Software Requirements Specification

for

VetCare

Version 1.0 approved

Harmandeep Singh (s4009171), Gurnoor Kaur (s3991487), Fazila Qurban Ali (s3667195), Krishitaa Purusothaman (s3962111), Ninh Duy Huynh (s4003174), Mohamed Bilal Naeem (s3967700)

Prepared by Developers of Vetcare

25/08/2024

Revision History (No Revisions as of 25/08/2024)

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the software requirements for the VetCare Online Vet Clinic Management System. This document specifies the functional and non-functional requirements for the VetCare platform, including both the web-based and mobile applications. The SRS will serve as a basis for system development, ensuring all stakeholder needs are met effectively. The scope of this SRS covers all aspects of the VetCare system, from user interfaces to backend processes.

1.2 Document Conventions

This Software Requirements Specification (SRS) adheres to the IEEE SRS standard format to ensure clarity and uniformity throughout the document. The following conventions have been applied to maintain consistency and ease of understanding

Section headings are bold and numbered hierarchically to facilitate easy navigation and referencing for all readers. The document uses typographical conventions to enhance readability. Bold text is used for section titles, key terms, and critical information that requires attention. The distinctions help clarify the document's content, making it more accessible to its varied audience.

All communications within the system and external interactions adhere to standardized protocols. The document specifies the use of HTTP/HTTPS for web services and SSL/TLS for secure communications, ensuring data integrity and security throughout the VetCare platform.

1.3 Intended Audience and Reading Suggestions

This Software Requirements Specification (SRS) is intended for a diverse group of readers, each with specific roles and interests in the VetCare project. Developers are the primary audience, as they need detailed and clear software requirements to guide the design and development processes. The document provides them with the necessary functional and non-functional requirements to build the system accurately and efficiently.

Users, particularly veterinarians, pet owners, and documentation writers, will also benefit from this document. Users can verify that their needs and expectations are captured, while documentation writers can use the requirements to create accurate user manuals and guides. This ensures that all audiences have the necessary information to interact with the system effectively.

Project owners will find this document useful for planning and resource allocation. By understanding the scope and priorities of the software requirements, they can better manage timelines, budgets, and team assignments. This SRS helps project managers ensure that the development process aligns with project goals and stakeholder expectations.

1.4 Product Scope

The VetCare Online Vet Clinic Management System is designed to facilitate efficient management of veterinary clinics by providing functionalities such as appointment scheduling, patient records management, communication tools, and reporting. The software aims to improve clinic operations, enhance customer satisfaction, and provide a robust platform for managing pet healthcare. VetCare aligns with the corporate goals of improving veterinary service delivery through innovative technology solutions.

2. Overall Description

2.1 Product Perspective

Context and Origin:

Project Title: VetCare – Online Vet Clinic Management System
Product Description: VetCare is a cutting-edge web application designed to change how pet owners manage their pets' health. It offers solutions for scheduling appointments, accessing medical records, and consulting with veterinarians from a variety of clinics. This application itself is a new and self-contained system aiming of combining all components of vet care into a single user-friendly application for both pet owners, store owners and also medical professionals.
Type of Product: This is a new, standalone product and is not a follow-on member of any existing product family or a replacement for existing systems (as it integrates various systems together).

Integration and Interfaces:

Integration: The VetCare system will integrate with local veterinary clinics, pharmacies and product stores to provide users with real-time data on the nearby available services/products and their relevant information/ pricing details. This integration will ensure that users have access to accurate and up-to-date information about availability and accessibility to a variety of different options for pet care.
Interfaces:

- **Veterinary Clinics:** Interface to manage appointment scheduling, medical records, and prescriptions.
- **Product Providers:** Integration for displaying product information, prices, and availability.
- **User Interface (UI):** Frontend platform for users to access services, manage records, and interact with the system.

Diagram:

A simple diagram of the major components of the VetCare system might include:

- **User Interface:** Frontend platform for users
- **Appointment Scheduling System:** Component for managing bookings
- **Medical Records Database:** System for storing and retrieving pet medical histories
- **Prescription Management System:** Component for handling prescriptions and deliveries
- **Educational Resources Library:** Repository for pet care content
- **External Interfaces:** Integrations with veterinary clinics and product providers

Each component will interact with each other to provide a seamless experience for the user, ensuring that all functionalities are data flows across the system.

2.2 Product Functions

2.2.1 Major Functions Summary

The VetCare system must perform or enable users to perform the following major functions:

- **User Account Management:**
  - Sign-up, sign-in, and profile management for both pet owners and their pets.
- **Search and Exploration:**
  - Search for clinics.
  - Search for pet services (e.g., grooming, boarding).
  - Search for pet products (e.g., food, toys, medications).
- **Appointment Management:**
  - Book appointments with veterinarians at clinics.
  - Edit appointments (e.g., change consulting veterinarian, timing, service).
  - Cancel appointments.
  - Manage calendar/schedule (e.g., free up or fill-in time slots for appointments).
- **Medical Records Management:**
  - Insert, edit, delete, and view pet medical records.
  - Manage prescriptions and medication records.
- **E-commerce:**
  - Browse through pet products.
  - Filter and sort pet products.
  - Purchase pet products online.
- **Educational Content:**
  - Browse through articles and videos about pet care and wellness.

These major functions are designed to meet the needs of various user classes, including pet owners, veterinarians, front desk adminstrators at clinics and pharmacists ensuring a comprehensive and user-friendly experience.

2.3 User Classes and Characteristics

1. Pet Owners

- **Frequency of Use:** Variable; depends on the health needs of their pets.
- **Subset of Functions Used:** Booking appointments, accessing pet medical records, making payments, purchasing products, and communicating with vets, pharmacists and other clinic staff.
- **Technical Expertise:** Low to Moderate; familiarity with basic web applications and mobile apps.
- **Security/Privilege Levels:** Low to Moderate; access to their pet's records and payment information.
- **Educational Level:** Varies widely; general public with varying levels of tech savviness.
- **Experience:** Typically, little to no professional experience with medical systems.

Pertinent Characteristics:

Pet owners use the Vetcare system to tend to only their own pets. This can range from looking for pet services, purchasing pet products, getting in touch with expert veterinarians to treat their pets...etc. Thus, the system should make access to own pet medical records, prescriptions, vaccination records, appointment booking and communication tools comprehensive and approachable for them.

Importance: High. Pet owners will make up most of the stakeholders, going by the demand to supply ratio – where demand is the demand for pet care by pet owners and supply is the treatment to pets from veterinarians, pharmacies and clinic staff.

2. Veterianians

- **Frequency of Use:** High
- **Subset of Functions Used:** Viewing and updating medical records, managing patient consultations, prescribing medications, and communicating with pet owners.
- **Technical Expertise:** Moderate; familiarity with medical records systems and diagnostic tools.
- **Security/Privilege Levels:** Very High; access to all medical records and sensitive information.
- **Educational Level:** Veterinary degrees and specialized medical training.
- **Experience:** Typically, experienced professionals with a veterinary background.

Pertinent Characteristics:

Vets need quick access to medical histories before an appointment to refer to and the ability to update patient records *during* the appointments. The system should be intuitive and minimize data entry time, and provide easy access to updating medications for the pet on the vetcare system.

Importance: High. Vets are the primary source of contact for pet owners, as they have the highest expertise in treating pets out of all the stakeholders of the VetCare system. They do work that directly impacts the health of pets, hence they are one of the most important user classes.

3. Front desk administrators at Clinics

- **Frequency of Use:** High
- **Subset of Functions Used:** Appointment scheduling, patient record management, communication with pet owners, billing, and prescription management.
- **Technical Expertise:** Moderate; familiar with medical software and basic IT skills.
- **Security/Privilege Levels:** High; access to sensitive patient and financial data.
- **Educational Level:** Typically, administrative or medical office training.
- **Experience:** Varies from entry-level to experienced office staff.

Pertinent Characteristics:

Front desk administrators are responsible for the day-to-day operations of veterinary practices, including managing appointments, maintaining pet medical records, processing payments, and handling communications with pet owners. They need a user-friendly interface and efficient workflows to manage their tasks effectively.

Importance: High: The system must cater effectively to clinic staff as they are primary users and rely on the system for critical operational tasks.

4. Pharmacists

- **Frequency of Use:** Moderate
- **Subset of Functions Used:** Medication management, prescription fulfillment, inventory tracking, and interaction with vets and pet owners.
- **Technical Expertise:** Moderate; knowledge of pharmaceutical software and inventory management systems.
- **Security/Privilege Levels:** High; access to prescription data and inventory information.
- **Educational Level:** Pharmacy degrees or related qualifications.
- **Experience:** Typically experienced in pharmaceutical management.

Pertinent Characteristics:

Pharmacists require the system to handle prescription orders, track inventory levels, and ensure compliance with medical standards. They also need to communicate with vets for prescription clarifications and with pet owners for medication instructions.

Importance: Medium: Pharmacists are important for ensuring the correct medications are provided to pets, and the system should support their workflow efficiently.

User class importance: Pet Owners, Veterinarians and Front desk administrators are the **most important** user classes for the VetCare system (in that order). Whereas pharmacists are the **least important** user classes for this system.

2.4 Operating Environment

- The VetCare - Online Vet Clinic Management System is designed to operate within a modern web-based environment, ensuring compatibility and performance across various hardware and software platforms. Below is a detailed description of the operating environment:

Hardware Platform

2.4.1.1 **Server Hardware:** The application will be hosted on cloud-based servers, such as AWS, Azure, or Google Cloud Platform, providing scalability, reliability, and high availability.

