Fpt University

TRƯỜNG ĐẠI HỌC FPT

CAPSTONE PROJECT REPORT

Report 4 – Software Design Document

– DaNang, Oct 2022 –

Table of Contents

Table of Tables

Table of Figures

I. Record of Changes

Date	A*M, D	In charge	Change Description
09/10/2022	A*	DangHH	Create new
15/10/2022	M*	SangDT	Add class diagram
20/11/2022	M*	ThanhNPN	Add Sequence diagram

*A - Added M - Modified D - Deleted

II. Software Design Document

1. System Design

The project has successfully applied a data analysis method to make suggestions. Basically, these suggestions are based on location parameters (the city the customer lives in).

As for the client side, the recommendations online system has been developed in web applications. The web application used the framework NodeJS built on top of React which provided a short page load time and fast scan of the data taken from the user.

As for the server side, the recommendations online system have been developed with the ReactJS web framework, the Rest API and the database management system MySQL is a database management system that allows you to manage relational databases.

Figure 1:User interface

1.1 System Architecture

1.1.1 Client

Reactjs is an open-source Javascript library that helps build user interface components quickly and easily. Usually, programmers will embed javascript into HTML code through attributes like AngularJS but Reactjs works as a library that allows embedding HTML into javascript through JSX. You can easily nest HTML snippets into JSX making components easier to understand and use.

The basic components of React are called components. Syntax to write HTML using Javascript to render. You can create a component by calling the createClass method of the React object, the starting point when accessing this library. Multiple components can be nested through the return statement of the render method.

Benefits of using Reactjs

- ReactJS makes it easier to write Javascript code because it uses a special syntax that is the JSX syntax. Through JSX allows embedding HTML and Javascript code.
- ReactJS allows Developers to break complex UI structures into independent components. Devs won't have to worry about the overall web application, now it's easier for developers to break down complex UI/UX structures into simpler components.
- Included with ReactJS are many development tools that make debugging code easier.
- One of the more advantages of ReactJS is its SEO friendliness. Most of the JS Frameworks are not search-friendly, although much improved, but with the help of rendering data returned in the form of web pages, it helps to make SEO more standard.

Figure 2: Application Architecture

1.1.2 Server

1.1.2.1 Serverless

Serverless (also known as a serverless platform) is a platform that creates an environment that allows programmers to code applications or services without having to worry too much about server problems. Serverless application can be understood as a server that takes care of internal system operations such as allocation, system resource management, upgrade and security.

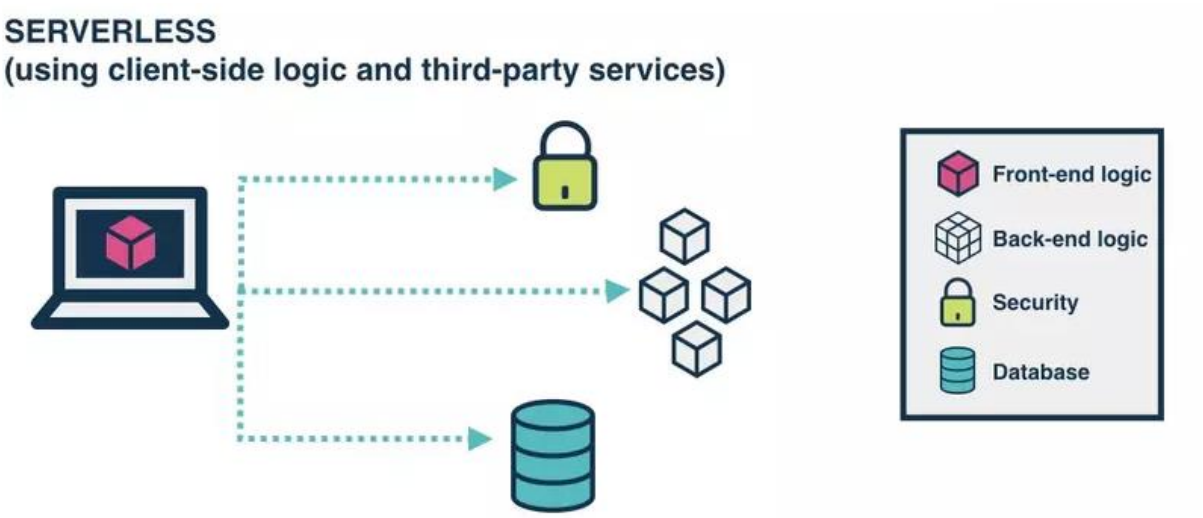

Figure 3: Serverless Architecture

1.1.2.1.1 Cognito

Amazon Cognito allows you to quickly and easily add registration, sign-in, and user access control to websites and mobile apps. Amazon Cognito scales to millions of users and supports logins through social identity providers such as Apple, Facebook, Google, and Amazon, as well as enterprise identity providers through SAML 2.0 and OpenID Connect.

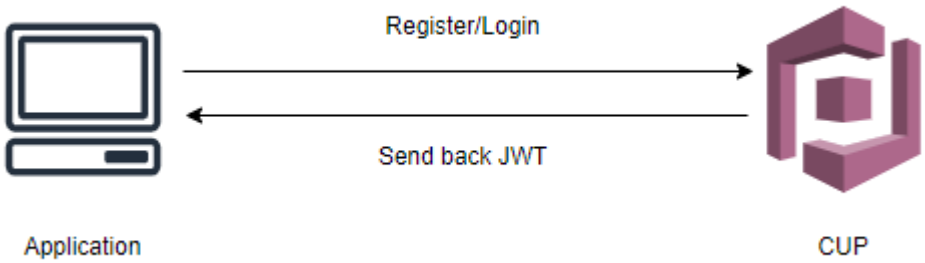

Figure 4: Cognito

1.1.2.1.2 SQS Service

Amazon Simple Queue Service (SQS) is a fully managed message queue service that allows you to unlink and scale your microservices, distributed systems, and serverless applications. SQS eliminates the complexity and indirect costs associated with managing and operating message-oriented middleware and allows developers to focus on other things. Using SQS, you can send, store, and receive messages between software components at any volume, without losing messages or forcing other services to be available at all times.

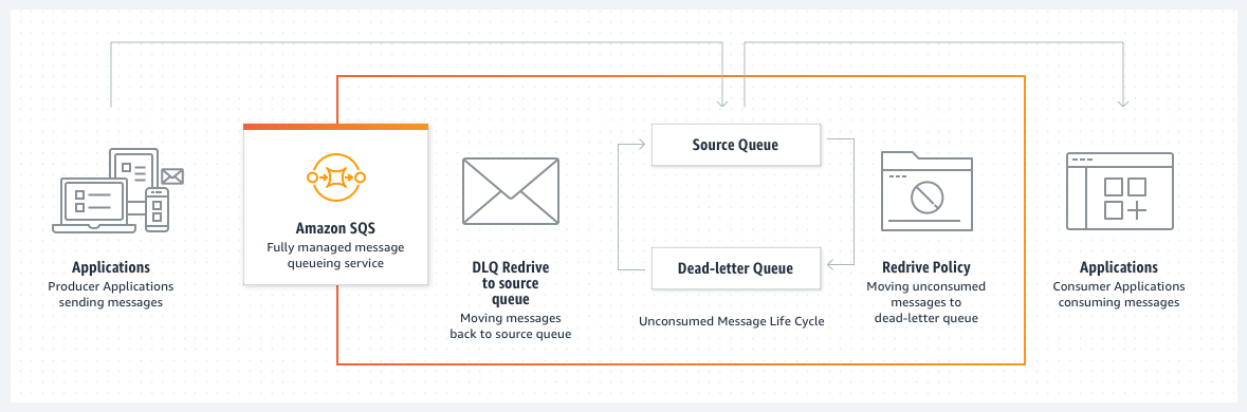

Figure 5: SQS Service

1.1.2.1.3 SES Service

Amazon Simple Email Service (SES) is a cost-effective, flexible, and scalable email service that allows developers to send email from within any application. You can quickly configure Amazon SES to support a number of email use cases including mass email communications, marketing, or transactions. Amazon SES' flexible IP deployment and email authentication options help drive greater deliverability and protect sender reputation, while delivery analytics measure the impact of individual emails. With Amazon SES, you can securely send email globally at scale.

Figure 6: SES Service

1.1.2.2 Lambda function

AWS Lambda is a computer service where you can upload your code, and the AWS Lambda service helps you run that code using available AWS resources. After you upload your code, and you create a Lambda function, AWS provision and manage the servers that you use to run the code. As for using AWS, you can do the following:

- An event-driven compute service where AWS Lambda runs your code and returns events, data changes pushed into an Amazon S3 bucket or an Amazon DynamoDB table. (you can learn more about Amazon DynamoDB)
- A computer service to run code and return HTTP requests using Amazon API Gateway or APIs using AWS SDKs.

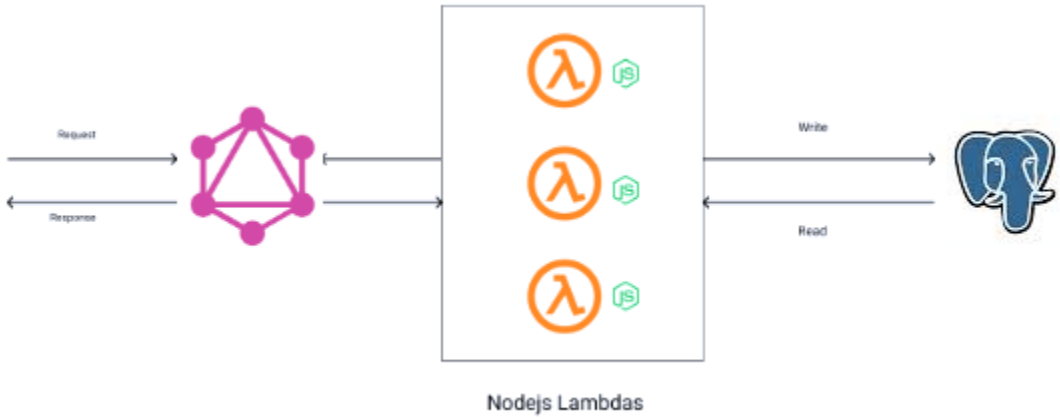

Figure 7: Lambda function

1.1.2.3 Type Graphql

GraphQL is a query language for APIs that provides a complete description of the data in your API, allowing the client side to request exactly the data it needs without being redundant or missing. In other words, it provides a common interface between the client and the server for retrieving and manipulating data.

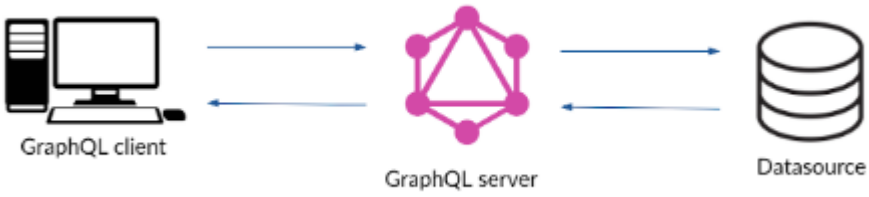

Figure 8: Type Graphql

1.1.2.4 Apollo server

Apollo Server is a GraphQL server implementation for JavaScript, specifically for the Node.js platform.It supports many popular Node.js frameworks, including:

- Express
- Hapi
- Koa
- Adjust

Apollo Server provides us with 3 basic things:

- Gives us a way to describe our data with schema.
- Provides a framework for resolvers, which are functions we write to fetch the data needed to make a request.
- Facilitate authentication processing for our API.

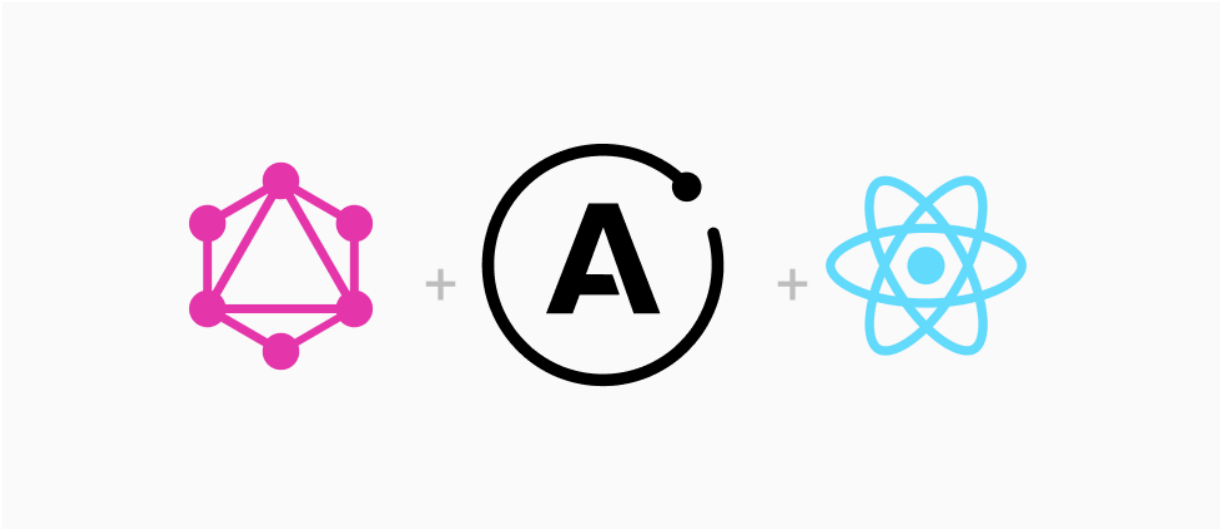

Figure 9: Apollo server

1.1.2.5 PostgreSQL

PostgreSQL is a general-purpose object-relational database management system, the most advanced open source database system available today.PostgreSQL is designed to run on UNIX-like platforms. However, PostgreSQL was also dynamically adapted to run on many different platforms such as Mac OS X, Solaris, and Windows.

Figure 10: MySQL

1.1.2.6 NodeJS & JavaScript

NodeJS is an open-source and cross-platform JavaScript runtime environment that is used to run web applications outside of the client's browser. This platform, developed by Ryan Dahl in 2009, is considered a perfect solution for data-intensive applications thanks to its asynchronous event-driven model.

Figure 11: NodeJS

JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved.

Figure 12: JavaScript

1.1.3 Recommendation System

1.1.3.1 Overview of recommendation system

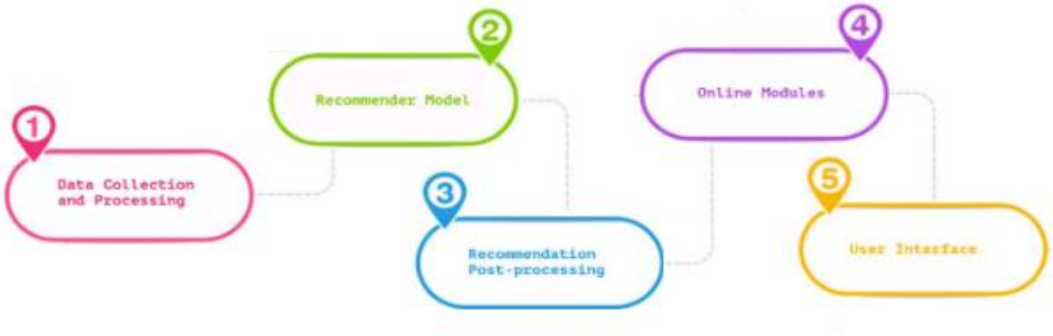

Figure 13: Flow of recommendation system

Health and fitness is the area that users are most concerned and worried about, so we build a recommendation system based on each user's desire and self-improvement criteria through a survey, because so we used a Collaborative Filtering System (Item-Item Filtering) for the suggestion and it had to respond to the following location parameters.

