KTPM1 – Group 06

E-commerce Web Application
Software Architecture Document

Version <2.0>

Revision History

Date	Version	Description	Author
01/12/2020	1.0		Phạm Vũ Duy, Hồ Nguyễn Huy Hoàng
09/12/2020	2.0	Add use case diagram and specification V2.0	Phạm Vũ Duy, Hồ Nguyễn Huy Hoàng

Table of Contents

Software Architecture Document

1. Introduction

"Dang's Company" is an E-commerce Web Application, its aim is to bring the buyers and the sellers together.

This document elaborates the software architecture document for the "Dang's Company E-commerce Web Application". The system architecture is abstracted into many view and components which will be explain in this document.

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture the significant architectural decisions which have been made on the system.

1.2 Scope

The software architecture document applies to each static and dynamic aspect of the system.

Under the static behavior of the system, the document discusses the class diagram and other static architecture designs. Dynamic aspects of the system are elaborated using case realizations.

1.3 Definition, Acronyms and Abbreviations

MVC – Model View Control architecture

DWA - Dang's Company E-commerce Web Application

DB – DataBase

1.4 Reference

https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm#Definitions,%20Acronyms%20and%20Abbreviations

https://www.slideshare.net/PasinduTennage/sample-software-architecture-document

https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/process/artifact/ar_sadoc.htm

1.5 Overview

This document will present a detailed analysis of the architecture of Dang's Company E-commerce Web Application. The further section will cover the architectural goals including the architectural constraints.

2. Architectural Goals and Constraints

2.1 Server side

The DWA will be hosted at "Heroku" JSP server. Mongo DB will be used as central database server. All communication between server and client will using HTTP/HTTPS (free SSL come along with Heroku) – a standard communication protocol.

2.2 Client side

User will be able to access DWA only online. Users/Clients are expected to use a modern web browser which

can support Bootstrap 4.0 to get full experience:

Recommend mobile devices browser

	Google Chrome	**Firefox**	**Safari 9.0 +**	**Android Browser 4.4+**
Android	✓	✓	N/A	✓
iOS	✓	✓	✓	N/A

Desktop

	Chrome 45.0+	**Firefox 38.0+**	**IE 10.0+**	**Microsoft Edge 12.0+**	**Opera 30.0+**	**Safari 9.0+**
Mac	✓	✓	N/A	N/A	✓	✓
Windows	✓	✓	✓	✓	✓	N/A

2.3 Security and privacy

The central security will be handled by the member of the development team. They will be given the full access not only in the application level but also in database level. Creating account for the staff and the owner of the shop are done by administrator. When creating an account, user can choose their password and this password can be changed anytime by them. All the password is encrypted both on database or on communication process between client and server in order to ensure high level of security. The user information will only be seen by the shop owner and the administrator.

2.4 Reliability and Availability

The system will be subjected to several testing step (Unit testing, Integration testing, System testing including Security and Performance testing) before being released to ensure that the system is reliable and worked as intended. The DB Central which is placed on MongoDB, ensure both the security with TLS/SSL encryption and performance such as low latency and response time.

2.5 Performance

The server responds to any request from client within the web script timeouts (30 seconds), also the system performance can depend on available hardware, networks and internet connection capabilities. Therefore, the actual performance can be determined only after the system is deployed and tested. Our aim is to make the loading time on client side become ideal which is lower than 2 seconds.

2.6 Portability and Reuse

The DWA is design to be a complete cosmetic website. But can be extend to sale many kinds of product. In order to maintain reusability, the web using Handlebar Template Engines which can be reuse. Best practice of RUP during development combine with Mongo DB make the structured is well layered.

2.7 Development tools

The project is the combination of many tool:

