**Requirements Specifications**

**PO1:PETSWALA**

| **Student ID** | **Name** |
| --- | --- |
| **22100300** | **Muhammad Aaish Javed** |
| **22100098** | **Muhammad Ibrahim Bhalli** |
| **20100049** | **Syed Raza Abbas** |
| **22100200** | **Muhammad Tayyab** |

| **Content** | **Totals** | **Obtained** |
| --- | --- | --- |
| Introduction & system actors | 5 | 5 |
| Use case diagram | 10 | 9 |
| Use case descriptions | 20 | 10 |
| Class diagram | 20 | 8 |
| Sequence diagram | 20 | 2 |
| State diagram | 5 | 5 |
| Non-functional requirements | 5 | 5 |
| Who did what | 5 | 5 |
| Review checklist | 5 | 5 |
| Overall formatting/template | 5 | 4 |
| Late submission penalty | -20 |  |
| **Total** | **100** | **58** |
| Review | 20 |  |
| **Grand Total** |  |  |

**Table of Contents**

1. Introduction. 4

2. System Actors. 5

3. Use Cases. 6

3.1 Use Case Diagrams. 6

3.2 Description of Use Cases. 10

3.2.1 Signup of General Users

3.2.2 Signup of Service Providers

3.2.3 Signup of Veterinary Doctors

3.2.4 Signup of Rescue Services

3.2.5 Login

3.2.6 Logout

3.2.7 Change Password

3.2.8 Reset Password

3.2.9 Create Blog Posts

3.2.10 Edit Blog Posts

3.2.11 Chat

3.2.12 Comment on Blog Posts

3.2.13 Review and Rate Services

3.2.14 Report User or Service

3.2.15 Approve Service Provider/Vet/Rescue Service (Admin)

3.2.16 Search on Blog

3.2.17 Search Marketplace

3.2.18 Add User Profile (Admin)

3.2.19 Edit User Profile (Admin)

3.2.20 Delete User Profile (Admin)

4. Class Diagram. 39

4.1 Diagram. 39

4.2 Description. 40

5. Sequence Diagrams. 41

5.1 Signup of General User. 41

5.2 Signup of Service Provider. 42

5.3 Signup of Vet. 43

5.4 Signup of Rescue Services. 43

5.5 Login. 44

5.6 Logout. 44

5.7 Change Password. 45

6. State Diagrams. 46

6.1 Diagram details (Search and Add to Cart). 46

6.2 Diagram (Search and Add to Cart). 47

6.3 Diagram details (Approve Vet/Service Registration). 48

6.4 Diagram (Approve Vet/Service Registration). 48

6.5 Diagram details (Create/Edit Blog Posts). 49

6.6 Diagram (Create/Edit Blog Posts). 49

7. Non-functional Requirements / Quality Attributes. 50

7.1 Quality Attributes. 51

8. Who Did What? 53

9. Review checklist. 54

**1.** **Introduction**

**1.1 Document Purpose**

This document is the system requirements specification for an online web-based platform for pet owners. Our product aims to provide pet owners and others in the veterinary or pet service industry with an online platform where both service providers and customers can interact and engage in commerce. The document will provide an overview of what our web application would offer, including but not limited to its functionalities, constraints, assumptions and dependencies.

**1.2 Product Scope**

We will be developing a web application which registers pet owners, vets, other pet service and accessories providers and possibly pet experts and allows them to interact based on their needs. For instance, pet service providers and vets would be able to advertise their services and products and pet owners would be able to contact these service providers as well as connect with other pet owners for any purpose such as breeding, purchasing or selling.

**1.3 Potential Users**

As mentioned earlier, potential users of the web-application will be key characters in the pets’ communities: pet owners, service providers, people interested in getting a pet, and as well as application and business administrators.

# **2.** **System Actors**

| **Actor Name** | **Description** |
| --- | --- |
| General User | Users can find nearby service providers. They can post pets or accessories to sell in the marketplace. They can give services related feedback, ratings and reviews. They can create posts for articles and videos on the Blog. They can chat with other users using the in-app messenger. |
| Administrator | Admins can Add/Edit/Delete Service Provider profiles. Verify and approve service providers. Supervise the Marketplace. Provide feedback to user complaints. |
| Service Providers | Service providers can advertise their services, sell products in the marketplace and post on the blog.  For instance, Pet shops can provide their location, the pets they have, and their prices so that the users can contact them in case they need to buy a pet. They can chat with other users using the in-app messenger.  . |
| Rescue Services | Rescue services can deal with user reports and resolve them. They can be contacted by the users urgently in case of emergencies. They can also post on the blog. They can chat with other users using the in app messenger. |
| Veterinary Doctors | Veterinary doctors can provide their location, services and timings so the users can contact them. They can chat with other users using the in-app messenger. |

