

MINI PACK

Lovable Credit Saver

Build more, burn less: get further on every Lovable credit.

For Lovable · plus any AI chat

9 prompts · 4 sections

Welcome! How to Use This Pack

You're halfway through building your app and the credits are almost gone. Most of them went on retries, vague requests, and big changes that broke something else and had to be fixed again.

This pack helps you spend credits on progress instead of rework. You'll learn to see where your credits go, pick the cheapest mode for each job, write requests that land the first time, stop paying for fix loops, and keep big builds from running away with your balance.

Lovable changes its plans, pricing, and features often, so this pack doesn't quote prices or credit amounts. When a tip mentions a setting or a cost rule, it comes from Lovable's official docs at docs.lovable.dev as of October 2026. If your screen or your plan looks different, trust Lovable's current docs and your own Usage details page.

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. Each tip also says where to run the prompt: in Lovable (and in which mode) or in a separate AI chat.

Works best with: Lovable, in the mode each tip names. Prompts 1.1, 2.3, and 4.2 also work in any AI chat, like ChatGPT or Claude, so you can think things through there without using Lovable credits.

The three modes, in one line each. Lovable's project chat has three modes. **Chat mode** discusses your app and never changes code. **Plan mode** investigates and writes a plan you approve, and also never changes code. **Build mode** makes the changes. Lovable's docs describe Chat mode as the cheaper way to ask questions, and Build mode costs depend on how big and complex the work is.

Using the free Lovable prompts on our site too? Those cover writing your project Knowledge, planning a bigger feature, and debugging. This pack goes deeper on making every credit count.

Quick Start: Find Your Problem

- **No idea where your credits went?** Prompt 1.1.
- **Not sure which mode to use?** Prompt 1.2.
- **Lovable keeps changing things you didn't ask for?** Prompts 2.1 and 3.2.
- **Lots of tiny tweaks to make?** Prompt 2.2.
- **Stuck in a fix loop?** Prompt 3.1.
- **About to build something big?** Prompts 4.1 and 4.2.

Golden rule: Decide in Chat mode, plan in Plan mode, and only spend Build mode credits on a change you've already decided on.

What's Inside

1. See Where Your Credits Go (2 prompts)
2. Get It Right the First Time (3 prompts)
3. Stop Paying for Fix Loops (2 prompts)
4. Keep Big Builds Under Control (2 prompts)

SECTION 1

See Where Your Credits Go

You can't save what you can't see.

1.1 Find What's Eating My Credits

Use this when your credits are running out faster than you expected and you don't know why. It's for turning your Lovable usage numbers into a short list of habits to change, without spending more Lovable credits to analyze them.

Fill in: [WHAT I'M BUILDING], [USAGE SUMMARY] (e.g. totals from Usage details for the last 30 days: Build, Chat, Cloud, AI), [MY 5 MOST EXPENSIVE MESSAGES] (what you asked and the credits used for each), [HOW I USUALLY WORK] (e.g. "I mostly use Build mode", "I retry when something breaks").

```
I build with Lovable and my credits run out too fast. Help me find the habits that
cost the most. Use only the numbers I give you, and don't guess any prices or
credit amounts.
```

```
What I'm building: [WHAT I'M BUILDING]
```

```
My usage summary: [USAGE SUMMARY]
```

```
My 5 most expensive messages and what each cost: [MY 5 MOST EXPENSIVE MESSAGES]
```

```
How I usually work: [HOW I USUALLY WORK]
```

```
Please:
```

1. Sort my spending into: building features, fixing things that broke, retries of the same request, small tweaks, and running the app (hosting, backend, AI features).
2. Point out patterns, like many retries on the same problem, big vague requests, or Build mode used for questions.
3. Give me the 3 habit changes that would save the most, in order, each with a one-line example of what to do instead.
4. Tell me which of my expensive messages could have been a Chat mode question, a Plan mode message, or several smaller requests.



Tips to customize: Run this in a separate AI chat, not in Lovable, so the analysis itself doesn't use Lovable credits. In Lovable, you'll find your numbers under Workspace settings → Plans & credit usage → Usage details, and the exact cost of any single message under Credits used in the More options menu below Lovable's reply.

1.2 Pick the Right Mode Before I Send

Use this when you're about to send a request and aren't sure whether it should be a question, a plan, or a build. It's for getting a quick recommendation on the cheapest mode that will still get the job done.

Fill in: [WHAT I WANT] (describe the change or question in a sentence or two).

```
Before you do anything, help me pick the right mode for this request. Don't change any code.
```

```
What I want: [WHAT I WANT]
```

```
Tell me:
```

1. Is this a question, a decision, a plan, or a change? Explain in one sentence.
2. Which mode fits best: Chat (discuss, no code changes), Plan (write a plan I approve, no code changes), or Build (make the change)?
3. If it's a change: is it small and clear enough to build directly, or should I plan it first?
4. If it's big: how would you split it into smaller requests I can build and check one at a time?
5. Anything I should decide or clarify first so the build works the first time.

 **Tips to customize:** Send this in Chat mode. Lovable shows a "Start building" card when you ask for a change in Chat mode, so nothing gets built until you choose Build.

