit app data

Create the **migrations** in a local SQL file first, then run in Supabase.

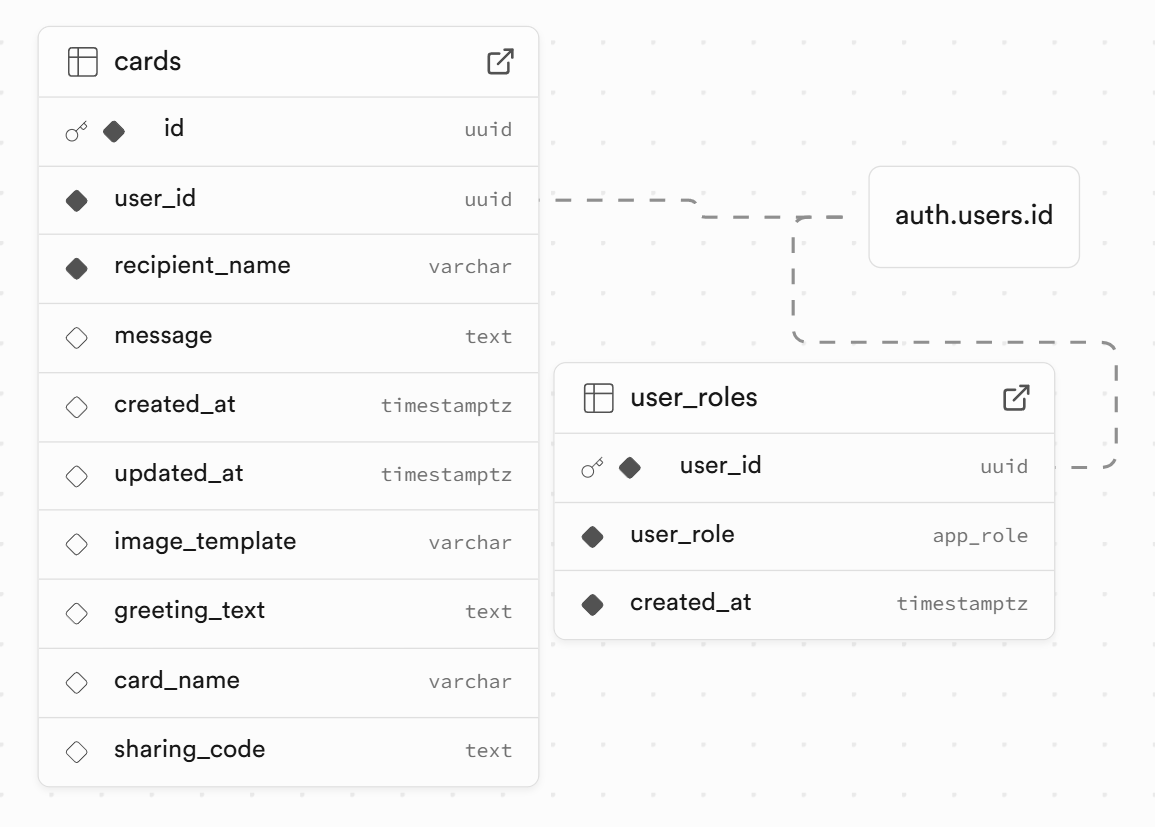
#### The AI Prompt



#### The Local Migration Files



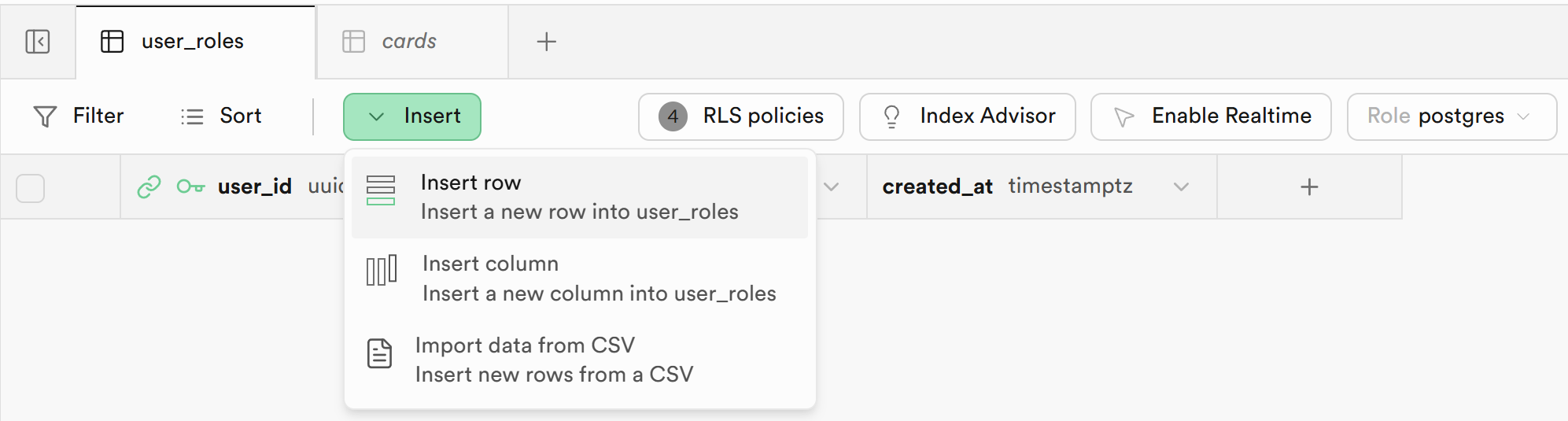
#### The Supabase Tables



### Create an Admin

Create the **first admin** in the database: just add a new row in the user\_roles table.

