Source Code: AI Generated Report

Generated on: 2026-02-07 13:28:03

```
from fastapi import FastAPI, HTTPException, Depends
from fastapi.security import OAuth2PasswordBearer, OAuth2PasswordRequestForm
from pydantic import BaseModel
from sqlalchemy import create_engine, Column, Integer, String, Float
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, Session
from datetime import datetime
import json
import hashlib

DATABASE_URL = "sqlite:///./test.db"
engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()

app = FastAPI()
oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")

class User(Base):
    __tablename__ = "users"
    id = Column(Integer, primary_key=True, index=True)
    username = Column(String, unique=True, index=True)
    hashed_password = Column(String)
```

```
class Transaction(Base):
    __tablename__ = "transactions"
    id = Column(Integer, primary_key=True, index=True)
    user_id = Column(Integer)
    amount = Column(Float)
    category = Column(String)
    transaction_type = Column(String)  # income or expense
    created_at = Column(String, default=datetime.utcnow)

Base.metadata.create_all(bind=engine)

class UserCreate(BaseModel):
    username: str
    password: str

class TransactionCreate(BaseModel):
    user_id: int
    amount: float
    category: str
    transaction_type: str

def hash_password(password: str):
    return hashlib.sha256(password.encode()).hexdigest()

@app.post("/register")
```

```
def register(user: UserCreate):
    db: Session = SessionLocal()
    db_user = User(username=user.username, hashed_password=hash_password(user.password))
    db.add(db_user)
    db.commit()
    db.refresh(db_user)
    return {"username": db_user.username}


@app.post("/token")
def login(form_data: OAuth2PasswordRequestForm = Depends()):
    db: Session = SessionLocal()
    user = db.query(User).filter(User.username == form_data.username).first()
    if not user or user.hashed_password != hash_password(form_data.password):
        raise HTTPException(status_code=400, detail="Incorrect username or password")
    return {"access_token": user.username, "token_type": "bearer"}


@app.post("/transactions/")
def create_transaction(transaction: TransactionCreate):
    db: Session = SessionLocal()
    db_transaction = Transaction(**transaction.dict())
    db.add(db_transaction)
    db.commit()
    db.refresh(db_transaction)
    return db_transaction


@app.get("/reports/")
```

```
def get_reports(user_id: int):
    db: Session = SessionLocal()
    transactions = db.query(Transaction).filter(Transaction.user_id == user_id).all()
    total_income = sum(t.amount for t in transactions if t.transaction_type == "income")
       total_expenses = sum(t.amount for t in transactions if t.transaction_type ==
"expense")
    return {
        "total_income": total_income,
        "total_expenses": total_expenses,
        "budget_adherence": total_income - total_expenses
    }

if __name__ == '__main__':
    from fastapi.testclient import TestClient
    client = TestClient(app)

    # Mock requests
       response = client.post("/register", json={"username": "testuser", "password":
"testpass"})
    print("Register Response:", response.json())

        response = client.post("/token", data={"username": "testuser", "password":
"testpass"})
    print("Login Response:", response.json())

    token = response.json()["access_token"]
```

```
        response = client.post("/transactions/", headers={"Authorization": f"Bearer {token}"}, json={"user_id": 1, "amount": 100.0, "category": "Salary", "transaction_type": "income"})
    print("Transaction Response:", response.json())


    response = client.get("/reports/?user_id=1")
    print("Reports Response:", response.json())
```