

MINI PACK

Graduate From Lovable

Move your app to GitHub and keep building in Cursor or Claude Code, without losing work.

For Lovable + GitHub · with Cursor or Claude Code

9 prompts · 4 sections

Welcome! How to Use This Pack

Your Lovable app works, but every new chat edit seems to break something else, and you want to open the code in Cursor or Claude Code. The scary part is the move itself: what if Lovable and your editor fight over the same code, and a week of work disappears?

Here's the good news. Lovable's GitHub sync is two-way. Edits you make in Lovable get pushed to your GitHub repository, and commits you push to the synced branch show up back in Lovable. You don't have to pick one tool forever. You just need a few habits so the two sides never step on each other.

This pack walks you through it: getting your code out safely, understanding what you exported, running it on your own computer, and working in both places without overwriting anything.

Lovable changes often. Every tool-specific tip here comes from Lovable's official docs as of October 2026. If a button or menu looks different for you, check the current docs.

How each prompt is laid out

- **Title:** what the prompt is.
- **Use this when...:** one or two sentences on what the prompt is for and when to reach for it.
- **Fill in:** the blanks to replace before you paste.
- **The prompt:** the box you copy.
- **Tips to customize:** optional ways to change it.

Anything in **[BRACKETS IN CAPS]** is a blank to fill in.

Works best with: Lovable plus GitHub, with Cursor or Claude Code for the work on your computer. Prompts 1.1 and 4.2 only need what you paste, so you can also run them in any AI chat.

The words, in one line each. **GitHub sync** is Lovable's link between your project and a GitHub repository. The **synced branch** is the one branch Lovable reads and writes, usually `main`. A **feature branch** is your own branch for local work. **Cloning** means downloading the repository to your computer so you can edit it there.

Quick Start: Find Your Problem

- **Haven't connected GitHub yet?** Prompt 1.1.
- **Connected, but not sure how branches should work?** Prompt 1.2.
- **Opened the code and have no idea what you're looking at?** Prompt 2.1.
- **Want Cursor or Claude Code to understand the project?** Prompt 2.2.
- **Trying to run the app on your computer for the first time?** Prompt 3.1.
- **Works in Lovable but breaks on your computer?** Prompt 3.2.
- **Switching back and forth between Lovable and your editor?** Prompt 4.1.
- **Lovable says it and GitHub "have diverged"?** Prompt 4.2.
- **Changed a backend function or the database outside Lovable?** Prompt 4.3.

Golden rule: Lovable stays on the synced branch. You work on a feature branch. Never force-push the synced branch.

What's Inside

1. Get Your Code Out Safely (2 prompts)
2. Understand What You Exported (2 prompts)
3. Run It on Your Computer (2 prompts)
4. Work in Both Without Overwriting (3 prompts)

SECTION 1

Get Your Code Out Safely

A few minutes of setup now saves you from the "where did my work go?" moment later.

1.1 My Safe GitHub Connection Checklist

Use this when you're about to connect your Lovable project to GitHub for the first time. It's for getting a short, ordered checklist for your project, so nothing is mid-edit and you have a known-good version to fall back on.

Fill in: [WHAT MY APP DOES] (e.g. "a booking app for a dog groomer"), [WHAT IT USES] (e.g. "Lovable Cloud for the database and login, Stripe for payments"), [WHERE I'LL EDIT LOCALLY] (e.g. "Cursor" or "Claude Code").

```
I'm about to connect my Lovable project to GitHub so I can also work on it in
[WHERE I'LL EDIT LOCALLY]. I'm a beginner with Git.
```

```
My app: [WHAT MY APP DOES]
```

```
It uses: [WHAT IT USES]
```

```
Here's what I know about how Lovable's GitHub sync works:
```

- Connecting creates a brand-new GitHub repository. It can't link to a repository I already have.
- Sync is two-way: Lovable pushes its edits to GitHub, and commits pushed to the synced branch come back into Lovable.
- Lovable only edits and syncs one branch at a time, usually main.
- If I disconnect and reconnect later, Lovable creates a new repository instead of reusing the old one.

```
Please give me a short checklist, in order, for:
```

1. What to do in Lovable before I connect, so nothing is half-finished.
2. Connecting, in plain words.
3. Checking afterward that the code really landed on GitHub.
4. The three mistakes beginners make most often in the first week, and how to avoid each one.

```
Keep it under 250 words. If you're not sure about something specific to Lovable,
tell me to check Lovable's docs instead of guessing.
```

 **Tips to customize:** Before you connect, bookmark a known-good version in Lovable's History panel, so it's one click away if something goes wrong. Lovable's docs suggest this before any big change. Wait for any Lovable edit in progress to finish before you connect or clone. Don't delete the GitHub repository later, because that stops the sync. Renaming it is fine. If you only want a one-time

copy of your code, paid plans can also download the codebase from **Project settings** → **Git** without connecting GitHub at all.