Programing tool: Visual Code

Database: MongoDB

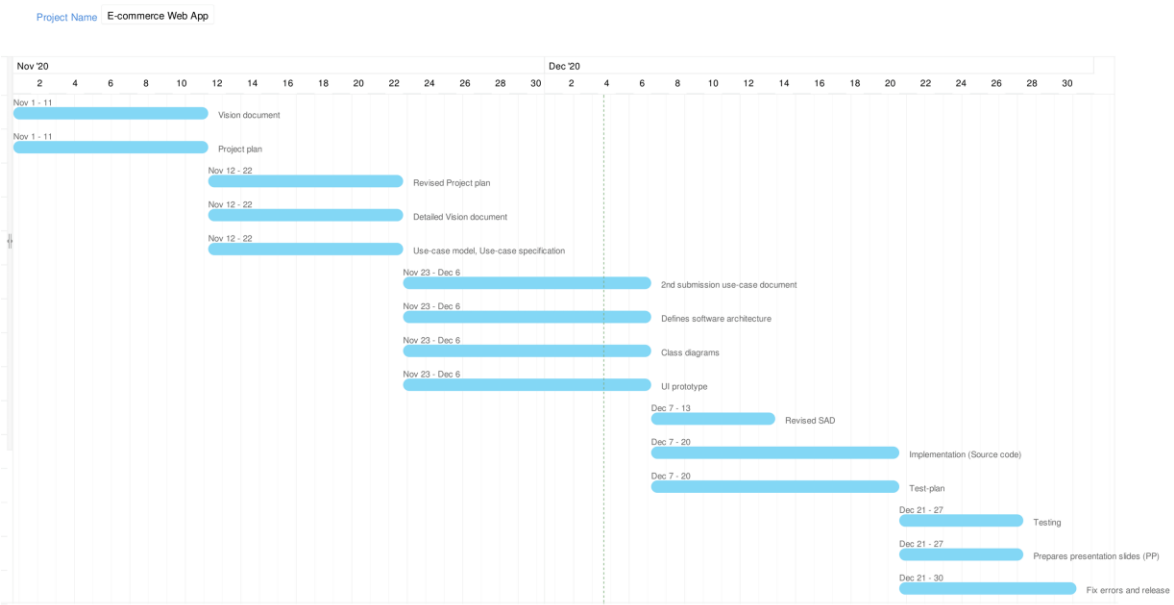
UI Prototype: figma

Meeting platform: Slack, Discord

Schedule: Trello

2.8 Schedule

The development process is follows by the combination of Agile and RUP workflow. There are six sprints, each has their own workload and document:



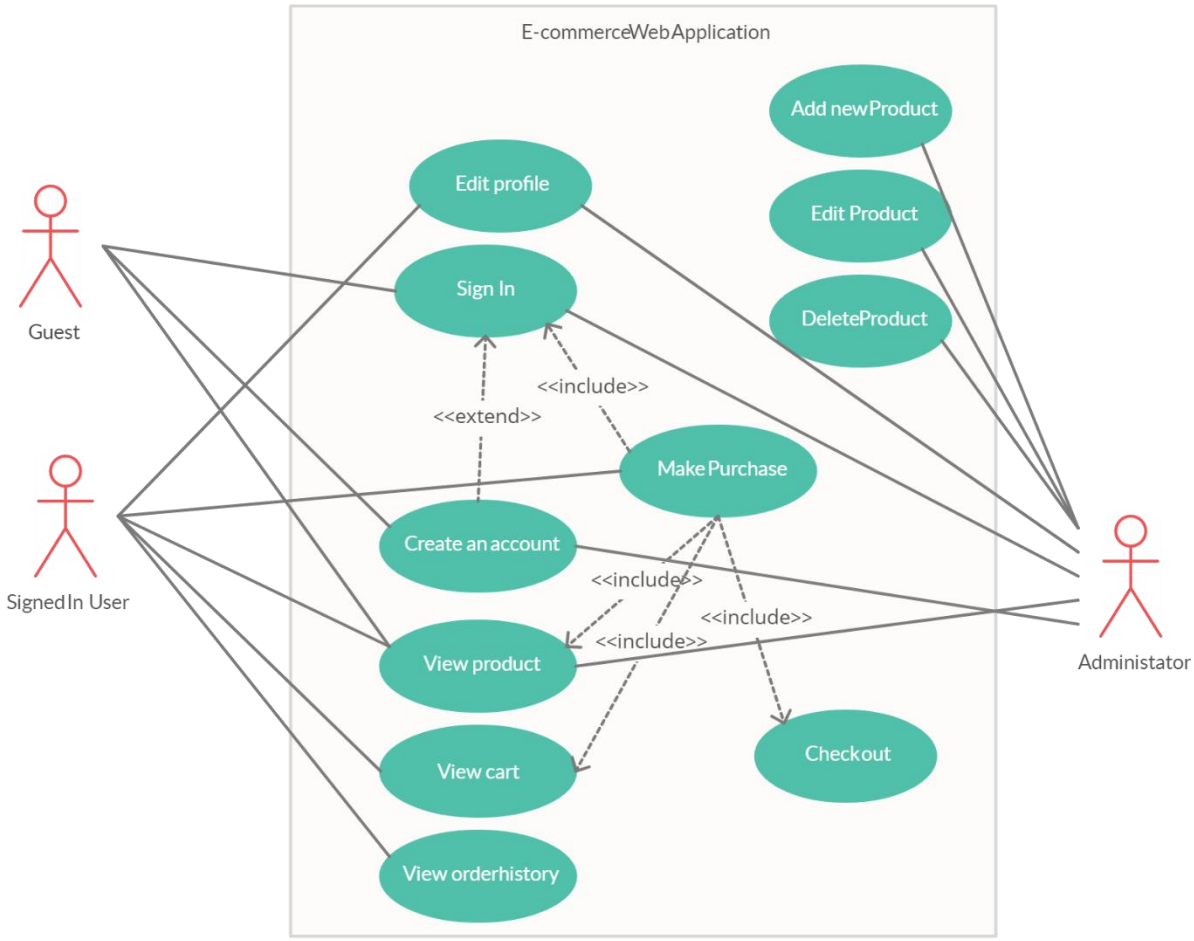
*Due to the limitation of Zoho Free-trial version that only allow maximum 3 member in each project. So, we are not able to show the specific assignment for each member in the Gantt chart above. This is the schedule with the specific assignment for each team member:

Phases	Interation No.	Tasks and Artifacts	Assignee	Start Date	End Date
Inception	1	- Vision document	Nguyễn Phúc Thịnh	1/11/2020	11/11/2020
		- Project plan	Huỳnh Nhật Nam		
Elaboration	1	- Revised project plan	Huỳnh Nhật Nam	12/11/2020	22/11/2020
		- Detailed vision document	Nguyễn Phúc Thịnh		
		- Use-case model, use-case specification	Huỳnh Nhật Nam, Nguyễn Phúc Thịnh		
	2	- 2nd submission use-case document	Huỳnh Nhật Nam, Nguyễn Phúc Thịnh	23/11/2020	6/12/2020
		- Defines software architecture	Phạm Vũ Duy, Hồ Nguyễn Huy Hoàng		
		- Class diagrams	Phạm Vũ Duy, Hồ Nguyễn Huy Hoàng		
Construction	1	- Revised SAD	Phạm Vũ Duy, Hồ Nguyễn Huy Hoàng	7/12/2020	13/12/2020
		- UI prototype	Phạm Vũ Duy, Hồ		

