Petswala -SPROJ Report

Roshan Aziz 22100298

Adil Aslam Chaudry 22100004

Mohid Yousaf 22100201

Advisor: Waqar Ahmad

School of Science and Engineering

Lahore University of Management Sciences

**Submission Date: 30-4-22**

**Acknowledgement and Dedication**

**Certificate**

I certify that the senior project titled “**Add project title here**” was completed under my supervision by the following students:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

and the project deliverables meet the requirements of the program.

------------------------------------- Date:

**Advisor (Signature)**

------------------------------------- Date:

**Co-advisor (if any)**

# 

# **Table of Contents**

# 

[**Table of Contents**](#_bvd6xrxj5uxv) **3**

[**1. Introduction**](#_8y3qhnkcsbvn) **6**

[a. Introduction](#_xd7nkv7q0q5) 6

[b. Objective and Scope](#_ia4uj5dkjy2f) 6

[c. Development Methodology](#_8lo11bfrdgcb) 6

[d. Contributions](#_xza44816hvmg) 7

[**2. System Requirements**](#_w3xhbslyoa7b) **7**

[a. System Actors](#_lqopvqgusqlp) 7

[b. Functional Requirements](#_tttmor5ml6qy) 9

[c. Non-functional Requirements](#_gg508a5825jz) 12

[**3. System Architecture**](#_gt9ryhqeh6h2) **14**

[a. Architecture Diagram](#_7rx9ngrbux7r) 14

[b. Architecture Description](#_k222h0x27tlm) 15

[c. Justification of the Architecture](#_4ud8hi8jfor) 17

[d. Tools and Technologies](#_f3xg301nd9q8) 18

[**4. Requirements Specifications**](#_2fpuostpb4sf) **19**

[a. Use Cases](#_mmpv933ltnb4) 19

[Use Case Diagram](#_frkbkf3ia4zq) 19

[Use Case Descriptions](#_91agbca4vqpi) 20

[2.1 Report Stray Animals](#_ds3pu8kg5dr0) 20

[2.2 Find Veterinary Doctors](#_3vmludfe5tzs) 21

[2.3 Receive Rescue Request](#_99f3lurt730t) 23

[2.4 Buy Pet Accessories](#_2kcwdn687nsv) 24

[2.5 Get in contact with the Customer](#_q6sp4cw8czuy) 26

[2.6 Give Location Coordinates](#_luo2ywdwf8f9) 27

[2.7 Set up Store](#_qdvcvy8uxk0d) 29

[2.8 Upload Pictures of Pets](#_46mt2b7docr2) 30

[2.9 Updating Profile](#_qslruyb4i6rg) 32

[2.10 Sell Pets](#_fb27h2hw4x16) 33

[b. Class Diagram](#_m6u64mmfyhqj) 35

[Diagram](#_tefnmfluo35) 35

[Description](#_xa7w3g47yw5l) 35

[c. Sequence Diagrams](#_f5fgj236gxi6) 37

[5.1 Sell Accessories](#_2kyi7kgu9h90) 37

[5.2 Respond via Messages](#_i6lzz66eol3r) 38

[5.3 Login](#_1g9m4g34ggl) 39

[5.4 Report Stray Animals](#_2avvb8n292xh) 40

[5.5 Receive Rescue Request](#_4xeehx9x4zzg) 41

[5.6 Buy Pet Accessories](#_gaxwio1scam8) 42

[5.7 Get in Contact with the Customer](#_cvl4y5gtn5xf) 43

[5.8 Upload Pictures of Pets](#_iu6kady1f1h8) 44

[5.9 Update Profile](#_9nyx2yp49f58) 45

[5.10 Sell Pets](#_9l7363aa87fv) 46

[**5. Software Development Methodology and Plan**](#_12fqxpynu57m) **47**

[a. Software Process Selection](#_dcataxgw08b) 47

[Model Comparison](#_swfm9vltqegc) 47

[Selection of Process for Project Development](#_bmk2ewrdzwa9) 48

[b. Gantt Chart](#_penq935wbzie) 49

[**6. Database Design and Web Services**](#_b1qoqz9hhnca) **50**

[a. Database Design](#_vuc671vodqx6) 50

[Diagram](#_tq9dizehd3k9) 50

[Description](#_q7i281aul2t8) 50

[b. API Specification](#_44eqyna2ey5a) 52

[Signup](#_h0l35ddve2mc) 53

[Home](#_7pshsxox8spl) 54

[Adopting a Pet](#_bz8o9y5al9te) 54

[Check Your NewsFeed](#_a3qbk3g0hnmj) 55

[Make a Post](#_690nn94yrhi4) 55

[Finding a Service](#_xcdvdk4ianlg) 56

[Post Rescue Request](#_tygzzxeo1b9a) 56

[View Purchased Items](#_o6j887wte4cw) 57

[Edit Profile Picture](#_407d4t2159vl) 58

[Edit Profile Info](#_6ur3nonjkm9o) 58

[Access Seller Homepage](#_neowg4rik7ej) 59

[See Product List (As Seller)](#_h4r6fehkbc0p) 59

[**8. Project Security**](#_n6g7n7vfjba5) **60**

[a. Project Threats](#_be3nq3qtx41k) 60

[Sensitive Data Exposure](#_e48zml902ajf) 60

[Insecure Communication](#_tmbun46bakn7) 61

[b. Potential Losses](#_wbv8ntdvygu0) 61

[User Monetary and Data Losses](#_syps2huh38ad) 61

[Business Reputation Loss](#_x5gqsegccf23) 61

[Litigation Losses](#_q732l3h0pljl) 61

[c. Security Controls](#_56ffbfolxo91) 61

[d. Static and Dynamic Security Scanning Tools](#_ggaewx4trg28) 63

[Dynamic Testing](#_mlzi4hma7d1q) 63

[Bloc\_Test With Flutter](#_lnpc1tj63yxs) 63

[OWASP ZAP](#_l09ivwxzsp2b) 63

[Static Testing](#_3kuawba4bp4q) 63

[Dart Analysis](#_kybntojcjkse) 63

[NodeJSScan](#_jtik5k3a6b19) 63

[**9. Risk Management**](#_9y2cj474ojhc) **64**

[Potential Risks and Mitigation Strategies](#_7tv2r34s2a5b) 64

[**10. Testing and Evaluation**](#_ichg8tu8r43m) **66**

[Test Case 1](#_lm27fphcwhx7) 66

[Test Case 2](#_hitu0pgg8k7e) 67

[Test Case 3](#_h6fkyukh8g6g) 67

[Test Case 4](#_wl7skwm3tjj9) 68

[**11. Deployment Guidelines**](#_khjup95yt0gu) **68**

[**12. Conclusion**](#_qu6h16ye0854) **69**

[a. Summary](#_uj8bgpu7uwaa) 69

[b. Challenges](#_u5f0vzdak866) 69

[c. Future](#_8b4va6rr64dp) 70

[**13. Review checklist**](#_mdtkfl4hknka) **70**

# 

# 

# 

# 

# **1.** **Introduction**

## 

## **a.** **Introduction**

This application is created to cater for the growing community in Pakistan. This community is one that revolves around animals and pets. The project comprises a mobile application catered for the community involving; pet owners, pet shops, sellers of pet accessories, veterinary doctors, pet rescue volunteers, and pet shelters/rescue teams. The application exists in the domain of a social media application, as well as an e-commerce one. The application aims to connect these individual community members and bring them together under one roof, this is done by providing certain functionalities, pet owners would be able to search for different accessories, as well as find good veterinary doctors for their pets, making their lives much more comfortable and better. They would also be able to interact with other pet owners and share photos and stories of their own pets as well. They could share helpful information with other pet owners. Any user of the app can also help with the animal rescue process by reporting pets to be rescued to the relevant rescue teams in the area.

## **b.** **Objective and Scope**

The objective of this application is to bring together members of a community and provide them with tools and platforms to perform certain functions related to that community. This application aims to provide a social media hub and e-commerce platform for all the members of the pets or animal community to interact with each other, pet owners and casual users can post, chat, buy pets and pet merchandise, while traders can also set up a store page to provide services or products. This is a relatively untapped market in Pakistan as of yet since there had not been a massive demand for pet-related services and stores before. However, this demand has increased recently. People are buying pets or even gifting them at an increasing rate and there are new services being provided locally for these people, but there is not a vast virtual platform to cater to their needs.

## **c.** **Development Methodology**

PetsWala is an android application that is build in flutter, we could implement the front end as well as backend in dart, however we build a RESTful API to separate the client and server logics, so we deployed this API in heroku, and all the requests from apk are handled through the deployed Server, in this way it makes it secure as well as make the project scalable. Similarly we set up our backend on Mongodb, so our RESTful API is connected with a Mongodb cluster named as ‘PetsWala’, and we used Mongoose and Express as middlewares to update all the user initiated information based on specific events to our database. All the external APIs are also deployed on respective servers, for instance we have OneSignal API for push notifications, GetStream API for chat system, Google Maps and Google Authentication API that are all configured in RESTful API.

## **d.** **Contributions**

The application that we have made distinguishes itself by a platform providing all the needed functions for our targeted demographic. The application provides a “one-stop” service for our customers to fulfill all their needs with regard to pets with just one app. Based on our research, there are no such applications providing all these functionalities under one roof, in such an accessible package.

# **2.** **System Requirements**

In this section, we will highlight the potential users of the system and highlight the needs of these customers that our app aims to fulfill. This section would elaborate on both the functional and non-functional requirements of our application.

## **a.** **System Actors**

| **Actor Name** | **Description** |
| --- | --- |
| Admin | The admin is in charge of managing the entire application. The admin can add/delete/edit/filter any of the profiles and data in the application. |
| Casual App Owners | They can register themselves and can access various facilities including veterinary opportunities and buying pets and accessories. They can also report pets to the rescue teams. They can also publish pet-related events in their surroundings. |
| Pet Owners | Those who are looking for guidance to treat their pets in the best possible way, can also rate the veterinary-related facilities and give their feedback and reviews. |
| Veterinary Doctors | They can register themselves on the platform, so pet owners can look for veterinary opportunities nearby. |
| Pet Stores | They are selling pets mainly, so people can search for the desired pets. |
| Pet Merchandise Shops | They are selling accessories required for pet care and grooming. |
| Pet Service Providers | These are service providers who help the pet owners with various tasks such as pet daycares, dog walkers, pet grooming, etc. |
| Pet Shelters | Pet shelters are places where the rescued pets are provided with a temporary home until they are adopted. A lot of the pet shelters have their own rescue teams. |

## 

## **b.** **Functional Requirements**

| **Requirements** | |
| --- | --- |
| **Sr#** | **Requirement** |
| 1 | As a Casual App Owner, I can register an account on the application. |
| 2 | As a Casual App Owner, I can report stray animals to the nearest rescue teams. |
| 3 | As a Casual App Owner, I can post any pet-related blogs or activities in the surrounding area. |
| 4 | As a Casual App Owner, I can also buy pets or any accessories. |
| 5 | As a Pet Store Owner, I want to set up my store on the platform so I can sell pets. |
| 6 | As a Pet Store Owner, I want to upload pictures and details of the animals and products we have. |
| 7 | As a Pet Store Owner, I want to be notified when an application user buys something. |
| 8 | As a Pet Store Owner, I want to display the details of my store including address, contact information, etc. |
| 9 | As a Pet Store Owner, I want to interact with my customers to convey order information or address complaints. |
| 10 | As an Admin, I want to approve or remove requests from service providers. |
| 11 | As an Admin, I want to add and remove veterinary doctors, pets stores and shelters from the app. |
| 12 | As an Admin, I need to be able to investigate the reporting of content that the application users do. |
| 13 | As a Veterinary Doctor, I want to register myself on the platform, so pet owners can look for veterinary opportunities nearby. |
| 14 | As a Veterinary doctor, I want to display or modify my contact information, address and other details on my profile. |
| 15 | As a Pet Owner, I want to get various veterinary recommendations that are best suited to me and I also want to rate them and give my feedback. |
| 16 | As a Pet Owner, I want to look for service providers and pet accessories stores near me. |
| 17 | As a Pet Owner, I want to report fake or scam service providers. |
| 18 | As a Pet Owner, I want to be able to chat in a safe environment with shop owners, service providers and vets. |
| 19 | As a Pet Owner, I should be able to create my own blog to post videos or articles. |
| 20 | As a Pet Service Provider, I can add, edit and delete services on my profile, like pet daycare, dog walking, dog grooming e.t.c. |
| 21 | As a Pet Service Provider, I should be able to see and respond to the reviews that are left on my services. |
| 22 | As a Pet Service Provider, I should be able to view if my services are needed in the area via contact information or posts. |
| 23 | As a Pet Shelter Provider, I can accept rescue requests and complete the rescue with my rescue team |
| 24 | As a Pet Shelter Provider, I will receive a notification with location, when a user puts up a rescue request. |
| 26 | As a Pet Shelter Provider, I should be able to share via Post, a successful rescue operation |
| 27 | As a Pet Shelter Provider, I can accept or reject adoption requests from interested users |
| 28 | As a Pet Merchandise Shop Owner, I can add accessories required for pet care and grooming on my page. |
| 29 | As a Pet Merchandise Shop Owner, I can sell these accessories from the page via cash on delivery services or providing shop location for in-person purchase. |
| 30 | As a Pet Merchandise Shop Owner, I want to check for reviews of my shop and respond to them. |
| 31 | As a Pet Merchandise Shop Owner, I need to be able to check for reviews of items that my shop sells. |
| 32 | As a Pet Merchandise Shop Owner, I need to be able to establish contact with the application user whether on phone or in-application chat. |

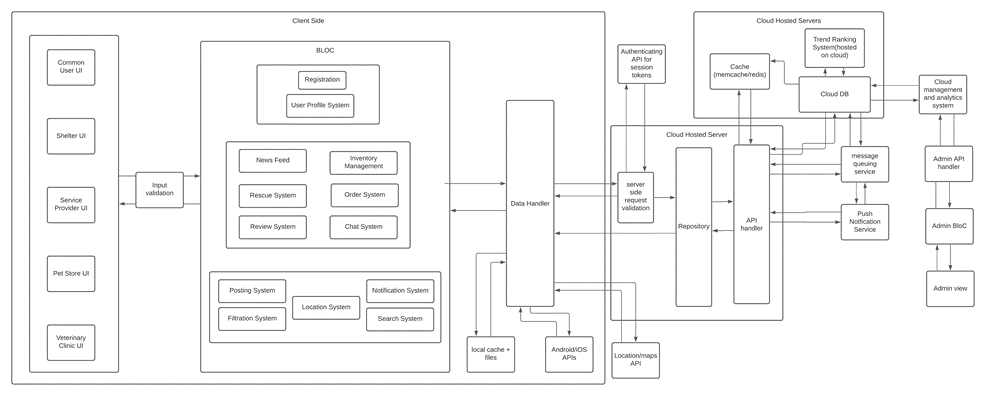
## **c.** **Non-functional Requirements**

| **Sr#** | **Requirements** |
| --- | --- |
| 1 | The system should not utilize more than 1 GB of memory at any time during its execution. |
| 2 | The system should not fail more than 3 times every 24 hours. In case of a failure, the system should restore to normal operations within 5 minutes of a failure. |
| 3 | The application should take no more than 5 seconds to load. |
| 4 | The system should protect all data inside it against malware attacks or unauthorized access. |
| 5 | User requests (ordering pets/accessories/offering services) should be processed within 5 seconds. |
| 6 | The user interface must be easily understandable by laymen having a basic understanding of English language, with users not having to spend more than 3 seconds to figure out how to proceed with a task. |
| 7 | Considering the application is targeted towards local people, measurements, time zones and currencies will be adjusted accordingly. (Kg, PKR etc.) |
| 8 | The time to notify rescue teams after the initiation of rescue requests should be no more than 20 seconds. |
| 9 | The system should initially be able to cater to a minimum of 1000 users simultaneously. |
| 10 | The system should have a clear distinction between the rights of the admin and the rights of the app users. |
| 11 | Users should be able to navigate their way around the application with ease after continuous use and exploration of it for at least 15 minutes. |
| 12 | The user should be able to make a post within 5 seconds |
| 13 | When the app gets interrupted by a call it should save state for when the call ends. |
| 14 | The locations of the stores or shops should be accurate within a radius of 10 meters. |
| 15 | The user should be able to reorder an item in under a minute. |
| 16 | The system should be compatible with Android OS versions 10 and above. |

# **3.** **System Architecture**

This chapter would elaborate on the architecture used to build the application, providing the reason for the selection of that algorithm. It would describe the architecture in detail and provide a visual depiction of it. The reasoning for the selection of this architecture would also be argued, highlighting why this architecture works best for the type of application being created by us.

## **a.** **Architecture Diagram**



## 

## 

## 

## **b.** **Architecture Description**

There are a number of subsystems used in our architecture. These are divided into user, user actions and, utility. The subsystems include:

**2.2.1 User**

1. Registration:

This subsystem handles the entire process of registration, including generating and processing the forms.

1. User Profile System:

This subsystem is responsible for all the user preferences and the user profile data. If a user makes changes to their profile, this would be the subsystem that would be responsible for that.

**2.2.2 User Actions**

1. News Feed:

This subsystem would handle the way the news feed is displayed and whatever actions are performed on the news feed. This would use the Utility subsystems,as well as the User Profile System to generate its news feed.

1. Rescue System:

This subsystem would be responsible for creating the rescue requests that the pet shelters can then use to rescue pets.

1. Inventory Management:

This subsystem would be responsible for all the shops, stores, shelters and services on the application. This subsystem would manage their inventory, generate the store page.

1. Order System:

This subsystem is closely linked to the inventory management subsystem. This subsystem manages all the orders and the purchases. It allows the users to buy from their stores and handles all the necessary processing for order confirmation.

1. Review System:

This subsystem deals with the reviews that the user leaves on the application. It allows the users to leave the reviews and the stores to check them.

**2.2.3 Utility**

1. Posting System:

This subsystem makes posts and submits them to be posted on the news feed or for the rescue system.

1. Notification System:

This subsystem is responsible for sending notifications and responding to received notifications.

1. Filtration System:

This subsystem handles all the filtration tasks, for all the various interfaces. It also includes processing reports for hateful content or spam.

1. Location System:

This subsystem handles getting the user location or fetching coordinates from a location to send to Google Maps.

1. Search System:

This subsystem handles all the search related tasks of the application, whether it be searching for stores, services, posts etc.

These subsystems are interconnected through data streams, they take data from each other. The user actions all take into account and depend on the preferences of the user. While the user actions all use the utility functions, which are the reusable components of the software, for various tasks and needs.

## **c.** **Justification of the Architecture**

Our primary concern was not only to make different layers of our application separate but to make complex subsystems out of simple blocks, hence we came out of BloC Architecture keeping in mind the technologies we are using and the context of our system. We have a user profiling system that has multiple UI interfaces, for instance, common user UI, Shelter UI, Service Provider UI etc.

The business logic layer of the architecture contains the business logic of the subsystems present in a semi-restricted layered structure, keeping the most used subsystems in the lowest layer, hence maintaining a useful layered structure. As our application is based on flutter, our application and business logic reside client side in order to ensure low-latency and a responsive UI design. The business logic is kept entirely separate from the UI and data layers to keep the system modular and scalable, both in terms of functionality and load. In order to communicate with the UI and repository layers, we use an event-based architecture with streams. Events trigger functions which perform business logic and state is updated via streams. This architecture is common and optimized for flutter and works well in separating state from UI.

The repository layer is used to handle multiple sources of data from APIs. For example, data could be fetched from a cache or if the cache is expired then from the database. So in order to facilitate multiple API calls and data sources the repository layer handles streams from the API handler and forwards optimal data to the Business Logic layer.

The API handler layer will handle all API calls and connections with the DB etc. it will handle connection timeouts as well as responses for API calls e.t.c. This is kept separate from the logic since these data sources and APIs can change and in order to facilitate modular design, only the API handler layer has to handle it, the logic layers will not have to cater to it.

We used cloud DB and storage options in order to ensure consistent latency and up-time and to ensure that there is not a single point of failure. Moreover, having scalable cloud storage helps in easy load scaling and it also facilitates a flexible load, so that costs are dependent on usage only. The system can also quickly adapt to very frequent changes in load and we will not have to physically adapt to such change in resource requirement.

We also added caching servers to serve pre-processed content with < 1ms latency. This will ensure quick loading times for the news feed. The caching servers will reside on the cloud and will fetch data from the db periodically.

In order to serve a live news feed from the many posts by many users, we thought that we should pre-process and rank tags and posts in order to prevent high processing time when a user opens the news feed. So we added a cloud-based Trend-ranking system which will periodically run algorithms and calculate top trends in order to bring relevant content on news feeds. This will run separate from client requests and will ensure that there is not a high client-side load when a user requests the news feed. The calculated trends will be served to both the db and the cache.

We also have a message queuing and push notification system to ensure deliverability of messages including rescue notifications and comments/post notifications e.t.c.

Keeping the above in mind, our architecture has the following pros:

1. Easy to make changes in UI because of separation of state and UI
2. Easy to change data sources and APIs because dependence on only one layer
3. Scalable and flexible due to scalable cloud services and client-side logic
4. Client-side application logic to ensure low latency and increased responsiveness in UI
5. Data heavy tasks and logic delegated to a separate cloud backend to minimize heavy client-side computation.
6. Caching to ensure low latency and quick data delivery
7. Pre-computing heavy tasks ensures low run-time latency

However, there are a few cons:

1. Client side is thick as application logic resides there making the app larger.
2. Regularly computing trends is very resource intensive.
3. Client requests need to be re-validated server-side because we can never trust client-side data.

## **d.** **Tools and Technologies**

1. Flutter 2.0 for development on Android.
2. MongoDB Atlas for cloud DB.
3. OneSignal for push notification.
4. Stream API for chat services.
5. Google Maps location and distance API.
6. Heroku for Hosting server.

