Software Requirements Specification for VetCare

Christian Nieves, Seanghai Heng, Keenan Phillips, William Dash, Heethasha Sandeep Kumar, & Paul Johny Mampily
RMIT University

8/25/24

Table of Contents

1. Introduction

1.1 Overview and Purpose

For pet owners, sometimes it isn't easy to find a vet clinic when they are sick or cannot find valuable information regarding their behaviour and welfare. For these reasons, The VetCare project aims to provide a convenient way for pet owners to ensure their pets receive pet care services with just a few clicks of a button. The VetCare project is an innovative online web application that utilizes modern technologies to meet the needs of pet owners. VetCare is also designed to be used by pet owners as well as admins to enhance the wellness of pets since it is integrated with local veterinary clinics and pet stores.

1.2 Scope

The primary purpose of VetCare is to deliver an accessible solution for managing various aspects of pet care. By making use of cutting-edge technology, the VetCare aims to:

- Allow pet owners to easily book, reschedule, or cancel appointments, view and manage medical records, and request prescription refills with minimal effort.
- Offer secure and convenient access to detailed medical histories, vaccination records, and treatment plans, as well as up-to-date information about veterinary providers and products.
- Allow users to export data and share it with professional's veterinary.
- Allow pet owners to request their prescriptions with delivery options.
- Provide a comprehensive library of articles, videos, and guides on pet care, keeping users informed about the latest trends and best practices.
- Provide users with a user-friendly platform that provides a smooth user experience.
- Provide users with reviews and rating options within the application.

2. Functional Requirements

2.1 Booking Appointments

- The system shall allow the user to book an appointment.
- The system must allow the pet owner to select a preferred veterinarian.
- The system must provide an interface where pet owners can select a date and time for their appointment.
- The system shall allow users to view upcoming appointments.
- The system must allow users to reschedule or cancel appointments.
- The system must validate that the selected time slot is still available at the time of booking.
- The system must send confirmation emails for appointments, including details of the appointment.

2.2 Account Creation

- The system shall allow the user to create an account using an email.
- The system must provide a registration form where users can enter their details.
- The system must check if the account already exists.
- The system must verify the password and confirmation password before proceeding to account creation.
- The system must send a verification email to the user after registration, requiring confirmation before the account is activated.

2.3 Update Pet’s Medical Data

- The system must allow veterinarians to access the pet's medical information.
- The system must allow veterinarians to update pet medical data such as vaccination records and medical history.
- The veterinarian must be able to confirm or cancel the changes.

- The system must display a status message when updating the pet's information (e.g., "Update successful").
- The system must log all changes made to a pet's medical data for audit purposes.

2.4 Payments

- The system must integrate with a secure payment gateway (Stripe).
- The system must support various payment methods, including credit/debit cards.
- The system must send a confirmation message when payment is received.
- The system must send an error message when the payment fails.
- The system must allow users to save payment methods securely for future use.
- The system must provide an option for users for request refunds and display the status of refund requests.

2.5 Stay Informed with the Latest Trends

- The system must allow users to turn on notifications for the latest updates.
- The trends must be sorted by published date.
- The system must allow users to manage their notification preferences (e.g., frequency, type of content).

2.6 Search and Filters

- The system must provide a search bar where users can enter keywords to search for veterinary services.
- The system must provide various filters that users can apply to narrow down the list of veterinary clinics.
- The system shall display the results page of clinics with their name, location, and service.
- The system must provide sorting options to allow users to order search results by name or location.
- The system must provide relevant search results even if no exact matches are found (e.g., fuzzy matching).

- The system must handle cases where no search results are found and provide suggestions.

2.7 Updating Personal Details

- The system must ensure that only authenticated users can access and update their details.
- The system must provide an interface where users can view and update their details.
- The system must validate the input for each field (e.g., email format, phone number format).
- The system must allow the user to delete his/her account.
- The system must log all updates to personal details, including the date and time of changes.
- The system must enforce a cooldown period between account deletions and re-registration.

