**Simple Steps to fast-Api CRUD setup in Win10**

**Download Python Form Office Site**

**https://www.python.org/ftp/python/3.13.2/python-3.13.2-amd64.exe**

**Install File & Restart OS Windows**

1. Make sure you have **pip install fastapi[all] sqlalchemy sqlite** Framework installed,
2. then install database : pip install fastapi[all] sqlalchemy psycopg2-binary – postgres

Cereate main.py (under Main APP)

from fastapi import FastAPI, Depends, HTTPException

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

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import sessionmaker, Session

import os

# PostgreSQL Database URL (Modify with your credentials)

DATABASE\_URL = "postgresql://postgres:admin@localhost:5432/ecom"

# Create Database Engine

engine = create\_engine(DATABASE\_URL)

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

Base = declarative\_base()

app = FastAPI()

# Database Model

class User(Base):

    \_\_tablename\_\_ = "users"

    id = Column(Integer, primary\_key=True, index=True)

    name = Column(String, index=True)

    email = Column(String, unique=True, index=True)

# Create Tables

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

# Dependency to Get DB Session

def get\_db():

    db = SessionLocal()

    try:

        yield db

    finally:

        db.close()

# Create User

@app.post("/users/", response\_model=dict)

def create\_user(name: str, email: str, db: Session = Depends(get\_db)):

    db\_user = User(name=name, email=email)

    db.add(db\_user)

    db.commit()

    db.refresh(db\_user)

    return {"id": db\_user.id, "name": db\_user.name, "email": db\_user.email}

# Get All Users

@app.get("/users/", response\_model=list)

def get\_users(db: Session = Depends(get\_db)):

    return db.query(User).all()

# Get Single User

@app.get("/users/{user\_id}", response\_model=dict)

def get\_user(user\_id: int, db: Session = Depends(get\_db)):

    user = db.query(User).filter(User.id == user\_id).first()

    if user is None:

        raise HTTPException(status\_code=404, detail="User not found")

    return {"id": user.id, "name": user.name, "email": user.email}

# Update User

@app.put("/users/{user\_id}", response\_model=dict)

def update\_user(user\_id: int, name: str, email: str, db: Session = Depends(get\_db)):

    user = db.query(User).filter(User.id == user\_id).first()

    if user is None:

        raise HTTPException(status\_code=404, detail="User not found")

    user.name = name

    user.email = email

    db.commit()

    db.refresh(user)

    return {"id": user.id, "name": user.name, "email": user.email}

# Delete User

@app.delete("/users/{user\_id}", response\_model=dict)

def delete\_user(user\_id: int, db: Session = Depends(get\_db)):

    user = db.query(User).filter(User.id == user\_id).first()

    if user is None:

        raise HTTPException(status\_code=404, detail="User not found")

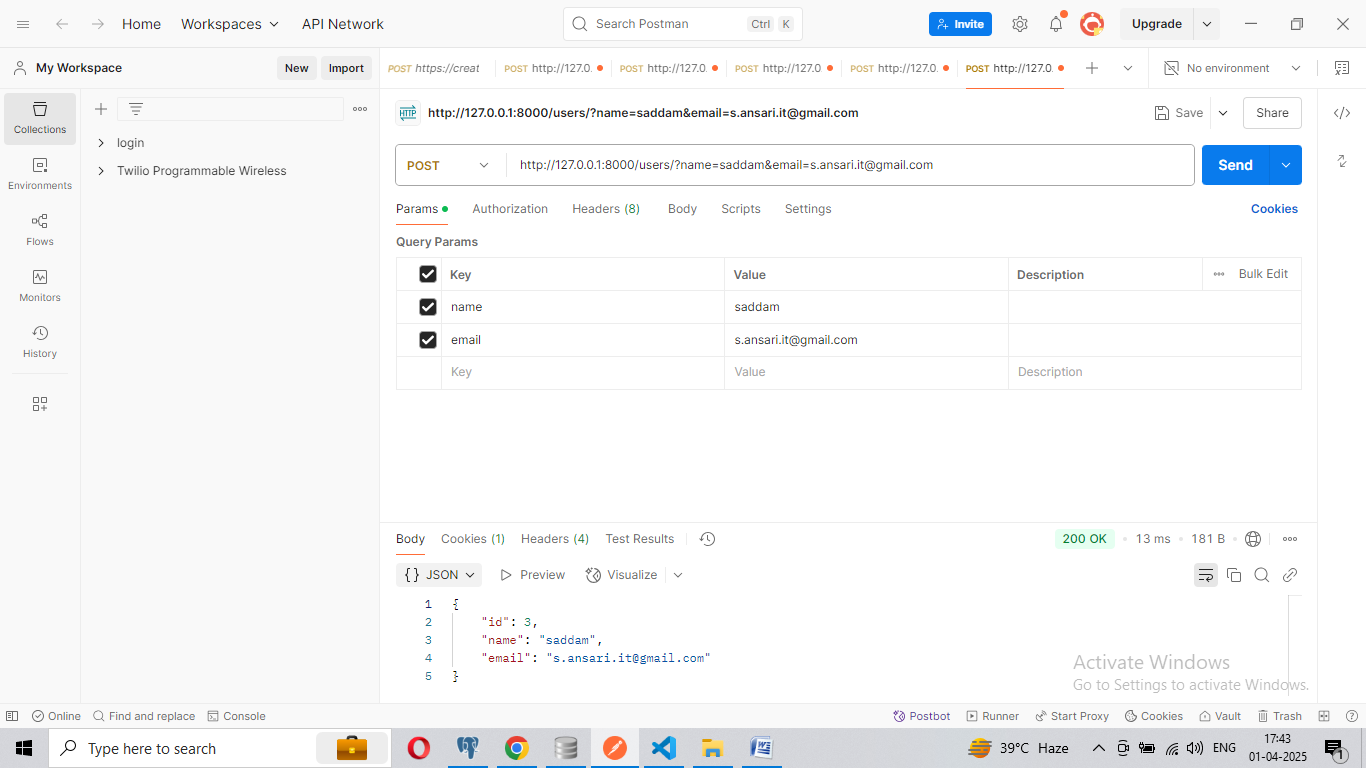
    db.delete(user)

