
Module 16: Lovable: Basic to Advanced

Go from a prompt to a published web app with a database, login, AI features and payments, then sync it to GitHub and keep improving it with Claude Code.



AUTHOR

Varun Kumar

Product Management Master Handbook
v0.1 · 23 October 2026

Built from the 55-lecture course study guide and extended to the 12-module Product Core curriculum.
Hypothetical examples are labelled; no real company results are invented.

Contents

16.1 What Lovable is	3
16.2 The editor	4
16.3 Level 1 · Your first app in 30 minutes	4
16.4 Level 2 · Building like a pro	5
16.5 Level 3 · Database, login, AI, payments and GitHub	7
16.6 Labs	10
16.7 Prompt library	11
16.8 Saving credits	11
16.9 Common mistakes	11
16.10 Exercises with model answers	12
16.11 Key takeaways	12
16.12 Check your understanding	13
16.13 Recommended resources	13

MODULE 16

Lovable: Basic to Advanced

Go from a prompt to a published web app with a database, login, AI features and payments, then sync it to GitHub and keep improving it with Claude Code.

≈ 8 h study

Lovable changes fast

Lovable adds and renames features often. If a button below has moved, search the [Lovable docs](#). Screens are **illustrative wireframes**.

16.1 What Lovable is

Lovable builds web apps from a conversation. You describe the app, and Lovable writes a real React application, shows a live preview, and can add a **database, user login, file storage and server functions** for you. You can publish with one click, connect your own domain, and sync the code to GitHub.

When to use Lovable

Clickable MVPs, internal tools, landing pages with forms, dashboards, booking and directory apps, and portfolio projects. For deep changes to a large existing codebase, use Claude Code (Class 15). For production at scale, an engineering team reviews security, performance and data.

16.1.1 What it costs (checked 23 Oct 2026)

Plan	Price	What you get
Free	\$0	5 credits a day (up to 30 a month), public projects on a Lovable address
Pro	about \$25 a month	100 monthly credits (more tiers available), private projects, custom domains, remove the badge, credit top-ups
Business	about \$50 a month	Team features such as single sign-on and security controls

A **credit** is used each time Lovable does work for you. Bigger changes use more. Backend services (database, AI calls, email) may have their own usage limits. Check lovable.dev/pricing.

16.2 The editor

1 Agent mode builds and edits. Chat mode discusses and plans without changing code, so use it to think before spending build credits.

2 Each message becomes a version in History. You can restore any version.

3 The preview updates live. Use Visual edits (click an element) for small text, colour and spacing changes.

4 Publish puts the app on a public link. Paid plans add custom domains.

The Lovable editor (illustrative wireframe, not a real product screenshot). Knowledge holds your project instructions, and Integrations is where you connect a database, GitHub, payments and email.

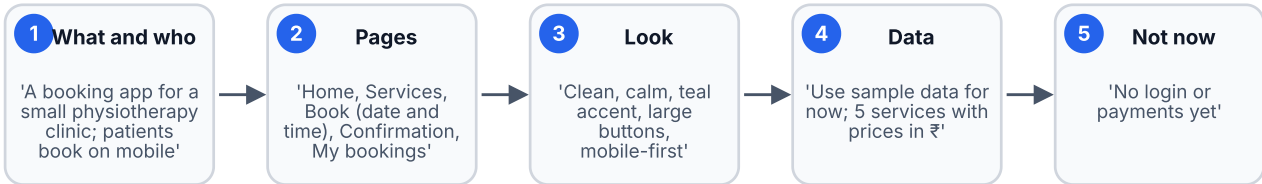
16.3 Level 1 · Your first app in 30 minutes

Level 1 · Beginner

Goal: build and publish a small, good-looking app, and learn the prompt pattern that saves credits.

16.3.1 The first-prompt template

A strong first prompt saves many fixes later. Include five things:



The Lovable first prompt. Saying what not to build yet keeps the first version small, cheap and easy to check.

16.3.2 Then iterate in small steps

1. Look at the preview on desktop **and** in the phone view.
2. Ask for **one change per message**: "Make the service cards show duration and price."
3. Use **Visual edits** for tiny text or colour changes, which is faster and cheaper than a prompt.
4. If something breaks, click **Try to fix**, or restore the last good version from **History**.
5. Click **Publish** and open the link on your phone.

16.4 Level 2 · Building like a pro

Level 2 · Working PM

Goal: plan before you build, keep the app consistent, debug calmly, and avoid wasting credits.

16.4.1 Plan in chat mode, then build

The wireframe shows a chat window titled "Lovable · Chat mode". It contains three numbered steps: 1. A user prompt: "I want patients to reschedule a booking. Don't build yet. What's the simplest approach, and what could go wrong?". 2. A system response: "Option A: a Reschedule button on My bookings that reopens the date picker... Risks: double bookings, time zones...". 3. A user confirmation: "Go with option A. Prevent double bookings. Build it.". Below the chat window, a green bar indicates "Switching to agent mode: building the reschedule flow".