# 

# 

# 

# 

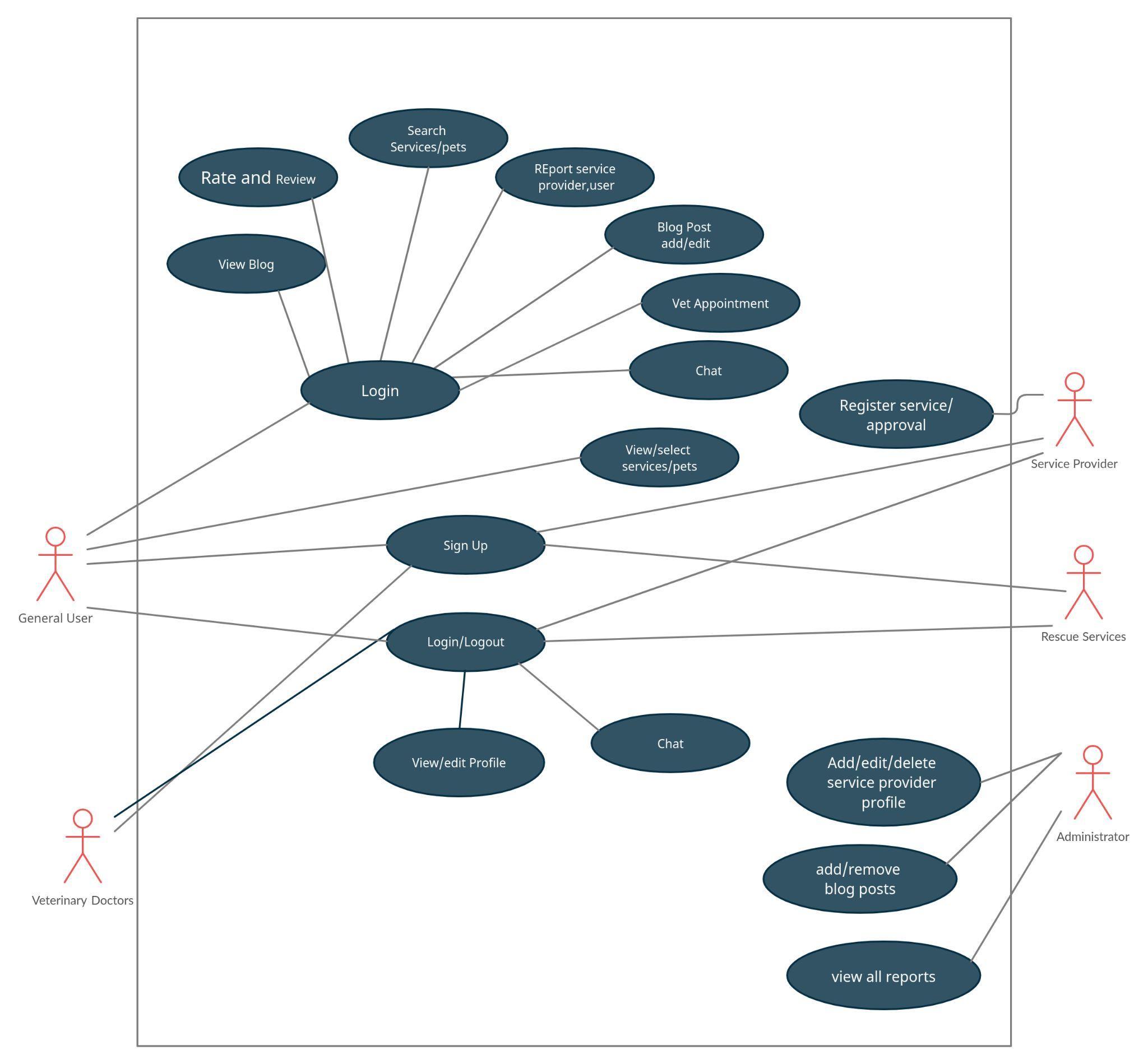
# 

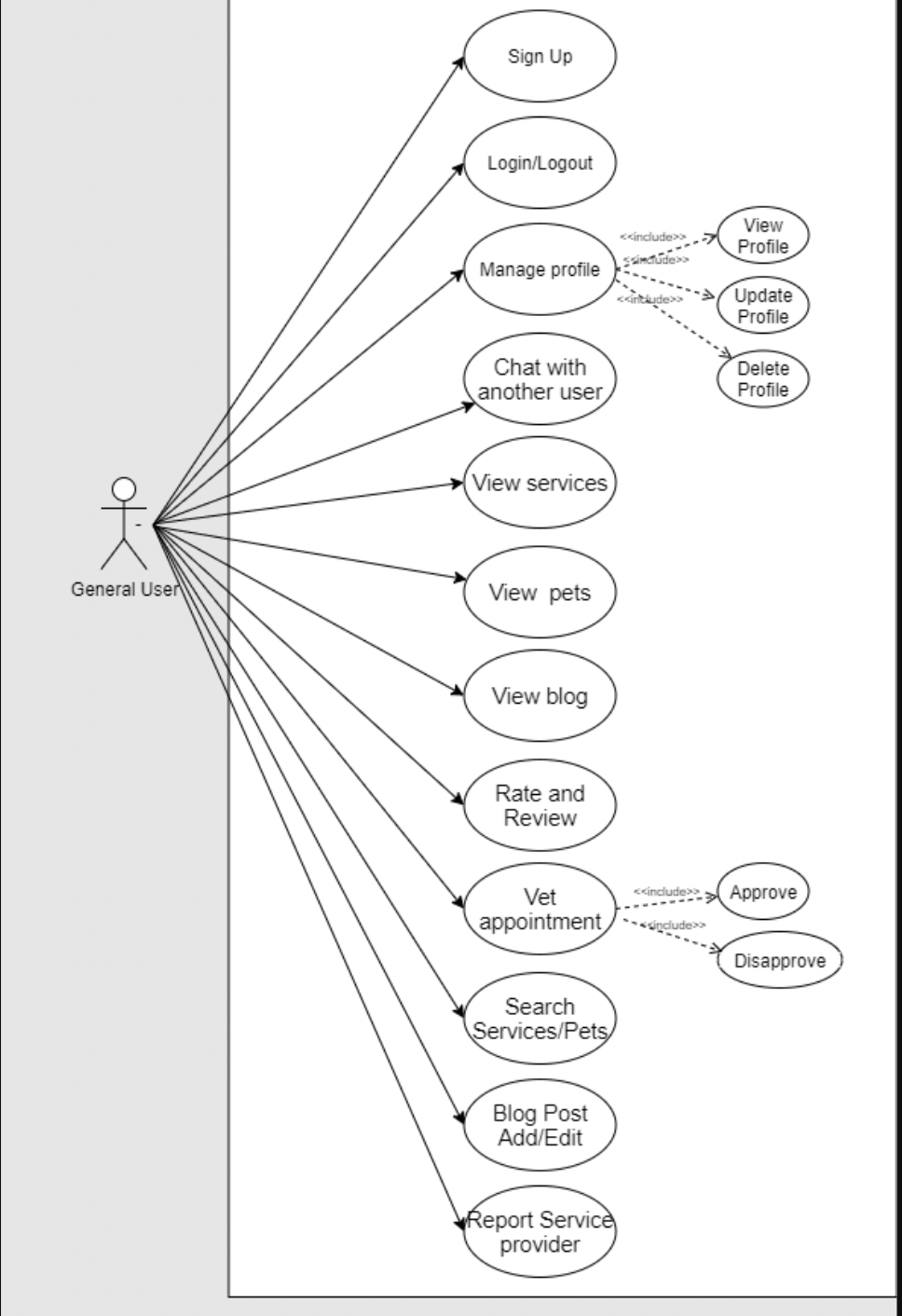
# 

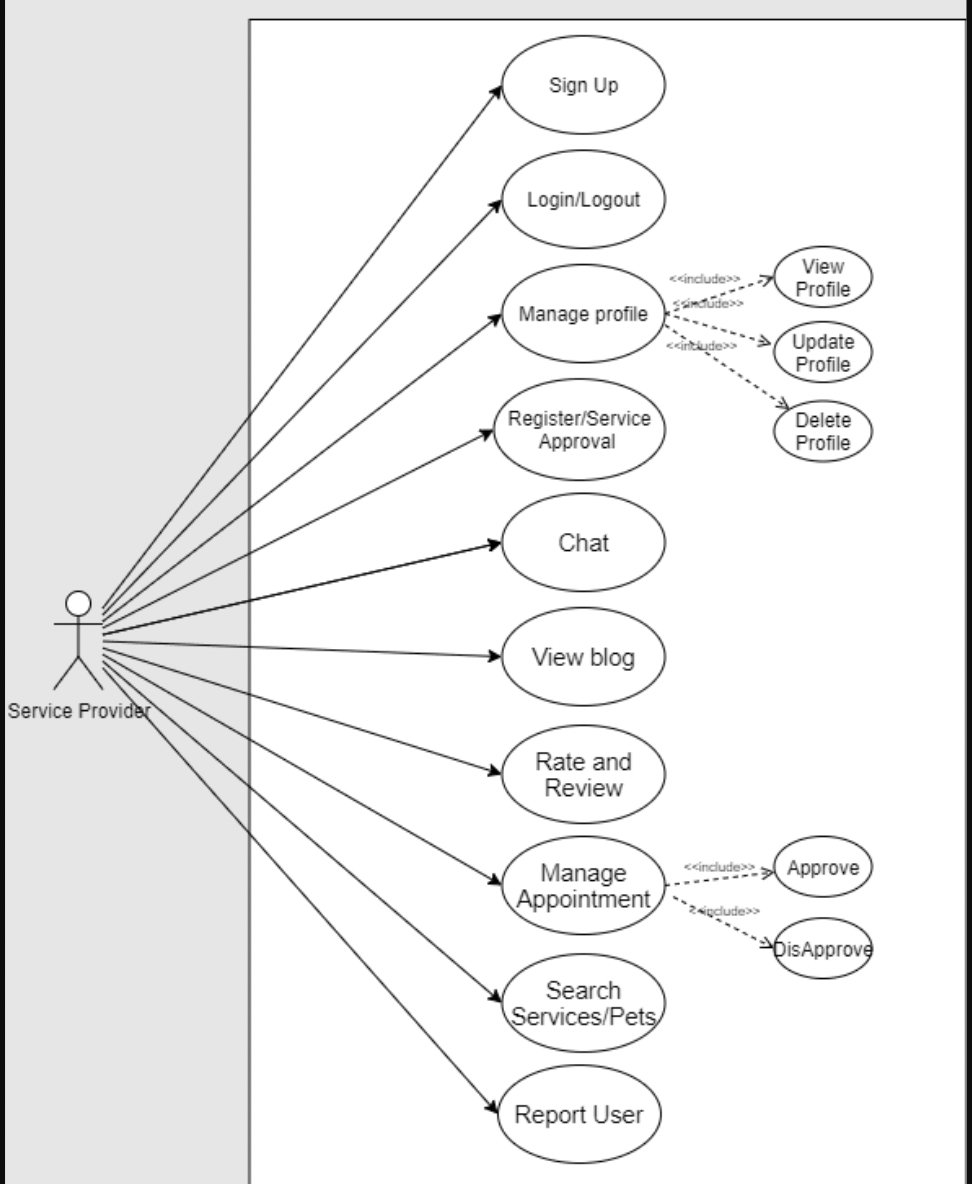
# **3.** **Use Cases**

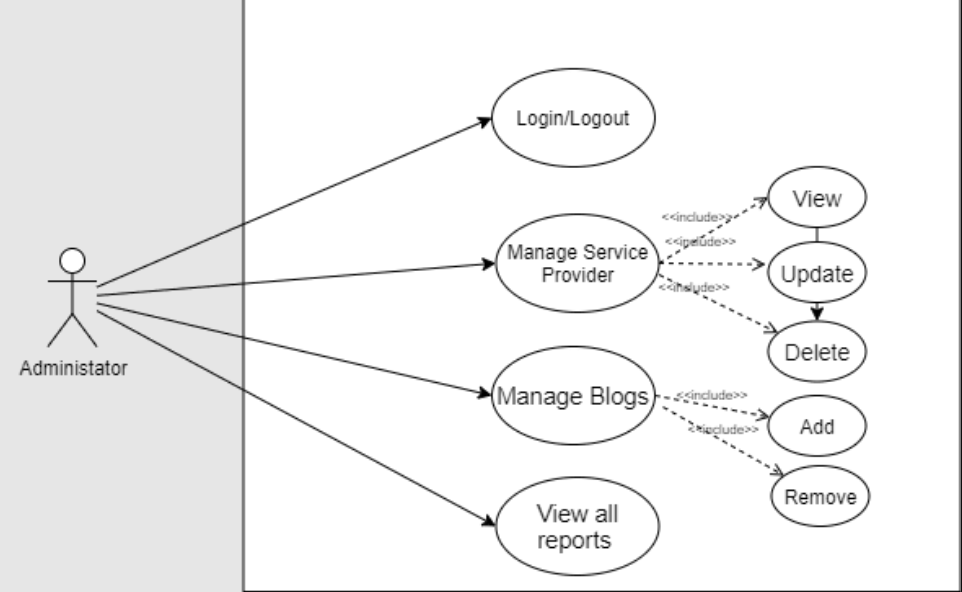
## **3.1 Use Case Diagrams**

[In the diagrams below, how is a *User* different from *General User* and *Admin* from *Administrator*? It is better if you follow proper naming convention and review the document.]









## 

## 

## 

## 

## 

## 

## 

## 

## 

## **3.2** **Description of Use Cases**

[Carefully look at the use case sample provided in the template. Write use case descriptions that distinguish user actions and system responses.

In general, there are issues with overall content including pre-conditions, format of description and alternate and exception paths. See my comments on some selected use cases below for specific examples.]

### **3.2.1** **Sign Up of General Users**

| **Identifier** | | UC-001 | |
| --- | --- | --- | --- |
| **Purpose** | | The Users register their account on the website. | |
| **Pre-conditions** | | The *User* should have a valid email address.  The *User* must not already have an account. | |
| **Post-conditions** | | The Sign Up is successful and the account is created.  If User already has an account, they will be directed to the login prompt. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The user clicks on the sign-up button for pet owners. | |  |
| 2. | The system redirects the user to the sign up page. | |  |
| 3. | The system prompts the user with a form to enter his personal information including name, contact, date of birth, email address, and asks the user to upload a profile picture . | |  |
| 4. | The user enters his personal information including name, contact and date of birth. | |  |
| 4. | The user enters his email address. | |  |
| 5. | The user uploads his profile picture. | |  |
| 6. | The user submits the form. | |  |
| **7.** | The system validates the form and if the user has entered the information correctly then the account is created. | |  |
| 8. | The use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | After step 1 the user can choose the signup with Google option and go directly to step 7. | |  |
|  | **Exception Paths** | |  |
|  | In step 7, if the user enters invalid information or if the email address is already registered an error message is shown. | |  |

### **3.2.2** **Sign Up of Service Providers**

| **Identifier** | | UC-002 | |
| --- | --- | --- | --- |
| **Purpose** | | The Service Providers register their account on the website. | |
| **Pre-conditions** | | The *Service Providers should have a valid company letterhead and email address.* | |
| **Post-conditions** | | The Sign Up is successful and the account is created. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The Service Provider clicks on the sign-up button for Services. and is redirected to the sign up page. | |  |
