# **📑 Requirements Document**

**Project:** AI Audience Agent

**Tech:** LangGraph + LangSmith + FastAPI

**Scope:** Parse user prompts (English & Arabic) into structured filters for campaign audience creation

## **1. Objective**

Build an **AI-powered agent** that converts **natural language prompts** into **structured audience filters**.

* Agent orchestrated with **LangGraph**.
* Observability & tracing via **LangSmith**.
* Deployed as a **FastAPI application**.
* Supports prompts in **Arabic and English**.

## **2. Input → Output**

### **Input**

Example user prompt:

* **EN:** *“Find customers who joined after Jan 2023 with more than 5 orders and store rating between 3 and 5.”*
* **AR:** *“اعثر على العملاء الذين انضموا بعد يناير 2023 ولديهم أكثر من 5 طلبات وتقييم المتجر بين 3 و5.”*

### **Output**

Structured JSON filter:

{

"filters": [

{ "field": "joining\_date", "operator": ">", "value": "2023-01-01" },

{ "field": "total\_orders", "operator": ">", "value": 5 },

{ "field": "store\_rating", "operator": "between", "value": [3, 5] }

]

}

## **3. Supported Fields**

### **Customer Attributes**

* gender
* birthday
* birthday\_days (relative, e.g., “in next 7 days”)
* joining\_date
* last\_login

### **Behavioral Attributes**

* doesnt\_have\_orders
* have\_cancelled\_orders
* latest\_purchase

### **Sales & Engagement Metrics**

* total\_sales
* total\_orders
* store\_rating

### **Contact Attributes**

* doesnt\_have\_email

### **Geographic Attributes**

* country
* city

## **4. Supported Operators**

* **Equality / Negation**
  + = (equals)
  + != (not equals)
* **Comparisons**
  + < (less than)
  + > (greater than)
  + <= (less than or equal to)
  + >= (greater than or equal to)
  + between (numeric or date ranges)

## **5. Supported Value Types**

* **Integers** → 5, 1000
* **Floats** → 3.5, 4.2
* **Dates** → YYYY-MM-DD, relative (e.g., “last 30 days”)
* **Strings** → "Male", "Riyadh"
* **Lists** → "Riyadh or Jeddah" → ["Riyadh", "Jeddah"]

## **6. Agent Architecture**

### **LangGraph Orchestration**

* **Input Node** → receives user prompt
* **Parsing Node** → LLM extracts filters
* **Validation Node** → checks field/operator support, normalizes values
* **Output Node** → returns JSON filters

### **LangSmith Observability**

* Trace every step: input → parse → validation → output
* Log: raw prompt, extracted filters, errors
* Dashboard for debugging and replay

## **7. Error Handling**

* **Unsupported field** →  
    
   *“The field ‘email\_open\_rate’ is not supported. Please use one of: [list of supported fields].”*
* **Missing value** →  
    
   *“Please provide a value for total\_sales (e.g., greater than 1000).”*
* **Ambiguous date** →  
    
   *“Do you mean joining date after 2023-01-01 or in the last 30 days?”*
* **Invalid operator** →  
    
   Normalize if possible, else reject with guidance.

## **8. Testing Dataset**

### **Structure**

Each test case contains:

* id
* prompt\_en
* prompt\_ar
* expected\_output

### **Examples**

#### **Equality**

{

"id": 1,

"prompt\_en": "Find female customers",

"prompt\_ar": "اعثر على العملاء الإناث",

"expected\_output": {

"filters": [

{ "field": "gender", "operator": "=", "value": "female" }

]

}

}

#### **Greater Than**

{

"id": 2,

"prompt\_en": "Customers with more than 10 orders",

"prompt\_ar": "العملاء الذين لديهم أكثر من 10 طلبات",

"expected\_output": {

"filters": [

{ "field": "total\_orders", "operator": ">", "value": 10 }

]

}

}

#### **Between**

{

"id": 3,

"prompt\_en": "Stores rated between 3 and 5 stars",

"prompt\_ar": "المتاجر التي تقييمها بين 3 و 5 نجوم",

"expected\_output": {

"filters": [

{ "field": "store\_rating", "operator": "between", "value": [3, 5] }

]

}

}

👉 Full dataset: **50–100 test cases** covering all fields, operators, and edge cases, in **both English and Arabic**.

## **9. API Deployment (FastAPI)**

### **Endpoint**

POST /parse\_prompt

### **Request**

{

"prompt": "Find customers in Riyadh with more than 500 sales"

}

### **Response**

{

"filters": [

{ "field": "city", "operator": "=", "value": "Riyadh" },

{ "field": "total\_sales", "operator": ">", "value": 500 }

]

}

### **Deployment Plan**

* **Dev**: Run via uvicorn main:app --reload
* **Prod**:  
  + Containerize with Docker
  + Deploy via AWS ECS / GCP Cloud Run / Azure App Service
  + Secure with HTTPS + API Gateway

## **10. Validation Plan**

* Run automated tests (pytest) with bilingual dataset.
* Measure accuracy: ≥ 90% prompts parsed correctly.
* Track parsing errors in LangSmith.
* Continuous improvement from satisfaction logs & error reports.

## **11. Success Criteria**

* **Accuracy:** ≥ 90% of prompts parsed successfully.
* **Performance:** API response < 2s average.
* **Coverage:** All supported fields/operators validated in test dataset.
* **Observability:** 100% of requests traced in LangSmith.