Software Requirements Specification

for

VetCare

Version 1.0 approved

Prepared by Francis Zaldarriaga

RMIT University

15/08/2024

Table of Contents Error! Bookmark not defined.

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 2

1.3 Intended Audience and Reading Suggestions 2

1.4 Product Scope 5

1.5 References 7

2. Overall Description 7

2.1 Product Perspective 7

2.2 Product Functions 8

2.3 User Classes and Characteristics 9

2.4 Operating Environment…………………………………………………………......10

2.5 Design and Implementation Constraints 12

2.6 User Documentation 14

2.7 Assumptions and Dependencies 14

3. External Interface Requirements 15

3.1 User Interfaces 15

3.2 Hardware Interfaces 16

3.3 Software Interfaces 16

3.4 Communications Interfaces 18

4. Nonfunctional Requirements 19

4.1 Performance Requirements 19

4.2 Safety Requirements 20

4.3 Security Requirements 20

4.4 Software Quality Attributes 21

4.5 Business Rules 22

5. Other Requirements 23

6. System Architecture Error! Bookmark not defined.

6.1 Architecture Overview 21

6.2 Architectural Decisions 22

7. User Interface Design 22

Appendix A: Glossary 29

Appendix B: Analysis Models 29

Appendix C: To Be Determined List 29

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

This document’s purpose is to provide a detailed overview of the Online Vet Clinic management system – VetCare. Specifically, a management system that serves to provide pet owners with managing their pet’s healthcare needs. This will include in-depth information on performance, functionalities, system constraints, and documentation of architectural decisions that may change during the development process. The current version outlining this SRS document's application is 1.1.

## Document Conventions

Visual Representation:

* + Information that can be effectively summarized is presented visually using graphs, diagrams, or tables for clarity and ease of understanding.

Formatting for Assumptions and Dependencies:

* + Section 2.7 is listed in a bulleted format for easy reference. Each bullet point is presented as a concise statement to ensure the reader understands the foundational expectations and conditions the system is based on.

Use of Bold Lettering:

* + Key sections, including 1.4 (Features in Product Scope), 2.3 (User Classes/Intended Target Group), 3.1 (Characteristics of the Webpages of the Product), and 4.1 (Performance Requirements), utilize bold text to highlight important information.

Lists and Bullet Points:

* + Lists and bullet points are used to label features, user characteristics, requirements, and rationale to enhance readability and organization.

Text Alignment:

* + The document's text is aligned to 'Justify' maintaining a clean and professional appearance.

Font Style and Size:

* + The font style is set to 'Arial'. The main text font size is set at 12 points, while headers are set at 14 points and section headers at 18 points.

Use of Hyphens:

* + Hyphens are employed as needed to enhance reading clarity and comprehension throughout the document.

Document Structure:

* + The document is divided into clearly numbered sections, each with a corresponding header. A table of contents is provided at the beginning for easy navigation and quick reference to specific sections.

## Intended Audience and Reading Suggestions

|  |  |  |
| --- | --- | --- |
| Sequence order | Developers | Project Managers |
| 1 | **Introduction:** (Section 1): Purpose (1.1) | **Introduction:** (Section 1): Purpose (1.1) and Product Scope (1.4) |
| 2 | **Overall Description :** Product Functions (2.2) and User Classes and Characteristics (2.3) | **Overall Description** (Section 2) especially Assumptions and Dependencies (2.7), |
| 3 | **All of External Interface requirements** | **Design and Implementation Constraints** : (Section 2.5): |
| 4 | **Nonfunctional Requirements:** (Section 4): Performance Requirements (4.1),  Security Requirements (4.3), and Software Quality Attributes (4.4). | **Nonfunctional Requirements** (Section4): Performance Requirements (4.1) and Business Rules (4.5) |
| 5 | **All of System Architecture (**Section 6) | **All of System Architecture (**Section 6) |

|  |  |  |
| --- | --- | --- |
| Sequence order | Marketing Staff | Users |
| 1 | **Introduction** (Section 1): Focus on Product Scope (1.4) | **Introduction** (Section 1) |
| 2 | **Overall Description** (Section 2): Product Functions (2.2) and User Classes and Characteristics (2.3) | **Overall Description** (Section 2): User Classes and Characteristics (2.3) and Product Functions (2.2) |
| 3 | **References** (Section 1.5) | **User Documentation** (Section 2.6) |

|  |  |  |
| --- | --- | --- |
| Sequence order | Testers | Documentation Writers |
| 1 | **Introduction** (Section 1): Start with the Purpose (1.1) | **Introduction** (Section 1): Purpose (1.1) and Intended Audience and Reading Suggestions (1.3) |
| 2 | **Overall Description** (Section 2): Product Functions (2.2) and Assumptions and Dependencies (2.7) | **Overall Description** (Section 2): User Classes and Characteristics (2.3) and Product Functions (2.2), **User Documentation** (Section 2.6) |
| 3 | **External Interface Requirements** (Section 3): User Interfaces (3.1) and Software Interfaces (3.3) | **External Interface Requirements** (Section 3) |
| 4 | **Nonfunctional Requirements** (Section 4): Performance Requirements (4.1),Security Requirements (4.3), and Software Quality Attributes (4.4) |  |

The remaining sections of this Software Requirements Specification (SRS) are organised to give a thorough and in-depth overview of the system that is being created. The document is divided into several sections, each of which focuses on a distinct aspect of the software's requirements, functionalities, and constraints. The contents of each part and the structure of the document are broken down as follows:

1. Introduction

This section focuses on the main objective of the entire project. It starts with Purpose (1.1) that defines the goals of the SRS document and what it covers and its intended use. It also outlines the Document Conventions (1.2) which describes the standards and conventions used in the document to ensure consistency and clarity. It identifies the Intended Audience and Reading Suggestions (1.3) that provides reading suggestions for different reader types. Product Scope (1.4) provides a brief overview of the software, including its purpose, main features, and the business goals it supports. Finally, a list of references is given at the end for any external documents or resources used in this document.

2. Overall Description

Section 2 Overall Description describes the overall view on the software side of the project. It ranges from discussing the Product Perspective (2.1), which describes how the software will integrate within the system, and the main functionalities of the software (Product Functions (2.2). It identifies the different User Classes and Characteristics (2.3), explaining who will be using the software and their specific needs. It goes more in depth by discussing the nitty and gritty side of the software/hardware from webserver, operating system, databases etc. (Operating Environment (2.4). It also includes a list of possible Design and Implementation Constraints (2.5). It also touches on the User Documentation (2.6) that will be provided, such as user manuals or online help, and lists any Assumptions and Dependencies (2.7) that could impact the project’s success

3. External Interface Requirements

The External Interface Requirements section delves into User Interfaces (3.1), which relates to how the software interacts with the user. And Hardware Interfaces (3.2) talks about the integration of the software and hardware components of the system. (Software Interfaces 3.3) dives more into the connection between software and other systems, databases, operating systems, or software components. Lastly, the end of this section covers Communications Interfaces (3.4), which outlines the communications requirements that will ensure the system operates smoothly between users, application server, and outside services.

4. The Nonfunctional Requirements

The Nonfunctional Requirements focuses more on user experience needed for a smooth transition for the VetCare website. It focuses on the Performance (4.1) of the website such as speed of processing data, or Safety measures (4.2) that is provided to user to safeguard them from data leak and keeping their information private ensuring privacy (Security Requirements (4.3). It then focuses on the useability of the software, such as ease-of-use and reliability, that users notice when using the website, while all adhering to rules that the software must adhere to, which might influence its design and functionality Business Rules (4.5).

5. Other Requirements

Includes any additional requirements not covered in the previous sections, such as legal, regulatory, or internationalization requirements.

6. System Architecture

The System Architecture section includes a view of the Architecture Overview (6.1), often represented through diagrams, to show the major components and how they interact. It also discusses the rationale behind significant architectural choices and how these decisions impact the system’s performance, scalability, and maintainability (Architectural Decisions (6.2))

7. User Interface Design

Offers a detailed description of the user interface design, including layouts, navigation, and other UI elements that will shape the user experience.

8. Appendices

The Appendices provide additional information that adds to the main content of the SRS document. This may include a glossary of terms, acronyms, or abbreviations used in the document, ensuring clarity and understanding (Appendix A: Glossary). The appendices might also contain Analysis Models, such as diagrams or flowcharts, that offer additional insights into the system’s design or functionality. Lastly, this section lists any "To Be Determined" (TBD) items, allowing these issues to be tracked and resolved as the project progresses. The appendices serve as a valuable resource for readers who need more detailed explanations or additional context

An overview of the document:

This SRS is structured to give an overview and introduction to the software, and then to go into detail into its external interfaces, nonfunctional requirements, and system architecture. Details on the user interface design and any supplementary or appendix material are included at the end.

## Product Scope

This product is an intuitive vet clinic web application designed to provide pet owners with a convenient and comprehensive platform for managing their pet’s healthcare which can include:

* **Appointment Scheduling:** Pet owners can easily find and book appointments with their preferred vet, selecting from a database of local clinics and stores. The system allows users to reschedule or cancel appointments if needed, ensuring flexibility and ease of use.
* **Medical Records Management:** The application will utilize a secure database that allows authorized users to retrieve, update, and manage medical records in real-time. Role-based access control ensures that pet owners and veterinarians have an appropriate level of access to the records. This meets the business’ aim to adhere to relevant privacy laws and regulations when handling sensitive data.
* **Multi-Tenancy:** As multiple clinics will be using the system, the application supports multi-tenancy, allowing each clinic to securely access and manage their own records without interference from other clinics. This supports the business strategy of scalability, enabling the website to serve multiple clients.
* **Prescription Management:** Following consultations, veterinarians can issue electronic prescriptions, enabling pet owners to conveniently order medications online with the best prices available. This feature enhances the overall user experience by streamlining the prescription process, improving operational efficiency. By automating prescription handling, the software can reduce administrative work, cost and speed up service delivery.
* **Educational Resources:** Pet owners will have access to a library of educational materials tailored to their pet’s specific needs, helping them provide better care and stay informed about common health issues.

The implementation of these features enables the software to offer valuable services that enhance customer satisfaction. This software aims to address the specific needs of pet owners, building upon the goals of clinics to improve their visibility and attract more clients, all to provide accessible and comprehensive pet care along with a comfortable user experience.