2.4.1.2 **Client Hardware:** Users will access the application via personal computers, laptops, tablets, and smartphones. The system is optimized to run efficiently on devices with at least 4GB of RAM and modern multi-core processors.

Operating System

- **Client-Side:** The application is platform-independent and can be accessed via any modern web browser (e.g., Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge) on various operating systems, including Windows 10 or later, macOS Big Sur or later, iOS 14 or later, and Android 10 or later.

2.4.1.3 Web Framework: Spring Boot

2.4.1.4 Build Tool: Maven

2.4.1.5 **Containerization: Docker**

2.4.1.6 Data Storing: MySQL

2.4.1.7 **Integration:** GitHub

Other Software Components

- **Java 17 or Later:** The backend of the application is developed using Java, ensuring performance, security, and compatibility with the latest Java features.
- **JUnit5:** Unit tests are conducted using JUnit5 to maintain code quality and ensure the reliability of the application's features.
- **Integration with External Systems: T**hird party notification and payment system

2.5 Design and Implementation Constraints

Time Constraints:

- **Fixed Deadlines:** Development and deployment must adhere to specific timelines, limiting flexibility for extensive changes or more advanced features. Can also limit testing and lead to errors after deployment.

Cost Constraints:

- **Budget Limits:** The project budget restricts the scope of features and the use of advanced technologies or tools. Moreover, this constraint limits scalability and throughput of our application potentially limiting user experience.

Database Limitations:

- **Scalability:** Database must handle expected data volumes and user load without performance degradation. Difficult to construct database to handle unknown amount of data and volume before development.
- **Query Complexity:** Constraints on query complexity may affect functionality and performance.

Technological Constraints:

- **Tech Stack:** Must use specified technologies, tools, and frameworks, potentially limiting development options. Moreover some Technolgies may prevent integration with other tools, limiting overall development.

Regulatory and Security Constraints:

- **Compliance:** Must meet data protection regulations and security standards for handling sensitive information.

Design Standards:

- **Programming Conventions:** Follow organizational coding practices and documentation standards to ensure maintainability. Developing to ensure future updates are easy to integrate without changing the architecture significantly.

2.6 User Documentation

- **Online Help:** Embedded help resources that provide users with contextual assistance based on the section of the application they are using. This includes tooltips, FAQs, and step-by-step guidance or a quick user guide
- **Format:** Directly accessible within the application through a help icon or menu.
- **Video Tutorials:** Visual presentation that demonstrates the use of the VetCare application, covering topics like account setup, appointment booking, and accessing pet records and other details would be provided by the all-team members
- **Format:** Video (MP4/Online) Hosted on the VetCare website and possibly on platforms like YouTube, with links provided within the application and documentation.

2.7 Assumptions and Dependencies

- *Notification system, online payment system, MySQL for database, JetBrains (operating environments), Web framework (springboot)...etc (more technology and frameworks).*
  The development and operation of the VetCare – Online Vet Clinic Management System is based on several assumptions and dependencies. These factors, while anticipated, could impact the project if they are incorrect or change during the project lifecycle
  Assumptions.
- **Cross-Platform Compatibility:** It is assumed that the web application will perform consistently across various devices (desktops, tablets, and smartphones) and operating systems (Windows, macOS, iOS, Android) as long as they use modern web browsers.
- **Third-Party Service Availability:**
- **Notification System:** It is assumed that the third-party notification system integrated into the application will remain operational, reliable, and continue to support the necessary APIs for sending reminders and alerts.
- **Payment Gateway:** The online payment gateway services are assumed to be available, secure, and capable of handling transactions without significant downtime or security breaches.
- **Dependencies**

3. **MySQL Database: The** project relies on the MySQL database for data storage and retrieval.

4. **Spring Boot Framework: The** application's development is dependent on the Spring Boot framework.

- **Docker for Containerization: The** application depends on Docker for consistent deployment across different environments.
- **Third-Party APIs:**

- **Notification and Payment Systems:** The project depends on third-party APIs for notifications and payments. Any changes in API endpoints, rate limits, pricing, or availability could significantly impact the project's functionality and user experience.
- **Continuous Integration on GitHub**

5. External Interface Requirements

5.1 User Interfaces

The VetCare system will provide a user-friendly, intuitive interface for pet owners, veterinarians, and admins. The user interface (UI) will follow modern GUI standards to ensure consistency and ease of use across all user roles.

Every screen includes a navigation bar at the top that features links to essential sections such as Home, Pet Records, Appointments, Resources, Notifications, Reviews, Contact, and My Profile. This bar ensures users can easily navigate the platform. A footer with essential links and social media icons appear on every page, providing a familiar structure.

Users can use keyboard shortcuts such as "Alt + H" for Home, "Alt + P" for Pet Records, and "Alt + A" for Appointments, enabling power users to navigate the platform more efficiently.

Error messages are displayed near the relevant fields or sections, in a clear and concise manner, ensuring that users understand the issue and how to correct it. Error messages use consistent language and design across the platform, with specific instructions to guide users back to the correct workflow.

5.2 Hardware Interfaces

The VetCare system will interact with various hardware components to ensure seamless operation across different devices. This section describes the logical and physical characteristics of each interface between the software product and the hardware components.

Supported Device Types
The VetCare system is designed to be compatible with a wide range of devices, including but not limited to:

- **Desktop Computers and Laptops:** Running operating systems such as Windows, macOS, and Linux.
- **Tablets:** Including devices running iOS (e.g., iPads) and Android.
- **Smartphones:** Supporting both iOS (e.g., iPhones) and Android devices.

Display and Input Devices

- **Display:** The VetCare UI is optimized for different screen sizes, ranging from desktop monitors to small smartphone screens. The system uses responsive design techniques to ensure that the interface adapts to various screen resolutions and orientations (portrait and landscape).
- **Input Devices:**
    - **Mouse and Keyboard:** Standard input for desktops and laptops. The system will support keyboard shortcuts for common actions (e.g., navigating between tabs, submitting forms).
    - **Touch Input:** Tablets and smartphones will rely on touch input, with the UI supporting gestures such as swipe, pinch, and tap for navigation and interaction.

Data and Control Interactions

- **Data Input:** Users will input data into the VetCare system through various forms and controls within the UI. This data is transmitted to the system for processing and storage.
- **Data Output:** The system will display data to users via the UI, ensuring that information such as pet records, appointments, and notifications are accessible and easy to read across all devices.

Communication Protocols

The VetCare system will communicate with hardware components using standard communication protocols:

- **USB Interface:** For connecting external devices like printers to print reports or medical records directly from the system.
- **Bluetooth:** To allow wireless connectivity with devices such as mobile printers, barcode scanners, or other medical devices that may interact with the system.
- **Wi-Fi:** Ensuring the system can connect to the internet for online features, including cloud storage, remote database access, and real-time notifications.
- **Peripheral Devices**
- **Printers:** The system will support printing capabilities, allowing users to print pet medical records, appointment schedules, and other documents. Both wired (USB) and wireless (Wi-Fi, Bluetooth) printers will be supported.
- **Scanners:** Barcode scanners or QR code readers may be used to quickly input pet identification details or access records. The system will support these devices via USB or Bluetooth connections.
- **Performance Considerations**
- **Latency:** The VetCare system will be optimized to minimize latency in interactions between the software and hardware components, ensuring a smooth user experience, particularly when accessing data or printing documents.
- **Power Consumption:** For mobile devices (smartphones and tablets), the system will be designed to minimize power consumption to extend battery life during use.

This hardware interface section ensures that the VetCare system is compatible with various devices and peripherals, providing a seamless and efficient user experience across all platforms.

5.3 Software Interfaces

The VetCare system interacts with various software components and external services. Below is an overview of the key connections:

- **Operating System:**
  - The application is designed to run on a variety of operating systems, primarily Linux-based environments (e.g., Ubuntu) in production, with support for Windows and macOS in development.
- **Database (MySQL):**
  - **Data Flow:** The Persistence Layer interacts with MySQL to store and retrieve data related to users, pets, appointments, medical records, and transactions.
  - **Integration:** Spring Data JPA (Java Persistence API) is used to abstract database operations and manage entities within MySQL. SQL scripts or migration tools like Flyway may be used to manage schema changes.
- **Web Services (Spring Boot):**
  - **Data Flow:** Spring Boot handles HTTP requests and responses, processing data through the Business Layer and interacting with the Persistence Layer as needed.
  - **APIs:** RESTful APIs are exposed for interaction with client applications, including mobile apps and third-party services.

- **User Interface (Thymeleaf):**
  - **Data Flow:** The Presentation Layer uses Thymeleaf to render dynamic web pages based on data processed by the Business Layer and fetched from the Persistence Layer.
  - **Templates:** Thymeleaf templates are used to create reusable components, ensuring consistency across the user interface.
- **Containerization (Docker):**
  - **Data Flow:** Docker containers package the application along with its dependencies, ensuring consistency across different environments.
  - **Integration:** Docker Compose may be used to manage multi-container setups, including the application, database, and other services.
- **Continuous Integration/Continuous Deployment (GitHub Actions):**
  - **Data Flow:** Code changes trigger automated workflows in GitHub Actions, which run tests, build the application, and deploy it to the desired environment.
  - **Integration:** GitHub Actions integrates with Docker for container management and deployment, as well as with MySQL for database migrations during deployment.
- **Testing Framework (JUnit5):**
  - **Data Flow:** JUnit5 is used to execute unit tests, ensuring that individual components of the system function as expected.
  - **Integration:** Test reports and code coverage are integrated with the CI/CD pipeline, ensuring quality control throughout development.