# 

# 

# **4.** **Requirements Specifications**

## **a.** **Use Cases**

## Use Case Diagram

### Use Case Descriptions

### **2.1** **Report Stray Animals**

| **Identifier** | | UC-002 | |
| --- | --- | --- | --- |
| **Purpose** | | The user can report the location of the stray animals to nearby animal rescue teams. | |
| **Pre-conditions** | | 1. The *Register Account* use case has been completed. 2. The user is a casual app user or a pet owner. 3. The user has their GPS turned on. | |
| **Post-conditions** | | A post is created with the location of the pet, this can be seen by nearby pet rescue teams. | |
|  | | | |
| **Step #** | **Typical Course of Action** | |  |
|  | The user will select the rescue option. | |  |
| 2. | The system displays a map, with the option to pinpoint users own location. | |  |
| 3. | The user selects the gps option to pinpoint their own location. | |  |
| 4. | The user selects the type of animal. | |  |
| 5. | The user selects the post option. | |  |
| 6. | An automatic post, with the location embedded, is created for the rescue teams. | |  |
|  | | | |
| **Step #** | **Alternate Courses of Action** | |  |
|  | There are none. | |  |
| **Step #** | **Exception Paths** | |  |
|  | If the GPS is not active, the user is given a pop-up to turn the location services on. | |  |
|  |  |  |  |

### **2.2** **Find Veterinary Doctors**

| **Identifier** | | UC-004 | |
| --- | --- | --- | --- |
| **Purpose** | | The user can find the location of a veterinary doctor. | |
| **Pre-conditions** | | 1. The user must have completed the *Register Account* use case. | |
| **Post-conditions** | | The user learns of the location of the veterinary doctor. | |
|  | | | |
| **Step #** | **Typical Course of Action** | |  |
|  | The user goes to the services section. | |  |
| 2. | The user browses the services section and looks for the right veterinary doctor. | |  |
| 3. | The user selects a veterinary doctor. | |  |
| 4. | The system shows the user the credentials as well as the information regarding the veterinary doctor. | |  |
| 5. | The user reads the location of the doctor. | |  |
|  | | | |
| **Step #** | **Alternate Courses of Action** | |  |
|  | After step 4, the user can click on a map to be led to google maps, with their coordinates. They can then visit the store themselves. | |  |
| **Step #** | **Exception Paths** | |  |
|  | In the alternative path, if the veterinary doctor has not uploaded their address, that path cannot happen. | |  |
|  |  |  |  |

### 

### **2.3 Receive Rescue Request**

| **Identifier** | | UC-008 | |
| --- | --- | --- | --- |
| **Purpose** | | The user will get the request of an animal in need and can update the status of the animal rescue. | |
| **Pre-conditions** | | 1. The user must have completed the *Register account* use case. 2. The user must be of the correct type. 3. There should be a request made by one of the casual app users or pet owners. | |
| **Post-conditions** | | The user will get the notification and use the location. | |
|  | | | |
| **Step #** | **Typical Course of Action** | |  |
|  | The user gets the notification. | |  |
| 2. | The user taps on the notification. | |  |
| 3. | The system takes the user to the post. | |  |
| 4. | The user selects the location coordinates. | |  |
| 5. | The system gives the location of the stray animal. | |  |
|  | | | |
| **Step #** | **Alternate Courses of Action** | |  |
|  | The user can reply to the customer afterwards about the success of the rescue or not. | |  |
| **Step #** | **Exception Paths** | |  |
|  | None. | |  |
|  |  |  |  |

### **2.4** **Buy Pet Accessories**

| **Identifier** | | UC-009 | |
| --- | --- | --- | --- |
| **Purpose** | | The user can buy accessories for their pets. | |
| **Pre-conditions** | | 1. The user must have completed the *Register Account* use case. 2. The user must be of type *Pet Owner.* | |
| **Post-conditions** | | The accessories are dispatched, and the user can buy cash on delivery or visit the store itself. | |
|  | | | |
| **Step #** | **Typical Course of Action** | |  |
|  | The user selects the marketplace option. | |  |
| 2. | The user browses the marketplace to find the merchandise shop they want to buy from. | |  |
| 3. | The user selects the store. | |  |
| 4. | The system presents the user with the store page. | |  |
| 5. | The user browses the store page to find the accessories they are looking for. | |  |
| 6. | The user selects the accessories they want to purchase. | |  |
| 7. | The user confirms the purchase and selects the cash on purpose option. | |  |
| 8. | The user enters their information. | |  |
| 9. | The user confirms the order. | |  |
| 10. | The system sends a notification to the seller. | |  |
|  | | | |
| **Step #** | **Alternate Courses of Action** | |  |
|  | 1. The user can at any point go back to the store list. 2. After step 4, the user can filter between types of accessories to make his/her search easier. 3. The user can continue to add items after step 6. | |  |
| **Step #** | **Exception Paths** | |  |
|  | If the item is out of stock, the user cannot order it. | |  |
|  |  |  |  |

### **2.5** **Get in contact with the Customer**

| **Identifier** | | UC-013 | |
| --- | --- | --- | --- |
| **Purpose** | | The pet stores,shelters, merchandise shops can chat with their customers. | |
| **Pre-conditions** | | 1. The types of communicator and the one they are communicating should be appropriate. 2. The shops have entered correct communication information. | |
| **Post-conditions** | | The customer and the seller can get in contact with each other. | |
|  | | | |
| **Step #** | **Typical Course of Action** | |  |
|  | The shop will open the customer's profile by looking through their orders. | |  |
| 2. | The shops will press the chat button. | |  |
| 3. | The shops will be led to the chat window, by the system. | |  |
| 4. | The shops will type the text they want. | |  |
| 5. | The shops will send the text. | |  |
|  |  | |  |
|  | | | |
| **Step #** | **Alternate Courses of Action** | |  |
|  | None | |  |
| **Step #** | **Exception Paths** | |  |
|  | There are none. | |  |
|  |  |  |  |

### **2.6** **Give Location Coordinates**

| **Identifier** | | UC-017 | |
| --- | --- | --- | --- |
| **Purpose** | | The store, veterinary doctors, shops, services can give their locations on their pages. | |
| **Pre-conditions** | | 1. The users must have completed the *Register account* use case. 2. The users must be of the correct type. | |
| **Post-conditions** | | The user's location will be given on their store page. | |
|  | | | |
| **Step #** | **Typical Course of Action** | |  |
|  | The user visits their profile. | |  |
| 2. | The user selects their store page. | |  |
| 3. | The user goes to the information section by pressing the button “i”. | |  |
| 4. | The user presses the pencil icon. | |  |
| 5. | A form is opened by the system. | |  |
| 6. | The user goes to the section of location. | |  |
| 7. | The user clicks on the add location button, | |  |
| 8. | The user uploads their location. | |  |
|  | | | |
| **Step #** | **Alternate Courses of Action** | |  |
|  | The location can be uploaded on the time of creating the account/store page. | |  |
| **Step #** | **Exception Paths** | |  |
|  | If GPS is off on step 7, the GPS will need to be turned on. | |  |
|  |  |  |  |

### **2.7** **Set up Store**

| **Identifier** | | UC-018 | |
| --- | --- | --- | --- |
| **Purpose** | | Set-up Store | |
| **Pre-conditions** | | The owner has a registered account. | |
| **Post-conditions** | | The store is visible to the general public upon search to buy pets/ accessories from. | |
|  | | | |
| **Step #** | **Typical Course of Action** | |  |
| 1. | The user submits a request for making a pet store, using relevant workflow.  (make a store etc.) | |  |
| 2. | The user enters relevant details, items available, address contact, pictures | |  |
| 3. | The account request goes to the admin for review. | |  |
| **4.** | The request gets approved. | |  |
| 5. | The user gets a notification and access to their store online, by the system. | |  |
|  | | | |
| **Step #** | **Alternate Courses of Action** | |  |
|  | In step 4, the request gets rejected, the user gets notified and their info is discarded. | |  |
| **Step #** | **Exception Paths** | |  |
|  | In step 2, if mandatory details are not provided, (contact, enough pictures etc.) the request doesn’t proceed. | |  |
|  |  |  |  |

### **2.8 Upload Pictures of Pets**

| **Identifier** | | UC-021 | |
| --- | --- | --- | --- |
| **Purpose** | | Sharing pets pictures on the platform | |
| **Pre-conditions** | | The user has a registered account. | |
| **Post-conditions** | | The shared pictures show up in other users' feeds. | |
|  | | | |
| **Step #** | **Typical Course of Action** | |  |
| 1. | The user logs in to their account. | |  |
| 2. | The user chooses the ‘share’ option and selects pictures from their gallery. | |  |
| 3. | The user enters a text description for the post. | |  |
| 4. | The user presses the post button. | |  |
| **4.** | The post gets uploaded on the News Feed, by the system. | |  |
| **5.** | The user is taken back to the News Feed, by the system. | |  |
|  | | | |
| **Step #** | **Alternate Courses of Action** | |  |
|  | In steps 2, 3, and 4, the user can press the cancel button and go directly to step 5. | |  |
| **Step #** | **Exception Paths** | |  |
|  |  | |  |
|  |  |  |  |

### **2.9 Updating Profile**

| **Identifier** | | UC-0028 | |
| --- | --- | --- | --- |
| **Purpose** | | The user can make desired changes to their profile. | |
| **Pre-conditions** | | The user has an account and is logged in. | |
| **Post-conditions** | | The changes are effectively displayed on the profile. | |
|  | | | |
| **Step #** | **Typical Course of Action** | |  |
|  | The user clicks on ‘profile settings’ on their account. | |  |
| 2. | The user makes the changes they want to (which may include,changing profile picture, updating interests and other details). | |  |
| 3. | The users saves the made changes. | |  |
| 4. | The new information is updated by the system and is visible to all the users visiting the profile. | |  |
|  | | | |
| **Step #** | **Alternate Courses of Action** | |  |
|  | The user can cancel the process before step 3. | |  |
| **Step #** | **Exception Paths** | |  |
|  | None | |  |
|  |  |  |  |

### **2.10 Sell Pets**

| **Identifier** | | UC-0030 | |
| --- | --- | --- | --- |
| **Purpose** | | The pet store owners can sell their pets. | |
| **Pre-conditions** | | The user has a store set up. | |
| **Post-conditions** | | The pet is sent to the customer and the amount is paid. | |
|  | | | |
| **Step #** | **Typical Course of Action** | |  |
|  | The user views list/ notifications of the reserved pets in the ‘orders in progress’ section. | |  |
| 2. | The user changes the status of the order after sending the pet for delivery/ handing in-person. | |  |
| 3. | The order is marked as completed after receiving cash on delivery or in-person. | |  |
| 4. | The sold pet is removed from available for sale pets and the changes are visible to the app users. | |  |
|  | | | |
| **Step #** | **Alternate Courses of Action** | |  |
|  | The customer can cancel the order before step 3, the order gets deleted from the list and the pet is available for sale for other users. | |  |
| **Step #** | **Exception Paths** | |  |
|  | In case of non-payment of the amount, the order is cancelled by the user and the pet is available for sale to other customers. | |  |
|  |  |  |  |

## 

## 

## 

## 

## 

## 

## 

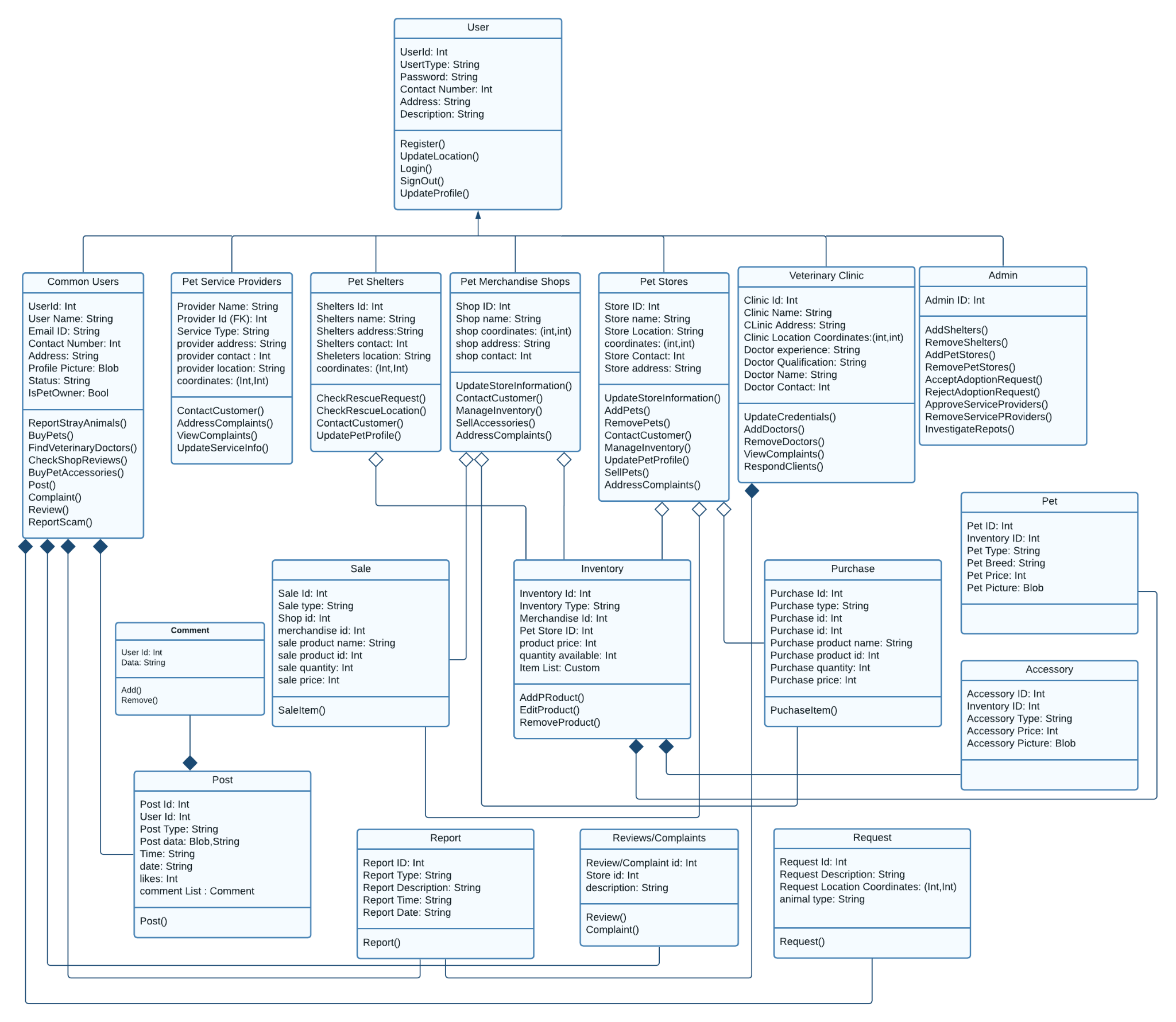
## 

## 

## 

## **b.** **Class Diagram**

### **Diagram**



### **Description**

**User** : It is a user of an application that can be subdivided into common users, Service Providers, Shelters , Veterinary Clinic, Pet Store and admin and it can register.

**Common Users:** Common users are those who are casual app owners and they can be pet owners, they have the functionalities of reporting stray animals, buy pets, finding veterinary options, post and review etc

**Pet Service Providers:** They are providing services related to pets, they can also update their services and address complaints if any.

**Pet Shelters:** Pet Shelters are entertaining rescue requests mainly.

**Pet Merchandise Shops:** Pet merchandise shops are the stores which sell products related to pets.

**Pet Stores:** Pet stores are the places where people can buy pets from. These are physical stores which are set up online on our app as well.

**Veterinary Clinic:** Veterinary clinic is a place where one or more than one veterinary doctor sits. These are also physical clinics with an online clinic set up on our app.

**Admin:** Admin is the owner of the app and they have special rights. They can add, remove people on the app, and modify several other things.

**Comment:** A comment is a message a user leaves on a post, video, blog or any advertisement of a product.

**Sale:** A sale is the exchange of goods (accessories or pets) in return for money. Sales on our app will be made in person for pets and can be cash on delivery for other accessories.

**Inventory:** An inventory is a list of items available at a particular store.

**Purchase:** A purchase is users buying the items in exchange for money. A purchase can be made for pets or accessories.

**Pet:** A pet is an animal kept by people for companionship or entertainment. A pet, on our app, can be in a store for sale, shelter for adoption or might belong to app users.

**Accessory:** An accessory is a product which might be used for decorative or useful purposes. An accessory is usually related to pets in the context of our app.

**Post:** A post is an image, video or text shard by an app user, which is visible to other app users and they can react or comment on it.

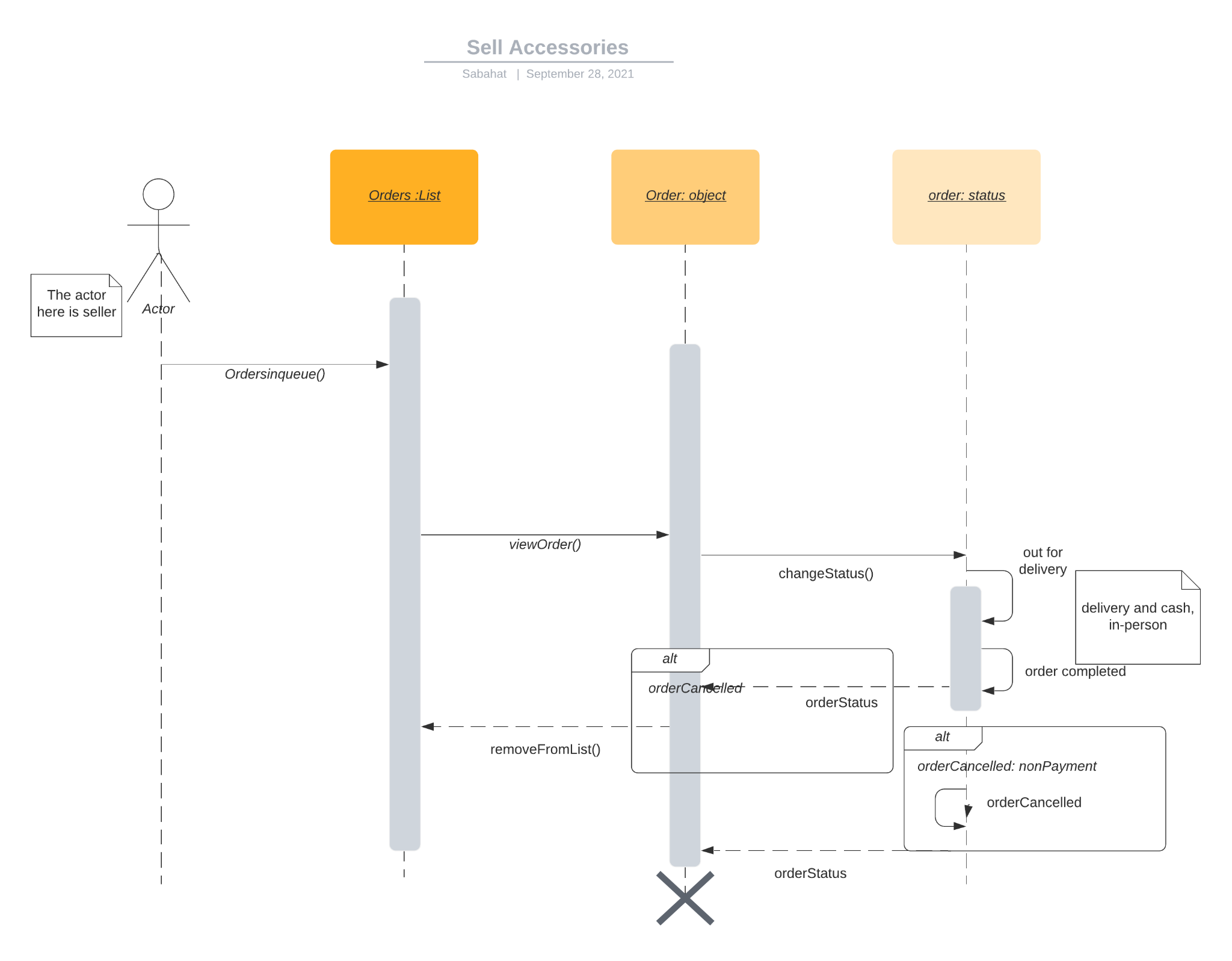
**Report:** A report is made by app users to draw admin’s attention to something which should not be on the app. The reports are reviewed by the admin and the relevant action is taken.

**Reviews / Complaints:** Reviews and complaints are the messages sent to store owners by buyers, to give feedback on their product or to register their complaints.

**Request:** A request is an application submitted by stores, shelters, vets and service providers to get permission to be allowed to sell their services on the app. The admin approves or deletes the request.