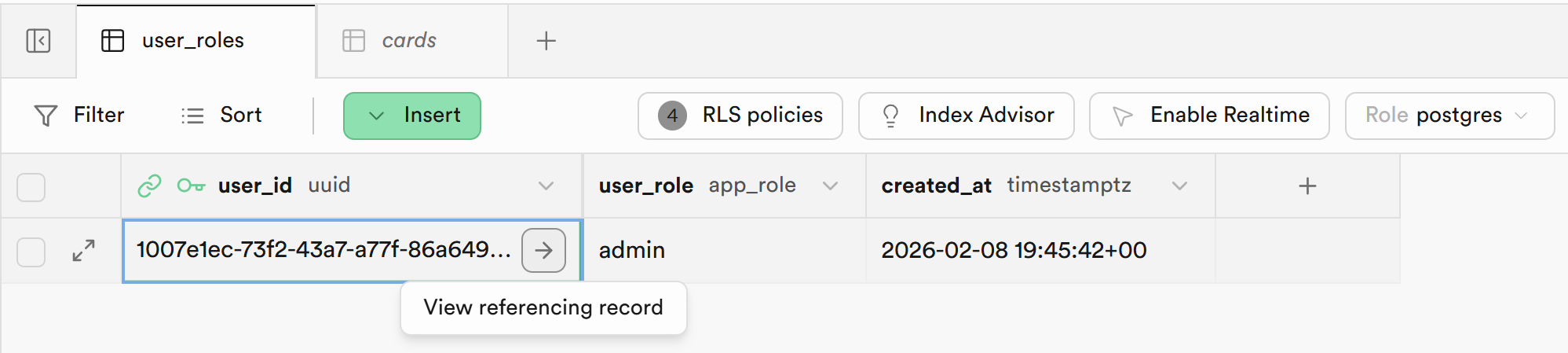
Open the **[Table Editor]** and then select the user\_roles table. Then click the **[Insert] button** and then **[Insert row]**:



A new tab will open. Choose the **user** to give **admin role** to from the **[Select Record] button**:



You will see the new role appear after you save the changes:

