Fpt University

TRƯỜNG ĐẠI HỌC FPT

CAPSTONE PROJECT REPORT

Report 4 – Software Design Document

– Danang, March 2022 –

Table of Contents

I. Record of Changes

Date	A* M, D	In charge	Change Description

*A - Added M - Modified D - Deleted

II. Software Design Document

1. System Design

The project successfully applied the data analysis method, advanced artificial intelligence algorithms to make the recommendations online. Basically, these recommendations were based on the amount of products obtained in their previous customers ' order when shopping online. By focusing on the process of decision-making of customer's behaviours, the system used data relating to the pictures of the 10 orders that users have ordered recently to recommend items that are most relevant to them. The quality of the recommendations is directly related to the quantity and quality of the data picture obtained. So, the diversity of data about the customer behaviours, the better and more personalized the recommendations can be, arousing that user's interest and helping in decision making.

As for the client side, the recommendations online system has been developed in both web and mobile applications. The web application used the framework NextJS built on top of React which provided a short page load time and fast scan of the data taken from the user. The mobile application used the cross-platform Flutter that provides the reactive views without requiring a JavaScript bridge, contributing to the fast app startup times and less performance issues.

As for the server side, the recommendations online system have been developed with the microservice-based Java web framework Spring Boot, the Rest API and the database management system MySQL is a database management system that allows you to manage relational databases. They are popularly used due to its outstanding high performance features. VNPAY Payment Gateway was chosen as the main payment method between consumers and the system due to its security and provided outstanding payment methods such as scanning QR code, domestic cards/accounts or international credit cards.

As for the algorithm of recommendation function, the method was executed in the three main steps including scan across all product in store images and automatically generate a group of recommended products based on user last purchased(1), create the "real-time" requirements of the pipeline (2) and recommend products in the store to users (3). The system was built based on CNN model (resnet50) to extract features of products, nearest neighbor (ANNOY) to recommend products; which proved to be a very robust and effective method for recommendation systems.

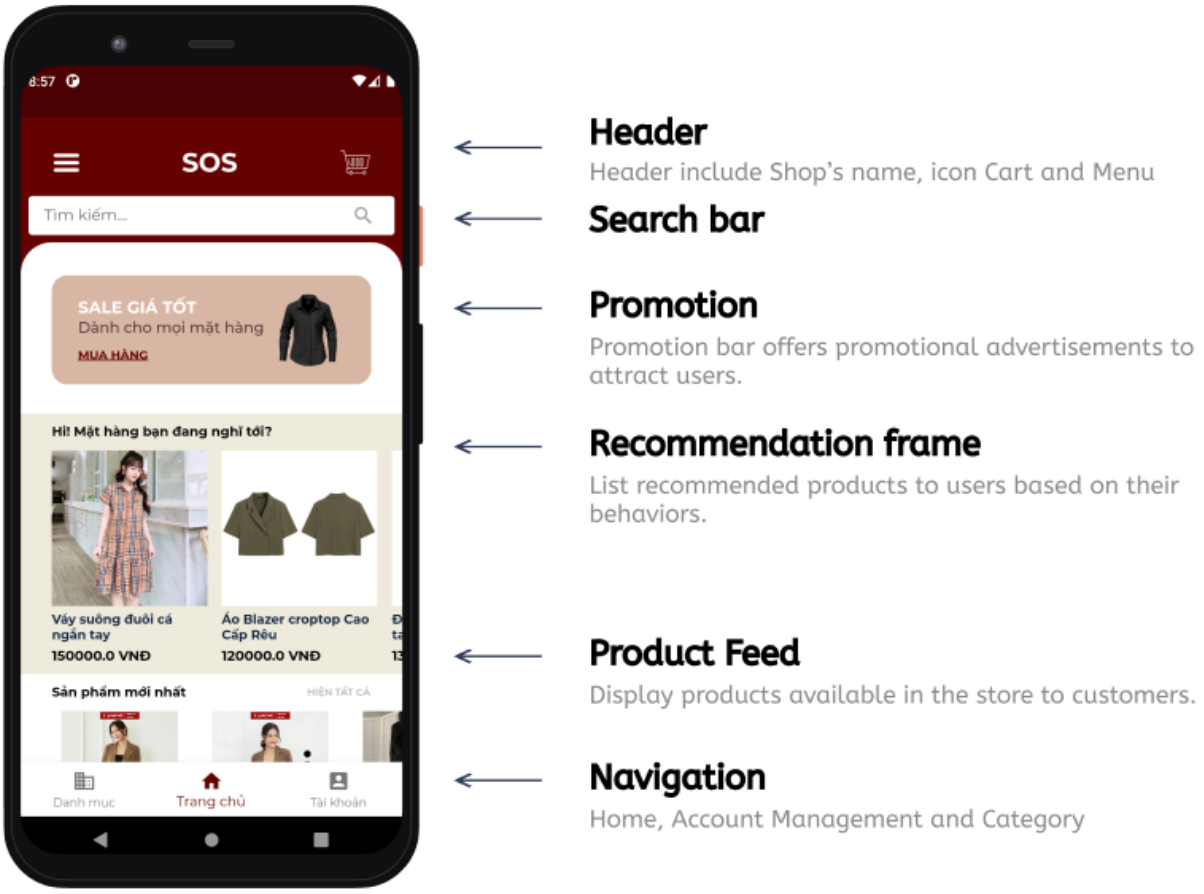


Figure 1: User interface

In our e-commerce recommendation systems, the suggestion of clothe products was appeared on the website's home page and on product pages. The process to make the online payment was completed based on the five main steps such as choose product, see the product detail, add to card, confirm the payment method and then complete the payment.

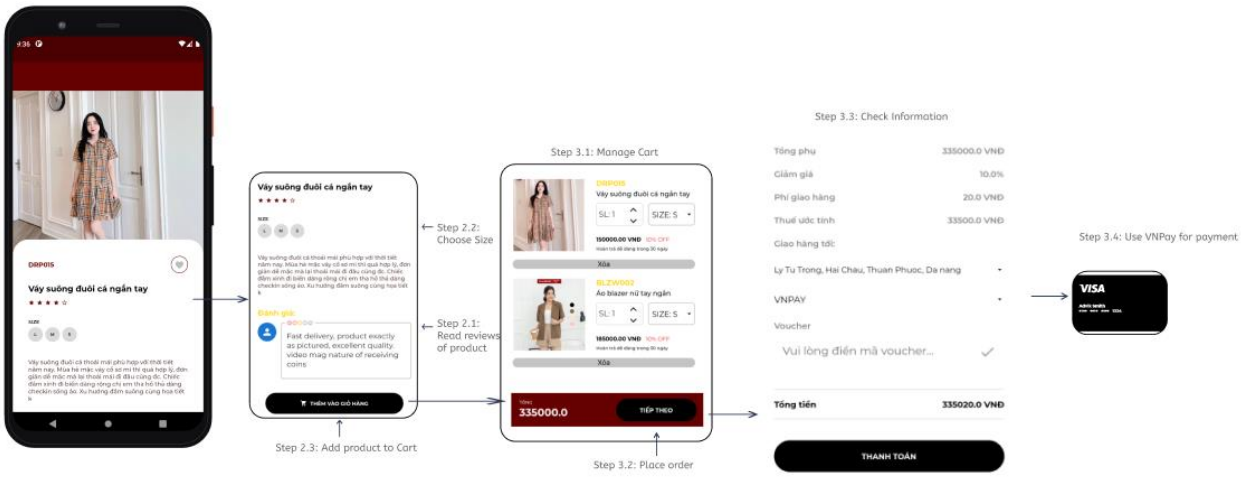


Figure 2: Order Flow for User

1.1. System Architecture Design

1.1.1. Client

1.1.1.1. Web Application Architecture

When Next.js came along, it was a perfect fit since we were naturally using JavaScript on the front and back-end of our clients' applications. The advantages of Next.js is that the Next.js application loads significantly faster than the React server-rendered application as well as supports Server-Side-Rendering. In Next.js, we divide a website into components, and we can re-use this component during coding.

All components in Next.js hold a state; however, it's not mandatory. By state, we mean the data needed to track for making the application work. Also, your application state keeps on changing from time to time, depending on the action taken by your user. It's only the state of data that determines what displays on the UI of an application. React already offers an out-of-the-box mechanism called the local state. However, the mechanism is not sufficient alone to use in real-world applications. That's where state management libraries like Redux come into play.

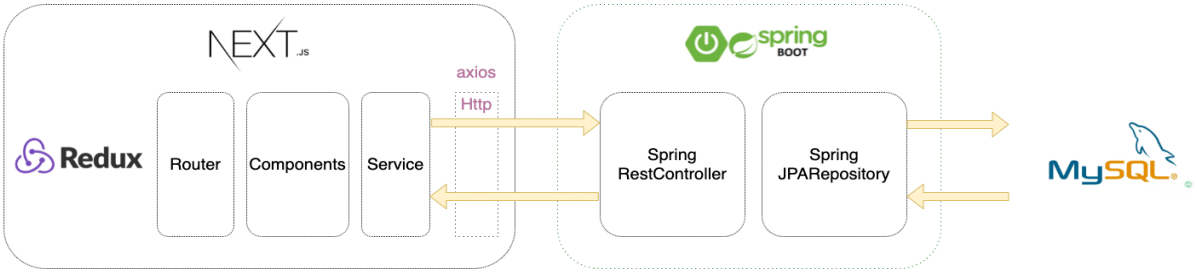


Figure 3: Web Application Architecture

1.1.1.2. Mobile Application Architecture

Flutter is a cross-platform application development platform for iOS and Android developed by Google. Flutter uses the DART language also developed by Google and flutter has also been

used to create native apps for Google. We chose it to use for this project to develop and build for the front-end.

Its advantages meet everything for our project, especially in terms of effects, very high application performance. Communicate almost directly with the system. Static language but with modern syntax (similar to JS, Python, Java), flexible compiler using AOT (for end product) and JIT (for development with hot reload). Can run mobile emulators right on the web, convenient for development. Supported performance metrics give developers great control over application performance.

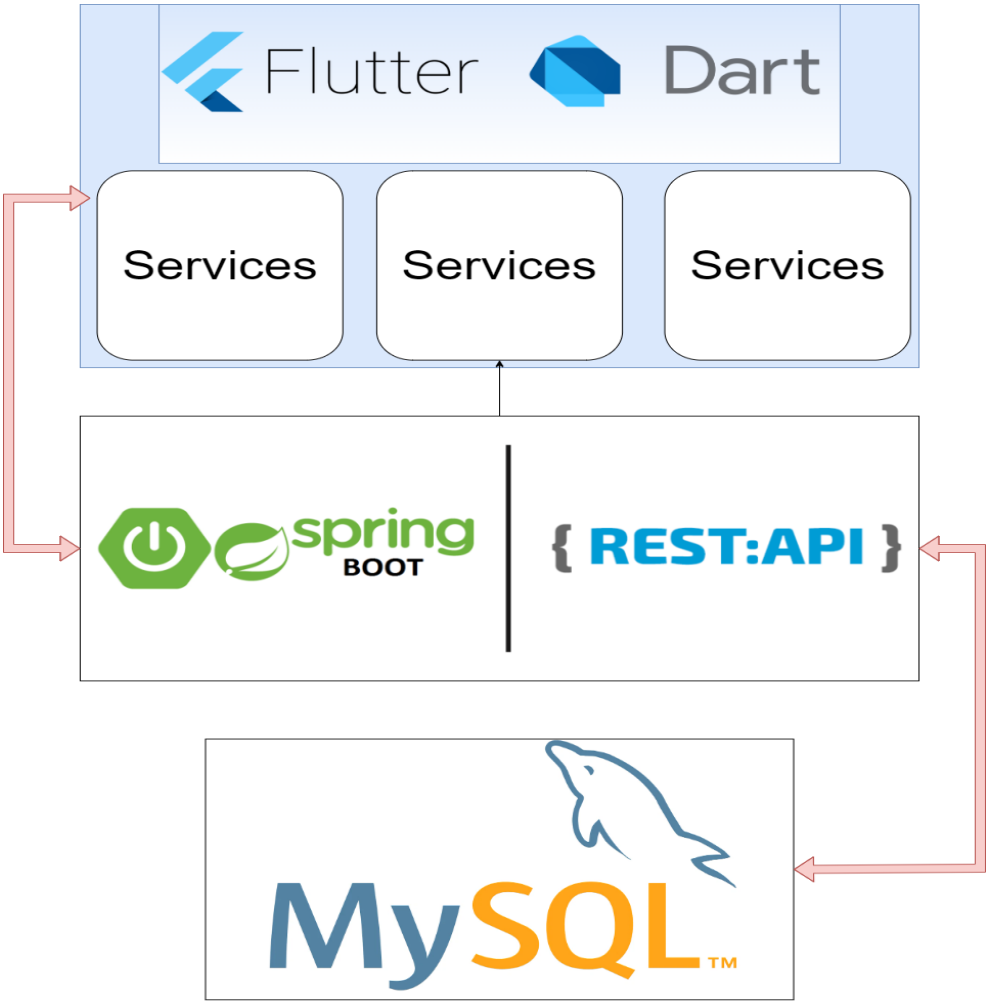


Figure 4: Mobile Application Architecture

1.1.2. Server

1.1.2.1. Spring Boot Application Architecture

Spring Boot is an open source, microservice-based Java web framework. The Spring Boot framework creates a fully production-ready environment that is completely configurable using its prebuilt code within its codebase.

Spring Boot follows a layered architecture:

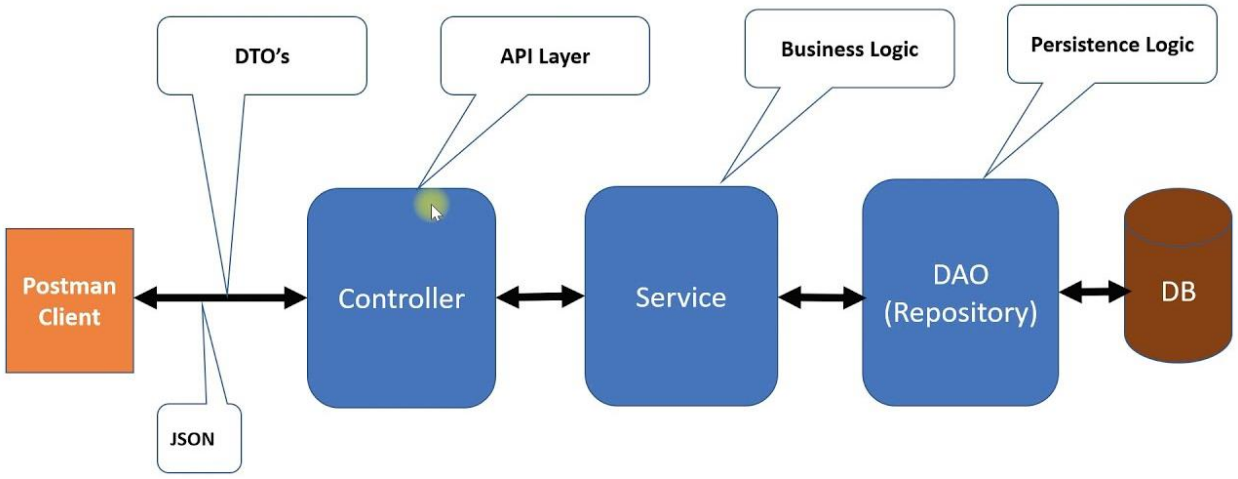


Figure 5: Spring Boot Application Architecture

We are going to use three layer or three-tier architecture to develop out spring boot application.We have controller layer, service layer and DAO or repository layer

- Controller layer is used to develop arrest apis and we can call controller layer as an api layer because we basically create spring mvc controller and we keep all rest apis in controller class.
- In the service layer, we keep all our business logic.
- In the DAO layer, we keep our database related logic or persistence logic, and the DAO layer is responsible for communicating with the database.

We use a postman client to call rest apis or to test the rest apis. We are going to use DTOs to transfer data between client and server, and we use json json as a media type. We use json as a message exchange format between client and server

1.1.2.2. Rest API Architecture

REST has quickly become the de-facto standard for building web services on the web because they're easy to build and easy to consume.

The below diagram shows the typical REST architecture:

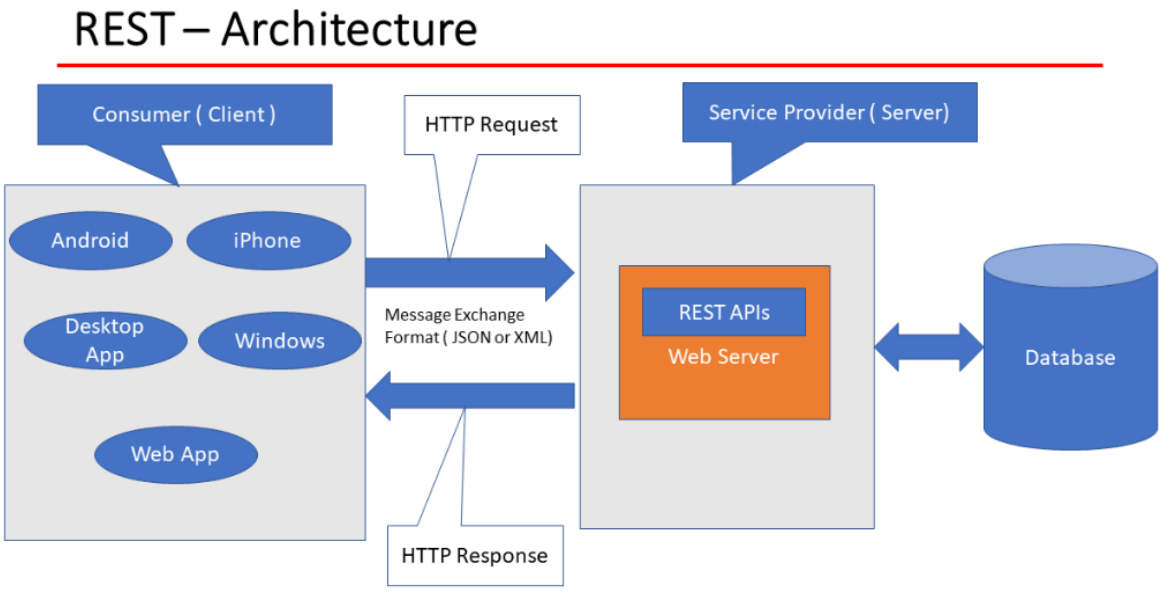


Figure 6: Rest API Architecture

- **Request and Response:** Request is the input to a web service, and the response is the output from a web service.
- **Message Exchange Format:** It is the format of the request and response. There are two popular message exchange formats: XML and JSON.
- **Service Provider or Server:** The service provider is one that hosts the web service.
- **Service Consumer or Client:** A service consumer is one who is using the web service.

Client application has the responsibility to prepare and send http request messages.

The business component(developed by a service provider) has the responsibility to prepare and send the HTTP response message.

1.1.2.3. System Architecture Explanation

1.1.2.3.1. Spring Boot

Figure 7: Spring Boot

Spring Boot is an open source, microservice-based Java web framework. The Spring Boot framework creates a fully production-ready environment that is completely configurable using its prebuilt code within its codebase. Spring Boot makes developing web application and microservices with Spring Framework faster and easier through three core capabilities:

+ Auto configuration: means that applications are initialized with pre-set dependencies that you don't have to configure manually.
+ An opinionated approach to configuration: uses an opinionated approach to adding and configuring starter dependencies, based on the needs of your project.
+ The ability to create standalone applications: you can launch your application on any platform. It lets you create standalone applications that run on their own, without relying on an external web server, by embedding a web server such as Tomcat into your app during the initialization process.

1.1.2.3.2. Spring JPA

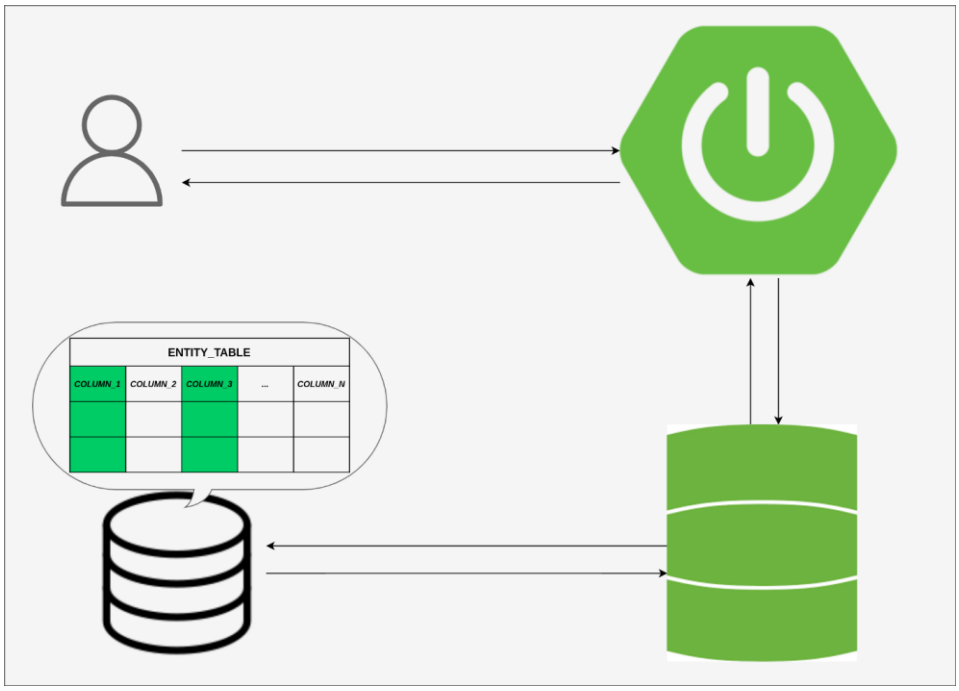


Figure 8: Spring JPA

Spring Boot JPA is a Java specification for managing relational data in Java applications. It allows us to access and persist data between Java objects/ classes and the relational database. JPA follows Object-Relation Mapping. It is a set of interfaces. It also provides a runtime EntityManager API for processing queries and transactions on the objects against the database. It uses a platform-independent object-oriented query language JPQL (Java Persistent Query Language).

JPA is widely used beaucase:

+ It is simpler, cleaner, and less labor-intensive than JDBC, SQL, and hand-written mapping.
+ It allows mapping in XML or using Java annotations.
+ When we need to perform queries using JPQL, it allows us to express the queries in terms of Java entities rather than the (native) SQL table and columns.

1.1.2.3.3. Spring Security

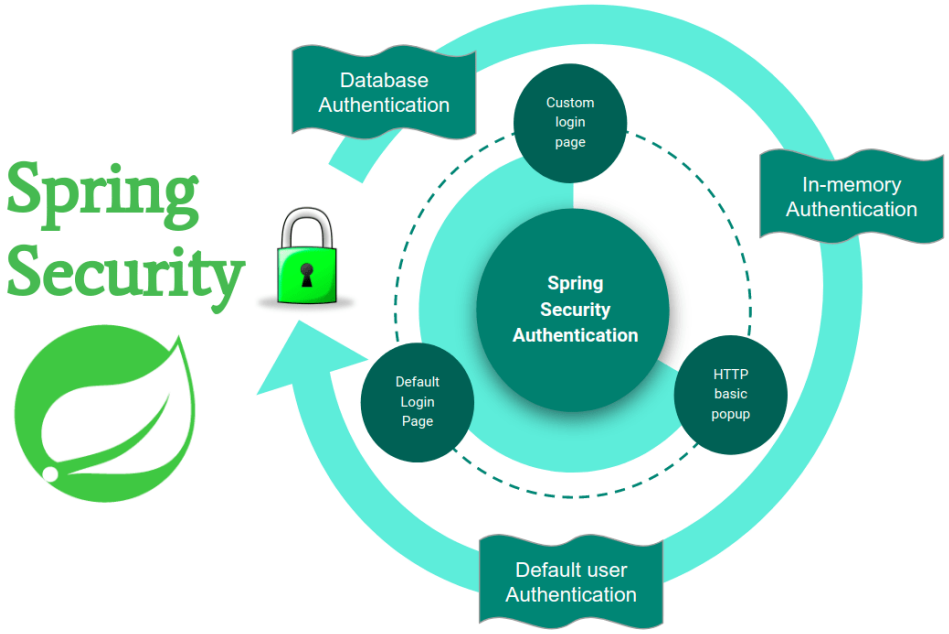


Figure 9: Spring Security

Spring Security is a framework that provides authentication, authorization, and protection against common attacks. With first class support for securing both imperative and reactive applications, it is the de-facto standard for securing Spring-based applications.

