ORAMA – FILE

Forntend – All detail

**🧰 Tech Stack Summary**

* **React (Vite)** – Fast and modular frontend
* **Tailwind CSS** – Clean styling
* **Axios** – API calls
* **React Router** – Page navigation
* **No Redux** – Uses simple React state/context

**🗂️ Folder Structure**

frontend/

├── index.html

├── package.json

├── tailwind.config.js

├── postcss.config.js

├── vite.config.js

├── public/

├── src/

│ ├── App.jsx

│ ├── main.jsx

│ ├── index.css

│ ├── pages/

│ │ ├── Home.jsx

│ │ ├── Dashboard.jsx

│ │ └── Outreach.jsx

│ ├── components/

│ │ ├── CampaignForm.jsx

│ │ └── ProspectCard.jsx

│ └── api.js

**🧱 Step-by-Step Setup**

**1. Create Project**

npm create vite@latest frontend -- --template react

cd frontend

npm install

**2. Install Dependencies**

npm install axios react-router-dom tailwindcss postcss autoprefixer

npx tailwindcss init -p

**⚙️ Configuration**

**tailwind.config.js**

export default {

content: ["./index.html", "./src/\*\*/\*.{js,jsx}"],

theme: {

extend: {},

},

plugins: [],

}

**src/index.css**

@tailwind base;

@tailwind components;

@tailwind utilities;

**🚀 Full Code Files**

**✅ src/main.jsx**

import React from 'react'

import ReactDOM from 'react-dom/client'

import App from './App.jsx'

import './index.css'

ReactDOM.createRoot(document.getElementById('root')).render(

<React.StrictMode>

<App />

</React.StrictMode>,

)

**✅ src/App.jsx**

import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'

import Home from './pages/Home'

import Dashboard from './pages/Dashboard'

import Outreach from './pages/Outreach'

function App() {

return (

<Router>

<div className="min-h-screen bg-gray-100 p-4">

<h1 className="text-2xl font-bold mb-4">LinkedIn Sales Automation</h1>

<Routes>

<Route path="/" element={<Home />} />

<Route path="/dashboard" element={<Dashboard />} />

<Route path="/outreach" element={<Outreach />} />

</Routes>

</div>

</Router>

)

}

export default App

**✅ src/pages/Home.jsx**

import CampaignForm from '../components/CampaignForm'

export default function Home() {

return (

<div>

<h2 className="text-xl mb-2">Create Campaign</h2>

<CampaignForm />

</div>

)

}

**✅ src/pages/Dashboard.jsx**

import { useEffect, useState } from 'react'

import axios from 'axios'

export default function Dashboard() {

const [metrics, setMetrics] = useState({})

useEffect(() => {

axios.get('http://localhost:8000/dashboard/metrics')

.then(res => setMetrics(res.data))

}, [])

return (

<div>

<h2 className="text-xl mb-2">Campaign Metrics</h2>

<ul className="list-disc ml-6">

<li>Connections Sent: {metrics.connections\_sent}</li>

<li>Replies Received: {metrics.replies\_received}</li>

<li>Conversion Rate: {metrics.conversion\_rate}</li>

</ul>

</div>

)

}

**✅ src/pages/Outreach.jsx**

import ProspectCard from '../components/ProspectCard'

const sampleProspects = [

{

name: "Anjali Mehta",

role: "HR Manager",

company: "HirePulse",

message: "Hi Anjali, loved your post on hybrid hiring..."

}

]

export default function Outreach() {

return (

<div>

<h2 className="text-xl mb-2">Top Prospects</h2>

<div className="space-y-4">

{sampleProspects.map((p, i) => (

<ProspectCard key={i} prospect={p} />

))}

</div>

</div>

)

}

**✅ src/components/CampaignForm.jsx**

import { useState } from 'react'

import axios from 'axios'

export default function CampaignForm() {

const [formData, setFormData] = useState({

product: '',

target\_industry: '',

job\_roles: '',

company\_size: '',

region: '',

goal: '',

brand\_voice: '',

triggers: ''

})

const handleChange = (e) => {

setFormData({...formData, [e.target.name]: e.target.value})

}

const handleSubmit = async (e) => {

e.preventDefault()

const payload = {

...formData,

job\_roles: formData.job\_roles.split(','),

triggers: formData.triggers.split(',')

}

await axios.post('http://localhost:8000/campaign/create', payload)

alert("Campaign created!")

}

return (

<form onSubmit={handleSubmit} className="space-y-4">

{["product", "target\_industry", "job\_roles", "company\_size", "region", "goal", "brand\_voice", "triggers"].map(field => (

<input

key={field}

name={field}

placeholder={field.replace('\_', ' ')}

onChange={handleChange}

className="w-full p-2 border rounded"

/>

))}

<button type="submit" className="bg-blue-600 text-white px-4 py-2 rounded">Submit</button>

</form>

)

}

**✅ src/components/ProspectCard.jsx**

export default function ProspectCard({ prospect }) {

return (

<div className="border p-4 rounded bg-white shadow">

<h3 className="font-bold">{prospect.name}</h3>

<p>{prospect.role} @ {prospect.company}</p>

<p className="mt-2 text-sm text-gray-700">{prospect.message}</p>

</div>

)

}

**🚀 Run Locally**

npm run dev

Visit: <http://localhost:5173>

Make sure your FastAPI backend is running at http://localhost:8000.

**🌐 Deploy Frontend**

**Option 1: Vercel**

* Push to GitHub
* Go to [vercel.com](https://vercel.com/)
* Import repo → Deploy

**Option 2: Netlify**

* Push to GitHub
* Go to [netlify.com](https://www.netlify.com/)
* Connect repo → Deploy

Would you like me to help you connect the backend scraper or message generator next? Or scaffold the frontend dashboard with live campaign tracking?

, 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.