كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

Helwan University

PRESENTED BY:

- MOHAMED TAREK
- MOHAMED SAMIR
- MOHAMED ADEL
- TOKA SALAH
- MOHAMED AHMED
- MOHAMED RAMADAN

Supervised by

DR. Hanan Fahmy

Index

Chapter 1: Introduction

Chapter 2: Literature Review

Chapter 3: Analysis

Chapter 4: Design

Chapter 5: Implementation

Chapter 6: Software Architecture and UI Design

Chapter 1:

In this chapter we are going to discuss and go deeper in the overview of the project and know more about its scope and limitations and explain some terminologies we will find throughout the document.

Chapter Headlines:

1.1 **Overview**

1.2 **Project Motivation**

1.3 **problem statement**

1.4 **Project Aim and Objectives**

1.5 **Project Software and Hardware Requirements**

1.6 Project **scope**

1.7 Project limitations

1.8 The expected output of the project

Chapter 1:

1.1 Overview

- PetCaretopia is a web application that enables pet owners to take care of their pets by finding nearby service providers such as veterinarians, groomers, and pet shelters.
- Pet owners can create their own accounts on the platform, browse available services, adopt pets, book appointments, and share posts and feedback about their experiences.
- Service providers and shelters can also create accounts to manage their services, display available pets for adoption, view bookings, and interact with pet owners through comments and messages.

PetCaretopia is a platform that connects pet lovers with trusted services in a smart and convenient way using the latest technology. It reduces the time and effort required to find reliable pet services or adopt a pet. PetCaretopia offers users multiple benefits such as:

- Discovering nearby service providers with ease

- Browsing available pets for adoption based on filters
- Booking appointments online without hassle
- A free platform that benefits both pet owners and providers by making pet care more accessible and organized

PetCaretopia creates a win-win situation for pet lovers, shelters, and service providers by simplifying the entire process and building a connected pet care community.

1.2 Project Motivation

It’s an irrefutable fact that many pet owners face real difficulties when it comes to finding trusted and nearby services for their pets — whether it's a vet, a groomer, a shelter, or even a place for adoption. Most of the time, they don’t know where to go or how to evaluate the quality of the service, especially in emergencies or when relocating to a new area.

Nowadays, the world has become smaller thanks to the Internet, and searching online for what we need has become much easier and faster.

However, when it comes to pet care, there is still a gap. You may have a sick pet and no idea where the nearest veterinary clinic is, or you might want to adopt but can’t find reliable shelter information around you. This is exactly where **PetCaretopia** comes in.

This project aims to make it easier for pet owners to find and connect with nearby, trusted service providers, all in one platform. What makes PetCaretopia unique is that it not only helps users discover services near them, but also gives them options to choose from, read reviews, share feedback, and engage with the community — all in a modern, smart, and accessible way.

1.3 problem statement:

One of the main problems that may arise when using an online platform like **Tair** is the user's dependency on having an active internet connection. In urgent situations — such as sudden car breakdowns in remote areas — the user might not have access to the internet, which prevents them from opening the website and contacting the nearest mechanic.

This limitation highlights a critical gap in accessibility, especially during emergencies where fast communication is essential. The lack of offline support or alternative communication methods can reduce the effectiveness of the platform in real-life scenarios where it's most needed.

1.4 Project Aim and Objectives

The main goal of **Tair** is to simplify and improve the communication between car owners and mechanics, wherever the car owners may be. The platform aims to ensure that users can easily search for, contact, and book mechanics based on proximity, reputation, and availability — all in one centralized system.

Objectives:

- Enable car owners to find the **nearest mechanic workshops** with ease.
- Allow users to **book appointments online** and **communicate directly** with mechanics.
- Collect **user feedback** and ratings for each mechanic to help others make informed decisions.
- Help mechanics build a **database of clients**, including their car models and previous issues, to streamline their workflow.

- Create a **user-friendly platform** that connects both sides efficiently and reduces the time and stress of car repair processes.

Tair brings mechanics and users together on a single platform, making it the most practical and reliable search and booking site for automotive repair services.

1.5 Project Software and Hardware Requirements

1.5.1 Software Requirements

In this project, we are using the following software technologies:

BACK-END:

- **Java as the main programming language**
- **Spring Boot framework for building RESTful APIs**
- **WebSockets for real-time communication between users (e.g., messaging, notifications)**
- **MySQL as the relational database management system**

FRONT-END:

- **React.js for building a dynamic, component-based user interface**
- **HTML5 & CSS3 for structuring and styling the web pages**

Other Tools & Libraries:

- **JWT for authentication and authorization**
- **Lombok and MapStruct for cleaner code and DTO mapping**
- **Axios for API calls from the frontend**
- **Socket.IO or native WebSocket support on both frontend and backend for real-time features**

1.4.2 Hardware Requirements

Users can use any labtop(winsows-linux) or mobile phone (Android - IOS)

1.5 Project, Product, and Schedule Risks

Risk	Discerption	Mitigation
Project Risk	Uncertain event or activity that can impact the project's progress, ex: Lack of communication, causing lack of clarity and confusion. Adding of features and functionality (project scope) without addressing the effects on time, costs, and resources and that may effect on the progress of main features.	Schedule regular team meetings to maintain clear communication. Apply change management practices to prevent uncontrolled scope expansion. Document all requirements and agreements to ensure alignment among team members.
Product Risk	The possibility that the system or software might fail to satisfy or fulfill the expectation of the customer, user, or stakeholders, ex: bad recommendation system or some crashes due to mistakes in connection between backend and frontend.	Perform regular testing, including unit and integration tests, throughout development. Share demos or prototypes with users early to gather feedback and adjust features accordingly. Conduct frequent code reviews to detect and fix issues early.
Schedule Risk	Failing to meet schedule plans and the effect of that failure, ex: Project schedule is not clearly defined or understood	Create a clear and detailed project timeline covering all development phases. Allocate buffer time for each phase to absorb unexpeced delays.

1.6 Project scope

1.6.1 Main Scope

- **Communication**:
  The platform enables smooth communication between pet owners and service providers (vets, groomers, shelters), helping users get fast support and build trust through direct messaging and notifications.
- **Booking Services:**

  Users can explore service providers near them, view their profiles and service details, then book appointments directly through the system.
- **Pet Adoption**:

  Animal shelters can list available pets for adoption with full details, and users can apply to adopt and track the status of their requests.
- **Rating and Feedback:**

  After receiving a service or adopting a pet, users can rate their experience and leave feedback, which enhances trust and improves the recommendation system.

- **Community Engagement:**

  Users can post, comment, react, and share pet-related content, creating a social space that builds connection and engagement.

1.7 Project limitations:

The platform currently supports pet-related services and adoption **within a specific region or country**. Cross-border services or international shelter listings are not supported in the current phase.

Additionally, the platform requires a stable internet connection to access services, meaning offline functionality is limited.

1.8 The expected output of the project:

A pet owner using **PetCaretopia** will be able to:

- Discover nearby and trusted pet care services based on their location
- Book appointments easily

- Apply for pet adoption
- Communicate directly with providers
- Interact with a growing pet-loving community

All of this through a centralized platform designed for convenience, trust, and engagement

Chapter 2

2.1 Introduction

In this chapter, we present similar applications that provide services related to pet care, adoption, or connecting pet owners with service providers. We analyze the common issues these platforms face and how **PetCaretopia** addresses and resolves them using modern development and design approaches.

2.2 Existing Systems to Connect Pet Owners with Pet Services

Some existing platforms that offer similar functionalities to our system include:

- **Petfinder**
- **Rover**
- **Wag!**
- **Adopt-a-Pet**
- **Vetster**

2.3 Overall Problems of Existing System

Despite the usefulness of these platforms, they face several common problems that affect user satisfaction and engagement:

- **Unfriendly User interface**
  Many apps suffer from outdated or overly complex designs, making it hard for first-time users or non-technical users to navigate the system smoothly.
- **Limited Integration of Services**

  Most existing platforms focus on only one function (e.g., either adoption or vet booking), forcing users to use multiple apps for complete care.
- **Lack of Personalization**

  Recommendation systems in some apps are either basic or absent, failing to provide customized suggestions based on user history or pet profile.
- **High Service Fees**

  Platforms often charge high commission rates or subscription fees for both service providers and users, limiting accessibility.

2.4 Overall Solution Approach

In **PetCaretopia**, we designed the platform while keeping in mind all the weaknesses of existing systems. We implemented the following enhancements:

- **Modern and Clean UI/UX**

  The platform offers a user-friendly interface designed with simplicity and clarity to allow users of all ages and backgrounds to navigate easily.

- **All-in-One Ecosystem**

  Our application integrates all essential pet-related services in one place: booking, adoption, communication, community interaction, and more.

- **Smart Recommendation System**

  We implemented a data-driven recommender system that uses user behavior and pet preferences to provide personalized service and pet suggestions.

- **Low-Cost Model**

  By offering most services for free and using a minimal, fair commission structure, we ensure both users and providers benefit without high costs.

- **Continuous Updates**

The platform is designed to evolve. Regular updates will be released based on user feedback and emerging needs in the pet care industry.

Chapter 3: Analysis

3.1 Functional Requirements – PetCaretopia

User Registration & Login

- Users (pet owners, service providers, shelters, admins) can register and log in with role-based access.

Profile Management

- Users can view and edit their personal information and pet profiles.
- Service providers can manage their service details and working hours.
- Shelters can manage pet listings.

Location-Based Search

- Users can search for nearby service providers (vets, groomers, shelters) using their current location or map.

- Filtering by service type, ratings, distance, and availability.

Pet Adoption System

- Shelters can add pets for adoption with details and photos.
- Users can view pets, apply for adoption, and track application status.
- Admins/shelters can approve or reject requests.

Booking Services

- Pet owners can book appointments with service providers (e.g., vet, groomer).
- Booking includes date/time selection, service type, and optional notes.
- Users receive booking confirmation and reminders.

Live Chat (WebSocket)

- Real-time chat between pet owners and service providers for inquiries and follow-up.
- Notifications for new messages.

Rating & Reviews

- After completing a service or adoption, users can rate and write reviews.
- Average ratings are shown in provider profiles.

Post Creation & Social Interaction

- Users can create posts (with images) about their pets or experiences.
- Other users can react, comment, and share posts.

Reporting System

- Users can report inappropriate posts or behavior.
- Admin reviews and takes action.

Admin Dashboard

- Admin can manage users, pets, posts, bookings, and view system statistics.
- Admins can ban/unban users, and view reports submitted.

Feedback System

- General feedback option for users to send suggestions or complaints to platform admins.

Pet Profile Management

- Users can register multiple pets, each with breed, age, medical info, and photos.
- Used for personalized recommendations and service tracking.

Service Package Display

- Service providers can list packages (e.g., grooming plans, health checkups) with pricing and descriptions.

Search History & Personalization

- System keeps search history and behavior to suggest services and content.

Session Management

- Users are logged out after inactivity for security.
- JWT tokens used for secure session handling.

Product Listing

- Admins or service providers can add products (e.g., pet food, toys, accessories).
- Each product includes name, description, price, category, stock, and images.

Product Browsing & Filtering

- Users can browse the store by category, price range, brand, or pet type.

- Product search functionality included.

Product Details Page

- Each product has a dedicated page showing full details, images, availability, and reviews.

Cart Management

- Users can add/remove items from their cart.
- Quantity adjustment available before checkout.
- Total price and estimated delivery calculated in real-time.

Wishlist Feature

- Users can add products to their wishlist to purchase later.

Checkout System

- Users can place orders by providing shipping details and selecting payment method (e.g., cash on delivery, online).
- Confirmation and order summary displayed after placing an order.

Order Tracking

- Users can view order history and current status (pending, confirmed, shipped, delivered).

Stock Management

- Admins can update product stock and availability.
- Automatic alerts when stock is low.

Product Reviews & Ratings

- Users can leave reviews for products they purchased.
- Ratings help future buyers make decisions.

Discounts & Promotions

- Admin can create discount codes or time-limited offers.
- Discounts applied automatically during checkout.

3.2 Non-functional Requirements

- **Availability**:
  The platform will be available 24/7. If the internet is disconnected, the last accessed screen will remain available in cache.
- **Reliability**:
  Built using stable and up-to-date technologies (Spring Boot, React, WebSocket, JWT) to ensure minimal downtime and consistent performance.

- **Efficiency**:
  The application uses clean code patterns, asynchronous processing, and optimized database queries to ensure a fast and smooth user experience.
- **Integrity**:
  Role-based access control (via Spring Security with JWT) is implemented to ensure that users access only the features relevant to their roles (e.g., admin, pet owner, service provider).
- **Portability**:
  The app is designed as a responsive web application and can later be wrapped into a mobile version for Android and iOS using frameworks like React Native or Flutter.
- **Usability**:
  The UI includes accessible design elements such as readable font sizes, intuitive navigation, and consistent layout to support users of all backgrounds.
- **Scalability**:
  The backend uses a cloud-ready architecture, with MySQL database and support for scalable hosting, to handle a growing number of users and services.
- **Flexibility**:
  The system follows Object-Oriented Programming and modular design principles, allowing each module

(Booking, Adoption, Social, etc.) to be updated or extended independently.

3.3 Stakeholders

3.3.1 Primary Stakeholders

Project Managers

Responsible for overseeing the project timeline, resource allocation, and ensuring that the platform is developed according to the planned scope.

- **Project Team (Developers, Designers, Testers)** The technical team involved in designing, developing, testing, and deploying the platform's features.
- **Pet Owners (Customers)**

  The main users of the platform who will book services, adopt pets, and interact socially with others in the pet community.
- **Service Providers**

  Includes veterinarians, groomers, pet sitters, and others who offer services through the platform.
- **Shelters**
  Institutions or individuals that list pets for adoption and manage adoption requests.

Admin Users

Responsible for monitoring the platform, managing content, reviewing reports, and ensuring system integrity.

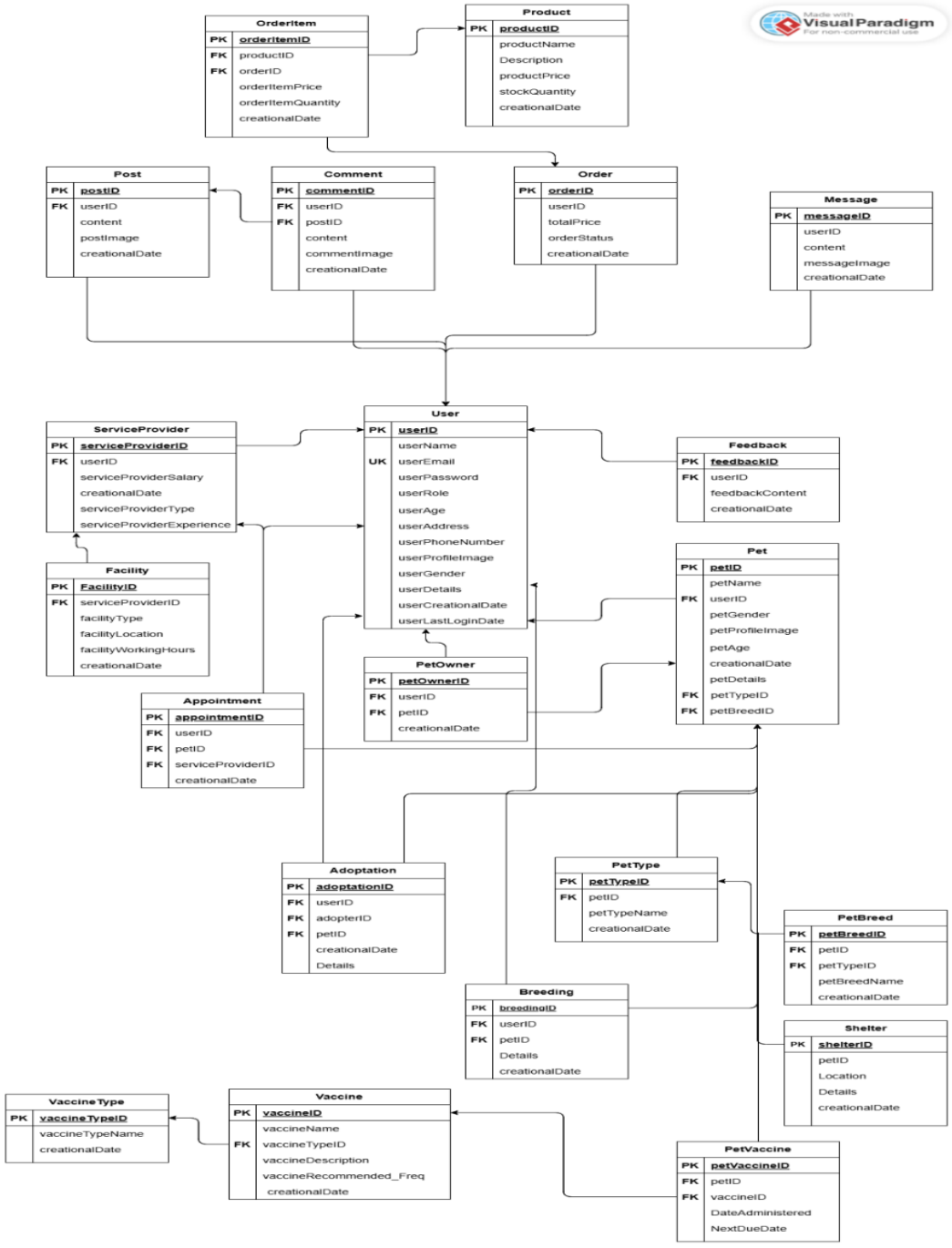
3.4 Constraints

- The application shall run primarily on mobile devices (Android or iOS), and must be responsive for different screen sizes.
- A stable internet connection is required to access most of the platform features such as real-time chat, booking, and browsing updated content.
- Only **registered users** can perform core actions like booking services, adopting pets, chatting with providers, and posting content.
- Service providers and shelters must verify their identity before being approved to offer services or list pets.
- The system must ensure role-based access to restrict users from accessing unauthorized functionalities.
- The application shall comply with data privacy standards to protect user and pet information.