2.8 Leaving Feedback and Reviews

- The system shall allow users to leave a review after they have an appointment with the clinics.
- The system must provide a form where users can enter their feedback and review details.
- The system must validate that all required fields are completed before allowing the submission of feedback.
- The system must provide a confirmation message upon successful submission of feedback.

3. Non-Functional Requirements

3.1. Usability Requirements

- The system interface must be intuitive and follow standard web design conventions.
- The system must provide help documentation or tooltips to assist users with complex tasks.
- The system must allow new users to complete key tasks (e.g., booking an appointment) within 5 minutes of first use.

3.2. Performance Requirements

- The system must load the homepage within 3 seconds under normal conditions.
- The system must be able to handle at least 300 concurrent users without significant degradation in performance.
- The system must aim for a query response times of under 500 milliseconds to ensure quick retrieval of data.
- The system must remain responsive during a high concurrent user count, with no more than a 5% increase in page load times.

3.3. Security Requirements

- The system must require users to authenticate using a secure login mechanism, including email and password, with two-factor authentication as an option.

3.4 Compatibility Requirements

- The system must be compatible with major web browsers, including Chrome, Firefox, Safari, and Edge.
- The system must display correctly on a most desktop and laptop screen sizes with a 16:9 aspect ratio.

- The system must support a range of operating systems, including Windows, macOS, Linux, iOS, and Android.

3.5 Reliability Requirements

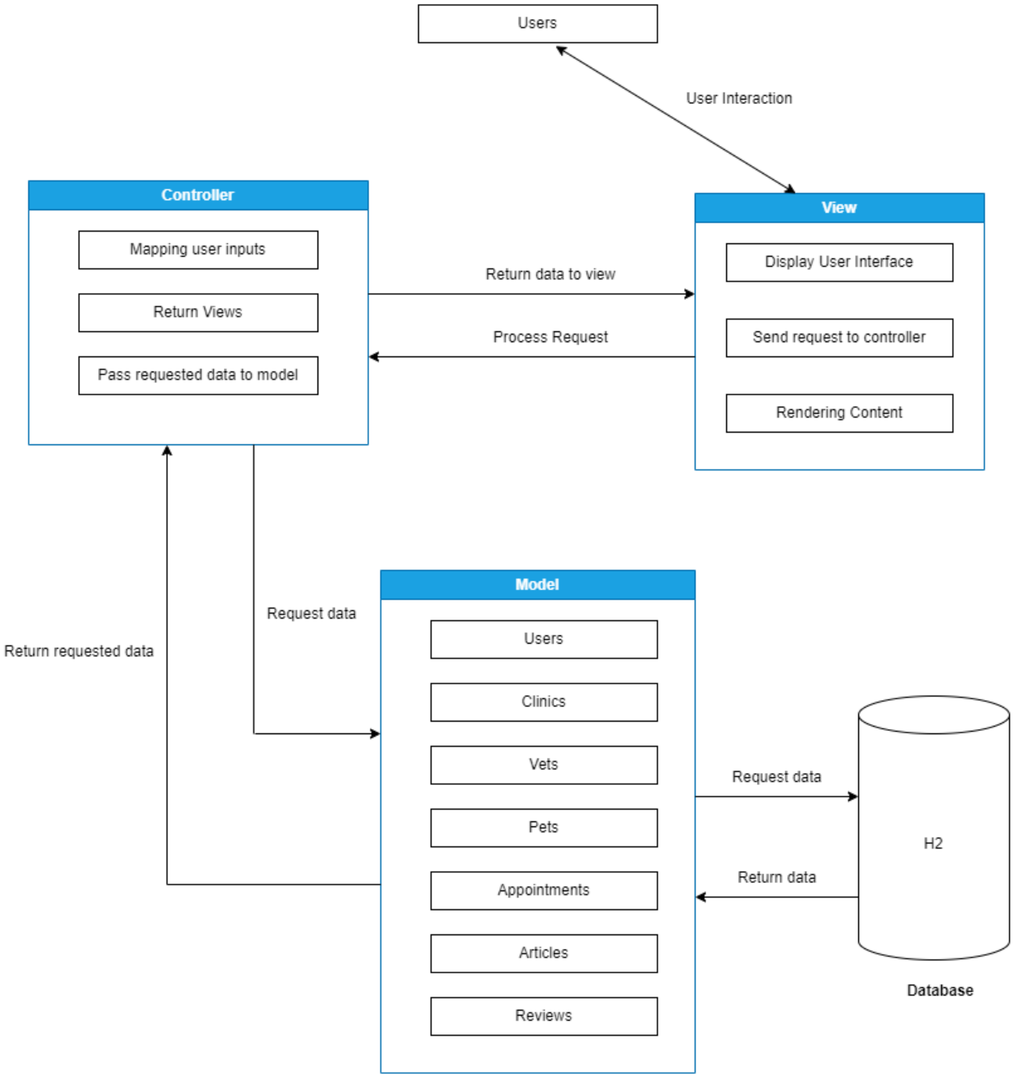
- The system must have an uptime of 99.9% or higher.
- The system must automatically back up data daily.

