PETCARE
HOME OF PETS

By Adam Kanj and Ahmad Malak

Streamlined pet care management for a purrfect pet parenthood journey.

1
Appointment Scheduling

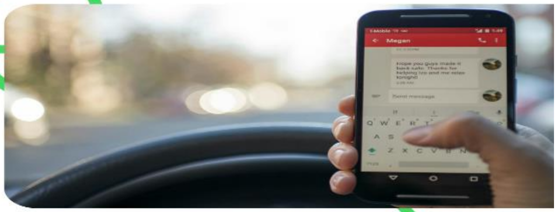
Effortless booking and management of pet consultations, ensuring timely and convenient care.

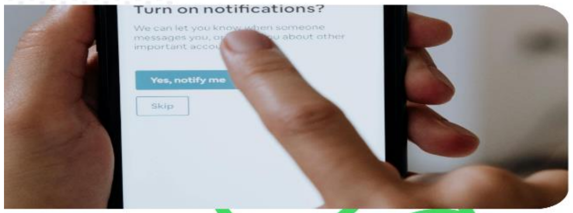
2
Medical Records

Secure storage and easy access to your pet's medical history, vaccinations, and treatment details.

3
Communication Gateway

Direct messaging with your veterinarian for any queries, followup questions, and medical advice.





4
Reminder Notifications

Stay on top of upcoming appointments, medication schedules, and preventive care routines.

5
Health Monitoring

Keep tabs on your pet's well-being by recording observations, symptoms, and behavioral changes.

Simplified Pet Care

PetCare empowers responsible pet ownership with organized medical records, accessible vet consultations, and user-friendly appointment management.

LAU
School of Arts and Sciences

Department of Computer Science & Mathematics

APPLICATION DESIGN FOR A VET APPLICATION

PETCARE
HOME OF PETS

By

Adam Kanj (202105510)

Ahmad Malak (202105744)

REPORT

Submitted to DR. NADINE ABBAS

Course “CSC590: Capstone Project” in Computer Science

Submitted on

May 9, 2024

Table of Contents

I. Introduction

PetCare is a cutting-edge mobile application revolutionizing the landscape of pet healthcare management. Developed as an intuitive and accessible solution, PetCare aims to enhance the well-being of pets by providing seamless access to essential veterinary services and resources. Through its user-friendly interface, PetCare empowers both pet owners and veterinarians to streamline pet health management efficiently.

The application focuses on the following key objectives:

1. **Comprehensive Appointment Management:** PetCare simplifies the process of scheduling veterinary appointments, ensuring timely access to healthcare services for pets. Pet owners can easily book appointments, receive reminders, and manage their pets' healthcare schedules.
2. **Enhanced Pet Profile Management:** PetCare facilitates the creation and maintenance of detailed profiles for each pet, encompassing vital information such as medical history, allergies, medications, and vaccination records. This comprehensive approach enables pet owners to track and manage their pets' health effectively.
3. **Convenient Telemedicine Consultations:** Through PetCare, pet owners can engage in remote consultations with veterinarians for non-emergency concerns or follow-up queries. This feature promotes accessibility to veterinary care while minimizing the need for physical visits.
4. **Effortless Access to Medical Records:** PetCare grants pet owners secure access to their pets' medical records, lab results, and treatment plans. By providing a centralized platform for medical information, PetCare facilitates informed decision-making and continuity of care.
5. **Timely Medication Reminders:** PetCare offers timely reminders for medication doses, vaccinations, and follow-up appointments, ensuring adherence to treatment plans and preventive care measures.
6. **Symptom Checker Tool:** PetCare includes an integrated symptom checker tool, allowing pet owners to assess their pets' conditions and seek appropriate veterinary care promptly.

7. **Secure Online Payments:** With PetCare, pet owners can make secure online payments for veterinary services, medications, and other expenses directly through the application.
8. **Access to Health Articles and Resources:** PetCare provides access to a vast library of articles and resources on pet care, nutrition, behavior, and preventive health measures, empowering pet owners with valuable information and guidance.

By prioritizing these functionalities, PetCare aims to foster better communication, coordination, and care for pets, ultimately enhancing the overall pet healthcare experience.

II. Background

PetCare is a pet healthcare management application designed to address the diverse needs of pet owners, veterinarians, and administrative staff in the veterinary industry. With a focus on enhancing the well-being of pets and promoting efficient healthcare management, PetCare offers a range of features tailored to each stakeholder group.

For pet owners, PetCare serves as a convenient and accessible platform for managing their pets' health. Through the application, pet owners can schedule veterinary appointments, access comprehensive pet profiles containing vital medical information, and engage in telemedicine consultations with veterinarians for non-emergency concerns. Additionally, PetCare provides medication reminders, symptom checking tools, and access to a wealth of pet health articles and resources, empowering pet owners to make informed decisions about their pets' care.

Veterinarians benefit from PetCare's robust appointment and patient management features, allowing them to efficiently manage their schedules, access patient records, and conduct remote consultations with pet owners. The application facilitates prescription management, billing, and communication with pet owners, streamlining veterinary operations and improving the quality of care provided to pets.

Administrative staff within veterinary clinics benefit from PetCare's administrative functionalities, including appointment scheduling, patient registration, inventory management, and billing. These features help streamline clinic operations, reduce administrative overhead, and ensure smooth communication and coordination between staff members.

In developing PetCare, extensive research was conducted within the veterinary industry to identify the unique challenges faced by pet owners, veterinarians, and administrative staff. This research informed the design and development of PetCare's features, ensuring that the application effectively addresses the needs of all stakeholders. By offering a comprehensive solution for pet healthcare management, PetCare aims to improve the efficiency of veterinary clinics, enhance the quality of care provided to pets, and ultimately, enrich the lives of both pets and their owners.

III. User Requirements

These user requirements aim to ensure that PetCare app fulfills the needs of pet owners, veterinarians, and administrative staff, offering a comprehensive solution for efficient pet healthcare management.
The following are the user requirements of the software:

General Requirements:

- Users shall be able to log in to the app using their unique credentials.
- The app shall provide a user-friendly interface for seamless navigation.
- The app shall ensure data security and privacy of users' information.

1. Pet Owner Requirements:

- Registration: Pet owners should be able to create profiles with their personal information and pet details.
- Appointment Management: Pet owners should be able to schedule appointments with veterinarians and receive reminders.
- Pet Health Management: Pet owners should be able to maintain detailed profiles for each pet, including medical history and vaccination records.
- Communication: Pet owners should be able to engage in remote consultations with veterinarians and make secure online payments for services.
- Location Services: Pet owners should have access to an emergency services locator for urgent situations.