Data Sharing Across Components

- **User and Pet Data:** Shared between the Presentation Layer (for displaying information), the Business Layer (for processing logic), and the Persistence Layer (for storage in MySQL).
- **Appointment and Medical Records:** Managed primarily by the Business Layer, with interactions between the UI (via Thymeleaf) and storage/retrieval operations in the database.
- **E-commerce Transactions:** Payment details and transaction records are handled by both the Business Layer and Persistence Layer, ensuring secure processing and storage.

Implementation Constraints

- **Database Access:** All data-related operations must be handled through the Persistence Layer using Spring Data JPA, ensuring that direct database access is controlled and secured.
- **Containerization:** The application must be deployable in Docker containers, requiring all components to be compatible with the Docker environment.
- **Security:** Data must be encrypted at rest and in transit, particularly sensitive information such as medical records and payment details.

5.4 Communications Interfaces

The platform sends automated email notifications and alerts to users regarding appointment reminders, medication schedules, and system updates. These emails are formatted in HTML to ensure a consistent and visually appealing presentation.

The platform utilizes RESTful APIs for communication between the front-end interface and the back-end server. These APIs are structured to handle standard HTTP methods (GET, POST, PUT, DELETE) for CRUD operations.

Sensitive data stored on the server (such as user credentials and health records) is encrypted at rest to protect against unauthorized access.

6. Nonfunctional Requirements

6.1 Performance Requirements

1. Response Time

Definition: Response time refers to the duration the system takes to react and respond to a user's action/request.

Features/Requirements:

Feature	Requirement	Rationale
Booking with Specific Veterinarian	Process booking in 2 seconds of request submission	Ensures a smooth user experience during appointment scheduling.
Communication with Other Clinics/Stores	Complete communication within 2 seconds	Key to coordinated care and timely service.
Email Confirmation	Sent within 1 minute	Builds user trust and operational efficiency.
Receive Confirmation Email for Payment/Receipt	Complete within 1 minute	Provides transaction assurance and builds trust.
Report Generation (Veterinarian)	Generate and print summary reports within 15 seconds	Ensures timely access to necessary information for decision-making.
Access Educational Resources	Complete within 2 seconds	Enhances user experience and encourages platform use.
Access Pet's Medical Records	Complete within 3 seconds	Critical for immediate information access, especially in emergencies.
Upload Educational Resources	Complete within 5 seconds for 100MB	Reduces downtime and allows focus on other tasks.

Rationale: Fast response times for various user interactions, such as loading pages, submitting forms, or updating records are essential for maintaining a smooth and efficient user experience, reducing frustration, and removing unnecessary delays which can also reduce system usability/reliability.

2. System Update

Definition: System update measures the time taken for changes to be reflected across the system for all users after an update, modification or feature. This metric tracks how quickly the system integrates and displays updates including schedule changes or record modifications.

Features/Requirements:

Feature	Requirement	Rationale

Set and Manage Clinic Schedule	Reflect changes to user-visible data (e.g., schedules, appointment details) within 1 second	Prevents scheduling conflicts and ensures accurate availability.
Update Appointment Schedule	Reflect within 1 second	Ensures timely updates and prevents scheduling conflicts.
Manage Appointment Schedule and Customer Details	Reflect within 1 second	Prevents errors and ensures smooth operations.
Examine and Modify Payment Information	Reflect immediately within 1 second	Maintains accurate financial records and prevents payment errors.
System Restart	Recover within 5 minutes post-shutdown	Minimizes downtime and disruption, allowing users to resume normal operations quickly.
Data Recovery	Restore data within 1 hour after a database crash	Ensures minimal data loss and quick recovery for continuity of operations.

Rationale: Rapid system updates are crucial for maintaining accurate and up-to-date information, which helps avoid conflicts and errors while maintaining system integrity.

3. Throughput

Definition: The number of transactions/operations the system can handle within a specified time frame.

Requirement:

Appointment Bookings: Process up to 100 bookings per minute.

Data Upload/Download: Handle up to 50MB per second.

Real-time Notification Delivery: Deliver up to 100 real-time notifications (e.g. appointment reminders, medication updates, emergency alerts etc) per second to users.

Prescription Management: Process up to 150 prescription orders per minute.

Inventory Management Updates: Process up to 100 inventory updates per minute for all clinics to add new stock, update quantities/prices or description.

Data Synchronization Across Devices: Synchronize data changes (e.g., appointment updates, medical records) across all devices within 2 seconds, supporting up to 100 concurrent changes

E-commerce Transactions: Handle up to 100 product purchases per minute through the VetCare system.

Rationale: High throughput is essential for VetCare to ensure smooth operation during peak usage periods, preventing performance bottlenecks and better user experience. By processing a high number of appointment bookings, prescription orders, and inventory updates per minute, it ensures the system not only maintains smooth operations but also meets user expectations. Moreover, fast data upload/download speeds and real-time notifications ensure consistent and timely communication for users, allowing for

efficient information management. These requirements collectively enhance user experience and increase operational efficiency, which is essential for a management system.

4. Scalability

Definition: The system's ability to grow and handle increased load (of users/data demand) without significant performance degradation.

Requirement:

Concurrent Users: Support up to 2000 concurrent users without noticeable slowdown or delay during peak usage period (e.g. weekend rushes).

Data Volume: Maintain process efficiency while handling a database of up to 100,000 records.

Geographical Scalability: Support operations for up to 100 clinics across various cities, ensuring each clinic accesses shared data with a maximum latency of 200 milliseconds.

Rationale: Scalability ensures the system can grow with user demand without significant reengineering. A scalable system can adapt to both predictable and unpredictable increases in load, ensuring consistent user experience and operational efficiency int he long term. Moreover, for a system that manages various clinics and stores, geographical and data vlume scalability is essential for system reliability and overall user experience.

5. Reliability and Uptime

Definition: The system's ability to operate without failure over a specified period.

Requirement:

Uptime: Achieve 99.9% uptime.

Backup and Recovery: Implement daily backups of critical data (e.g. pet medical records) and ensure ability to restore these backups within 30 minutes in event of data loss

System Crashes: Minimize frequency of crashes or errors affecting users.

Error Handling: Implement robust error handling and logging mechanisms to quickly identify any issues needed to be resolved.

Rationale: High reliability and uptime are crucial for maintaining user trust and ensuring system availability. Moreover, error handling and backups ensure system integrity.

6. Data Integrity

Definition: The accuracy and consistency of data throughout its lifecycle.

Requirement:

Data Accuracy: Ensure data such as medical records and appointment schedules remain correct and uncorrupted during operations.

Audit Trails: Maintain delated audit logs for all data modifications, including changes made, individuals involved, dates to track and ensure data accuracy over time.

Validation: Implementing input validation to prevent incorrect, malicious or malformed data from entering the system.

Automated Data Integrity Alerts: Notify administrators within 1 minute if any discrepancies, data corruption or unauthorized medications are detected in the system (e.g. for medical records or appointment schedules).

Rationale: By maintaining real-time data consistency, the system ensures that all users, including veterinarians and clinic staff, have access to the most up-to-date information. This prevents discrepancies and reduces the risk of errors in patient care. Moreover, the proactive approach of prompt alerts for administrators helps maintain user trust and prevents potential errors or defects ensuring compliance with data management standards.

7. Recovery Time

Definition: The time required for the system to recover from a failure.

Requirement:

System Restart: Ensure system can perform a complete shutdown and become operational within 5 minutes post-shutdown.

Error Resolution: Implement automated error detection and reporting mechanisms that allow administrators to identify system failures/inconsistencies within 30 minutes of occurrence.

Data Recovery: Restore all critical data including user/pet records and transaction history within 1 hour after a database crash.

Rationale: Fast recovery times minimize downtime and disruption, ensuring users can resume normal operations quickly. Furthermore, an efficient system restart and data recovery ensure high levels of service and data integrity which swift error resolution helps address issues before they escalate.

8. Security and Access Control

Definition: Protection from unauthorized access and efficient management of access permissions.

Requirement:

Authentication: Complete secure authentication within 2 seconds for user login and access verification.

Access Permission Updates: Any changes to user access permission should be applied across the system within 5 seconds.

Security Breach Detection: Implement real-time monitoring for unauthorized access or malicious activities by using alerts within 1 second

Data Transmission: Support encrypted data transmission using industry-standard protocols with no more than 10% performance overhead from encryption/decryption operations reducing performance impact.

Rationale: Balancing security with performance and efficient authentication is essential for protecting user data while maintaining a seamless experience. Ensuring timely encryption, quick application of permission changes, and prompt detection of potential threats helps protect sensitive data while supporting a smooth and responsive user experience.

6.2 Safety Requirements

Description: This section specifies the safety considerations that must be integrated into the VetCare's product's design and usage. It includes safeguards to mitigate risks, actions that must be avoided, compliance with relevant regulations, and any necessary safety certifications.