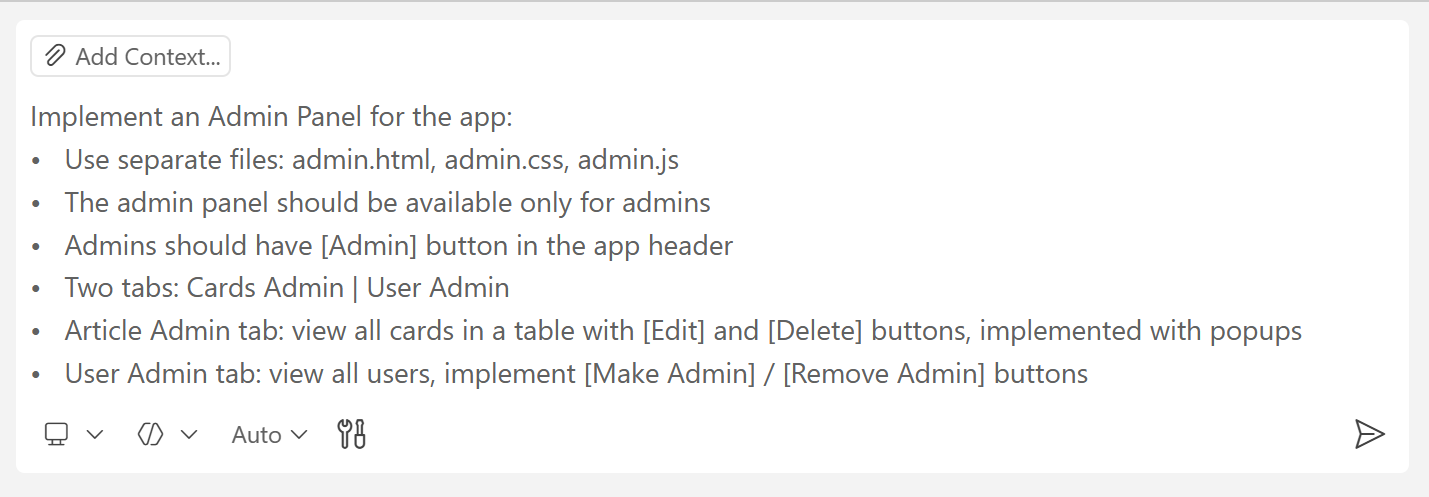


### Implement Admin Panel

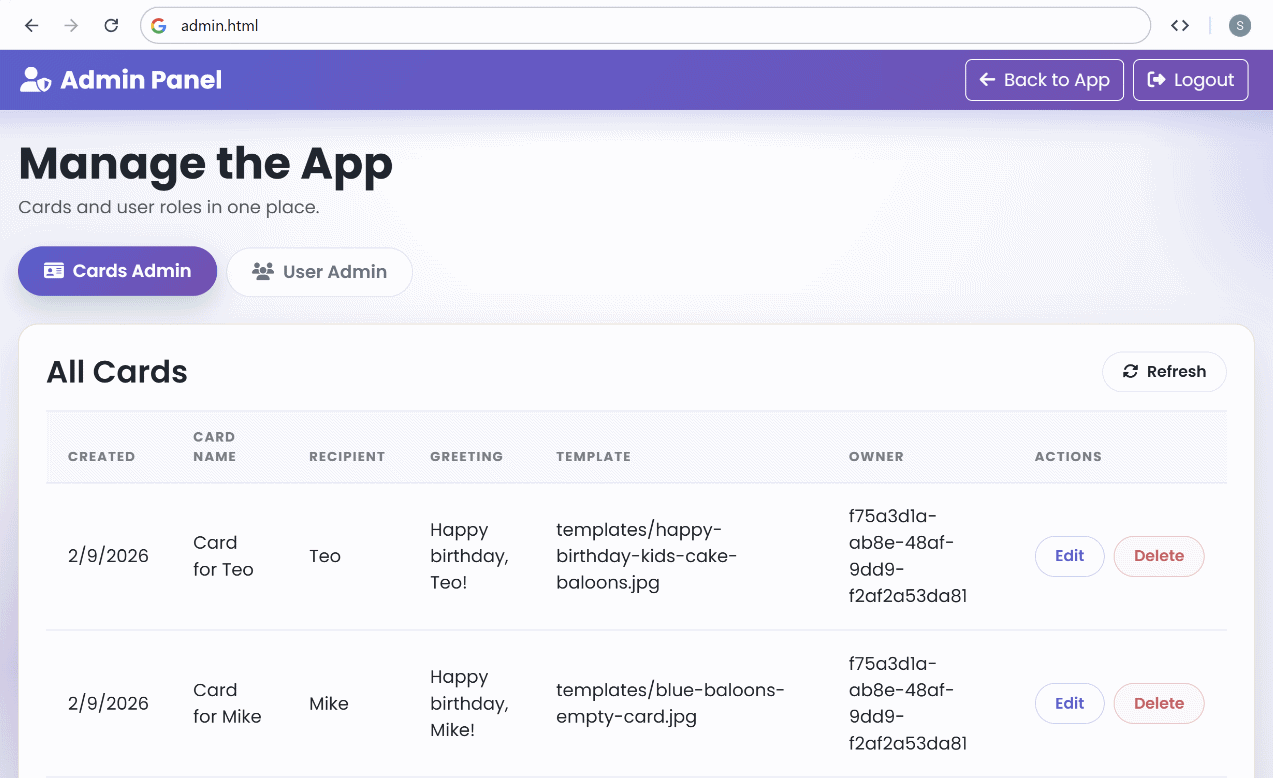
**Implement an Admin Panel** for the app:

* Use separate files: admin.html, admin.css, admin.js
* The **admin panel** should be available only for admins
* Admins should have [Admin] button in the app header
* Two tabs: **Cards Admin** | **User Admin**
* **Cards Admin** tab: view all cards in a table with [**Edit**] and [**Delete**] buttons, implemented with popups
* **User Admin** tab: view all users, implement [**Make Admin**] / [**Remove Admin**] buttons