2. Veterinarian Requirements:

- Registration: Veterinarians should be able to create profiles with their professional information and availability.

- Appointment and Patient Management: Veterinarians should be able to manage appointment schedules, patient records, and conduct remote consultations.
- Prescription and Billing: Veterinarians should be able to prescribe medications, generate invoices, and handle billing inquiries from pet owners.
- Communication: Veterinarians should be able to communicate securely with pet owners and access health articles and resources.

3. Admin Requirements:

- Appointment and Patient Management: Admins should be able to manage appointment calendars, patient registration, and scheduling.
- Inventory and Financial Management: Admins should be able to track inventory levels, process payments, and generate reports on clinic performance.
- Customer Support: Admins should provide assistance to pet owners and veterinarians with scheduling, billing, and general inquiries.

5. File Support:

- The app shall support various file formats including Microsoft files, PDFs, image files, HL7, and DICOM.
- The maximum document size allowed shall be 50 MB.

6. Notification System:

- The app shall send notifications to users about upcoming appointments, changes in schedules, and new messages.
- Users shall have the option to choose their preferred method of receiving notifications.

7. Prescription Management:

- Veterinarians shall be able to prescribe medications to pets online.

- Pet owners shall be able to view prescribed medications and request refills.
- The app shall alert veterinarians when pet owners request medication refills.

IV. System Requirements

Functional Requirements:

Functional requirements outline specific features and functionalities that the PetCare app must possess to meet the needs of its users. These requirements define how the system should behave under different conditions and what outputs it should produce in response to specific inputs. They focus on capabilities, performance, security, and usability.

1. Authentication and User Profile Management:

- Users shall register using their phone numbers and authenticate using unique usernames and passwords.
- The app shall provide a user profile tab for managing personal information and pet details.
- Profile pictures matching user IDs shall be uploaded upon registration.
- Users shall be able to delete their profiles permanently.
- Verification documents must be updated when expiration dates are due, with users unable to access the app until updated.

2. Search and Filter:

- Users shall search for veterinarians by name and filter results by specialty, gender preference, and ratings.
- Detailed information about veterinarians' credentials and work history shall be accessible.

3. Pet and Medical History Management:

- The system shall securely store personal and medical information, including pet profiles, allergies, medications, and vaccination records.
- It shall allow users to view and update pet medical records, lab results, and vital signs.

4. Ratings and Reviews:

- Users can rate and review veterinarians using a five-star scale based on their interactions.
- The system shall calculate average ratings for each veterinarian and display them for users' reference.

5. Notifications:

- Users shall receive notifications for upcoming appointments, changes in schedules, test results, and prescription renewals.
- Various notification formats such as push notifications, emails, or SMS messages shall be supported.

6. Staff and Inventory Management:

- Veterinarians shall prescribe medications, order diagnostic tests, and schedule follow-up appointments through the app.
- Staff shall manage inventory, track equipment availability, and send maintenance requests.

7. Payment Management:

- Multiple payment options shall be available, including credit card, debit card, PayPal, and bank transfer.

- The system shall generate invoices, billing statements, and manage insurance information securely.
- It shall comply with regulations like HIPAA and GDPR for secure payment processing.

Non-Functional Requirements:

Non-functional requirements define characteristics related to the system's performance, reliability, security, scalability, maintainability, and usability.

1. Usability:

- The app shall have an intuitive interface accessible from various devices.
- It shall minimize user errors and ensure efficient workflows.

2. Security:

- Robust security measures shall protect user data and comply with privacy laws.
- Authentication and authorization features shall prevent unauthorized access.

3. Performance:

- The app shall perform well with fast load times and minimal lag, even with large data loads.
- It shall be scalable to accommodate increasing users and data volume.

4. Integration:

- The app shall integrate with other pet healthcare systems and software.
- Data import and export capabilities shall facilitate interoperability.

5. Support and Maintenance:

- Comprehensive documentation and training resources shall be provided.
- Ongoing technical support and maintenance shall address issues and ensure system updates.

V. Software Architecture Design

Software architecture refers to the structures of a software system and the discipline of creating such structures and systems. Our platform is a Flutter-based application, therefore the software architecture is defined as follows:

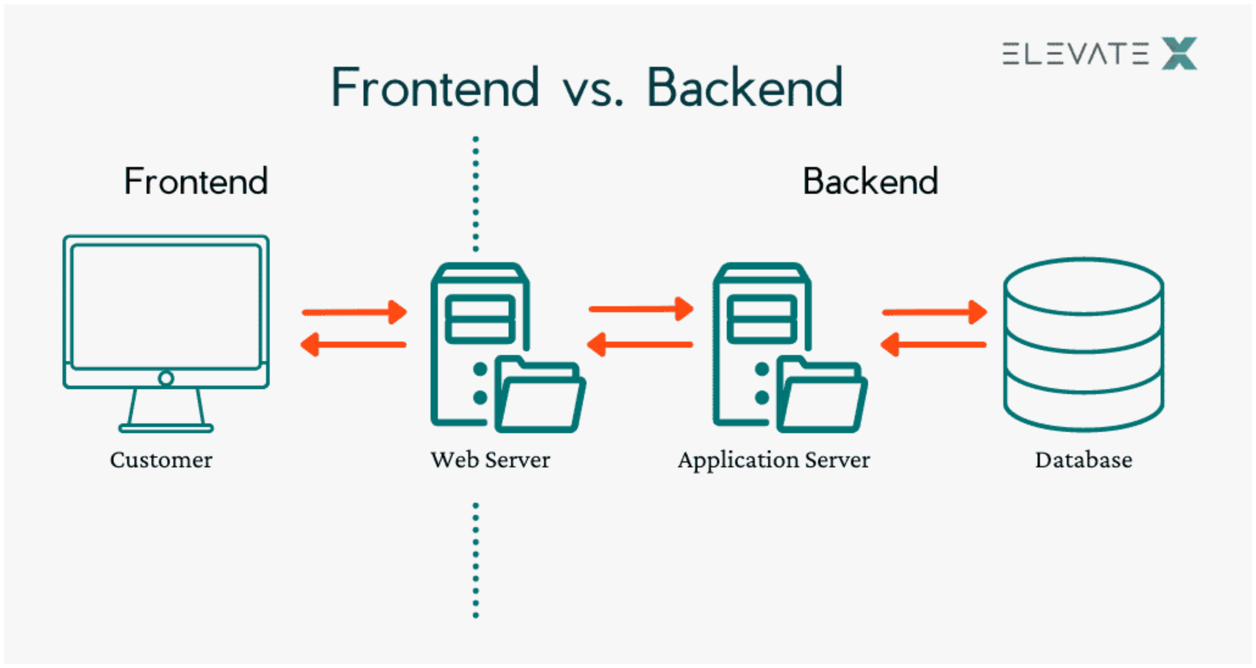


Figure 1

The program is split into two parts:

• **Front-end:** The presentation and visuals of the software. In other words, what the user sees. These are done using Dart as a programming language.