| 2. | The system redirects the service provider to its sign up page. | |  |
| 3. | The system prompts the service provider with a form to enter the relevant information, upload a profile picture, enter email address, username and password. | |  |
| 4. | The Service Provider enters their company information. | |  |
| 5. | The Service Provider uploads their profile picture. | |  |
| 6. | The Service Provider enters their email address. | |  |
| 7. | The Service Provider chooses their username and password. | |  |
| 8. | The Service Provider submits the form. | |  |
| 9. | The system validates the form and displays a pending verification message on the user’s screen. | |  |
| 10. | The system redirects the user to the home screen. | |  |
| 11. | The system forwards the sign-up request to the Admin team for verification. | |  |
| 12. | After verification the system sends the Service Provider a confirmation email. | |  |
| 10. | The use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | The admin team rejects the sign up request, and the system sends a rejection email to the service provider. | |  |
|  | **Exception Paths** | |  |
|  | In step 5, if the Service provider enters invalid information or if the email address is already registered an error message is shown. | |  |

### **3.2.3** **Sign Up of Veterinary Doctors**

| **Identifier** | | UC-003 | |
| --- | --- | --- | --- |
| **Purpose** | | The Vets register their account on the website. | |
| **Pre-conditions** | | The *Vets should have a valid medical license and email address.* | |
| **Post-conditions** | | The Sign Up is successful and the account is created. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The Vet clicks on the signup button for Vets. | |  |
| 2. | The system redirects the vet to its sign up page. | |  |
| 3. | The system prompts the vet with a form to enter medical information, email address, username, password and upload a profile picture. | |  |
| 4. | The Vet enters their medical information. | |  |
| 5. | The Vet uploads their profile picture. | |  |
| 6. | The Vet enters their email address. | |  |
| 7. | The Vet chooses their username and password. | |  |
| 8. | The Vet submits the form. | |  |
| 9. | The system validates the form and displays a pending verification message on the vet’s screen. | |  |
| 10. | The system redirects the vet to the home screen. | |  |
| 11. | The system forwards the Sign-up request to the Admin team for verification. | |  |
| 12. | After verification the system sends the Vet a confirmation email. | |  |
| 10. | The use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | The admin team rejects the sign up request, and the system sends a rejection email to the vet | |  |
|  | **Exception Paths** | |  |
|  | In step 5, if the Vet enters invalid information or if the email address is already registered an error message is shown. | |  |

