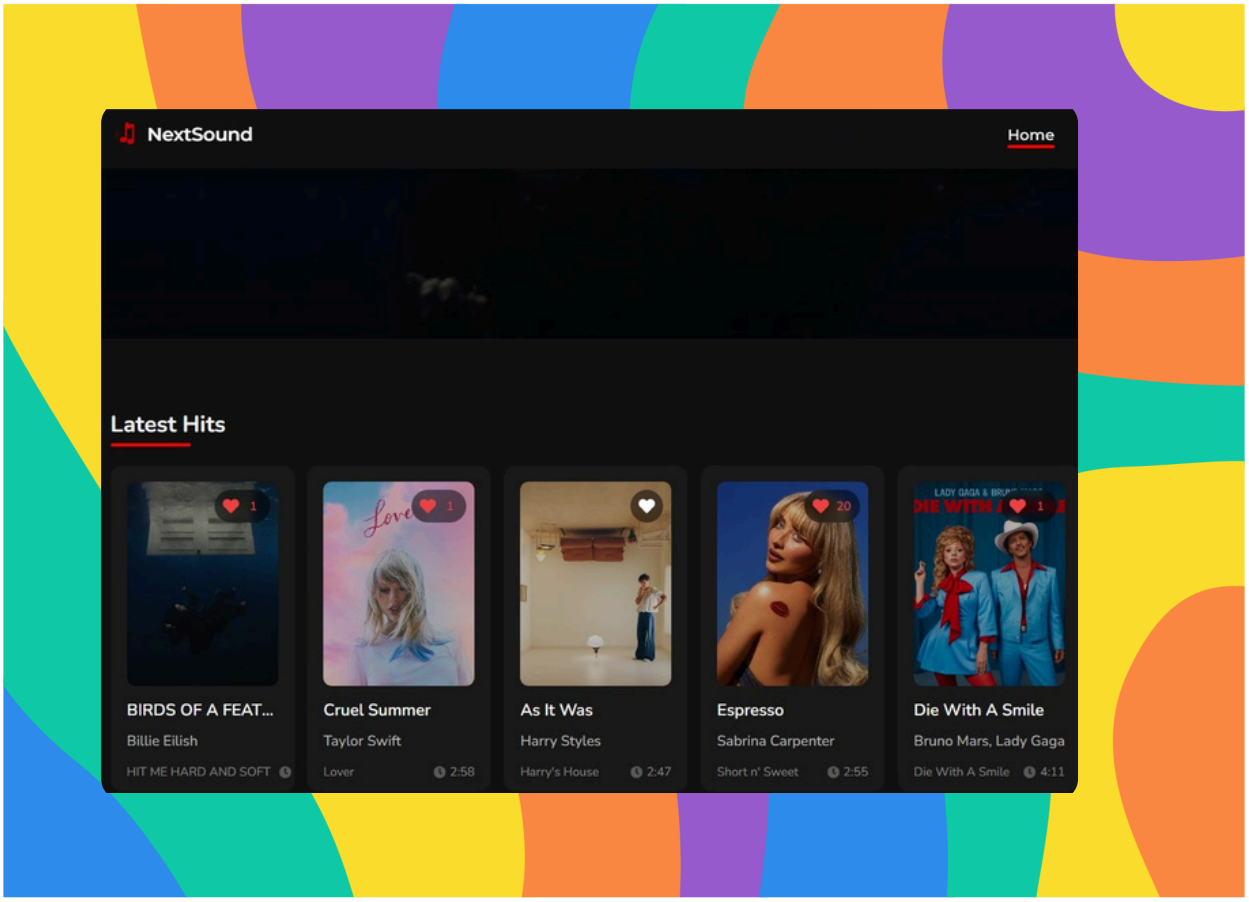
nextwork.org

Build a Database With Cursor and Supabase MCP

Nikhil Bhan



Nikhil Bhan
NextWork Student

Introducing this project!

In this project, I'm going to use Cursor and Supabase MCP to set up and configure a database for a web app called NextSound; NextSound is a music app that allows users to explore and discover the latest and trending music around the world. I'm doing this project to learn what an MCP is, how it works and how to set it up. I'll learn what Supabase is and how to configure a database for the NextSound web app.