1.2 Set Up My Branches So Nothing Gets Overwritten

Use this when your project is connected to GitHub and you're about to make your first change on your own computer. It's for setting up a simple branch plan that keeps your local work and Lovable's work apart until you're ready to combine them.

Fill in: [MY REPO URL] (from GitHub, e.g. <https://github.com/you/dog-booking>), [SYNCED BRANCH] (shown in Lovable's GitHub settings, usually `main`), [MY FIRST CHANGE] (e.g. "rewrite the booking form validation").

My Lovable project syncs with GitHub. I want to work on it locally without ever overwriting Lovable's work.

Repository: [MY REPO URL]

Lovable's synced branch: [SYNCED BRANCH]

My first local change: [MY FIRST CHANGE]

Please:

1. Give me the exact Git commands to clone the repo and create a feature branch for my first change, with a good branch name. Explain each command in one line.
2. Explain in plain words why Lovable stays on [SYNCED BRANCH] while I work on my feature branch.
3. Show me how to merge my feature branch into [SYNCED BRANCH] when it's done, using a normal merge, not a squash or a rebase.
4. List the commands I must never run on [SYNCED BRANCH] (like force-push), and tell me what to do instead when I want to undo something.

 **Tips to customize:** Lovable's docs say exactly this: keep Lovable on the synced branch, make your local changes on a feature branch, and merge when you're ready. Lovable doesn't see a feature branch until you merge it, so half-finished work can't reach your project by accident. Don't force-push, rebase, or squash commits that are already on the synced branch. To undo a change, add a new commit that reverts it. If you use Claude Code, it can run these Git commands for you, but read each one before you say yes.

SECTION 2

Understand What You Exported

Before you change anything, get a map of what Lovable built.

2.1 Explain My Exported Codebase

Use this when you've just opened your Lovable code in Cursor or Claude Code and don't know where anything lives. It's for getting a plain-English map of the folders, the data calls, and the settings your app needs, without changing a single file.

Fill in: [WHAT MY APP DOES], [THE PART I WANT TO CHANGE FIRST] (e.g. "the booking page").

```
Don't change any files. This repository was built with Lovable, and I'm opening it
in an editor for the first time. I'm a beginner.
```

```
What the app does: [WHAT MY APP DOES]
```

```
The part I want to change first: [THE PART I WANT TO CHANGE FIRST]
```

```
Please read the repository and explain:
```

1. The main folders and what each one is for, in one line each.
 2. Where the pages and components for [THE PART I WANT TO CHANGE FIRST] live.
 3. Every place the app talks to the database, login, or an outside service, with file paths.
 4. Every environment variable the code reads, where it's read, and whether it ends up in the browser. Don't print any secret values.
 5. Anything in supabase/functions/ or a migrations folder, and what each one seems to do.
 6. The commands from package.json to install, run, and build the app.
- ```
If something is unclear, say so instead of guessing.
```



**Tips to customize:** Run this in plan mode (press Shift+Tab in Cursor or Claude Code until it's selected), so the AI looks around and explains before it touches anything. The prompt also says not to change files, as a backup. Lovable's docs point you to the README and [package.json](#) for what to install and which commands start the app. Save the answer as a notes file in your repository. It's the starting point for Prompt 2.2.

## 2.2 Write One Rules File for My Exported Repo

Use this when your exported project has no instructions for your coding AI yet. It's for creating one AGENTS.md file that tells Cursor and Claude Code how your Lovable project works and what they must not break.

**Fill in:** [MY CODEBASE NOTES] (paste the answer from Prompt 2.1), [THINGS THAT MUST KEEP WORKING] (e.g. "bookings save to the database, login with Google, the Stripe checkout").

```
Write an AGENTS.md file for the root of this repository. This project was built in Lovable and still syncs with Lovable through GitHub, so Lovable and I both edit it.
```

```
My codebase notes: [MY CODEBASE NOTES]
```

```
Things that must keep working: [THINGS THAT MUST KEEP WORKING]
```

```
Include short sections for:
```

1. What the app is and the main tech it uses.
  2. How to install, run, and build it (exact commands).
  3. Where things live (pages, components, database calls, backend functions).
  4. Branch rules: never commit directly to the synced branch, never force-push, rebase, or squash it, and work on feature branches.
  5. Backend rules: changes in supabase/functions/ or migration folders aren't deployed automatically, so flag them in your summary.
  6. Never put secrets in variables that start with VITE\_, because those end up in the browser.
  7. The features that must keep working, and how to check each one.
- ```
Keep it under 120 lines. Show me the file before you save it.
```

 **Tips to customize:** Cursor reads an AGENTS.md in your project root. Claude Code reads AGENTS.md too when there's no CLAUDE.md, as long as you're on version 2.1.277 or later. Otherwise, import it from a CLAUDE.md. Claude Code's `/init` command can also draft a CLAUDE.md for you, if you'd rather start that way. Lovable has its own project knowledge (under **Project settings** → **Knowledge**, up to 10,000 characters), so copy the branch rules and the must-keep-working list there too. Our One Rules File pack goes deeper on keeping these files in step.