## **c.** **Sequence Diagrams**

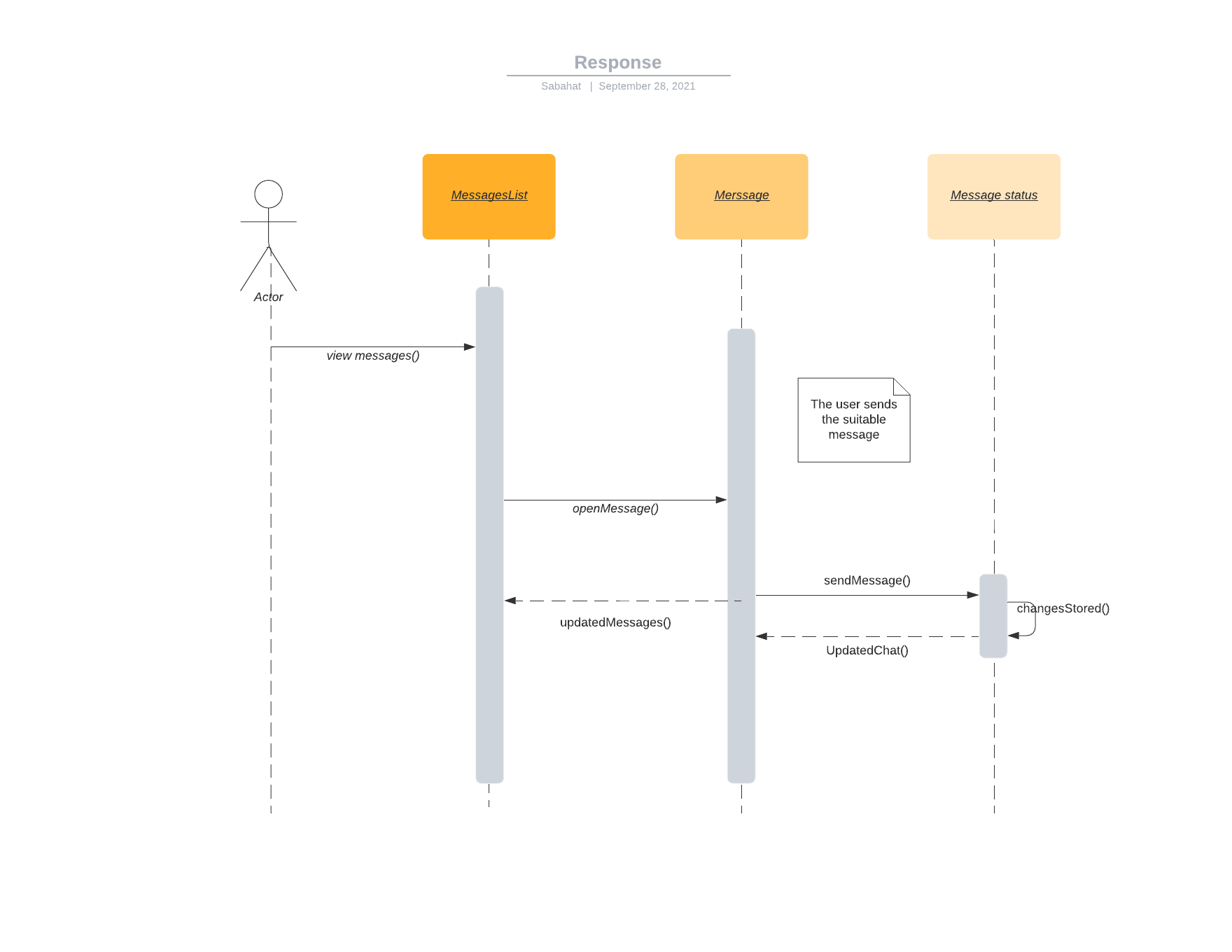
### **5.1 Sell Accessories**

****

### 

### 

### **5.2 Respond via Messages**



### 

### 

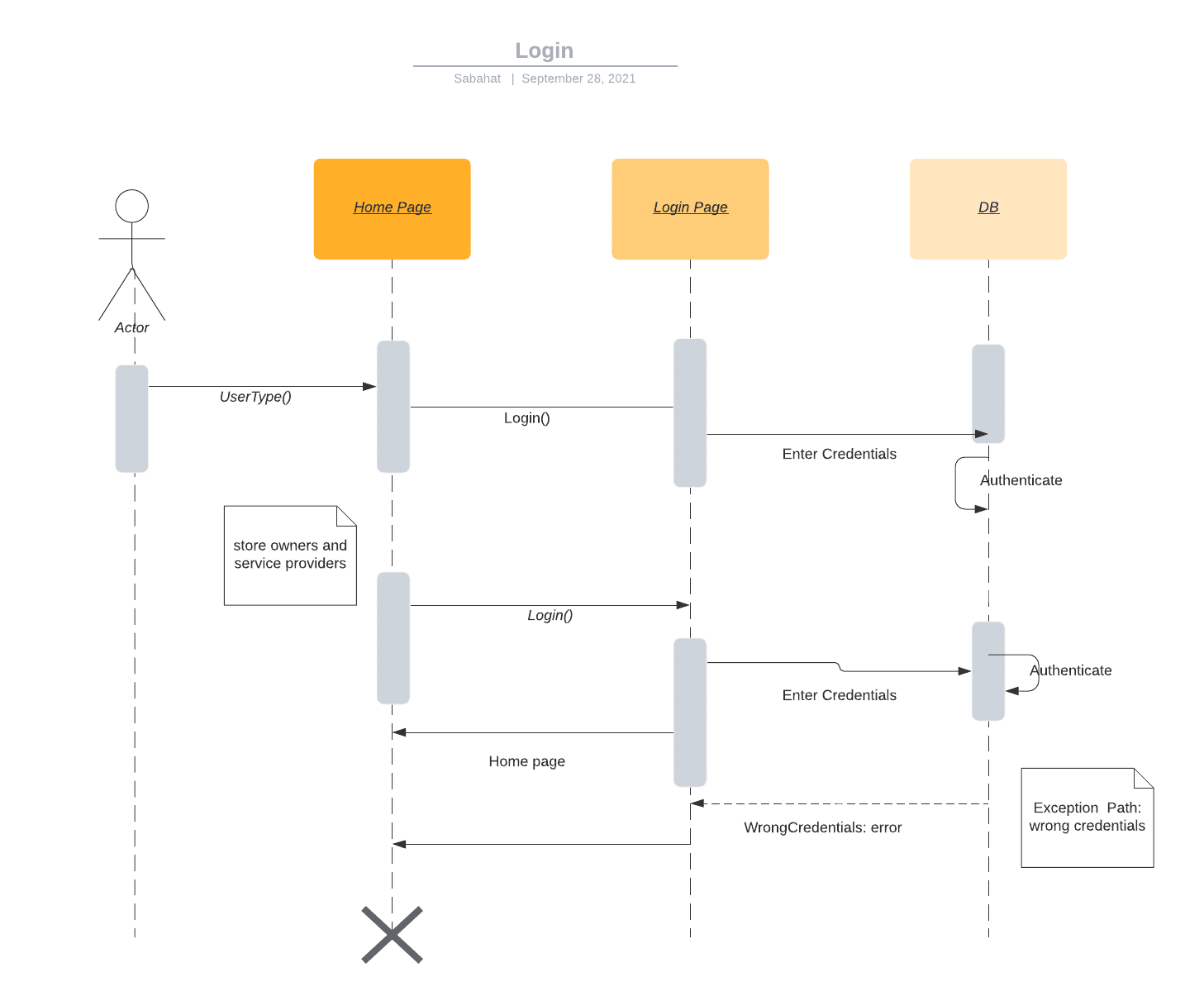
### 

### 

### 

### 

### **5.3 Login**



### 

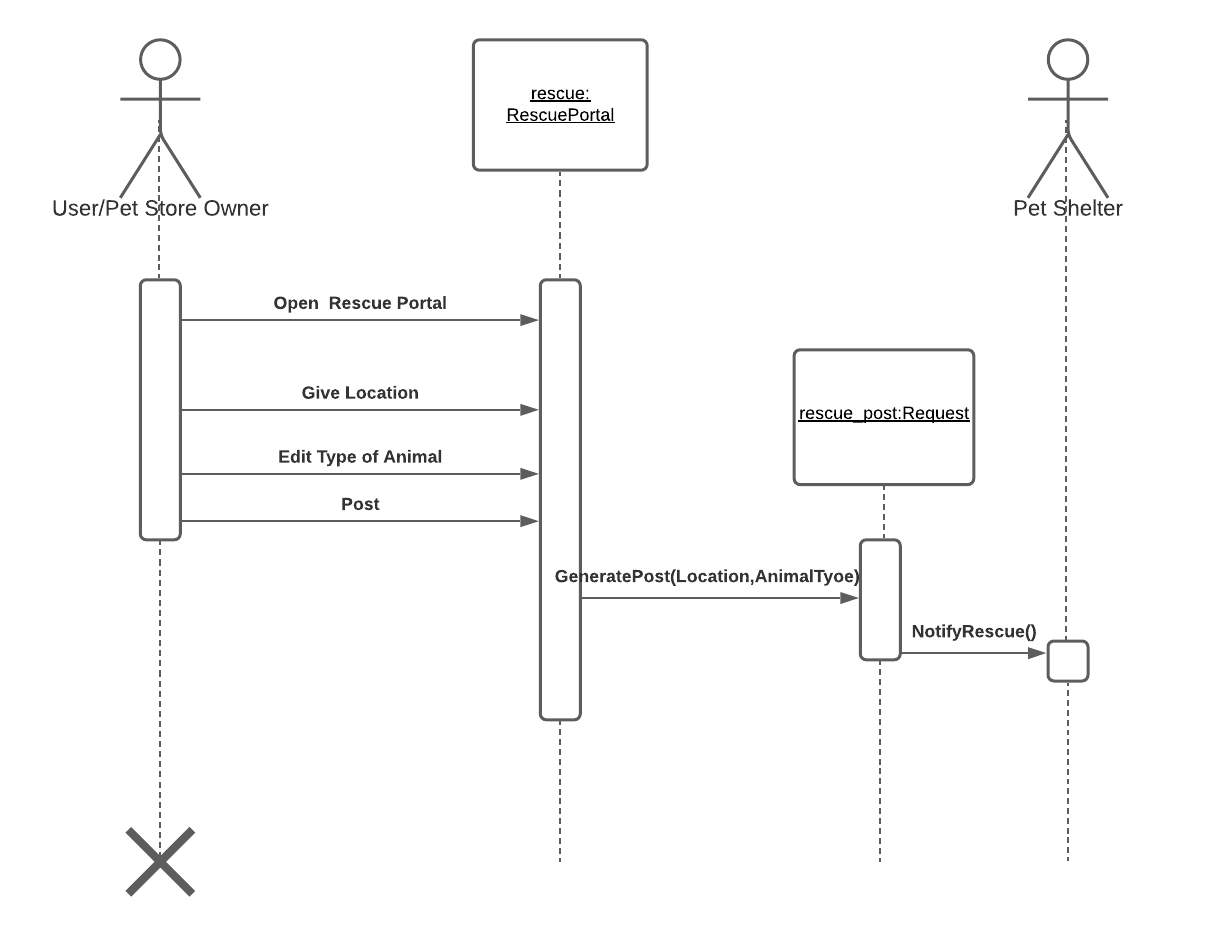
### 

### 

### 

### 

### **5.4** **Report Stray Animals**



### 

### 

### 

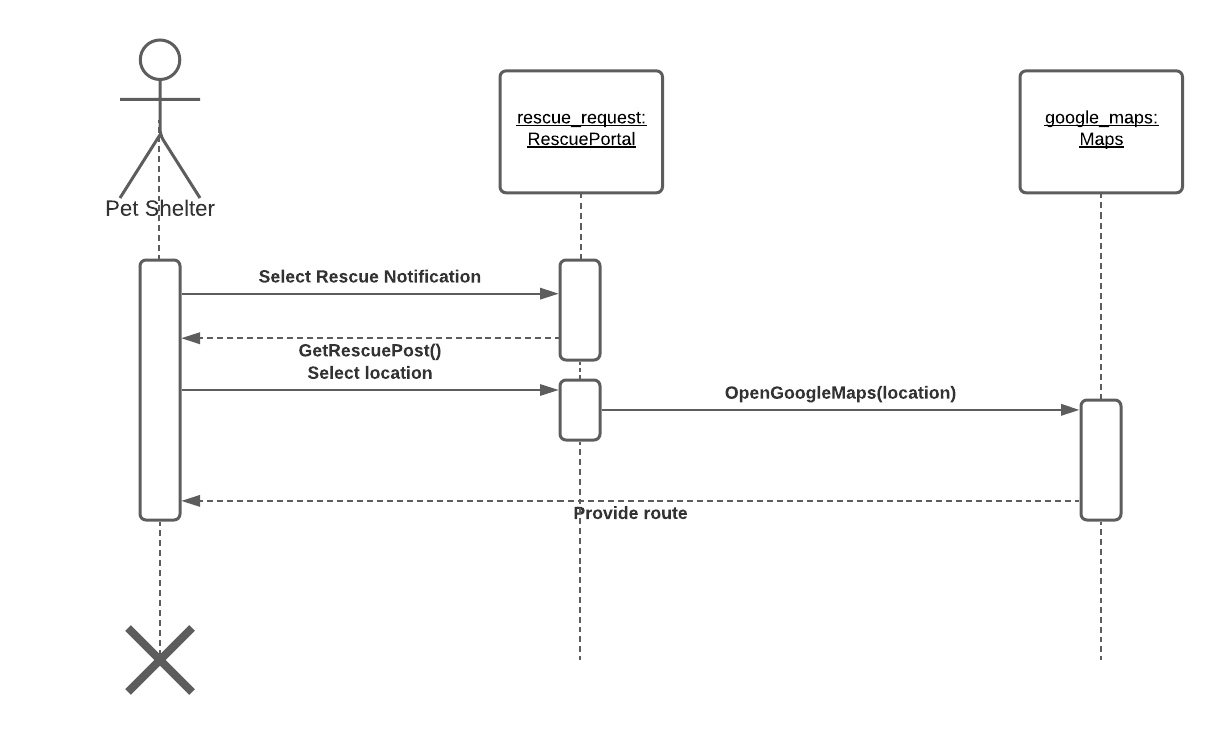
### 

### 

### 

### 

### **5.5** **Receive Rescue Request**



### 

### 

### 

### 

### 

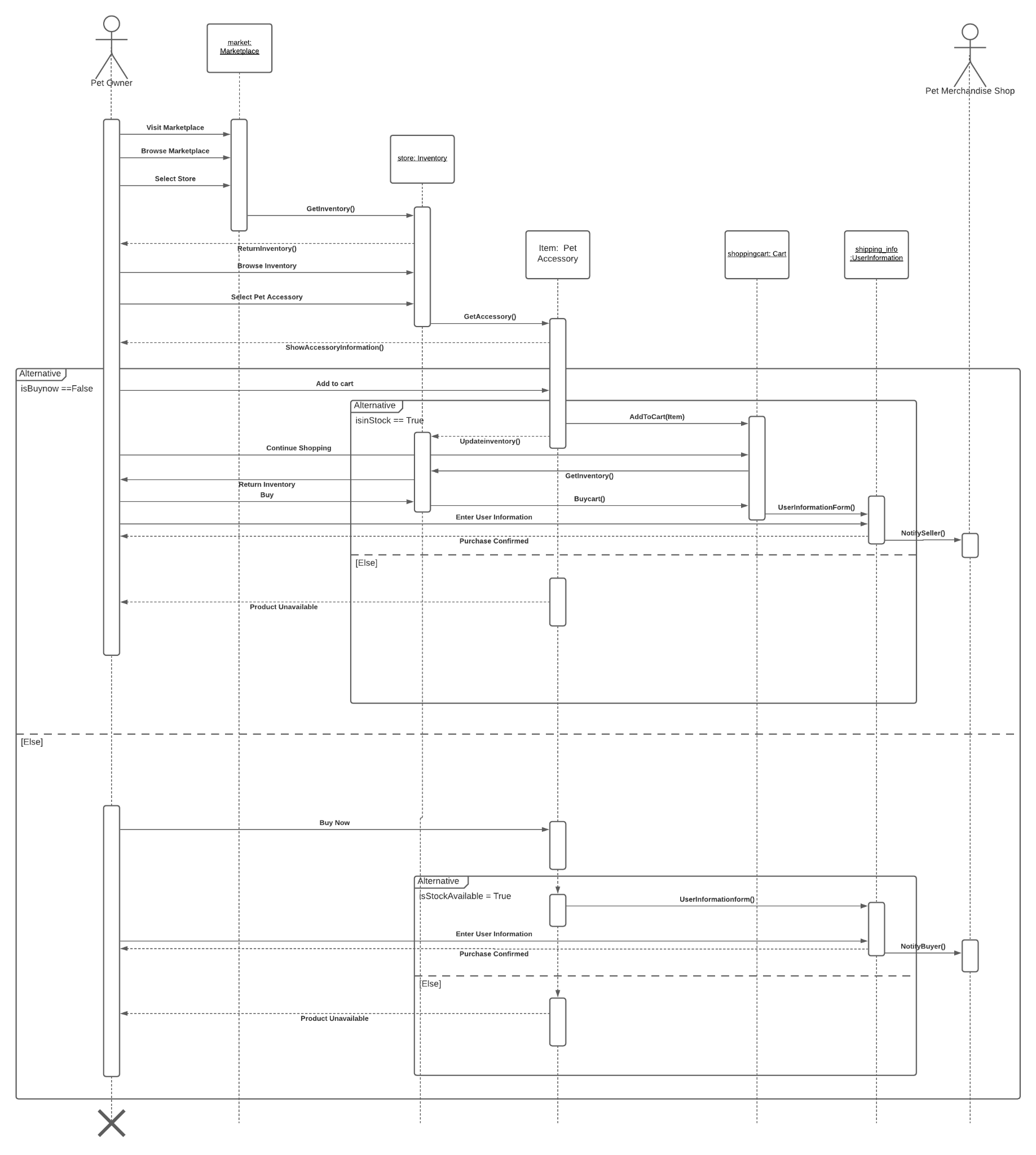
### 

### 

### 

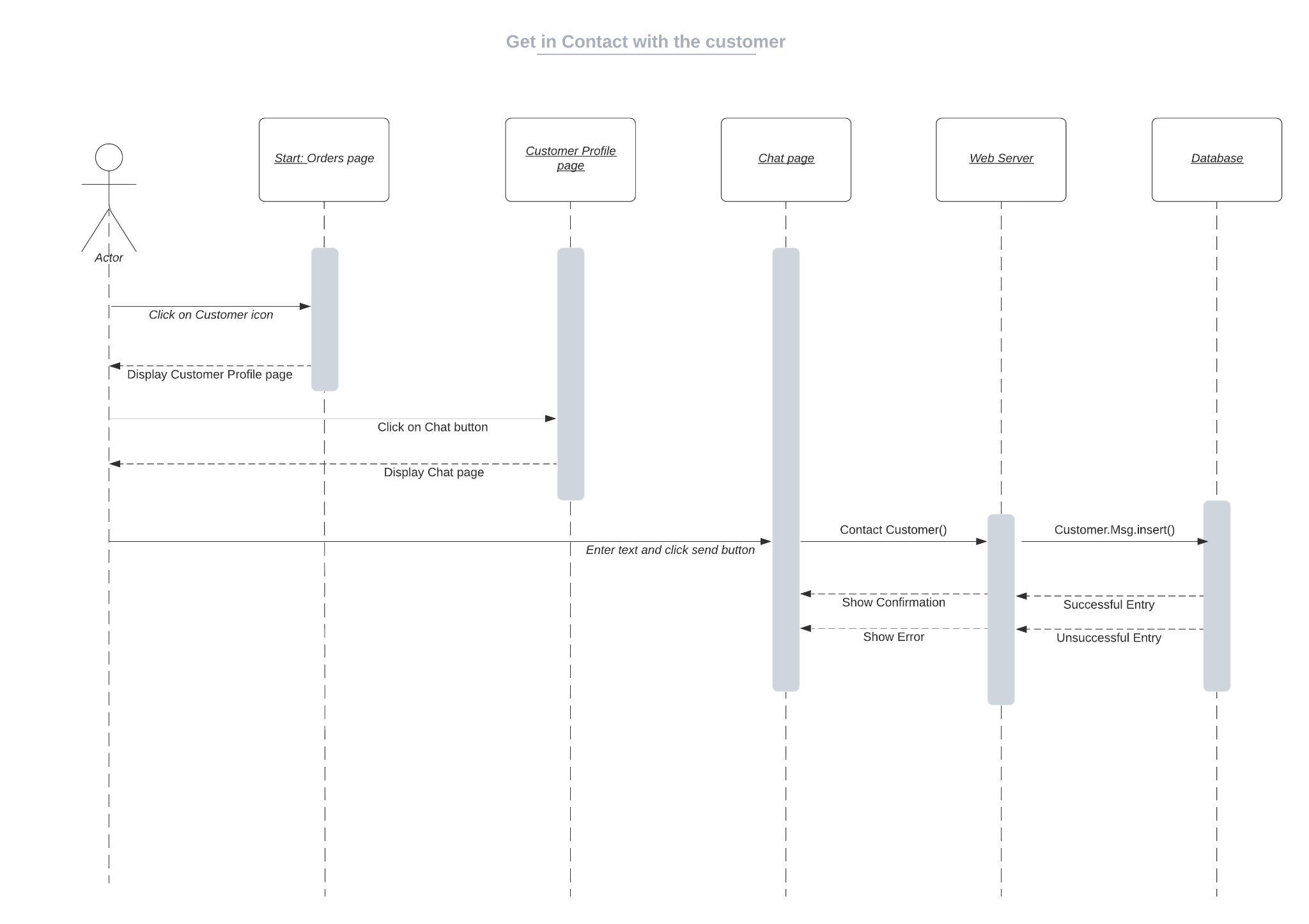
### 

### **5.6** **Buy Pet Accessories**



### 

### **5.7 Get in Contact with the Customer**



### 

### 

### 

### 

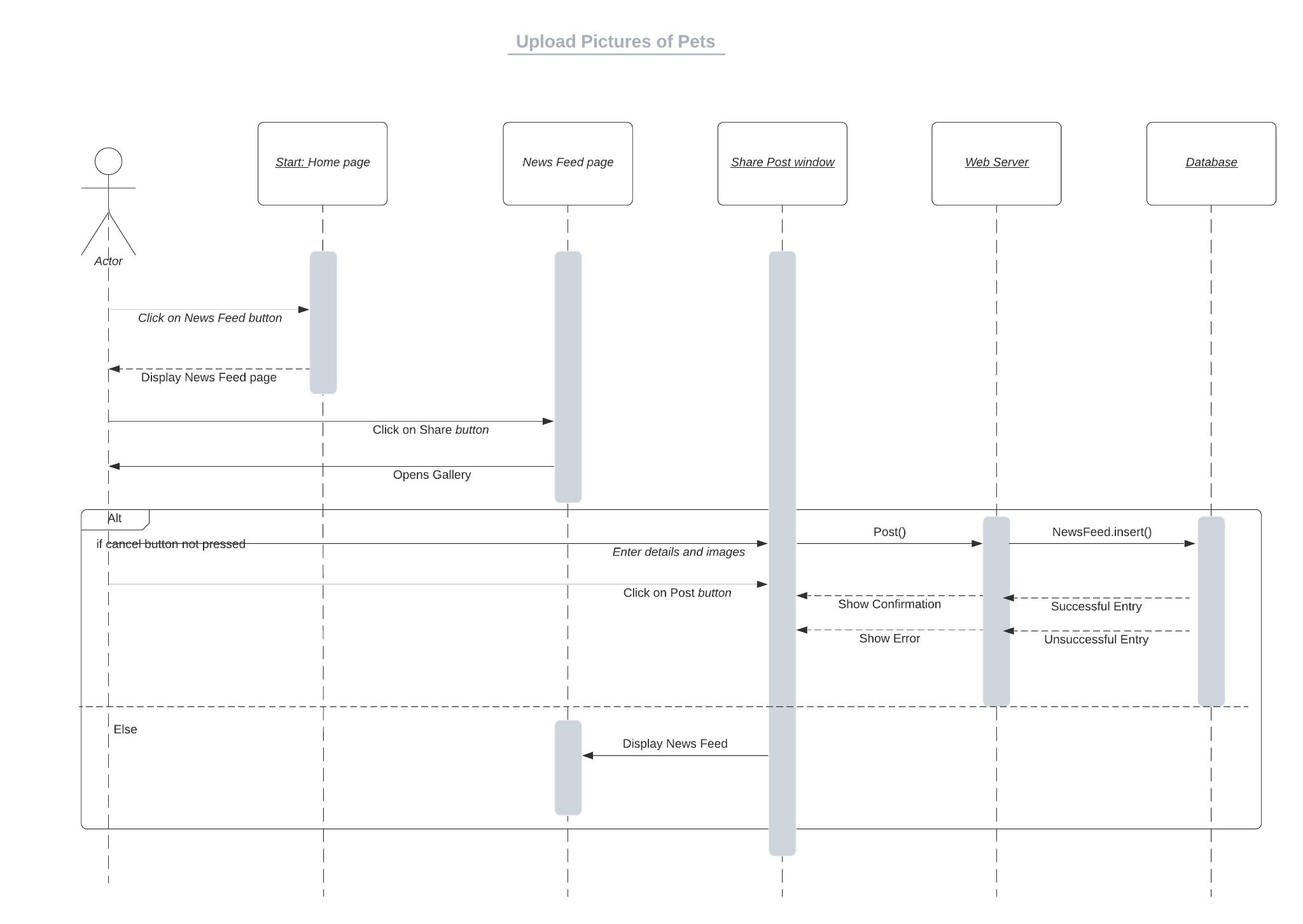
### 

### 

### 

### 

### **5.8 Upload Pictures of Pets**



### 

### 

### 

### 

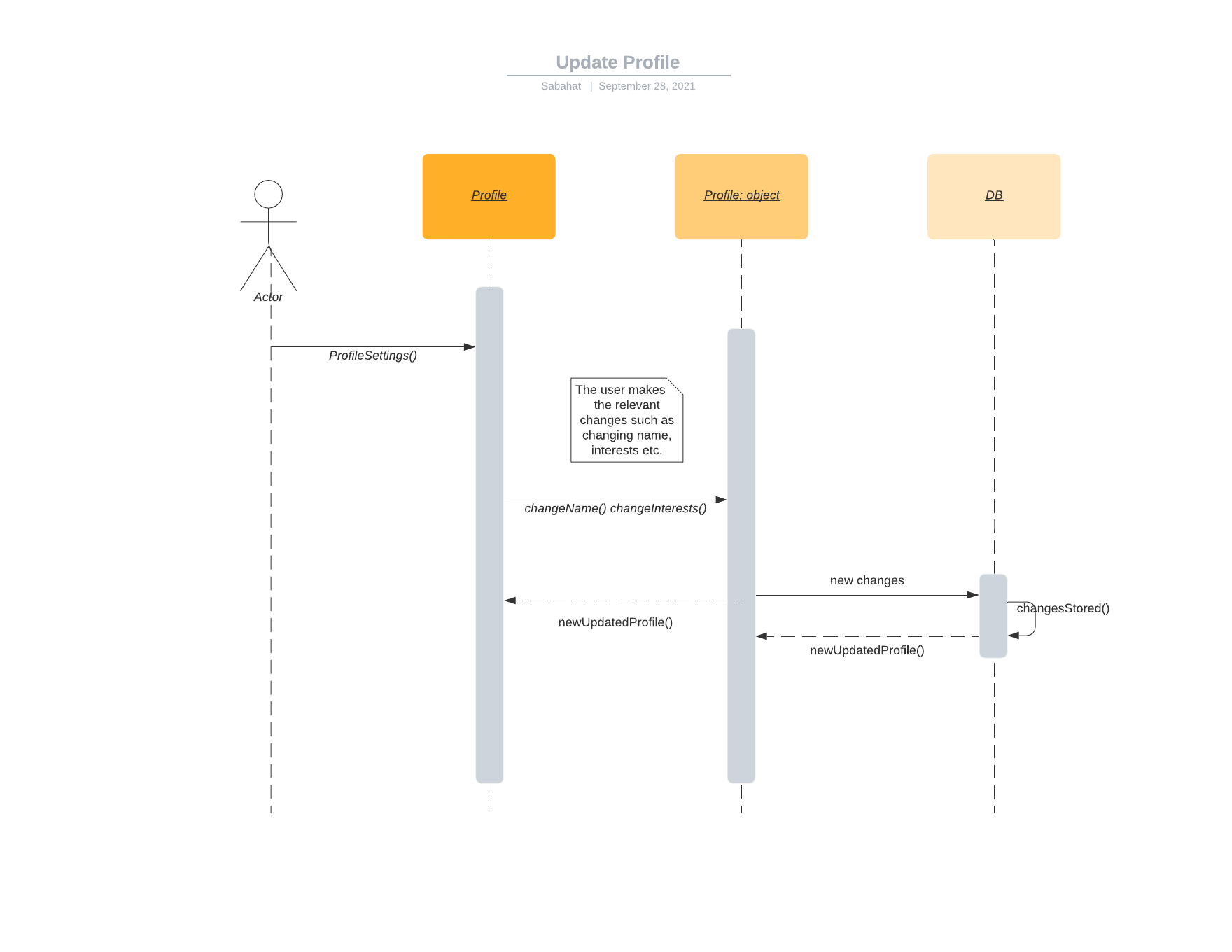
### 

### 

### 

### 

### **5.9 Update Profile**



### 

### 

### 

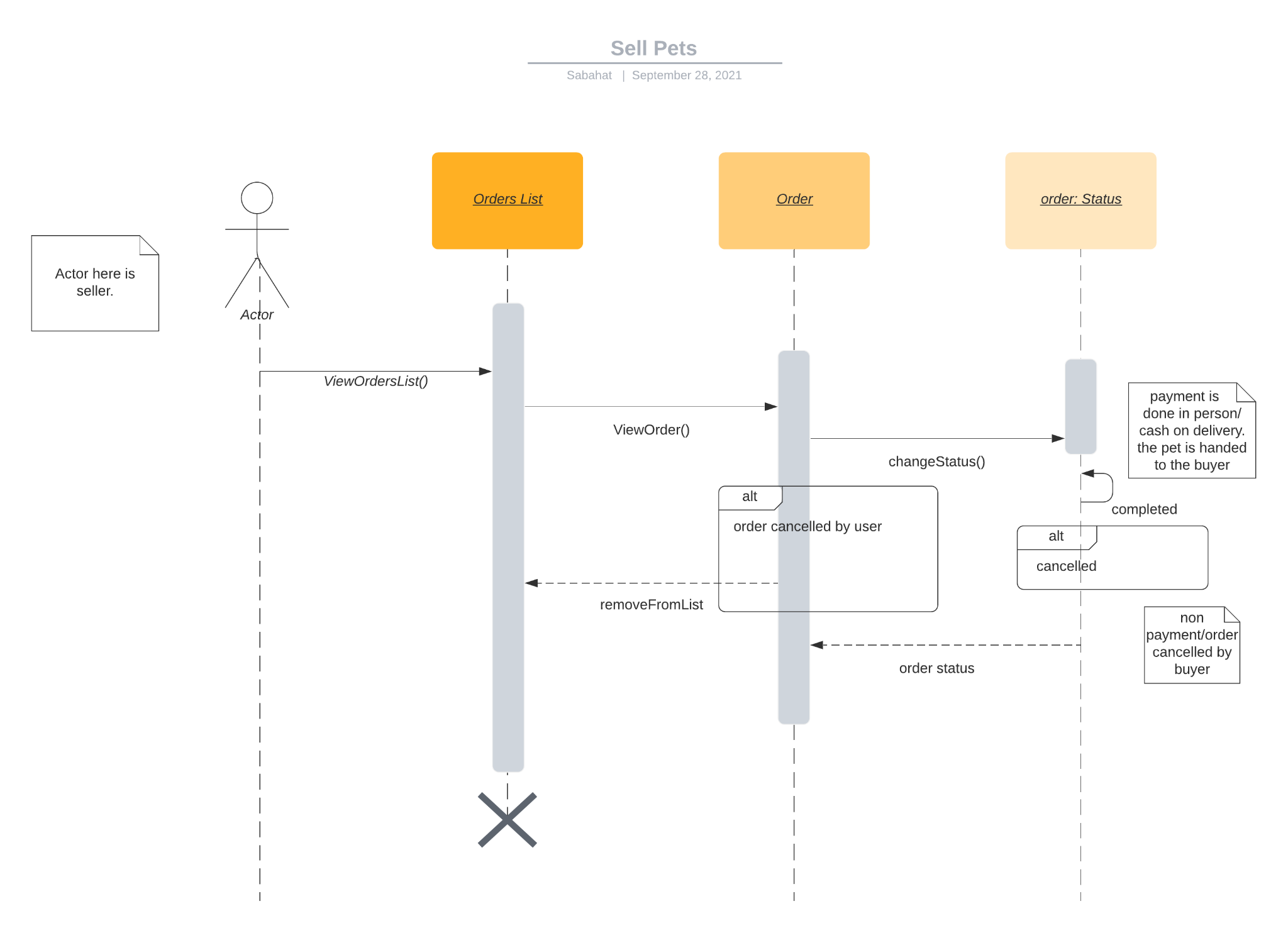
### 

### 

### 

### 

### **5.10** **Sell Pets**



# 

# 

# 

# 

# **5.** **Software Development Methodology and Plan**

This section would discuss the development pipeline of the application. It would elaborate on the software development process that was selected and elaborate on the reasoning for that choice. It would also compare the waterfall and agile models to illustrate why one was selected over the other.

## **a.** **Software Process Selection**

#### **Model Comparison**

**Waterfall Model:**

In a waterfall model, the project requirements, costs and timelines are worked upon in the start and are set in place. This is a more predictable approach where expectations and requirements are clearly communicated and are fixed in place. There is no room for change and the timelines are rigid. There is one final product that is delivered at the end of the whole process and it is subsequently tested.

**Pros:**

1. The client is given a clear expectation of when and what to expect for a finished product.
2. Rigid timelines allow for better planning and resource allocation
3. There is less dependency on specialized individuals and it is easy to find replacements
4. The system architecture can be specially designed since there is no need to account for changes.

**Cons:**

1. There is a lot of overhead time in starting the project as all requirements and design needs need to be figured out in the start.
2. Slow rollout of software since there is no iterative process and the product is only available at the end.
3. Cannot completely test products during production since there is no Working product. This it is not suitable for user centered design
4. There is no flexibility for change in client requirements. If the circumstances around the product change then there is no way to make changes to the requirements or design etc. The product direction cannot be realigned with respect to market changes.

**Agile:**

In the agile process, our project is divided into a large number of small tasks that are called sprints and for each sprint, a team works through a full software development cycle including planning, requirements analysis, design, coding, testing, accepting testing and release.

Each sprint is of a fixed time period i.e. four weeks and size of sprint is mostly based on team size (5-10 people).

**Pros:**

1. If the market for the software is poorly understood or the requirements are poorly understood, or expected to change, select a process that keeps flexibility, and we will use an iterative process containing sprints and phased implementation.
2. Revisits and rewrites of steps are encouraged to achieve the desired results.
3. Allowing for faster delivery and a better project.
4. Frequent delivery allows for quick changes in project direction while maintaining project scope.

**Cons:**

1. If a big software system has many interrelated components, and we want to avoid major changes to the design of a system during development, we can’t use agile methodology.
2. Agile doesn’t set a strict schedule, which, if not managed, can be difficult under a tight deadline.
3. Changing project requirements may cause problems in other areas of the organization.
4. Agile requires a consistent team. A weak link in the Agile team or management could result in wasted time and money.

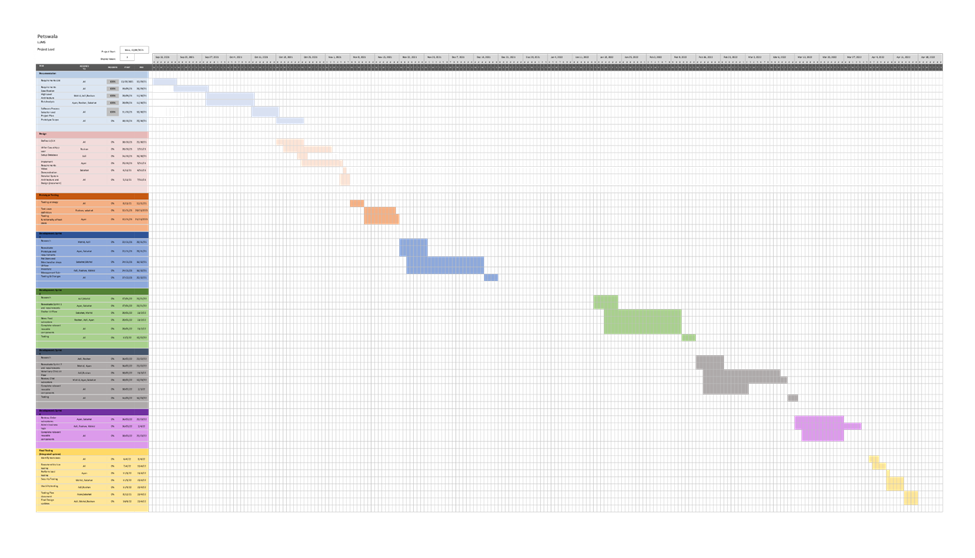
### **Selection of Process for Project Development**

We are going to use scrum methodology which is based on the agile philosophy.