Key tools and concepts

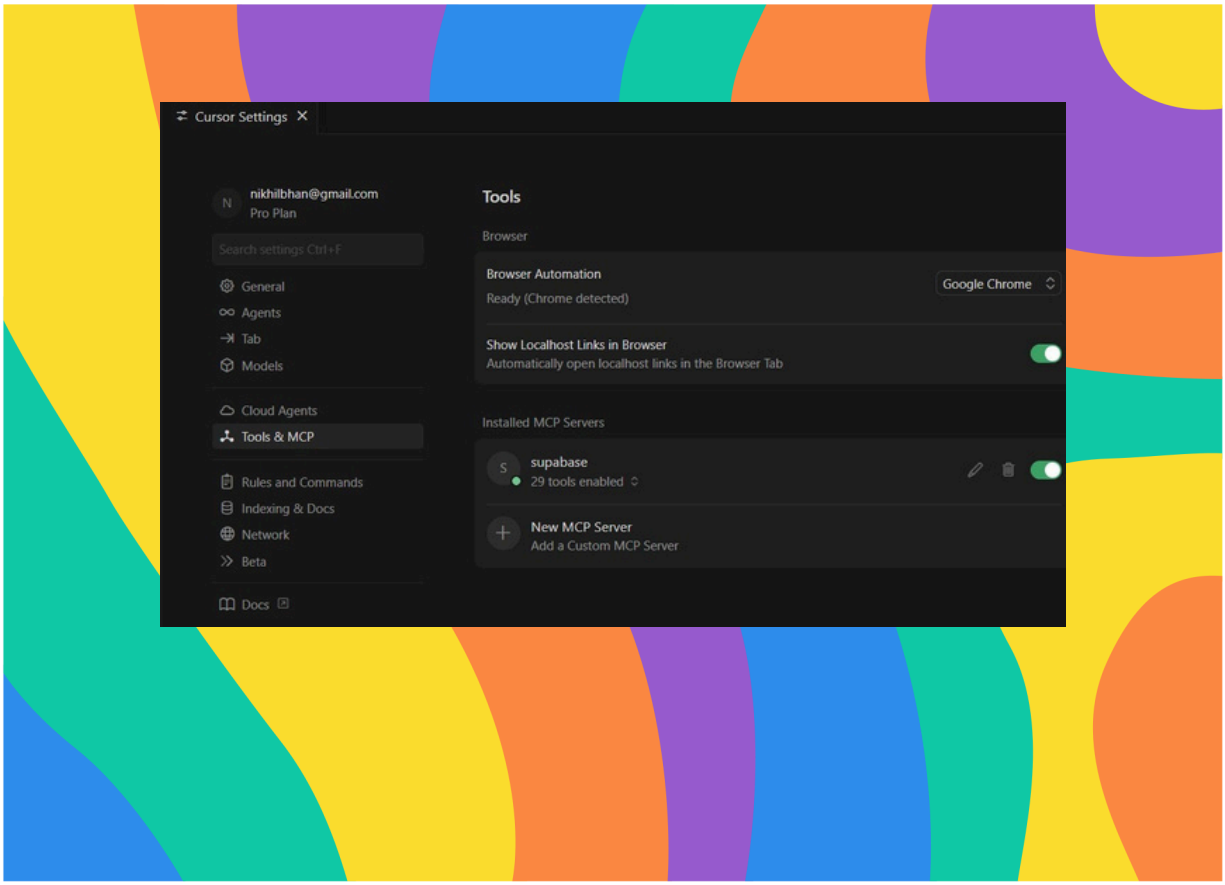
The key tools I used include Cursor, Supabase and MCPs. Key concepts I learnt include how to set up a Supabase account, Organization and Project, how to set up an MCP, understand what database schemas are, user authentication, SQL and performing MCP tool calls. The most important thing I learnt was that MCPs allow me to perform more tasks within an IDE (e.g. Cursor) without having to leave the environment, making it a significantly more powerful tool.

Project completion time

This project took me approximately 2 hours and 30 minutes to complete. I took another 30 minutes to finish my documentation. The most challenging part was understanding the structure of the data and how the 'Likes' are identified for each user. It was most rewarding to see the tracks get 'Liked' and 'Unliked' in the web app, as well as viewing these updates in the Supabase tables. I want to improve my skills in MCPs by using the AWS Security MCP that will allow me to perform security analysis in my AWS environment.

