API Riko - Panduan Kompatibilitas untuk Herta’s Vibe Checker

Dokumen ini menjelaskan cara membuat Herta’s Vibe Checker tampil sebagai ’API Riko’ supaya bisa langsung dipakai oleh layanan yang mengikuti kontrak API. API Riko berjalan berdamping dengan endpoint asli, jadi tetap bisa memakai fitur vibe-check yang sudah ada.

Beberapa integrasi membutuhkan pola endpoint tertentu (misalnya /register, /oauth/token, /custom-words, /detect). Dengan lapisan kompatibilitas ini, Vibe Checker bisa berbicara dengan mereka tanpa merombak arsitektur yang sudah berjalan.

Endpoint API Riko yang tersedia

- POST /register : mendaftarkan aplikasi; balasan berisi client_id dan client_secret.

- POST /oauth/token : pakai Basic Auth dan grant_type=client_credentials; balasan token JWT.

- POST /custom-words : kelola kata kunci per-klien (action=add/remove, category=blacklist/whitelist).

- POST /detect : kirim {text}; balasan berisi isProfane dan daftar detected_words jika ada.

Mode jalan (pilih sesuai kebutuhan)

1) Mode API asli (math challenge wajib):

```
export JWT_SECRET=change_me

export REQUIRE_MATH_CHALLENGE=true

uvicorn app.main:app --host 0.0.0.0 --port 8000 --proxy-headers
```

2) Mode API Riko (kompat; client_credentials aktif, math challenge opsional):

```
export JWT_SECRET=change_me

export REQUIRE_MATH_CHALLENGE=false

uvicorn app.main:app --host 0.0.0.0 --port 8000 --proxy-headers
```

Catatan: mengganti nilai REQUIRE_MATH_CHALLENGE tidak mematikan endpoint asli. Endpoint vibe check/single, /vibe-check/batch) tetap bisa dipakai pada kedua mode.

Alur cepat (API Riko)

1) Register klien baru:

```
curl -s -X POST http://127.0.0.1:8000/register \
-H ’Content-Type: application/json’ \
-d ’{"name":"AplikasiSaya","email":"dev@example.com"}’
```

2) Ambil token (client_credentials pakai Basic Auth):

```
CID="(client_id)"; CSEC="(client_secret)"; BASIC=$(printf "%s:%s" "$CID" "$CSEC" |
base64)
curl -s -X POST http://127.0.0.1:8000/oauth/token \
-H "Authorization: Basic $BASIC" \
-H "Content-Type: application/x-www-form-urlencoded" \
-d "grant_type=client_credentials"
```

3) Tambahkan blacklist kata:

```
TOKEN="(access_token)"
    curl -s -X POST http://127.0.0.1:8000/custom-words \
    -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
    -d ’{"action":"add","category":"blacklist","words":["badword","heck"]}’
```

4) Deteksi:

```
curl -s -X POST http://127.0.0.1:8000/detect \
-H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
-d ’{"text":"ini mengandung badword tolong tandai"}’
```

Pengujian otomatis (pytest)

Simpan sebagai tests/test_riko_api.py lalu jalankan ’pytest -q tests/test_riko_api.py’.

```
import os, base64
from fastapi.testclient import TestClient
```

```
os.environ.setdefault("DATABASE_URL", "sqlite:///./test_riko_api.db")
os.environ.setdefault("REQUIRE_MATH_CHALLENGE", "false")
from app.main import app
c = TestClient(app)
def b64(cid, csec): return base64.b64encode(f"{cid}:{csec}".encode()).decode()
def test_riko_flow():
r = c.post("/register", json={"name":"RikoCompat","email":"dev@example.com"})
assert r.status_code == 200
cid, csec = r.json()["client_id"], r.json()["client_secret"]
r = c.post("/oauth/token",
headers={"Authorization": f"Basic {b64(cid,csec)}"},
data={"grant_type":"client_credentials"})
assert r.status_code == 200
tok = r.json()["access_token"]; bear = {"Authorization": f"Bearer {tok}"}
c.post("/custom-words", headers=bear,
json={"action":"add","category":"blacklist","words":["badword"]}).raise_for_status()
r = c.post("/detect", headers=bear, json={"text":"ini berisi badword mohon tandai"})
assert r.status_code == 200 and r.json()["isProfane"] is True
r = c.post("/vibe-check/single", headers=bear, json={"text":"Aku suka banget fitur ini!"})
assert r.status_code == 200
```