JS Apps with Supabase

Database Apps with Users and Roles, Supabase Auth, Supabase Storage

Svetlin Nakov, PhD

Co-founder @ SoftUni

SoftUni AI https://ai.softuni.bg

Agenda

SoftUni AI

1. **Database apps with users**: auth, users, register, login, logout
2. **Database migrations**: concepts, SQL migration scripts, Supabase migrations infrastructure
3. **Locations on the Map**: example of JS app with database, auth, users, RLS policies, DB schema migrations
4. **Row-Level Security (RLS)** in Supabase
5. **Users and roles** with Supabase Auth and RLS
6. **Edge functions**: server-side JS logic in the backend
7. **Supabase storage**: file uploads and downloads

Sli.do Code

#Soft-Tech-AI

Join at

slido.com

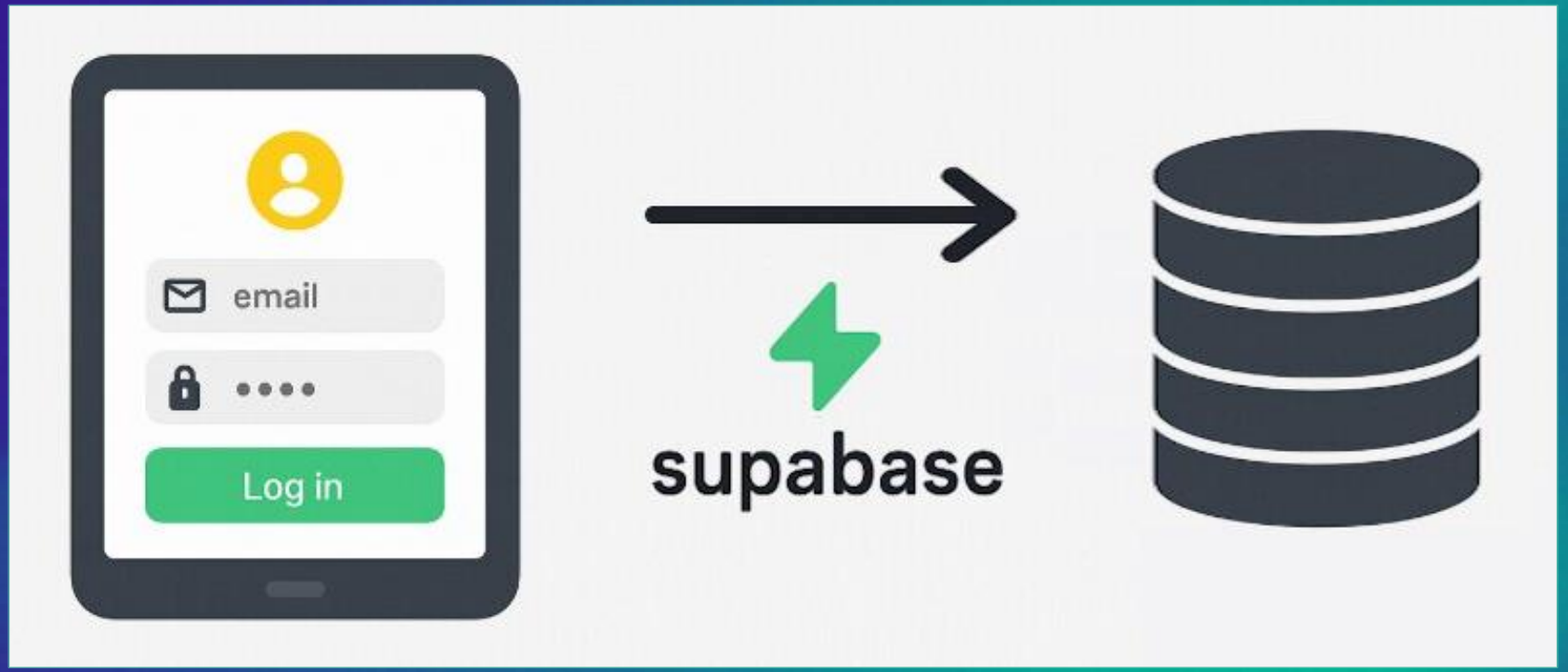
#Soft-Tech-AI

Breaks

20:00 / 21:00

Database Apps with Users

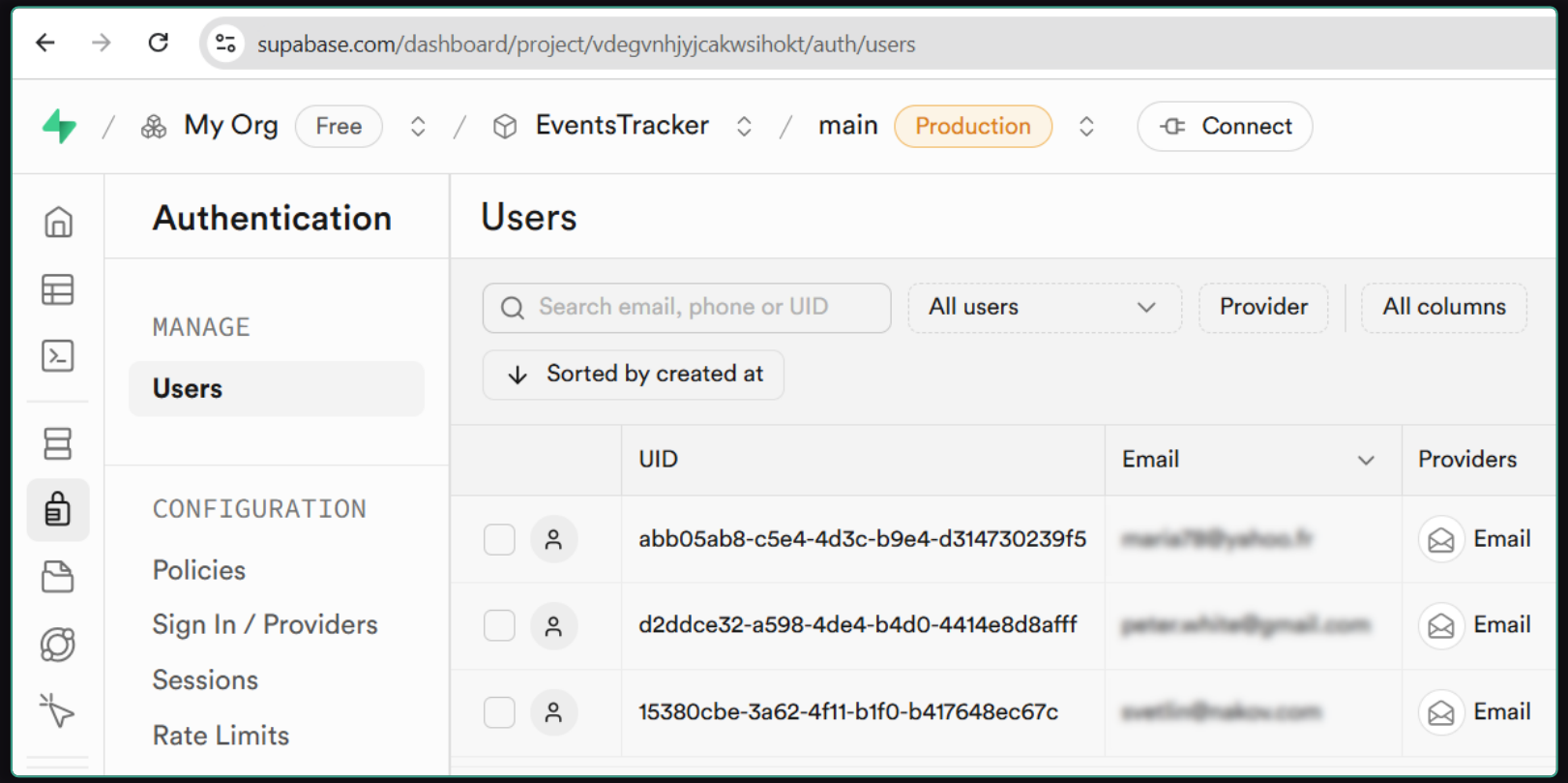
Auth, Users, Register / Login / Logout

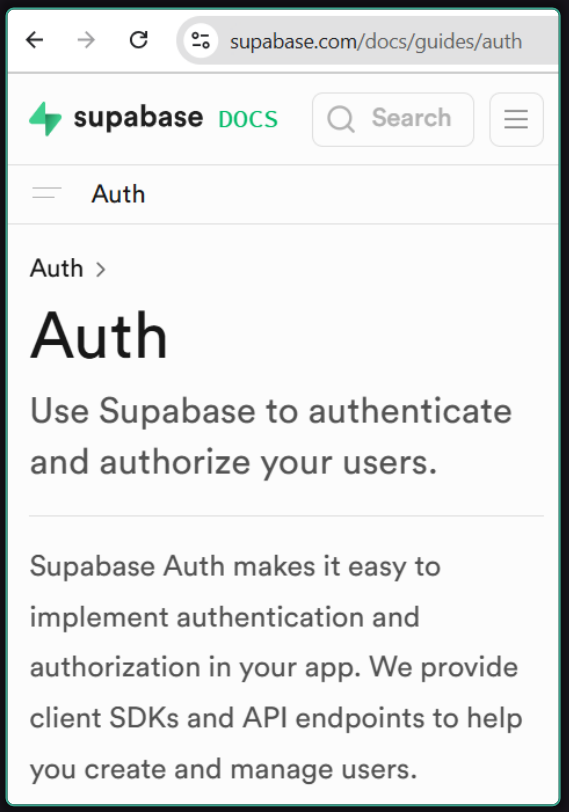


Authentication in Supabase

SoftUni AI

- **Supabase** has **built-in authentication** back-end
  - Users, roles, external login providers, multi-factor auth, …

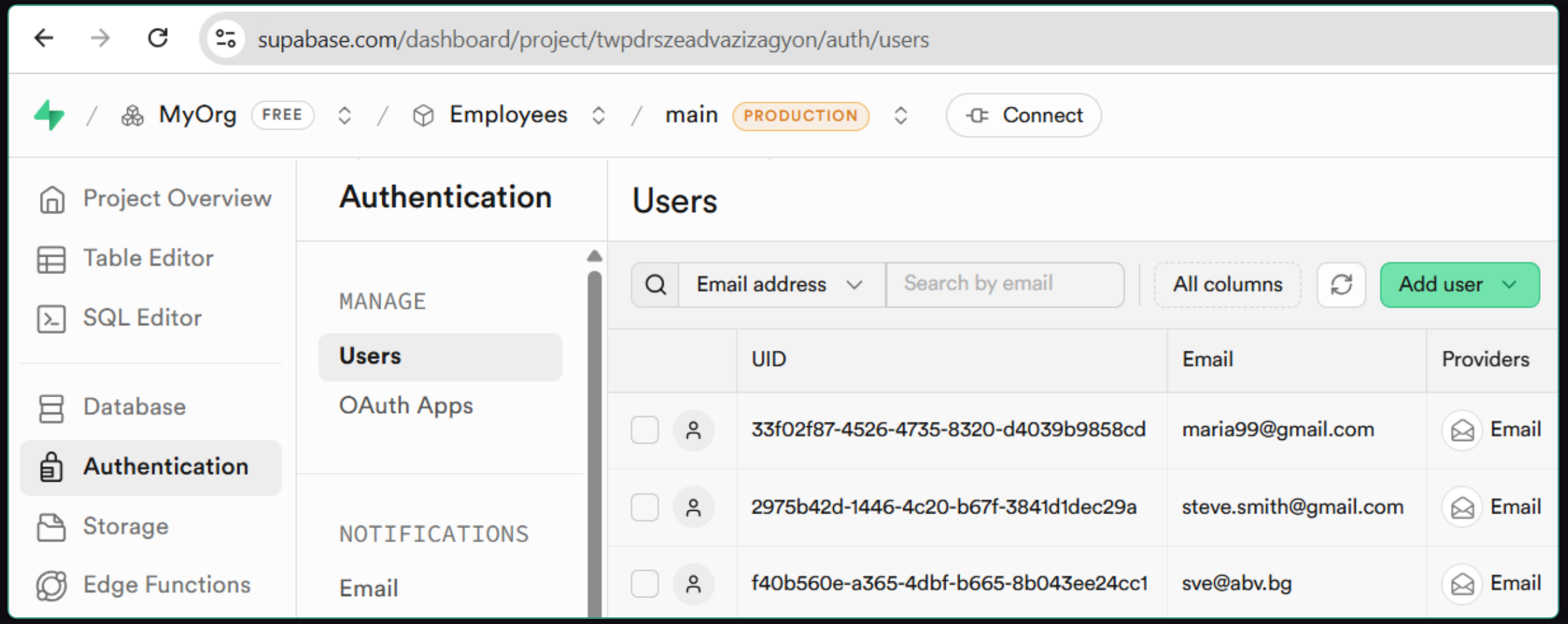




Supabase Auth → Users

SoftUni AI

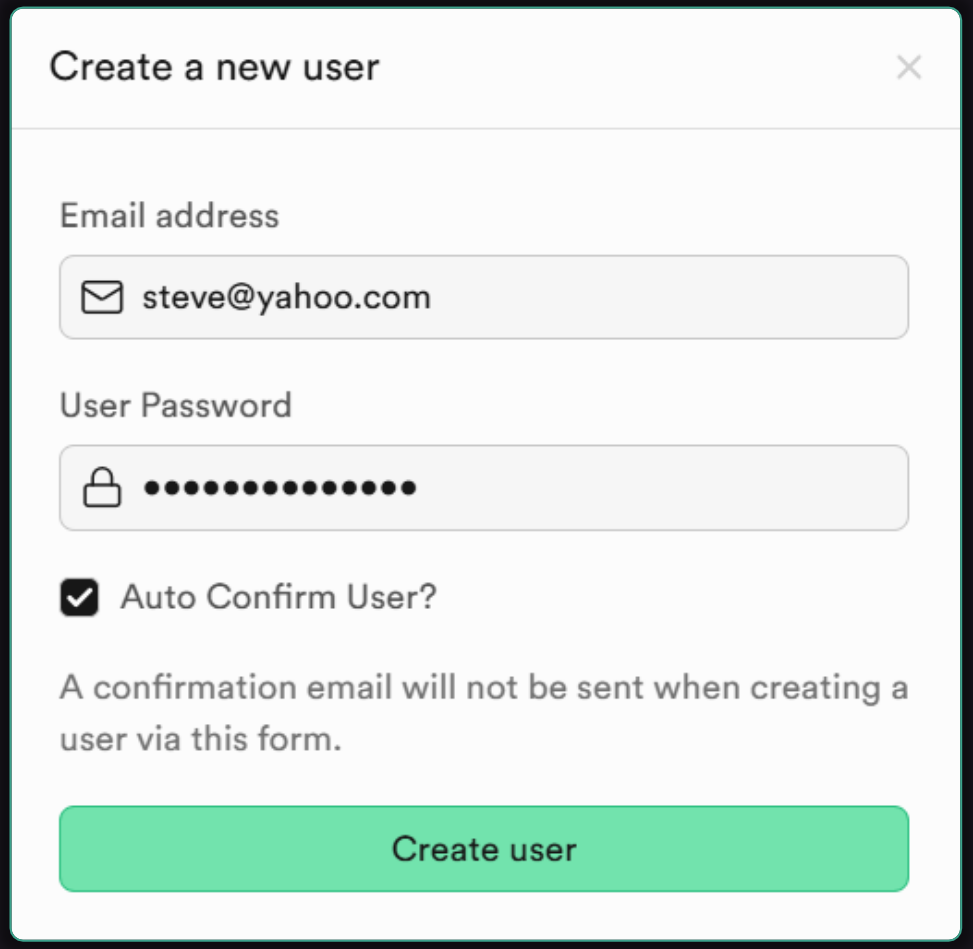
- Dashboard → Authentication → Users



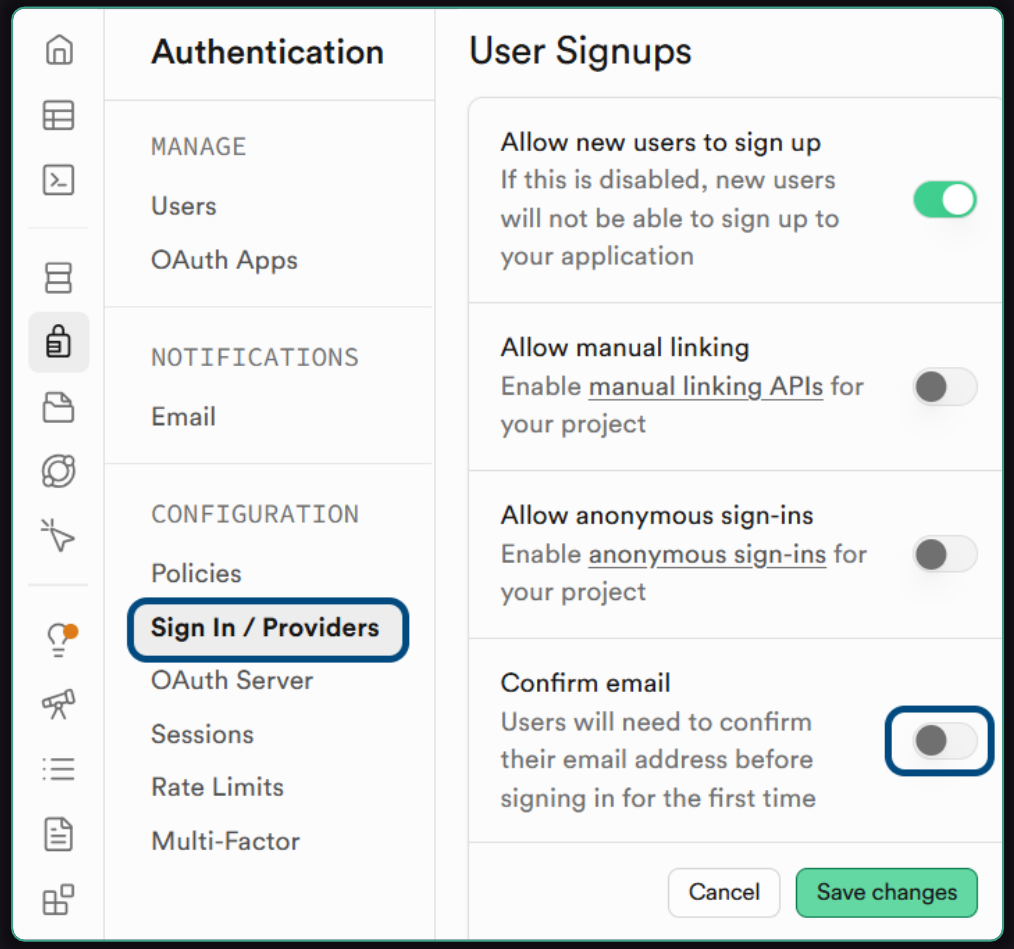
Supabase Auth Configuration

SoftUni AI

- Create a **new user**



- Switch off "**Confirm email**"



Where are Users Stored in DB?