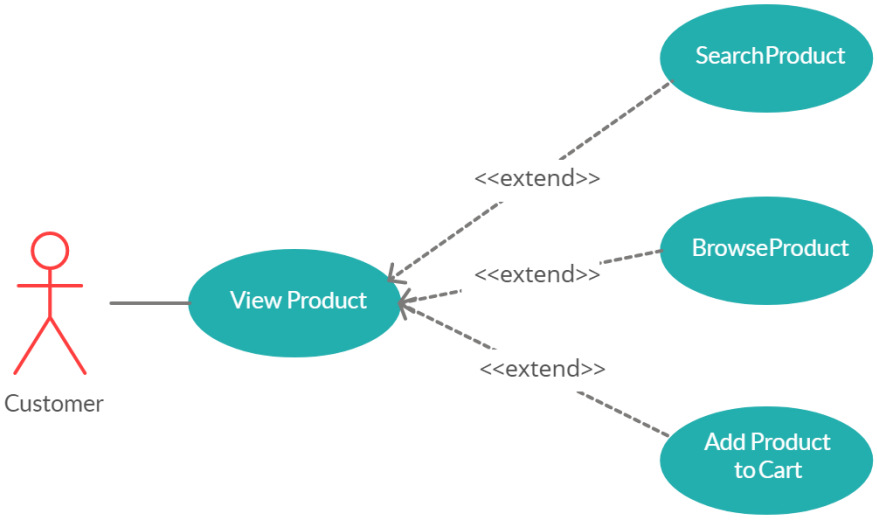
			Nguyễn Huy Hoàng		
		- Implementation (Source code)	All team members	7/12/2020	20/12/2020
		- Test-plan	Mai Đăng Khánh		
		- Release (See 4.2.3 for better details)	Hồ Nguyễn Huy Hoàng	14/12/2020	20/12/2020
	2	- Testing	Mai Đăng Khánh	21/12/2020	27/12/2020

3. Use-Case Model

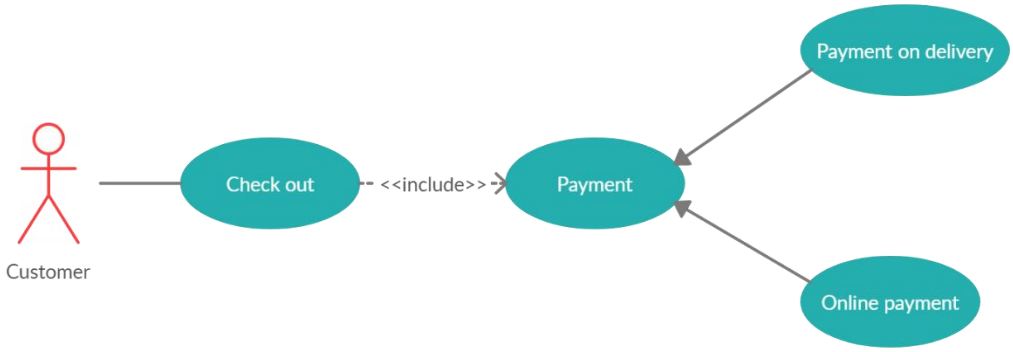
3.1 Use-case diagram



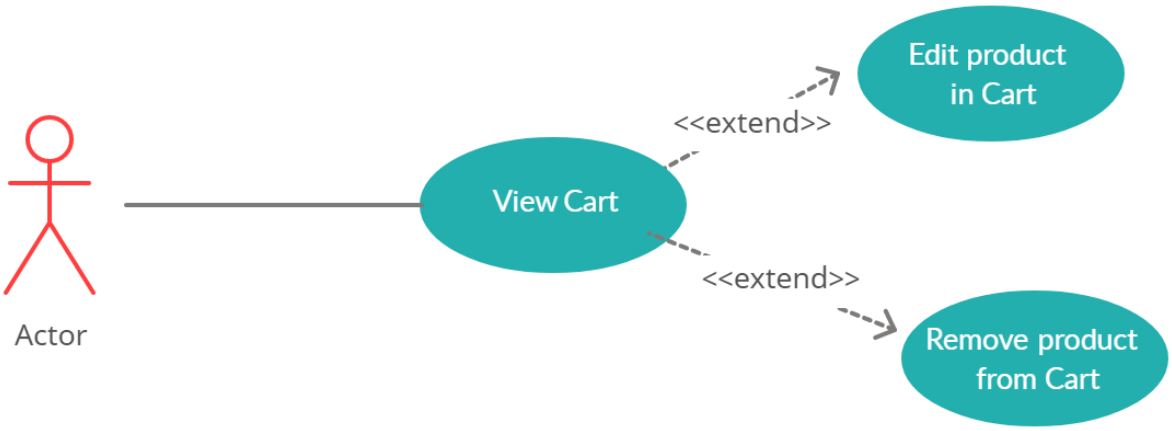
- View product model:



- Check Out model:



