# Usecases for PetCare app

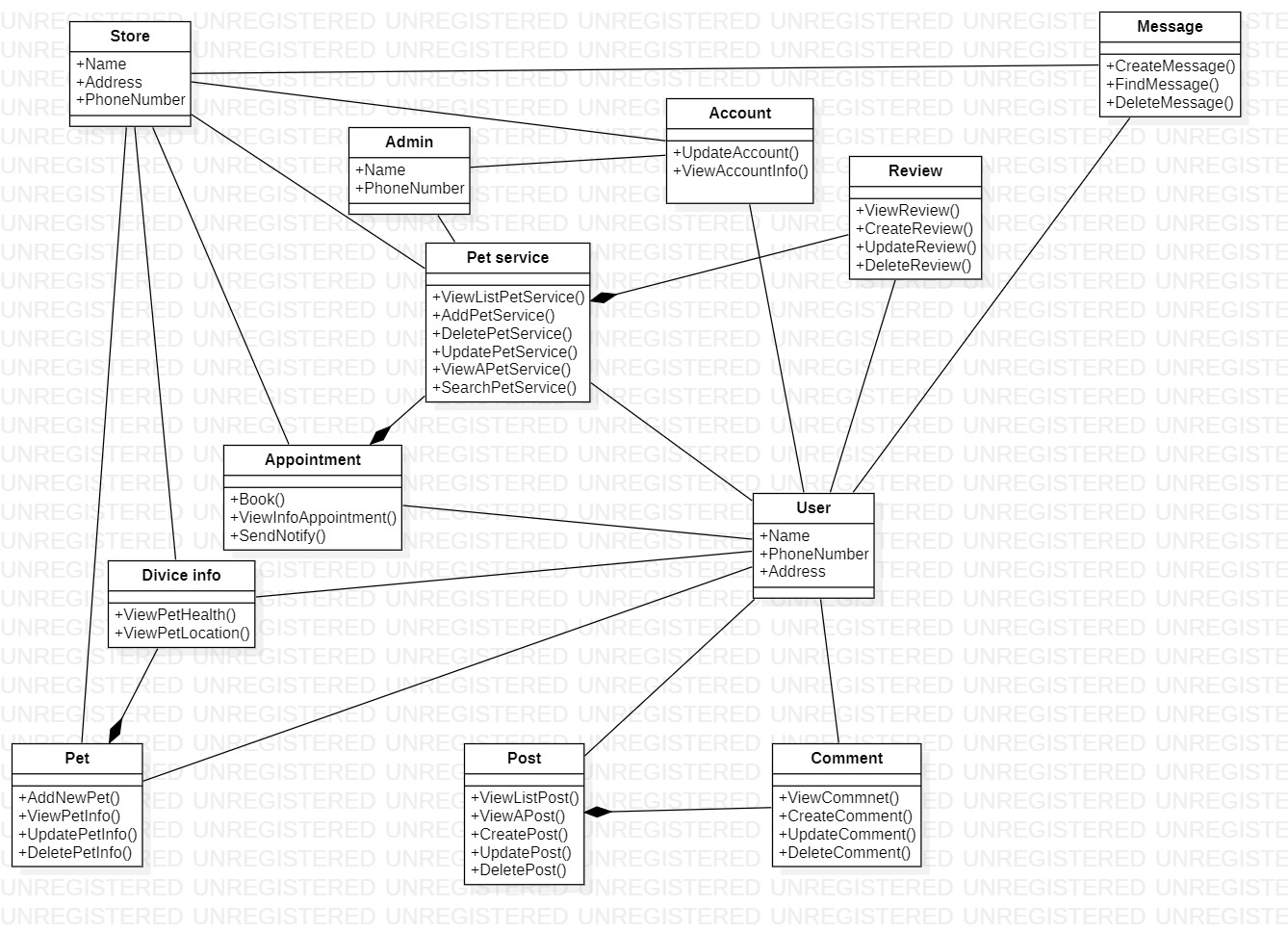
UR8- Users can book an appointment for their pet base on the service.

UR9- Users can make friend and chat with other users

StOR3- Store owners can create a new product to showcase

StOR6- Store owners can receive the status and information about the pet that the pet owner register with

