from fastapi import FastAPI,Depends

from pydantic import BaseModel

from sqlalchemy import create\_engine, Column, Integer, String, ForeignKey

import jwt

import sqlalchemy

from datetime import datetime

from sqlalchemy.orm import sessionmaker, Session, relationship

import sqlalchemy.orm

app = FastAPI()

engin = create\_engine("sqlite:///./fast.db")

session = Session(bind=engin)

Base = sqlalchemy.orm.declarative\_base()

key = "rM9Ookf7XOyuBi5R"

def create\_token(user):

payload = {"exp": datetime.utc + 60\*12\*7, "sub": user.id}

return jwt.encode(payload, key, algorithm="HS256")

def verify\_token(token):

try:

return jwt.decode(token)

except:

return False

class User(Base):

\_\_tablename\_\_ ="user"

user\_id = Column(Integer,primary\_key=True)

username = Column(String)

password = Column(String)

token = Column(String)

class Product(Base):

\_\_tablename\_\_ = "product"

product\_id =Column(Integer,primary\_key=True)

name = Column(String)

price = Column(Integer)

description = Column(String)

stock = Column(Integer)

user = relationship('user',foreign\_keys = "user.user\_id")

Base.metadata.create\_all(bind=engin)

# def db\_session():

# db = session()

class User\_data(BaseModel):

username: str

password: str

token : str

class Product\_details(BaseModel):

name : str

price : int

description : str

stock :int

@app.post('/login\_create')

def create(user:User\_data):

token = create\_token(user)

user.token=token

user = User(\*\*user.model\_dump())

db = session()

db.add(user)

db.commit()

db.refresh(user)

return {"user":user.username,"token":user.token}

@app.get("/protected")

def protected(token:User\_data,Session = Depends(Session)):

token= db.query(User).filter(User.username == token.username).first()

if verify\_token(token.token):

return {"message": f"Welcome {token.username}"}

else:

return {"message":"invalid token"}

@app.post('/create')

def create(item:Product\_details):

item = Product(\*\*item.model\_dump())

db = session()

db.add(item)

db.commit()

db.refresh(item)

return {"message": "item created"}

@app.get('/products')

def retrive():

db = session()

products = db.query(Product).all()

return {"products":products}