### **3.2.4** **Sign Up of Rescue Services**

| **Identifier** | | UC-004 | |
| --- | --- | --- | --- |
| **Purpose** | | The Rescue Services register their account on the website. | |
| **Pre-conditions** | | The *Rescue Services should have a valid organization and email address.* | |
| **Post-conditions** | | The Sign Up is successful and the account is created. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The Rescue Service clicks on the signup button for Vets. and is redirected to the sign-up page. | |  |
| 2. | The system redirects the rescue service to the sign up page. | |  |
| 3. | The system prompts the rescue service with a form to enter company information, email address, username, password and upload a profile picture. | |  |
| 3. | The Rescue Service enters their company information. | |  |
| 3. | The Rescue Services upload their profile picture. | |  |
| 4. | The Rescue Service enters their email address. | |  |
| 5. | The Rescue Service chooses their username and password. | |  |
| 6. | The Rescue Service submits the form. | |  |
| 7. | The system validates the form and displays a pending verification message on the rescue service’s screen. | |  |
| 8. | The system redirects the rescue service to the home screen. | |  |
| 9. | The system forwards the Sign-up request to the Admin team for verification. | |  |
| 10. | After verification the system sends the Rescue Service a confirmation email. | |  |
| 11. | The use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | The admin team rejects the sign up request, and the system sends a rejection email to the rescue service. | |  |
|  | **Exception Paths** | |  |
|  | In step 6, if the Rescue Service enters invalid information or if the email address is already registered an error message is shown. | |  |

### **3.2.5** **Login**

| **Identifier** | | UC-005 | |
| --- | --- | --- | --- |
| **Purpose** | | The Users can Login to their account. | |
| **Pre-conditions** | | The *Users should have already registered their account.*  The Users should be logged out at the time of they wish to login. | |
| **Post-conditions** | | The Users login and are redirected to their dashboard. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The user clicks on the login button | |  |
| 2. | The system redirects the user to the login page | |  |
| 3. | The system prompts the user with a form to enter their username and password. | |  |
| 4. | The User enters their Username. | |  |
| 5. | The User enters their password. | |  |
| 6. | The user clicks on login | |  |
| 7. | The system validates the form and if the credentials match it redirects the user to the dashboard | |  |
| 10. | The use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | In case of failed validation, the user can click on the forgot password button. | |  |
|  | **Exception Paths** | |  |
|  | In step 3, if the Username/Password combination is incorrect an error message is shown. | |  |

### 

### 

### **3.2.6** **Logout**

| **Identifier** | | UC-006 | |
| --- | --- | --- | --- |
| **Purpose** | | The Users can Log Out from the system. | |
| **Pre-conditions** | | The *Users should already be logged in.* | |
| **Post-conditions** | | The Users are logged out and redirected to the home page. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The User clicks on the Logout Button | |  |
| 2. | The system logs the user out. | |  |
| 2. | The system redirects the user to the home page. | |  |
| 3. | The use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  |  | |  |
|  | **Exception Paths** | |  |
|  |  | |  |

### **3.2.7** **Change Password**

| **Identifier** | | UC-007 | |
| --- | --- | --- | --- |
| **Purpose** | | The Users can change their password. | |
| **Pre-conditions** | | The *Users should know their old password.* | |
| **Post-conditions** | | The password is changed | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The User navigates to the Edit profile page | |  |
| 2. | The User clicks on the change password button. | |  |
| 3. | The system redirects the user to the password change page. | |  |
| 4. | The system prompts the user with a form asking for their old password and new password. | |  |
| 5. | The User enters their old password. | |  |
| 6. | The User enters a new password. | |  |
| 7. | The system validates the form, updates the password in the database and redirects the user to the password change successful page. | |  |
| 8. | The use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | If the form is not valid and the credentials are incorrect, the system displays an error message to the user. | |  |
|  | **Exception Paths** | |  |
|  | In step 5, if the old Password is incorrect an error message is shown. | |  |

### **3.2.8** **Reset Password**

| **Identifier** | | UC-008 | |
| --- | --- | --- | --- |
| **Purpose** | | The Users can reset their password in case they forget. | |
| **Pre-conditions** | | The *Users should know their registered email address.* | |
| **Post-conditions** | | The password is changed. The previous password should not work. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The User clicks on the forgot password button on the home page. | |  |
| 2. | The system redirects User to the password reset page. | |  |
| 3. | The User enters his email address. | |  |
| 4. | The User gets an email link to reset their password. | |  |
| 5. | The User clicks on the link and is redirected to the change password page. | |  |
| 6. | The User enters a new password. | |  |
| 7. | The password is validated and reset. | |  |
| 8. | The User is redirected to the homepage. | |  |
| 7. | The use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | In step 7, if the user enters an invalid password or the old password, an error message will be shown and User will be prompted to re-enter the password | |  |
|  | **Exception Paths** | |  |
|  | In step 2, if the email address is incorrect an error message is shown. | |  |

### **3.2.9** **Create Blog Posts**

| **Identifier** | | UC-009 | |
| --- | --- | --- | --- |
| **Purpose** | | The Users can create a new blog post. | |
| **Pre-conditions** | | The *User* should be present on the Blog Page. | |
| **Post-conditions** | | A new Blog post is created. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The user clicks on the Add Post option. | |  |
| 2. | The system redirects the user to an Add New Post page. | |  |
| 2. | The user enters text and/or images. | |  |
| 3. | The user clicks on the submit button. | |  |
| 4. | The System shows a “Post Successfully added” message. [The description should make it clear that system shows this message. ] Fixed | |  |
| 5. | The post is published. | |  |
| 6. | The user is redirected to the Blog Page. | |  |
| 7. | The use case ends. | |  |
|  | [How is it published without approval? You should mention that Approval use case is completed and then the post is published.] Fixed | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  |  | |  |
|  | **Exception Paths** | |  |
|  | [What happens in this case? Can a user resubmit after editing?] Fixed | |  |

### **3.2.10** **Edit Blog Post**

| **Identifier** | | UC-010 | |
| --- | --- | --- | --- |
| **Purpose** | | The Users can edit their Blog Post. | |
| **Pre-conditions** | | The *Post should already be published.* | |
| **Post-conditions** | | The Post is updated. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The user clicks on the edit post button. | |  |
| 2. | The system redirects the user to an edit blog post page | |  |
| 2. | The user makes changes to their Blog Post. | |  |
| 4. | After making the changes the user clicks on save changes | |  |
| 5. | The system redirects the user to the home page with a success message that the edit has been successful. | |  |
| 6. | The system updates the database | |  |
| 7. | The use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  |  | |  |
|  | **Exception Paths** | |  |
|  | In step 4, if the post is not approved the user receives a message in their inbox. | |  |

### **3.2.11** **Chat (TENTATIVE)**

| **Identifier** | | UC-011 | |
| --- | --- | --- | --- |
| **Purpose** | | The Users can chat with other users in the in-app messenger. | |
| **Pre-conditions** | | The *User should be logged in to their account.* | |
| **Post-conditions** | | The Users can chat with each other. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The user clicks messenger tab | |  |
| 2. | The system redirects the user to the messenger page. | |  |
| 3. | The user clicks on the new message button  or opens an existing chat. | |  |
| 4. | The system redirects the user to the compose new message page. | |  |
| 5. | The user has the option to send texts, images and videos. | |  |
| 3. | The user can chat with the desired recipient. The user can only chat with one person. There is no group chat.  [Will it just be text only chat or will the users be allowed to send images, videos as well? Can it be a group chat?] | |  |
| 4. | The use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
| 3. | Continued from step 3:  The user clicks on an existing chat  The system redirects the user to an existing chat page.  Continue from step 5. | |  |
|  | **Exception Paths** | |  |
|  |  | |  |

