Scalable and Distributed Computing

Management system for Trung Nguyen Coffee Company

Pham Hoang Minh - ITDSIU18012

Phan Quoc Khoi - ITDSIU18028

1. Project requirement

Distributed system has become a trend in the industry. Most of the major corporation such as Google and Facebook use distributed server to handle massive number of requests around the world. In this project, we are required to design a system for a corporation named Trung Nguyen. The main products that the company provides are coffee and foods. After a sequence of discussion, a project requirement is provided in [1] and the output system should be 3-tier.

2. Methodology

Before going any further to the design details, we want to provide some assumptions:

i. All activities related to buying and selling a product will take place offline. This assumption is reasonable since if it happens online, the system should be capable of choosing branch nearest to the customer location, which is inaccessible within the budget of implementation.

ii. All discount or promotional programs focus solely on reducing prices of one or more products during a certain amount of time. The aim of the assumption is to simplify the system design so that both employees in business and IT can understand.

iii. If a product has a discount applied by general manager (global discount), a product will not receive any discount from local manager (local discount) until the global discount expires. If a product already has a local discount, a general manager can overwrite that local discount into a global discount (and the local discount will no longer be valid).

2.1. System design

2.1.1 Use case diagram

Before implementation, we must model the real system. The initial step that needs to be taken is to identify people participating in the system and their incentives. From [1], we find out that there are four users participating in the system: general managers, local managers, sales staffs, and customers. [1] also clearly states the incentives of each user. A customer can only see the product information (and cannot buy products due to our assumption). A sales staffs can only record transactions and provide bills. A local manager can generate a report on the shop's sales and apply local discount. A general manager can get sales report from any branch and can apply global discount. Local and global discount mechanisms should follow the assumptions. All mentioned information can be presented by the Use Case Diagram (See Figure 1). Details on each use case is mentioned in Appendix A.

Figure 1: System use case diagram

2.1.2. Entity-relationship diagram (ERD):

After constructing use case diagram, the next step is to design the database management system to model the data storage model. In this scenario, we use the entity-relationship diagram (ERD), which is a diagram modelling tables in the database as entities, and references from one table to another as relationship. From the use case diagram and project description, we can derive the following ERD. We do not use weak entity since weak entity will make the model more vulnerable to bugs in the future. Owing to our assumptions, we only implement discount or promotional programs focusing on reducing products' price for a certain amount of time.

