Restful API And Microservices with Python

Day 5

Day 5 - Overview

- Demo of User Management Service
- Security in REST APIs
- Authentication and logging in—part 1
- Logging in with Flask-JWT-Extended
- Generating JWT token
- Using the JWT token to access a protected resource
- Token expiry
- Testing from postman

Prerequisite

- VM with windows OS
- Python 3.8 or >
- Visual Studio Code - Code Editor
- Postman
- GIT - *https://github.com/saurav-samantray/flask-microservices-training/blob/main/slides/Setup%20GIT%20in%20your%20Local%20system.pdf*
- Docker - Not Mandatory for current training

Sync your fork for Day 5 activities

- Follow the below document to sync your fork and update local repository.

  https://github.com/saurav-samantray/flask-microservices-training/blob/main/slides/Setup%20GIT%20in%20your%20Local%20system.pdf

Local Setup for Day 5

- Navigate to the below folder

  C:\workspace\flask-microservices-training\day5\user-management-service

- Create a virtual environment and activate it

  python -m venv venv

  .\venv\Scripts\activate

- Install the dependencies, initialize DB and start server

  pip install -r requirements.txt

  python init_db.py

  python server.py

Security in REST API

- API security is the protection of the integrity of APIs—both the ones you own and the ones you use. But what does that mean?
- Businesses use APIs to connect services and to transfer data. Broken, exposed, or hacked APIs are behind major data breaches. They expose sensitive medical, financial, and personal data for public consumption. That said, not all data is the same nor should be protected in the same way. How you approach API security will depend on what kind of data is being transferred.

Generating JWT token

```
@app.route("/api/login", methods=["POST"])
def login():
    email = request.json.get("email", None)
    password = request.json.get("password", None)
    if email != "test@gmail.com" or password != "test":
        return {"msg": "Bad email or password"}, 401
    access_token = create_access_token(identity=email)
    return jsonify(access_token=access_token)
```

Accessing Protected Resource

```
@app.route("/api/protected", methods=["GET"])
@jwt_required()
def protected():
    # Access the identity of the current user with get_jwt_identity
    current_user = get_jwt_identity()
    return jsonify(logged_in_as=current_user), 200
```

Q and A