1. Data Protection and Privacy

Requirement: Ensure compliance with data protection regulations such as *Privacy Act 1988(Cth), Spam Act 2003 (Cth), My Health Records Act 2012 (Cth),* and other relevant privacy laws.

Safeguards/Actions:

- Provide users with clear and accessible privacy policies and consent mechanisms.
- Implement encryption for data at rest and in transit.
- Regularly audit and update security measures to address new threats.

Actions to Prevent:

- Unauthorized access to personal data.
- Data corruption, unauthorized access or malicious activity
- Data breaches or leaks.

Rationale: Protecting user data is critical to prevent identity theft, and breaches of privacy (e.g. for personal user data or pet medical information). Moreover, Compliance with these data protection regulations not only maintains user trust but also provides legal adherence.

2. User Authentication and Authorization

Requirement: Implement authentication/authorization mechanisms to ensure that only authorized users have access to the system, with permissions based on their defined roles and responsibilities.

Safeguards/Actions:

- Regularly review and update access controls based on user roles and responsibilities.
- Use multi-factor authentication (MFA) for user accounts.
- Role-based access control (RBAC) to enforce varying levels of access and permissions depending on user roles.
- Implement session timeouts and automatic logouts after periods of inactivity using tokens.

Actions to Prevent:

- Unauthorized access to user accounts and sensitive information.
- User impersonation and privilege escalation.

Rationale: Effective authentication and authorization prevents unauthorized access and protect sensitive data from being exposed/manipulated by unauthorized users.

3. System Security

Requirement: Ensure that the system is protected against various security threats and vulnerabilities.

Safeguards/Actions:

- Regularly update and patch software to fix known vulnerabilities or potential future vulnerabilities.

- Use firewalls, intrusion detection systems (IDS), SSL/TLS encryption, secure content management systems, and strict user permission controls to protect against external and internal threats.
- Conduct regular security assessments and penetration testing.

Actions to Prevent:

- Cyberattacks, including malware, data breaches, Phishing schemes, ransomware, and denial-of-service (DoS) attacks or DDoS attacks.
- System vulnerabilities that could be exploited by attackers.

Rationale: Maintaining system security is essential to protect the integrity of the system, user data, and to not only prevent disruptions/threats/attacks but also reduce the damage caused by cyber threats or system vulnerabilities.

4. Safety Certifications

Requirement: Obtain necessary safety certifications and comply with industry standards.

Certifications:

- **ISO/IEC 27001:** Information security management.
- **ISO/IEC 27018:** Protection of personal data in the cloud.
- **Australian Signals Directorate (ASD) Essential Eight:** Strategies to mitigate cybersecurity incidents.

Safeguards/Actions:

- Regularly review and maintain certification compliance.
- Ensure all components/processes are certified according to relevant standards.

Actions to Prevent:

- Non-compliance with industry standards and regulations.
- Potential legal adherence and financial repercussions due to lack of certification.

Rationale: Compliance with safety certifications ensures the system meets recognized standards for security and quality. This protects users and data, enhances credibility, and prevents legal and financial repercussions.

5. Emergency Response Recovery

Requirement: Establish procedures for emergency response and recovery to handle unexpected incidents effectively to reduce/prevent further damage.

Safeguards/Actions:

- Develop and document an emergency response plan for admins/system, including steps for system recovery and data restoration.
- Conduct regular drills and training for staff to handle emergencies.
- Implement backup systems and redundant infrastructure to minimize potential downtime.

Actions to Prevent:

- Extended system outages or data loss due to emergencies.
- Downtime or critical damage to system

Rationale: Having a robust emergency response and recovery plan ensures that the system can quickly recover from failures or incidents and that staff are knowledge about procedure during emergencies, which increase security and minimizes the impact on users and operation.

6.3 Security Requirements

Product Protection:

Ensuring the privacy and security of personal and medical data on the VetCare website is of outmost importance, as it must comply with relevant data protection regulations such as GDPR and HIPAA. It is essential to ensure that all user data, such as pet information, prescription details, and payment information, is encrypted using industry-standard encryption protocols (e.g., TLS 1.2 or higher) both at rest and in transit.

User Authentications:

- Password policies: The system should incorporate robust authentication measures, such as strong password requirements,
- Multi-factor authentication (MFA): Implementing MFA involves utilizing multiple verification methods, such as combining a password with a one-time code sent via SMS or using biometric verification, which is an effective session management, to ensure that user accounts remain secure and protected from unauthorized access. It is important to ensure that all individuals, whether they are pet owners or veterinary staff, are properly authenticated before being granted access to sensitive information.
- Access control: Implementing role-based access controls (RBAC) is crucial for ensuring that users are granted access only to the specific data and functions required for their roles. As an illustration, only individuals in the veterinary field should have the ability to make changes to prescription details, whereas pet owners have the option to view and ask for refills.
- Single Sign-On (SSO): Enable SSO to streamline user authentication across multiple systems, eliminating the need for repeated authentication.
- Biometric authentication: For enhanced security, it is worth considering the implementation of biometric authentication methods such as fingerprint or facial recognition.

External Policies or regulations:

- General Data Protection Regulation (GDPR): Ensure compliance with GDPR if your product operates in or handles data of EU citizens. Pay special attention to user consent, data protection, and rights to data access/erasure.
- Health Insurance Portability and Accountability Act (HIPAA): When it comes to products involving healthcare data, strict security measures are mandated by HIPAA. These measures include encryption and user access controls.
- Payment Card Industry Data Security Standard (PCI DSS): Ensuring the security of payment data is crucial. Compliance with PCI DSS requirements is necessary to safeguard the storage, transmission, and processing of cardholder information.

Security or privacy certifications:

- ISO/IEC 27001: Make sure the product meets the requirements of this certification, which establishes a structure for implementing information security management systems (ISMS).

- SOC 2: Ensure that your product meets the necessary security, availability, processing integrity, confidentiality, and privacy standards by obtaining SOC 2 certification.
- Cybersecurity Maturity Model Certification (CMMC): Ensuring compliance with cybersecurity practices is crucial for products working with U.S. Department of Defence data, as CMMC certification demonstrates.
- FIPS 140-2: Ensuring compliance with the Federal Information Processing Standard (FIPS) 140-2 is crucial for government-related projects involving cryptographic modules.

6.4 Software Quality Attributes

1. **Adaptability:**

   - **Requirement:** The system should be able to easily accommodate future changes in payment methods, prescription formats, or user interfaces without the need for significant architectural modifications.
   - **Performance Measure:** Ensuring that a significant portion, around 60%, of code modules can be reused without any modifications in future expansions or related projects is crucial.
   - **Verification:** Code should be organized into modules and well-documented, ensuring that different aspects are clearly separated. Conduct thorough code reviews with a focus on identifying opportunities for future reuse.

2. **Correctness:**

   - **Requirement:** Ensuring the precise management of prescriptions and consistently presenting accurate information to users is of utmost importance.
   - **Performance Measure:** Prescription records should ideally maintain an error rate of less than 0.1% across a total of 10,000 transactions.
   - **Verification:** Ensure that automated tests are implemented, encompassing unit and integration tests, with a minimum code coverage of 95%. Conduct routine audits to ensure the accuracy and reliability of data.

3. **Reusability:**

   - **Requirement:** It is important to design code modules that can be reused across different components of the system. This will ensure efficient payment processing, user authentication, and prescription management.
   - **Performance Measure:** Ensuring that a significant portion, at least 60%, of code modules can be reused without any modifications in future expansions or related projects is crucial.
   - **Verification:** Code should be organized into modules and well-documented, ensuring that different aspects are clearly separated. Conduct thorough code reviews with a focus on identifying opportunities for future reuse.

4. **Robustness:**

   - **Requirement:** The system should be able to handle any incorrect inputs, unexpected user actions, and hardware or network issues without crashing.
   - **Performance Measure:** The system must maintain a minimum uptime of 99.9% and ensure that no individual failure exceeds a duration of 2 minutes.
   - **Verification:** It is important to conduct stress tests, fault injection, and automated error recovery tests. Monitor system performance in real-time and establish alerts for critical failures.

5. **Availability:**

- **Requirement:** The system needs to be up and running smoothly, ensuring that customers can access it without any significant interruptions.
- **Performance Measure:** Consistently maintaining a 99.9% uptime during peak hours. Ensuring a maximum downtime of 10 minutes per month.
- **Verification:** Consistent uptime reports and diligent monitoring logs. Alerts will be generated automatically for any instances of downtime that exceed 1 minute.

6. Usability:

- **Requirement:** The system should be user-friendly and straightforward, allowing customers and staff to navigate and complete tasks like scheduling appointments, checking prescriptions, or making payments with ease and efficiency.
- **Performance Measure:** It is important to ensure that the average task completion time for common tasks, such as booking appointments, does not exceed 2 minutes. It is important to ensure that the user error rate remains below 2% when carrying out key actions.
- **Verification:** Conducting usability testing with a minimum of 20 participants, achieving success rates exceeding 90%. Striving for a user satisfaction rating of at least 4 out of 5 in ease of use through post-release surveys.

7. **Reliability**:

- **Requirement:** The system needs to consistently function without any failures that hinder users from successfully completing important tasks such as booking or making payments.
- **Performance measure:** With an Mean Time Between Failures (MTBF) of 1000 hours and an Mean Time to Repair (MTTR) of 1 hour, the system's reliability and efficiency are well-balanced.
- **Verification:** Providing logs and incident reports that demonstrate meeting the targets for Mean Time Between Failures (MTBF) and Mean Time to Repair (MTTR).
  Ensuring regular automated testing for critical functions.