3.6 Maintainability Requirements

- The system's codebase must be well-documented.
- The system must use version control (e.g., Git) to manage code changes and ensure that all updates are tracked and reversible.
- The system must be designed with scalability in mind.
- The system must include automated testing options to verify the functionality of new code before it is deployed.

4. System Architecture

4.1 System Architecture Diagram



4.2 Model

The Model component handles data and logic for the VetCare application. It receives data from the Controller and then passes it to a database for queries. After it gets the data from the database, it will pass the data back to the Controller.

4.3 View

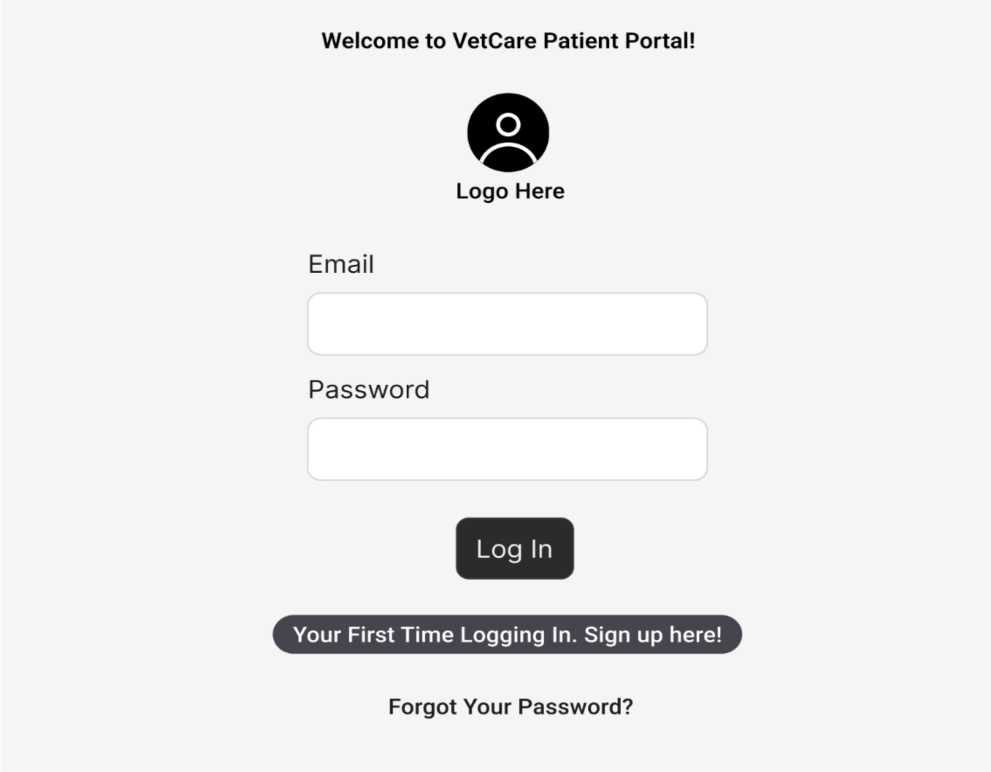
The View component displays user interfaces and handles user interactions. It receives data inputs from the user and sends it to the controller for further processing. After getting the data from the Controller, it then re-renders the content and displays it to the user.

