**TDA-11 – Final Comment**

**Samenvatting**

* AI-classificatie (keep/ignore) voor bakkerijen werkt end-to-end.
* Endpoints:
  + GET /dev/ai/classify-one?id=… → test result
  + POST /dev/ai/classify-apply?id=… → schrijft naar DB (state, category, confidence, notes).
* Batch worker:
  + app.workers.classify\_bot ondersteunt dry-run en drempel voor confidence.
* Evaluatie op gold set:
  + Laatste meting (n=100): **KEEP precision = 1.000**, **Category accuracy = 1.000** ✅
* DB mutaties:
  + keep → state=PENDING\_VERIFICATION, category=bakery, confidence\_score, + reden in notes
  + ignore → state=RETIRED, + reden in notes

**Korte README (runbook)**

1. **Activatie & pad**

cd "/Users/metehankul/Desktop/TurkishProject/Turkish Diaspora App/Backend"

source .venv/bin/activate

1. **Confidence-drempel**

* Optie A (zonder code): meegeven per run  
  --min-confidence 0.80
* Optie B (vaste default): zet in .env  
  CLASSIFY\_MIN\_CONF=0.80  
  en gebruik de kleine parser-default tweak in app/workers/classify\_bot.py.

1. **Dry-run (veilig testen)**

python -m app.workers.classify\_bot --limit 50 --min-confidence 0.80 --dry-run

1. **Echte run**

python -m app.workers.classify\_bot --limit 200 --min-confidence 0.80

1. **Spotcheck na run (SQL)**

SELECT id, name, category, confidence\_score, state, notes

FROM locations

WHERE state IN ('PENDING\_VERIFICATION','RETIRED')

ORDER BY id DESC

LIMIT 50;

1. **Labels importeren (optioneel)**

* CSV moet kolommen hebben: location\_id,name,address,type,label\_action,label\_category,notes

python -m app.workers.import\_training\_labels \

--infile "data/candidates\_to\_label.only\_labeled.with\_notes.csv" \

--only-labeled

1. **Evaluator (kwaliteitscheck)**

python -m app.workers.eval\_classify --limit 100

# Target: KEEP precision >= 0.90

1. **Rollback/correctie (SQL)**

UPDATE locations

SET state='CANDIDATE',

notes = COALESCE(notes,'') || E'\n[manual revert]'

WHERE id = {ID};

**Mini-Sanity Job (voor snelle menselijke review)**  
Kopieer in terminal (in Backend root, venv actief):

python - <<'PY'

import asyncio

from sqlalchemy import text

from services.db\_service import async\_engine

async def main():

async with async\_engine.begin() as conn:

rows = (await conn.execute(text("""

SELECT id, name, address, category, confidence\_score, notes

FROM locations

WHERE state='PENDING\_VERIFICATION'

ORDER BY id DESC

LIMIT 5

"""))).mappings().all()

if not rows:

print("Geen PENDING\_VERIFICATION records.")

return

print("Laatste 5 PENDING\_VERIFICATION:")

for r in rows:

last\_line = (r["notes"] or "").splitlines()[-1] if r["notes"] else "-"

print(f"[{r['id']}] {r['name']} | {r['address']} | cat={r['category']} | conf={r['confidence\_score']}\n note: {last\_line}\n")

asyncio.run(main())

PY

**Huidige status**

* Feature: **gereed**
* Kwaliteit: **≥ target** (KEEP precision 1.000 @ n=100)
* Runbook + sanity: **bijgevoegd**
* Aanbeveling: hanteer **conf drempel 0.80** als standaard.

—

**3) Mini-sanity job meenemen**

Staat hierboven onder de README in de Jira comment (het Python-blok). Het print de laatste 5 PENDING\_VERIFICATION met naam/adres/categorie/confidence en de meest recente regel uit de notes; ideaal om snel te beoordelen of de auto-updates logisch zijn.