**Justification:**

Our product is very modular in nature. We have a lot of subsystems which will be incrementally added with subsequent iterations. We also aim to incrementally make performance improving additions while focusing on delivering functionality first. We have a small fixed team which is perfect for an agile approach. We also aim to test our products after every sprint and incorporate a user-centered approach in order to maintain a correct direction with regards to fulfilling user needs. For this we need to be flexible in adapting to changing market dynamics. An agile approach is necessary to achieve this.

## **b.** **Gantt Chart**



# 

# 

# 

# 

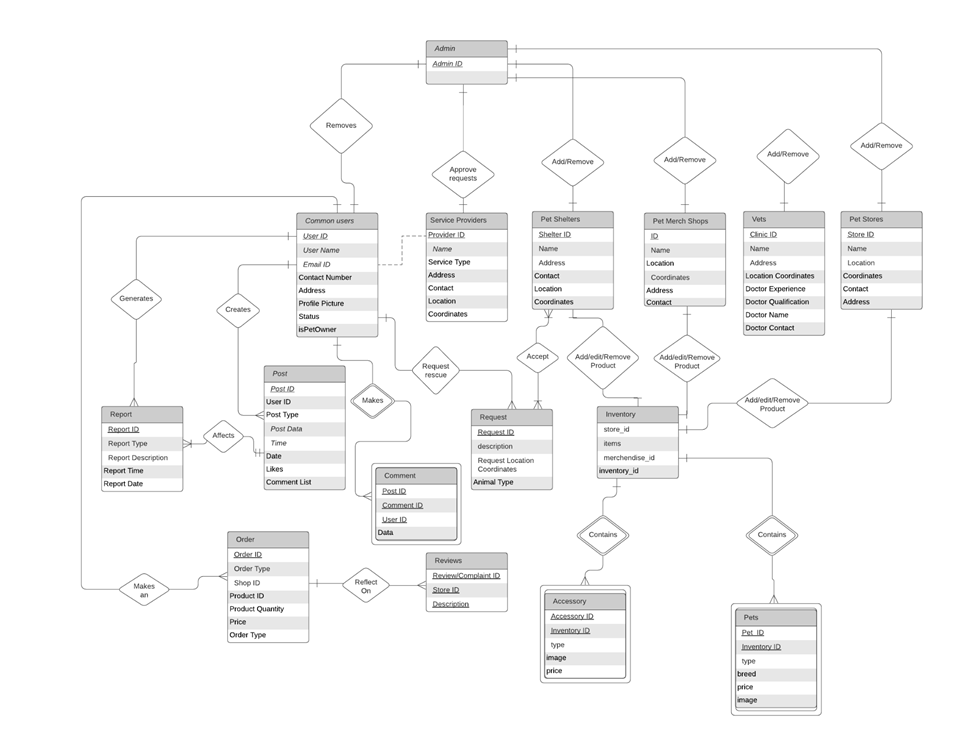
# 

# **6.** **Database Design and Web Services**

Brief introduction of this chapter in a paragraph

## **a.** **Database Design**

### **Diagram**



### **Description**

The data fields have been taken from the class diagram since they represent the variables or data that would be stored in the database. The entities chosen for the E/R diagram were also taken from the class diagram on the basis of them having or requiring a representation in the database. These entities include:

● Admin:

This contains the ids of the different admin to help them log in.

● Common users:

This entity contains all the information about the common user or pet owner type of user.

● Service Providers:

This entity contains all the information about the service providers.

● Pet Shelters:

This entity contains all the information about the Pet Shelters user type.

● Pet Merch Shops:

This entity contains all the information about the Pet Merchandise Shops.

● Vets:

This entity contains all the information about Veterinary doctors.

● Pet Stores:

This entity contains all the information about the Pet Stores.

● Report:

This entity is for the table that contains all the reports that have been generated on posts.

● Post:

This entity is for the table that contains all the data for posts, including the post data, the id of post and user, number of likes and comment list.

● Comment:

This entity is for the table that contains all the comments, this is a weak entity since it is dependent on the post.

● Request:

This entity is for the table that represents all the rescue requests for rescuing animals and contains the location of the animals and the type of animal.

● Inventory:

This is the entity that represents each shop, merch store and shelters and their goods, this entity contains ids for the different item type and the store.

● Accessory:

This entity has all the accessories data, basically, all the accessories in stock. This is a dependent entity and hence a weak entity. This is dependent on inventory.

● Pets:

This entity has all the Pets data, basically, all the Pets in stock. This is a dependent entity and hence a weak entity. This is dependent on inventory

● Order:

This entity represents all the transactions that happen on the application whether they be sales or purchases. This entity represents the table that stores the data for these transactions. It includes the order id, the order type(sales or purchase), quantity and price.

● Reviews:

This entity represents all tables containing the reviews left on the Pet stores, merchandise shops, vets etc. It contains the review id, the id of the store on which the review is left on and the review itself.

## **b.** **API Specification**

In Petswala, for rescue alerts, we have used One Signal API for sending push notifications to specific list of people and we can filter them accordingly, the API takes care of all the subscribed users. In order to implement a chat system effectively, we used Stream API that handles all the channels initiated by different users so they can chat with different users, and we can also track API calls and usage and track channels effectively. We have also built a RESTful API in our project that handles all the client side requests and deal with accordingly. We have defined different routes, models and controllers for it and all the one signal, stream API configurations are managed through the RESTful API, we have also used Google authentication API that is used for authentication and signup through Email Id that makes it secure and makes the sign up simple. We have also used Google Maps API in order to find the current location when we initiate a rescue request, that effectively update the location and we can connect with shelters.

**7.** **System User Interface**

This subsection should explain the functionality of your application to the end user with supporting screenshots.