• **Back-end:** Involves all servers and databases that the software utilizes. Here, we’re mainly using ASP.NET Web API and SQL.

Since our software is flutter-based, we decided to go with the traditional 3-tier software architecture which involves the clients, server, and database. The clients can be from any device. The server is a midway communication between the clients and the database.

VI. Database Design and Implementation

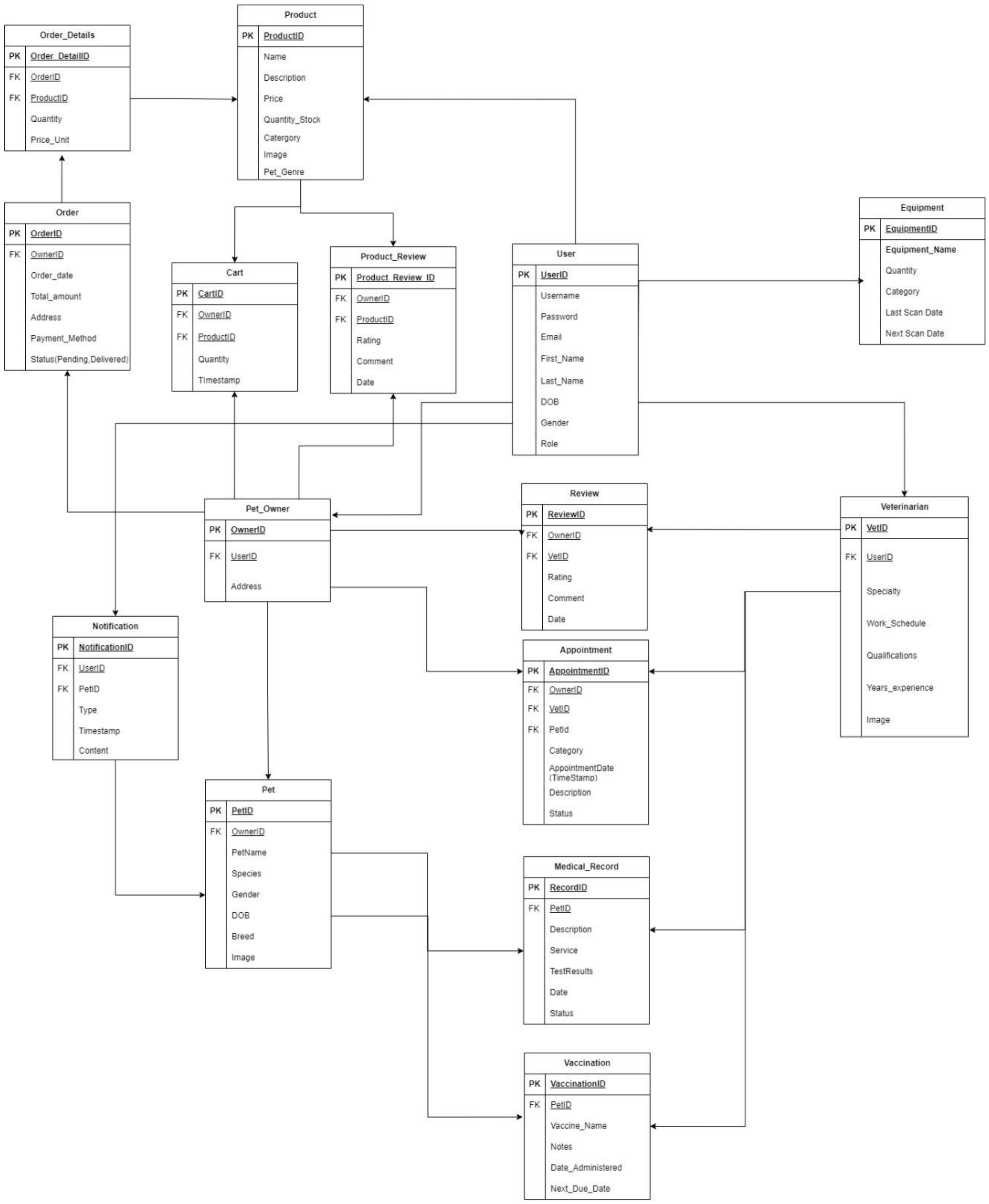


Figure 2: Database ER Diagram

1. Introduction

This is a detailed overview of the database design for a veterinary clinic app tailored for various users including pet owners, veterinarians, and administrative staff. The design is illustrated through an Entity-Relationship (ER) diagram, which encapsulates entities, relationships, and key attributes to support the app's functionalities.

2. ER Diagram Overview

The ER diagram represents the structured data model for our app. It includes entities such as Users, Pets, Products, Orders, Veterinarians, and more, linked through relationships that facilitate operations like scheduling appointments, managing medical records, and processing orders.

3. Entity Descriptions

Each entity in the diagram corresponds to a database table. Below are the entities detailed with their respective attributes:

1. **User**
   - **UserId** (PK): Unique identifier for each user.
   - **Username**: User's chosen login name.
   - **Password**: Encrypted password for account security.
   - **Email**: User's email address for communication.
   - **FirstName**: User's given name.
   - **LastName**: User's family name.
   - **DOB**: Date of birth, formatted as YYYY-MM-DD.
   - **Gender**: User's gender.
   - **Role**: Role within the app (e.g., Admin, Veterinarian, Pet Owner).
2. **Pet**
   - **PetId** (PK): Unique identifier for each pet registered on the app.
   - **OwnerId** (FK): Links to the UserId of the pet's owner.
   - **PetName**: The pet's name.

   - **Species**: The species of the pet (e.g., Dog, Cat).
   - **Gender**: The pet's gender.
   - **DOB**: The pet's date of birth.
   - **Breed**: The breed of the pet if applicable.
   - **Image**: A link to or blob of the pet's image.
3. **Product**
   - **ProductId** (PK): Unique identifier for each product.
   - **Name**: Product name.
   - **Description**: Detailed description of the product.
   - **Price**: Cost per unit.
   - **Quantity_Stock**: Number of items currently in stock.
   - **Category**: Category of the product (e.g., Food, Toy).
   - **Image**: Image of the product.
   - **Pet_Genre**: Targeted pet genre for the product.
4. **Order**
   - **OrderId** (PK): Unique identifier for each order placed.
   - **UserId** (FK): References the user who placed the order.
   - **Order_Date**: Date when the order was placed.
   - **Total_amount**: Total amount of the order.
   - **Address**: Shipping address for the order.
   - **Payment_Method**: Method of payment used.
   - **Status**: Status of the order (e.g., Pending, Delivered).

