```
# main.py

from fastapi import FastAPI
from pydantic import BaseModel
from typing import Literal
from datetime import datetime
import uuid

from sqlalchemy import create_engine, Column, String, Float, DateTime
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

# ──────────────────────────────────────────────
#  MySQL 연결 (DB 정보는 자기 걸로 바꿔야 함)
# ──────────────────────────────────────────────
SQLALCHEMY_DATABASE_URL = "mysql+pymysql://root:1234@localhost/factshield"  # <- 이건 예시, 본인 DB 정보 쓰셈

engine = create_engine(SQLALCHEMY_DATABASE_URL)
SessionLocal = sessionmaker(bind=engine)
Base = declarative_base()

# ──────────────────────────────────────────────
#  판별 모델 (임시로 만들어둔 클래스)
# ──────────────────────────────────────────────
class FakeNewsClassifier:
    def __init__(self):
        self.version = "v1.0.0"

    def predict(self, title: str, content: str):
        # 진짜 모델 붙일 때는 여기만 바꾸면 됨
        return {
            "result": "fake",
            "confidence": 0.87,
            "grade": "주의 필요"
        }

# ──────────────────────────────────────────────
#  DB에 들어갈 테이블 구조 정의
# ──────────────────────────────────────────────
class NewsPrediction(Base):
    __tablename__ = "news_predictions"

    id = Column(String(36), primary_key=True, index=True)
    title = Column(String(500))
    content = Column(String(5000))
    result = Column(String(10))
    confidence = Column(Float)
```

```
    grade = Column(String(50))
    model_version = Column(String(20))
    analysis_time = Column(DateTime)

# 테이블이 없으면 자동 생성됨
Base.metadata.create_all(bind=engine)

# ──────────────────────────────────────────────
#  FastAPI 기본 세팅
# ──────────────────────────────────────────────
app = FastAPI()
model = FakeNewsClassifier()

# 요청 형식 정의 (사용자가 입력한 제목 + 본문)
class NewsInput(BaseModel):
    title: str
    content: str

# ──────────────────────────────────────────────
#  API: 뉴스 입력받고 → 모델 판별 → 결과 저장
# ──────────────────────────────────────────────
@app.post("/predict")
def predict(news: NewsInput):
    # 예측
    prediction = model.predict(news.title, news.content)

    # 예측 결과를 DB 테이블 형식에 맞게 정리
    result = NewsPrediction(
        id=str(uuid.uuid4()),
        title=news.title,
        content=news.content,
        result=prediction["result"],
        confidence=prediction["confidence"],
        grade=prediction["grade"],
        model_version=model.version,
        analysis_time=datetime.now()
    )

    # DB 저장
    db = SessionLocal()
    db.add(result)
    db.commit()
    db.refresh(result)
    db.close()

    # 결과 리턴 (프론트로 응답 보내는 부분)
    return {
        "id": result.id,
```

```
    "result": result.result,
    "confidence": result.confidence,
    "grade": result.grade,
    "model_version": result.model_version,
    "analysis_time": result.analysis_time
}
```