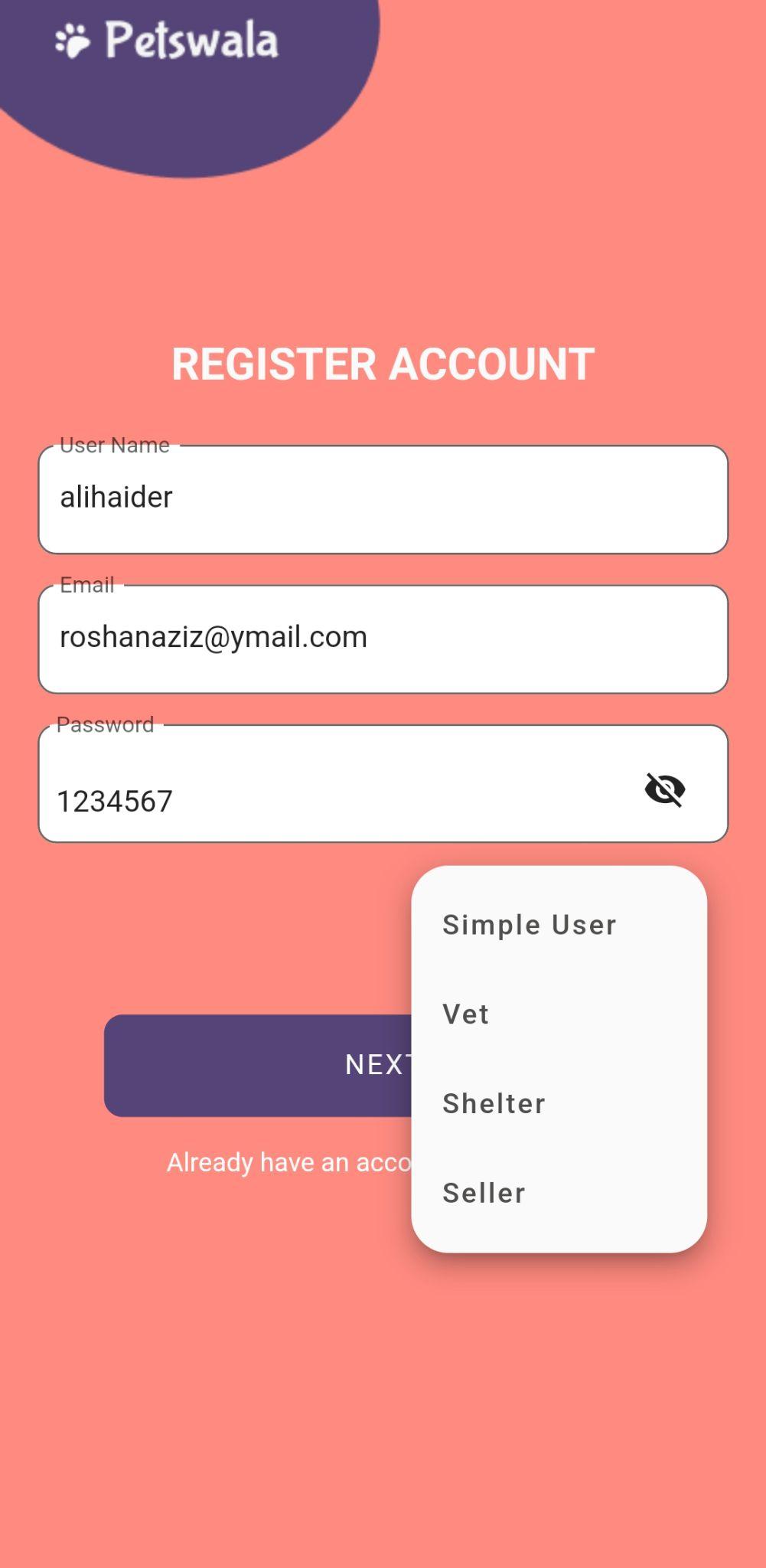
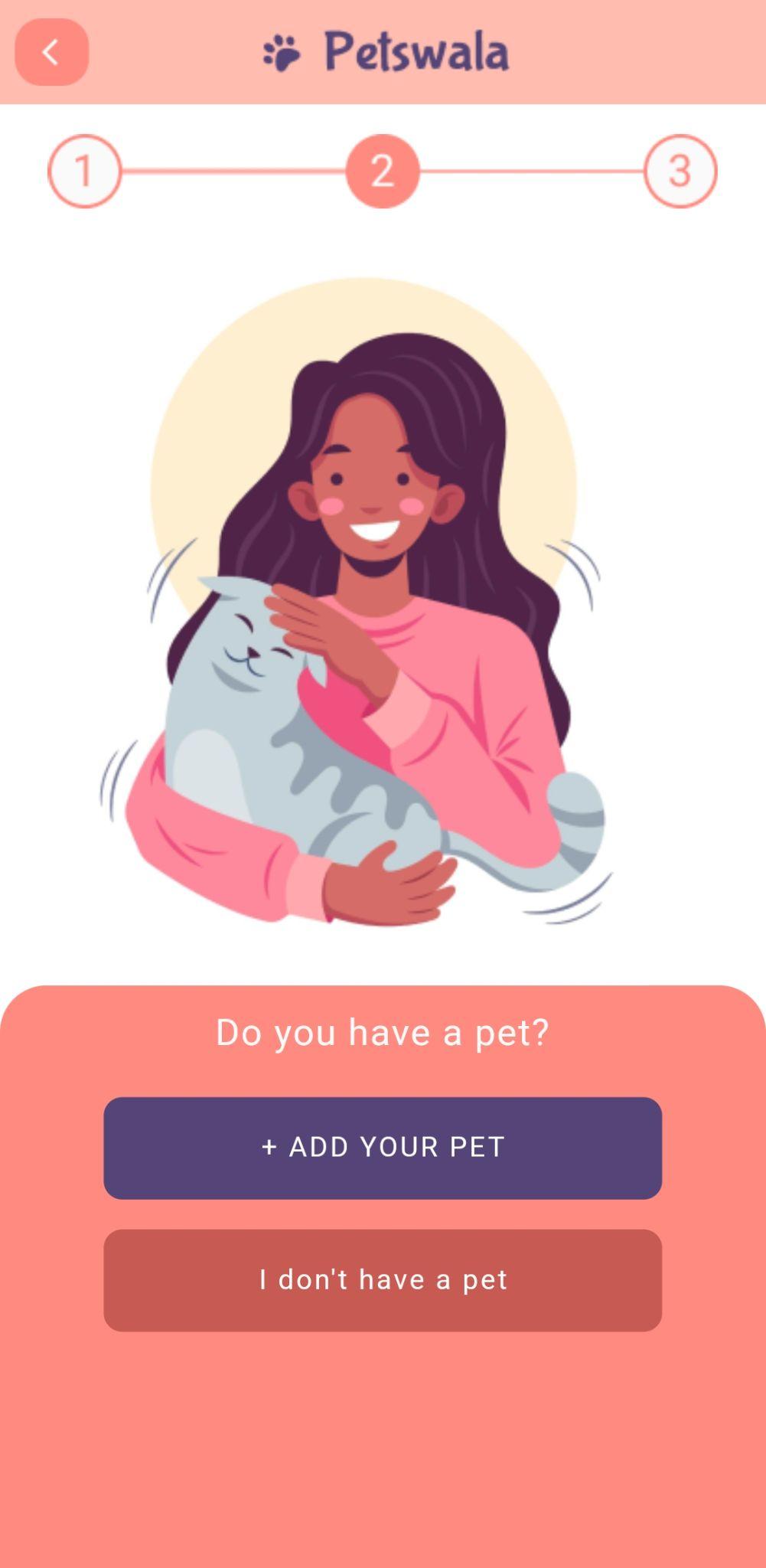
This section would guide the user through the system and help them get started with it. It will act as a user manual for them to understand how to perform the tasks they want to want to perform.

### **Signup**

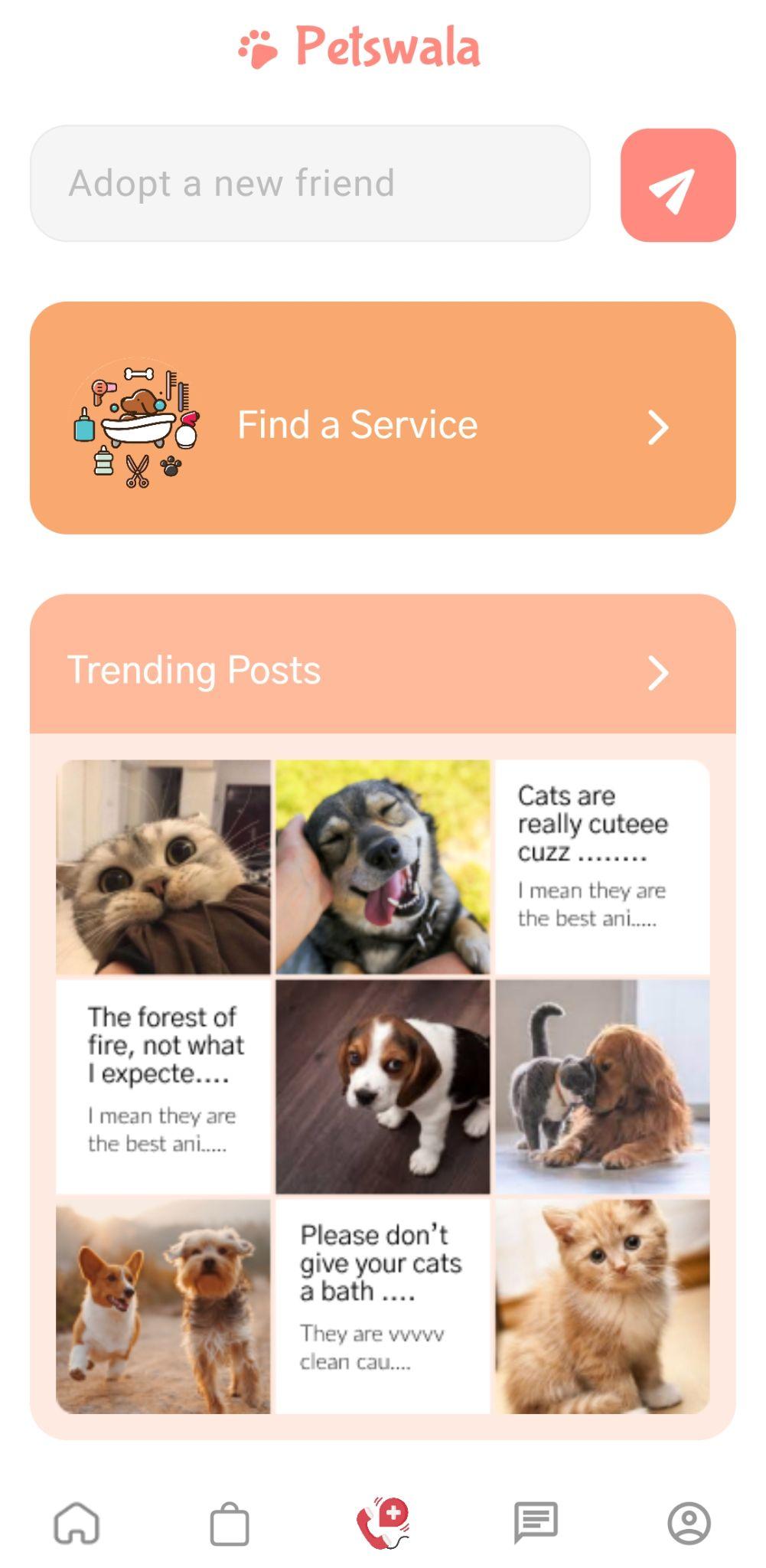
When you open the application, you will be greeted by this page and provided the options of skipping the login process and signing in as a guest or to login if you already have an account or signup if you do not.

To signup, you would tap on the join button and will be transferred to the signup form that would ask you for all the information required.

# 

On the left is the signup form, it would ask for your username, email and password. All of this information you could later utilize to log in (you can also log in through your Google account). The form would also ask you the type of user you want to be. You can be a simple user (one who owns a pet or not), a vet (service provider), a shelter (for rescue requests) and a seller (see Section 2a for more information). If you choose a simple user, you will be asked if you have a pet or not and you can provide the information for this pet. 

### **Home**

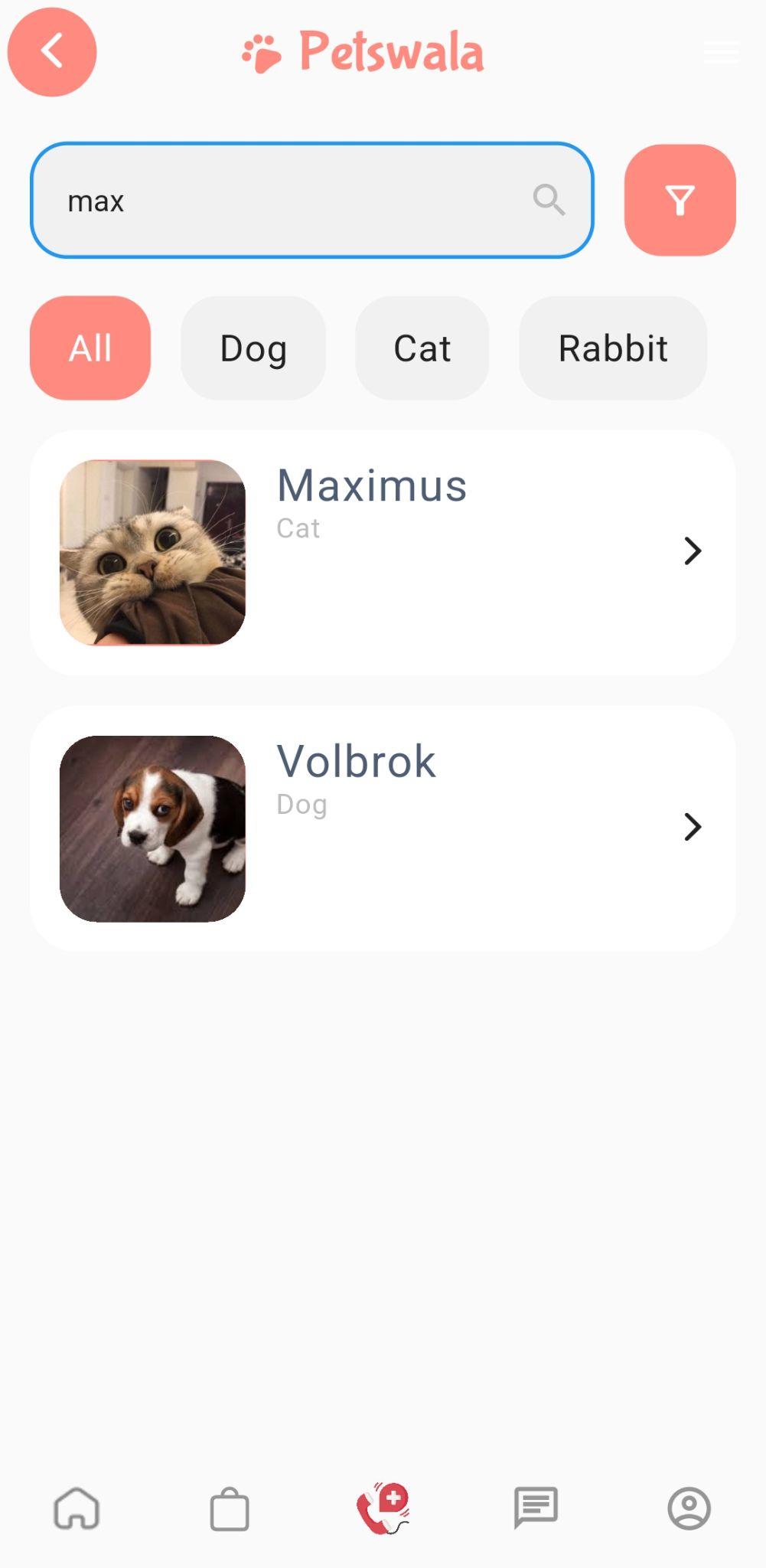


This is the center of all your operations. The portal to all the functionality that brought you to our app, home sweet home. On this homepage, you have a chinbar to navigate to the different functionalities that our app provides; (starting from the left) you have the home page, the marketplace, the rescue system, the newsfeed and your user profile. On the home page itself, you can search for a pet to be adopted, you can see the posts that trending and, you can find services for you pets that you might need.

# 

### 

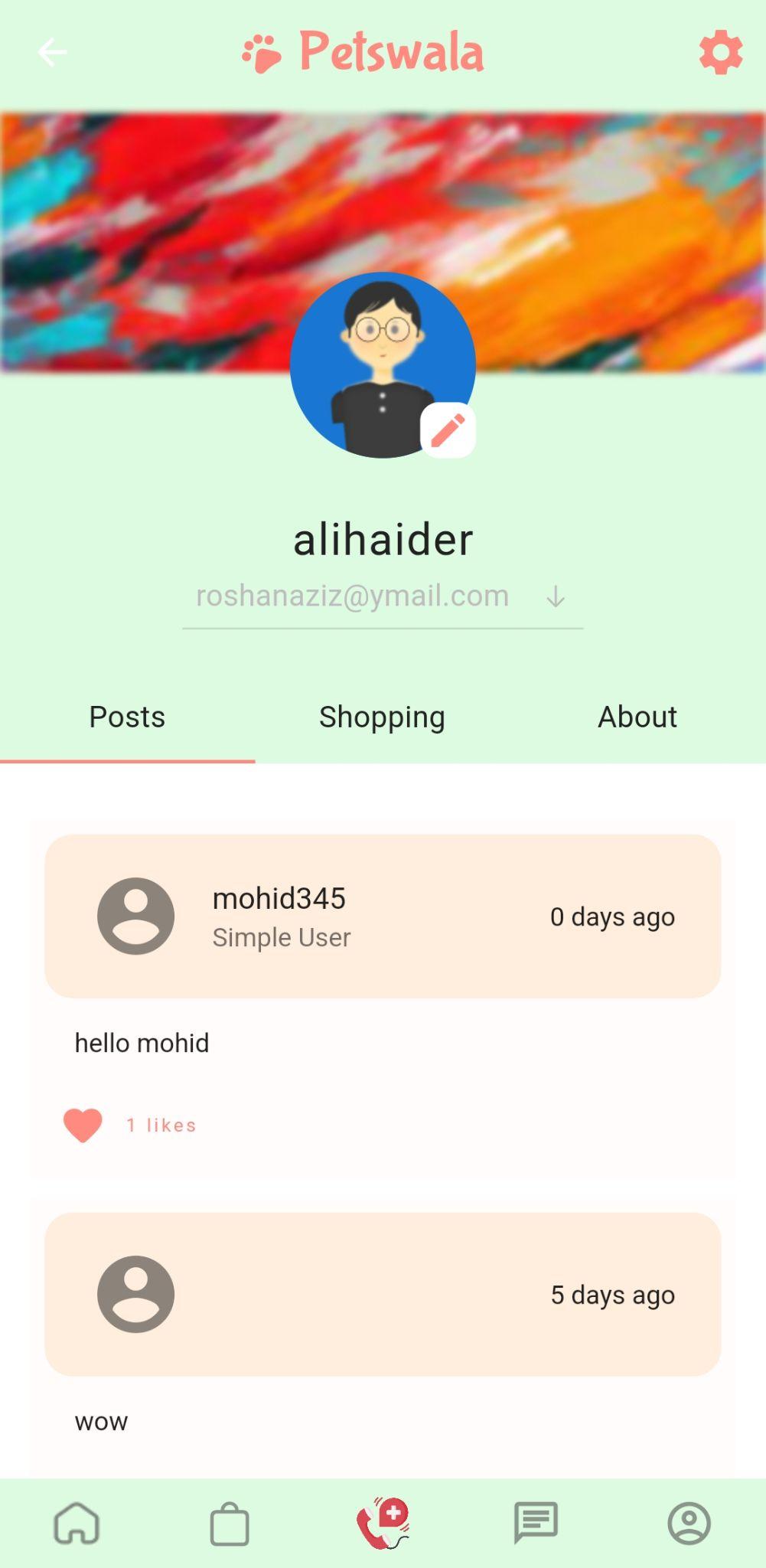
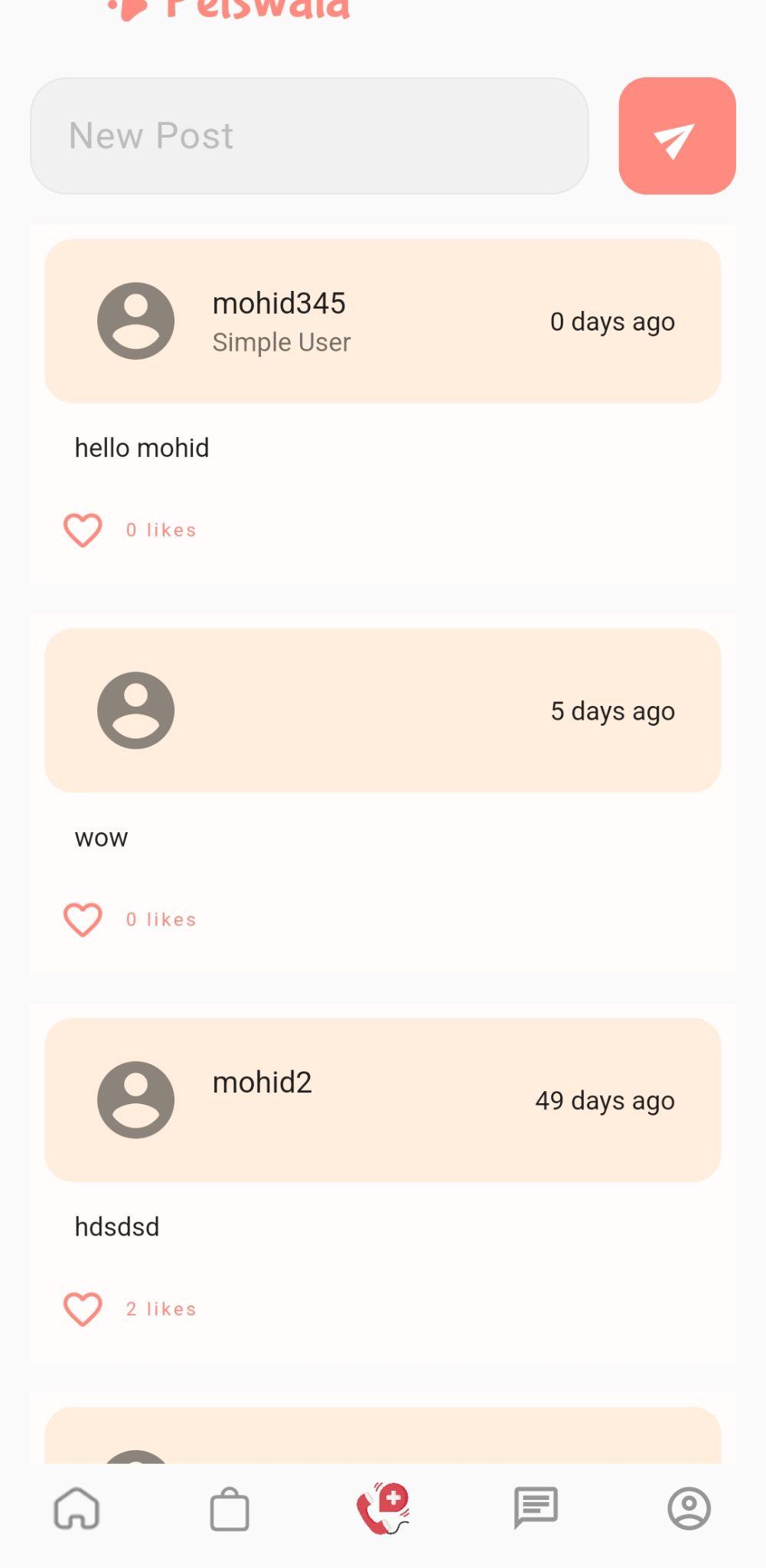
### **Adopting a Pet**



You can search on the homepage for adoption. Once you do, you would be presented with pets that you can adopt (see image on the left). You can filter these animals based on tags provided on the top to select your furry friend. Once you have found a pet or as they say in the pet community, once the pet has found you, you can select that pet and be led to its page. This page (see image on the right) will show you the name of the pet, some information about the pet, its age, type and gender, as well as being presented by its location. You can select the option to chat with the shelter associated with this pet or tap on “Adopt Pet” to directly adopt it.