- 1 Ask for options and risks first. Chat mode doesn't change code.
- 2 Read the plan. Remove anything you don't need.
- 3 Approve one clear plan, then let agent mode build it.

Planning a feature in chat mode (illustrative wireframe, not a real product screenshot).
Planning first is the biggest credit saver: one good build beats five corrections.

16.4.2 Knowledge: rules for every prompt

Open **Knowledge** (project instructions) and write rules Lovable follows on every change:

- What the app is, who uses it, and the tone of the copy.
- Design rules: colours, fonts, spacing, "mobile-first", "large tap targets".
- Data rules: "sample data only", "prices in ₹", "dates as 23 Oct 2026".
- Constraints: "don't change the booking flow unless asked".

16.4.3 Debugging without panic

Symptom	What to do
Red error in the preview	Click Try to fix once. If it fails twice, restore the last working version and ask in smaller steps.
A change broke something else	Restore from History , then say exactly what must not change.
It keeps repeating a mistake	Add a rule to Knowledge ; describe the bug with a screenshot.
Layout wrong on mobile	Switch the preview to phone size and name the element and the problem.
You are lost	Chat mode: "Explain how this page works and which files control it."

16.4.4 Reusing and sharing

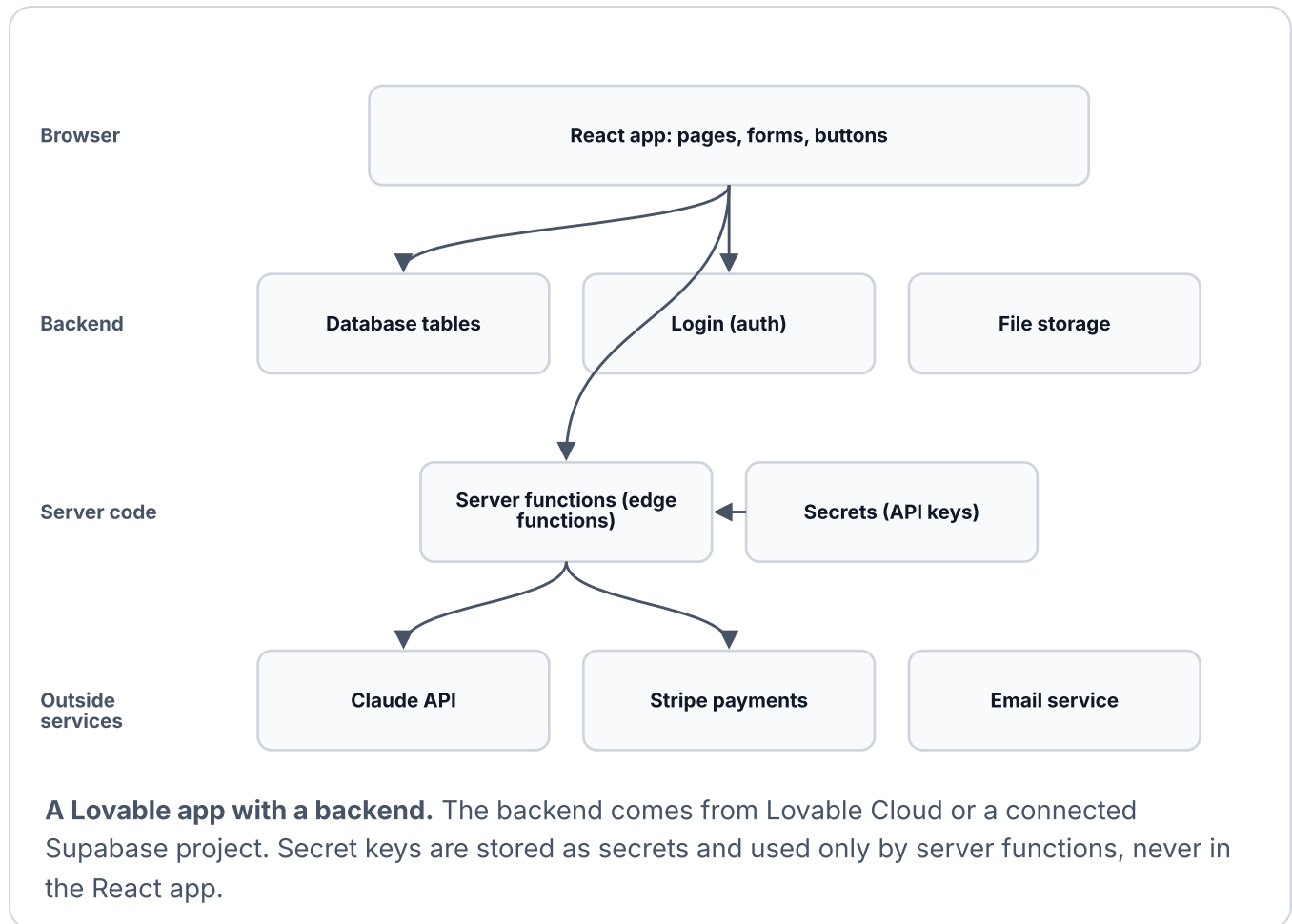
- **Remix** copies a project so you can try a risky idea without touching the original.
- **Share** a preview link for feedback before you publish.
- Upload images, logos or a screenshot of a design, and say "match this".

