AGENTIC QUOTATION MINI-ASSESSMENT (BAKERY)

GOAL

Build a small **agent** that proactively gathers job details from a user and then **orchestrates three simple tools** to produce a professional quotation document for a bakery order.

Tools/resources to integrate:

1) **SQL material-costs DB**
2) **Pricing/BOM tool API** (very simple)
3) **Quotation template** (fill and export)

The agent should ask missing details, call tools, calculate prices, and output a ready-to-send quote.

WHAT YOU ARE GIVEN:

- **SQLite** file with schema + provided seed data.
- **Minimal HTTP BOM API** (a tiny local server you can run via *docker compose up --build*)
- **Quotation template** file with placeholders.

WHAT YOU WILL DELIVER

- A runnable agent (CLI or simple chat UI) that:
  - Greets and **asks for missing info** (job type, quantity, due date, customer/company details, currency, VAT rate).
  - Calls the **BOM API** to get required materials + labor.
  - Queries the **SQL DB** for unit prices of materials.
  - Calculates costs, applies markup and VAT, **fills the quotation template**, and saves the result (Markdown or PDF).
- A short **README** (how to run, assumptions)

Keep it small and readable. Any language/runtime is fine.

RESOURCE 1. SQL MATERIAL-COSTS DATABASE

A file will be provided for convenience but you can use **SQLite** with this schema and seed rows:

```
CREATE TABLE materials (
  id INTEGER PRIMARY KEY,
  name TEXT UNIQUE NOT NULL,
  unit TEXT NOT NULL,          -- 'kg', 'L', or 'each', 'ml' (if you need it)
  unit_cost REAL NOT NULL,     -- price per unit
  currency TEXT NOT NULL,      -- e.g., 'GBP'
  last_updated TEXT NOT NULL   -- ISO date
);

INSERT INTO materials (name, unit, unit_cost, currency, last_updated) VALUES
  ('flour', 'kg', 0.90, 'GBP', '2025-09-01'),
  ('sugar', 'kg', 0.70, 'GBP', '2025-09-01'),
  ('butter', 'kg', 4.50, 'GBP', '2025-09-01'),
  ('eggs', 'each', 0.18, 'GBP', '2025-09-01'),
  ('milk', 'L', 0.60, 'GBP', '2025-09-01'),
  ('cocoa', 'kg', 6.00, 'GBP', '2025-09-01'),
  ('vanilla', 'ml', 0.05, 'GBP', '2025-09-01'),
  ('baking_powder', 'kg', 3.00, 'GBP', '2025-09-01'),
  ('salt', 'kg', 0.40, 'GBP', '2025-09-01'),
  ('yeast', 'kg', 2.50, 'GBP', '2025-09-01');
```

CONVERSION RULES TO IMPLEMENT

- grams ↔ kilograms: 1 g = 0.001 kg
- ml ↔ L: 1000 ml = 1 L

RESOURCE 2. PRICING/BOM TOOL API

A tiny HTTP API that **only returns materials per unit** and **labor hours** for a job and can scale by quantity.

Three job types: `cupcakes`, `cake`, `pastry_box`.

ENDPOINTS

- `GET /job-types` → `["cupcakes","cake","pastry_box"]`
- `POST /estimate`
  - **Request**

    ```
    {
      "job_type": "cupcakes",
      "quantity": 24
    }
    ```
  - **Response**

    ```
    {
      "job_type": "cupcakes",
      "quantity": 24,
      "materials": [
        {"name":"flour","unit":"kg","qty":1.92},
    ```

```
        {"name":"sugar","unit":"kg","qty":1.44},
        {"name":"butter","unit":"kg","qty":0.96},
        {"name":"eggs","unit":"each","qty":12.0},
        {"name":"milk","unit":"L","qty":1.2},
        {"name":"vanilla","unit":"ml","qty":24.0},
        {"name":"baking_powder","unit":"kg","qty":0.024}
      ],
      "labor_hours": 1.2
    }
```