4. Relationships and Mapping

The relationships between entities are crucial for database functionality:

- **User-Pet**: A "one-to-many" relationship, as a user can own multiple pets but each pet has only one owner.
- **User-Order**: A "one-to-many" relationship, indicating a user can place multiple orders.
- **Pet-Appointment**: A "one-to-many" relationship, as a pet can have several appointments.

- **Veterinarian-Appointment**: Many-to-One. An appointment is linked to one veterinarian who can have multiple appointments.

5. Table Structure and Normalization

The tables are designed to ensure minimal redundancy and high data integrity:

- **Primary Keys** (PK) uniquely identify each record within a table.
- **Foreign Keys** (FK) create logical links between tables.
- The database is normalized to the third normal form to prevent data redundancy and ensure data integrity.

6. Data Operations (using MS SQL)

- **Inserting Data**: Example SQL for inserting a new pet:

```
SQL
INSERT INTO Pet (OwnerId, PetName, Species, Gender, DOB, Breed,
Image) VALUES (1, 'Buddy', 'Dog', 'Male', '2017-04-12', 'Golden
Retriever', '/images/buddy.jpg');
```

- **Sample Queries**: Example SQL to retrieve all products for cats:

```
SQL
SELECT * FROM Product WHERE Pet_Genre = 'Cat';
```

7. Conclusion

The database design outlined supports the critical functionalities required for the smooth operation of a veterinary clinic app. It is scalable, ensuring that as the app grows, new features and entities can be integrated without major overhauls. Future enhancements could include more detailed user profiles, integration with external medical record systems, and advanced analytics for veterinary use.

VII. Backend

Introduction

This report explores the backend functionalities of the petcare app, which is built using ASP.NET Web API. This framework is selected for its robustness, scalability, and ability to handle RESTful services efficiently. The backend serves three types of users: pet owners, veterinarians, and administrators, each with specific interaction flows and data needs.

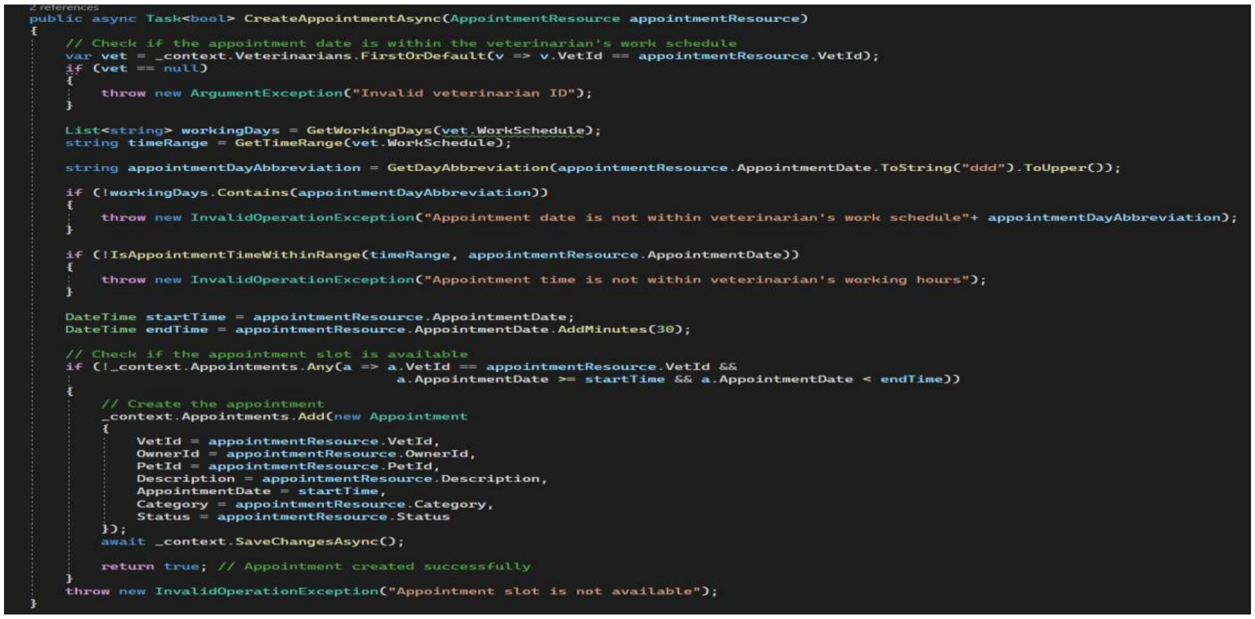
Core Functionalities

The backend, developed with ASP.NET Web API, is responsible for handling all server-side logic, database interactions, and providing endpoints for the frontend to consume.

Appointment Management

A central feature of our application, appointment management, is critical for both pet owners and veterinarians. This system manages the scheduling, updating, and cancellation of appointments.

CreateAppointmentAsync Method



```
2 references
public async Task<bool> CreateAppointmentAsync(AppointmentResource appointmentResource)
{
    // Check if the appointment date is within the veterinarian's work schedule
    var vet = _context.Veterinarians.FirstOrDefault(v => v.VetId == appointmentResource.VetId);
    if (vet == null)
    {
        throw new ArgumentException("Invalid veterinarian ID");
    }

    List<string> workingDays = GetWorkingDays(vet.WorkSchedule);
    string timeRange = GetTimeRange(vet.WorkSchedule);

    string appointmentDayAbbreviation = GetDayAbbreviation(appointmentResource.AppointmentDate.ToString("ddd").ToUpper());

    if (!workingDays.Contains(appointmentDayAbbreviation))
    {
        throw new InvalidOperationException("Appointment date is not within veterinarian's work schedule"+ appointmentDayAbbreviation);
    }

    if (!IsAppointmentTimeWithinRange(timeRange, appointmentResource.AppointmentDate))
    {
        throw new InvalidOperationException("Appointment time is not within veterinarian's working hours");
    }

    DateTime startTime = appointmentResource.AppointmentDate;
    DateTime endTime = appointmentResource.AppointmentDate.AddMinutes(30);

    // Check if the appointment slot is available
    if (!_context.Appointments.Any(a => a.VetId == appointmentResource.VetId &&
                                        a.AppointmentDate >= startTime && a.AppointmentDate < endTime))
    {
        // Create the appointment
        _context.Appointments.Add(new Appointment
        {
            VetId = appointmentResource.VetId,
            OwnerId = appointmentResource.OwnerId,
            PetId = appointmentResource.PetId,
            Description = appointmentResource.Description,
            AppointmentDate = startTime,
            Category = appointmentResource.Category,
            Status = appointmentResource.Status
        });
        await _context.SaveChangesAsync();

        return true; // Appointment created successfully
    }
    throw new InvalidOperationException("Appointment slot is not available");
}
```

