Flask application Building

Team ID:PNT2022TMID41082

```
import os
from flask import Flask,request,jsonify, json, Response, make_response, render_template
from flask_pymongo import PyMongo
from flask_bcrypt import Bcrypt
from flask_cors import CORS
import db
import mail_client
import auth
import pickle

app = Flask(__name__)
bcrypt = Bcrypt(app)
CORS(app)
model = pickle.load(open('model.pkl','rb'))

class UserAuthUtil:

    @app.route("/", methods=['GET'])
    def hello_world():
        return "Working"

    @app.route("/login", methods=['POST'])
    def login_user():
```

```
    try:
        if request.method == 'POST':
            form_data = request.get_json()
            email = form_data['email']
            password = form_data['password']
            if(email != '' and password != ''):
                data = list(db.users.find({'email': email}))
                if(len(data) == 0):
                    return Response(status=404, response=json.dumps({'message': 'user does not exist'}), mimetype='application/json')
                else:
                    print(data)
                    data = data[0]
                    print(str(data['_id']))
                    if(bcrypt.check_password_hash(data['password'], password)):
                        #token =jwt.encode({'email': email}, app.config['SECRET_KEY'])
                        return make_response(jsonify({'message':'User logged in successfully','token':auth.create_bearer_token(str(data['_id']))}), 201)
                        #,'token':auth.create_bearer_token(data['_id'])['id']
                    else:
                        return Response(status=402, response=json.dumps({'message': 'Invalid password'}), mimetype='application/json')
            else:
                return Response(status=400, response=json.dumps({'message': 'Bad request'}), mimetype='application/json')
        else:
```

```
            return Response(status=401, response=json.dumps({'message': 'invalid request type'}),
mimetype='application/json')
        except Exception as Ex:
            print('\n\n\n**********************************')
            print(Ex)
            print('**********************************\n\n\n')
            return Response(response=json.dumps({'message': "Internal Server error"}), status=500,
mimetype="application/json")



    @app.route("/register", methods=['POST'])
    def register_user():
        try:
            if request.method == "POST":
                user_details = request.get_json()
                full_name = user_details["fullName"]
                email = user_details["email"]
                password = user_details["password"]
                password_hash = bcrypt.generate_password_hash(password).decode('utf-8')
                response_message = 'User created successfully'
                if (full_name != '' and email != '' and password_hash != ''):

db.users.insert_one({'fullName':full_name,'email':email,'password':password_hash,'results':[]})
                    try:
                        mail_client.send_conf_mail(email)
                    except Exception as exx:
                        response_message += ', but confirmation mail could not be delivered'
```

```
            return Response(response=json.dumps({'message': response_message}), status=200, mimetype="application/json")

        else:

            return Response(status=400, response=json.dumps({'message': 'Please enter your details'}), mimetype='application/json')

    else:

        return Response(status=400, response=json.dumps({'message': 'Bad request'}), mimetype='application/json')

    except Exception as Ex:

        print('\n\n\n**********************************')

        print(Ex)

        print('**********************************\n\n\n')

        return Response(response=json.dumps({'message': "Internal Server Error"}), status=500, mimetype="application/json")


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

def test_protected_route():

    user_token = request.headers.get('Authorization').split(" ")[1]

    is_valid_user = auth.decode_and_verify_user(user_token)[0]

    if is_valid_user:

        return Response(response=json.dumps({'message': 'You are authorised to access this page'}), status=200, mimetype="application/json")

    else:

        return Response(response=json.dumps({'message': 'You are not authorised to access this page'}), status=401, mimetype="application/json")


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

def predict_heart_disease():
```

```
    try:
        user_token = request.headers.get('Authorization').split(" ")[1]
        is_valid_user = auth.decode_and_verify_user(user_token)[0]
        if is_valid_user:
            params = request.get_json()
            result =
model.predict([[params['age'],params['sex'],params['chest_pain_type'],params['bp'],params['cholestrol'
],params['fbs_over_120'],params['ekg_results'],params['max_hr'],params['exercise_angina'],params['st_
depression'],params['slope_of_st'],params['number_of_vessels_fluro'],params['thallium']]])
            return Response(response=json.dumps({'prediction': result[0]}), status=200,
mimetype="application/json")
        else:
            return Response(response=json.dumps({'message': 'You are not authorised to access this
page'}), status=401, mimetype="application/json")
    except Exception as ex:
        return Response(response=json.dumps({'message': "Internal Server Error"}), status=500,
mimetype="application/json")


@app.route("/save_result", methods=['PUT'])
def save_result():
    try:
        user_token = request.headers.get('Authorization').split(" ")[1]
        user_details = auth.decode_and_verify_user(user_token)
        is_valid_user = user_details[0]
        user = user_details[1]
        if is_valid_user:
            params = request.get_json()
            past_results = user[0]['results']
```

```
            past_results.append(params)

            db.users.update_one({'_id':user[0]['_id']}, {"$set":{'results':past_results}})

            return Response(response=json.dumps({'message': 'Results Saved'}), status=200,
mimetype="application/json")

        else:

            return Response(response=json.dumps({'message': 'You are not authorised to access this
page'}), status=401, mimetype="application/json")

    except Exception as ex:

        print(ex)

        return Response(response=json.dumps({'message': "Internal Server Error"}), status=500,
mimetype="application/json")

if __name__ == '__main__':

  app.run(port=8000)
```