1.1.3.2 The setting up steps of the recommendation System

The suggestion system provides value to customers by understanding individual user behavior and then recommending foods & exercises that they may find appropriate. First, the system collects data from the survey, from which the system calculates the BMI data and sends to the customer the most suitable suggestions

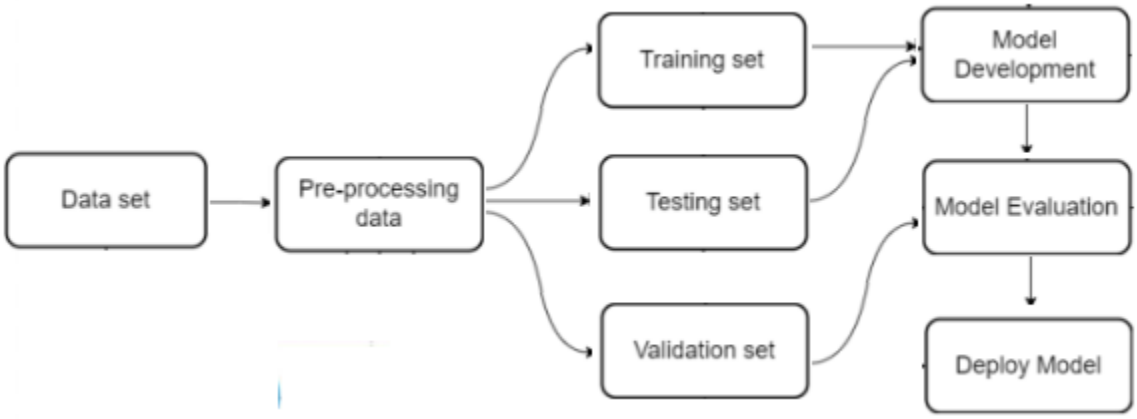

Figure 14: Flow of building model

1.1.4 System Architecture

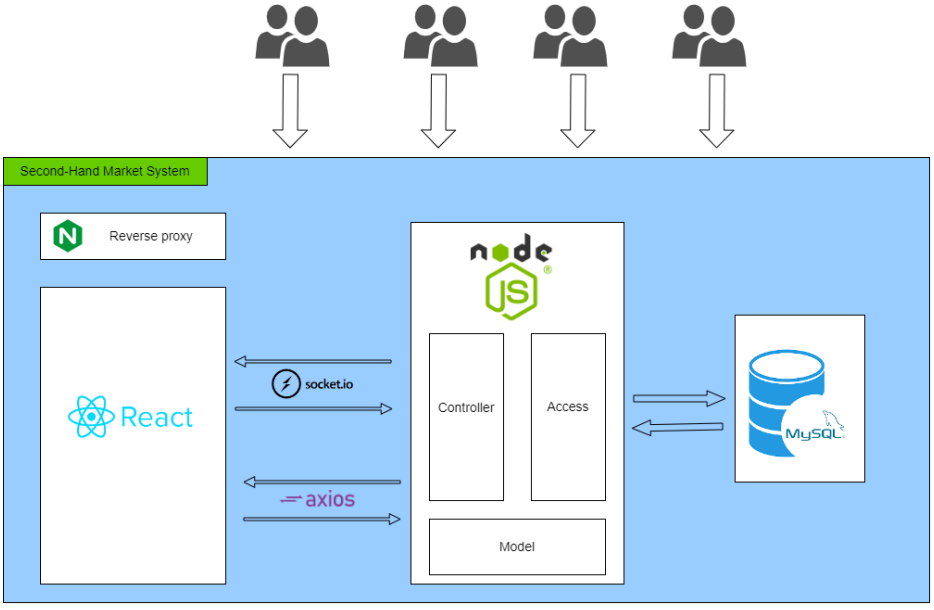

Figure 15: System Architecture

1.2 Package Diagram

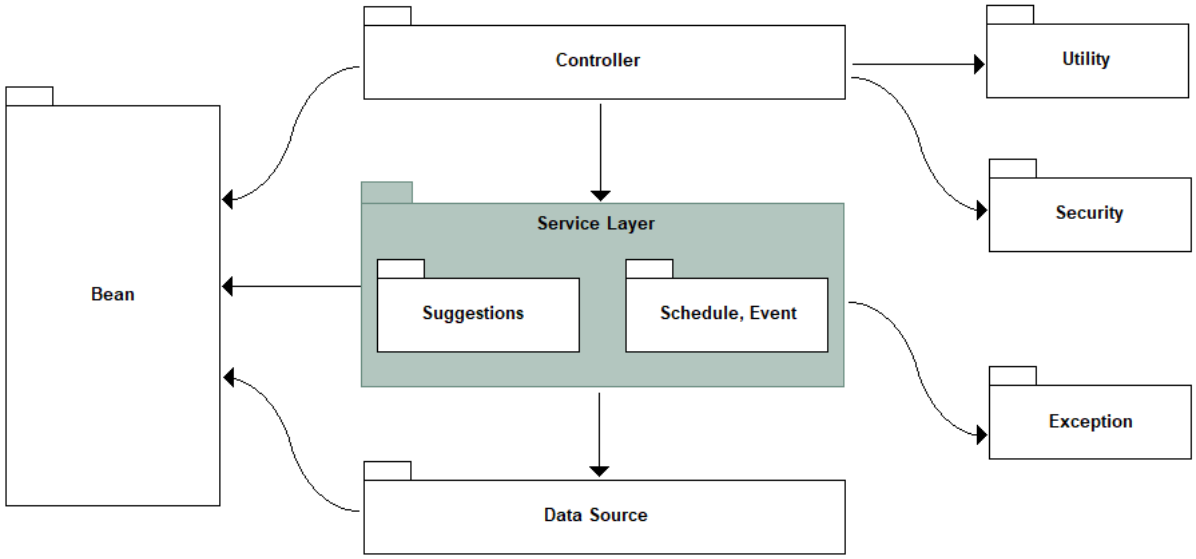

Figure 16: Package Diagram

- **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	Suggestions	Contains suggestion classes that are used to write business logic in another layer, separate from the controller
03	Schedule, Event	Contains schedule, event classes that are used to write business logic in another layer, separate from the controller
04	Utility	Contains entity classes which are the persistence objects stores as a record in the database
05	Security	Implements authentication, authorization, and protection against common attacks
06	Exception	Provides a mechanism to treat exceptions that are thrown during execution of handlers
07	Data Source	A factory for connections to the PostgreSQL database
08	Bean	Exploits the Inversion of Control feature by which an object defines its dependencies without creating them

2. Database Design

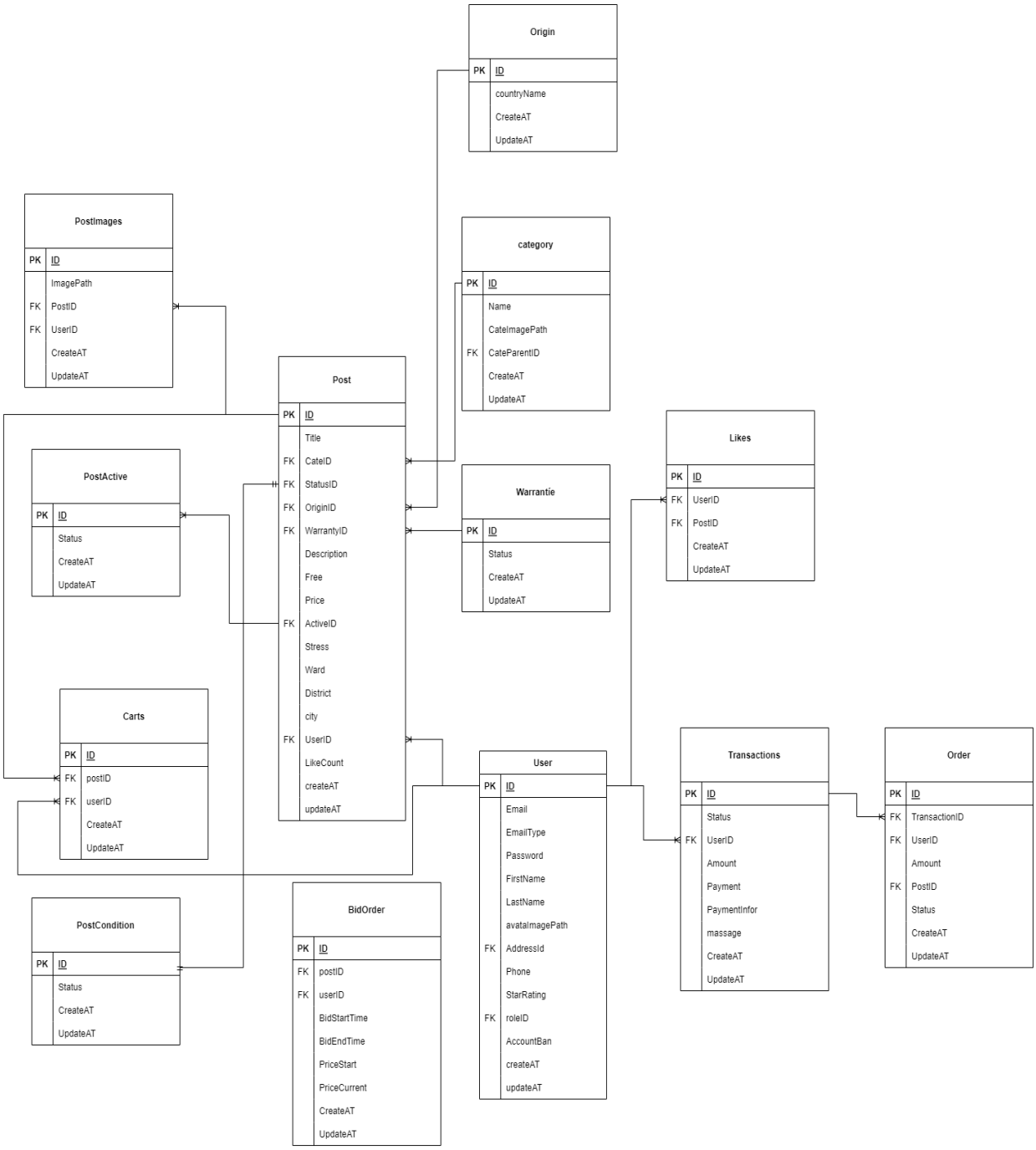

Figure 17: Database Design

- **Table Descriptions**

No	Table	Description
01	Users	- **Primary keys:** userid - **Foreign keys:** None
02	Posts	- **Primary keys:** id - **Foreign keys:** cateId,statusId,warrantyId,originId,activeId,userId
03	Origins	- **Primary keys**: id - **Foreign keys:** none
04	Postmages	- **Primary keys:** id - **Foreign keys:** postId,userId
05	Categories	- **Primary keys:** id - **Foreign keys:** cateParentId
06	Likes	- **Primary keys:** id - **Foreign keys:** postId,userId
07	PostCondition	- **Primary keys:** id - **Foreign keys:** none
08	PostActive	- **Primary keys:** id - **Foreign keys:** None
09	Waranties	- **Primary keys**: id - **Foreign keys:** None
10	Carts	- **Primary keys:** id - **Foreign keys:** postId,userId
11	Transactions	- **Primary keys**: id - **Foreign keys:** userid
12	Order	- **Primary keys**: id - **Foreign keys:** postId,transactionId
13	BidOrder	- **Primary keys**: id - **Foreign keys:** postId,userId

Table 1: Table Descriptions

3. Detailed Design

3.1 Class Diagram

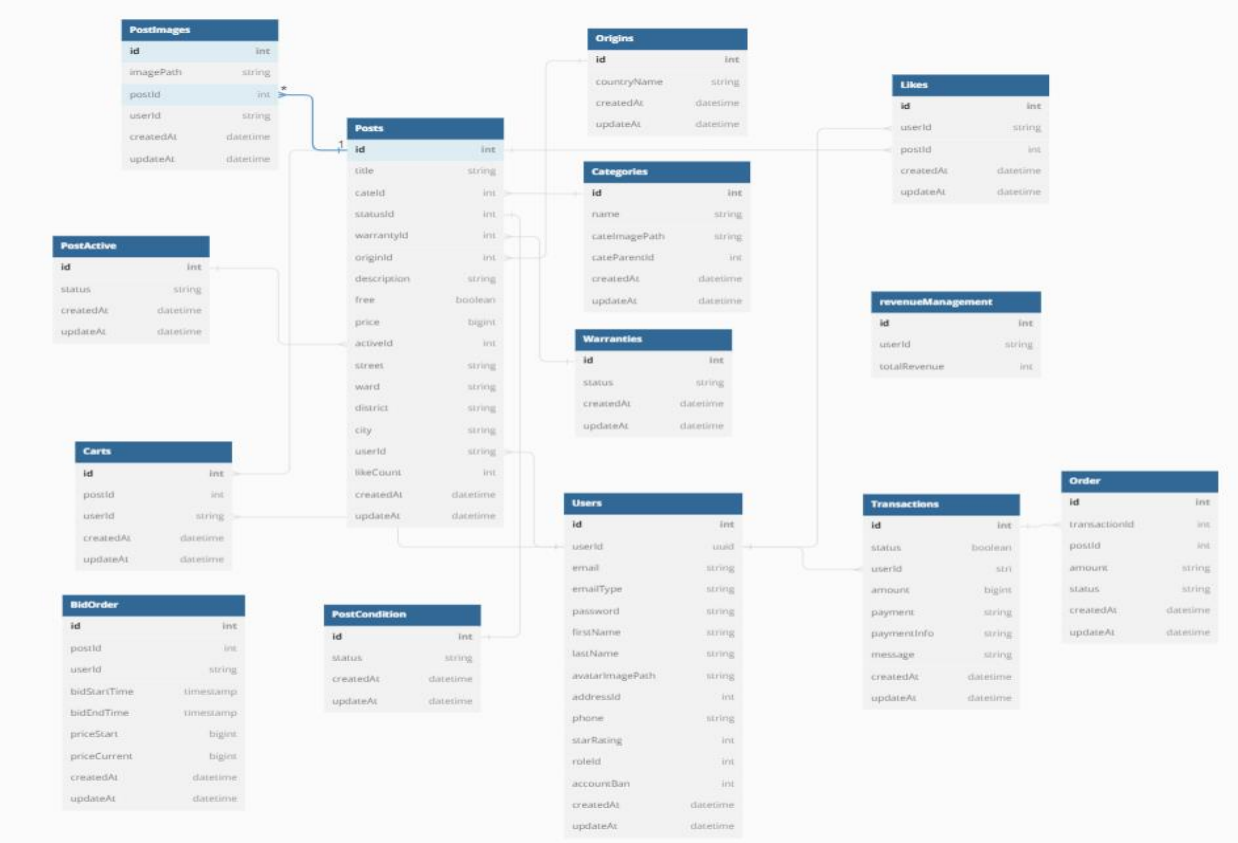

Figure 18: Class Diagrams

3.2 Class Specifications

3.2.1 Users

No	Attributes	Type	Description
01	id	int	Unique identifier, auto increment
02	UserID	uuid	Unique identifier by userlist ,auto increment
03	email	string	email of user
04	emailType	string	type of user email
05	Password	string	password
06	FirstName	string	User Firstname
07	LastName	string	User Lastname
08	avataImagePath	string	link to avata image
09	addressID	int	address of user
10	phone	String	phone number of user
11	starrating	int	average starpoint of user
12	roleid	int	role of user
13	accountBan	int	number for ban/unban user
14	createAT	datetime	create time
15	updateAT	datetime	update time

Table 2: Users

3.2.2 Posts

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	title	String	Title of product
03	CateID	int	Unique identifier by category, auto increment
04	StatusId	int	status of product
05	warrantyId	int	status warranty of product
06	originId	int	origin of product
07	description	string	description
08	free	boolean	free or no
09	price	bigint	price of product
10	activeId	int	
11	street	string	address of user who post the product
12	ward	string	address of user who post the product
13	district	string	address of user who post the product
14	city	string	address of user who post the product
15	userId	int	id of user post
16	likeCount	int	total likes of post
17	createdAt	datetime	time create post
18	updateAt	datetime	time update post

Table 3: Posts

3.2.3 Origins

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	countryName	string	country of manufacture of the product
03	RoleID	int	Unique identifier by role, auto increment
04	createdAt	datetime	time create post
05	updateAt	datetime	time update post

Table 4: Origins

3.2.4 PostImages

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	imagePath	int	link to image
03	postId	int	Unique identifier by post list
04	userId	id	Unique identifier by user list
05	createdAt	datetime	time create post
06	updateAt	datetime	time update post

Table 5: PostImages

3.2.5 Categories

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	Name	string	Name of category
03	cateImagePath	string	path of image category
04	cateParentId	int	Unique identifier by post list
05	createdAt	datetime	time create post
06	updateAt	datetime	time update post

Table 6: Categories

3.2.6 Likes

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	userId	string	Unique identifier by user list
03	postId	int	Unique identifier by post list
05	createdAt	datetime	time create post
06	updateAt	datetime	time update post

Table 7: Likes

3.2.7 PostCondition

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	Status	int	Status condition of product
03	createdAt	datetime	time create post
04	updateAt	datetime	time update post

Table 8: PostCondition

3.2.8 PostActive

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	Status	int	Status active of product
03	createdAt	datetime	time create post
04	updateAt	datetime	time update post

Table 9: PostActive

3.2.9 Warranties

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	Status	int	Status warranties of product
03	createdAt	datetime	time create post
04	updateAt	datetime	time update post

Table 10: Warranties

3.2.10 Carts

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	userId	string	Unique identifier by user list
03	postId	int	Unique identifier by post list
05	createdAt	datetime	time create post
06	updateAt	datetime	time update post

Table 11: Carts

3.2.11 Transactions

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	status	boolean	Status Transaction of product
03	userId	string	Unique identifier by user list
04	amount	bigint	amount product transaction
05	payment	string	Status of payment
06	paymentInfo	string	information of payment
07	message	string	message of user for product
08	createdAt	datetime	time create post
09	updateAt	datetime	time update post

Table 12: Transactions

3.2.12 Order

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	transactionId	int	Unique identifier by transaction
03	postId	int	Unique identifier by post
04	amount	string	amount product order
05	status	string	Status order of product
08	createdAt	datetime	time create post
09	updateAt	datetime	time update post

Table 13: Order

3.2.13 BidOrder

No	Attributes	Type	Description
01	ID	int	Unique identifier, auto increment
02	userId	string	Unique identifier by user list
03	postId	int	Unique identifier by post list
04	bidStartTime	timestamp	time start auctions
05	bidEndTime	timestamp	time end auctions
06	priceStart	bigint	price start auctions
07	priceCurrent	bigint	price current auctions
08	createdAt	datetime	time create post
09	updateAt	datetime	time update post