SoftUni AI

- Users, managed by Supabase Auth, are stored in the `**auth.users**` DB table:

auth.users

Filter | Sort | Index Advisor | Enable Realtime

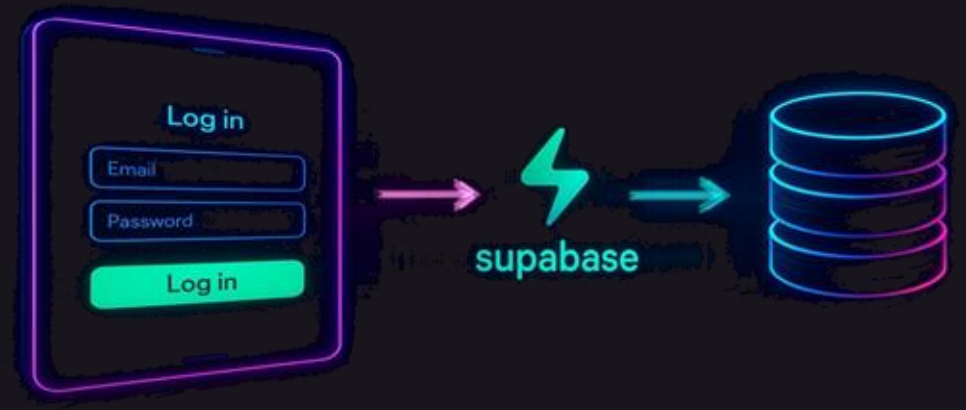
id uuid	email varchar	encrypted_password varchar	email_confirmed_at timestamptz
2975b42d-1446-4c20-b67f-3841d1dec29a	steve.smith@gmail.com	$2a$10$NQxWNxpOUsFd9qhJMUl8N	2026-02-02 10:23:42.507229+00
33f02f87-4526-4735-8320-d4039b9858cd	maria99@gmail.com	$2a$10$i5yWCpkn86ce4TyppXZvou0j	2026-02-02 10:22:43.734484+00
f40b560e-a365-4dbf-b665-8b043ee24cc1	sve@abv.bg	$2a$10$cfFAPtAknLMK59186rvkZ.hl8	2026-02-02 10:16:32.979504+00

- **Passwords** are encrypted with **bcrypt**
  - Your app can compare a password with the DB, but **cannot decrypt** a password back to plaintext

Adding Supabase Authentication

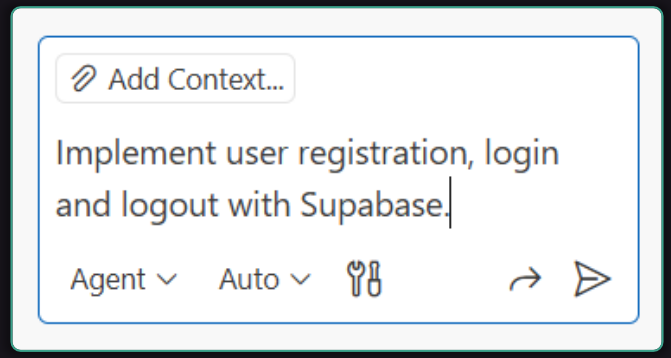
SoftUni AI

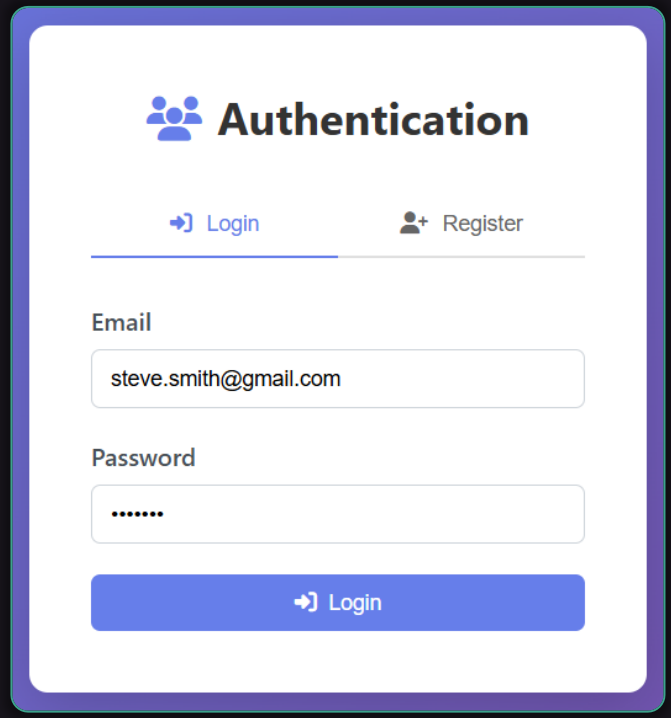
- Implementing **Supabase authentication**
  - User registration, login, logout



- Just run an AI prompt:

> Implement **user registration**, **login** and **logout** with Supabase.





Contactbook App with Users

SoftUni AI

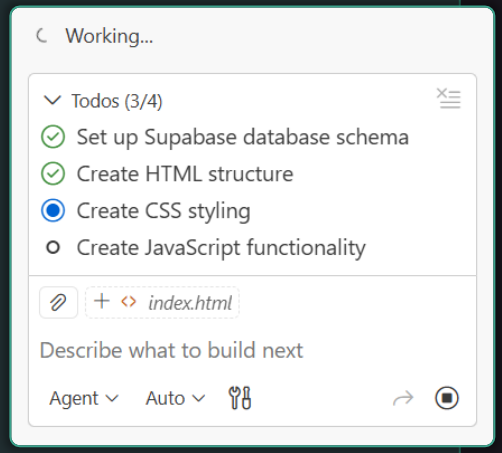
- We want to create a "**Contactbook**" app
  - Users **register** / **login** and manage their own contacts in their accounts in the app
  - **Contacts** hold **name** + **phone** + **email** + **comments**
- **Step 1**: Create a **Supabase project**
  - Supabase Dashboard → Projects → New project
- **Step 2**: Connect **Supabase MCP** to VS Code
  - Supabase Dashboard → Connect → MCP → VS Code

Contactbook App with Users (2)

SoftUni AI

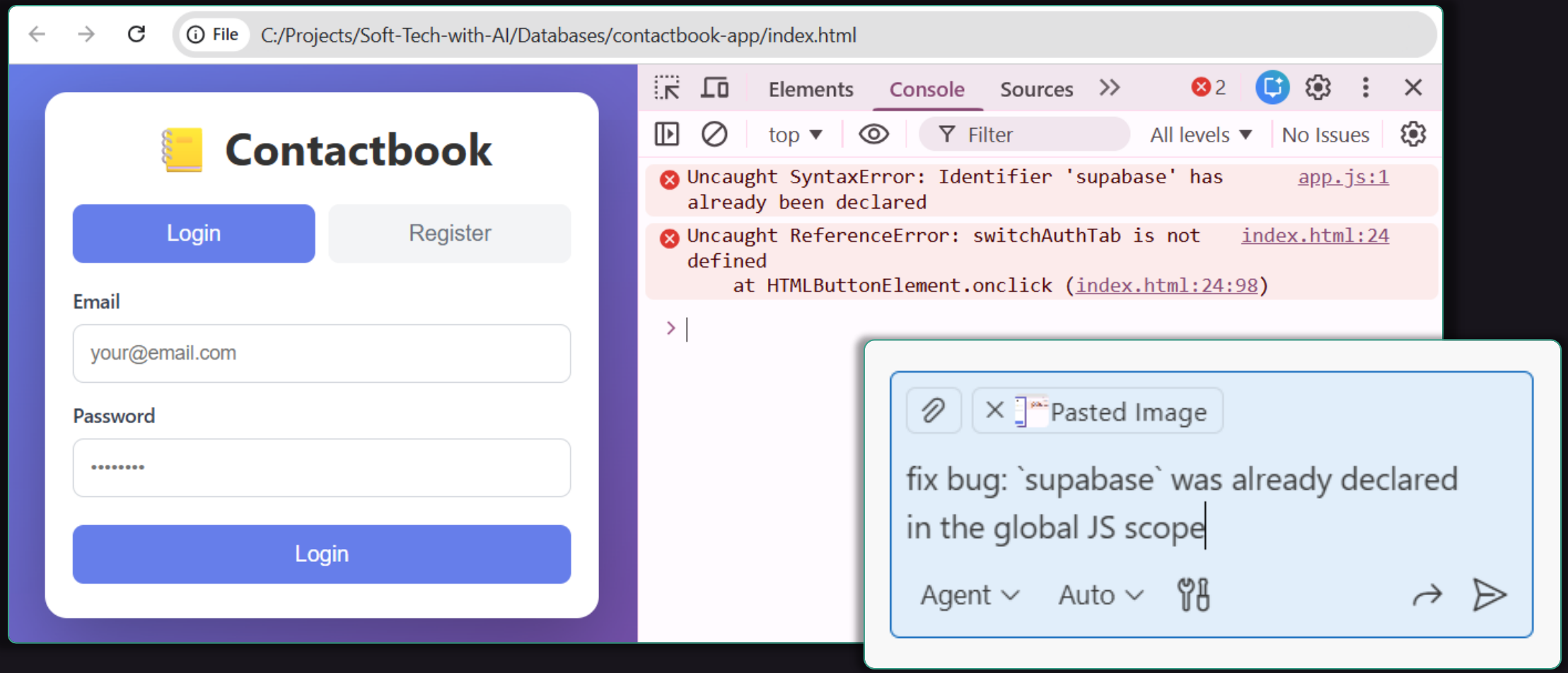
- **Step 3**: Create the **Contactbook** app with users:

Create a "Contactbook" app: login, view / add / edit / delete contacts, logout
- Use **Supabase as backend** (acess it through the Supabase MCP)
  - Each user has **own list of contacts**, displayed as cards
  - Each **contact** holds **name** + **email** + **phone** + **comments**
- Implement user **register** / **login** / **logout** using Supabase Auth
- Use simple **HTML** + **CSS** + **JavaScript** code
  - At startup display the **login** / **register** form
  - After login, **list contacts** as cards (with view / edit / delete buttons)
  - Implement **view**, **add**, **edit** and **delete** confirm with **popups**
  - Use **toast notifications** for errors and info messages (auto-close)
  - Implement a **loading spinner** at the top of the app screen



Contactbook App with Users (3)

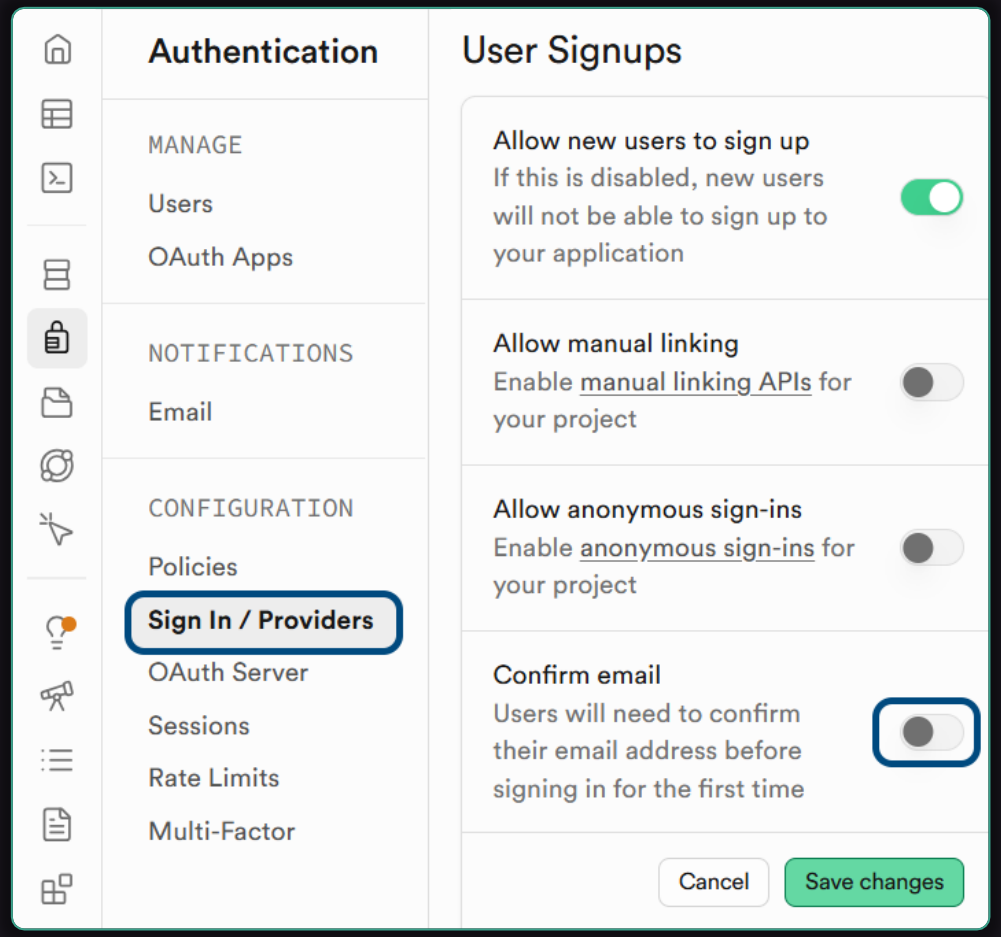
- **Step 4**: Run the app and **fix bugs** (if any)

