Copy-Paste Technical Build Guide – QuoteForge MVP

No business/legal – just code, commands, and prompts

Target: Full-stack app ready in 6–8 weeks with Cursor + Claude 4

Date: December 08, 2025

Copy this entire guide into Notion/Google Docs → follow sequentially. Use Cursor for all prompts (Cmd+K on files).

Prerequisites (1–2 Hours Setup)

1. **Install Cursor**: Download system installer (not user): https://download.cursor.com/windows/cursor-setup-1.7.46-x64.exe (or latest from cursor.com). Run → Sign in → Add Claude API key (from console.anthropic.com) in Settings → AI. Enable Privacy Mode.
2. **Accounts & Keys**:
    - Claude Pro (API key)
    - Supabase (create project → get URL/anon key)
    - Vercel (free)
    - Render (free)
    - Stripe (test keys)
    - Resend (API key for emails)
3. **Local Environment**:

```
Bash

# Install Node.js (v22 latest) + Python 3.12
nvm install 22 && nvm use 22  # For Node
pyenv install 3.12 && pyenv global 3.12  # For Python (or use official i
```

4. **GitHub Repo**: Create private repo "quoteforge-mvp" → clone locally.

Phase 1: Scaffold Project (1 Hour – One Mega-Prompt)

1. Open Cursor → New Folder "quoteforge-mvp" → Empty file "setup.ts" → Cmd+K → Paste:

```
text

Scaffold full-stack quoting app:
Frontend: Next.js 15 App Router + Tailwind CSS v4 + shadcn/ui (install a
Backend: FastAPI in /backend folder with Python 3.12.
Database: Supabase Postgres + Auth.
Add .env.example with SUPABASE_URL, SUPABASE_ANON_KEY, ANTHROPIC_API_KEY
Include basic README.md and gitignore.
```

2. Run:

```
Bash

cd frontend && npm install
cd backend && pip install fastapi uvicorn supabase python-multipart anth
```

3. Commit: `git add . && git commit -m "Initial scaffold" && git push`.

Phase 2: Backend – 78-Rule Costing Engine (2–3 Hours)

1. In /backend/rules_engine.py → Cmd+K → Prompt:

```
text

Convert this Google Sheet to Python code: [paste your sheet link].
Create dicts: materials (name: {'price_lb': float, 'density': float}), t
Write function calculate_quote(inputs: dict) -> dict: Apply all 78 rules
Add pytest unit tests covering 10 rules.
```

2. In /backend/main.py → Add endpoint:

```
text

from fastapi import FastAPI, UploadFile
app = FastAPI()
@app.post("/calculate")
async def calc_quote(inputs: dict):
    return calculate_quote(inputs)
```

3. Test locally: `uvicorn main:app --reload` → Use Postman: POST http://localhost:8000/calculate with sample JSON.

Phase 3: AI Parsing for PDF/STEP (3–4 Hours)

1. Install extras: `pip install pypdf2 pillow` (for PDF to image).
2. In /backend/parse_pdf.py → Cmd+K → Prompt:

```
text

Create /parse-pdf endpoint (FastAPI UploadFile).
Convert PDF pages to base64 images using pypdf2 + Pillow.
Send to Claude-4o Vision API: Prompt "Extract JSON: {material: str, tole
Return parsed JSON + flags for low confidence.
```

3. In /backend/parse_step.py → Cmd+K → Prompt:

```
text

Add /parse-step endpoint (UploadFile).
Use python-occ to load STEP file → compute volume.
Return {'volume_mm3': float, 'weight_lbs': float (using material density
Install python-occ-core if needed.
```

4. Test: POST file to endpoints → Verify extraction on sample PDF (e.g., material = "6061", tolerance = "±0.001").

Phase 4: Frontend – Form & Upload (2 Hours)

1. In frontend/app/quotes/new/page.tsx → Cmd+K → Prompt:

```
text

Build form with shadcn/ui: Material dropdown, quantity input, bounding b
Add drag-drop for PDF/STEP (shadcn dropzone). On upload, call backend /p
Show confidence highlights (red for <85%).
```

2. On submit → Call /calculate → Display breakdown.

Phase 5: PDF Generation (1 Hour)

1. Install: `npm install @react-pdf/renderer`.
2. In frontend/components/QuotePDF.tsx → Cmd+K → Prompt:

```
text

Use @react-pdf/renderer to generate PDF:
- Shop logo (placeholder image)
- Line-item table from breakdown JSON
- Total, lead time, validity
- QR code to /q/[id] (use qr-code lib)
Create API route /api/generate-pdf → return Blob for download.
```

3. Test: Render PDF in browser preview.

Phase 6: Customer Portal & Accept (2 Hours)

1. In frontend/app/q/[quoteId]/page.tsx → Cmd+K → Prompt:

```
text

Public page (no auth): Fetch quote from Supabase by ID.
Embed PDF preview.
"Accept Quote" button → Form for PO number + optional 10% deposit (Strip
On accept: Call backend to generate PO PDF + email shop via Resend.
```

2. Backend /accept-quote:
   Prompt: "Endpoint to create PO PDF (using reportlab or pdfkit) + send email with Resend (HTML template + attachment)."
   Install: `pip install resend reportlab`.

Phase 7: Dashboard & Settings (1–2 Hours)

1. In frontend/app/dashboard/page.tsx → Cmd+K → Prompt:

```
text

Use Recharts: Bar chart quotes/mo, line chart win rate, pie chart avg ma
Query Supabase for data.
Add CSV export button (papaparse).
```

2. In frontend/app/settings/page.tsx → Cmd+K → Prompt:

```
text

Form to edit hourly rate, margin %, upload logo.
Also editable table for all 78 rules (use react-table).
Save to Supabase user_meta.
```

Phase 8: Stripe Billing (1 Hour)

1. Install: `npm install stripe @stripe/stripe-js`.
2. In frontend/app/billing/page.tsx → Cmd+K → Prompt:

```
text

Integrate Stripe Checkout: Buttons for $99 Starter, $299 Pro.
Create session in backend /create-checkout-session (FastAPI).
Webhook /webhook for fulfill (update Supabase user plan).
Use test mode first.
```

3. Backend webhook: Verify signature + update user.

Phase 9: Polish & Deploy (1 Day)

1. Responsive: Add Tailwind mobile classes.
2. Notifications: Prompt "Add Resend emails for quote sent/accepted."
3. Test E2E: Upload sample → Parse → Quote → Accept → Email.
4. Deploy:

```
Bash

# Frontend
cd frontend && vercel --prod  # Connect GitHub for auto-deploys

# Backend
Create Render Web Service → GitHub /backend → Python → free tier → add e
```

5. Domain: Point quoteforge.io to Vercel (CNAME + A records).

Final Notes

- All testing: pytest (backend), vitest (frontend).
- Security: Supabase RLS policies (prompt to generate).
- Costs: <$50/mo total.

This is 100% copy-paste – every prompt generates 80–90% of the code.
Start with Phase 1 now.