ANIMAL-CARE VETERINARY CLINIC MANAGEMENT SYSTEM

Part I - Information System Analysis & Design (UML Diagrams)

Negar Pirasteh - Betty Dang - Hope Jeanine Ukundimana - Ngoc Yen Nhi Pham

Contents

Team Contribution Summary

Team Member	Responsibilities and Contributions
Negar Pirasteh	Designed the one **Class Diagram**, created the full **documentation file**, and wrote detailed explanations for the system overview and UML diagrams.
Betty Dang	Created the two **Use Case Diagrams** (general and specific), identified system actors and functionalities, and defined user interactions.
Hope Jeanine Ukundimana	Designed the four **Sequence Diagrams**, showing message flow and interactions between system objects and users.
Ngoc Yen Nhi Pham	Developed the four **Activity Diagrams**, illustrating the workflows for main system processes and aligning them with the use cases.

1. Introduction

The Animal-Care Veterinary Clinic Management System is a centralized web application developed to automate and streamline the daily operations of the clinic. It eliminates problems caused by paper files and spreadsheets such as double-booked appointments, lost records, and inefficient communication among staff.

The system provides secure, role-based access for different users and enables:

- Registration and management of animals and their owners
- Scheduling and management of veterinary appointments
- Recording of medical visits and treatments
- Generation of invoices and payments
- Centralized storage of all data for easy access and reporting

Its goal is to create an intuitive, secure, and scalable platform that supports the clinic's growth while reducing administrative workload and human error.

2. System Analysis

2.1 Problem Definition

- The clinic currently faces multiple operational challenges:
- Appointment scheduling conflicts between veterinarians
- Manual recording of patient data leading to lost or inconsistent information
- Difficulty generating bills or tracking payments
- Lack of centralized communication between veterinarians, receptionists, and administrators

2.2 System Objectives

The application is designed to:

- Maintain a complete database of veterinarians, owners, and animals.
- Automate the appointment scheduling process and prevent time conflicts.
- Provide each veterinarian with a clear, up-to-date schedule.
- Record all visits, diagnoses, and treatments accurately.
- Generate invoices and register payments securely.
- Restrict access through authentication and user roles.

2.3 Actors and Roles

Actor	Description
Administrator	Full access to the system. Manages users, reports, and billing data.
Receptionist	Registers new animals and owners, books appointments, and handles billing.
Veterinarian	Views personal schedule, records medical visits, and manages availability.
Pet Owner	Provides pet information and receives appointment confirmations.

3. Design Overview

The system is modeled using UML (Unified Modeling Language) to describe both its structure and behavior.

The following diagrams illustrate the application's design from different perspectives:

- **Use Case Diagrams:** show the system's main functionalities from the users' point of view and define what the system should do.
- **Class Diagram:** presents the static structure of the system, including its classes, attributes, methods, and relationships.
- **Activity Diagrams:** describe the detailed workflow of major processes and how each operation is executed step by step.
- **Sequence Diagrams:** show the order of interactions between actors and system components over time for each process.

3.1 Use Case Diagrams

3.1.1 General Use Case Diagram

Purpose:
The general use case diagram provides an overview of all system functionalities and how each actor interacts with the system. It represents the high-level behavior of the Animal-Care Veterinary Clinic Management System**,** focusing on what the system does rather than how it does it.

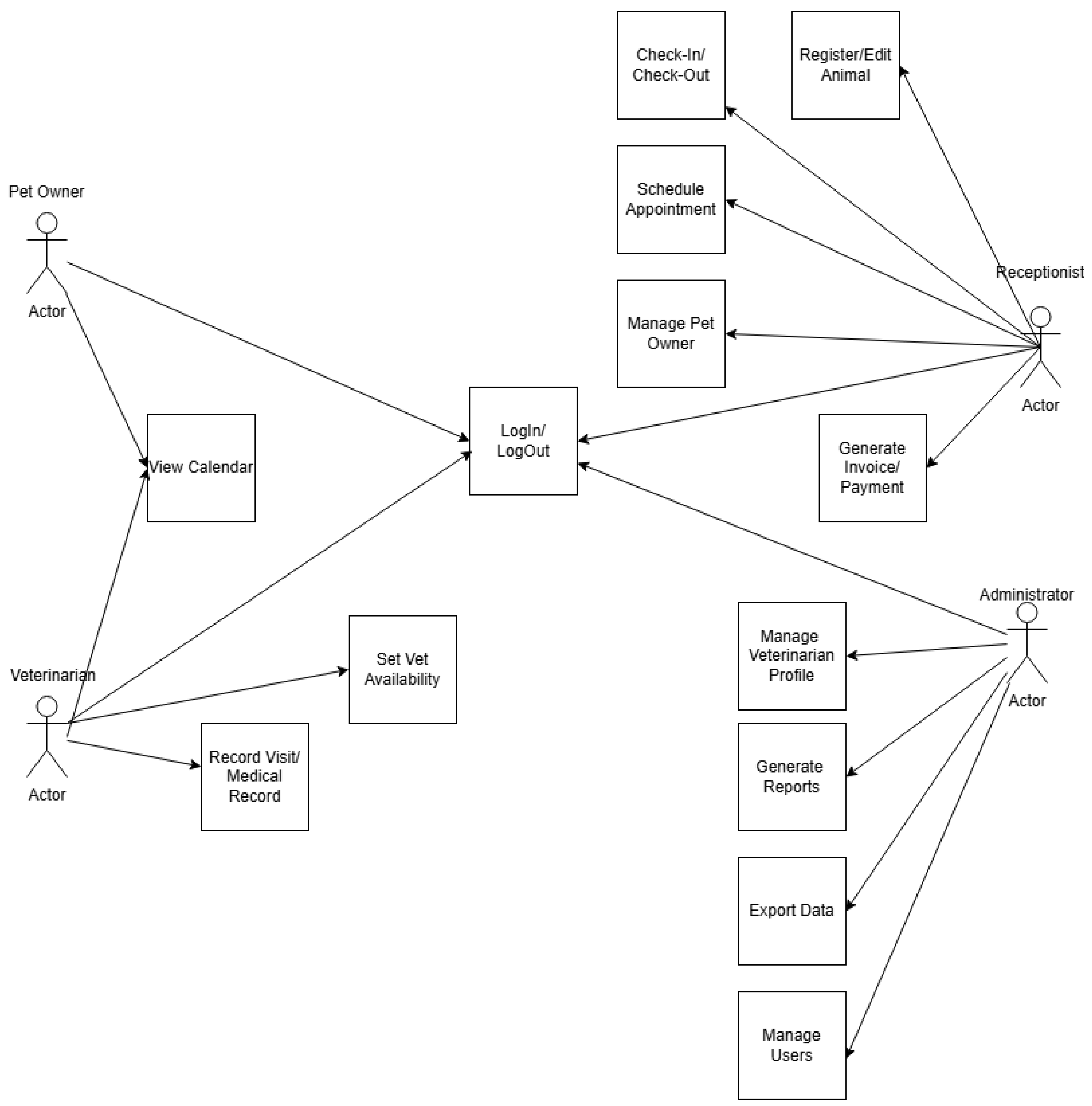
Description:
This diagram identifies four main actors, Administrator**,** Receptionist**,** Veterinarian, and Pet Owner**,** each responsible for different parts of the clinic's operations.