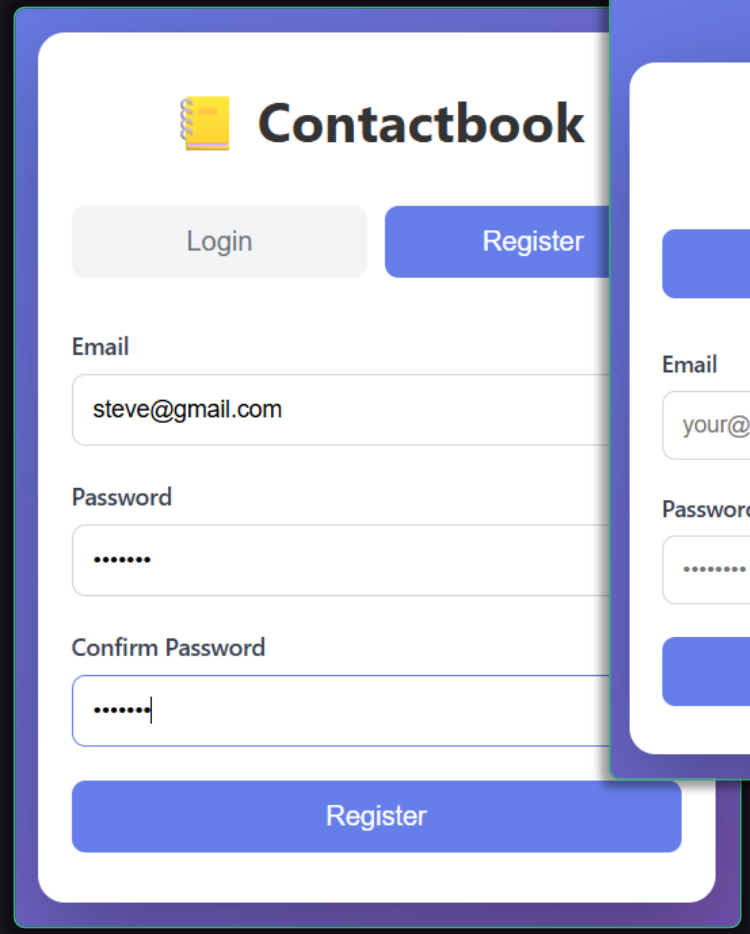


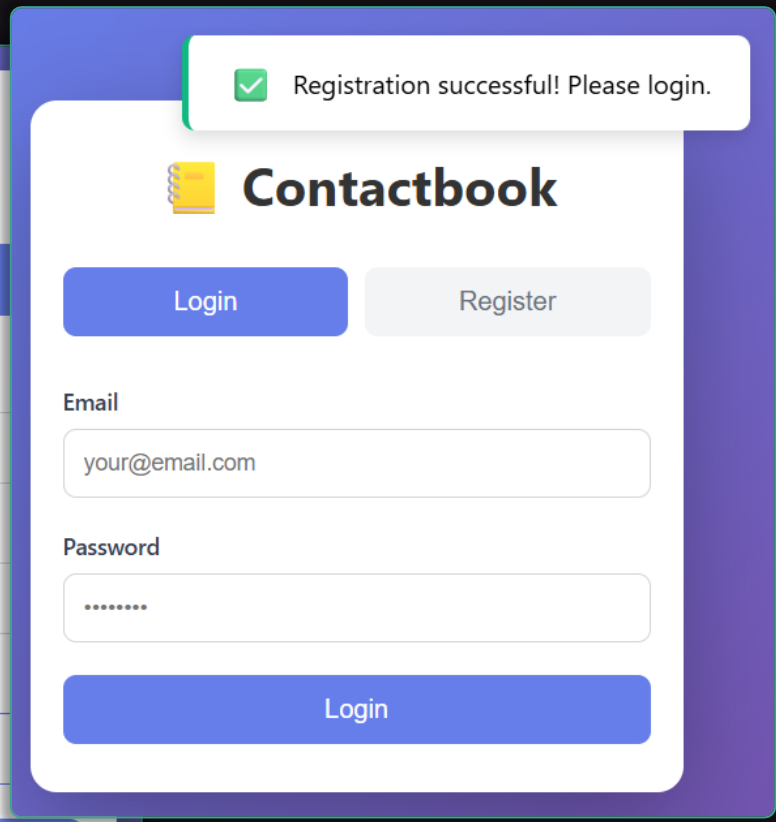
Register and Login in the App

SoftUni AI

- Switch off "**Confirm email**" in Supabase

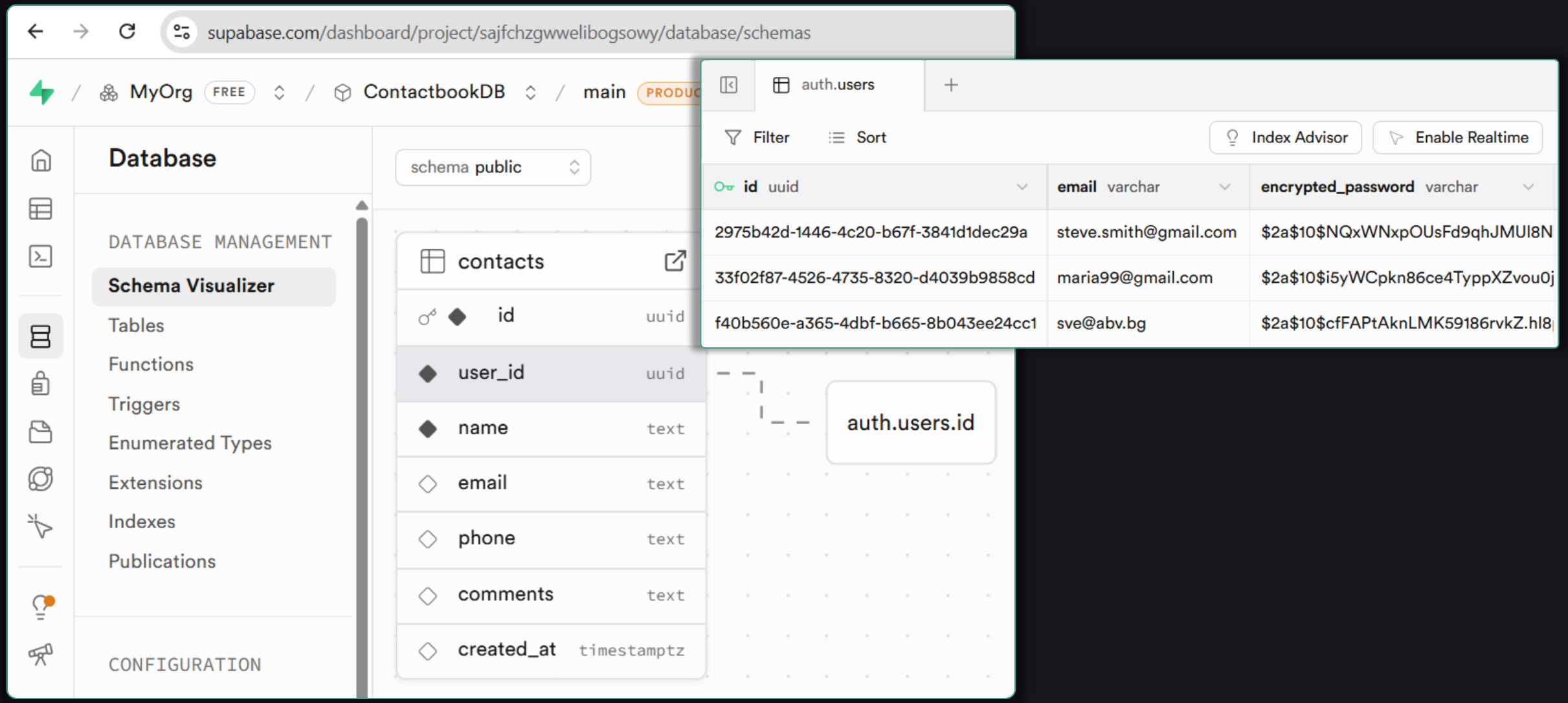






App Tables in Supabase

SoftUni AI

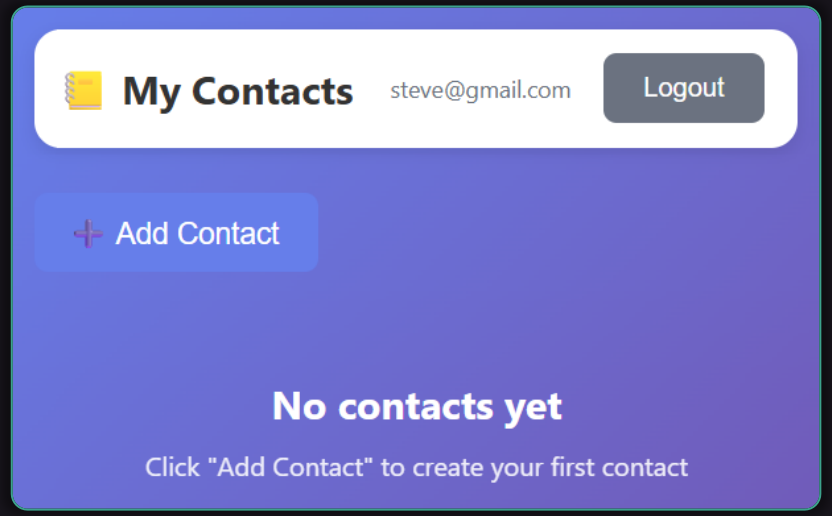


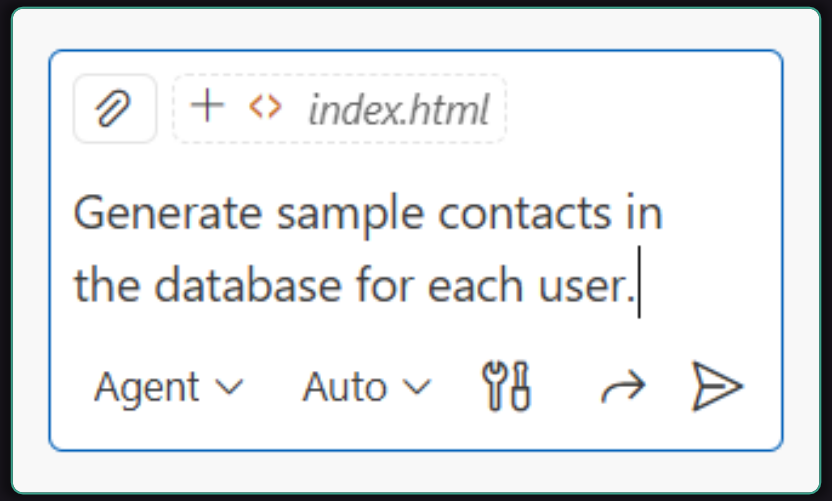
Contactbook App after Login

SoftUni AI

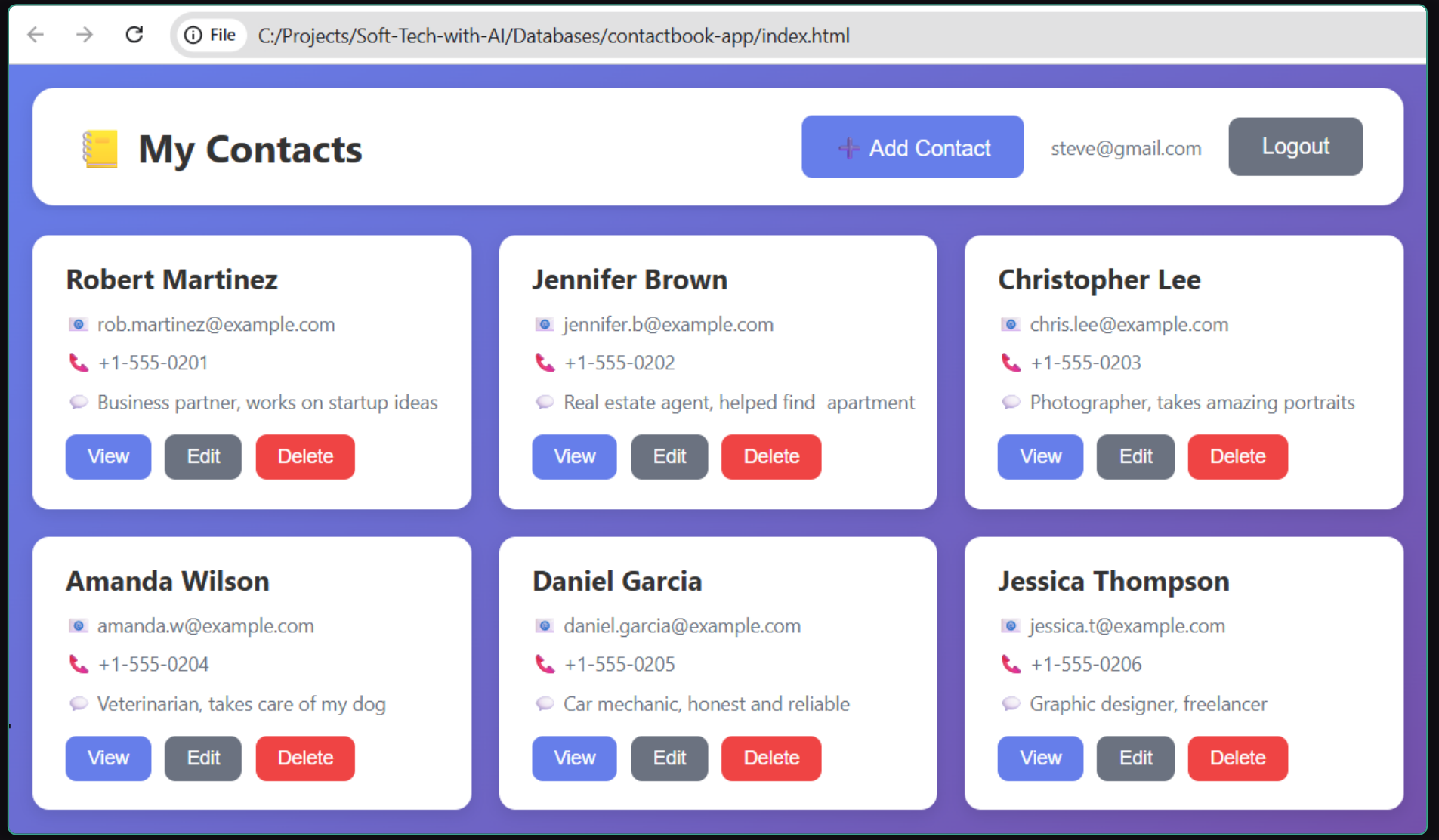
- The database is initially **empty** (no contacts yet)
- Register a 2-3 **users**
- You can **generate sample data** in the DB with an AI prompt:

> Generate **sample contacts** in the database for each user.

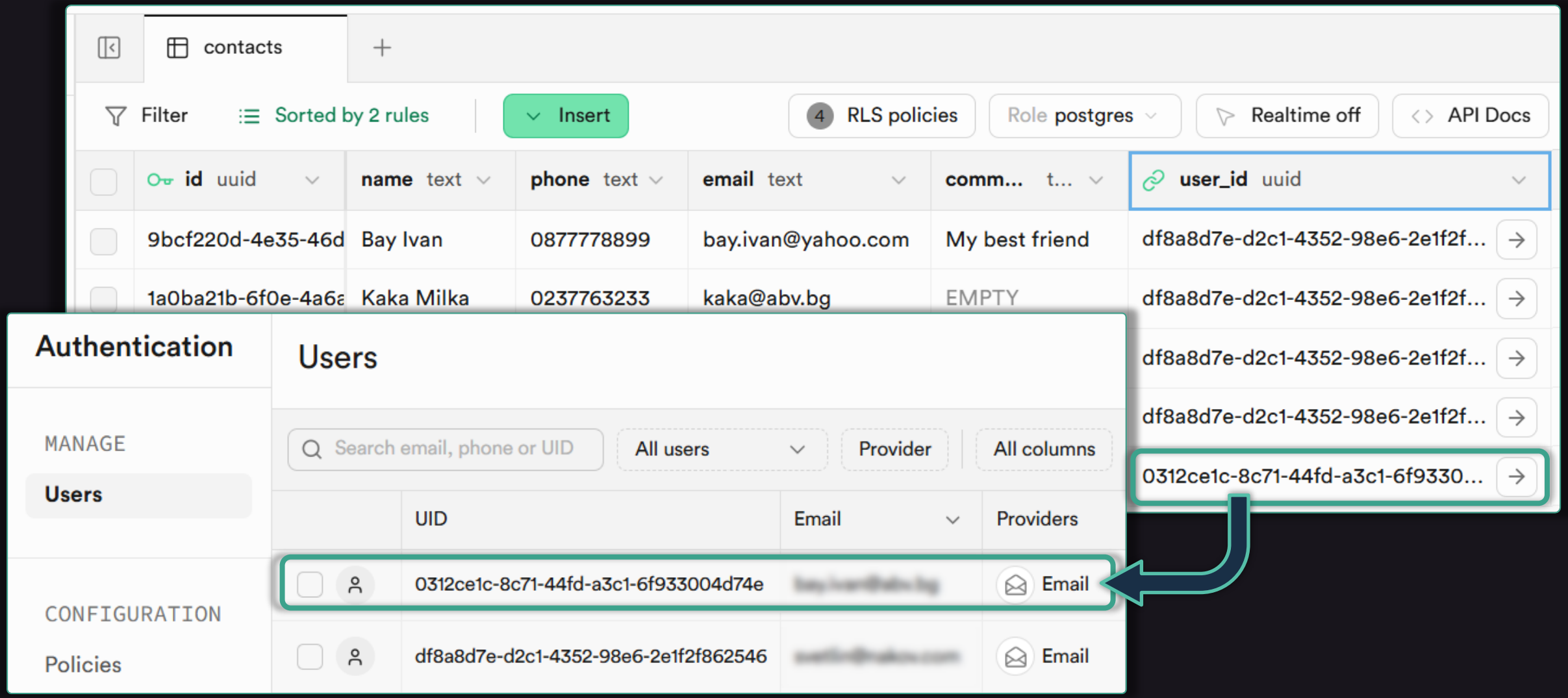




Contactbook with Sample Contacts

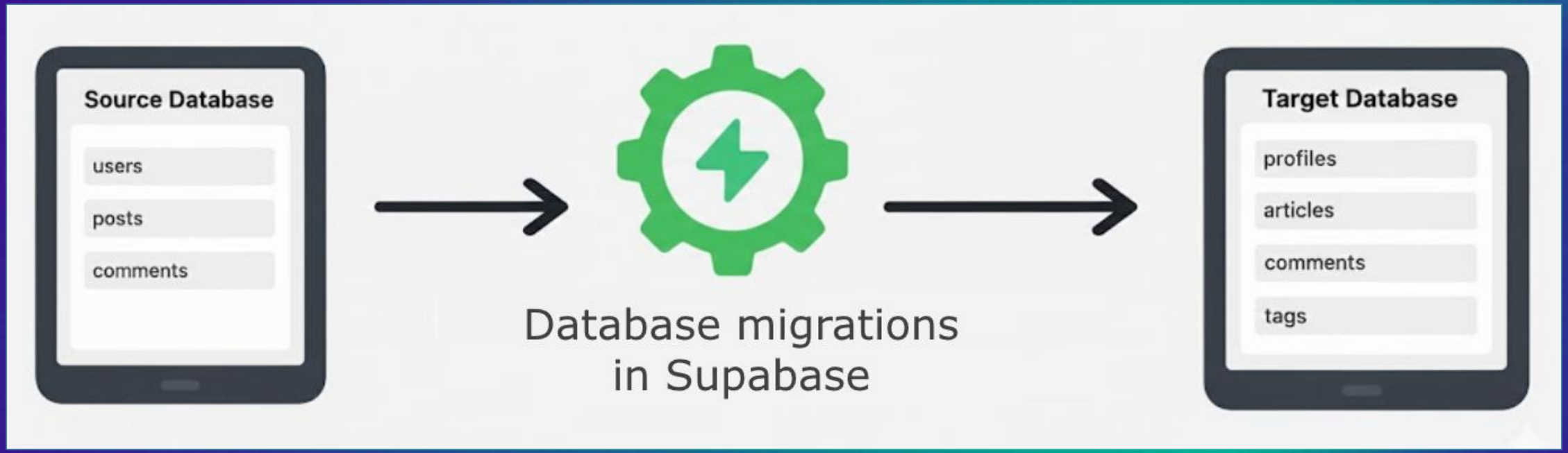


Users and Contacts Tables in the DB



Database Migrations

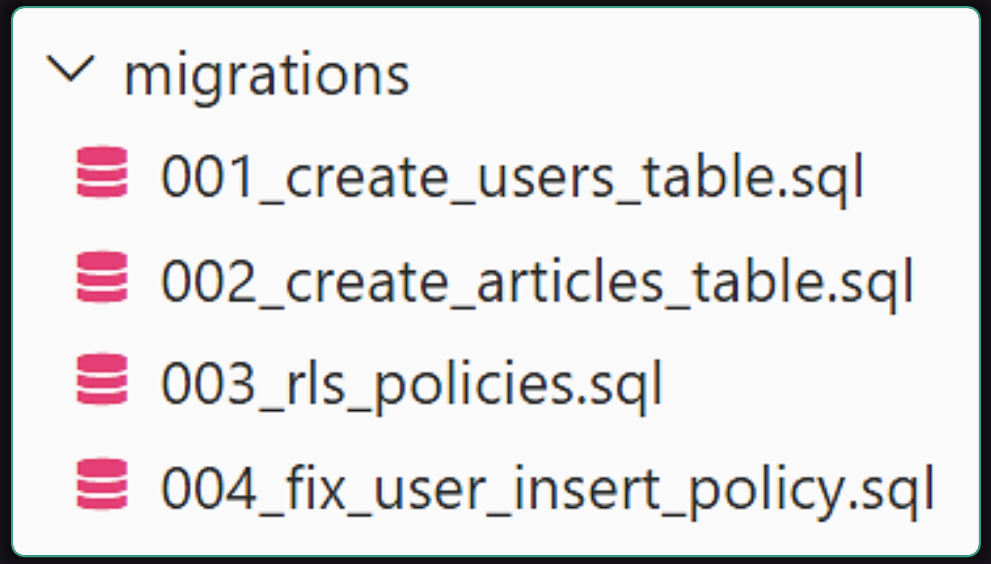
Tracking DB Schema Modifications in Supabase



Database Migrations

SoftUni AI

- **Database migrations** are how developers safely version, share, and deploy changes to the PostgreSQL schema
  - Track changes in tables, columns, functions, policies, etc. are stored as **SQL scripts**, ordered by **time**



migrations > 002_create_articles_table.sql

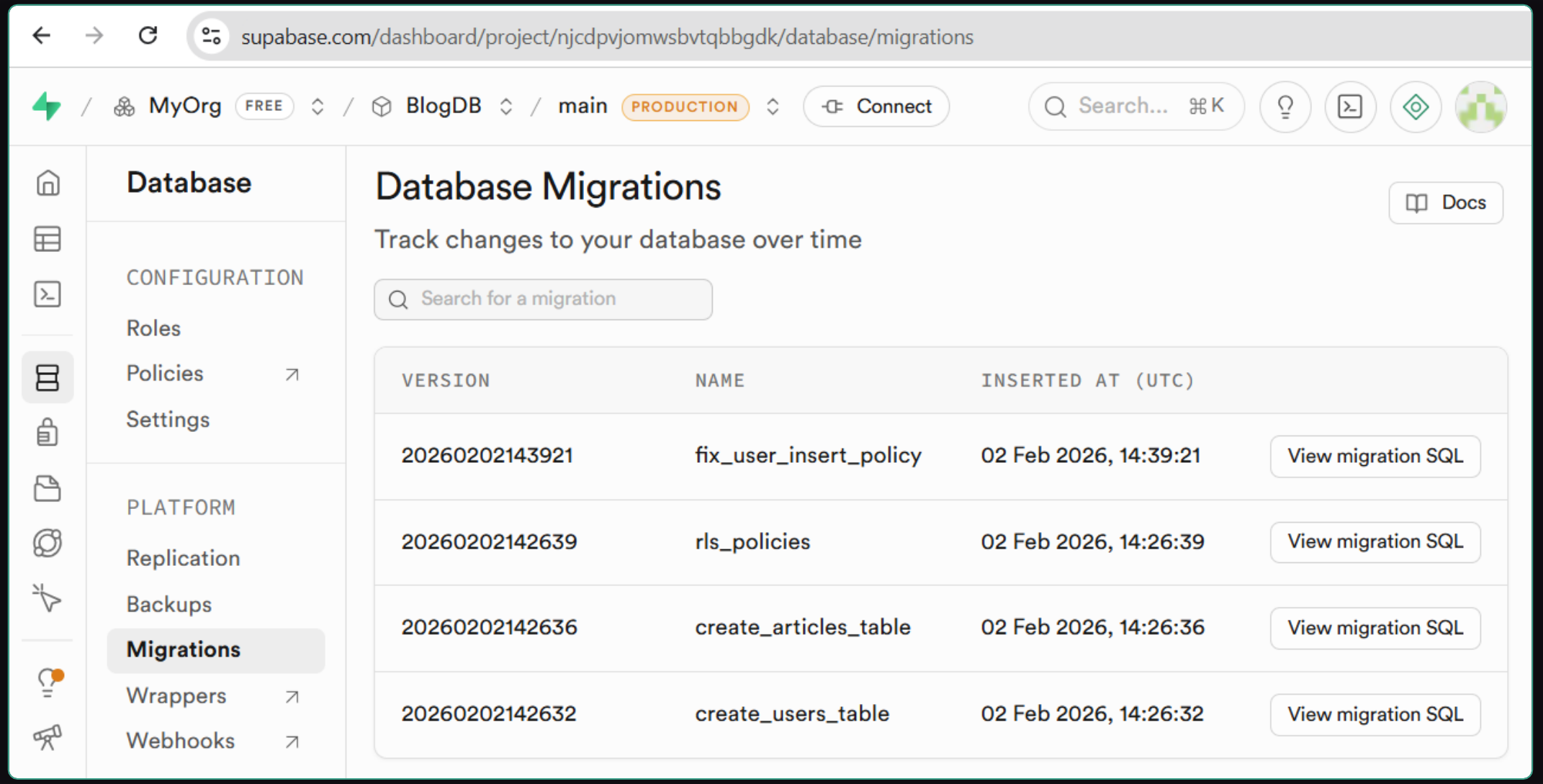
```sql
-- Create articles table
CREATE TABLE articles (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  owner_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  title TEXT NOT NULL,
  content TEXT NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Create indexes for better query performance
CREATE INDEX idx_articles_owner_id ON articles(owner_id);
CREATE INDEX idx_articles_created_at ON articles(created_at DESC);

-- Enable Row Level Security
ALTER TABLE articles ENABLE ROW LEVEL SECURITY;
```

Supabase Migrations

SoftUni AI

- **Supabase** has its own **migration infrastructure**
  - Local **SQL files** in the `**supabase/migrations**` folder
    - These files are part of your project **source code**
    - It is committed to **Git** and version tracked
    - Once created, a migration SQL file is **never changed**
  - **Migration history table** in the DB: **supabase_migrations.schema_migrations**
    - It holds the migrations already **applied** to the DB schema

Supabase Migrations – Example

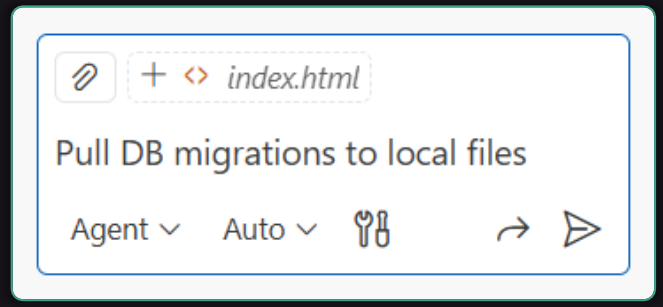


Working with Migrations

SoftUni AI

- **Supabase MCP** typically creates **migrations** for each change it makes in the DB schema
- If not explicitly requested, Supabase migrations stay inside the **database only** (local SQL files are missing)
  - This is **wrong**: DB schema migrations are part of **the code**
- Run this **AI prompt** to store all DB migrations to **local files**

