**PET APPOINTMENT AND VETERINARY RECORD SYSTEM FOR PETLANDIA IN MALOLOS CITY BULACAN**

A Thesis Project Presented to

the Faculty of Datamex College of Saint Adeline, Inc.

In Partial Fulfillment of the Requirements for the

Degree of Bachelor of Science in Information Technology

By:

Amponin, John Alexis

Ausa, Justin

Dueñas, Jhazen Mackenzie

Geronimo, ReymarK

September 15, 2025

**REQUIREMENT**

**SPECIFICATION**

**INTRODUCTION**

This document aims to provide a description of the requirements for the development of the Pet Appointment and Veterinary Record System for Petlandia Veterinary Clinic in Malolos, Bulacan. It serves as a reference for the development team and management of the clinic to align the system under construction with the needs of the clinic. By defining the system's functionalities, limitations, and goals, this document helps to arrive at a lucid understanding of what is expected to be delivered by the system and how it will help to strengthen the existing processes.

The system has been developed as an independent application based on PHP, installed locally using XAMPP. In effect, the system shall replace the conventional manual approaches of appointment scheduling, tracking information about the owners, details about the pets, and keeping veterinary records that lead to scattered files and time-consuming searches. For this system, all information is stored in one secure database with accuracy and efficiency. There are two primary types of users for the application: Staff and Admin. Staff can create owner and pet records, book appointments, Admin users, meanwhile, have the most control of the system and are allowed to view totals, make updates to records, and delete incorrect entries. This differentiation of duties promotes a more organized workflow while reducing the chance of error and maintaining suitable oversight in the clinic.

This requirements document's scope includes specific use cases, data needs, assumptions, and limitations in addition to the system's functional and non-functional components. The clinic's primary functions—managing pet owners, pets, appointments, and veterinarian records—are the only things the system is designed to handle. It can operate independently within the clinic's computers and is not dependent on an internet connection because it is a stand-alone solution. The present scope didn't include an external function like the third-party integration, SMS/email notifications, and processing of the payment. The system is made to be scalable in the future, so if the clinic's requirements change or grow, more features can be added later.

**FUNCTIONAL REQUIREMENTS**

The functional requirements describe the specific features and capabilities of the system. They define what staff and admin users can do within the application, ensuring that the system supports the clinic’s daily operations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirement ID** | **Requirement Description** | **Priority** | **Dependencies** | **Acceptance Criteria** |
| FR-S1 | Staff can create owner records by entering full name, contact number, email, and address. | High | Database connection (owners table) | Owner record is successfully stored and viewable in the system. |
| FR-S2 | Staff can create pet records under an existing owner, including pet name, species, breed, gender, and age. | High | Owners table (must exist before adding pets) | Pet record is saved and linked to the correct owner. |
| FR-S3 | Staff can book appointments by selecting a pet, date, and reason. The system generates a unique authenticated code. | High | Pets and users’ tables | Appointment is saved and displays an authenticated code for reference. |
| FR-S4 | Staff can input veterinary records using an authenticated code, including diagnosis, treatment, medication, and remarks. | High | Appointment with valid authenticated code must exist | Veterinary record is stored and linked to the appointment. |
| FR-S5 | Staff can view a list of booked appointments with owner, pet, date, and status. | Medium | Appointments table | Appointment list is displayed in chronological order. |
| FR-A1 | Admin can view dashboard summaries showing totals for appointments, veterinary records, owners, and pets. | High | All tables must be accessible | Dashboard displays accurate counts in real time. |
| FR-A2 | Admin can update appointment, and veterinary records. | High | Existing records must be available | Updated records are reflected immediately in the system. |
| FR-A3 | Admin can delete appointment, and veterinary records. | High | Existing records must be available | Deleted records are permanently removed and no longer visible. |
| FR-A4 | Admin can search and filter records to quickly find information. | Medium | Database indexing | Search returns correct results based on keywords or filters. |
| FR-A5 | Admin can manage staff accounts (add, edit, or remove) if required by the clinic. | Low | Users table | Admin successfully modifies staff accounts. |

*Table 1. Functional Requirements*

**NON-FUNCTIONAL REQUIREMENTS**

The non-functional requirements define how the system should operate rather than what it should do. They describe the qualities that make the system efficient, secure, and user-friendly.