### 

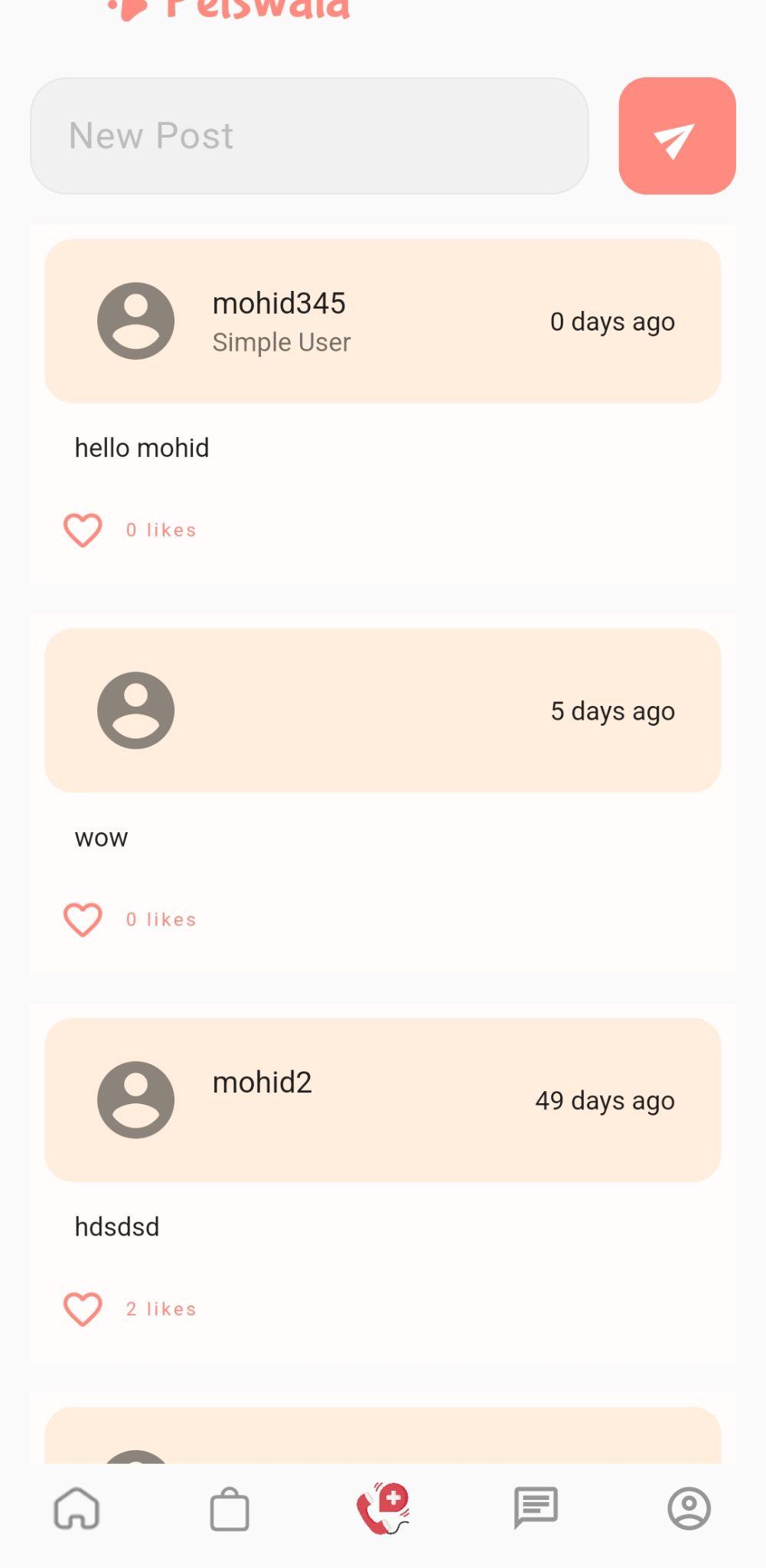
### **Check Your NewsFeed**



There are two ways to access all pets-related posts and pictures that your heart may desire. One is to click on the newsfeed icon on the chin bar (see homepage), this will take you to your newsfeed (as can be seen in the image on the right) and you can proceed to view all the posts and like the ones that you enjoy. You can also select the user profile button from the chin bar (see homepage) and be brought to a multi-select screen. One of the options presented would be that of “Posts”, you can select this option to view all the posts as well.

### 

### **Make a Post**



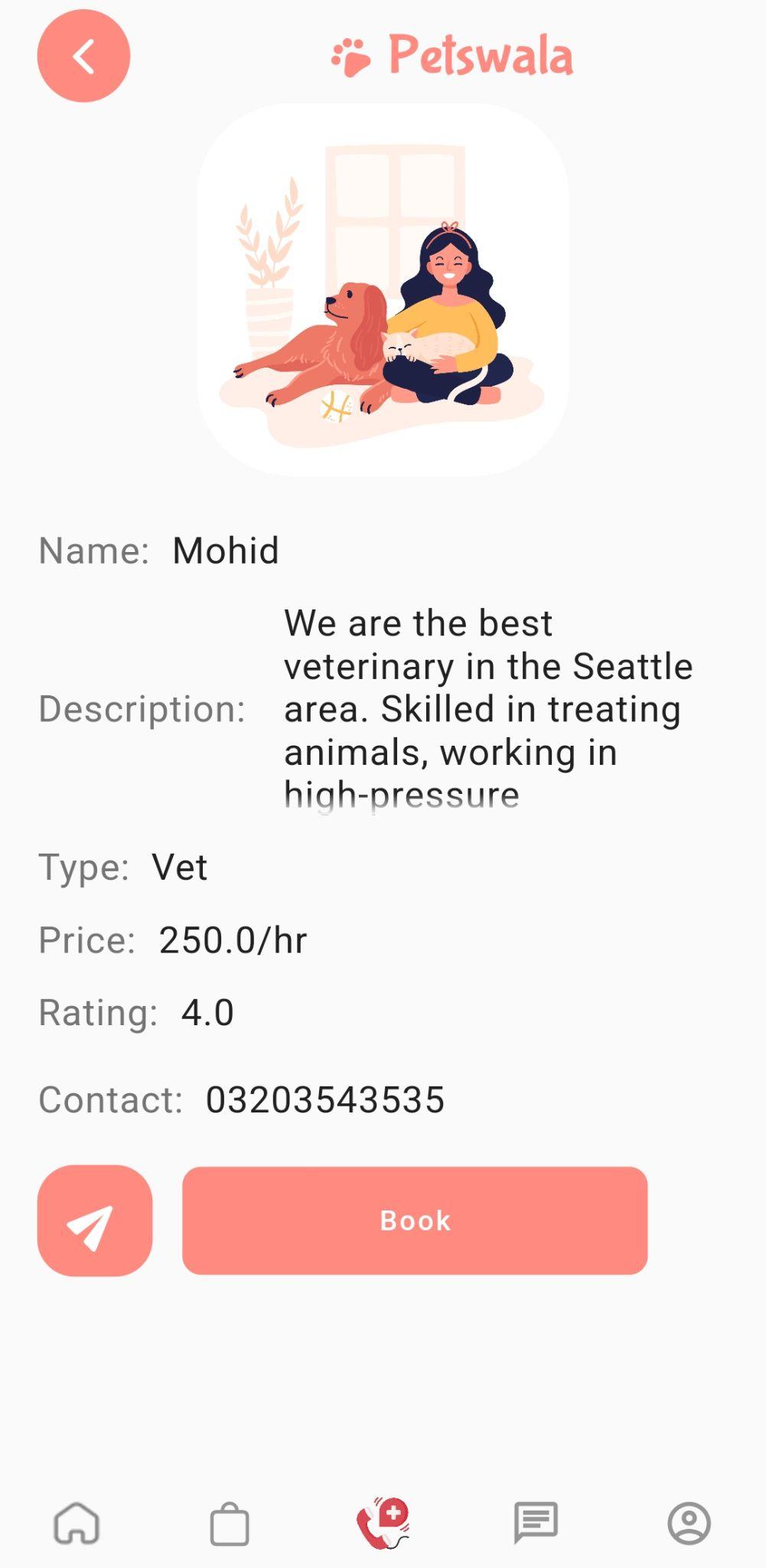
After tapping on the newsfeed icon on the chin bar, the news feed page will open (see homepage). You can tap on the text field that says “New Post” and type out a text post, you can tap the send button next to it to send it.

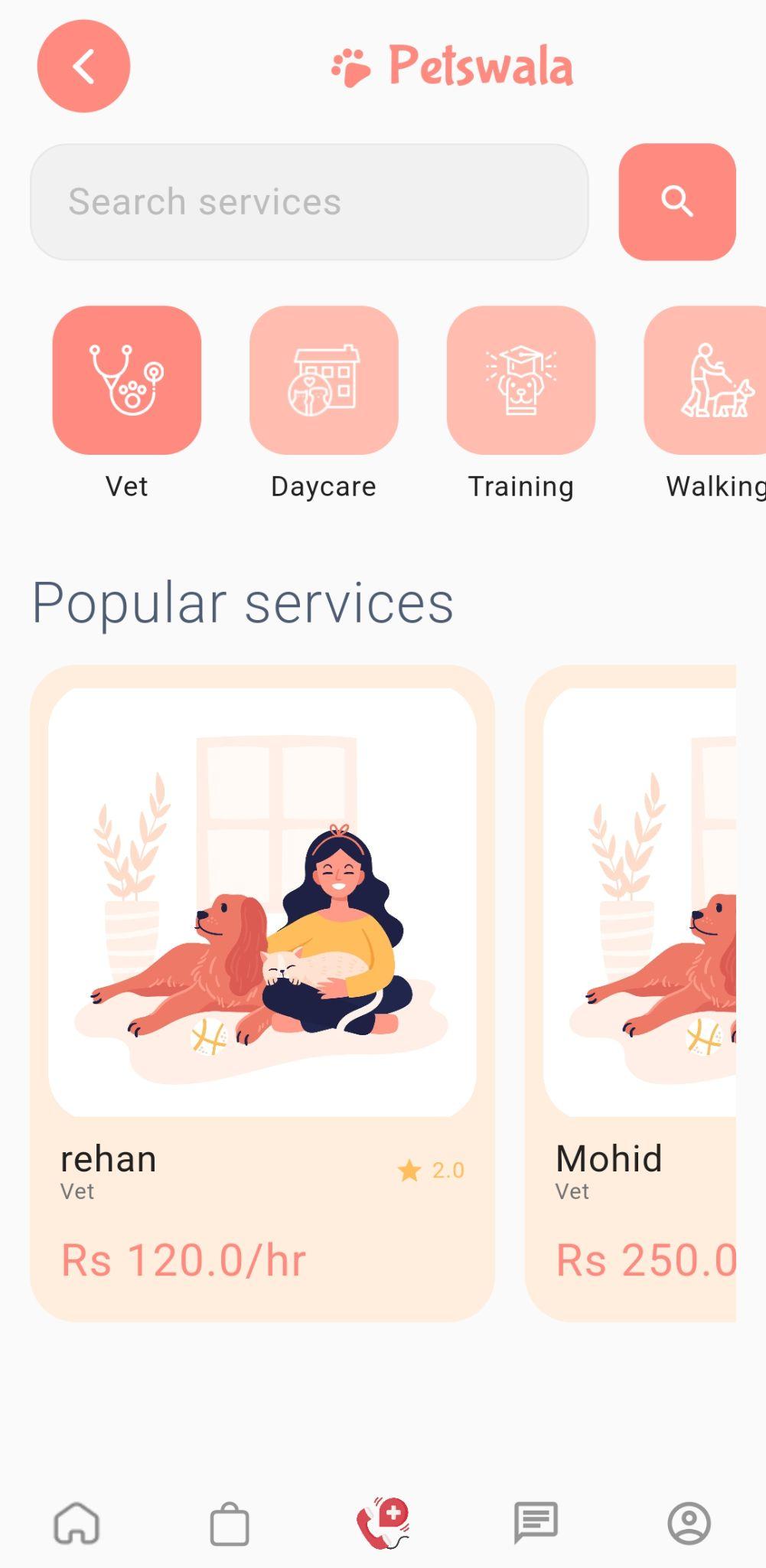
# 

# 

### 

### **Finding a Service**



You tap on the Find a Service button on the homepage (see homepage). This will take you to the page displaying all the services that have signed on to the app (see the image on the left). You can then search for a specific service, scroll through a list of popular services or filter the services further by using the tags under the search bar. After narrowing dowm your choice of service and selecting the service you need, you tap on that service and are taken to the page of that service (see image on the right). On this page, the information for the service provider is displayed, this includes; the name of the service provider, the description, the type, price, rating and contact number. You can select to either book the service provider directly by tapping the “Book” button or chat with this service provider by pressing the chat button. 

### **Post Rescue Request**

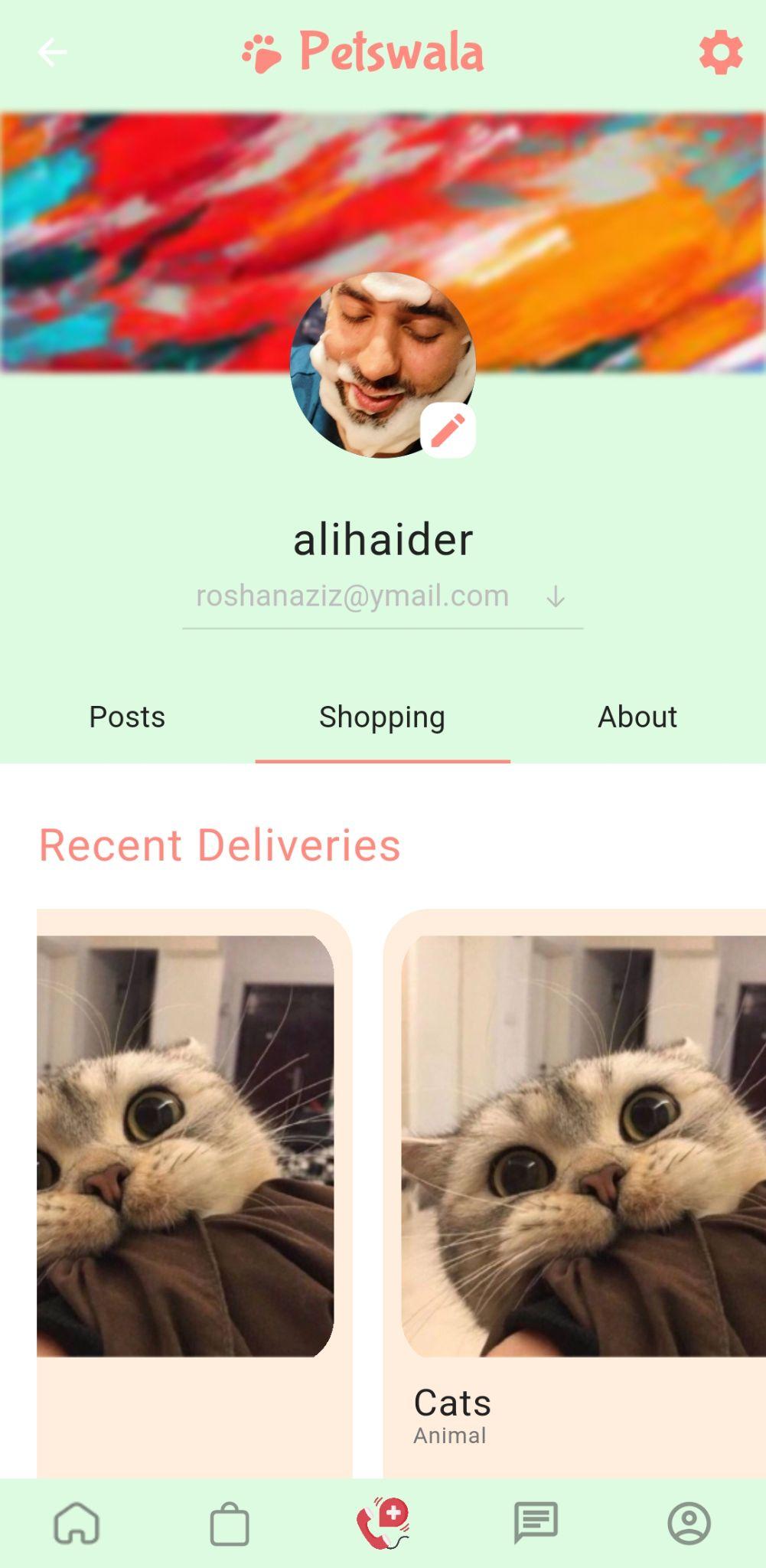
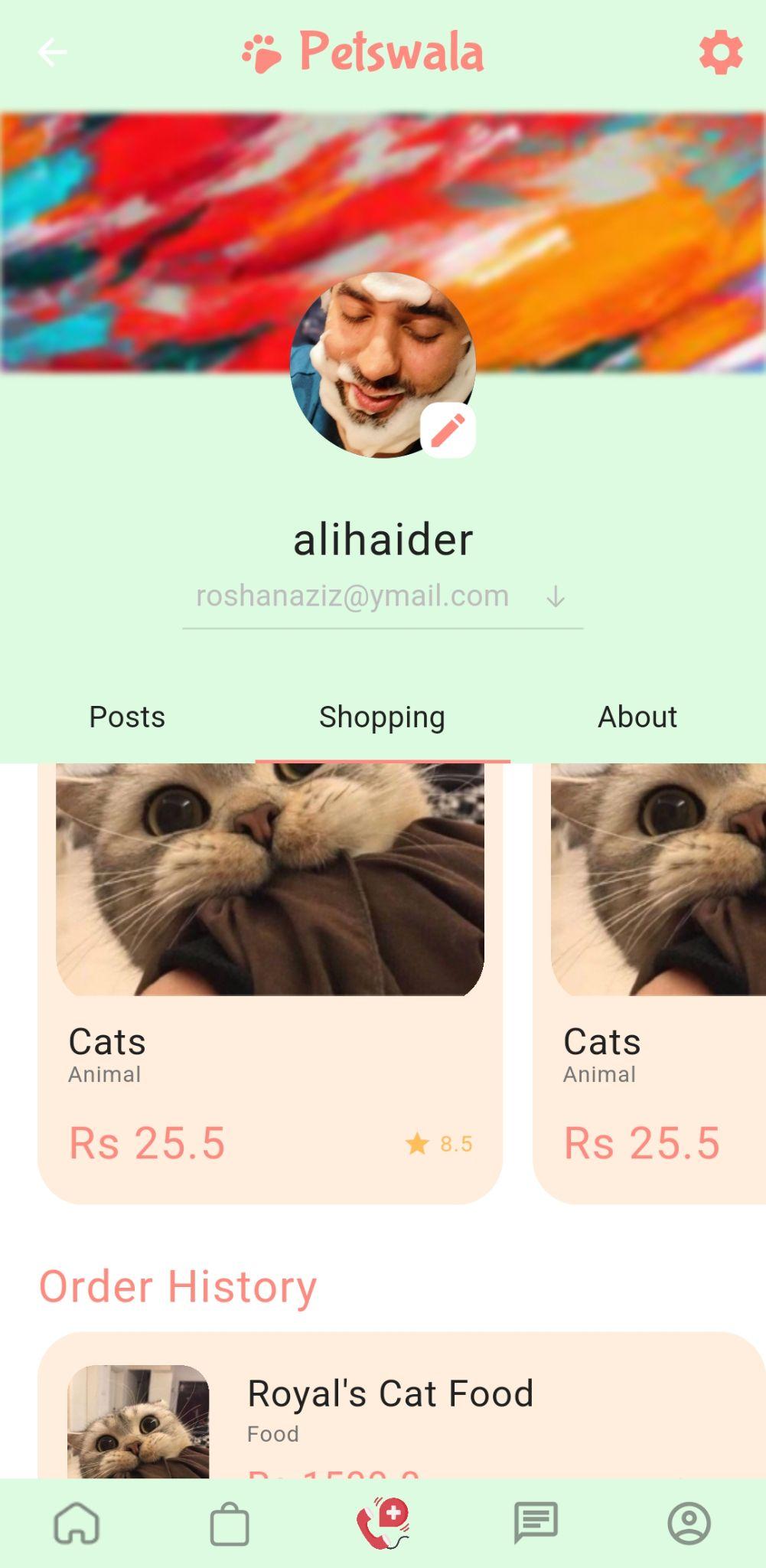
### 

# 

# 

To save the life of one of the stray animals, you can participate in helping to identify and provide coordinates for their rescue. The process is extremely simple and fast to make it as easy for you as possible. You tap the rescue button on the chin bar (see homepage). You will be taken to a screen presenting you with the option to report a stray animal or to check for any request of one (see image on the left). You can select to report a stray animal and will be taken to a map that will pinpoint your location to learn of the location of the stray animal (see the image in the centre), you can confirm this location. After this, you will be asked to fill in a small amount of info, identifying you and the animal (see the image on the right), you will be asked for your phone number to be notified of the rescue’s success, you will also have to input the type of animal and the the description of it.