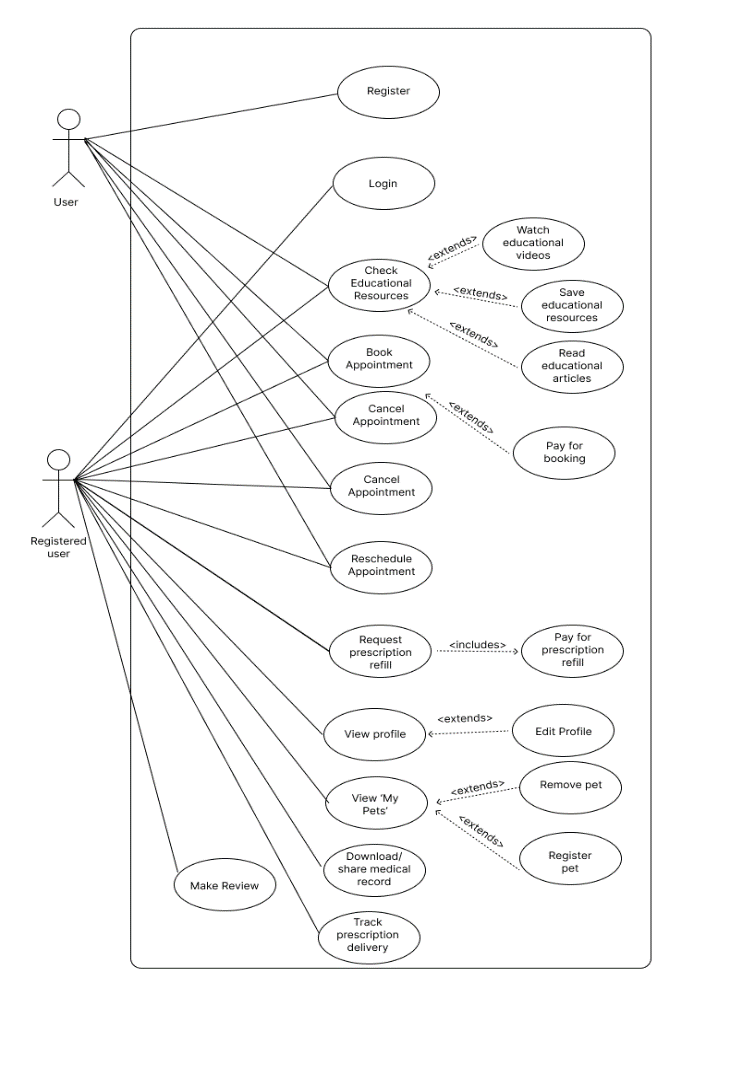
## References

1. Kosutic, Dejan. What Is ISO 27001? Understanding Data Security Standards. <https://advisera.com/27001academy/what-is-iso-27001/>. Accessed 22 Aug. 2024. (Used in 4.3 Security Requirements section)
2. Unsplash. Beautiful Free Images & Pictures | Unsplash. <https://unsplash.com/>. Accessed 23 Aug. 2024. (Used in section 7- User interface design to import free and licensed pictures)
3. ‘Docker Overview’. Docker Documentation, 20 Aug. 2024, <https://docs.docker.com/get-started/docker-overview/>. (Used in 2.4 to cite the definiton of Docker, in ‘Containerisation’ towards the end of 2.4 section)
4. Monolithic vs Microservices Architecture: Pros and Cons | OpenLegacy. https://www.openlegacy.com/blog/monolithic-application. Accessed 25 Aug. 2024.
5. JUnit 5. https://junit.org/junit5/. Accessed 25 Aug. 2024.
6. Thymeleaf. https://www.thymeleaf.org/. Accessed 25 Aug. 2024.
7. Establishing a Connection (The JavaTM Tutorials > JDBC Database Access > JDBC Basics). https://docs.oracle.com/javase/tutorial/jdbc/basics/connecting.html. Accessed 25 Aug. 2024.
8. ‘Spring Boot’. Spring Boot, https://spring.io/projects/spring-boot. Accessed 25 Aug. 2024.
9. Web Content Accessibility Guidelines (WCAG) 2.1. https://www.w3.org/TR/WCAG21/. Accessed 25 Aug. 2024.
10. ‘General Data Protection Regulation (GDPR) – Legal Text’. General Data Protection Regulation (GDPR), https://gdpr-info.eu/. Accessed 25 Aug. 2024.

# Overall Description

## Product Perspective

The VetCare Clinic System is a web application used to address the needs of clients (pet owners) of veterinary clinics. The application provides a platform for pet owners to manage the healthcare of their pets, which includes medical records, scheduling appointments, accessing medical records, and managing prescriptions. There is no pre-existing follow-on member / product family of this product, nor a replacement for any certain existing systems. Rather, the VetCare Clinic System represents a new, self-contained product, developed entirely from scratch. As such, the application itself is the main system, which will be facilitated by external components and systems such as the vet clinic databases.



## Product Functions

**Appointment Management**

* Booking System: Facilities the scheduling, rescheduling and cancellation of appointments
* Appointment Tracking: Provides interfaces to view and manage upcoming and past appointments efficiently

**Viewing Medical Records**

* Records Access: Ensures the user can access the pet’s complete medical history
* Record Sharing: Allows the sharing of the medical records to specific veterinary practitioners

**Prescription Management**

* Buying Prescriptions: User has direct access to buy certain prescriptions for their pet. This will be linked with the medical records as well
* Prescription Details: Provides access to prescription details, including dosage and instructions for usage.

**Educational/Helpful Resources**

* Resource Library: Provides the user with a page of videos, guides and forums where the user can see guides and personal reviews from other pet lovers on how to properly care for and maintain proper health of their pets.
* Regular Updates: Regular updates to this page will be conducted to ensure that users are provided with the latest research and practices.
* FAQs: Users can view FAQs, so they are able to navigate and understand the website clearly.

**User Account and Profile Management**

* Customization: Users can manage their own profiles and well as their pets' profiles
* Settings: Allows users to set and modify notification preferences (i.e. upcoming appointments)
* Order Tracking: Users can track orders for the prescriptions they have ordered

**Notifications and Reminders**

* Reminders: Users are sent reminders for upcoming appointments as well as when the pets should be taking their medication
* System Alerts: Notifies the users about upcoming updates as well as new educational recourses

**Payment System**

* Transactions: When users are going to buy medication for their pet, they will have a secure payment point and they will be sent to a different page for this
* Payment Options: User will be provided with multiple payment options and after will receive a receipt.

## User Classes and Characteristics

The VetCare system will serve mainly the pet owner user class, with consideration to veterinary professional. Each with distinct roles, needs, and access levels. The primary user classes identified are as follows:

**Pet Owners:**

* + Characteristics: Users with varying computer proficiency. Pet owners are interested in managing their pet’s healthcare with as much convenience as possible. The age group ranges from young adults to elderly individuals.
  + Frequency of Use: Moderate to High; they will interact with the system as needed.
  + Key Features Used: Appointment scheduling, medical records access, prescription management, and educational resources.
  + Technical Expertise: Low to moderate; the system should not expect users to be experts and should be intuitive and easy to use with clear instructions.
  + Security & Privilege Levels: Pet owners will have limited access, being restricted to their own pet’s data. To access this data, they will require to login to their accounts.

**Veterinary Professionals:**

* + Characteristics: These users include veterinarians, assistants and receptionists who have technical knowledge. Their needs would be access to medical records, viewing appointments and being able to upload prescriptions.
  + Frequency of Use: High; veterinary professionals will interact with the system daily to update pet medical records, check appointments, and issue electronic prescriptions.
  + Key Features Used: Medical records management, appointment viewing, prescription management
  + Technical Expertise: High; they are expected to be comfortable navigating and utilizing various system features efficiently.
  + Security & Privilege Levels: Veterinary professionals will have access to higher levels with privileges which include features that enable the key features aforementioned

## Operating Environment

**Hardware Platform:**  
**Servers:**

Amazon Web Services (AWS) cloud-based servers will host the VetCare website. It gives clients freedom in selecting resources like CPUs, memory, and storage, which may be dynamically assigned according to the demands of the application. It will be simpler to design the application in tandem using Docker, which enables applications to build, test, and deploy rapidly.

**Operating System:**

**Server-Side OS:**

The backend of the VetCare application will operate on a Linux-based operating system, such as Ubuntu Server. The operating systems are best for stability, security, and compatibility with the chosen software stack (Spring Boot, MySQL, Docker).

**Client-Side OS:**

The VetCare application will be compatible with major operating systems used by end-users, including:

* Windows: Windows 10 and 11
* macOS: Versions such as Big Sur, Monterey, or later
* iOS: iOS 13 or later for iPhone and iPad users
* Android: Android 8.0 (Oreo) or later

**3. Software Components:**

**Web Server:**

* The application will use Apache HTTP Server or Nginx as the web server. These are robust, reliable, and capable of handling high volumes of web traffic efficiently.
* The web server will be configured to work seamlessly with the Spring Boot framework, ensuring dynamic and static content works. (CSS, JavaScript, images).

**Database:**

* VetCare will utilize MySQL for data storage and retrieval. MySQL is chosen for its performance, scalability, and robust support for transactional operations.

**Backend Framework:**

For the back-end framework, Spring Boot will be used to build the core of the application. It provides ease in creating applications that can handle a large network of users with its tools and features that it provides. It will handle the backend logic by processing user requests, handling data, and performing calculations.

**Front-end Framework:**

HTML, CSS, and JavaScript will be used for the front-end development. The mixture of these three languages will help to create the best possible user experience on desktop and mobile platforms.

**Continuous Integration (CI) and Continuous Deployment (CD)** **tools:**

CI and CD processes will be managed using GitHub Actions. It is a helpful tool to ensure code has minimal errors from merge conflicts and reduce manual correction of code. It also does a code quality check and most importantly runs automated testing to ensure new changes don’t break existing code.

**Containerization:**

Docker will be used to containerize the application, making it easier to deploy across different environments (development, testing, production). This ensures a reduction in the delay between writing code and running it in production.