- The Administrator manages users, veterinarian profiles, reports, and data export.
- The Receptionist handles client-facing tasks such as registering or editing animal records, scheduling appointments, managing owners, and generating invoices or payments.
- The Veterinarian records medical visits, updates availability, and views the appointment calendar.
- The Pet Owner accesses the system to view appointment details and veterinarian availability through the calendar.

Shared Functionality:

All actors interact with the Login/Logout use case, ensuring secure authentication before accessing system features. This shared use case maintains role-based access control throughout the application.

Interpretation:
The diagram demonstrates how the system integrates multiple user roles into one cohesive environment where data flows seamlessly between administrative, clinical, and client activities. It clearly shows that each actor has distinct yet interconnected responsibilities, supporting efficient clinic management.


Check-In/ Check-Out
Register/Edit Animal
Pet Owner
Schedule Appointment
Receptionist
Actor
Manage Pet Owner
Actor
LogIn/ LogOut
View Calendar
Generate Invoice/ Payment
Administrator
Manage Veterinarian Profile
Actor
Veterinarian
Set Vet Availability
Record Visit/ Medical Record
Actor
Generate Reports
Export Data
Manage Users

3.1.2 Specific Use Case Diagram (focused view)

Actors and access

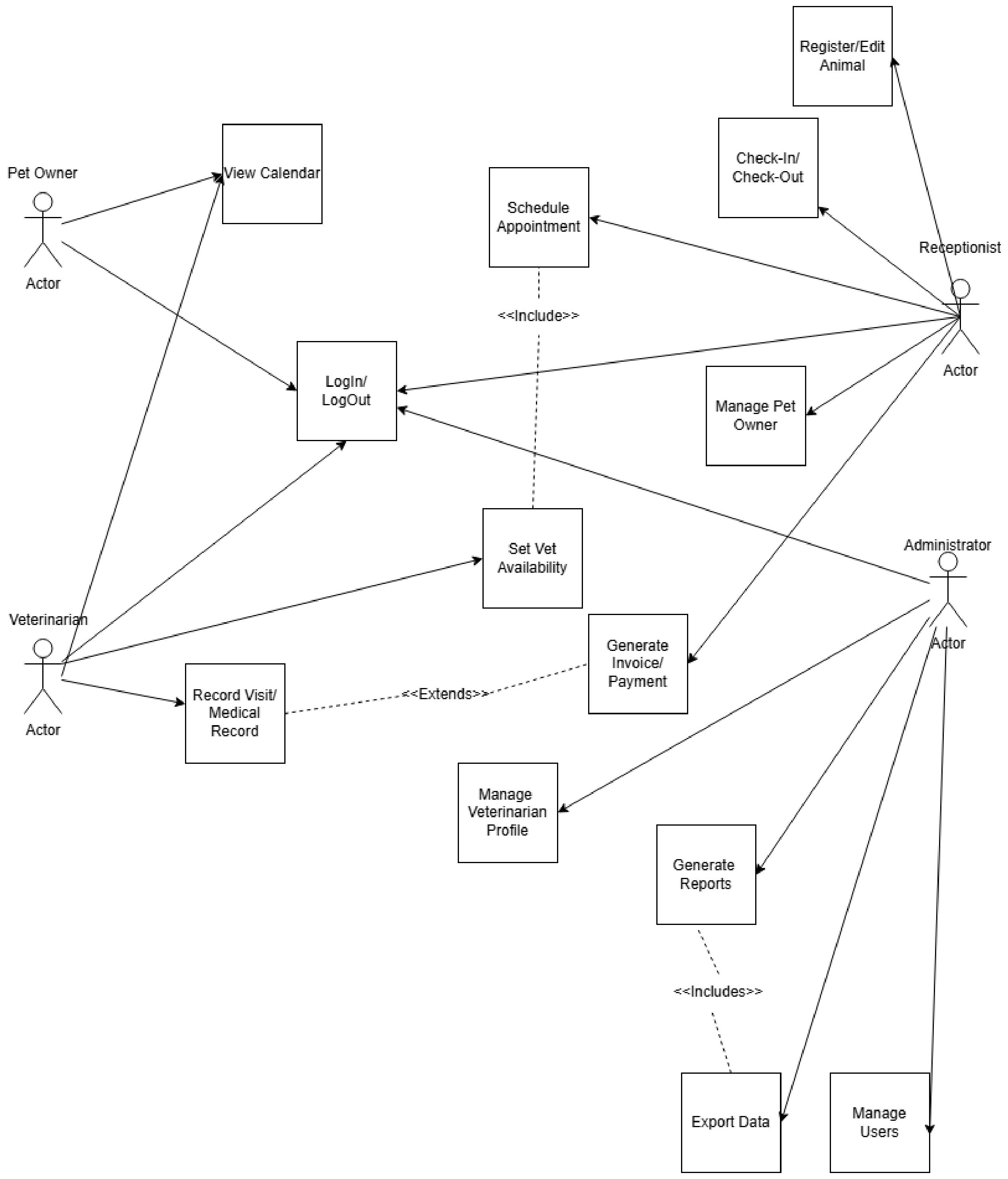
- **Receptionist:** Schedule Appointment, Register/Edit Animal, Check-In/Check-Out, Manage Pet Owner, Generate Invoice/Payment.
- **Veterinarian:** Set Vet Availability, Record Visit/Medical Record, View Calendar, Login/Logout.
- **Pet Owner:** View Calendar, Login/Logout.
- **Administrator:** Manage Veterinarian Profile, Generate Reports, Export Data (via Reports), Manage Users, Login/Logout.

Key relationships shown

- Schedule Appointment includes Set Vet Availability
  - The system must consult availability while booking.
- Record Visit/Medical Record extends Generate Invoice/Payment
  - Billing is optionally triggered as part of documenting a visit.
- Generate Reports includes Export Data
  - Reporting depends on export capability.
- Login/Logout is not an include; it is a separate common use case directly associated with all actors.

Interpretation

- Only the Receptionist books appointments in this diagram. Owners do not book; they only view the calendar.
- Veterinarians control capacity through Set Vet Availability, which the booking flow must use.
- After a visit is recorded, invoicing can occur as an extension, keeping billing optional and context-driven.
- Admin functions stay orthogonal: user management, vet profile admin, and reporting/export.


Register/Edit Animal
Check-In/ Check-Out
Pet Owner
View Calendar
Schedule Appointment
Receptionist
Actor
<<Include>>
LogIn/ LogOut
Manage Pet Owner
Set Vet Availability
Administrator
Veterinarian
Generate Invoice/ Payment
<<Extends>>
Record Visit/ Medical Record
Manage Veterinarian Profile
Generate Reports
<<Includes>>
Export Data
Manage Users

3.2 Class Diagram

Purpose:

The class diagram defines the static structure of the Animal-Care Veterinary Clinic Management System.

It identifies the main classes, their attributes, methods, and how they interact through relationships.

This diagram forms the backbone of the system's database and application logic, showing how data and functionality are organized.