Nikhil Bhan
NextWork Student

Setting up Cursor and MCPs



Why MCPs?

MCP stands for Model Context Protocol and it functions as a universal translator between AI agents and external tools. It helps AI agents establish a connection to external tools and perform a variety of actions the tools provide. These tools optimize the productivity of the AI Agents. Without MCPs, AI agents wouldn't be as productive and they would be unable to convert user prompts into actions.

I connected Cursor to the Supabase MCP. I was able to verify the connection by sending a prompt from the AI Pane in Cursor. The prompt I sent was 'What is my Supabase Organization called?'.

After sending it, the Supabase MCP allowed Cursor to run the 'list_organizations' tool call. It provided Cursor access to the Organizations in Supabase and the response to my prompt was the organization I had created, which is 'NextWork MCP Project - NextSound'. Cursor also has access to other tool calls as well, such as 'list_projects', 'list_branches' and 'list_tables'.

Nikhil Bhan
NextWork Student

Building with Cursor

How Cursor set up the database

To set up my database, Cursor used a set of tool calls to perform specific tasks for me. It used the 'get_cost' tool call to perform a safety check to make sure that the action taken won't incur any charges. Next it used the 'confirm_cost', which asked for my confirmation before proceeding; this allowed me to be in control of the spending. The last tool call it used was 'create_project' and this creates a new project in my Supabase Organization. The project includes parameters such as the Project URL and API Key, which I can save in an environment file.

Understanding .env files

An environment file (.env) stores sensitive information like database passwords, API keys, tokens and other credentials. For NextSound, my .env file contains the 'VITE_SUPABASE_URL' and the 'VITE_SUPABASE_ANON_KEY' parameters. These parameters are important because they will allow the NextSound web app to have access to my database in Supabase.

Implementing the ‘Likes’ feature

The feature I'm building is a 'Likes' feature for the NextSound web app; the feature will allow users to select their favorite music tracks. Instead of manually setting up the database, I'm going to send natural language prompts to Cursor so it can configure my Supabase database for me.