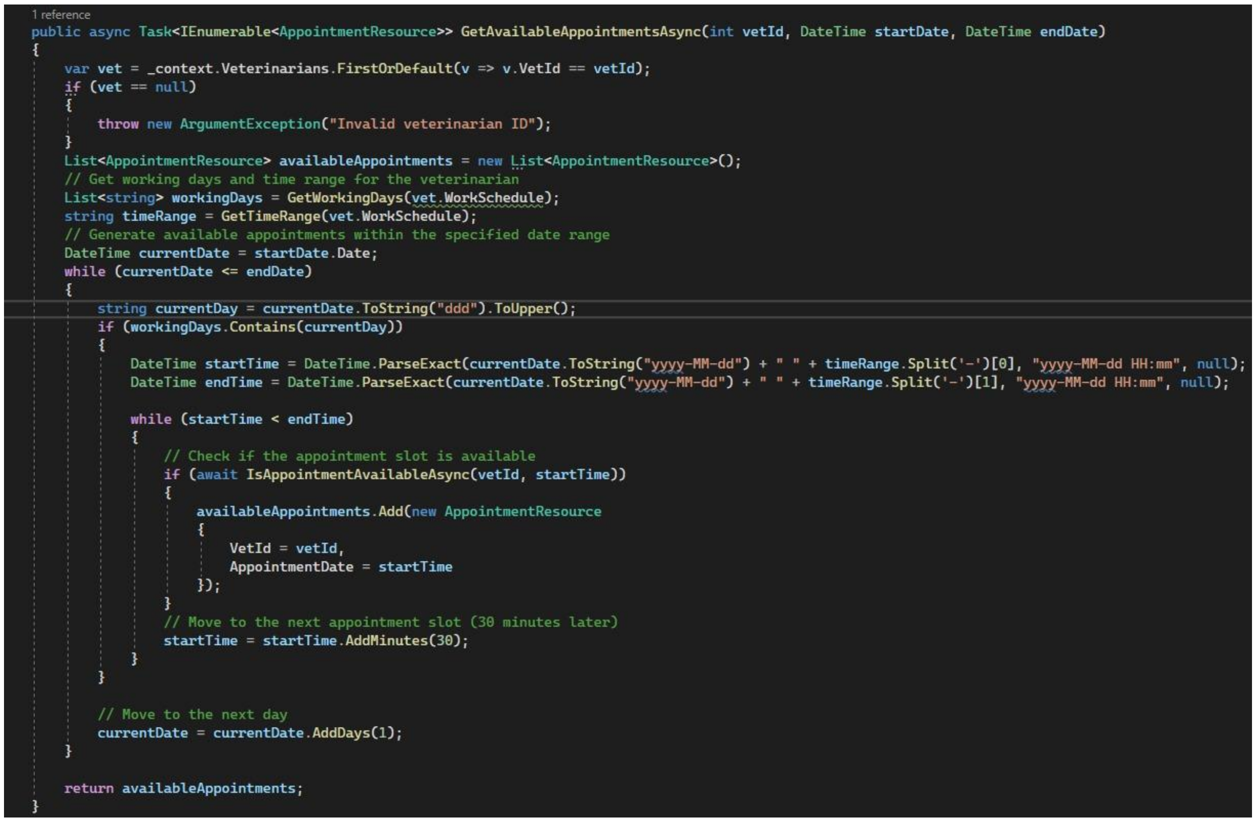
Figure 3.1: Create appointment function

- **Purpose**: Allows users to create new appointments, ensuring no scheduling conflicts and validating against the veterinarian's available hours.
- **Process**:
  - **Validation**: Checks that the appointment date and time are within the available slots and does not overlap with existing appointments.
  - **Execution**: Adds the appointment to the database after successful validation.
- **Technology**: ASP.NET Web API handles the request, and Entity Framework is used for database operations.
- **Frontend Connection**:
  - **Pet Owner Interface**: Pet owners use this feature through the appointment booking section.
  - **Veterinarian Interface**: Allows veterinarians to add appointments directly.
- **Code Snippet**:

```
csharp
[HttpPost]
public async Task<IActionResult> CreateAppointment(AppointmentResource
appointmentResource) {
    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    if (!IsSlotAvailable(appointmentResource))
        return BadRequest("Appointment slot is not available.");

    var appointment = new Appointment {
        VetId = appointmentResource.VetId,
        PetId = appointmentResource.PetId,
        Date = appointmentResource.Date
    };

    await _context.Appointments.AddAsync(appointment);
```

```
        await _context.SaveChangesAsync();

        return Ok();
    }
```

GetAvailableAppointmentAsync Method



```
1 reference
public async Task<IEnumerable<AppointmentResource>> GetAvailableAppointmentsAsync(int vetId, DateTime startDate, DateTime endDate)
{
    var vet = _context.Veterinarians.FirstOrDefault(v => v.VetId == vetId);
    if (vet == null)
    {
        throw new ArgumentException("Invalid veterinarian ID");
    }
    List<AppointmentResource> availableAppointments = new List<AppointmentResource>();
    // Get working days and time range for the veterinarian
    List<string> workingDays = GetWorkingDays(vet.WorkSchedule);
    string timeRange = GetTimeRange(vet.WorkSchedule);
    // Generate available appointments within the specified date range
    DateTime currentDate = startDate.Date;
    while (currentDate <= endDate)
    {
        string currentDay = currentDate.ToString("ddd").ToUpper();
        if (workingDays.Contains(currentDay))
        {
            DateTime startTime = DateTime.ParseExact(currentDate.ToString("yyyy-MM-dd") + " " + timeRange.Split('-')[0], "yyyy-MM-dd HH:mm", null);
            DateTime endTime = DateTime.ParseExact(currentDate.ToString("yyyy-MM-dd") + " " + timeRange.Split('-')[1], "yyyy-MM-dd HH:mm", null);

            while (startTime < endTime)
            {
                // Check if the appointment slot is available
                if (await IsAppointmentAvailableAsync(vetId, startTime))
                {
                    availableAppointments.Add(new AppointmentResource
                    {
                        VetId = vetId,
                        AppointmentDate = startTime
                    });
                }
                // Move to the next appointment slot (30 minutes later)
                startTime = startTime.AddMinutes(30);
            }
        }

        // Move to the next day
        currentDate = currentDate.AddDays(1);
    }

    return availableAppointments;
}
```

Figure 3.2: Get Available Appointment Function

UpdateAppointmentAsync Method

- **Purpose**: Updates the details of an existing appointment, ensuring the new timing is available.
- **Process**:
  - **Validation**: Confirms new timing does not conflict with other bookings.
  - **Execution**: Updates the appointment in the database.
- **Technology**: Managed by ASP.NET Web API with Entity Framework for ORM.
- **Frontend Connection**:

- **Veterinarian Interface**: Used for managing and rescheduling appointments.