The benefits of Spring Security are not limited to helping us with Authentication and Authorization. It can also help us to apply best practices in saving users, building a sign-up feature.

1.1.2.3.4. Rest API

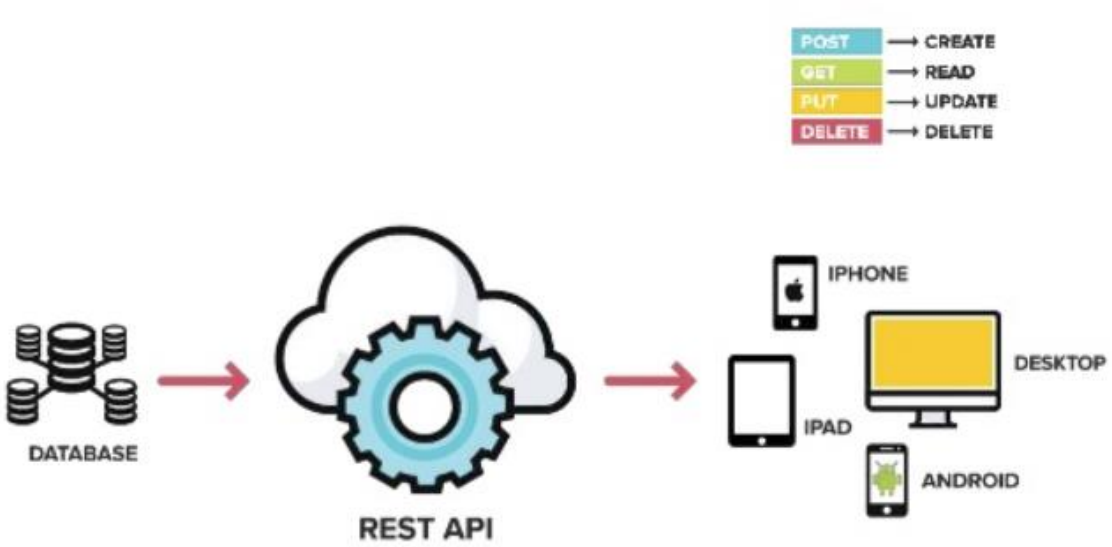


Figure 10: Rest API

An API (application programming interface) is a set of rules that define how applications or devices can connect to and communicate with each other. REST APIs provide a flexible, lightweight way to integrate applications, and have emerged as the most common method for connecting components in microservices architectures.

1.1.2.3.5. MySQL

Figure 11: MySQL

MySQL is a database management system that allows you to manage relational databases. It is open source software backed by Oracle. A is popularly used for its outstanding features:

+ High Performance: MySQL can meet performance of a high-volume web site that services a billion queries a day.
+ Web and Data Warehouse Strengths: MySQL is the de-facto standard for high-traffic web sites because of its high-performance query engine, tremendously fast data insert capability, and strong support for specialized web functions like fast full text searches.

+ Strong Data Protection: MySQL offers exceptional security features that ensure absolute data protection.
+ Comprehensive Application Development: MySQL provides comprehensive support for every application development need.
+ Management Ease: MySQL offers exceptional quick-start capability and a complete suite of graphical management and migration tools

1.1.2.3.6. VNPay (External service)

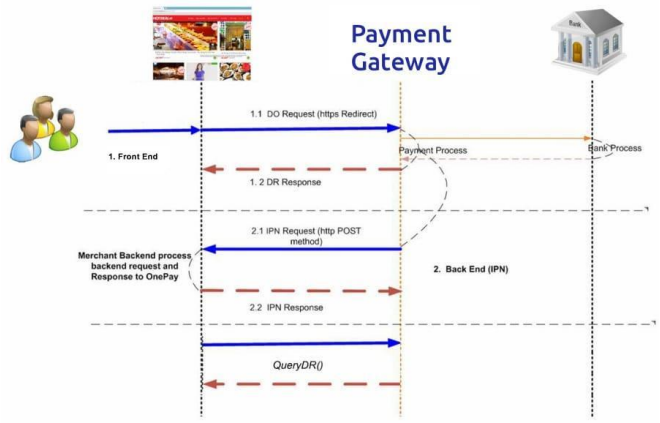


Figure 12: VNPay (External service)

VNPAY Payment Gateway is an intermediate system to transmit, exchange and process payment transactions between consumers owning card, bank account or e-wallet and enterprises providing goods, services on the Internet. VNPAY Payment Gateway provides customers with 3 outstanding payment methods: scanning QR code via Mobile Banking app, payment by domestic cards/accounts, payment by international credit cards. Using VNPAY as our system's online payment method makes it easier for customers to pay for their orders.

1.1.3. Recommendation System

1.1.3.1. Overview of recommendation system

Figure 13: Flow of recommendation system

Fashion is an area where usually users will make purchasing decisions based on the beauty of that product, so we build a recommendation system based on each user's shopping preferences through various categories characteristics of the product they have purchased in the store so we used Collaborative filtering System (Item-Item filtering) for recommend and that system must be reply the following lists:

- The model should be able to scan across all product in store images and automatically generate a group of recommended products that are customized to what the user has viewed/bought.
- Speed is of the essence in an eCommerce firm. The group of recommended products should be generated to meet the “real-time” requirements of the pipeline.
- The model should be able to adapt to new products and new users.

So we built a system based on CNN model (resnet50) to extract features of products so we can embedding image products and approximate nearest neighbor (ANNOY) to recommend products for users based on top 10 products purchased by the user. This flow will be show as diagram below:

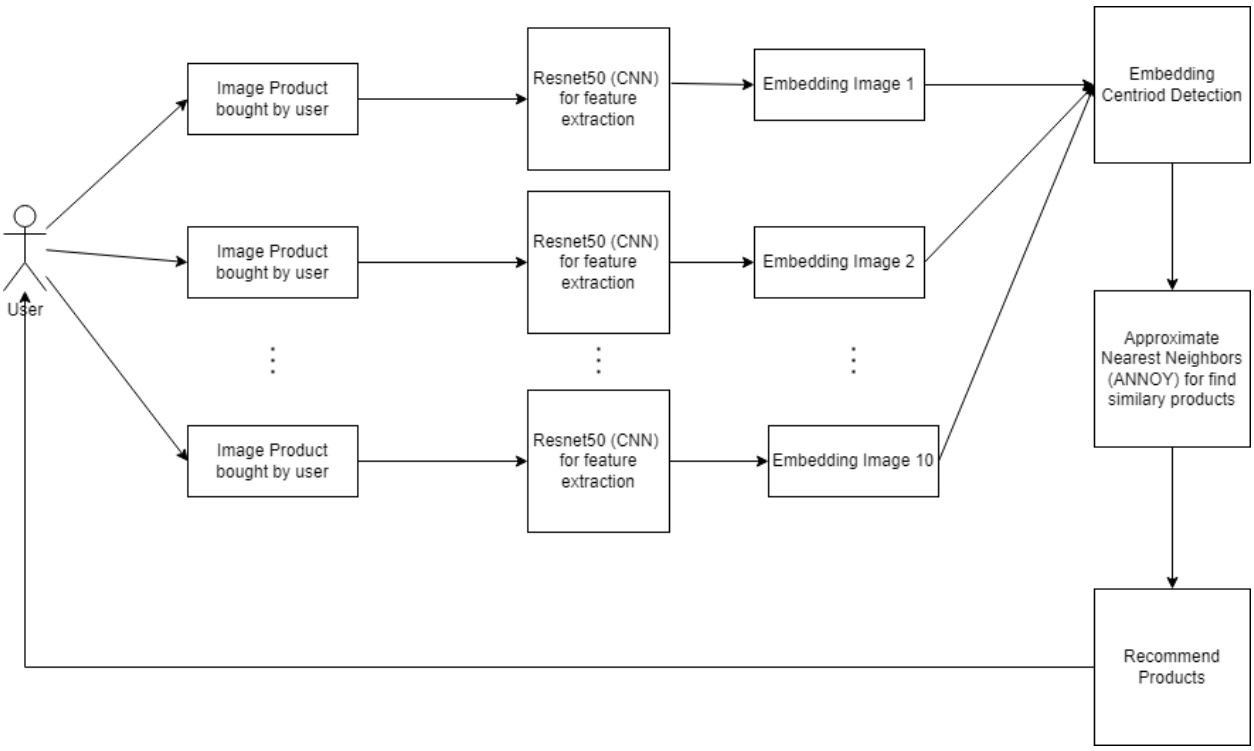


Figure 14: Flow chart of recommendation system

1.1.3.2. The setting up steps of the recommendation System

The recommendation systems provide value to customers by understanding an individual user's behaviour and then recommending to them items they might find useful. For customers, recommendation systems are "smart" filters that help them discover what they like. For businesses, recommendation systems drive engagement with content and increase revenue. The algorithms for excuting the system includes three main step. First, the system collect the data on users order. Second, the system used collaborative filtering and item-based filtering and the embeddings system to find and recommend items that are similar to those a customer has already bought, based on a domain-specific notion of item picture. Finally, the system used the Convolutional Neural Network Architecture (CNN) for evaluating and testing the system.

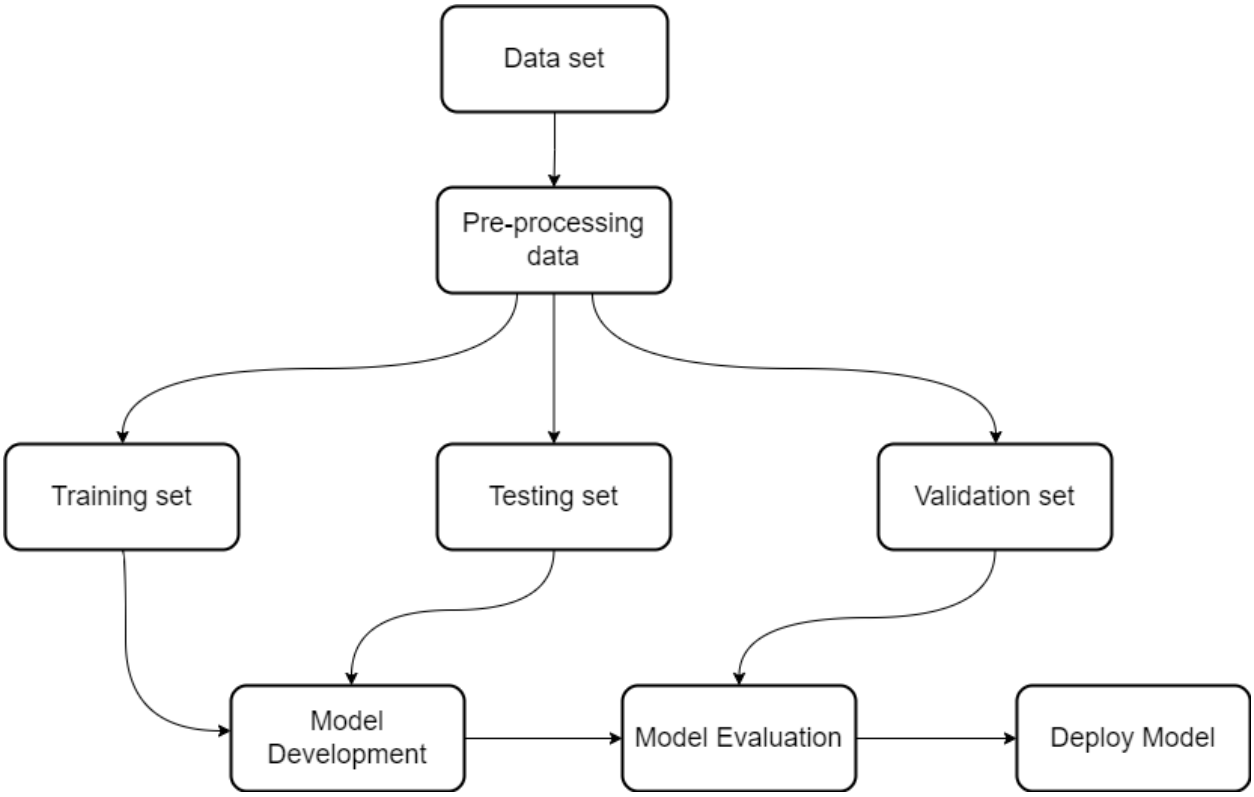


Figure 15: Flow of building model

1. Collect data on productsThe Data:

- We will be using a subset of DeepFashion data open-sourced by Liu Z. et al., The Chinese University of Hong Kong. Our data consists of 280K fashion images across 46 categories.

2. Using the filtering systems

- **Collaborative filtering System**: Collaborative does not need the features of the items to be given. Every item is described by a feature vector or embedding.
- It creates embedding for both items on its own. It embeds items in the same embedding space.
- **Item-Item filtering**: Here, if user A likes an item x, then, the items y and z which are similar to x in property, then y and z are recommended to the user. As a statement, it can be said, "Because you liked this, you may also like those".

Let us take a look at the input/output of the model to gain intuition on the problem and proposed solution:

Input:

- I simulate outfits viewed/bought by users by selecting 10 outfits of similar styles.
- I will only pass images into the model; the model does not ingest additional attributes/descriptions/product details; this is because we want the model to automatically

learn and detect the style of fashion images passed into it without further human/machine labeling.

Output:

- The goal is for collections generated by the model to contain some aspects of the input but with serendipity to pleasantly surprise users.

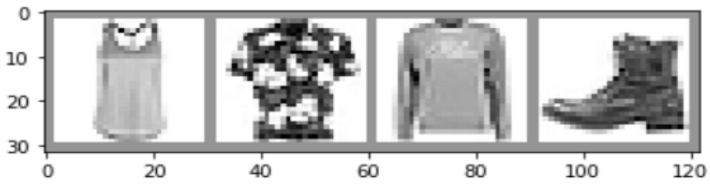
Figure 16: Recommend product for user with flower dress

Figure 17: Recommend products for user with short categories

3. Using the Embeddings system to represent items in a matrix

To explain more about the concept of embedding, we will use of a open-source data set known as FashionMNIST:

```
# show images
matplotlib_imshow(img_grid, one_channel=True)
```




```
img_grid
tensor([[[[ 0.,  0.,  0.,  ...,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  ...,  0.,  0.,  0.],
          [ 0.,  0., -1.,  ..., -1.,  0.,  0.],
          ...,
          [ 0.,  0., -1.,  ..., -1.,  0.,  0.],
          [ 0.,  0.,  0.,  ...,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  ...,  0.,  0.,  0.]],

         [[ 0.,  0.,  0.,  ...,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  ...,  0.,  0.,  0.],
          [ 0.,  0., -1.,  ..., -1.,  0.,  0.],
          ...,
          [ 0.,  0., -1.,  ..., -1.,  0.,  0.],
          [ 0.,  0.,  0.,  ...,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  ...,  0.,  0.,  0.]],

         [[ 0.,  0.,  0.,  ...,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  ...,  0.,  0.,  0.],
          [ 0.,  0., -1.,  ..., -1.,  0.,  0.],
          ...,
          [ 0.,  0., -1.,  ..., -1.,  0.,  0.],
          [ 0.,  0.,  0.,  ...,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  ...,  0.,  0.,  0.]]])
```

Figure 18: FashionMNIST image

Traditionally, we represent an image as a large array of integers (3D array for RGB images and 1D array for grayscale images). These arrays are unwieldy and grow exponentially — we need to keep track of millions of numbers to analyze hundreds of high-resolution images. It will be impossible to scale any modeling using arrays of integers, so the modern approach of embeddings is created.

Traditionally, we made use of techniques like one-hot encoding to represent items in a matrix. For example, given three books, we will have a 3x3 matrix, where each item is represented by an array of discrete numbers that grows with every new item added (4x4 with 4 items, 5x5 with 5 items…). Furthermore, there is no sense of similarity nor distance as the items are not connected with any form of relationship.

```
# One Hot Encoding Categoricals

books = ["Harry Potter and The Philosopher's Stone",
         "Harry Potter and The Chamber of Secrets",
         "The Lean Startup"]

books_encoded = [[1, 0, 0],
                 [0, 1, 0],
                 [0, 0, 1]]

Similarity (dot product) between First and Second = 0
Similarity (dot product) between Second and Third = 0
Similarity (dot product) between First and Third = 0
```

Figure 19: Example of one-hot encoding

Once we apply a transformation to convert the objects into embeddings, we now limit the number of elements in the array representing each item (the limit is 2 in this example) on a

continuous scale, and the values have a relationship-based meaning. Objects close to each other based on similarity (dot product) are highly related. Example of Embedding:

```
# Idealized Representation of Embedding

books = ["Harry Potter and The Philosopher's Stone",
         "Harry Potter and The Chamber of Secrets",
         "The Lean Startup"]

books_encoded_ideal = [[0.53,  0.85],
                       [0.60,  0.80],
                       [-0.78, -0.62]]

Similarity (dot product) between First and Second = 0.99
Similarity (dot product) between Second and Third = -0.94
Similarity (dot product) between First and Third = -0.97
```

Figure 20: Example of embedding

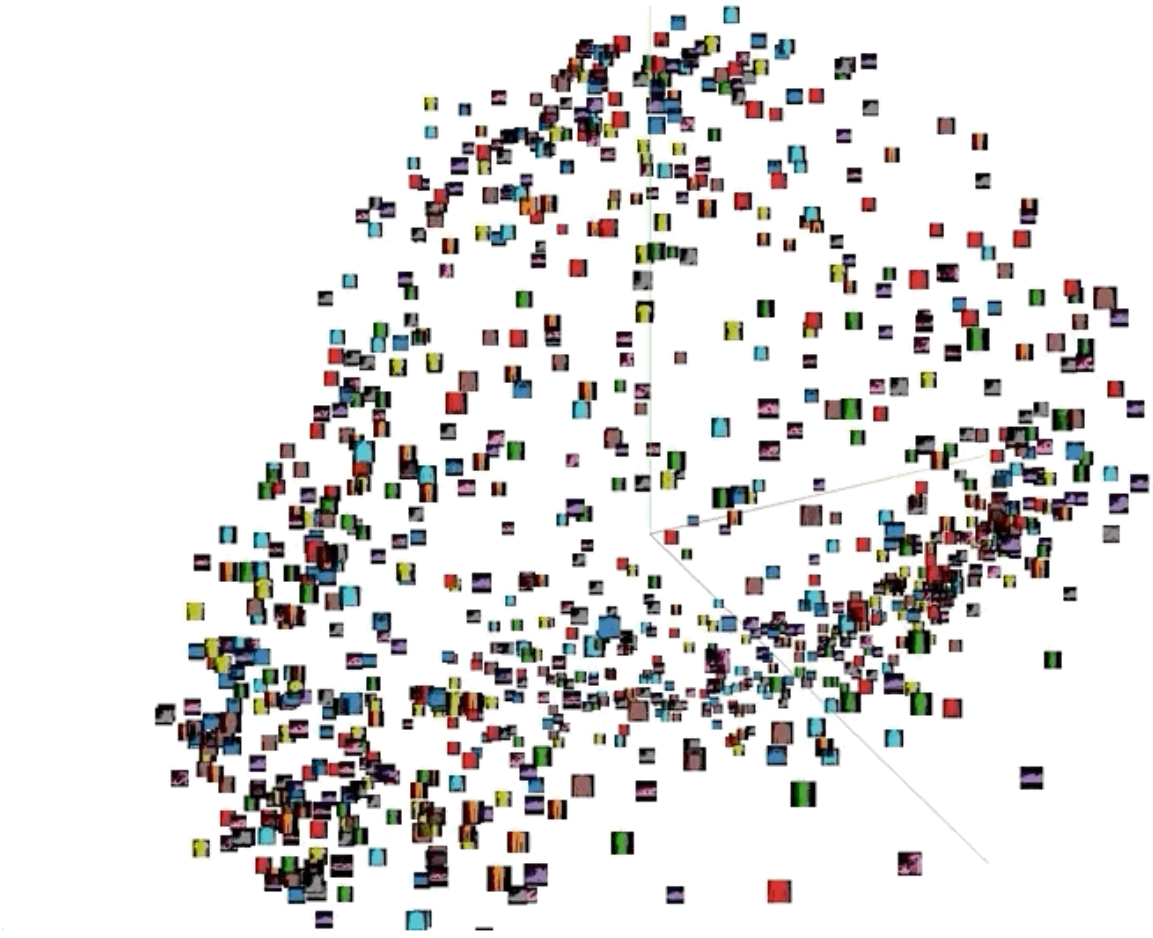


Figure 21: Embedding Projection

1.1.3.3. Using the neural networks

Convolutional Neural Network Architecture (CNN) definition:

Convolution is not a new technique. In essence, we are applying a kernel to every pixel in the image to achieve a goal, usually to blur, sharpen or detect edges/objects. For each pixel, we will do an elementwise product with the kernel, followed by summing the result to get a single number.

CNN is a fascinating model as it combines the power of convolution and neural networks. There are many different architectures but generally consist of a combination of convolution, subsampling, activation and full connectedness.

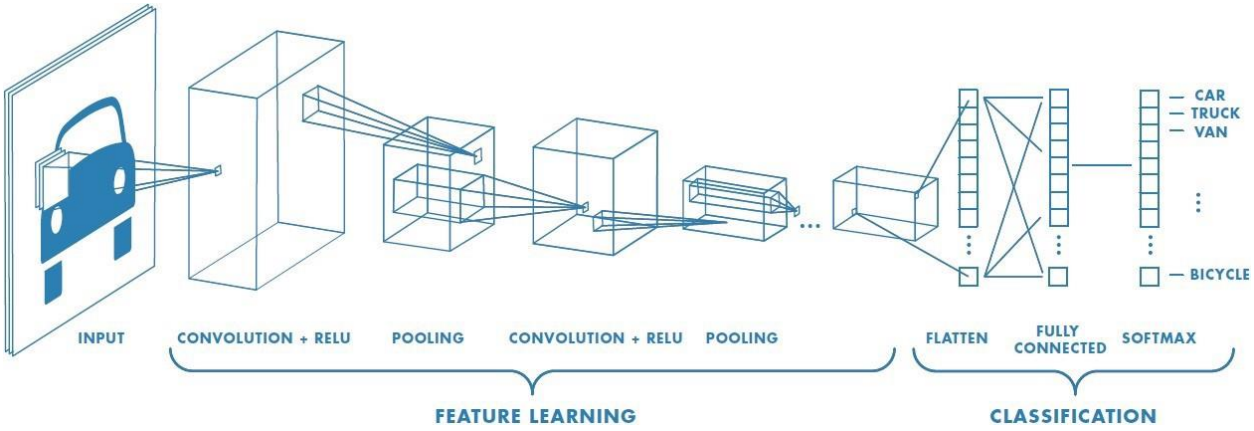


Figure 22: Convolutional neural networks

1.1.3.4. Transfer Learning Architecture

Concretely, we take a large pre-trained CNN like resnet50 , freeze most of the layers, and train on the last few layers. The intuition is that for most of the DCNN, the knowledge gained by the filters is transferable components such as detection of edges, patterns, gradients, colors, etc. Thus, we fine-tune the last few layers to our problem, and we can have a working solution.