RESOURCE 3. QUOTATION TEMPLATE

You are given a `quote_template.md` with placeholders (Mustache-style or your own simple `${placeholder}`).

```
# {{company_name}} — Quotation

**Quote ID:** {{quote_id}}
**Date:** {{quote_date}}
**Valid Until:** {{valid_until}}
**Customer:** {{customer_name}}
**Project:** {{job_type}} × {{quantity}}
**Delivery / Due:** {{due_date}}

## Bill of Materials & Labor
| Item | Qty | Unit | Unit Cost ({{currency}}) | Line Cost |
|---|---:|:---:|---:|---:|
{{#lines}}
| {{name}} | {{qty}} | {{unit}} | {{unit_cost}} | {{line_cost}} |
{{/lines}}
| **Labor (@ {{labor_rate}}/h)** | {{labor_hours}} | h | — | {{labor_cost}} |

**Materials Subtotal:** {{materials_subtotal}} {{currency}}
**Labor Subtotal:** {{labor_cost}} {{currency}}
**Subtotal (pre-markup):** {{subtotal}} {{currency}}
**Markup ({{markup_pct}}):** {{markup_value}} {{currency}}
**Price before VAT:** {{price_before_vat}} {{currency}}
**VAT ({{vat_pct}}):** {{vat_value}} {{currency}}
**Total:** **{{total}} {{currency}}**

**Notes:** {{notes}}

---
```

`*Thank you for your business!*`

Output can be the filled Markdown (and optionally a PDF export).

PRICING LOGIC (TO IMPLEMENT IN THE AGENT)

1) Call BOM API → get `materials[]` and `labor_hours`.
2) For each material:
   - Look up `unit_cost` in SQLite; ensure unit compatibility via the conversion rules.
   - `line_cost = qty * converted_unit_cost`.
3) `materials_subtotal = Σ line_cost`.
4) `labor_cost = labor_hours * labor_rate`.
5) `subtotal = materials_subtotal + labor_cost`.
6) Apply `markup_pct` (e.g., 30%): `price_before_vat = subtotal * (1 + markup_pct)`.
7) Apply `vat_pct` (e.g., 20%): `total = price_before_vat * (1 + vat_pct)`.
8) Derive a **unit price** if helpful: `unit_price = total / quantity`.

CONFIGURABLE PARAMETERS (WITH SANE DEFAULTS)

- `labor_rate`: **15.00 GBP/hour**
- `markup_pct`: **30%**
- `vat_pct`: **20%**
- `currency`: **GBP**

  These can be overridden via user input or a `.env` file.

AGENT BEHAVIOR REQUIREMENTS

- **Proactive Q&A**: If anything is missing, the agent asks (job type, quantity, due date, customer/company, email, currency, VAT).
- **Validation**: Ensure job type ∈ {cupcakes, cake, pastry_box}; quantity is a positive integer.
- **Tool use**:
  - Call **/estimate** with job type & quantity.
  - Query **SQLite** for each material's unit price (single SQL SELECT per material or an IN query).
  - Load and fill **quote_template.md**; save to `out/quote_<id>.md` (and optional PDF). (Hint: use a library like *chevron* for this)
- **Transparency**: Show a short calculation summary in the chat before writing the file.
- **Graceful errors**: If a material is missing from the DB, report it and suggest adding it.

SUGGESTED PLAN OF ATTACK:

Expose each of the three resources an agentic tool or equivalent abstraction:

- `db_query(sql: string, params?: any[]) -> rows`
- `bom_estimate(job_type: string, quantity: number) -> {materials[], labor_hours}`
- `template_render(template_path: string, data: object) -> markdown_string`

Agent plan (sketch):

1. Collect/validate inputs → confirm summary with user.
2. `bom.estimate(...)` → get BOM.
3. `db.query(...)` to fetch unit costs for all `materials.name IN (...)`.
4. Compute totals → build `data` map.
5. `templater.render(...)` → write `out/quote_<id>.md`.
6. Return path + brief summary in chat.

README NOTES TO INCLUDE

- How to run (one command).
- How to change default rates/percentages.
- How to add a new material or job type.
- Limitations

STRETCH (OPTIONAL)

- Export PDF.
- Currency switch with live FX (mock OK).
- Basic “order number” generation and storage.