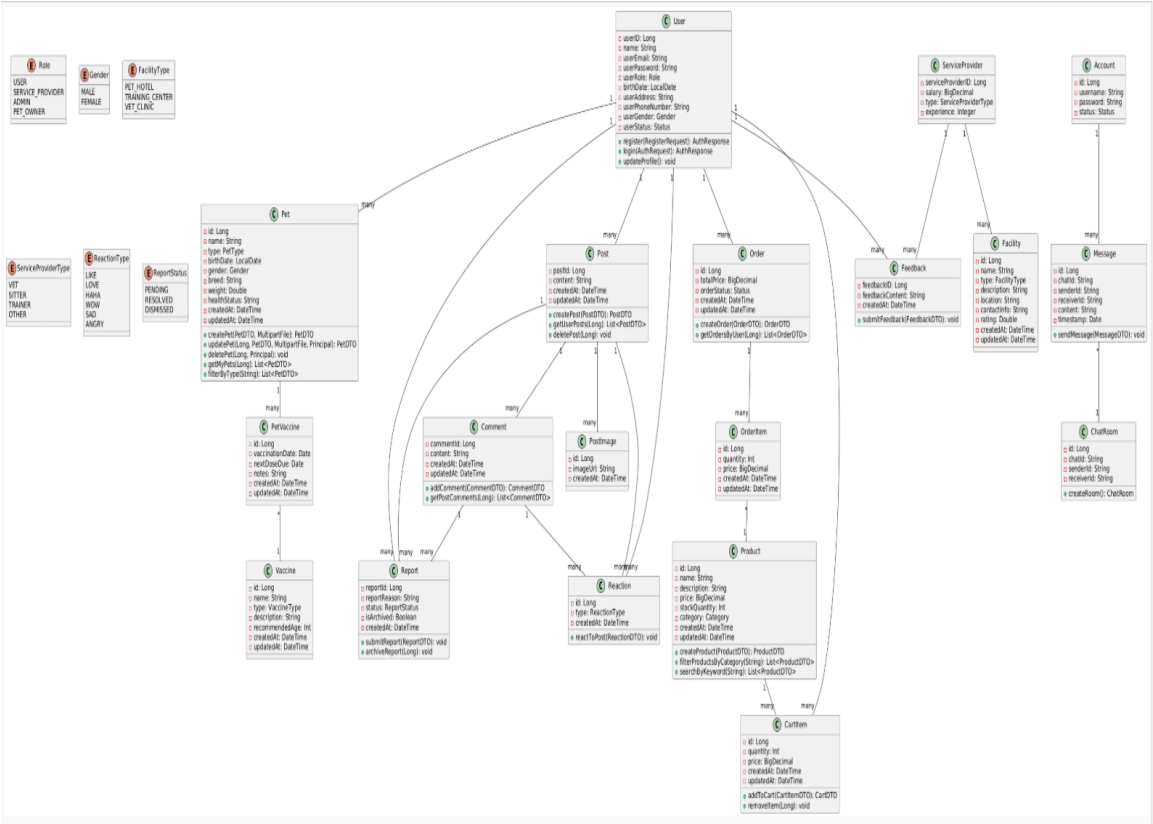
. Chapter 4

4.1 Database Design

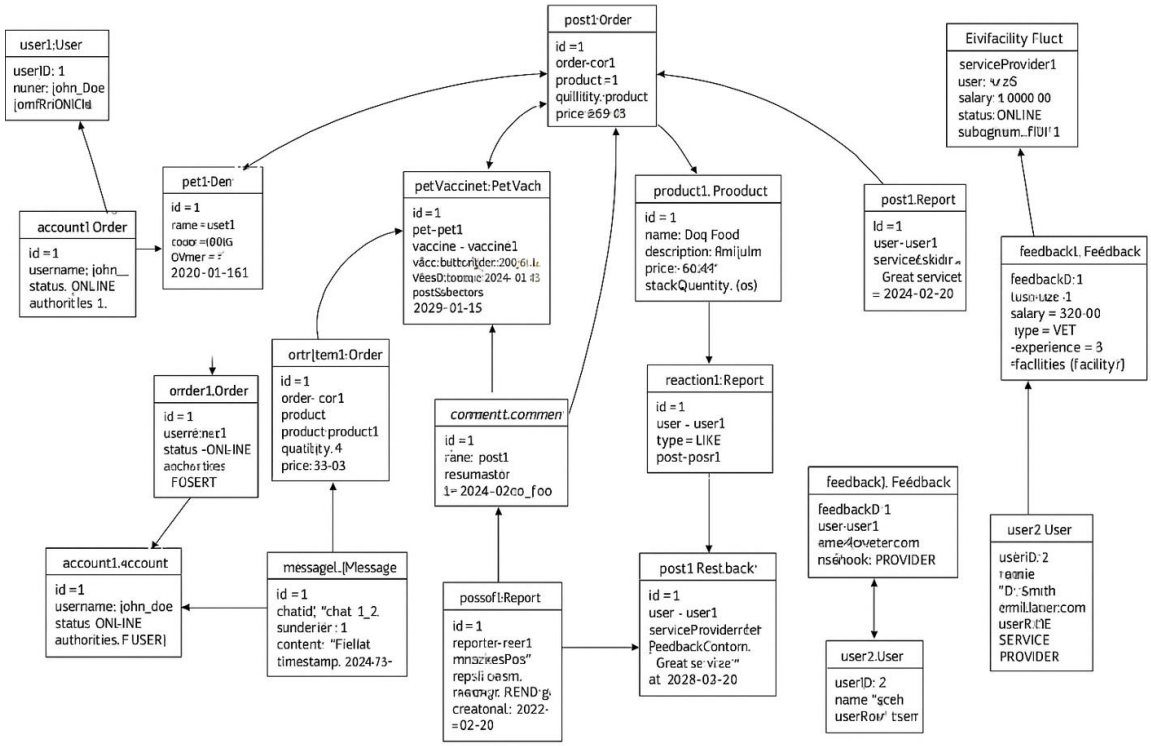
ERD diagram 4.2



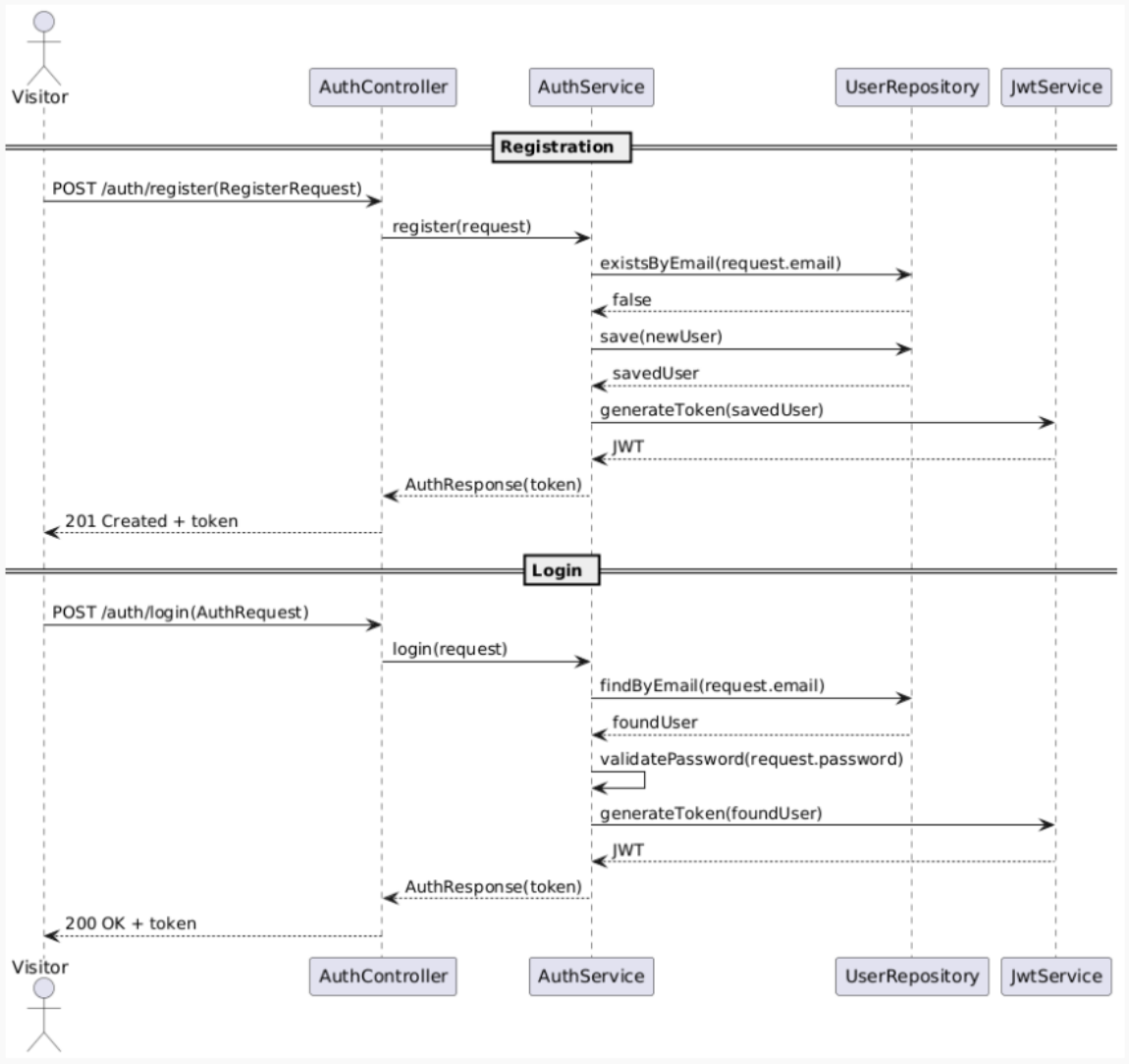
Class Diagrams 4.3

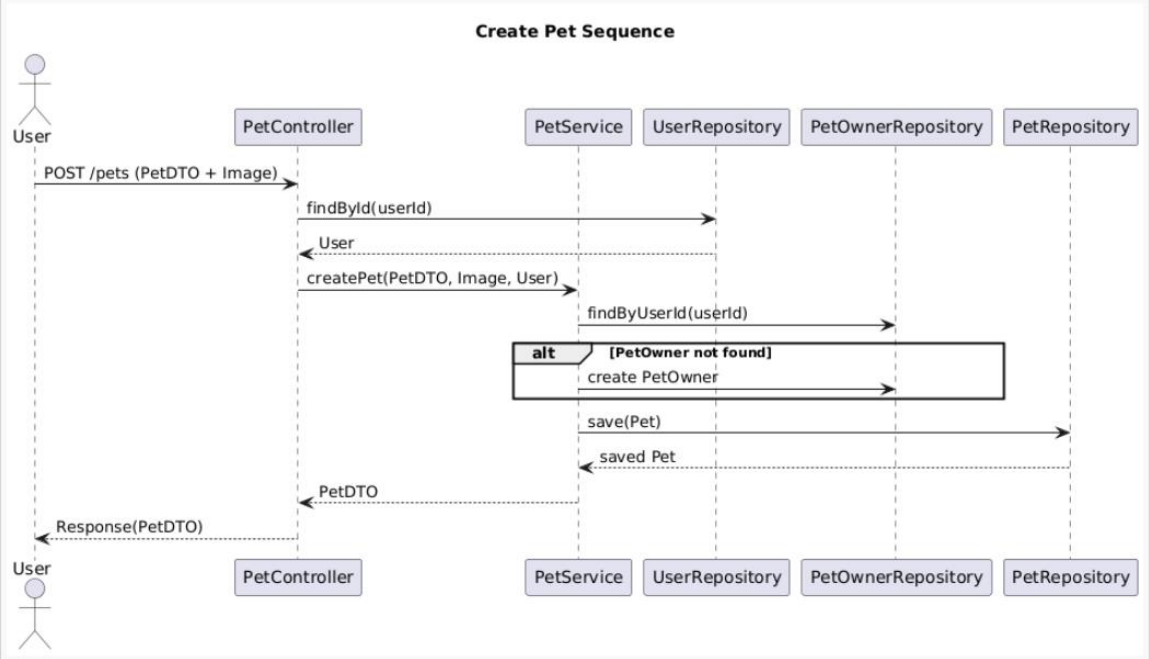


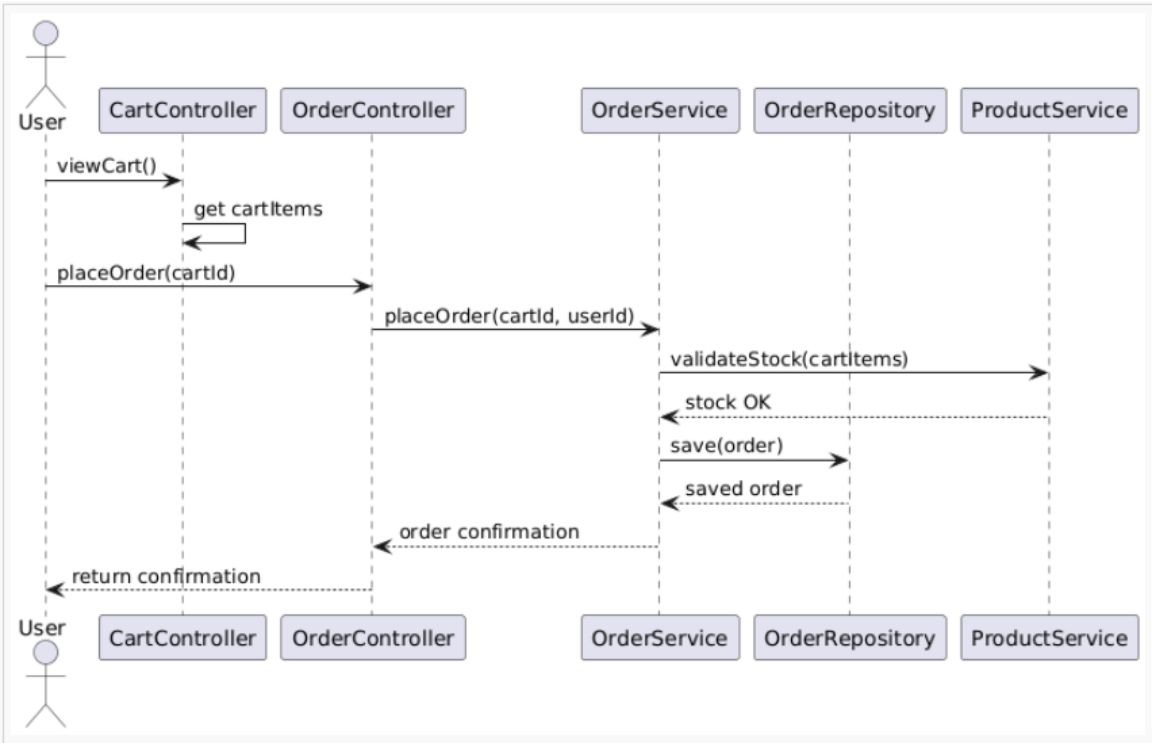
Object Diagrams 4.4

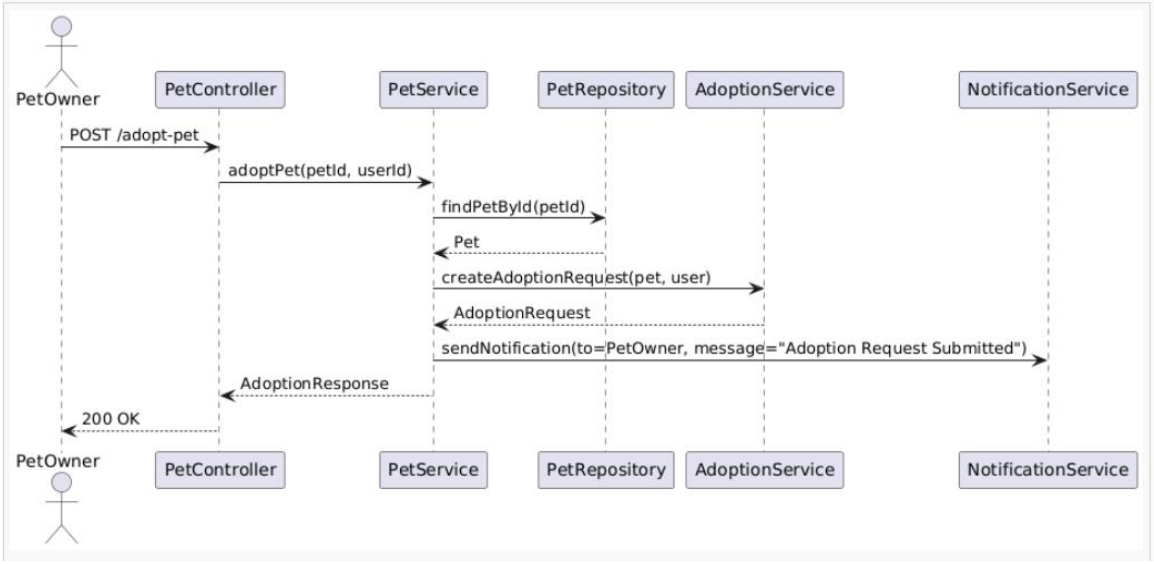


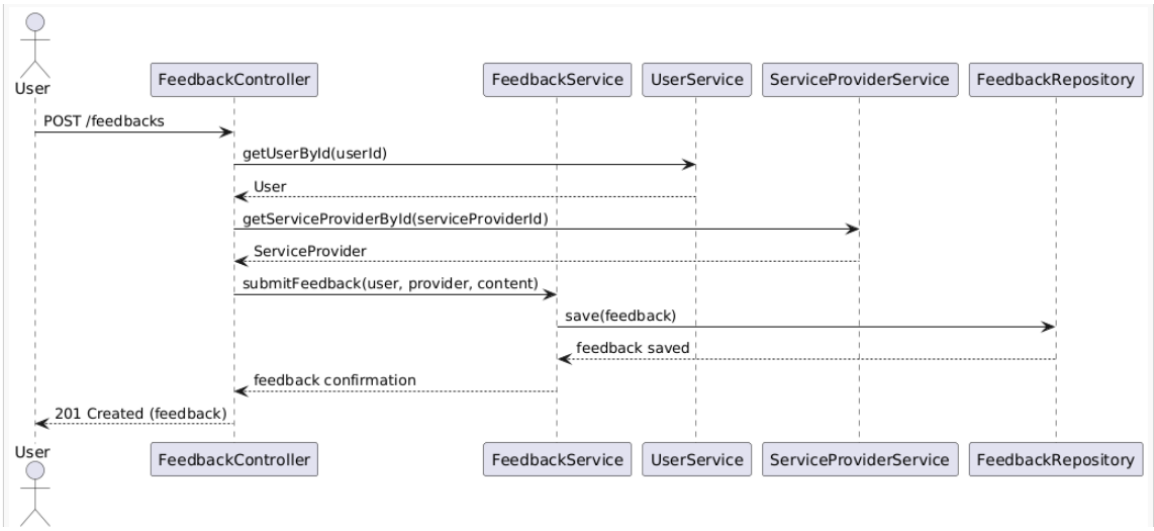
Sequence Diagram 4.4

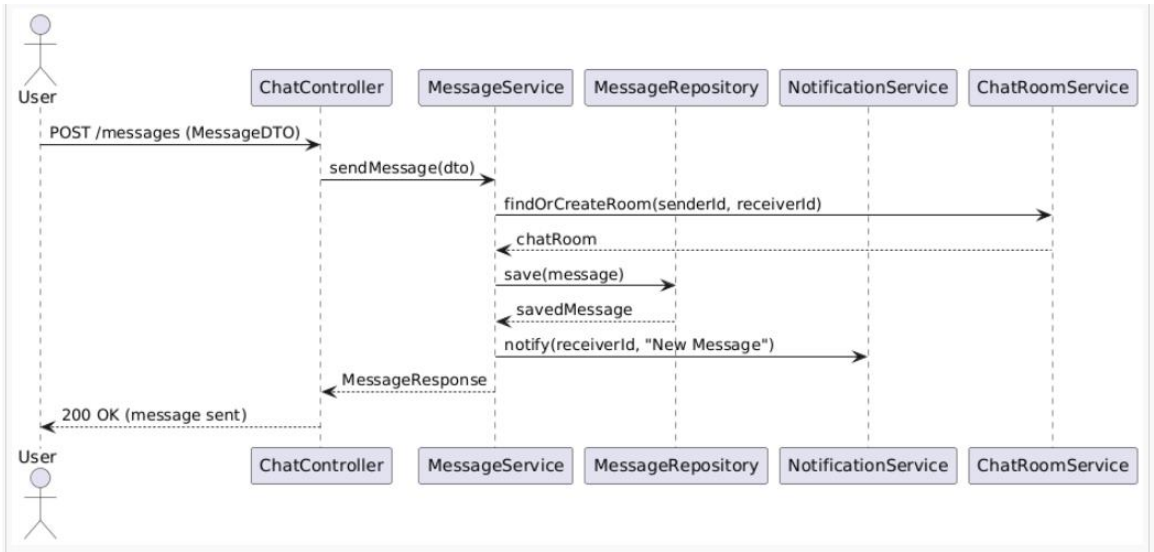


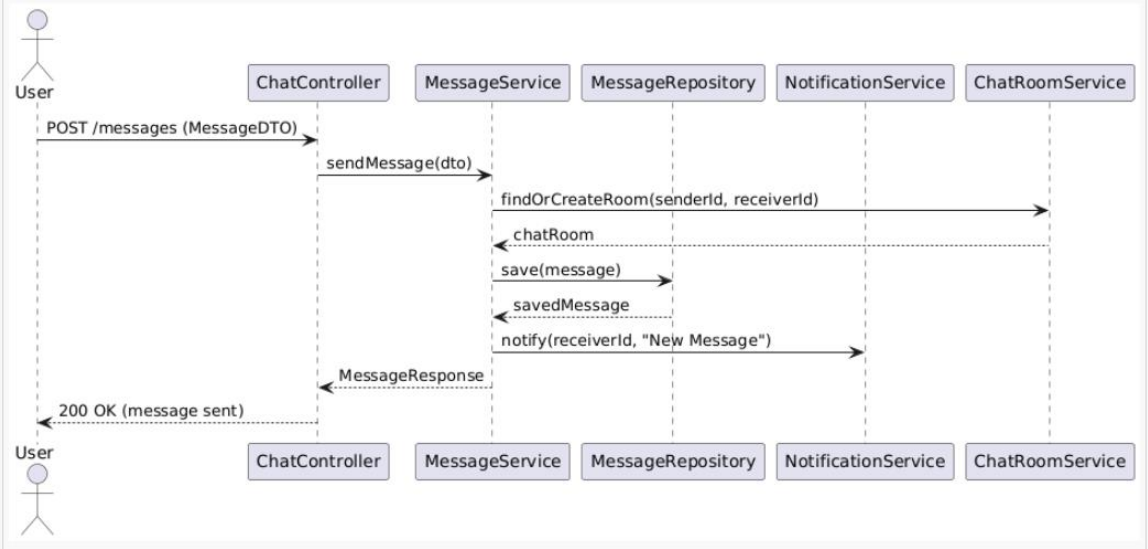

Create Pet Sequence
User
PetController
PetService
UserRepository
PetOwnerRepository
PetRepository
POST /pets (PetDTO + Image)
findById(userId)
User
createPet(PetDTO, Image, User)
findByUserId(userId)
alt
[PetOwner not found]
create PetOwner
save(Pet)
saved Pet
PetDTO
Response(PetDTO)
User
PetController
PetService
UserRepository
PetOwnerRepository
PetRepository