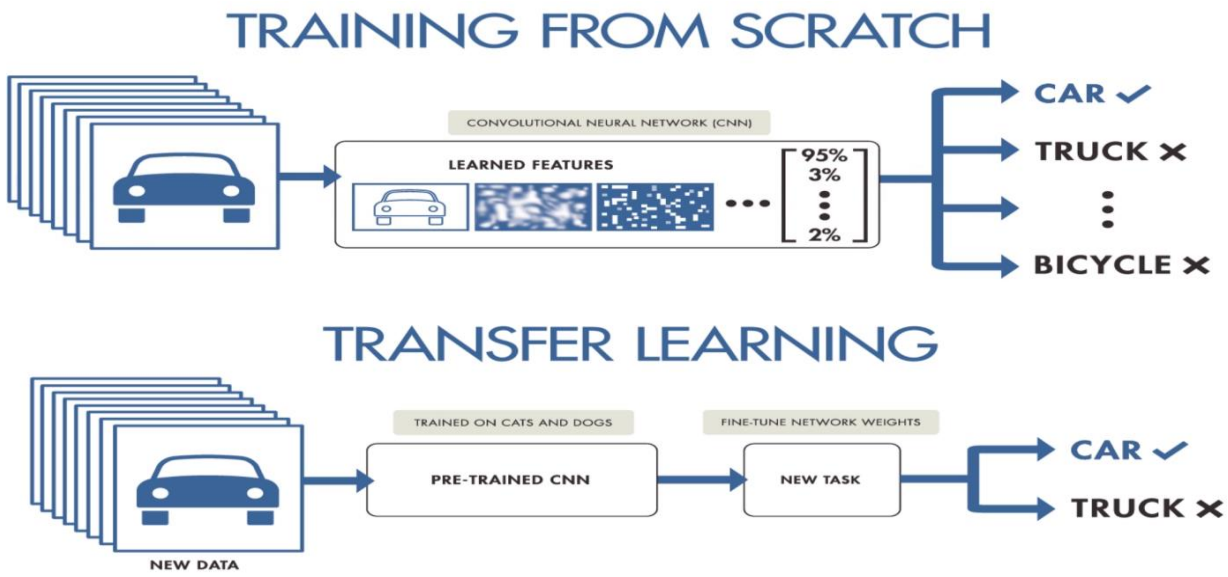


Figure 23: Transfer learning

1.1.3.5. Resnet Architecture

Residual Network (ResNet) is one of the famous deep learning models that was introduced by Shaoqing Ren, Kaiming He, Jian Sun, and Xiangyu Zhang in their paper. The paper was named "Deep Residual Learning for Image Recognition" [1] in 2015. **The ResNet model is one of the popular and most successful deep learning models so far.**

The problem of training very deep networks has been relieved with the introduction of these Residual blocks and the ResNet model is made up of these blocks. Resnet blocks is showed below:

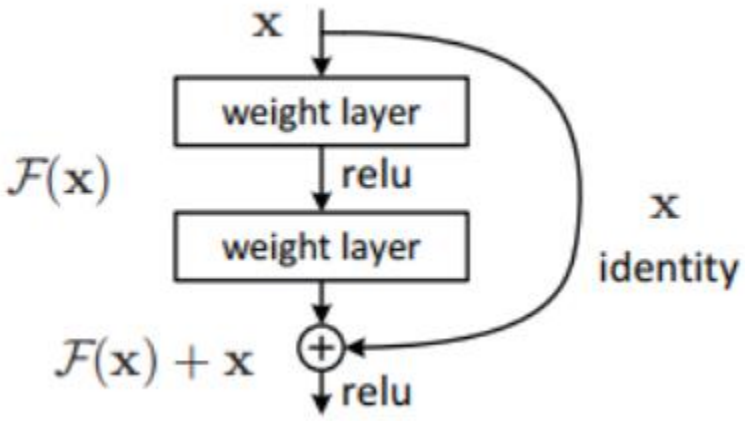


Figure 24: Skip connection

In the above figure, the very first thing we can notice is that there is a direct connection that skips some layers of the model. **This connection is called 'skip connection' and is the heart of**

residual blocks. The output is not the same due to this skip connection. Without the skip connection, input ‘X gets multiplied by the weights of the layer followed by adding a bias term.

This skip connections technique in ResNet solves the problem of vanishing gradient in deep CNNs by allowing alternate shortcut paths for the gradient to flow through. Also, the skip connection helps if any layer hurts the performance of architecture, then it will be skipped by regularization. that's why we chose this model.

Training Accuracy with DeepFashion dataset:

We train both Resnet18 model and Resnet50 model to compare which model achieves better results and we obtained that Resnet50 model achieve higer accuracy (93%) compared to Resnet18 (91%)

Confusion matrix after training:

This is confusion matrix based on our training data

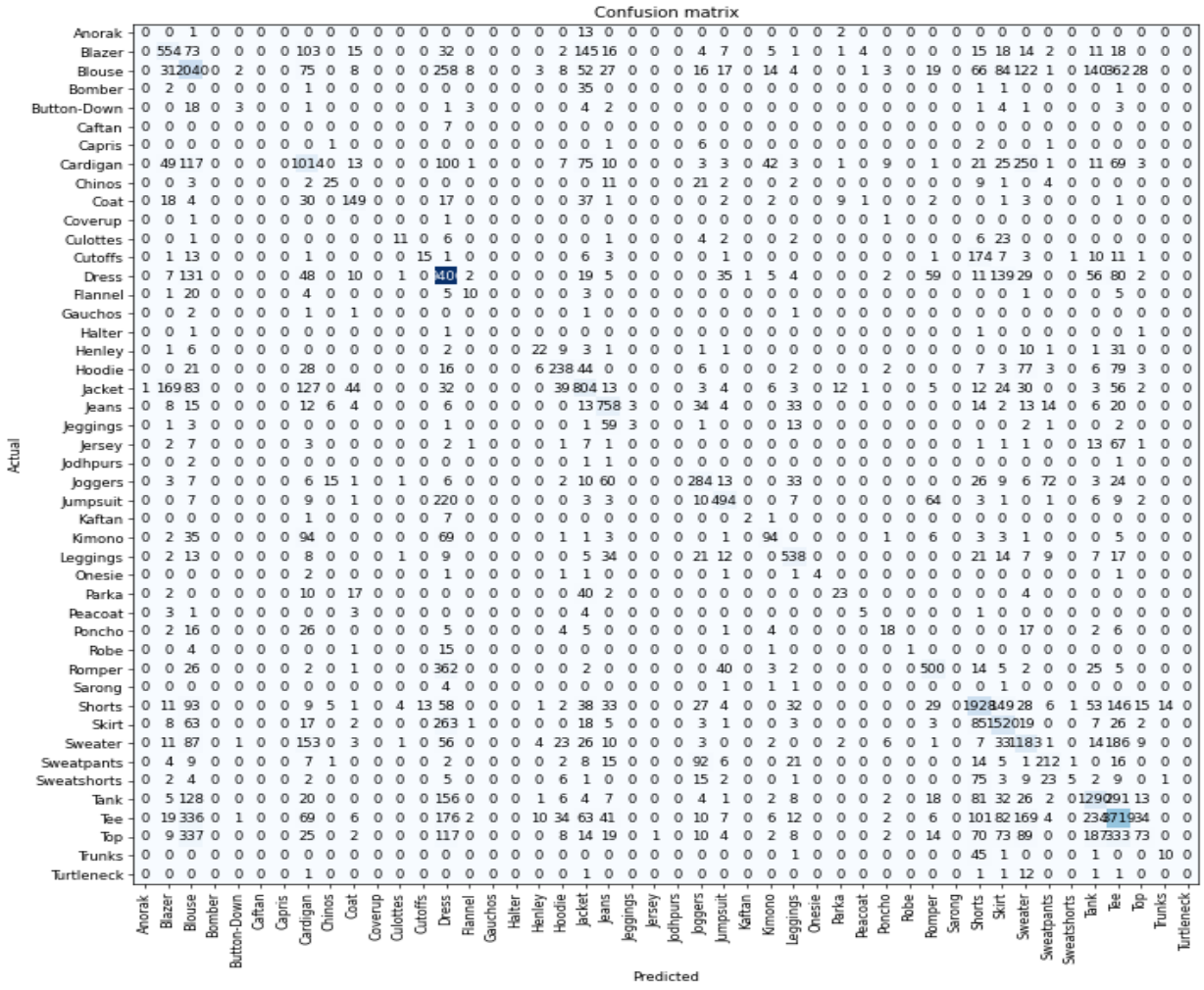


Figure 25: Confusion matrix

Confusion Matrix allows us to measure Recall, Precision. Precision represents the reliability of the model, Recall represents the sensitivity of the model.

1.1.3.6. Nearest Neighbors with Annoy & Embeddings Centroid Detection

We run Annoy through each row in the dataframe and find the nearest neighbors.

Finally, we insert the embeddings of all 10 (or more) user-selected items into a list and average the embeddings values across each of the dimensions; This creates a phantom object that represents the aggregated value across all the selected items. We can then find the nearest neighbor to this phantom item.

If we only have a small corpus of images to search, simple distance metrics like cosine similarity will work. In real-world deployments, such as e-commerce, we usually have millions of images to compare with each other, and it will be impractical for the API to execute pairwise comparisons across every single image. Annoy (Approximate Nearest Neighbors Oh Yeah) is by Erik Bernhardsson from Spotify, with an easy to use API t/hat can be integrated into our PyTorch workflow.

1.1.4. System Architecture

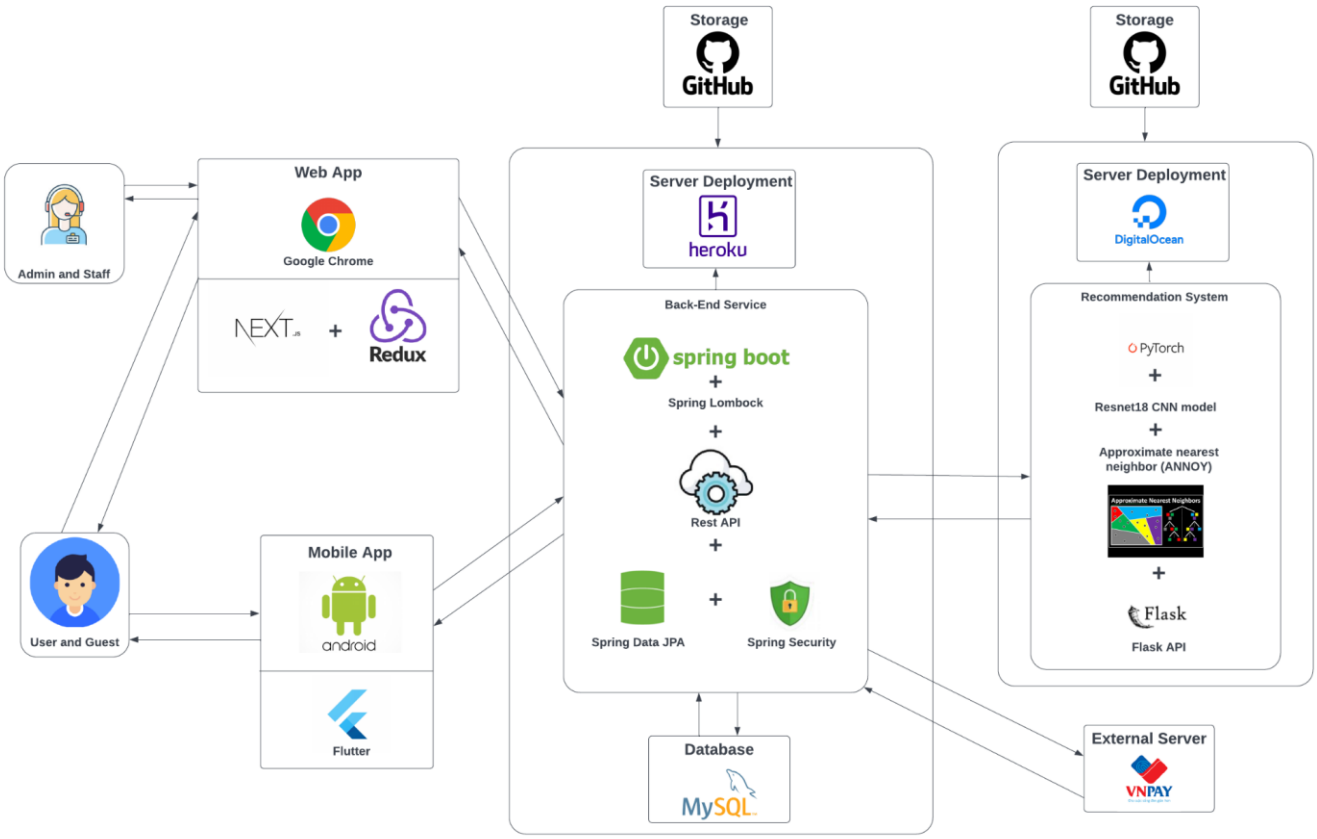


Figure 26: System Architecture

1.2. Package Diagram

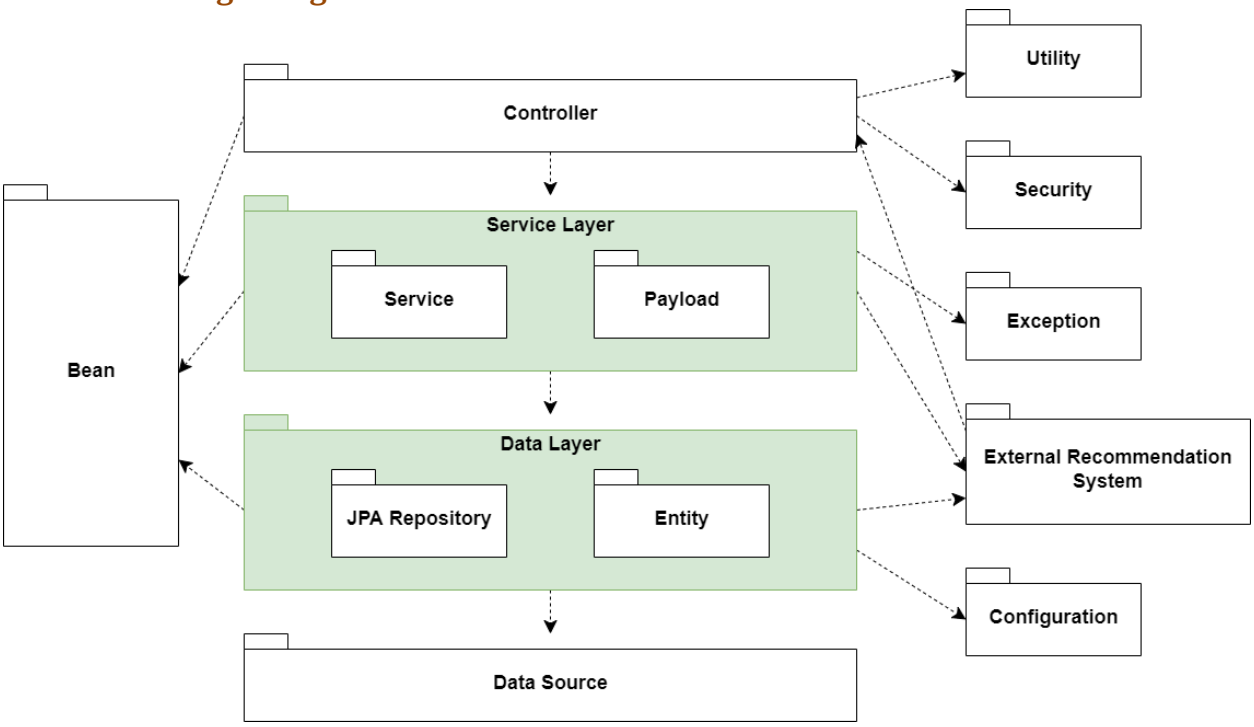


Figure 27: Package Diagram

Package Descriptions

No	Package	Description
01	Controller	Contains controller classes which are responsible for processing incoming REST API requests, preparing a model, and returning the view to be rendered as a response.
02	Service	Contains services classes which are used to write business logic in a different layer, separated from controller
03	Payload	Contains payload classes which are the essential information in a data block that is sent to or received from the server when making API requests.
04	JPA Repository	Contains JPA repository interface that encapsulate the logic required to access data sources
05	Entity	Contains entity classes which are the persistence objects stores as a record in the database

06	Data Source	A factory for connections to the MySql database
07	Utility	Contains utility classes which perform common, often reused functions
08	Security	Implements authentication, authorization, and protection against common attacks
09	Exception	Provides a mechanism to treat exceptions that are thrown during execution of handlers
10	External Recommendation System	Model contains Resnet50 (CNN model) and Nearest Neighbors to predict products based last purchased user's products
11	Configuration	Contains configuration classes which consist principally of @Bean - annotated methods that define instantiation, configuration, and initialization logic for objects that are managed by the Spring IoC container
12	Bean	Exploits the Inversion of Control feature by which an object defines its dependencies without creating them

2. Database Design

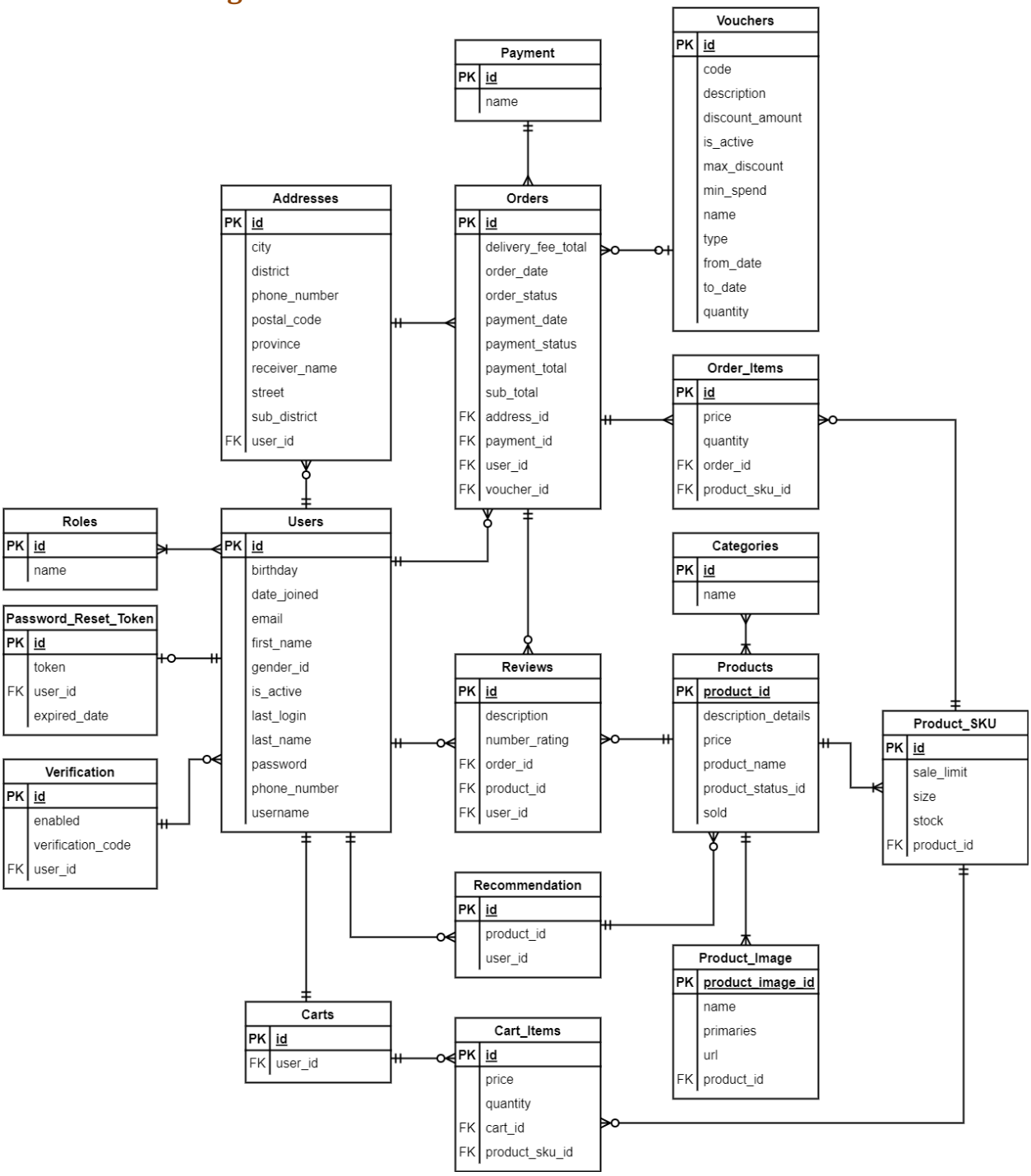


Figure 28: Database Design

Table Descriptions

No	Table	Description
01	Users	- **Primary keys:** id

		- **Foreign keys:** None
02	Roles	- **Primary keys:** id - **Foreign keys:** None
03	Addresses	- **Primary keys:** id - **Foreign keys:** user_id
04	Verification	- **Primary keys:** id - **Foreign keys:** user_id
05	Password_Reset_Token	- **Primary keys:** id - **Foreign keys:** user_id
06	Products	- **Primary keys:** product_id - **Foreign keys:** None
07	Categories	- **Primary keys:** id - **Foreign keys:** None
08	Product_SKU	- **Primary keys:** id - **Foreign keys:** product_id
09	Product_Image	- **Primary keys:** product_image_id - **Foreign keys:** product_id
10	Recommendation	- **Primary keys:** id - **Foreign keys:** product_id, user_id
11	Carts	- **Primary keys:** id - **Foreign keys:** user_id
12	Cart_Items	- **Primary keys:** id - **Foreign keys:** cart_id, product_sku_id
13	Orders	- **Primary keys:** id - **Foreign keys:** address_id, payment_id, user_id, voucher_id
14	Order_Items	- **Primary keys:** id - **Foreign keys:** order_id, product_sku_id
15	Payments	- **Primary keys:** id - **Foreign keys:** None
16	Reviews	- **Primary keys:** id - **Foreign keys:** order_id, product_id, user_id
17	Vouchers	- **Primary keys:** id - **Foreign keys:** None

3. Detailed Design

3.1. Class Diagrams

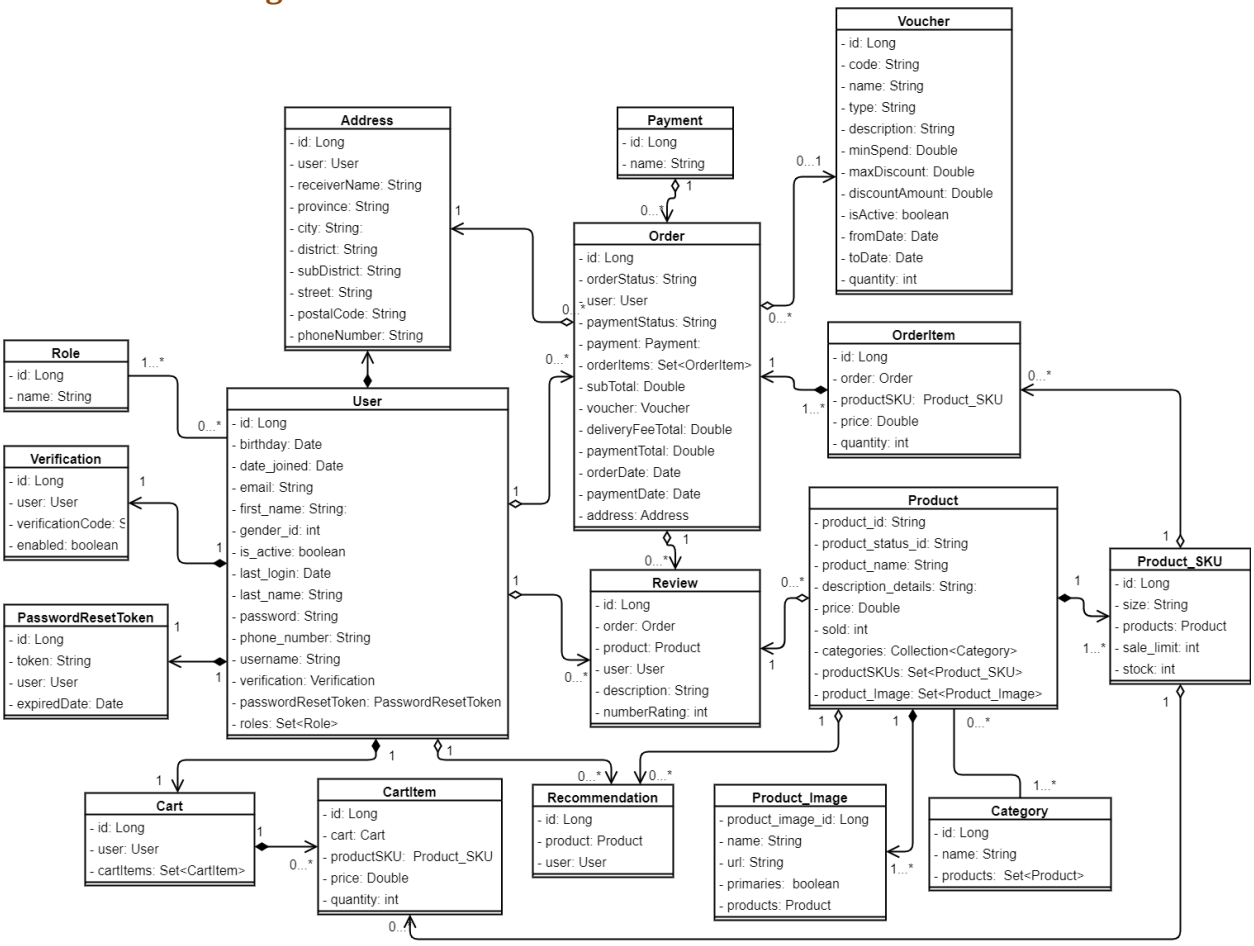


Figure 29: Class Diagrams

3.2. Class Specifications

3.2.1. User Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	birthday	Date	Day of birth of user
03	date_joined	Date	The day user register to the system
04	email	String	Email of user, using for login and forgot password
05	fisrt_name	String	First name of user
06	gender_id	int	Gender of user
07	is_active	boolean	Login status of user