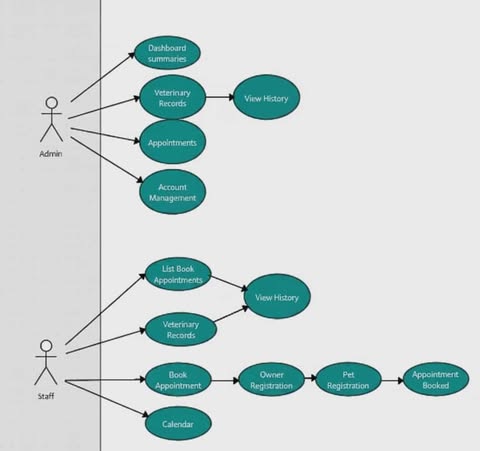
|  |  |
| --- | --- |
| **Category** | **Requirement Description** |
| **Performance** | The system should respond within seconds when loading appointment lists, veterinary records, or dashboards. It should handle multiple records efficiently without significant delays. |
| **Usability** | The interface should be simple, intuitive, and easy to navigate for both staff and admin users. Forms must be organized logically to minimize user errors. |
| **Reliability** | The system should ensure that records are stored accurately and remain accessible as long as the database is intact. It should recover gracefully from unexpected shutdowns without data loss. |
| **Security** | Only authenticated users (staff and admin) should be able to access the system. Passwords must be encrypted, and sensitive data such as veterinary records should be protected against unauthorized access. |
| **Scalability** | Although designed as a stand-alone system, the database structure should allow the addition of more features, records, or user roles in the future without major redesign. |
| **Maintainability** | The system’s code and database should be structured clearly, making it easy for future developers to update, fix, or expand the system if needed. |

*Table 2. Non-Functional Requirements*

**USE CASES**

The use cases describe how different users interact with the Pet Appointment and Veterinary Record System. They define the step-by-step scenarios in which staff and admin perform their tasks, such as booking appointments, managing records, and viewing summaries. By outlining these use cases, the system’s functionality becomes clearer, ensuring that both user roles are properly supported in their responsibilities.

**USE CASE DIAGRAM- PET APPOINTMENT AND VETERINARY RECORD SYSTEM**

****

*Figure 1. Use Cases*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | **Use Case Name** | **Description** | **Actors** | **Preconditions** | **Postconditions** | **Alternate Flows** |
| UC-1 | Book Appointment | Staff books an appointment by selecting a pet, date, and reason. The system generates an authenticated code. | Staff | Owner and pet records must exist; staff must be logged in. | Appointment is saved and linked with a generated authenticated code. | If no pet record exists, staff must first create one. |
| UC-2 | Enter Veterinary Records | Staff inputs diagnosis, treatment, medication, and remarks using an authenticated code from an appointment. | Staff | Appointment with a valid authenticated code must exist. | Veterinary record is stored and linked to the correct appointment. | If code is invalid, the system rejects the entry. |
| UC-3 | View Appointment List | Staff views a list of booked appointments with owner, pet, date, and status. | Staff | Staff must be logged in. | Appointment list is displayed in order. | If no appointments exist, the list appears empty. |
| UC-4 | View Dashboard Summaries | Admin views totals for appointments, records, owners, and pets. | Admin | Admin must be logged in. | Dashboard displays updated counts. | None. |
| UC-5 | Update Records | Admin updates veterinary records. | Admin | Auth code must already exist in the system. | Updated record replaces old data. | If record change auth code, an error message is shown. |
| UC-6 | Delete Records | Admin deletes outdated or incorrect records. | Admin | Record must exist in the system. | Record is permanently removed. | None |
| UC-7 | Search and Filter Records | Admin searches for and filters appointment records. | Admin | Admin must be logged in. | System displays records matching criteria. | If no match is found, the system shows “No results found.” |
| UC-8 | Manage Staff Accounts | Admin manages staff user accounts (add, edit, delete). | Admin | Users table must exist; admin must be logged in. | Staff account changes are saved. | If username already exists, system rejects creation. |
| UC-9 | Create Owner Records | Staff creates owner records with full name, contact, email, address, and date appointment | Staff | Staff must be logged in. | Owner record is saved in the database. | If mandatory fields are empty, system rejects the entry. |
| UC-10 | Create Pet Records | Staff registers pets under an owner with details like name, species, breed, gender, age. | Staff | Owner record must exist. | Pet record is saved and linked to the correct owner. | If owner does not exist, system prevents entry. |