Table 14: BidOrder

3.3 Sequence Diagrams

3.3.1 Guest

3.3.1.1 Register

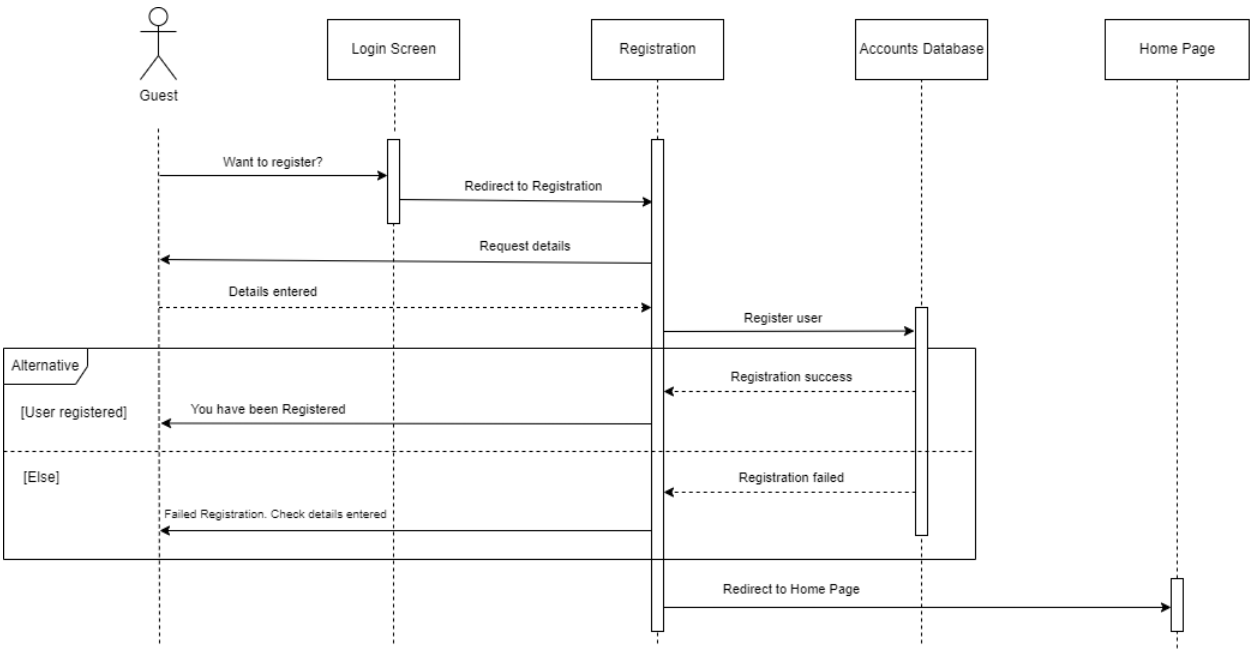

Figure 19: Guest Register

3.3.1.2 Search product by name

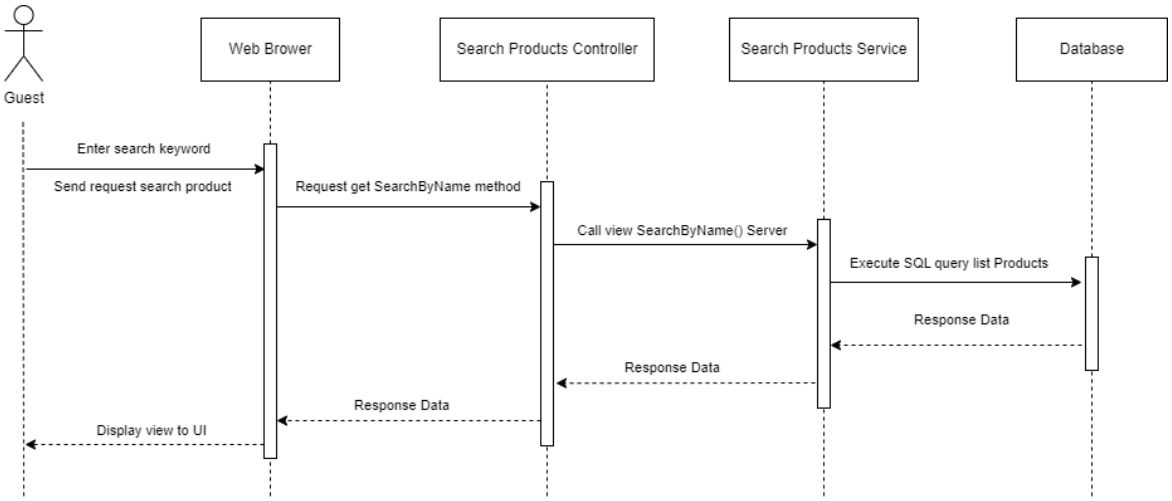

Figure 20: Guest Search product by name

3.3.1.3 View product

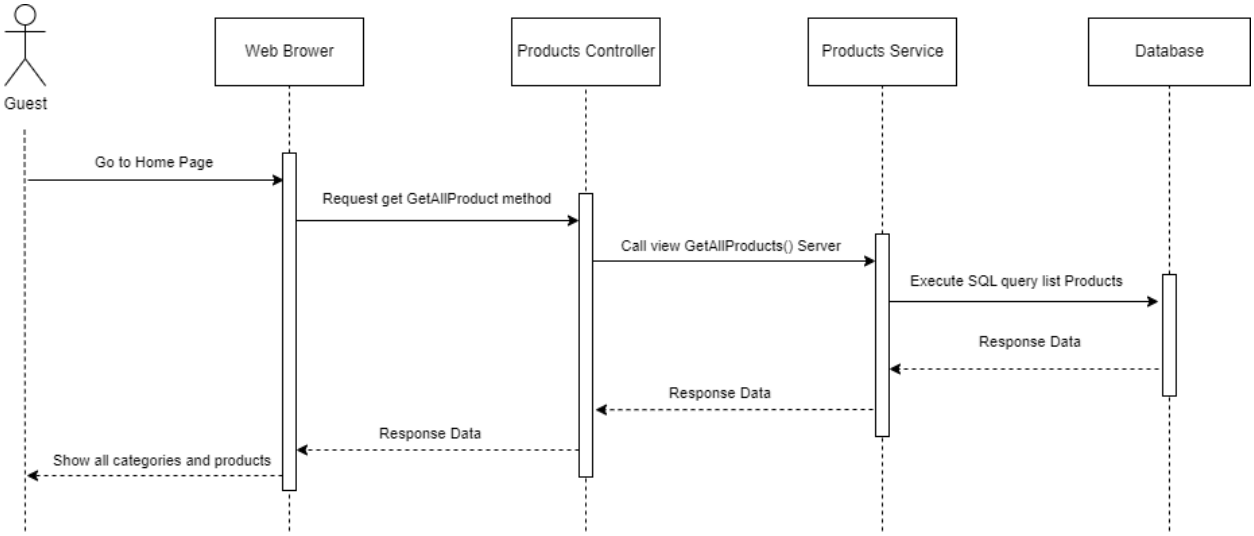

Figure 21: Guest View product

3.3.2 User

3.3.2.1 Login/Register

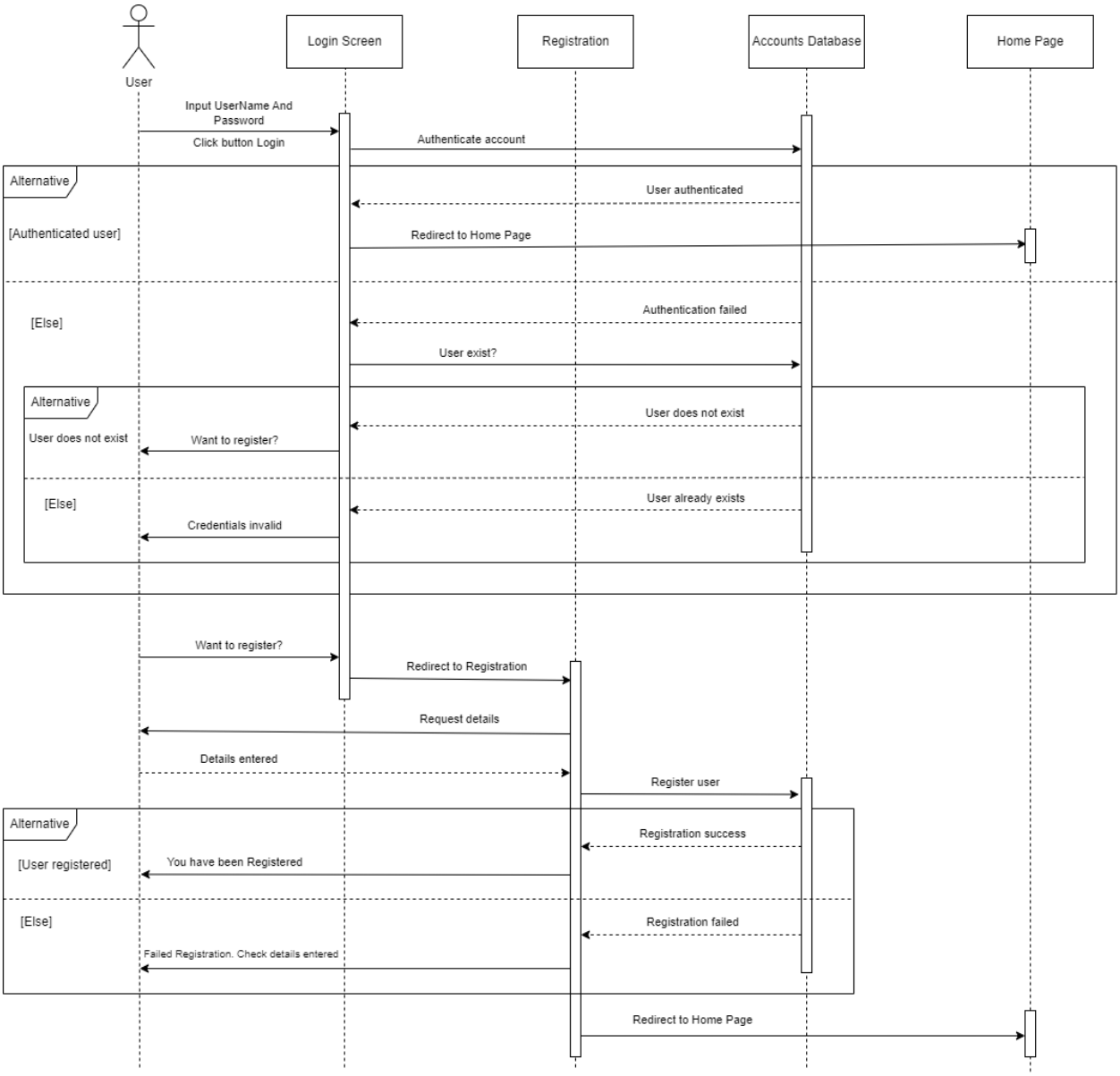

Figure 22: User Login/Register

3.3.2.2 Forgot password

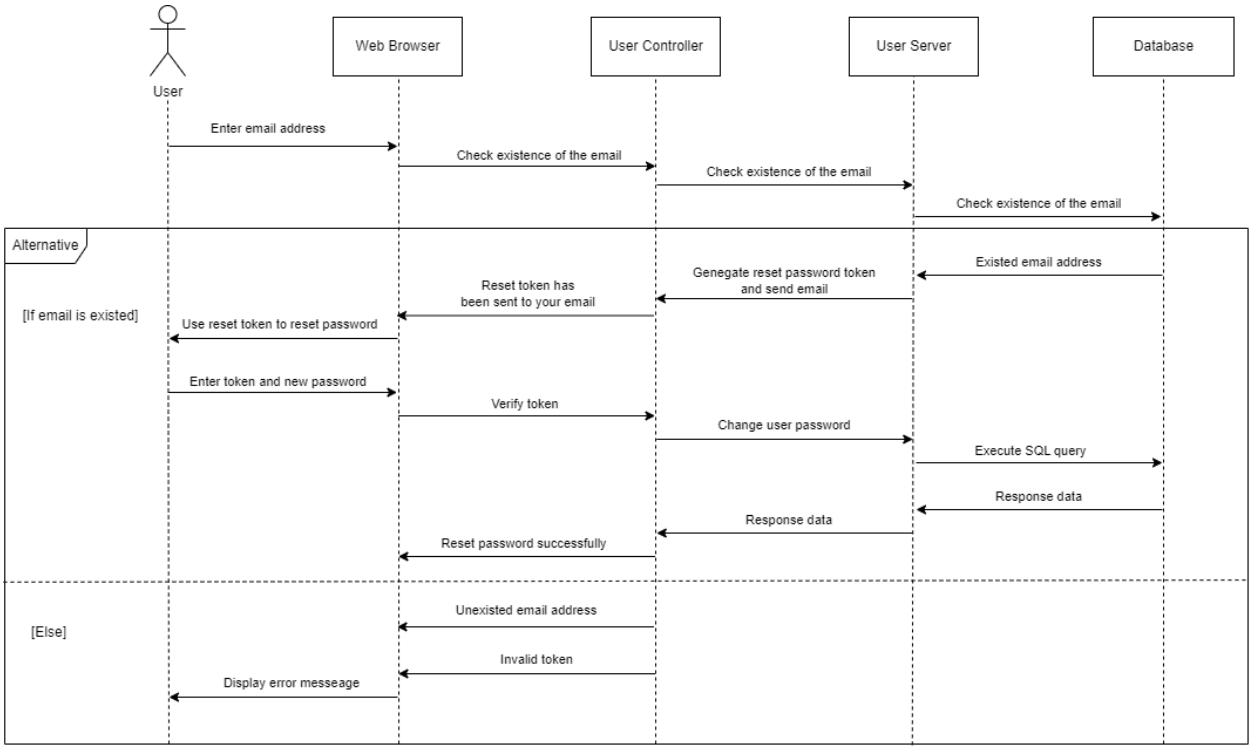

Figure 23: User Forgot password

3.3.2.3 Logout

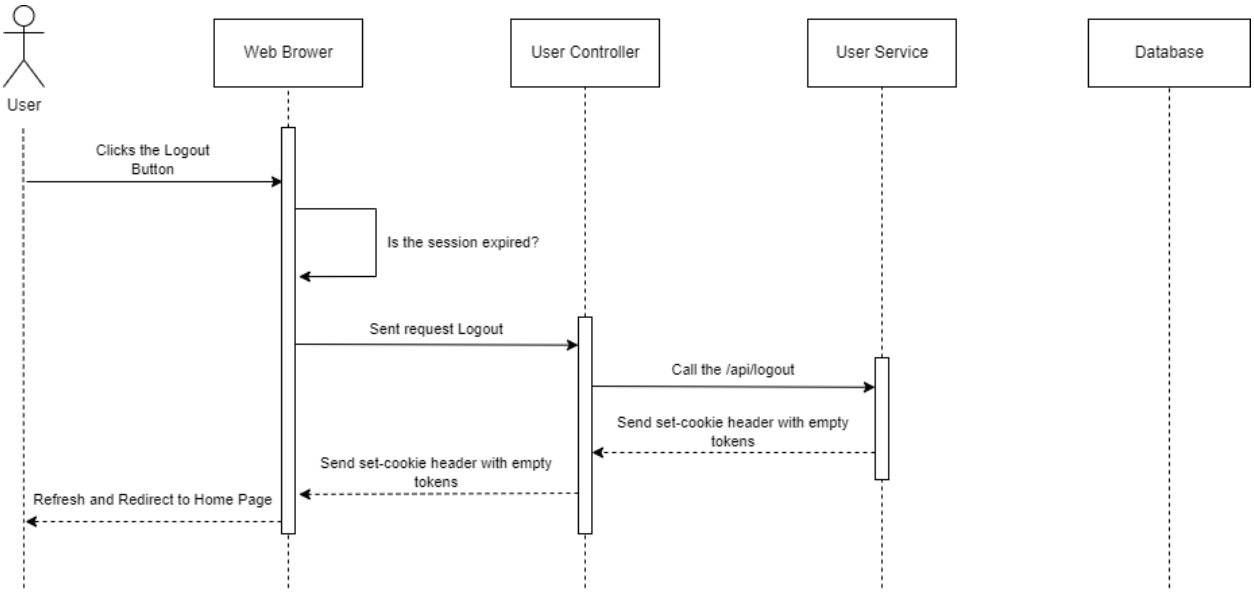

Figure 24: User Logout

3.3.2.4 View product

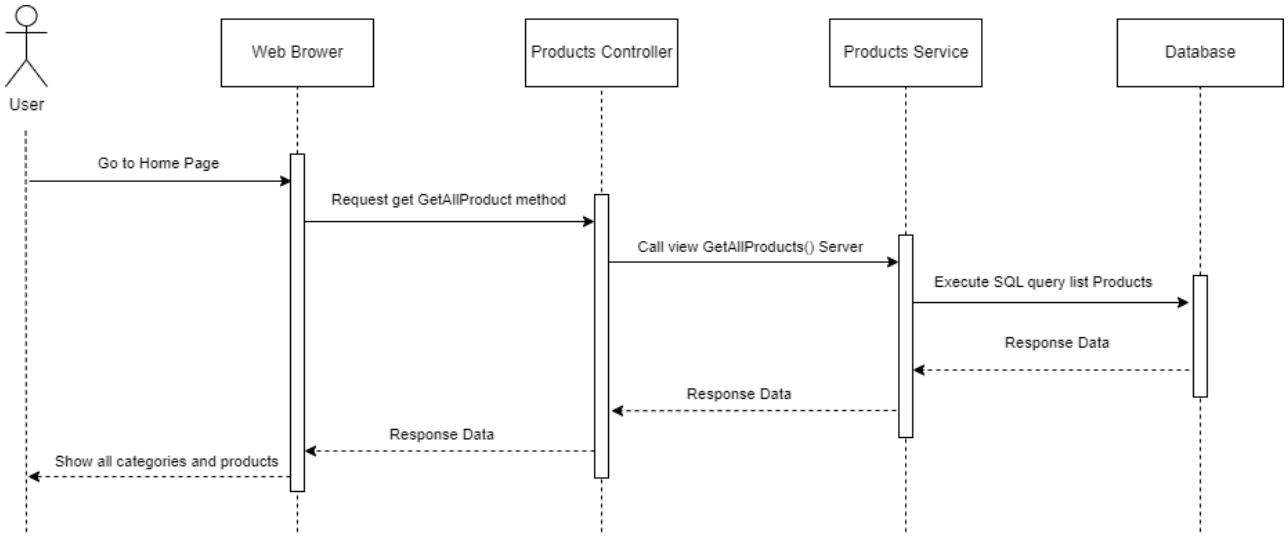

Figure 25: User View product

