```python
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}
```

```python
@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}
```