Source Code: AI Generated Report

Generated on: 2026-02-07 13:56:50

```
from fastapi import FastAPI, HTTPException, Depends

from fastapi.testclient import TestClient

from pydantic import BaseModel

from sqlalchemy import create_engine, Column, Integer, String, Float, Date

from sqlalchemy.orm import sessionmaker, DeclarativeBase, Session

from sqlalchemy.exc import IntegrityError

import json


DATABASE_URL = "sqlite:///:memory:"


engine = create_engine(DATABASE_URL, connect_args={"check_same_thread": False})

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)


class Base(DeclarativeBase):

    pass


class Transaction(Base):

    __tablename__ = "transactions"

    id = Column(Integer, primary_key=True, index=True)

    date = Column(Date)

    description = Column(String)

    amount = Column(Float)

    category = Column(String)
```

```
class TransactionCreate(BaseModel):

    date: str

    description: str

    amount: float

    category: str


class TransactionResponse(BaseModel):

    id: int

    date: str

    description: str

    amount: float

    category: str


app = FastAPI()


@app.on_event("startup")

def startup():

    Base.metadata.create_all(bind=engine)


@app.post("/transactions/", response_model=TransactionResponse)

def     create_transaction(transaction:     TransactionCreate,     db:     Session     =
Depends(SessionLocal)):

    db_transaction = Transaction(**transaction.dict())

    db.add(db_transaction)

    try:
        db.commit()
        db.refresh(db_transaction)
    except IntegrityError:
        db.rollback()
        raise HTTPException(status_code=400, detail="Transaction could not be created.")
    return db_transaction


@app.get("/transactions/", response_model=list[TransactionResponse])
def read_transactions(skip: int = 0, limit: int = 10, db: Session =
Depends(SessionLocal)):
    transactions = db.query(Transaction).offset(skip).limit(limit).all()
    return transactions


client = TestClient(app)


def test_create_transaction():
     response = client.post("/transactions/", json={"date": "2023-10-01", "description":
"Salary", "amount": 5000, "category": "Income"})
    if response.status_code == 200:
        data = response.json()
        assert 'id' in data
    else:
        print(response.status_code, response.text)


def test_read_transactions():
    response = client.get("/transactions/")
    if response.status_code == 200:

        data = response.json()

        assert isinstance(data, list)

    else:

        print(response.status_code, response.text)



test_create_transaction()

test_read_transactions()
```