



Restful API And Microservices with Python

Day 6



Day 6 - Overview

- Signing up and writing Users to a database
- Logging in and retrieving Users from a database
- Preventing duplicate usernames when signing users up
- Securing our Item(Address) resources from a database - Task
- Advanced Flask-JWT Configuration
- Unit test cases for Users and User API
- Unit test case for Addresses and Address API - Task



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 6 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 6

- Navigate to the below folder

C:\workspace\flask-microservices-training\day6\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



Creating a User Registration API

```
class RegisterApi(Resource):  
    def post(self):  
        errors = user_schema.validate(request.json)  
        print("errors: "+str(errors))  
        if errors:  
            raise InvalidUserPayload(errors, 400)  
        user = User.from_json(request.json)  
        user.password = flask_bcrypt.generate_password_hash(user.password).decode('utf-8')  
        #user_dict['password'] = flask_bcrypt.generate_password_hash(user_dict['password'])  
        conn = get_db_connection()  
        user_db.create_users(conn, user)  
        commit_and_close_db_connection(conn)  
        return user, 201
```



Using JWT token to retrieve list of users

```
class UsersApi(Resource):  
    decorators = [jwt_required()]  
    def get(self):  
        conn = get_db_connection()  
        users = user_db.get_users(conn)  
        close_db_connection(conn)  
        return users
```



Preventing duplicate email/username

```
class RegisterApi(Resource):
    def post(self):
        #Payload Validation
        errors = user_schema.validate(request.json)
        print("errors: "+str(errors))
        if errors:
            raise InvalidUserPayload(errors, 400)

        #Validating duplicate user
        conn = get_db_connection()
        email = request.json.get("email", None)
        existing_user = get_user_details_from_email(conn, email)
        if existing_user is not None:
            raise UserExistsException(f"User with email [{email}] already exists in DB")

        user = User.from_json(request.json)
        user.password = flask_bcrypt.generate_password_hash(user.password).decode('utf-8')
        #user dict['password'] = flask_bcrypt.generate_password_hash(user dict['password'])

        user_db.create_users(conn, user)
        commit_and_close_db_connection(conn)
        return user, 201
```




Securing the Addresses API

- Task for the class



Writing Unit Test cases

- Covering all testing scenarios for Users and User API
- Covering all testing scenarios for Addresses and Address API - Task



Q and A