## 2.5 Design and Implementation Constraints

The development of the VetCare Online Vet Clinic Management System will face several constraints that may limit the options available to the developers. Below are the detailed constraints:

1. Regulatory Policies

Data Privacy Regulations:

* Users' personal information and sensitive personal data, such as pet medical records, will be handled by VetCare. Consent procedures must be put in place and users must be made aware of data collecting methods in compliance with regulatory policies

Payment’s security:

* The VetCare system will include an online payment feature which mandates strict security protocols for handling and storing payment information, including encryption, and secure authentication.

2. Hardware Limitations

Server Resource Constraints:

* While VetCare will be hosted on cloud infrastructure, there may still be limitations in server resources such as CPU, memory, and storage, especially for clinics that may opt for a more cost-effective hosting plan. These resource limitations could impact the performance of the application during peak usage periods.

Client Device Limitations:

* The application has to be optimised to function well on low-resource devices, including outdated smartphones or tablets with a little CPU, RAM, and/or little storage. Due to this limitation, a front-end architecture that is lightweight with minimal resource consumption

External Service Dependencies:

* The application will rely on AWS S3 for cloud storage and a possible constraint will be rate limits, costs, and data retrieval times.

3. Real-Time Data Processing

* Real-time data processing for tasks like appointment scheduling and product availability updates must be supported by the system. This places limitations on the system's ability to handle several jobs at once.

4. Security Considerations

Data Encryption:

* Sensitive information must always be encrypted. This requirement may impose performance constraints, especially for tasks requiring big data sets or repeated encryption/decryption cycles.

## 2.6 User Documentation

**Quick Start Guide**:

* Aimed at new users, this is a guide that offers easy-to-follow instructions on the basic operations needed to get started with the website, such as account creation and first-time login.

**FAQs Page**:

* A standalone page on the website that allows users to view the most common questions and challenges they might encounter.

**Video Tutorials**:

* These are step-by-steps videos which will help the user navigate through the website so they can learn about the features that are present and how to use all available functions.

## 2.7 Assumptions and Dependencies

**Assumptions**

* The system assumes the user will want to receive notifications via email or SMS about appointments, as well as share medical records by downloading it themselves via email
* Assumes that the user will want the option to pay for their bookings either online or in person
* The accuracy of the application’s database is up to par, and is updated regularly
* Local clinics are inclined to provide access to their data/APIs and thereby willing to integrate them with the VetCare Platform.

**Dependencies**

* The platform will be dependent on the databases of the veterinary clinics, and thus any downtime or maintenance of this dependency directly affects the usability of the platform
* Hosting and server infrastructure (AWS) are also dependencies of the platform, such that any issues impacting the AWS infrastructure will affect the hosting of the website

# External Interface Requirements

## User Interfaces

The VetCare system’s UI is intuitive and user-friendly, focusing on the Pet Owner and Veterinary Professional interfaces, and is optimized for desktop use.

**Interface Overview & Components:**

* Pet Owner: Manages appointments, views medical records, and accesses educational resources.
* Veterinary Professional: Manages patient records, schedules, and prescriptions.

**Design Standards:**

* Consistency: Consistent button styles, fonts, and colors (detailed info and color palettes can be found in the wireframes)
* Responsiveness: Designed for desktops, so it should be adaptable to various screen sizes.

**Screen Layout and Navigation:**

* Layout: Clear structure with a header including all the links to main pages and account, main content area, and footer with the quick links and contact details.
* Navigation: Top navigation bar.
* Buttons: Consistent placement of save/next and cancel/back buttons for relevant forms.

**Error Handling:**

* Messages: Error messages for relevant form fields e.g. incorrect form inputs, scheduling conflicts
* Validation: Form validation to ensure correct input and action (visible system status)

## Hardware Interfaces

The VetCare web application is designed to be accessed via web browsers on desktop and laptop computers. The following outlines the logical and physical characteristics of the interfaces between the software and the hardware components of the system:

# Supported Device Types:

# Desktop and Laptop Computers:

# The application is optimized for use on standard desktops, laptop computers, mobile devices, tablets, and other devices.

# Supported operating systems include Windows, macOS, and Linux distributions that can run modern web browsers such as Chrome, Firefox, Safari, and Edge.

# Nature of Data and Control Interactions

# User Input Devices:

# Users will interact with the application through standard input devices such as keyboards. No specialized hardware is required for interaction with the application.

The software captures input data from these devices to manage tasks such as appointment scheduling, record retrieval, and form submissions.

* **Display Hardware**:
  + The application is designed to render content optimally on screens with a minimum resolution of 1024x768 pixels, ensuring a clear and accessible interface on typical desktop and laptop monitors.
  + It is not software to be downloaded and installed on a device, therefore, it does not need any specific hardware requirements to have, before using it.

#### Network Requirements:

* **Internet Connection:**
  + Since VetCare is an online web application, a stable and active internet connection is essential for accessing the application. Users will need either a wired (Ethernet) connected to device directly or wireless (Wi-Fi) connection to the internet, provided through a router or network infrastructure.

## Software Interfaces

The VetCare web application interacts with several software components to ensure smooth functioning, data management, and communication. This section details the key connections between the system and other software components, including databases, operating systems, tools, libraries, and integrated commercial services.

**Operating Systems:**

* **Operating System:**
  + This application will be hosted on cloud-based servers, specifically utilizing Amazon Web Services (AWS) EC2 instances.
  + The VetCare application will be running on a Linux-based operating system, specifically Ubuntu Server 23.04 LTS or later. This choice is made due to Ubuntu’s stability, security features, and compatibility with a wide range of server hardware.
  + **Server Configuration:** The server will be configured with at least 4 virtual CPUs (vCPUs) and 8 GB of RAM. As the application scales, the server resources can be increased by adding more vCPUs and RAM, or by deploying additional instances through an auto-scaling group.

**Data Storage:**

* Using SSD-backed storage from AWS (Amazon Web Services) as part of its cloud infrastructure will provide optimal speed and dependability for the VetCare application.
* To handle real-time data, like appointment scheduling and medical records, this cloud-based storage system provides enhanced I/O operations and faster data access.
* Instead of requiring physical hardware to be on-site, the application can effectively handle fluctuating loads by utilizing AWS's scalable storage choices, guaranteeing that data is kept safe and easily accessible even during periods of high user demand.

**Database Management System (DBMS):**

* **Database:**
  + The application will utilize MySQL 8.0 as the primary database management system. MySQL is chosen for its reliability, performance, and support for complex queries and transactions.
  + **Data Interaction:** The VetCare application will use the Spring Data JPA (Java Persistence API) to interact with the MySQL database. This interaction will include operations such as CRUD (Create, Read, Update, Delete) for managing user information, pet records, appointment schedules, and transaction logs.
  + **Data Exchange:** Data sent to the database includes user-submitted forms, appointment details, and updates to medical records. Data retrieved includes user profiles, appointment history, and prescription details.

**External Libraries and Frameworks:**

* **Spring Boot (v3.3+):** The backend of the VetCare application is built using Spring Boot, which provides a comprehensive framework for building and running web applications. Spring Boot manages the application’s dependency injection, security configurations, and web layer abstraction.
* **Thymeleaf (v3.1+):** Thymeleaf is used for server-side rendering of HTML templates. It processes the data provided by the backend and dynamically generates web pages that are sent to the client’s browser.
* **JDBC Library:** JDBC will be used to manage the interactions between the Java application and the MySQL database. It enables the execution of SQL queries directly within the Java code, offering flexibility and control over the database operations.
* **Maven**: The VetCare program uses Maven as both a dependency manager and a build automation tool. It controls all project dependencies, guaranteeing that the appropriate framework and library versions are applied consistently in various development contexts. The build process, which includes code compilation, test execution, and application packaging for deployment, is likewise automated by Maven.
* **JUnit:** JUnit offers a framework for creating and executing automated tests to make sure certain code segments—like methods or classes—work as intended. Therefore, the JUnit library is for unit testing in this application. Furthermore, Test-driven development (TDD) techniques are also supported by JUnit library, which helps in the early detection of bugs in codes throughout the development process.

**Data Format and Data Sharing**:

* **Data Format:** JSON is the primary data format used in API communications, enabling platform-independent data exchange between the client, server, and third-party services.
* Throughout software components, information about user sessions, appointment data, and user profiles will be shared. To ensure consistency and integrity, this data will be handled in a centralized database and accessible via secure APIs.

## Communications Interfaces

To guarantee smooth connection between users, the application server, and outside services, the VetCare program needs several different communication features. The main needs for communication, such as data transfer techniques, security, message formatting, and protocols, are covered in this section.

**Web Browser Communication**

* **Purpose:** Users will access the VetCare application through modern web browsers. The application’s frontend, rendered by Thymeleaf, communicates with the backend using standard web technologies.
* **Compatibility:** The application will support major web browsers like Chrome, Firefox, Safari, and Edge, ensuring compatibility across different platforms and operating systems.

**Network Server Communications Protocols:**

The core networking protocols used in this server are TCP/IP and HTTPS, which encrypt data transmissions using SSL/TLS to protect sensitive data including user credentials (username, email, password) and pets’ medical records. HTTPS also provides data exchange security. This guarantees dependable, safe connection between the clients and the server.

**Electronic Forms**

* The application will utilize electronic forms for user input, including registration, appointment scheduling, and medical record updates.
* User input will be validated on the client-side using JavaScript, and on the server-side using backend validation methods provided by Spring Boot.
* All form data submitted by users will be transmitted over HTTPS to ensure it is securely encrypted in transit.

# Nonfunctional Requirements

## Performance Requirements

# Response Time:

* Requirement: The system must provide a response time of less than 3 seconds for all key user interactions, such as booking appointments, retrieving medical records, and issuing prescriptions.
* Rationale: Quick response times are essential to maintain user satisfaction, especially for pet owners who expect a seamless experience when managing their pets' healthcare.

# System Availability:

* Requirement: The system must be available 99% of the time.
* Rationale: High availability is important to ensure that in cases of emergency or urgent care, both pet owners and veterinary professionals can access the system at any given time.

# Scalability:

* Requirement: The system must be able to handle up to at least 1000 simultaneous users without any significant change in performance.
* Rationale: Since the VetCare system will be used by many clinics and pet owners, it is important that the website can scale effectively to accommodate a growing number of users.