08	last_login	Date	The latest time user has logged into the system
09	last_name	String	Last name of user
10	password	String (Encode)	Password of user, using for login
11	phone_number	String	Phone number of user
12	username	String	Username of user, using for login
13	verification	Verification	Contain verification code and status when user first register to the system, using for email verification and activate account
14	passwordResetToken	PasswordResetToken	Contain token for user when password is forgot, using for user verification and reset password
15	roles	Set<Role>	List of roles of user

3.2.2. Role Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	name	String	Name of role

3.2.3. Address Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	user	User	User who create address
03	receiverName	String	Name of receiver of the order
04	provine	String	Province of address
05	city	String	City of address
06	district	String	District of address
07	subDistrict	String	Sub District of address
08	street	String	Street of address
09	postalCode	String	Postal code of the address
10	phoneNumber	String	Phone number of the receiver

3.2.4. Verification Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	user	User	User of verification
03	verificationCode	String	Unique code attached to a link that send to user's email to verify and activate user account
04	enabled	boolean	Activation status of user

3.2.5. PasswordResetToken

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	user	User	User of password reset token
03	token	String	Unique token that send to user's email and user will enter this token at reset password screen to verify
04	expiredDate	Date	The date the token is expired

3.2.6. Product Class

No	Attributes	Type	Description
01	product_id	String	Unique identifier
02	product_status_id	String	Status of product (instock/outstock)
03	product_name	String	Name of product
04	description_details	String	Description details of product
05	price	Double	Unit price of product
06	sold	int	Number of sold products
07	categories	Collection<Category>	List of categories of product
08	productSKUs	Set<Product_SKU>	List of product SKU of product
09	product_Image	Set<Product_Image>	List of images of product

3.2.7. Category Class

No	Attributes	Type	Description

01	id	Long	Unique identifier, auto increment
02	name	String	Name of category
03	products	Set<Product>	list of product have this category

3.2.8. Product_SKU Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	size	String	Size of product SKU
03	product	Product	Product that has this product SKU
04	sale_limit	int	Max number of item allowed in 1 order
05	stock	int	Number of item available in stock

3.2.9. Product_Image Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	name	String	Name of image
03	url	String	url of image
04	primaries	boolean	Use to identify main image of the product
05	product	Product	Product that has this image

3.2.10. Recommendation

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	product	Product	Product that recommend to user
03	user	User	User who get recommended product

3.2.11. Cart Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	user	User	User who own the cart

03	cartItems	Set<CartItem>	List of items in cart

3.2.12. CartItem Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	cart	Cart	cart that store items
03	productSKU	Product_SKU	product SKU of cart item
04	price	Double	price total of cart item
05	quantity	int	quantity of cart item

3.2.13. Order Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	orderStatus	String	Status of order
03	user	User	User who create order
04	paymentStatus	String	Payment status of order
05	payment	Payment	Payment method of order
06	orderItems	Set<OrderItem>	List of items in the order
07	subTotal	Double	Total price of items excluding delivery fee and voucher discount
08	voucher	Voucher	Voucher that applied to the order
09	deliveryFeeTotal	Double	Delivery fee of order
10	paymentTotal	Double	Total price of the order including delivery fee and voucher discount
11	orderDate	Date	Date that order is created
12	paymentDate	Date	Date that order is paid
13	address	Address	Address to deliver order to the user

3.2.14. OrderItem Class

No	Attributes	Type	Description

01	id	Long	Unique identifier, auto increment
02	order	Order	Order that store items
03	productSKU	Product_SKU	product SKU of order item
04	price	Double	price of order item
05	quantity	int	quantity of order item

3.2.15. Payment Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	name	String	Name of payment method

3.2.16. Review Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	order	Order	Completed order that user create and contain reviewed product
03	product	Product	Product that is reviewed
04	user	User	User who create review
05	description	String	Feedback comment of user
06	numberRating	int	Number rating for the product

3.2.17. Voucher Class

No	Attributes	Type	Description
01	id	Long	Unique identifier, auto increment
02	code	String	Unique code of voucher
03	name	String	Name of voucher
04	type	String	Type of voucher (percentage/fix value)
05	description	String	Description details of voucher
06	minSpend	Double	Minimum price total of order to apply voucher
07	maxDiscount	Double	Max discount of price to the order

08	discountAmount	Double	Number of discount percentage of the voucher if the type is percentage
09	isActive	boolean	Activation status of voucher
10	fromDate	Date	The date the voucher is available
11	toDate	Date	The date the voucher is expired
12	quantity	int	The number of times that a voucher can be applied

