Finance Stack Prompt Pack

A step-by-step set of prompts to build your own AI-powered finance system from scratch — no coding required.

Each section is a stage. Copy the prompt, adapt the details in [brackets] to your business, paste it into the relevant tool.

by Nik McFly

nikmcfly.com

STAGE 1

Database Schema

Paste into Claude, then paste the output SQL into Supabase SQL Editor.

```
Design a PostgreSQL database schema for a small company
finance tracker.
Requirements:
- Accounts table: [list your accounts, e.g.,
"PayPal (USD), local bank (local currency),
crypto wallet (USDT)"]
- Transactions table with fields: date, amount,
currency, exchange_rate, type, category,
description, payee, note
- Transaction types: Income, Expense,
Transfer (between accounts), Salary
- Budget items table: monthly budget per category
(planned vs actual)
- Team members table: name, role, monthly rate in USD
- Subscriptions table: service name, cost,
billing cycle, status
- A database VIEW called "account_balances" that
calculates current balance per account by summing
all transactions
- Sign convention: income is positive, everything else
(expense, salary, transfer) is stored as negative
- amount_usd should be a computed column:
amount x exchange_rate
Output the complete SQL I can paste into Supabase SQL
Editor. Include CREATE TABLE statements, the VIEW, and
any CHECK constraints.
```

STAGE 1.5

Visual Design

Paste into Variant.

```
Create a nice minimalistic but functional financial
dashboard for a [music label / your industry] company.
```

Variant generates a polished design layout you can use as the visual foundation. Export or screenshot the result — then bring it into Lovable as the base for your working app.

STAGE 2

Web Dashboard

Paste into Lovable.

```
Create a nice minimalistic but functional financial
dashboard for a [music label / your industry] company.
Connect to my Supabase database (I'll provide the URL
and anon key).
Pages needed:
1. Dashboard - account balances as cards, income vs
expense chart (last 6 months), recent transactions
2. Transactions - full table with filters (date range,
type, category, search), pagination
3. Budget - monthly plan vs actual spending, deviation
4. Payroll - team members with roles and monthly rates
5. Settings - basic app settings
Design: warm minimalistic theme, light background
(#F5F4F0), clean cards, no heavy shadows.
Use shadcn/ui components.
The base currency is USD. Show all amounts as $X,XXX.
```

Follow-up prompts for iteration:

```
Make the transaction table sortable by date and amount.
Add a button to add new transactions with a dialog form.
```

```
The budget page should show a progress bar for each
category - green if under budget, red if over.
Show the percentage used.
```

```
Add a monthly summary at the top of the dashboard:
total income, total expenses, net profit/loss, and
percentage change from last month.
```

STAGE 3

API Server

Paste into Claude.

```
Write a Node.js Express server that:
1. Connects to Supabase using the service_role key
(environment variable SUPABASE_KEY)
2. Exposes these REST API endpoints:
- GET /api/balances - account balances from VIEW
- GET /api/transactions - with query params:
date_from, date_to, type, category, search, limit
- POST /api/transactions - add new transaction
(auto-negate for expense/salary/transfer types)
- DELETE /api/transactions/:id - delete by UUID
- GET /api/budget?month=YYYY-MM - plan vs actuals
- GET /api/team - active team members
- GET /api/subscriptions - active subscriptions
- GET /api/summary?months=6 - monthly breakdown
3. All /api/* endpoints require Bearer token auth
(API_SECRET_KEY env var)
4. Also serves as an MCP (Model Context Protocol)
server on POST/GET/DELETE /mcp - so AI assistants
like Claude can call the same functions as tools.
Use @supabase/supabase-js for database access.
Use @modelcontextprotocol/sdk for MCP.
Environment variables:
SUPABASE_URL, SUPABASE_KEY, API_SECRET_KEY
Output the complete index.js and package.json.
```

STAGE 4

Telegram Bot

Paste into Claude.

```
Add a Telegram bot to the server from Stage 3.
The bot should:
1. Use grammy library
2. Receive messages in a group chat, respond only
when @mentioned or replied to
3. Use Claude Haiku (claude-haiku-4-5-20251001) to
understand natural language queries
4. Claude Haiku gets the same finance tools
(get_balances, get_transactions, add_transaction,
etc.) and calls them as needed
5. Support conversation history per chat
(last 20 messages)
6. When adding transactions, always confirm with
the user before inserting
Environment variables:
TELEGRAM_BOT_TOKEN, ANTHROPIC_API_KEY
Security:
- Whitelist specific Telegram user IDs
(ALLOWED_TELEGRAM_USERS env var)
- Whitelist specific chat IDs
(ALLOWED_TELEGRAM_CHATS env var)
- If no whitelist configured, block everyone
(fail-safe)
Set up webhook mode (not polling) for production.
```

STAGE 5

Security Hardening

Paste into Claude after everything works.

```
Audit my finance server for security issues and fix:
1. REST API: add Bearer token authentication on all
/api/* endpoints
2. Telegram bot: add user/chat whitelist (only
approved users can interact)
3. Webhook: verify Telegram's secret token header
(X-Telegram-Bot-Api-Secret-Token)
4. Search inputs: sanitize for SQL injection
(escape %, _, \ in ilike queries)
5. MCP endpoint: add optional Bearer auth
(separate MCP_SECRET_KEY env var)
For each fix, show me the code and explain what
attack it prevents.
Generate secure random tokens I can use:
- One for API_SECRET_KEY
- One for TELEGRAM_WEBHOOK_SECRET
```

STAGE 6

Deployment

Supabase: Already hosted. Just save your project URL and keys.

Railway (server + bot):

1. Push code to a GitHub repo (make it PRIVATE)
2. Connect Railway to the repo
3. Add all environment variables in Railway dashboard
4. Set the webhook URL: https://your-railway-url.up.railway.app/webhook

Lovable (dashboard): Already hosted by Lovable. Connect your Supabase credentials in app settings.

Register the Telegram webhook:

```
curl -X POST \
"https://api.telegram.org/bot<TOKEN>/setWebhook" \
-H "Content-Type: application/json" \
-d '{"url": "https://your-url.up.railway.app/webhook",
"secret_token": "<YOUR_WEBHOOK_SECRET>"}'
```

Tips

Start with Stage 1 and 2. Get the database and dashboard working first. Add the API server and bot later.

Iterate in conversation. When something breaks, paste the error into Claude and describe what you expected. It fixes most things in one turn.

Adapt the categories. Replace music label categories with whatever fits your business — e-commerce, agency, SaaS, whatever.

Budget ~$7/month for the core system. Supabase free tier + Railway (~$5) + Telegram (free) + Claude API for the bot ($1-2). Lovable is free for building; hosting can be moved to Vercel or Netlify for $0.
