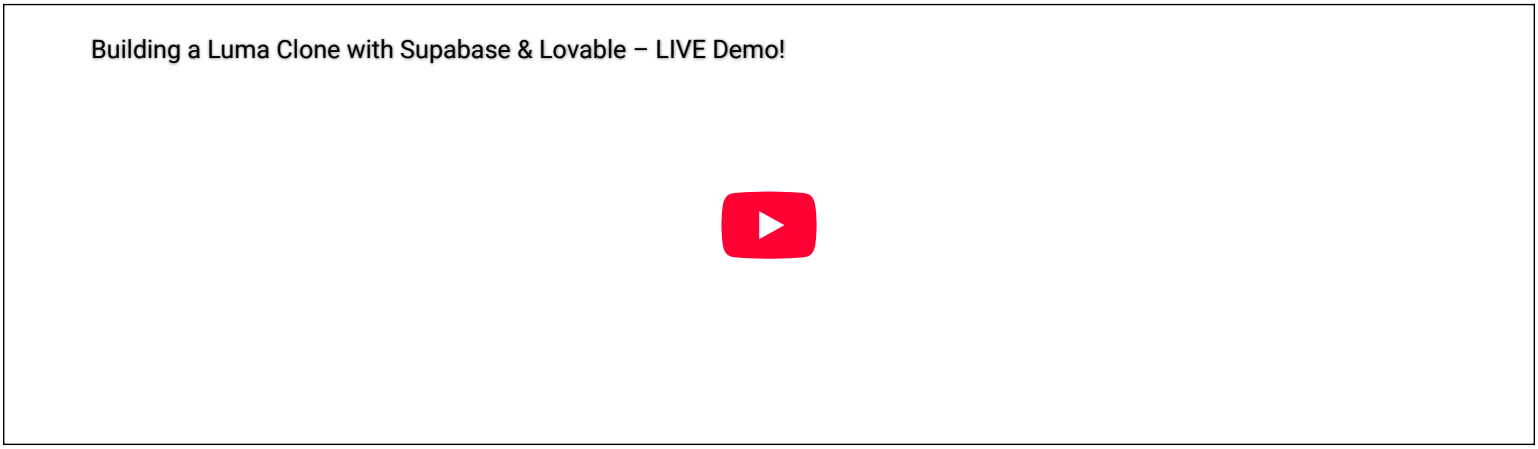
Integrations

Supabase Integration

Integrate a full-featured backend into your Lovable application with Supabase.

Lovable's **native Supabase integration** lets you manage both your front-end UI and your back-end database through a single, easy-to-use chat interface. In other words, you can design your app's screens *and* set up a cloud PostgreSQL database without leaving Lovable. This unified approach makes powerful app development accessible to everyone – non-technical users can rely on Lovable's guidance, while experienced developers can tap into advanced Supabase features as needed.

Overview



Supabase is an open-source alternative to Firebase, providing a hosted PostgreSQL database with real-time capabilities, user authentication, file storage, and serverless functions. By connecting Supabase to your Lovable app, you instantly gain a production-ready backend without writing any boilerplate code or manually configuring servers. Supabase's intuitive web dashboard makes it easy to manage your data and users, and its robust SQL foundation means you retain the full power and scalability of a PostgreSQL database.

Key features unlocked by Supabase integration:

- **Database (PostgreSQL)** – Store and query your app data with full SQL support. Lovable can automatically generate the necessary tables and schema based on your prompts.
- **User Authentication** – Securely manage user sign-ups, logins, and access control. Lovable can add pre-built authentication flows (email/password, etc.) to your app with a simple prompt.
- **File Storage** – Upload and serve images or other files via Supabase Storage. Great for user profile photos, uploads, or any static media your app needs to handle.
- **Real-time Updates** – Supabase can stream live data changes to your app. This enables features like live chat, activity feeds, or collaborative dashboards that update instantly for all users.
- **Edge Functions (Serverless)** – Run custom backend logic (in JavaScript/TypeScript) on Supabase's infrastructure. Lovable will create and deploy these functions for tasks like sending emails, processing payments, or integrating with external APIs.

Why use Lovable's Supabase integration?