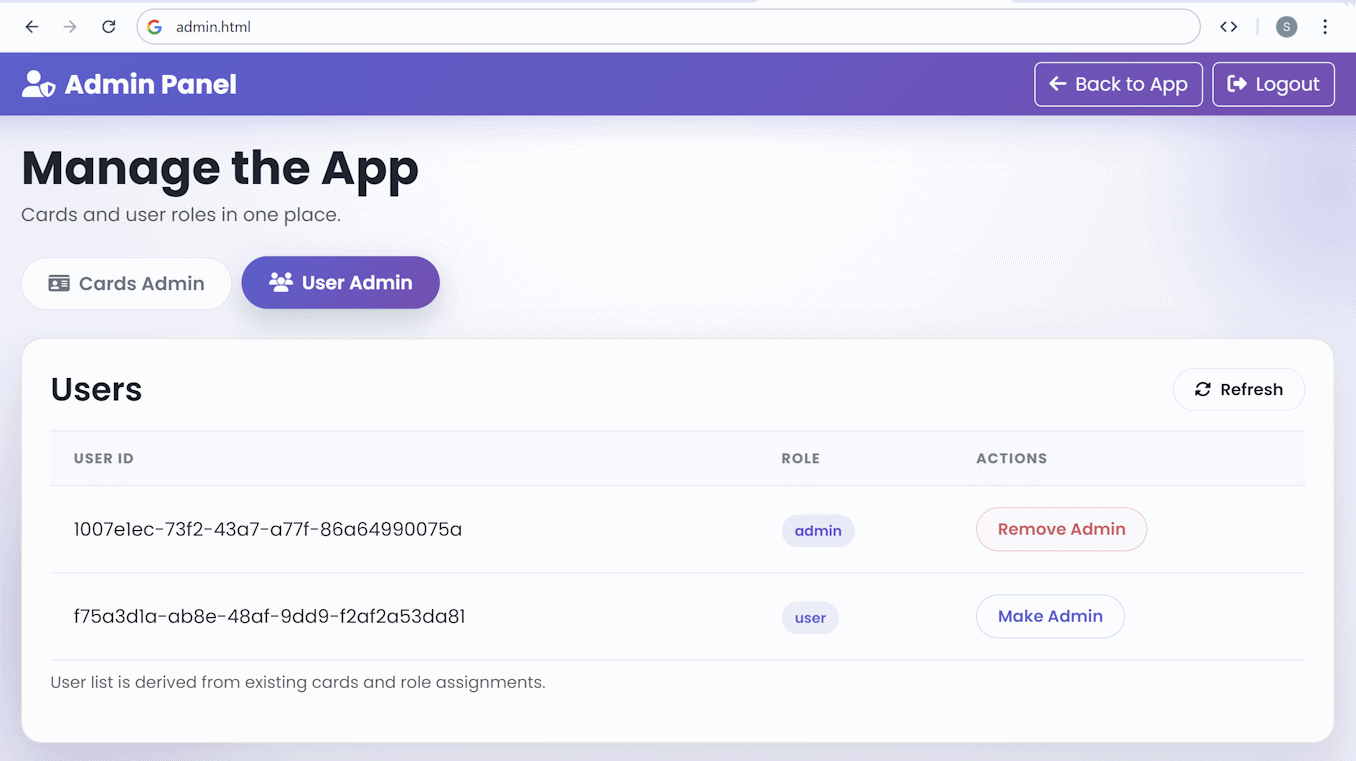
#### The AI Prompt



#### The Cards Admin Tab



#### The User Admin Tab



## Birthday Cards Templates

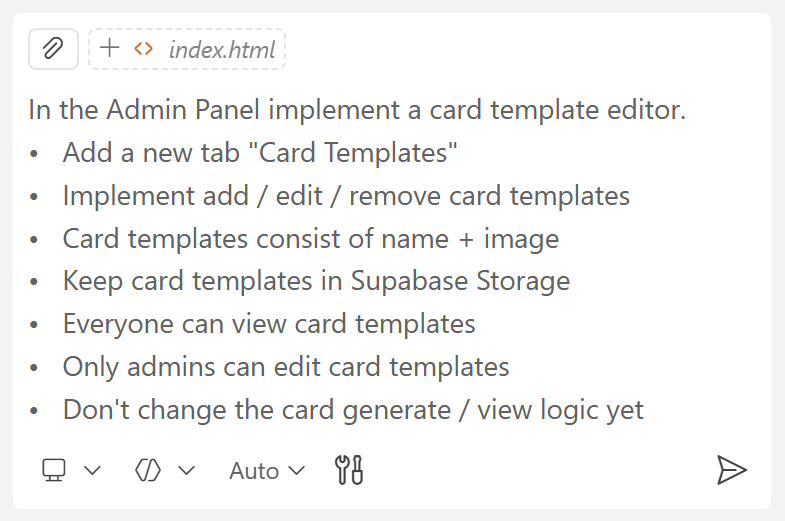
In this exercise we shall **implement an editor for card template images**. We shall **store card template images** in **Supabase Storage**.