Description:

The system is composed of several interconnected classes that represent the clinic's core entities:

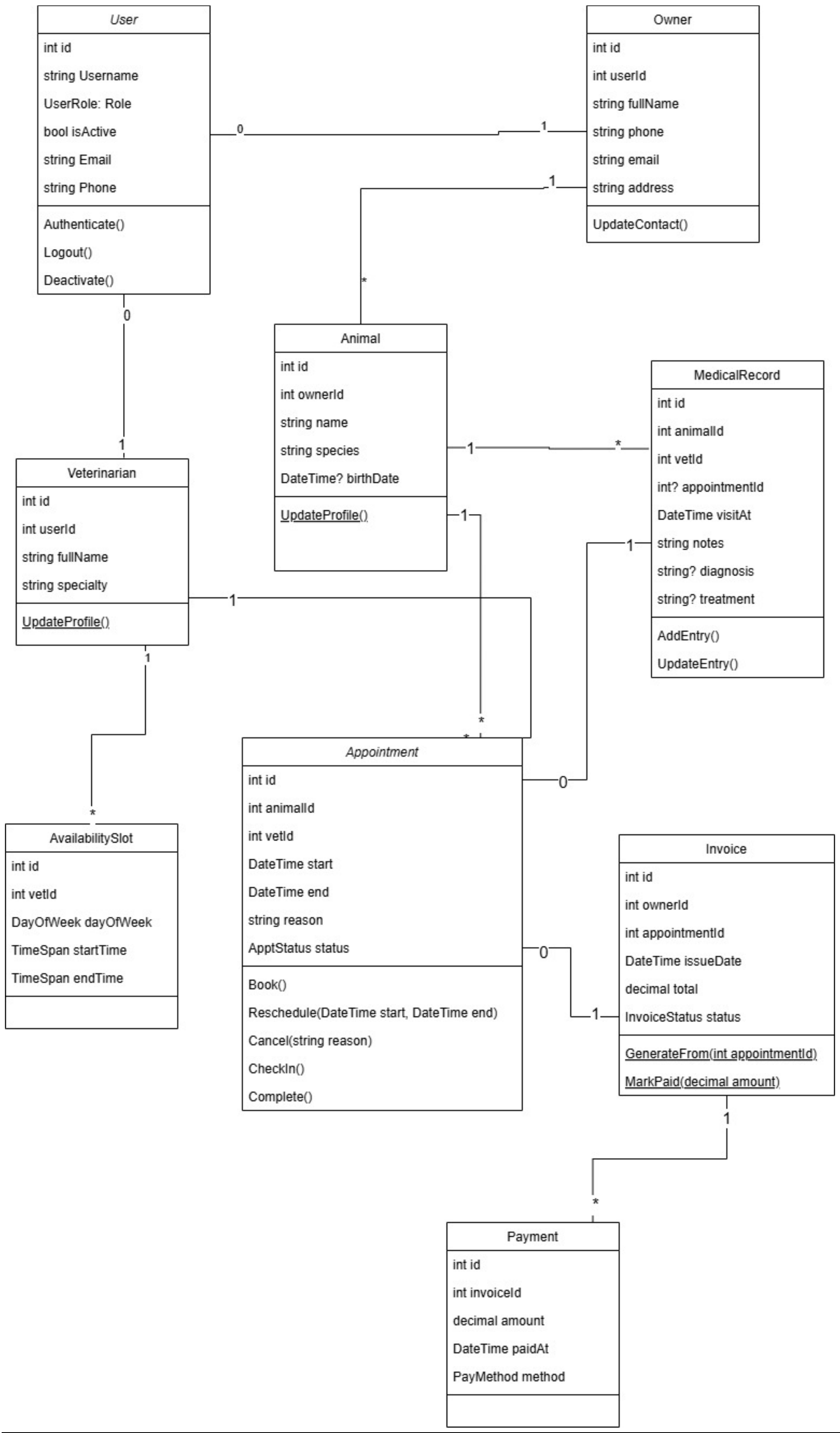
- **User:** Handles authentication and role management. Each user can log in, log out, or deactivate their account.
    - Roles include administrators, veterinarians, receptionists, and owners.
- **Owner:** Represents pet owners with their contact details and a link to the system's user record when applicable.
    - Each owner can register and manage multiple animals.
- **Veterinarian:** Stores details about each veterinarian, including their name, specialty, and linked user profile.
    - Veterinarians can update their profile and define their work availability.
- **Animal:** Stores information about animals such as name, species, and birth date.
    - Each animal belongs to one owner and can have multiple appointments and medical records.
- **Appointment:** Connects animals and veterinarians.
    - It includes start and end times, appointment status, and actions such as booking, rescheduling, cancelling, check-in, and completion.
- **Availability Slot:** Defines the weekly availability of veterinarians.
    - Each slot records the day of the week and start/end time for appointments.
- **Medical Record:** Keeps details of veterinary visits, diagnoses, and treatments.
    - It is linked to both an animal and a veterinarian and can optionally be tied to a specific appointment.
- **Invoice:** Created after an appointment to handle billing.
    - It records the appointment, owner, date, total amount, and payment status.
    - Includes methods to generate invoices and mark them as paid.
- **Payment:** Represents payments made for invoices, including amount, date, and payment method.

Relationships:

- One User can be linked to one Owner or one Veterinarian.
- One Owner can have many Animals.
- One Veterinarian can have many Appointments and Availability Slots.
- One Animal can have many Appointments and Medical Records.
- Each Appointment can generate one Invoice, and each Invoice can have multiple Payments.
- Each Medical Record may optionally link to one Appointment.

Interpretation:

This class diagram accurately models the information flow and data hierarchy of the clinic system.
It ensures clear relationships between people, animals, appointments, and financial operations.
The structure is modular, making future extensions, such as adding reporting or notifications, easy to integrate without breaking existing functionality.


User
int id
string Username
UserRole: Role
bool isActive
string Email
string Phone
Authenticate()
Logout()
Deactivate()
Owner
int id
int userId
string fullName
string phone
string email
string address
UpdateContact()
Animal
int id
int ownerId
string name
string species
DateTime? birthDate
UpdateProfile()
MedicalRecord
int id
int animalId
int vetId
int? appointmentId
DateTime visitAt
string notes
string? diagnosis
string? treatment
AddEntry()
UpdateEntry()
Veterinarian
int id
int userId
string fullName
string specialty
UpdateProfile()
AvailabilitySlot
int id
int vetId
DayOfWeek dayOfWeek
TimeSpan startTime
TimeSpan endTime
Appointment
int id
int animalId
int vetId
DateTime start
DateTime end
string reason
ApptStatus status
Book()
Reschedule(DateTime start, DateTime end)
Cancel(string reason)
CheckIn()
Complete()
Invoice
int id
int ownerId
int appointmentId
DateTime issueDate
decimal total
InvoiceStatus status
GenerateFrom(int appointmentId)
MarkPaid(decimal amount)
Payment
int id
int invoiceId
decimal amount
DateTime paidAt
PayMethod method
0
1
1
*
0
1
1
*
1
*
1
1
*
1
*
*
0
0
1
1
*

3.3 Activity Diagrams

3.3.1 Activity Diagram 1: Check-In, Visit, Check-Out

Purpose: Models the end-to-end flow on visit day: reception check-in, vet consultation, billing, and check-out.

Swimlanes: Receptionist, Veterinarian, System, Pet Owner.

Preconditions:

- Owner and animal exist.
- Appointment may already exist for today. Walk-ins allowed.

