

PET VISIT RECORD SYSTEM FOR PETLINK CALOOCAN

A Project Proposal 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:

Perez, Jocella Mae V.

October 2025

TECHNICAL DOCUMENT

INTRODUCTION

The goal of this Technical Documentation is to clearly describe the technical design and detailed specifications for the Pet Visit Record System (PVRS) built for PetLink Caloocan. This document acts as the main guide for the development team and technical staff, providing deep details on how the system will be built, set up, and maintained over time. It ensures everyone understands how the system works, turning the clinic's business needs into exact technical steps.

The PVRS is a vital web-based program that will update clinic work by changing the old manual, paper-based way of keeping pet records to a modern digital system. It's made to make work flow smoothly, keep data accurate, and lower mistakes for all clinic staff. The system helps staff book and change appointments, record walk-in pets, check a pet's full visit history, and create important reports.

This full guide covers all important technical parts of the PVRS: the system's structure, the database layout, the user interface design, the different system features (modules), how data moves, security features, performance goals, installation steps, and plans for maintenance.

SYSTEM OVERVIEW

The structure of the Pet Visit Record System (PVRs) is based on a three-tier client-server setup. This smart design carefully separates the whole system into three main parts: the Presentation Layer, the Application Layer, and the Data Layer. This separation is key to making sure the system can grow easily (scalability), is simple to fix and update (maintainability), and can handle new features later (flexibility). Because of this design, the PVRs can manage all pet visit data and clinic tasks well, keeping patient information safe and correct while being easy to use for all clinic staff.

High-Level Components and Their Interactions: The system is divided into two main parts: the front-end and the back-end.

Front-End (Presentation Layer) - This is the interface that users see and interact with. It will be a web browser interface that is responsible for displaying information and collecting user input from clinic staff.

Back-End (Application & Data Layers) - This part contains the core logic of the system and all the stored data. It is divided into the Application Layer, which handles all business rules and processes, and the Data Layer, which is a relational database that stores all information, including pet records, owner information, and visit logs

Data moves in a planned way: it goes from the Front-End, through the Application Layer for processing, to the Data Layer for saving or looking up information, and then the results flow back to the Front-End to be displayed

Deployment Architecture

The system follows a client-server model for its setup. This means the main application and database will be installed and deployed on a local server or a dedicated, powerful desktop computer within the clinic. Clinic staff access the system from their workstations (the clients) using a web browser, which connects directly to this central server via the local network.

INSTALLATION GUIDE

This section outlines the basic steps and requirements needed to install the Pet Visit Record System (PVRs) on the clinic's computer system. The goal is to successfully set up and start the application on the server.

System Requirements (Hardware, Software, Dependencies):

Hardware: The server or desktop running the system should have a minimum of 8GB of RAM and an i5 8th gen processor.

Software/Dependencies: Required tools include a local environment program like XAMPP for the server setup and Git for managing and updating the application's code.

Step-by-step instructions for installing the software:

Step 1: Pre-Deployment Setup

1. **Check Hardware** - Make sure the server or desktop has at least 8GB of RAM and an i5 8th gen processor.
2. **Install Required Software:** Install all necessary software, including the local environment tool XAMPP and the version control system Git.
3. **Ensure Network:** Check that the network connection is working and the system can communicate with all client computers.
4. **Backup Data (If Needed):** If the clinic has any existing digital records, create a backup before continuing.
5. **Set Up Database:** Install and set up the relational database (the Data Layer) on the server, ready to accept the PVRs data.

Step 2: Deployment Execution

1. **Deploy Application Files** - Move all the PVRs program files (the Back-End and Front-End code) from the development source onto the local server's correct folder.
2. **Configure System Settings** - Access the system's configuration files to set up key connections:
 - Enter the correct server address and login details for the database connection.
 - Set up any needed API keys for future use.
 - Define all other environment variables that control how the system works.
3. **Perform System Initialization** - Run the final setup steps to load initial data, create necessary folders, and start the application services.
4. **Check for Errors** - Immediately check the server logs and system interface for any startup problems or errors.

CONFIGURATION GUIDE

The Configuration Guide is a key document that gives clear instructions on how to set up the system's main settings right after installation. This step is essential to make sure the software works exactly as needed for the clinic's daily tasks.