# Class Diagram



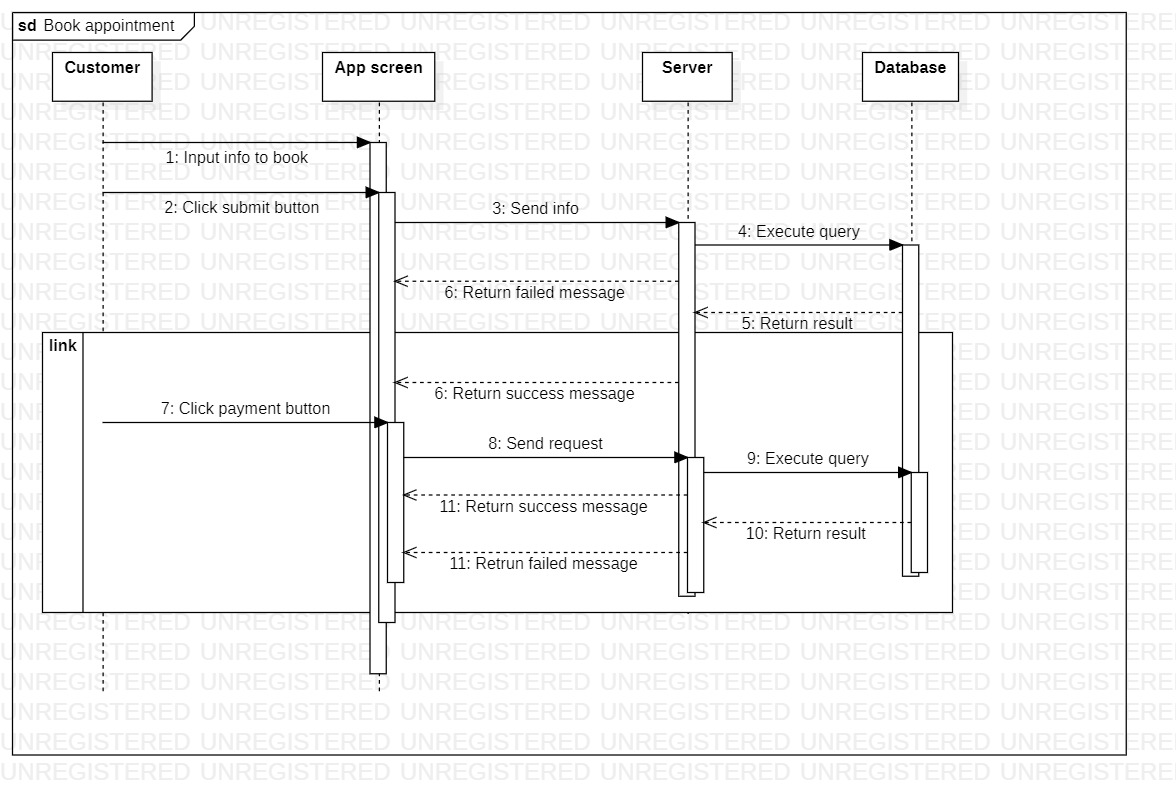
# The details for each use case

|  |  |
| --- | --- |
| ID and Name: | UR2- Users can book an appointment for their pet base on the service |
| Actor: | User |
| Description: | The user can choose services of a pet store which is for their pet. After choosing the user can book appointment to use that service |
| Normal Flow: | 1. The user choose the “Book an appointment” on the home screen 2. User choose the profile of the pet that you want 3. Choose the service that you want 4. Choose the time that the store has available 5. Press the “Book” button 6. The system displays the “Book successfully” |
| Pre-conditions: | User has logged in to the system |
| Post-conditions: | User successfully book an appointment  The appointment get add to the their notification |

Architect styles: MVC

Reason:

* MVC architecture will separate the user interface from business logic and business logic
* Components are reusable.
* Easy to maintain.
* This architecture help to test components independently.

Sequence diagram: 

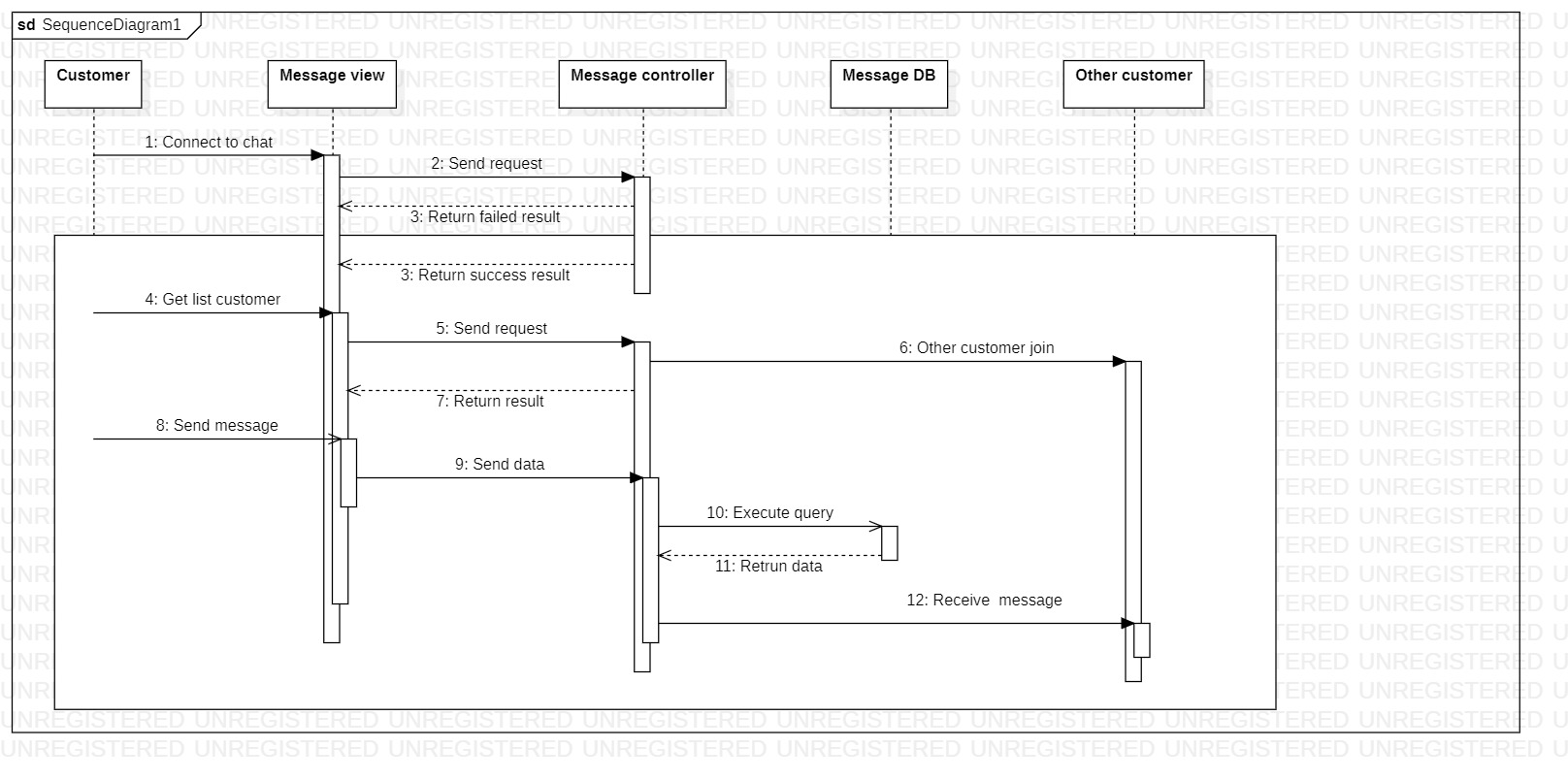
2)