Main flow:

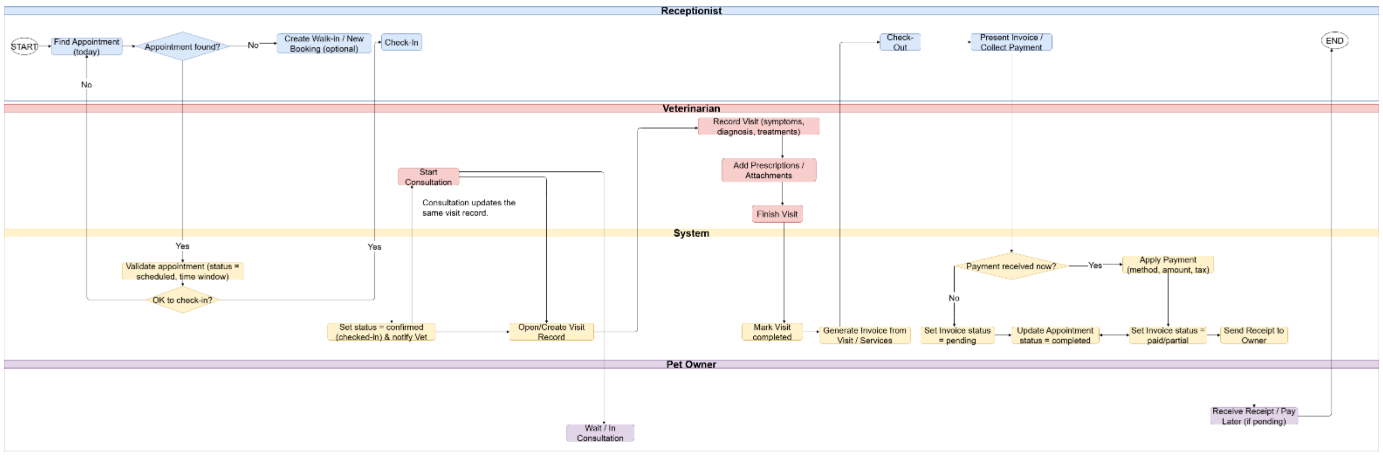
1. Receptionist**:** Find Appointment (today).
2. Decision: Appointment found?
    a. No → Create Walk-in / New Booking (optional) → Check-In.
    b. Yes → Validate appointment (status = scheduled, within time window) → Decision: OK to check-in?
        i. Yes → Check-In.
3. System**:** set appointment status = confirmed (checked-in) and notify vet; Open/Create Visit Record.
4. Veterinarian: Start Consultation → Record Visit (symptoms, diagnosis, treatments) → Add Prescriptions/Attachments → Finish Visit.
5. System: Mark Visit completed → Generate Invoice from Visit/Services.
6. Decision**:** Payment received now?
    a. Yes → Apply Payment (method, amount, tax) → set invoice status = paid/partial → Send Receipt to Owner.
    b. No → set invoice status = pending → Update Appointment status = completed → Send Receipt to Owner (or pay later).
7. Receptionist: Check-Out → Present Invoice / Collect Payment (if pending).
8. **Pet** Owner**:** Wait / In Consultation → Receive Receipt / Pay Later (if pending).

Postconditions:

- Appointment ends in completed.
- Visit record saved.
- Invoice created; payment recorded or marked pending.

Traceability:

- Matches use cases: Check-In/Check-Out, Record Visit/Medical Record, Generate Invoice/Payment.
- Uses classes: Appointment (status transitions), MedicalRecord, Invoice, Payment.


Receptionist
START
Find Appointment (today)
Appointment found?
No
Create Walk-In / New Booking (optional)
Check-In
No
Check-Out
Present Invoice / Collect Payment
END
Veterinarian
Record Visit (symptoms, diagnosis, treatments)
Add Prescriptions / Attachments
Start Consultation
Consultation updates the same visit record.
Finish Visit
System
Yes
Yes
Validate appointment (status = scheduled, time window)
OK to check-in?
Set status = confirmed (checked-in) & notify Vet
Open/Create Visit Record
Payment received now?
Yes
Apply Payment (method, amount, tax)
No
Mark Visit completed
Generate Invoice from Visit / Services
Set Invoice status = pending
Update Appointment status = completed
Set Invoice status = paid/partial
Send Receipt to Owner
Pet Owner
Wait / In Consultation
Receive Receipt / Pay Later (if pending)

3.3.2 Activity Diagram 2: Register / Edit Animal

Purpose:

Create or update an animal profile linked to an owner.

Swimlanes:

Receptionist, System, Pet Owner.

Preconditions:

Receptionist is authenticated.

Main flow:

1. Receptionist searches owner by phone/email.
2. Decision: Owner found?
   a. No → create owner profile → go to form.
   b. Yes → open Register/Edit Animal form.
3. Decision: New or Editing?
   a. New → enter animal details (name, species, breed, sex, DOB, weight, microchip, notes).
   b. Editing → select existing animal → update fields.

4. Optionally upload photos/documents.
5. Save / Submit.

System validations:

- Validate owner link.
- All required fields valid? If **No** → return error and highlight fields.
- Animal already exists for owner? If **Yes** → return duplicate error.
- Microchip unique? If **No** → return duplicate-chip error.

On success:

- Create/Update animal record.
- Store attachments with secure path/ACL.
- Return confirmation.

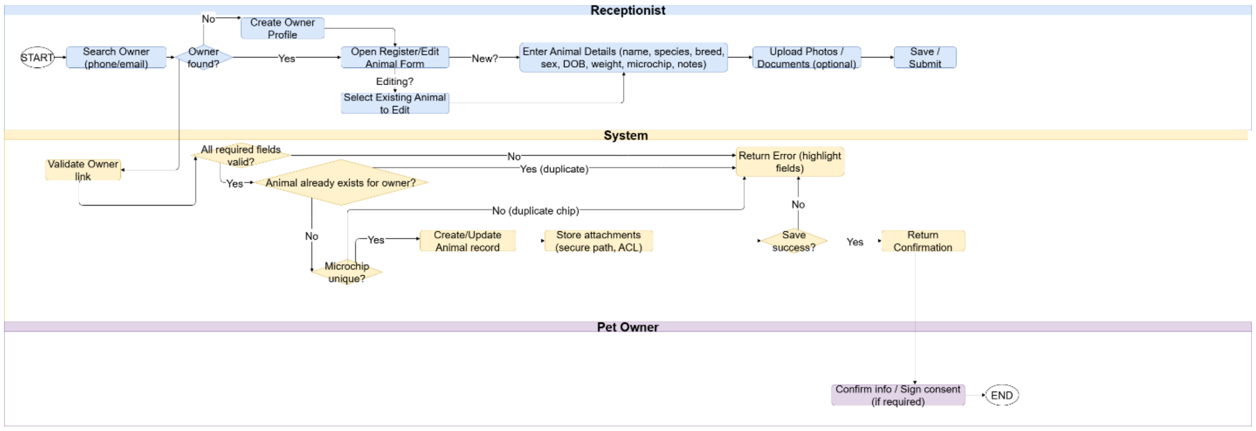
Pet owner action (if policy requires):

Confirm info / sign consent.

Postconditions:

- Animal profile is saved and linked to the owner; attachments stored.
- Errors are surfaced immediately when duplicates or invalid data are detected.

Traceability: Covers use cases Register/Edit Animal and Manage Pet Owner; uses classes Owner, Animal.