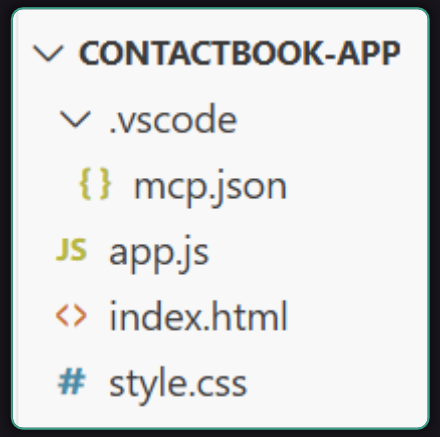
Pull DB migrations to local files



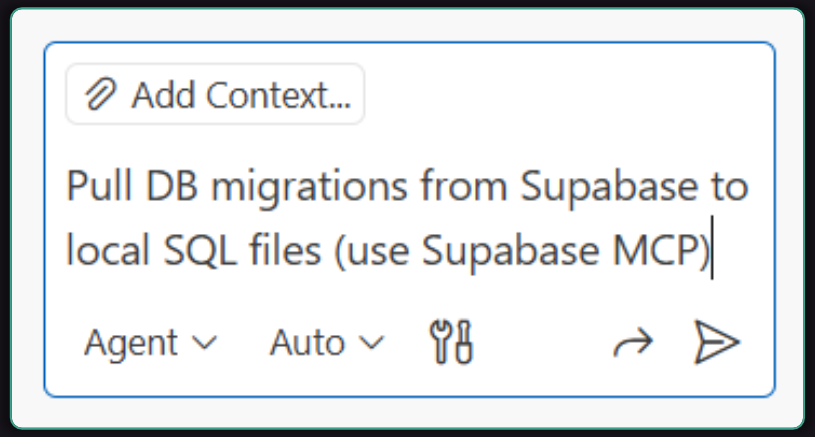
Example: Contactbook Migrations

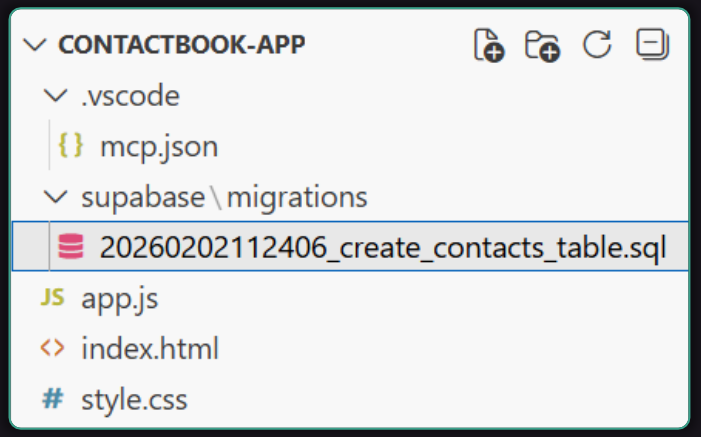
SoftUni AI

- Let's take the **Contactbook app** from the previous example
  - Simple app, which holds contacts in **Supabase** and implements **login**, **view** / **add** / **edit** / **delete** contacts, **logout**
- The app code does not have any DB migrations scripts:



- Let's **pull the migrations** from Supabase to local SQL files:



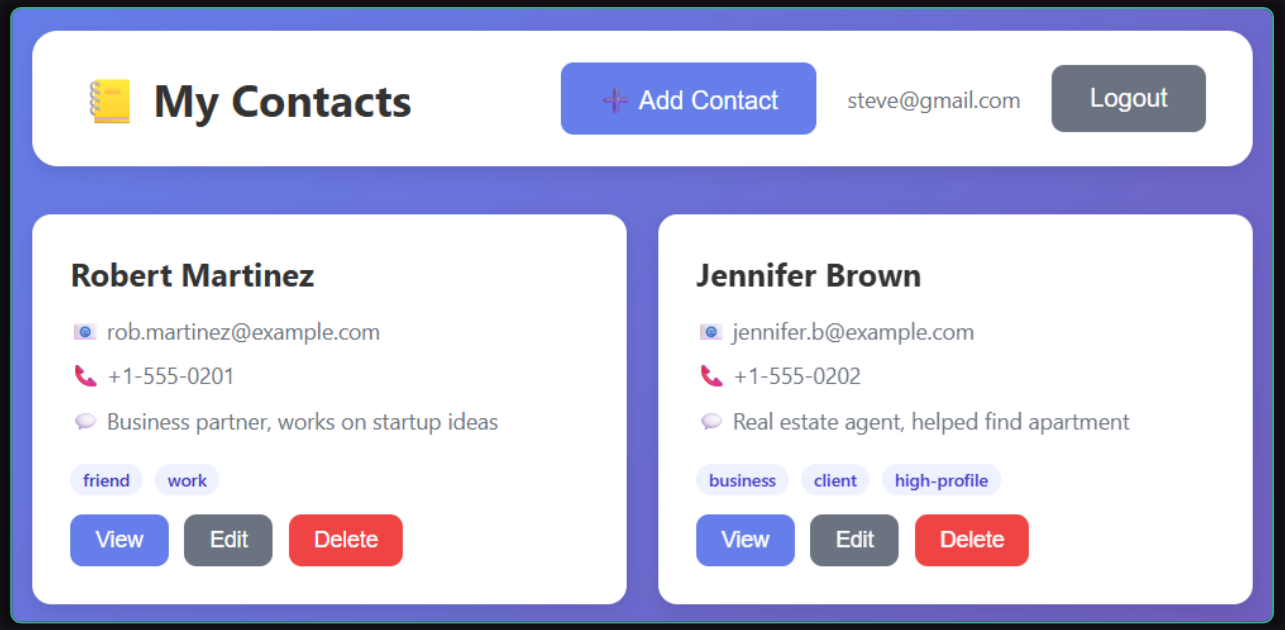


Example: Creating New Migrations

SoftUni AI

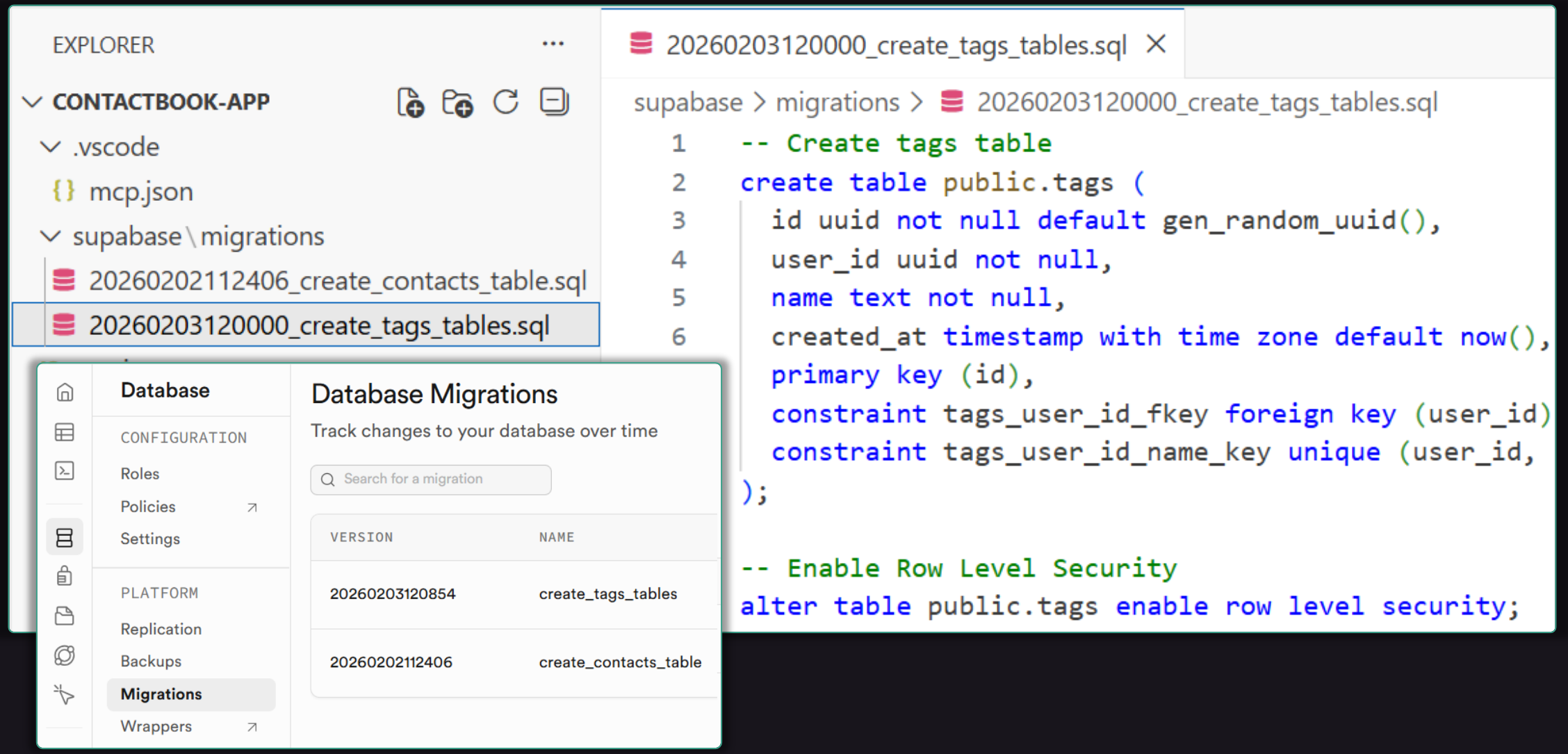
- Let's **implement tags** in the Contactbook app
  - This will create a **migration** to add **new tables** in the DB
  - It will also **extend the app UI** to display and edit tags

Implement tags in the Contactbook **contacts**:
- Add and run migrations to add relevant tables to the DB
- Extend the app **UI** to display and edit tags



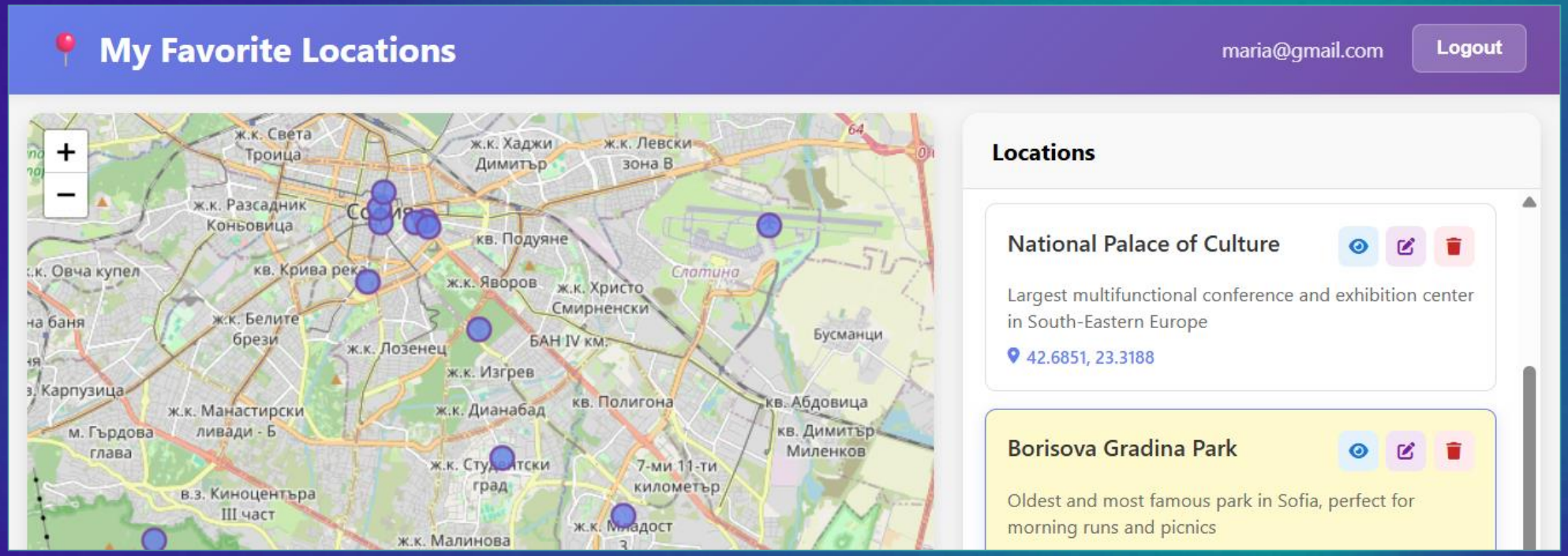
Migrations as SQL in the DB

SoftUni AI



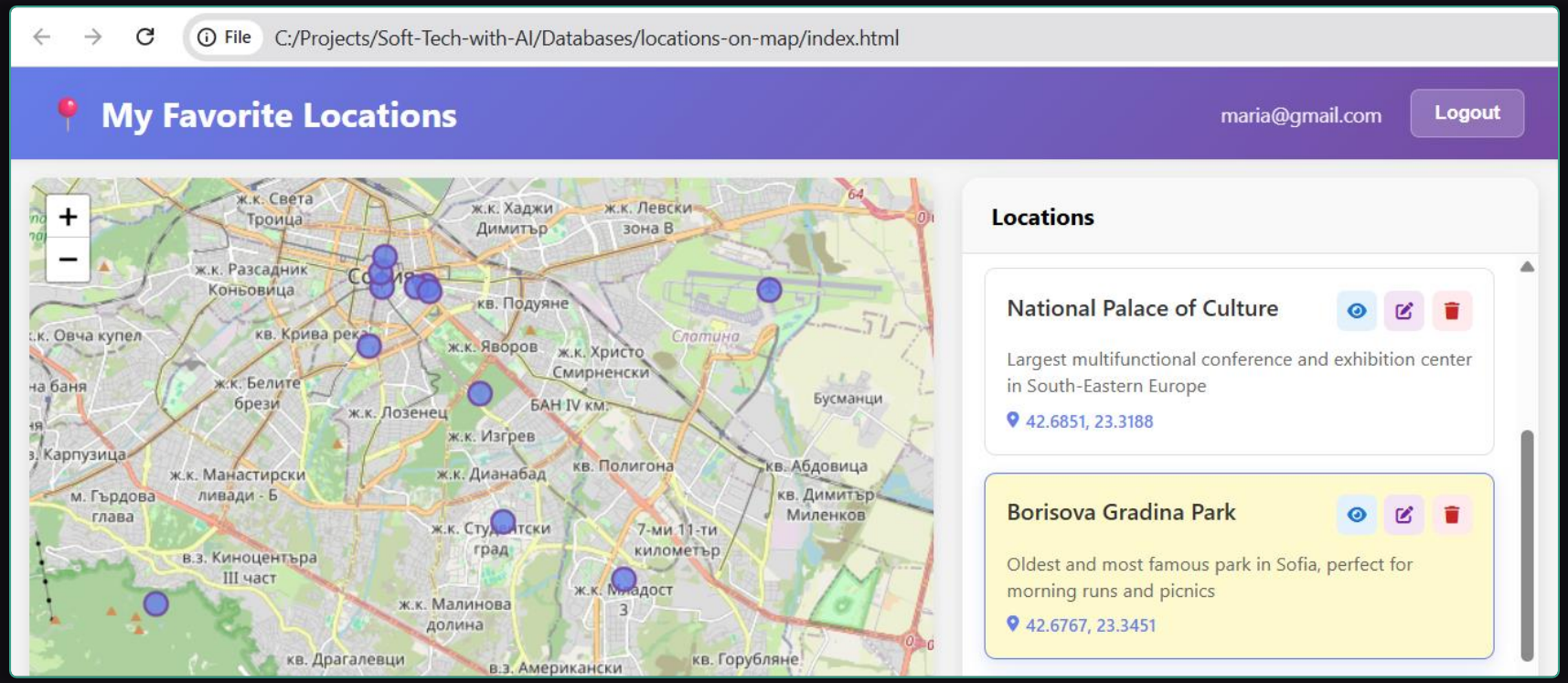
Locations on the Map

Building an App with Supabase Back-End with Users, Data, RLS Policies and Migrations



Example: Locations on the Map

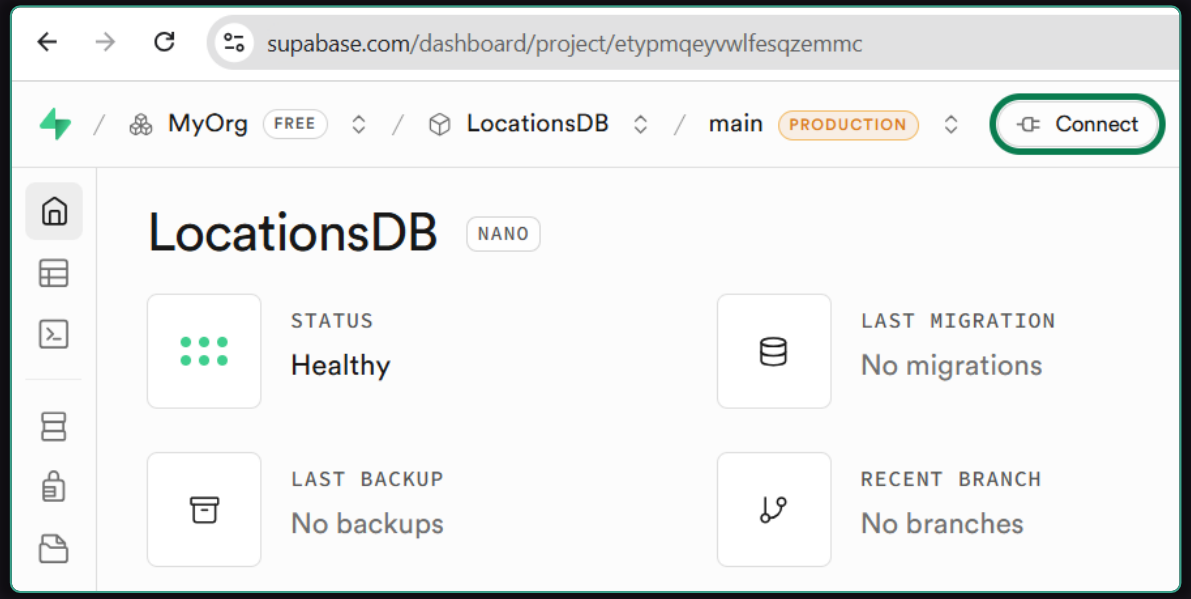
- Let's create an app with **Supabase MCP** and **DB migrations**
- **Locations on the Map** app: users log-in and pin locations on a visual map with locations editor of the right

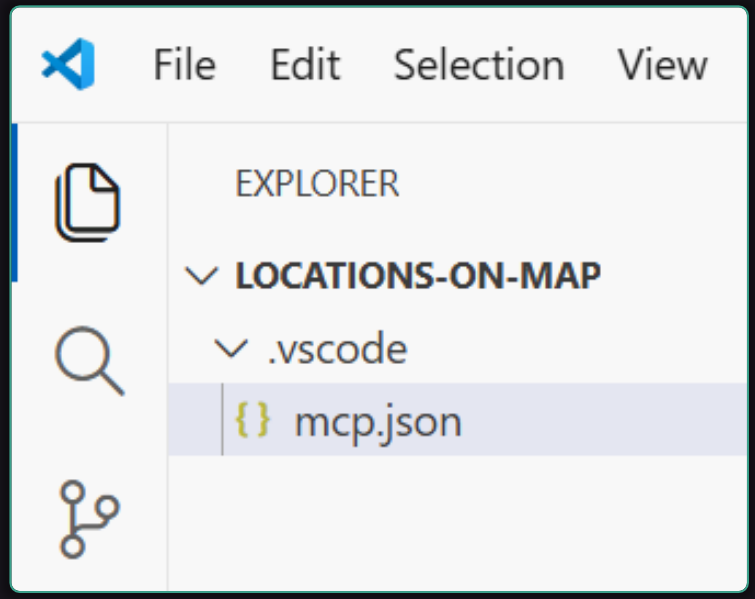


Locations on the Map: Supabase DB

SoftUni AI

- **Step 1: Supabase Project + MCP Server**
  - Create a new **project in Supabase**
  - Install the **Supabase MCP**: Supabase Dashboard → Connect → MCP → VS Code → Install in Workspace → Authorize