3.3. Sequence Diagrams

3.3.1. Admin

3.3.1.1. Login/Register

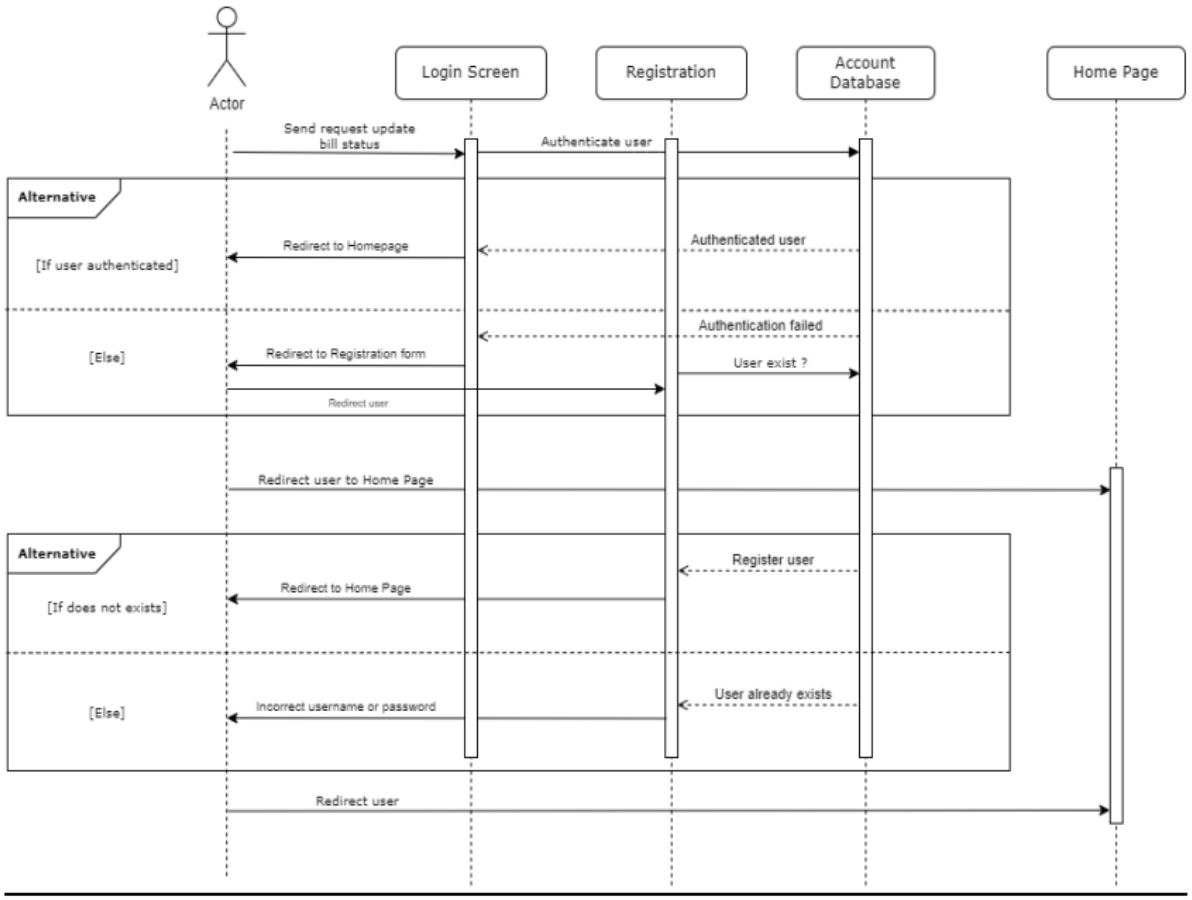


Figure 30: Login/Register Diagram with Admin, Staff Roles

3.3.1.2. Forgot Password

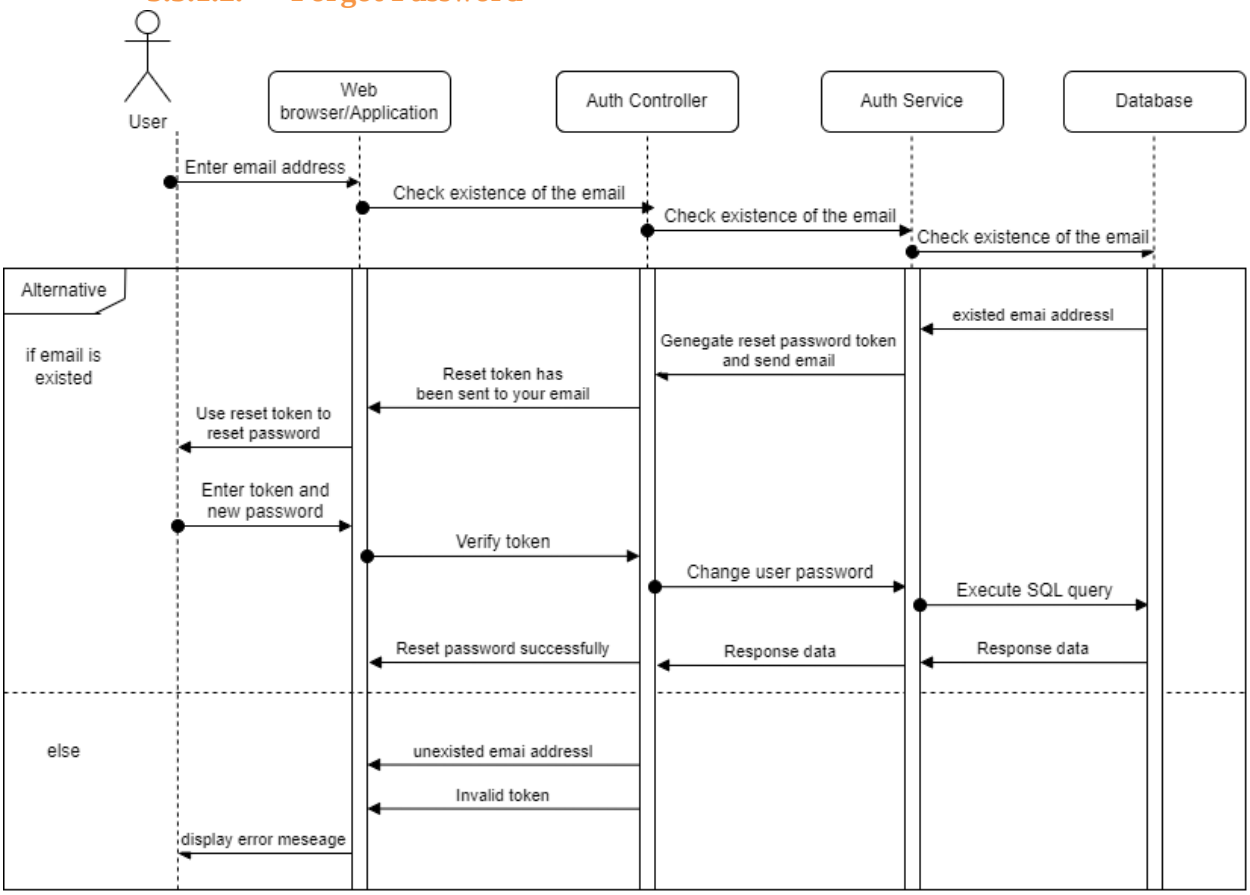


Figure 31: Forgot Password Diagram with Admin, Staff Roles

3.3.1.3. View Profile

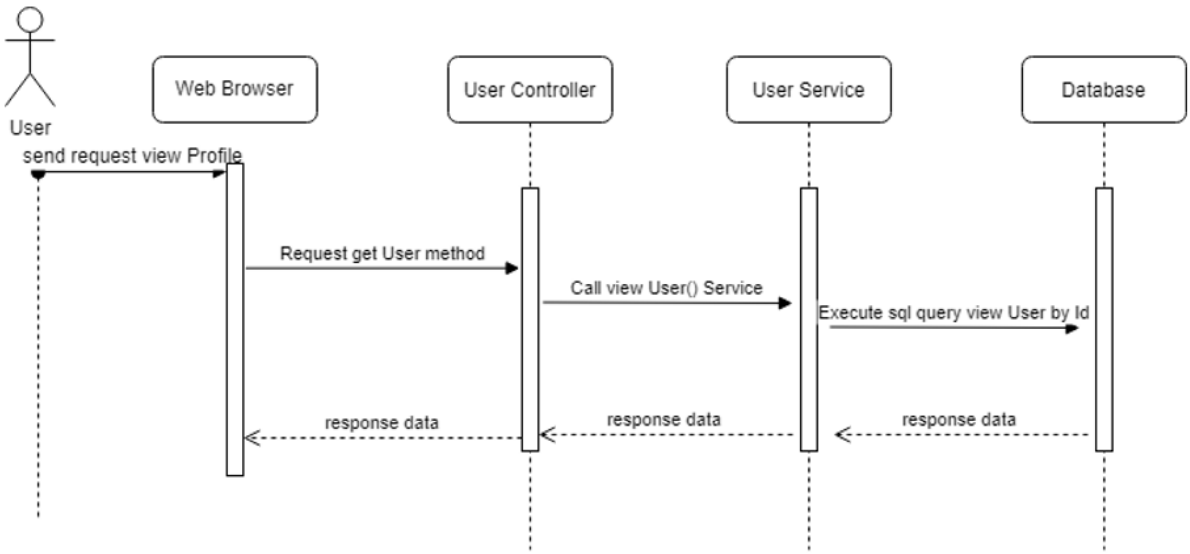


Figure 32: View Profile Diagram with Admin, Staff Roles

3.3.1.4. Update Profile

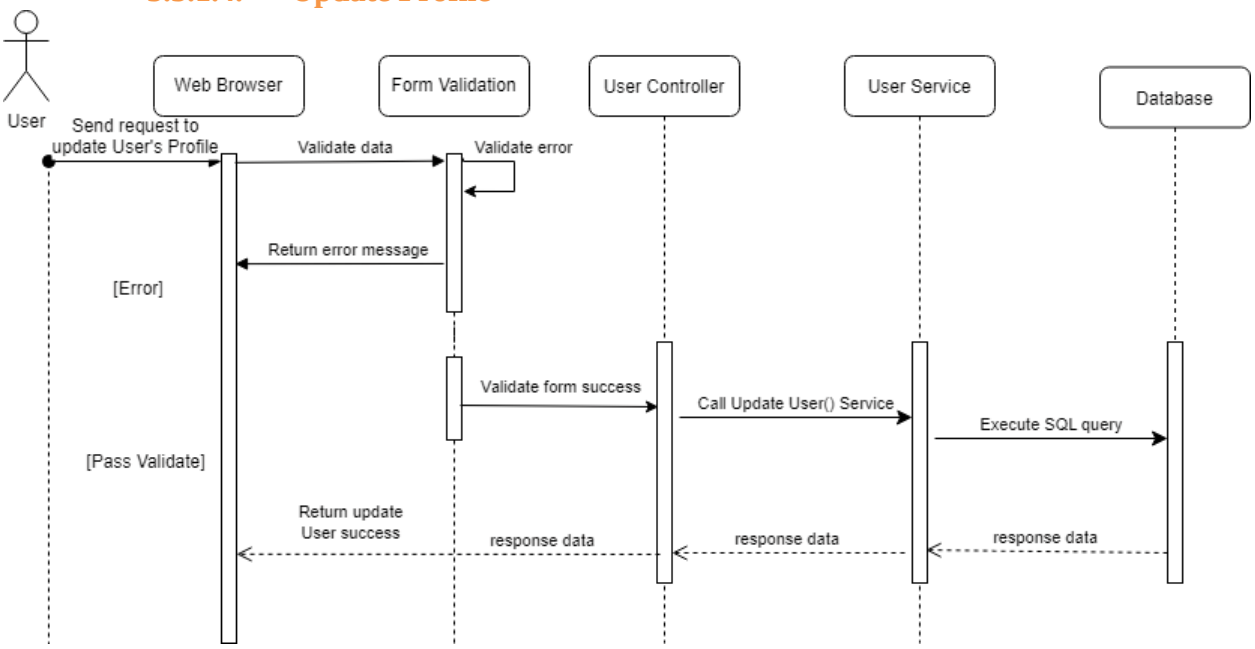


Figure 33: Update Profile Diagram with Admin, Staff Roles

3.3.1.5. Change Password

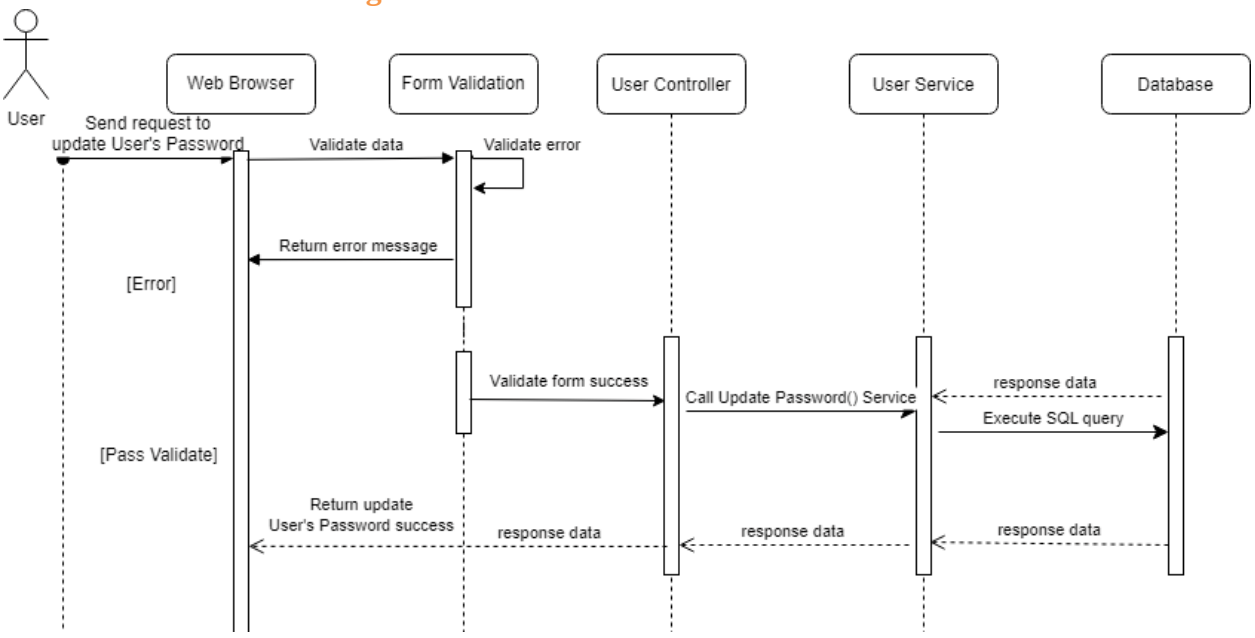


Figure 34: Change Password Diagram with Admin, Staff Roles

3.3.1.6. View Products/Product Details

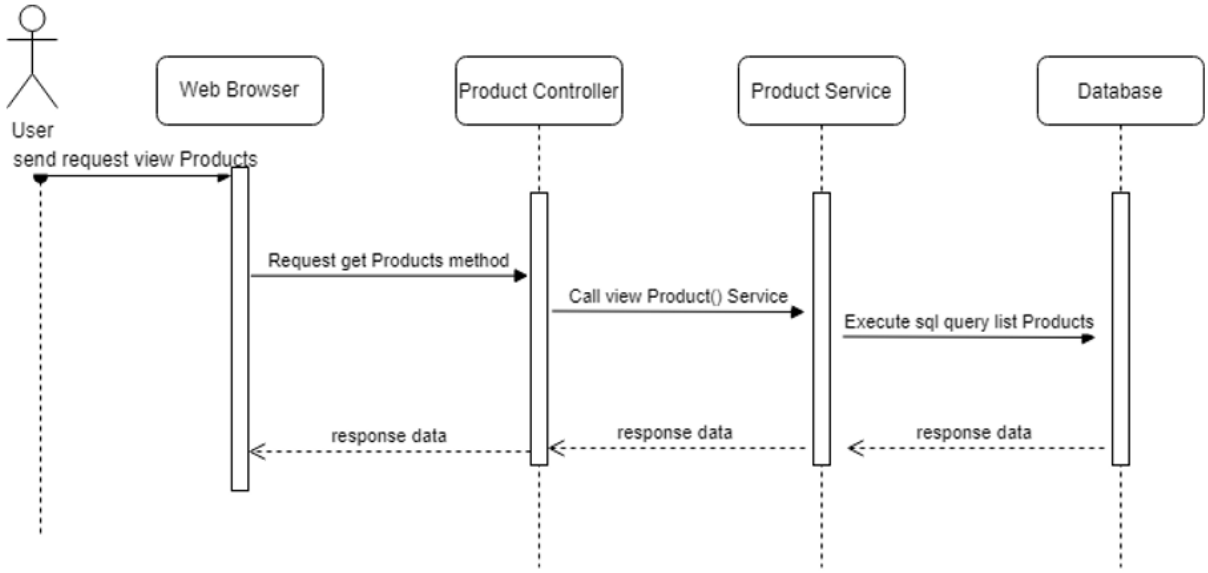


Figure 35: View Products/Product Details Diagram with Admin, Staff Roles

3.3.1.7. Create Product

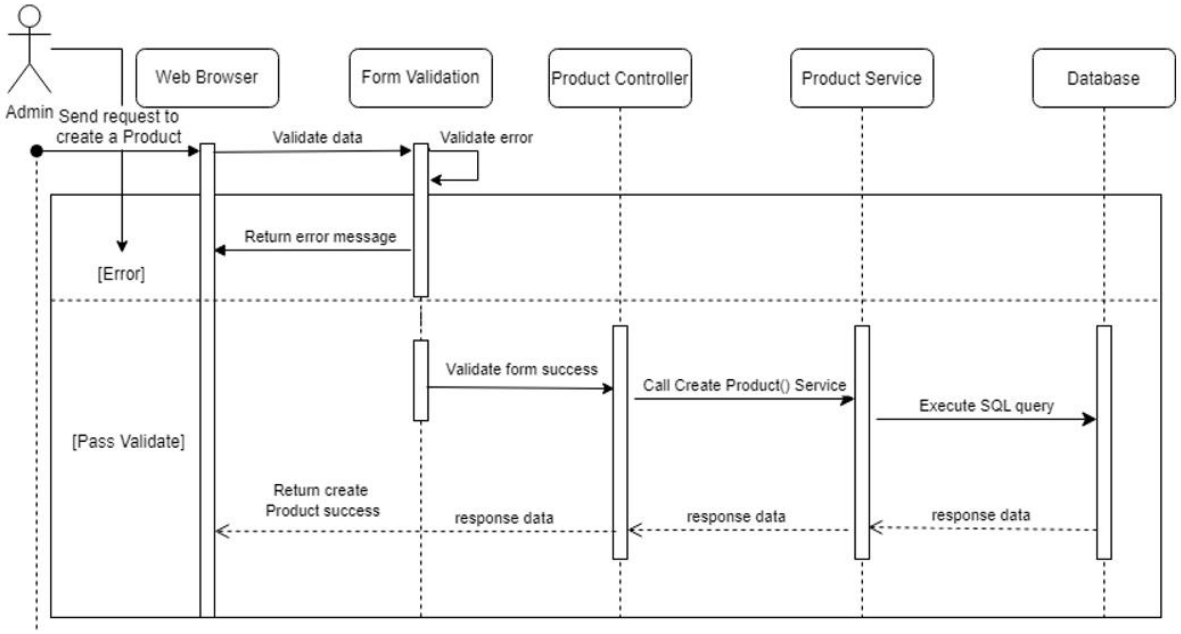


Figure 36: Create Product Diagram with Admin Role

3.3.1.8. Update Product

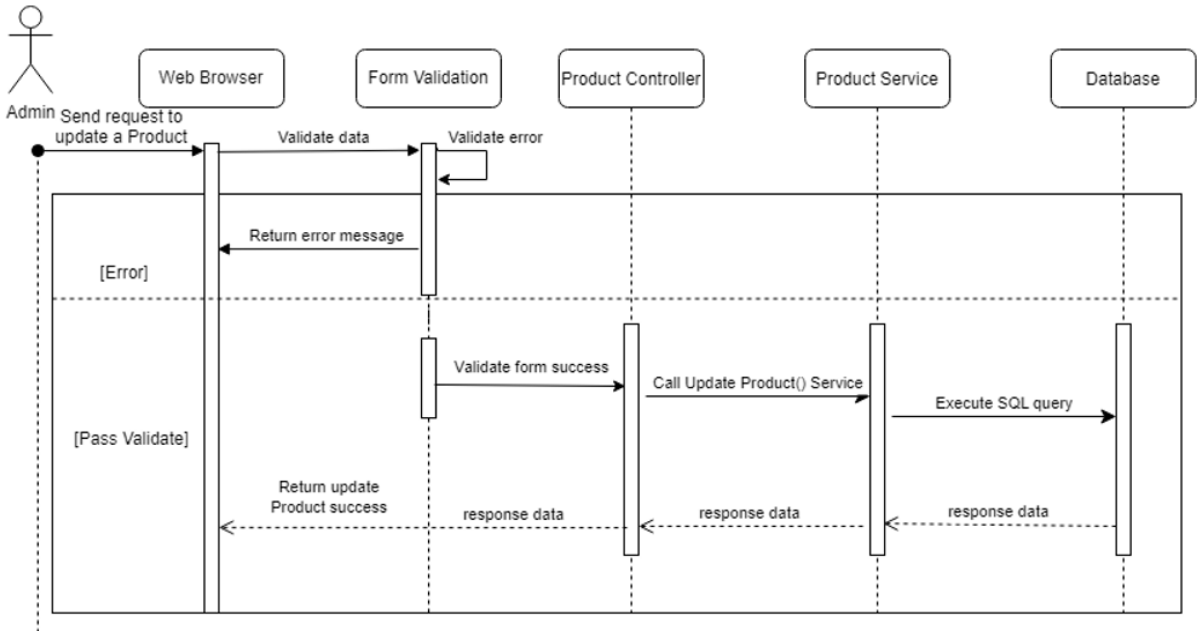


Figure 37: Update Product Diagram with Admin Role

3.3.1.9. Delete Product

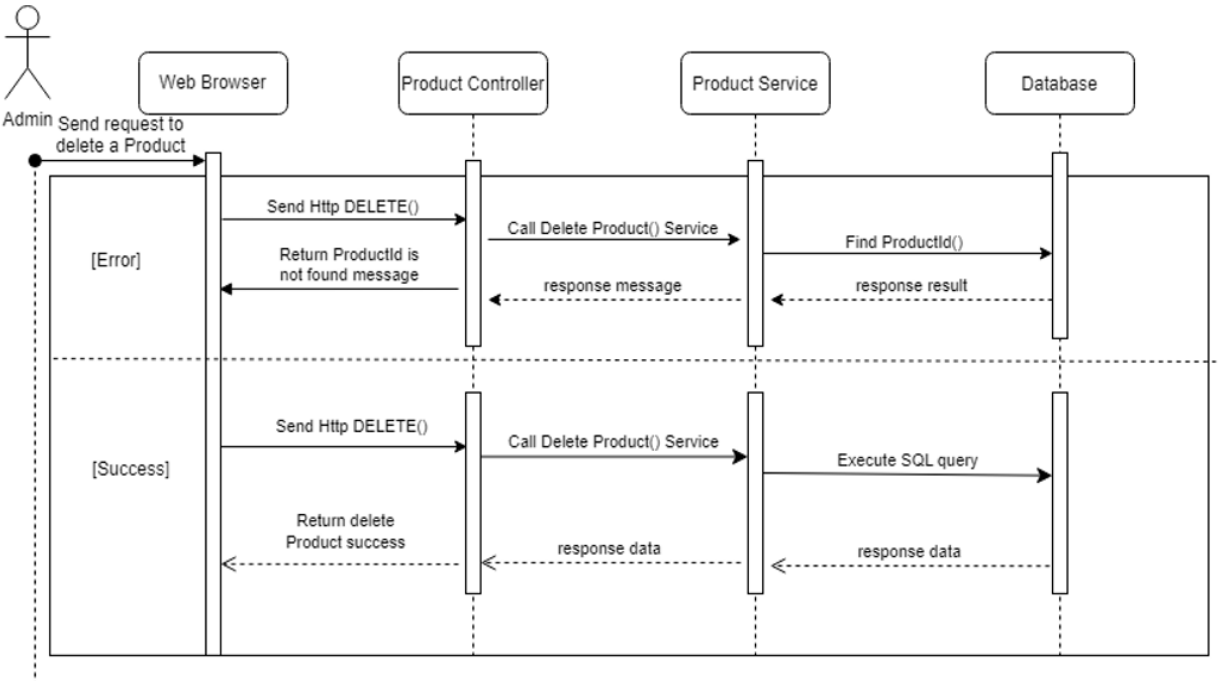


Figure 38: Delete Product Diagram with Admin Role

3.3.1.10. View Orders

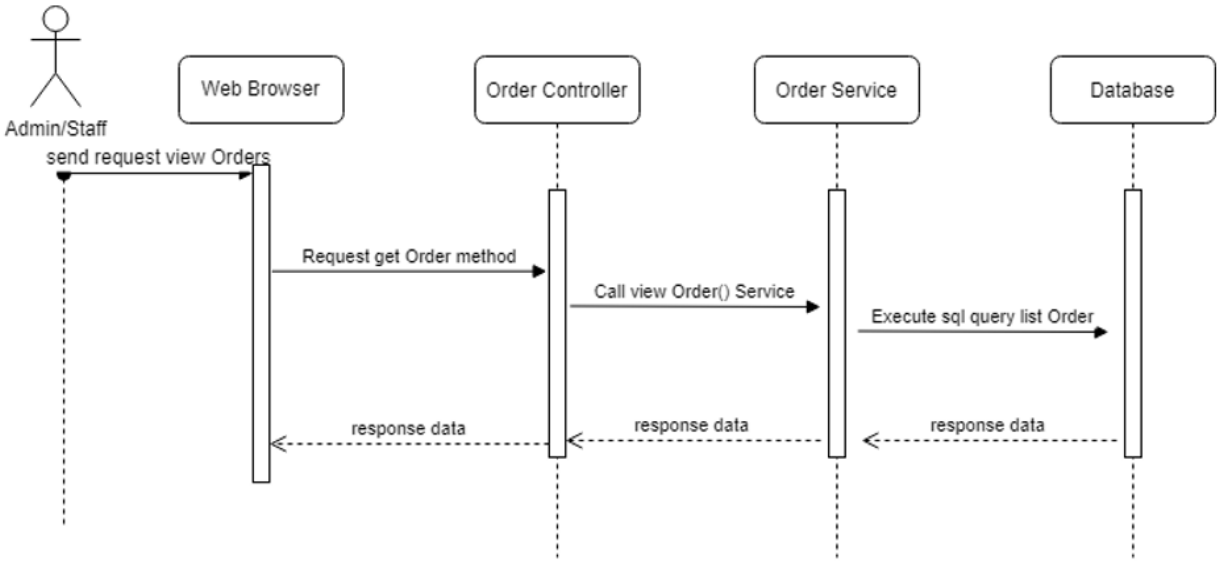


Figure 39: View Orders Diagram with Admin, Staff Roles

3.3.1.11. Change Order Status

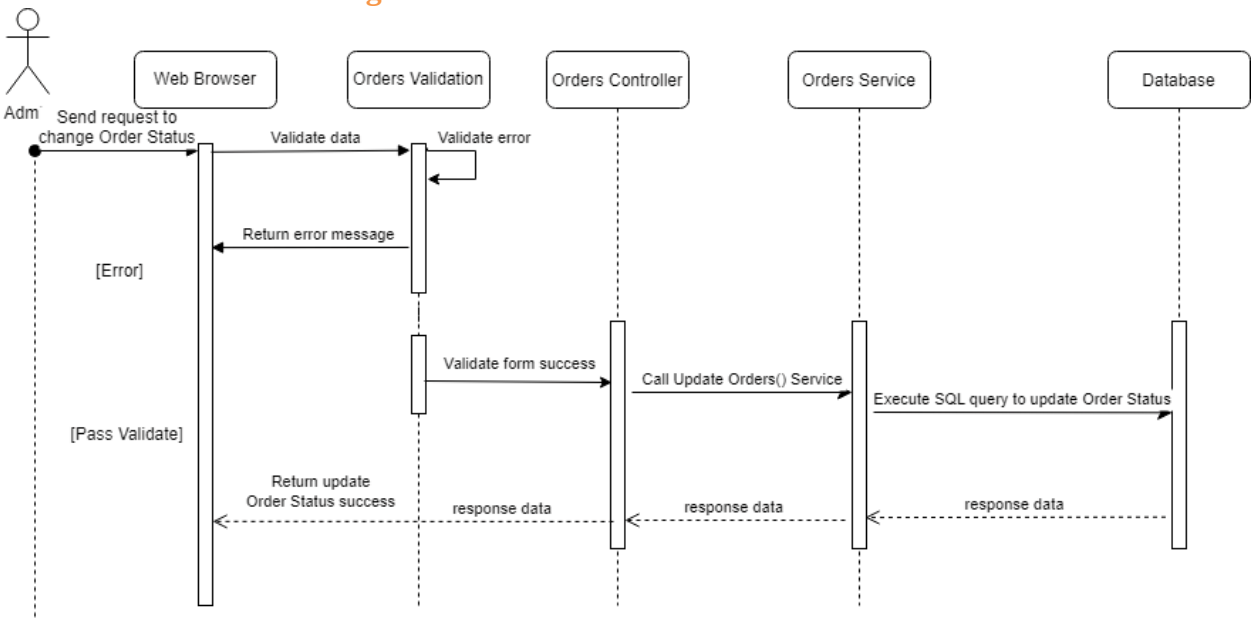


Figure 40: Change Order Status Diagram with Admin, Staff Roles

3.3.1.12. View Account

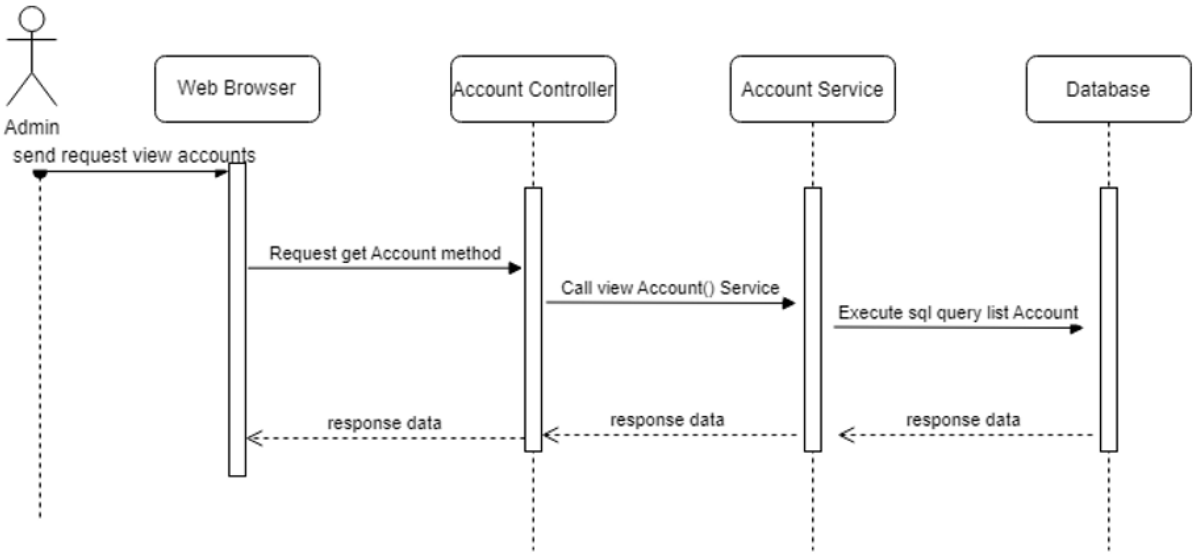


