set up basic fastapi application for crud operations for climate data using in memory space

set up use python virtual environment and use requirements.txt file to specify and manage project specific libraries

define pydantic model for climaate data and create end points for crud operations

test the api using tools swagger

list of end points

create an end point to add climate data

create an end point to get climate data with given id

create an end point to updata existing climatedata with given id

create an end point to delete existing climate data with given id

create an end point to get climate data history

and climate data should be as follows

url :<base url>/climatedata

method: Post

payload:{

temperature,

humidity,

air quality

}

python3 -m venv venv

source venv/bin/activate # On Windows, use `venv\Scripts\activate`

fastapi uvicorn pydantic

pip install -r requirements.txt

from typing import List, Optional

from fastapi import FastAPI, HTTPException

from pydantic import BaseModel

app = FastAPI()

# Pydantic model

class ClimateData(BaseModel):

id: Optional[int]

temperature: float

humidity: float

air\_quality: float

# In-memory storage for climate data

climate\_data\_store = {}

current\_id = 1

# Create an endpoint to add climate data

@app.post("/climatedata", response\_model=ClimateData)

def add\_climate\_data(data: ClimateData):

global current\_id

data.id = current\_id

climate\_data\_store[current\_id] = data

current\_id += 1

return data

# Create an endpoint to get climate data with a given id

@app.get("/climatedata/{data\_id}", response\_model=ClimateData)

def get\_climate\_data(data\_id: int):

if data\_id not in climate\_data\_store:

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

return climate\_data\_store[data\_id]

# Create an endpoint to update existing climate data with a given id

@app.put("/climatedata/{data\_id}", response\_model=ClimateData)

def update\_climate\_data(data\_id: int, data: ClimateData):

if data\_id not in climate\_data\_store:

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

data.id = data\_id

climate\_data\_store[data\_id] = data

return data

# Create an endpoint to delete existing climate data with a given id

@app.delete("/climatedata/{data\_id}")

def delete\_climate\_data(data\_id: int):

if data\_id not in climate\_data\_store:

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

del climate\_data\_store[data\_id]

return {"message": "Climate data deleted successfully"}

# Create an endpoint to get climate data history

@app.get("/climatedata", response\_model=List[ClimateData])

def get\_climate\_data\_history():

return list(climate\_data\_store.values())

uvicorn main:app –reload

<http://127.0.0.1:8000/docs>

1. **Add Climate Data**
   * **URL:** /climatedata
   * **Method:** POST
   * **Payload:**

json

Copy code

{

"temperature": 25.0,

"humidity": 60.0,

"air\_quality": 30.0

}

1. **Get Climate Data with a Given ID**
   * **URL:** /climatedata/{data\_id}
   * **Method:** GET
2. **Update Existing Climate Data with a Given ID**
   * **URL:** /climatedata/{data\_id}
   * **Method:** PUT
   * **Payload:**

json

Copy code

{

"temperature": 26.0,

"humidity": 65.0,

"air\_quality": 28.0

}

1. **Delete Existing Climate Data with a Given ID**
   * **URL:** /climatedata/{data\_id}
   * **Method:** DELETE
2. **Get Climate Data History**
   * **URL:** /climatedata
   * **Method:** GET

This setup allows you to perform basic CRUD operations on climate data using FastAPI and test it using Swagger.