SECTION 3

Run It on Your Computer

A clone gives you the code, not the services behind it.

3.1 My First Local Run

Use this when you want to start your Lovable app on your own computer for the first time. It's for getting it running step by step, and for knowing what's safe to test before you touch real data.

Fill in: [MY COMPUTER] (e.g. "Mac" or "Windows"), [WHAT'S INSTALLED] (e.g. "Node.js, not sure which version", or "nothing yet"), [ANY ERROR I GOT] (paste it, or write "haven't tried yet").

```
Help me run this Lovable project on my computer for the first time. I'm a beginner.
```

```
My computer: [MY COMPUTER]
What's installed: [WHAT'S INSTALLED]
Any error I got: [ANY ERROR I GOT]
```

Please:

1. Read the README and package.json, and tell me what I need installed and the exact commands to install and start the app.
2. Check which environment variables the app needs, and whether a .env file is already in the repo. Tell me which values I still need and where to find them, without printing any values.
3. Warn me clearly if my local app will read and write the same database as my live app, and tell me how to test safely.
4. If I pasted an error, explain it in one line and give me the fix.
5. Tell me exactly what I should see when it's working.

 **Tips to customize:** Lovable's docs give a big warning here. In a project on Lovable Cloud, the repository's `.env` file points at your real backend, so a record you create or delete on your computer changes real data. Make a test account and use test records while you try things out. A Git branch doesn't give you a separate database. Variables that start with `VITE_` are sent to the browser, so never put a service role key or any other secret in one.

3.2 Works in Lovable, Breaks on My Computer

Use this when the app runs fine in Lovable's preview or the live site, but something fails when you run it locally. It's for checking the usual reasons a clone behaves differently, so you fix the right thing.

Fill in: [WHAT FAILS] (e.g. "Google login sends me to an error page", or "the AI summary button does nothing"), [THE ERROR] (paste it from the browser console or terminal), [WHEN I CREATED THE PROJECT] (e.g. "June 2026", or "not sure").

```
My Lovable app works in Lovable but fails when I run it on my computer.
```

```
What fails: [WHAT FAILS]
```

```
The error: [THE ERROR]
```

```
When I created the project in Lovable: [WHEN I CREATED THE PROJECT]
```

```
Check these possible causes and say "likely", "possible", or "unlikely" for each,
with one line why:
```

1. Login: my login provider only redirects to URLs it knows, and my local URL isn't on the list yet.
2. Secrets: secrets stored in Lovable aren't in the repository, so server code that needs them is missing values on my computer.
3. Connectors and Lovable AI: their credentials are managed by Lovable and can't be copied out.
4. Missing install step or wrong environment variable.

```
Then give me the fix for the most likely cause, step by step. If the fix is "test
this in Lovable's preview instead", say so plainly.
```

 **Tips to customize:** Lovable's docs say newer projects (created from May 13, 2026, built on TanStack Start) run their server code on your computer, so calls through Lovable's connectors and Lovable AI fail locally. Test those in the Lovable preview or your published app. In older React + Vite projects, those calls run in backend functions on Lovable Cloud, so they can still work locally. If you're not sure which kind you have, ask the AI to check the project's files for TanStack Start.

SECTION 4

Work in Both Without Overwriting

Two tools, one codebase. A short routine keeps them friends.

4.1 My Handoff Routine Between Lovable and My Editor

Use this when you're going to switch back and forth between Lovable and Cursor or Claude Code on a regular basis. It's for getting a short routine you follow every time, so both sides always start from the latest code.

Fill in: [HOW I SPLIT THE WORK] (e.g. "UI tweaks in Lovable, logic and bug fixes in Cursor"), [SYNCED BRANCH] (usually `main`).

I edit my app in both Lovable and my own editor. Write me a short handoff routine I can follow every time.

How I split the work: [HOW I SPLIT THE WORK]
Lovable's synced branch: [SYNCED BRANCH]

Facts about Lovable's GitHub sync:

- Lovable shows a sync status card in its project Git settings. "In sync" means both sides are on the same commit.
- Commits on other branches don't appear in Lovable until they're merged into the synced branch.
- After a merge, I should tell Lovable what I changed so its next edit builds on my work.
- A merge doesn't change my published app. I have to publish again in Lovable.

Please write:

1. A "before I leave Lovable" checklist (3 to 5 steps).
 2. A "before I go back to Lovable" checklist (3 to 5 steps), including the git commands to update my local copy and merge my feature branch.
 3. A fill-in message I can paste into Lovable's chat after a merge that tells it which files I changed and what it must keep working.
- Keep each checklist short enough to fit on a sticky note.