User
CartController
OrderController
OrderService
OrderRepository
ProductService
viewCart()
get cartItems
placeOrder(cartId)
placeOrder(cartId, userId)
validateStock(cartItems)
stock OK
save(order)
saved order
order confirmation
return confirmation
User
CartController
OrderController
OrderService
OrderRepository
ProductService


PetOwner
PetController
PetService
PetRepository
AdoptionService
NotificationService
POST /adopt-pet
adoptPet(petId, userId)
findPetById(petId)
Pet
createAdoptionRequest(pet, user)
AdoptionRequest
sendNotification(to=PetOwner, message="Adoption Request Submitted")
AdoptionResponse
200 OK
PetOwner
PetController
PetService
PetRepository
AdoptionService
NotificationService


User
FeedbackController
FeedbackService
UserService
ServiceProviderService
FeedbackRepository
POST /feedbacks
getUserById(userId)
User
getServiceProviderById(serviceProviderId)
ServiceProvider
submitFeedback(user, provider, content)
save(feedback)
feedback saved
feedback confirmation
201 Created (feedback)
User
FeedbackController
FeedbackService
UserService
ServiceProviderService
FeedbackRepository

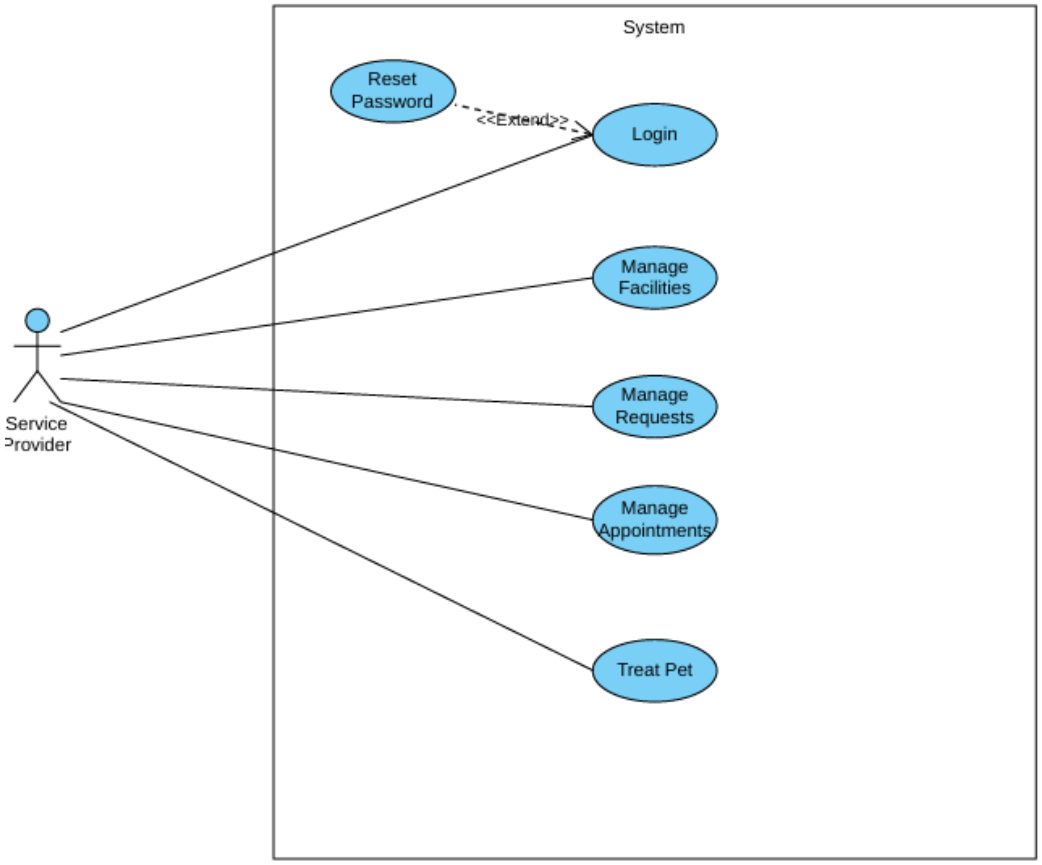

User
ChatController
MessageService
MessageRepository
NotificationService
ChatRoomService
POST /messages (MessageDTO)
sendMessage(dto)
findOrCreateRoom(senderId, receiverId)
chatRoom
save(message)
savedMessage
notify(receiverId, "New Message")
MessageResponse
200 OK (message sent)
User
ChatController
MessageService
MessageRepository
NotificationService
ChatRoomService


User
ChatController
MessageService
MessageRepository
NotificationService
ChatRoomService
POST /messages (MessageDTO)
sendMessage(dto)
findOrCreateRoom(senderId, receiverId)
chatRoom
save(message)
savedMessage
notify(receiverId, "New Message")
MessageResponse
200 OK (message sent)
User
ChatController
MessageService
MessageRepository
NotificationService
ChatRoomService

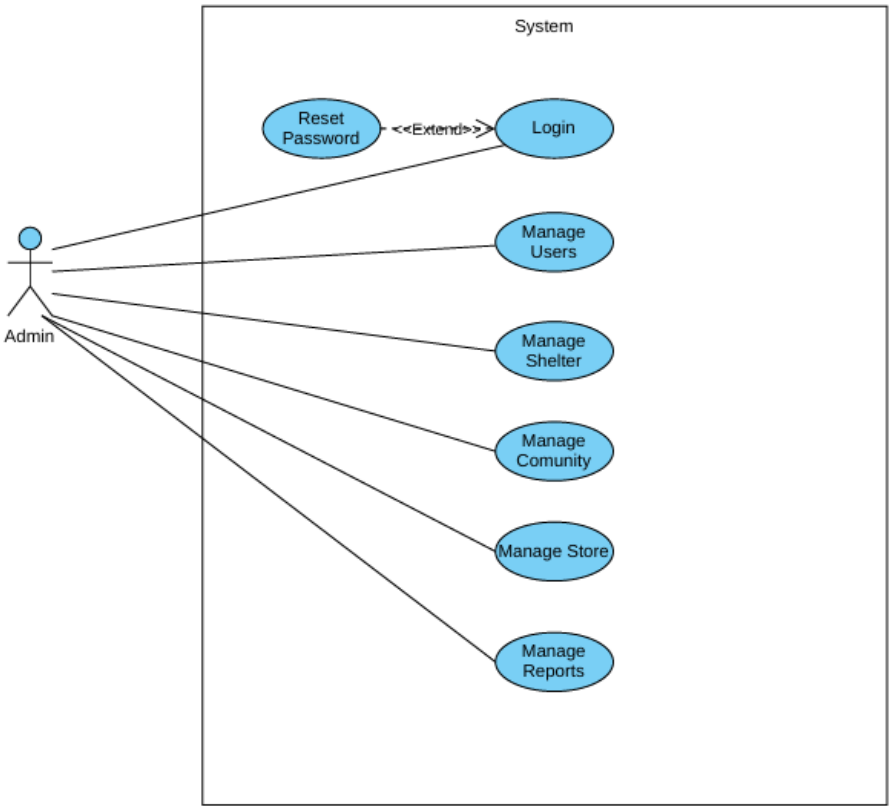
Usecase Diagram 4.5

User/PetOwner:

Service Provider:

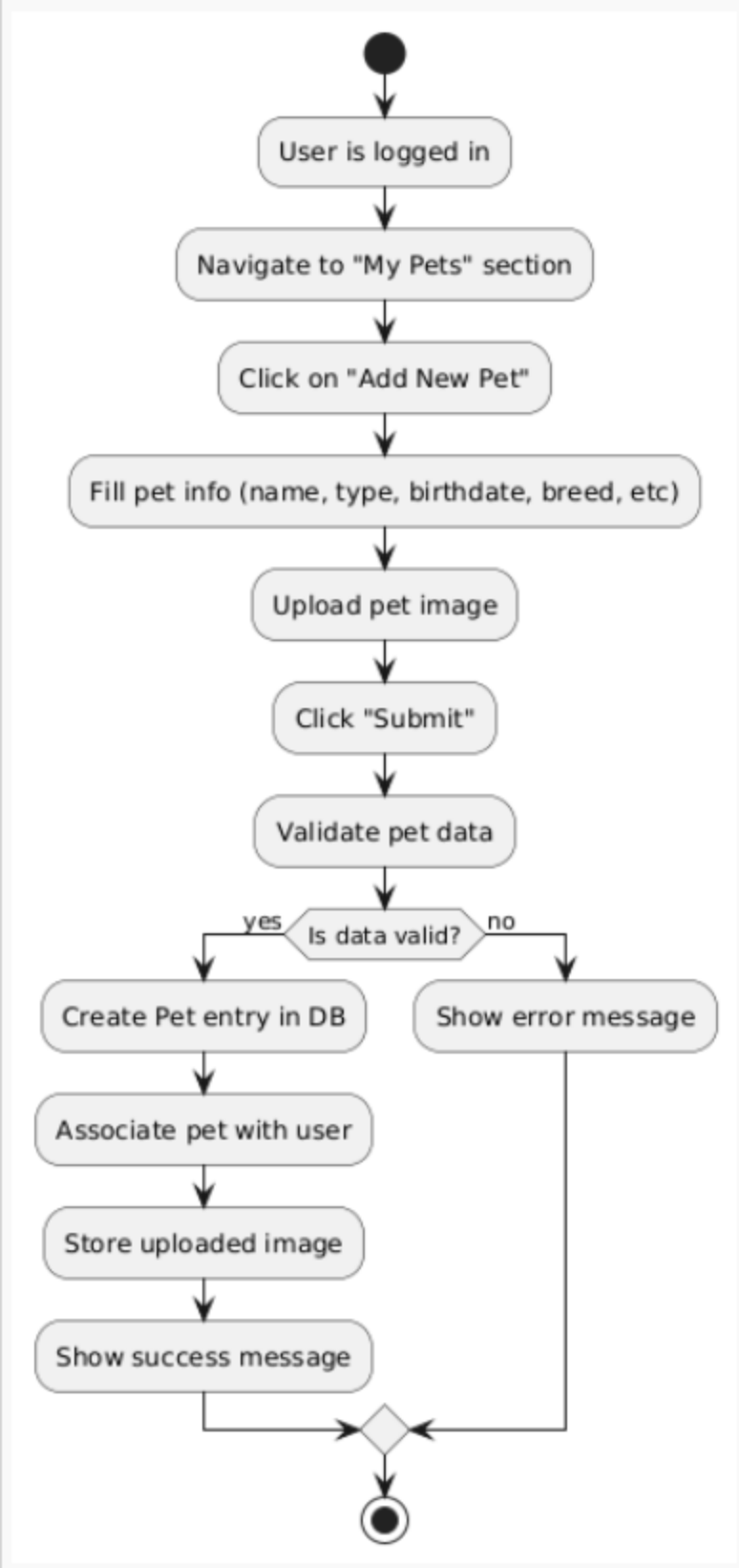


Admin:

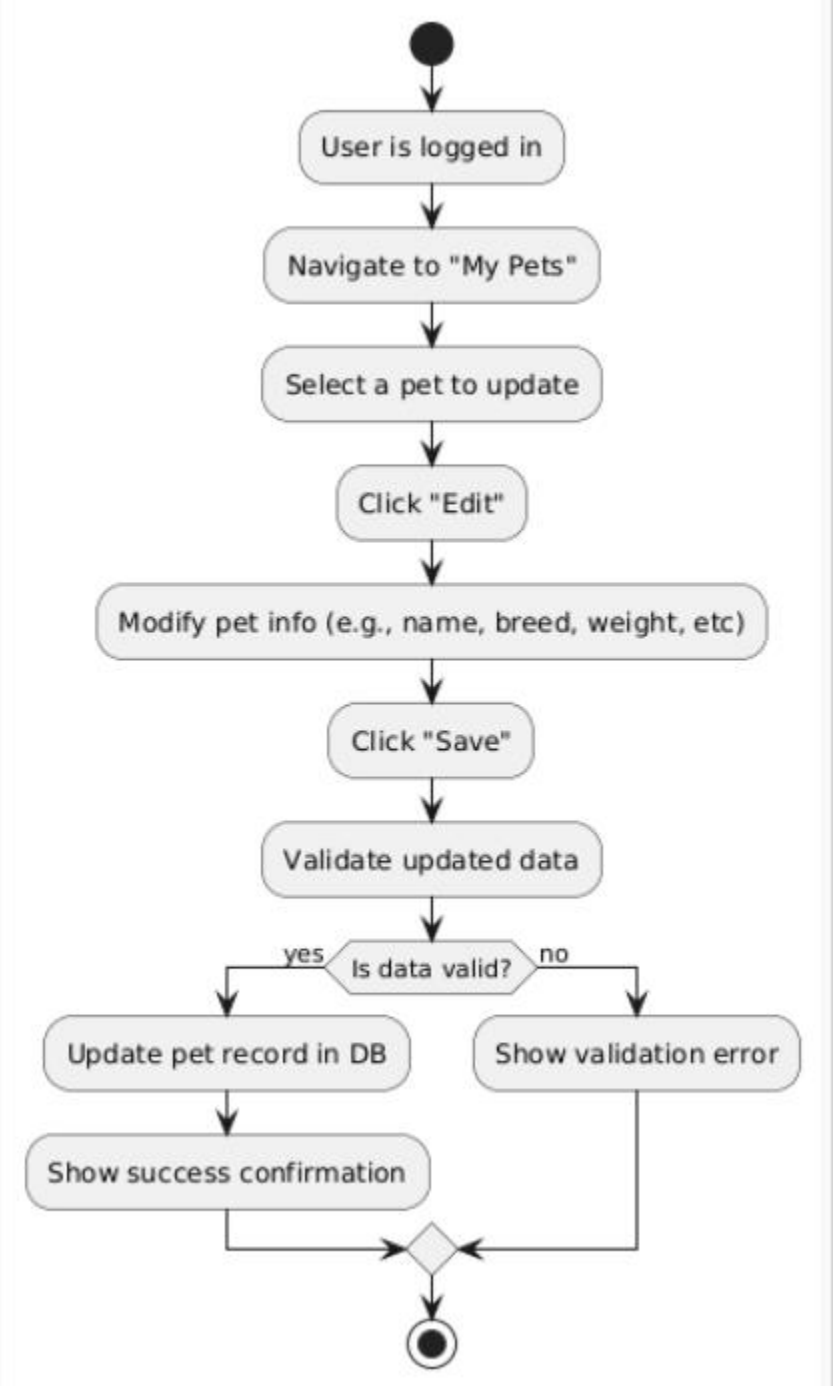


Activity diagram 4.5

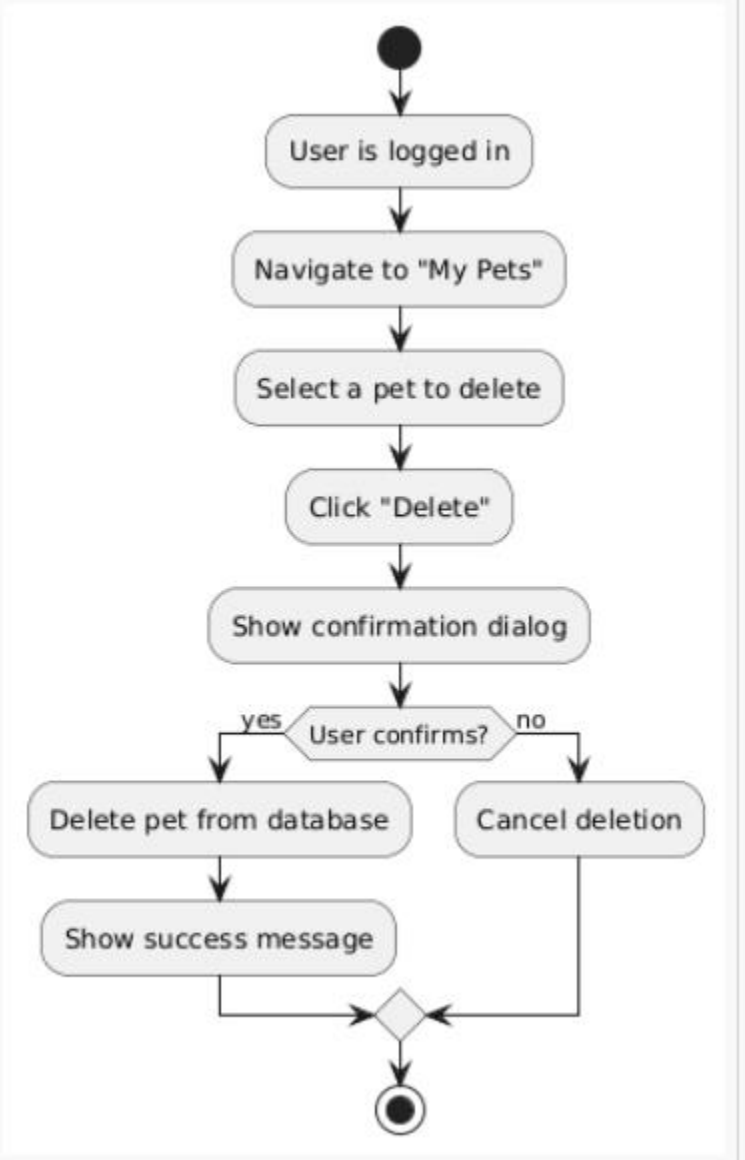
Create New Pet


User is logged in
Navigate to "My Pets" section
Click on "Add New Pet"
Fill pet info (name, type, birthdate, breed, etc)
Upload pet image
Click "Submit"
Validate pet data
Is data valid?
yes
no
Create Pet entry in DB
Show error message
Associate pet with user
Store uploaded image
Show success message