16.5 Level 3 · Database, login, AI, payments and GitHub

Level 3 · Advanced

Goal: turn the prototype into a working app with real data and accounts, safely.

16.5.1 How a Lovable app with a backend fits together



16.5.2 Database and login

Ask in plain words: "Add a backend. Store bookings in a database with patient name, phone, service, date and time. Add email login so patients see only their own bookings." Lovable creates the tables and the login screens.

Row-level security is not optional

Row-level security (RLS) rules decide which rows each user can read or change. Without them, a logged-in user might read everyone's bookings. Ask: "Turn on row-level security so each user can only read and edit their own bookings; staff can read all." Then **test with two accounts**. Run Lovable's security check before you publish.

16.5.3 AI features with a secret key

To add an "Ask the clinic" chatbot powered by Claude:

1. Get an API key from console.anthropic.com and set a spend limit.

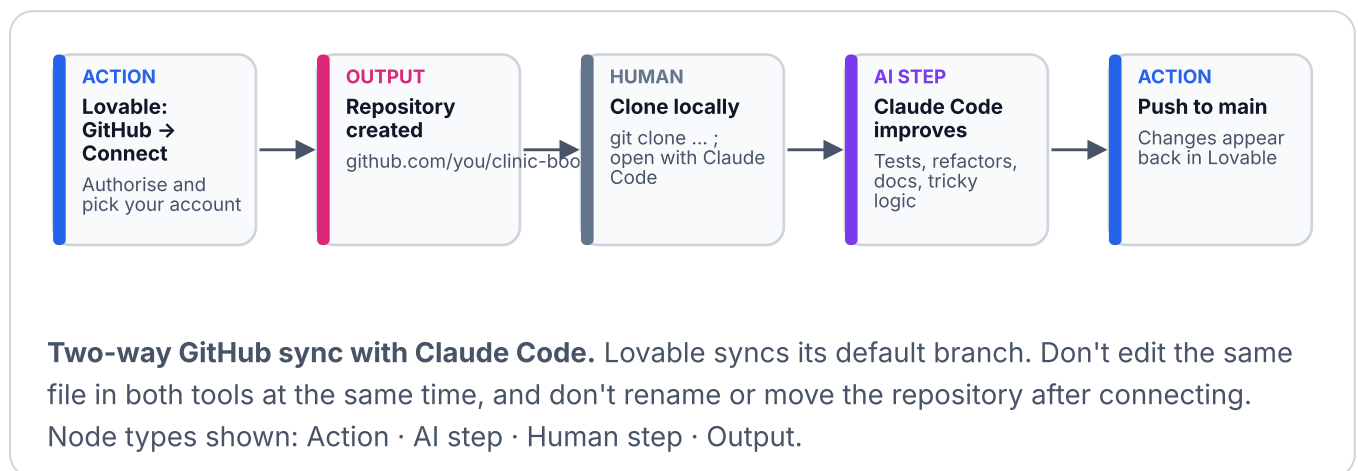
2. In Lovable, add it as a **secret** named `ANTHROPIC_API_KEY`. Never paste it in the chat or the code.
3. Prompt: "Create a server function that calls the Anthropic Messages API with the ANTHROPIC_API_KEY secret. Answer only from the clinic FAQ text in the database. Connect it to an 'Ask us' chat button."
4. Test it with five questions, including one it should refuse ("Can you diagnose me?").

Lovable may also offer **built-in AI** that needs no key of your own. It is the quickest option for a demo, with usage limits.

16.5.4 Payments and email

- **Stripe**: "Add a Pay deposit button using Stripe Checkout in test mode." Use Stripe's test cards and switch to live keys only after review.
- **Email**: connect an email service (for example Resend) and "send a confirmation email after a booking".

16.5.5 GitHub sync: work in Lovable and Claude Code



16.5.6 Publish and custom domain

1. Click **Publish**. Check the app in a private browser window.
2. On a paid plan: Settings → **Domains** → add your domain and follow the DNS steps shown.
3. Re-publish after changes. The live site updates only when you publish.