Receptionist
START
Search Owner (phone/email)
Owner found?
No
Create Owner Profile
Yes
Open Register/Edit Animal Form
New?
Editing?
Select Existing Animal to Edit
Enter Animal Details (name, species, breed, sex, DOB, weight, microchip, notes)
Upload Photos / Documents (optional)
Save / Submit
System
Validate Owner link
All required fields valid?
No
Yes
Animal already exists for owner?
Yes (duplicate)
No
Microchip unique?
No (duplicate chip)
Yes
Create/Update Animal record
Store attachments (secure path, ACL)
Return Error (highlight fields)
No
Save success?
Yes
Return Confirmation
Pet Owner
Confirm info / Sign consent (if required)
END

3.3.3 Activity Diagram 3: Set Veterinarian Availability

Purpose:

Models how a veterinarian defines or updates their working hours, breaks, and exceptions in the clinic system. It ensures that available time slots respect clinic opening hours and avoid conflicts with existing appointments.

Swimlanes:

Veterinarian, System, Administrator.

Preconditions:

Veterinarian is logged in and has permission to modify their own schedule.

Main flow:

1. Veterinarian logs in → opens Availability & Calendar.
2. Chooses mode: Weekly Pattern or Single Day.
3. Sets working hours (e.g., 09:00–17:00).
4. Adds breaks (lunch, surgery block) and exceptions (vacations, training).
5. Defines buffer time between appointments.
6. Clicks Save Availability.
7. System checks if schedule is within clinic opening hours.
8. Decision: Conflicts with existing appointments?
    a. Yes → system may suggest alternatives / auto-adjust.
        i. If accepted → Apply suggested changes.
        ii. If rejected → show error (outside hours or conflict unresolved).
    b. No → proceed.
9. System generates bookable time slots, stores availability (with recurrence and exceptions), and notifies reception to update calendars.
10. System writes audit log (who, when, what) and returns confirmation.
11. Administrator may adjust clinic-wide settings (opening hours, holidays) when required.

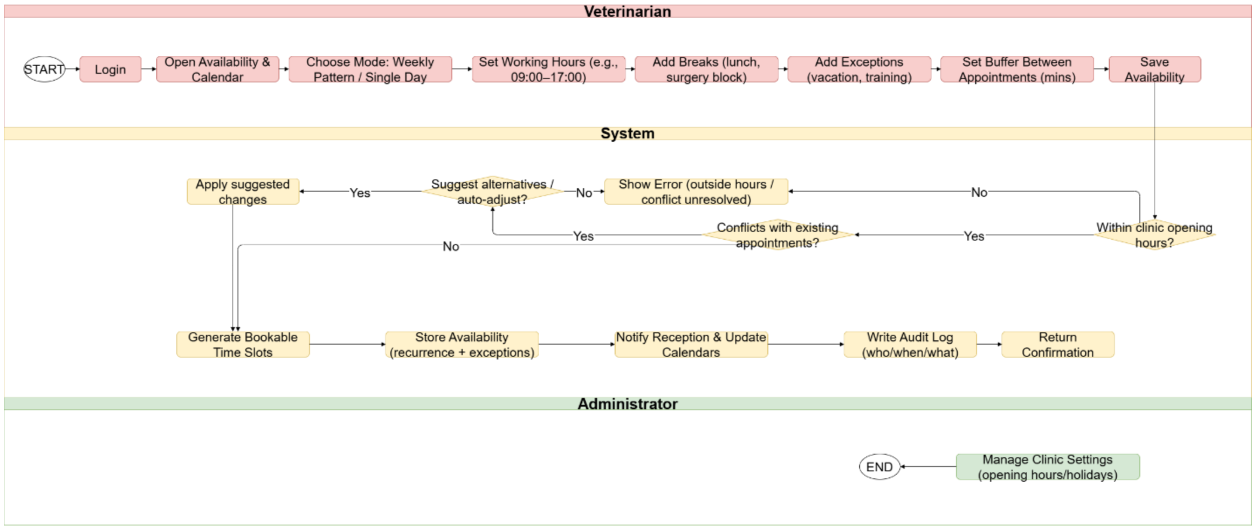
Postconditions:

- Updated schedule is stored in the system.
- Reception and system calendars reflect new bookable time slots.
- Audit record created for traceability.

Traceability:

Relates to the “Set Vet Availability” and “View Calendar” use cases.

Uses classes Veterinarian, AvailabilitySlot, and Appointment.


Veterinarian
START
Login
Open Availability & Calendar
Choose Mode: Weekly Pattern / Single Day
Set Working Hours (e.g., 09:00–17:00)
Add Breaks (lunch, surgery block)
Add Exceptions (vacation, training)
Set Buffer Between Appointments (mins)
Save Availability
System
Apply suggested changes
Yes
Suggest alternatives / auto-adjust?
No
Show Error (outside hours / conflict unresolved)
No
Yes
Conflicts with existing appointments?
Yes
Within clinic opening hours?
No
Generate Bookable Time Slots
Store Availability (recurrence + exceptions)
Notify Reception & Update Calendars
Write Audit Log (who/when/what)
Return Confirmation
Administrator
END
Manage Clinic Settings (opening hours/holidays)

3.3.4 Activity Diagram 4: Schedule Appointment

Purpose:

Shows how a receptionist books an appointment for a pet owner while the system validates schedules, clinic hours, and veterinarian availability.

Swimlanes:

Pet Owner, Receptionist, System, Veterinarian.

Preconditions:

The receptionist and system users are logged in. The veterinarian's availability is already configured.

Main flow:

1. Pet Owner logs in or views the calendar, selects an animal and the required service, and optionally picks a preferred veterinarian and date/time.
2. Receptionist logs in and searches for the owner.
    a. Decision: Owner found?
        i. Yes → continue to choose service and veterinarian.

ii. No → create new owner and register animal, then proceed.

3. Receptionist views the selected veterinarian's availability.
4. System computes appointment duration and filters available veterinarians.
5. Decision: Clinic open?
   a. No → show message "Clinic closed."
   b. Yes → generate available time slots.
6. Decision: Slot free?
   a. No → suggest other available slots.
   b. Yes → create appointment (status = scheduled).
7. System notifies both veterinarian and owner, logs the action to the audit trail, and returns confirmation.
8. Veterinarian views the updated calendar reflecting the new appointment.