3.3.2.5 Search product by name

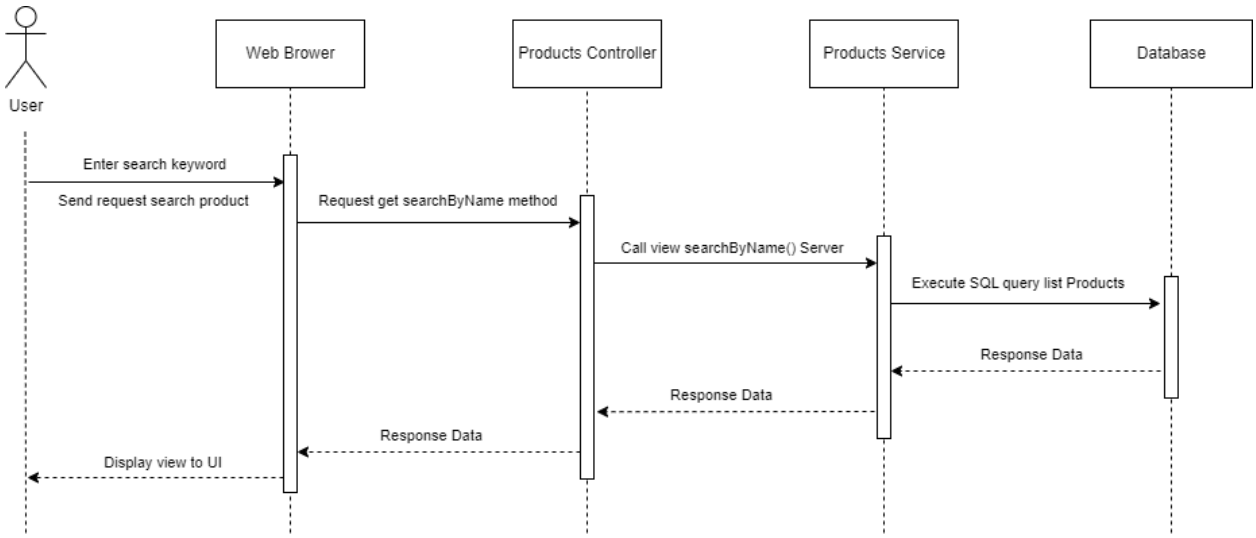

Figure 26: User Search product by name

3.3.2.6 Manage product to sell

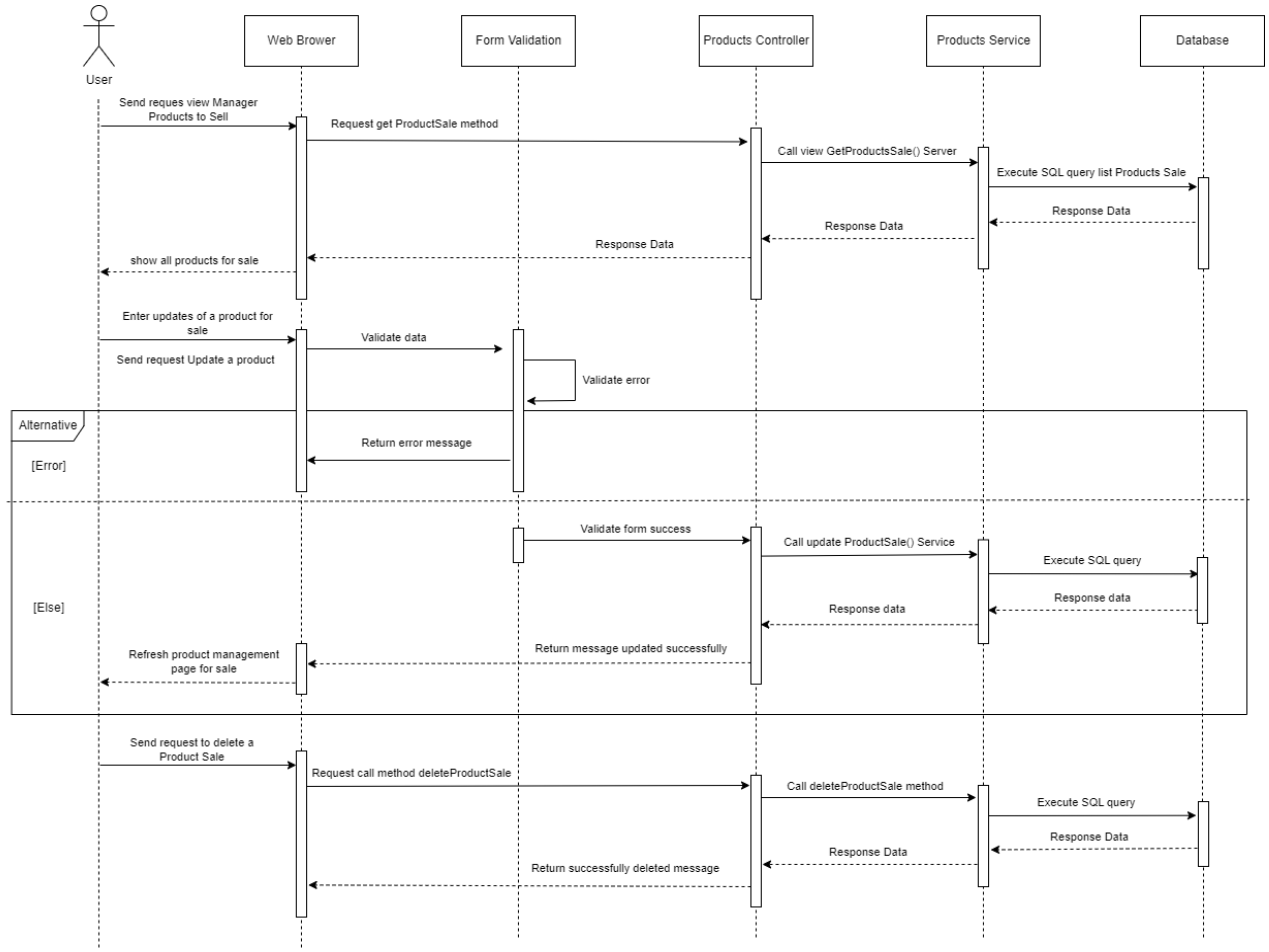

Figure 27: User Manage product to sell

3.3.2.7 Post product to sell

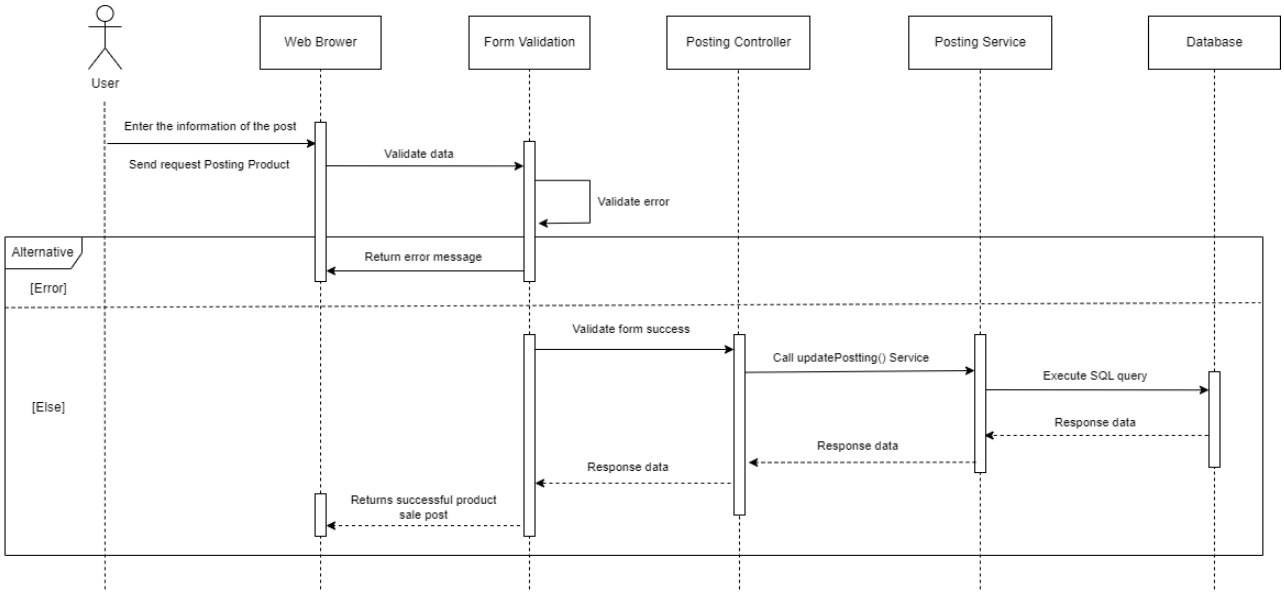

Figure 28: User Post product to sell

3.3.2.8 Create auction

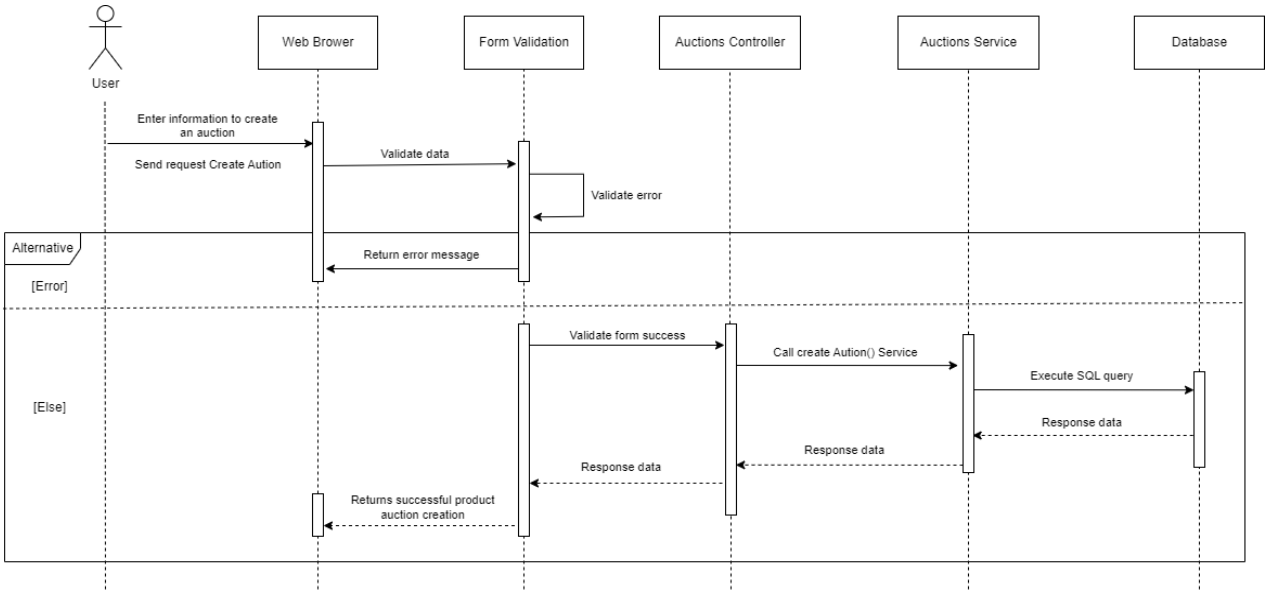

Figure 29: User Create auction

3.3.2.9 Manage auction product

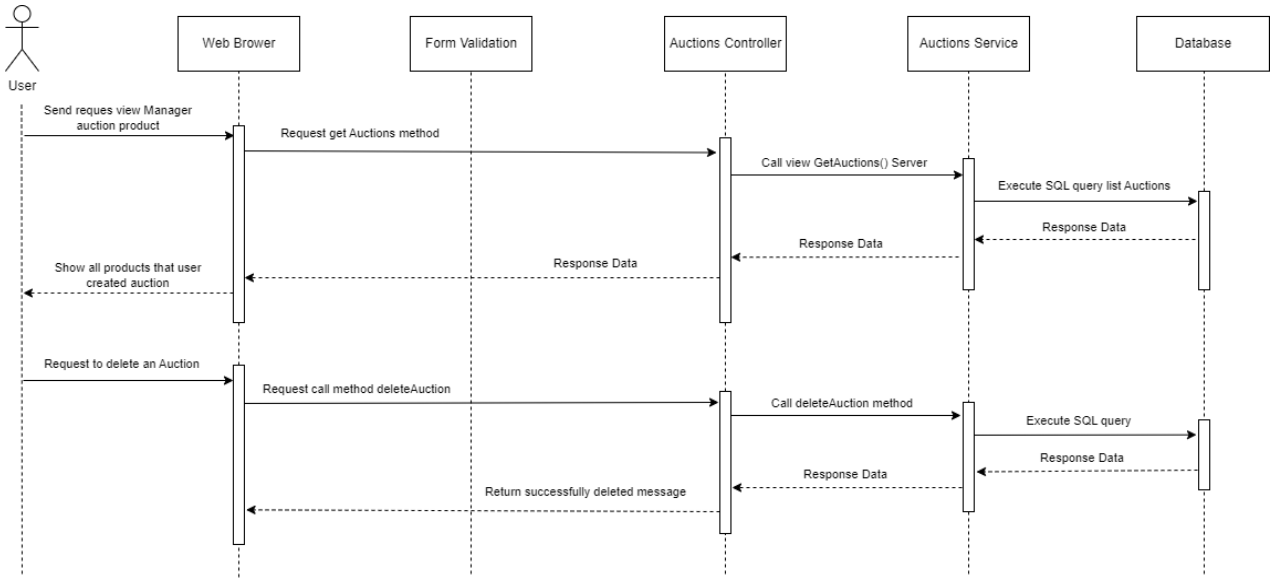

Figure 30: User Manage auction product

3.3.2.10 View other user's account information

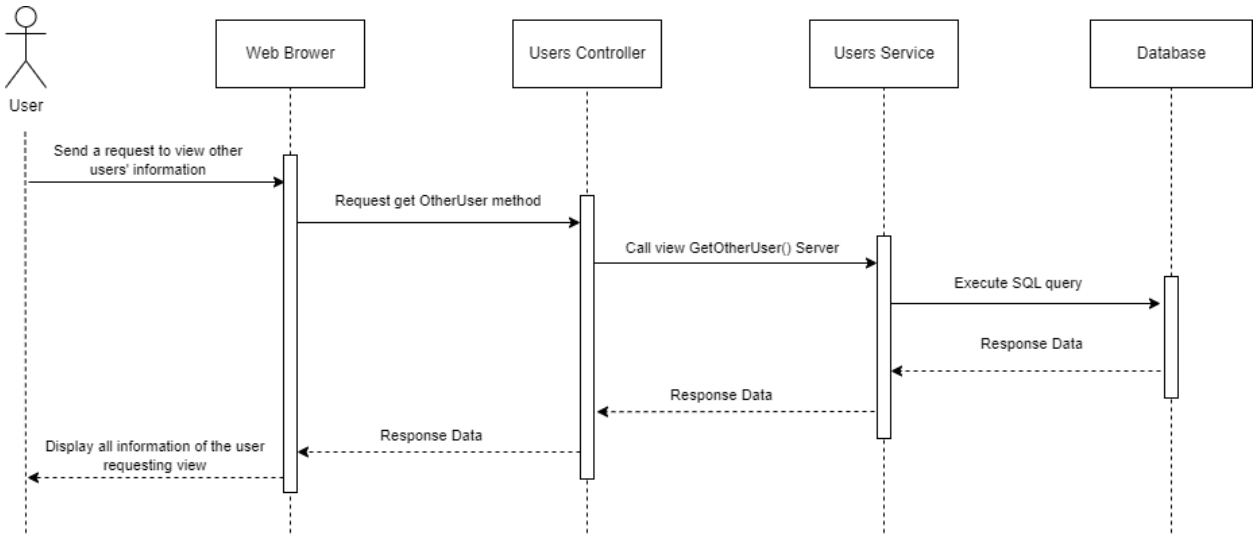

Figure 31: User View other user's account infomation

3.3.2.11 Manage order

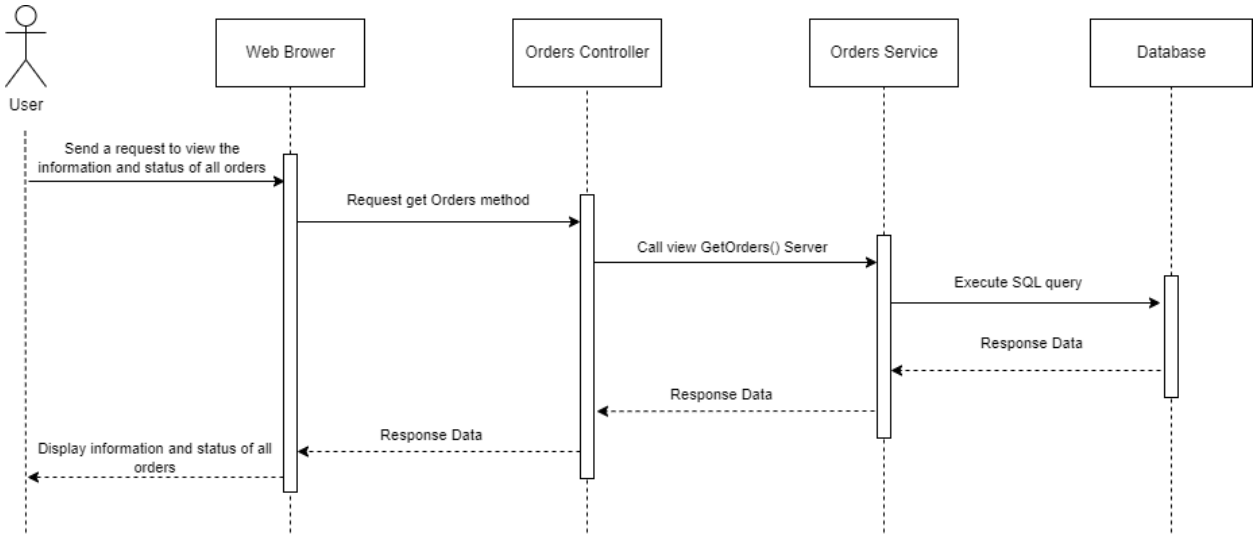

