Source Code: AI Generated Report

Generated on: 2026-02-07 13:42:33

```
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 fastapi.testclient import TestClient
import hashlib
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)
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 Expense(Base):
    __tablename__ = "expenses"
    id = Column(Integer, primary_key=True, index=True)
    user_id = Column(Integer)
    amount = Column(Float)
    description = Column(String)

class Budget(Base):
    __tablename__ = "budgets"
    id = Column(Integer, primary_key=True, index=True)
    user_id = Column(Integer)
    amount = Column(Float)
    category = Column(String)

Base.metadata.create_all(bind=engine)

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

class ExpenseCreate(BaseModel):
    user_id: int
    amount: float
    description: str
```

```
class BudgetCreate(BaseModel):
    user_id: int
    amount: float
    category: str


@app.post("/token")
async def login(form_data: OAuth2PasswordRequestForm = Depends()):
    user = get_user(form_data.username)
    if not user or not verify_password(form_data.password, user.hashed_password):
        raise HTTPException(status_code=400, detail="Incorrect username or password")
    return {"access_token": user.username, "token_type": "bearer"}


@app.post("/users/", response_model=UserCreate)
def create_user(user: UserCreate, db: Session = Depends(SessionLocal)):
    hashed_password = hashlib.sha256(user.password.encode()).hexdigest()
    db_user = User(username=user.username, hashed_password=hashed_password)
    db.add(db_user)
    db.commit()
    db.refresh(db_user)
    return db_user


@app.post("/expenses/", response_model=ExpenseCreate)
def create_expense(expense: ExpenseCreate, db: Session = Depends(SessionLocal)):
    db_expense = Expense(**expense.dict())
    db.add(db_expense)
    db.commit()
```

```
    db.refresh(db_expense)
    return db_expense

@app.post("/budgets/", response_model=BudgetCreate)
def create_budget(budget: BudgetCreate, db: Session = Depends(SessionLocal)):
    db_budget = Budget(**budget.dict())
    db.add(db_budget)
    db.commit()
    db.refresh(db_budget)
    return db_budget

def get_user(username: str):
    db = SessionLocal()
    return db.query(User).filter(User.username == username).first()

def verify_password(plain_password, hashed_password):
    return hashlib.sha256(plain_password.encode()).hexdigest() == hashed_password

if __name__ == '__main__':
    client = TestClient(app)
        response  =  client.post("/users/",  json={"username":  "testuser",  "password":
"testpass"})
    print("User Creation Response:", response.json())
         response  =  client.post("/expenses/",  json={"user_id":  1,  "amount":  50.0,
"description": "Groceries"})
    print("Expense Creation Response:", response.json())
```

```
    response = client.post("/budgets/", json={"user_id": 1, "amount": 200.0, "category":
"Food"})
    print("Budget Creation Response:", response.json())
```