- **Code Snippet**:

```
    csharp
[HttpPut("{id}")]
public async Task<IActionResult> UpdateAppointment(int id, AppointmentUpdateResource
resource) {
   var appointment = await _context.Appointments.FindAsync(id);
   if (appointment == null)
      return NotFound();

   appointment.Date = resource.Date;
   _context.Appointments.Update(appointment);
   await _context.SaveChangesAsync();

   return Ok();
}
```

User Management

Handles registration, authentication, and profile updates, ensuring a secure and personalized experience for each user.

Authentication Methods

- **Purpose**: Secures user access by validating credentials and issuing tokens.
- **Process**:
  - **Validation**: Checks credentials against the database.
  - **Execution**: Issues a JWT token for authenticated sessions.
- **Technology**: ASP.NET Web API for handling requests and JWT for session management.
- **Frontend Connection**:

- **Common Pages**: Utilized in the login and registration processes.

- **Code Snippet**:

```
    csharp
[HttpPost("login")]
public async Task<IActionResult> Authenticate(LoginModel model) {
    var user = await _context.Users
            .FirstOrDefaultAsync(u => u.Username == model.Username && u.Password ==
model.Password);

    if (user == null)
        return Unauthorized();

    var token = GenerateJwtToken(user);
    return Ok(new { Token = token });
}
```

Conclusion

The backend of the petcare app, built with ASP.NET Web API, provides a powerful, scalable, and secure platform for managing complex data interactions and user functionalities. Its integration with frontend technologies ensures a seamless and dynamic user experience, essential for the effective management of veterinary services. Future developments could include enhancing API security, integrating more third-party services for expanded functionality, and employing machine learning for predictive analytics in pet healthcare. This comprehensive backend architecture facilitates robust application performance and offers extensive capabilities to meet the growing demands of the veterinary industry.

VIII. Frontend

1. **Introduction**

This section provides an in-depth overview of the frontend design and functionality of the petcare app developed using Flutter. Flutter's robust framework allows for the creation of a seamless, cross-platform user experience. The application caters to three main user roles: Pet Owners, Veterinarians, and Administrators, each with specialized interfaces to efficiently manage their interactions and responsibilities.

2. **Common Features Across All User Types**

Login and Sign Up Pages

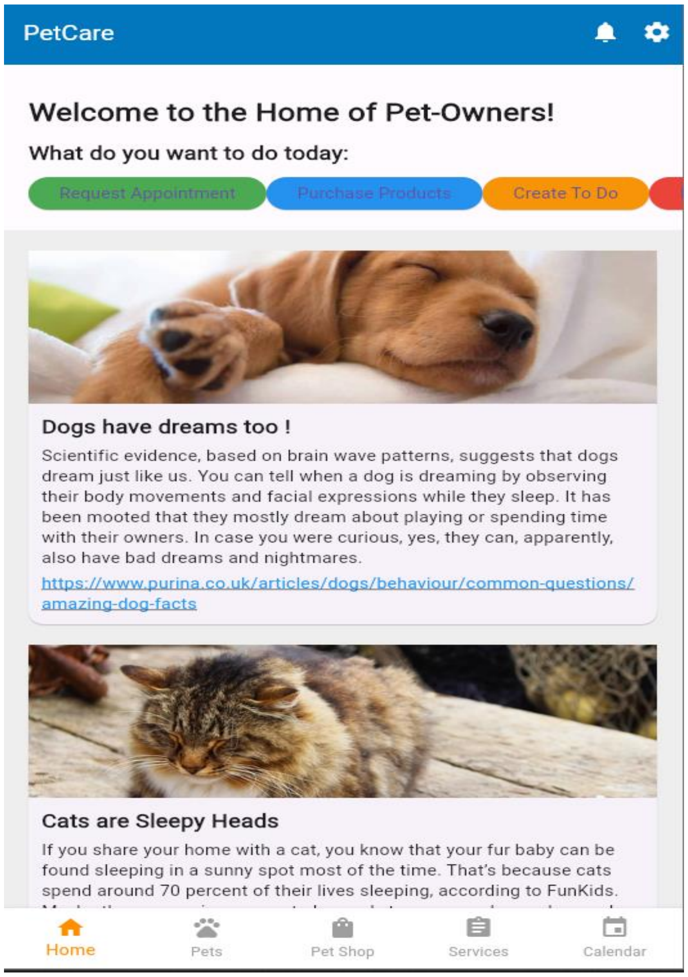
- **Functionality**: Both pages serve as the entry point for all users, featuring form inputs validated for email and secure password entry.
- **Design**: Clean and user-friendly, with clear navigation to either log into existing accounts or register a new account.
- **Code Snippet** (Login Page):

```
   dart
TextField(
  decoration: InputDecoration(labelText: 'Email', hintText: 'Enter your email'),
),
TextField(
  decoration: InputDecoration(
    labelText: 'Password',
    hintText: 'Enter your password',
    obscureText: true,
  ),
),
ElevatedButton(
  onPressed: () {
    // Authentication logic
  },
  child: Text('Login'),
),
```

1. Pet Owner Interface

Pet owners have a comprehensive dashboard to manage their pets, appointments, and purchases.

Home Page



Welcome to the Home of Pet-Owners!

What do you want to do today:

Dogs have dreams too !

Scientific evidence, based on brain wave patterns, suggests that dogs dream just like us. You can tell when a dog is dreaming by observing their body movements and facial expressions while they sleep. It has been mooted that they mostly dream about playing or spending time with their owners. In case you were curious, yes, they can, apparently, also have bad dreams and nightmares.

https://www.purina.co.uk/articles/dogs/behaviour/common-questions/amazing-dog-facts

Cats are Sleepy Heads

If you share your home with a cat, you know that your fur baby can be found sleeping in a sunny spot most of the time. That's because cats spend around 70 percent of their lives sleeping, according to FunKids.