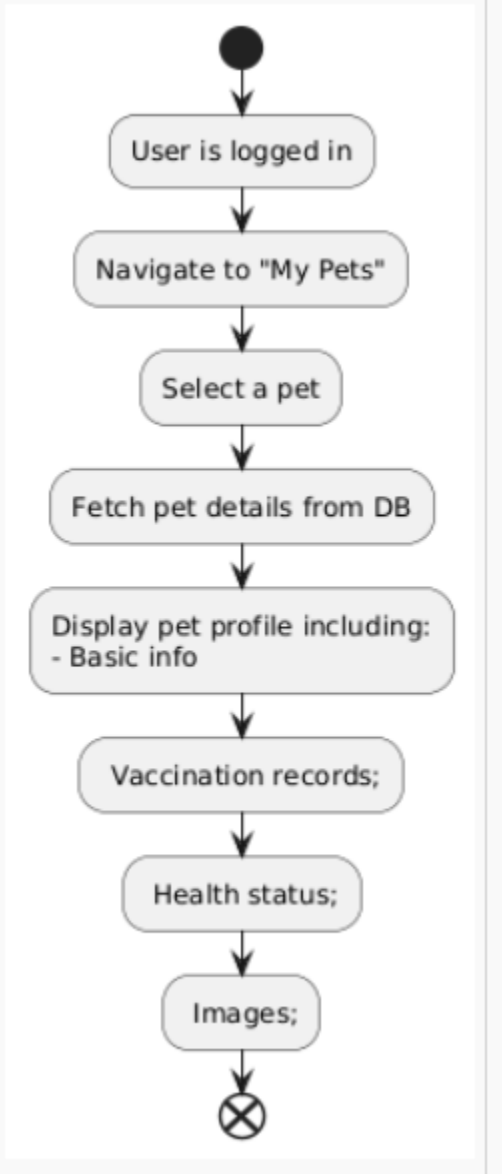
Update Existing Pet



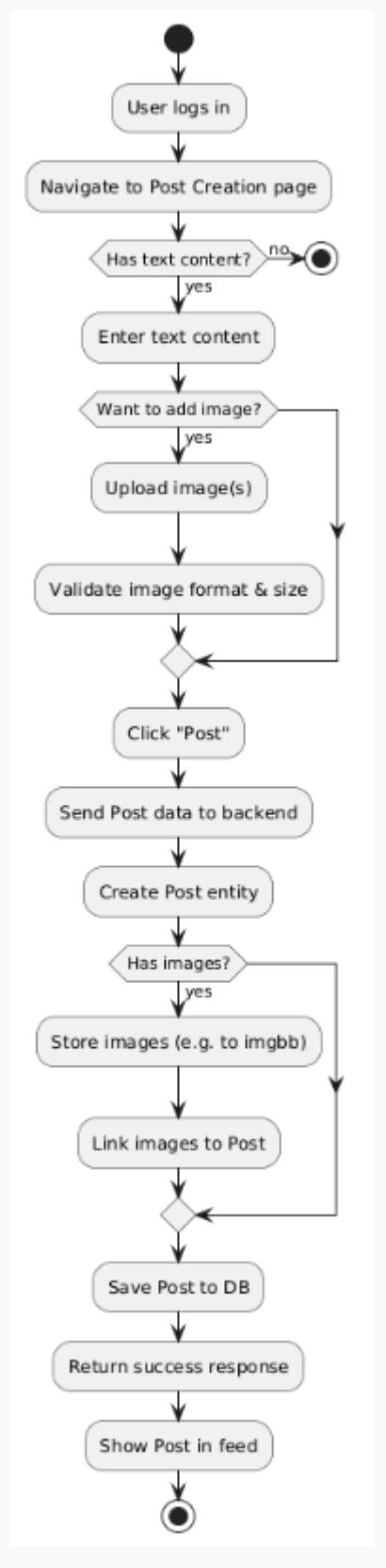
Delete Pet


User is logged in
Navigate to "My Pets"
Select a pet to delete
Click "Delete"
Show confirmation dialog
User confirms?
yes
no
Delete pet from database
Cancel deletion
Show success message

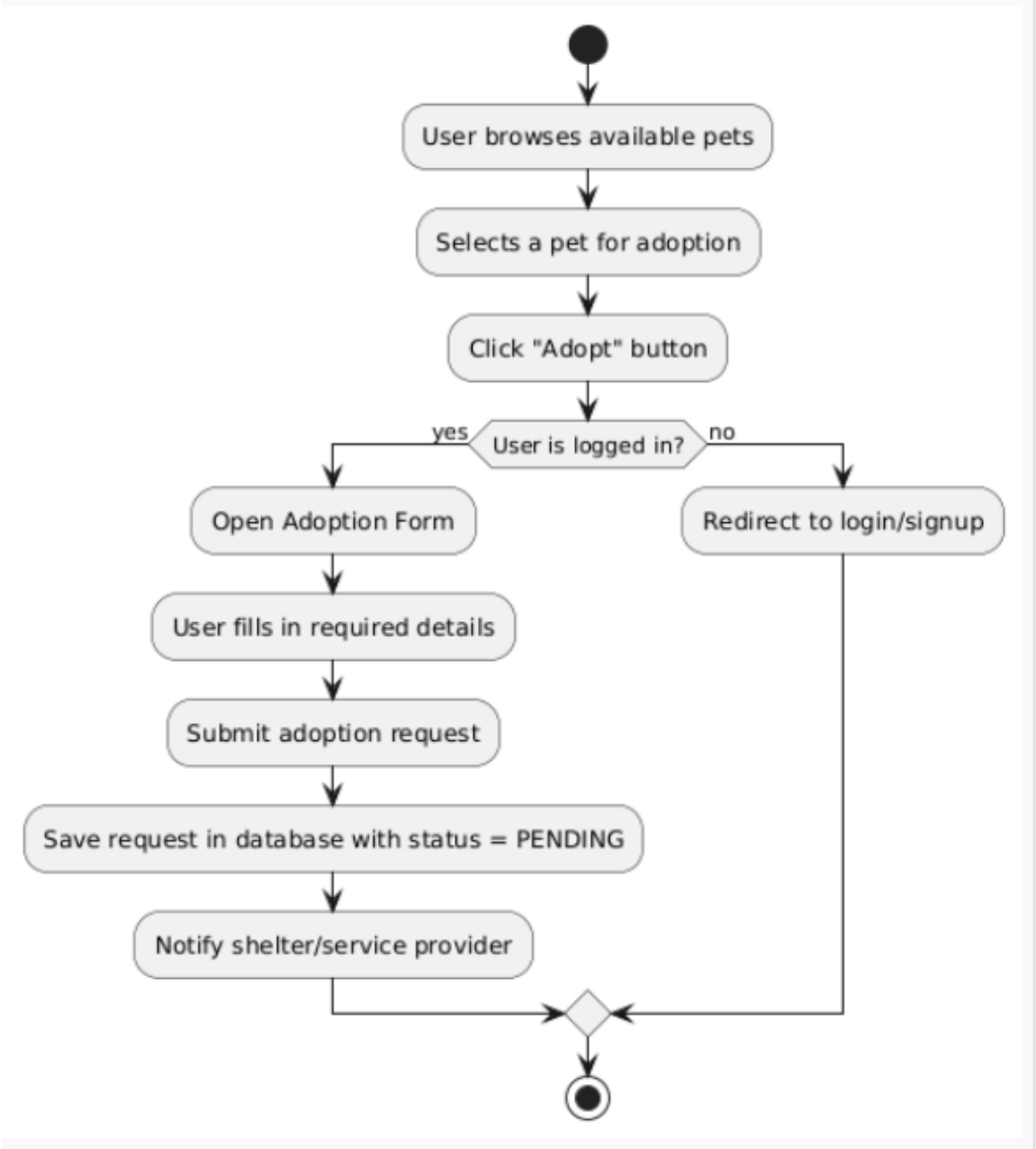
View Pet Profile



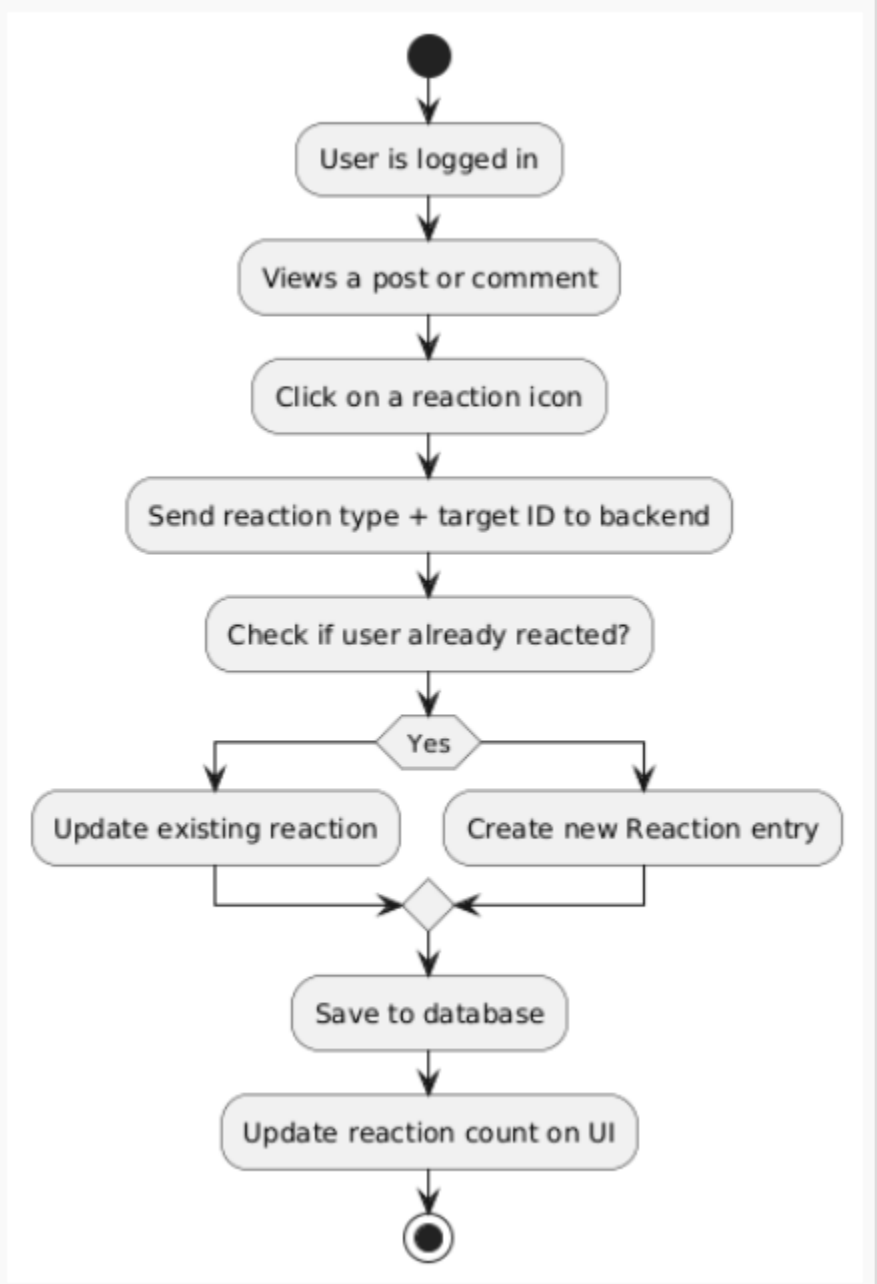
Create a Post



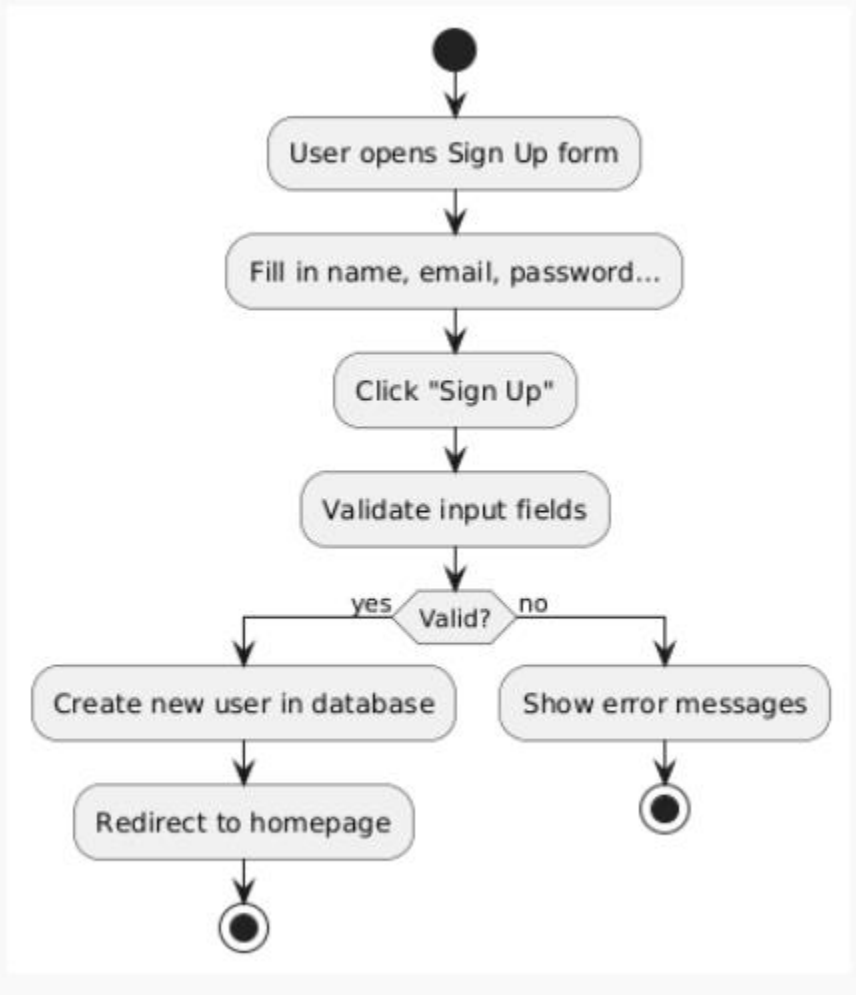
Adoption Activity



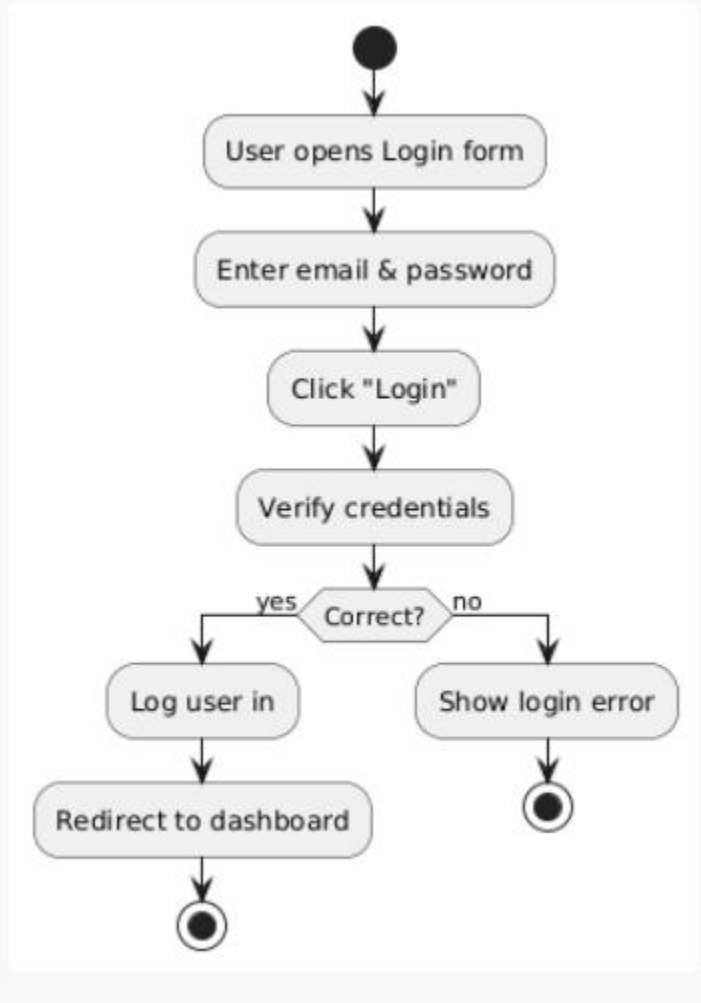
Add reaction



Sign up


User opens Sign Up form
Fill in name, email, password...
Click "Sign Up"
Validate input fields
Valid?
yes
no
Create new user in database
Show error messages
Redirect to homepage

Login


User opens Login form
Enter email & password
Click "Login"
Verify credentials
Correct?
yes
no
Log user in
Show login error
Redirect to dashboard

Chapter5: Implementation

5.1 Description of Implementation

In this subsection, we describe in detail how the main components of **PetCaretopia** were implemented from the backend and frontend perspectives.
The system follows a modular and scalable architecture, relying on Spring Boot for backend logic, React for the frontend interface, MySQL for persistent storage, and WebSockets for real-time communication.

5.1.1 Authentication and Role-Based Access Control (RBAC)

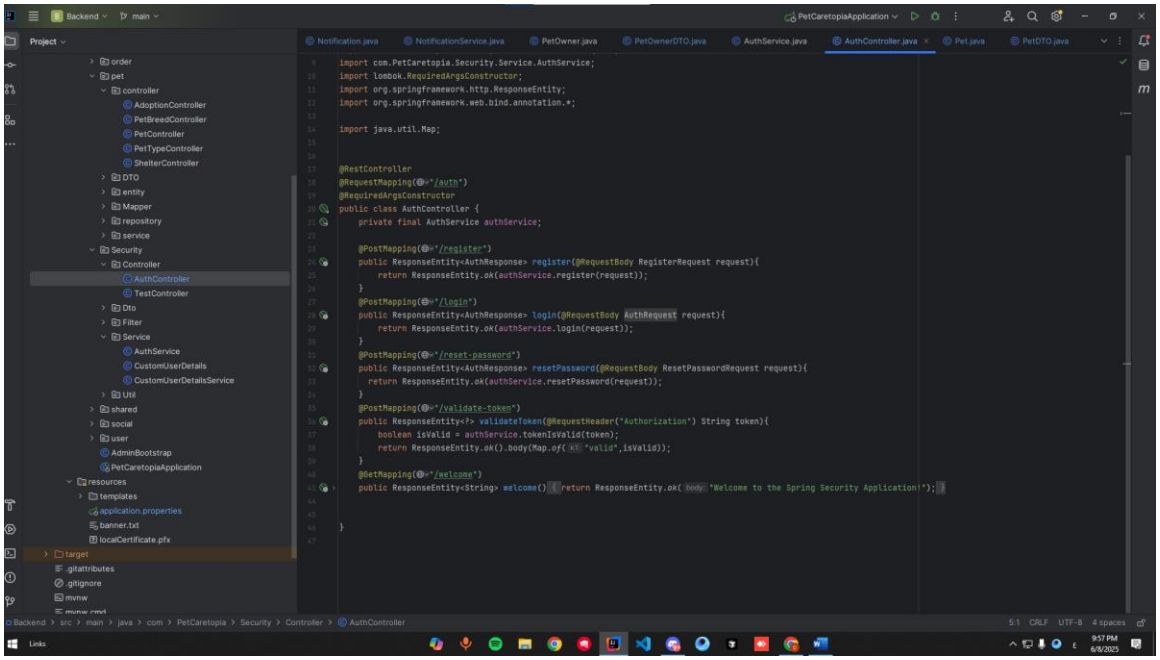
One of the core implementations in **PetCaretopia** is the **Role-Based Access Control** system that manages **what actions each user type can perform** based on their role (e.g., PET_OWNER, SERVICE_PROVIDER, ADMIN, SHELTER).

We implemented this using **Spring Security** along with **JWT tokens** for stateless authentication.
Each user, upon registration and login, receives a JWT that

encodes their identity and role. This token is used to protect API endpoints and validate access throughout the application.

