Task 1

from flask import Flask, request, jsonify

import uuid

import re

app = Flask(\_\_name\_\_)

# In-memory data structure for users

users = {}

# Utility function to validate email

def is\_valid\_email(email):

email\_regex = r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'

return re.match(email\_regex, email) is not None

# Create a new user

@app.route('/users', methods=['POST'])

def create\_user():

data = request.get\_json()

if not data:

return jsonify({"error": "No data provided"}), 400

name = data.get("name")

email = data.get("email")

age = data.get("age")

if not name or not email or not age:

return jsonify({"error": "Missing required fields: name, email, or age"}), 400

if not is\_valid\_email(email):

return jsonify({"error": "Invalid email format"}), 400

try:

age = int(age)

if age <= 0:

raise ValueError

except ValueError:

return jsonify({"error": "Age must be a positive integer"}), 400

user\_id = str(uuid.uuid4())

users[user\_id] = {"id": user\_id, "name": name, "email": email, "age": age}

return jsonify(users[user\_id]), 201

# Get all users

@app.route('/users', methods=['GET'])

def get\_users():

return jsonify(list(users.values())), 200

# Get a single user by ID

@app.route('/users/<user\_id>', methods=['GET'])

def get\_user(user\_id):

user = users.get(user\_id)

if not user:

return jsonify({"error": "User not found"}), 404

return jsonify(user), 200

# Update a user by ID

@app.route('/users/<user\_id>', methods=['PUT'])

def update\_user(user\_id):

user = users.get(user\_id)

if not user:

return jsonify({"error": "User not found"}), 404

data = request.get\_json()

if not data:

return jsonify({"error": "No data provided"}), 400

name = data.get("name", user["name"])

email = data.get("email", user["email"])

age = data.get("age", user["age"])

if not is\_valid\_email(email):

return jsonify({"error": "Invalid email format"}), 400

try:

age = int(age)

if age <= 0:

raise ValueError

except ValueError:

return jsonify({"error": "Age must be a positive integer"}), 400

user.update({"name": name, "email": email, "age": age})

return jsonify(user), 200

# Delete a user by ID

@app.route('/users/<user\_id>', methods=['DELETE'])

def delete\_user(user\_id):

user = users.pop(user\_id, None)

if not user:

return jsonify({"error": "User not found"}), 404

return jsonify({"message": "User deleted successfully"}), 200

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)