Figure 41: View Account Diagram with Admin, Staff Roles

3.3.1.13. Create Staff

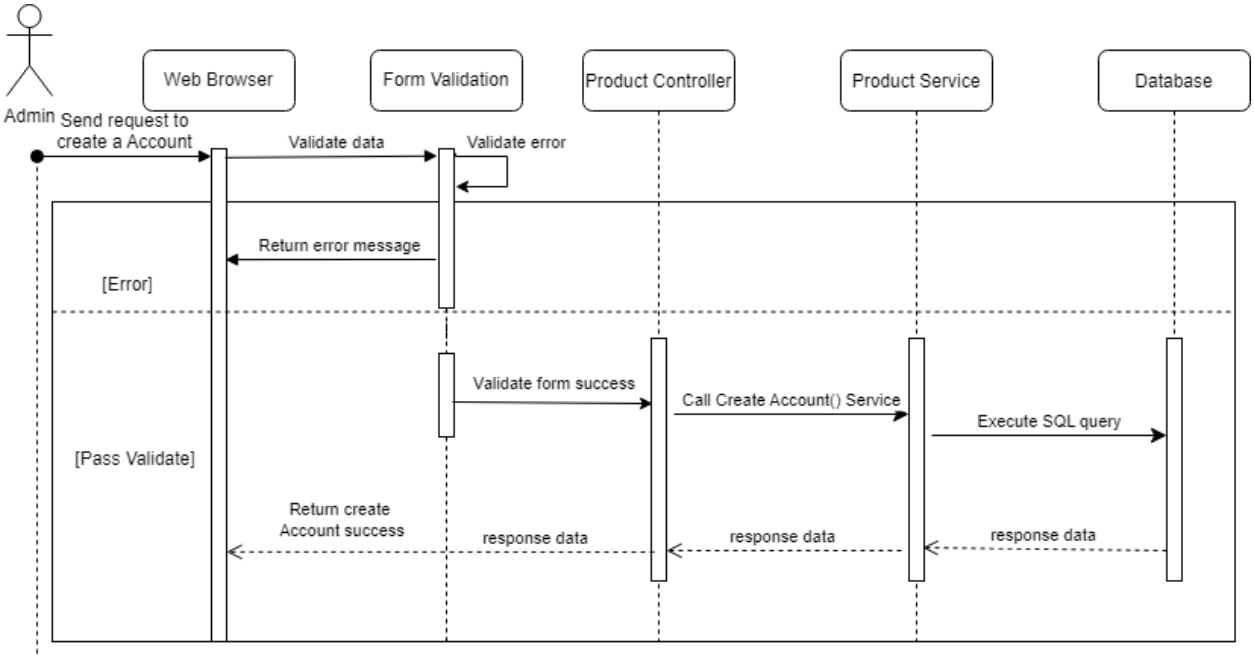


Figure 42: Create Staff Diagram with Admin Role

3.3.1.14. Change Roles

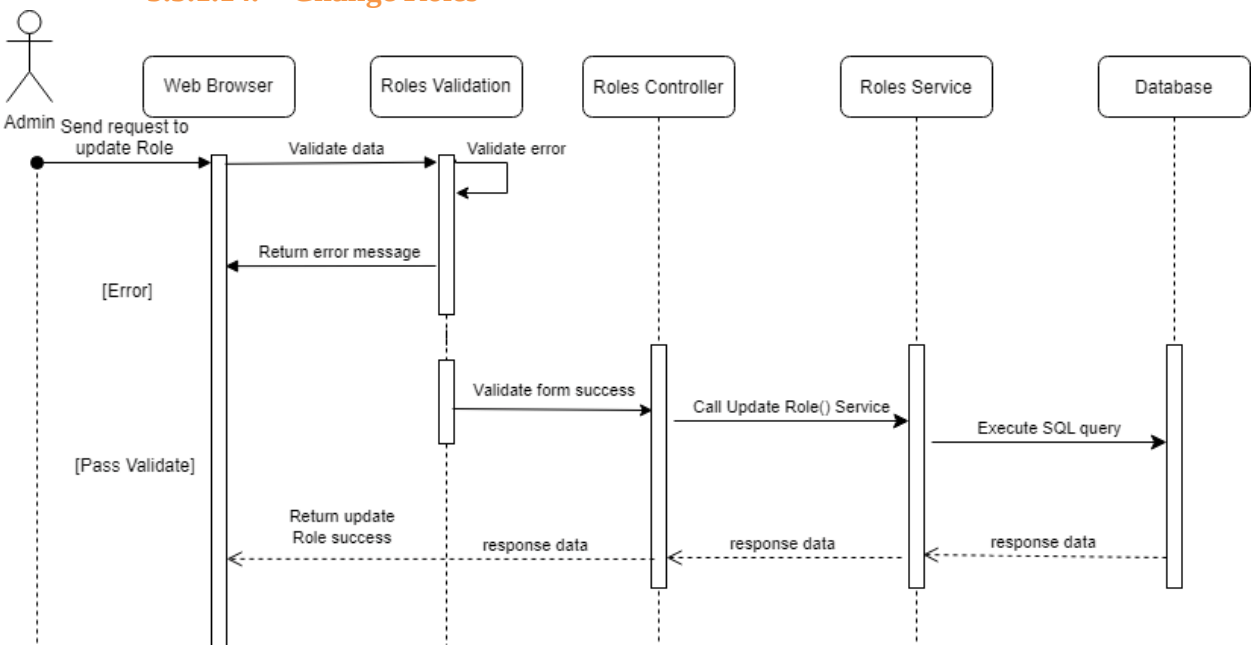


Figure 43: Change Roles Diagram with Admin Role

3.3.1.15. Delete Account

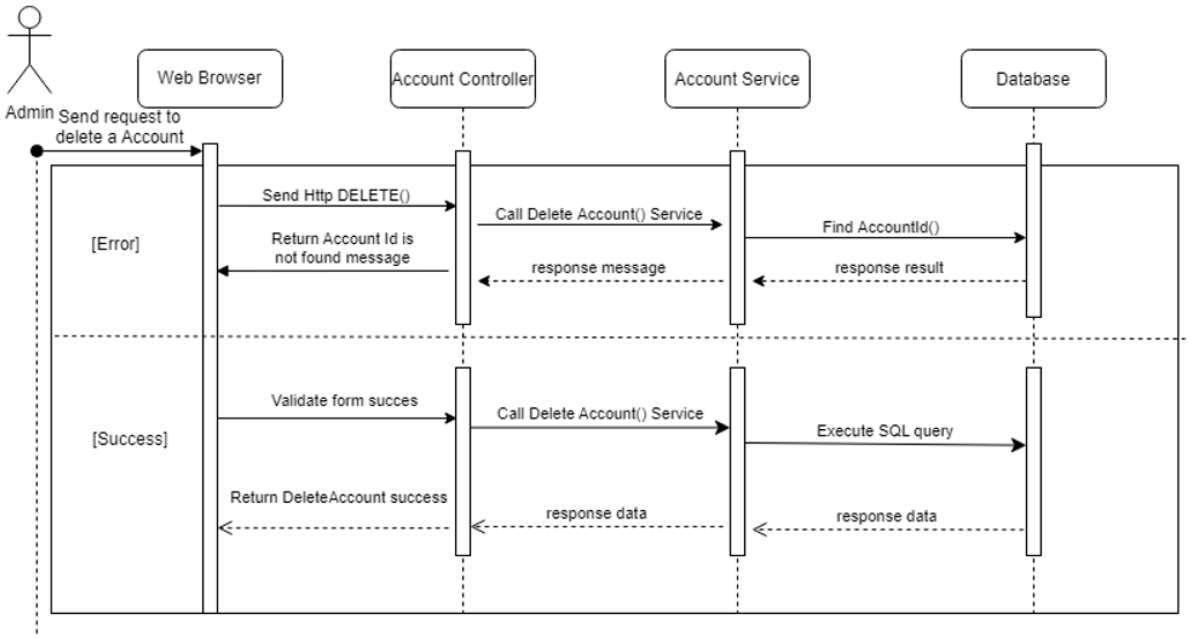


Figure 44: Delete Account Diagram with Admin Role

3.3.1.16. View Category

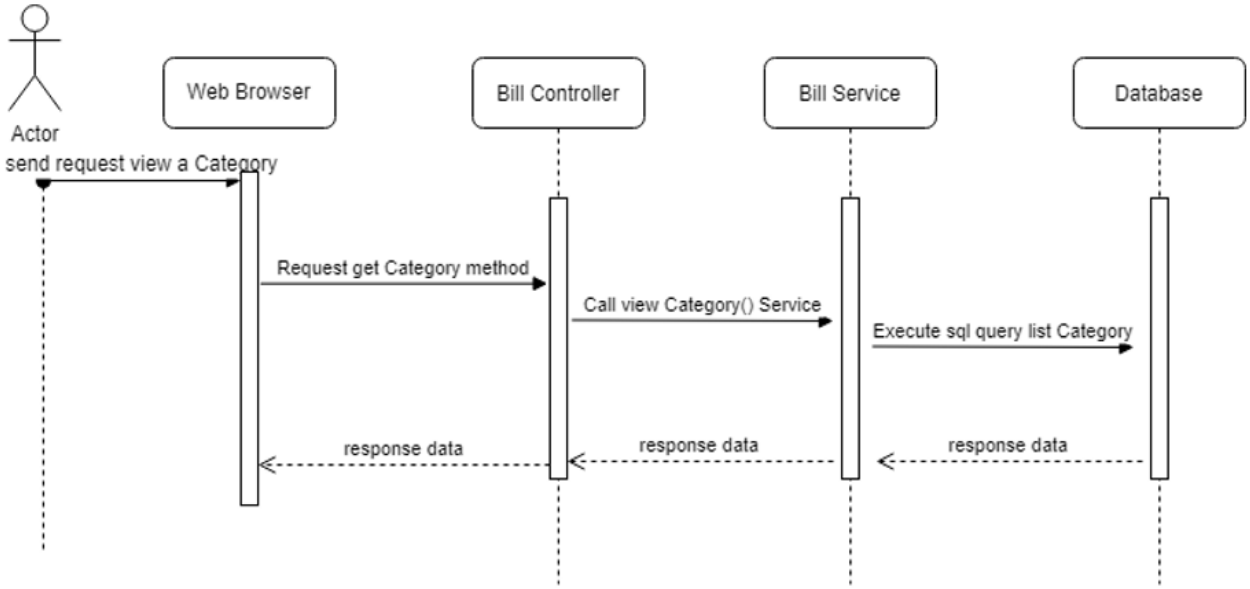


Figure 45: View Category Diagram with Admin, Staff Roles

3.3.1.17. Create category

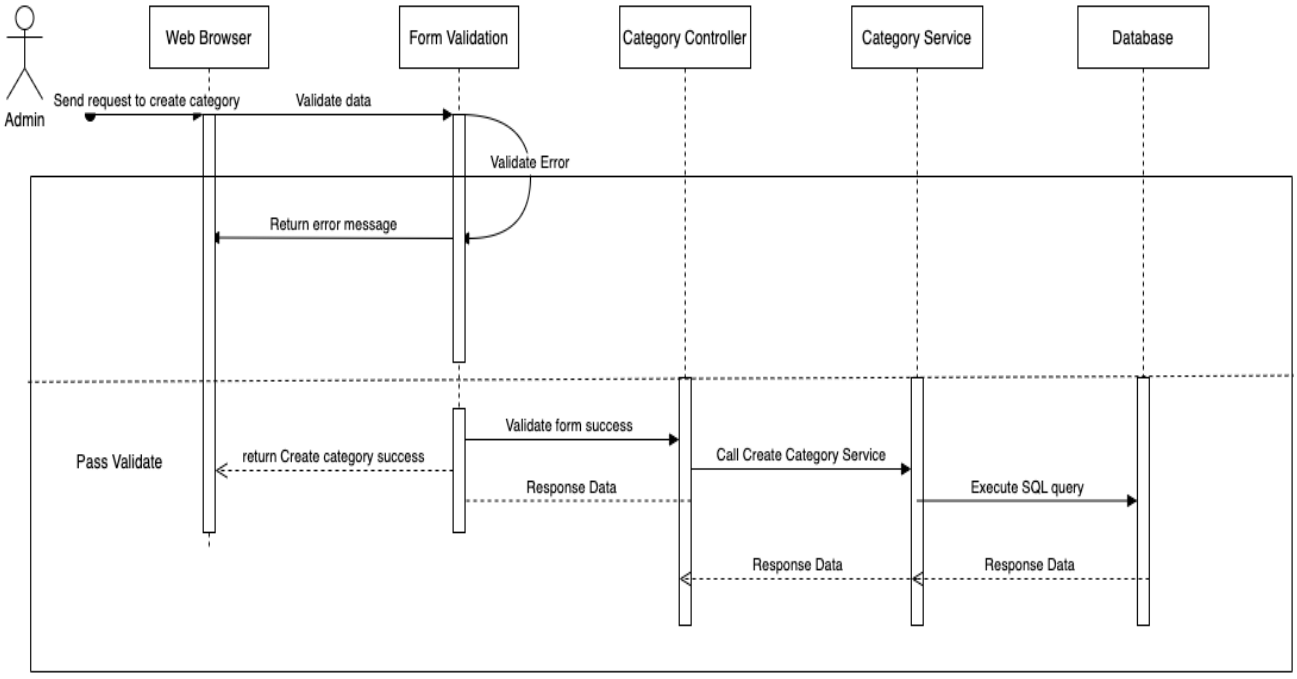


Figure 46: Create Category Diagram with Admin Role

3.3.1.18. Update category

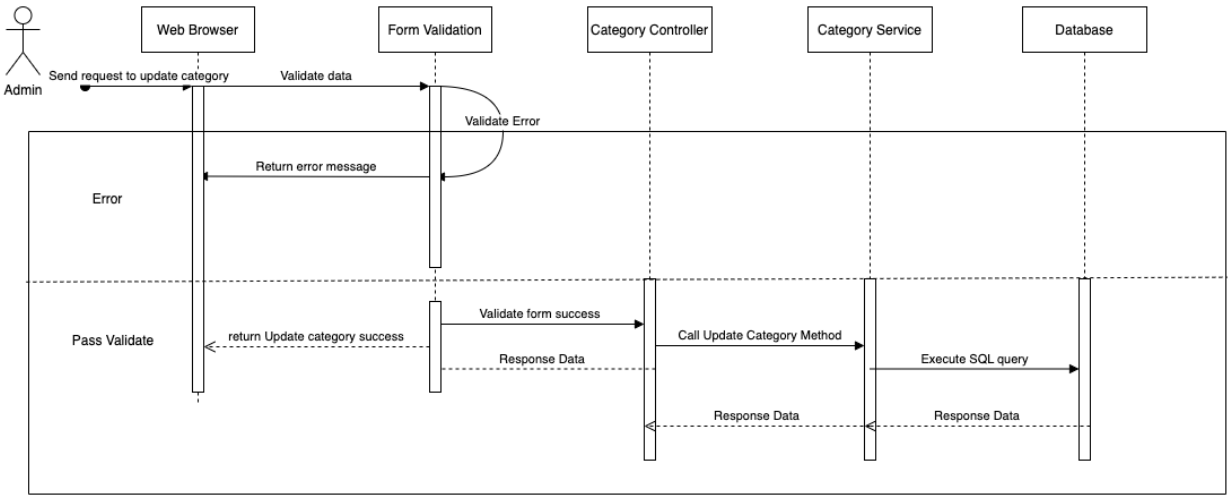


Figure 47: Update Category Diagram with Admin Role

3.3.1.19. Delete category

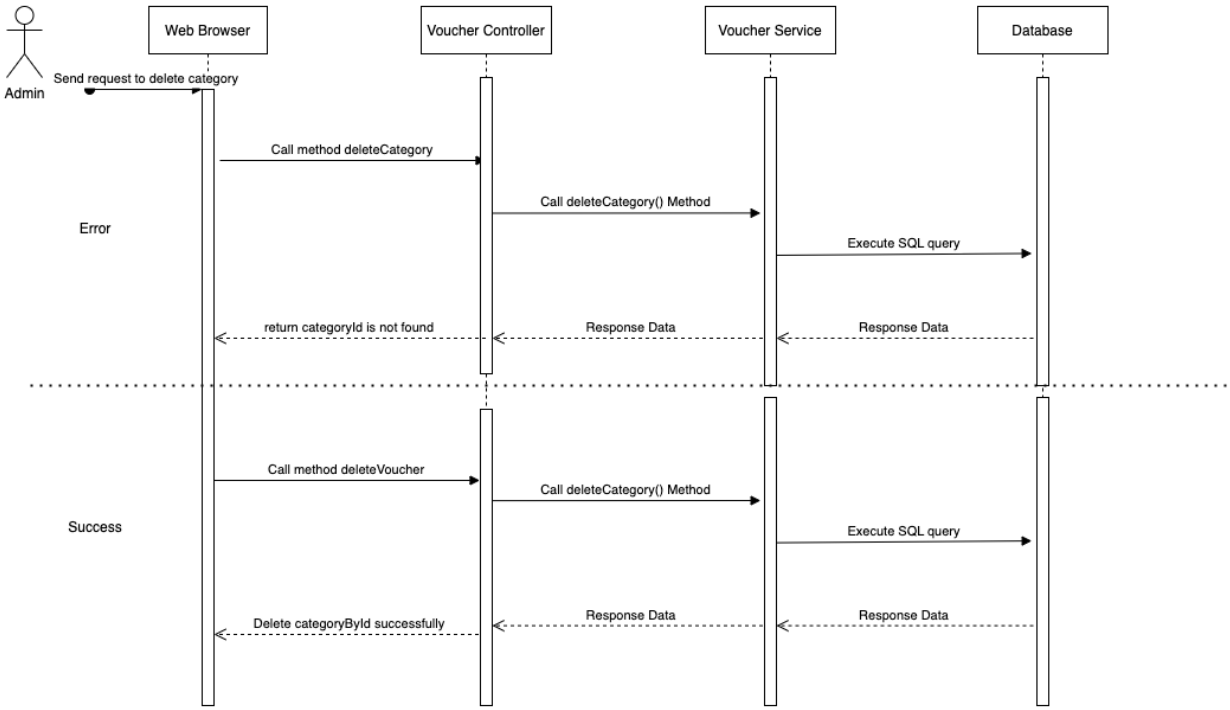


Figure 48: Delete Category Diagram with Admin Role

3.3.1.20. View Vouchers

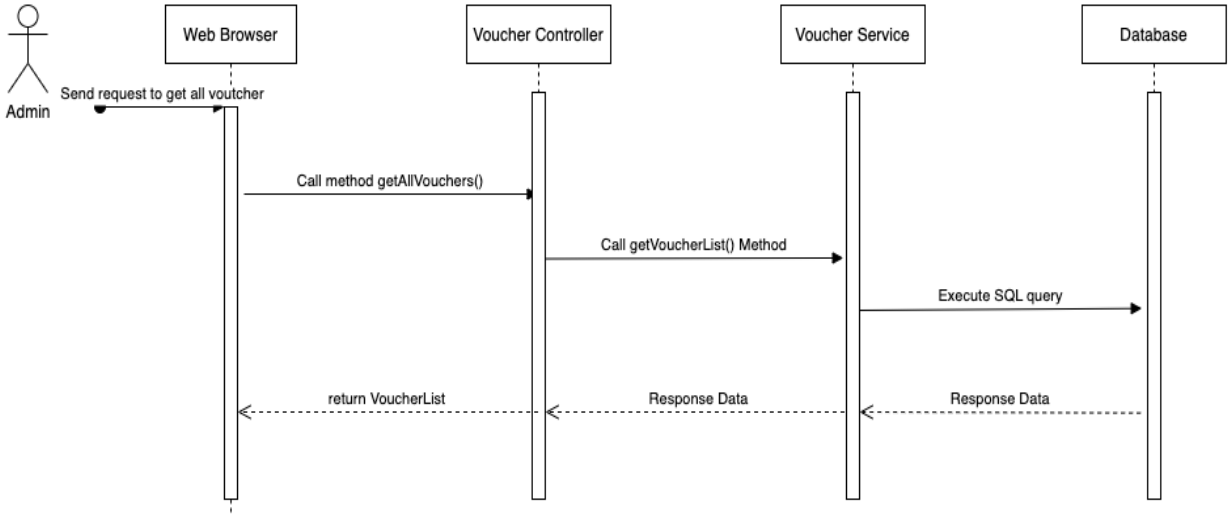


Figure 49: View Vouchers Diagram with Admin, Staff Roles

3.3.1.21. Create Voucher

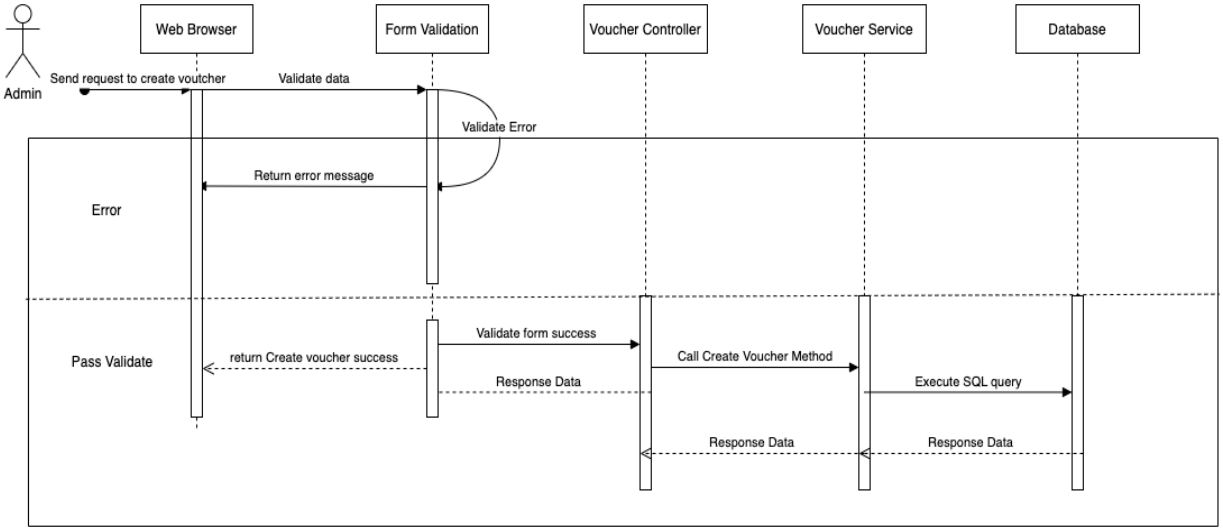


Figure 50: Create Voucher Diagram with Admin Role

3.3.1.22. Update Voucher

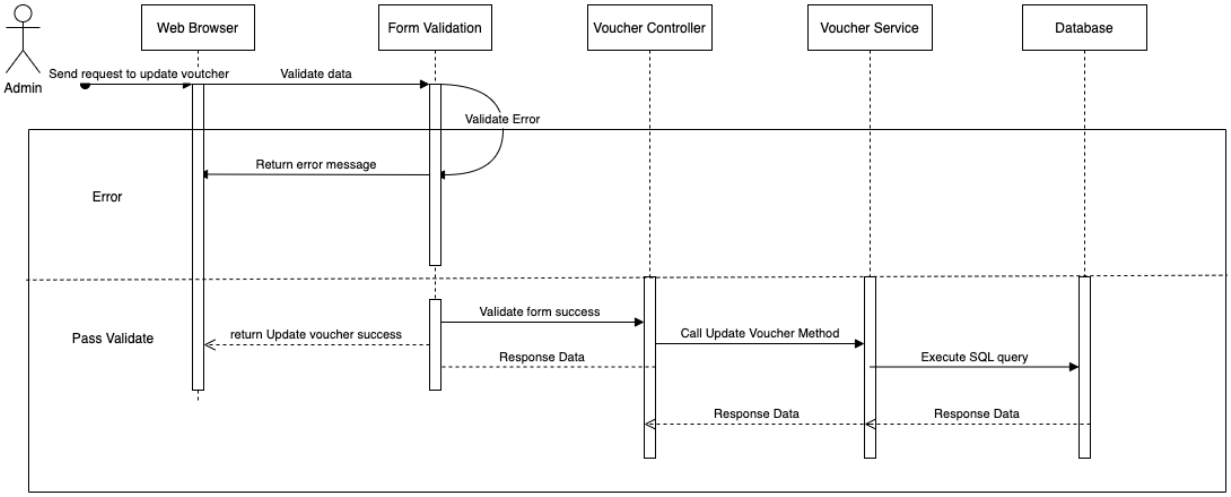


Figure 51: Update Voucher Diagram with Admin Role

3.3.1.23. Change Activation

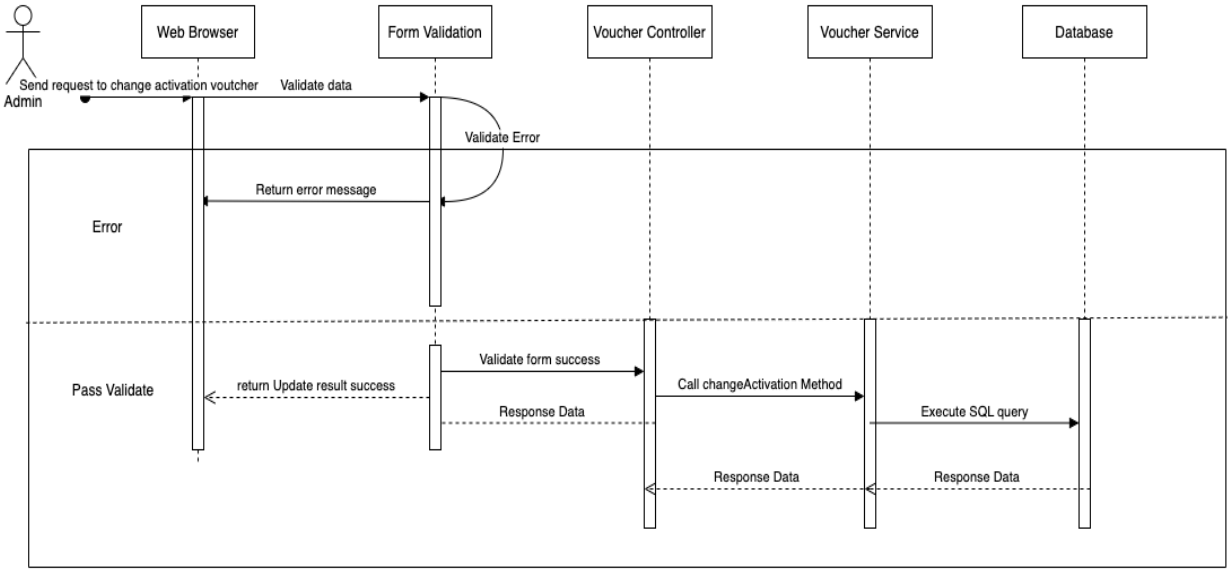


Figure 52: Change Activation Diagram with Admin Role

3.3.1.24. Delete Voucher

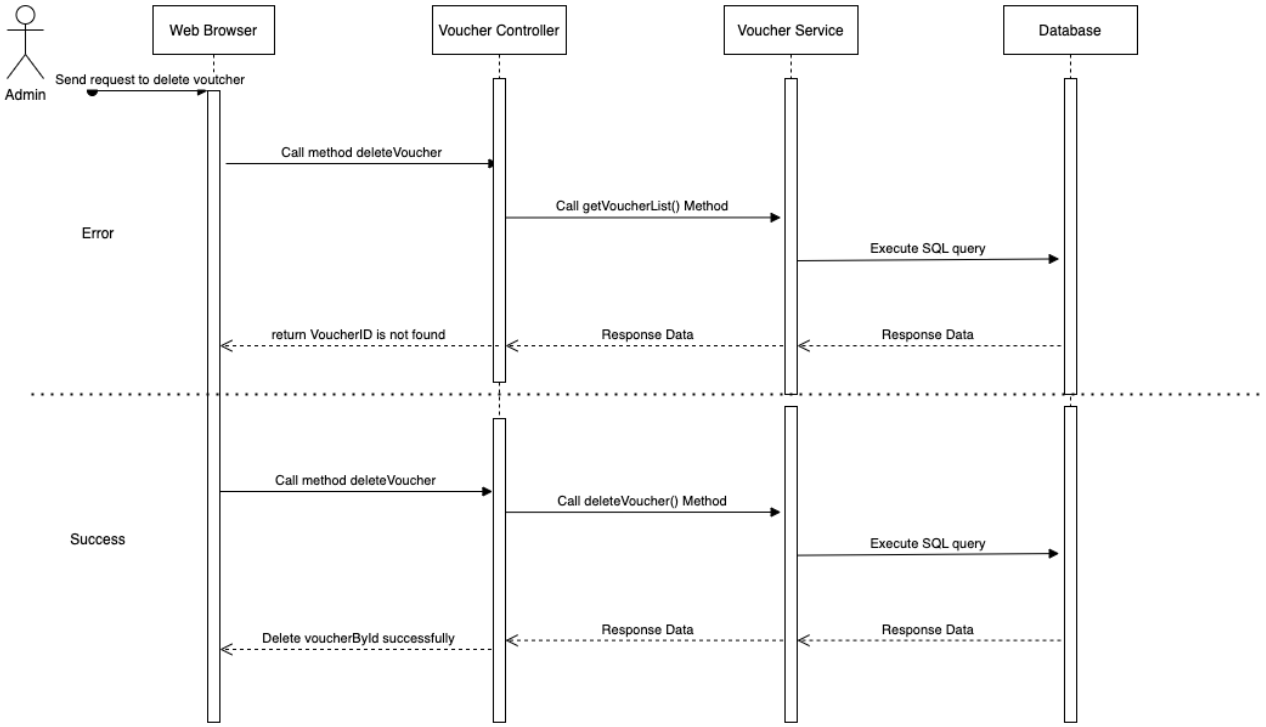


Figure 53: Delete Voucher Diagram with Admin Role