### Card Template Editor

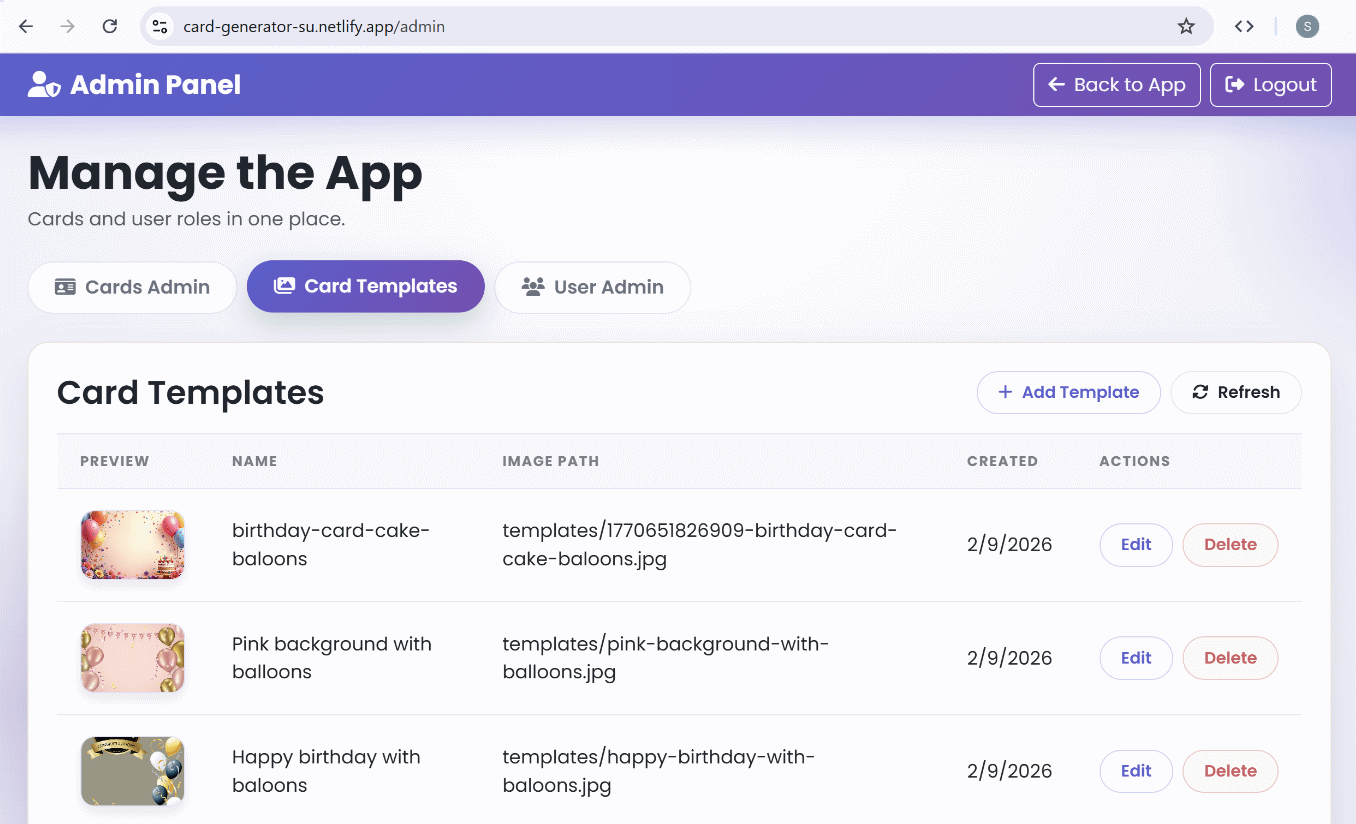
In the **Admin Panel** implement a **card template editor**.

* Add a new **tab "Card Templates"**
* Implement **add** / **edit** / **remove** card templates
* Card templates consist of **name** + **image**
* Keep card templates in **Supabase Storage**
* Everyone can view card templates
* Only admins can edit card templates
* Don't change the card generating / card view logic yet