Tips to customize: Before you clone or merge, let any Lovable edit finish, then check that the sync status card says the two sides are in sync. Lovable's docs suggest a message like this after a merge: "I changed [files] outside Lovable to [describe the change]. Keep [behavior or constraint] when making further edits." Lovable can edit code you wrote, just like its own code, so put the rules you depend on in project knowledge and cover them with tests. Editing and committing on your computer uses no Lovable credits.

4.2 Rescue Me: Lovable and GitHub Have Diverged

Use this when Lovable's sync status says the two sides "have diverged", a `lovable-sync` branch showed up on GitHub, or you see a notice that a repository push replaced your Lovable work. It's for understanding what happened and getting everything back onto one branch without losing anything.

Fill in: [WHAT LOVABLE SAYS] (copy the exact message from Lovable's Git settings or editor), [WHAT I DID RECENTLY] (e.g. "pushed from Cursor while Lovable was still editing", or "I ran a force-push"), [BRANCHES I SEE ON GITHUB] (e.g. "main, lovable-sync, my-feature").

```
My Lovable project and GitHub got out of step. Help me fix it without losing work.
Don't run any commands yet. Explain first.
```

```
What Lovable says: [WHAT LOVABLE SAYS]
```

```
What I did recently: [WHAT I DID RECENTLY]
```

```
Branches I see on GitHub: [BRANCHES I SEE ON GITHUB]
```

```
What I know from Lovable's docs:
```

- "Diverged" means Lovable and GitHub each have commits the other doesn't. Lovable then pushes its changes to a branch called `lovable-sync`, and the next sync from GitHub replaces Lovable's version of the branch with the GitHub version.
- When a push replaces Lovable's work, Lovable saves a backup first, and the project owner can choose "Restore Lovable work" or "Keep repository version".
- A `lovable-sync` branch should be merged with a normal merge commit, not a squash or rebase.

```
Please:
```

1. Explain in plain words what most likely happened.
2. Tell me which work could be at risk right now, and the safest first step to protect it.
3. Give me the steps to get back to one clean synced branch, with each Git command explained in one line.
4. Tell me the one habit that would have prevented this.

 **Tips to customize:** Only the project owner can restore Lovable's backup. Lovable shows the restore notice for 24 hours after the push. After that, the owner can click **Check for saved Lovable work** in the project's Git settings. Pushes can also land on `lovable-sync` when GitHub rejects them, for example because of branch protection rules. Review that branch on GitHub and merge it into your synced branch.

4.3 I Changed the Backend Outside Lovable

Use this when you edited a backend function or the database structure in your own editor and pushed it. It's for making sure those changes actually go live, because Lovable doesn't deploy them for you when your commits sync.

Fill in: [FILES I CHANGED] (e.g. `supabase/functions/send-reminder/index.ts`), [WHAT THE CHANGE DOES] (e.g. "sends the reminder email an hour earlier").

```
I changed backend files in my Lovable project from my own editor, and the commit has synced to Lovable.
```

```
Files I changed: [FILES I CHANGED]
```

```
What the change does: [WHAT THE CHANGE DOES]
```

```
I know that Lovable doesn't deploy a changed Edge Function or run a new migration file when commits sync.
```

```
Please:
```

1. Sort my changed files into: Edge Functions (`supabase/functions/`), migrations (`supabase/migrations/` or `drizzle/migrations/`), or other.
2. For each Edge Function, write the short message I should send in Lovable's chat to deploy it, and tell me how to check it worked.
3. For any database change, explain what it changes in plain words, whether it could delete or damage existing data, and write the message I should send in Lovable's chat to make the change properly.
4. Give me a 3-step test I can run in the Lovable preview to confirm everything works.

 **Tips to customize:** Lovable's docs say to ask Lovable in the project chat to deploy a changed Edge Function, then check it under Lovable Cloud's Edge functions page. For database structure changes, ask in the chat too, and Lovable records the change as a new migration file in your repository. You can also run the SQL yourself in Lovable's SQL editor, but only if you understand exactly what it does. Reverting a version in Lovable restores code, not database data, so be extra careful with anything that deletes or changes data.

Bonus: Magic Phrases for Working Outside Lovable

- "Don't change any files yet. Explain first."
- "Work on a new feature branch, not main."
- "Never force-push, rebase, or squash the synced branch."
- "Use a test account. Don't touch real data."
- "Flag any change to backend functions or migrations."
- "Show me the exact Git command and what it does before you run it."

Graduate From Lovable

That's the whole pack: **9 prompts across 4 sections**. Connect GitHub safely, understand what Lovable built, run it on your own computer, and switch between Lovable and your editor without ever losing a week of work.

Happy building!