### **3.2.12** **Comment on Blog Posts**

| **Identifier** | | UC-012 | |
| --- | --- | --- | --- |
| **Purpose** | | The Users can comment on Blog Posts. | |
| **Pre-conditions** | | The *User* should be present on the Blog Post they want to comment on. | |
| **Post-conditions** | | The User must be able to see their comment posted under the post. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The user clicks on the Add comment button. | |  |
| 2. | The system opens a text box for the User to type in. | |  |
| 3. | The user types in their comment. | |  |
| 4. | The user clicks on the post comment button. | |  |
| 5. | The system shows a “comment successfully added” message. | |  |
| 6. | The comment is posted | |  |
| 7. | The use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | None | |  |
|  | **Exception Paths** | |  |
|  | None | |  |

### **3.2.13** **Review and Rate Services**

| **Identifier** | | UC-013 | |
| --- | --- | --- | --- |
| **Purpose** | | The Users can review and rate services | |
| **Pre-conditions** | | The *user* should have availed the service.  The *user* must be present on the service profile. | |
| **Post-conditions** | | The User’s review and rating is updated on the service profile. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The user clicks on the review button/field. | |  |
| 2. | The system opens a text box for the user to type in. | |  |
| 3. | The user types in their review.  The user rates the service out of 5 using stars. | |  |
| 4. | The user clicks on the post review button. | |  |
| 5. | The system shows a “successfully posted review” message to the user. | |  |
| 6. | The user review and rating are updated on the service profile. | |  |
| 7. | Use case ends. | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | If the user has not availed the service, they will not be able to click on the review and ratings button. | |  |
|  | **Exception Paths** | |  |
|  | None | |  |

### **3.2.14** **Report User or Service**

| **Identifier** | | UC-014 | |
| --- | --- | --- | --- |
| **Purpose** | | The User can report inappropriate content posted by other users or service providers on the blog. | |
| **Pre-conditions** | | * The user should be logged in and viewing the post. * Precondition removed as per instructions[This is not a pre-condition, it is a step performed by the actor in the use case.] | |
| **Post-conditions** | | The Users review and rating is updated on the service profile. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The user clicks on the report button/field on a post. | |  |
| 2. | The system redirects the user to the report page. | |  |
| 3. | The system prompts the user with a form to enter the reason for reporting | |  |
| 4. | The user types in the reason for reporting. | |  |
| 5. | The user clicks on the submit report button. | |  |
| 6. | The system validates the form. | |  |
| 7. | The system updates the database with the report | |  |
| 8. | The system forwards the report to the administrator | |  |
| 9. | The system redirects the user after a success message that the report has been submitted | |  |
| 10. | Use Case ends | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  |  | |  |
|  | **Exception Paths** | |  |
|  |  | |  |

### **3.2.15** **Approve Service Provider/Vet/Rescue Service Profile (Admin)**

| **Identifier** | | UC-015 | |
| --- | --- | --- | --- |
| **Purpose** | | The Admin can Approve or Reject account registration requests | |
| **Pre-conditions** | | * The admin should be logged in to their dashboard | |
| **Post-conditions** | | The New account is registered. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The admin clicks on view registration request | |  |
| 2. | The system displays the registration request to the admin (redirects the admin to the request page. | |  |
| 3. | The admin clicks on approve request | |  |
| 4. | The system displays the message to the admin that request approved and redirects the admin to the registration requests page. | |  |
| 5. | The account is registered. | |  |
| 6. | The system sends an email confirmation to the user. | |  |
| 7. | Use Case ends | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | Continuing from step 3:  The admin clicks on reject request  The system displays the message to the admin that request is rejected and redirects the admin to the registration requests page.  The account is not registered  Use case end | |  |
|  | **Exception Paths** | |  |
|  |  | |  |

### **3.2.16** Search on Blog

| **Identifier** | | UC-016 | |
| --- | --- | --- | --- |
| **Purpose** | | The user can search for relevant stuff on the blog | |
| **Pre-conditions** | | None | |
| **Post-conditions** | | None [This is part of the use case] | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The user clicks on the search field. | |  |
| 2. | The user types in query or keywords | |  |
| 3. | The system uses the typed in query to search and filter the database | |  |
| 4. | The system presents the fetched data to the user | |  |
| 5.. | Use Case ends | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | Removed as instructed[It is not an alternate course of action. Is it really an error?] | |  |
|  | **Exception Paths** | |  |
|  | If the system does not find any matches, it displays a message to the user that no results found | |  |

### **3.2.17** **Search Marketplace**

| **Identifier** | | UC-017 | |
| --- | --- | --- | --- |
| **Purpose** | | The user searches for a specific service, or vet in the marketplace | |
| **Pre-conditions** | | The user should be logged into the account  The user should be in the marketplace | |
| **Post-conditions** | | Search results are filtered according to the user’s query and displayed | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The user clicks on the search field | |  |
| 2. | The user types in query or keywords | |  |
| 3. | The system uses the typed in query to search and filter the database | |  |
| 4. | The system presents the fetched data to the user | |  |
| 5.. | Use case ended | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | Continued from step 4:  The user has the option to filter based on location  The user clicks on filter by location  The system filters the results by location  The system displays the results to the user  Use case ended | |  |
|  | **Exception Paths** | |  |
|  |  | |  |

### **3.2.18** **Add User Profile (Admin) (TENTATIVE)**

| **Identifier** | | UC-018 | |
| --- | --- | --- | --- |
| **Purpose** | | Admin can add a user profile | |
| **Pre-conditions** | | The admin should be logged into the admin account  The admin should have valid reason for this use case | |
| **Post-conditions** | | User profile will be added | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The admin goes to the user table / tab | |  |
| 2. | The admin clicks on add user button | |  |
| 3. | The system redirects the admin to the add user page | |  |
| 4. | The system prompts the admin to fill in the details of the user. | |  |
| 5. | The admin fills in the details of the user | |  |
| 6. | The admin finalizes adding user and clicks on add confirmation button | |  |
| 7. | The system updates the New user data in the database | |  |
| 8. | The system displays a success message to the admin and redirects the admin back to the admin page | |  |
| 9. | Use case Ends | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  | Continuing from step 6:  The user with that username and password already exists.  The system displays an error message to the admin and prompts him to try again with another username and password. | |  |
|  | **Exception Paths** | |  |
|  | None | |  |

### **3.2.19** **Edit User Profile (Admin) TENTATIVE**

| **Identifier** | | UC-019 | |
| --- | --- | --- | --- |
| **Purpose** | | Admin can edit a user profile. | |
| **Pre-conditions** | | The admin should be logged into the admin account.  The admin should have a valid reason for this use case. | |
| **Post-conditions** | | User profile will be edited. | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The admin goes to the user table / tab | |  |
| 2. | The admin clicks on edit user button of the desired user | |  |
| 3. | The admin edits the required details of the user | |  |
| 4. | The admin finalizes editing details and clicks on confirmation button | |  |
| 5. | New user data is updated in the database | |  |
| 6. | Use case Ends | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  |  | |  |
|  | **Exception Paths** | |  |
|  |  | |  |

### **3.2.20** **Delete User Profile (Admin)**

| **Identifier** | | UC-020 | |
| --- | --- | --- | --- |
| **Purpose** | | Admin can delete a user profile | |
| **Pre-conditions** | | The admin should be logged into the admin account  The admin should have valid reason for this use case | |
| **Post-conditions** | | User profile will be deleted | |
|  | | | |
|  | **Typical Course of Action** | |  |
| 1. | The admin goes to the user table / tab | |  |
| 2. | The admin clicks on delete user button of the desired user | |  |
| 3. | The system prompts the admin with a confirmation message to delete user | |  |
| 4. | The admin clicks on yes upon the confirmation message | |  |
| 5. | The system deletes the desired user from the database | |  |
| 6. | Use case Ends | |  |
|  | | | |
|  | **Alternate Courses of Action** | |  |
|  |  | |  |
|  | **Exception Paths** | |  |
|  |  | |  |

# 

# 

# **4.** **Class Diagram**

## Missing classes for reviews, blogs.

* Missing classes for entities that enable approval process.