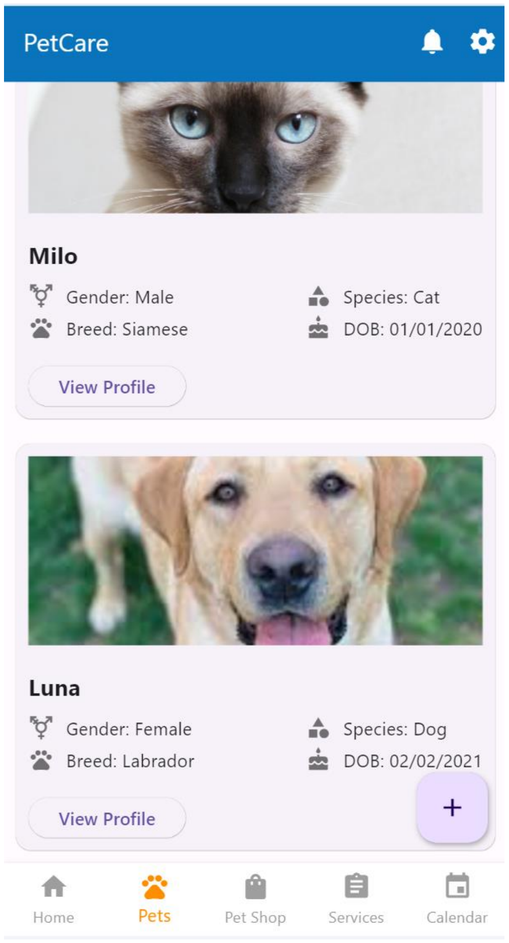
Figure 4.1: Pet Owner Home Page

- **Description**: Acts as a central dashboard providing quick links to essential features such as appointment scheduling, product purchases, and informational articles.
- **Features**: Widgets for quick actions and educational content snippets.

- **Code Snippet**:

```
        dart
Column(
  children: <Widget>[
    ElevatedButton(
      onPressed: () { Navigator.pushNamed(context, '/appointments'); },
      child: Text('Request Appointment'),
    ),
    ElevatedButton(
      onPressed: () { Navigator.pushNamed(context, '/petShop'); },
      child: Text('Purchase Products'),
    ),
  ],
)
```

Pets Page



Milo
Gender: Male
Species: Cat
Breed: Siamese
DOB: 01/01/2020

Luna
Gender: Female
Species: Dog
Breed: Labrador
DOB: 02/02/2021

Figure 4.2: Pet Manager Page

- **Description**: Displays a list of the user's pets, providing quick access to detailed profiles and health records.
- **Features**: List view of pets, with navigation to detailed pet profiles.
- **Code Snippet**:

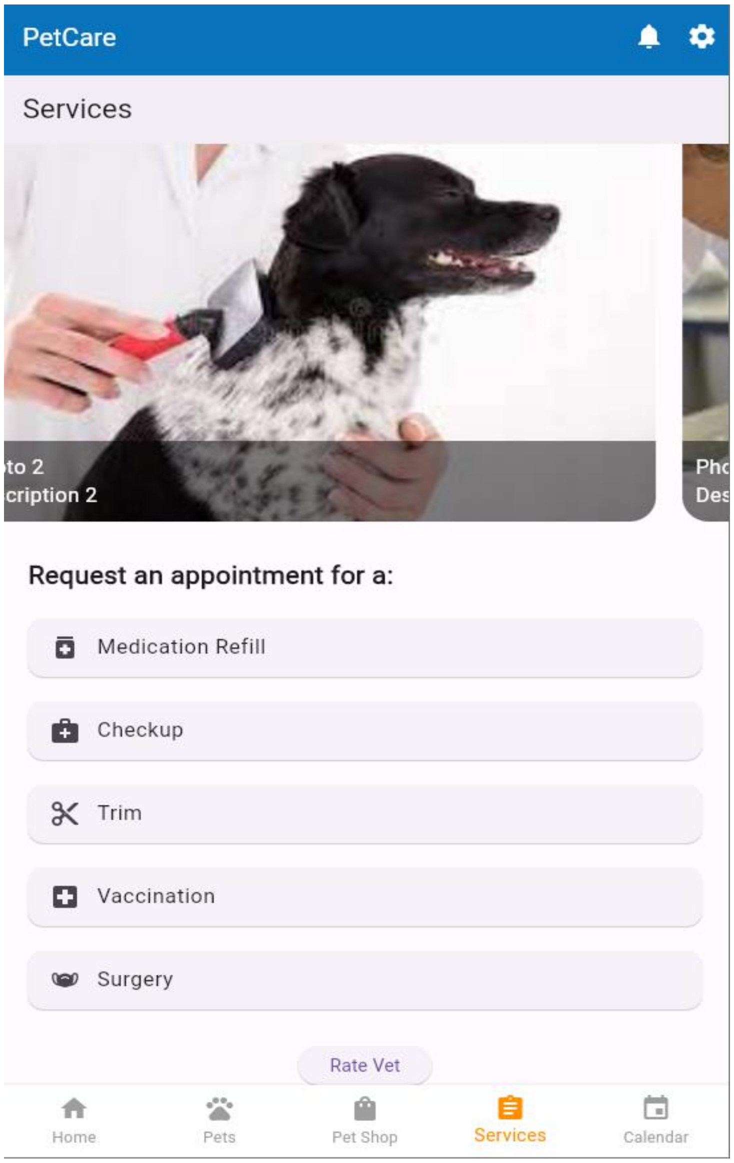
```
   dart
ListView.builder(
  itemCount: pets.length,
  itemBuilder: (context, index) {
    return ListTile(
      title: Text(pets[index].name),
      subtitle: Text('${pets[index].species} - ${pets[index].breed}'),
      onTap: () {
        // Details page navigation
      },
    );
  },
)
```

Pet Shop Page

- **Description**: A marketplace for pet-related products categorized for easy browsing.
- **Features**: Search functionality, category filters, and an add-to-cart feature.
- **Code Snippet**:

```
   dart
GridView.builder(
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 2),
  itemCount: products.length,
  itemBuilder: (context, index) {
    return Card(
      child: ListTile(
        title: Text(products[index].name),
        leading: Image.network(products[index].imageUrl),
        onTap: () {
          // Add to cart functionality
        },
      ),
    );
  },
)
```