With Lovable, you don't have to juggle separate tools for front-end design and back-end setup. By simply conversing with Lovable's AI, you can build your UI *and* have the underlying database and server functions created for you automatically. This means faster development and fewer integration headaches. For example, if you prompt Lovable with "Add a user feedback form and save responses to the database," Lovable will generate the form UI **and** set up a Supabase table to store the feedback – all in one go. This seamless end-to-end generation is Lovable's unique strength, empowering beginners to build complex apps and allowing power users to move faster.

From Zero to Backend – Supabase Introduction & Setup Guide

1. **Create a Supabase account**

   Register a new Supabase account **here** or **sign in** if you already have one.

   How to Connect Your Lovable App to Supabase – Quick & Easy!

2. **Create a new project in Supabase.**

   Click on + New Project, complete the necessary fields, and allow a few minutes for setup.

   Connecting To Supabase

3. **Connect Supabase to Lovable**

   1. **Initiate the Supabase connection in Lovable.**

      In the Lovable editor, open the **Integrations** menu (usually found in the top-right toolbar). Click **Connect Supabase**. Lovable will prompt you to log in to Supabase if you aren't already.

(Lovable's chat will guide you through creating a new Supabase project if needed).

3 **Wait for configuration to complete.**

After selecting the project, Lovable will automatically configure the connection. Within a few seconds, you should see a confirmation message in the chat (e.g. "✅ Supabase connected"). At this point, your Lovable app is now linked to the Supabase database – ready to use authentication, data storage, and other backend features.

Both Lovable and Supabase offer generous free tiers for development. If your app grows or requires advanced features, you may eventually need paid plans for each service separately. For now, you can experiment without incurring costs.

Adding User Authentication

One of the first things you'll likely want is user authentication (so people can sign up and log into your app). Lovable's Supabase integration makes this trivial to set up via chat.

Email and Password (Basic Setup)

Once your project is connected to Supabase, Lovable can generate authentication pages for you. For example, you can simply prompt: **"Add login"**. This will typically create a basic login page (and related signup flow) in your app, wired up to Supabase's authentication system.

After Lovable adds the login UI, you have a couple of ways to create users for testing:

Example prompt:

```
Add login
```

- **Via your app's signup form:** Use the newly added interface in your app to register a user (this will communicate with Supabase to create the account).
- **Via Supabase Dashboard:** Go to your Supabase project's dashboard, navigate to **Authentication > Users**, and manually add a new user (email and password). This is handy to set up a test account quickly.

For development convenience, we recommend disabling email confirmation in Supabase while you test. This way, users can log in immediately without needing to verify an email. To do this, open your Supabase dashboard and go to **Authentication > Settings (or Providers) > Email**, then **disable the "Confirm email" requirement**. You can re-enable confirmations later in production for security.

Social Logins (e.g. Google)

Supabase supports OAuth logins like Google, GitHub, Twitter, and more. You can integrate these into your Lovable app as well:

1. **Enable the provider in Supabase:** In your Supabase dashboard, navigate to **Authentication > Providers**. You'll see a list of external login providers. Toggle on **Google** (for example) and follow the instructions to provide the required OAuth Client ID and Secret from Google. *(Supabase will guide you on how to obtain these credentials from Google's developer console.)* Save the settings – your Supabase project now knows how to handle Google logins.
2. **Update your Lovable app's UI:** Next, you can prompt Lovable to add a social login option. For instance: **"Add a 'Sign in with Google' button to the login page."** Lovable will modify the authentication page, adding a Google sign-in button and the necessary code to initiate the OAuth flow via Supabase.

Once this is done, users of your app will be able to click "Sign in with Google", be redirected to Google for authentication, and then return to your app as logged-in users. You can enable other providers (GitHub, Facebook, etc.) in a similar way – just remember to configure each in Supabase and then adjust your Lovable UI accordingly.

Managing Data with Supabase

Beyond authentication, the core of most apps is reading and writing data. With Lovable and Supabase, you can create database tables and connect them to your UI without leaving the chat prompt.

Creating Database Tables via Lovable

After connecting Supabase, whenever you need to store data persistently, you can instruct Lovable and it will coordinate the database setup for you. The process typically looks like this:

1 **Describe the feature and data you need**

2 **Review the generated SQL snippet**

Lovable's AI will propose updates to your app. Since your request involves storing data, Lovable will also produce a SQL schema snippet – essentially the commands to create the necessary table(s) or columns in Supabase. For example, it might say something like: _"I will create a table `feedback` *with columns* _ `id` , `message` , *and* `rating` " and provide the SQL code for that. This snippet appears in the chat or sidebar for you to copy.