The guide covers three main areas:

This guide ensures the PVRS runs with the correct permissions, connects to the right data, and follows all necessary clinic rules.

1. **Detailed Instructions for Configuring the Software:** After the PVRS is installed, all system settings must be carefully set up to match how the clinic works. This ensures the system runs with the correct user permissions, connects to the right database, and follows the necessary clinic rules.
2. **Configuration File Formats and Parameters:** This section describes the types of files and information needed to set up key connections. This specifically covers entering the correct information for database connections, any external service credentials like API keys, and necessary environment variables that tell the application how to behave in the local setting.
3. **Best Practices for Customization:** The best way to set up the system is to focus on securing access. This means using the User Management module to set up role-based access permissions. By giving staff and administrators different rights, you protect sensitive data and ensure everyone only accesses what they need.

API DOCUMENTATION

This section formally defines the Application Programming Interfaces (APIs) of the Pet Visit Record System (PVRs), detailing the communication protocols that facilitate the core functions within its three-tier architecture.

List of APIs Exposed by the System

The PVRs exposes a set of programmatic interfaces that enable the Presentation Layer to interact seamlessly with the Application Layer. These APIs are essential for executing all system functions, such as creating appointments, retrieving comprehensive pet records, and executing reporting commands, thereby serving as the primary communication bridge across all functional components.

Endpoint URLs, Request/Response Formats, and Parameters

This subsection specifies the necessary technical details for each service call. It will include the Uniform Resource Locators (URLs) for specific endpoints (e.g., /api/v1/appointments/schedule), the required data structure for incoming requests (parameters), and the expected format of the resulting data sent back to the client (response format). The specific implementation details of these endpoints are determined by the Application Layer's programming logic and will be documented upon finalization of the code.

Authentication and Authorization Requirements

Security is enforced through strict access controls on all API functions. Access to critical endpoints is secured by the User Management module. This measure mandates that all accessing users must successfully authenticate and possess the necessary role-based access credentials (permissions) to execute the specific function. This security protocol is crucial for ensuring the confidentiality and integrity of sensitive patient data by preventing unauthorized access or manipulation.

DATABASE DOCUMENTATION

The Pet Visit Record System (PVRs) is built from several software components, each designed to handle a specific function of the clinic's workflow.

Entity-Relationship Diagram (ERD) and Schema

The database schema illustrates the relationships between the system's core entities. The key entities include Owners, Pets, Visits, and Staff. The ERD visually confirms how these entities connect to one another to ensure a structured and functional data model.

Description of Database Tables, Fields, and Relationships

The database is designed using a relational model and is normalized up to the Third Normal Form (3NF). This level of normalization is applied to reduce data redundancy and guarantee data integrity and consistency. Key relationships within the schema include:

- A One-to-Many relationship between the Owner entity and the Pet entity (one owner can have many pets).
- A One-to-Many relationship between the Pet entity and the Visit entity (one pet can have many visits over time).

Data Migration and Backup Procedures

To ensure both a smooth launch and long-term data safety, specific procedures for data management are mandatory.

- Data Migration: The initial deployment requires the manual transfer of existing paper-based records into the new digital database. This ensures a complete patient history is available at launch.
- Backup & Recovery: Ongoing procedures are in place for creating and storing regular backups of the database (the Backup & Recovery Plan) to prevent data loss in case of a system failure.

USER MANUAL

The primary instruction for staff is to use the software's user-friendly interface to manage all daily clinic operations. The manual is designed to simplify complex digital processes into clear, easy-to-follow steps, allowing staff to quickly become proficient in tasks like patient check-in, record retrieval, and scheduling. The PVRs feature a clean layout with consistent navigation menus to minimize confusion, and the interface is intentionally structured so that the most frequently used features are immediately accessible. Navigation guidelines will walk users through accessing different System Modules via the main menu, ensuring quick and logical movement between tasks. This section details the step-by-step procedures for the system's core functional workflows: the Dashboard Overview provides immediate instructions on how to interpret the main screen for a summary of the day's tasks; Appointment Management procedures cover booking, rescheduling, and viewing the clinic's calendar; Pet Records (Visit History) explains how to search for a pet and access its complete historical log of past treatments; and finally, the Reporting workflow guides users on generating essential summaries, such as the total number of patients or visit trends over a specified period.