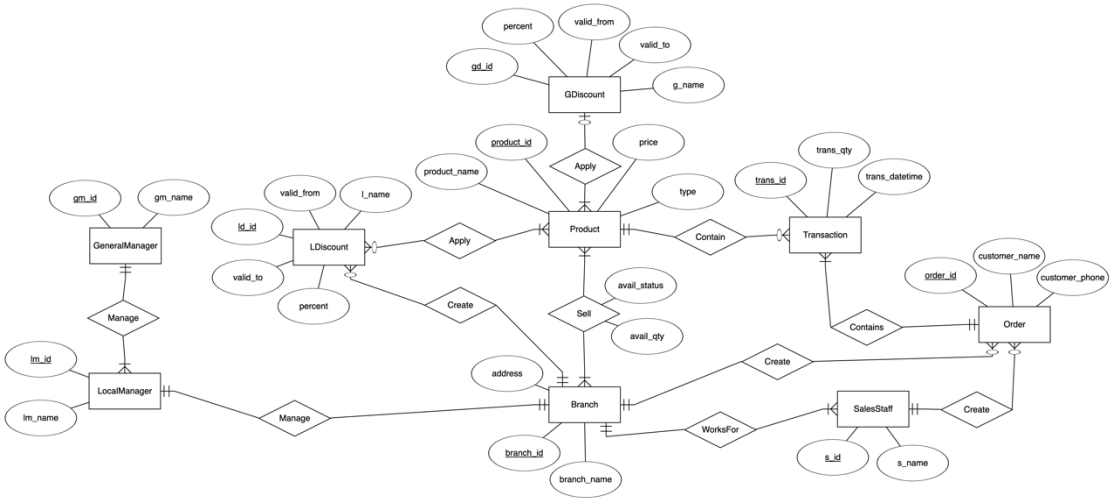

Figure 2: Entity-Relationship Diagram for Trung Nguyen Management System

2.1.3. Deployment diagram:

To describe the resemblance of the management system, we use the deployment diagram, which treats physical devices as nodes, and communications between two devices as edges. We want to propose two deployment model for the Trung Nguyen business. The first one is based on the assumption that Trung Nguyen starts up with relatively high budget. We call it model A (Figure 3).

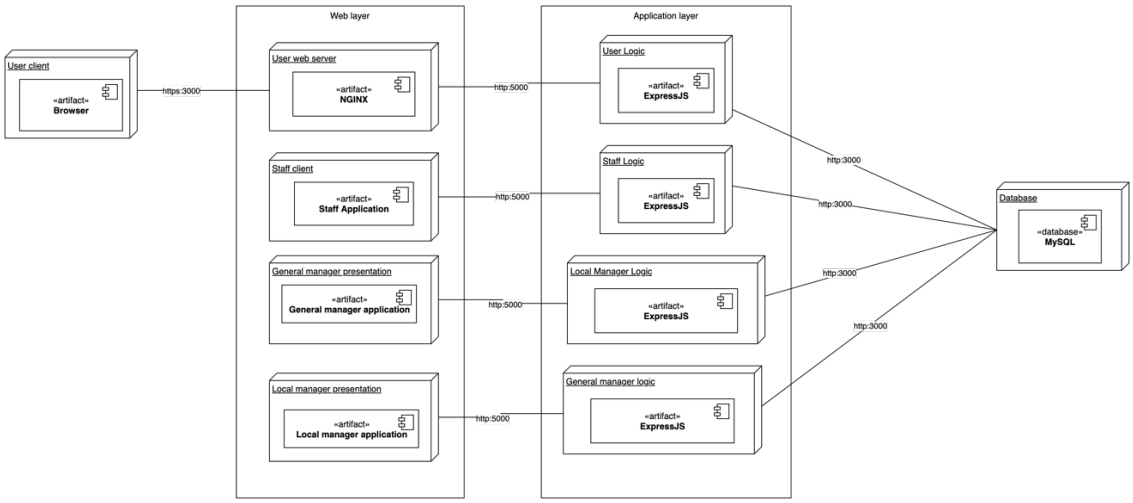

Figure 3: Deployment diagram for model A.

In model A, we provide a complete version of the management system of the company. The goal of this model is that each type of user (general manager, local manager, sales staff, and customer) will have a separate communication line, preventing one user from gaining access to another user's privilege. Nonetheless, due to the separation in each participant's communication channel, the number of physical servers needed to deploy is unachievable if the company does not have sufficient budget. According to [3], the average maintenance cost for one server is roughly $150-$250/month. As a result, in case of model A, the company needs a total budget of $600-$1,000/month to maintain the operation of the system. Hence, we propose another system model in order to mitigate the high maintenance price issue (see Figure 4).

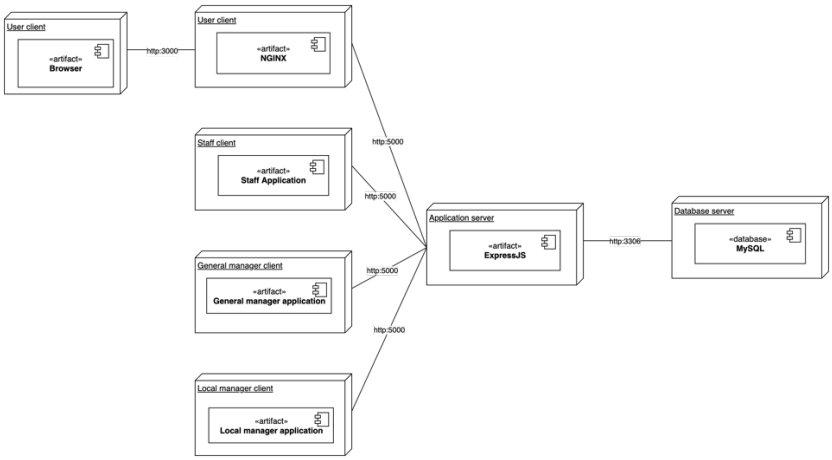

Figure 4: Deployment diagram for model B.