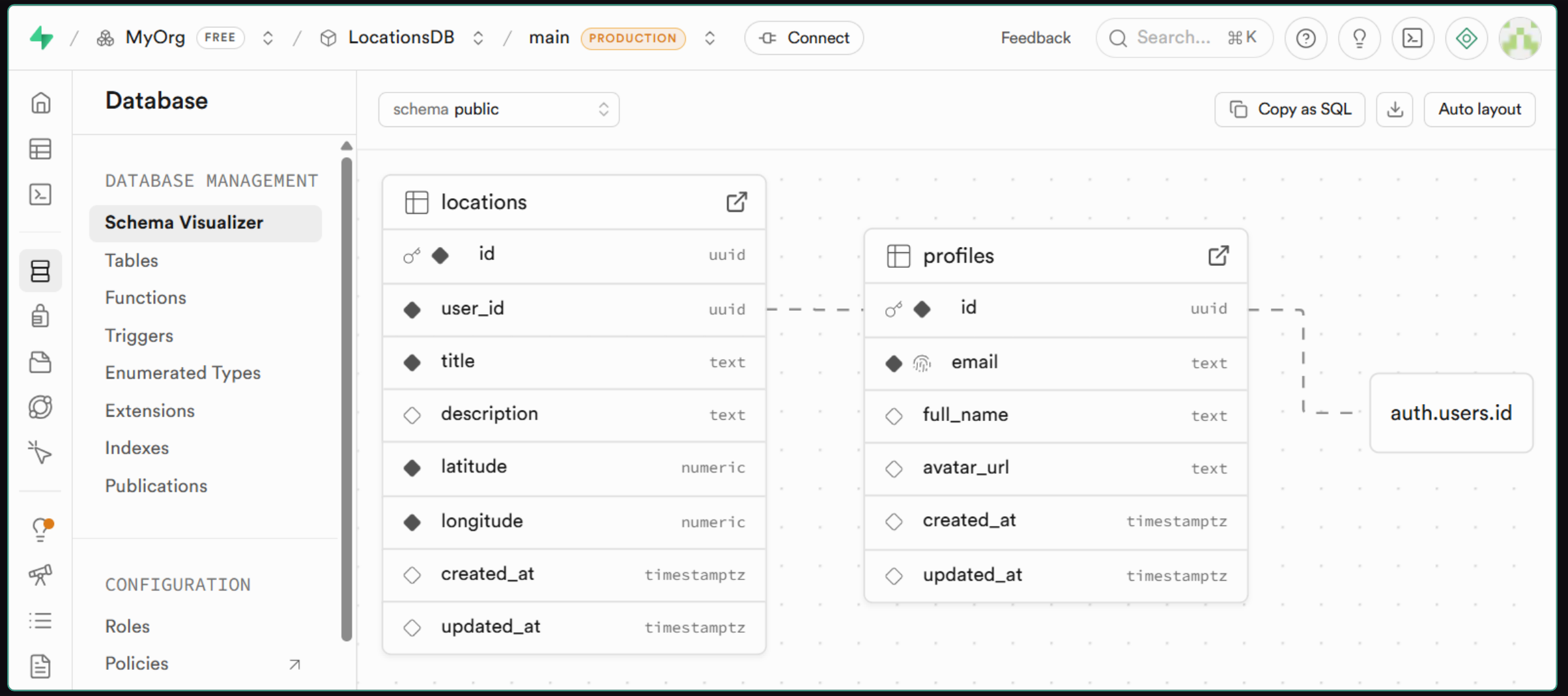
Locations on the Map: DB Schema

SoftUni AI

- **Step 2**: **Create the DB Schema** for the app

Create the **DB schema** for the app "**Locations on the Map**"
- Use **Supabase** as backend to store **users** and **locations**
- Create a **DB schema** to hold locations on a map
  - **Users** have their own **lists of locations**
  - Each **location** holds **GPS coordinates** + **title** + **description**
- Use the **Supabase MCP** to access the DB directly
- Keep track of **DB migrations** as local SQL files
- Define **RLS policies** to protect users' data

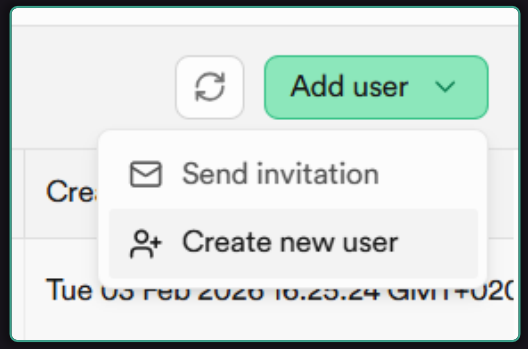
Locations DB Schema – Example

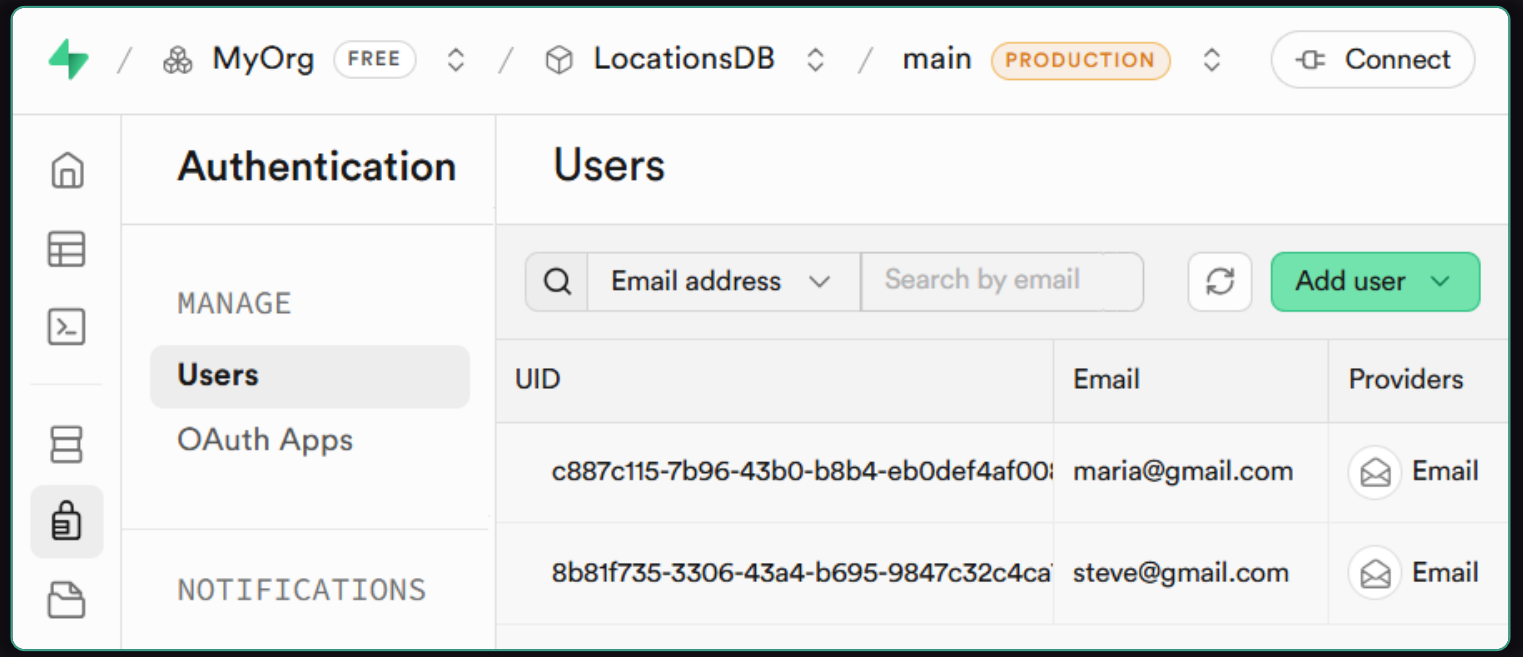


Locations on the Map: Create Users

SoftUni AI

- **Step 3**: Create **users** in Supabase
  - Using Supabase Studio, **register sample users** in the Supabase Auth system:
  - maria@gmail.com / pass123
  - steve@gmail.com / pass123
- **Note**: Supabase MCP cannot register users



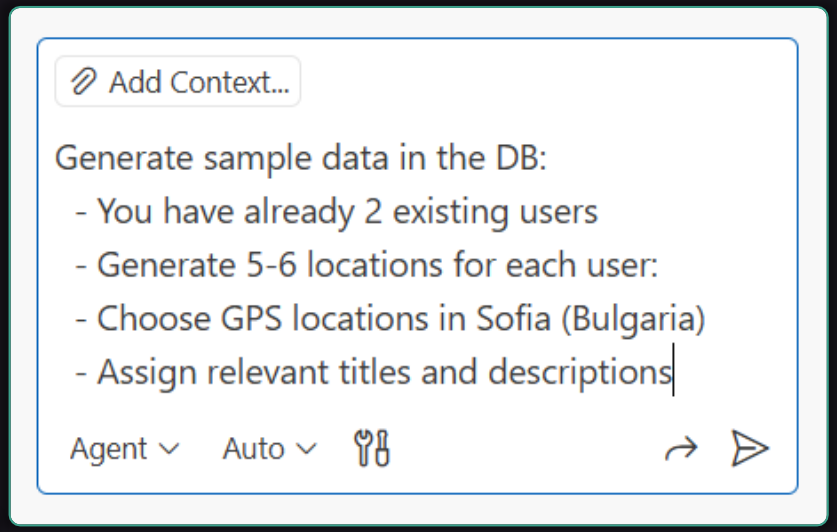


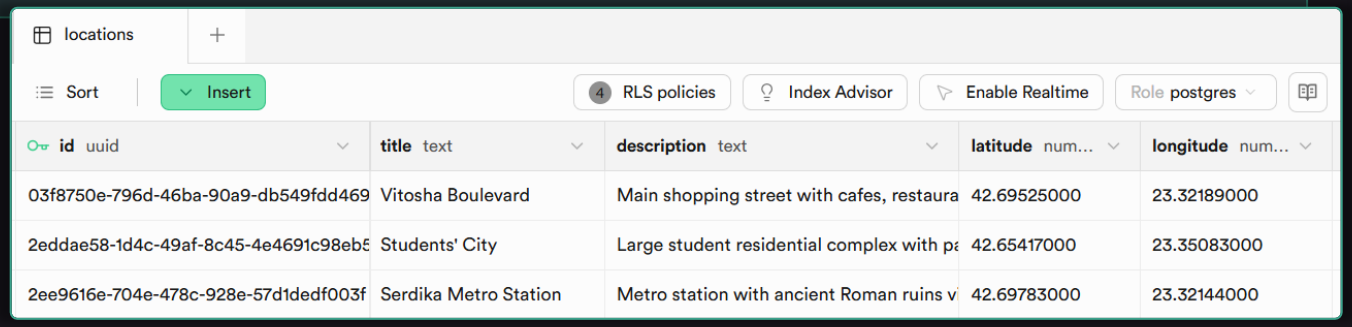
Locations on the Map: Sample Data

SoftUni AI

- **Step 4**: Generate **sample data**
  - Using Supabase MCP, **generate sample data** in the DB:

Generate **sample data** in the DB:
- You have already 2 **existing users**
- Generate **5-6 locations** for each user:
- Choose **GPS locations** in Sofia (Bulgaria)
- Assign relevant **titles** and **descriptions**





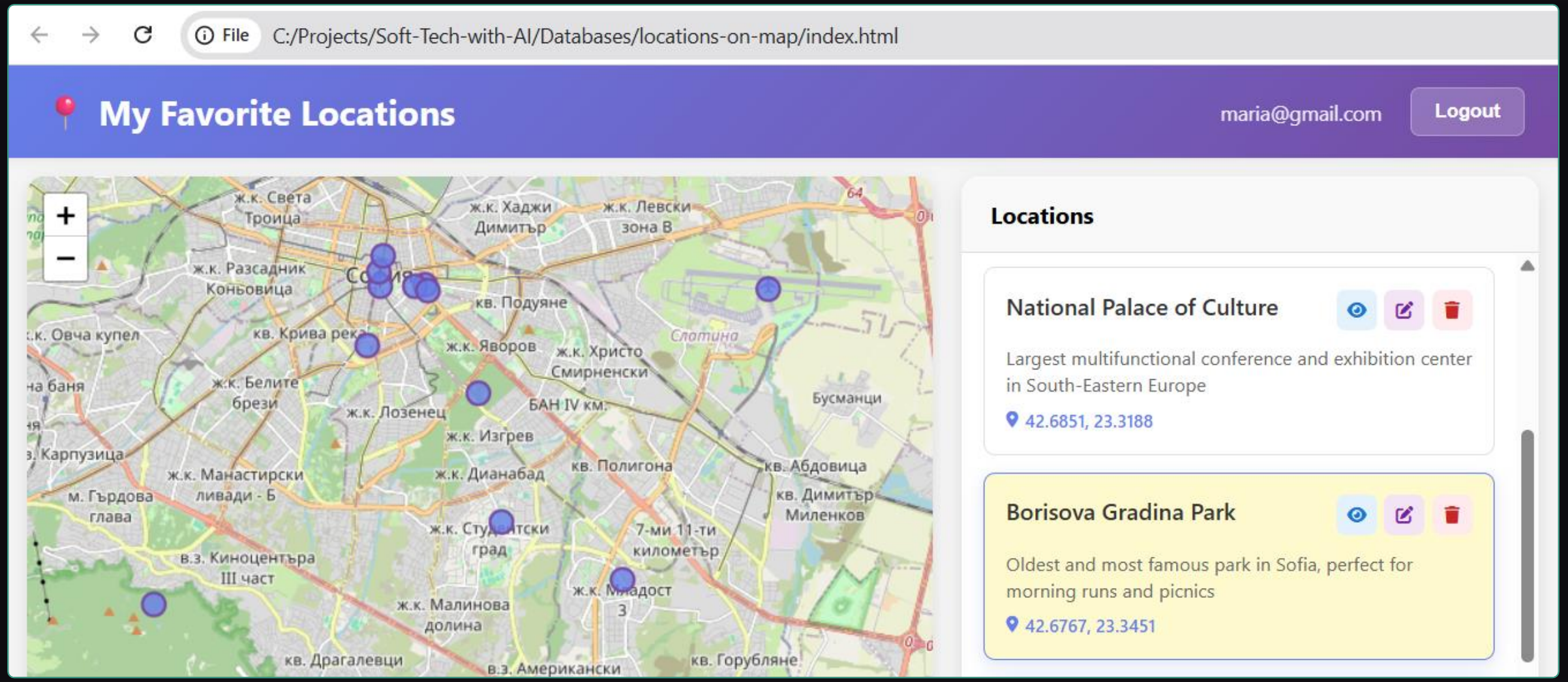
id uuid	title text	description text	latitude num...	longitude num...
03f8750e-796d-46ba-90a9-db549fdd469	Vitosha Boulevard	Main shopping street with cafes, restaura	42.69525000	23.32189000
2eddae58-1d4c-49af-8c45-4e4691c98eb5	Students' City	Large student residential complex with pa	42.65417000	23.35083000
2ee9616e-704e-478c-928e-57d1dedf003f	Serdika Metro Station	Metro station with ancient Roman ruins v	42.69783000	23.32144000

Locations on the Map: Create the App

- **Step 5**: Create the **app "Locations on the Map"**

Create an app to visualize favorite locations on a map
- Features: **login** / **logout**, **manage** locations, **visualize** as pins
- Use simple **HTML** + **CSS** + **JavaScript** code
- Use a standard **map UI** component like **Leaflet**
- Display the map visually with **pins** (pinned locations)
- All **locations** are shown on the right side as list of **cards**
  - Click on the map to **create a pin** at certain location (use a popup)
  - Implement **view**, **edit** and **delete** location cards with popups
- **Pin click** shows the location on the right; **location click** shows the pin
- Use **icons** to make view, edit and delete buttons small

Locations on the Map – The App



Row-Level Security (RLS)

Defining Database-Level Access Control

SQL CREATE POLICY

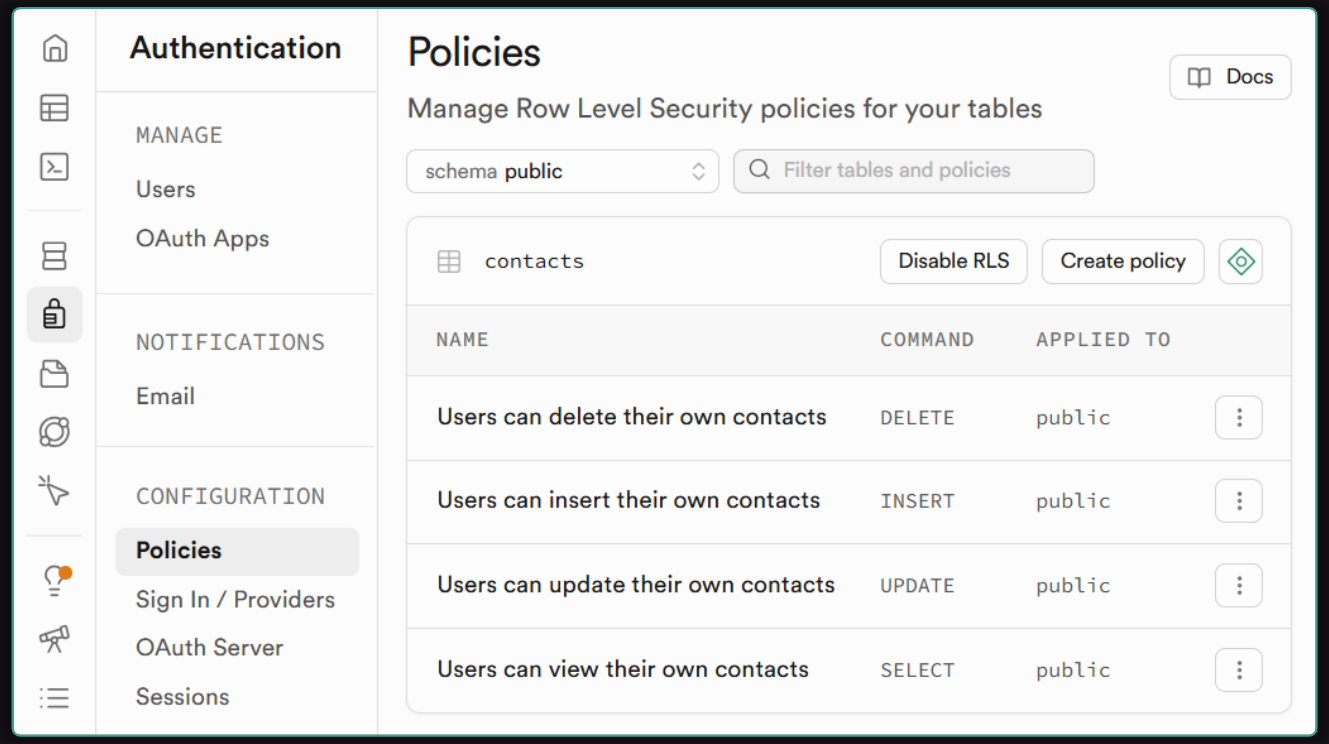
Row-Level Security (RLS)

SoftUni AI

- **Row-Level Security** (RLS) defines which rows a user can read or write (access control)
  - Defined by **security policies** in the database (SQL code)
  - Based on the **logged-in user**: anonymous / authenticated
- Typical RLS policy:
  - **Anonymous** users: can **read all public data**, e. g. products
  - **Authenticated users**: can read / write their **own data**, e. g. own products, own messages, own user profile
  - **Admin users**: can read / write **any data** in the DB

Sample RLS Policies

- Sample **RLS policies** in Supabase:




```
CREATE POLICY "Users can
update their contacts"
ON contacts FOR UPDATE
USING (
   (auth.uid() = user_id)
);
```

Blog System with Users and RLS

SoftUni AI

- We want to create a "**Blog**" web site
  - Site **visitors** can **view** and **read** all published articles
  - Visitors can **register** / **login**
  - Logged-in **users** can **post** articles, **edit** / **delete** own articles
  - **Articles** hold **title** + **owner** + **date** + **content**
- **Step 1**: Create a **Supabase project**
  - Supabase Dashboard → Projects → New project
- **Step 2**: Connect **Supabase MCP** to VS Code
  - Supabase Dashboard → Connect → MCP → VS Code

Blog System with Users and RLS (2)

