FIELD GUIDE · NO JARGON

The Anatomy of an App

A plain-English map of what every app is made of — and how the pieces talk to each other.
Written for builders who think in ideas, not infrastructure. · Bluemoonkey

Every app — from Instagram to your local booking tool — is built from the same handful of parts. You don't need to become a developer to understand them. You just need to know **what each part does** and **why it exists**.

This guide explains the five building blocks you'll see in every blueprint: the **frontend**, the **backend**, the **database**, the **admin panel**, and **APIs**. By the end, you'll know exactly what to plan before you open Cursor or Claude Code.

01 · Frontend

The face of your app

The frontend is everything your user sees, clicks, and interacts with. It's the website in their browser, the screen on their phone, the buttons, forms, menus, and images. If a user can touch it or read it on screen — that's the frontend.

Think of it like the storefront of a shop. It's designed to look good, be easy to navigate, and guide people toward doing something — signing up, buying, posting, booking.

> **Remember** — The frontend shows information and collects input. It doesn't store data permanently or run heavy logic on its own — it asks the backend to do that.

> **Real-world examples**
> The login page where someone enters their email. The dashboard showing their recent orders. The “Post” button on a social app. The checkout form in an online store.

02 · Backend

The engine room

The backend is the part of your app that runs behind the scenes. Users never see it directly. It handles the logic: checking passwords, processing payments, sending emails, deciding who can see what, and coordinating everything between the frontend and the database.

If the frontend is the storefront, the backend is the stockroom, the till, and the manager's office combined. It receives requests from the frontend (“this user wants to log in”), does the work, and sends back a

response (“login successful, here's their profile”).

> **Remember** — When something needs to be secure, calculated, or coordinated — that happens in the backend. Never put secret keys or payment logic in the frontend.

> **Real-world examples**
>
> Verifying a password when someone logs in. Charging a credit card. Sending a “Welcome!” email after signup. Checking if a user is allowed to edit a post.

03 · Database

Where your data lives

The database is where your app stores information permanently. User accounts, posts, orders, settings, messages — anything that needs to survive after someone closes the browser lives here.

Think of it as a very organised filing cabinet. The backend reads from it (“get this user's orders”) and writes to it (“save this new post”). The frontend never talks to the database directly — that would be like letting customers walk into the stockroom and change prices themselves.

> **Remember** — If you'd be upset to lose it when someone refreshes the page, it belongs in the database.

> **Real-world examples**
>
> A table of user profiles (name, email, password hash). A list of blog posts with titles and content. Order history for an e-commerce store. Saved preferences and settings.

04 · Admin Panel

Your control room

An admin panel is a special frontend — but only for you (or your team), not regular users. It's where you manage the app: view users, moderate content, change settings, see analytics, and fix problems without touching code.

Not every app needs one on day one, but most apps grow into needing it. A social app needs somewhere to ban spam accounts. A booking app needs a calendar view of all appointments. A shop needs to update products and process refunds.

> **Remember** — The admin panel is still a frontend — it just connects to backend endpoints that regular users can't access.

> **Real-world examples**
>
> A dashboard showing signups this week. A page to approve or reject user-submitted content. A settings screen to change pricing or feature flags.

05 · APIs

How the parts talk to each other

API stands for Application Programming Interface — but forget the acronym. An API is simply a **defined way for one part of your app to ask another part for something**.

When your frontend needs data, it calls an API on your backend ("give me this user's profile"). When your backend needs to store something, it talks to the database. APIs can also connect your app to outside services — Stripe for payments, SendGrid for emails, OpenAI for AI features.

> **Remember** — Every arrow in a Bluemoonkey blueprint between a frontend, backend, or external service is usually an API connection. If two parts need to exchange data, there's an API between them.

> **Real-world examples**
>
> `GET /users/123` — frontend asks backend for user #123's profile.
> `POST /orders` — frontend sends a new order to the backend to process.
> Stripe API — backend asks Stripe to charge a card.

06 · How it all connects

The full picture

Here's the flow that happens in almost every app, every time a user does something meaningful:

User — Clicks, types, taps

↓

Frontend — What they see

↓ API call

Backend — Logic & security

↓ read / write

Database — Stored data

The admin panel sits alongside the regular frontend — same pattern, different access. External services (payments, email, AI) plug into the backend via their own APIs.

When you blueprint an app in Bluemoonkey, you're mapping exactly this: which screens exist, what logic runs where, what data gets stored, and how everything connects. Get this map right before you build, and your AI coding tool has a fighting chance of building what you actually meant.