# Data Processing:

* Requirement: For features such as medical record management, which include inserting new records and retrieval of existing ones, the data must be processed in less than 5 seconds.
* Rationale: The system must be efficient in data processing to ensure that veterinary professionals can access and update patient information promptly to use for treatment.

**Real-Time Notifications:**

* Requirement: The system should deliver real-time notifications (e.g., appointment reminders, prescription alerts) within 5 seconds of the action.
* Rationale: Real-time notifications are necessary to keep both pet owners and veterinary professionals informed in cases of changes in scheduling and prescription uploading, to reduce the likelihood of missed appointments or required medication doses.

## Safety Requirements

If any damage is done to the web application or database due to a bug or a crash in the system, a previous working version of the application and/or database can be restored and any changes to the system can be reapplied prior to the damage.

## Security Requirements

**[Data Security and Privacy]**

VetCare must ensure that user data is protected and always secured. This includes user login information which should be encrypted using industry-standard technology such as AES-256, and guarantee that information such as user profile information, pet medical records, and prescriptions are only accessible to that user and any relevant parties such as vet clinics.

Failure to uphold these safety requirements runs the risk of user’s personal data being breached, including sensitive information such as user’s login and payment information.

**[Safety Certifications and Regulations]**

Regulations within the safety certification, ISO 27001, should be followed to ensure that VetCare maintains high standards of online safety and reliability. The safety certification, developed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), is a globally recognized framework that lays out a set of standards that Information Security Management Systems must follow so that user’s personal, and valuable, information is protected. The ISO 27001 contains 3 main principles which include **Confidentiality;** where only authorized users have the right to access information, **Integrity;** where only authorized users can change certain information, and **Availability**; where information must be accessible to authorized users when required (Kosutic, 2023).

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some things to consider are adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

#### Reliability

* We aim to have the website running almost all the time (excluding downtime for updates) to ensure that all users have access to the systems, thus avoiding disruptions.
* As for the developers, they will ensure the system maintains minimal errors across all transactions and interactions

**Usability**

* We will prioritize a simplistic design to ensure that new users can perform basic tasks such as scheduling an appointment or accessing medical records without prior knowledge of the system.
* We aim to comply with WCAG 2.1 AA standards, ensuring it is accessible to all types of users.

**Performance**

* The developers will ensure that customers will be provided with a wait time of no more than 3 seconds given all user interactions under normal conditions, enhancing the user experience
* The website will be able to handle at least multiple simultaneous user sessions without hindering performance

**Maintainability**

* The system's architecture will be modular, allowing for easy updates and maintenance.
* Proper documentation will be provided, including code comments and guides to facilitate easy understanding.

**Portability**

* We aim to ensure that the website can be reached on all operating systems including Windows, macOS, and Linux.

**Security**

* All user data will be encrypted both at rest and in transit using industry-standard protocols such as TLS.
* The website will follow relevant legal frameworks, including GDPR

**Testability**

* Developers will ensure proper testing, including unit, integration, and system tests, are conducted to ensure that each component functions correctly both independently and when integrated.

## Business Rules

**Individual Roles**

* **Administrators**: Have full access to the system, including the ability to manage user roles, access all records and configure system settings.
* **Veterinarians**: Can access and update medical records, view appointment schedules and manage prescriptions. Veterinarians are also allowed to input medical notes so the user can view them.
* **Clinic Staff**: Limited to scheduling appointments, managing client information and handling billing and payments. Clinic staff do not have access to detailed medical records unless explicitly authorized by a veterinarian.
* **Users**: Can book, reschedule, or cancel appointments, view their pet’s medical history, request prescription refills and access educational resources. Pet owners cannot access medical records of other pets or any administrative functionalities.

**Data Privacy and Confidentiality**

* Access to pet medical records is strictly limited to authorized personnel (veterinarians and authorized clinic staff) to maintain privacy and comply with legal standards.
* Sharing medical records with other veterinary clinics or professionals requires consent from the pet owner, except in emergencies where animal health is at risk.

#### System Availability

* Scheduled maintenance will be communicated in advance with users to minimize disruption and ensure system reliability. These will be conducted during off-peak hours.

**Compliance and Legal Adherence**

* The website will follow all regulations and data protection laws.
* All users must agree to the terms of service and privacy policy before accessing the system.

# Other Requirements

TBD.

# System Architecture

< Summary of the fundamental decisions and solution strategies that shape the architecture. Can include technology, top-level decomposition, approaches to achieve top quality goals and relevant organizational decisions.>

VetCare is designed in mind to provide a fast, responsive, and reliable web application for pet owners. There are several fundamental decisions and solutions that shape the architecture for VetCare.

The technology choices include utilising Java with the Spring Boot framework, along with Maven for managing dependencies, and using MySQL for VetCare’s database management system.

VetCare adopts a monolithic architecture with a 3-tier architecture pattern which comprise of the user interface layer which is responsible for handling the presentation of the web application, the business logic layer which manages tasks like user management and data processing, and the persistence layer which handles the interactions with VetCare’s database.

A diagram of a computer

Description automatically generated

## 6.1 Architecture Overview

<Static decomposition of the system, abstractions of source-code, shown as hierarchy of white boxes (containing black boxes), up to the appropriate level of detail.>

The VetCare web application is organized into a 3-tiered monolithic architecture design comprising of the user interface layer, the business logic layer, and the persistence layer. Each layer is presented with a white box that consists of components and functions - which are presented as black boxes - that all coexist and communicate with each other for VetCare to be functional.

Within the user interface layer, it contains components that handle the presentation of the web application and its user interactions. For example, the controllers manage the HTTP requests from users and map those interactions to the corresponding services from the business logic layer. For instance, ProfileController.java is responsible for displaying the user’s profile page in order to manage their information. The user interface layer also contains templates and resources in the form of HTML & CSS files to formulate a consistent and aesthetic user interface and user experience for VetCare.

The business logic layer is responsible for the core functionalities of VetCare. This layer processes the user’s inputs from the application and communicates with the user interface layer and the persistence layer to produce an output. For example, this layer contains services such as the AppointmentService.java which handles user’s appointments, including scheduling and notifications. The business logic layer also contains Data Access Objects (DAOs) which are objects that are passed to the persistence layer.

Finally, the persistence layer is composed of components that handle the interactions with VetCare’s MySQL database. This includes several repositories such as the UserRepository.java which encapsulates the interface for creating, reading, updating, and deleting (CRUD) operations on ‘User’ entities. Speaking of, this layer also contains entities which represent their respective tables within the database.

A screenshot of a computer diagram

Description automatically generated

## 6.2 Architectural Decisions

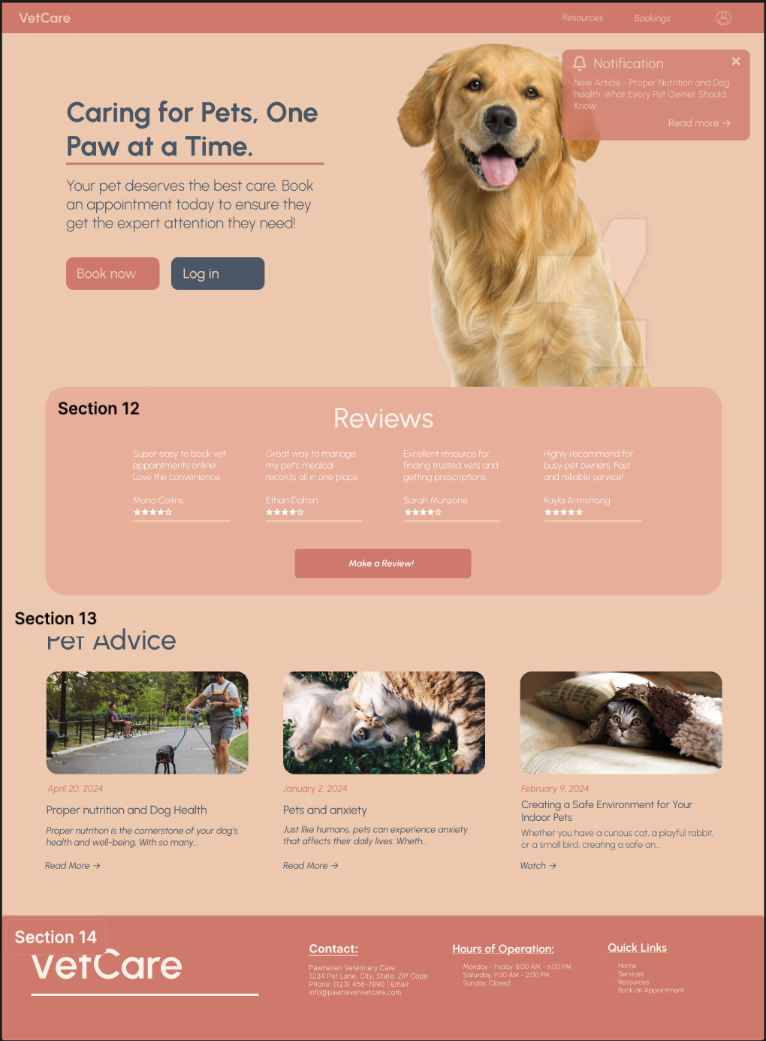
<Important, expensive, critical, large scale or risky architecture decisions including rationales.>

The decision to build VetCare with a 3-tiered monolithic architecture is due to its simplicity and homogenous structure, which allows the team to quickly develop the web application. Another reason for this decision is that it’s also efficient and easier to debug as a small team since the web application is encapsulated as one big program

