Source Code: AI Generated Report

Generated on: 2026-02-07 13:35:02

```
from fastapi import FastAPI, Depends, HTTPException, status

from fastapi.security import OAuth2PasswordBearer, OAuth2PasswordRequestForm

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

from sqlalchemy.orm import sessionmaker, declarative_base, relationship

from sqlalchemy.ext.declarative import DeclarativeMeta

from pydantic import BaseModel

from passlib.context import CryptContext

from datetime import datetime

from fastapi.testclient import TestClient



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



engine = create_engine(DATABASE_URL)

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

Base: DeclarativeMeta = declarative_base()



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, ForeignKey("users.id"))

    amount = Column(Float)

    description = Column(String)

    created_at = Column(String, default=datetime.utcnow)


class Budget(Base):

    __tablename__ = "budgets"

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

    user_id = Column(Integer, ForeignKey("users.id"))

    limit = Column(Float)

    created_at = Column(String, default=datetime.utcnow)


Base.metadata.create_all(bind=engine)


class UserCreate(BaseModel):

    username: str

    password: str


class UserInDB(UserCreate):

    hashed_password: str


class ExpenseCreate(BaseModel):

    user_id: int

    amount: float
    description: str


class BudgetCreate(BaseModel):
    user_id: int
    limit: float


class UserService:
    pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

    @classmethod
    def hash_password(cls, password: str) -> str:
        return cls.pwd_context.hash(password)

    @classmethod
    def verify_password(cls, plain_password, hashed_password):
        return cls.pwd_context.verify(plain_password, hashed_password)


app = FastAPI()


@app.post("/users/", response_model=UserInDB)
def create_user(user: UserCreate):
    db = SessionLocal()
    hashed_password = UserService.hash_password(user.password)
    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 = 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 = SessionLocal()

    db_budget = Budget(**budget.dict())

    db.add(db_budget)

    db.commit()

    db.refresh(db_budget)

    return db_budget



@app.get("/expenses/summary")

def get_expense_summary(user_id: int):

    db = SessionLocal()

    total_expenses = db.query(Expense).filter(Expense.user_id == user_id).count()

    return {"total_expenses": total_expenses}
```

```python
@app.get("/budgets/alerts")
def check_budget_alerts(user_id: int):
    db = SessionLocal()
    budgets = db.query(Budget).filter(Budget.user_id == user_id).all()
    alerts = []
    for budget in budgets:
        total_expenses = db.query(Expense).filter(Expense.user_id == user_id).count()
        if total_expenses > budget.limit:
            alerts.append(f"Budget limit exceeded for budget ID {budget.id}")
    return {"alerts": alerts}


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, "limit": 100.0})
    print("Budget Creation Response:", response.json())
    response = client.get("/expenses/summary?user_id=1")
    print("Expense Summary Response:", response.json())
    response = client.get("/budgets/alerts?user_id=1")
    print("Budget Alerts Response:", response.json())
```