View Cart model:



3.2 Use-case Specifications

3.2.1 Use-case: Sign in

Use case Name	Sign in
Brief description	This use case describes how a user can sign in
Actors	Guest, Administrator
Basic Flow	1. User enters username and password 2. Return to homepage if user signed in successfully
Alternative Flows	**Alternative flow 1: User enters wrong password** 1. System displays an error message 2. Continue step #1 of basic flow **Alternative flow 2: User enters non-existent username** 1. System displays an error message

	2. Continue step #1 of basic flow
Pre-conditions	User goes to login page through 1 of 3 ways: - Click on sign in button at the top right of the home page - User is not signed in and clicks on view cart button at the top right of the home page - User is not signed in and adds a product to cart
Post-conditions	Guest successfully signs in and return to homepage

3.2.2 Use-case: Edit profile

Use case Name	Edit profile
Brief description	This use case describes how a user can edit their profile information
Actors	Signed-in user, Administrator
Basic Flow	1. User choose my-profile in navigation bav 2. User choose edit profile in my-profile page 3. User choose and edit their information they want 4. User click on save button 5. System comeback to my-profile page
Alternative Flows	**Alternative flow 1: User enters invalid type of information** 1. System displays an error message 2. Continue step #3 of basic flow **Alternative flow 2: Blank on any line** 1. System displays an error message 2. Continue step #3 of basic flow
Pre-conditions	User signed-in system with their account
Post-conditions	User successfully edit their information and return to my-profile page

3.2.3 Use-case: Create an account

Use case Name	Create an account
Brief description	This use case describes how a user can create an account
Actors	Guest, administrator
Basic Flow	1. At sign in page, user clicks on sign up button. 2. User goes to sign up page. 3. User enters account's name 4. User enters password 5. User enters password again 6. User clicks sign up 7. Goes back to sign in page
Alternative Flows	**Alternative flow 1: User enters username that's already taken** 1. From #6 of the basic flow, system displays error message 2. Continue from step #3. **Alternative flow 2: User re-enters wrong password** 1. From #6 of the basic flow, system displays error message 2. Continue from step #5.
Pre-conditions	User goes to login page and clicks sign up button
Post-conditions	User successfully signs up and returns to sign in page

3.2.4 Use-case: Make purchase

Use case Name	Make purchase
Brief description	This use case describes how a user can make a purchase
Actors	Signed in customer
Basic Flow	1. At homepage, user goes to their cart by clicking on cart button 2. User click on purchase button 3. User click on checkout button. 4. User check their information again 5. User click on confirm button

Alternative Flows	**Alternative flow 1: User's shopping cart is empty** 1. From #3 of the basic flow, systerm display error message 2. Systerm goes to homepage **Alternative flow 2: User information is wrong** 1. From #4 of the basic flow, user re-enter their information 2. Continue step #5
Pre-conditions	User goes to homepage User has product in shopping-cart User has correctly information in profile
Post-conditions	User successfully make a purchase and waiting for an email to confirm

3.2.5 Use-case: View order history

Use case Name	View order history
Brief description	This use case describes how a user can view details about their order history
Actors	Signed in user
Basic Flow	1. User goes to homepage 2. User clicks on my-profile 3. User choose order history on my-profile page 4. System show order history
Alternative Flows	**Alternative flow 1: User don't have any order before** 1. From #4 of the basic flow, system displays no order to show 2. System comeback to my-profile page
Pre-conditions	User signed in succesfully User go on my-profile page
Post-conditions	User view order history

3.2.6 Use-case: View product

Use case Name	View product
Brief description	This use case describes how a user can view details about product
Actors	Guest, administrator, signed in user
Basic Flow	5. User goes to homepage 6. User clicks on product 7. System displays details of products
Alternative Flows	**Alternative flow 1: User can't view the product details** 3. From #3 of the basic flow, system displays no details to show 4. Continue step #2
Pre-conditions	User goes to homepage
Post-conditions	User view a product's details information

3.2.7 Use-case: Search product

Use case Name	Search product
Brief description	This use case describes how a user can search a product by search bar
Actors	Guest, administrator, signed in user
Basic Flow	8. User goes to homepage 9. User clicks on search bar 10. User enter a text 11. System displays list of products according to the searching result 12. User clicks on a product 13. User goes to a page which displays that product's information
Alternative Flows	**Alternative flow 1: User can't not find the product**

	5. From #4 of the basic flow, system displays no product is found 6. Continue step #2
Pre-conditions	User goes to homepage
Post-conditions	User view a product's information

3.2.8 Use-case: Browse product

Use case Name	Browse product
Brief description	This use case describes how a user can browse product
Actors	Guest, administrator, signed in user
Basic Flow	1. User goes to homepage 2. Using navigation bar, user can see types of product 3. User clicks on a type on navigation bar 4. System displays a list of products belongs to that specific type 5. User clicks on a product 6. System displays a page of that product's information
Pre-conditions	User goes to homepage
Post-conditions	User views a product's information

3.2.9 Use-case: View cart

Use case Name	View cart
Brief description	This use case describes how a user can view his/her cart
Actors	Signed in user
Basic Flow	1. User clicks on icon of the cart 2. System displays user's cart (a page)
Pre-conditions	- User goes to homepage - User has already signed in