3 **Run the SQL in Supabase**

Open your Supabase dashboard and go to the **SQL Editor**. Paste the SQL snippet from Lovable and execute it, which will create the new table (or alter an existing table) in your database. You can verify in the Supabase **Table Editor** that the table now exists with the specified columns.

4 **Confirm and let Lovable finish integration**

Back in Lovable, confirm that you ran the SQL (the chat might ask for confirmation, or you can simply tell Lovable "done" or click a provided confirmation). Lovable will then finalize the integration: it updates your app's UI to bind form inputs, lists, or other components to the new Supabase table. Continuing our example, the feedback form in your app is now connected—when a user submits feedback, it will be saved into the `feedback` table, and you can retrieve and display those entries as well.

That's it\! You didn't have to manually design a database schema or write any backend code – Lovable and Supabase handled it from your description. You can repeat this process for any new data feature (blog posts, comments, products, etc.).

For example, if you want users to be able to post articles in your app, you could prompt: **"Allow users to create posts with a title and content, and store these in the database."** Lovable would generate a `posts` table (with fields for title, content, author, timestamps, etc.), give you the SQL to add it to Supabase, and then wire up the front-end forms and pages to that table.

Supabase's dashboard offers a rich interface to manage your data. You can view and edit table rows in a spreadsheet-like UI, define relationships between tables, and even import data from CSV or Excel. Under the hood, it's all PostgreSQL – which means you can perform complex queries or use SQL features as needed. Supabase even provides an **AI SQL Assistant** in its SQL editor to help you write queries if you're not familiar with SQL. This can be handy for advanced analysis or troubleshooting.

File Storage (Images & Media)

When your Lovable app needs to handle file uploads (for example, user profile pictures, attachments, or any media), Supabase integration has you covered. Supabase includes a **Storage** service for hosting files (images, videos, PDFs, etc.) conveniently alongside your database.

If you add an **Upload** component or an image upload feature in your Lovable app, Lovable will utilize Supabase Storage behind the scenes. Uploaded files will be stored in a storage bucket within your Supabase project, and you'll get a URL or reference to use for displaying or downloading the file later.

By default, Supabase's free tier allows files up to **50MB** each to be uploaded. This is plenty for most images or short videos. If your app needs to handle larger files, Supabase's paid plans support bigger uploads (including **resumable uploads** for very large files). You can organize files into folders (buckets) and manage access permissions via the Supabase dashboard as needed.

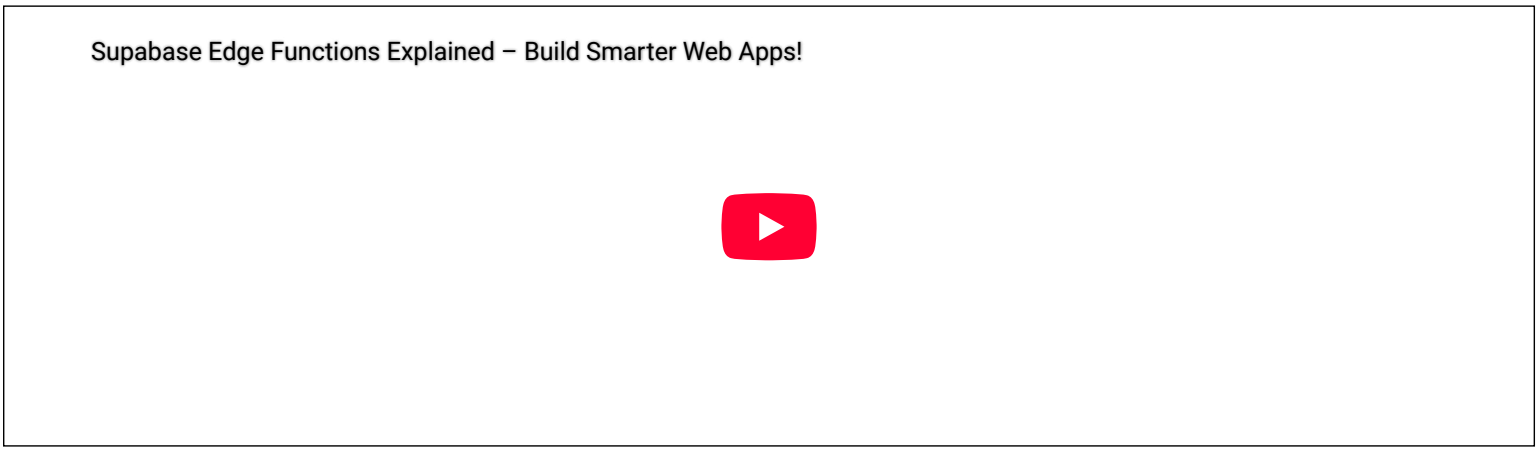
Suppose your app has user profiles and you want users to add a profile picture. You might prompt Lovable: *"Add a profile picture upload to the account settings page."* Lovable will create the UI for uploading an image. Thanks to the Supabase integration, when a user uploads a file, it's saved in your Supabase Storage (in a bucket, e.g. `public/avatar-images`), and Lovable will handle retrieving that image URL to display the profile picture in your app. All of this happens without you writing any storage-handling code.