Figure 32: User Manage order

3.3.2.12 Buy product

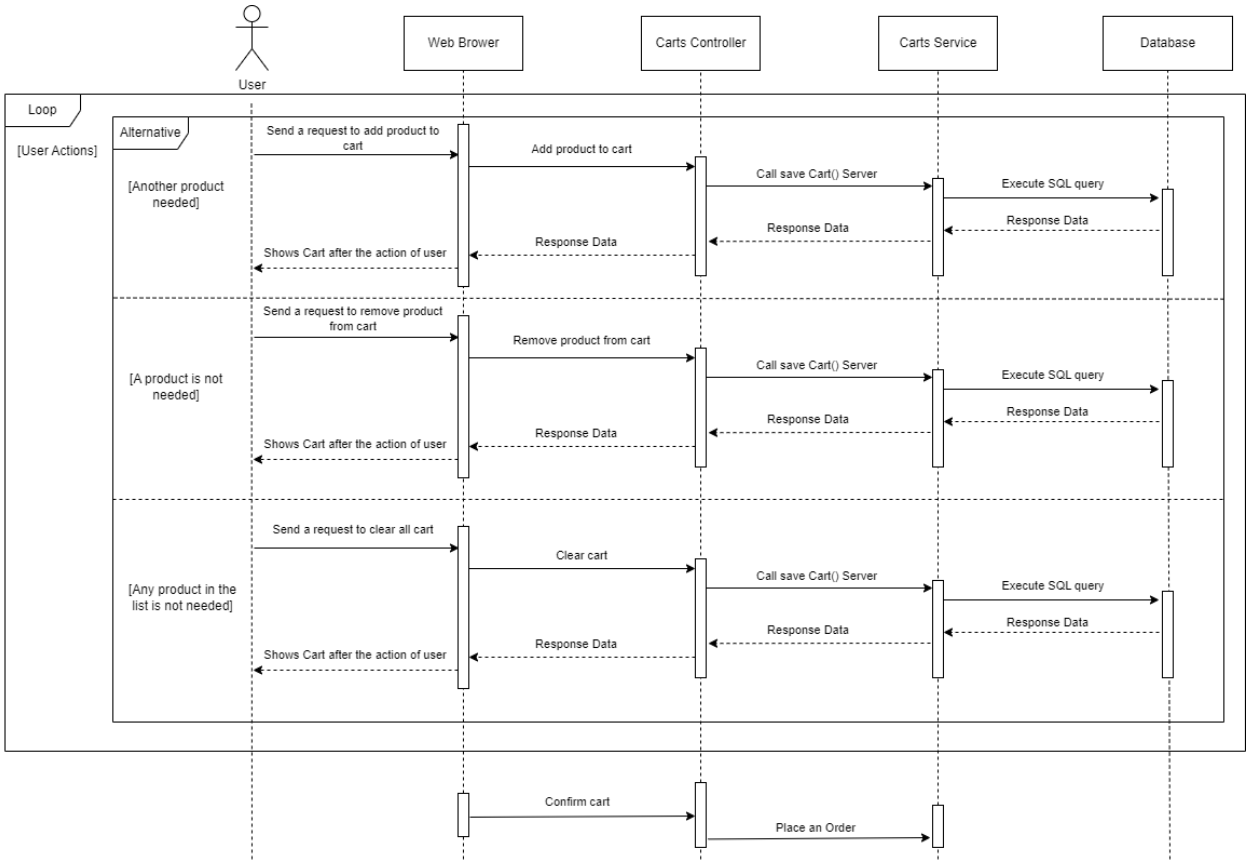

Figure 33: User Buy product

3.3.2.13 Payment

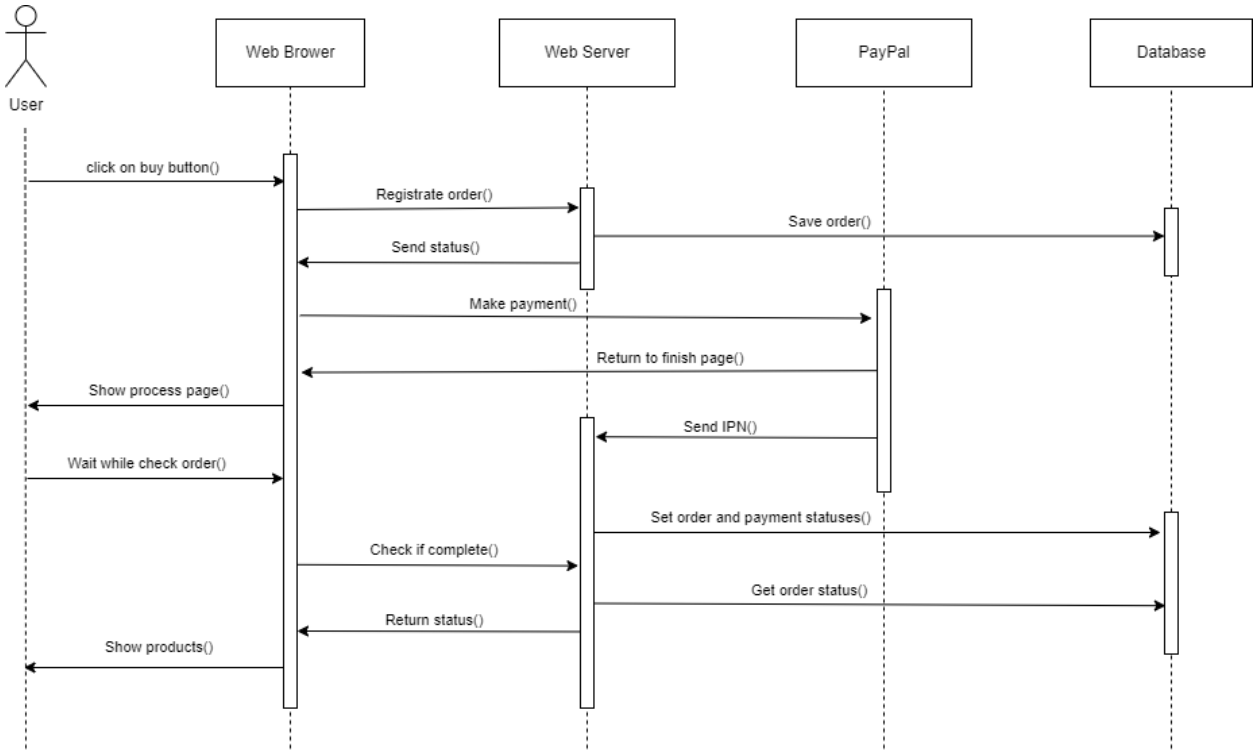

Figure 34: User Payment

3.3.2.14 Add favorite product

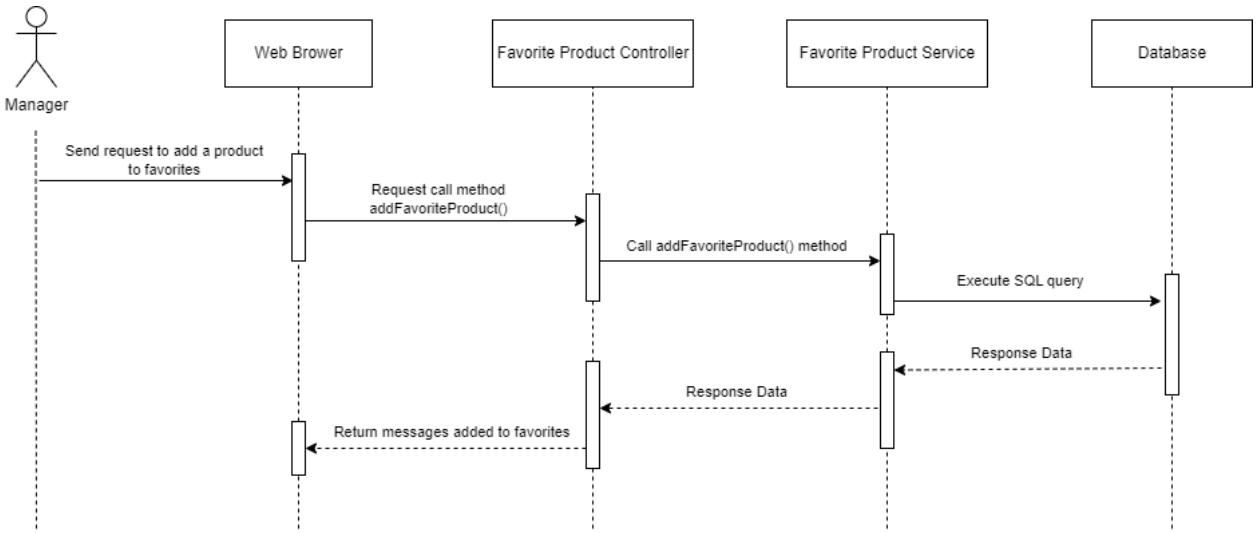

Figure 35: User Add favourite product

3.3.2.15 Mange favorite product list

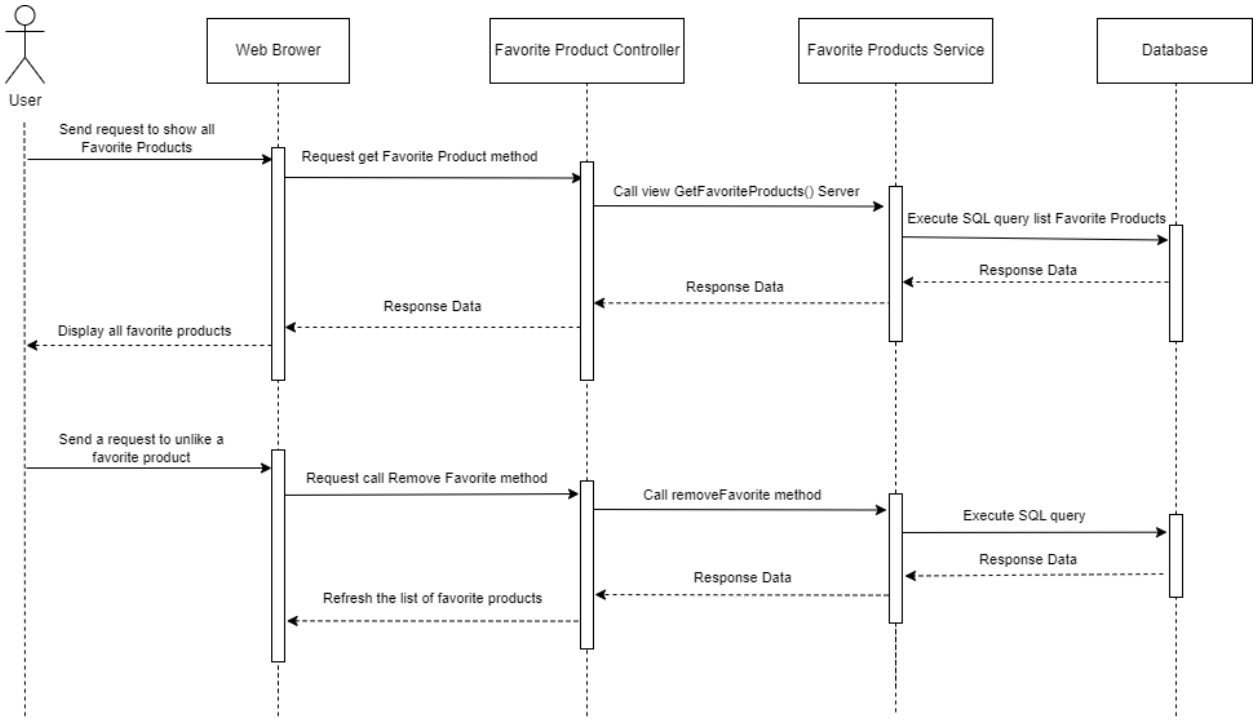

Figure 36: User Manage favourite product list

3.3.2.16 Manage account

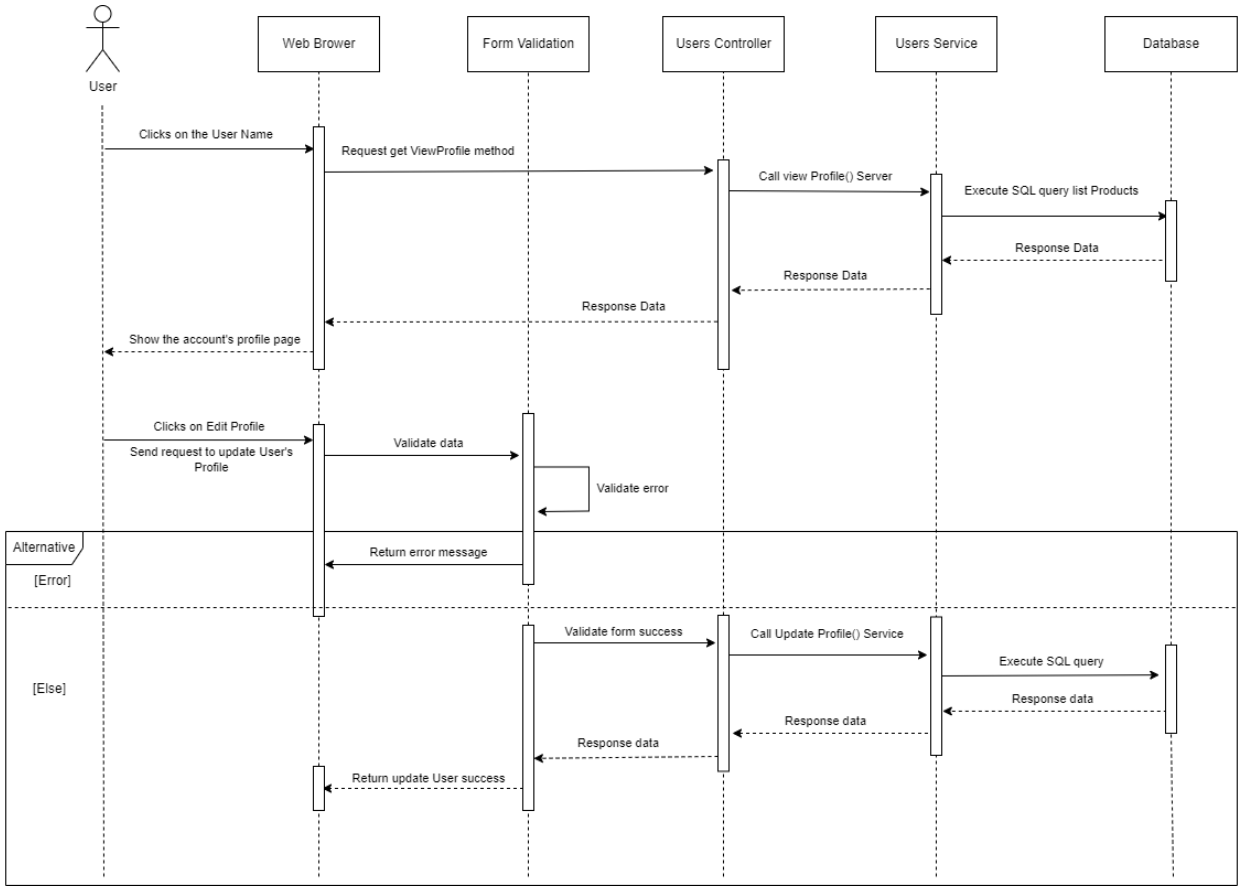

Figure 37: User Manage account

3.3.3 Manager

3.3.3.1 Login

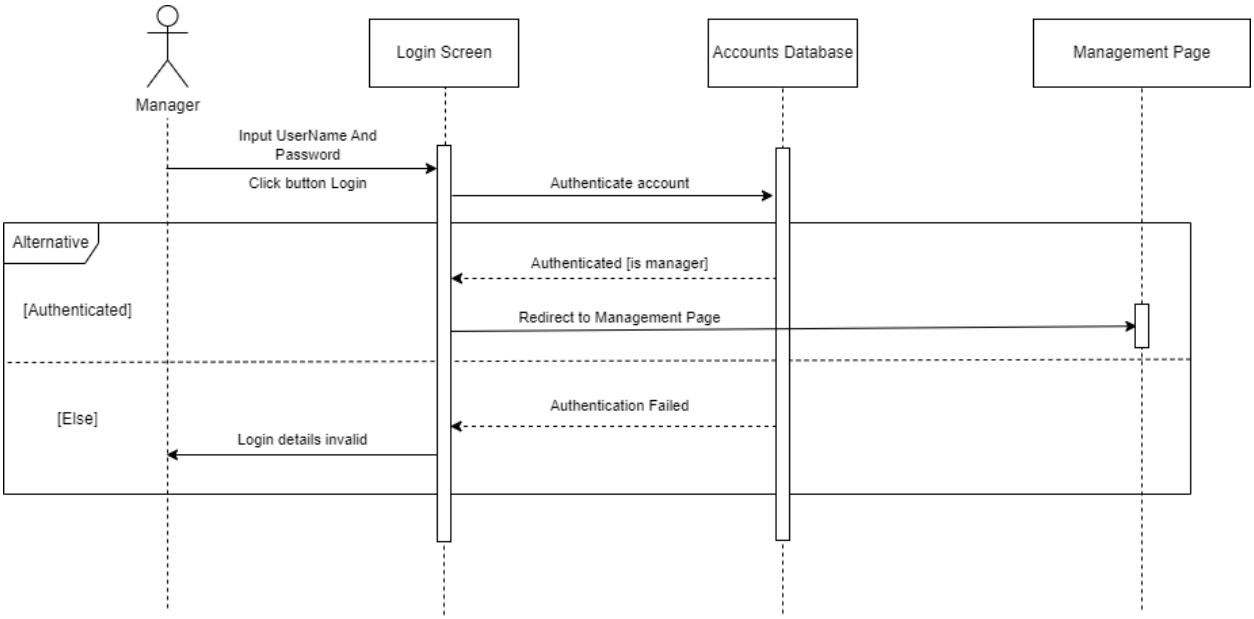

Figure 38: Manager Login

3.3.3.1 Logout

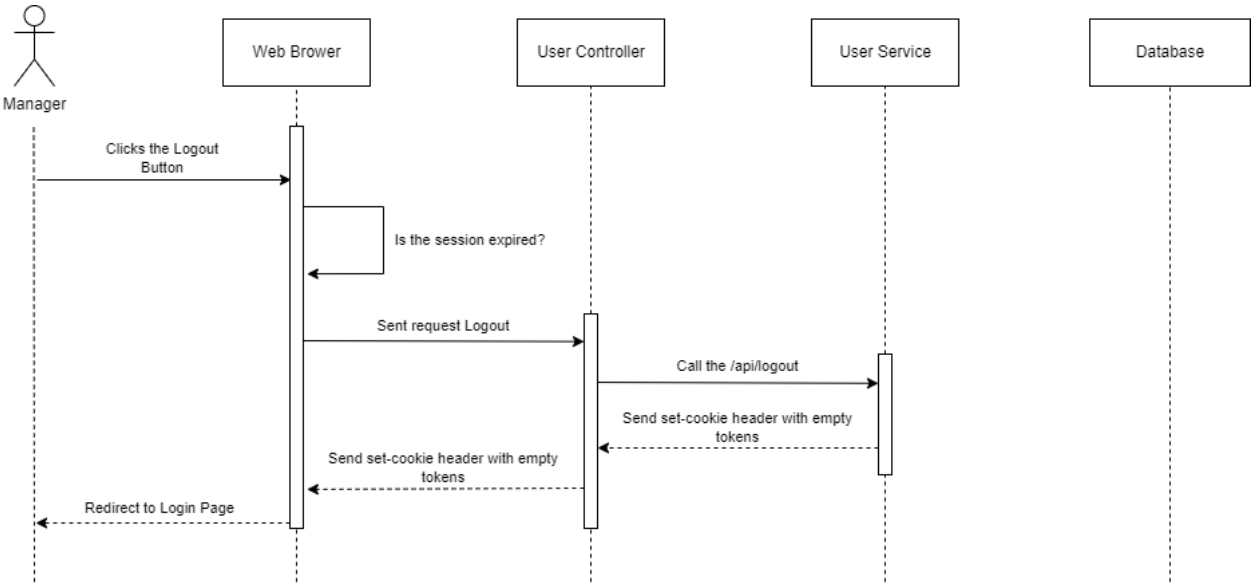

Figure 39: Manager Logout

3.3.3.1 Manage category