8. Performance and Scalability:

- **Requirement:** The system needs to ensure fast response times and minimal delays, even during high-demand situations such as Black Friday sales or health emergencies.
- **Performance measure:** The system needs to be capable of handling a large number of users simultaneously, ensuring a quick response time of under 2 seconds.
- **Verification:** Perform load testing in simulated high-traffic scenarios, closely monitor system performance, and maintain consistent results.

9. Interoperability:

- **Requirement:** The system needs to have the capability to seamlessly exchange and utilise information with various external systems such as third-party payment gateways, pharmacies, or vet databases.
- **Performance measure:** Ensuring seamless integration with industry-standard APIs (such as REST and JSON) and lightning-fast response times of under 2 seconds for data exchanges. 100% data transfer rate achieved during integration testing.
- **Verification:** The integration test results indicate that there were no critical errors detected during the exchanges.
  Ensuring adherence to the necessary API documentation and industry standards.

10. Maintainability:

- **Requirement:** The system should be designed for easy maintenance, enabling swift updates and bug fixes with minimal downtime.
- **Performance measure:** The system should be designed to accommodate updates or fixes with minimal disruption. For example, updates should be completed within one hour and downtime should be limited to no more than five minutes.
- **Verification:** Perform routine maintenance simulations, monitor update durations, and monitor system logs to ensure that downtime remains within acceptable thresholds.

11. Security:

- **Requirement:** The system should ensure the security of sensitive information, such as prescription data and payment details, by implementing industry-standard encryption and authentication mechanisms.
- **Performance measure:** Ensuring a 12-month period free from any security breaches or data leaks is of utmost importance.
- **Verification:** Perform routine penetration testing, security code reviews, and audits in accordance with Open Web Application Security Project (OWASP) guidelines. Monitor the system's adherence to data protection regulations such as General Data Protection Regulation (GDPR).

12. Portability:

- **Requirement:** The system should be able to run on different operating systems such as Windows, macOS, and Linux, and should work well with popular web browsers like Chrome, Safari, and Firefox.
- **Performance measure:** The system needs to work properly on at least 90% of the commonly used platforms and browsers.
- **Verification:** Ensure that core functionalities are thoroughly tested across multiple platforms to verify seamless performance.

13. Flexibility:

- **requirement:** The system should be flexible enough to accommodate new features, like the inclusion of more payment methods or the integration with third-party services, such as prescription management.
- **performance measure:** When adding a new feature or service, it should be manageable to integrate without extensive code refactoring or a lengthy development period.
- **verification:** Ensure accurate tracking of the time and effort needed for implementing new features or making changes, while also conducting thorough regression tests to maintain the stability of existing functionalities.

14. Testability:

- **Requirement:** The system should be designed with testing in mind, allowing developers to easily identify and address bugs, ensuring that new code does not introduce any issues.
- **Performance measure:** It is important to ensure that unit tests adequately cover a significant portion of the system's codebase, ideally reaching at least 80%. Additionally, it is crucial that automated tests can be executed efficiently within a maximum timeframe of 10 minutes.
- **Verification:** Keep an eye on code coverage reports and keep track of the execution time for the test suite after every release.

6.5 Business Rules

1. **Veterinarians:**

Role: Medical professional

Functions:

- Efficiently handle pet medical records with ease.
- Perform tasks related to prescription management.
- Reviewing and approving appointment requests from pet owners.
- Offer virtual consultations and deliver expert medical guidance.
- Evaluate and make decisions on prescription renewals requested by pet owners.

2. **Pet Owners:**
   Role: user

   Functions:
   - Sign up and oversee personal and pet profiles.
   - Access pet medical history, which includes vaccination records and past prescriptions.
   - Schedule appointments with your preferred veterinarians.
   - Gain access to and download approved prescriptions.
   - Ensure timely prescription renewals by requesting them before they expire.
   - Choose from a variety of payment options, including credit card and PayPal, to conveniently make payments for services.
   - Access information about both upcoming and past appointments.
   - Get notified about your appointments, prescription renewals, and medical reminders.

3. **Receptionist:**
   Role: administrative support

   Functions:
   - Oversee user accounts for pet owners and veterinarians.
   - Manage appointment schedules and coordinate bookings.
   - Evaluate appointment requests and make decisions based on availability.
   - update the service offerings, pricing, and business hours on the system.
   - Assist customers with enquiries regarding account management, appointments, and payments.
   - handle refunds and accommodate special requests for cancellations or rescheduling.
   - Ensure accurate record-keeping for compliance and reporting purposes.

4. **System administrator:**
   Role: Technical and System Management

   Functions:
   - Ensure optimal system performance, security, and updates are managed and maintained.
   - Make sure to back up data, comply with data protection regulations, and store it securely.
   - Set up role-based access control and permissions.
   - Monitor system usage and generate reports to provide valuable business insights.
   - Address technical issues and bugs that impact the system's functionality.

7. Other Requirements

1. Ethics

Requirement:

- **User Consent and Privacy:** Obtain explicit consent for data collection, usage, and sharing. Ensure transparency regarding the utilization of user data and enable users to manage their data preferences.
- **Data Anonymization:** Where possible, anonymize user data to safeguard privacy. Ensure that personally identifiable information (PII) is not exposed or misused and is securely stored.
- **Ethical Use of AI:** If the system uses AI or machine learning, ensure that algorithms are structured to avoid biases and discriminatory practices. This can be achieved by regularly auditing AI models for fairness and accuracy.

Rationale: Maintaining ethical standards fosters user trust, allows compliance with regulations, and ensures fair treatment of all users. Ethical practices in AI and data handling contribute to responsible technology development and use.

2. Legal Requirements

Requirement:

- **Compliance with Data Protection Laws:** Ensure compliance with relevant data protection regulations such as GDPR (General Data Protection Regulation), Australian Privacy Act 1988 and other applicable local laws. This includes data collection, processing, storage, and user rights.
- **Legal Record Keeping:** Maintain records of user consent, data processing activities, and compliance audits to demonstrate adherence to legal requirements.
- **Contractual Obligations:** Adhering to any contractual obligations or (SLAs) service level agreements that impact system functionality or data handling

Rationale: Following legal requirements prevents legal disputes and ensures that the system operates within the bounds of applicable laws, protecting the organization from repercussions and its users with ethical data handling.

3. Internationalization

Requirement:

- **Multi-Language Support:** Provide support for multiple languages to accommodate users from different regions ensuring that the VetCare's interface, notifications, and documentation can be translated into the supported languages to provide further support.
- **Date and Time Formats:** Support various date and time formats based on user location or preferences. Ensure correct display during scheduling or medical record updates.
- **Currency and Measurement Units:** Adapt currency and measurement units based on user location to ensure relevance and accuracy in transactions.

Rationale: Internationalization allows the system to be accessible and usable by a global audience, enhancing user experience and broadening market reach to people of different backgrounds or regions.

4. Reuse Objectives

Requirement:

- **Code Reusability:** Design the system with modularity in mind, enabling reuse of components/services across different parts of the system or in future projects.
- **Third-Party Integrations:** Utilize reusable third-party libraries and services which can help avoid redundant efforts and allow system to leverage existing solutions.
- **Documentation:** Provide comprehensive documentation for reusable components/services to facilitate their integration and use in other projects or third-party tools (e.g. other similar systems which may need to integrate with VetCare).

Rationale: Reusability reduces development time, improves maintainability, and encourages consistent and more streamlined development across projects and other tools. Documentation also supports the effective reuse and integration of components/services which allow for a cost-effect development process.

8. System Architecture

8.1 Architecture Overview

This section provides a top-level decomposition of the system, showing how various components interact within the architecture. The architecture is divided into three main layers:

Presentation Layer: This layer is responsible for the user interface and handles all interactions with the end users. It consists of:

- **User Management UI**: Interfaces for sign-up, sign-in and managing user accounts and profiles.
- **Appointment & Schedule Management UI**: Interfaces for booking and managing appointments.
- **E-commerce Page UI**: Interfaces for handling e-commerce-related activities, such as browsing and purchasing pet products.
- **Pet Records Management UI**: Interfaces for viewing and managing pet medical records.
- **Web Interface**: The primary interface that connects users to the backend services.

Business Layer: This layer contains the core business logic of the application, including:

- **User Account Management**: Handles account creation, user authentication, authorization, and profile management.
- **Search Functionality**: Allows users to search for clinics, products in the e-commerce page, or medical records on the front desk administrator's page of the VetCare system.
- **Appointment & Schedule Management**: Manages scheduling, booking, and calendar functionality (where veterinarians and clinic staff can fill or free-up time slots for appointments).
- **Order Management**: Manages the lifecycle of product orders on the e-commerce page of pet products, creation to completion.
- **Pet Medical Record Management**: Handles the insertion, deletion and in some instances, editing of pet medical history, vaccination records and prescriptions - mostly, carried out by the clinic staff/ front desk administrator's account. Whereas the viewing of pet medical records can be done by pet owners.

Integration Layer (sublayer of Business Layer)

- **Notification System**: Sends reminders, alerts, and updates to users about upcoming appointments, administration of regular medication to pets, clinical surgery statuses of their pets, order placements, order arrivals, payment confirmations…etc

- **Payment System**: Manages transactions and payment processing for purchasing of products on the e-commerce page, and in some instances of booking appointments at clinics.
- **API**: Facilitates communication between different services and components.