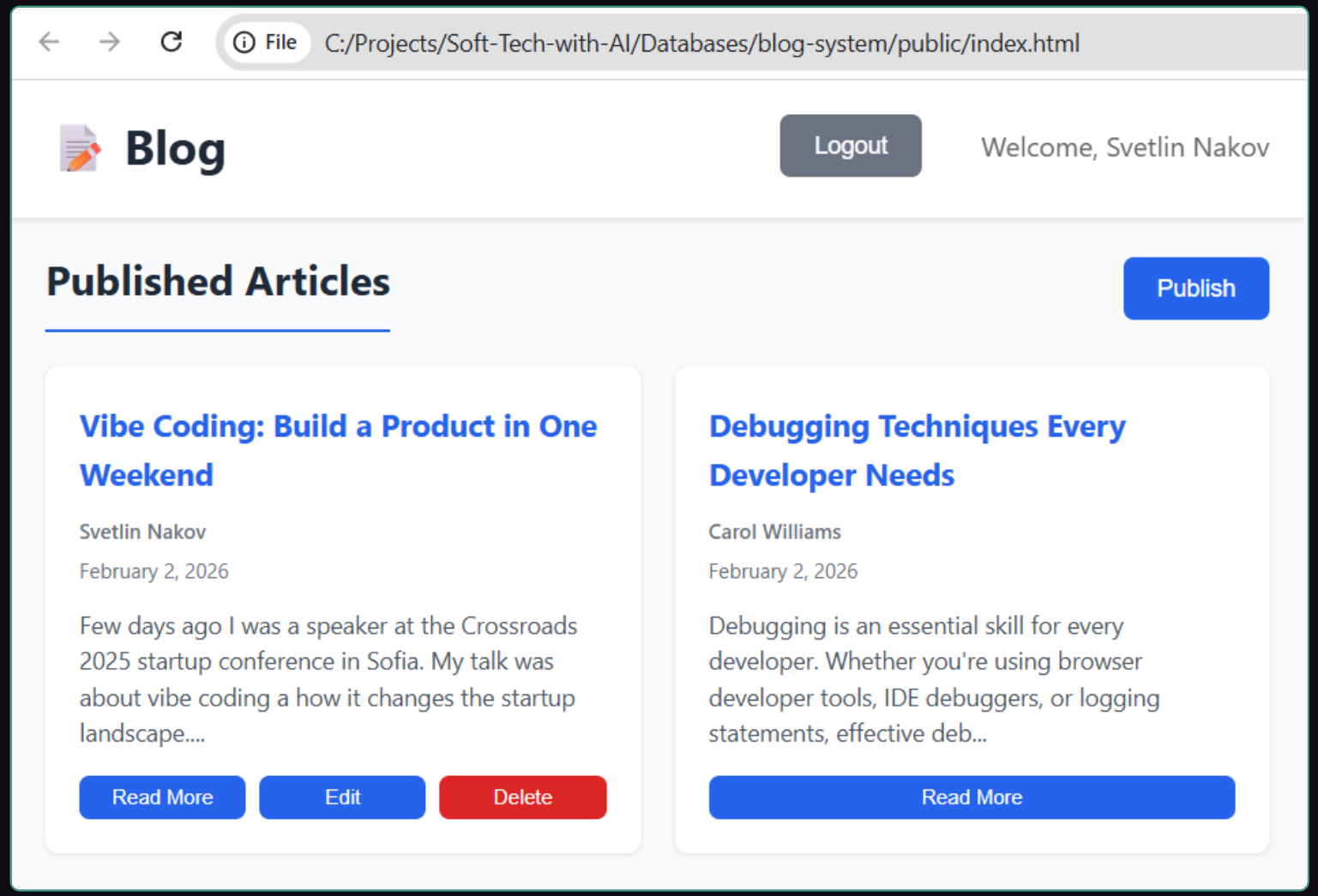
SoftUni AI

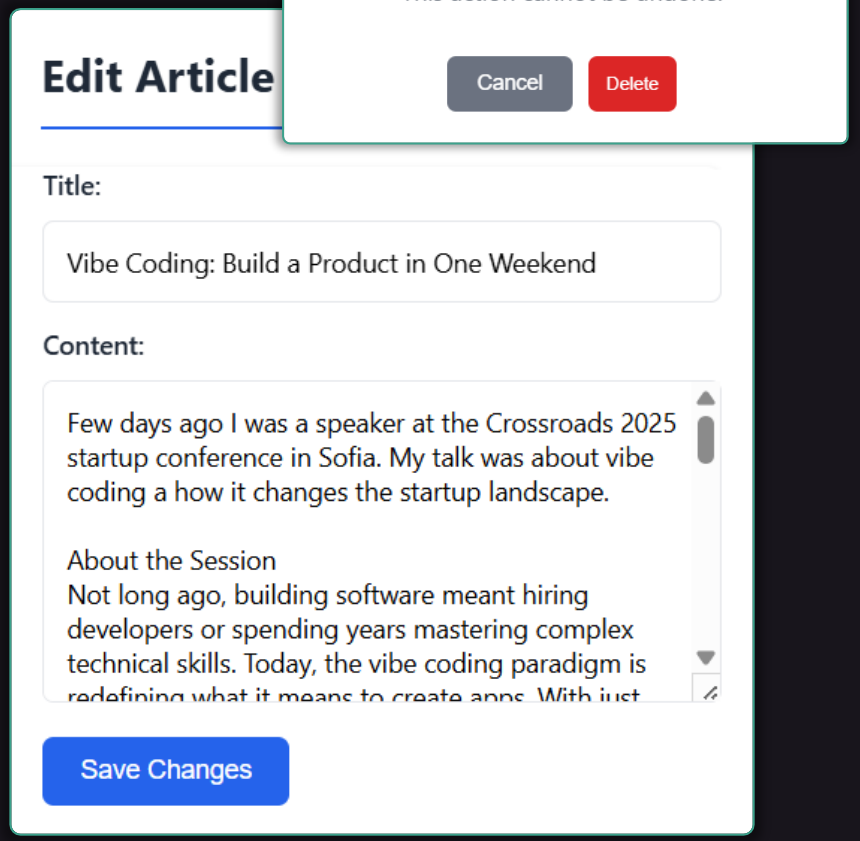
- **Step 3**: Create the **Blog web site** with users and RLS:

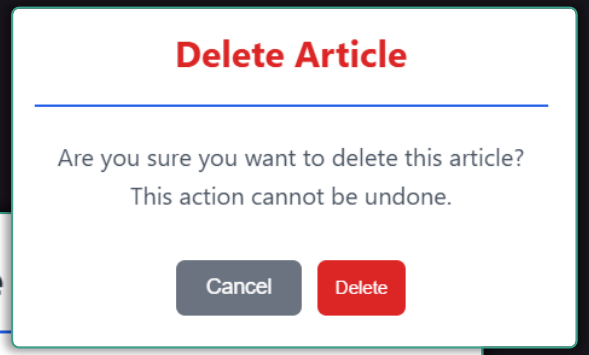
Create a "**Blog**" web site to view / publish blog articles:
  - Site **visitors** can **view** and **read** all published articles
  - **Articles** hold **title** + **owner** + **date** + **content**
  - Site visitors can **register** / **login**
  - Logged-in **users** can **post** articles, **edit** / **delete** own articles
- Hold **app data** (users, articles) in **Supabase DB**
    - Keep **DB migrations** in a local folder, synced with the DB
    - Define **RLS policies**: users create, edit, delete their own data
    - Generate **sample data** in the DB: a few blog articles (anonymous)
  - Use simple **HTML** + **CSS** + **JS**

Blog System with Users and RLS (3)

SoftUni AI



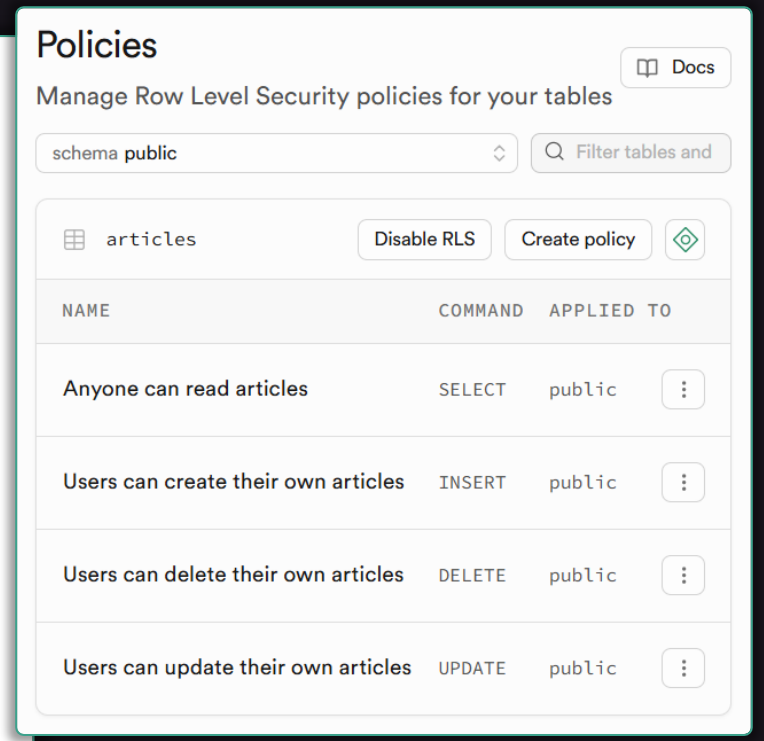




Blog System: RLS Policies

SoftUni AI

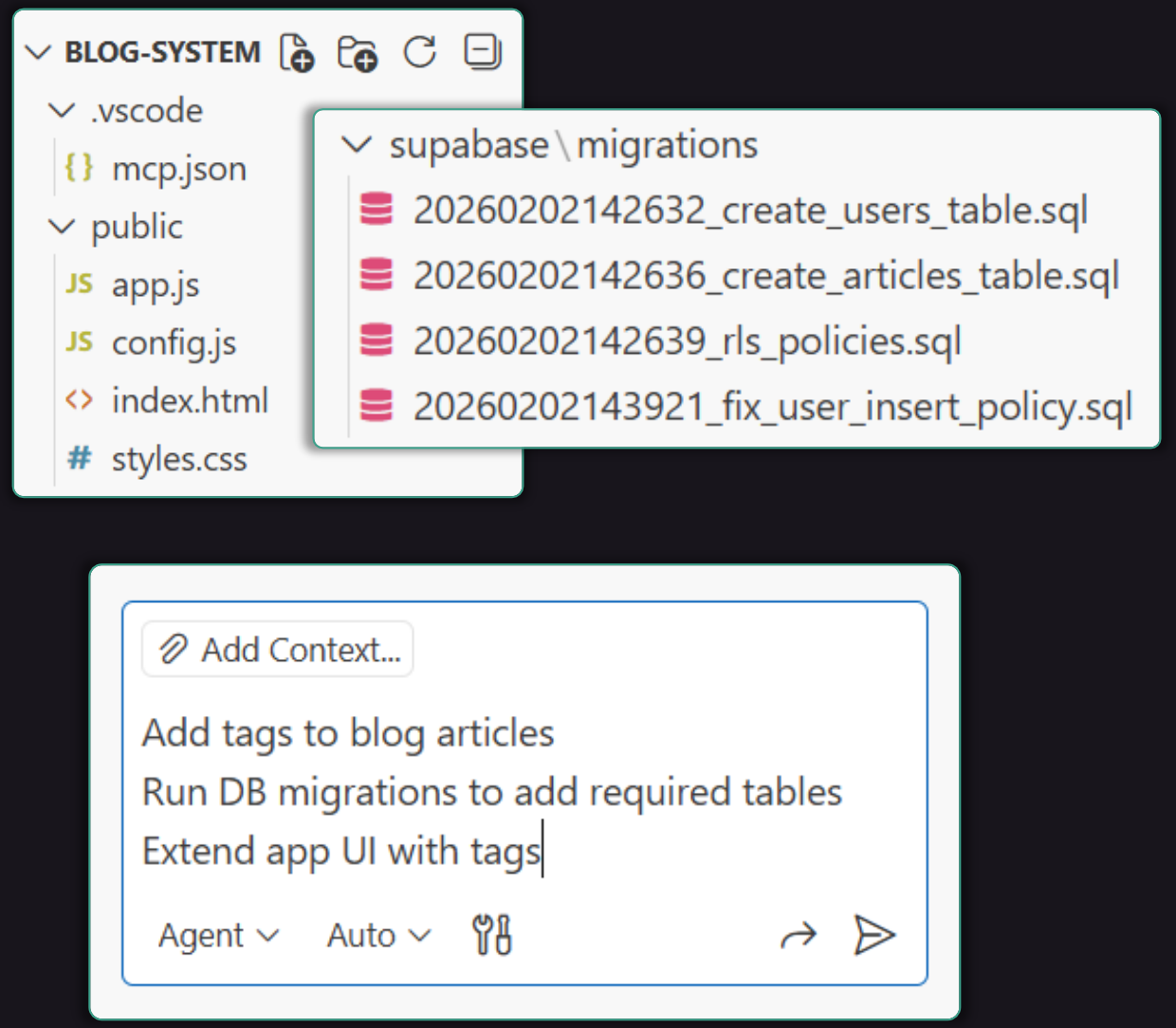
```sql
-- RLS Policies for articles table
-- Anyone can read published articles
CREATE POLICY "Anyone can read articles" ON articles
  FOR SELECT USING (true);

-- Users can only insert articles for themselves
CREATE POLICY "Users can create their own articles" ON articles
  FOR INSERT WITH CHECK (auth.uid() = owner_id);

-- Users can only update their own articles
CREATE POLICY "Users can update their own articles" ON articles
  FOR UPDATE USING (auth.uid() = owner_id);

-- Users can only delete their own articles
CREATE POLICY "Users can delete their own articles" ON articles
  FOR DELETE USING (auth.uid() = owner_id);
```



Example: Blog System Migrations

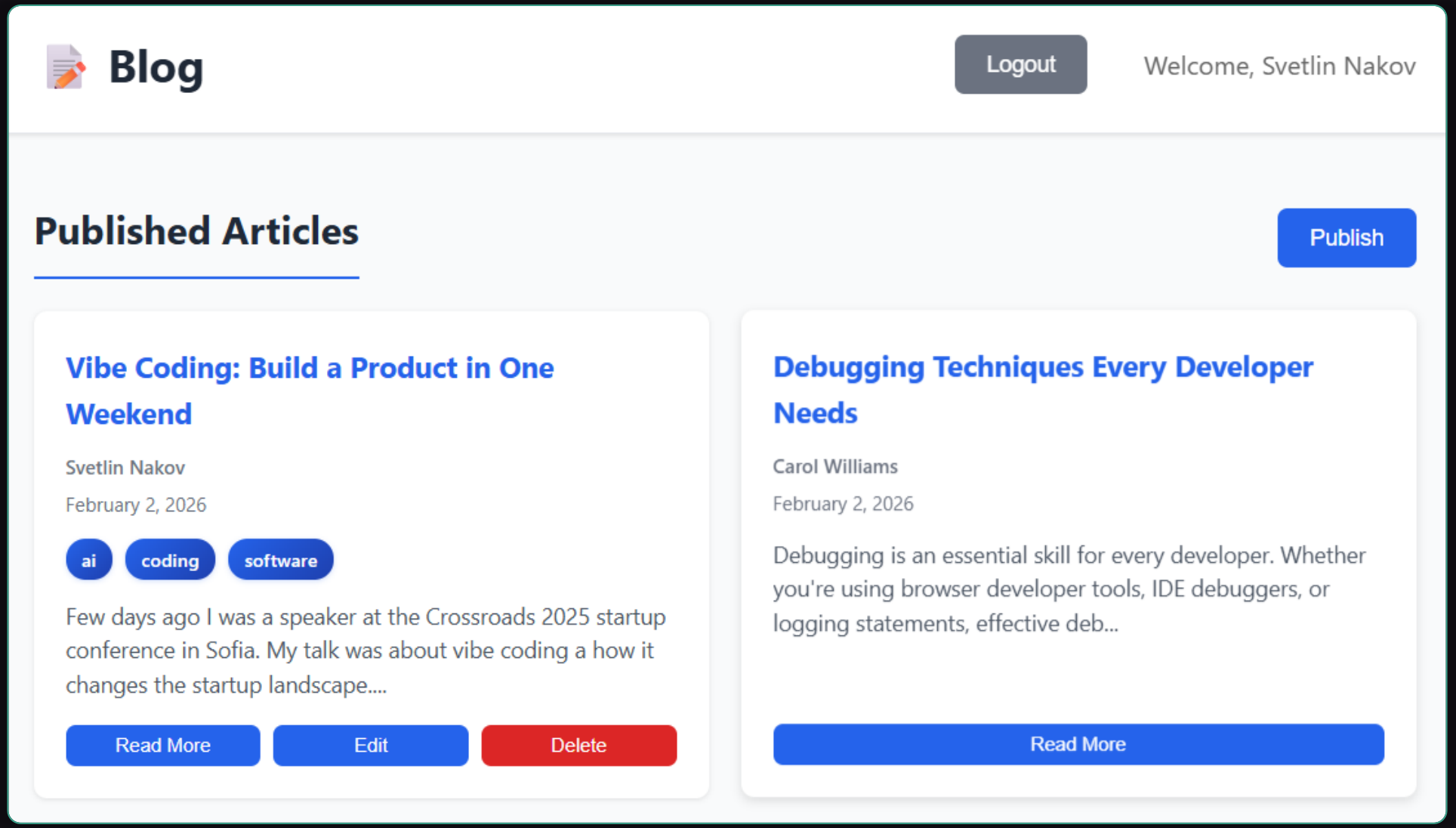
SoftUni AI

- The source code now holds:
  - **App code** (HTML, CSS, JS)
  - **DB migrations** (SQL)
- Now, let's implement **new functionality** in the blog:
  - **Add tags** to blog articles
  - This will add **DB migrations**
  - Will extend **app UI** with tags



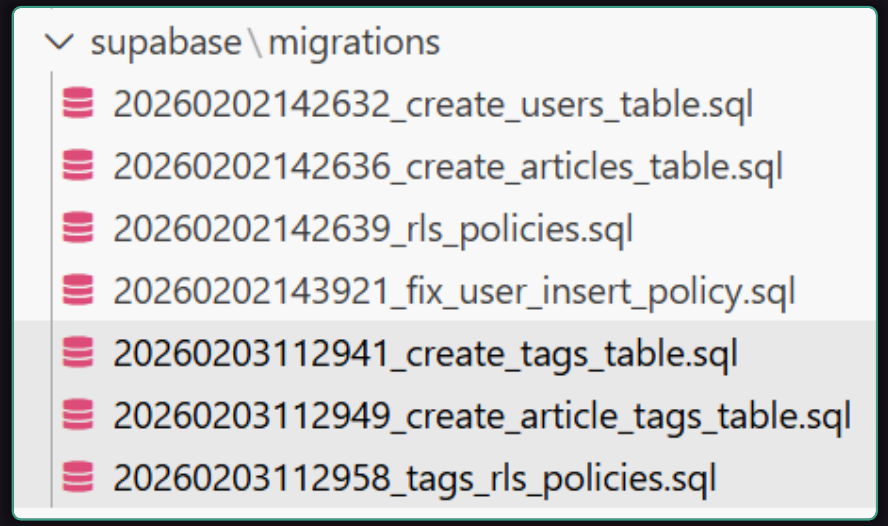
Example: Blog with Tags

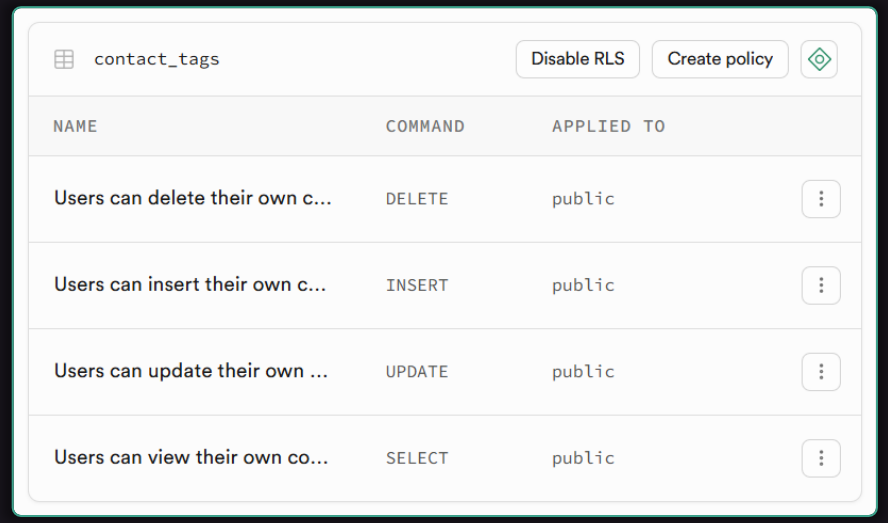
SoftUni AI



Blog System: RLS Policies

SoftUni AI



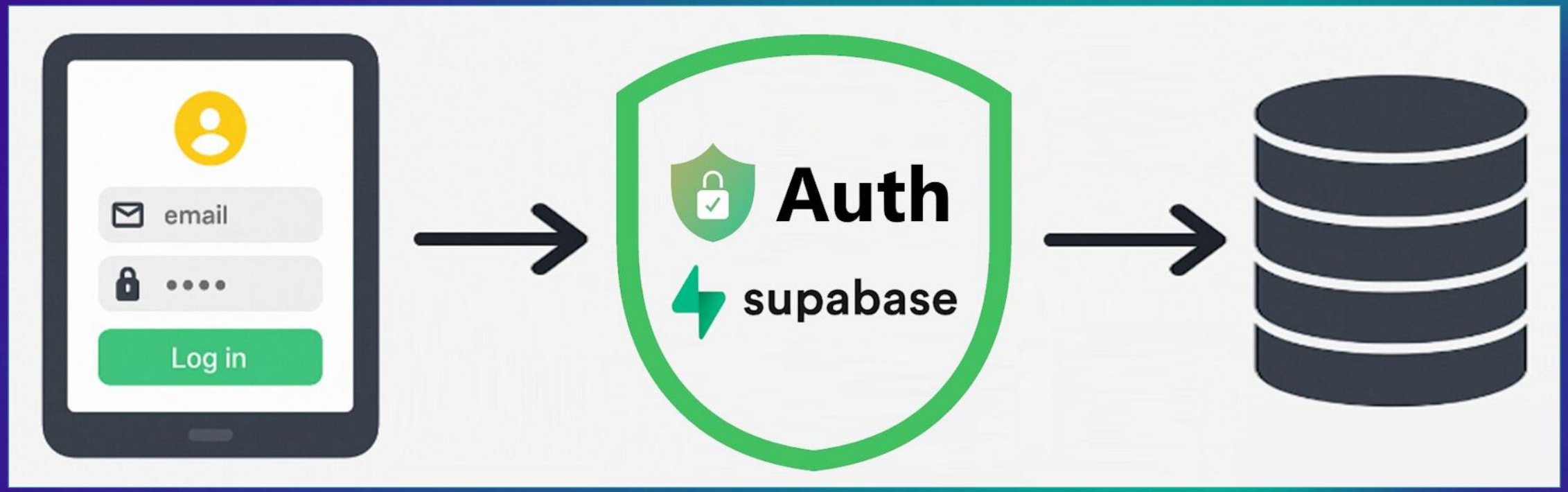


supabase > migrations > 20260203112958_tags_rls_policies.sql

```sql
-- RLS Policies for tags table
-- Anyone can read tags
CREATE POLICY "Anyone can read tags" ON tags
  FOR SELECT USING (true);

-- Authenticated users can create tags
CREATE POLICY "Authenticated users can create tags" ON tags
  FOR INSERT WITH CHECK (auth.uid() IS NOT NULL);

-- No one can update or delete tags (once created, they persist)
-- If you want to allow updates/deletes, uncomment these:
-- CREATE POLICY "Authenticated users can update tags" ON tags
--   FOR UPDATE USING (auth.uid() IS NOT NULL);
-- CREATE POLICY "Authenticated users can delete tags" ON tags
--   FOR DELETE USING (auth.uid() IS NOT NULL);

-- RLS Policies for article_tags table
-- Anyone can read article-tag relationships
CREATE POLICY "Anyone can read article tags" ON article_tags
  FOR SELECT USING (true);

-- Users can only add tags to their own articles
CREATE POLICY "Users can tag their own articles" ON article_tags
  FOR INSERT WITH CHECK (
    EXISTS (
      SELECT 1 FROM articles
      WHERE articles.id = article_id
      AND articles.owner_id = auth.uid()
    )
  );
```