16.6 Labs

Lab 1 • Landing page with waitlist (Free, 20 minutes)

1. Prompt with the five-part template for a waitlist landing page.
2. Make two improvements, one prompt each. Use Visual edits for one text change.
3. Publish and open the link on your phone.

Lab 2 • Clinic booking app with a database (40 minutes)

1. Build the booking app with sample services.
2. Chat mode: "How should we store bookings?" Then approve the plan.
3. "Add a backend and save bookings to the database."
4. Make three bookings, refresh, and confirm they are still there.

Lab 3 • Login and row-level security (30 minutes)

1. "Add email login. Patients see only their own bookings."
2. "Turn on row-level security for bookings: owners read and edit their own; staff role reads all."
3. Create two test accounts. Check account A cannot see B's bookings.
4. Run the security check and fix anything it reports.

Lab 4 • 'Ask us' AI assistant (Pro or API, 30 minutes)

1. Add the clinic FAQ as a database table or Knowledge text.
2. Store `ANTHROPIC_API_KEY` as a secret.
3. Ask for the server function and chat button as described above.
4. Test five normal questions and one it must refuse.

Lab 5 • GitHub sync and Claude Code (30 minutes)

1. Connect GitHub and let Lovable create the repository.
2. Clone it and run Claude Code: "Write a README explaining how to run and deploy this app."
3. Commit and push to main. Check the README appears in Lovable's code view.

Lab 6 · Publish with a custom domain (paid, 20 minutes)

1. Publish the app.
2. Add your domain under Domains and follow the DNS instructions.
3. Wait for the certificate, then test on mobile.

16.7 Prompt library

Goal	Prompt
Polish	"Make this feel premium: consistent spacing, larger headings, subtle shadows, no new features."
Empty and error states	"Add empty, loading and error states to My bookings."
Accessibility	"Check colour contrast and keyboard navigation; fix issues without changing the layout."
Admin view	"Add a staff-only page listing today's bookings with a status dropdown."
Analytics	"Track button clicks for Book and Pay deposit; show counts on the admin page."
Explain	"In chat mode: explain how login works in this app, in plain language."

16.8 Saving credits

- Plan in **chat mode**, then build once.
- Use **Visual edits** for small text, colour and spacing changes.
- One change per prompt, and be specific about the element.
- Restore a version instead of asking Lovable to "undo everything".
- Move heavy refactoring and documentation to **Claude Code** via GitHub.

16.9 Common mistakes

- **Building everything in the first prompt:** start small and add features one at a time.
- **No row-level security:** test with two accounts before anyone real signs up.

- **API keys in prompts or code:** always use secrets and server functions.
- **Publishing real data in a public free project:** free projects are public.
- **Fighting the same bug for many credits:** restore, simplify the request, and add a Knowledge rule.

16.10 Exercises with model answers

Exercise 1 · Write the first prompt

Write a first Lovable prompt for an internal tool that tracks customer feedback by theme.

Model answer

"Build an internal feedback tracker for a SaaS product team (what and who). Pages: Feedback list with filters, Add feedback form, Themes view with counts, Feedback detail (pages). Clean, neutral, compact tables, desktop-first (look). Use 20 sample feedback items across 5 themes and 3 customer segments (data). No login, no integrations yet (not now)."

Exercise 2 · What's the security bug?

A booking app shows "My bookings", but when you change the booking ID in the address bar you can open someone else's booking. What is missing and how do you fix it?

Model answer

Row-level security, or the server check behind it. The app filters bookings in the page but the database still returns any row. Ask Lovable to enable RLS so users can only read rows where they are the owner, then retest with two accounts and the edited address.

16.11 Key takeaways

- Use the **five-part first prompt**, then change one thing at a time.
- **Chat mode** to plan, **agent mode** to build, **History** to undo.
- Real apps need a **backend, login and row-level security**. Test with two accounts.
- Keep keys in **secrets** used by **server functions**.
- **GitHub sync** lets you combine Lovable's speed with Claude Code's depth.

16.12 Check your understanding

CHECK YOUR UNDERSTANDING

Which Lovable mode lets you discuss and plan without changing the code?

- A. Agent mode
- B. Chat mode
- C. Publish
- D. Remix

Why: Chat mode plans and explains; agent mode makes changes.

What stops a logged-in user from reading other users' rows?

- A. A nicer login page
- B. Row-level security rules
- C. A custom domain
- D. More credits

Why: RLS rules decide which rows each user can access.

Where should a Claude API key go in a Lovable app?

- A. In the chat prompt
- B. In the React code
- C. In a secret used by a server function
- D. In the page title

Why: Secrets used only by server-side functions keep keys away from visitors.

16.13 Recommended resources

- [Lovable docs](#) · [Lovable pricing](#)
- [Supabase docs: row-level security](#)
- [Stripe test cards](#)
- Lovable's official YouTube channel and community Discord, for feature walkthroughs and launch-week tutorials