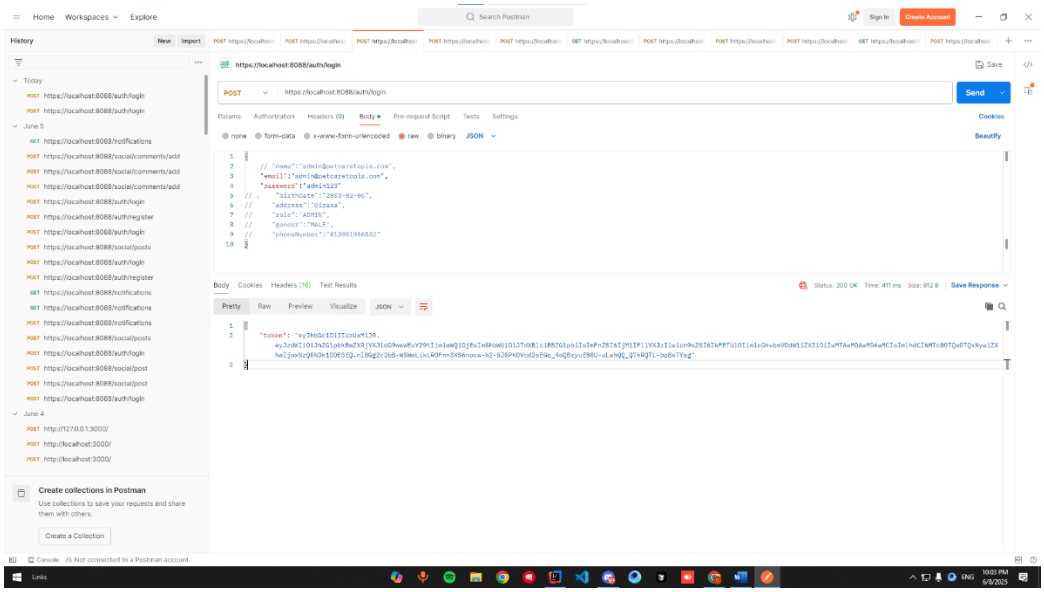
The following screenshot shows the AuthController, where we handle core authentication endpoints such as:

- /register: Registers a new user with a specified role.
- /login: Authenticates a user and returns a JWT token.
- /reset-password: Allows password updates.
- /validate-token: Validates the JWT to ensure it's still active.
- /welcome: Test endpoint to check secured access.



JWT Token Response

When a user logs in via the /login endpoint, a JWT token is returned as shown below.
This token is required in the Authorization header for accessing protected routes.



Role & Permission Structure

Each user has a defined role in the system. We store this relationship in a **many-to-many mapping** between User and Role entities using JPA.

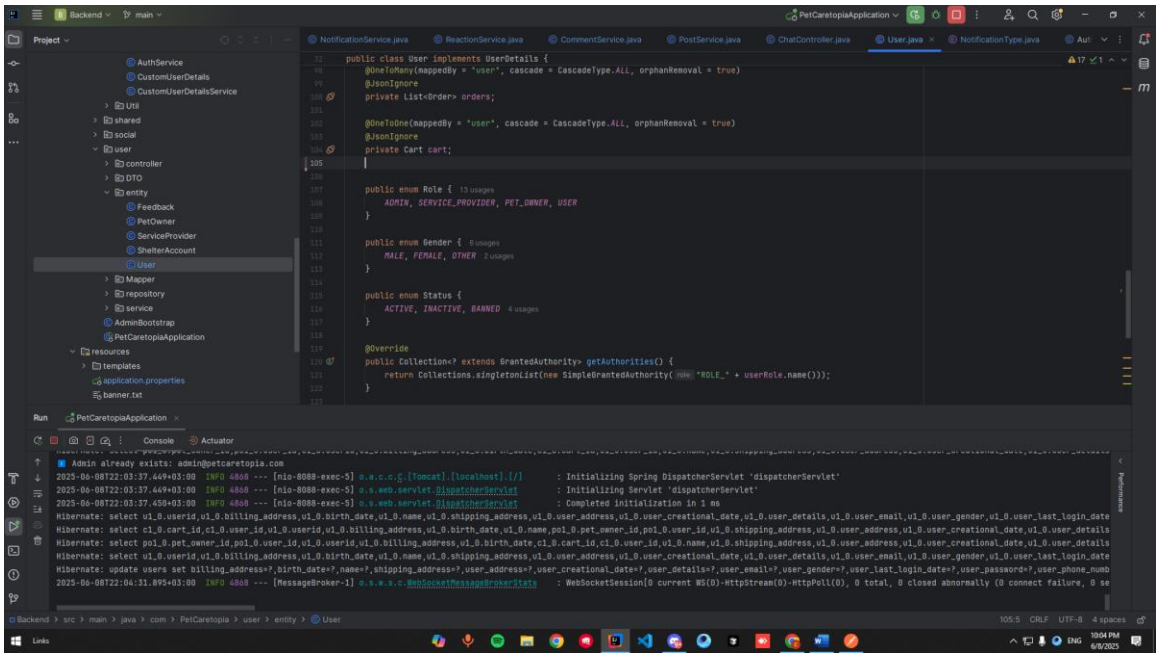
In the **User.java** entity class, we defined multiple enumerated roles including ADMIN, SERVICE_PROVIDER, PET_OWNER, and USER.

These roles are mapped into authorities using Spring Security's **GrantedAuthority** interface through the **getAuthorities**() method.

This allows the application to dynamically determine access rights for each user at runtime. The structure is modular and ready to be extended with more roles or permissions if needed.

Additional enums such as Gender and Status provide more metadata for user management.

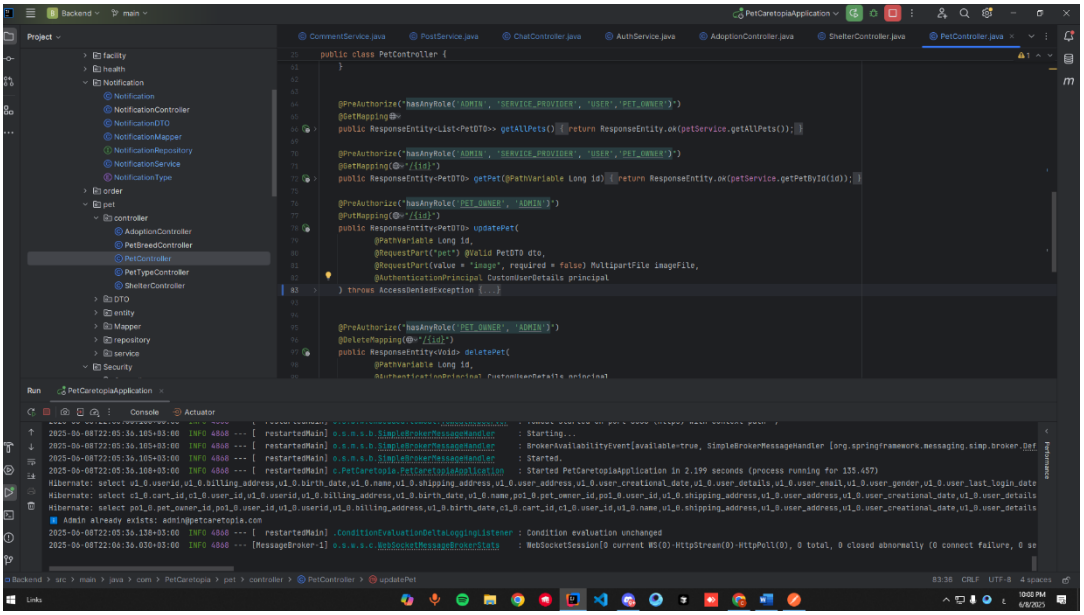
The screenshot below illustrates this implementation:



Spring Security Configuration

To restrict access based on user roles, we define secured endpoints using either:

- @PreAuthorize("hasRole('ADMIN')") on controller methods



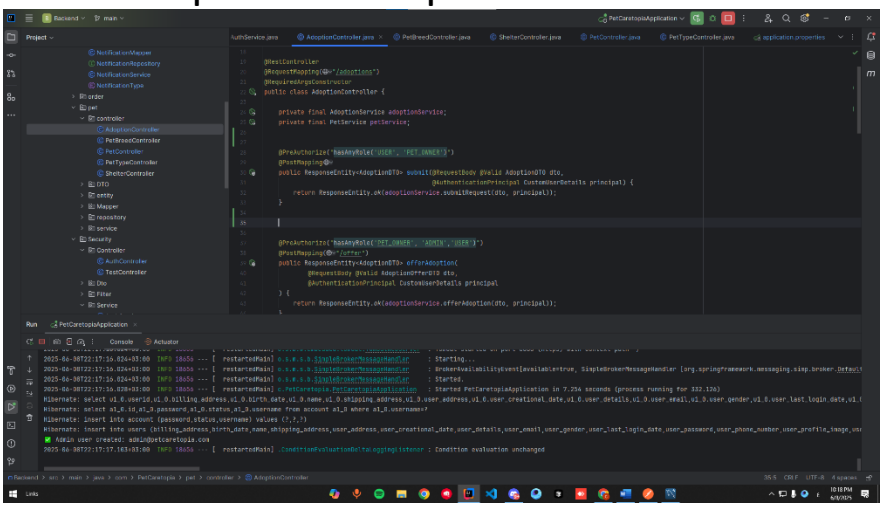
Adoption Module – Submit Request

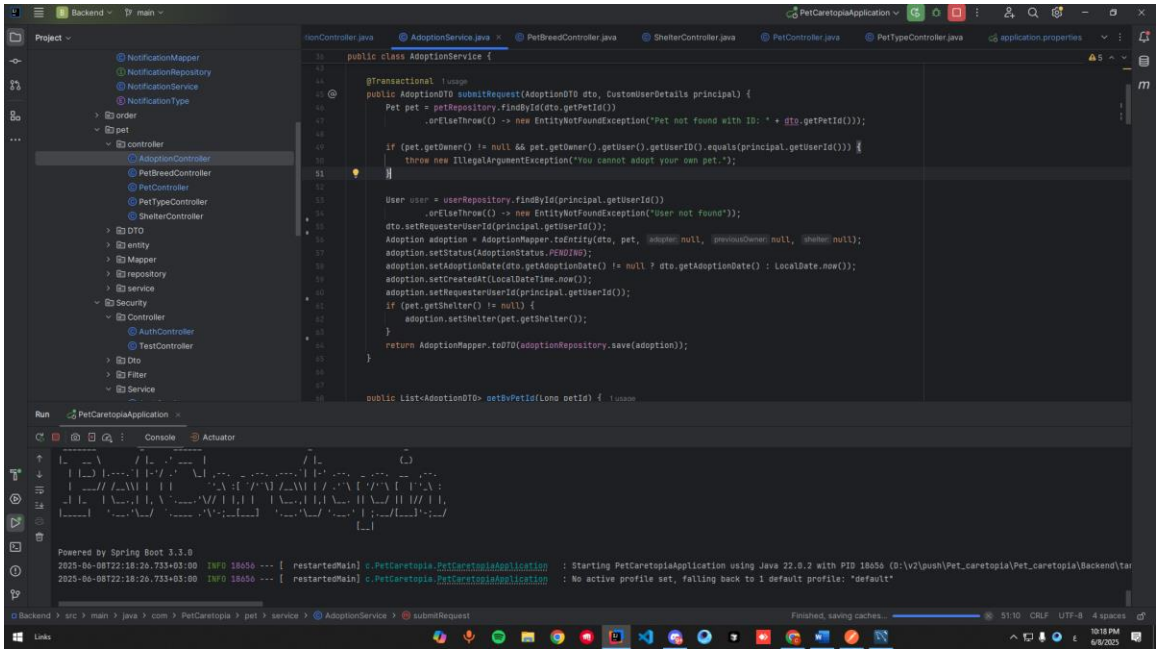
This endpoint allows a USER or PET_OWNER to submit a request to adopt a pet.

- **Endpoint:** POST /adoptions
- **Roles Allowed:** USER, PET_OWNER

- **Description:**

  The user provides the pet ID and their details in the request body. The system registers the request and links it to both the pet and adopter.

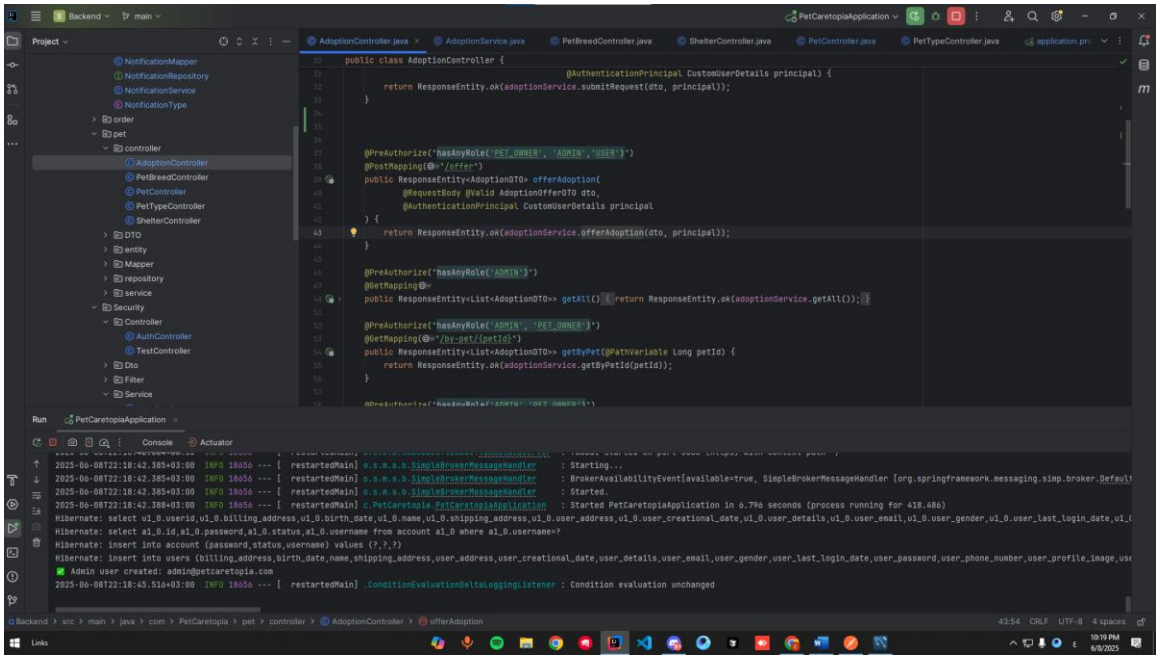


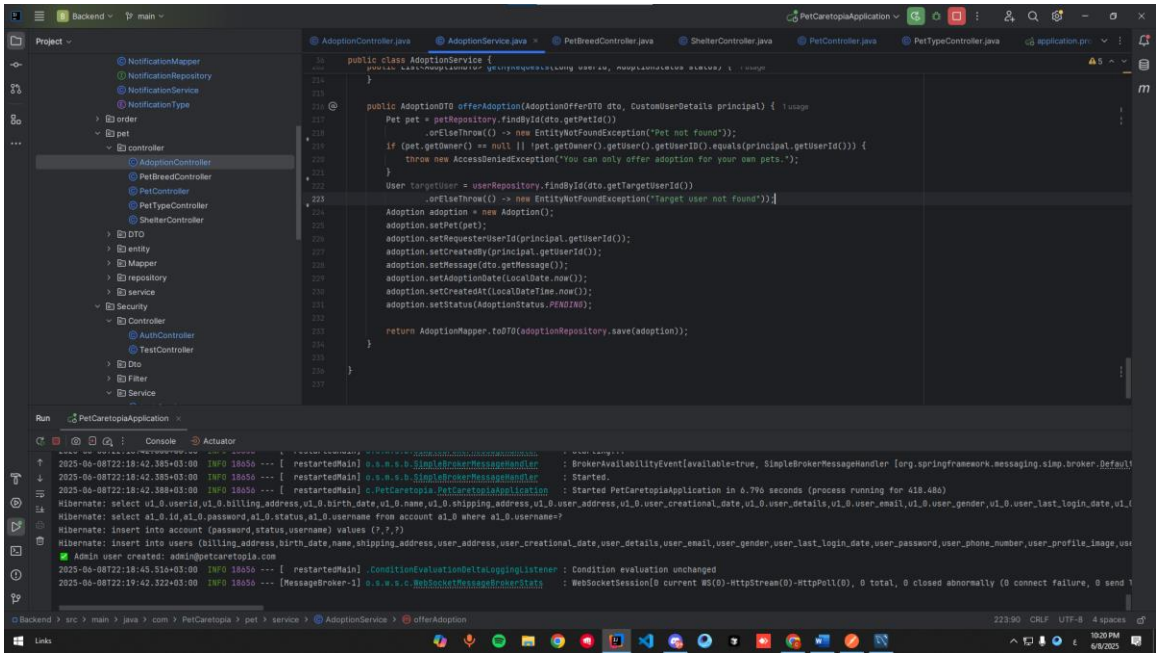


Offer Pet for Adoption

This API is used by pet owners to offer their pets for adoption.

- **Endpoint:** POST /adoptions/offer
- **Roles Allowed:** PET_OWNER, ADMIN
- **Description:**
  The pet owner specifies the pet ID and conditions. The pet's status is updated to "Available for Adoption".

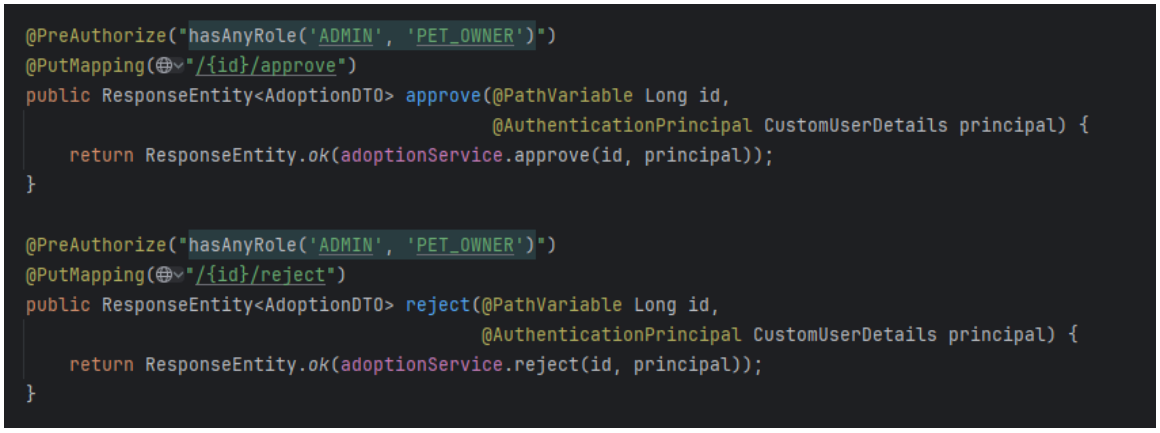




Approve/Reject Adoption Request

Used by shelter admins or pet owners to **approve or reject** adoption requests.