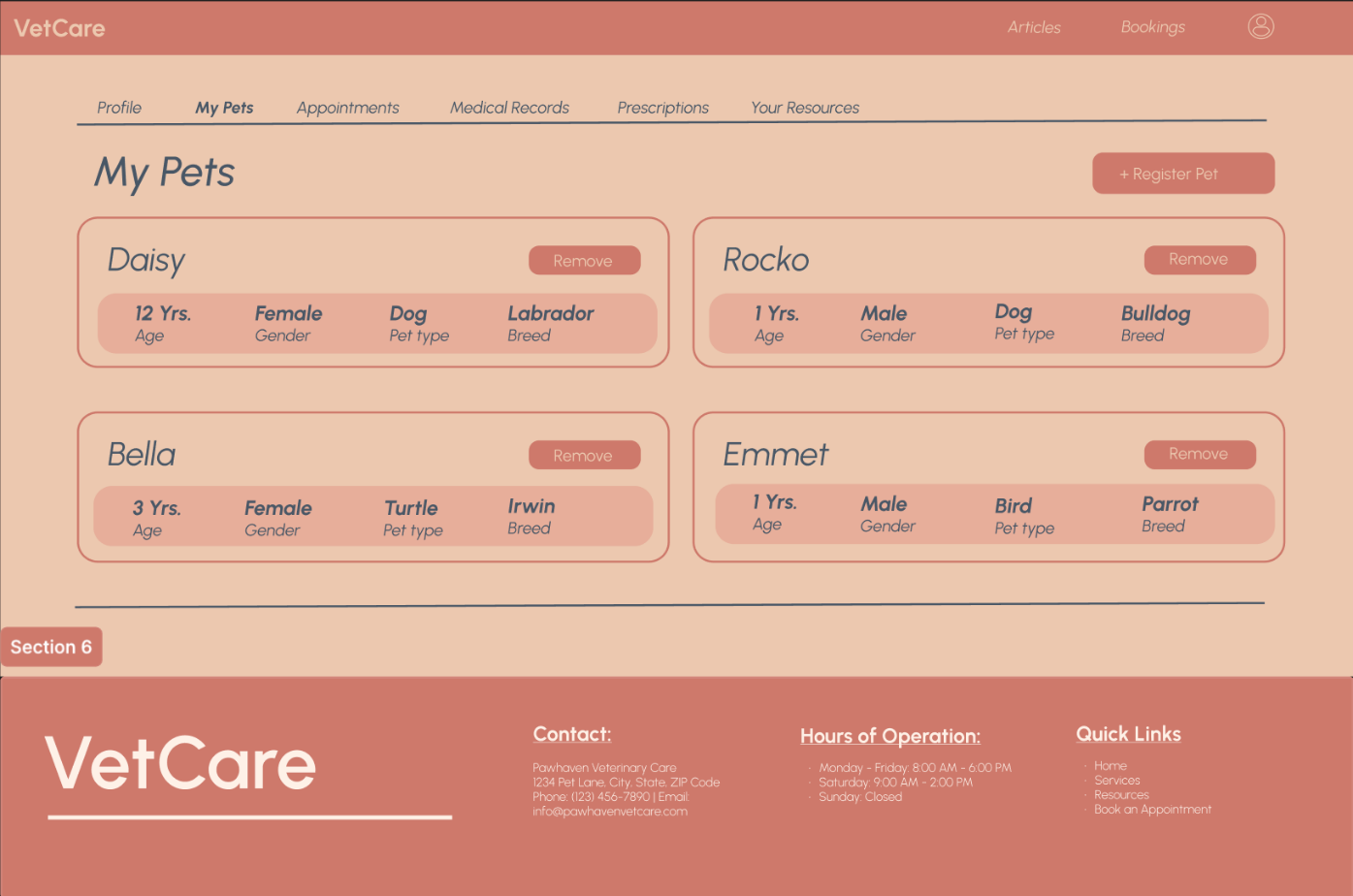
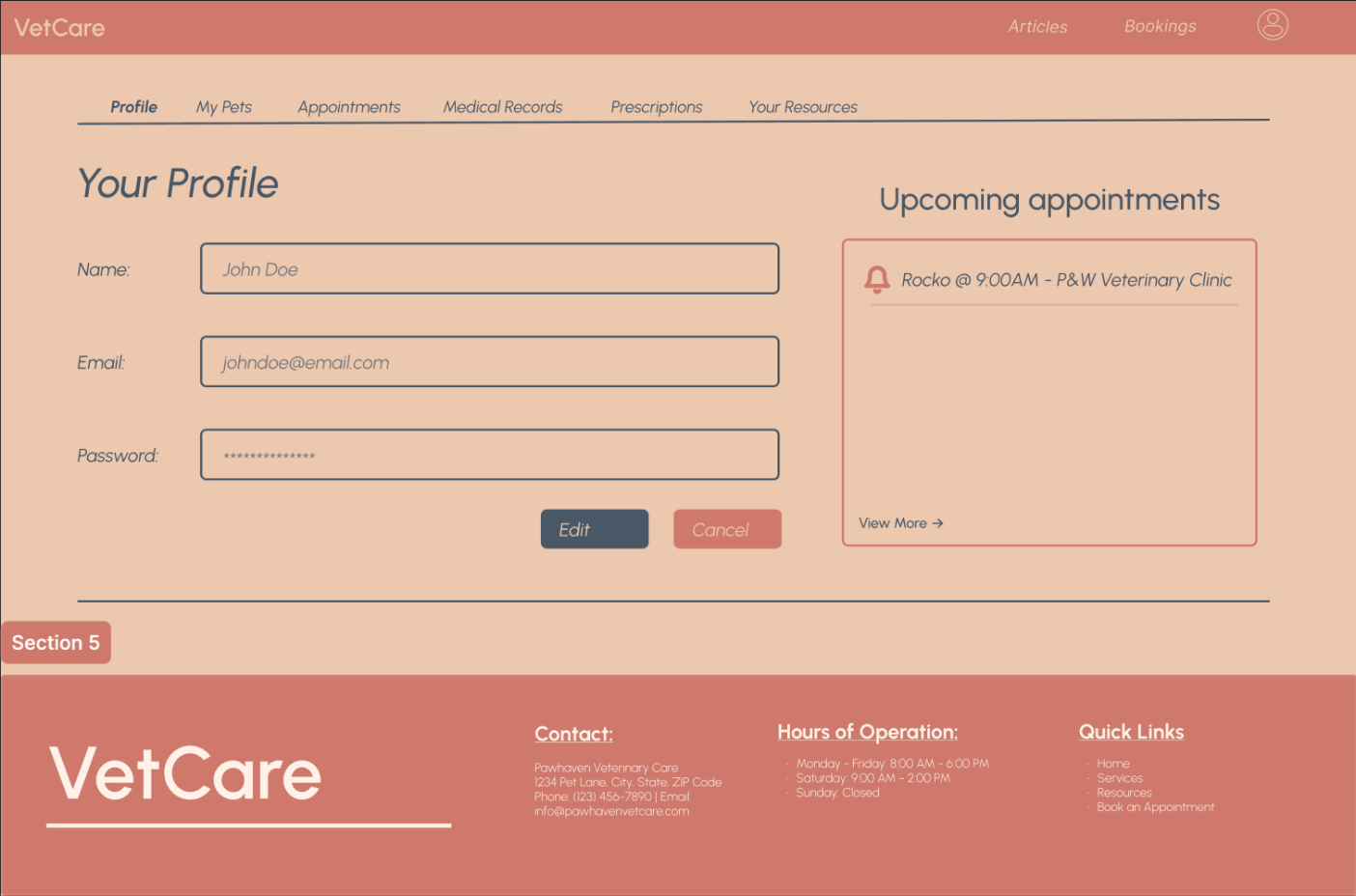
However, utilizing this architecture design may also come with risks. For example, it is more difficult to maintain and scale if the development team wants to add more features to VetCare in the future. Another example is that functions within VetCare are highly dependent on one another. Since all the functions are contained in one big program, if one function stops working, other functions of VetCare may stop working too.