Users and Roles

Database Apps with Users and Roles. Visitors, Regulars Users, Admins



Users and Roles

- **Access control** in most apps works with **users** and **roles**
  - Simple apps use hard-coded **roles** (e. g. **user** / **admin**)
  - Complex apps assign **permissions** for each role (e. g. "*view payments*" permission, "*edit payments*" permission)
- Typical access control policy:
  - **Anonymous** users: can **read all public data**, e. g. articles
  - **Authenticated users**: can read / write their **own data**, e. g. own articles, own messages, own user profile
  - **Admin users**: can read / write **any data** in the DB

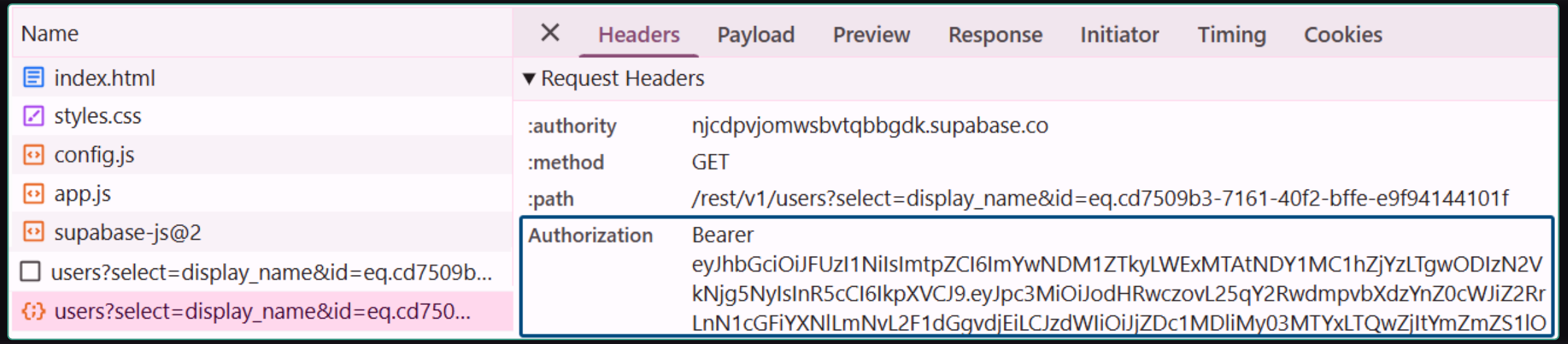
JWT Tokens

SoftUni AI

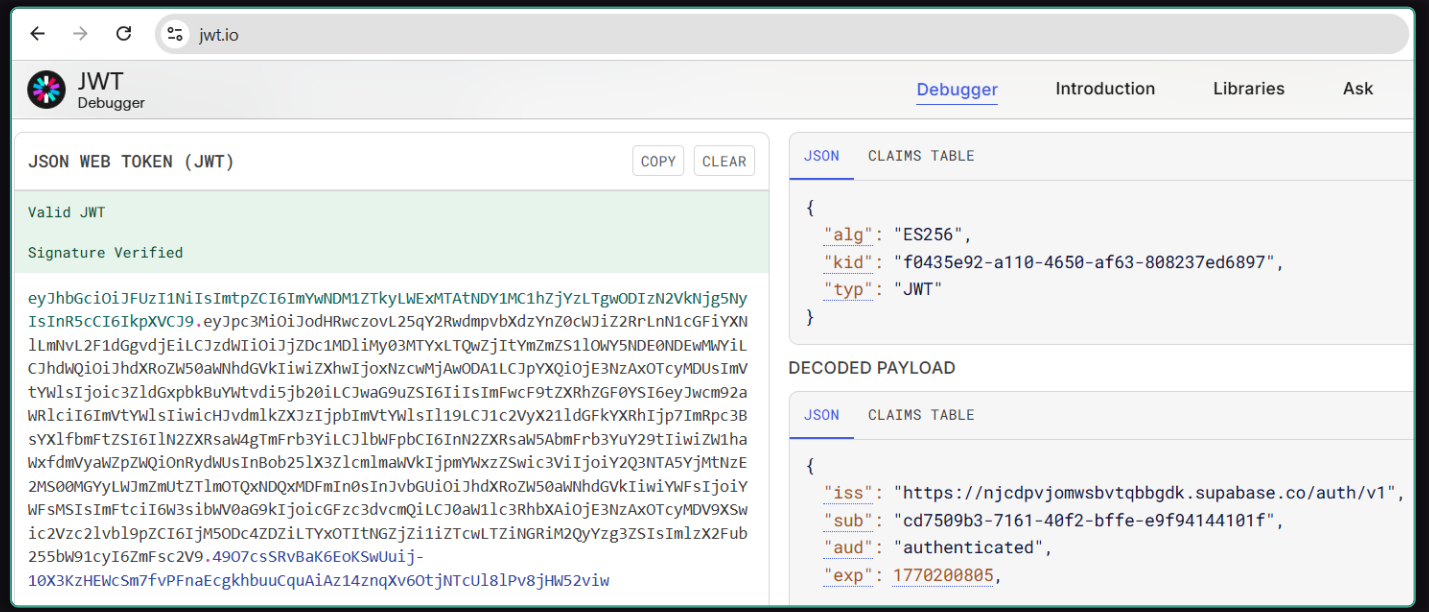
- **JWT** (JSON Web Token) is modern **authentication** and **authorization** standard
  - Holds info about the **currently logged-in user**: **user_id** / **username**, **roles** and **permissions** (optional)
  - Holds a **digital signature**, created by the token issuer
- The **JWT token** is:
  - **Issued at login** (after credentials are verified)
  - **Sent with each request** to identify and authorize the user
  - **Removed** from the client at **logout** (to forget logged-in user)

Inside JWT Tokens

SoftUni AI



header	payload	signature
{ "alg": "ES256", "typ": "JWT" }	{ "sub": "jonny86", "iat": 1516239022 }	KMUFsIDTnF myG3nMiG M6H9FNFUR Of3wh7S...



Users and Roles in Supabase

SoftUni AI

- **Supabase Auth** have built-in **user management** (register, login, external login, magic link login, JWT tokens, logout)
  - **Supabase Auth** does not provide **user roles** (e. g. admin)
  - Developers should implement **roles** by **custom code**
- Several **approaches** to define user roles in Supabase:
  - Use **auth.users.app_metadata** → simple, but can't edit roles after signup (only with edge functions)
  - Use **user_roles** table + **RLS joins** → easy and correct
  - Use **user_roles** table + **RLS** + **Auth hooks** → better performance

Users and Roles in Supabase

SoftUni AI

- Recommended approach to **implement 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**
    - **Everyone** can **read public app data**
    - Only **owners** and **admins** can **edit app data**

Example: Blog with Admins

- We already have a **blog system**:
  - **Visitors** list and **view** blog articles
  - **Users** log-in, **create** / **edit** their **own articles**

- Now we want to **implement admins**:
  - **Admins** will have **unrestricted read / write access** to all app data
  - View / edit / delete user's articles
  - **Manage users**: view / create / edit / delete users

Implementing Blog Admins in DB

SoftUni AI

- **Step 1**: Define **users and roles** in the Supabase DB

Implement **user roles** in Supabase:
- Enum: **app_role** (**user** | **admin**)
- Table: **user_roles** (**user_id**, **user_role**)

user_id uuid	user_role app_role
311ab519-1500-4023-b9...	admin
cd7509b3-7161-40f2-bff...	admin

Define **RLS policies**:
- Everyone can **read user_roles**
- Database function **is_admin()**
- Only **admins** can **write user_roles**
- **Everyone** can **read** public app data
- Only **owners** and **admins** can **edit** app data

Implementing Admin Panel

SoftUni AI

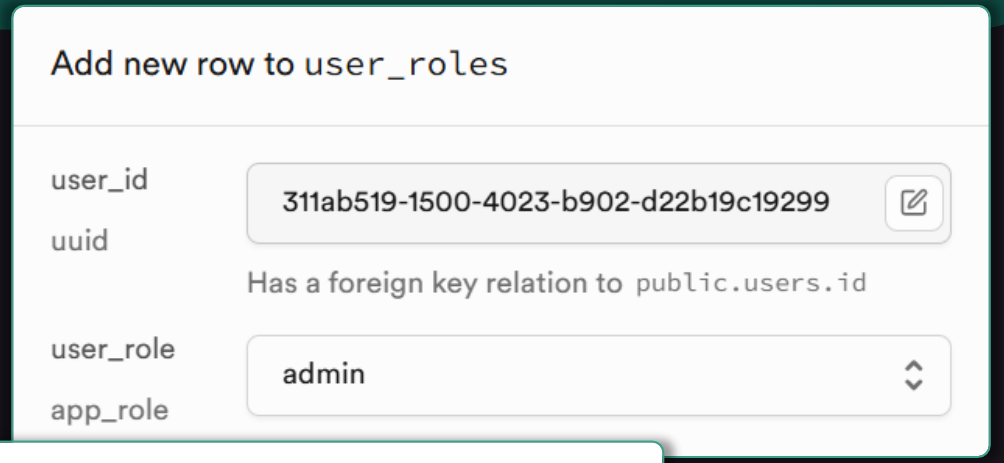
- **Step 2**: Implement the Admin Panel

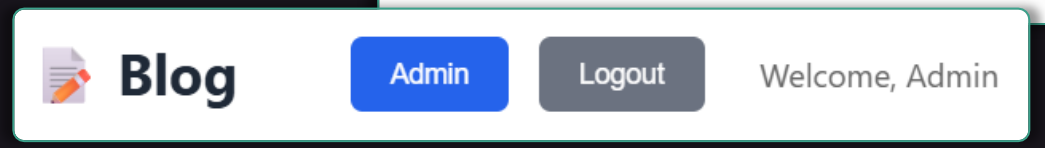
Implement the app **Admin Panel**:
- Use separate files: **admin.html**, **admin.css**, **admin.js**
- The admin panel should be available **only for admins**
  - Admins should have **[Admin]** button in the app header
- Two tabs: **Article Admin** | **User Admin**
- **Article Admin** tab: view all articles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

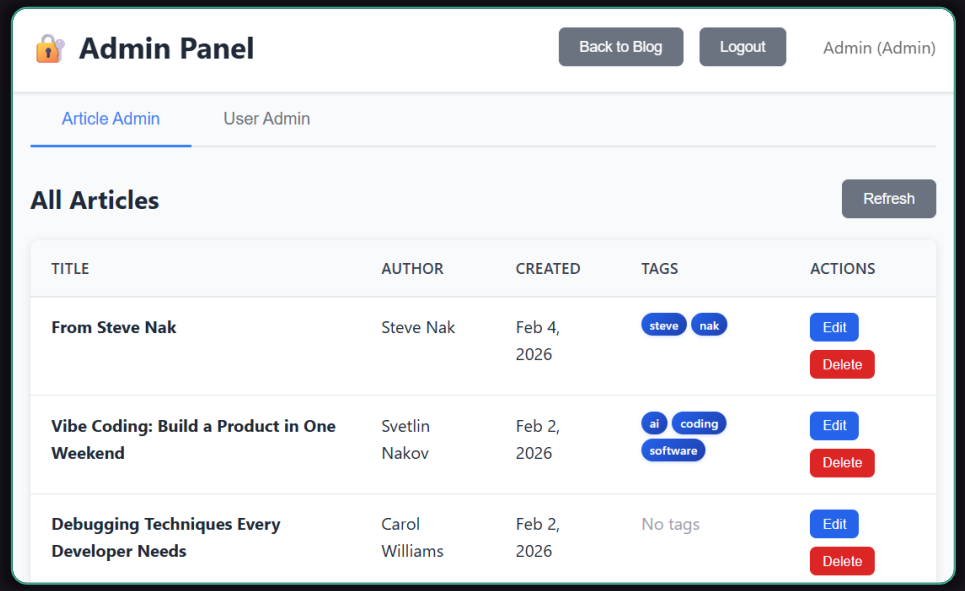
The Admin Panel in Action

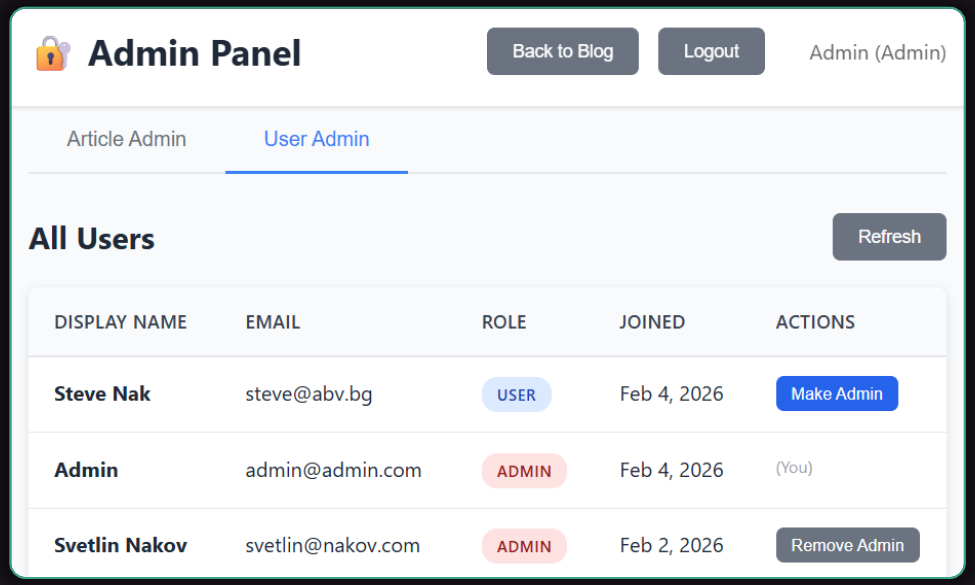
SoftUni AI

- First, **create an admin** from Supabase Studio
- Next, **login** with the admin user
- Open the **Admin Panel**



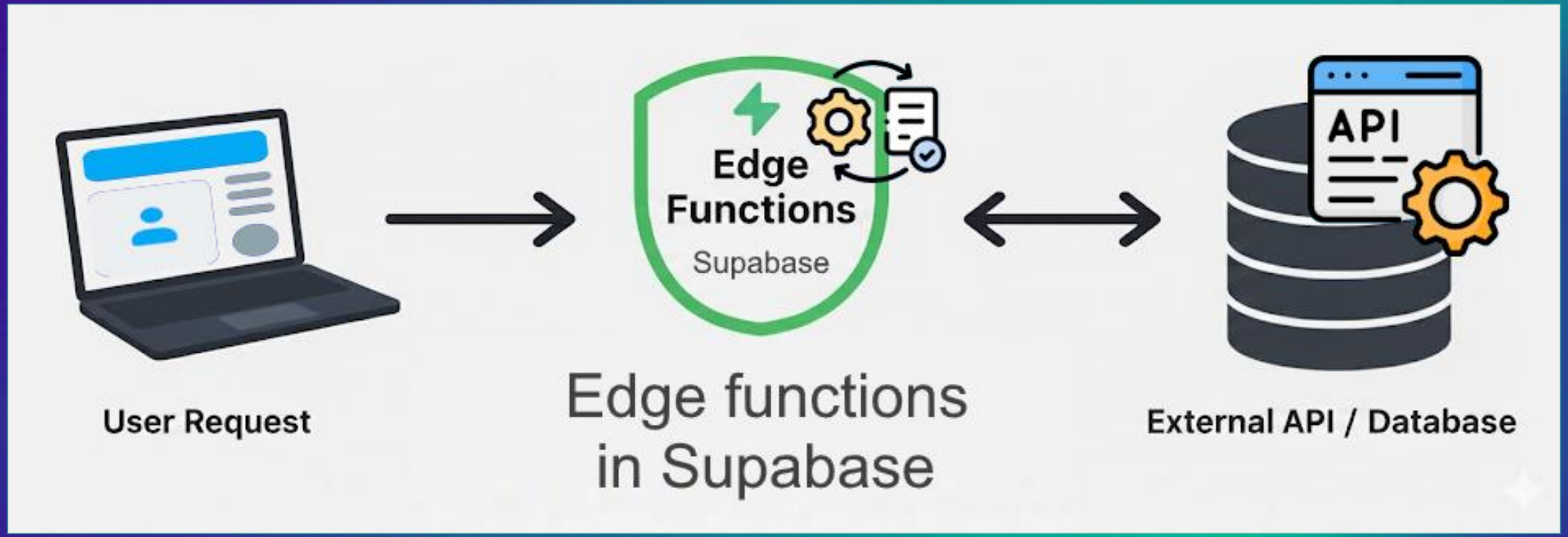






Edge Functions

JS Logic in the Supabase Backend



Supabase Edge Functions

SoftUni AI

- **Supabase Edge Functions**
  - **Serverless functions**, acting as a lightweight back-end
  - Implemented in **JavaScript**, accessed through the API
  - Implement **payments**, **emails**, **webhooks** (Web callbacks)
  - Have **unrestricted DB access**, skipping the RLS policies
- **When** to use edge functions?
  - DB operations with **elevated privileges**: e. g. delete user
  - Access **external APIs**, e. g. send email, accept a payment, integrate an LLM (like ChatGPT or Gemini)

Serverless Example

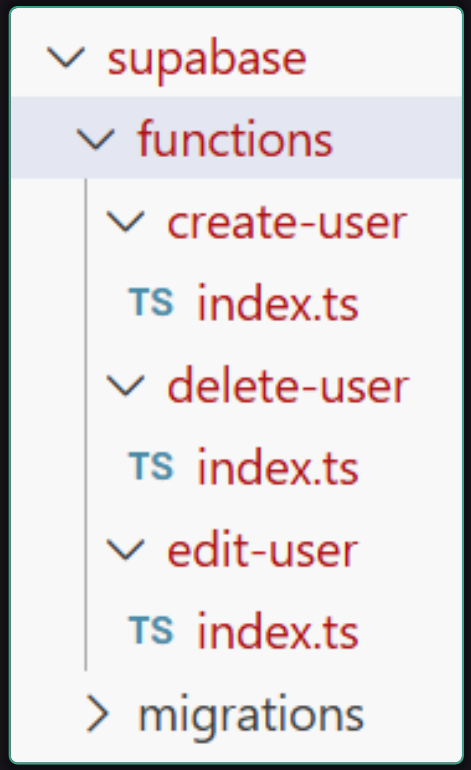
- We already have a **blog system** with **admin panel**
  - **Visitors** read / view blog articles
  - **Users** log-in and **create** articles, **edit** / **delete** own articles
  - **Admins** have **admin panel**: edit articles and manage users
- Now we want to **implement full user management**
  - **Create** / **edit** / **delete** users
  - This requires **elevated privileges** for the DB
  - Can be implemented with **edge functions**

