Assignment #1

2.1 Software Set-up.

Basically, how I understood it is that this is a program that does both ordering and monitoring. The program has two components that do continuous monitoring; first, “Hystrix Dahsboard” which has two other components which monitor the state of the Hystrex. Second, the Eureka which also has two components which do service discovery. The other part of the application is ordering. In the process of ordering, the user can first browse a catalog. After browsing, the user can add items from the browsed catalog to which I assume is a shopping cart, the user then have the option to go back to browsing the catalog or removing am item from the cart. The other user is the root user who is able to add materials to the catalog the see the system’s states and services. At the end, everything is sent to the server which display’s it on the screen.

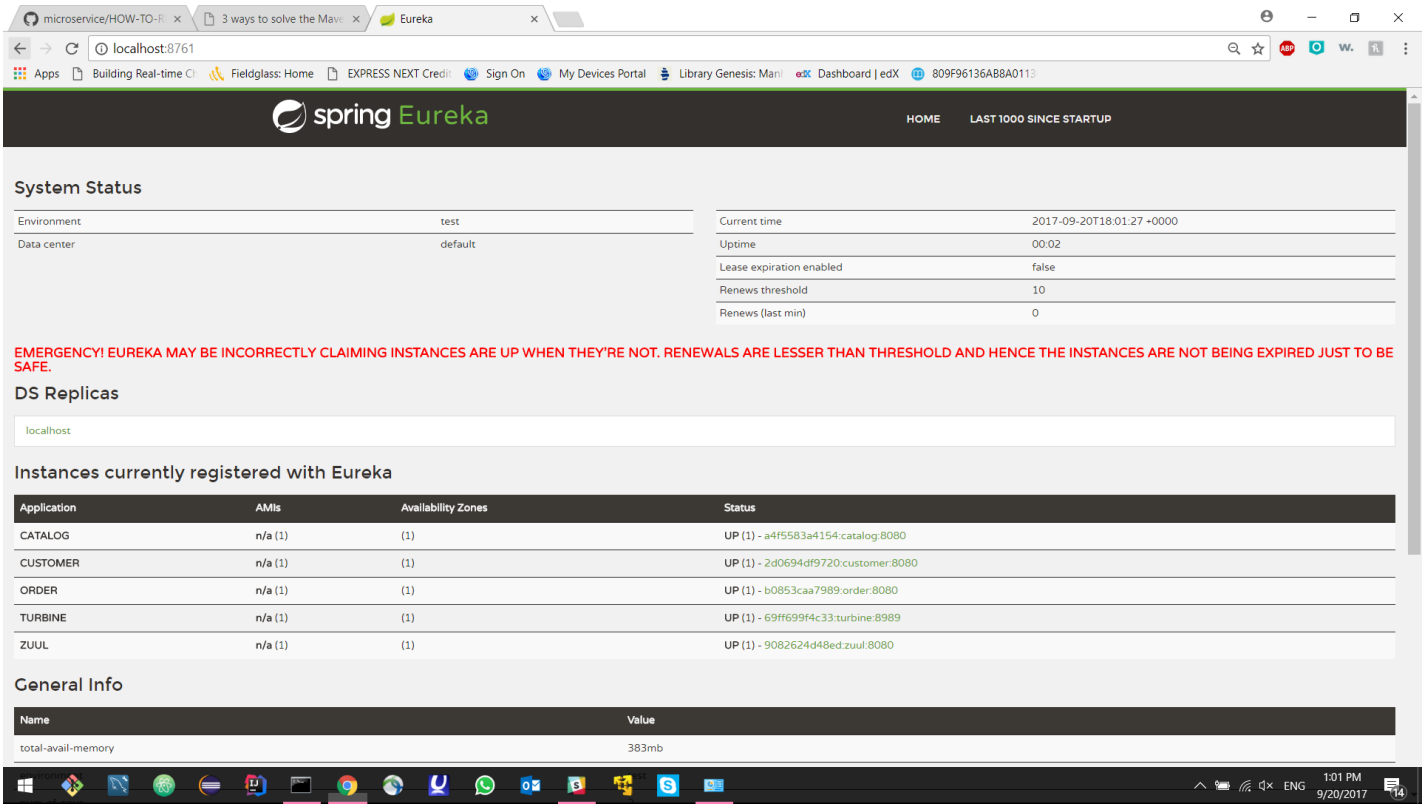


Figure 1_Microservices Status

2.3 Use case diagram.

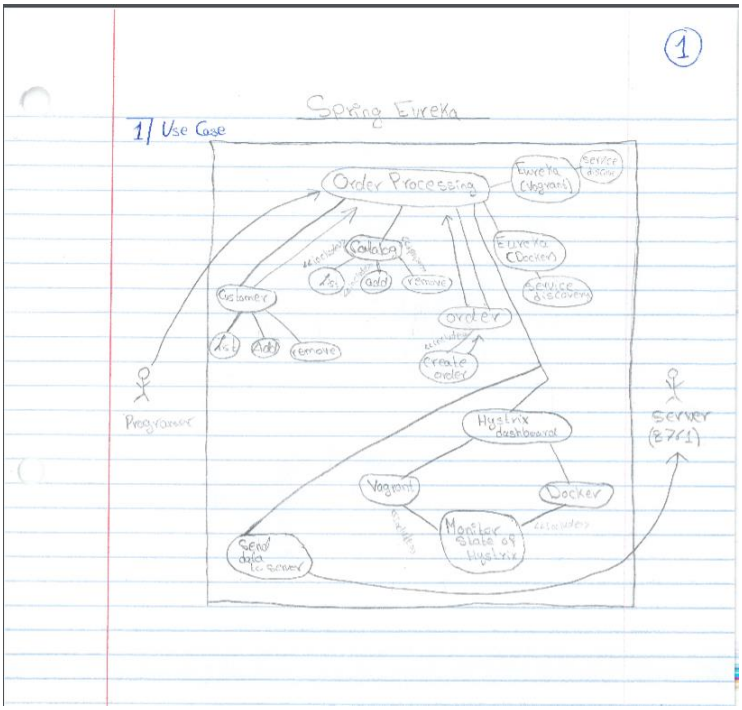


Figure 2_Use case diagram

This is a use case diagram. It shows the use cases and the actors of the project. The description of the actors, use cases, and their functionality are shown in the tables below.

Use Case	**Functionality**	**Actor**	**Actor's functionality**
Eureka (Vogrant)	Service Discovery	Programmer	Sets the system
Eureka (Docker)	Service Discovery	User	Selections (e.g. browsing, adding item, removing item)
Catalog	View catalog	Server	View data & final result.
List (Catalog)	List the catalog		
List (Customer)	List the customer's items		
Add (Catalog)	Add an item to the catalog		
Add (Customer)	Add an item to the customer's list		
Remove (Catalog)	Remove an item from the catalog		
Remove (Customer)	Remove an item from the customer's list		

Hystrix (Vagrant)	Monitoring state of Hystrax and send result to the dashboard		
Hystrix (Docker)	Monitoring the state of the Hystrax and send the result to the dashboard		
Order	Create an order		
Send data to server	Sending the data to the server		
Order processing	Process the order		