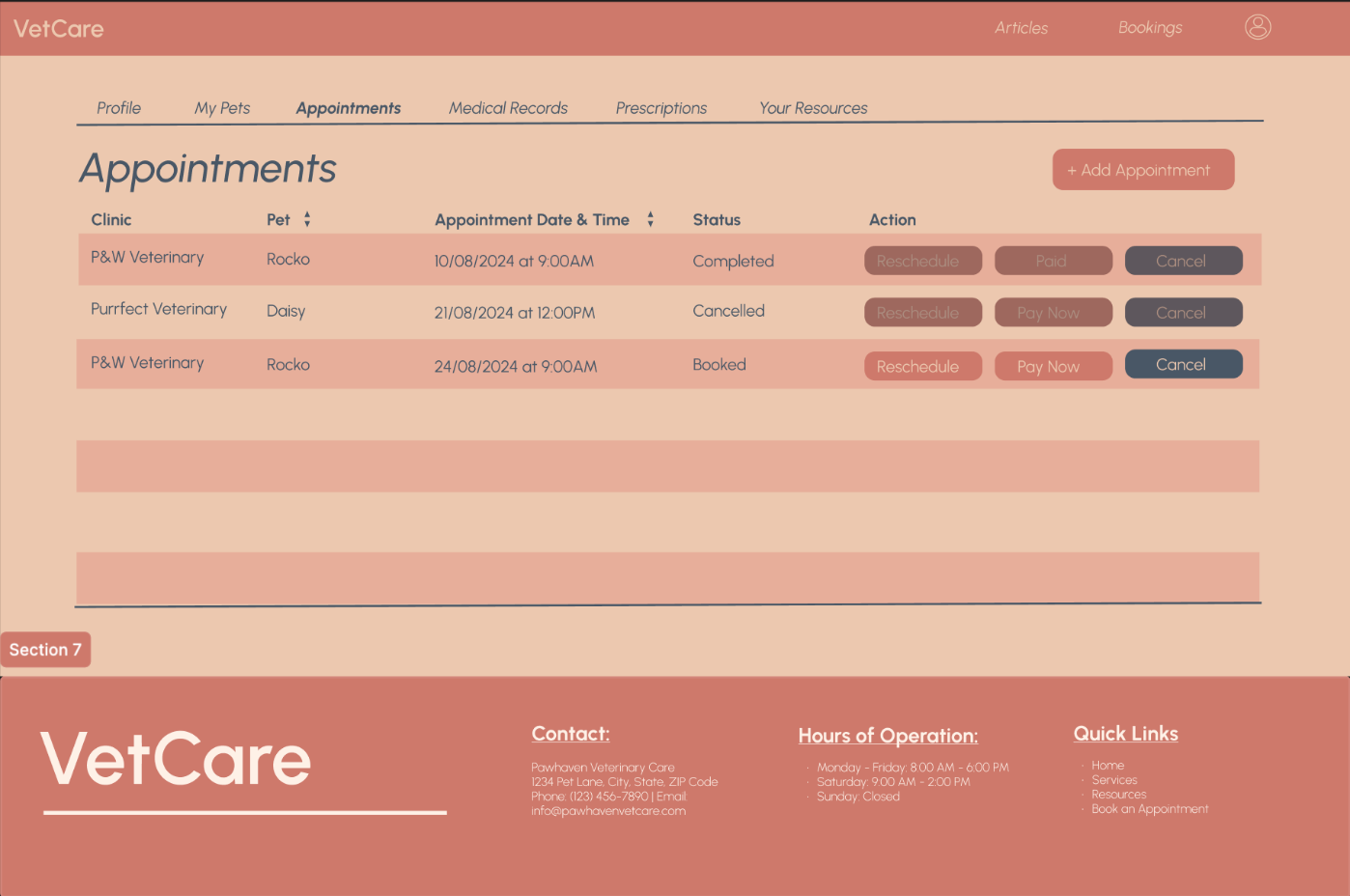
# User Interface Design

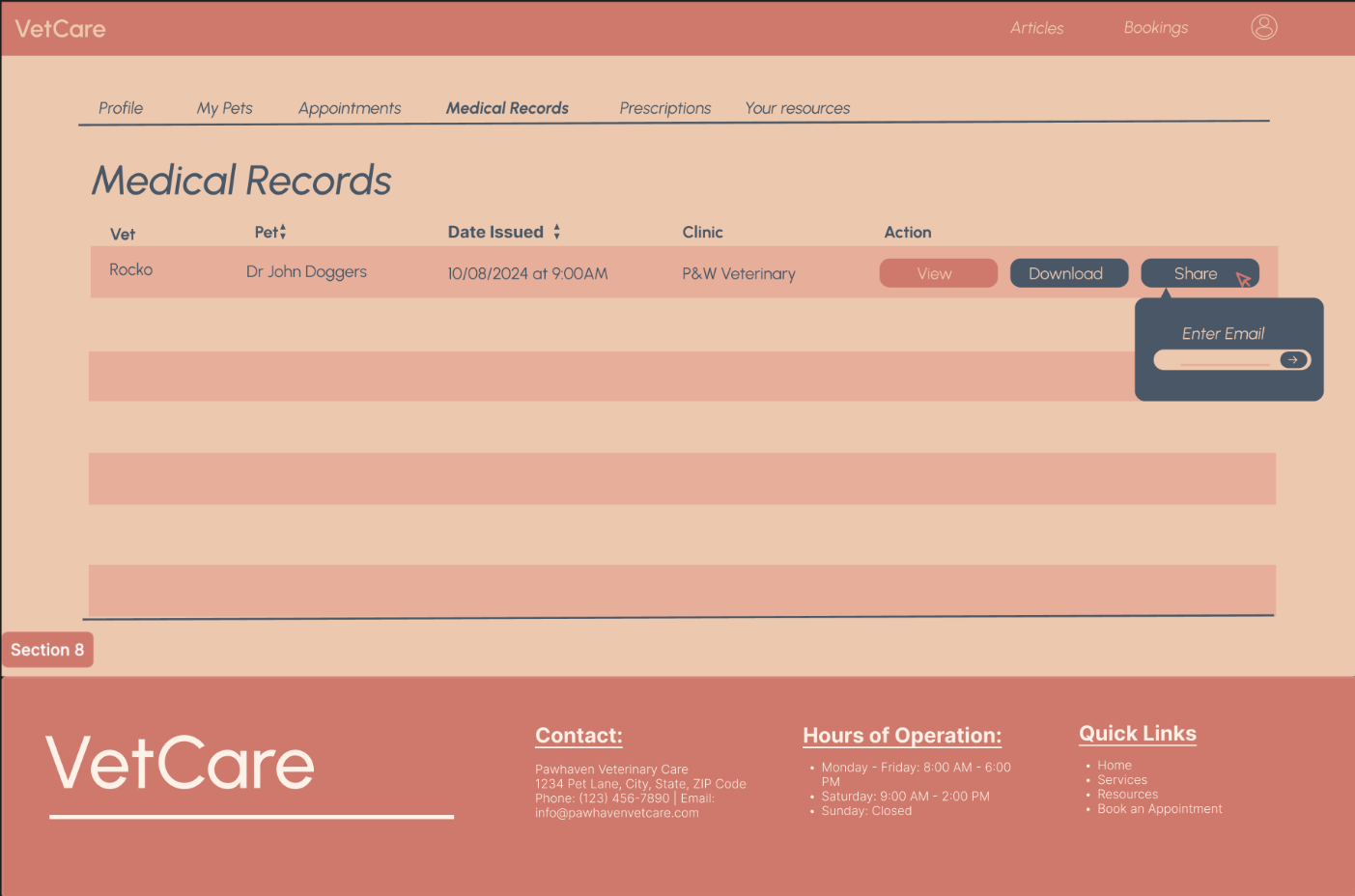
**Landing Page:**

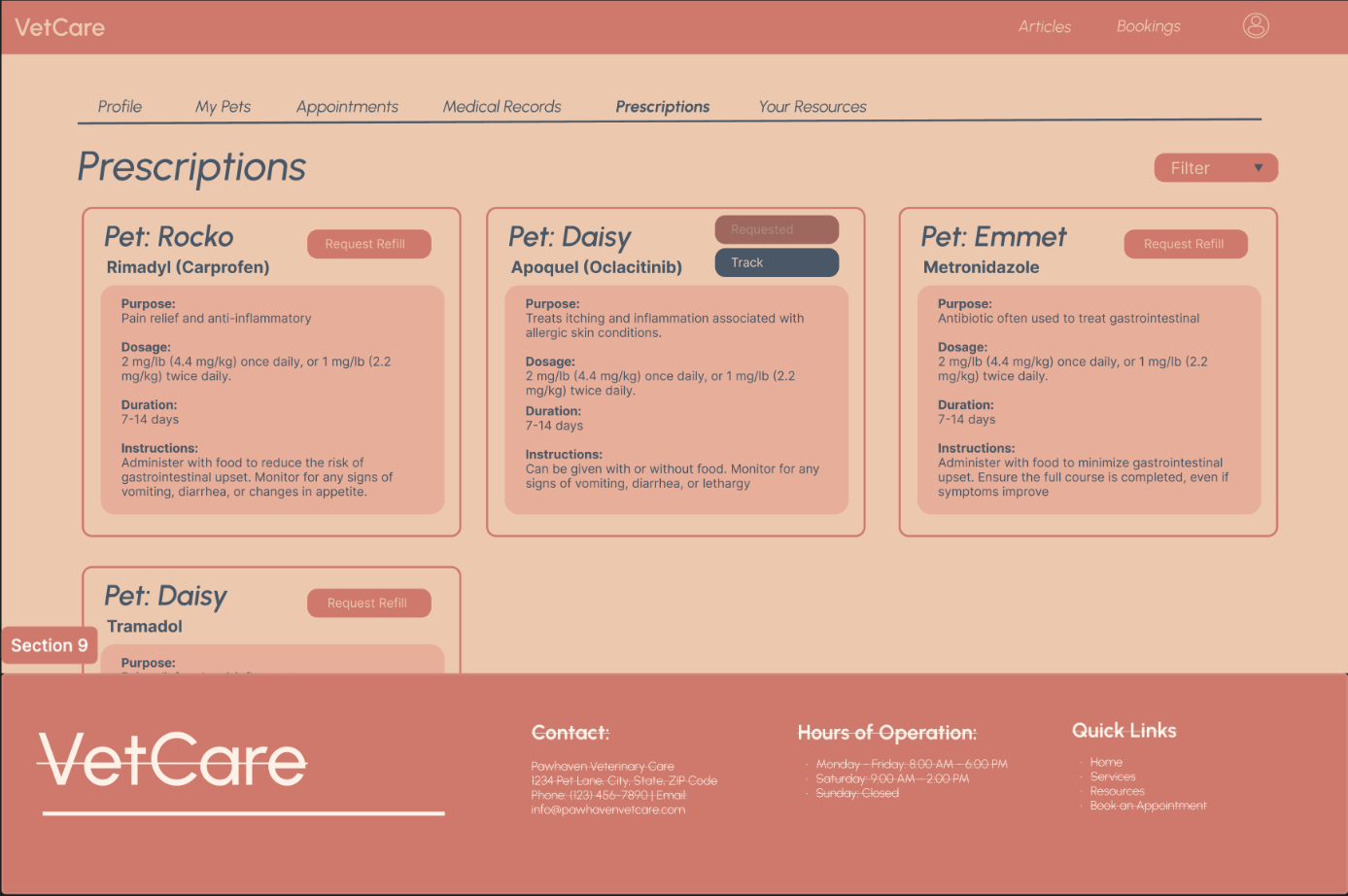


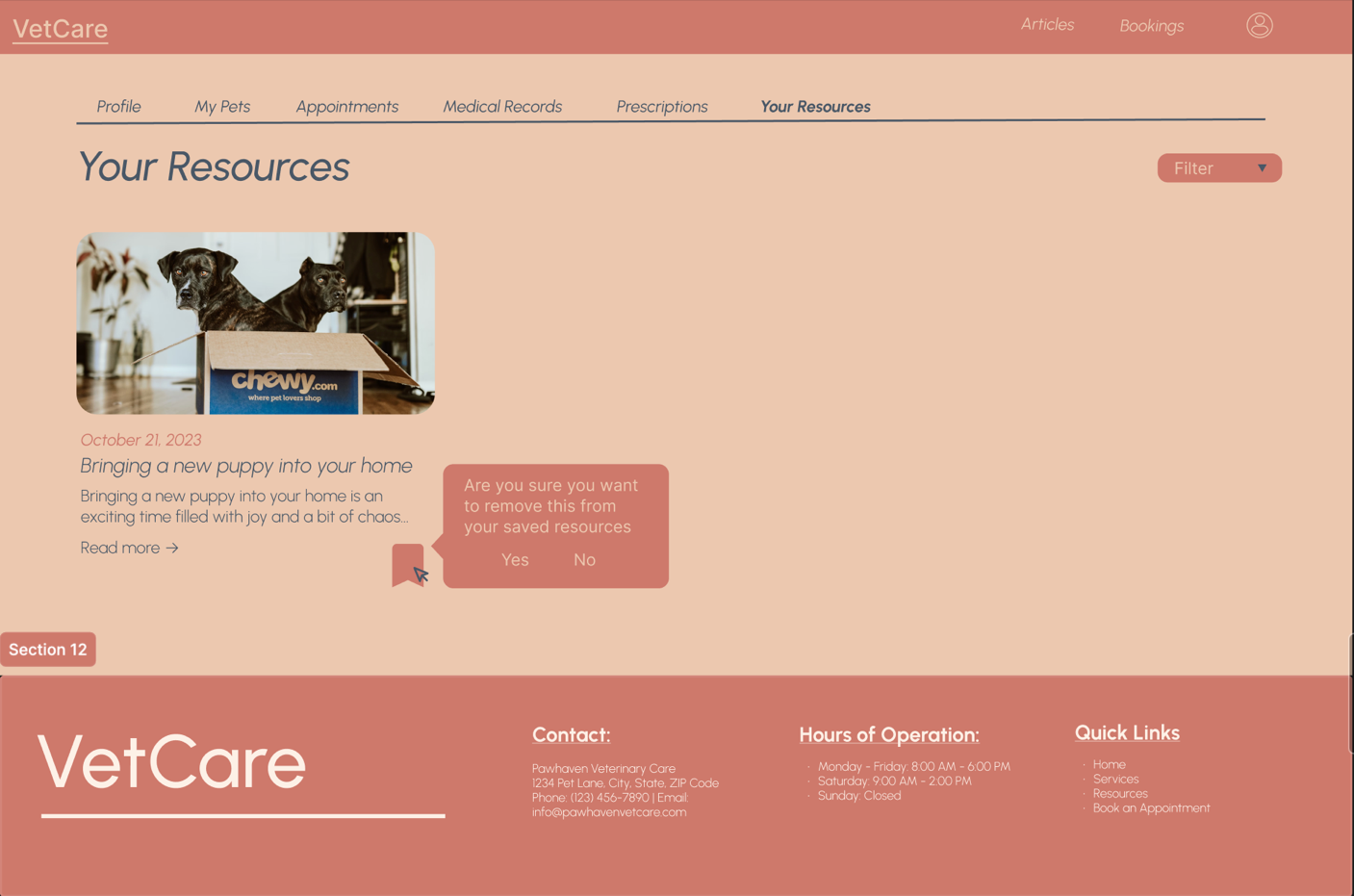