Lovable automatically detects when a feature requires a secret and prompts you with a UI to input the necessary values.

These secrets are stored securely in Supabase's Edge Function secret manager for your project. They are encrypted and kept safe on the backend. When you deploy Edge Functions (see next section), they can access these secrets to connect with external services.

For example, if you integrate Stripe for payments, you would store your Stripe Secret Key as a secret. When Lovable deploys a payment-processing function to Supabase, it will automatically include that key from the secret store so the function can authenticate with Stripe. This way, you never have to hard-code secrets into your app, and you avoid exposing them publicly.

Backend Logic with Edge Functions



Sometimes your app needs custom backend logic beyond basic data CRUD (Create, Read, Update, Delete). **Supabase Edge Functions** are serverless functions (similar to AWS Lambda) that let you run code on the backend triggered by events or requests. Lovable's integration means you can define desired backend behavior in plain language, and Lovable will write and deploy the necessary Edge Function code to Supabase for you.

Typical use cases for Edge Functions in Lovable include:

- **Using AI services:** e.g. processing some input with OpenAI or Anthropic APIs (with the API key stored as a secret as described above).
- **Sending emails or notifications:** e.g. sending a welcome email when a user signs up, via an email API like Resend.
- **Processing payments:** e.g. creating a checkout session or fulfilling an order using Stripe's API.
- **Scheduled tasks:** e.g. performing a cleanup or summary job every hour/day (Supabase Edge Functions can be triggered on a schedule).
- **Complex computations or third-party integrations:** any code that you don't want to run on the client-side can be done in an Edge Function.

To add a backend function, simply describe what you need in the Lovable chat. For example: *"When a user submits the feedback form, analyze the text using OpenAI and store a sentiment score."* Lovable will generate the code for this logic as a Supabase Edge Function (in this case, calling the OpenAI API) and deploy it to your Supabase project. It will also update your Lovable app to call this function at the right time (e.g., on form submission) and handle the response.

You can find and monitor your Edge Functions in the Supabase dashboard under **Functions**. Each function will have logs that show recent executions and any output or errors. Lovable's Supabase Integration 2.0 improves this experience by automatically reading those logs when something goes wrong – if your function errors out, Lovable will surface the error message in the chat to help you troubleshoot. Of course, you can always check the Supabase logs yourself for more details or for peace of mind.

Before going live: Supabase's default security rules are permissive for development, but you should set up **Row Level Security (RLS)** policies to protect your data in production. RLS allows you to define who can read or write each row in your database tables (for example, ensuring users can only access their own data). Lovable can assist in generating basic RLS policies if you prompt it (for instance, "Apply security policies so users can only edit their own feedback"). However, always review and test these policies in the Supabase dashboard under **Auth > Policies**. Proper security setup is crucial before you invite real users to your app.

It gives your Lovable app a fully managed backend. Without it, Lovable can still build your UI, but you'd have nowhere to persist data or manage users out-of-the-box. With Supabase connected, Lovable can create user accounts (authentication), store and retrieve data in a database, upload files, run server-side code, and more – all automatically. Essentially, Supabase provides the databases and servers behind your app, and Lovable drives it through prompts.

Do I need separate accounts for Lovable and Supabase?

Yes. Lovable and Supabase are two separate platforms. You will need an account on Supabase (to host your database) in addition to your Lovable account. The good news is both have free tiers, so you can get started without any cost. Just remember that if you later upgrade for more usage or features, you'd handle billing for each service individually.

How do I connect Lovable with Supabase?