- **Endpoints:**
  - PUT /adoptions/{id}/approve
  - PUT /adoptions/{id}/reject
- **Roles Allowed:** ADMIN, PET_OWNER
- **Description:**
  This marks the request as approved or rejected. Once approved, the pet becomes unavailable for further adoption.



```
@PreAuthorize("hasAnyRole('ADMIN', 'PET_OWNER')")
@PutMapping("/{id}/approve")
public ResponseEntity<AdoptionDTO> approve(@PathVariable Long id,
                                           @AuthenticationPrincipal CustomUserDetails principal) {
    return ResponseEntity.ok(adoptionService.approve(id, principal));
}

@PreAuthorize("hasAnyRole('ADMIN', 'PET_OWNER')")
@PutMapping("/{id}/reject")
public ResponseEntity<AdoptionDTO> reject(@PathVariable Long id,
                                          @AuthenticationPrincipal CustomUserDetails principal) {
    return ResponseEntity.ok(adoptionService.reject(id, principal));
}
```

```
public class AdoptionService {                                                                    ⚠ 5
    public AdoptionDTO approve(Long id, CustomUserDetails principal) {  1 usage
        Adoption adoption = adoptionRepository.findById(id)
                .orElseThrow(() -> new EntityNotFoundException("Adoption not found"));
        if (adoption.getStatus() == AdoptionStatus.APPROVED || adoption.getStatus() == AdoptionStatus.REJECTED) {
            throw new IllegalStateException("This adoption request has already been processed.");
        }
        Pet pet = adoption.getPet();
        boolean isAdmin = principal.getRole().equals("ADMIN");
        if (!isAdmin) {
            if (pet.getOwner() == null || !pet.getOwner().getUser().getUserID().equals(principal.getUserId())) {
                throw new AccessDeniedException("You can only approve adoptions for your own pets.");
            }
        }
        User user = userRepository.findById(adoption.getCreatedBy())
                .orElseThrow(() -> new EntityNotFoundException("User who submitted the adoption request not found"));
        PetOwner adopter = petOwnerRepository.findByUser_UserID(user.getUserID())
                .orElseGet(() -> {
                    PetOwner newOwner = new PetOwner();
                    newOwner.setUser(user);
                    PetOwner saved = petOwnerRepository.save(newOwner);

                    user.setUserRole(PET_OWNER);
                    userRepository.save(user);
                    return saved;
                });
        pet.adopt(adopter);
        pet.setAvailableForAdoption(false);
        adoption.setAdopter(adopter);
        adoption.setStatus(AdoptionStatus.APPROVED);
        adoption.setPreviousOwner(pet.getOwner());
        petRepository.save(pet);
        return AdoptionMapper.toDTO(adoptionRepository.save(adoption));
    }
```

```
@Transactional  1 usage
public AdoptionDTO reject(Long id, CustomUserDetails principal) {
    Adoption adoption = adoptionRepository.findById(id)
            .orElseThrow(() -> new EntityNotFoundException("Adoption not found"));
    Pet pet = adoption.getPet();
    if (adoption.getStatus() == AdoptionStatus.APPROVED || adoption.getStatus() == AdoptionStatus.REJECTED) {
        throw new IllegalStateException("This adoption request has already been processed.");
    }
    boolean isAdmin = principal.getRole().equals("ADMIN");

    if (!isAdmin) {
        if (pet.getOwner() == null || !pet.getOwner().getUser().getUserID().equals(principal.getUserId())) {
            throw new AccessDeniedException("You can only reject adoptions for your own pets.");
        }
    }
    adoption.setStatus(AdoptionStatus.REJECTED);
    Adoption savedAdoption = adoptionRepository.save(adoption);
    Long requesterUserId = adoption.getCreatedBy();
    if (requesterUserId != null) {
        User user = userRepository.findById(requesterUserId).orElse( other: null);
        if (user != null && petOwnerRepository.existsByUser_UserID(user.getUserID())) {
            PetOwner adopter = petOwnerRepository.findByUser_UserID(user.getUserID()).orElse( other: null);
            if (adopter != null) {
                List<Adoption> otherAdoptions = adoptionRepository.findByAdopter_PetOwnerId(adopter.getPetOwnerId()).stream()
                        .filter(a -> !Objects.equals(a.getId(), adoption.getId()))
                        .filter(a -> a.getStatus() != AdoptionStatus.REJECTED)
                        .toList();
                if (otherAdoptions.isEmpty()) {
                    petOwnerRepository.delete(adopter);
                    user.setUserRole(USER);
                    userRepository.save(user);
                }
            }
```

View Available Pets for Adoption

This is a public endpoint that lists all pets currently available for adoption.

- **Endpoint:** GET /adoptions/available-for-adoption
- **Roles Allowed:** Public (no token required)
- **Description:**
  Returns a list of pets marked as available, including their type, age, and shelter details.

```
@GetMapping("/available-for-adoption")
public ResponseEntity<List<PetDTO>> getAvailablePets() {
    List<PetDTO> pets = petService.getAvailableForAdoption();
    return ResponseEntity.ok(pets);
}
```

```
public List<PetDTO> getAvailableForAdoption() {  1 usage
    return petRepository.findByIsAvailableForAdoptionTrue().stream()  Stream<Pet>
            .map(PetMapper::toDTO)  Stream<PetDTO>
            .toList();
}
```

Pet Module – Add Pet

This endpoint allows a PET_OWNER to add a new pet profile (including optional image upload).

- **Endpoint:** POST /pets
- **Roles Allowed:** USER (converted to PET_OWNER), PET_OWNER, ADMIN

- **Description:**

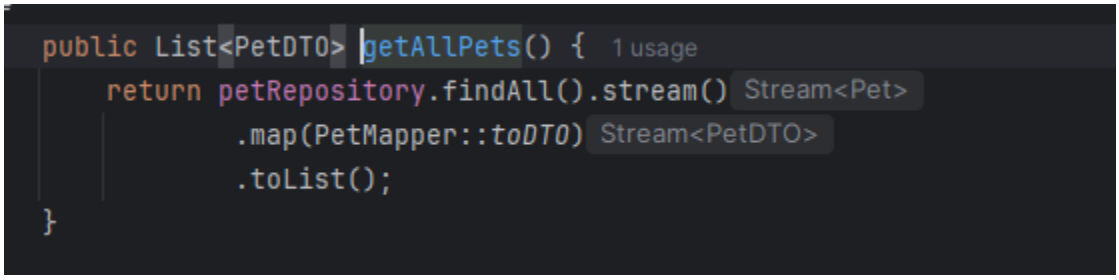
  Creates a new pet entity with attributes like name, type, breed, age, and optionally image. If the user is of type USER, they are upgraded to PET_OWNER.

```java
@PreAuthorize("hasAnyRole('USER','PET_OWNER','ADMIN')")
@PostMapping
public ResponseEntity<PetDTO> createPet(
        @RequestPart("pet") @Valid PetDTO dto,
        @RequestPart(value = "image", required = false) MultipartFile imageFile,
        @AuthenticationPrincipal CustomUserDetails principal
) {
    Long userId = principal.getUserId();
    User user = userRepository.findById(userId)
            .orElseThrow(() -> new IllegalArgumentException("User not found"));
    if (dto.getShelterId() == null) {
        PetOwner owner = user.getPetOwner();

        if (owner == null) {
            owner = new PetOwner();
            owner.setUser(user);
            user.setPetOwner(owner);
            if(user.getUserRole().equals(User.Role.USER)) {
                user.setUserRole(User.Role.PET_OWNER);
            }
            user = userRepository.save(user);
            owner = user.getPetOwner();
        }
        dto.setOwnerId(owner.getPetOwnerId());
    }
    return ResponseEntity.ok(petService.createPet(dto, imageFile));
}

public PetDTO createPet(PetDTO dto, MultipartFile imageFile) {  1 usage
    PetType type = petTypeRepository.findByTypeName(dto.getPetTypeName())
            .orElseThrow(() -> new IllegalArgumentException("Invalid pet type"));
    PetBreed breed = petBreedRepository.findByBreedNameAndPetType(dto.getPetBreedName(), type)
            .orElseThrow(() -> new IllegalArgumentException("Invalid pet breed"));
    PetOwner owner = null;
    Shelter shelter = null;
    if (dto.getOwnerId() != null) {
        owner = petOwnerRepository.findById(dto.getOwnerId()).orElse( other: null);
        if (owner == null) {
            User user = userRepository.findById(dto.getOwnerId())
                    .orElseThrow(() -> new IllegalArgumentException("User not found"));
            if (user.getUserRole() != User.Role.PET_OWNER) {
                user.setUserRole(User.Role.PET_OWNER);
            }
            owner = new PetOwner();
            owner.setUser(user);
            user.setPetOwner(owner);
            userRepository.save(user);
        }
    }
    if (dto.getShelterId() != null) {
        shelter = shelterRepository.findById(dto.getShelterId())
                .orElseThrow(() -> new IllegalArgumentException("Shelter not found"));
    }
    if (owner != null && shelter != null) {
        throw new IllegalArgumentException("Pet cannot have both owner and shelter.");
    }
    Pet pet = PetMapper.toEntity(dto, type, breed, owner, shelter);
    if (imageFile != null && !imageFile.isEmpty()) {
        String imageUrl = imageUploadService.uploadMultipartFile(imageFile);
        pet.setImageUrl(imageUrl);
    }
    return PetMapper.toDTO(petRepository.save(pet));
}
```

Pet Module – Get My Pets

This endpoint returns all pets registered by the currently authenticated pet owner.

- **Endpoint:** GET /pets/mine
- **Roles Allowed:** PET_OWNER
- **Description:**
  Helpful for viewing and managing personal pets.



Pet Type and Breed – Filtering by Type

Used to get breeds based on selected type (e.g., Dog, Cat).

- **Endpoint:** GET /pet-breeds/by-type?type=CAT
- **Roles Allowed:** ALL

Description:

Returns filtered breeds to assist users when adding new pets.

```java
@PreAuthorize("hasAnyRole('ADMIN', 'SERVICE_PROVIDER', 'USER','PET_OWNER')")
@GetMapping("/filter")
public ResponseEntity<List<PetDTO>> filterByType(@RequestParam String type) {
    return ResponseEntity.ok(petService.getPetsByType(type));
}
```

```java
public List<PetDTO> getPetsByType(String typeName) {  1 usage
    PetType type = petTypeRepository.findByTypeName(typeName)
            .orElseThrow(() -> new IllegalArgumentException("Invalid type"));
    return petRepository.findByPetType(type).stream().map(PetMapper::toDTO).toList();
}
```

Shelter – Get Pets in Specific Shelter

This allows any user to see all pets hosted by a particular shelter.

- **Endpoint:** GET /shelters/{shelterId}/pets
- **Roles Allowed:** USER, PET_OWNER, ADMIN
- **Description:**
  Displays a list of pets available in a specific shelter with IDs and brief info.

```java
@PreAuthorize("hasAnyRole('ADMIN', 'PET_OWNER', 'USER')")
@GetMapping("/{shelterId}/pets")
public ResponseEntity<List<PetDTO>> getPetsByShelter(@PathVariable Long shelterId) {
    return ResponseEntity.ok(shelterService.getPetsByShelter(shelterId));
}
```

```java
public List<PetDTO> getPetsByShelter(Long shelterId) {  1 usage
    Shelter shelter = shelterRepository.findById(shelterId)
            .orElseThrow(() -> new EntityNotFoundException("Shelter not found"));

    List<Pet> pets = petRepository.findByShelter(shelter);

    return pets.stream()  Stream<Pet>
            .map(PetMapper::toDTO)  Stream<PetDTO>
            .toList();
}
```

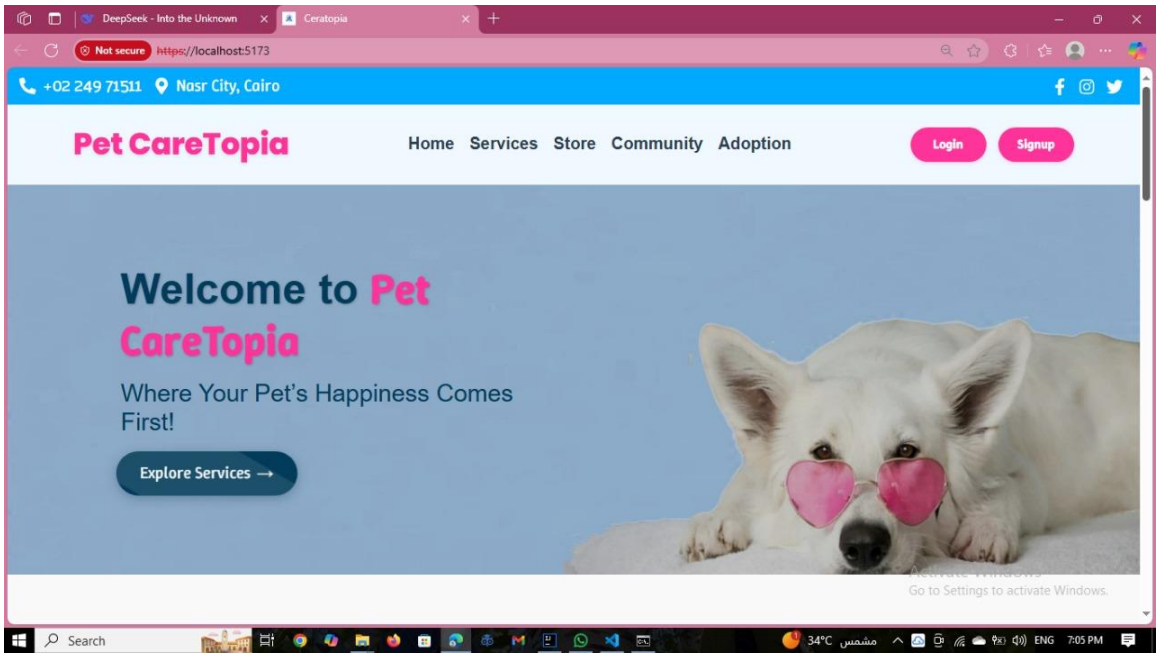
These APIs are secured using Spring Security with role-based access, and their implementation ensures clear separation of concerns, data validation, and user permissions at every interaction level.

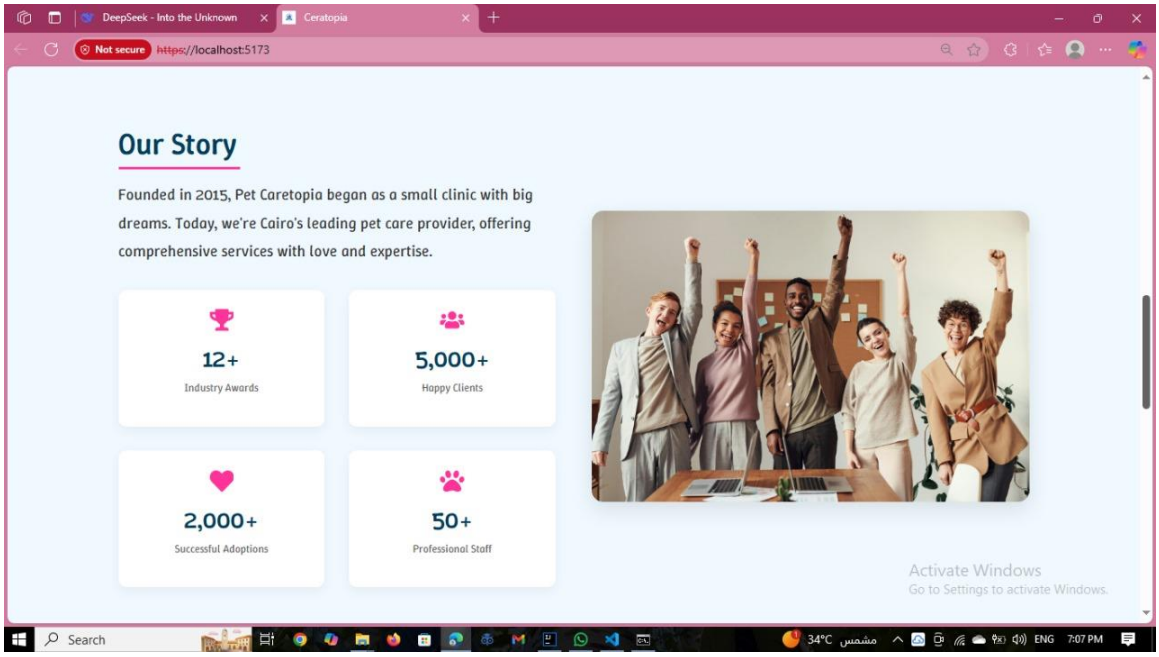
Chapter 6: Software

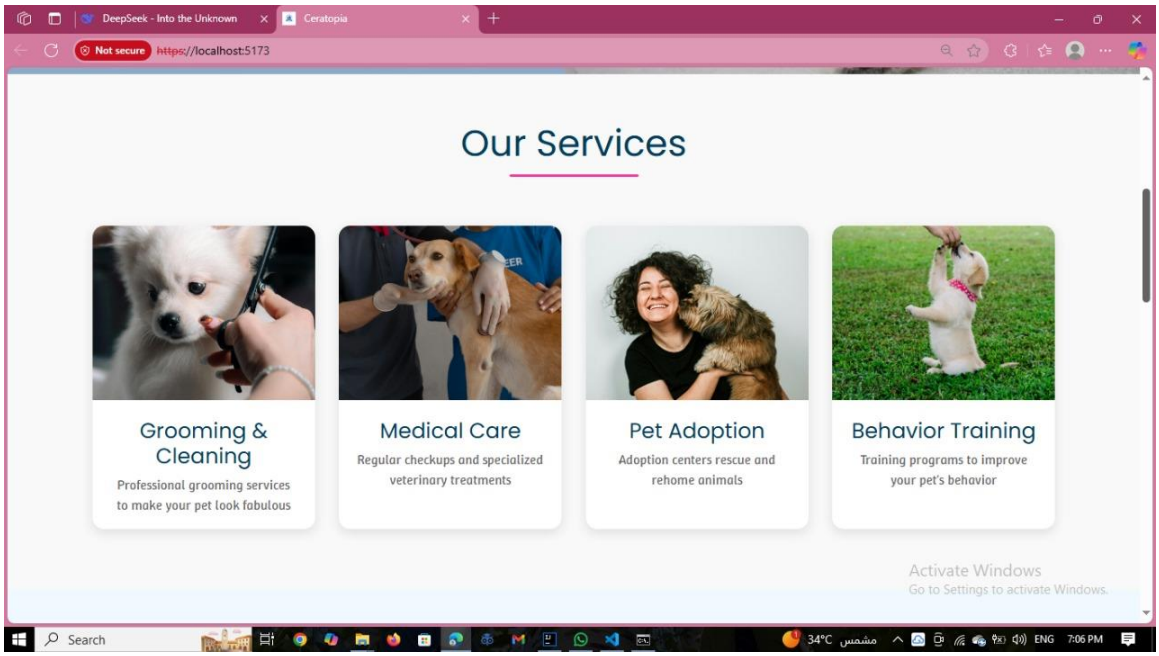
4.1 Overview

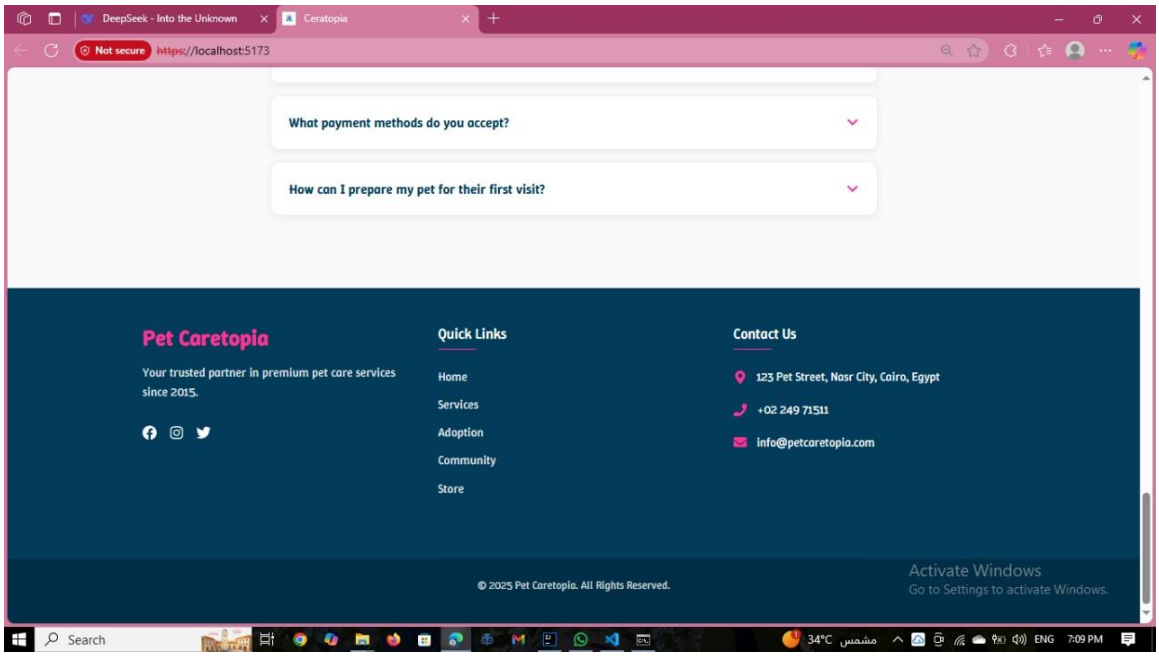
This chapter is covering the system architecture and the design of the system by using diagrams