4.4 Controller

The Controller handles user interaction such as user inputs and mouse clicks. For each user interaction, it passes the data to the Model for performing operations. After receiving data from the Model, it will return the data to View.

5. User Interface Design

5.1 Login page



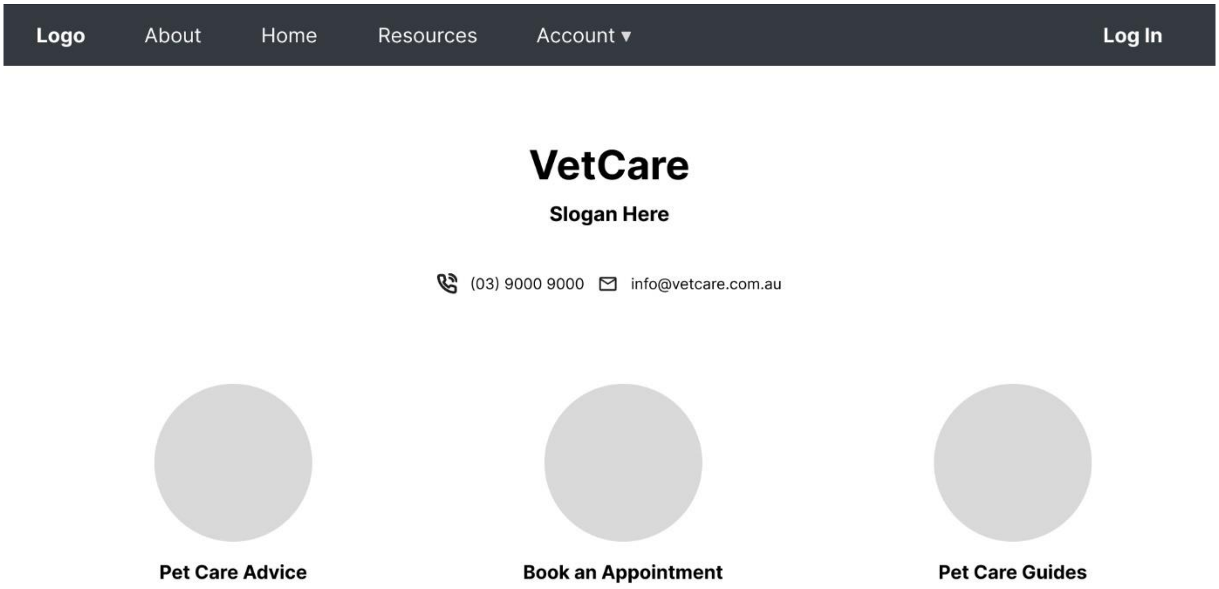
Welcome to VetCare Patient Portal!