SECTION 2

Get It Right the First Time

Clear requests cost less than retries.

2.1 The One-Change Request

Use this when you know exactly what you want changed and want Lovable to do only that. It's for writing a tightly scoped Build request that names the files, the expected result, and what must not change.

Fill in: [THE CHANGE] (one change only), [WHERE] (e.g. @src/components/Header.tsx or "the pricing section on the homepage"), [DONE LOOKS LIKE] (e.g. "the button is gray on every page"), [DON'T TOUCH] (e.g. "the navigation, the login flow, any database tables").

Make one change only.

The change: [THE CHANGE]

Where: [WHERE]

Done looks like: [DONE LOOKS LIKE]

Don't touch: [DON'T TOUCH]

Rules:

- Change only what this task needs. Keep existing names, styles, and structure.
- Don't refactor, rename, or "clean up" anything else.
- If the change needs edits outside the place I named, stop and tell me which files and why before you edit them.
- When you're done, list every file you changed in one line each.



Tips to customize: Send this in Build mode. Type @ in the project chat to reference a specific file, which helps Lovable find the right place without exploring the whole project. Lovable's docs say small, focused edits usually cost less than large or multi-step changes.

2.2 Batch the Small Fixes

Use this when you have a list of small copy, color, or spacing tweaks and don't want to send them one by one. It's for bundling related small fixes into one clear, numbered request, and keeping pure typo fixes out of chat entirely.

Fill in: [PAGE OR AREA] (e.g. "the homepage"), [MY LIST OF TWEAKS] (one per line, each with the exact new text or value).

```
Make these small fixes in one pass. They're all in [PAGE OR AREA].
```

```
[MY LIST OF TWEAKS]
```

```
Rules:
```

- Do exactly what each item says, nothing more.
- Don't change layout, components, or logic beyond these items.
- If any item is unclear or would need a bigger change, skip it and tell me why instead of guessing.
- When you're done, give me the list back with "done" or "skipped" next to each item.

 **Tips to customize:** Send this in Build mode. Keep each batch to one page or area so a mistake is easy to spot and undo. For plain wording changes, try the preview toolbar's Edit text inline mode first: Lovable's docs say a limited number of inline text edits are free before they start using credits.

2.3 Working Rules for My Project Knowledge

Use this when you keep typing the same warnings ("don't touch the login", "ask first") into every request. It's for writing a short set of working rules for your project Knowledge, so Lovable follows them on every message without you repeating them.

Fill in: [WHAT I'M BUILDING], [PARTS THAT ALREADY WORK] (e.g. "login, the dashboard, Stripe checkout"), [THINGS LOVABLE KEEPS DOING THAT I DON'T WANT] (e.g. "redesigns pages I didn't mention", "adds new libraries").

```
Write a short "Working rules" section for my Lovable project Knowledge. Keep it under 150 words, as a bullet list of direct rules.
```

```
What I'm building: [WHAT I'M BUILDING]
```

```
Parts that already work: [PARTS THAT ALREADY WORK]
```

```
Things Lovable keeps doing that I don't want: [THINGS LOVABLE KEEPS DOING THAT I DON'T WANT]
```

```
Include rules that:
```

1. Keep changes limited to what each request asks for.
2. Treat the working parts above as locked unless I name them.

3. Require asking me before adding new libraries, new database tables, or deleting anything.
4. Ask a clarifying question when a request could mean several things, instead of guessing.
5. List the files changed at the end of each change.

Write each rule as one short, specific sentence. No vague rules like "write clean code".



Tips to customize: Run this in a separate AI chat or in Chat mode, then paste the result into Project settings → Knowledge. Lovable includes Knowledge with every message, so keep it short and specific. Lovable also reads an AGENTS.md file in your project's repository, if you use one.

SECTION 3

Stop Paying for Fix Loops

Every failed retry costs credits. Break the loop early.

3.1 Fix or Revert? Decide Before I Spend More

Use this when a change went wrong and you've already tried to fix it once or twice. It's for deciding, in Chat mode, whether to keep fixing or go back to a working version and try again in smaller steps.

Fill in: [WHAT I WAS TRYING TO DO], [WHAT BROKE], [WHAT WE'VE TRIED] (list each fix attempt), [LAST VERSION THAT WORKED] (e.g. "before I asked for the redesign", "my bookmarked version from Tuesday"), [DID THE CHANGE TOUCH MY DATABASE?] (yes / no / not sure).

Don't change any code. Help me decide whether to keep fixing this or revert.

What I was trying to do: [WHAT I WAS TRYING TO DO]

What broke: [WHAT BROKE]

What we've tried so far: [WHAT WE'VE TRIED]

Last version that worked: [LAST VERSION THAT WORKED]

Did the change touch my database? [DID THE CHANGE TOUCH MY DATABASE?]

Please:

1. Explain the most likely root cause in plain English, and how confident you are.
2. Tell me whether the fixes so far have added code that hides the real problem.
3. Compare two options: (a) one more targeted fix, with exactly what it would change, or (b) revert to the last working version and redo the change in smaller steps.
4. Recommend one option and say why.
5. If I revert: remind me what reverting does and doesn't undo, and give me the first small step to re-send afterward.