    db.commit()

    return {"message": "User deleted successfully"}

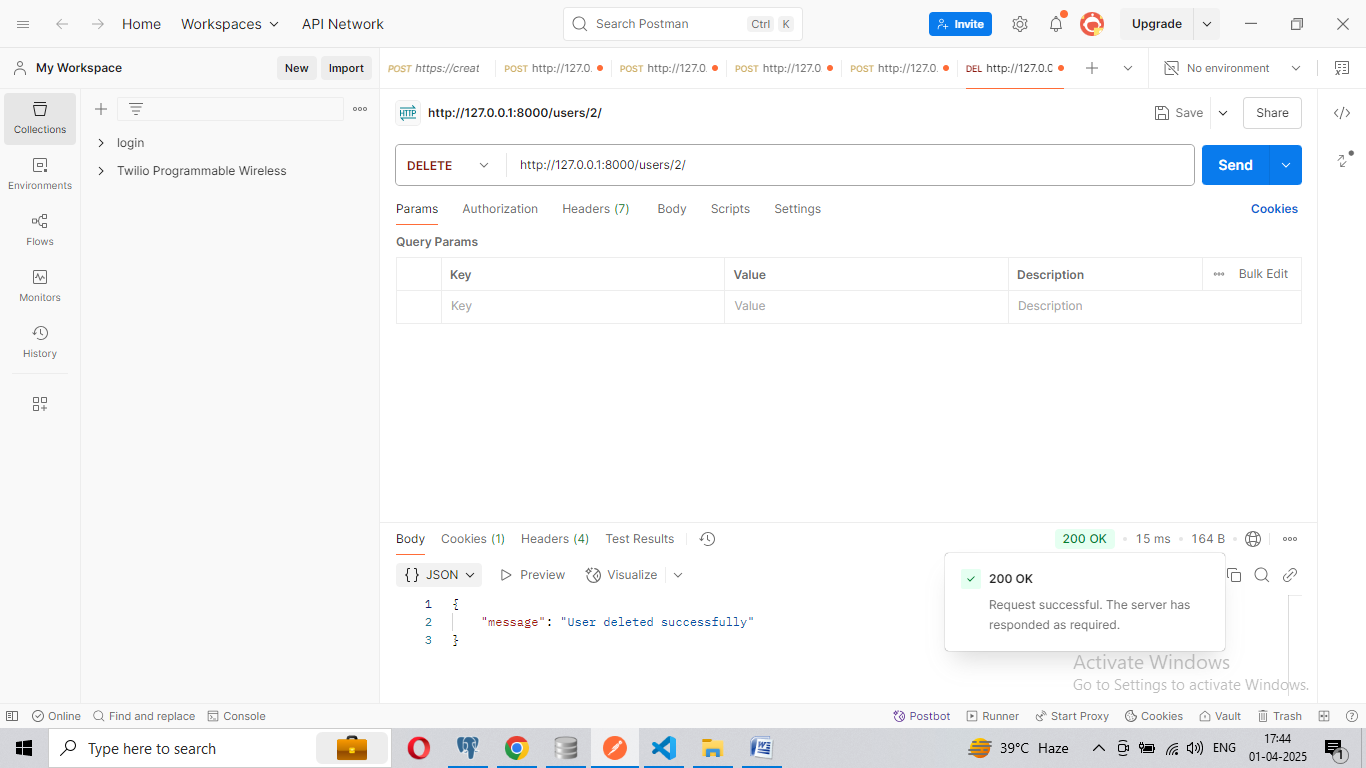
**Save & Exit;**

**Run own server : uvicorn main:app –reload**

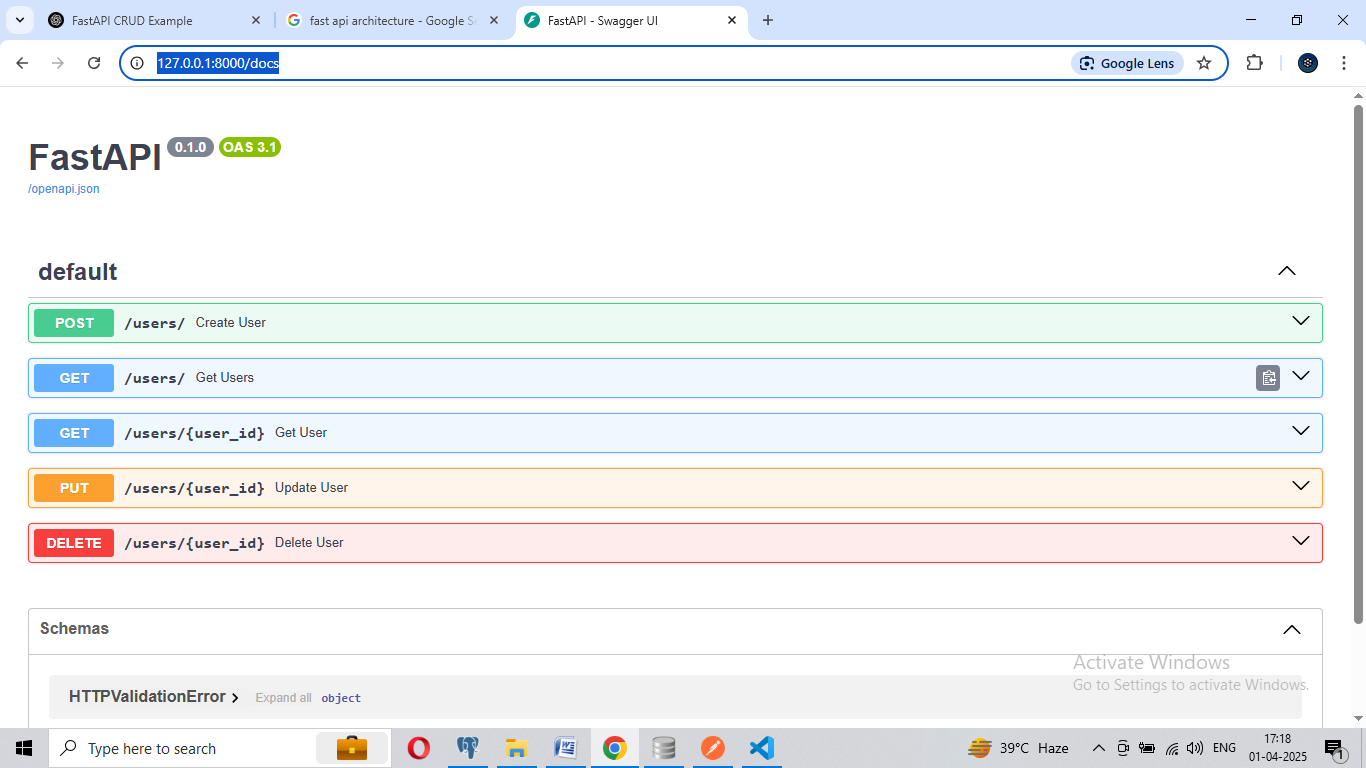
****

**API Endpoints**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /users/ | Create a new user |
| GET | /users/ | Get all users |
| GET | /users/{id} | Get a user by ID |
| PUT | /users/{id} | Update user details |
| DELETE | /users/{id} | Delete a user |

****

**Database login**

In your project, open <http://127.0.0.1:8000/docs> 

Swagger UI: its free install