Post-conditions	User views cart

3.2.10 Use-case: Edit product in cart

Use case Name	Edit product in cart
Brief description	This use case describes how a user can edit product in cart
Actors	Signed in user
Basic Flow	1. User views cart 2. User changes quantity of products 3. User adds requirements to products
Pre-conditions	- User goes to homepage - User has already signed in.
Post-conditions	User changes product's quantity in cart or add requirements

3.2.11 Use-case: Remove product from cart

Use case Name	Remove product from cart
Brief description	This use case describes how a user can remove product from cart
Actors	Signed in user
Basic Flow	1. User views cart 2. User clicks on remove button of a product
Pre-conditions	- User goes to homepage - User has already signed in.
Post-conditions	A product is removed from cart

3.2.12 Use-case: Add a product to cart

Use case Name	Add a product to cart

Brief description	This use case describes how a user can add a product to cart
Actors	Signed in user
Basic Flow	1. User browses or searchs a product 2. At product's information page, user clicks on add to cart button 3. System adds product to user's cart
Alternative Flows	**Alternative flow 1: Product is out of stock** 1. After step #2 of basic flow, system displays error message 2. Continue from step #1. **Alternative flow 2: Product is already in cart** 1. From step #3 of basic flow, system increases product's quantity in cart
Pre-conditions	User goes to homepage User has already signed in
Post-conditions	User adds product to cart or increases product's quantity

3.2.13 Use-case: Payment

Use case Name	Add a product to cart
Brief description	This use case describes how a user can do a payment
Actors	Signed in custormer
Basic Flow	1. User goes to shopping cart 2. User click on purchase button 3. User click on checkout button 4. User choose payment methods 5. User click on confirm button
Alternative Flows	**Alternative flow 1: User's shopping cart is empty** 1. From #2 of the basic flow, systerm display error message 2. Systerm goes to homepage **Alternative flow 2: Payment methods is not available** 1. From step #4 of basic flow, system display error message

	2. Goes to #4
Pre-conditions	User goes to shopping cart User has already signed in User has already product in shopping cart
Post-conditions	User successfully to pay for a purchase.

3.2.14 Use-case: Payment on delivery

Use case Name	Payment on delivery
Brief description	This use case describes how a user can choose payment method is pay when receive product
Actors	Signed in custormer
Basic Flow	1. User goes to shopping cart 2. User click on purchase button 3. User click on checkout button 4. User choose payment on delivery 5. User click on confirm button
Alternative Flows	**Alternative flow 1: User's shopping cart is empty** 1. From #2 of the basic flow, systerm display error message 2. Systerm goes to homepage **Alternative flow 2: Payment methods is not available** 1. From step #4 of basic flow, system display error message 2. Goes to #4 and choose other payment methods
Pre-conditions	User goes to shopping cart User has already signed in User has already product in shopping cart
Post-conditions	User successfully to pay for a purchase.

3.2.15 Use-case: Payment online

Use case Name	Payment online
Brief description	This use case describes how a user can choose payment method is pay when receive product
Actors	Signed in custormer
Basic Flow	1. User goes to shopping cart 2. User click on purchase button 3. User click on checkout button 4. User choose payment online 5. User click on confirm button
Alternative Flows	**Alternative flow 1: User's shopping cart is empty** 1. From #2 of the basic flow, systerm display error message 2. Systerm goes to homepage **Alternative flow 2: Payment methods is not available** 1. From step #4 of basic flow, system display error message 2. Goes to #4 and choose other payment methods **Alternative flow 3: Credit card not available** 1. From step #4 of basic flow, systerm display error message 2. Goes to #4 and re-enter.
Pre-conditions	User goes to shopping cart User has already signed in User has already product in shopping cart
Post-conditions	User successfully to pay for a purchase.

3.2.16 Use-case: Add new product

Use case Name	Add new product
Brief description	This use case describes how administrator can add new product

Actors	Administrator
Basic Flow	1. At administrator page, admin clicks on add new product button 2. System displays a new page 3. Admin fills in product's information 4. Admin clicks on add button 5. System adds product to database
Pre-conditions	Administrator signed in with admin account
Post-conditions	Admin adds new product to database

3.2.17 Use-case: Edit product

Use case Name	Edit product
Brief description	This use case describes how administrator can edit product's information
Actors	Administrator
Basic Flow	1. At administrator page, admin clicks on search button 2. Admin enters products name 3. System displays product's information 4. Admin clicks on a field 5. Admin changes its content 6. Admin clicks OK 7. System changes product's content in dadabase
Alternative Flows	**Alternative flow 1: Product don't exist in database** 1. From step #2 of basic flow, admin enters another term 2. Continue to step #6.
Pre-conditions	Administrator signed in with admin account
Post-conditions	Admin edits product's information in database

3.2.18 Use-case: Delete product

Use case Name	Delete product

Brief description	This use case describes how administrator can delete a product
Actors	Administrator
Basic Flow	1. At administrator page, admin clicks on search button 2. Admin enters products name 3. System displays product's information 4. Admin clicks on delete button 5. Admin confirms 6. Systems delete product in database
Alternative Flows	**Alternative flow 1: Product don't exist in database** 1. From step #2 of basic flow, admin enters another term 2. Continue to step #6.
Pre-conditions	Administrator signed in with admin account
Post-conditions	Admin deletes a product in database

4. Logical View

4.1 Overview

4.1.1 The subsystem

The DWA can be divided into 3 main sub-system.