3.3.3.1.1 View Category

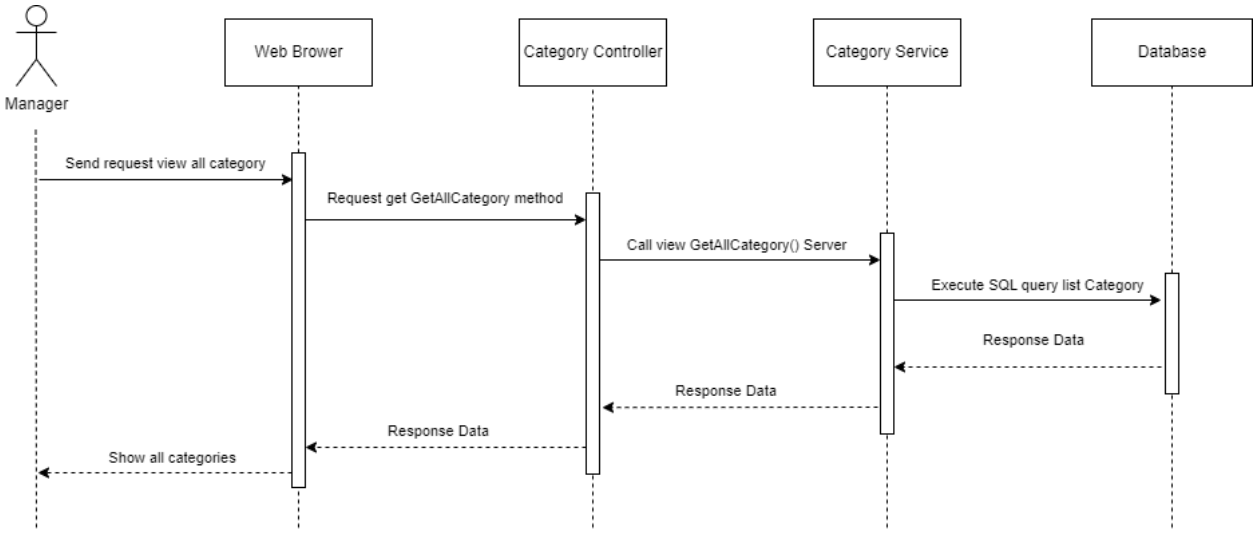

Figure 40: Manage View category

3.3.3.1.2 Create Category

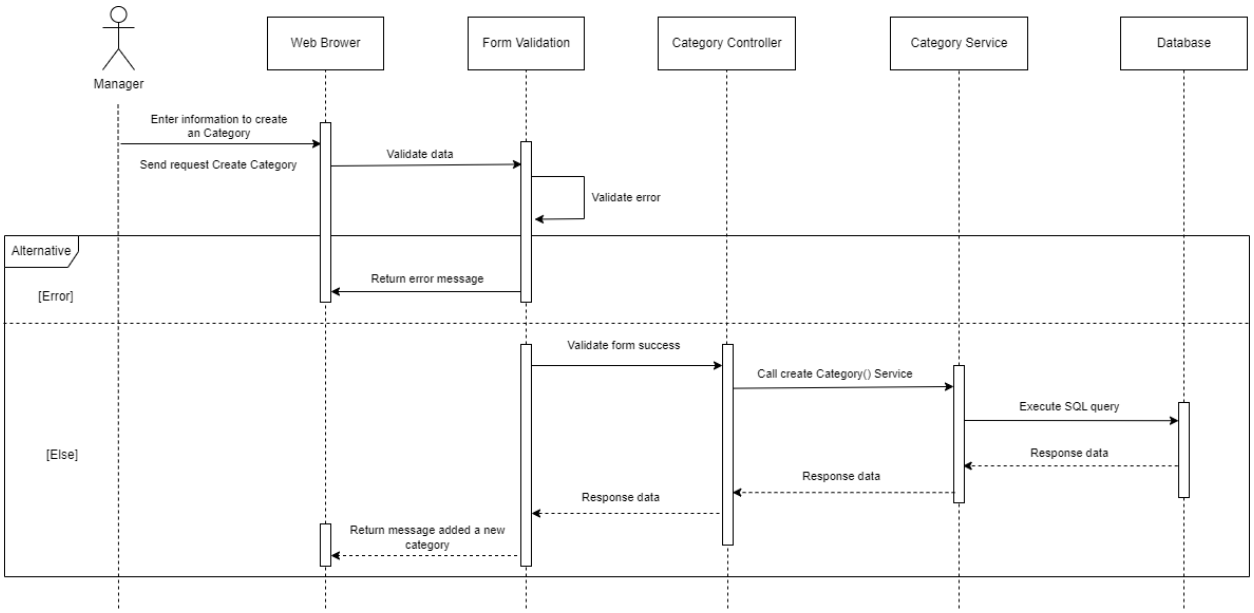

Figure 41: Manage Create category

3.3.3.1.3 Update Category

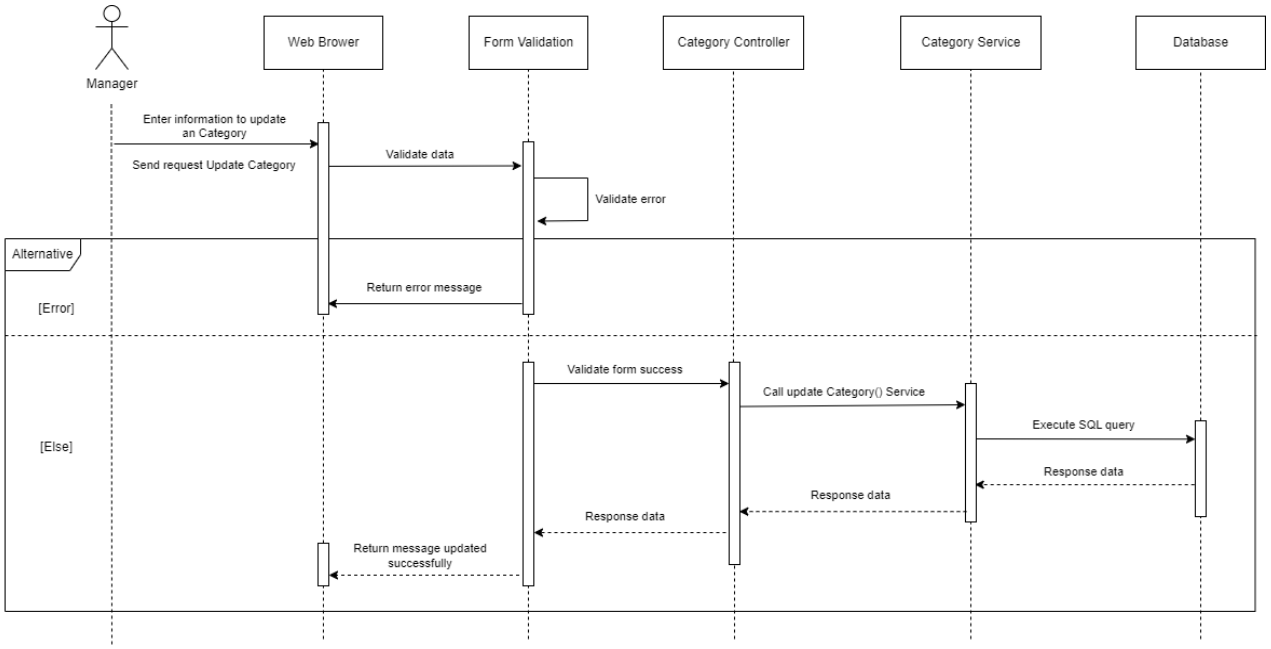

Figure 42: Manager Update category

3.3.3.1.4 Delete Category

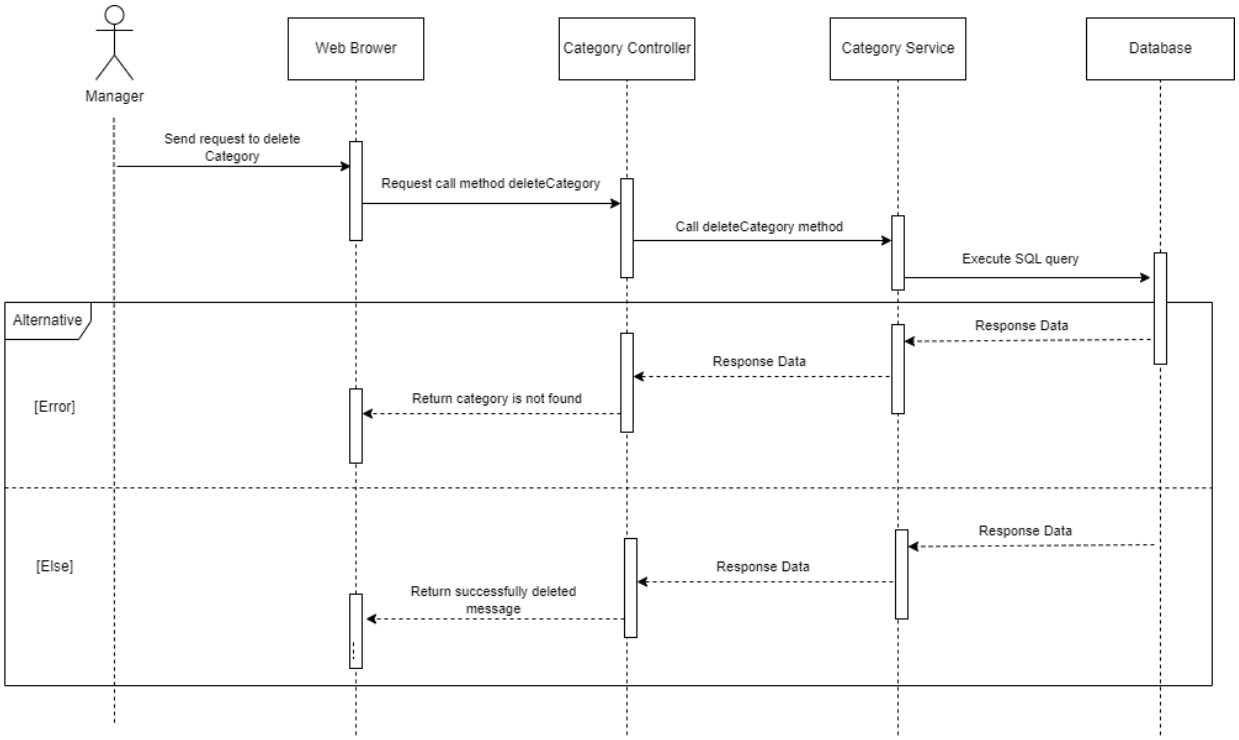

Figure 43: Manager Delete category

3.3.3.1 Manage all posting

3.3.3.1.1 View all posting

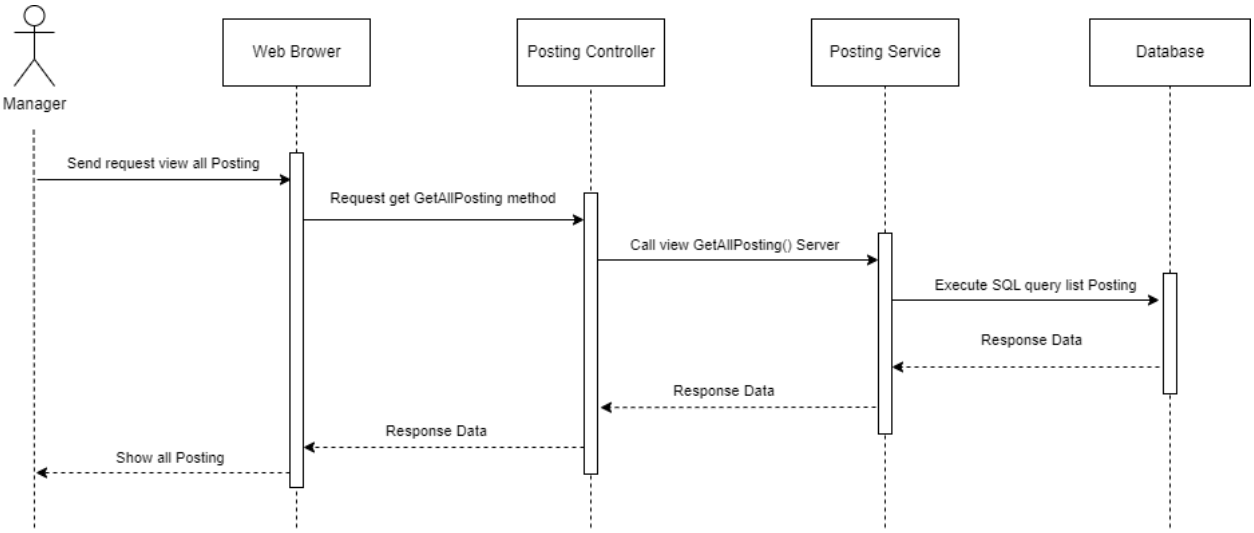

Figure 44: Manager View all posting

3.3.3.1.2 Create posting

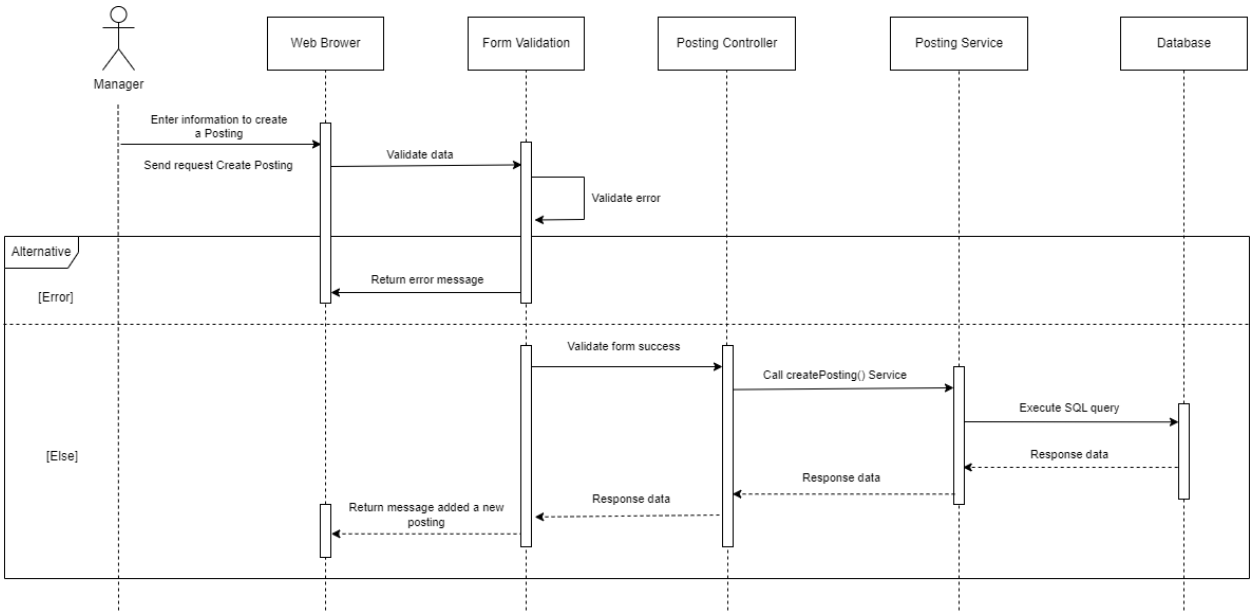

Figure 45: Manager Create posting

3.3.3.1.3 Update posting

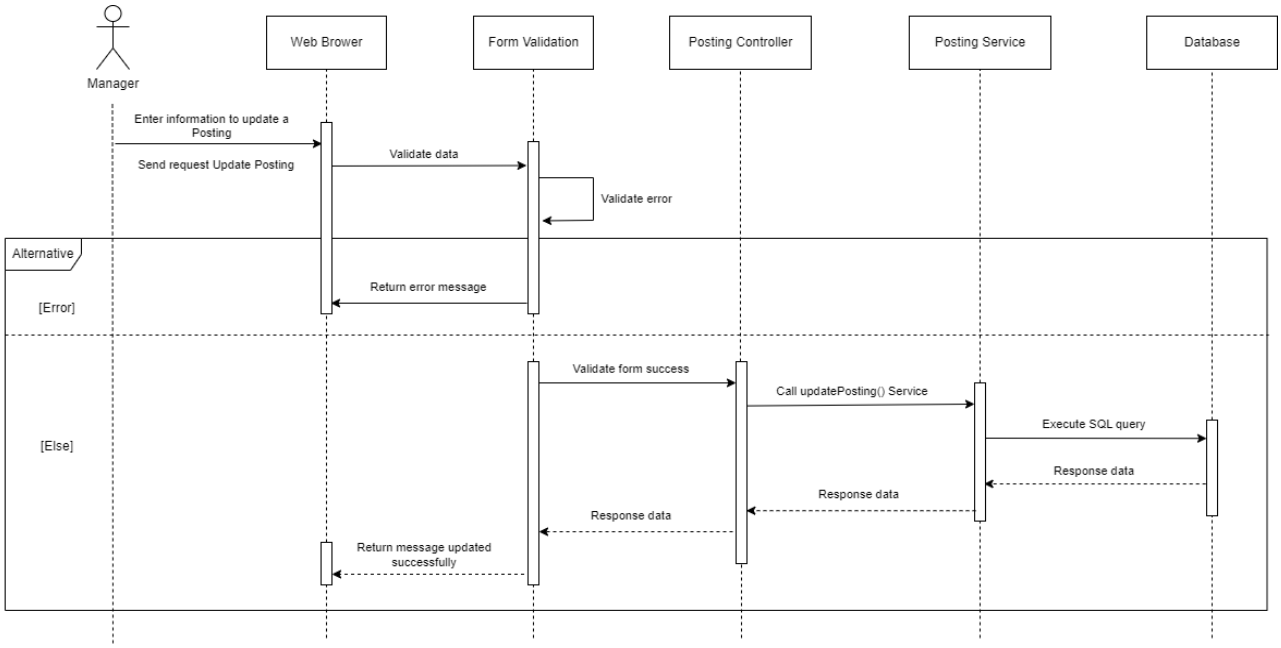

Figure 46: Manager Update posting