Persistence Layer: This layer is responsible for data storage and retrieval, consisting of:

- **Pet Repository**: Stores data related to pets, including medical records, prescriptions and profiles.
- **Clinics**: Stores data related to veterinary clinics, services provided and veterinarians.
- **Appointments**: Stores information on scheduled appointments (such as different appointments scheduled for individual pet owners and appointments from clients booked at clinics).
- **Products**: Manages product inventory and details for the e-commerce platform.
- **Customer Accounts**: Stores user profiles, account information and pet profiles of the pet owner.
- **Security**: Handles data encryption and user authentication/authorization.
- **Educational Resources**: Stores articles, videos, and other educational content about pet care and wellness.

There are also domains created across layers which are coupled into: User accounts, Appointments & Scheduling, Pet Records and E-commerce. These are combined as they are planned to have similar functionalities, UI and modules in code. These domains also represent the abstractions of code, which will be materialized into modules further into the project.

8.2 Architectural Decisions

Justification for chosen architecture style:

The chosen architecture style for the VetCare system is the **N-tiered architecture style**. This decision was made because, according to the VetCare system's specifications, the quality attributes that the N-tiered architecture style can achieve are significantly more advantageous and essential than those of other popular architecture styles, such as microservices or monolithic (refer to section 1.2 Architecture decisions for more details).

For example, **data integrity breaches** are more difficult to occur in an N-tiered architecture. This is especially important as we are dealing with pet medical records that can be tampered with, posing a great threat to the lives of pets. Similarly, tampering of appointments and schedules can also result in significant problems for pet owners and clinics. In addition to ensuring data integrity, the N-tiered architecture system provides an overall, **secure system** that is vital for protecting other relevant aspects like login passwords and online payment credentials.

From the perspective of developing the Vetcare System, this architecture style comes in particularly handy for allowing the **reuse of components** across the codebase (example: the categorised domains as illustrated in the given architecture diagram are planned to have similar functionalities and UI). Furthermore, it is **easily scalable** allowing developers to add resources and features well after the system’s initial development. This, consequently, allows **ease of management** with regards to code development.

Technologies and frameworks

This section outlines the critical architectural decisions made during the design of the system

1. Layered Architecture:

    - Rationale: A layered architecture (N tier) separates concerns, making the system easier to manage, scale, and maintain, easy to maintain with small scale projects
    - Decision: Adopted a 3-layer architecture (Presentation, Business, Persistence) to handle the different aspects of the system.

2. Spring Boot as Web Framework:

    - Rationale: **Part of specification of project.**
    - Boot is a widely used, robust framework that supports rapid development and offers strong integration with various components such as databases, APIs, and security modules. It's auto-configuration process allows it to focus more on building features rather than configuring the infrastructure. Moreover, embedded servers like Tomcat, Jetty, or Undertow allow to run web applications directly without needing to deploy them to an external server. This makes it easier to develop, test, and deploy applications.
    - Decision: Use Spring Boot for building and deploying web services.

3. MySQL as the Primary Database:

    - Rationale: MySQL is a reliable, scalable, and widely supported relational database system that aligns well with the project's requirements for handling complex queries and ensuring data integrity.
    - **Our whole team has experience and expertise using MySQL.**
    - Decision: Utilise MySQL for managing the application's data, ensuring consistent performance and scalability.

4. Docker for Containerization:

- Rationale: **Part of specification of project**.

5. CI/CD with GitHub Actions:

    - Rationale: **Part of specification of project**. GitHub Actions also offers integrated CI/CD capabilities, enabling automated testing and deployment, which supports agile development practices.
    - Our whole team has experience and expertise using GitHub Decision: Use JUnit5 for writing and running unit tests, ensuring high code quality and reliability.

6. Thyme leaf for Server-Side Templating

- Rationale: Thyme leaf is a modern server-side Java template engine for web and standalone environments. It is well-suited for Spring Boot applications, providing natural templating, which is simple and intuitive for developers to use. Thyme leaf supports HTML5 and offers strong integration with Spring Boot, making it easier to develop dynamic web pages that bind seamlessly with the

backend data. This choice enhances the maintainability of the UI code while ensuring that the presentation layer is tightly integrated with the backend logic.
- Decision: Use Thyme leaf as the server-side templating engine for generating dynamic HTML views within the VetCare system

8. J Unit for testing: specified in instructions

- These decisions are aimed at building a scalable, maintainable, and secure system that meets the project's functional and non-functional requirements.

ADR 1: Rejection of Microservices Architecture

- **Title:** Microservices Architecture
- **Status**: Rejected
- **Context**: Microservices architecture was considered as an option for building the VetCare system. Microservices would allow us to break down the system into independent services, each with its own database and functionality. This could provide benefits such as independent scaling, deployment, and technology diversity.
- However, the team evaluated the complexity, operational overhead, and the project's scope. Given the current size and scope of the VetCare system, implementing microservices would introduce unnecessary complexity. The system doesn't require the level of scalability or independence that microservices are designed to support.
- **Decision**: We have decided to reject the use of microservices architecture in favour of a more straightforward layered architecture. This decision aligns with the current needs of the project, reducing complexity and focusing on delivering a maintainable and scalable system within the existing project constraints.
- **Consequences**:
- **Positive**:(Of rejecting microservices)
    - Reduced complexity in system design and development.
    - Easier to manage and deploy the system as a single monolithic application.
    - Lower operational overhead, as there is no need for managing multiple services, inter-service communication, or complex deployment pipelines.
    -
    - Decision: Implement CI/CD pipelines using GitHub Actions to automate the testing, building, and deployment processes, ensuring continuous integration and delivery.
- **Negative**:
    - Scalability could become a concern if the system grows beyond the expected scope.
    - Future enhancements may require significant refactoring if microservices become a better fit later on.

ADR 2: Choose MySQL as the Primary Database

- **Status**: Accepted
- **Context**: The system requires a reliable and scalable database to manage complex queries, ensure data integrity, and support various transactions. MySQL, being a proven relational database, fits these requirements and aligns well with the project's technology stack. Moreover our team has good experience in using MySQL.
- **Decision**: MySQL will be used as the primary relational database for the project.
- **Consequences**:

- **Easier**: Data integrity and performance will be consistent, with support for complex queries and transactions.
- **More Difficult**: MySQL's relational model may introduce complexity when handling highly flexible or unstructured data.

ADR 3: Reject PostgreSQL as the Primary Database

- **Status**: Rejected
- **Context**: PostgreSQL was considered as the primary database for the project due to its advanced features, including support for complex data types, powerful indexing, and strong ACID compliance. However, after evaluating the team's current expertise and the project timeline, it became evident that the adoption of PostgreSQL would introduce significant challenges. The team is not familiar with PostgreSQL, and the time required for learning and training would delay the project's progress.
- **Decision**: PostgreSQL will not be used as the primary database for the project. Instead, we will continue with a database that the team is already comfortable with, ensuring smoother development and faster delivery.
- **Consequences**:
- **Easier**:
    - **Faster Development**: By sticking with a database, the team is familiar with, such as MySQL, the development process will be faster, and the team can leverage existing knowledge and skills.
    - **Reduced Training Time**: The team can focus on development rather than spending significant time learning and mastering a new database system.
- **More Difficult**:

    - **Advanced Features**: The project may miss out on some of the advanced features offered by PostgreSQL, such as better support for complex queries and data types.
    - **Future Scalability**: While MySQL is reliable and scalable, PostgreSQL's advanced capabilities may have offered better scalability for certain complex use cases. However, these can be managed with the current database through careful design and optimization.

ADR 4: Adoption of Thyme leaf for Server-Side Templating

- **Status**: Accepted
- **Context**: The VetCare system requires a way to dynamically generate HTML views on the server side. The views will be used for the various UIs such as user management, appointment scheduling, and e-commerce pages. We needed a templating engine that integrates seamlessly with Spring Boot, supports modern HTML5 standards, and provides flexibility in handling the data sent to the views.

  Thyme leaf was considered as a suitable templating engine due to its strong integration with Spring Boot, ease of use, and the ability to create rich, dynamic web pages.

- **Decision**: We have decided to adopt Thyme leaf as the server-side templating engine for the VetCare system. Thyme leaf will be used to generate the HTML views dynamically, leveraging its features to bind model data directly to the UI components.
- **Consequences**:
- **Positive**:
    - Seamless integration with Spring Boot, reducing the time and effort needed for configuration.

- Supports modern HTML5 standards, making it easier to create responsive and well-structured web pages.
- Provides natural templating, which is easy for developers to read and maintain.
- Strong community support and extensive documentation, which aids in solving any issues that arise during development.

- **Negative**:
  - Performance overhead compared to client-side rendering, as the server needs to process the templates before sending the HTML to the client.
  - Limited dynamic capabilities compared to modern client-side frameworks like React or Angular, which might restrict the UI's responsiveness without additional client-side scripting.

ADR 5: Adoption of Twilio for Notification System

- **Status:** Accepted
- **Context:**

  The VetCare system requires a flexible notification system to manage various alerts and updates. Some examples include appointment confirmations, payment confirmation, clinical surgery updates of their pets, reminders for scheduled appointments and reminders for administering prescribed medication to their pets. The chosen notification system must support multi-channel communication (SMS, email, push notifications) to ensure users receive timely and reliable updates they would see.