In model B, instead of splitting communication line for each user, we merge all communication stream into one and use only one server to handle communication of users. The design cuts the price by half; however, the security level will be reduced if the request-handling mechanism is not properly designed. In addition, the design increases the traffic per one server, which will worsen the system performance and degrade the server's life expectancy. As a result, it is advisable that the company should have a long-term plan for converting from model B to model A if they start up with low budget.

2.1.3. Class diagram

We also want to provide a conceptual class diagram in [5] so that readers can have an overview about the project scenario (See Figure 5 and please go to link in [5] for better image quality).

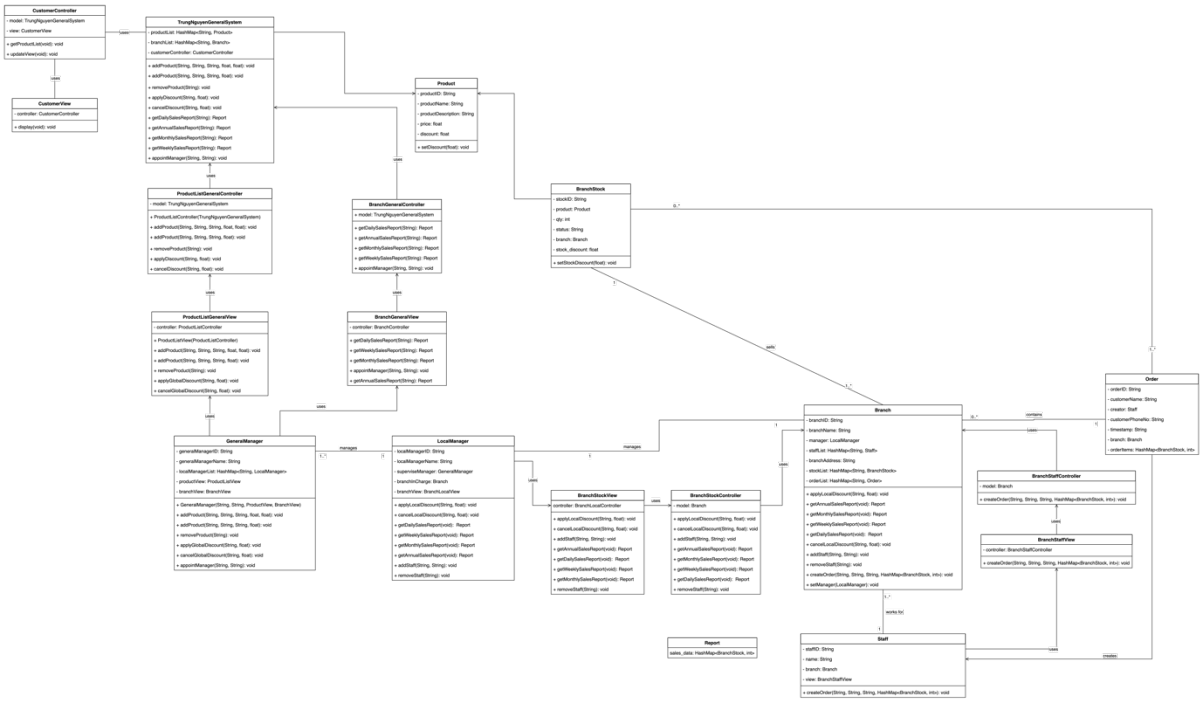

Figure 5: Class diagram for Trung Nguyen management system

2.2. System implementation

In terms of implementation, we implement a web service that can be used for further development (see Appendix A). Below is one example of a customer wants to retrieve information from our web service using Postman as a client application.