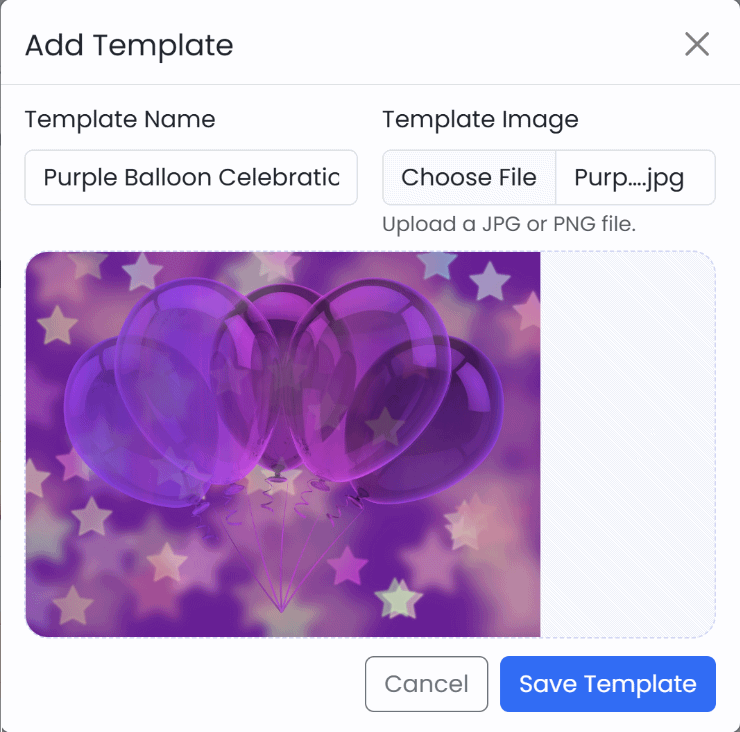
#### The AI Prompt



#### The Templates in the App



#### Adding Templates

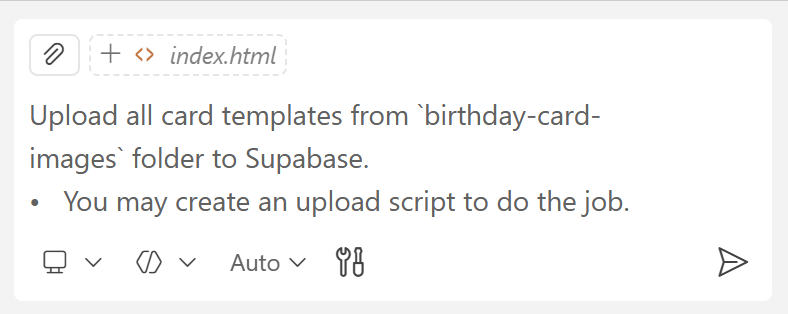


### Upload Existing Templates

Upload all card templates from `birthday-card-images` folder to Supabase.

* You may create a temporary **upload script** to do the job.

