Restful API And Microservices with Python

Day 7

Day 7 - Overview

- Updating User and Address relationship
- What is a fresh token?
- Performing token refresh in our REST API
- Requiring a fresh token in an endpoint
- Custom exception handling for

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 7 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 7

- Navigate to the below folder

```
C:\workspace\flask-microservices-training\day7\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
```

SQL One to Many Relationship

- Establish a One-to-Many relationship between User and Address
- Update Addresses and Address APIs to handle the relationship

What is Refresh Token

- Access token is usually short lived, around 30 minutes to few hours. After the expiry time, the token expires and user has to login again which is not a good experience.
- Refresh Token is also a type of bearer token which live longer than access token and can be used to generate a new user token automatically.
- No need to store or ask for username and password.

Updating the Auth API to add refresh Token

```
class AuthApi(Resource):
    def post(self):
        email = request.json.get("email", None)
        password = request.json.get("password", None)
        conn = get_db_connection()
        user = get_user_details_from_email(conn, email)
        if user is None:
            raise UserNotFoundException(f"User with email [{email}] not found in DB", 400)
        if email != user.email or not flask_bcrypt.check_password_hash(user.password, password):
            return {"msg": "Bad email or password"}, 401

        access_token = create_access_token(identity=email)
        refresh_token = create_refresh_token(identity=email)
        return {"access_token":access_token, "refresh_token": refresh_token}
```

New Refresh API to refresh the expired token

```
class RefreshTokenApi(Resource):
    decorators = [jwt_required(refresh=True)]
    def post(self):
        identity = get_jwt_identity()
        access_token = create_access_token(identity=identity)
        return {"access_token":access_token}
```

Q and A