FastAPI Items API Assignment

Overview

This project is a simple RESTful API built using FastAPI and SQLite. The API allows users to perform CRUD operations (Create, Read, Update, Delete) on a single table named Items. Additionally, it provides a search functionality to filter records based on specific fields.

Table Structure

The table Items has the following fields:

- id (Integer, Primary Key, Auto Increment)
- name (String)
- description (String)
- price (Float)
- quantity (Integer)

API Endpoints

- POST /items/: Create a new item.
- GET /items/: Retrieve all items.
- GET /items/{item_id}: Retrieve an item by ID.
- PUT /items/{item_id}: Update an item by ID.
- DELETE /items/{item_id}: Delete an item by ID.
- GET /items/search/: Search items based on name, description, price range, and quantity.

Project Structure

```
fastapi_items/
├── database.py
├── crud.py
├── main.py
├── models.py
├── __init__.py
├── venv/
└── items.db
```

Code Implementation

database.py

```python
# database.py
from sqlalchemy import create_engine, Column, Integer, String, Float
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

DATABASE_URL = "sqlite:///./items.db"

engine = create_engine(DATABASE_URL, connect_args={"check_same_thread": False})
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()

class Item(Base):
    __tablename__ = "items"

    id = Column(Integer, primary_key=True, index=True, autoincrement=True)
    name = Column(String, index=True)
    description = Column(String)
    price = Column(Float)
    quantity = Column(Integer)

def init_db():
    Base.metadata.create_all(bind=engine)
```

crud.py

```python
# crud.py
from sqlalchemy.orm import Session
from models import Item

def create_item(db: Session, name: str, description: str, price: float, quantity: int):
    db_item = Item(name=name, description=description, price=price, quantity=quantity)
    db.add(db_item)
    db.commit()
    db.refresh(db_item)
    return db_item

def get_items(db: Session, skip: int = 0, limit: int = 10):
    return db.query(Item).offset(skip).limit(limit).all()

def get_item(db: Session, item_id: int):
    return db.query(Item).filter(Item.id == item_id).first()

def update_item(db: Session, item_id: int, name: str, description: str, price: float, quantity: int):
    db_item = db.query(Item).filter(Item.id == item_id).first()
    if db_item:
        db_item.name = name
        db_item.description = description
        db_item.price = price
        db_item.quantity = quantity
        db.commit()
        db.refresh(db_item)
    return db_item
```

crud.py > ...

```python
def delete_item(db: Session, item_id: int):
    db_item = db.query(Item).filter(Item.id == item_id).first()
    if db_item:
        db.delete(db_item)
        db.commit()
    return db_item

def search_items(db: Session, name: str = None, description: str = None, price_min: float = None, price_
    query = db.query(Item)
    if name:
        query = query.filter(Item.name.like(f"%{name}%"))
    if description:
        query = query.filter(Item.description.like(f"%{description}%"))
    if price_min is not None:
        query = query.filter(Item.price >= price_min)
    if price_max is not None:
        query = query.filter(Item.price <= price_max)
    if quantity is not None:
        query = query.filter(Item.quantity == quantity)
    return query.all()
```

models.py

models.py > ...

```python
# models.py
from sqlalchemy import Column, Integer, String, Float
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()

class Item(Base):
    __tablename__ = "items"

    id = Column(Integer, primary_key=True, index=True, autoincrement=True)
    name = Column(String, index=True)
    description = Column(String)
    price = Column(Float)
    quantity = Column(Integer)
```

main.py

main.py > ...

```python
# main.py
from fastapi import FastAPI, HTTPException, Depends
from sqlalchemy.orm import Session
from database import init_db, SessionLocal
import crud

app = FastAPI()

# Initialize database
init_db()

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

@app.get("/")
def read_root():
    return {"message": "Welcome to the FastAPI Items API"}

# Existing endpoints
# ...

@app.post("/items/")
def create_item(name: str, description: str, price: float, quantity: int, db: Session = Depends(get_db))
    return crud.create_item(db, name, description, price, quantity)

@app.get("/items/")
def read_items(skip: int = 0, limit: int = 10, db: Session = Depends(get_db)):
    items = crud.get_items(db, skip=skip, limit=limit)
```

main.py > ...

```python
@app.get("/items/")
def read_items(skip: int = 0, limit: int = 10, db: Session = Depends(get_db)):
    items = crud.get_items(db, skip=skip, limit=limit)
    return items

@app.get("/items/{item_id}")
def read_item(item_id: int, db: Session = Depends(get_db)):
    item = crud.get_item(db, item_id)
    if item is None:
        raise HTTPException(status_code=404, detail="Item not found")
    return item

@app.put("/items/{item_id}")
def update_item(item_id: int, name: str, description: str, price: float, quantity: int, db: Session = De
    item = crud.update_item(db, item_id, name, description, price, quantity)
    if item is None:
        raise HTTPException(status_code=404, detail="Item not found")
    return item

@app.delete("/items/{item_id}")
def delete_item(item_id: int, db: Session = Depends(get_db)):
    item = crud.delete_item(db, item_id)
    if item is None:
        raise HTTPException(status_code=404, detail="Item not found")
    return {"detail": "Item deleted"}

@app.get("/items/search/")
def search_items(name: str = None, description: str = None, price_min: float = None, price_max: float =
    items = crud.search_items(db, name, description, price_min, price_max, quantity)
    return items
```

Testing the Endpoints

Use the interactive API documentation at http://127.0.0.1:8000/docs to test the endpoints. You can also use curl commands or any API client like Postman.

Output

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS                                uvicorn

INFO:     Started server process [16168]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:50761 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:50761 - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     127.0.0.1:50768 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:50784 - "POST /items/?name=Item1&description=This%20is%20item%201&price=10.0&quantity=5 HTTP/1.1" 200 OK
INFO:     127.0.0.1:50785 - "GET /items/?skip=0&limit=10 HTTP/1.1" 200 OK
INFO:     127.0.0.1:50790 - "GET /items/1 HTTP/1.1" 200 OK
INFO:     127.0.0.1:50792 - "PUT /items/1?name=updated%20item&description=this%20item%20is%20updated&price=15&quantity=10 HTTP/1.1" 200
OK
INFO:     127.0.0.1:50796 - "DELETE /items/1 HTTP/1.1" 200 OK
INFO:     127.0.0.1:50797 - "GET /items/search/?name=Item%26Price&description=search&price_min=5&price_max=20&quantity=10 HTTP/1.1" 200
OK
```