3.3.3.1.4 Delete posting

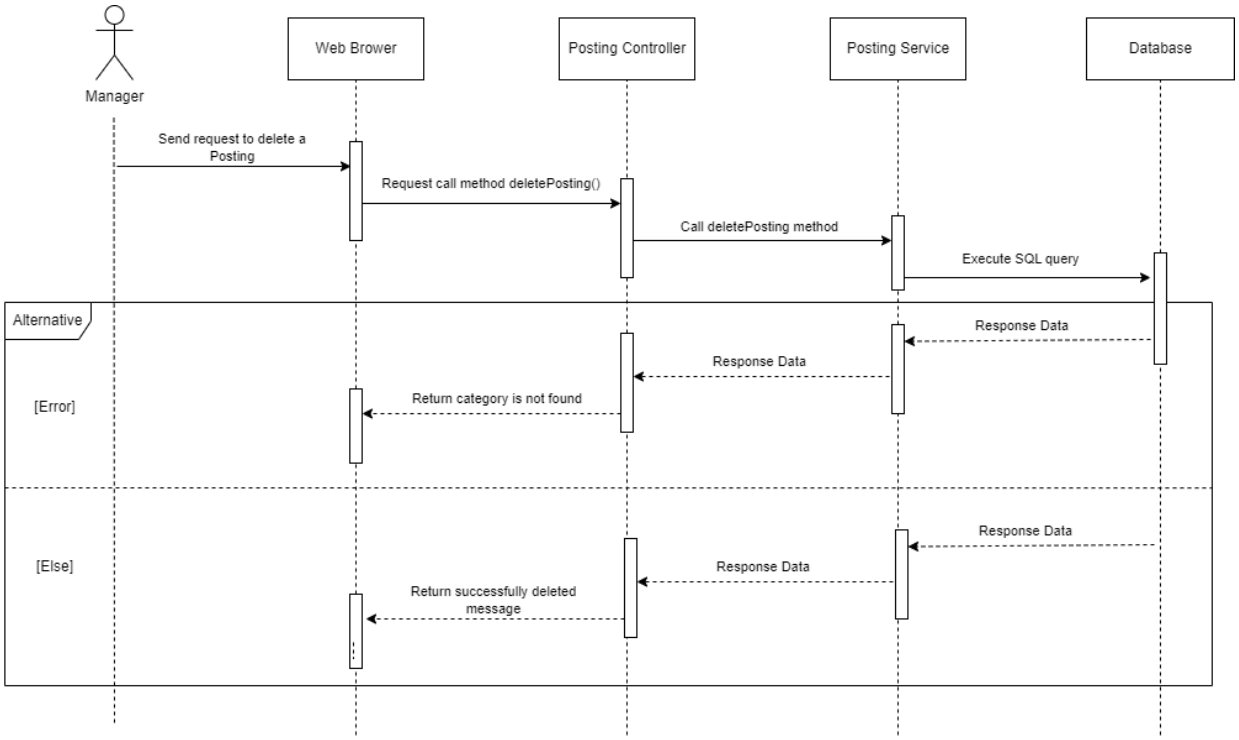

Figure 47: Manager Delete posting

3.3.4 Admin

3.3.4.1 Login

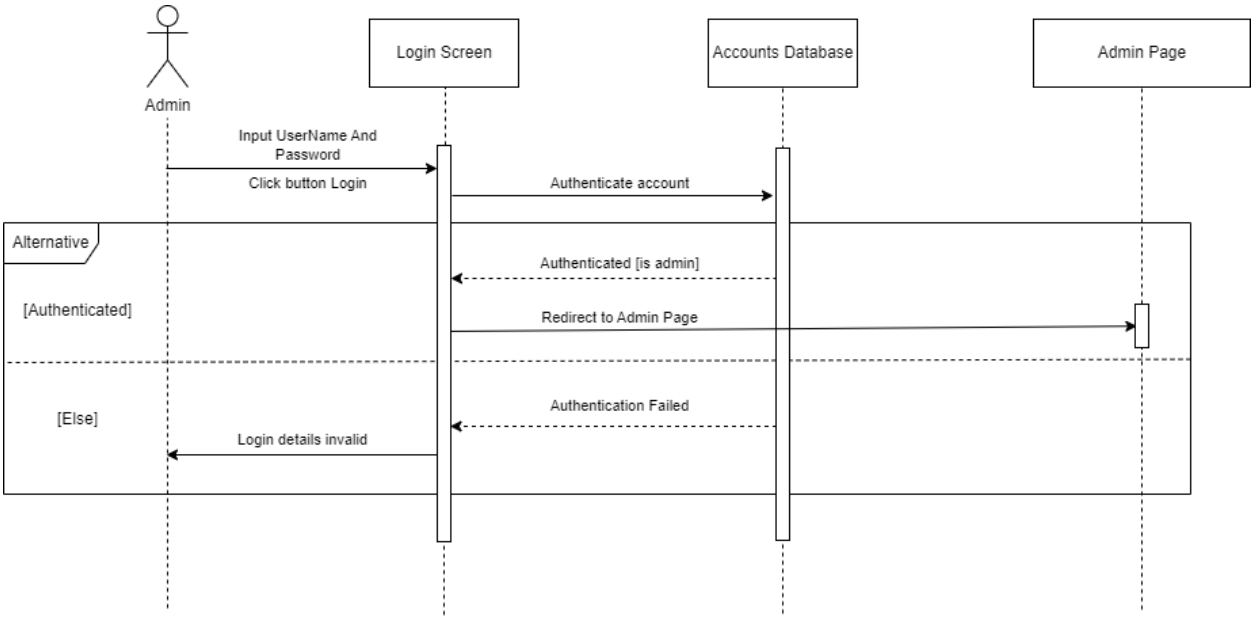

Figure 48: Admin Login

3.3.4.2 Logout

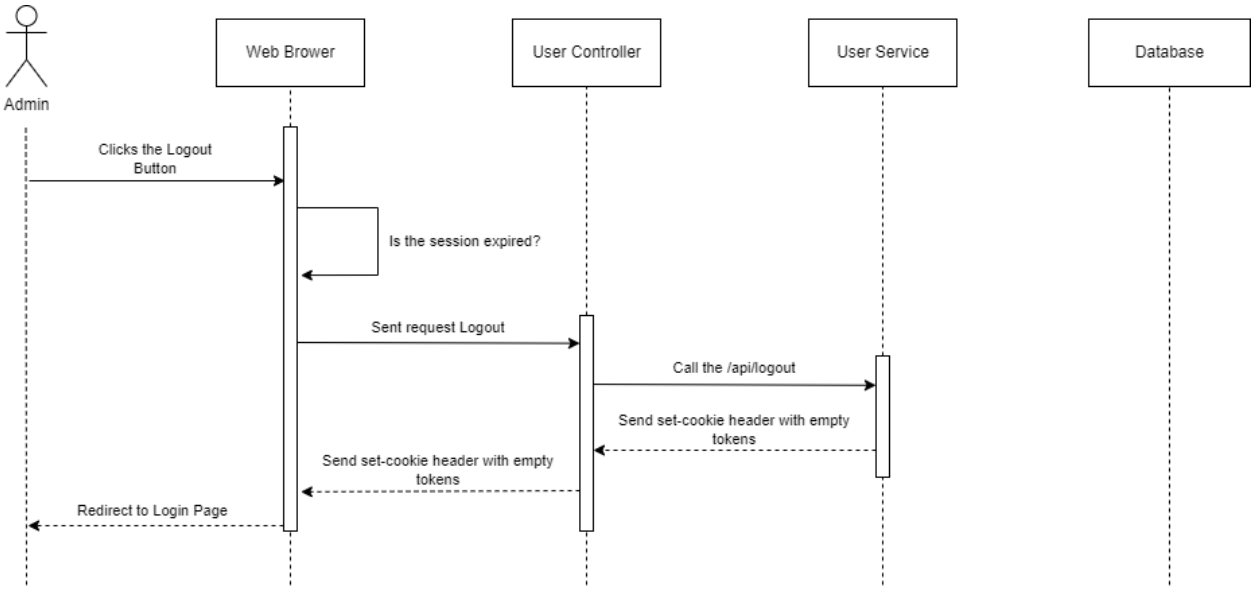

Figure 49: Manager Logout

3.3.4.3 Manage Category

3.3.4.3.1 View Category

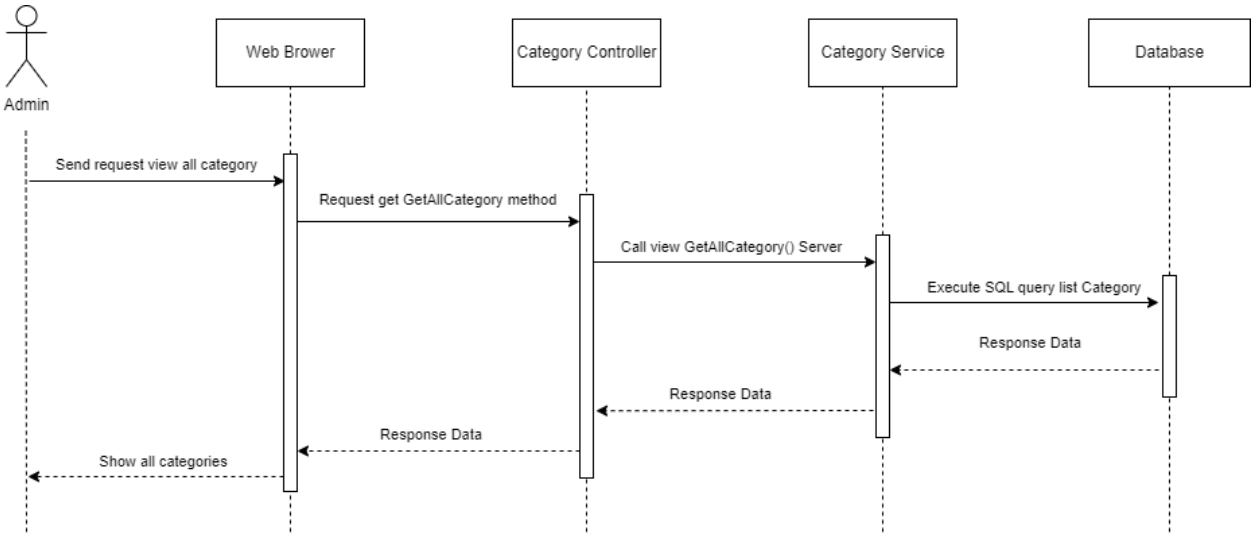

Figure 50: Admin View category

3.3.4.3.2 Create Category

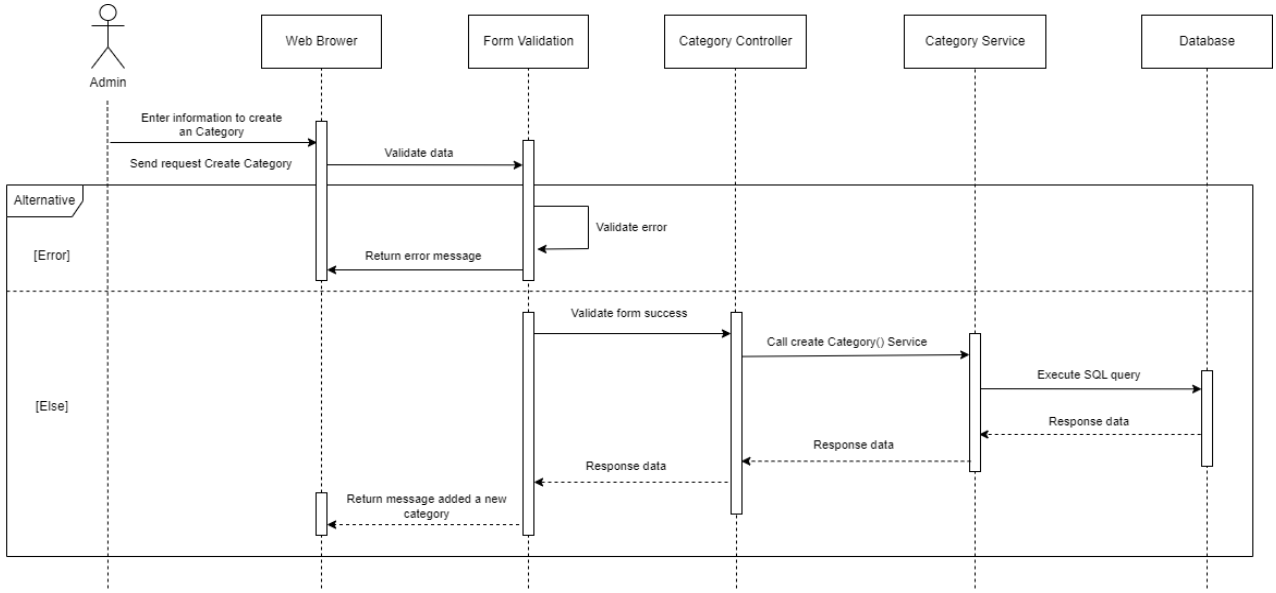

Figure 51: Admin Create category

3.3.4.3.3 Update Category

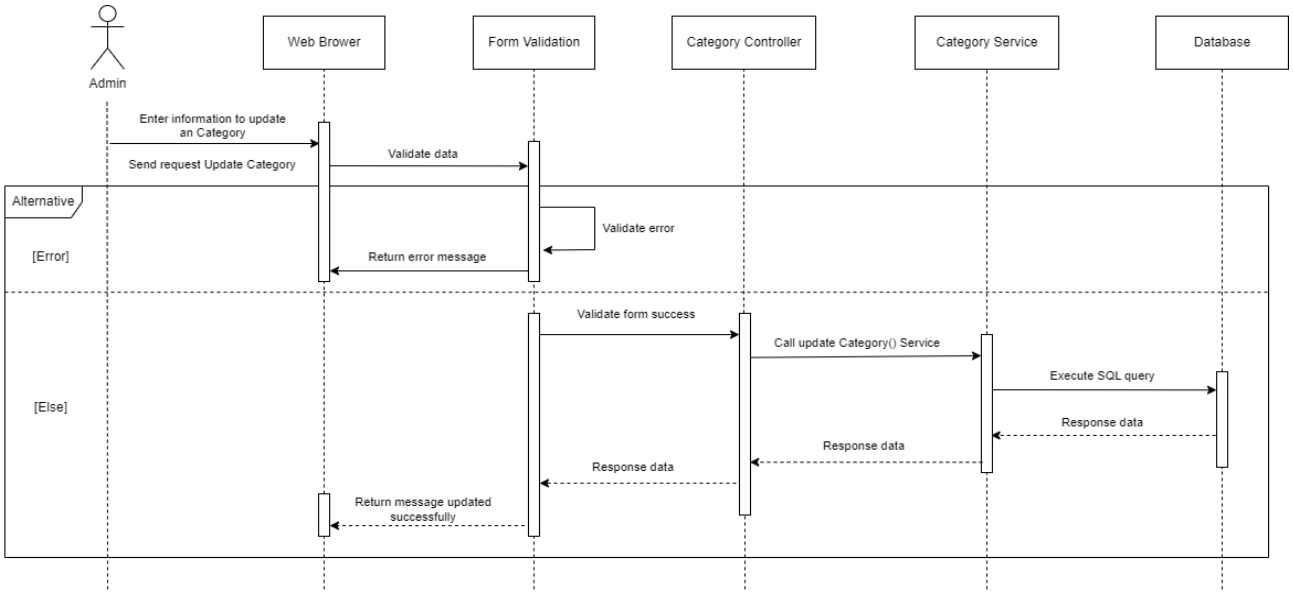

Figure 52: Admin Update category

3.3.4.3.4 Delete Category

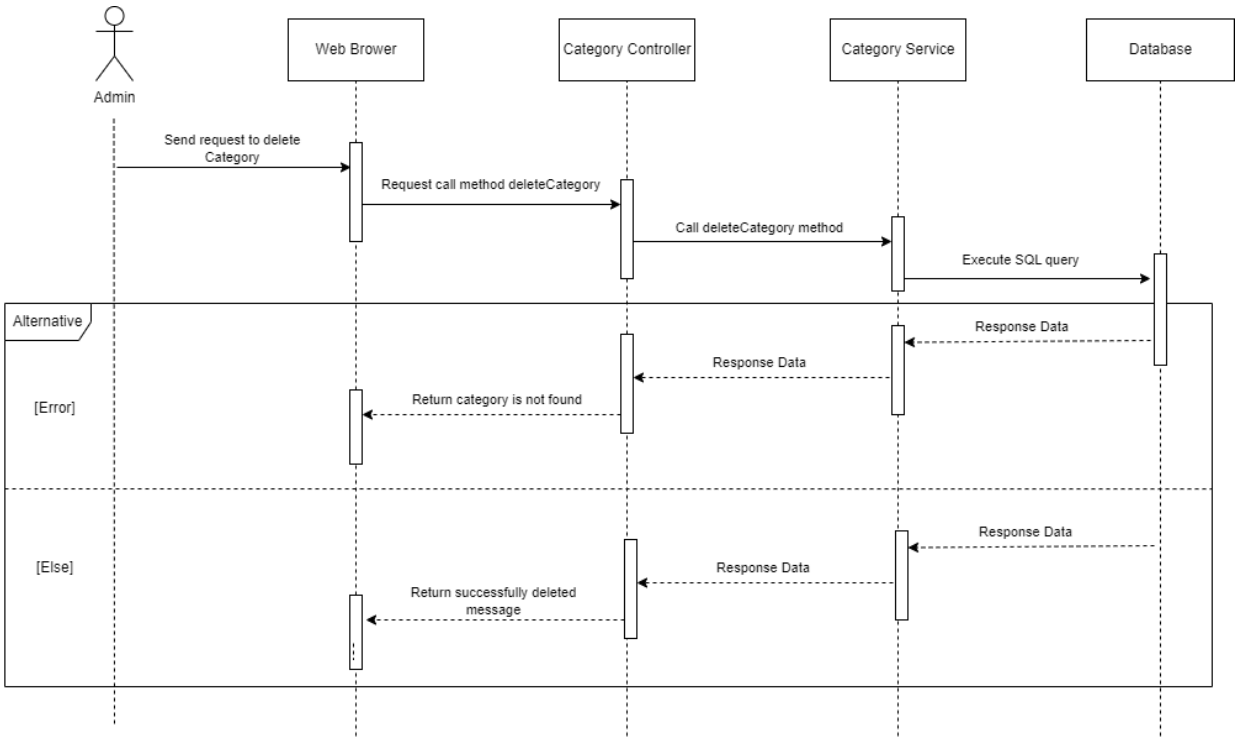

Figure 53: Admin Delete Category

3.3.4.4 Manage all posting

