**🧩 TDA-10 — OpenAI Service with Structured JSON Output (Final Implementation Summary)**

**Epic:** AI-Powered Classification & Verification  
**Status:** ✅ *Done — committed and verified end-to-end*  
**Environment:** Local FastAPI (Backend), PostgreSQL (Supabase), OpenAI GPT-4.1-mini

**Overview**

We implemented and verified a production-grade OpenAIService that enforces structured JSON output from the OpenAI API, validates it against a strict Pydantic schema, retries on invalid responses, and logs every interaction to the ai\_logs table in PostgreSQL.

This service now provides the foundation for all AI-based classification and enrichment tasks (TDA-11+).

**Key Deliverables**

**1️⃣ OpenAIService (services/openai\_service.py)**

* Handles full lifecycle of AI calls:
  + JSON-only output enforcement using response\_format={"type": "json\_object"}.
  + Pydantic v2 validation with model\_validate.
  + Exponential backoff retry on JSONDecodeError / ValidationError.
  + Robust \_extract\_first\_json() fallback parser.
  + Runtime check require\_openai() ensures environment key is loaded.
* All OpenAI SDK objects are converted to plain JSON via \_to\_jsonable() to avoid serialization errors.
* Logs every call (success & failure) via ai\_log() into the ai\_logs table with:
* location\_id
* action\_type
* prompt (TEXT)
* raw\_response (JSONB)
* validated\_output (JSONB)
* model\_used
* is\_success
* error\_message
* created\_at
* Tested and confirmed to write clean, serializable data to the database.

**2️⃣ Database Logging (services/db\_service.py)**

* Self-contained async SQLAlchemy engine built from settings.DATABASE\_URL (no external dependency).
* Async insert into the ai\_logs table using exact column names from schema:
* id bigint
* location\_id bigint
* action\_type text
* prompt text
* raw\_response jsonb
* validated\_output jsonb
* model\_used text
* is\_success boolean
* error\_message text
* created\_at timestamptz
* Synchronous façade (ai\_log(...)) automatically handles async contexts.
* Confirmed data consistency:
* | id | action\_type | model\_used | is\_success | created\_at |
* |----|----------------|--------------|-------------|--------------------------------|
* | 2 | classify\_demo | gpt-4.1-mini | true | 2025-10-12 22:23:36.233033+00 |
* | 1 | classify\_demo | gpt-4.1-mini | false | 2025-10-12 22:18:48.553458+00 |

**3️⃣ Configuration (app/config.py)**

* Deterministic .env loading from absolute path /Backend/.env using dotenv + pydantic-settings.
* Pydantic Settings v2 configuration:
  + APP\_VERSION, DATABASE\_URL, OPENAI\_API\_KEY, OPENAI\_MODEL, GOOGLE\_\*.
* Runtime require\_openai() check provides explicit error if missing.

**4️⃣ API Integration**

* **Router:** api/routers/dev\_ai.py
  + /dev/ai/ping — health check.
  + /dev/ai/classify-demo — runs OpenAI classification demo via ClassificationResult schema.
* **Model:** app/models/ai.py
  + Defines ClassificationResult with fields:
  + action: Literal["keep", "ignore"]
  + category: str
  + confidence\_score: float (0–1)
  + reason: Optional[str]
* **Main App:** app/main.py
  + Integrates dev\_ai router with CORS, health endpoints, and exception handlers.

**End-to-End Verification**

|  |  |  |
| --- | --- | --- |
| **Step** | **Action** | **Result** |
| 1 | Start API | ✅ uvicorn app.main:app --reload |
| 2 | Test Ping | ✅ {"ok":true,"service":"dev\_ai"} |
| 3 | Test Classification | ✅ HTTP 200 with structured JSON |
| 4 | DB Write | ✅ is\_success=true logged in ai\_logs |
| 5 | Retry logic test | ✅ first failed run logged with is\_success=false |
| 6 | JSON serialization | ✅ \_to\_jsonable() fixed CompletionTokensDetails serialization issue |

**Example successful response:**

{

"ok": true,

"parsed": {

"action": "keep",

"category": "bakery",

"confidence\_score": 0.9,

"reason": "Kaya is een Turkse naam en bakery kan Turkse producten verkopen."

},

"meta": {

"ok": true,

"model": "gpt-4.1-mini",

"duration\_ms": 1219

}

}

**Acceptance Criteria**

|  |  |  |
| --- | --- | --- |
| **Criterion** | **Description** | **Status** |
| JSON output validated vs schema | Enforced with Pydantic v2 | ✅ |
| Retries on invalid JSON | Exponential backoff implemented | ✅ |
| Logging of all calls | Full DB insert with success/failure | ✅ |
| Structured response model | ClassificationResult used | ✅ |
| Works end-to-end | Verified locally with real OpenAI API | ✅ |

**Definition of Done**

✅ OpenAI integration working  
✅ Retry logic operational  
✅ Comprehensive logging functional  
✅ Tested end-to-end with DB  
✅ Code committed & documented