*Table 3. Use Cases*

**DATA REQUIREMENTS**

The Pet Appointment and Veterinary Record System relies on structured data to function effectively. This section outlines the key data entities, their attributes, and the relationships between them. Clearly defining these requirements ensures that information about users, owners, pets, appointments, and veterinary records is stored consistently and linked accurately within the database.

|  |  |
| --- | --- |
| Data Entity | Attributes |
| Users | user\_id (PK), username, password, role (admin/staff), created\_at |
| Owners | owner\_id (PK), full\_name, contact\_number, email, address, created\_at |
| Pets | pet\_id (PK), owner\_id (FK), pet\_name, species, breed, gender, age, created\_at |
| Appointments | appointment\_id (PK), pet\_id (FK), appointment\_date, reason, status, auth\_code (unique), created\_by (FK), created\_at |
| Veterinary Records | record\_id (PK), auth\_code (FK), diagnosis, treatment, medication, remarks, recorded\_by, created\_at |

*Table 4. Data Requirements*

**Relationships Between Entities**

1. **Owners ↔ Pets**
   * One-to-One Relationship.
   * Each owner is linked to exactly one pet, and each pet belongs to only one owner.
2. **Pets ↔ Appointments**
   * One-to-One Relationship.
   * Each pet can only have one appointment at a time. To book another, a new appointment replaces the old one.
3. **Appointments ↔ Veterinary Records**
   * One-to-One Relationship.
   * Each appointment generates exactly one veterinary record.
4. **Users → Appointments**
   * One-to-Many Relationship.
   * A staff user can create multiple appointments, but each appointment is created by only one user.

**ASSUMPTIONS AND CONSTRAINTS**

The development of the Pet Appointment and Veterinary Record System is guided by certain assumptions and limited by specific constraints. These assumptions ensure that the system is designed with realistic expectations of its users and environment, while the constraints define the boundaries within which the system must operate. Clearly identifying these factors helps maintain focus and ensures that the final product meets the clinic’s actual needs

**Assumptions**

* The system will be deployed as a stand-alone application within Petlandia Veterinary Clinic and does not require internet connectivity.
* Users (staff and admin) have basic knowledge of using a computer and web-based forms.
* All data entered into the system (e.g., owner details, pet details, veterinary records) is assumed to be accurate and provided by the clinic staff.
* The clinic operates under the rule of one owner per pet, meaning each new pet must be registered with a new owner record.

**Constraints**

* The system is limited to two user roles: Staff and Admin. No additional roles are included in this version.
* The system only runs on computers with XAMPP (Apache, MySQL, PHP, phpMyAdmin) properly installed.
* External integrations such as online payment processing, SMS/email notifications, or cloud backup are outside the scope of this project.
* Since the system is stand-alone, multi-branch or remote access is not supported. It can only be accessed within the local environment where it is installed.
* Database relationships are designed as mostly one-to-one, except for users to appointments, which follows a one-to-many structure.

**GLOSSARY**

**Admin** – A user role with full access to the system. Admins can view summaries, update and delete records, manage staff accounts, and oversee the entire system.

**Staff** – A user role responsible for creating owner records, pet records, booking appointments, and entering veterinary records. Staff users have limited permissions compared to admins.

**Owner Record** – Information about a pet’s owner, including full name, contact details, email, and address. In this system, one owner corresponds to exactly one pet.

**Pet Record** – Information about a pet, such as name, species, breed, gender, and age. Each pet is linked to a single owner.

**Appointment** – A scheduled booking for a pet to visit the clinic, created by staff. Each appointment is tied to one pet and generates an authenticated code.

**Veterinary Record** – Medical information recorded during or after an appointment, including diagnosis, treatment, medication, and remarks. Each appointment corresponds to one veterinary record.

**Dashboard** – The admin’s main interface, displaying totals of appointments, veterinary records, owners, and pets. It also provides access to management functions.

**Stand-alone System** – A software application that operates independently on a local computer without requiring internet or external systems.

**XAMPP** – A free and open-source platform that provides Apache, MySQL, PHP, and phpMyAdmin, used to run and manage the system locally.