3.3.4.4.1 View all posting

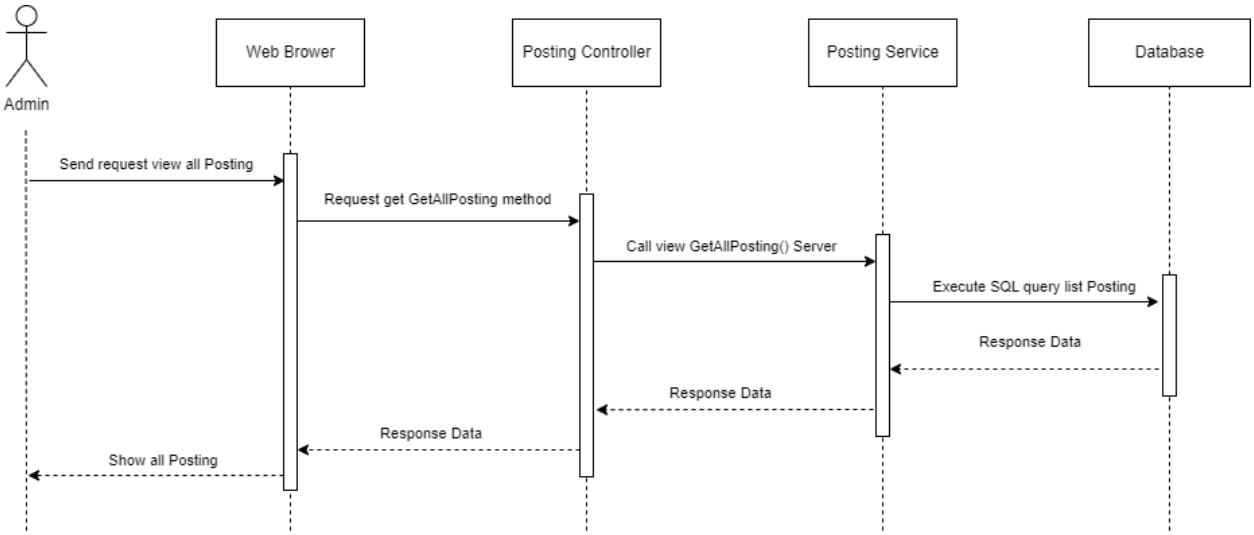

Figure 54: Admin View all posting

3.3.4.4.2 Create posting

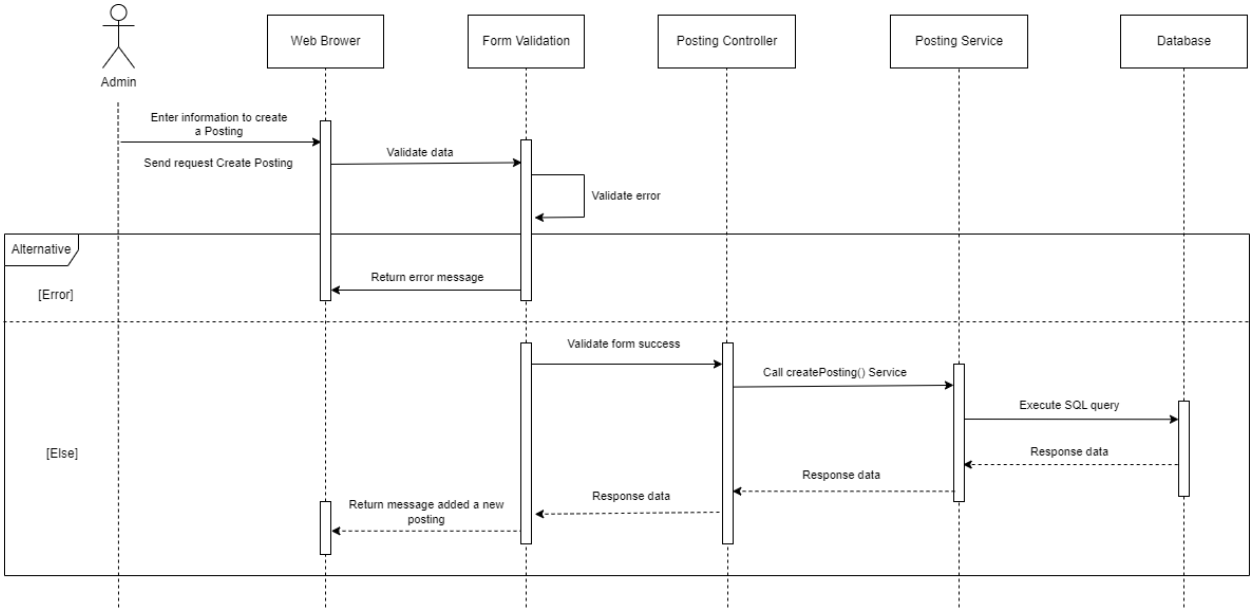

Figure 55: Admin Create posting

3.3.4.4.3 Update posting

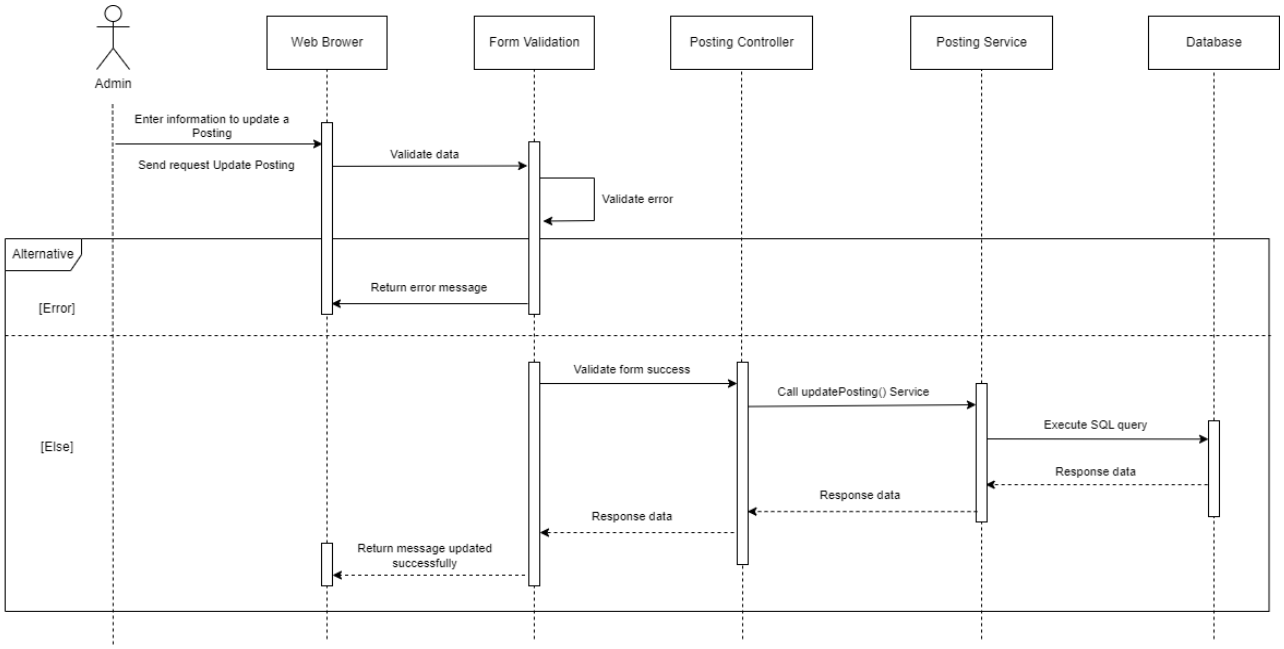

Figure 56: Admin Update posting

3.3.4.4.4 Delete posting

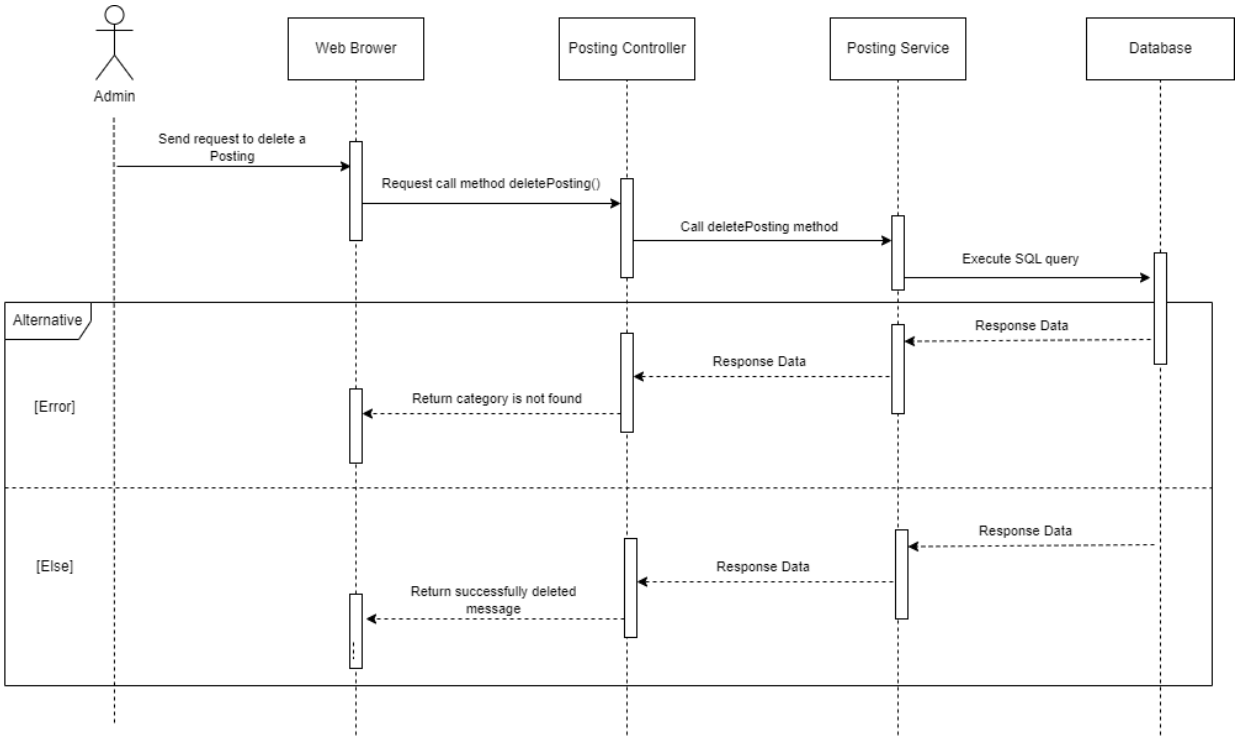

Figure 57: Admin Delete posting

3.3.4.5 Manage all account User and Manager

3.3.4.5.1 Create Account

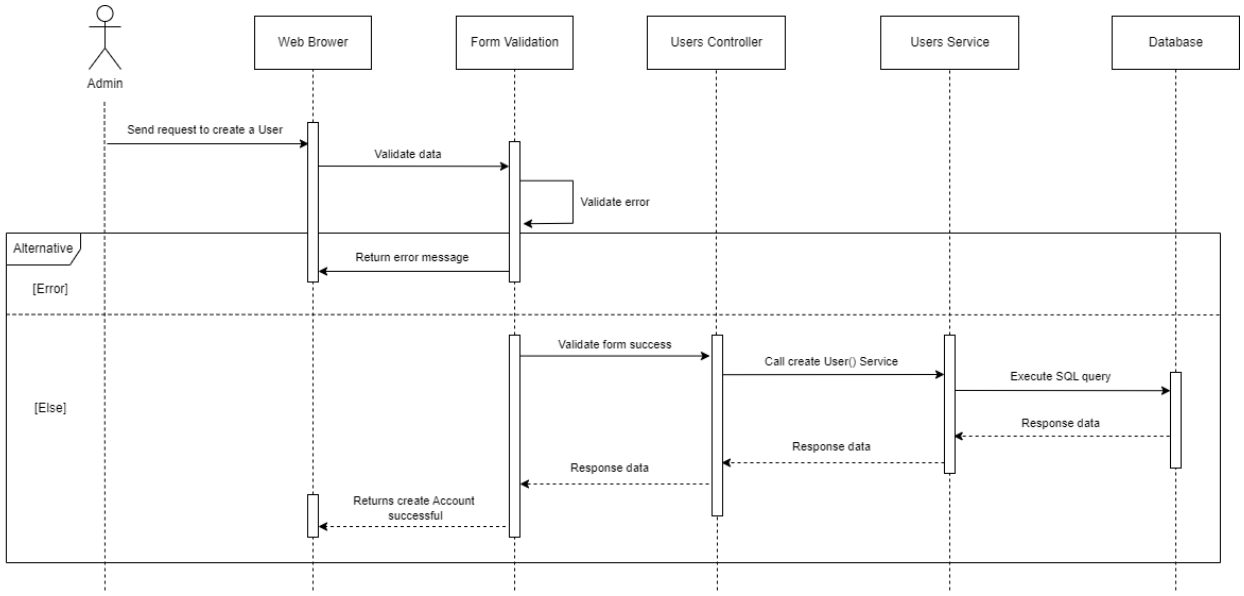

Figure 58: Admin Create Account

3.3.4.5.2 Update Account Role

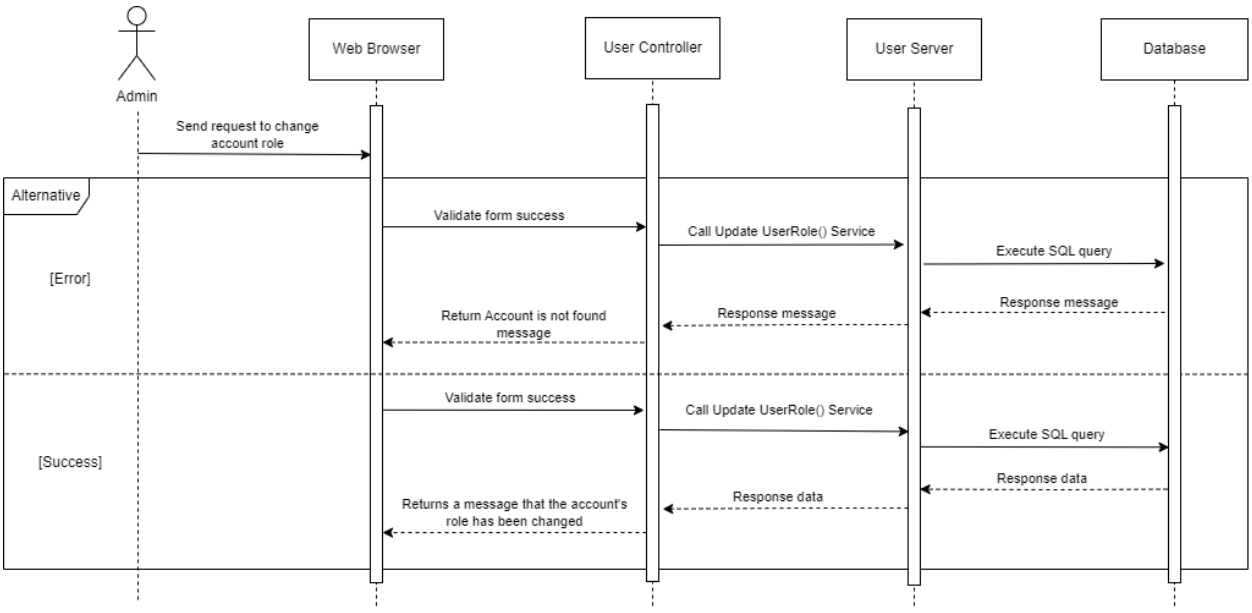

Figure 59: Admin Update account role

3.3.4.5.3 Ban Account

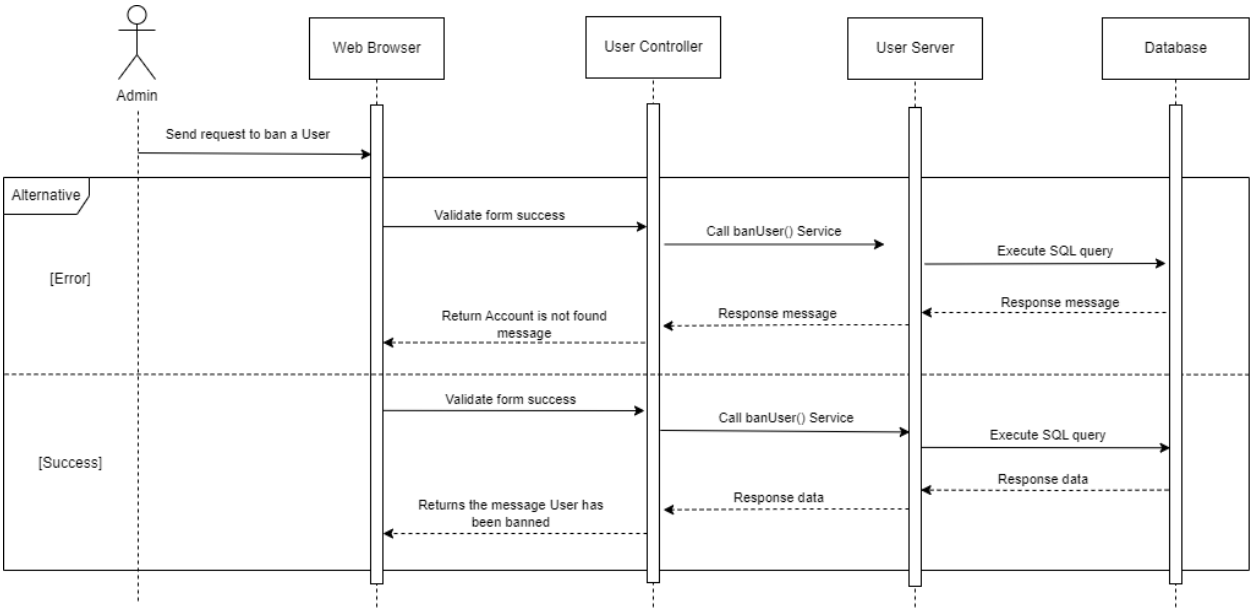

Figure 60: Admin Ban Account