Name : Jash Chauhan
Email: jashchauhan1999@gmail.com
Contact no. : +91 953799097
Topic:- Task of Python Backend Developer

This is task created in flask in which I have explained full task in video here also I am going share link of that video and code of particular task also.

Video link :- Detail video

Project repo link:-
https://github.com/JASHCHAUHAN1999/Employees_RestAPI_Flask

Code:-

Models.py

```
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
import secrets
from werkzeug.security import generate_password_hash, check_password_hash

database = SQLAlchemy()

class Login(database.Model):
    emp_id = database.Column(database.Integer, primary_key = True)
    emp_name = database.Column(database.String(50), unique = True)
    password = database.Column(database.String(255), nullable = False)
    token = database.Column(database.String(255), unique=True, nullable=True)

    def set_password(self,pwd):
        self.password = generate_password_hash(pwd)
    def check_password(self,pwd):
        return check_password_hash(self.password,pwd)

    def generate_token(self):
        self.token = secrets.token_hex(32)
        return self.token




class Employees(database.Model):
    id = database.Column(database.Integer, primary_key = True, autoincrement = True)
    name = database.Column(database.String(50), nullable = False)
    email = database.Column(database.String(100), nullable = False, unique = True)
    department = database.Column(database.String(100))
    role = database.Column(database.String(100))
    date_joined = database.Column(database.DateTime, default = datetime.now)

    def data(self):
        return{
            'id':self.id,
            'name':self.name,
            'email':self.email,
            'department':self.department,
            'role':self.role,
            'date_joined':self.date_joined
        }
```

App.py

```
from flask import Flask, jsonify, request
from config import Config
from models import Employees, database ,Login

app = Flask(__name__)
app.config.from_object(Config)
database.init_app(app)
```

```
@app.route('/')
def Home():
   return jsonify({'Home':"Home Page of Api"}), 200

@app.route('/employees')
def employee():
   employees = Employees.query.all()
   return jsonify([i.data() for i in employees]), 200

@app.route('/employees',methods = ['POST'])
def add_employee():
   data = request.get_json()
   new_employee = Employees(name = data['name'], email = data['email'],department =
data['department'],role = data['role'])
   database.session.add(new_employee)
   database.session.commit()
   return jsonify(new_employee.data()), 201

@app.route('/employees/<int:emp_id>',methods =['GET'])
def get_emp(emp_id):
   emp = Employees.query.get_or_404(emp_id)
   return jsonify(emp.data()), 200

@app.route('/employees/<int:emp_id>',methods =['PUT'])
def update_emp(emp_id):
   data = request.get_json()
   emp = Employees.query.get_or_404(emp_id)
   for i in emp.data():
      if i in data:
         # print(type(i),i)
         # emp.i = data[i]
         setattr(emp,i,data[i])
   database.session.commit()
   return jsonify(emp.data()), 201

@app.route('/employees/<int:emp_id>',methods =['DELETE'])
def del_emp(emp_id):
   emp = Employees.query.get_or_404(emp_id)
   database.session.delete(emp)
   database.session.commit()
   return jsonify('User Deleted'), 204

@app.route('/employees/',methods =['GET'])
def filtered_emp():
    dep = request.args.get("department")
    rl = request.args.get("role")
    page = request.args.get("page",1,type=int)
    if dep:
        emp = Employees.query.filter(Employees.department==dep)
        return jsonify([i.data() for i in emp]), 200

    if rl:
        emp = Employees.query.filter(Employees.role==rl)
        return jsonify([i.data() for i in emp]), 200
    if page:
        max_emp=10
        pagination = Employees.query.paginate(page=page, per_page=max_emp,
error_out=False )
        emp = pagination.items
        return jsonify({
        'page':page,
        'max employees':10,
        'data':[i.data() for i in emp]
        }), 200


@app.route('/employees/login',methods = ['POST'])
def login():
    data = request.get_json()
    emp = Login.query.filter_by(emp_name = data['name']).first()

    if not emp or not emp.check_password(data['password']):
        return jsonify({'Msg':'Invalid credenials'}), 401

    """existing_user = Login.query.filter_by(emp_name=data['name']).first()
    if existing_user:
        return jsonify({"msg": "Username already exists"}), 409"""

    token = emp.generate_token()
    database.session.commit()

    return jsonify({'token':token}, 200)

def token_required(f):
    def decorator(*args,**kwargs):
        token = request.headers.get("X-API-TOKEN")

        if not token:
            return jsonify({"msg":'Token missing'}),401

        emp = Login.query.filter_by(token=token).first()

        if not emp:
            return jsonify({"msg":'invalid token'}),401
        return f(*args,**kwargs)
    return decorator

@app.route('/employees/login/detials',methods = ['GET'])
@token_required
def get_employee():
    emp = Employees.query.all()
    return jsonify([i.data() for i in emp])


if __name__ == "__main__":
    with app.app_context():
       database.create_all()
    with app.app_context():
        if not Login.query.filter_by(emp_name="admin").first():
            login = Login(emp_name="admin")
            login.set_password("123")
            database.session.add(login)
            database.session.commit()

    app.run(debug = True, port = 5555)
```

config.py

```
class Config:
    SQLALCHEMY_DATABASE_URI =
'postgresql://postgres:password@localhost:5432/My_Flask'
```