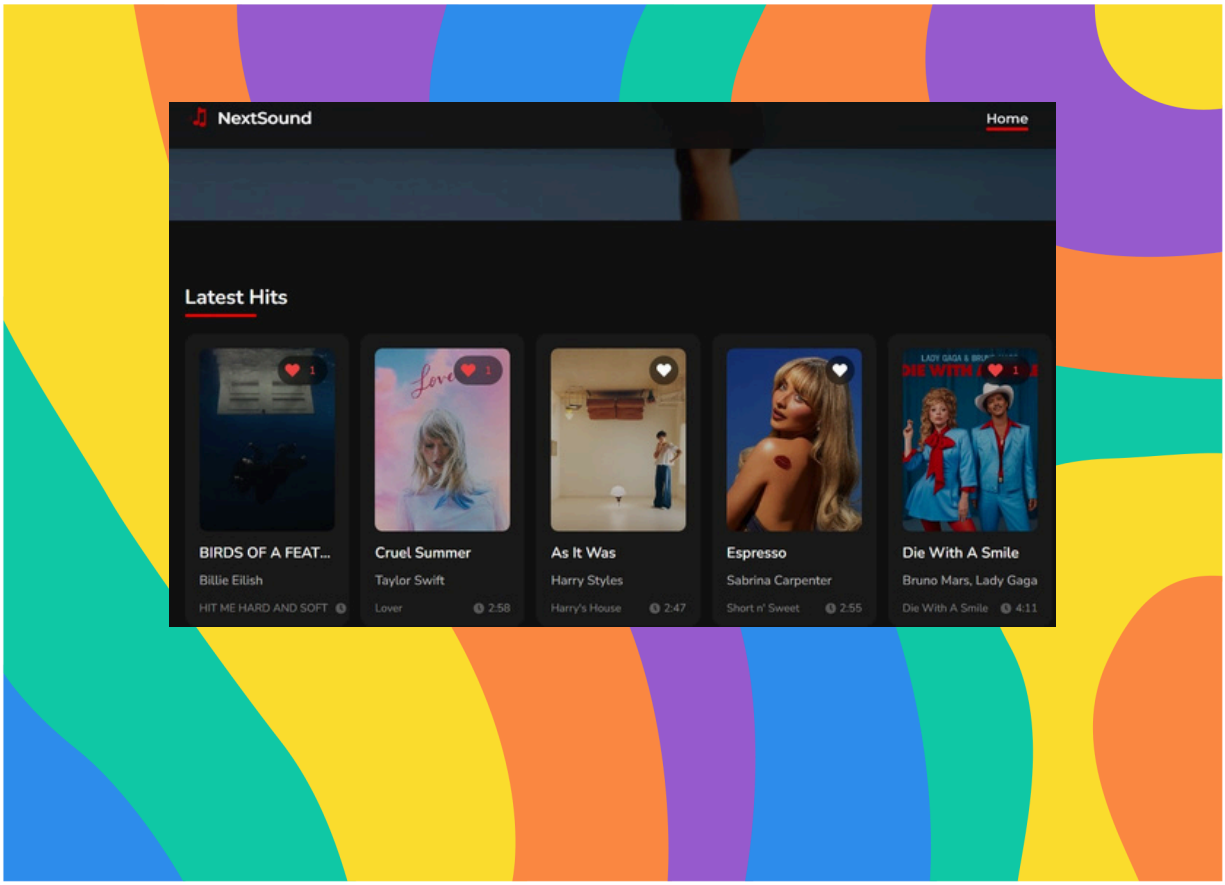
Implementation Process

To implement the feature, Cursor analyzes the structure of my web app's codebase. Next it analyzes the structure of my Supabase project to verify if there's any existing tables. There weren't any, so it made a new one for me.

It then creates and runs a service within my codebase that generates the code for the 'Likes' functionality and handles the different states. It also updates the TrackCard component, where it adds the heart button UI on the top right of every track. During the implementation Cursor used MCP tools such as 'list_projects' and 'list_tables' to understand my current setup in Supabase; it uses the 'apply_migration' to create the table.

Nikhil Bhan
NextWork Student

Testing the Likes Feature



How likes connect to the database

When I clicked the heart icon on a specific track, I could see it's color turn from white to red. The change in the heart icon's state verifies that the track was 'Liked'. I was able to see the number of likes for that track increase by 1; the number of likes was indicated on the right side of the heart icon. Behind the scenes, my Supabase database updates the number of likes for the 'Liked' tracks by 1 as well.

Data in the track_likes table

In Supabase, the 'track_likes' table stores details on the tracks that are liked from the web app, such as 'id', 'track_id' and 'likes_count'. I verified that the web app's connection with my database works by selecting the 'Likes' button on a few tracks.

I then checked the 'track_likes' table in Supabase and I saw that new rows were created. I compared a track's 'track_id' value in the database and looked it up in the project codebase to verify that they matched.

One limitation I observed is that the 'track_id' column is not a practical way to know which track is being referenced from the web app when it's selected via the 'Likes' button. I found that I could improve my database by editing the 'track_likes' table and adding a 'track_name' column. This would help in identifying the tracks in the Supabase table.