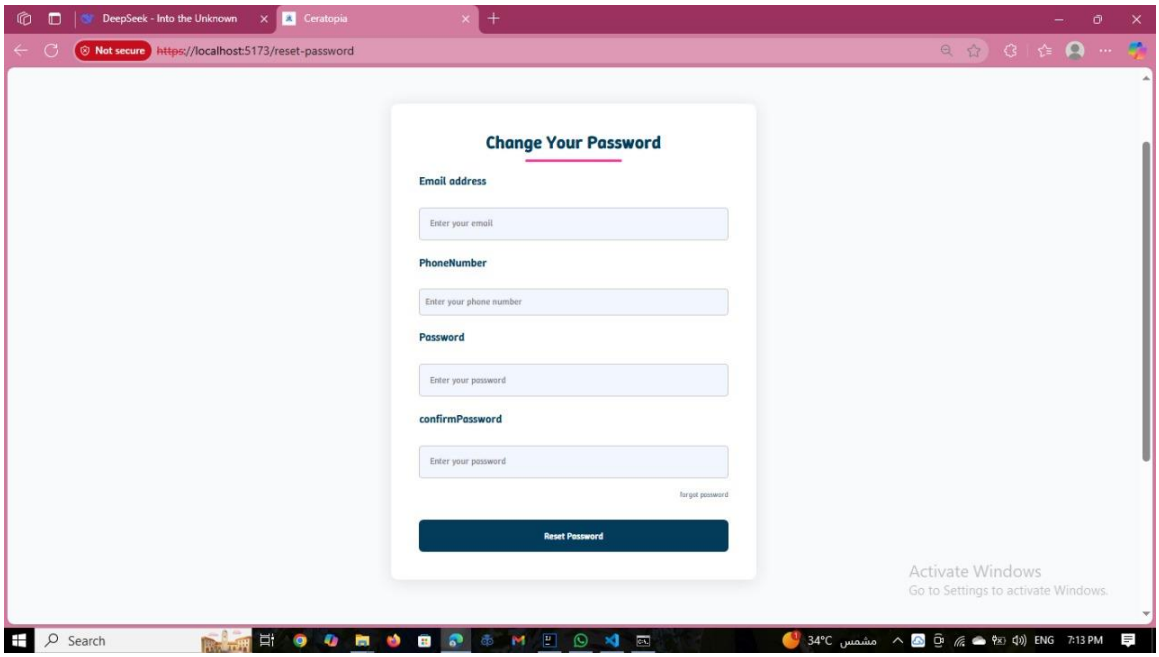
4.2 User Interface Design

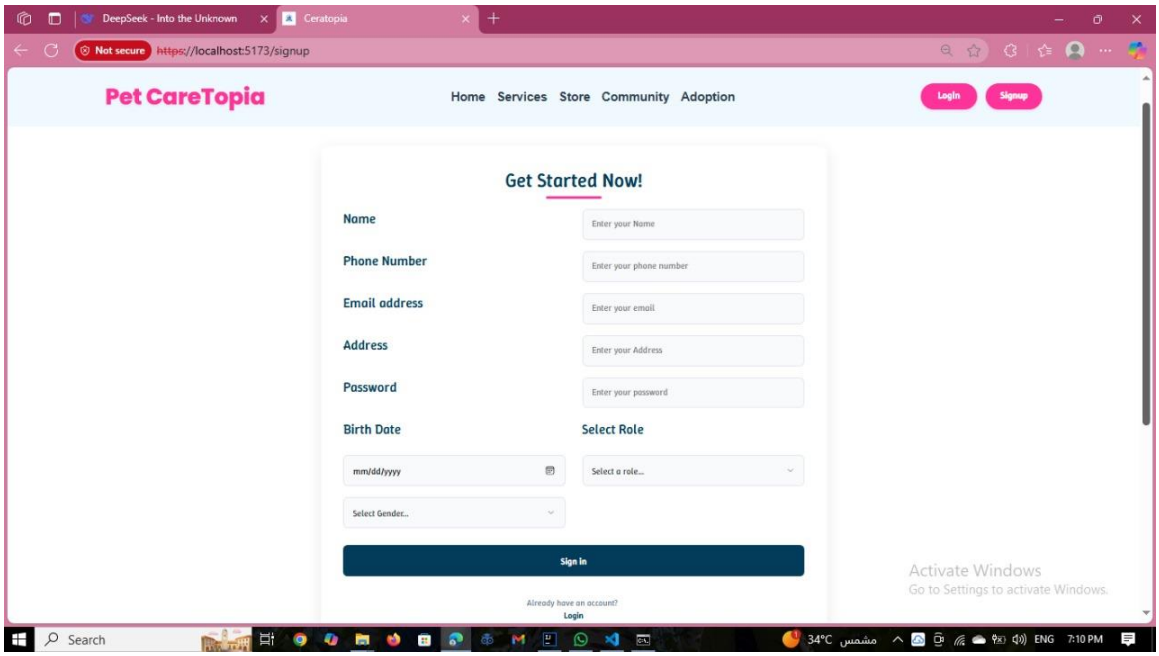

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173
+02 249 71511
Nasr City, Cairo
Pet CareTopia
Home
Services
Store
Community
Adoption
Login
Signup
Welcome to Pet CareTopia
Where Your Pet's Happiness Comes First!
Explore Services →
Go to Settings to activate Windows.
Search
34°C
ENG
7:05 PM

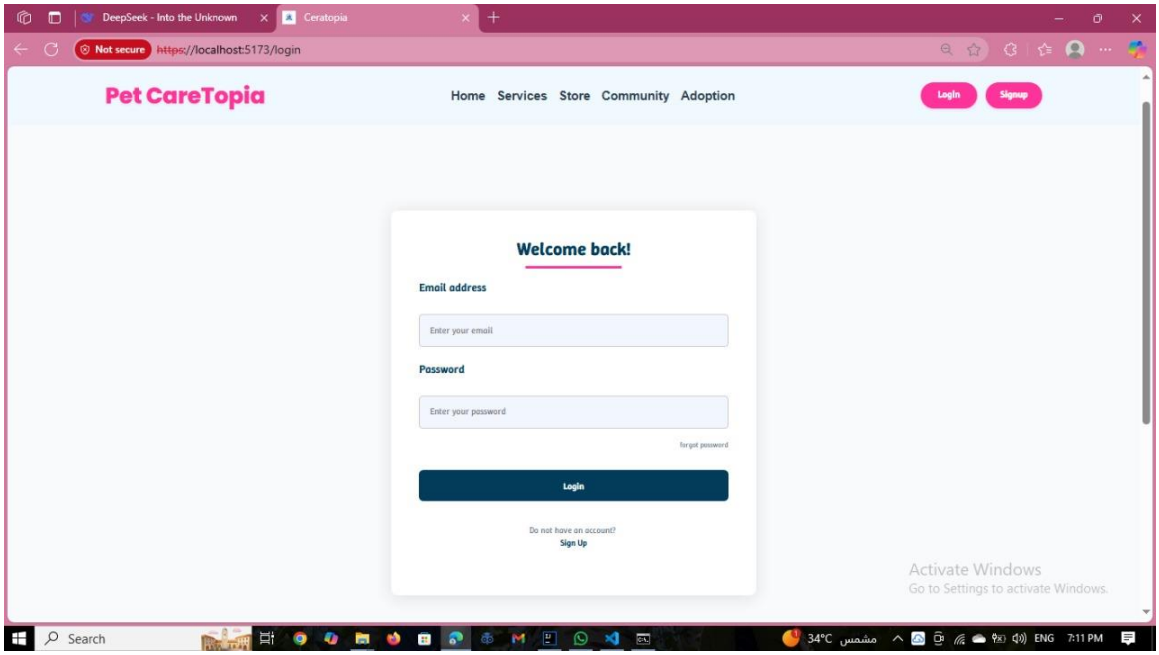

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173
Our Story
Founded in 2015, Pet Caretopia began as a small clinic with big dreams. Today, we're Cairo's leading pet care provider, offering comprehensive services with love and expertise.
12+
Industry Awards
5,000+
Happy Clients
2,000+
Successful Adoptions
50+
Professional Staff
Activate Windows
Go to Settings to activate Windows.
Search
34°C
ENG
7:07 PM

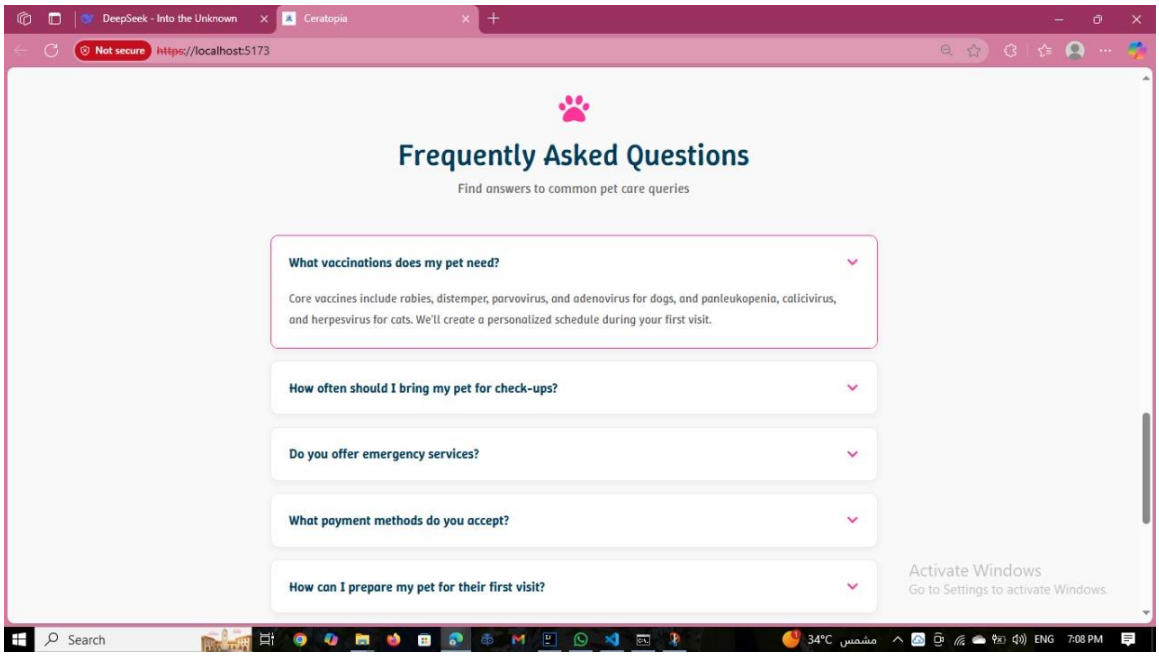

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173
Our Services
Grooming & Cleaning
Professional grooming services to make your pet look fabulous
Medical Care
Regular checkups and specialized veterinary treatments
Pet Adoption
Adoption centers rescue and rehome animals
Behavior Training
Training programs to improve your pet's behavior
Activate Windows
Go to Settings to activate Windows.
Search
34°C
ENG
7:06 PM

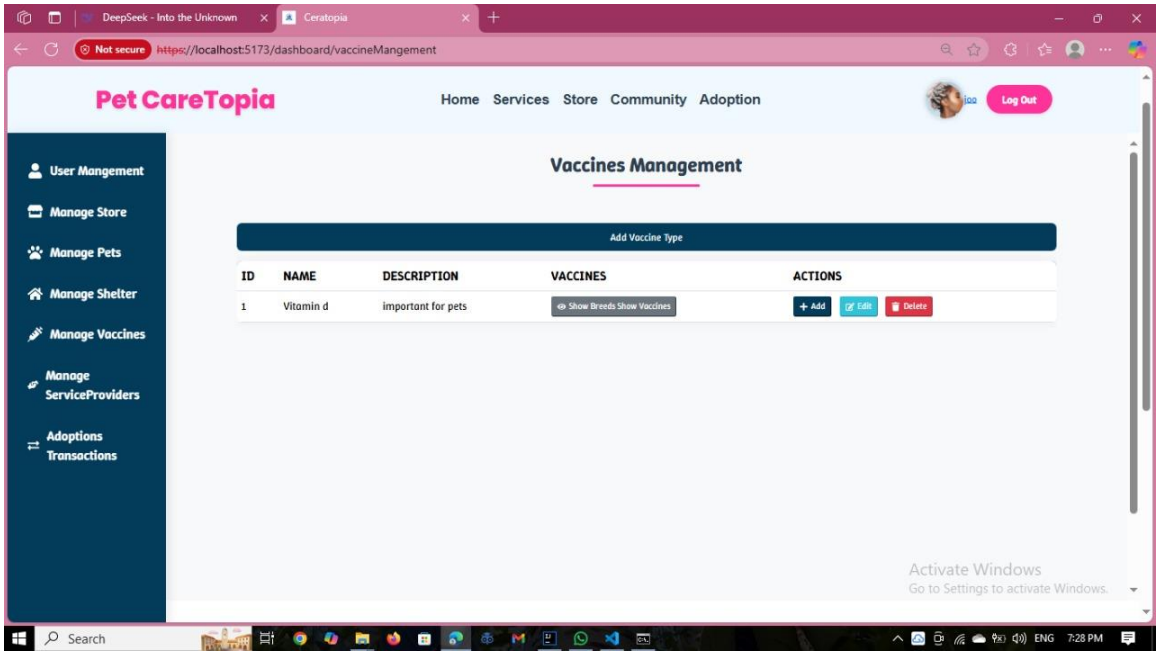

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173
What payment methods do you accept?
How can I prepare my pet for their first visit?
Pet Caretopia
Your trusted partner in premium pet care services since 2015.
Quick Links
Home
Services
Adoption
Community
Store
Contact Us
123 Pet Street, Nasr City, Cairo, Egypt
+02 249 71511
info@petcaretopia.com
© 2025 Pet Caretopia. All Rights Reserved.
Activate Windows
Go to Settings to activate Windows.
Search
34°C مشمس
ENG
7:09 PM

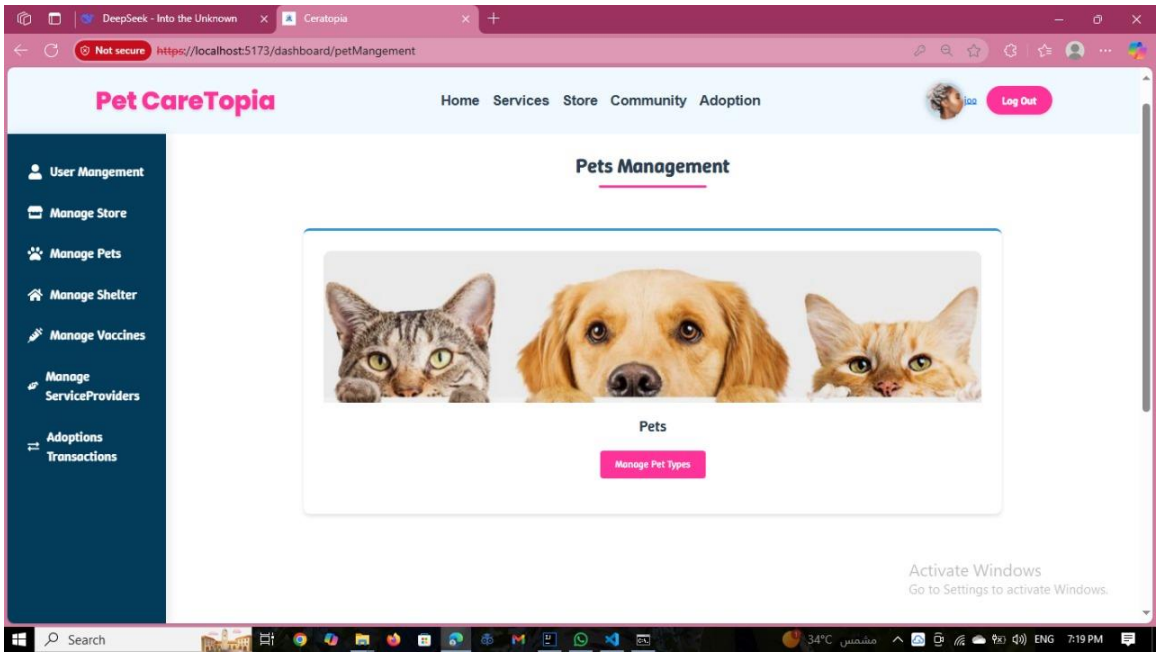

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/reset-password
Change Your Password
Email address
Enter your email
PhoneNumber
Enter your phone number
Password
Enter your password
confirmPassword
Enter your password
forgot password
Reset Password
Activate Windows
Go to Settings to activate Windows.
Search
34°C مشمس
ENG
7:13 PM

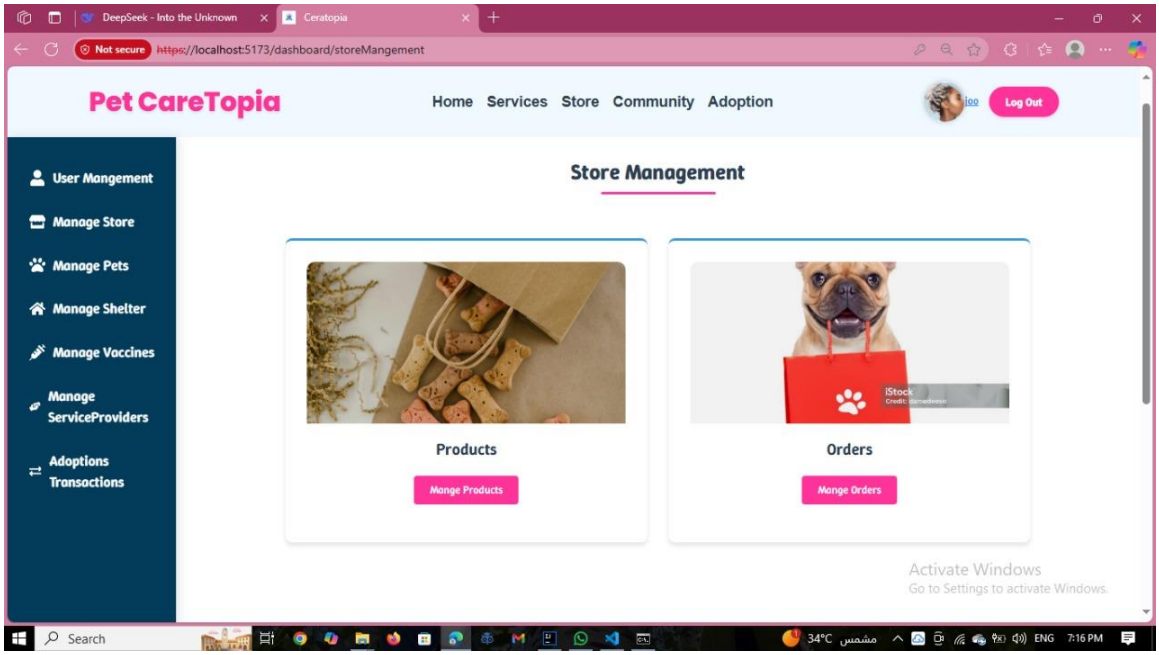

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/signup
Pet CareTopia
Home Services Store Community Adoption
Login
Signup
Get Started Now!
Name
Enter your Name
Phone Number
Enter your phone number
Email address
Enter your email
Address
Enter your Address
Password
Enter your password
Birth Date
Select Role
mm/dd/yyyy
Select a role...
Select Gender..
Sign In
Already have an account?
Login
Activate Windows
Go to Settings to activate Windows.
Search
34°C
ENG
7:10 PM