2.4 Component Diagram.

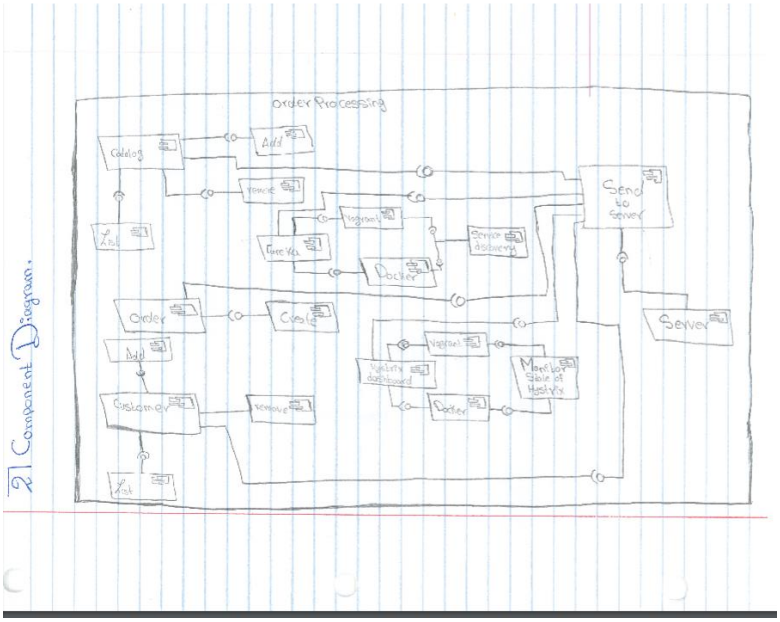


Figure 3_Component Diagram

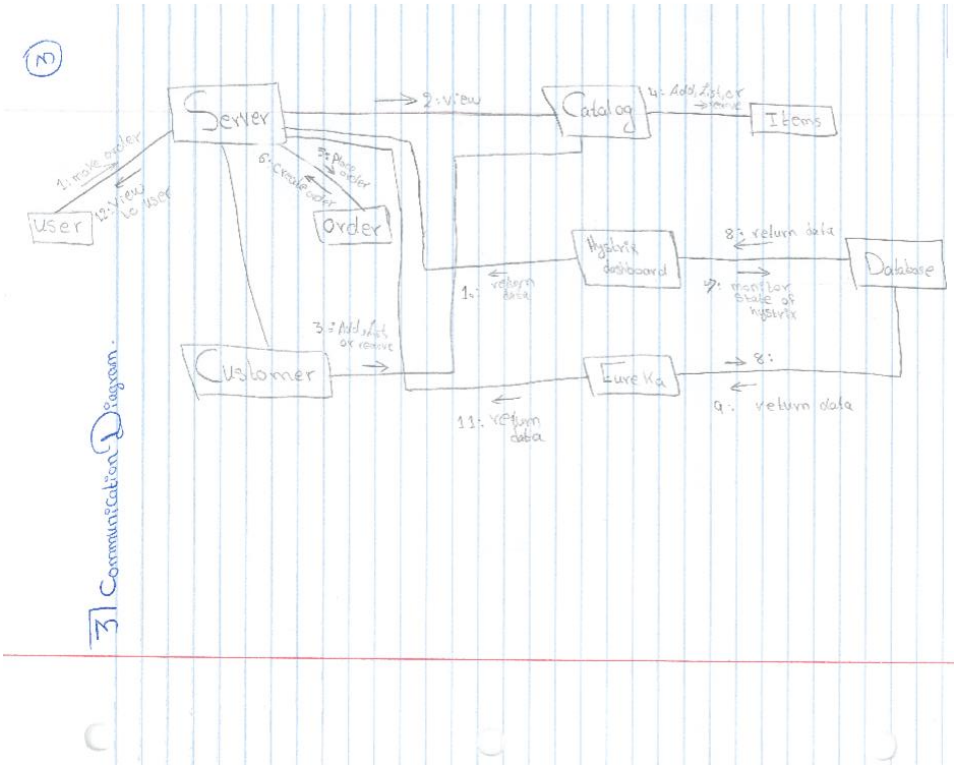
The components are as follows:

Components	**Functionality**
Eureka (Vogrant)	Service Discovery
Eureka (Docker)	Service Discovery
Catalog	View catalog
List (Catalog)	List the catalog
List (Customer)	List the customer's items
Add (Catalog)	Add an item to the catalog

Add (Customer)	Add an item to the customer's list
Remove (Catalog)	Remove an item from the catalog
Remove (Customer)	Remove an item from the customer's list
Hystrix (Vagrant)	Monitoring state of Hystrax and send result to the dashboard
Hystrix (Docker)	Monitoring the state of the Hystrax and send the result to the dashboard
Order	Create an order
Send data to server	Sending the data to the server
Order processing	Process the order

Some packages that the program is using are: Java JDK, Maven, and Docker. Some other packages that the system might be using are: A security package, Persistence package, and Data Base package.

2.5 Communication Diagram.



Basically, the communication diagram shows the relationships between the components of the project. In this project the server (localhost) receives all the data and displays it on the screen. Of course the programmer has the ability to list, remove, or add data to and from the system (e.g. Catalog) and has the ability to manipulate the system's security features. The user has the ability to view catalog, has the ability to add, remove, or list items that he/she have, and has the ability to order. All of this benefits are defined as use cases that draws a guideline to better software. On the diagram above you can see that the server is linked to nearly to every use case in the diagram because the server is the place the users navigate to view all the data thus being linked with the use cases fulfil this requirement. The catalog is linked with items and the customer because as a customer you need a catalog to select items from and make an order thus the catalog being linked with the customer and items fulfils this requirement. Finally, there are some back-end requirements for the project. In this project, the back-end development were monitoring the state of the Hystrix, view the data on the "Hystrex Dashboard", and service discovery (Eureka). These data needed to be saved in some sort of a database and needed to be sent to the server. The "Hystrix dashboard" and Eureka being connected to a database whilst everything is connected to the server fulfils that requirement.

2.6 Reflection.

In trying to understand the project, I noticed a couple of reasons why these diagrams where different than the diagrams that were formulated in the class. One of the reasons that this project needed more requirements or use cases to be implemented in order to reach one's goal (the project scope is larger than the one we had in class). The other reason is that the one in class doesn't have monitoring on the system whereas this project requires monitoring the state of the hystrex and service directory which required more work and cases to be implemented in order to fulfil those requirments.