Nikhil Bhan
NextWork Student

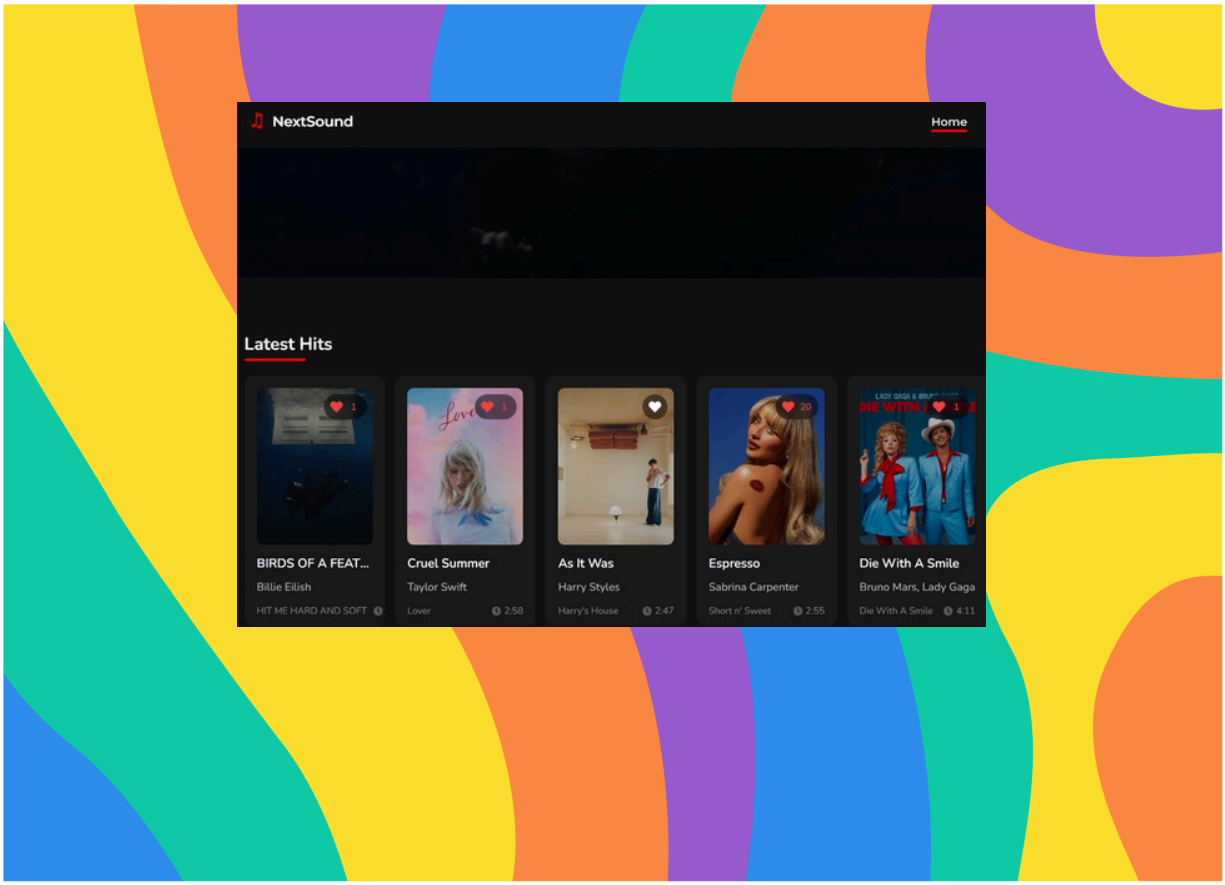
Querying with Natural Language

MCP tools used

When I asked Cursor which tracks were currently liked in NextSound, it used the Supabase MCP to query my 'liked_tracks' table. It used tool calls to query the database, such as 'list_projects', 'list_tables' and 'execute_sql'. The outcome of the results displayed a formatted list of tracks that were 'Liked'. The tracks that were displayed were 'Die With a Smile' by Bruno Mars and Lady Gaga, 'Birds of a Feather' by Billie Eilish and 'Cruel Summer' by Taylor Swift.