1. Web store
2. Order management subsystem
3. Statical data management subsystem

4.1.1.1 Web store

This subsystem provides all the functionalities that is related to user. The main use cases of this subsystem include

1. User login / Shop owner login
2. Create new user
3. Change password
4. Edit profile
5. Searching for product
6. Add product to cart
7. Make comment and rating

Depending on the level of account (access level), some action may be not be allowed.

4.1.1.2 Order management subsystem

The Order Management System (OMS) is play an important role as a median subsystem between the Web Store and the Data manager subsystem. The OMS main functionalities relate to all the customer ordering task.

- Order management
- Checkout handling

4.1.1.3 Statical data manager subsystem

This subsystem is the main system in charge of managing the central database. The subsystem involved in data access and processing operation which require special algorithms and processing capabilities. Only a few required data are fetched from the database

1. Numbers of product in stock
2. User information
3. Report about shop's monthly income
4. Change product status (add, remove, modify)
5. Comment and rating management

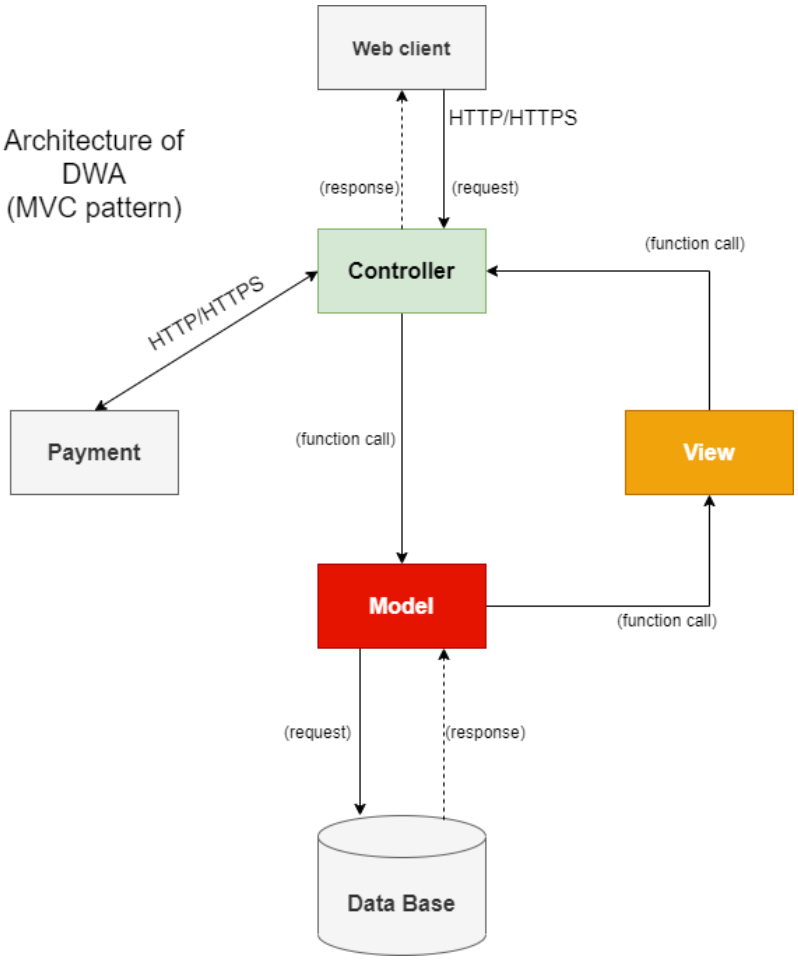
4.2 MVC model

The DWA (Dynamic Web Application) is divided into 3 main components: *Controller*, *Model* and *View*. Those components connected with each other in a strict rule, each of them has a specification job. The reason why this pattern has been chosen is that relate to each specification job of each main components, implementers can work parallel on several component to reduce the developing time. Following that, implementers easy to update and debug each component separately. However, there are also contrast, MVC pattern is hard to deeply understand and it must have

strict rules on method to avoid errors and mixed structured.