## Missing classes for authorization. Each user must have a role having certain rights. It should be possible to grant and revoke access rights on a need basis.

## 

## 

## 

## 

## 

## 

## 

## 

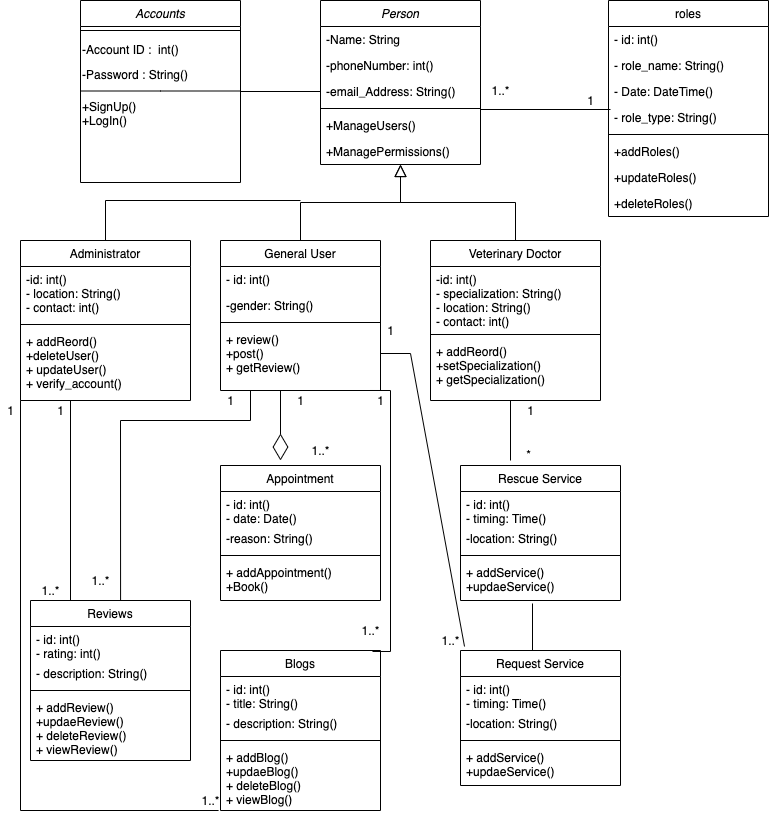
## 

## 

## 

## 

## **4.1** **Diagram**



## 

## **4.2** **Description**

This Class Diagram contains most of the classes [These cannot be all classes. There would be classes such as UI classes, data related classes, and others] that comprise the system. The Doctor, User, Service Provider and Admin all will inherit the attributes by their parent class, Person, since they have the common attributes and methods that are defined in the Person class. The other classes like the appointment class will have methods related to the appointment of the doctor and the Pet owner, like the detail of the issue for which they need an appointment. And last but not the least the Accounts class which will deal with the security and the privacy and have attributes such as account IDs and passwords. This class will also take care of signup/login and service registration approvals. The class diagram is providing a high-level idea and abstraction of how the many different classes interact and function. Moreover the sequence diagrams in the following section provide details about the flow of data in specific use case scenarios.

# 

# **5.** **Sequence Diagrams**

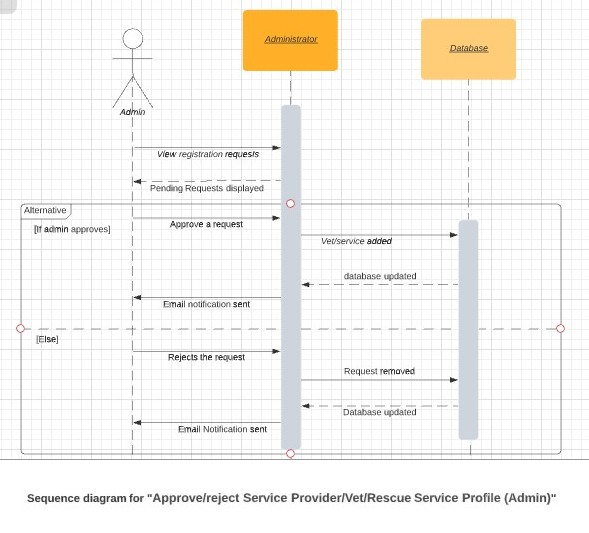
* You have not created diagrams for most of the important use cases.
* At least, some of the classes in the sequence diagrams should be from the class diagram as well. You used no class from the class diagram.
* Your diagrams give minimal details. No parameters are included.
* There are formatting problems with diagrams as well.
* I suggest you to refresh your concepts about sequence diagrams.

# **5.1** **Signup of general user:**

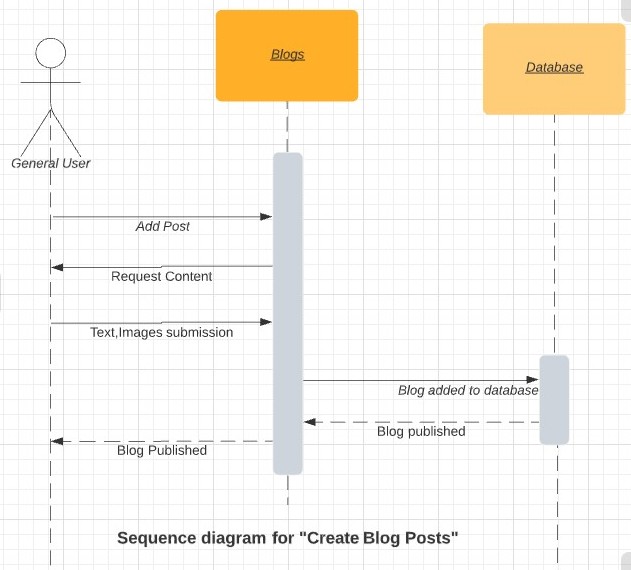
# 

# 

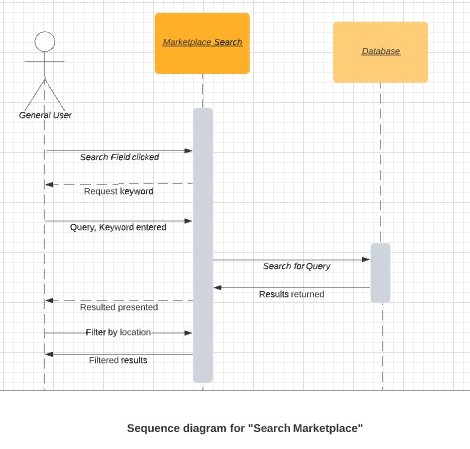
## **5.2** **Approve Service Provider/Vet/Rescue Service Profile (Admin)**



