from flask import Blueprint, jsonify, request

from backend import conn, config

import bcrypt

import jwt

import ibm\_db

auth = Blueprint("auth", \_\_name\_\_)

LOGIN\_FEILDS = ('email', 'password')

SIGNUP\_FEILDS = ('name', 'email', 'phone\_number', 'password')

@auth.route("/login", methods=['POST'])

def login\_user():

# Check if all the required feild are present

for feild in LOGIN\_FEILDS:

if not (feild in request.json):

return jsonify({"error": f"All feilds are required!"}), 409

email = request.json['email']

password = request.json['password']

sql = f"select \* from users where email='{email}'"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.execute(stmt)

user = ibm\_db.fetch\_assoc(stmt)

if not user:

return jsonify({"error": "Invalid credentials!"}), 401

if bcrypt.checkpw(password.encode('utf-8'),

user["PASSWORD"].encode('utf-8')):

token = jwt.encode(

{"email": email},

config["APP\_SECRET"],

algorithm="HS256"

)

return jsonify({"name": user["NAME"], "email": email, "phone\_number": user["PHONE\_NUMBER"], "token": token}), 200

else:

return jsonify({"error": "Invalid credentials!"}), 401

@auth.route("/signup", methods=['POST'])

def register\_user():

# Check if all the required feild are present

for feild in SIGNUP\_FEILDS:

if not (feild in request.json):

return jsonify({"error": f"All feilds are required!"}), 409

email = request.json['email']

phone\_number = request.json['phone\_number']

name = request.json['name']

password = request.json['password']

# Sql stmt to check if email/number is already in use

sql = f"select \* from users where email='{email}' or phone\_number='{phone\_number}'"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.execute(stmt)

user = ibm\_db.fetch\_assoc(stmt)

if user:

return jsonify({"error": f"Email/Phone number is alread in use!"}), 409

# If user does not exist, then create account

hashed\_password = bcrypt.hashpw(

password.encode('utf-8'), bcrypt.gensalt())

sql = f"insert into users(name,email,phone\_number,password) values('{name}','{email}','{phone\_number}',?)"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, hashed\_password)

ibm\_db.execute(stmt)

token = jwt.encode(

{"email": email},

config["APP\_SECRET"],

algorithm="HS256"

)

return jsonify({"name": name, "email": email, "phone\_number": phone\_number, "token": token}), 200