**Profile + Profile Tabs:**





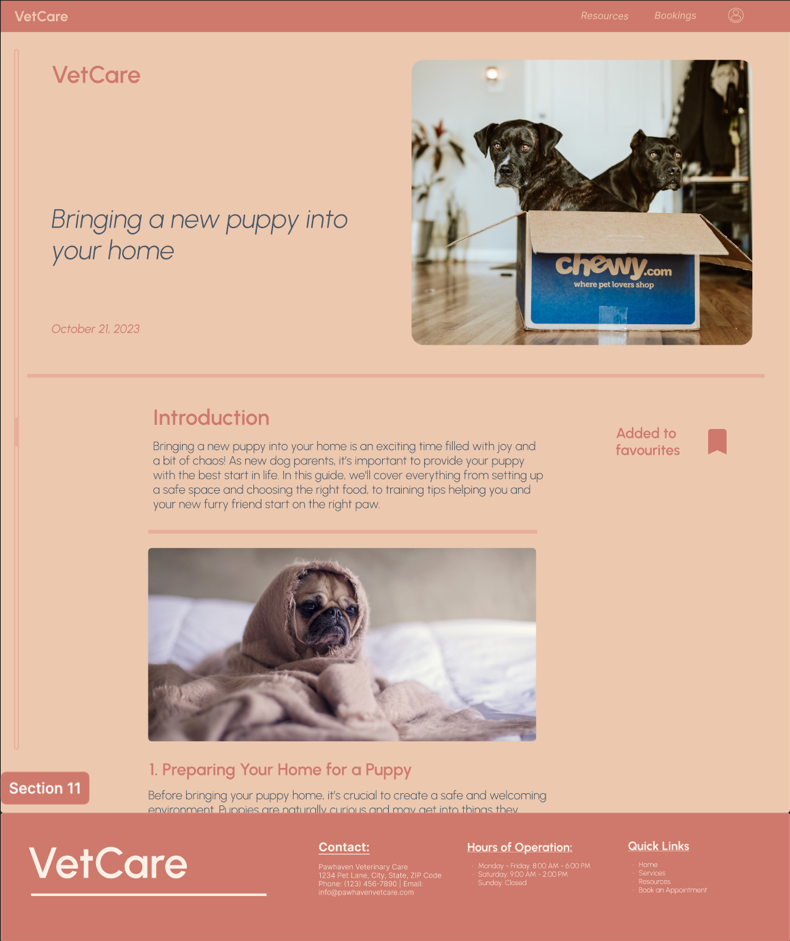




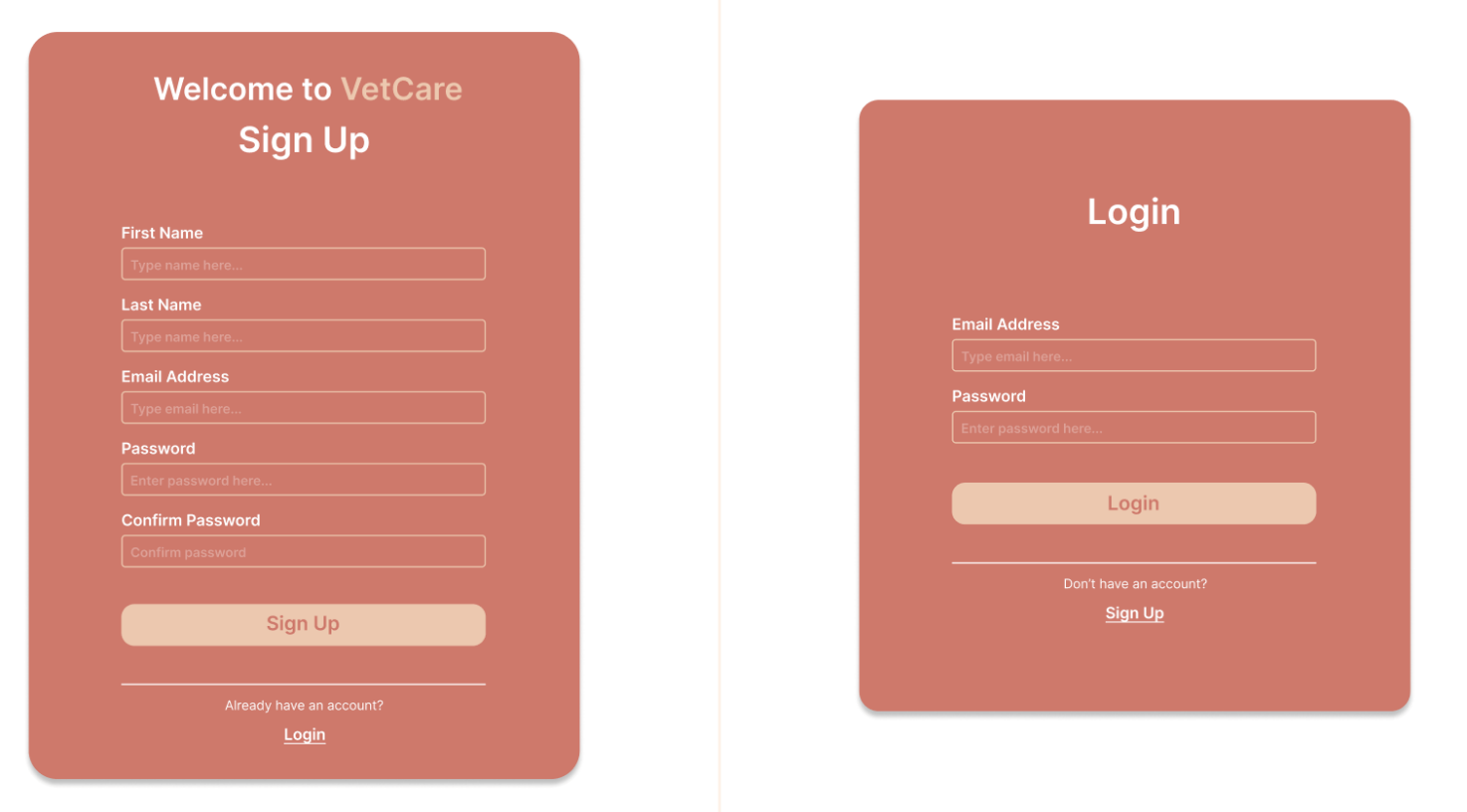


Educational Resources Page + Educational Resources Page > Article:

