ORAMA – FILE

, here’s a streamlined backend prototype plan for your **LinkedIn Sales Automation Tool** — complete from setup to deployment.

**🎯 Simplified USP (Still Powerful)**

**“Event-Triggered Outreach + Personality-Tailored Messaging”**

* Uses NLP to analyze profile tone and recent activity.
* Matches outreach tone to inferred personality.
* Triggers outreach based on events like job change or funding.
* Automates sequences with human-like pacing.

**🧰 Simplified Tech Stack**

| **Layer** | **Tech Used** | **Why It Works** |
| --- | --- | --- |
| Backend | **FastAPI** | Lightweight, async-ready |
| Database | **SQLite** (for prototype) | Easy local setup |
| AI & NLP | **OpenAI GPT-4**, **spaCy** | Profile analysis + message generation |
| Scraping | **BeautifulSoup** (demo) | Simpler than Puppeteer |
| Scheduling | **APScheduler** | Lightweight cron alternative |
| Deployment | **Render** or **Heroku** | Free, fast deployment |
| Auth | **Manual token input** | Skip OAuth for now |

**🗂️ Project Structure**

linkedin\_automation/

├── app/

│ ├── main.py

│ ├── config.py

│ ├── database.py

│ ├── models.py

│ ├── schemas.py

│ ├── routers/

│ │ ├── campaign.py

│ │ ├── outreach.py

│ │ └── dashboard.py

│ ├── services/

│ │ ├── profile\_analyzer.py

│ │ ├── disc\_classifier.py

│ │ ├── message\_generator.py

│ │ └── scheduler.py

│ └── utils/

│ └── scraper.py

├── requirements.txt

├── .env

└── README.md

**🧱 Setup Instructions**

**1. Install Tools**

* Python 3.10+
* VS Code
* Extensions: Python, REST Client, Prettier

**2. Create Virtual Environment**

python -m venv venv

source venv/bin/activate

**3. Install Dependencies**

pip install fastapi uvicorn openai spacy apscheduler sqlalchemy python-dotenv beautifulsoup4

python -m spacy download en\_core\_web\_sm

**🧬 Core Modules**

I’ll now generate the full content of each file so you can copy-paste directly into VS Code. Let’s start with:

**✅ main.py**

from fastapi import FastAPI

from app.routers import campaign, outreach, dashboard

app = FastAPI(title="LinkedIn Sales Automation Tool")

app.include\_router(campaign.router)

app.include\_router(outreach.router)

app.include\_router(dashboard.router)

**✅ config.py**

import os

from dotenv import load\_dotenv

load\_dotenv()

OPENAI\_API\_KEY = os.getenv("OPENAI\_API\_KEY")

**✅ .env**

OPENAI\_API\_KEY=your\_openai\_key\_here

**✅ database.py**

from sqlalchemy import create\_engine

from sqlalchemy.orm import sessionmaker, declarative\_base

SQLALCHEMY\_DATABASE\_URL = "sqlite:///./linkedin.db"

engine = create\_engine(SQLALCHEMY\_DATABASE\_URL, connect\_args={"check\_same\_thread": False})

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative\_base()

**✅ models.py**

from sqlalchemy import Column, Integer, String, JSON

from app.database import Base

class Campaign(Base):

\_\_tablename\_\_ = "campaigns"

id = Column(Integer, primary\_key=True, index=True)

product = Column(String)

target\_industry = Column(String)

job\_roles = Column(JSON)

company\_size = Column(String)

region = Column(String)

goal = Column(String)

brand\_voice = Column(String)

triggers = Column(JSON)

**✅ schemas.py**

from pydantic import BaseModel

from typing import List, Optional

class CampaignInput(BaseModel):

product: str

target\_industry: str

job\_roles: List[str]

company\_size: str

region: str

goal: str

brand\_voice: str

triggers: Optional[List[str]] = []

Would you like me to continue with the routers and services next? I’ll also include deployment instructions once the code is complete. Let’s build this step-by-step so it’s easy to follow and run locally.