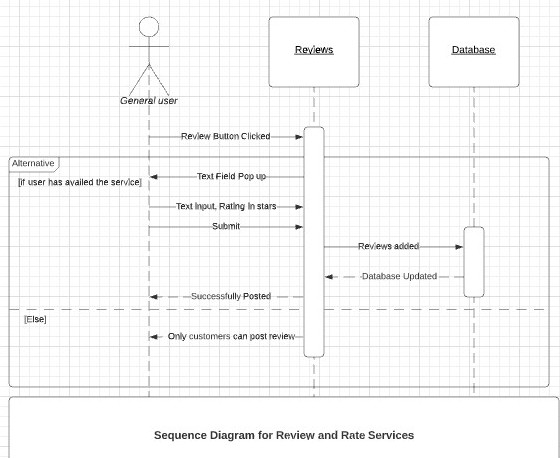
**5.3** **Create Blog Posts**

****

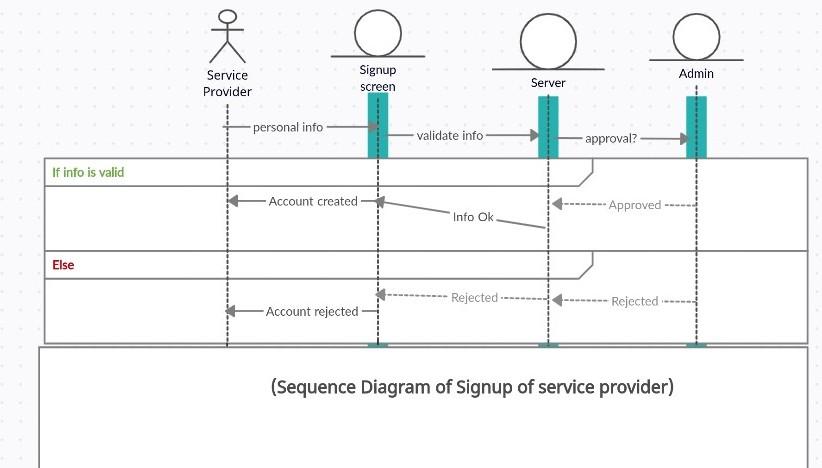
**5.4 Search Marketplace**

****

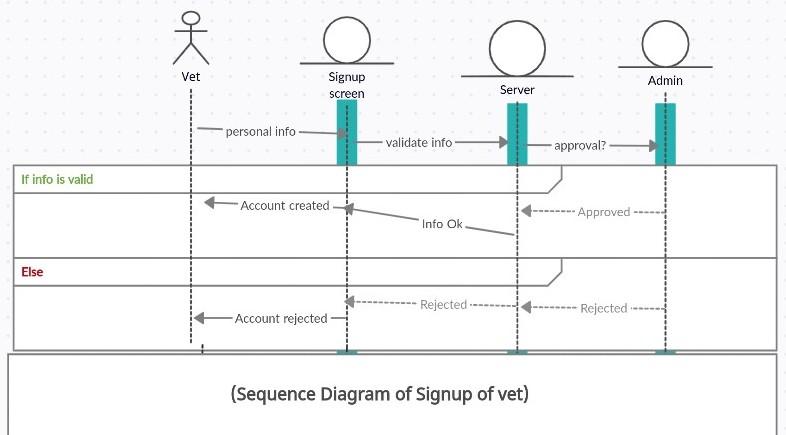
### **5.5 Review and Rate Services**



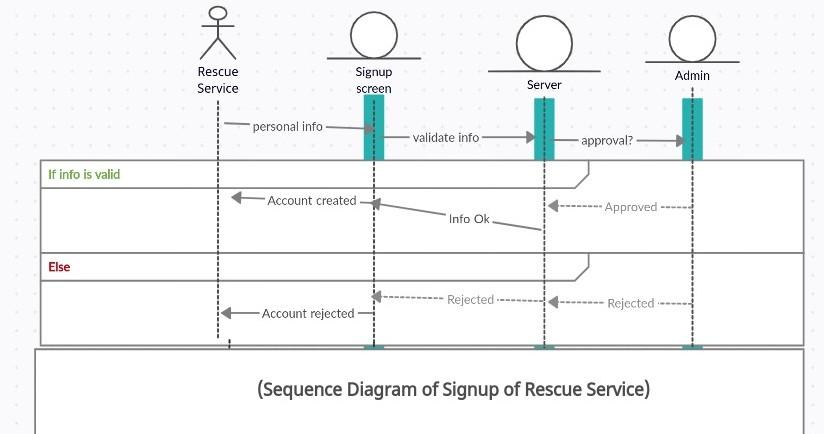
## **5.6** **Signup of Service Provider**



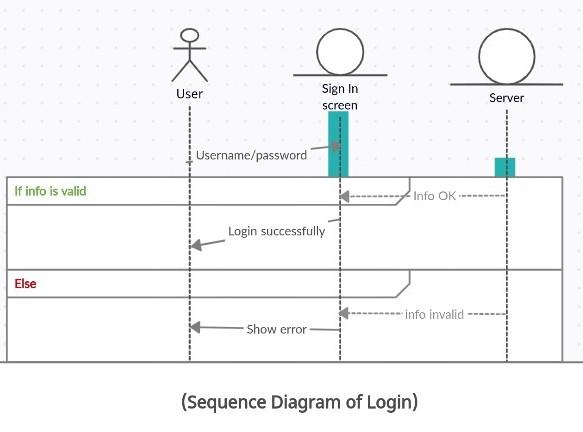
**5.7** **Signup of Vet**

****

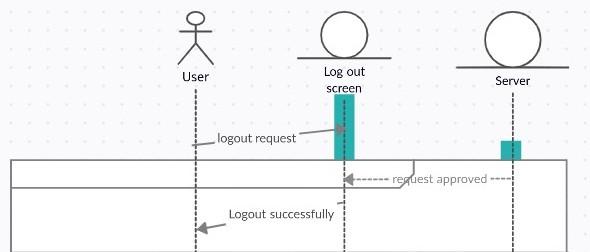
**5.8** **Signup of Rescue Services**

****

**5.9** **Login**

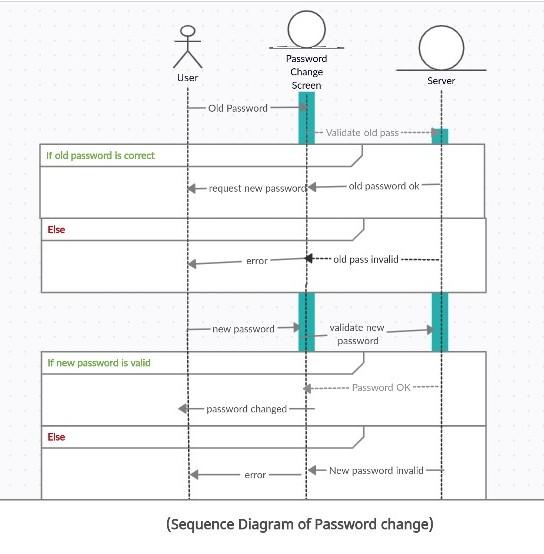
****

**5.10** **Logout**

****

# 

**5.11** **Password Change**

****

# 

# 

# 

# 

# **6.** **State Diagrams**

## **6.1** **Diagram details (Search and Add to Cart)**

General user performs the search on the marketplace and adds items to the cart.