Nikhil Bhan
NextWork Student

Updating Votes with Cursor



How Cursor updated the votes

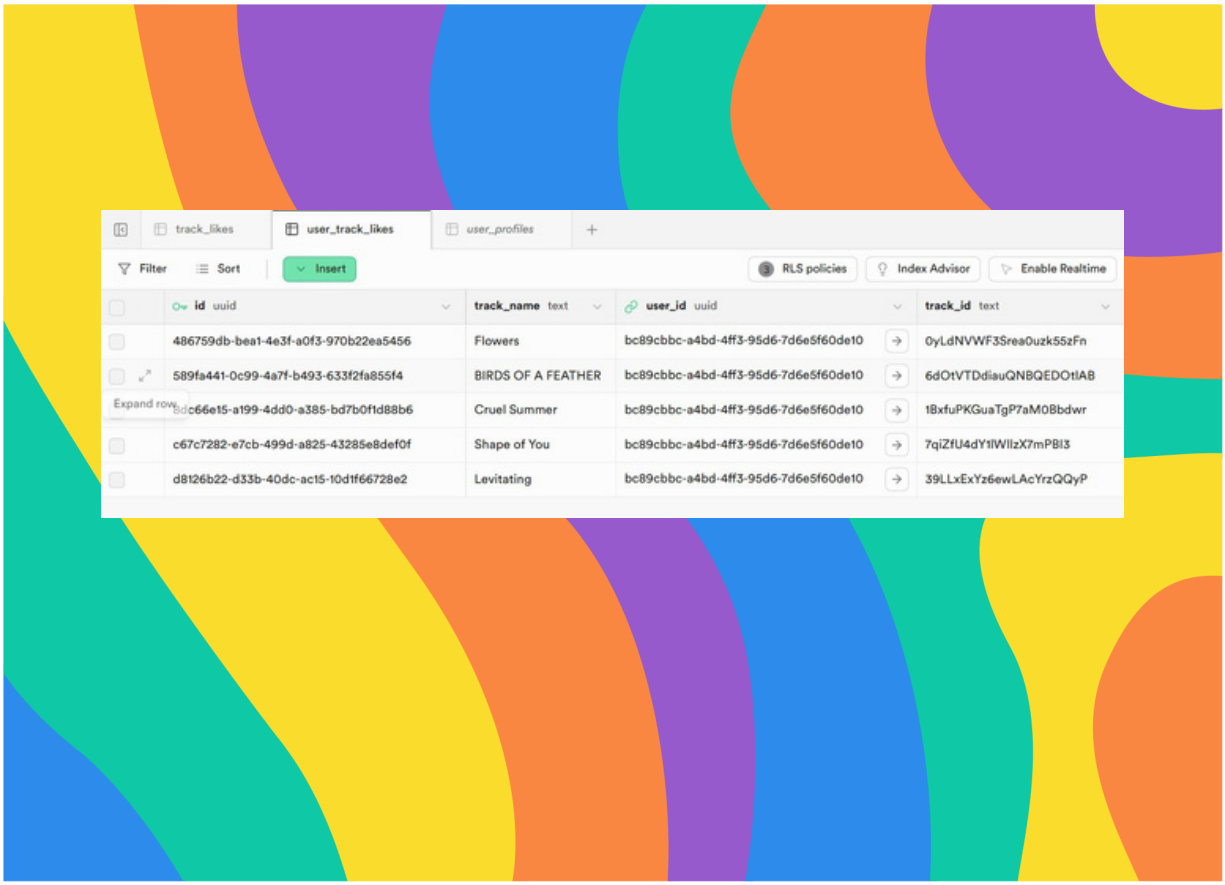
Aside from querying the database, I also used Cursor to update the number of likes for the 'Espresso' track. Cursor used an 'execute_sql' tool call to query the database and it found that the 'Espresso' track had 0 likes in the 'track_likes' table. Another 'execute_sql' tool call is run and this time it inserts a new row to the table, which specifies the 'Espresso' track and sets the value of 'Likes' to 20. I verified that the update was performed successfully by checking the database and the web app after they had a refresh on the data. Both of them displayed the 'Espresso' track and with a total count of 20 'Likes'.

Verifying the update

Being able to update data with natural language is powerful because you can do this while staying within the Cursor environment and you wouldn't have to use CLI commands or access your Supabase account to perform actions. I could use this skill to perform actions such as creating databases and tables, inserting new rows into tables and updating the data in the database much faster.

Nikhil Bhan
NextWork Student

Secret Mission: User Authentication



user_profiles vs user_track_likes

The 'user_profiles' table stores the users that have signed up and have been authenticated on the NextSound web app. The 'user_track_likes' table stores the tracks that have been liked and identifies the users that liked them. Each row contains one specific track that was liked by a user, as well as the name of the track and when it was liked.