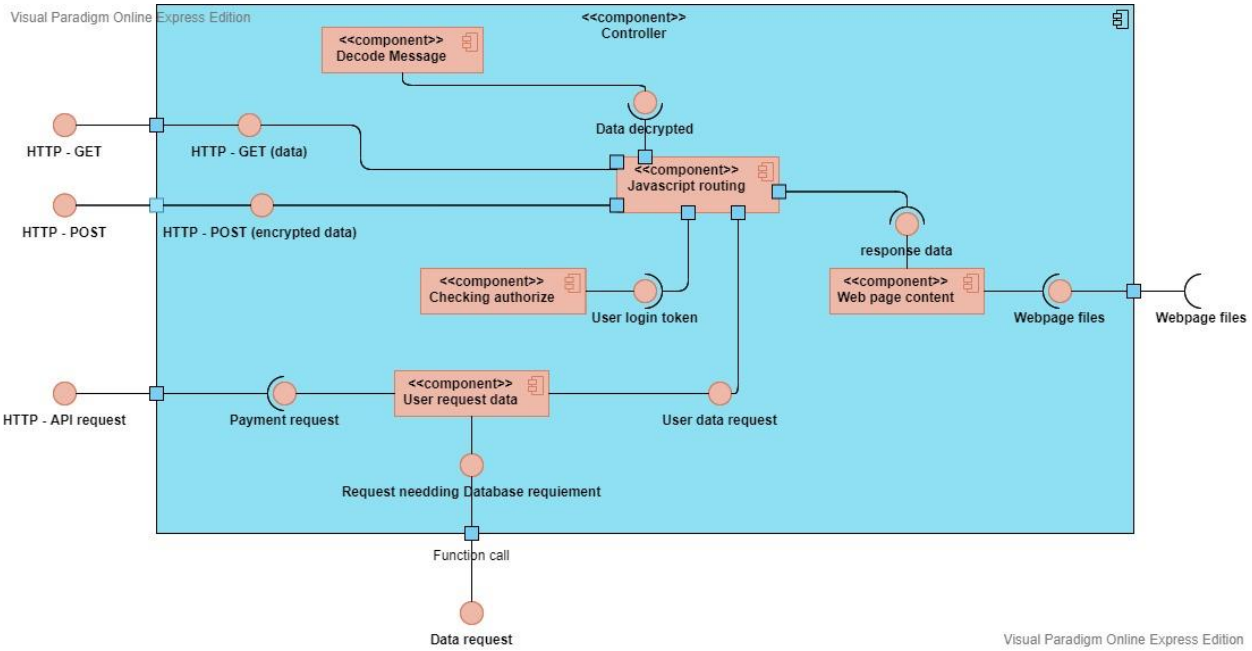
- The Controller handle the request of the client and make function calls to model to get the corresponding data.
- The Model handle the database, that means it make a request to the database such as get data, insert, update, delete. And its also send the data it has got to the view.
- The View in this case is import data to Html, Javascript and stick css style then send back to controller in order to respond the client request.

The pattern is present bellow to explain the communication method between those main components themself and other components



The MVC pattern for DWA architecture

4.2.1 Controller



In controller, client request and response by 2 main HTTP method, they are controlling by Javascript framework in order to receive and response data to client effectively:

- ***HTTP - GET:*** this is the method used to receive or send un-sensitive data from and to client.
- ***HTTP – POST:*** this is the method used to receive and send sensitive data such as User information (username, password, email…), or send webpage content to client.

The main sub-component inside Controller is Javascript routing, its job is to receive the HTTP request, and also response the Webpage content to client through HTTP/HTTPs protocol.

When POST data received, the routing catching that request and decode the message in its, then it pass the data to checking authorize:

- First it can make a request to User request data to get data for checking login success or register criteria.
- Secondly, if client request an authorize action the sub-component will check by using JSON Web Tokens to consider that user is login or not. Then response corresponding to that user action (restrict an un-authorized client from doing authorized action or passing successfully authorized client's request to User request data to go to Model).

The GET request does not need to decrypt, this request is pass through to User request data component to go to Model component.

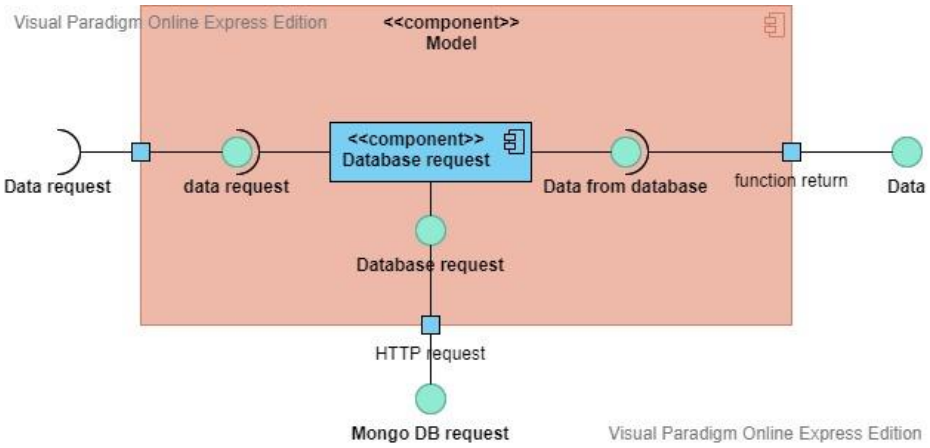
The User request sub-component is used as a bridge to pass request corresponding to third-party or to Model.

The webpage receive the web content from view to response the client.

The main programing language in this component is: Javascript.

Framework: ExpressJS.

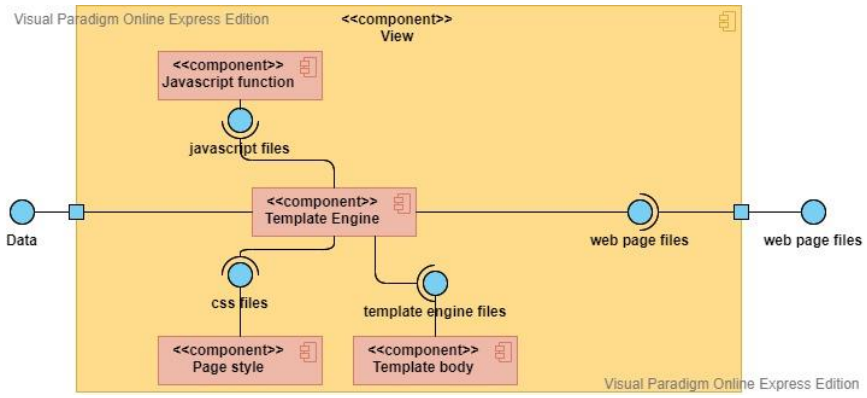
4.2.2 Model



In Model, the database request is control by Database request component, base on request it make a HTTP call to MongoDB's database for: get data, insert data, update data, delete data. Then the response from the MongoDB's will pass to the View component as javascript object.