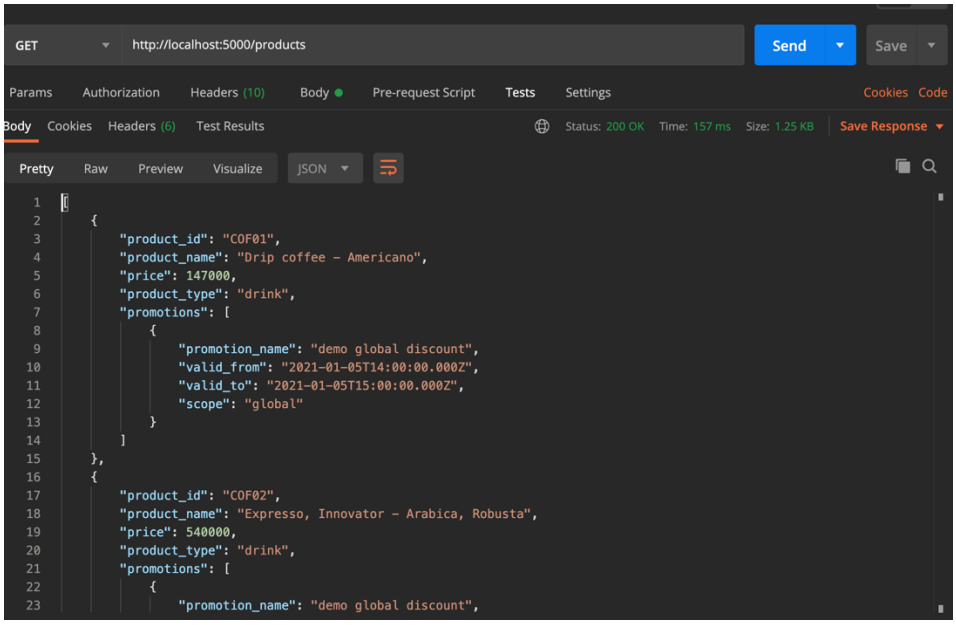

Figure 5: An illustration on web service

In case of model B, we implement a RESTful API with JSON serialization as a mean of communication between client and our system. In this project, we have the following routes:

Routes	Type	Description
/products	GET	Retrieving all products' information
/sales_staff/login	POST	Authentication for sales staff
/sales_staff/payment	POST	Sales staffs' adding new order to the system
/local_manager/login	POST	Authentication for local manager
/local_manager/local_discount	POST	Local manager's applying local discount on certain products
/general_manager/login	POST	Authentication for general manager
/general_manager/global_discount	POST	General manager's applying global discount on certain products

It is obvious that route /products will be publicly available to every user; however, routes for payment and discount should be protected. Therefore, for any users with POST privilege, we want to authenticate them before they can perform their authorized access. Thus, they need

to send a POST request to their relevant login with the provided username and password to get a "ticket" (or so-called a token) for performing actions, and for subsequent request, they have to include the token in the header of the request to validate their identity (token-based authentication - see Appendix B). For the web server tier, we can implement an appropriate front-end to handle login operation; however, this is out of the scope of this report. The implementation code is published in [2].

In case of model A, instead of implementing a login route for each user involving in the POST operation, we can deploy local area network (LAN) to handle the communication line of general manager to the database server and wide area networks among shops and the headquarter to handle communication between sales staff, local manager, and database server.

5. Conclusion

In brief, there are two models that Trung Nguyen company needs to consider: model A with high security level but high budget, and model B with low cost but low security level. If a business starts up with model B, an authentication mechanism should be implemented to prevent unauthorized access. A long-term plan for converting from model B to A is advisable for system performance and security.