### **View Purchased Items**



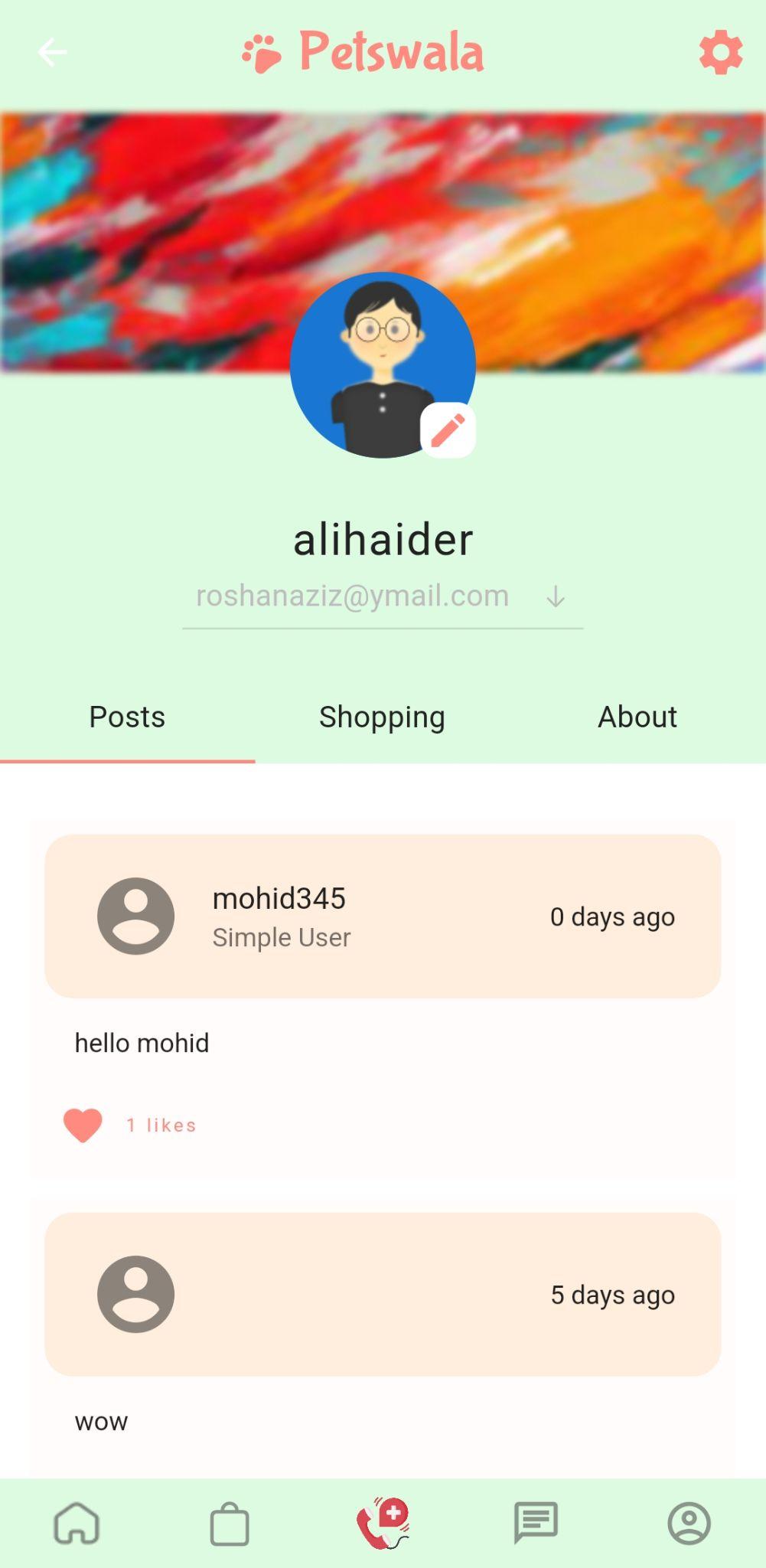
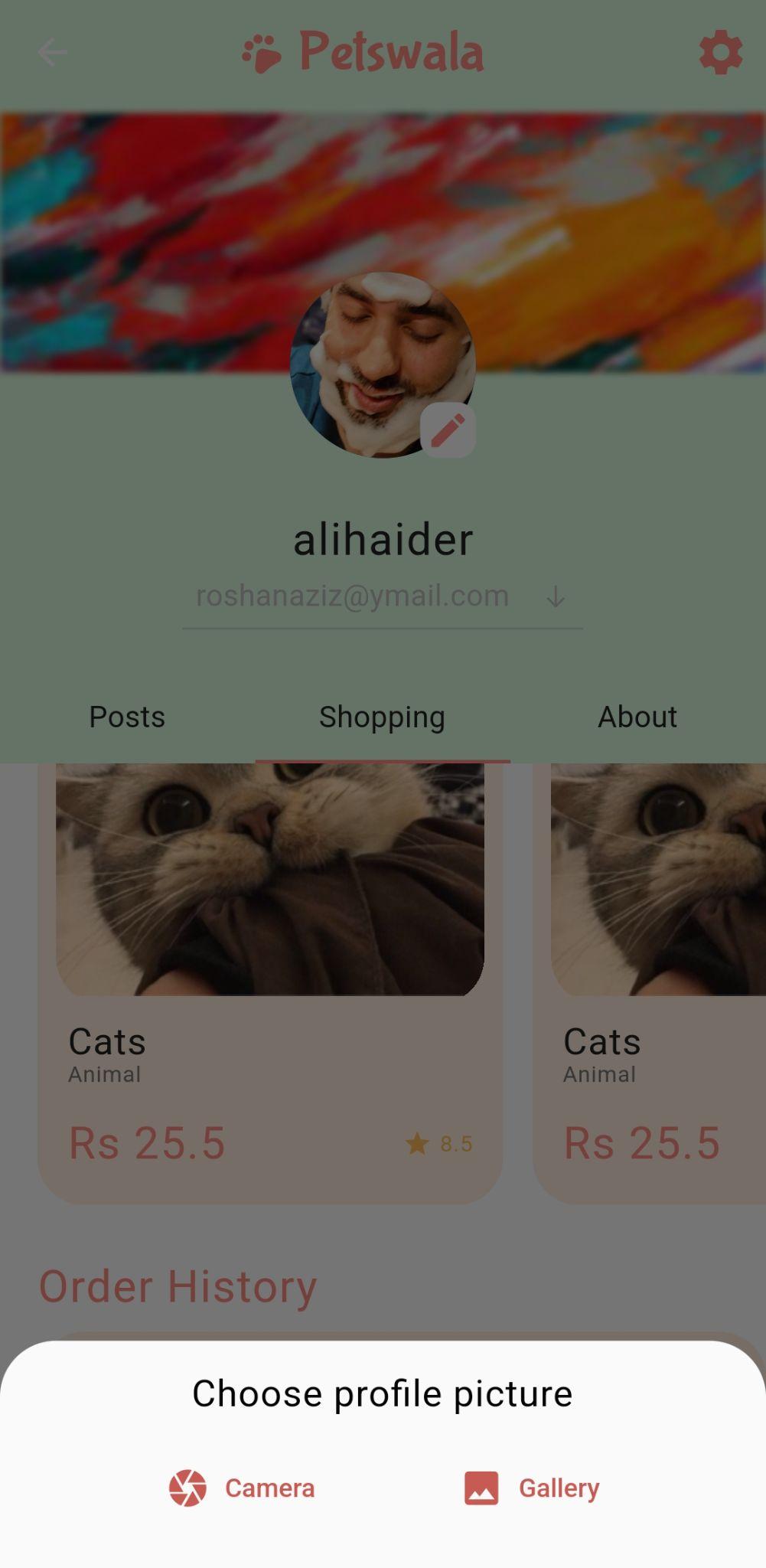
Tap on the user profile button on the chin bar (see homepage). You will be brought to the user profile page (as seen on the image on left), you would be looking at the posts when you arrive at this page, and tap on the shopping button on the multi-select bar. Then the bottom half of the page would change to show you your recent deliveries (as seen on the image on the left) and your order history (as seen on the image on the right).

### 

### 

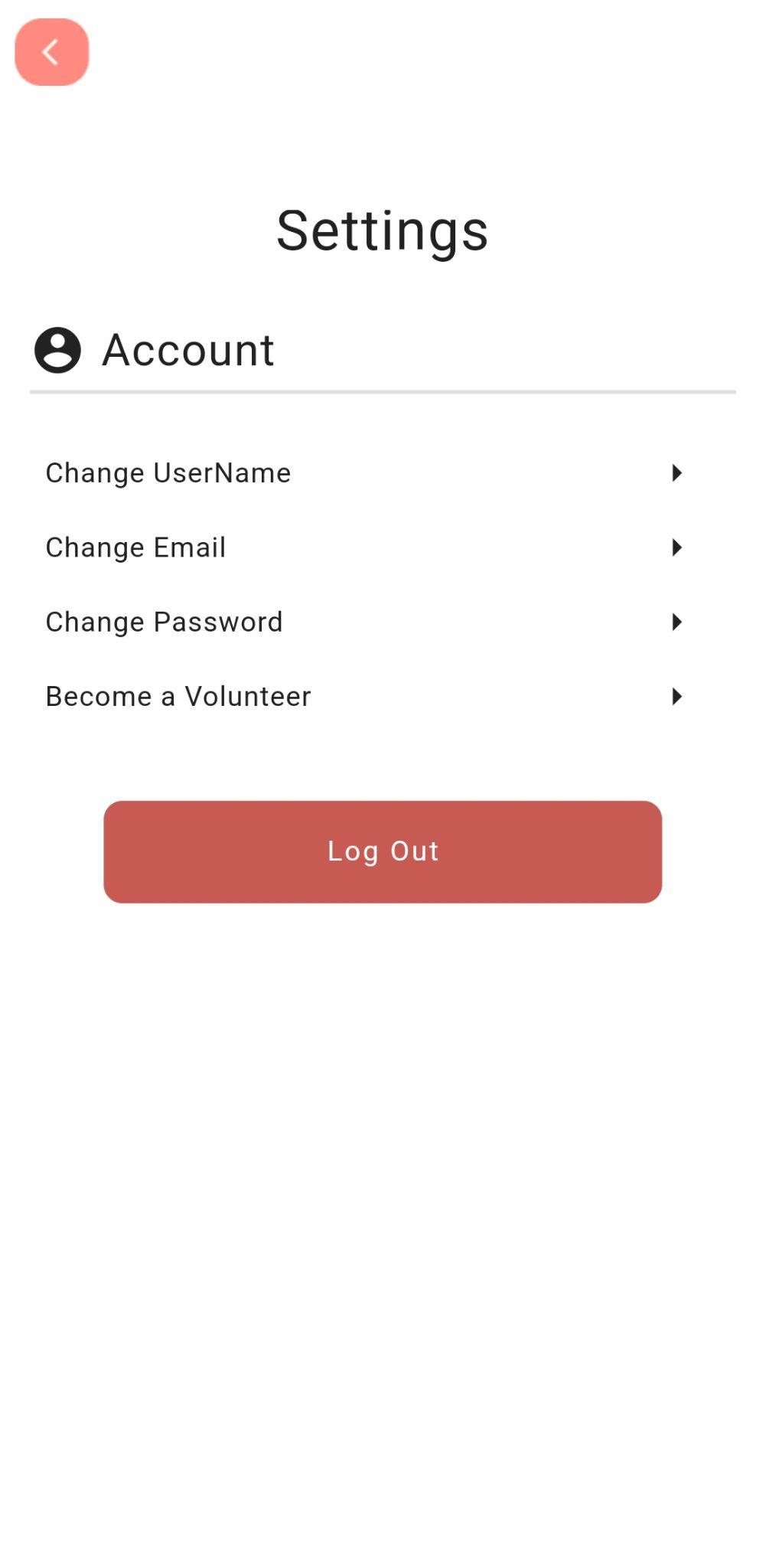
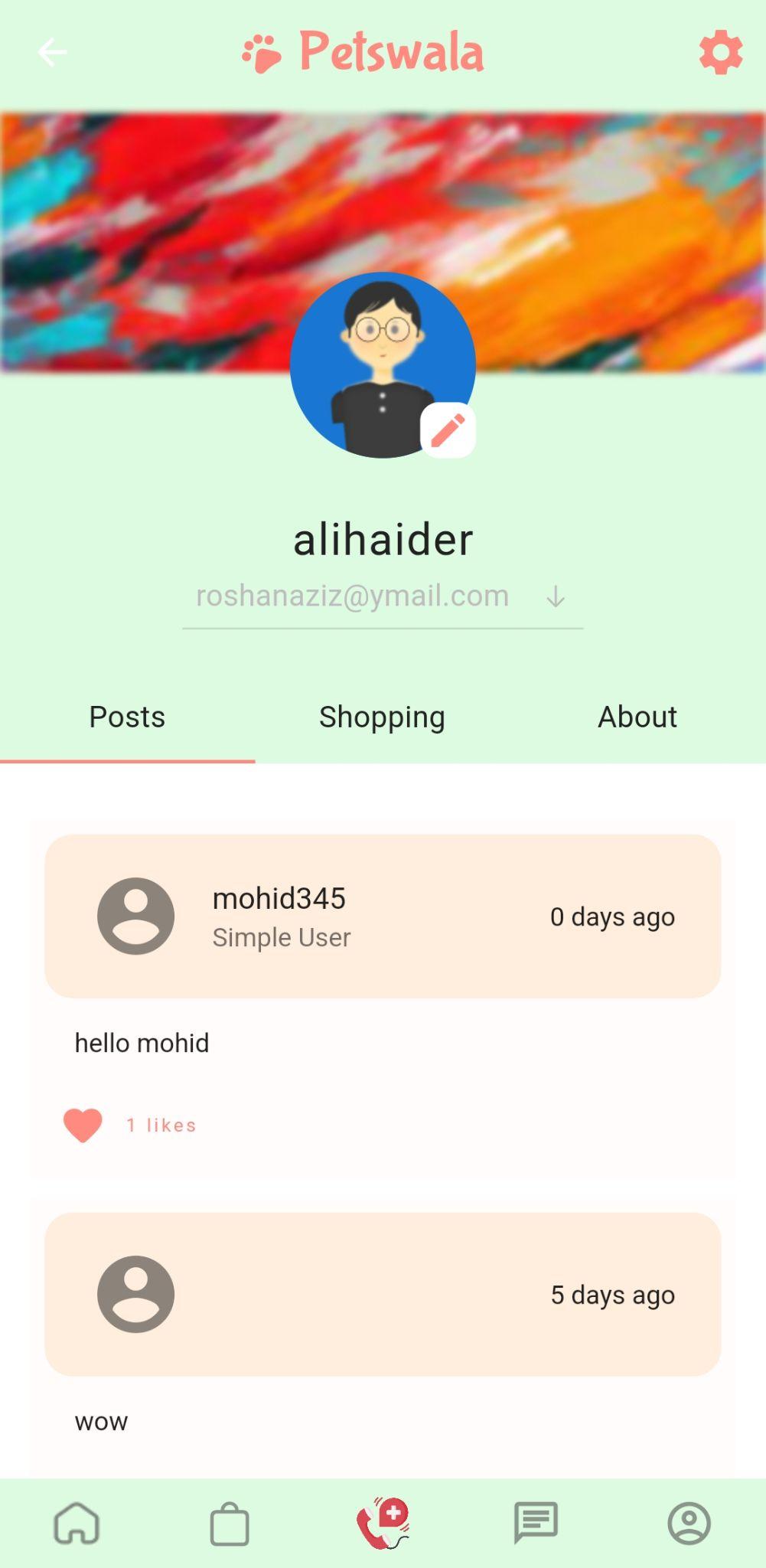
### 

### **Edit Profile Picture**



Tap on the user profile button on the chin bar (see homepage). You will be brought to the user profile page (as seen on the image on left). You will then tap on the profile picture itself and be presented with the option to either take a picture through your phone camera or select one from your gallery (as seen in the image on right). You can then take or select a custom image for your profile.

### **Edit Profile Info**

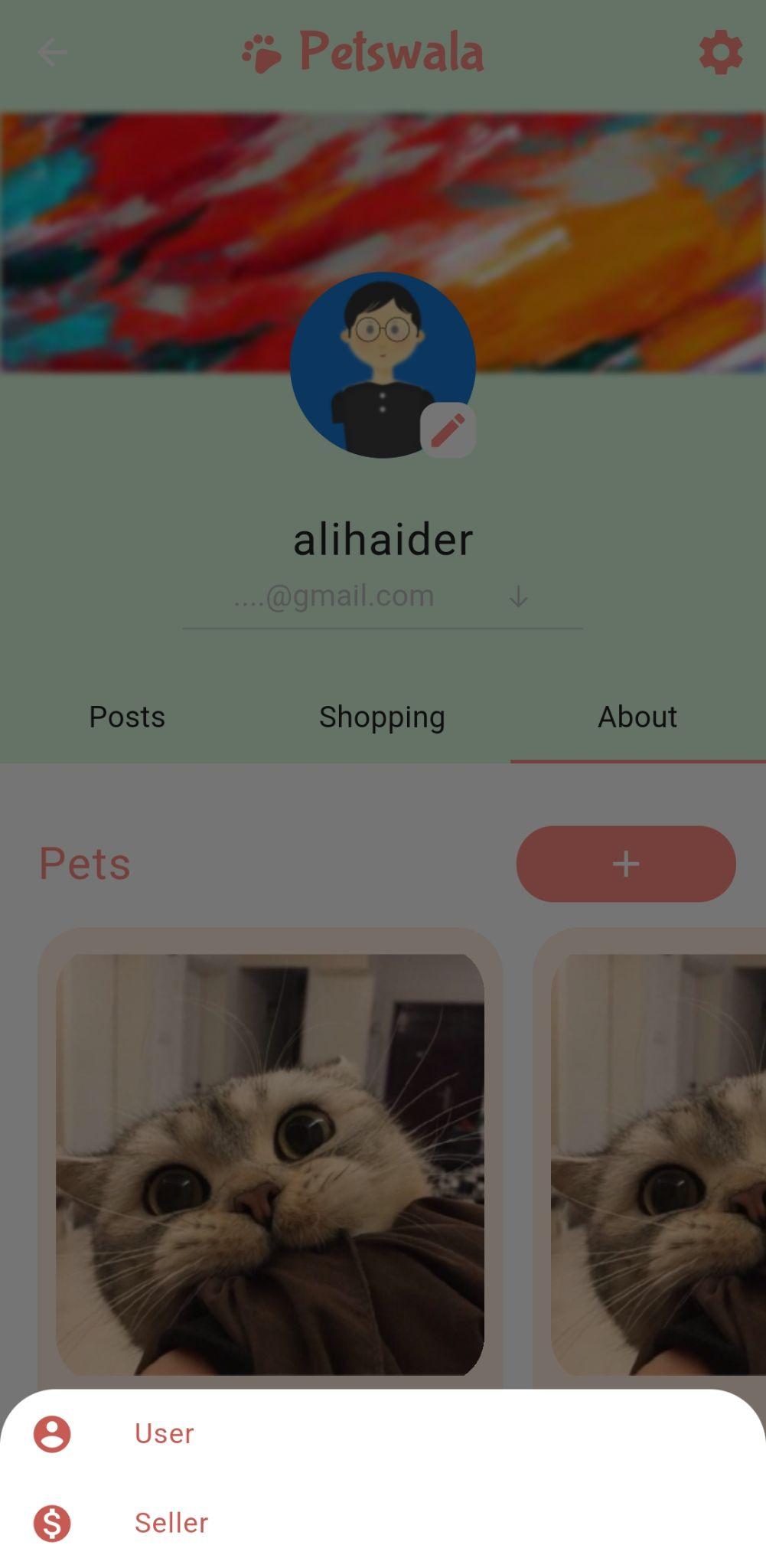
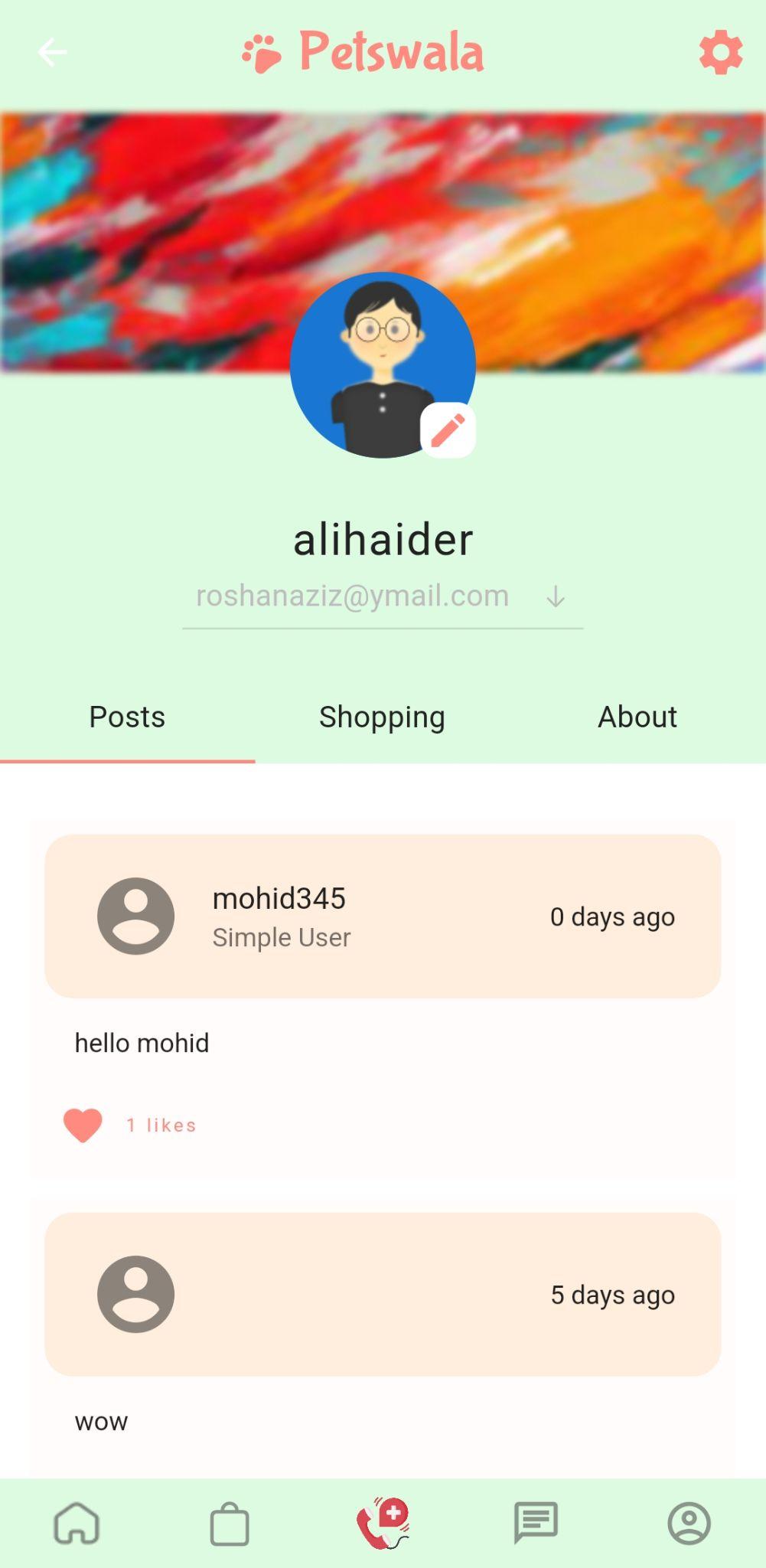


Tap on the user profile button on the chin bar (see homepage). You will be brought to the user profile page (as seen on the image on left). You can tap on the gear icon to go to the setting and edit information regarding your profile.