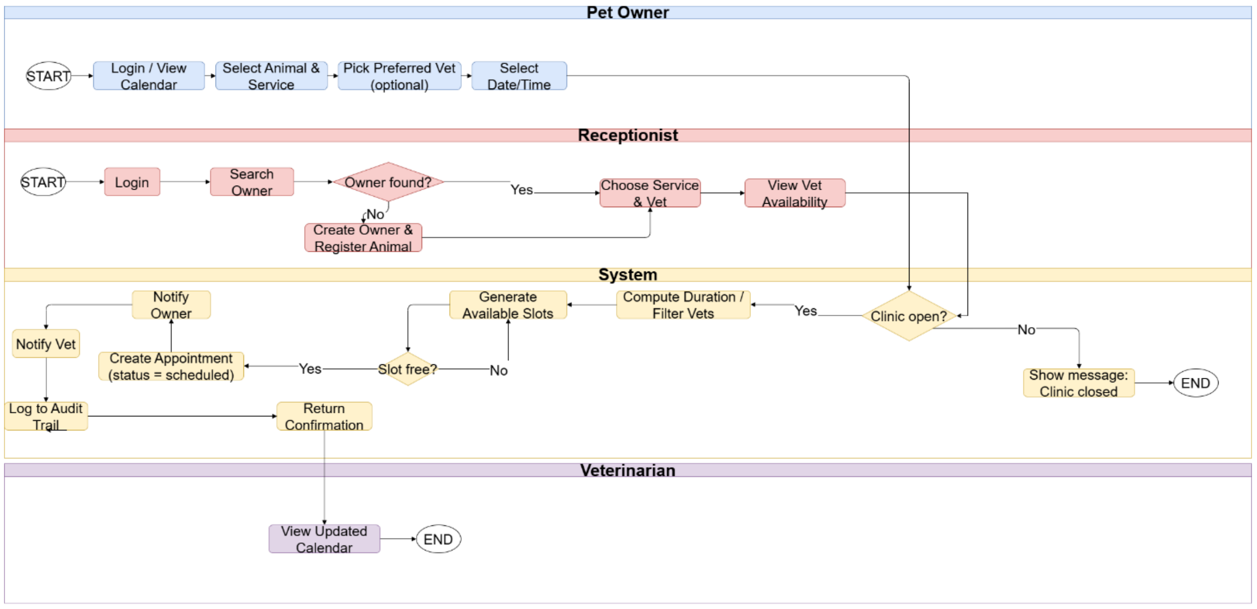
Postconditions:

- A new appointment is successfully created with a confirmed schedule.
- The veterinarian and owner receive automatic notifications.
- Audit logs capture booking details for traceability.

Traceability:

Relates to the Schedule Appointment and View Calendar use cases.

Uses classes Appointment, Animal, Veterinarian, and Owner.



3.4 Sequence Diagrams

3.4.1 Sequence Diagram 1: Schedule Appointment

Purpose:

Show the runtime interaction to book an appointment.

Lifelines:

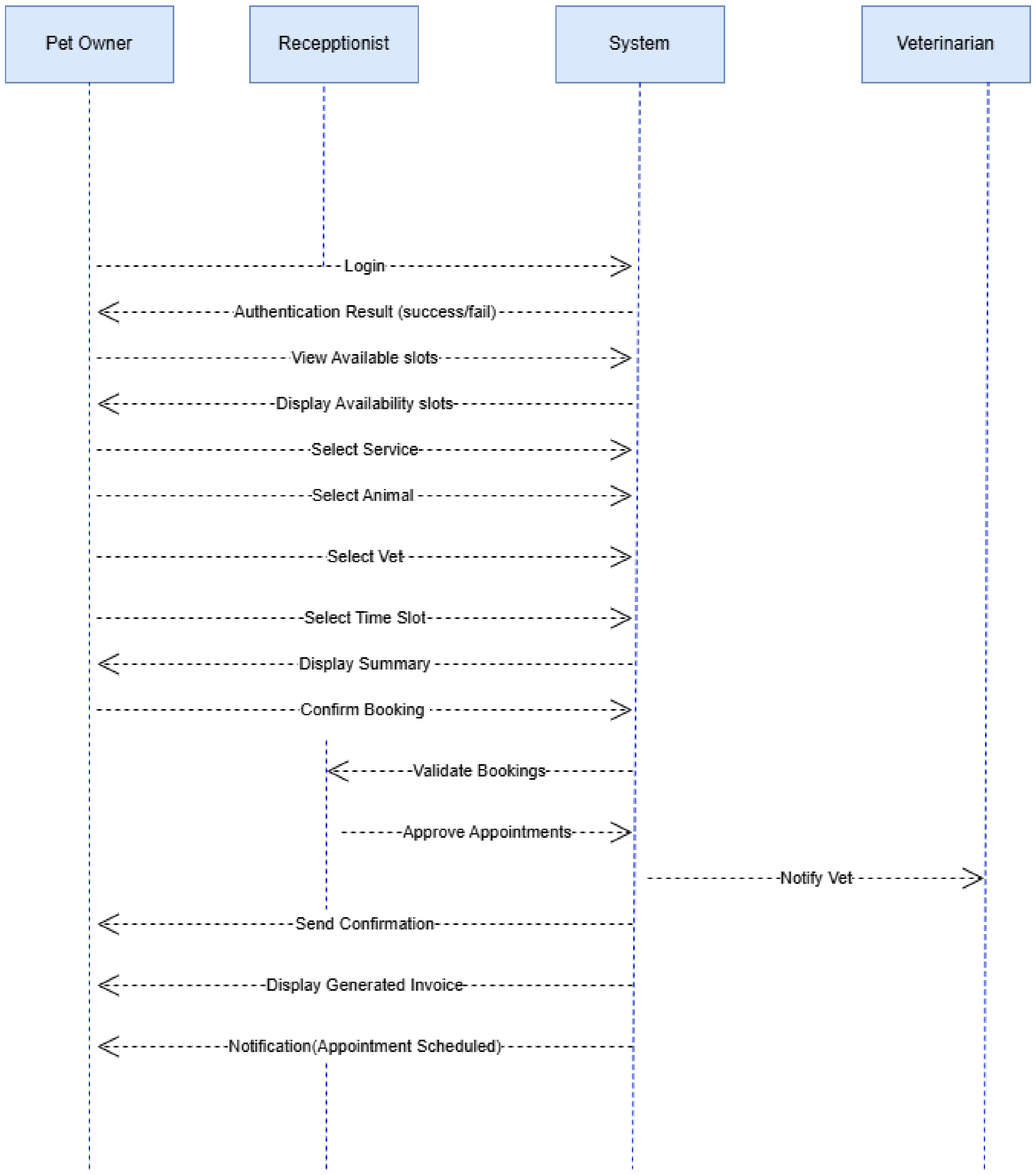
Pet Owner, Receptionist, System, Veterinarian.

Flow:

1. Receptionist → System: Login.
   a. System → Receptionist: Authentication result (success/fail).
2. Receptionist → System: View available slots.
   a. System → Receptionist: Display availability slots.
3. Receptionist → System: Select service.
4. Receptionist → System: Select animal.
5. Receptionist → System: Select vet.
6. Receptionist → System: Select time slot.
   a. System → Receptionist: Display summary.
7. Receptionist → System: Confirm booking.
   a. System → Receptionist: Validate bookings (conflict/clinic-hours checks).
   b. System → System: Approve appointment (create record, status = scheduled).
8. System → Veterinarian: Notify vet (new booking in calendar).
9. System → Receptionist: Send confirmation (booking details).
   a. System → Receptionist: Display generated invoice (if your process issues a booking invoice/deposit; otherwise treat as optional).
10. System → Pet Owner: Notification: appointment scheduled.

Postconditions:

Appointment exists with status scheduled; vet calendar updated; owner and receptionist receive confirmation.


Pet Owner
Recepptionist
System
Veterinarian
Login
Authentication Result (success/fail)
View Available slots
Display Availability slots
Select Service
Select Animal
Select Vet
Select Time Slot
Display Summary
Confirm Booking
Validate Bookings
Approve Appointments
Notify Vet
Send Confirmation
Display Generated Invoice
Notification(Appointment Scheduled)

Sequence Diagram 2: Set Veterinarian Availability

Purpose:

Shows how a veterinarian configures or updates their schedule through the system and how the system ensures proper synchronization with the reception and administrators.