REFERENCES

[1] Dr. Mai Hoang Bao An. "Course Project"
[2] https://github.com/minhrongcon2000/TrungNguyenCoffeeSystem
[3] https://bobcares.com/blog/server-maintenance-cost/
[4] https://www.w3.org/2002/ws/
[5]https://github.com/minhrongcon2000/TrungNguyenCoffeeSystem/blob/main/diagram/class_diagram_coffee_shop.png

APPENDIX

A. Background

1. Web service

According to W3C [3], a web service is a software system designed to support interoperable machine-to-machine interaction over a network. In other words, a web service is an API that allows machines to communicate to each other via networks. Noticing that a web service does not need a graphical user interface (GUI) to improve user experience as web site does; instead, they usually act as a middleware to handle user request after filling out forms in the web GUI (e.g., login form). In this report, we used RESTful protocol with JSON serialization, in which requests from client to a web service API are divided into four categories:

- GET: retrieve information from a relation
- POST: add new instance to a relation
- PUT: update an existing instance in a relation
- DELETE: delete an existing instance out of a relation

A web service has its own IP address so that clients can perform a request to the server underlying the service without a GUI. Furthermore, to provide GET requests to more than one relation in the database, a web service has a number of different routes for each relation. For example, if Trung Nguyen company wants to provide a route for a client to send GET request to its server to get product information, they could set the route as

http://api.trungnguyen.com/product (1)

and if they want to make another route for company's branches, they could set another route for it as

http://api.trungnguyen.com/branch (2)

Noticing that one route can handle more than one kind of request to a relation (for example, route (1) can handle customer's request to get product information and general manager's request to add a new product to the database). For convenience, in some documentations, the route is denoted as /<name of routes> since the IP address (or domain name) of the web service is publicly known.

2. Token-based authentication

Token-based authentication is a web authentication protocol that allows users to enter their username and password in order to obtain a token which allows them to fetch a specific resource - without using their username and password.

B. Use case description

UC	Use case name	Use case info	Flows
1	Make payment	Triggers/Goals: Customer wants to pay for one or more products at a specific Trung Nguyen shop Actors: Customer, sales staff	Main flow: 1. Customer brings products to the counter 2. Sales staff will scan for the product's ID 3. System returns product details 4. Sales staff takes customer's name, customer's phone number and enter it to the form 5. Sales staff tells customer the total price 6. Customer gives money and sales staff returns change (if any) 7. Sales staff submits the form 8. System saves the details and notify the transaction is processed successfully 9. Sales staff gives customer bill Exception: 8a. Transaction is failed to process 1. Sales staff uses paper to record the bill and give it to the customer 2. Sales staff immediately contact the local manager

2	Get product information	Triggers/Goals: Customer wants to search for Trung Nguyen product's information Actors: Customer	Main flow: 1. Customer accesses to Trung Nguyen's web page 2. System returns a list of products Exception: 2a. Server has internal error 1. System terminates
3	Get shop's sales report	Triggers/Goals: Local shop's manager wants to have sales report of his own shop Actors: Local manager	Main flow: 1. Local manager chooses which kind of report to be generated -- if the local manager chooses weekly report, do step 2 -- 2. System returns the total price that the shop earns, and the product with its total sales this week. -- if the local manager chooses monthly report, do step 3 -- 3. System returns the total price that the shop earns, and the product with its total sales this month. -- if the local manager chooses yearly report, do step 4 -- 4. System returns the total price that the shop earns, and the product with its total sales this year. Exception: 2a, 3a, 4a. System has internal error 1. System terminates
4	Create local discount	Triggers/Goals: Local shop's manager wants to launch a local discount program for a product Actors: Local manager Invocation constraints: Product ID exists Product does not have any discount program going on	Main flow: 1. Local manager enters the product ID 2. Local manager enters the name of discount program 3. Local manager enters the discount factor 4. Local manager enters the valid period of discount program 5. System saves the details, and updates the new local prices in the web pages Exception: 4a. System has internal error 1. System terminates