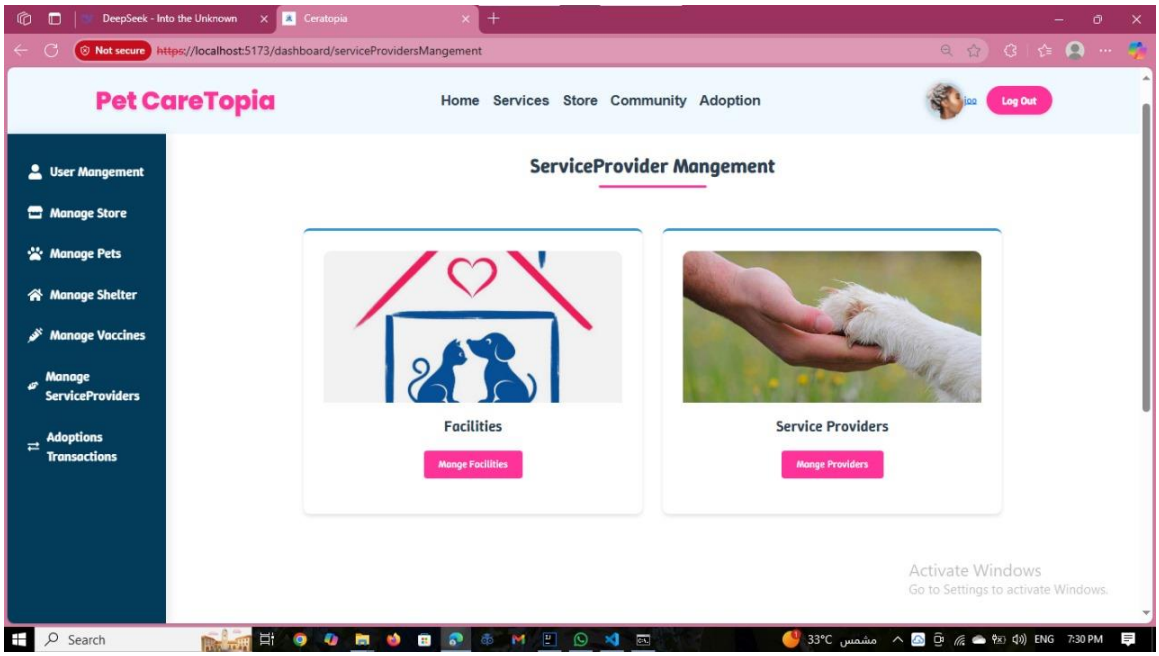

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/login
Pet CareTopia
Home Services Store Community Adoption
Login
Signup
Welcome back!
Email address
Enter your email
Password
Enter your password
forgot password
Login
Do not have an account?
Sign Up
Activate Windows
Go to Settings to activate Windows.
Search
34°C
ENG
7:11 PM

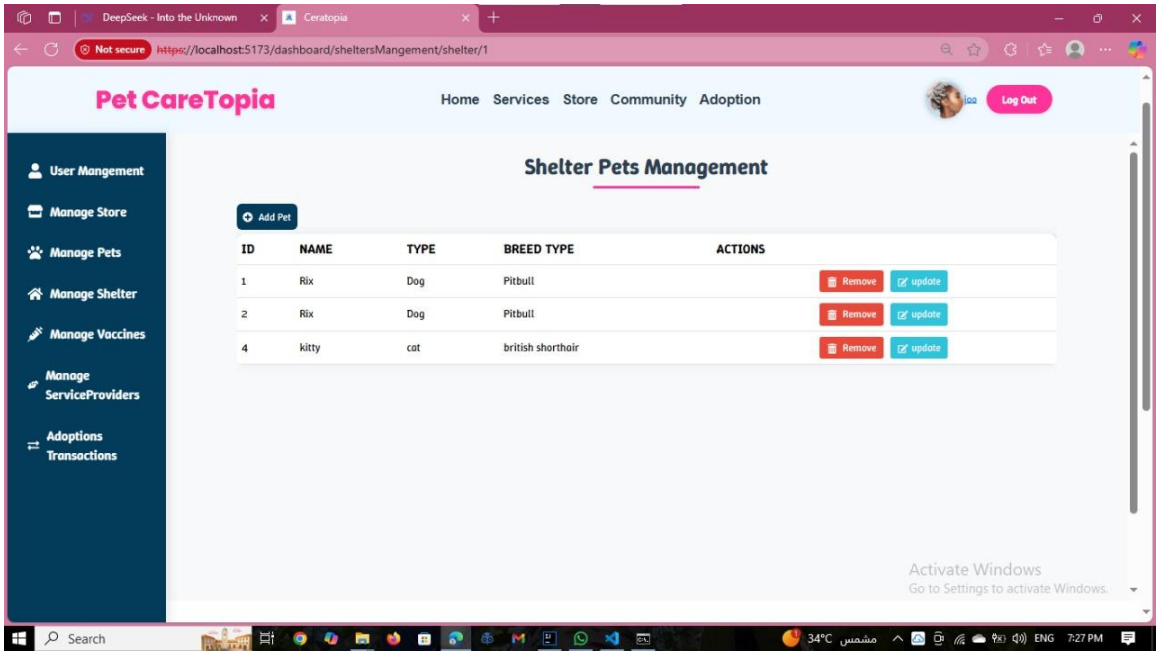

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173
Frequently Asked Questions
Find answers to common pet care queries
What vaccinations does my pet need?
Core vaccines include rabies, distemper, parvovirus, and adenovirus for dogs, and panleukopenia, calicivirus, and herpesvirus for cats. We'll create a personalized schedule during your first visit.
How often should I bring my pet for check-ups?
Do you offer emergency services?
What payment methods do you accept?
How can I prepare my pet for their first visit?
Activate Windows
Go to Settings to activate Windows.
Search
34°C
ENG
7:08 PM

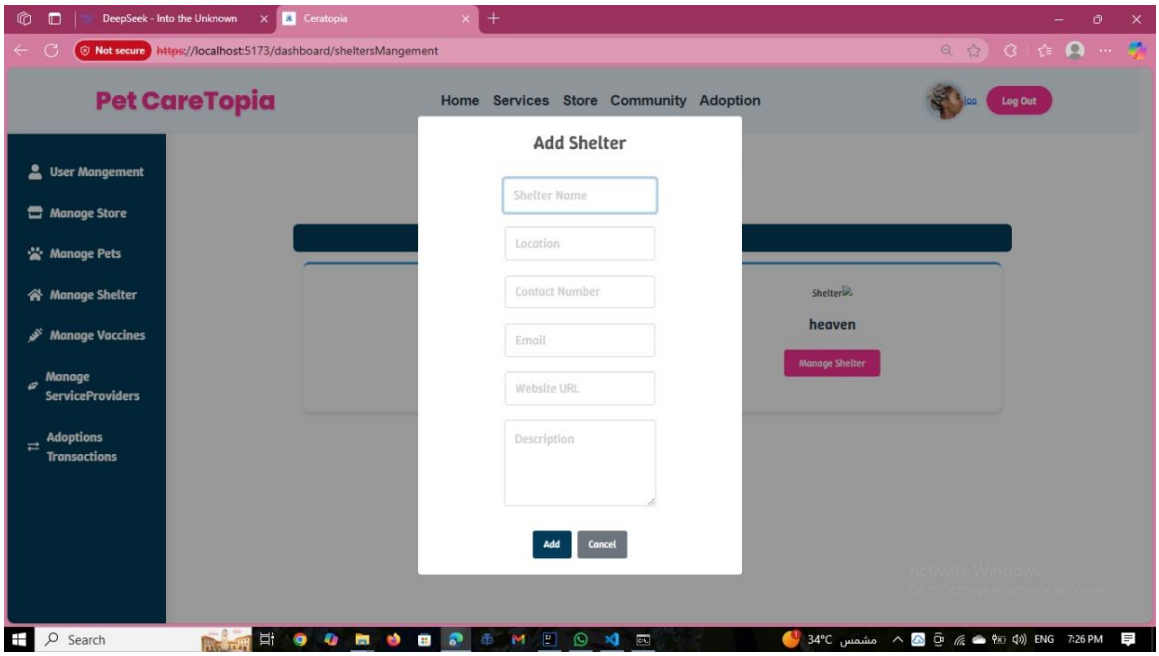

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/dashboard/vaccineMangement
Pet CareTopia
Home
Services
Store
Community
Adoption
Log Out
User Mangement
Manage Store
Manage Pets
Manage Shelter
Manage Vaccines
Manage ServiceProviders
Adoptions Transactions
Vaccines Management
Add Vaccine Type
ID
NAME
DESCRIPTION
VACCINES
ACTIONS
1
Vitamin d
important for pets
Show Breeds Show Vaccines
+ Add
Edit
Delete
Activate Windows
Go to Settings to activate Windows.
Search
ENG
7:28 PM

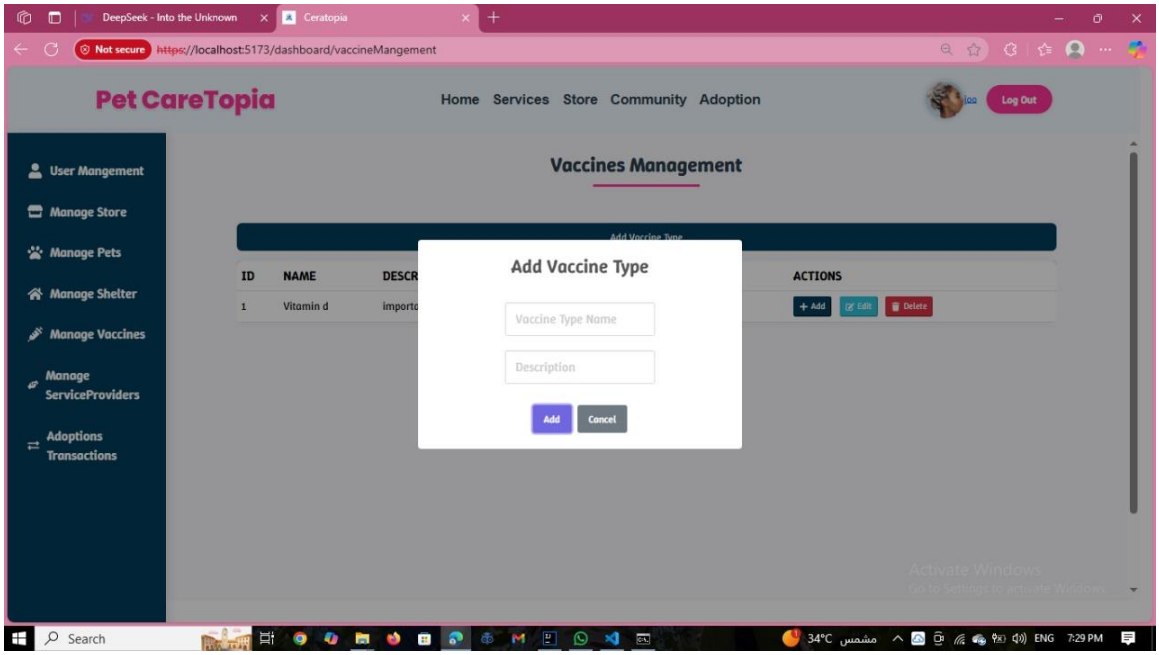

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/dashboard/petMangement
Pet CareTopia
Home Services Store Community Adoption
Log Out
User Mangement
Manage Store
Manage Pets
Manage Shelter
Manage Vaccines
Manage ServiceProviders
Adoptions Transactions
Pets Management
Pets
Manage Pet Types
Activate Windows
Go to Settings to activate Windows.
Search
34°C
ENG
7:19 PM

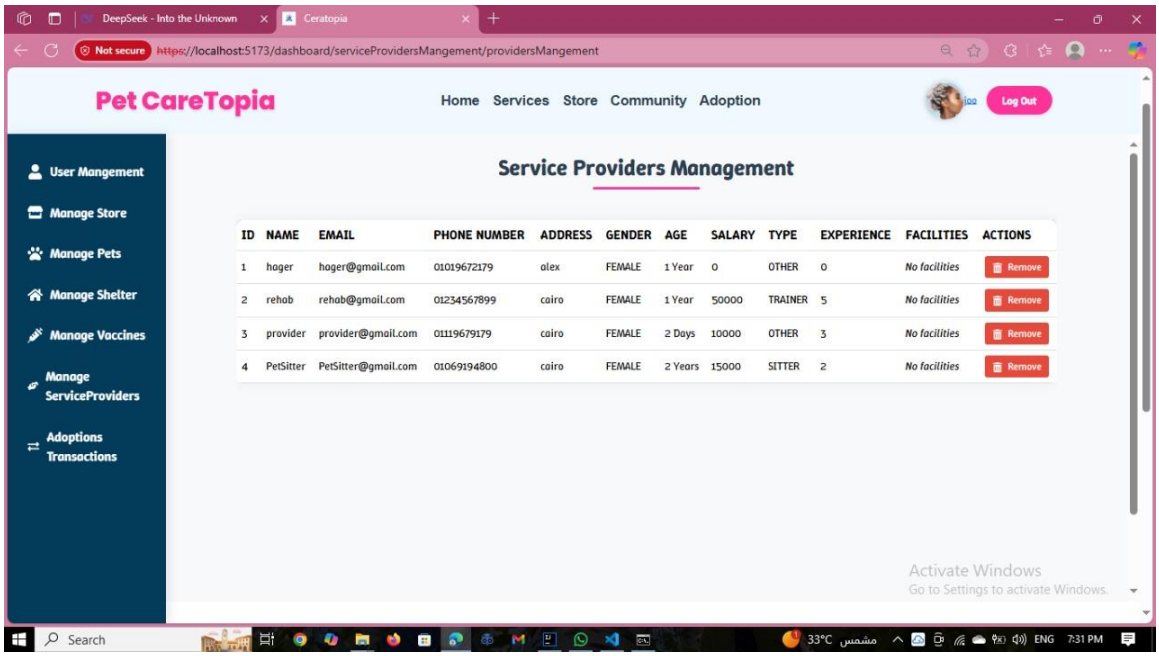

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/dashboard/storeMangement
Pet CareTopia
Home Services Store Community Adoption
Log Out
User Mangement
Manage Store
Manage Pets
Manage Shelter
Manage Vaccines
Manage ServiceProviders
Adoptions Transactions
Store Management
Products
Mange Products
Orders
Mange Orders
Activate Windows
Go to Settings to activate Windows.
Search
34°C
ENG
7:16 PM

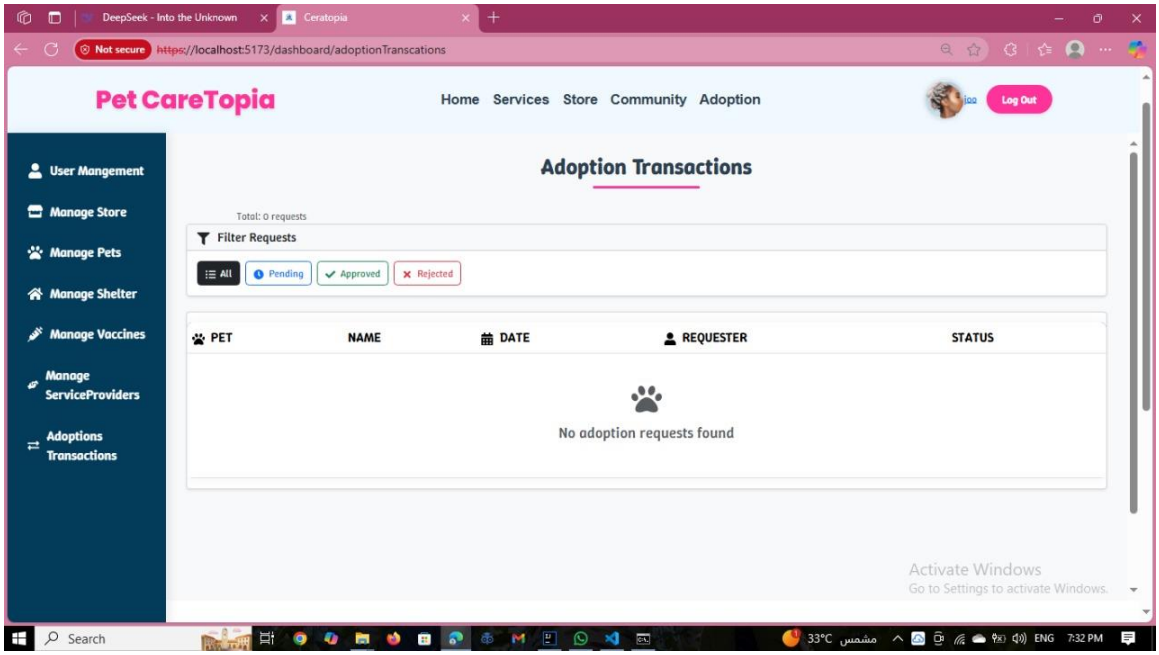

DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/dashboard/serviceProvidersMangement
Pet CareTopia
Home Services Store Community Adoption
Log Out
User Mangement
Manage Store
Manage Pets
Manage Shelter
Manage Vaccines
Manage ServiceProviders
Adoptions Transactions
ServiceProvider Mangement
Facilities
Mange Facilities
Service Providers
Mange Providers
Activate Windows
Go to Settings to activate Windows.
Search
33°C
ENG
7:30 PM


DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/dashboard/sheltersMangement/shelter/1
Pet CareTopia
Home Services Store Community Adoption
Log Out
User Mangement
Manage Store
Manage Pets
Manage Shelter
Manage Vaccines
Manage ServiceProviders
Adoptions Transactions
Shelter Pets Management
Add Pet
ID NAME TYPE BREED TYPE ACTIONS
1 Rix Dog Pitbull Remove update
2 Rix Dog Pitbull Remove update
4 kitty cat british shorthair Remove update
Activate Windows
Go to Settings to activate Windows.
Search
34°C
ENG
7:27 PM


DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/dashboard/sheltersMangement
Pet CareTopia
Home Services Store Community Adoption
Log Out
User Mangement
Manage Store
Manage Pets
Manage Shelter
Manage Vaccines
Manage ServiceProviders
Adoptions Transactions
Add Shelter
Shelter Name
Location
Contact Number
Email
Website URL
Description
Add
Cancel
Shelter
heaven
Manage Shelter
Activate Windows
Search
34°C
ENG
7:26 PM


DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/dashboard/vaccineMangement
Pet CareTopia
Home Services Store Community Adoption
Log Out
User Mangement
Manage Store
Manage Pets
Manage Shelter
Manage Vaccines
Manage ServiceProviders
Adoptions Transactions
Vaccines Management
ID
NAME
DESCR
ACTIONS
1
Vitamin d
+ Add
Edit
Delete
Add Vaccine Type
Vaccine Type Name
Description
Add
Cancel
Activate Windows
Search
34°C
ENG
7:29 PM


DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/dashboard/serviceProvidersMangement/providersMangement
Pet CareTopia
Home Services Store Community Adoption
Log Out
User Mangement
Manage Store
Manage Pets
Manage Shelter
Manage Vaccines
Manage ServiceProviders
Adoptions Transactions
Service Providers Management
ID NAME EMAIL PHONE NUMBER ADDRESS GENDER AGE SALARY TYPE EXPERIENCE FACILITIES ACTIONS
1 hager hager@gmail.com 01019672179 alex FEMALE 1 Year 0 OTHER 0 No facilities Remove
2 rehab rehab@gmail.com 01234567899 cairo FEMALE 1 Year 50000 TRAINER 5 No facilities Remove
3 provider provider@gmail.com 01119679179 cairo FEMALE 2 Days 10000 OTHER 3 No facilities Remove
4 PetSitter PetSitter@gmail.com 01069194800 cairo FEMALE 2 Years 15000 SITTER 2 No facilities Remove
Activate Windows
Go to Settings to activate Windows.
Search
33°C
ENG
7:31 PM


DeepSeek - Into the Unknown
Ceratopia
Not secure
https://localhost:5173/dashboard/adoptionTranscations
Pet CareTopia
Home Services Store Community Adoption
Log Out
User Mangement
Manage Store
Manage Pets
Manage Shelter
Manage Vaccines
Manage ServiceProviders
Adoptions Transactions
Adoption Transactions
Total: 0 requests
Filter Requests
All
Pending
Approved
Rejected
PET NAME DATE REQUESTER STATUS
No adoption requests found
Activate Windows
Go to Settings to activate Windows.
Search
33°C
ENG
7:32 PM

THANK YOU