1. In the Lovable editor, go to the **Integrations** section.
2. Click **Connect to Supabase** and follow the authentication steps.
3. If needed, create a new Supabase project within Lovable.
4. Lovable will automatically generate the necessary database schema and connect it to your project.

Can I integrate my Supabase-connected Lovable app with external automation tools?

Absolutely. When you use Lovable + Supabase, your data lives in Supabase's database and Supabase also provides auto-generated RESTful APIs for your tables (as well as a client library). This means you can use tools like Zapier, **Make.com**, or any other service to interact with your app's backend data via HTTP requests. For example, Zapier could fetch or add records to a Supabase table of your app. Additionally, you can create custom API endpoints using Supabase Edge Functions (which Lovable can help create) to trigger more complex workflows. In short, integrating with third-party automation services is doable – it just might involve a bit of configuration with Supabase's API keys or webhooks.

How scalable is Supabase for when my app grows?

Supabase is built on **PostgreSQL**, which can handle large amounts of data and high traffic. Out of the box, your free database can handle a decent workload (millions of rows, multiple connections). As your needs grow, you can upgrade your Supabase plan for more storage, throughput, and features. Many production apps run entirely on Supabase, so you're in good hands. Just be mindful of the usage limits on the free tier (which Supabase documents on their site) and plan to scale up if you approach those limits.

How can I add real-time features like a chat or live feed to my app?

Supabase has built-in real-time subscriptions on your database. This means your app can listen for changes (inserts, updates, deletes) on specific tables and react instantly. To leverage this, you would design your Lovable app feature as usual (e.g. a chat room that writes messages to a `messages` table). Lovable knows about Supabase's real-time capabilities, so it can set up the front-end to subscribe to that table's changes. In practice, after you've created a table for, say, chat messages, you can prompt Lovable to **"enable real-time updates for the chat"** and it will use Supabase's real-time API under the hood. Users will then see new messages appear live without needing to refresh. This works for any scenario where live updates are useful (comments, notifications, dashboards, etc.).

Is there a limit to file uploads with Supabase storage?

How do I configure authentication in Supabase?

Lovable automatically sets up authentication, but you may need to:

- Go to **Supabase Dashboard > Authentication**.
- Enable **Email Sign-in/Sign-up**.
- Disable email confirmation for easier local testing.

Can I use one Supabase database for multiple Lovable projects?

Yes, you can. You might build multiple front-end applications in Lovable that all connect to the same Supabase project (and thus share the same database and auth). This is advanced, but it's possible – for example, a main app and an admin dashboard as separate Lovable projects using one common database. When connecting Supabase in each project, just select the same Supabase project. Keep in mind all those apps will read/write the same data, so design accordingly.

What if I want to test changes to my database without affecting the live app?

At the moment, each Lovable project connects to one Supabase project, and Lovable doesn't have a built-in staging mode. If you want a safe playground to experiment, Supabase offers a feature called **Branching** which lets you create a temporary copy of your database (like a git branch) to test changes. You could connect a separate Lovable project to a branch or duplicate of your database for testing. In general, for serious projects you'd use caution making schema changes on a live app – perhaps create a backup or use branching, then merge changes when ready. Lovable's integration is evolving, so future updates might introduce more seamless staging workflows.

Does Supabase or Lovable help me write custom SQL or database logic?

Yes. Supabase's web interface includes an **AI SQL Assistant** that can generate SQL queries from natural language. So if you need a complex query or are not comfortable writing SQL, you can try that tool in the Supabase SQL editor. On Lovable's side, you typically don't need to hand-write SQL at all – the AI handles most of the schema creation and queries for you. But if you're an advanced user and want to do something custom, you can always use Lovable's GitHub integration to inspect or edit the code, or run raw SQL on the database as needed.

How do I handle payments in my Lovable app?

Payments are handled via integrations like Stripe, which you can use in conjunction with Supabase Edge Functions. For instance, you might prompt Lovable: "Add a checkout button and process payments with Stripe." Lovable would then create an Edge Function that talks to Stripe's API (using your Stripe secret key stored as a secret) and perhaps store the transaction details in your Supabase database. The UI would be updated to include the checkout/pay button. Essentially, Supabase provides the environment (Edge Functions, database) to implement payments, and Lovable can scaffold the code for you. For more details, you can also refer to the Stripe & Payments integration guide in Lovable's documentation, which covers setting up Stripe specifically.