|  |  |
| --- | --- |
| ID and Name: | UR9- Users can make friend and chat with other users |
| Actor: | User |
| Description: | User can make friend with others user by interact with, share details and updates related to their pets |
| Normal Flow: | 1. Press the “Pet social” on the main screen 2. You can check out others pet owner shared info 3. You can chat with that pet owner on their info |
| Pre-conditions: | User has logged in to the system |
| Post-conditions: | User get their posts upload successfully  User comment successfully to other posts |

Architect styles: Client-Server

Reason:

* Easy to maintain.
* Servers have better [control](https://cio-wiki.org/wiki/Control) and access resources to ensure that only authorized clients can access or manipulate data
* It is easy to manage, and the data can be easily delivered to the client.

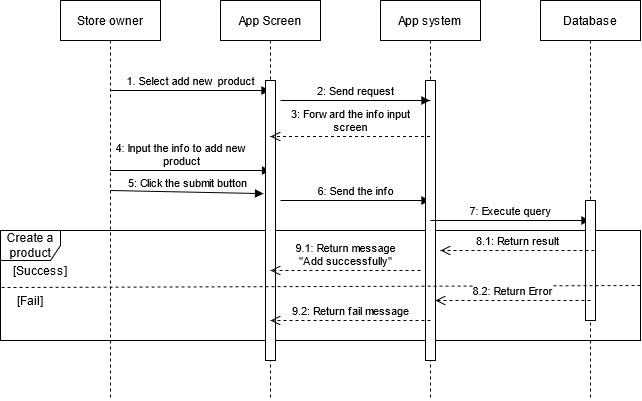
Sequence diagram: 

|  |  |
| --- | --- |
| ID and Name: | StOR3- Store owners can create a new product to showcase |
| Actor: | Store owner |
| Description: | Store owner can provide not only service but also the product that they have gotten on the store. They can add their product’s info on the main store to showcase the product |
| Normal Flow: | 1. Click on the “Add new product” 2. Input all the necessary information 3. Press the “Submit” button 4. The system show the message “The product has been added to your store” |
| Pre-conditions: | Store owner has logged in to the system |
| Post-conditions: | The product get add to their store  The product show up on the store if the user search for |

Architect styles: MVC

Reason:

* MVC architecture will separate the user interface from business logic and business logic
* Components are reusable.
* Easy to maintain.
* This architecture help to test components independently.

Sequence diagram: 

|  |  |
| --- | --- |
| ID and Name: | StOR6- Store owners can receive the status and information about the pet that the pet owner register with |
| Actor: | Store owner |
| Description: | The store can get the necessary information of the pet that the pet owner book the appointment. The store owner can prepare and ask the pet owner any problems about their pet |
| Normal Flow: | 1. Check the appointment that the user book 2. Click on the pet profile 3. The app show all the information that the pet has |
| Pre-conditions: | Store owner has logged in to the system  Store owner check the store‘s appointment |
| Post-conditions: | The user’s pet profile add to the appointment that has been made  Store owner can check the user’s pet conditions |

Architect styles: Blackboard

Reason:

* Provides scalability which provides easy to add or update knowledge source.
* Provides concurrency that allows all knowledge sources to work in parallel as they are independent of each other.
* The system sends notifications known as trigger and data to the clients when changes occur in the data.
* Easy to add new applications

Sequence diagram: 