2-Hour Supabase Mini–Project – Custom Note Service

# schema.sql

CREATE TABLE notes (  
 id uuid PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id uuid REFERENCES auth.users ON DELETE CASCADE,  
 title text NOT NULL,  
 content text NOT NULL,  
 created\_at timestamptz DEFAULT now()  
);

# functions/post\_notes.js

// Why: POST is used to create a new note, path is /notes for clarity, params read from JSON body.  
  
import { serve } from 'https://deno.land/std@0.168.0/http/server.ts'  
import { createClient } from 'https://esm.sh/@supabase/supabase-js'  
  
serve(async (req) => {  
 const supabase = createClient(  
 Deno.env.get('SUPABASE\_URL') ?? '',  
 Deno.env.get('SUPABASE\_ANON\_KEY') ?? '',  
 { global: { headers: { Authorization: req.headers.get('Authorization')! } } }  
 )  
  
 const { title, content } = await req.json()  
 const {  
 data: { user }  
 } = await supabase.auth.getUser()  
  
 const { data, error } = await supabase.from('notes').insert([  
 { user\_id: user?.id, title, content }  
 ])  
  
 if (error) return new Response(JSON.stringify({ error }), { status: 400 })  
  
 return new Response(JSON.stringify(data), { headers: { 'Content-Type': 'application/json' } })  
})

# functions/get\_notes.js

// Why: GET is used for retrieval, path is /notes for listing, user ID is from auth context.  
  
import { serve } from 'https://deno.land/std@0.168.0/http/server.ts'  
import { createClient } from 'https://esm.sh/@supabase/supabase-js'  
  
serve(async (req) => {  
 const supabase = createClient(  
 Deno.env.get('SUPABASE\_URL') ?? '',  
 Deno.env.get('SUPABASE\_ANON\_KEY') ?? '',  
 { global: { headers: { Authorization: req.headers.get('Authorization')! } } }  
 )  
  
 const {  
 data: { user }  
 } = await supabase.auth.getUser()  
  
 const { data, error } = await supabase  
 .from('notes')  
 .select('\*')  
 .eq('user\_id', user?.id)  
 .order('created\_at', { ascending: false })  
  
 if (error) return new Response(JSON.stringify({ error }), { status: 400 })  
  
 return new Response(JSON.stringify(data), { headers: { 'Content-Type': 'application/json' } })  
})

# README.md

# Custom Note Service – Supabase Mini Project  
  
## Setup & Deploy  
  
1. Create a Supabase project at https://supabase.com  
2. In your SQL editor, run `schema.sql` to create the `notes` table.  
3. Enable Edge Functions and deploy:  
  
 supabase functions deploy post\_notes  
 supabase functions deploy get\_notes  
  
4. Set required env vars in your local `.env` or Supabase Function config:  
 - SUPABASE\_URL  
 - SUPABASE\_ANON\_KEY  
  
## Why? (Schema)  
  
- `id` is `uuid` for uniqueness across distributed systems.  
- `user\_id` links to Supabase auth for multi-user support.  
- `title` and `content` are text for flexible note data.  
- `created\_at` helps sort and show recent notes.  
  
## Functions  
  
### post\_notes.js  
// Why: POST is used to create a new note, path is /notes for clarity, params read from JSON body.  
  
### get\_notes.js  
// Why: GET is used for retrieval, path is /notes for listing, user ID is from auth context.

## Demo (with curl commands)  
  
### Create a Note  
  
curl -X POST https://<PROJECT>.functions.supabase.co/post\_notes \  
 -H "Authorization: Bearer <YOUR\_JWT>" \  
 -H "Content-Type: application/json" \  
 -d '{"title": "First Note", "content": "This is a note from curl"}'  
  
Expected Output: JSON Responses  
  
[  
 {  
 "id": "b1f7...",  
 "user\_id": "a3d4...",  
 "title": "First Note",  
 "content": "This is a note from curl",  
 "created\_at": "2025-05-03T12:00:00.000Z"  
 }  
]  
  
### List All Notes  
  
curl -X GET https://<PROJECT>.functions.supabase.co/get\_notes \  
 -H "Authorization: Bearer <YOUR\_JWT>"  
  
Expected Output:  
  
[  
 {  
 "id": "b1f7...",  
 "user\_id": "a3d4...",  
 "title": "First Note",  
 "content": "This is a note from curl",  
 "created\_at": "2025-05-03T12:00:00.000Z"  
 }  
]

# Design Rationale – Key Choices ("Why?")

## Schema Design – Why?

- `id` as UUID with `gen\_random\_uuid()`: Ensures globally unique identifiers that are secure and efficient across distributed systems.  
- `user\_id` referencing `auth.users`: Ties each note to an authenticated Supabase user, enabling per-user access control and data isolation.  
- `title` and `content` as `text`: Allows flexibility in input length and content format.  
- `created\_at` with default `now()`: Automatically timestamps notes for easy sorting and tracking when they were created.  
- `ON DELETE CASCADE`: Ensures that all user-related notes are automatically deleted if the user is removed from the system, preserving data integrity.

## Edge Functions – Why?

✅ post\_notes.js

// Why: POST is used to create a new note, path is /notes for clarity, params read from JSON body.  
- POST method: Standard for creating resources in REST APIs.  
- `/notes` path: Simple and intuitive URL representing a collection of notes.  
- Reading from request body: POST typically carries data in the request body for new resource creation.

✅ get\_notes.js

// Why: GET is used for retrieval, path is /notes for listing, user ID is from auth context.  
- GET method: Standard for reading/fetching data.  
- `/notes` path: Retrieves all notes; same base path for consistency.  
- Reading user ID from auth context: Securely fetches notes only belonging to the authenticated user using Supabase Auth.