TROUBLESHOOTING GUIDE

This section formally defines the procedures for identifying, reporting, and resolving issues within the Pet Visit Record System (PVRs), ensuring minimal downtime and a stable operating environment.

Common Issues and Error Messages

The project anticipates and addresses key risks, including the potential for technical issues or bugs that could cause the system to crash or function incorrectly, leading to downtime and operational disruptions. This guide lists common error messages and system faults that may occur during regular use.

Troubleshooting Steps and Resolutions

A clear, structured process is mandated for handling all software defects to ensure timely resolution:

1. **Issue Reporting:** Clinic staff are strictly instructed to report all observed errors and technical issues in a shared document (e.g., an Issue Log). This ensures a centralized record of all problems.
2. **Issue Prioritization:** A System Analyst reviews the reported problems and assigns a priority level to each based on its severity and impact on clinic operations.
3. **Resolution and Implementation:** A Developer then addresses the prioritized issues by implementing the necessary bug fixes (Corrective Maintenance).

Contact Information for Technical Support

Technical assistance and support are provided to the clinic immediately following deployment. This includes a dedicated three-month support period to resolve initial problems and assist staff in fully adapting to the system. Contact details for the support team will be made available to staff through the system's user manual.

CODE DOCUMENTATION

1. Code Structure and Organization

a. script.js (Frontend Logic)

- A large, event-driven JavaScript module that manages the entire web application's client-side behavior.
- Organized around **modular functional sections**, including:
 - **Session Management** (startInactivityTimer, extendSession, handleLogout)
 - **Authentication** (handleLogin, showLogin, showApp)
 - **Data Loading** (loadInitialData, loadAppData, fetchAndRenderAppointments)
 - **UI Rendering** (tables, modals, dashboards)
 - **Form Handlers** for CRUD operations (Users, Clients, Pets, Appointments)
 - **Action Logging and Recycle Bin Management**
 - **Reports/Analytics** (Chart.js integration for visit statistics)
 - **Routing** within SPA-style navigation (navigateToPage)
- The code executes after the DOMContentLoaded event, ensuring DOM elements are available.

b. generate_hash.php (Backend Utility)

- A simple PHP script used to **generate password hashes** securely.
- Implements PHP's password_hash() with the PASSWORD_DEFAULT algorithm, ensuring future compatibility.

2. Inline Comments and Logic Explanation

Session Timeout Logic (script.js)

```
function startInactivityTimer() {  
  // Clear any existing timers  
  clearTimeout(inactivityTimer);  
  clearInterval(countdownInterval);  
  
  // Hide the modal in case it was left open  
  document.getElementById('session-timeout-modal').classList.remove('active');  
  
  const warningTimerDuration = Math.max(0, (sessionTimeoutSeconds - warningTimeSeconds) *  
1000);  
  inactivityTimer = setTimeout(showTimeoutWarning, warningTimerDuration);  
}
```

Explanation:

- Resets any previous timers to avoid duplicate warnings.
- Hides any active timeout modals.
- Sets a delayed timer to show a warning before session expiry.

CRUD Handler (Clients)

```
async function handleClientFormSubmit(e) {
  e.preventDefault();
  const clientData = { fullName, phone, email, address };
  const method = clientId ? 'PUT' : 'POST';
  const url = 'api/clients.php';
  // Sends request to backend API with CSRF protection
  const response = await fetch(url, { method, headers, body });
}
```

Explanation:

- Prevents default form submission.
- Chooses between POST (create) or PUT (update).
- Sends data via fetch() with CSRF token for security.

Modular UI Rendering

```
function renderPetRecords(patients) {
  if (!patients.length) {
    petRecordsGrid.innerHTML = `<p>No pet records found.</p>`;
    return;
  }
  // Dynamically builds pet record cards with icons based on species
}
```

Explanation:

- Handles conditional rendering for empty states.
- Creates HTML dynamically for each patient record.

3. Coding Standards and Conventions**General**

- Uses **modern ES6+ features** (arrow functions, async/await, const/let).
- Follows **functional modular design** — each function handles a specific responsibility.
- All main actions (login, CRUD, navigation) are event-driven and consistently registered through attachEventListeners().

