Backend & Database Implementation Guide

Follow these steps to build the Python (FastAPI) + PostgreSQL + Gemini architecture.

Phase 1: Database Setup (PostgreSQL)

You need to create the database and enable the "Fuzzy Search" extension (`pg_trgm`) that makes voice search accurate.

1. **Install PostgreSQL** (if not installed).
2. **Open your SQL Tool** (pgAdmin or terminal `psql`).
3. **Run these SQL Commands:**

```
-- 1. Create the database
CREATE DATABASE shop_voice_db;

-- 2. Connect to the database
\c shop_voice_db  -- (Only if using terminal, otherwise select DB in pgAdmin)

-- 3. Enable the "Trigram" extension for fuzzy matching
CREATE EXTENSION IF NOT EXISTS pg_trgm;

-- 4. Create the Products Table
CREATE TABLE products (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    keywords TEXT, -- "namak" for Salt, "thanda" for Coke
    price DECIMAL(10,2) NOT NULL,
    stock_qty INTEGER DEFAULT 0
);

-- 5. Create the Ledger/Customers Table
CREATE TABLE customers (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    phone VARCHAR(20),
    balance DECIMAL(10,2) DEFAULT 0.00 -- Positive = They owe you
);

-- 6. Seed some dummy data to test immediately
INSERT INTO products (name, keywords, price, stock_qty) VALUES
('Tata Salt', 'namak salt', 20.00, 50),
('Maggi Noodles', 'maggi snack noodles', 14.00, 100),
('Dove Soap', 'sabun soap body wash', 45.00, 30),
('Fortune Oil', 'tel oil cooking', 150.00, 20);

INSERT INTO customers (name, balance) VALUES
('rahul', 500.00),
('amit', -200.00);
```

Phase 2: Python Project Setup

1. Folder Structure

Create a folder named `shop_backend` . Inside it:

```
shop_backend/
├── .env                <-- Stores your API Keys (Security)
├── main.py             <-- The API code (updated backend.py)
└── requirements.txt    <-- List of libraries
```

2. Create the Virtual Environment

Open your terminal in `shop_backend` and run:

```
# Create virtual environment
python -m venv venv

# Activate it (Windows)
venv\Scripts\activate

# Activate it (Mac/Linux)
source venv/bin/activate
```

3. Install Dependencies

Create a file named `requirements.txt` with this content:

```
fastapi
uvicorn[standard]
sqlalchemy
psycopg2-binary
google-generativeai
python-dotenv
thefuzz
```

Run the install command:

```
pip install -r requirements.txt
```

4. Security (.env)

Create a file named `.env` (no extension) and add your keys:

```
DATABASE_URL=postgresql://postgres:password@localhost:5432/shop_voice_db
GEMINI_API_KEY=your_google_api_key_here
```

Note: Replace `postgres:password` with your actual Postgres username and password.

Phase 3: How to Run

1. **Start the Server:**

```
uvicorn main:app --reload
```

2. **Test with Postman or Browser:**
    - Open `http://127.0.0.1:8000/docs` (FastAPI automatically creates a UI for you).
    - Click on **POST /process-command**.
    - Click **Try it out**.
    - Enter JSON: `{"text": "Add 2 packets of Maggi"}`.
    - Execute and check the response.

Phase 4: Understanding the Search Logic

We use a Hybrid Search approach in the code:

1. **Gemini** extracts the *intent* (e.g., User wants "Maggi").
2. **Python** queries the database for *all* products.
3. **TheFuzz** (Python library) compares the spoken word "Magi" with database "Maggi Noodles" to find the match.

Why not just SQL? While Postgres `pg_trgm` is powerful, setting up SQLAlchemy to use raw SQL `similarity()` functions can be complex for beginners. Using Python's `thefuzz` on the result set is often faster to implement and easier to debug for a shop inventory of <5000 items.