  Twilio was evaluated as a suitable notification system due to its extensive support for multiple communication channels, reliable performance, and ease of integration with web applications. Twilio is also proven feasible for the project's current scope and the team members' skills. It offers a free tier that allows you to send a limited number of messages. This is more than enough considering the small-scale development of the vetcare system. It is also highly compatible with Spring Boot. There are official and community-supported libraries that make it easy to send messages from within a Spring Boot application.

- **Decision:**

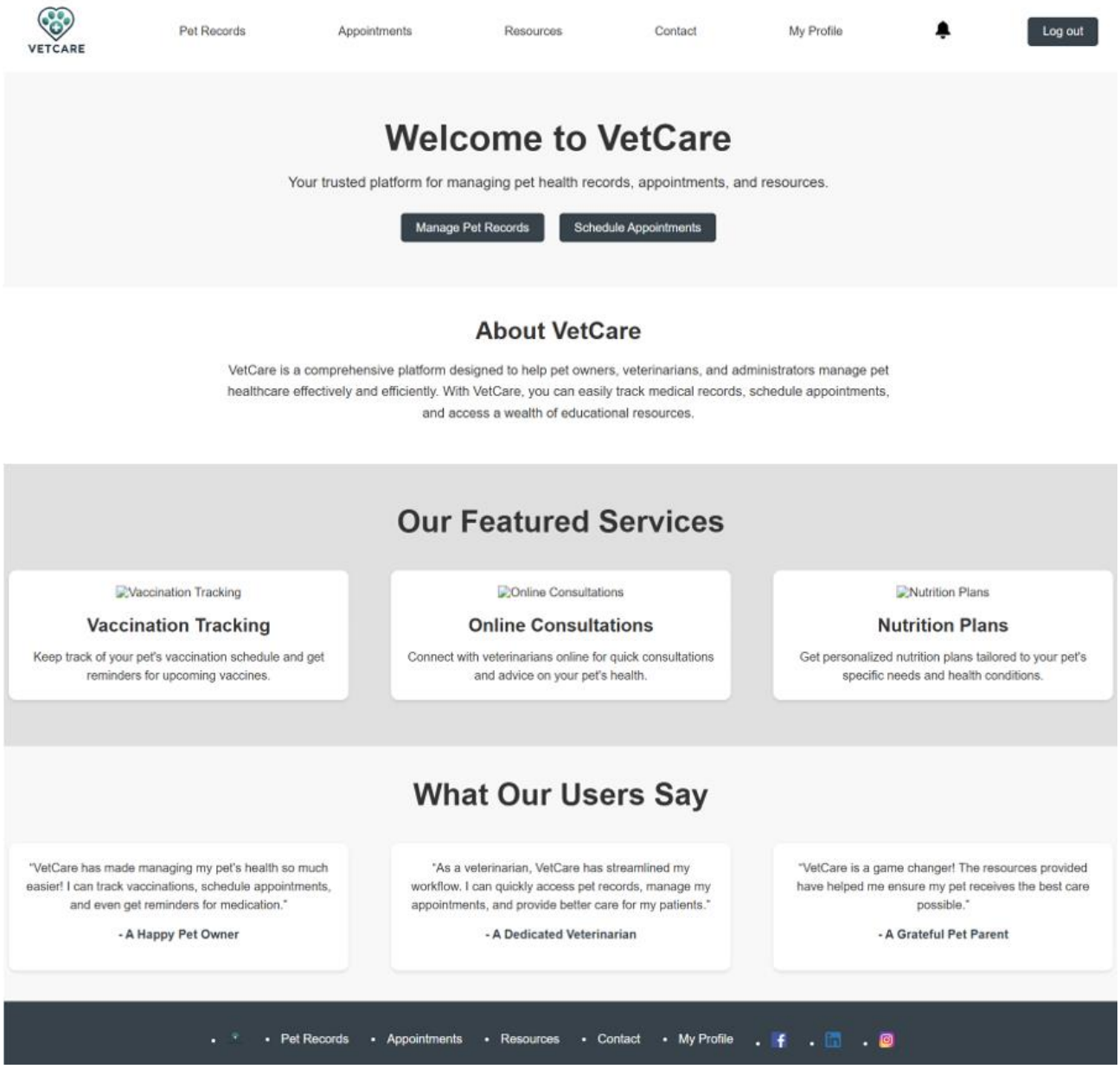
  We have decided to adopt Twilio as the notification system for the VetCare web application. Twilio will be used to send various notifications to pet owners, ensuring that all critical updates are communicated effectively across different channels (SMS, email, push notifications).

- **Consequences:**
- **Positive:**
  - **Multi-channel support**: Twilio provides comprehensive support for SMS, email, and push notifications, allowing the VetCare system to meet diverse user communication preferences.
  - **Ease of integration**: Twilio's API is well-compatible with the current web framework, Spring Boot, and it’s easy to integrate with the existing web application. This reduces the time and effort required for implementation.
  - **Easy to learn**: Twilio has well-detailed documentation and a strong developer community online to support our team’s attempts and help in case of shortcomings.
- **Negative:**
  - **Cost**: If the free tier is exhausted and there is a need to send more messages than what Twilio allows, the team might have to pay to scale up the notification system (however, this is unlikely due to the scale of the project and the team will be mindful of the number of messages we send, for testing).

- **Dependency on third-party service**: Relying on Twilio means the VetCare system will be dependent on a third-party service for critical communication functions, which could pose risks if there are service disruptions or changes in Twilio's offerings.