Naming Conventions

- **Variables:** camelCase (e.g., csrfToken, petToDeleteId)
- **Functions:** Verb-first naming for clarity (e.g., handleLogin, renderPetRecords)
- **Constants:** ALL_CAPS for config-like data (sessionTimeoutSeconds)
- **DOM Elements:** Prefixed semantically (clientForm, petRecordsGrid, etc.)
- **API Routes:** Stored as string paths, e.g. 'api/clients.php', maintaining backend consistency.

Comments and Documentation

- Inline comments exist in key areas, especially session management and fetch operations.
- Complex logic (like event delegation and rendering) is often self-explanatory through descriptive naming.

Error Handling

- Robust use of try...catch for all asynchronous operations.
- User feedback via alerts and console logs for debugging.
- Graceful fallbacks when data is missing or responses fail.

Security Practices

- Implements CSRF token usage for all API calls.
- Session timeout and inactivity handling.
- Server-side hashing (in PHP) ensures password safety.
- Prevents self-deletion of user accounts.

UI/UX Consistency

- SPA-like navigation without reloading.
- Dynamic modals for CRUD.
- Placeholder text for empty states.
- Pagination, search, and filter functions for better usability.

TESTING DOCUMENTATION

This section formally defines the testing framework for the Pet Visit Record System (PVRS), outlining the strategic approach used to ensure the system is reliable, meets all documented requirements, and is ready for clinical deployment.

Testing Documentation

This section formally defines the testing framework for the Pet Visit Record System (PVRS), outlining the strategic approach used to ensure the system is reliable, meets all documented requirements, and is ready for clinical deployment.

Test Plan Outlining Testing Objectives and Strategies

The test plan defines the official quality assurance goals for the PVRS. The main objective is to rigorously verify that the developed software fully meets all specified functional and non-functional requirements detailed in the design document. The strategy focuses on testing all major System Modules—including Appointment Management, Patient Records, and Reporting—to confirm accurate performance and adherence to security protocols.

Test Cases Covering Functional and Non-Functional Requirements

Specific Test Cases are developed for execution to prove the system's correctness. These scenarios cover both:

- **Functional Requirements:** Verifying that core business logic, such as successfully scheduling an appointment or generating a report, works as intended.
- **Non-Functional Requirements:** Verifying qualities like performance (speed), security (role-based access control), and usability.

These test cases are crucial during the Post-Deployment Steps to formally Verify Functionality immediately after the system is installed.

Test Results and Defect Reports

All testing activities generate documented evidence. This includes detailed Test Results, which record the outcome (Pass/Fail) of every test case executed. Any issues or failures are logged as Defect Reports, which contain a description of the error, steps to reproduce it, its severity level, and the current status (e.g., In Progress, Fixed). This process feeds directly into the Troubleshooting Guide and Maintenance Guide for formal bug resolution.

MAINTENANCE GUIDE

This section defines the mandatory procedures and protocols for maintaining and updating the Pet Visit Record System (PVRS) after its deployment, ensuring its long-term stability and optimal performance.

Procedures for Maintaining and Updating the Software

Maintenance activities are systematically categorized into four standard types to comprehensively address the system's needs:

- **Corrective Maintenance:** Focused on implementing immediate bug fixes and addressing errors reported through the Issue Tracking system.
- **Adaptive Maintenance:** Involves modifying the software to accommodate changes in the operating environment, such as new software versions or updated security standards.
- **Perfective Maintenance:** Dedicated to system optimizations and enhancements to improve efficiency, speed, or user experience.
- **Preventive Maintenance:** Consists of regular updates and checks designed to anticipate and prevent potential future issues before they disrupt operations.

Version Control and Release Management Practices

All code management will utilize Git as the designated tool for version control. This ensures a complete and traceable history of all code changes. Future updates and enhancements will be implemented through a structured release management process, with the prioritization of new features being based directly on feedback collected from the clinic. This feedback loop ensures that the system evolves in alignment with the clinic's practical operational needs.

Guidelines for Handling Bug Fixes and Enhancements

The process for addressing both bugs and new feature requests strictly follows the structured issue reporting and prioritization system defined in the Troubleshooting Guide (Section 8). This methodology mandates that all reported issues are logged, reviewed by a System Analyst for priority assignment, and then scheduled for resolution by a Developer to maintain accountability and a clear resolution timeline.