## **6.2** **Diagram (Search and Add to Cart)**

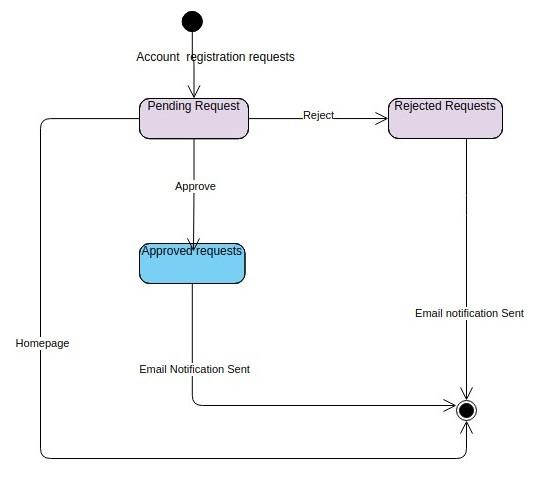
# 

# (Search Marketplace State Diagram)

**6.3** **Diagram details (Approve Vet/Service Registration)**

Admin Approves or rejects the pending account registration requests from Vet/Service Provider/Rescue Team.

## **6.4** **Diagram (Approve Vet/Service Registration)**

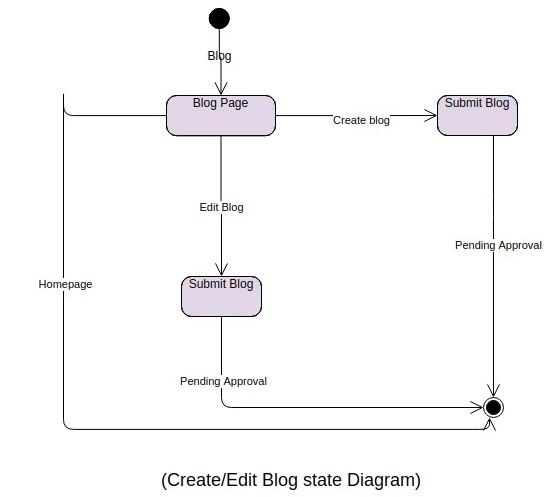


(Admin Pending accounts registration state diagram)

**6.5** **Diagram details (Create/Edit Blog Posts)**

This state diagram shows the general user creating/editing blog posts and submitting for approval.

## **6.6** **Diagram (Create/Edit Blog Post)**



# 

# 

# 

# **7.** **Non-functional Requirements / Quality Attributes**

| **Sr#** | **Requirements** |
| --- | --- |
| 1 | The system will process search requests in less than 3 seconds. |
| 2 | The system should have a secure database and should be protected from SQL injection attacks using form data input validation, parameterized queries, etc. This should prevent breach in sensitive data and will provide a more secure experience to our users. |
| 3 | Modified data in a database should be updated for all users accessing it within 3 seconds. If any data variable has been modified, the webpages for all the users should reflect that within 5 seconds. |
| 4 | The system should secure user identity and preventative measures should exist against potential security breach. The passwords will be hashed, and password change functionality will be provided to the users. The user will be able to contact the admin and disable his/her account in case of breach. |
| 5 | Anonymous users are not allowed to access and manipulate data. Only authenticated users are allowed to manipulate and access the database. |
| 6 | SSL Certification must be satisfied, which secures an internet connection and safeguards data which is being sent between server and browser. |
| 7 | The system must be fully operational at all times unless scheduled maintenance is required for which downtime will be pre-specified. |
| 8 | Users must login and be authorized in order to access their account. Admins and every other actor must also follow the authentication procedure in order to access their respective accounts. |
| 9 | Admins will have administrator privileges and the most access rights on the system. Every actor will act within its domain and cannot access other rights. Administrator will be responsible for granting or revoking these rights. |
| 10 | The system must automatically log out all customers after a period of inactivity. |
| 11 | Users can only place orders once they provide their relevant information. |
| 12 | The system shall not be shut down for maintenance more than once in a 24-hour period |
| 13 | The system shall provide a uniform look and feel between all the web pages. |
| 14 | The system should have an easy-to-use GUI, so that all functionality is self-explanatory and intuitive. |

**7.1 Quality Attributes**

*Availability*

The most critical components of the system should be always available; hence the user can access it using a web browser at any time. The only exception to availability is during down time of the server on which the system runs.

1. In the exceptional case of database corruption or hardware malfunction, a replacement page will be shown.
2. Unless the system is non-operational, the system shall present the user with a notification informing them that the system or one of its parts is unavailable.

*Reliability*

The system must continuously be able to consistently perform the specified functions without failure.

1. The inventory update process shall roll back all related updates when any update fails to commit.
2. The authorization transaction match process shall require a 100-percent match to post a transaction.
3. Reliability decreases because of bugs in the code hence we will ensure through rigorous testing that our code has minimum (if any) bugs.

*Maintainability*

The system design and implementation must conform to industry standards of development practices.

1. Maintenance developers must be able to understand the code with ease and be able to modify or change functionality as required.
2. The above-mentioned ease of understanding may be achieved by using extensive comments in the codebase to explain every step.
3. The code structure should be modular so that every module has a very specific and simple function.
4. The system shall not be shut down for maintenance more than once in a 24-hour period.

*Scalability*

The system should be able to expand its processing capabilities upward and outward to support business growth. This means that the system should be able to serve more users, process more data, and do more transactions at a time.

1. The effort needed to administer the inventory (as measured in hours per month of system administrators’ time) shall not increase with an increase in the number of products. If there is a significant increase in system operation work, it shall be proportionately less than an increase in the number of products.
2. The inventory and product display system shall be scalable to support unlimited growth in the number of products.
3. The account management system shall support unlimited customer, account, and transaction relationships.

*Usability*

The system should be easy for the user to learn, operate, prepare inputs, and interpret outputs through interaction.

1. The system shall provide a uniform look and feel between all the web pages.
2. The system shall be able to be used by adult members of the public without training.
3. The product shall be self-explanatory and intuitive.
4. The system shall provide a digital image for each product in the product menu.
5. The system shall be evaluated after testing and feedback and improvements will be made, if need be, to ensure the system is at least as easy to use as any existing similar system.
6. A user must be able to accomplish a particular task in a minimum number of steps.

*Correctness*

The system must be error free; we aim to achieve this through extensive testing toward the end of the development process.

1. During the development phase, the team will test each functionality to see if desired output is received.
2. Once the front end is developed, the team will test the system for any errors including any problems that may arise based on user experience.
3. Once the front and back end is linked, the team will test the entire system and cater to any error and changes that may need to be made.
4. Once the system is complete, a beta version will be provided to potential clients to test out.

# 

# 

# 

# **8.** **Who Did What?**

| **Name of the Team Member** | **Tasks done** |
| --- | --- |
| Muhammad Aaish Javed | Introduction, Quality Attributes, Use Case Table, Use Cases, Non-Functional Requirements, |
| Muhammad Ibrahim Bhalli | Introduction, Table of Contents, Use Case Table, System Actors, Use Cases |
| Syed Raza Abbas | System Actors, Use Cases, Use Case Diagram, Class Diagram |
| Muhammad Tayyab | Use Case Diagram, Sequence Diagrams, State Diagrams |

**9.** **Review checklist**

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

| **Section** **Title** | **Reviewer Name(s)** |
| --- | --- |
| Use Cases | Muhammad Aaish Javed, Muhammad Ibrahim Bhalli, Syed Raza Abbas, Muhammad Tayyab |
| Introduction | Muhammad Ibrahim Bhalli, Syed Raza Abbas |
| Non-Functional Requirements | Muhammad Ibrahim Bhalli, Muhammad Tayyab, Syed Raza Abbas, Muhammad Aaish Javed |
| Class, Sequence, State Diagrams | Muhammad Aaish Javed, Muhammad Ibrahim Bhalli, Muhammad Tayyab |