Lifelines:

Veterinarian, System, Administrator.

Flow:

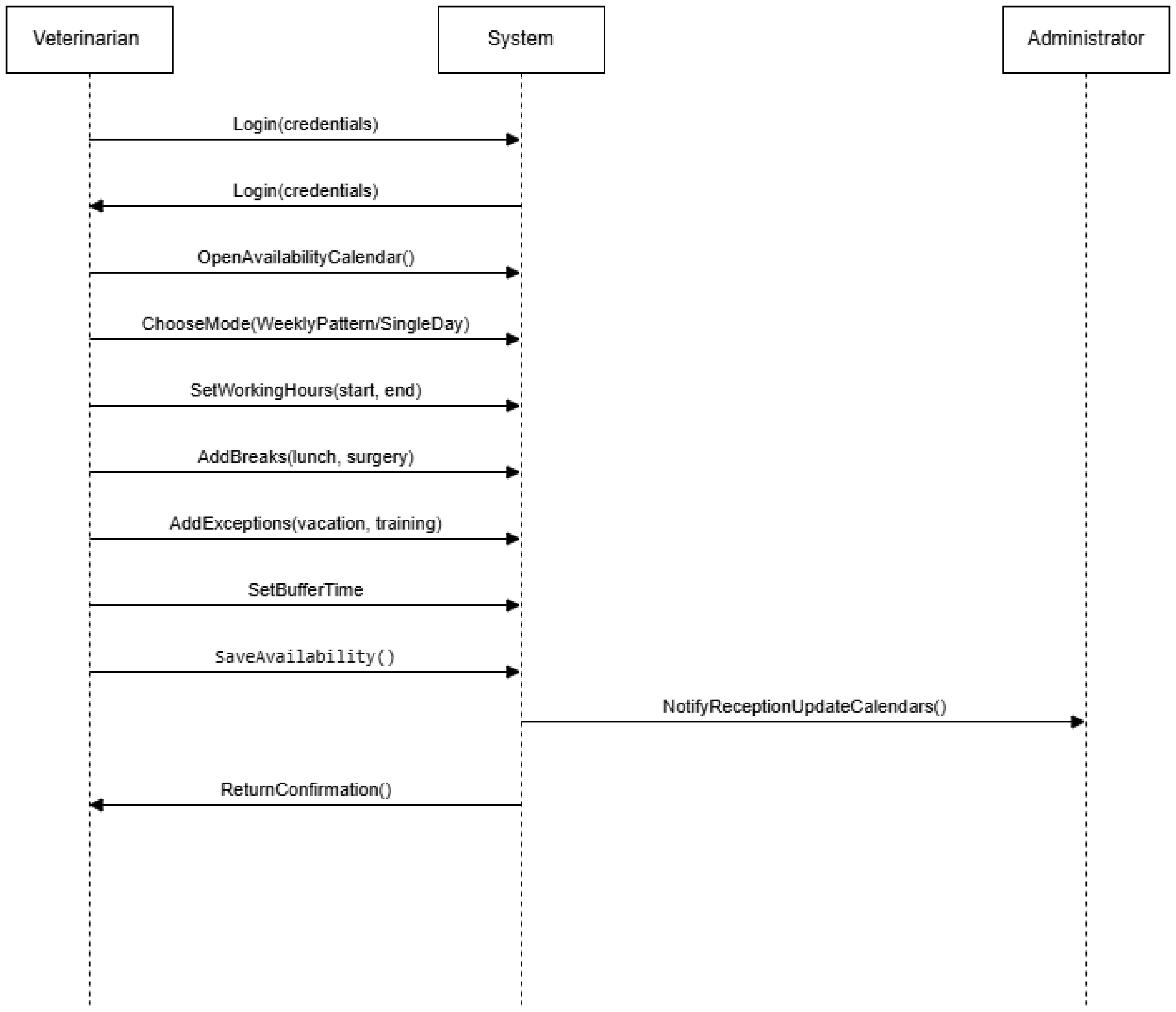
1. Veterinarian → System: Login(credentials)
    a. System → Veterinarian: Login(credentials) (confirmation).
2. Veterinarian → System: OpenAvailabilityCalendar()
3. Veterinarian → System: ChooseMode(WeeklyPattern / SingleDay)
4. Veterinarian → System: SetWorkingHours(start, end)
5. Veterinarian → System: AddBreaks(lunch, surgery)
6. Veterinarian → System: AddExceptions(vacation, training)
7. Veterinarian → System: SetBufferTime()
8. Veterinarian → System: SaveAvailability()
    a. System → Administrator: NotifyReceptionUpdateCalendars() (to refresh visible schedule and reception booking options).
9. System → Veterinarian: ReturnConfirmation() (successful save).

Postconditions:

- The new schedule is stored and reflected across reception and admin calendars.
- Any changes are logged and visible for appointment scheduling.

Traceability:

Relates to the Set Vet Availability use case and AvailabilitySlot class.


Veterinarian
System
Administrator
Login(credentials)
Login(credentials)
OpenAvailabilityCalendar()
ChooseMode(WeeklyPattern/SingleDay)
SetWorkingHours(start, end)
AddBreaks(lunch, surgery)
AddExceptions(vacation, training)
SetBufferTime
SaveAvailability()
NotifyReceptionUpdateCalendars()
ReturnConfirmation()

Sequence Diagram 3: Check-In / Check-Out Process

Purpose:

Shows how a patient visit proceeds from check-in through consultation to payment and checkout.

Lifelines:

Veterinarian, System, Receptionist, Pet Owner.

Flow:

1. Receptionist → System: FindAppointment(today)
    a. System → Receptionist: AppointmentFound? / NotFound()
2. Receptionist → System: CheckIn()
    a. System → Veterinarian: NotifyVet() (vet is informed that the client has arrived).
3. Veterinarian → System:
    a. StartConsultation()
    b. RecordVisit(symptoms, diagnosis, treatment)
    c. AddPrescriptionsOrAttachments()
    d. FinishVisit()
4. System → Receptionist: PresentInvoice()
    a. Pet Owner → Receptionist: MakePayment(method, amount)
    b. Receptionist → System: UpdateInvoiceStatus(paid / partial)
5. System → Pet Owner: SendReceipt()
6. Receptionist → System: CheckOut()