		Product does not have a global discount program	4b. Product already had discount program (global discount or on-going discount) 1. System notifies setting discount failed and shows the on-going discount Post-obligations: Reset price after valid time (stated in step 4) Note: Contact general manager if the promotion/discount is not implemented
5	Cancel local discount	Triggers/Goals: Local shop's manager wants to cancel discount program before expiration Actors: Local manager Invocation constraints: The discount program has been launched	Main flow: 1. Local manager enters the launched program ID 2. System returns the details 3. Local manager presses Cancel button 4. System notifies cancellation success Exception: 2a. ID not found 1. System asks the manager to re-enter the program ID
6	Get sales report from a shop	Triggers/Goals: General manager wants to get sales report from a specific shop Actors: General manager Invocation constraints: Shop ID exists	Main flow: 1. General manager enters the shop ID to get report 2. System returns the report Exception: 1a. shop ID not found 1. System asks user to re-enter the shop ID
7	Create global discount	Triggers/Goals: General manager wants to create a global discount for a product Actors: General manager	Main flow: 1. General manager enters the product ID 2. General manager enters the name of discount program 3. General manager enters the discount factor

		Invocation constraints: Product ID exists Product does not have any global discount program going on	4. General manager enters the valid period of discount program 5. System saves the details, and updates the new prices in the web pages Exceptions: 5a. Product has an ongoing local discount 1. System automatically overwrite the local discount to the global discount 5b. Product has an ongoing global discount 1. System notifies the discount program setting failed and shows the on-going global discount Post-obligations: Reset price after discount expired Note: Contact developer team if discount program is not implemented
8	Cancel global discount	Triggers/Goals: General manager wants to cancel a global discount program before expiration Actors: General manager Invocation constraints: The global discount program has already been launched	Main flow: 1. General manager enters the launched program ID 2. System returns the details 3. General manager presses Cancel button 4. System notifies cancellation success Exception: 2a. ID not found 1. System asks the manager to re-enter the program ID
9	Add product	Triggers/Goals: General manager wants to add new product Actors: General manager Invocation constraints: Product ID does not exist	Main flow: 1. General manager enters the new product ID, name, price, and description. 2. System saves the details and updates the web pages Exception: 2a. Product ID exists 1. System asks the manager to re-enter the product ID

10	Remove product	Triggers/Goals: General manager wants to stop providing a product Actors: General manager Invocation constraints: Product ID exists	Main flow: 1. General manager enters the product ID to be removed 2. System saves the details and update the web pages Exception: 2a. Product ID does not exist 1. System asks the manager to re-enter the ID
11	Add staff	Triggers/Goals: Local manager wants to add a new employee Actors: Local manager Invocation constraints: Staff ID does not exist	Main flow: 1. Local manager enters new employee ID and employee information (specified later in project) 2. System saves the detail to the database Exception: 1a. Employee ID exists 1. System notifies the local manager and asks the manager to repeat the process
12	Remove staff	Triggers/Goals: Local manager wants to fire an employee Actors: Local manager Invocation constraints: Staff ID exists	Main flow: 1. Local manager enters the ID of employee he wants to remove 2. System updates the database Exception: 1a. Employee ID does not exist 1. System notifies the local manager and asks local manager to repeat the process
13	Open a new branch	Triggers/Goals: General manager wants to add a new branch to the database Actors: General manager Invocation constraints: Branch ID does not exist	Main flow: 1. General manager enters the branch ID, name, and address 2. System updates the database Exception: 1a. Branch ID exists 1. System notifies the general manager and asks general manager to redo the process

		Desired branch has been opened in reality	
14	Appoint a new manager to a branch	Triggers/Goals: General manager wants to change the manager of a branch Actors: General manager Invocation constraints: Local manager ID does not exist Branch ID exists	Main flow: 1. General manager enters the branch ID that needs new manager 2. General manager enters the new local manager ID, as well as name, and other information 3. System updates the database Exception: 2a. Local manager ID exists 1. System notifies the error 2b. Branch ID does not exist 1. System notifies the error