<u>MicroService Exercises</u>

Implement the MicroServices with Spring Boot and **MySQL database**.

1. Design new MicroService users with following specifications.
    a. Model: User with attributes as id, name, password, email and account number.
    b. User Registration with POST method→to accept the new user details in JSON format without account number, add these to backend, assign a new account number and return this user details with account number.
    c. User validation: Get method→accept the user id and password as query-string/parameters and return the assigned current account number and id with name to the client.
    d. User Update→PUT method.--> accept the user id as parameter and name,password,email as json data. Update the received values into corresponding backend and return the user details as name, id, email and current account number to client in json format.
    e. User Delete→With Delete method→Accept the admin id and password as json data along with user id as parameter. Validate the admin request and upon successful validation remove the current user record identified by id and in case of validation failure return failure message with http forbidden status code.
    f. Test the MicroService operations with CURL and RESTED add-on in Firefox browser.
2. Design another MicroService **accounts** with following specifications.
    a. Model : Account with attributes as accountNumber, balance, username etc.
    b. Create a new account with minimum balance with http POST method→To accept the new account details in JSON format with account number, minimum balance of 100 and username. Add these to backend , return the user balance amount with account number in json format.
    c. Query the account to get the balance amount with GET method→Parameter accountNumber returns the balance amount with account number and username in json format.
    d. Deposit amount->Http PUT method→parameter accountNumber, accept the amount, and username in json format, locate the existing account, update the backend and return the updated balance amount with account number and username in json format. If the account record doesn't exist, return error with http Status code for NOT exists.
    e. GetAmount withdrawn and returns the withdrawn amount ->With http GET method.-->Accept the parameter accountNumber, accept the amount, and

username in json format, locate the existing account, update the backend by subtraction and return the updated balance amount in json format. If the account record doesn’t exist, return error with http Status code for NOT exists.

3. Design **frontend** MicroService to work with above two MicroServices.
    a. Model : Employee with attributes as id,name,password,email and account number and salary as amount deposited to account service.
    b. Client queries the service with Get method→To get the employee salary by validating the employee from user service and then getting the salary from account service as account balance in json format. In case the employee record doesn’t exist, return error with http status code.
4. Design **Employer** MicroService to work with **user** and **account** MicroService.
    a. Credit amount to account service→Http Put method→ parameter employee-id and salary amount as json data.→Validate the employee from user service, get the accountNumber and using this accountNumber put salary amount in **account service** backend.
5. Implement the MicroServices application inside docker as multiple linked containers.
6. With Kubernetes cluster framework manage multiple service instances, utilize the service registry and discovery for services running on external machines in the network.
7. Define load balancing with different algorithms.

****