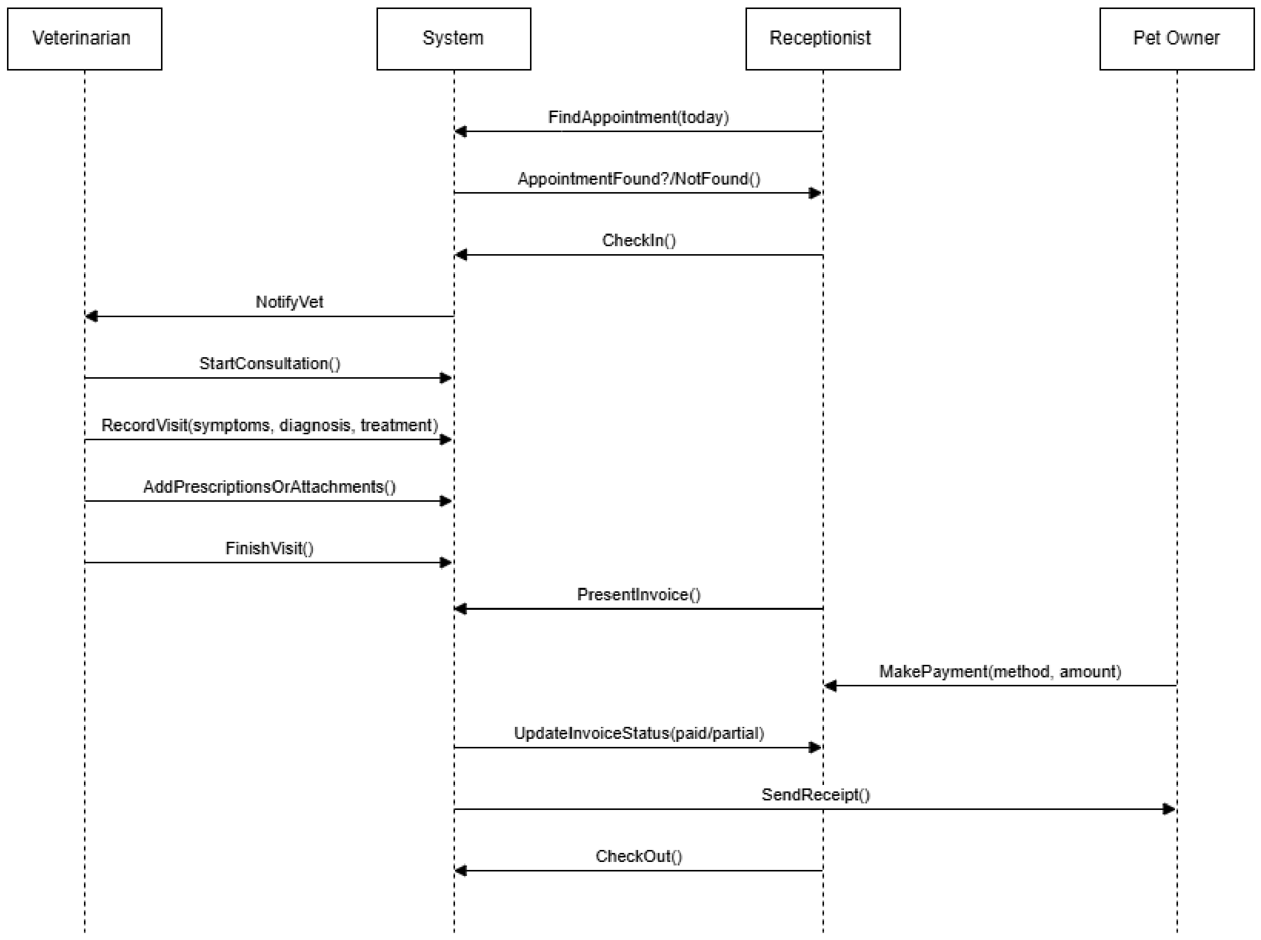
Postconditions:

- Appointment status becomes completed.
- Medical record, invoice, and payment entries are stored.
- Owner receives a receipt; veterinarian's log reflects the finished visit.

Traceability:

Relates to the Check-In/Check-Out, Record Visit/Medical Record, and Generate Invoice/Payment use cases.

Uses Appointment, MedicalRecord, Invoice, and Payment classes.


Veterinarian
System
Receptionist
Pet Owner
FindAppointment(today)
AppointmentFound?/NotFound()
CheckIn()
NotifyVet
StartConsultation()
RecordVisit(symptoms, diagnosis, treatment)
AddPrescriptionsOrAttachments()
FinishVisit()
PresentInvoice()
MakePayment(method, amount)
UpdateInvoiceStatus(paid/partial)
SendReceipt()
CheckOut()

Sequence Diagram 4: Register / Edit Animal

Purpose:

Capture the runtime steps to create or update an animal profile and link it to an owner.

Lifelines:

System, Receptionist, Pet Owner.

Flow:

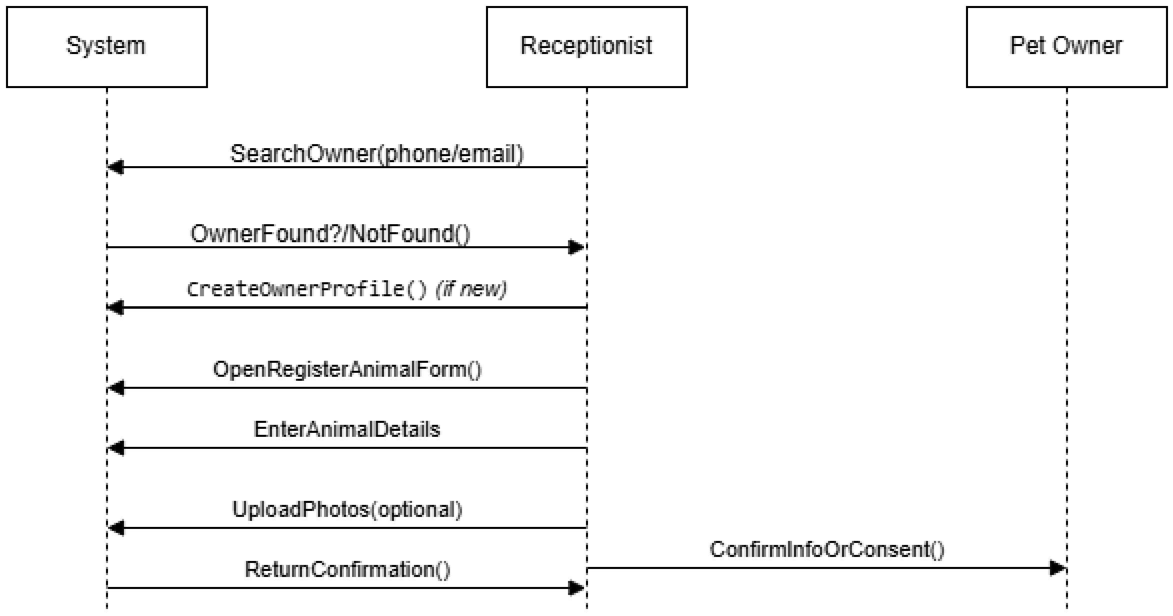
1. Receptionist → System: SearchOwner(phone/email)
    a. System → Receptionist: OwnerFound? / NotFound()
2. If not found: Receptionist → System: CreateOwnerProfile()
3. Receptionist → System: OpenRegisterAnimalForm()
4. Receptionist → System: EnterAnimalDetails (name, species, breed, DOB, microchip, notes)
5. Receptionist → System: UploadPhotos(optional)
6. System → Receptionist: ReturnConfirmation() (record saved)
7. Receptionist → Pet Owner: ConfirmInfoOrConsent() (if clinic policy requires)

Postconditions:

- Animal record exists and is linked to the owner.
- Optional attachments stored.
- Confirmation provided to the owner.

Traceability:

- Maps to use cases Register/Edit Animal and Manage Pet Owner.
- Uses classes Owner and Animal.


System
Receptionist
Pet Owner
SearchOwner(phone/email)
OwnerFound?/NotFound()
CreateOwnerProfile() (if new)
OpenRegisterAnimalForm()
EnterAnimalDetails
UploadPhotos(optional)
ConfirmInfoOrConsent()
ReturnConfirmation()