The main programing language in this component is: Javascript.

4.2.3 View



In View, using Template Engine which contain main layout of the web page, and then combine: the css style for that page, the javascript function for each button click or web page effect, and the template body of particular page (for example, product information page, home page, login page…). The Template Engine then convert all combination to HTML file, CSS files and Javascript files.

Page style may use ***Scss*** and then convert to ***css*** optimize.

Tempalte Engine: Handlebars

Javascript: ES6 structure.

4.2.4 Payment

In Payment, the DWA might use API provide by:

- Momo
- Zalo pay

Due to those third-party payment organization can support for Vietnamese people, and themselves also have many vouchers to encourage customer/ user to buy more products.

Especially, Zalo pay support user make a purchase through their ATM card.

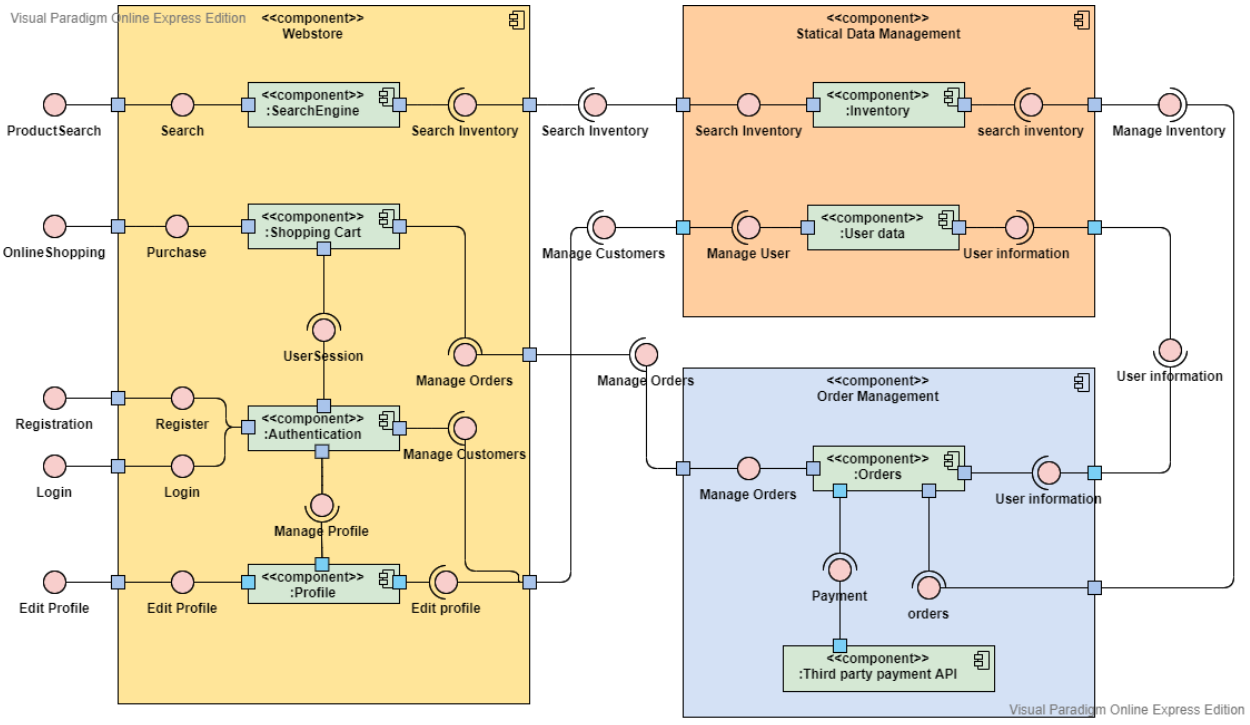
4.2.5 Database

This component is a third-party component, provided by MongoDB, the database is stored in MongoDB server. The recommend servers are:

- AWS/Azure Hongkong
- AWS/Azure US-East
- Google cloud Singapore (less recommend)

Because the free webserver provided by Heroku, which place in United States, So free MongoDB database server should have near to US due to get the low ping and fastest connection.

4.3 UML (Overview)



The UML – Component diagram show the overview of DWA to make the MVC model architecture above more explicit, this diagram is an overview stand at Customer view such as customer request to search a product or login, register, edit profile… Split DWA into 3 subsystem in order to easy understand at a customer point in the traditional way the real shop work. Each UML Overview component and interface are reference to MVC – component (Controller, Model, View, Database, Payment).

As MVC-View component is do the front-end job so in this UML Overview there is no MVC-View as an UML

component, in this diagram we can say that the response from Statical Data Management and Order Management is the front-end files provided by MVC – View.

4.3.1 Webstore UML explain:

In Webstore subsystem, it handles the requests from client such as *ProductSearch, OnlineShoping, Registration, Login* and *Edit Profile*. So, Webstore UML component is MVC – Controller.

The Authentication is the *checking authorize* of MVC – controller.

4.3.2 Statical Data Management explain:

In Statical Data Management, it does the request job to Database. So, Statical Data Management is the MVC – Model and MVC – Database.

4.3.3 Order Management explain:

In Order Management, it is the MVC – Payment (the Third party payment API - UML component), and a part of MVC – controller (Orders - UML component).