Tips to customize: Send this in Chat mode. Lovable's docs note that reverting restores your code but not your database data, and that reverting doesn't refund the credits already spent. Before your next big change, bookmark a known-good version in the History panel so going back is one click.

3.2 "Stop and Ask Me" Guardrail

Use this when you're about to change a delicate part of your app, like login, payments, or your database, and can't afford a rewrite. It's for adding guardrails to a Build request so Lovable checks the related code first and pauses before anything risky.

Fill in: [THE TASK], [THE DELICATE AREA] (e.g. "login and sign up", "Stripe checkout", "the orders table"), [WHAT MUST KEEP WORKING] (e.g. "email login and Google login").

```
This change is in a delicate part of my app, so go carefully.
```

```
Task: [THE TASK]
```

```
Delicate area: [THE DELICATE AREA]
```

```
Must keep working: [WHAT MUST KEEP WORKING]
```

```
Before you edit anything:
```

1. Look at all the code related to this area and tell me in 3-5 bullets what you plan to change.
2. Stop and wait for my OK if your plan includes any of these: rewriting a working component, deleting code or data, changing a database table, or touching files outside this area.

```
While you work:
```

- Change only what the task needs. Don't make "just in case" changes elsewhere.
- If anything is unclear, pause and ask instead of guessing.

```
When you finish, tell me how to check that everything under "Must keep working" still works.
```

 **Tips to customize:** Send this in Build mode. If you'd rather approve a full written plan first, send the same text in Plan mode instead: Plan mode never changes code, and Lovable starts building only after you approve the plan.

SECTION 4

Keep Big Builds Under Control

Big requests are where credits disappear fastest.

4.1 Plan Small, Build in Steps

Use this when you're about to add a big feature, like accounts, payments, or a new section of your app. It's for getting a focused plan that breaks the feature into small steps you can build and check one at a time.

Fill in: [THE FEATURE], [USERS SHOULD BE ABLE TO] (e.g. "save favorite recipes and see them on a Favorites page"), [WHERE IT LIVES] (e.g. @src/pages/Recipes.tsx or "the recipe pages").

Plan this feature as a series of small steps. Don't write any code yet.

Feature: [THE FEATURE]

Users should be able to: [USERS SHOULD BE ABLE TO]

Where it lives: [WHERE IT LIVES]

Plan rules:

1. Split the work into steps that each change one thing and can be checked in the preview.
2. For each step: what changes, which files, and how I'll know it worked.
3. Put anything that changes the database in its own step, and flag it.
4. Mark which steps are optional for a first version.
5. Keep the investigation focused on the files and areas I named. Ask me before exploring the rest of the project.

 **Tips to customize:** Send this in Plan mode. Lovable's docs say broad planning requests are more likely to trigger extra research, which costs more, so naming the files and area helps. After you approve, build one step at a time and check each in the preview before the next.

4.2 Set My Spending Guardrails

Use this when you want to stop a single long request, or a busy week, from draining your balance by surprise. It's for making a short, personal checklist of Lovable's spending controls and the habits that go with them.

Fill in: [MY PLAN] (e.g. Free, Pro, Business), [AM I THE WORKSPACE OWNER OR ADMIN?] (yes / no), [HOW I WORK] (e.g. "solo, a few hours each evening", "a team of 3"), [WHAT WORRIES ME] (e.g. "one long build using lots of credits", "teammates using it all up").

Help me set up spending guardrails for my Lovable workspace. Base your answer only on the settings I list below, and tell me to check Lovable's docs for anything you're not sure about. Don't guess prices or credit amounts.

My plan: [MY PLAN]

Am I the workspace owner or admin? [AM I THE WORKSPACE OWNER OR ADMIN?]

How I work: [HOW I WORK]

What worries me: [WHAT WORRIES ME]

Settings Lovable's docs describe:

- Credit check-ins for long-running Build mode messages (a pause card when one message passes a credit level you choose).
- Usage limits and alerts for members, projects, or the whole workspace (paid plans; owners and admins).
- A monthly spend limit for auto top-up, if I use auto top-up.
- The Usage details page and the cost shown under each message.

Please:

1. Recommend which of these I should turn on, given my plan and how I work, and why.
2. Suggest a simple weekly habit for checking my usage.
3. Give me 3 rules of thumb for my own requests that prevent surprise spending.



Tips to customize: Run this in a separate AI chat. Lovable's docs also say that a Build mode message sent as a /goal costs noticeably more, because Lovable keeps working until the goal is achieved, so save /goal for tasks where that's worth it.

Bonus: Magic Phrases for Saving Credits

- "Don't change any code yet."
- "Change only what this task needs."
- "Stop and ask me before touching anything outside this area."
- "List every file you changed."
- "What have we already tried, and why didn't it work?"
- "Split this into small steps I can check one at a time."

Lovable Credit Saver

That's the whole pack: **9 prompts across 4 sections**. See where your credits go, decide in Chat mode, plan in Plan mode, build one clear change at a time, and stop fix loops before they eat your balance.

Happy building!