# 

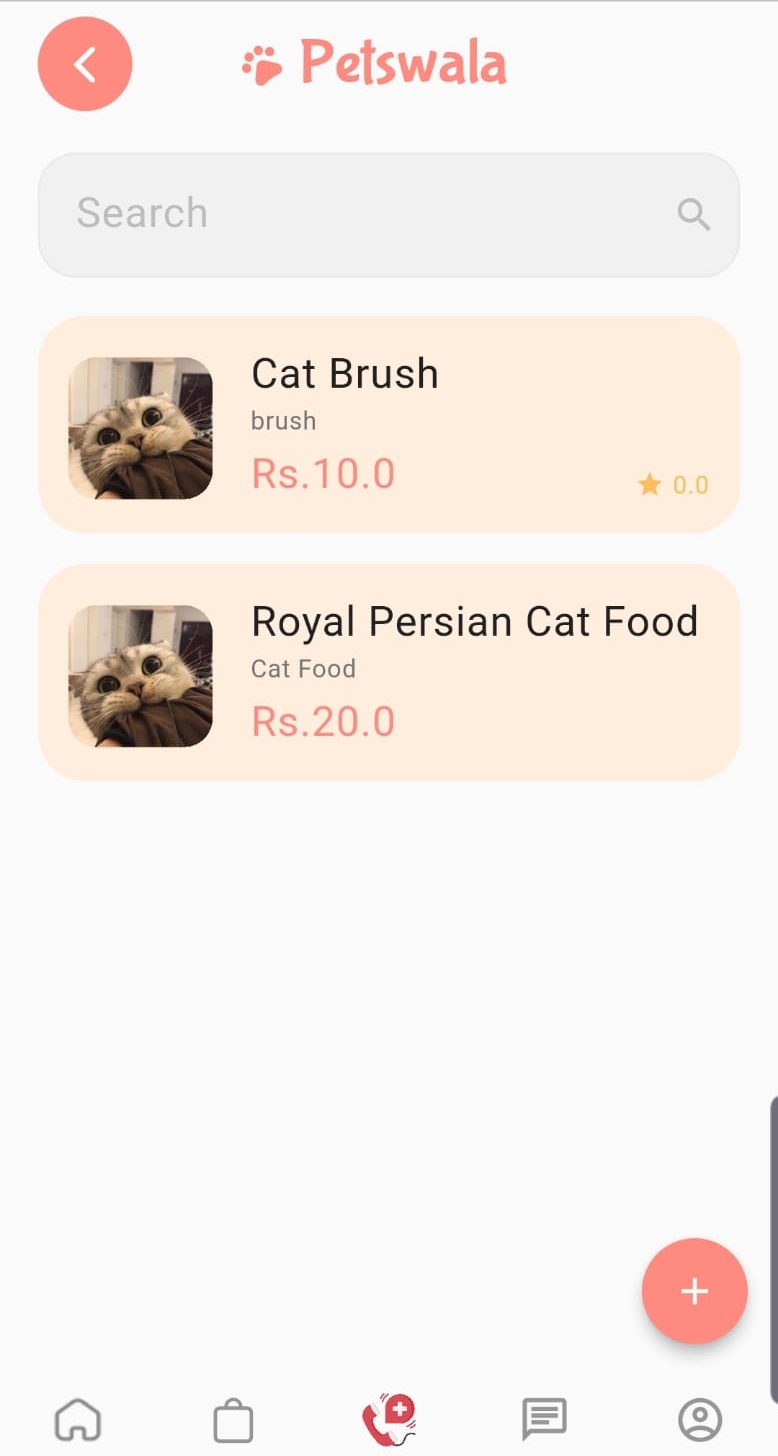
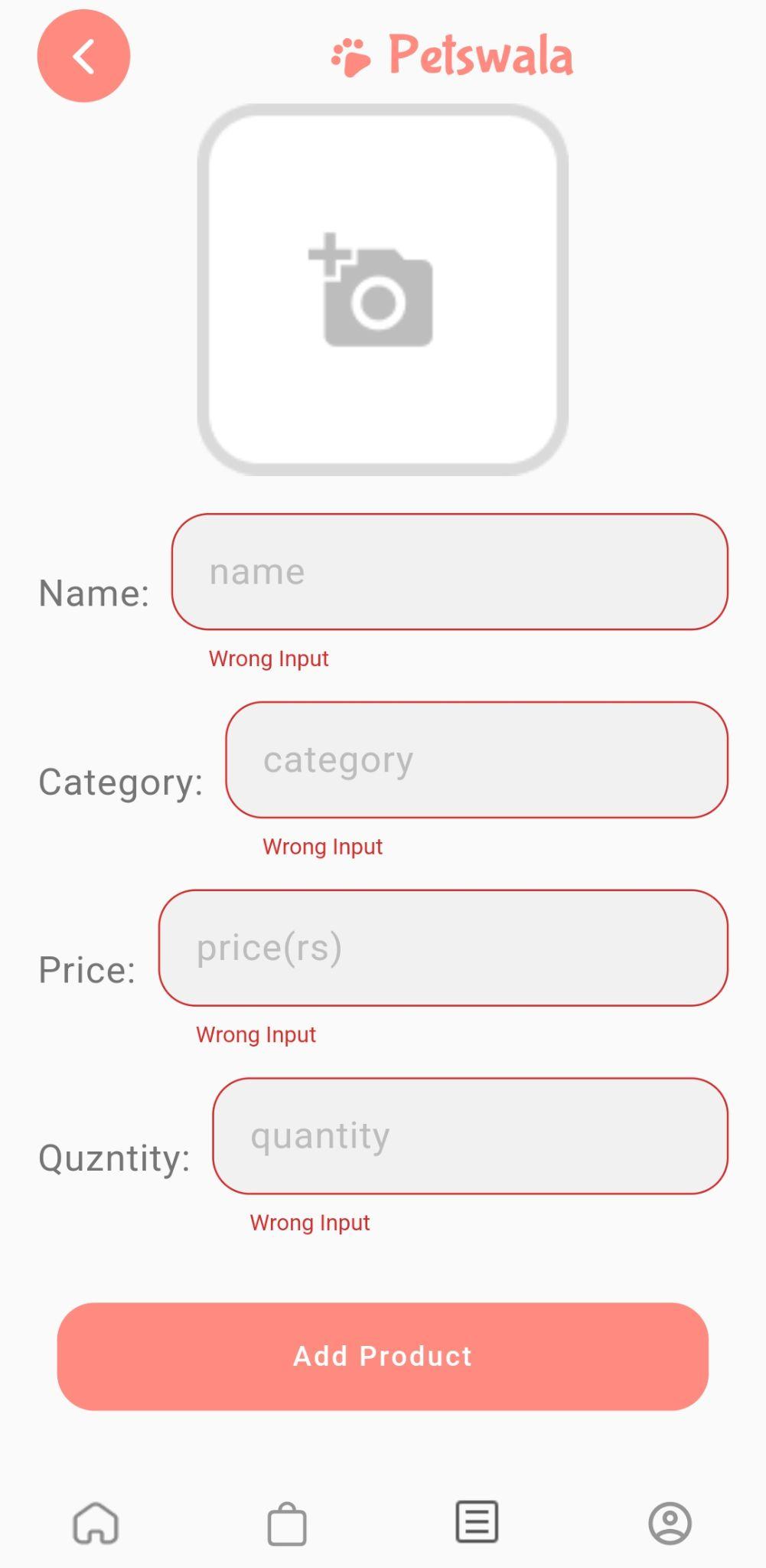
### 

### **Access Seller Homepage**



Tap on the user profile button on the chin bar (see homepage). You will be brought to the user profile page (as seen on the image on left). You can tap on the email address under the profile picture and be presented with the option to select the type of user you want to access at the moment (see the image in the center). You can select the seller option and be moved to the seller-side home page (see the image on the right). Here, you will be shown the recent transaction on your store and latest metrics for your stores performance. You can also access other functionalities available for the seller.

### **See Product List (As Seller)**



Once you have arrived at the seller home page (see Access seller homepage), you can tap on the button with the warehouse symbol. This will take you to the inventory page. The inventory page will be empty unless you have added anything, so you can tap on the “+” button to start adding items (see the image on the right). Once you tap this button, you will be taken to a form to enter the details of the product (see the image in the center). After you have entered the details of the product, a product tile will be formed which can show up on the inventory page (see the image on the right).

# **8.** **Project Security**

This section will refer to the security aspects of the app. Firstly it will go over the potential threats that the app faces through its various modes of interaction. It will detail some of these threats and mention the potential losses the app and its user would face if these threats are not mitigated in an appropriate manner. Finally, it will go over the possible controls which we have in place to deal with these threats as well as other controls which can be explored further.

## **a.** **Project Threats**

### **Sensitive Data Exposure**

Our App stores sensitive data for a lot of users. This includes live location data, credit card info, as well as addresses, passwords, tokens e.t.c. This data needs to be protected so that it does not fall into the wrong hands and cause potential losses. Moreover, tokens and passwords, if in the wrong hands, can be extremely detrimental.

**Broken Access Control**

Broken Access control is when an entity can access resources or perform functionality which they are not allowed to perform. Our app has lots of different user groups. These involve vets, pet shop owners, common users, volunteers, shelters e.t.c. If someone can gain access to functionality they can put the whole system at risk due to impersonation or worse.

### 

### **Insecure Communication**

Even if both ends of our Application are perfectly secure; although that would be a bizarre claim; If the channel of communication is not secure, then we can never trust the data which reaches either end-point. It could have been intercepted and manipulated with and can cause problems on both the user device and the server. Therefore it is important to ensure that the data channel is secure.

## **b.** **Potential Losses**

### **User Monetary and Data Losses**

This can be caused due to Sensitive Data loss. When a user loses their data, they obviously do face data loss. They may lose precious data, as well as their card details. If their location can be tracked, it can be a security threat as well. If their credit card info gets leaked, people can use it to purchase things, thereby inducing monetary losses as well

### **Business Reputation Loss**

Due to the inability to keep user data private as well as failing to prevent improper usage of the app due to broken access control, users of our app might be vary of using it. Bad Publicity will ward off any potential users and thus cause business reputation loss.

### **Litigation Losses**

Users who lose their data and/or are adversely affected in other ways due to our app can sue us which can cause lots of problems in terms of financial and time losses.

## **c.** **Security Controls**

| **Control** | **Type** | **Description** |
| --- | --- | --- |
| Server side authentication and authorization + Access Management(JWT tokens) or google authentication. | Detective | JWT tokens and oauth2 based authorization detects unauthorized requests as all users can be identified by unique tokens which are verified whenever requests are made. |
| Secure and Encrypted Storage (Flutter Secure Storage) | Protective | The storage of important data such as token and sensitive user data needs to be stored in encrypted storage. This involves document encryption for passwords and the like in our database as well as encrypted storage client side. Flutter secure storage is used client side to protect all such data. |
| Using SSL/TLS certificates | Protective | SSL/TLS certificates are important to ensure that all communication channels are encrypted and secure so that no third party or proxy can intercept our data and manipulate it. |
| Input validation (real time + server-side). | Detective | Validating all user inputs in terms of their expected inputs both server side and client side is important as any unexpected user input can cause bugs. If we validate all data input then we can be sure that we are dealing with data which our system is designed to handle. |
| API calls authenticated server side. | Protective | It is important to authenticate all API calls server side and have a set architecture for that. This is to ensure that no communication is done outside of the loop as prescribed by our architecture design. Unauthorized communication with APIs and services can cause data mismatches and therefore undefined behavior which can lead to vulnerabilities. |

## **d.** **Static and Dynamic Security Scanning Tools**

### **Dynamic Testing**

#### **Bloc\_Test With Flutter**

Flutter bloc\_test involves the individual testing of all business logic elements. All events are handled and state is managed in these components making it a necessary part of all flows. Bloc\_test is used to design tests to individually test those blocs and isolate state and event handling from the UI.

#### **OWASP ZAP**

This is a tool to test for possible vulnerabilities our app might have in terms of its interaction with the internet. It acts as a proxy between the app and the server. And intercepts all traffic. It can be used for traffic analysis to make sure that all communication is being appropriately carried out. It can also be used to manipulate data and test for weak spots in our Application security

### **Static Testing**

#### **Dart Analysis**

It is a built-in static testing tool to test for vulnerabilities in our dart code. It provides null-safety as well as other compile time checks to ensure our app does not have any weak points.

#### **NodeJSScan**

It is a static testing tool for NodeJS. We have written our RESTful API in NodeJS and as such, this tool helped us find possible vulnerabilities in our code.

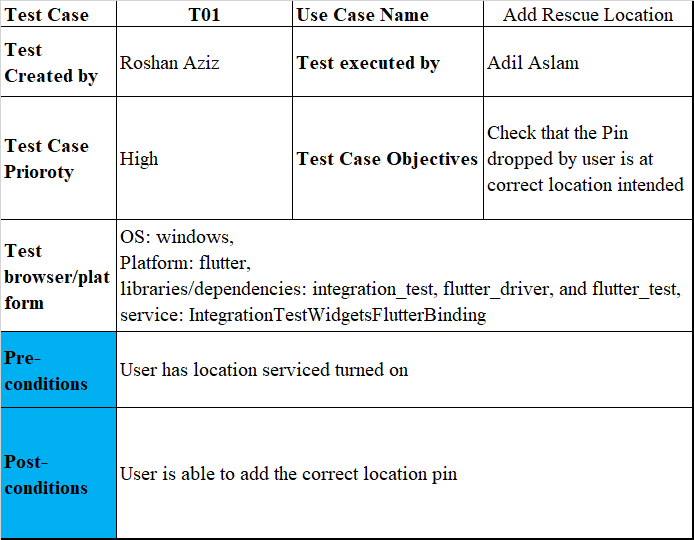
# **9.** **Risk Management**

## **Potential Risks and Mitigation Strategies**

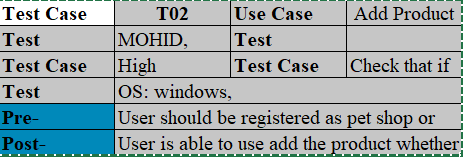
| **Sr.** | **Risk Description** | **Mitigation Strategy** |
| --- | --- | --- |
| 1. | User Engagement: After the release of the product, there might be a difference between stakeholders expectations and users actual engagement. | Incorporate user feedback in updates. Project will be designed in a way that is adaptable to changes. Moreover, testing, surveys and frequent updates will help avoid it. |
| 2. | Requirements Inflation: With the progress of the project, more requirements can be identified which were not stated in the beginning. | Involvement of customers at an early point in the development of the project, so that most of the requirements can be identified as soon as possible and relevant changes and estimations can be made. Moreover, frequent reporting in between the team would also help mitigate this risk. |
| 3. | Holes in software security: The application could have holes in it’s software security component and could be an unsecure application. | Use the appropriate APIs to have secure software. Additionally, plan your software architecture in a way that you have secure software. Logging the application use can also help in identifying these holes. |
| 4. | Unpredictable external risks: Sudden changes in the market, there can be a competitor with more resources and fast growth, some changes in consumer behavior and priorities, or any new laws. | Thorough research about market, competitors and laws to mitigate the chances of such changes affecting the product. Enhance user experience and good marketing of the product. |
| 5. | Lack of professional experience: Since we are students aiming to make a professional project, we are still learning. This inexperience could result in various problems. | Stay constantly updated on if we are on the right track by seeking help from seniors and the instructor and researching on Google. |
| 6. | Change in platform software: Due to the rapid changes in technology, changes can occur in the platform that we are developing for. Our application might risk falling off the cutting edge of technology and design. | Keep up to date on the release dates of these platform updates and either cater them during development or have an update path figured out beforehand. |
| 7. | Poor Management: Due to poor communication within the team, the developers might not share the important key information which is necessary for a smooth release of the product. | Scrum, weekly reports and frequent meetings. |
| 8. | Resources: Unable to obtain required resources | Because the project does not require many resources, it is critical to obtain them as early in the development phase as feasible, and alternate resources, such as local test servers, should be prepared for. |
| 9. | Inaccurate Estimates: The estimates of number of users can be inaccurate. | We will do thorough research and conduct surveys to estimate the number of different types of users. |
| 10. | Code Quality: The lack of time or rushing, might result in poor code quality. | Leaving enough time to check for bugs, and do thorough testing. |

# **10.** **Testing and Evaluation**

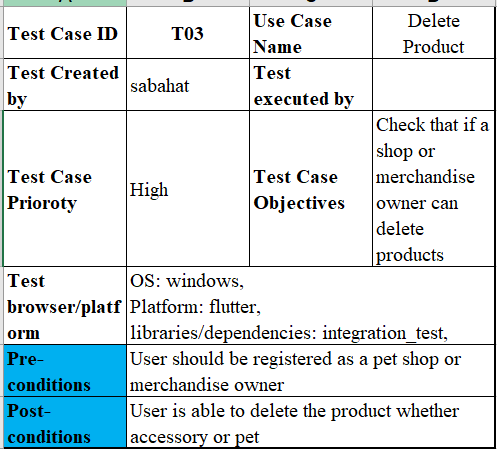
### **Test Case 1**



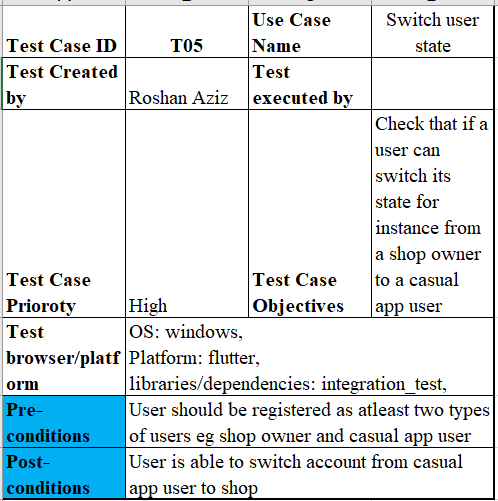
### **Test Case 2**



### **Test Case 3**



### **Test Case 4**



# **11.** **Deployment Guidelines**

<https://github.com/mohidyousaf/SPROJ.git>

<https://github.com/mohidyousaf/RESTful-Api.git>

https://intense-badlands-24863.herokuapp.com

PetsWala is an android application that is build in flutter, we could implement the front end as well as backend in dart, however we build a RESTful API to separate the client and server logics, so we deployed this API in heroku, and all the requests from apk are handled through the deployed Server, in this way it makes it secure as well as make the project scalable. Similarly we set up our backend on Mongodb, so our RESTful API is connected with a Mongodb cluster named as ‘PetsWala’, and we used Mongoose and Express as middlewares to update all the user initiated information based on specific events to our database. All the external APIs are also deployed on respective servers, for instance we have OneSignal API for push notifications, GetStream API for chat system, Google Maps and Google Authentication API that are all configured in RESTful API.

# **12.** **Conclusion**

## **a.** **Summary**

This application is created to cater for the growing community in Pakistan. This community is one that revolves around animals and pets. The project comprises a mobile application catered for the community involving; pet owners, pet shops, sellers of pet accessories, veterinary doctors, pet rescue volunteers, and pet shelters/rescue teams. The application exists in the domain of a social media application, as well as an e-commerce one. The application aims to connect these individual community members and bring them together under one roof, this is done by providing certain functionalities, pet owners would be able to search for different accessories, as well as find good veterinary doctors for their pets, making their lives much more comfortable and better. They would also be able to interact with other pet owners and share photos and stories of their own pets as well. They could share helpful information with other pet owners. Any user of the app can also help with the animal rescue process by reporting pets to be rescued to the relevant rescue teams in the area.

## **b.** **Challenges**

There were some major difficulties we faced while implementing the core functionalities of the system, majorly when we had to manage the states in the sub systems, as we started working on new sub systems, we had to manage the states used in those system efficiently so they can be shared as well, while we don’t need to put the states in top of the hierarchy for the entire subsystem and should be only used where needed, this not only makes things cleaner but it also makes our apk memory efficient and scalable as API calls will be made at specific context not at the start of our app for the very first time, so we researched accordingly and implemented the bloc architecture and its details can be found in the document as well. Other than that, we found some communication and task management issues as well, and for that we setup Clickup and github planner, and set task dependencies and deadlines to cater the issues. This also involved proper feedback and evaluation by our team members.

## **c.** **Future**

This Project was conceived as a platform to help solve the stray animals crisis in Pakistan. As such, the way forward is to collaborate with all parties involved in that area. This majorly includes shelters, volunteers, pet fostering services as well as common people who can keep pets for some time. Work needs to be done in increasing the reach of our app. Moreover, we need to work with adoption and on certifying vaccinations as well as credentials of vets and other service providers. We also aim to improve the community building aspect of the app and include various modes of interaction. We aim to include blog posts and forums as well as shared spaces where people can connect. The completion of these aims would help make petswala an enjoyable platform which would connect all members of the animal/pet related community in Pakistan.

# **13.** **Review checklist**

| **Chapter/Section Name** | **Reviewer Name(s)** |
| --- | --- |
| Introduction | Adil, Roshan |
| System Requirements | Mohid, Roshan |
| System Architecture | Adil |
| Requirement Specification | Roshan, Mohid |
| Software Dev methodology | Roshan |
| DB design | Mohid |
| System User Interface | Roshan, Adil |
| Security | Mohid |
| Risk Management | Adil |
| Testing and Evaluation | Mohid |
| Deployment Guidelines | Adil |

# 