

To set up your project including PostgreSQL database, Qdrant vector database, backend and frontend you should follow these steps:

**Step 1: Install Dependencies**

**System packages (macOS/Linux example)**

# Update package manager and install PostgreSQL, Python3, Node.js, npm/yarn  
brew update  
brew install postgresql  
brew install python  
brew install node

**Python virtual environment & packages**

python3 -m venv venv  
source venv/bin/activate # macOS/Linux  
  
pip install --upgrade pip setuptools wheel  
pip install fastapi uvicorn sqlalchemy psycopg2-binary alembic pydantic  
pip install openai requests  
pip install qdrant-client sentence-transformers

**Node.js frontend dependencies**

# In frontend folder (e.g., inside your Next.js project)  
npm install  
# or  
yarn install

**Step 2: PostgreSQL Setup**

# Start PostgreSQL service  
brew services start postgresql  
  
# Create database user and database (example)  
psql postgres  
  
# In psql shell:  
CREATE USER your\_user WITH PASSWORD 'your\_password';  
CREATE DATABASE your\_db;  
GRANT ALL PRIVILEGES ON DATABASE your\_db TO your\_user;  
\q

Create tables based on your SQLAlchemy models. If you use Alembic for migration, run:

# Initialize Alembic (if not already)  
alembic init alembic  
  
# Configure alembic.ini to your database settings  
  
# Generate migration scripts  
alembic revision --autogenerate -m "Initial migration"  
  
# Apply migrations  
alembic upgrade head

Or alternatively, directly create tables with SQLAlchemy in a Python script:

from app.database import Base, engine  
  
# Import all models so they get registered with Base  
from app.models import product, order, ...  
  
Base.metadata.create\_all(bind=engine)

**Step 3: Qdrant Vector Database Setup**

* Download and run Qdrant server (locally or use hosted service)

# Using docker  
docker run -p 6333:6333 qdrant/qdrant  
  
# Or install standalone from https://qdrant.tech/documentation/quick-start

* Configure .env or config variables:

QDRANT\_HOST=localhost  
QDRANT\_PORT=6333  
QDRANT\_API\_KEY= # if applicable

Your backend service (VectorService) will automatically create collections if missing.

**Step 4: Backend Setup**

* Configure environment variables (.env) for:

DATABASE\_URL=postgresql+psycopg2://your\_user:your\_password@localhost/your\_db  
QDRANT\_HOST=localhost  
QDRANT\_PORT=6333  
QDRANT\_API\_KEY=your\_key\_if\_any  
SHOPIFY\_API\_SETTINGS=... # your shopify credentials  
OPENAI\_API\_KEY=your\_openai\_key  
OPENAI\_MODEL=gpt-4o-mini

* Start backend API server

uvicorn app.main:app --reload

**Step 5: Frontend Setup**

* Navigate to frontend folder (Next.js React app)

npm run dev  
# or  
yarn dev

* Connect to backend API as per your React/Next.js code.

**Summary of key commands:**

|  |  |
| --- | --- |
| Purpose | Command |
| Install PostgreSQL | brew install postgresql |
| Start PostgreSQL service | brew services start postgresql |
| Create PostgreSQL user/db | Via psql shell: CREATE USER, CREATE DATABASE |
| Setup Alembic migrations | alembic revision --autogenerate, alembic upgrade head |
| Install Python packages | pip install -r requirements.txt |
| Run Qdrant (Docker) | docker run -p 6333:6333 qdrant/qdrant |
| Run backend dev server | uvicorn app.main:app --reload |
| Run frontend dev server | npm run dev or yarn dev |

If you want, I can prepare a detailed [README.md](http://README.md) file for your repo with all setup instructions and commands consolidated for easy onboarding. Would you like me to do that?

⁂

1. [order.py](http://order.py)
2. ChatInput.tsx
3. ChatMessage.tsx
4. OrderCard.tsx
5. ProductCard.tsx
6. page.tsx
7. data\_sync.py
8. openai\_service.py
9. shopify\_service.py
10. vector\_service.py
11. [product.py](http://product.py)
12. data\_sync.py
13. openai\_service.py
14. vector\_service.py
15. [chat.py](http://chat.py)
16. [orders.py](http://orders.py)
17. [products.py](http://products.py)
18. ChatBot.tsx