3.3.1.25. Extract feature product for recommend

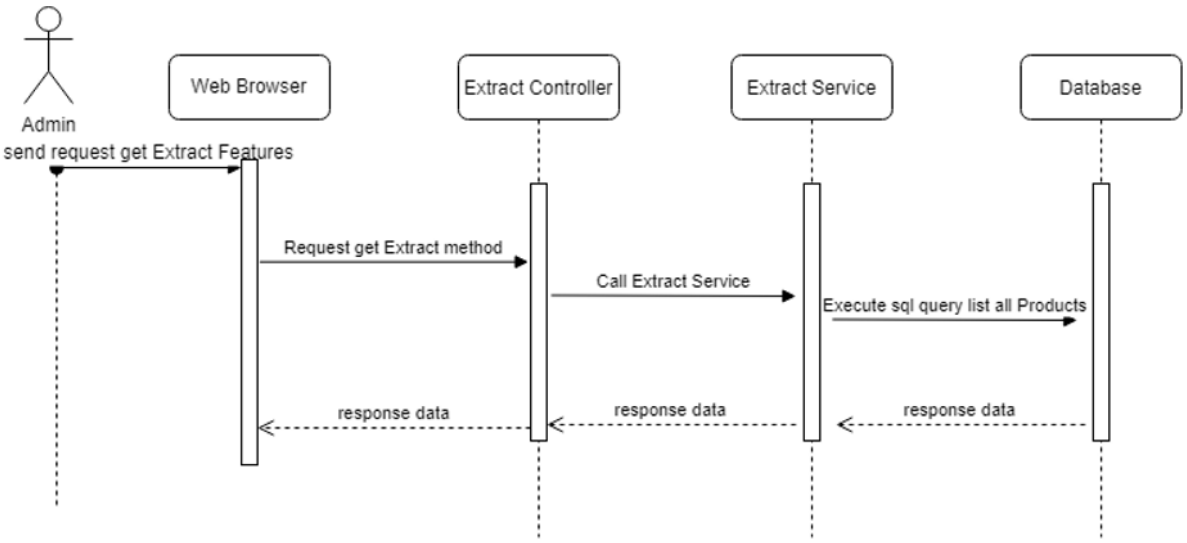


Figure 54: Extract feature product for recommend Diagram with Admin, Staff Roles

3.3.2. User

3.3.2.1. Login/Register

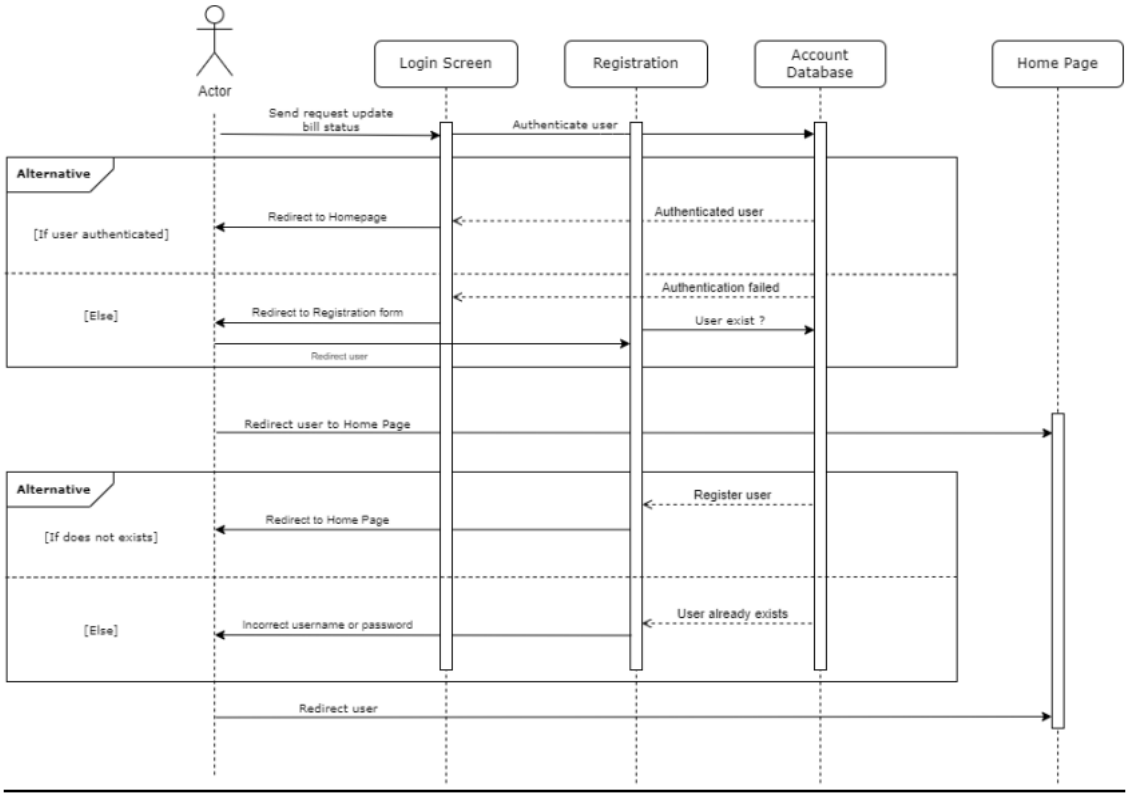


Figure 55: Login/Register Diagram with User Role

3.3.2.2. Forgot Password

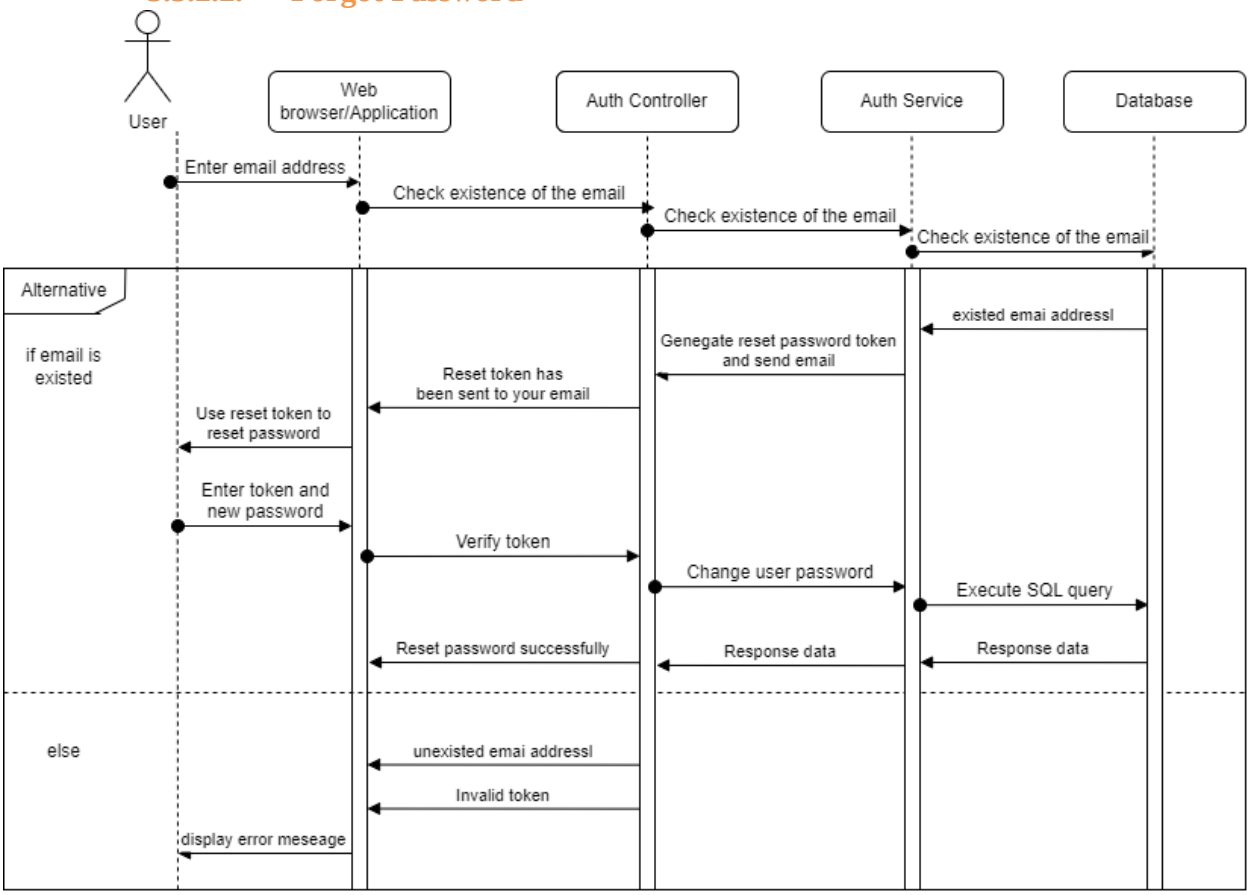


Figure 56: Forgot Password Diagram with User Role

3.3.2.3. View Profile

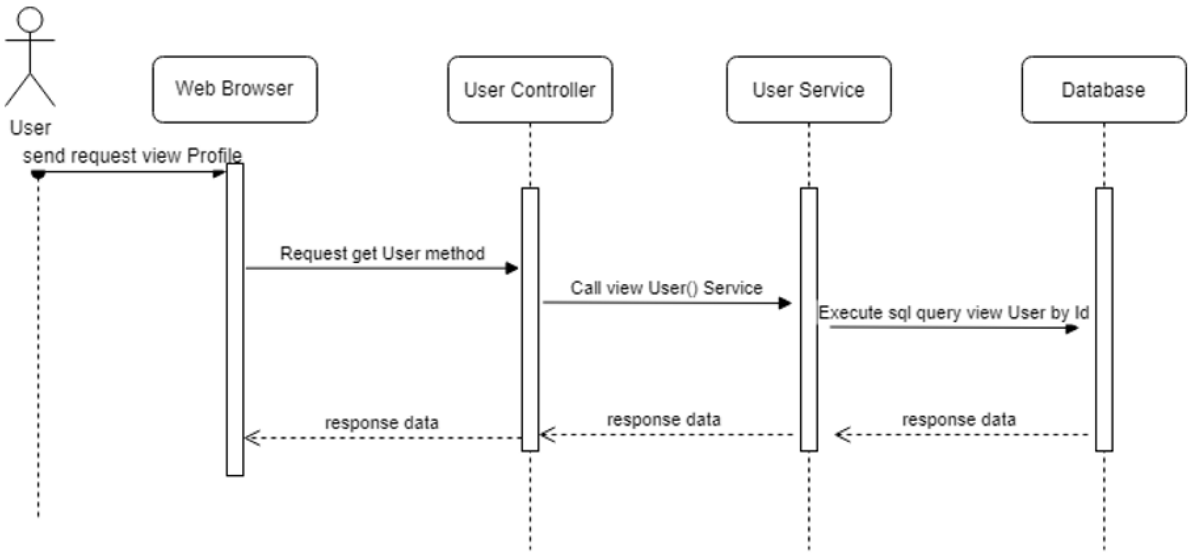


Figure 57: View Profile Diagram with User Role

3.3.2.4. Update Profile

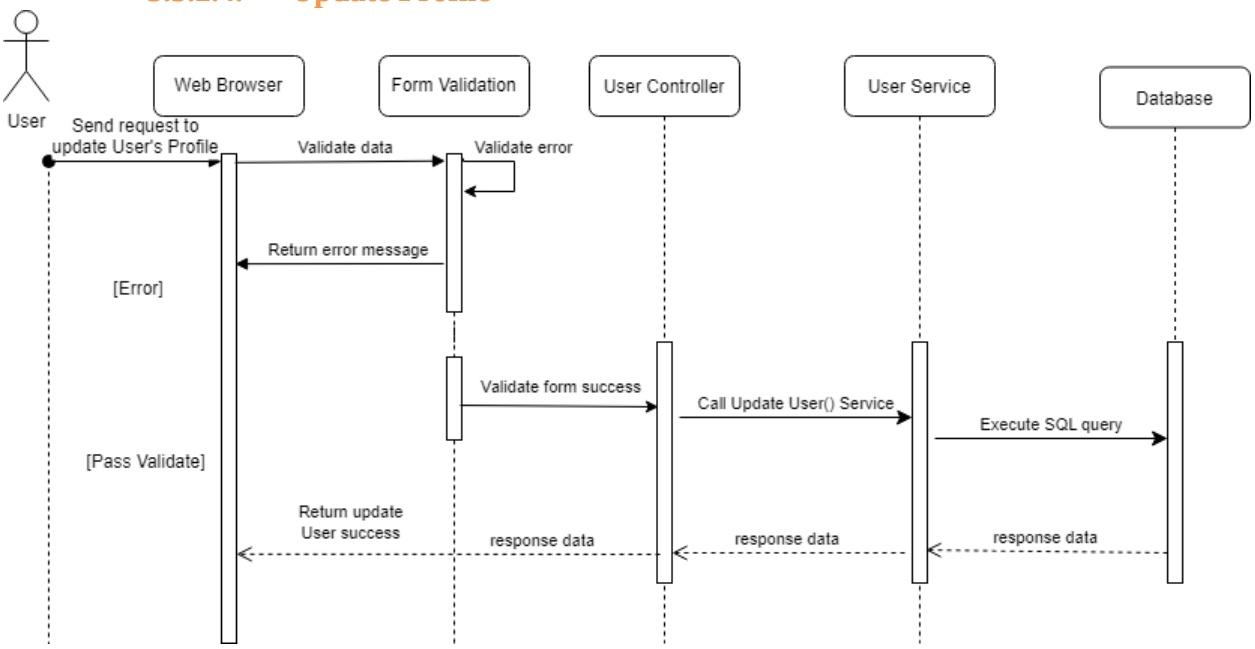


Figure 58: Update Profile Diagram with User Role

3.3.2.5. Change Password

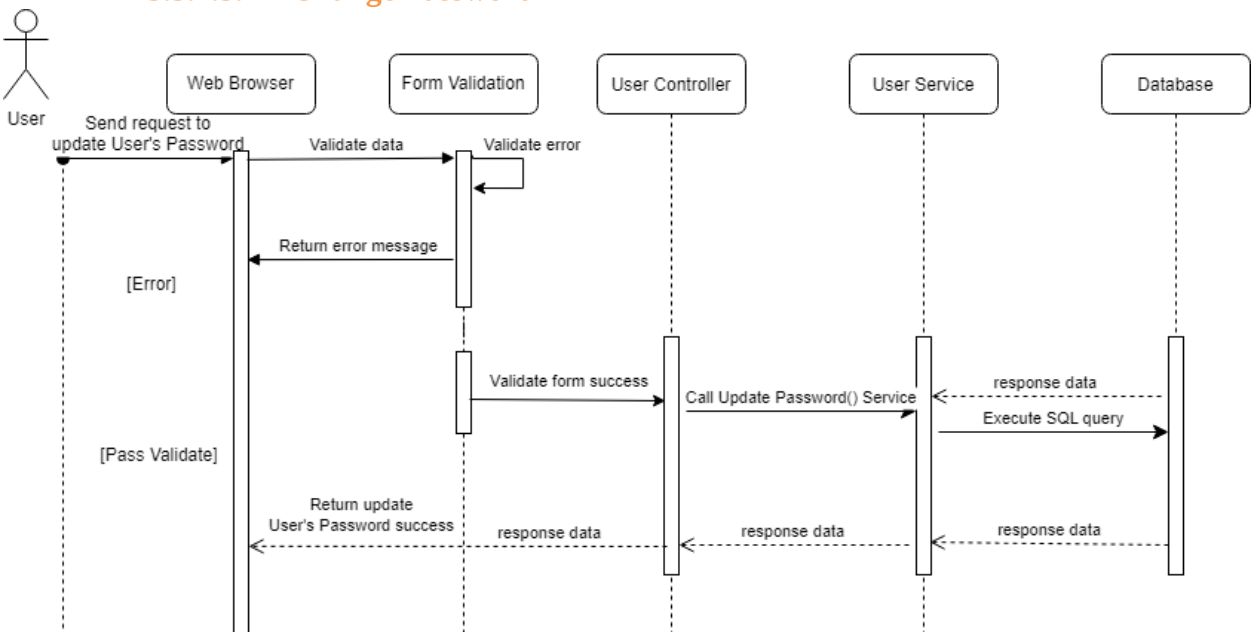


Figure 59: Change Password Diagram with User Role

3.3.2.6. View Products/Product Details

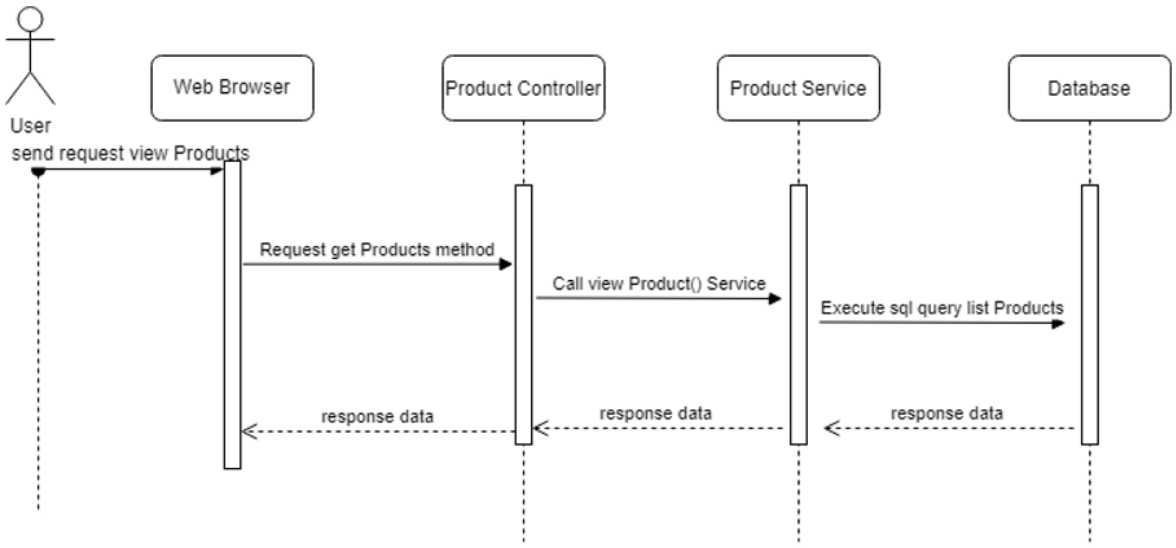


Figure 60: View Products/Product Details Diagram with User Role

3.3.2.7. Search product

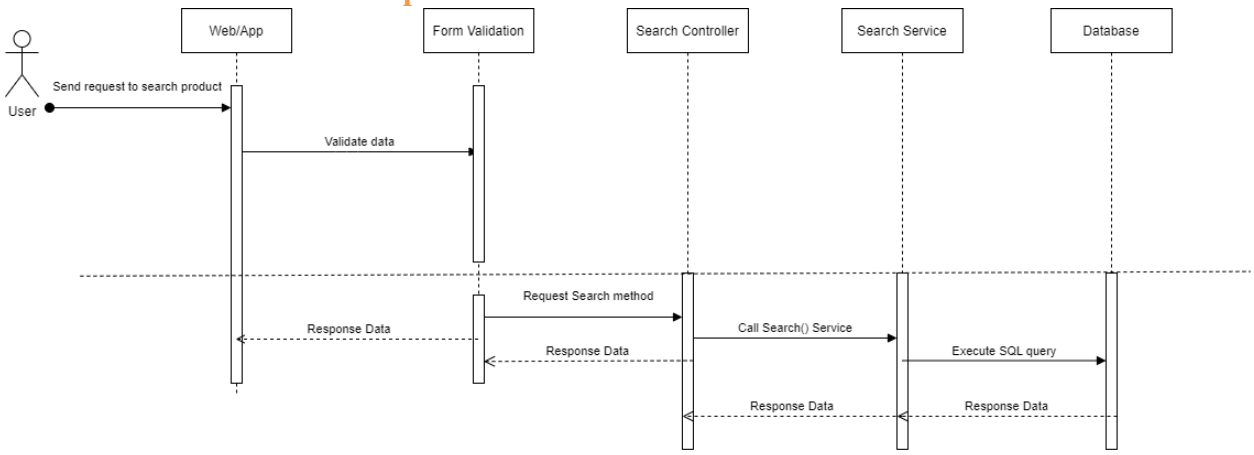


Figure 61: Search Product Diagram with User Role

3.3.2.8. View product reviews

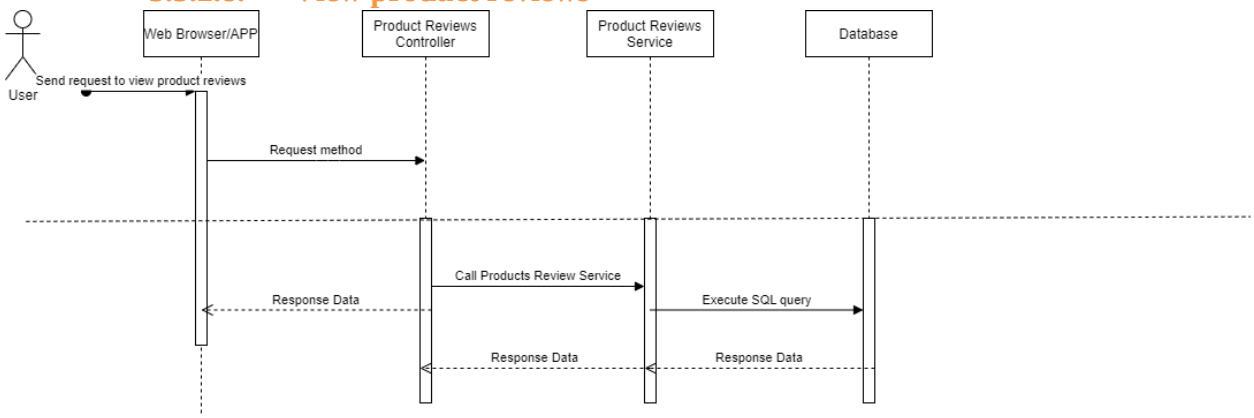


Figure 62: View Product Reviews Diagram with User Role

3.3.2.9. Add product to cart

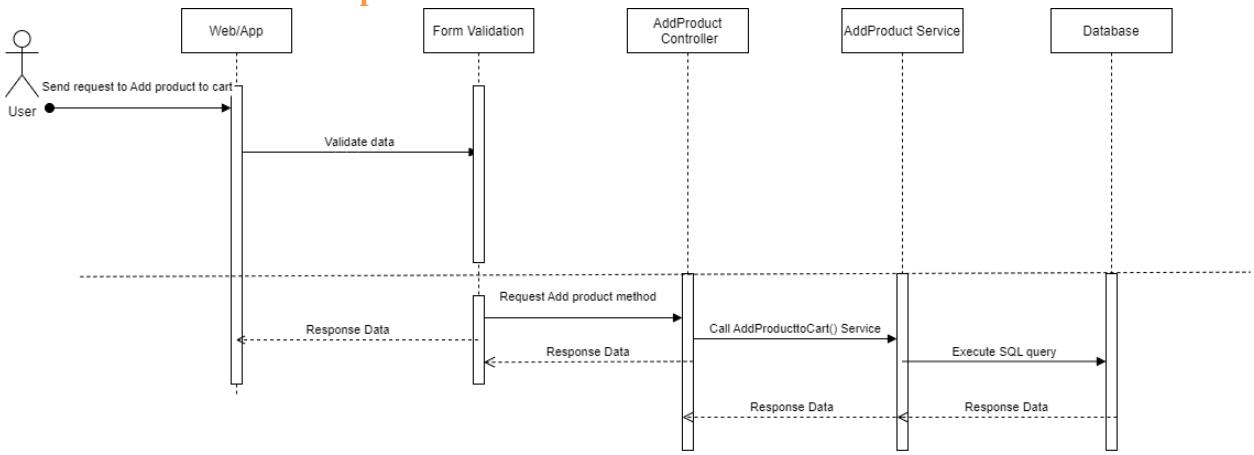


Figure 63: Add Product to Cart Diagram with User Role

3.3.2.10. View cart

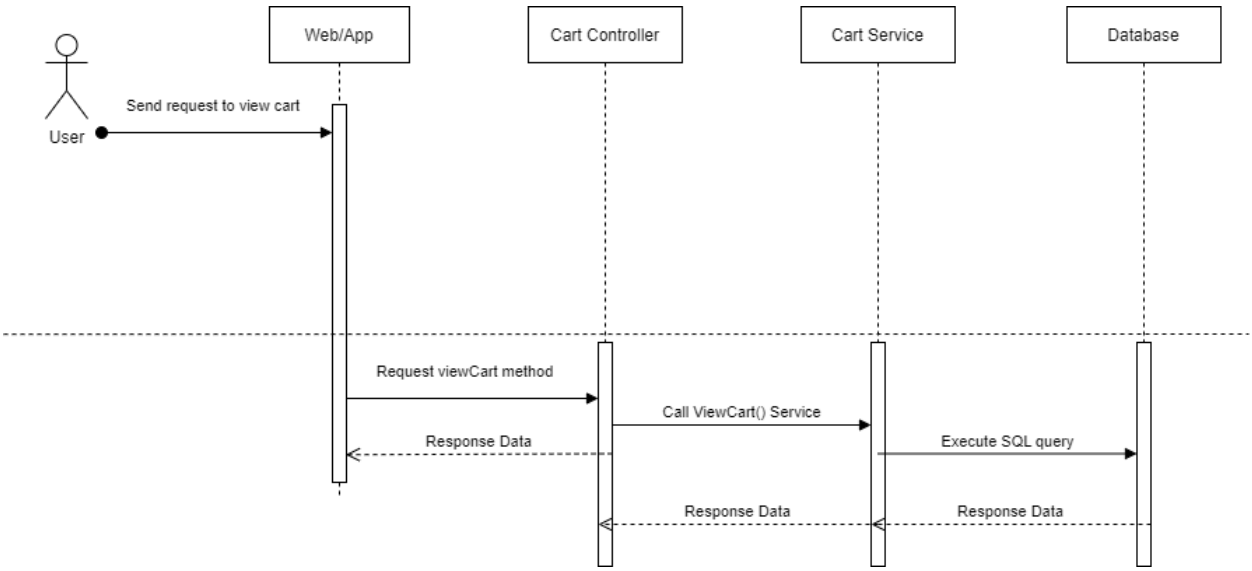


Figure 64: View Cart Diagram with User Role

3.3.2.11. Adjust quantity of cart items

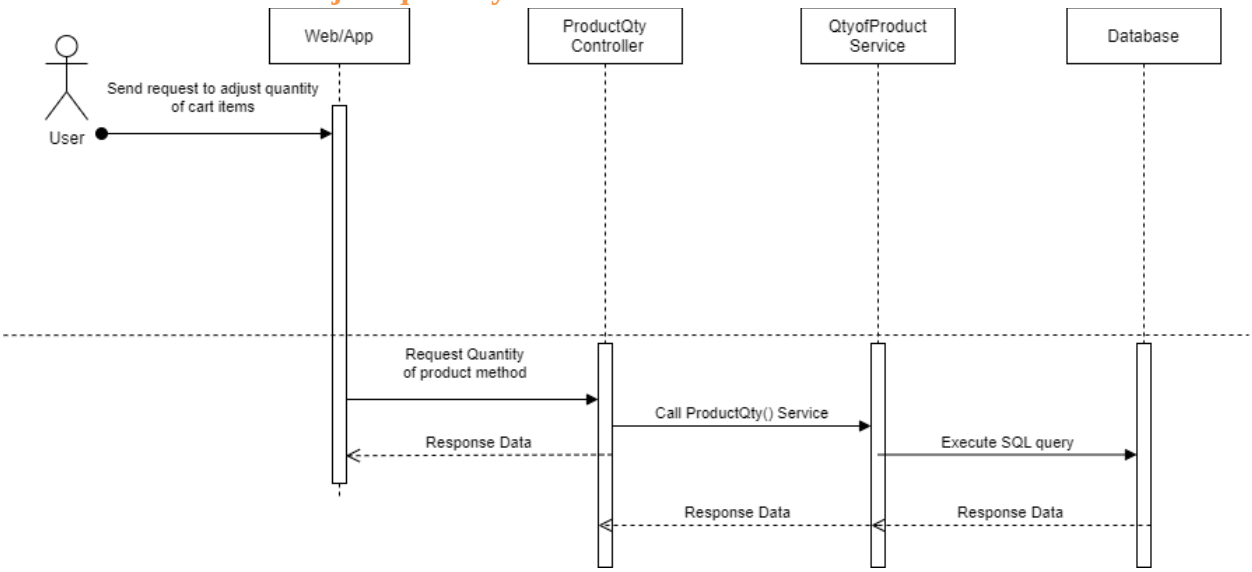


