# Source Code: AI Generated Report

*Generated on: 2026-02-07 13:31:38*

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

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

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 Income(Base):
```

```
    __tablename__ = "incomes"
    id = Column(Integer, primary_key=True, index=True)
    user_id = Column(Integer)
    amount = Column(Float)
    description = Column(String)
    date = 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)
    date = Column(String)

Base.metadata.create_all(bind=engine)

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

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

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


app = FastAPI()


def fake_hash_password(password: str):
    return "fakehashed" + password


@app.post("/token")
async def login(form_data: OAuth2PasswordRequestForm = Depends()):
                    user     =     SessionLocal().query(User).filter(User.username     ==
form_data.username).first()
    if not user or not fake_hash_password(form_data.password) == user.hashed_password:
        raise HTTPException(status_code=400, detail="Incorrect username or password")
    token = jwt.encode({"sub": user.username}, "secret", algorithm="HS256")
    return {"access_token": token, "token_type": "bearer"}


@app.post("/users/", response_model=UserCreate)
async def create_user(user: UserCreate):
    db = SessionLocal()
                                    db_user         =         User(username=user.username,
hashed_password=fake_hash_password(user.password))
```

```
    db.add(db_user)
    db.commit()
    db.refresh(db_user)
    return db_user

@app.post("/incomes/", response_model=IncomeCreate)
async def create_income(income: IncomeCreate):
    db = SessionLocal()
    db_income = Income(**income.dict())
    db.add(db_income)
    db.commit()
    db.refresh(db_income)
    return db_income

@app.post("/expenses/", response_model=ExpenseCreate)
async 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

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

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