Manage Users with Edge Functions

SoftUni AI

- Implement **full user management** with **edge functions**:

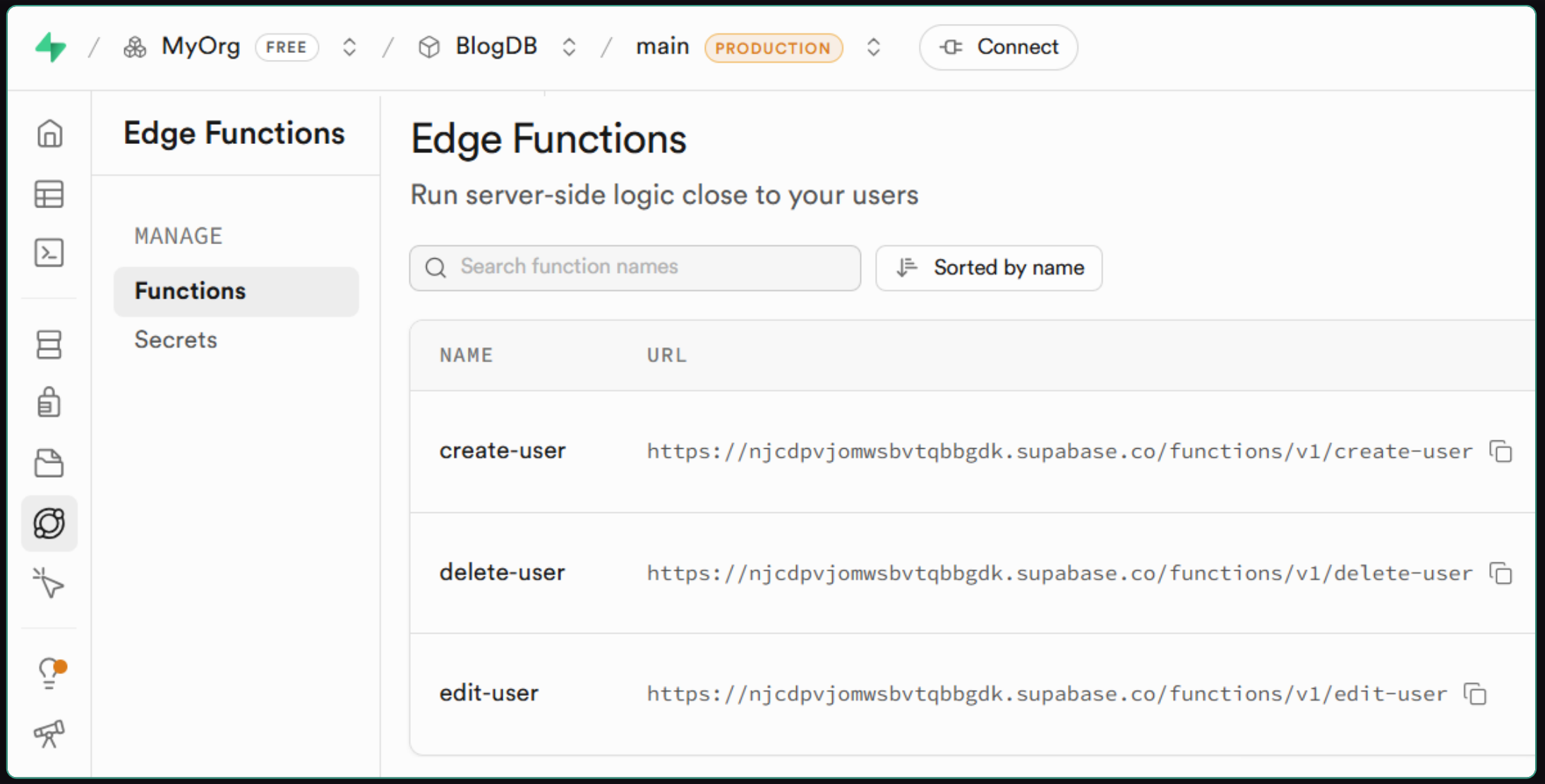
Implement user management in the **Admin Panel**:
- In the **User Admin** table add buttons [**Add**] / [**Edit**] / [**Delete**]
- Implement "Create user", "Edit user", "Delete user" with **edge functions** in the Supabase backend
- Use **popup dialogs** for add / edit / delete confirm
- Choose **user role** when creating / editing users



Deploy the edge functions to Supabase

Deployed Edge Functions

SoftUni AI



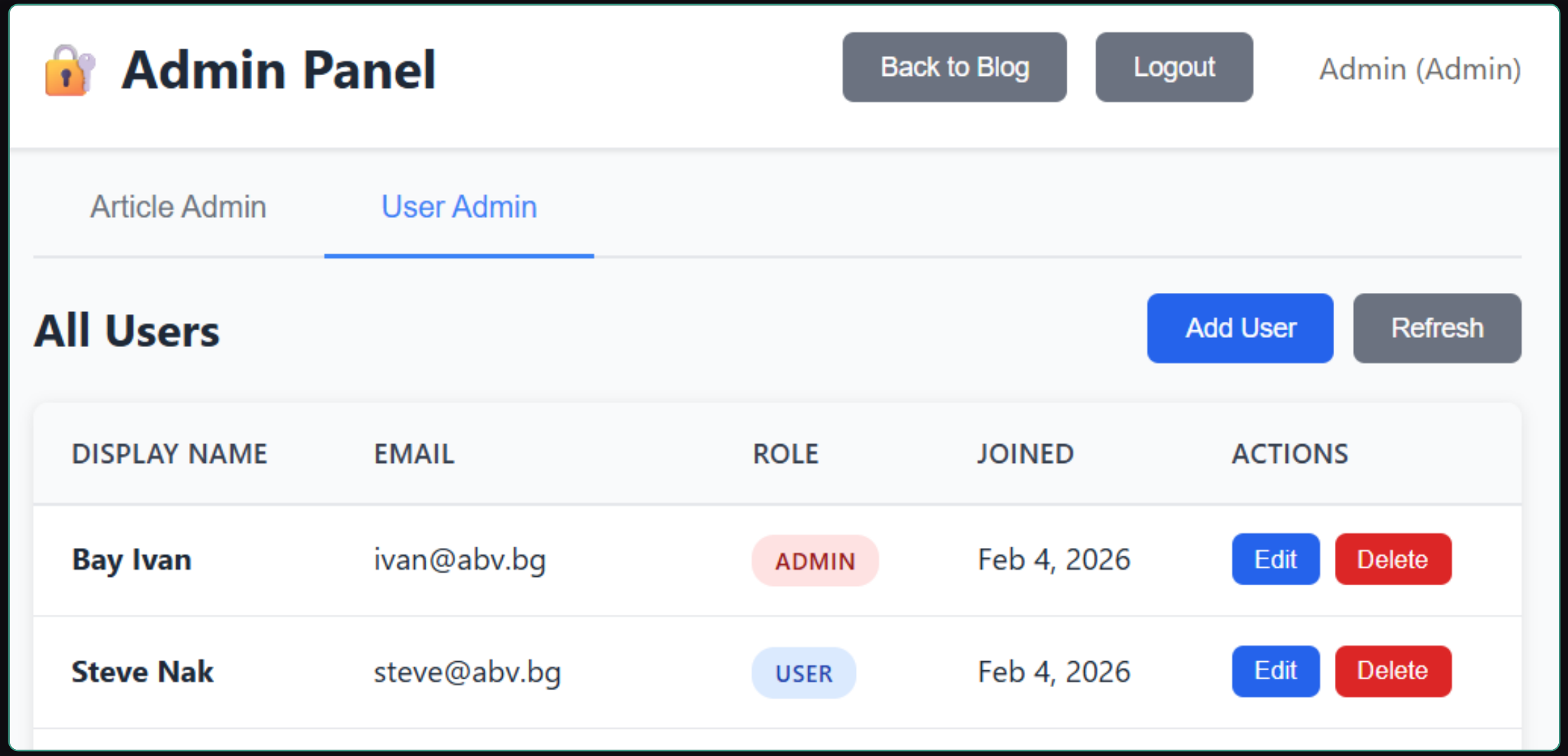
CORS Problems

- Your app may encounter the following problem:

- The edge functions **do not handle CORS** correctly → ask the AI agent to fix and re-deploy them

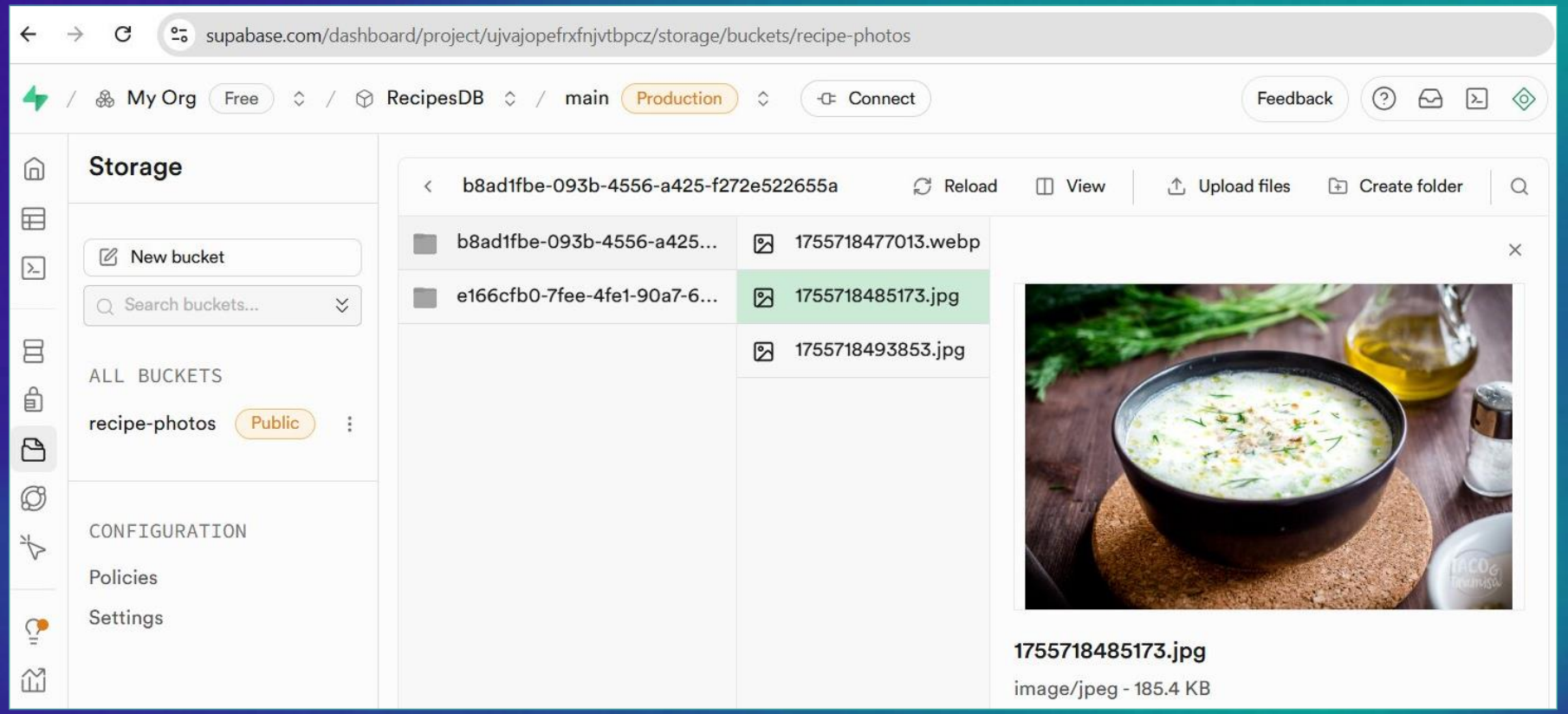
Admin Manage Users in Action

SoftUni AI



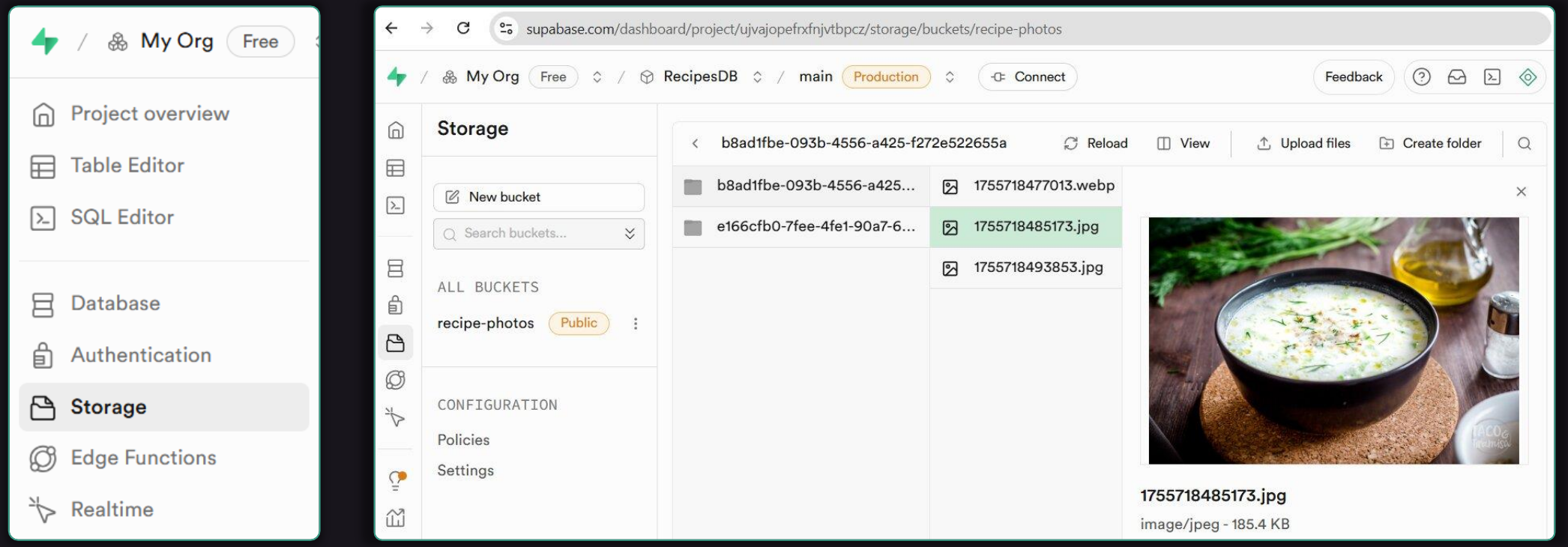
Supabase Storage

Upload and Download Images and Files



Photos and File Uploads

- Apps typically store **photos**, **images**, **documents**, **files** in the **back-end** or in a **cloud storage**
  - In Supabase images are stored in **Supabase Storage**



Supabase Storage

SoftUni AI

- **Supabase Storage** implements server-side **file uploads**, storage and **downloads**
  - Store images, photos, documents, videos, etc.
  - Storage structure:
    - **Buckets** (root folders, which hold files)
    - **Files** (with file paths, imitating **folders**)
  - Functionality: file **upload**, **download**, **share**
  - Access control: tables with **RLS** in the `**storage**` schema

Implementing Blog Images

SoftUni AI

- Implement **blog images** with **Supabase Storage**
  - We already have the **blog system**, which allows users to view and publish blog article
  - Now we shall **implement cover image** for the articles

Implement **cover images** for the blog articles
- Implement **image upload** in Add / Edit articles
- Keep image uploads in the **Supabase Storage**
- Implement proper **RLS policies**: everyone can view images, but only image owners can edit / delete them

Blog with Images in Action

SoftUni AI

Blog

Admin | Logout | Welcome, Svetlin Nakov

Published Articles

Tarator - the full recipe

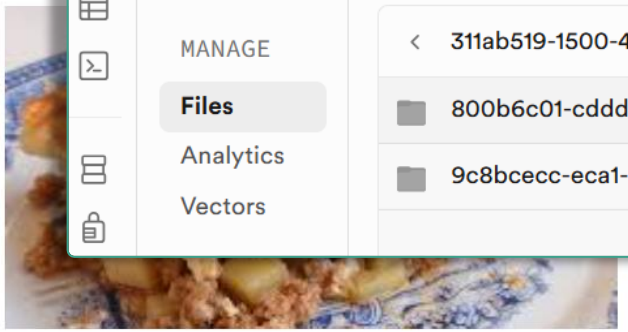
Admin

February 4, 2026

tarator | recipe | food

On days when the thermometers are showing nearly 40 degrees, everyone is looking for a way to escape the heat. Of course, the best solution to the pro...

Read More



Moussaka recipe

Steve Nak

February 4, 2026

food | moussaka | recipe | bulgarian

Ingredients Original recipe (1X) yields 12 servings ¾ pound ground beef (85% lean) ¾ pound ground pork ½ cup olive oil, divided 1 large carrot, f...

Read More

MyOrg FREE / BlogDB / main PRODUCTION | Connect | Feedback

Storage

MANAGE

Files

Analytics

Vectors

Files > Buckets > article-images PUBLIC

311ab519-1500-4023-b902-d22b19c19299 | Reload | View

800b6c01-cddd-48b9-847...

311ab519-1500-4023-b902-...

1770216403718_tarator4.jpeg

9c8bcecc-eca1-46a9-8c94...

```
async function publishArticle(e) {
    if (imageFile) {
      const timestamp = Date.now();
      const filename = `${articleData.id}/${currentUser.id}/

      // Upload file to storage
      const { error: uploadError } = await supabaseClient
        .storage
        .from('article-images')
        .upload(filename, imageFile);

      if (uploadError) throw uploadError;

      // Get public URL
      const { data: urlData } = supabaseClient
        .storage
        .from('article-images')
        .getPublicUrl(filename);
```

Lesson Summary

SoftUni AI

- **Supabase Auth** has built-in authentication system: users, login / logout, JWT tokens, RLS security, OAuth
- **Database migrations**: SQL scripts for DB schema changes, ordered by time, tracked in both DB and local files
- **Row-Level Security (RLS)**: defining policies for row-level access control (for SELECT, INSERT, UPDATE, DELETE)
- Implementing app **users** and **roles** (visitors, users, admins) in Supabase with **user_roles** table + **RLS joins**
- **Supabase Storage**: implementing file uploads / downloads

Questions?

Postbank – Exclusive Partner for SoftUni AI

- One of the leading **banking institutions** in Bulgaria
- Member of the Eurobank Group with € 103 billion assets
- Innovative trendsetter with next generation **beyond banking**, transforming today, empowering tomorrow
- Certified **Top Employer 2025** by the international Top Employers Institute
- Proven people care and **wellbeing initiatives**
- Benefits and unlimited access to professional, **personal and leadership trainings and programs**
- www.postbank.bg / careers.postbank.bg

Diamond Partners of Software University

Software University

Coca-Cola HBC Bulgaria

Postbank

top EMPLOYER България Bulgaria 2025

FOR A BETTER WORLD OF WORK

SUPER HOSTING .BG

encorp.ai

createX

DRAFT KINGS

THE CROWN IS YOURS

INDEAVR

Serving the high achievers

Diamond Partners of SoftUni Digital

SoftUni Digital

Coca-Cola HBC Bulgaria

Postbank

top EMPLOYER България Bulgaria 2025

FOR A BETTER WORLD OF WORK

SUPER HOSTING .BG

encorp.ai

createX

INDEAVR

Serving the high achievers

DRAFT KINGS

THE CROWN IS YOURS

Diamond Partners of SoftUni Digital

SoftUni Digital

HUMAN

NETPEAK
DIGITAL GROWTH PARTNER

MarmaLAB | E-комерс агенция

ETIENYANEV
Break Your *Limits*. Live Your *Brand*.

1 FOR FIT

Aleikov studio

Diamond Partners of SoftUni Creative

SoftUni Creative

Coca-Cola HBC Bulgaria

Postbank

top EMPLOYER България Bulgaria 2025

FOR A BETTER WORLD OF WORK

SUPER HOSTING .BG

encorp.ai

createX

DRAFT KINGS

THE CROWN IS YOURS

INDEAVR

Serving the high achievers

Organization Partners of SoftUni Creative

SoftUni Creative

creative corner

Studio π

Think Big • Design Smart

THE BUCKS TOWN'S WORK

DEFFECTO