Distributed Operating System

HW.2: Microwebservices

Student Name: Asem Jamil Ishtayah

Student ID: 11316990

Supervisor: Dr. Samer Arandi

**Introduction:**

On this project, or we might call it a lab, we will deal with some frameworks, database, virtual machines to operate gusts operating systems …etc. we will learn to deal with micro-webservices and make requests to these services and handle the responses to build a small distributed system project.

**Requirements:**

1. VMware: to operate gusts operating systems that will operate servers separately.
2. Gusts OS: ubuntu-18.04.3.
3. Micro-web framework: We use FLASK for Python language.
4. Database: We use SQLite3.
5. Text editor: Visual Studio Code.
6. Github account: <https://github.com/AsemIsht/DOS_HW-ASEM>

**Design:**

The project consists of three severs each server is installed on Flask framework for Python language that consist of micro-webservices, each server has separate purpose and works on separate gust operating system on VMware for virtualization, and these servers are:

1. Frontend Sever.
2. Order Server.
3. Catalo Server.

The client will request one of three micro-webservices to Frontend server:

1. search(topic).

http://Frontend-server/search/<topic>

1. lookup(item\_number).

http://Frontend-server/lookup/<item\_number>

1. buy(item\_number).

http://Frontend-server/buy/<item\_number>

Note that the **Database** is controlled by catalog server only. And if any of other servers need any data from Database will request from Catalog. As well as requests for update and modification any data.

Catalog Server will response to frontend and order servers, therefor it has these micro-webservices:

1. Query\_by\_topic(topic):

http://Catalog-server/query\_by\_topic/<topic>

1. Query\_by\_item\_number(item\_number):

http://Catalog-server/query\_by\_ item\_number/<item\_number>

1. Update(item\_number):

http://Catalog-server/update/<item\_number>

Order Server will response only to frontend server, therefor it has this micro-webservices:

buy(item\_number):

http://Order-server/buy/<item\_number>

**Sequence of operating:**

Search for a topic/item category(search, lookup): client requests a **GET** request to frontend server which handle this request and make a **GET** request from Catalog server which make a query to get all items that categorized on this topic to response to Frontend server. When Frontend server get the data will return it to client application with nicely formatted.

Host requests from frontend server 🡺 frontend server request the data from Catalog server 🡺 Catalog server check this query and return the results to frontend server which in turn will return the information in a nicely formatted to Client application.

Buy an item (Buy): Client requests a **GET** request to Frontend server, Frontend handle the request and requests a **PUT** request to order server which make some processes. Order server request a **GET** request to catalog server to get the quantity of this book which wanted from client. After the order server get the quantity and if it was positive number will request a **POST** request to catalog server to update the quantity (decrement 1 then send request with new\_value as parameter). When Catalog response positively to Order server, Order server will response to frontend server and inform it if the buy was done successfully. Then frontend server response to client application.

**How to run the program:**

At first clone the repository to your desk.

**Output generated:**

S

**Inline comments:**

اكتب comments على الكود

**Short demo video (ZOOM APP recorder)**