Logo Here

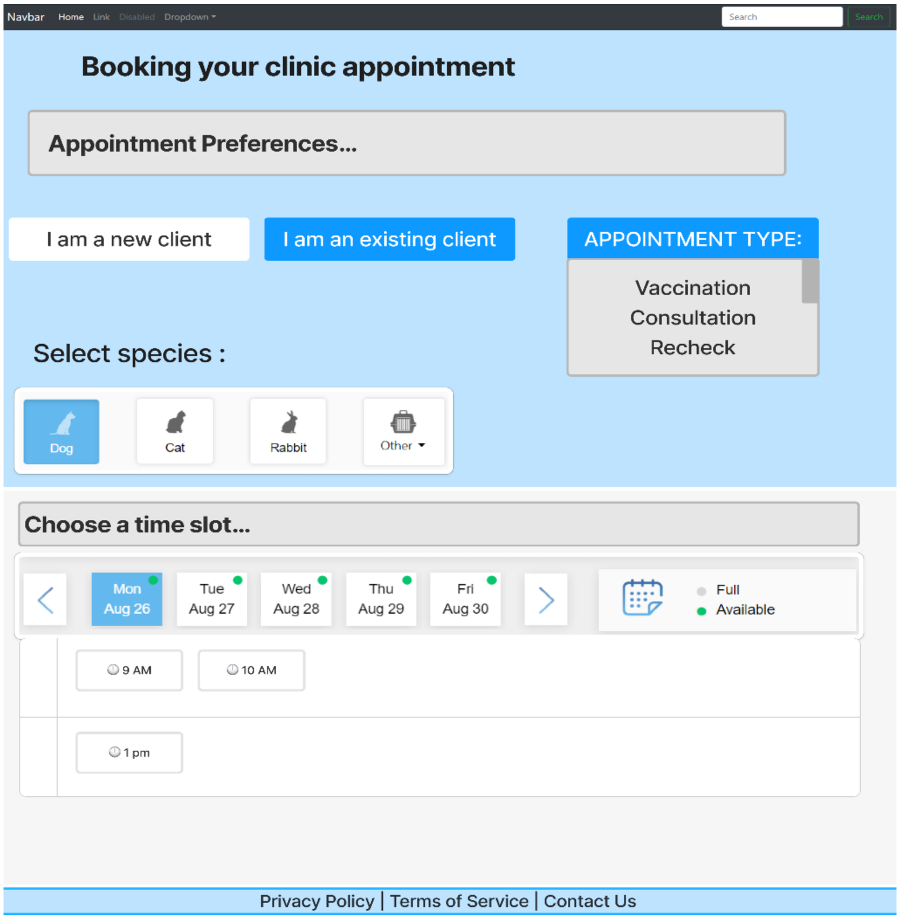
Email

Password

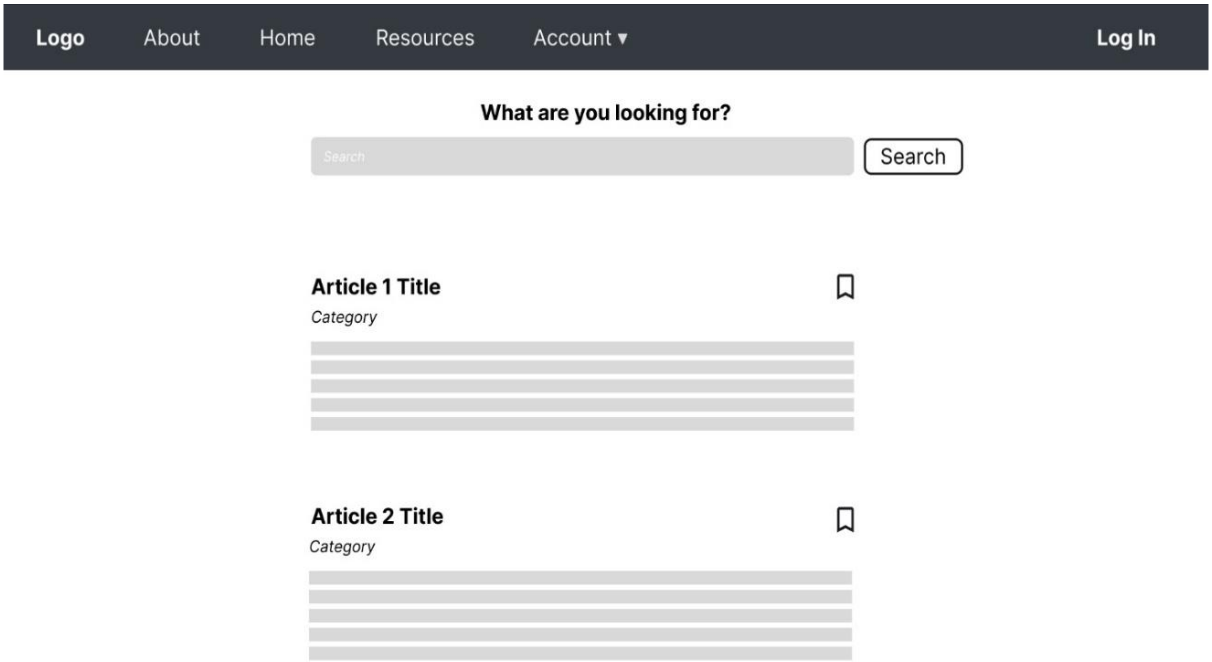
5.2 Landing page



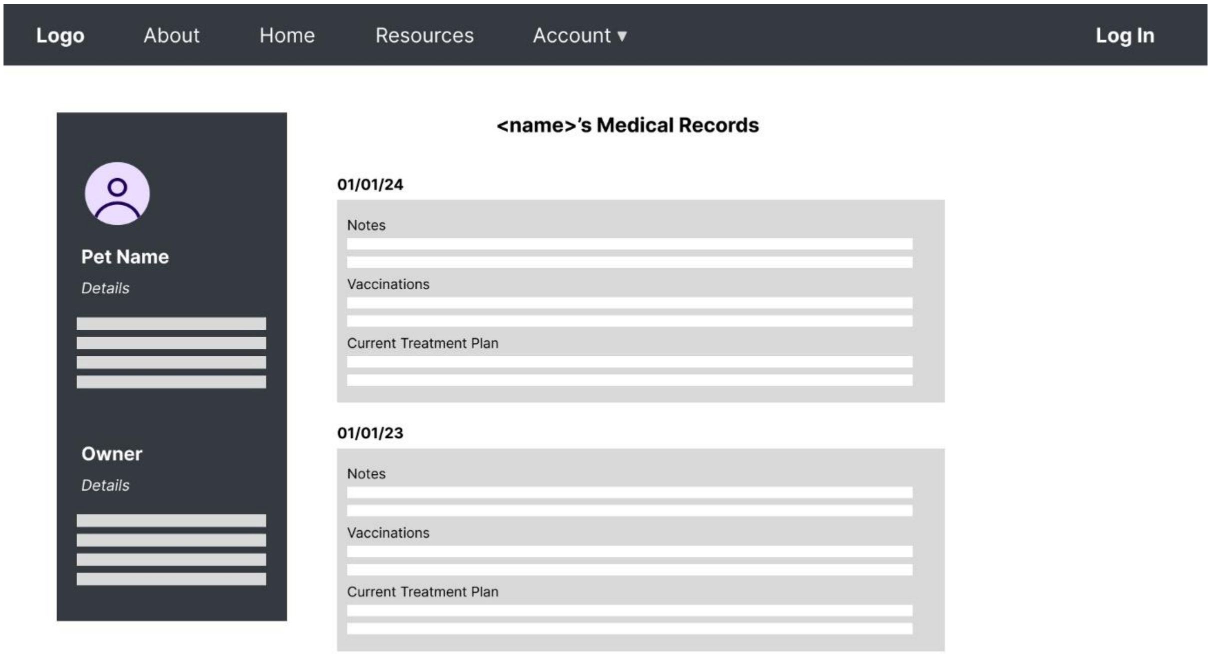
5.3 Appointments page



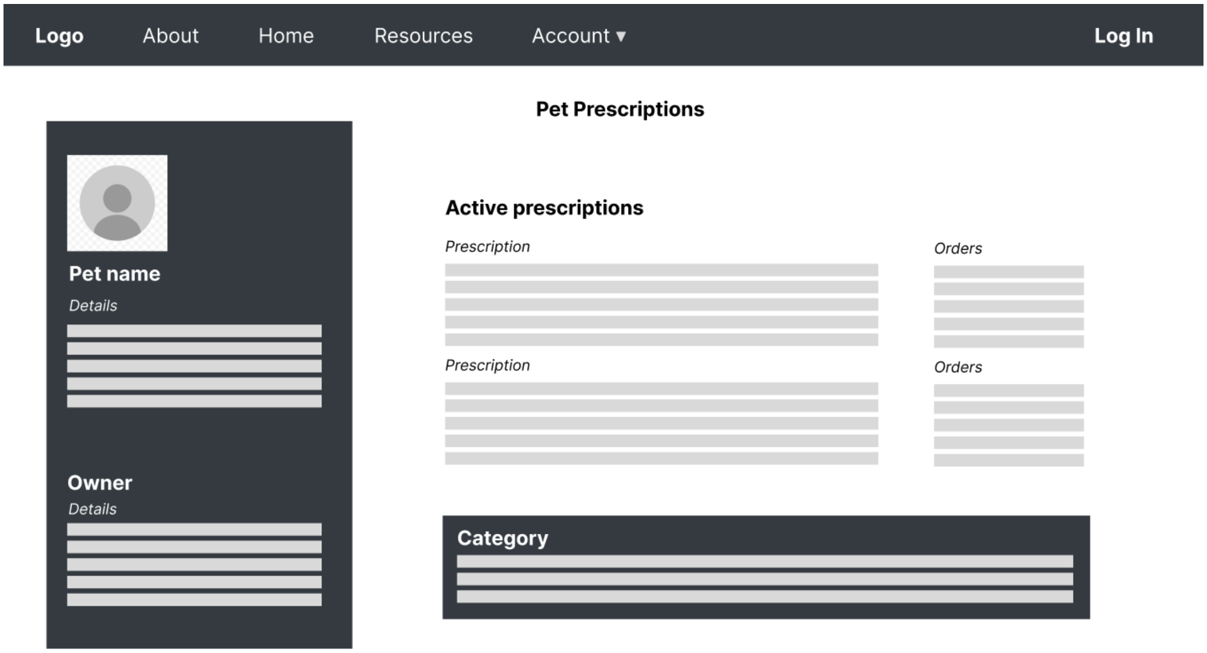
5.4 Educational Resources page



5.5 Medical Records page



5.6 Prescriptions page



6. Assumptions and Constraints

6.1 Assumptions

- The system should store and access appointment and pet information from a database as opposed to any other storage format.
- Users can access their pet's information securely through an account system.
- Veterinarians should have access to pet medical data and can edit this data.
- Veterinarians should have their own accounts separate from the users.
- Users should pay during booking with a third-party payment system such as Stripe.
- Users can stay up to date through notifications.
- Users should have the ability to delete their accounts.
- Users can leave reviews on veterinarians to give feedback on their experience.

6.2 Constraints

- We are limited by our resources, most notably time. This will impact the size of our scope for the system, reducing the quality of our implementations and the number of features we can include in our solution.
- Since we are students and are working with new technologies, our experience is limited. Due to this it will take a longer period for us to develop.
- To comply with local laws, we need to provide each user with the option to delete their account if they request. So, the feature for users to delete their account has been added.

7. Dependencies

VetCare's successful operation relies on the following external dependencies and libraries:

7.1 Payment Processing – Stripe API

- Stripe is a secure and reliable service for processing online payments. The system uses the Stripe API to handle all payment related processes, such as prescription ordering and online vet fee payments.
- The latest stable version of the API will be used.

7.2 Database – H2

- H2 Database is a lightweight relational database management system (DBMS). Since H2 is lightweight and provides an easy setup for testing, it will be the preferred DBMS for this application.