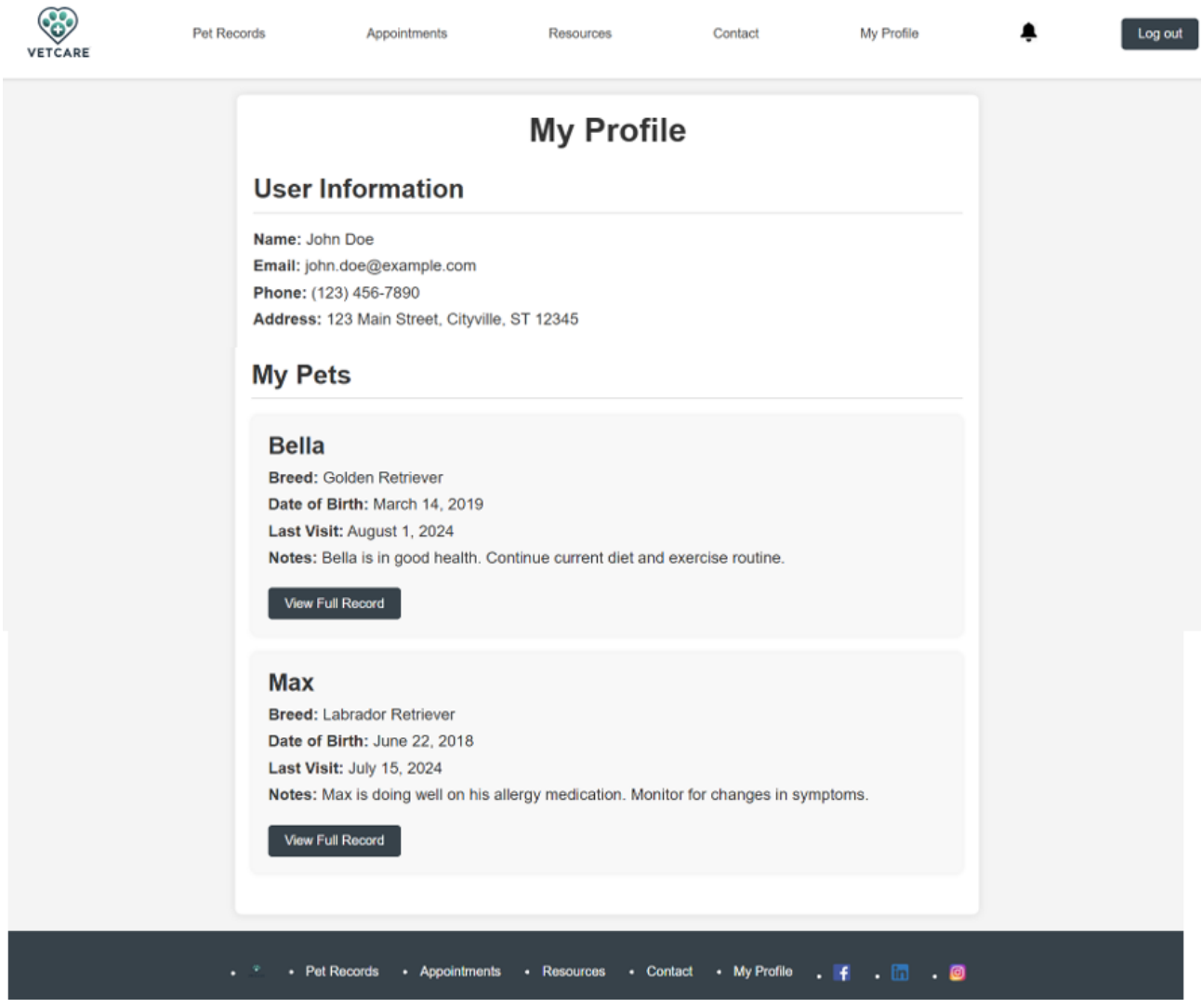
9. User Interface Design

Home Page



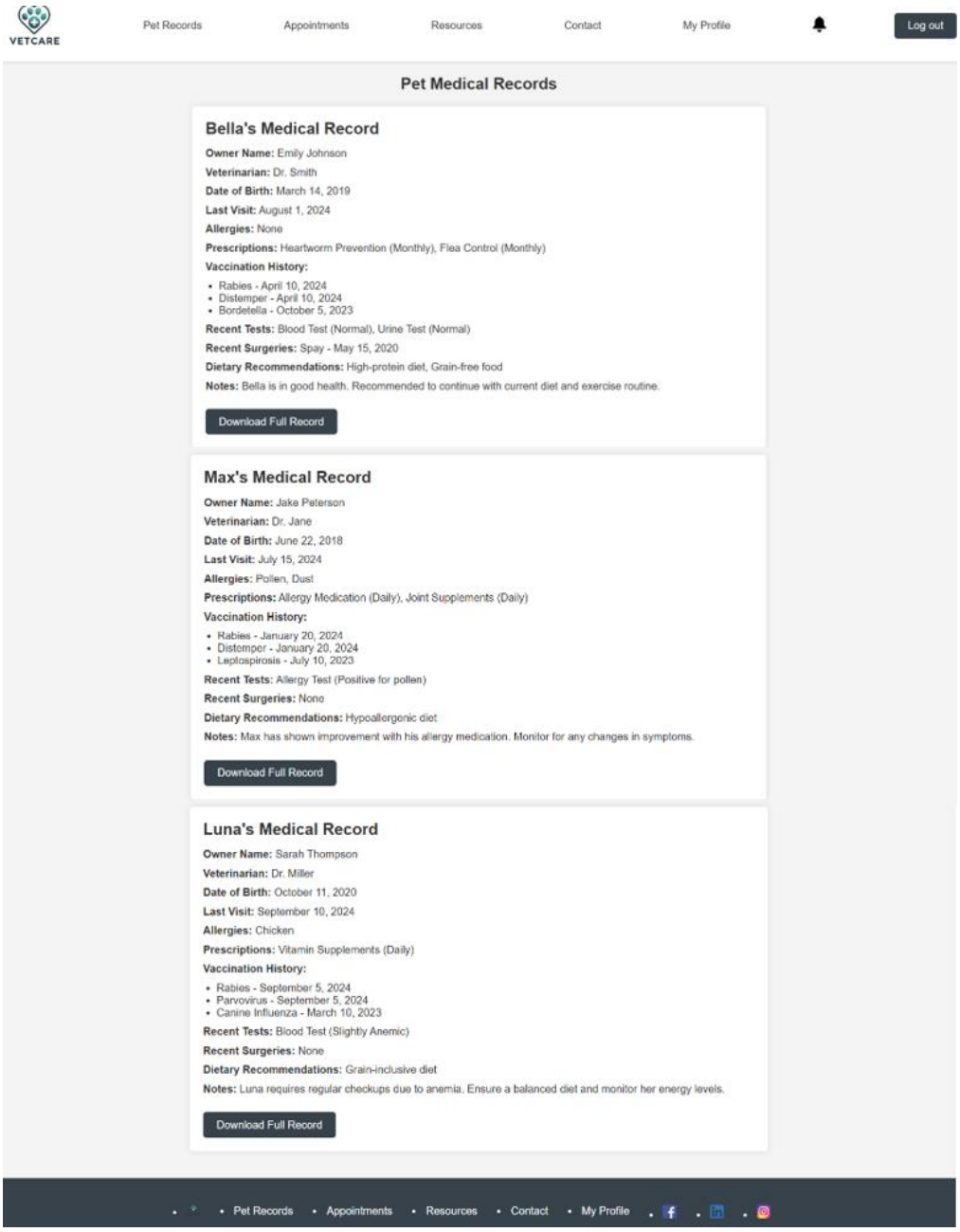
The VetCare homepage presents a clean and organised design which displays a navigation bar for easy access to key sections, a welcoming hero section with primary action buttons, along with a cool visual image. Under that are featured services like Vaccination Tracking, Online Consultations and nutrition plans. This is followed by an FAQ section addressing common user questions. The footer includes essential navigation links and social media icons, ensuring a cohesive and user-friendly experience throughout the site.

My Profile Page



This "My Profile" page mockup features a clean, structured layout with distinct sections for user and pet information. The header includes a navigation bar for easy access to various parts of the site, and a logout button is prominently displayed. User and pet details are presented clearly, with profile pictures aligned to the right, providing a balanced visual design. The footer includes links to essential pages and social media icons, reinforcing a consistent and user-friendly interface.

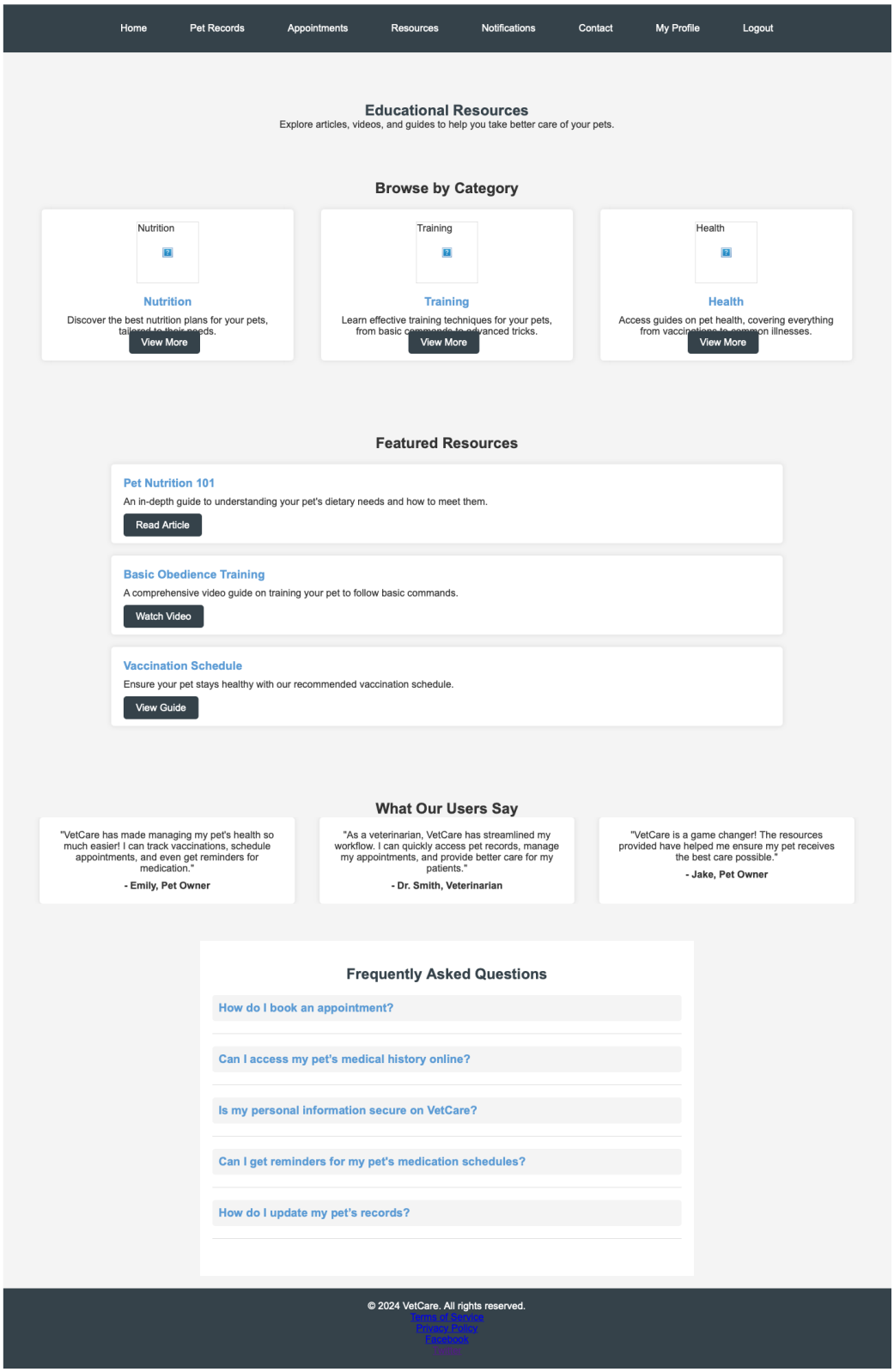
Pet Record Page



This "Pet Record" page mockup follows a similar structured and clean design as the "My Profile" page, with distinct sections for displaying pet record information. The header includes a navigation bar that allows users to access different sections of the site easily for user convenience. The pet record information is prominently displayed on the left, while a profile picture of the pet is aligned on the right, maintaining a balanced layout. Each section includes a "Download Record" button which enables users to download their pet's medical records. The footer is consistent with the rest of the site ensuring a cohesive user experience throughout the application.

My Appointments Page

Educational Resources page



The "Educational Resources" page is designed to provide pet owners with easy access to a wealth of knowledge regarding pet care. The page is structured to allow users to browse through various categories, such as Nutrition, Training, and Health, making it simple to find relevant information. The layout is clean and user-friendly, featuring both visual and textual elements to enhance comprehension. Users can access featured resources directly, ensuring that vital information is always within reach. This design supports VetCare's mission to empower pet owners with the knowledge they need to take better care of their pets.