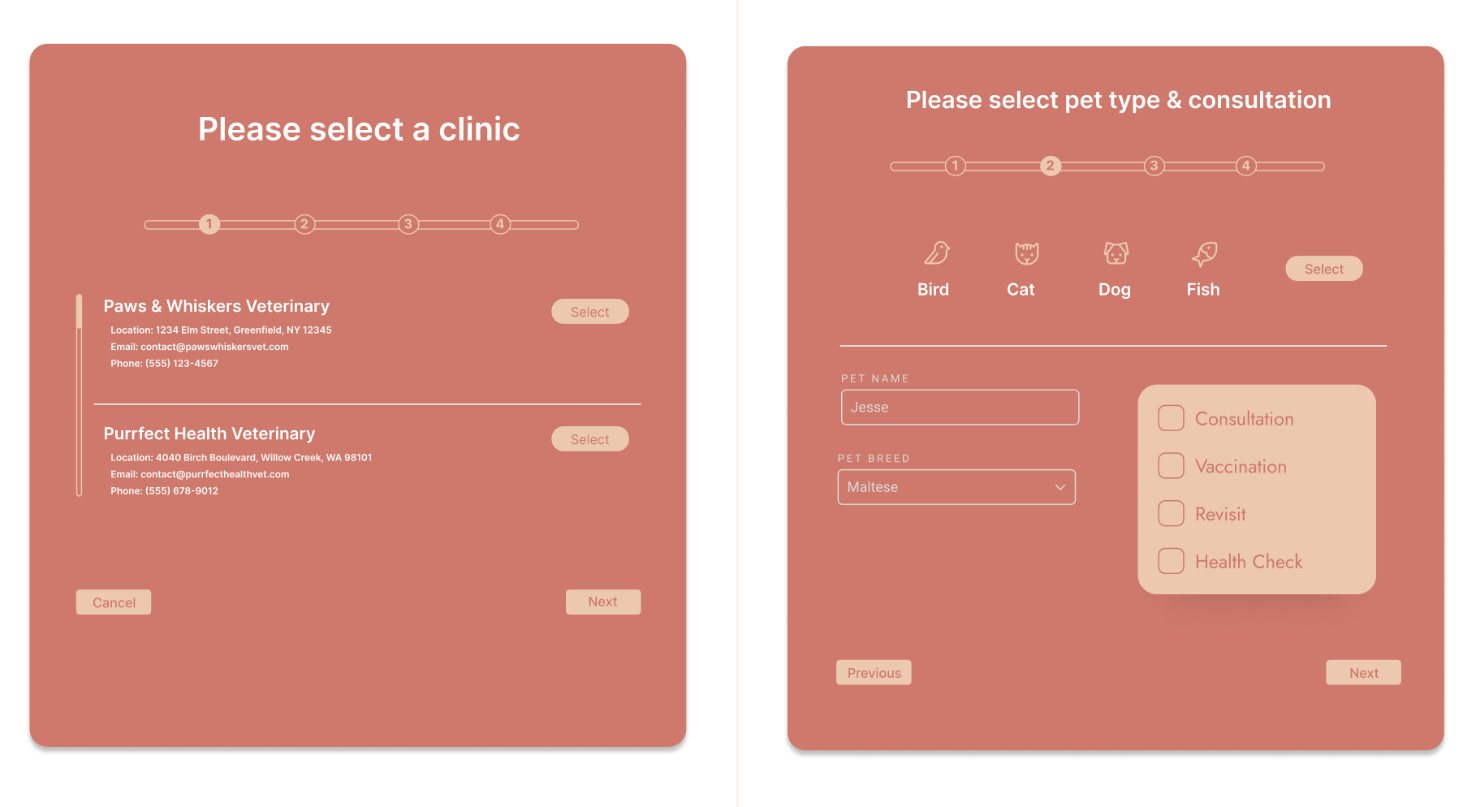




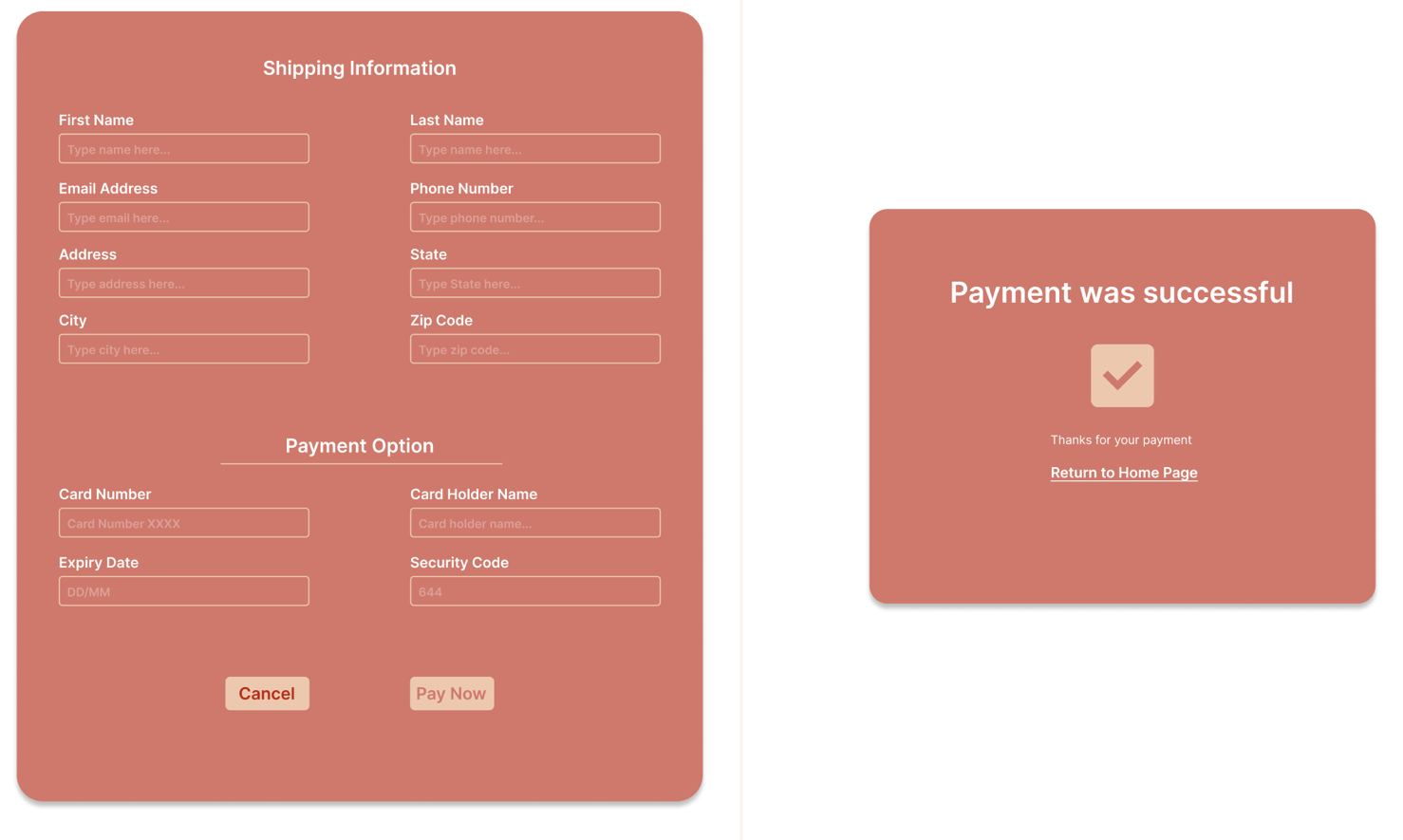
Login / Sign up modal:

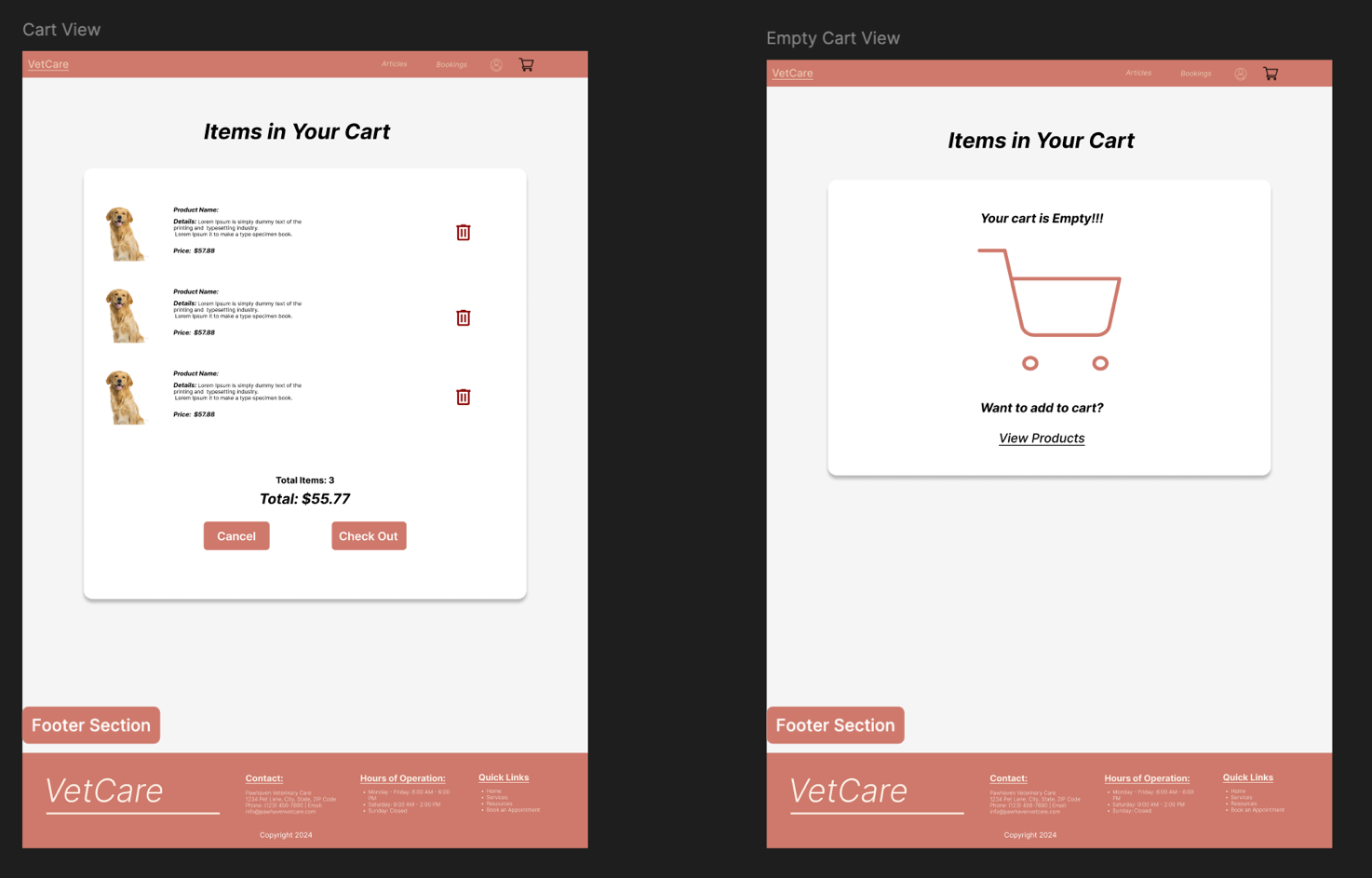


Booking modals:



Payment modal + Cart:





Appendix A: Glossary

* HTTP: Hypertext Transfer Protocol
* AWS: Amazon Web Services
* JDBC: Java Database Connectivity
* SQL: Structured Query Language
* HTML: Hypertext Markup Language
* FTP: File Transfer Protocol
* SSL: Secure Sockets Layer
* ISO: International Organization for Standardization
* IEC: International Electrotechnical Commission
* JSON: JavaScript Object Notation
* API: Application Programming Interface
* TDD: Test-driven development

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>