7.3 User Authentication and Authorisation – OAuth 2.0

- All users (pet owners and vets) will need login credentials. The system will use OAuth 2.0 as the protocol for user authentication and authorisation, which ensures secure access to VetCare.

7.4 Date and Time Handling – Java's Native Date/Time Library

- To ensure accuracy with appointment bookings and date/time calculations, the system will use Java's native time library (`java.time`).
- Since the system uses Java SE 17 or later, the native time library is included by importing the library.

7.5 Application Framework – Spring Boot

- The system is built using the Spring framework, which uses the model-view-controller (MVC) architecture for the application.

7.6 Unit Testing – JUnit

- The system will use JUnit for writing and running unit tests. Its purpose is to ensure that individual sections of code work as expected.
- JUnit 5.0 will be used.

7.7 Templating Engine – Thymeleaf

- The system will use Thymeleaf as a templating engine for displaying and rendering all the website's HTML content.
- Thymeleaf 3.1.2 will be used.

7.8 Utilities – Apache Commons

- The system will make use of Apache Commons, which comprises of reusable Java components for handling common operations like file I/O and strings. Implementing these will reduce development time for common programming tasks.

7.9 Front-End Framework – Bootstrap

- Bootstrap provides simple and reusable components for user interfaces, which reduces the need for custom CSS.
- Components like navigation bars, forms, and buttons will be integrated into the application.

7.10 PDF Generation – Apache PDFBox

- All users have the option of downloading their past appointment history. This API will allow us to create basic PDFs with the user's name, appointment date and time, vet name, and any costs associated with the appointment.

8. Glossary

API (Application Programming Interface): A set of functions and instructions that allow different software applications to communicate with each other. In this project, the Stripe API are used for payments, while OAuth 2.0 is used for user authentication.

Authentication: The process of verifying the identity of a user or system. In VetCare, authentication ensures that only registered users and veterinarians can access their accounts.

CRUD Operations: An acronym for Create, Read, Update, and Delete operations. These are the basic functions of persistent storage and are used throughout the VetCare system to manage data in the database.

Database Management System (DBMS): A tool that uses a database to store, retrieve, and manage data. VetCare uses the H2 database, a lightweight relational DBMS, for storing data related to users, pets, appointments, and more.

Entity-Relationship Diagram (ERD): A visual representation of the database schema, showing how different entities (e.g., Users, Pets, Appointments) are related to each other.

PDF (Portable Document Format): A file format used to present documents consistently across different devices and platforms. VetCare uses Apache PDFBox to generate PDF documents for appointment histories.

Thymeleaf: A modern Java-based templating engine used to process and generate HTML views in the VetCare application.

Two-Factor Authentication (2FA): An additional layer of security that requires not only a username and password but also something that only the user has access to, such as a mobile device, to verify their identity.

Unit Testing: A software testing method where individual units or components of a software are tested by themselves. VetCare uses JUnit for unit testing to ensure that individual parts of the code are correct.

User Story: A simple description of a feature from the perspective of the user or customer. User stories in VetCare guide the development of features, such as account creation or booking appointments.

Version Control: A system that records changes to files over time so that specific versions can be used later. VetCare uses Git for version control to manage changes to the codebase.

Vet: Short for veterinarian, a medical professional who treats animals. In VetCare, vets are users with specific roles that allow them to access and update pet medical data.

Web Framework: A software framework that is designed to support the development of web applications including web services, web resources, and web APIs. VetCare uses Spring Boot as its web framework.

Wireframe: A basic visual guide used to suggest the layout and structure of a web page or app interface without focusing on smaller design details, such as colours.

User Interface (UI): The means by which the user and a computer interact. In VetCare, UI components are designed to be intuitive and user-friendly to enhance user experience.

User Experience (UX): The experience of a user when interacting with the application, mostly in terms of how easy it is to use.