Figure 65: Adjust Quantity of Cart Items Diagram with User Role

3.3.2.12. Delete cart items

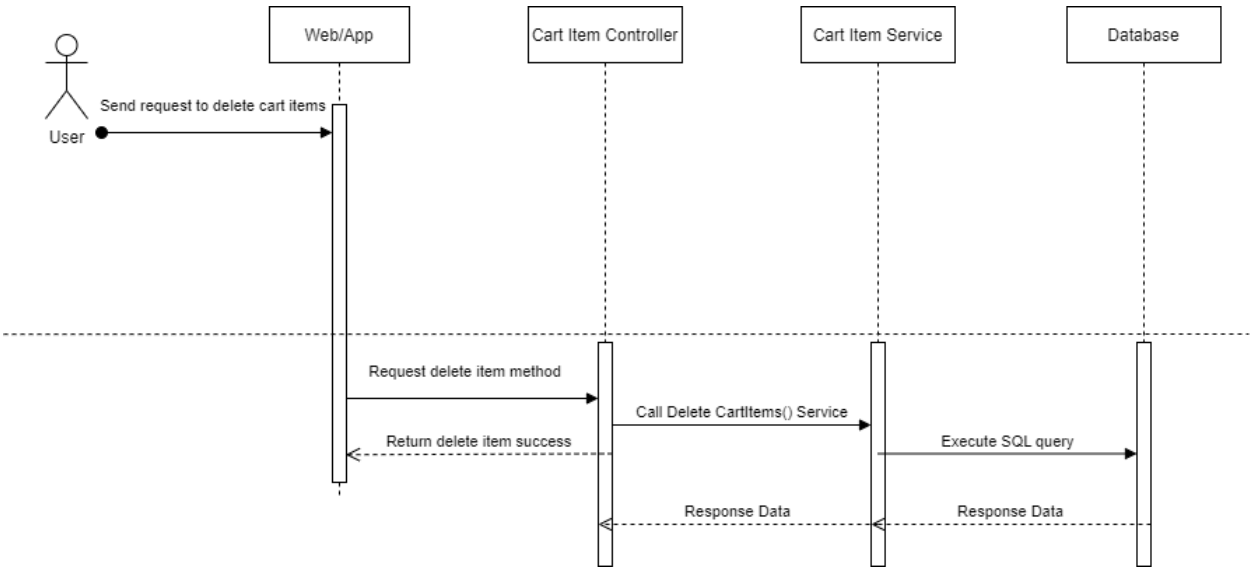


Figure 66: Delete Cart Items Diagram with User Role

3.3.2.13. Create order

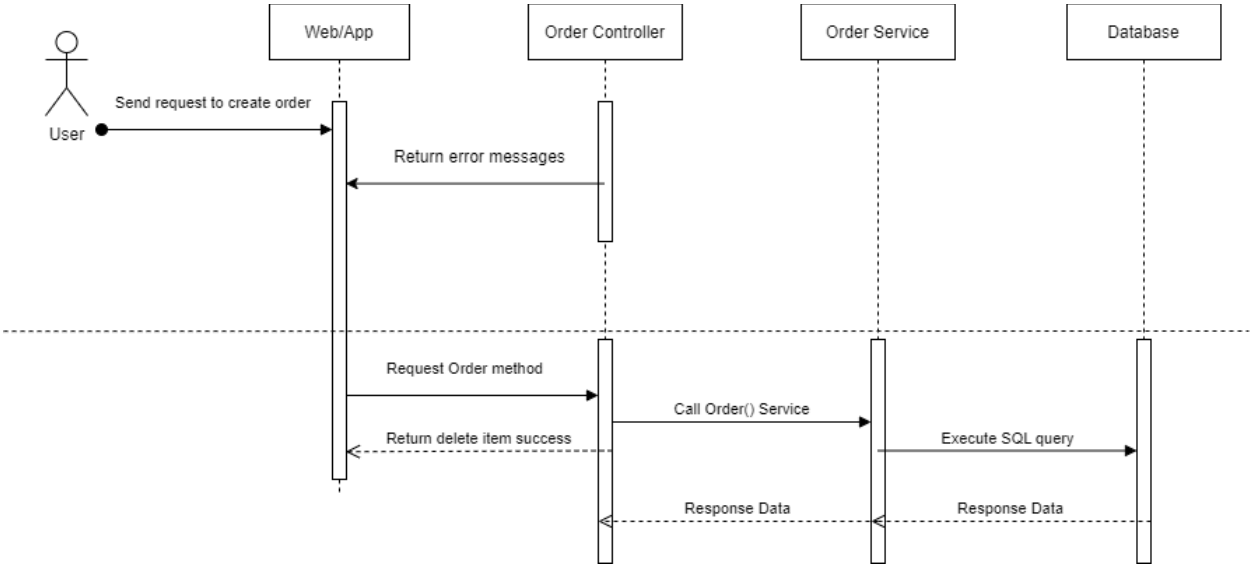


Figure 67: Create Order Diagram with User Role

3.3.2.14. Apply voucher

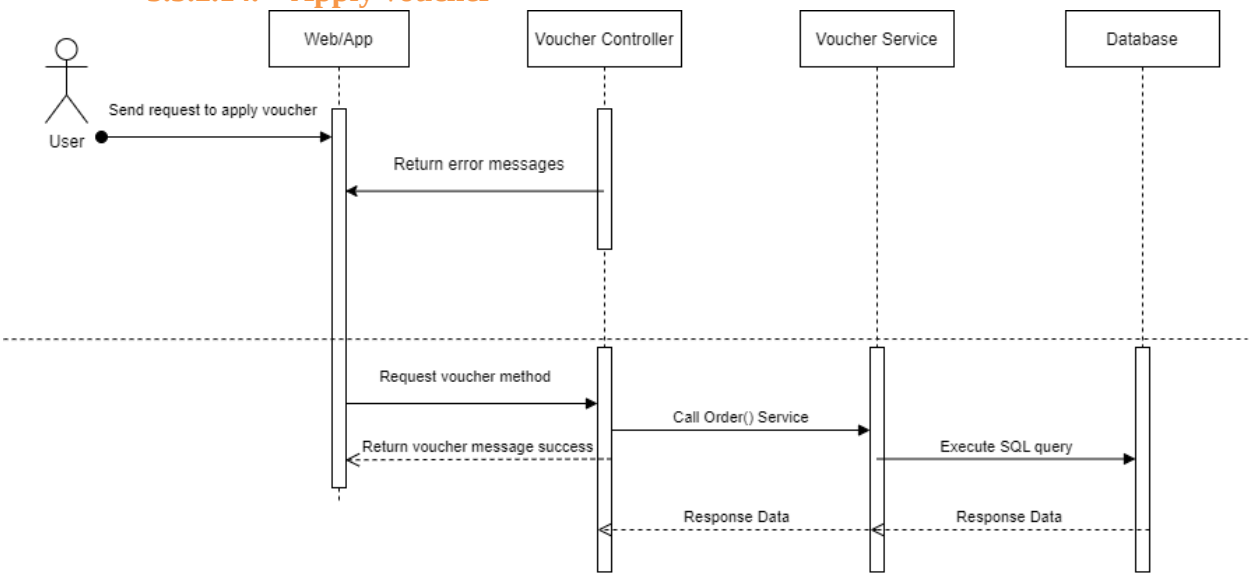


Figure 68: Apply Voucher Diagram with User Role

3.3.2.15. View orders

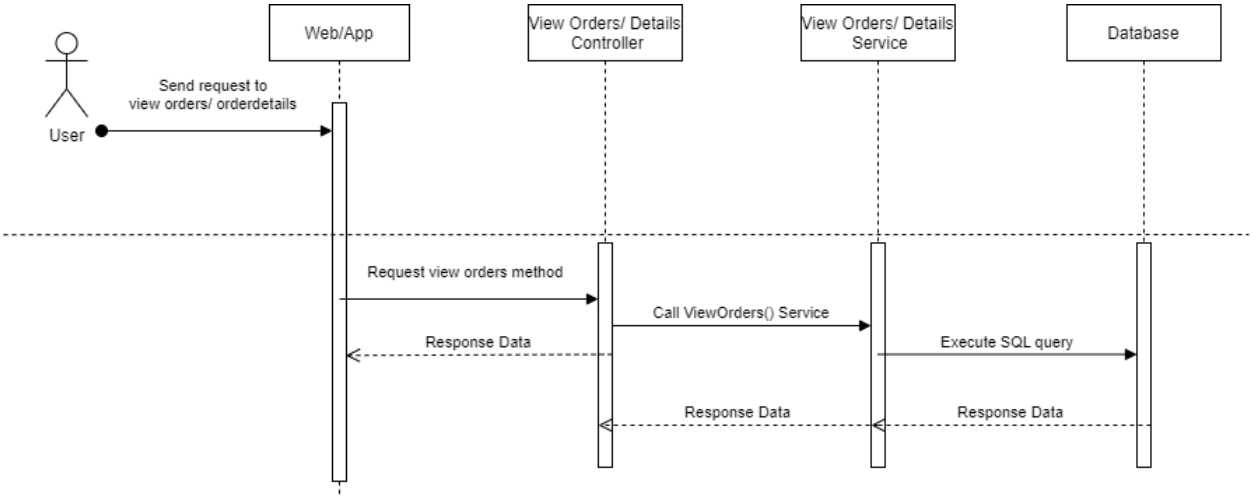


Figure 69: View Orders Diagram with User Role

3.3.2.16. View order details

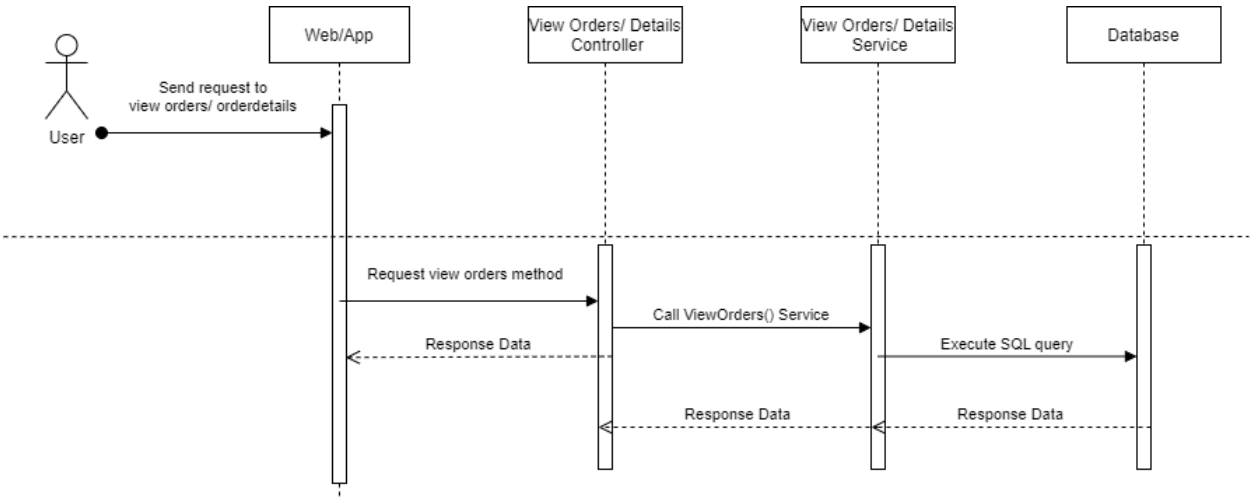


Figure 70: View Order Details Diagram with User Role

3.3.2.17. Cancel order

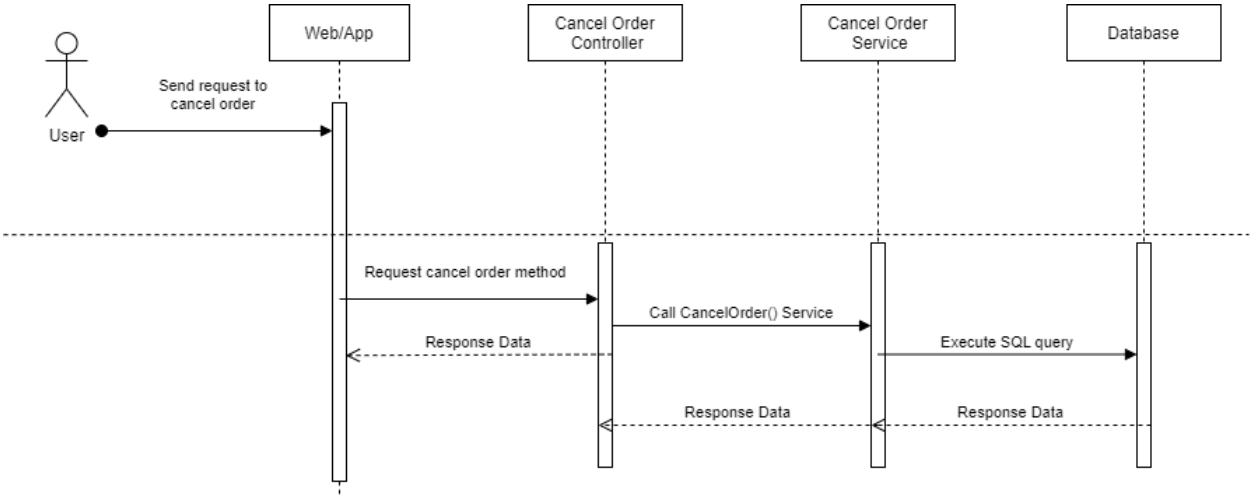


Figure 71: Cancel Order Diagram with User Role

3.3.2.18. Review product

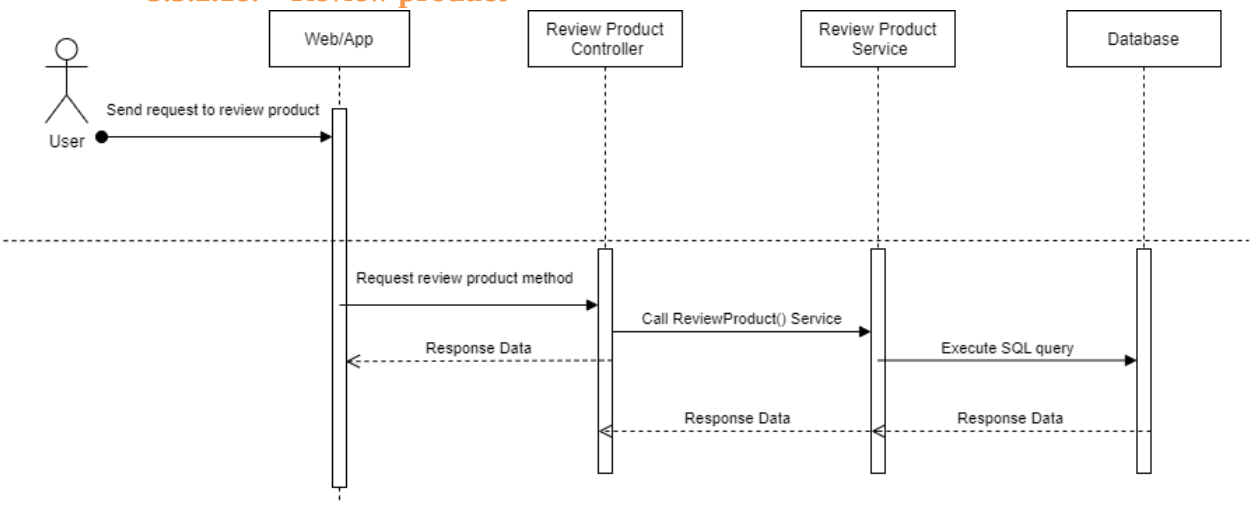


Figure 72: Review Product Diagram with User Role

3.3.2.19. View Addresses

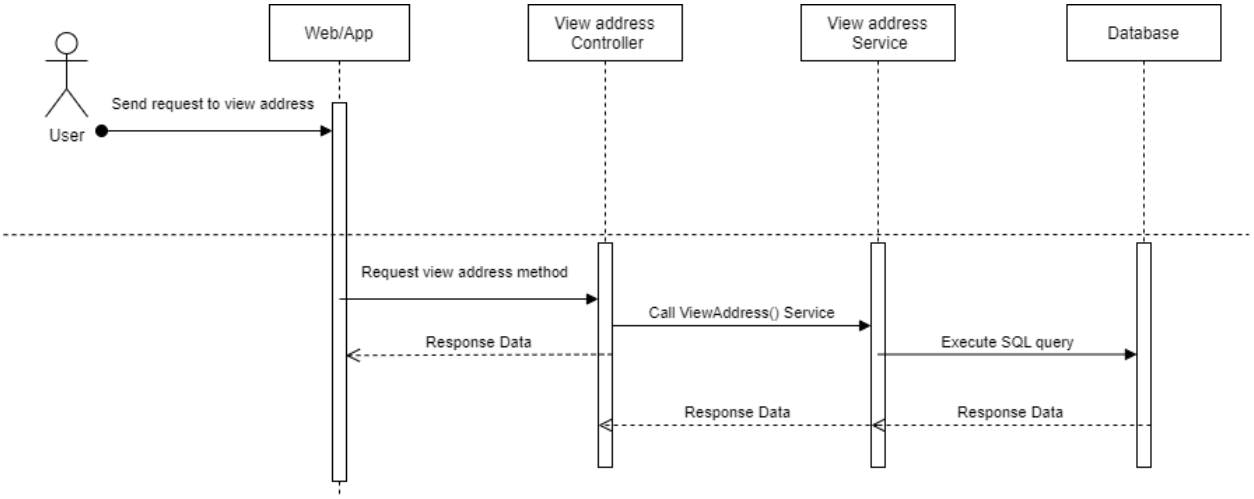


Figure 73: View Addresses Diagram with User Role

3.3.2.20. Create Address

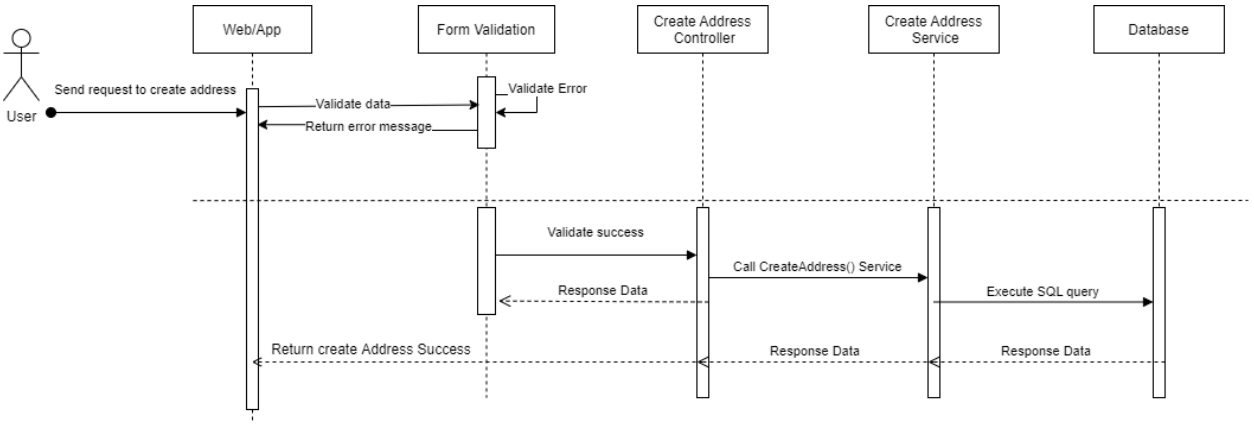


Figure 74: Create Address Diagram with User Role

3.3.2.21. Edit Address

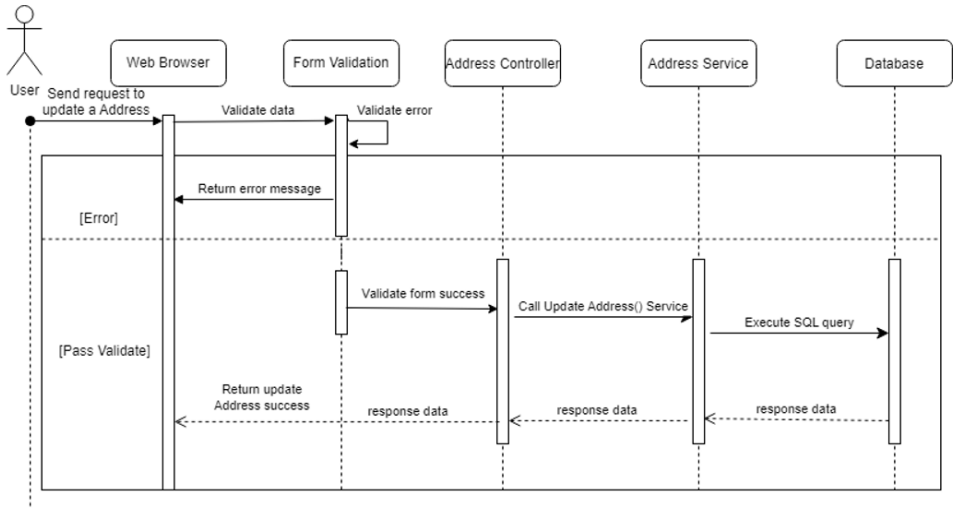


Figure 75: Edit Address Diagram with User Role

3.3.2.22. Delete Address

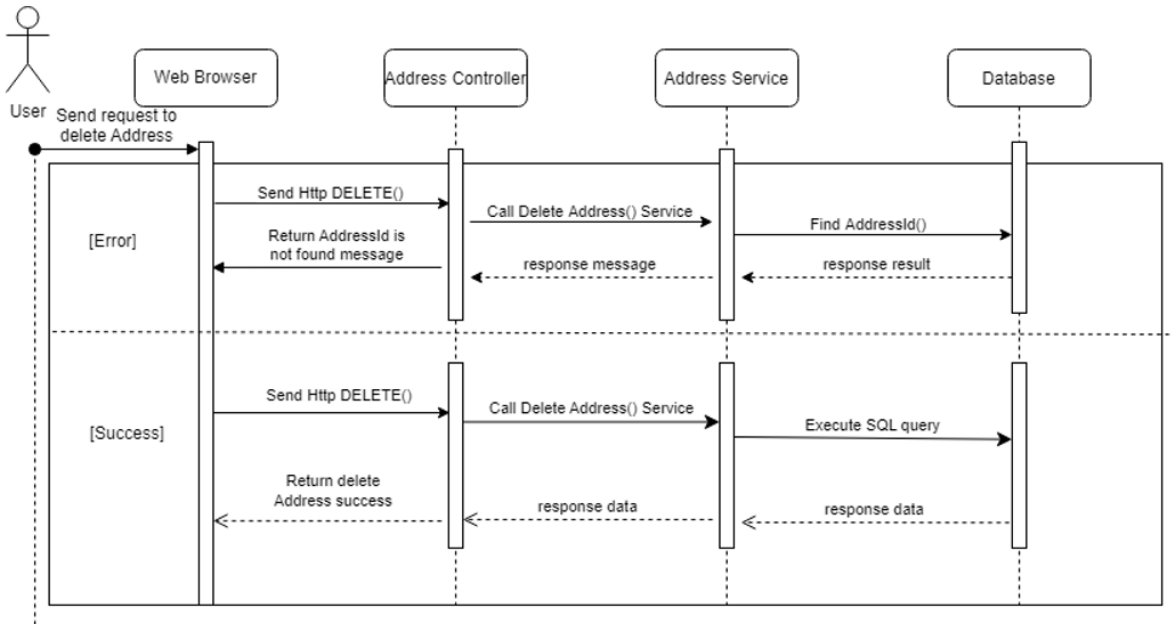


Figure 76: Delete Address Diagram with User Role

3.3.2.23. View Recommend Products

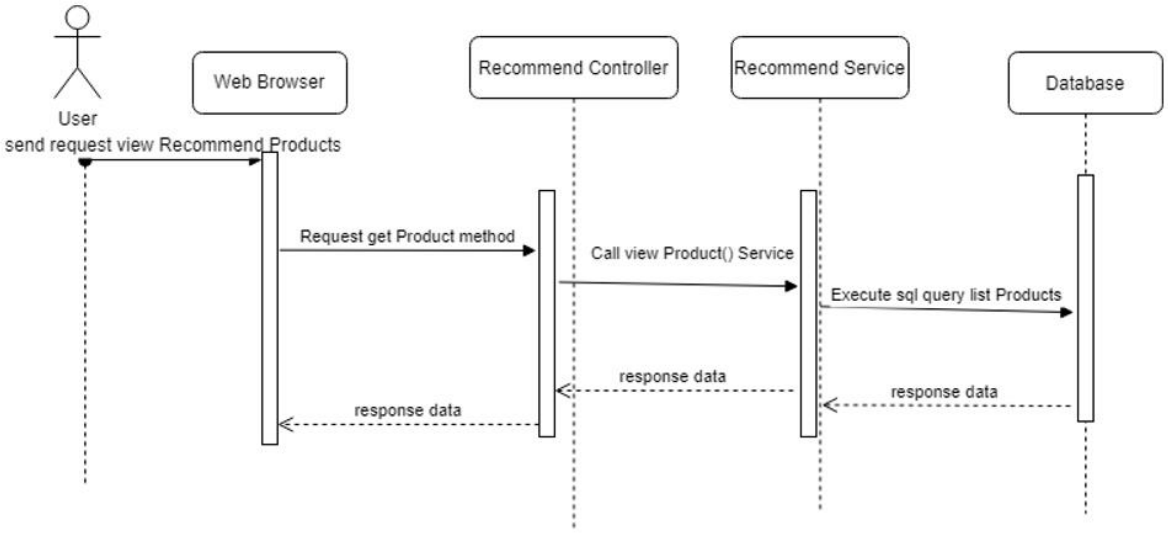


Figure 77: View Recommend Products Diagram with User Role