Services Page



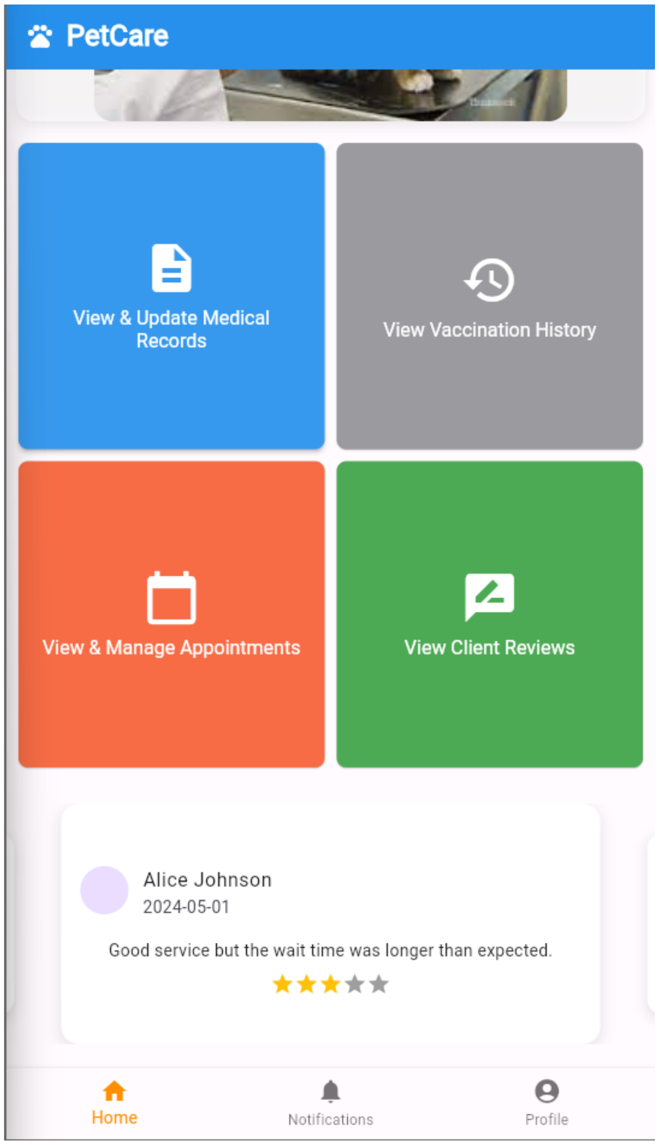
Request an appointment for a:

Figure 4.3: Pet Owner Services Page

2. Veterinarian Interface

Veterinarians can manage their appointments, patient records, and communicate with pet owners.

Services Page



Alice Johnson
2024-05-01
Good service but the wait time was longer than expected.
★★★☆☆

Figure 4.4: Veterinarian Home Page

- **Description**: Allows veterinarians to view their daily schedule, manage appointments, and upload patient records.
- **Features**: Overview of today's appointments and direct access to upload test results.

- **Code Snippet**:

```
        dart
ListView(
  children: appointments.map((appointment) {
    return ListTile(
      title: Text(appointment.petName),
      subtitle: Text(appointment.time.toString()),
      onTap: () {
        // View appointment details
      },
    );
  }).toList(),)
```

3. Administrator Interface

Administrators can oversee and manage user accounts, veterinarian schedules, and inventory.

Services Page

- **Description**: Provides tools for account management and resource allocation.
- **Features**: User management tools and inventory tracking systems.
- **Code Snippet**:

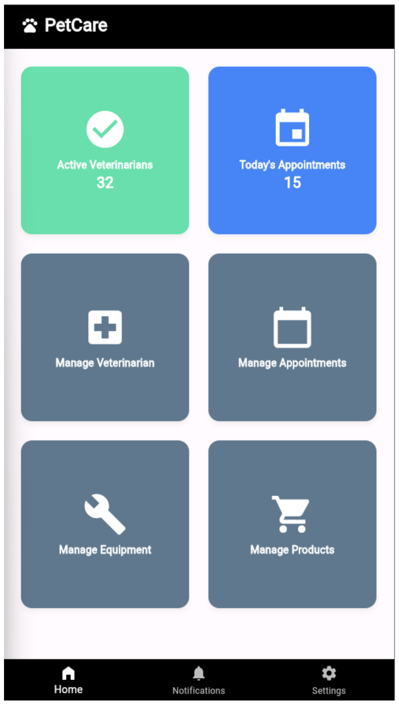


Figure 4.5: Admin Home Page

```
      dart
ListView.builder(
  itemCount: users.length,
  itemBuilder: (context, index) {
    return ListTile(
      title: Text(users[index].name),
      subtitle: Text('Role: ${users[index].role}'),
      onTap: () {
        // Manage user settings
      },
    );
  },
)
```

Conclusion

The frontend of the petcare app leverages Flutter's capabilities to provide a responsive, intuitive user experience across different platforms and user roles. Each interface is specifically designed to meet the unique needs of pet owners, veterinarians, and administrators, ensuring efficient management of pet care and clinic operations. Future enhancements can include more interactive features and integration with external APIs for expanded functionalities.

IX. Conclusion

In conclusion, the PetCare app emerges as a crucial asset for pet owners, veterinarians, and administrative staff within the veterinary industry. By providing a comprehensive suite of features tailored to meet the diverse needs of stakeholders, PetCare facilitates efficient pet healthcare management and enhances the overall well-being of pets.

One of the standout features of PetCare is its ability to centralize and securely manage pet information and medical records. This ensures that veterinarians have access to essential data, enabling them to deliver informed and personalized care to pets. Moreover, the app's emphasis on telemedicine consultations and symptom checking tools empowers pet owners to proactively monitor their pets' health and seek timely veterinary assistance when needed.

Beyond medical management, PetCare streamlines administrative tasks such as appointment scheduling, inventory management, and billing, thereby optimizing clinic operations and minimizing administrative burdens. This enables veterinary clinics to focus more on delivering high-quality care to pets and less on paperwork and logistical challenges.

Furthermore, PetCare fosters improved communication and collaboration between pet owners and veterinarians, facilitating seamless interaction and ensuring that pets receive comprehensive care tailored to their individual needs.

In essence, PetCare represents an innovative and indispensable solution for modern pet healthcare needs. Its user-friendly interface, comprehensive functionalities, and emphasis on efficiency and collaboration make it a valuable tool for enhancing the pet healthcare experience. As technology continues to play a pivotal role in veterinary medicine, PetCare is poised to play a significant role in shaping the future of pet healthcare management.

X. References:

I. Flutter & Dart:

1. Flutter Documentation: https://flutter.dev/docs
2. Dart Documentation: https://dart.dev/guides
3. Flutter Packages Repository: https://pub.dev/

II. ASP.NET Web API:

1. ASP.NET Core Documentation: https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0
2. Building Web APIs with ASP.NET Core: https://dotnet.microsoft.com/learn/aspnet/web-api-tutorial

III. Veterinary Clinic App Development:

1. Building Mobile Apps with Flutter: https://flutter.dev/docs/get-started/install
2. UI Design Patterns for Mobile Apps: https://flutter.dev/docs/development/ui/design-patterns
3. Integrating Backend with Flutter: https://flutter.dev/docs/development/data-and-backend
4. Building RESTful APIs with ASP.NET Core: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api