REVISION HISTORY

The Pet Visit Record System (PVRS) design document will maintain a revision history to track all changes and updates throughout its lifecycle. This practice ensures a transparent record of all modifications, from minor corrections to significant feature additions. The revision history table, as outlined in the document, is a crucial component for project management and accountability. The purpose of this history is to serve as a detailed log of the document's evolution, helping stakeholders and developers understand the rationale behind changes and ensuring that everyone is working with the most current version. By tracking modifications, it also facilitates a structured maintenance process and aids in troubleshooting or reviewing past decisions, which is essential for maintaining the integrity and accuracy of the design document over time.

Revision No.	Description	Date
Version 1.0	Technical Documentation	October 2025

APPENDIX

References (APA Style)

College of Computer and Information Sciences, Majmaah University. (2020). *Veterinary management system: Final year project report*.

<https://m.mu.edu.sa/sites/default/files/content/2020/09/final%20report%20Veterinary-RE.pdf>

PetLink Veterinary Clinic. (n.d.). *Clinic locations*. PetLink Vet PH. <https://www.petlinkvetph.com/clinic-locations>

PetLink Veterinary Clinic. (n.d.). *Our services*. PetLink Vet PH. <https://www.petlinkvetph.com/our-services>

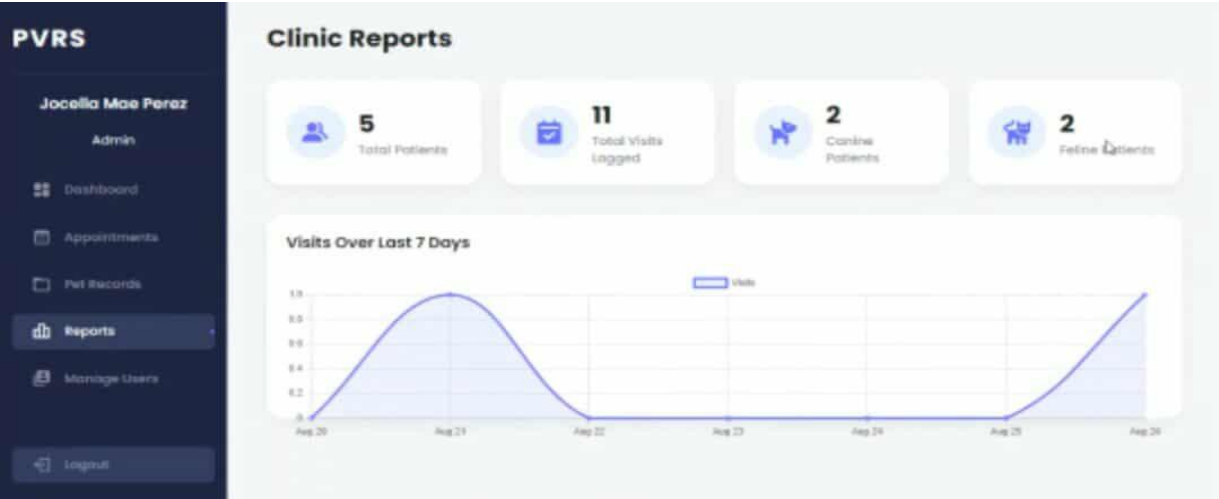
PetLink 10th Avenue – Caloocan. (n.d.). [Facebook page]. Facebook. <https://www.facebook.com/petlink10thAveCaloocan/>

u/milkmacchiato. (2024, July 7). *South Caloocan vet clinic recommendation* [Post]. Reddit. https://www.reddit.com/r/catsofrph/comments/1f7wlx8/south_caloocan_vet_clinic_recommendation/

Technaureus Info Solutions. (n.d.). *Pet management software*. <https://www.technaureus.com/product-details/pet-management-software>

Supporting Materials:






PVRS

Jocella Mae Perez
Admin

- Dashboard
- Appointments
- Pet Records
- Reports
- Manage Users**
- Logout

Manage Users

[+ New User](#)

FULL NAME	USERNAME	ROLE	ACTIONS
Jocella Mae Perez	admin	admin	
Brix Lumanog	staff	staff	