## **Dev 2 – AI Agent Logic: Starter Checklist**

### **1. Project Setup**

* Create a new repo: synthetix-ai-agent
* Set up a virtual environment (venv or poetry)

Install core libraries:  
  
 bash  
CopyEdit  
pip install openai langchain fastapi pydantic python-dotenv

### **2. Core Logic – GPT or LangChain**

* Set up GPT-4 call using OpenAI SDK or LangChain
* Define intents: booking, rescheduling, general inquiries

Write a prompt template:  
  
 text  
CopyEdit  
You are a helpful AI assistant for booking appointments. Extract user name, service, preferred date and time.

### **3. Entity Extraction (Initial MVP)**

* Parse model response to extract:  
  + name
  + service
  + date
  + time
* Validate fields (e.g., check date format, time availability)

### **4. FastAPI Endpoint (to connect with n8n or frontend)**

python

CopyEdit

from fastapi import FastAPI, Request

from pydantic import BaseModel

app = FastAPI()

class MessageRequest(BaseModel):

message: str

@app.post("/process-message/")

async def process\_message(data: MessageRequest):

# Call GPT or LangChain logic here

return {

"name": "John",

"service": "Massage",

"date": "2025-04-15",

"time": "2:00 PM"

}

### **5. Memory & Context (Optional Phase 2)**

* Use LangChain’s ConversationBufferMemory or custom session storage
* Keep track of previous messages for better flow

### **6. Integration Points**

* Sync with Dev 1 for sending data back to n8n via webhook or API
* Share API docs or request/response structure

**GitHub-ready starter repo**

**Dev 2 – AI Agent Logic using FastAPI + LangChain + OpenAI.**

## **✅ Project Name: synthetix-ai-agent**

### **📁 Folder Structure:**

**arduino**

**CopyEdit**

**synthetix-ai-agent/**

**├── app/**

**│ ├── main.py**

**│ ├── agent.py**

**│ ├── schemas.py**

**│ ├── config.py**

**│ └── utils.py**

**├── .env**

**├── requirements.txt**

**├── README.md**

**└── run.sh**

### **🧠 app/agent.py**

**Handles LangChain logic.**

**python**

**CopyEdit**

**from langchain.chat\_models import ChatOpenAI**

**from langchain.prompts import PromptTemplate**

**from langchain.chains import LLMChain**

**import os**

**llm = ChatOpenAI(model="gpt-4", temperature=0.2)**

**prompt\_template = PromptTemplate.from\_template("""**

**You are a booking assistant. Extract name, service, date, and time from this message:**

**"{user\_input}"**

**Respond in this format:**

**Name: <name>**

**Service: <service>**

**Date: <date>**

**Time: <time>**

**""")**

**chain = LLMChain(llm=llm, prompt=prompt\_template)**

**def extract\_booking\_info(user\_input: str) -> dict:**

**response = chain.run(user\_input)**

**lines = response.strip().split("\n")**

**return {line.split(":")[0].lower(): line.split(":")[1].strip() for line in lines if ":" in line}**

### **🚀 app/main.py**

**API endpoint.**

**python**

**CopyEdit**

**from fastapi import FastAPI**

**from app.schemas import MessageRequest, BookingResponse**

**from app.agent import extract\_booking\_info**

**app = FastAPI()**

**@app.post("/process-message/", response\_model=BookingResponse)**

**async def process\_message(payload: MessageRequest):**

**result = extract\_booking\_info(payload.message)**

**return result**

### **📦 app/schemas.py**

**Input/output models.**

**python**

**CopyEdit**

**from pydantic import BaseModel**

**class MessageRequest(BaseModel):**

**message: str**

**class BookingResponse(BaseModel):**

**name: str**

**service: str**

**date: str**

**time: str**

### **🔧 app/config.py**

**Reads your OpenAI key.**

**python**

**CopyEdit**

**import os**

**from dotenv import load\_dotenv**

**load\_dotenv()**

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

### **📄 .env**

**env**

**CopyEdit**

**OPENAI\_API\_KEY=sk-xxxxxxxxxxxxxxxx**

### **📋 requirements.txt**

**txt**

**CopyEdit**

**fastapi**

**langchain**

**openai**

**uvicorn**

**pydantic**

**python-dotenv**

### **▶️ run.sh**

**bash**

**CopyEdit**

**#!/bin/bash**

**uvicorn app.main:app --reload**

### **📘 README.md (short example)**

**md**

**CopyEdit**

**# Synthetix AI Agent**

**A FastAPI-based backend using LangChain & OpenAI to extract booking data from natural language.**

**## Setup**

**```bash**

**pip install -r requirements.txt**

**Add your OpenAI API key to .env, then run:**

**bash**

**CopyEdit**

**sh run.sh**

**Send a POST request to:**

**arduino**

**CopyEdit**

**POST /process-message/**

**{**

**"message": "Hi, I want to book a massage with John on Tuesday at 2 PM"**

**}**

**LangChain sample logic with memory + fallback handling**