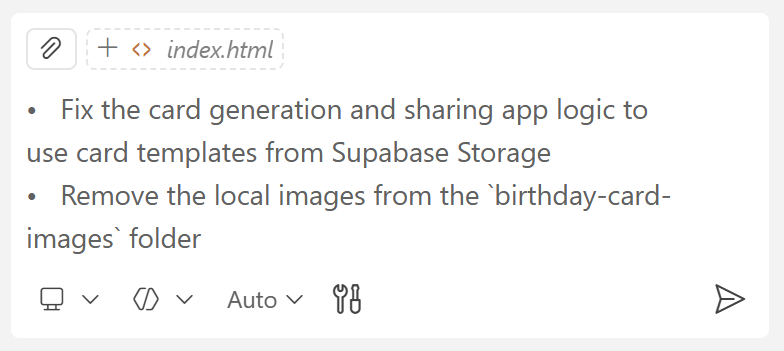
#### The AI Prompt



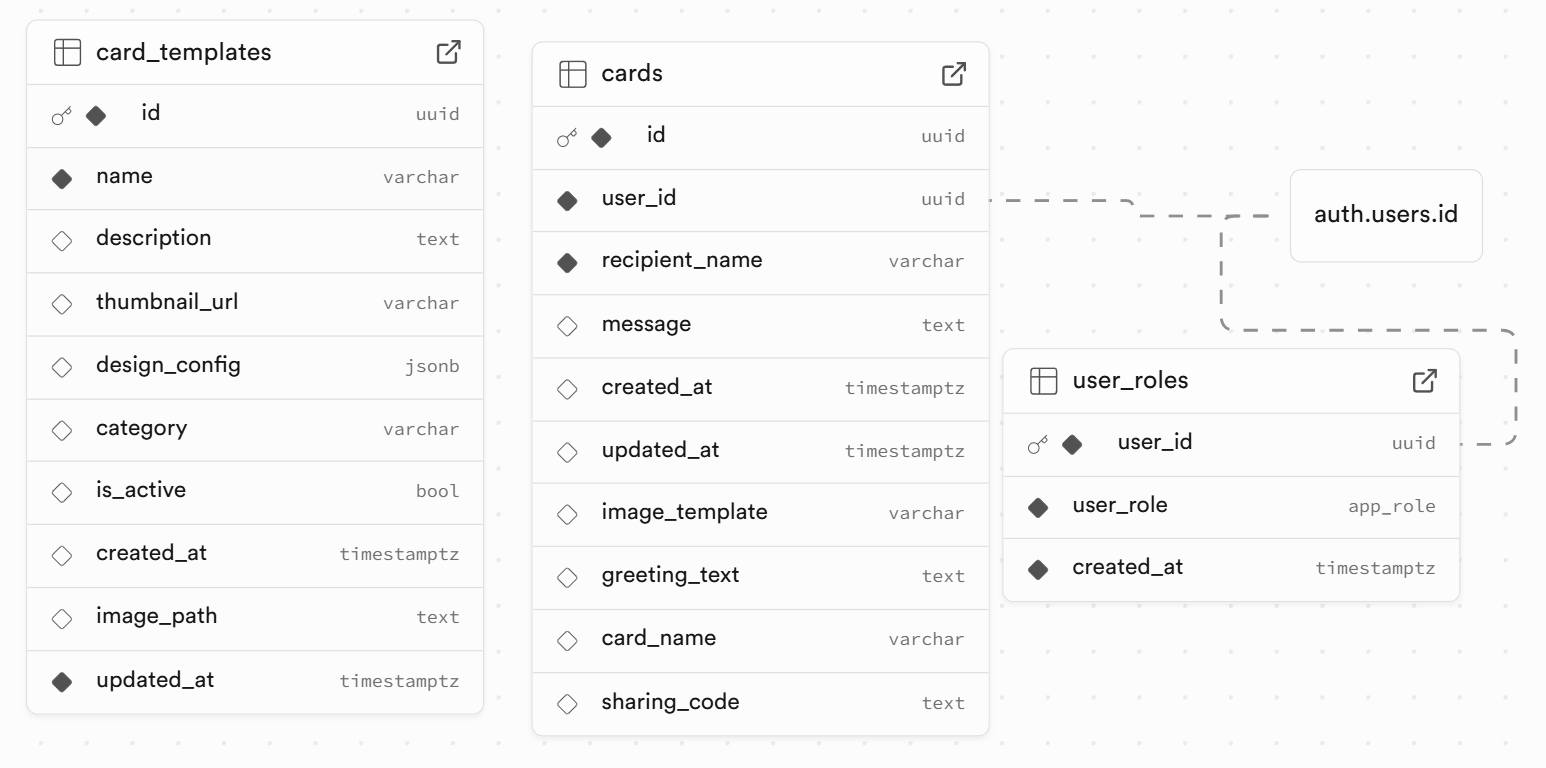
### Use Card Templates from Storage

* Fix the card generation and sharing app logic to **use card templates** from Supabase Storage
* Remove the local images from the `birthday-card-images` folder

#### The AI Prompt



#### The Tables in Supabase



#### The Storage in Supabase