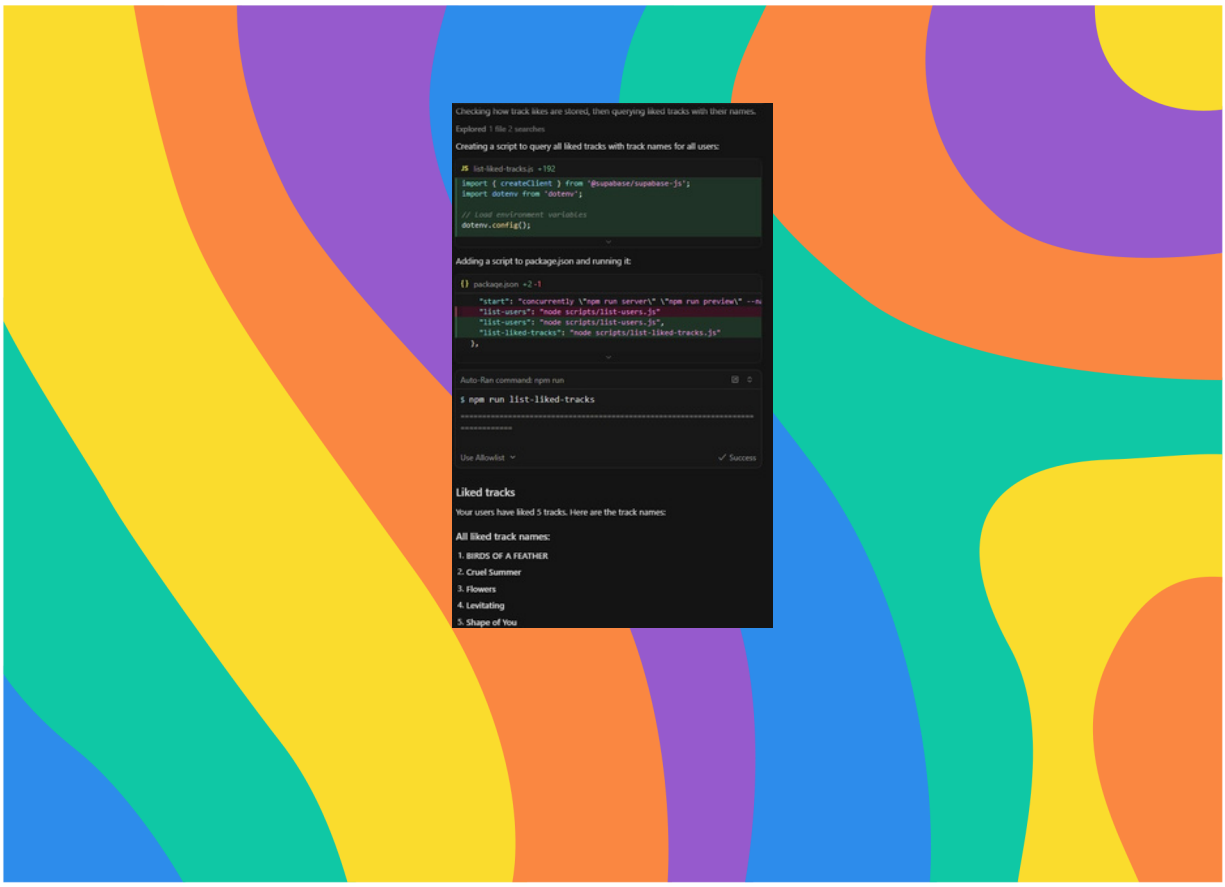
Linking users to tracks

The database matches a user to the tracks they've liked by matching the 'user_track_likes' table with the 'user.auth' table. The 'user.auth' table is located in the 'auth' schema while my tables are in the 'public' schema.

Schemas are used in databases because they provide the structure on how the data is organized. The 'user.auth' table is used to identify users and their unique user IDs. This makes it possible to know which user 'Liked' a specific track.

Nikhil Bhan
NextWork Student

Querying User Data



Benefits of natural language querying

Storing user emails means that I'll be storing sensitive and personal information in my database. One of the ways I reduced security risk in the database was structuring the data; I used the UserID to identify the users instead of using their email. The UserID is unique and each user has one; it helps to make users unidentifiable if any data gets leaked from the database. Passwords are also in the database, but they're not stored as plaintext, which would not be best security practices. Instead, Supabase secures passwords by using 'bcrypt hashing'. It makes it impossible to retrieve the original password and it prevents credentials from becoming compromised.

Other questions to explore

I could also ask Cursor other questions like providing the number of 'Likes' for each track in an artist's specific album, calculating popularity scores for each artist and then analyzing how the scores look over a period of time (e.g. weekly, monthly). I can even ask Cursor to review the data in my codebase and predict how popular future tracks will be.

nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects