**A PROPOSED OFFERING OF WEB BASED PET MEDICAL RECORDS SYSTEMFOR PETLINK VETERINARY CLINIC, 10TH AVENUE, CALOOCAN CITY**

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:

Mamay, John Carlo P.

Remoto, Rachel

Solidum, Jasmin E.

August 2025

**DESIGN DOCUMENT**

**INTRODUCTION**

The purpose of this design document is to provide a clear and comprehensive plan for the development of the **Pet Medical Records System**. It explains how the system will be designed based on the requirements and outlines the structure that will guide developers during implementation.

**Overview**

The system is a web-based application designed for veterinary clinics to assist staff in managing pet information and medical histories in a more reliable and organized manner. It is designed with a frontend built using HTML, CSS, JavaScriptfor the user interface, The backend is developed by Node.js or Python with APIs, which handle the processing of data, system logic, and communication between the frontend and database while SQLite serves as the database solution.

The system provides a straightforward yet effective interface where staff can register pet owners and their pets, record medical check-ups, treatments, and vaccinations, and generate accurate medical records when needed. To ensure accountability, Role is included, allowing staff to manage accounts and monitor all recorded activities within the system.

Overall, the design focuses on replacing manual paper-based processes with a secure, user-friendly, and digital solution that supports the daily operations of a veterinary clinic. It ensures data integrity, improves accessibility, and enhances the efficiency of managing pet health records.

**Scope**

The scope focuses on the system’s core features, which include user management where the can create, edit, and monitor staff accounts; pet and owner records where staff can register detailed profiles; medical records that cover consultations, diagnoses, treatments, prescriptions, and vaccination history; report generation for daily consultations, vaccination lists, and treatment summaries; and search and retrieval functions for quick access to records. Features beyond this scope, such as online appointment scheduling, billing integration, and mobile access, are not part of the current version but may be considered in future upgrades. By setting these boundaries, the document ensures that development efforts remain focused on delivering a secure, efficient, and user-friendly solution for veterinary clinics.

**SYSTEM ARCHITECTURE**

The system uses a client-server architecture where veterinary staff access the web app via a browser and interact with the backend server. The backend handles authentication, medical record processing, and database communication for secure data storage. Manages staff accounts and monitors activities, while staff record pet and medical information.

**High Level Components and Interactions**

* **Frontend:** Built with HTML, CSS, JavaScript; lets staff register pets/owners, record medical histories, manage treatments, and generate/search reports.
* **Backend:** Manages authentication, role-based access, business logic, and database communication using Node.js/Python/APIs.
* **Database:** SQLite stores pet, owner, staff, and medical records with lightweight but reliable management.
* **Dashboard:** To manage staff accounts, monitor logs, and access reports.

**Deployment Architecture**

* **Frontend:** Hosted on a server, accessible via web browsers.
* **Backend:** Runs on a server and connects frontend with the database.
* **Database:** Deployed locally with the backend, scalable for future upgrades.

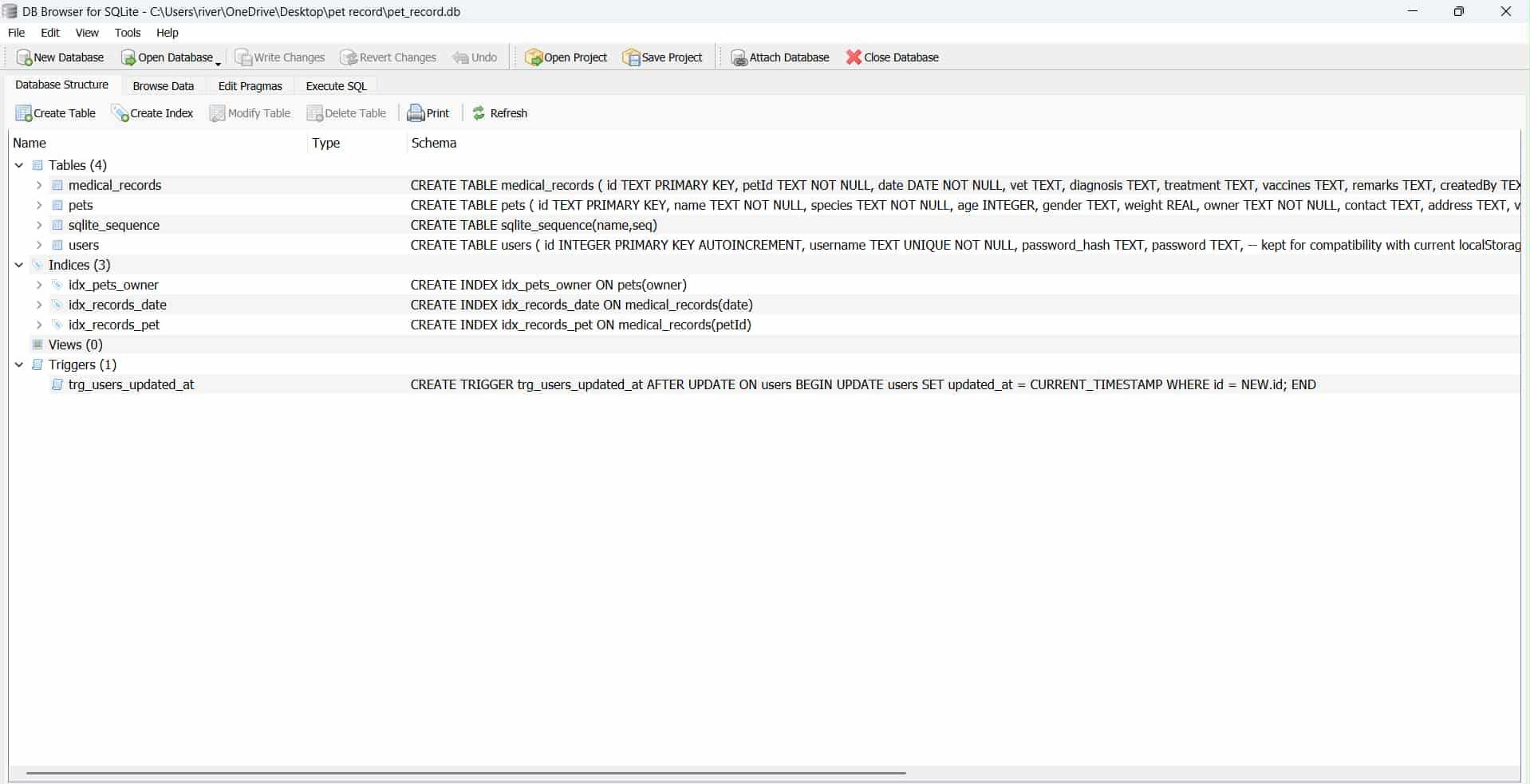
**Communication Protocols and Interface**

* **Frontend ↔ Backend:** Secure HTTP/HTTPS requests through APIs.
* **Backend ↔ Database:** SQL queries over a secure connection for storing, retrieving, and updating records.

**DATABASE DESIGN**

The database design provides the structure for storing and organizing system data. It ensures that user accounts, pet details, and medical records are properly managed, secured, and easily accessible to support the system’s functions.

**Entity-Relationship Diagram (ERD)**



*Figure 1. SQL Database*

**Description of Database Tables, Fields, and Relationships**

*Users Table*

* id (PK, INTEGER, AUTOINCREMENT) – Unique identifier for each user
* username (TEXT, UNIQUE, NOT NULL) – Login name
* password\_hash (TEXT) – Encrypted password
* password (TEXT) – Plain password (legacy support)
* updated\_at (TIMESTAMP) – Auto-updated using trigger

**Relationship:** Independent table for system authentication.

*Pets Table*

* id (PK, TEXT) – Unique identifier for each pet
* name (TEXT, NOT NULL) – Pet’s name
* species (TEXT, NOT NULL) – Pet species (dog, cat, etc.)
* age (INTEGER) – Pet’s age
* gender (TEXT) – Pet’s gender
* weight (REAL) – Pet’s weight in kg
* owner (TEXT, NOT NULL) – Pet owner’s name
* contact (TEXT) – Owner’s contact number
* address (TEXT) – Owner’s address

**Relationship:** One pet can have multiple medical records.

*Medical\_Records Table*

* id (PK, TEXT) – Unique identifier for each record
* petId (FK, TEXT, NOT NULL) – References Pets(id)
* date (DATE, NOT NULL) – Date of consultation
* vet (TEXT) – Veterinarian in charge
* diagnosis (TEXT) – Medical findings
* treatment (TEXT) – Treatment given
* vaccines (TEXT) – Vaccinations
* remarks (TEXT) – Additional notes
* createdBy (TEXT) – User who recorded the entry

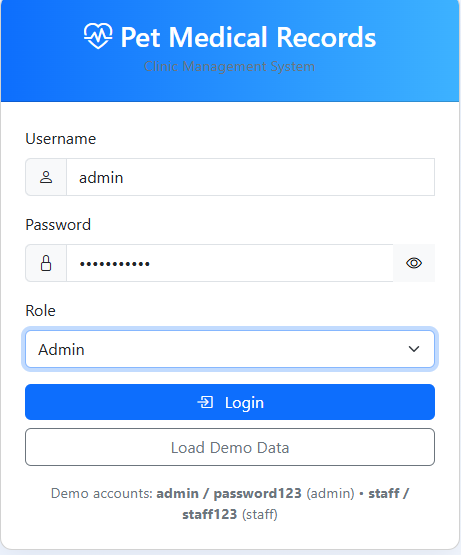
**Relationship:** Many records can be linked to one pet

**Data Normalization Techniques**

* **First Normal Form (1NF):**
* All attributes are atomic (pet details separated into name, species, age, etc.).
* No repeating groups.
* **Second Normal Form (2NF):**
* Non-key attributes fully depend on the primary key.
* Example: In Medical\_Records, details (diagnosis, treatment) depend entirely on id.
* **Third Normal Form (3NF):**
* No transitive dependency.
* Example: Owner’s details are stored only in Pets table, not repeated in Medical\_Records.

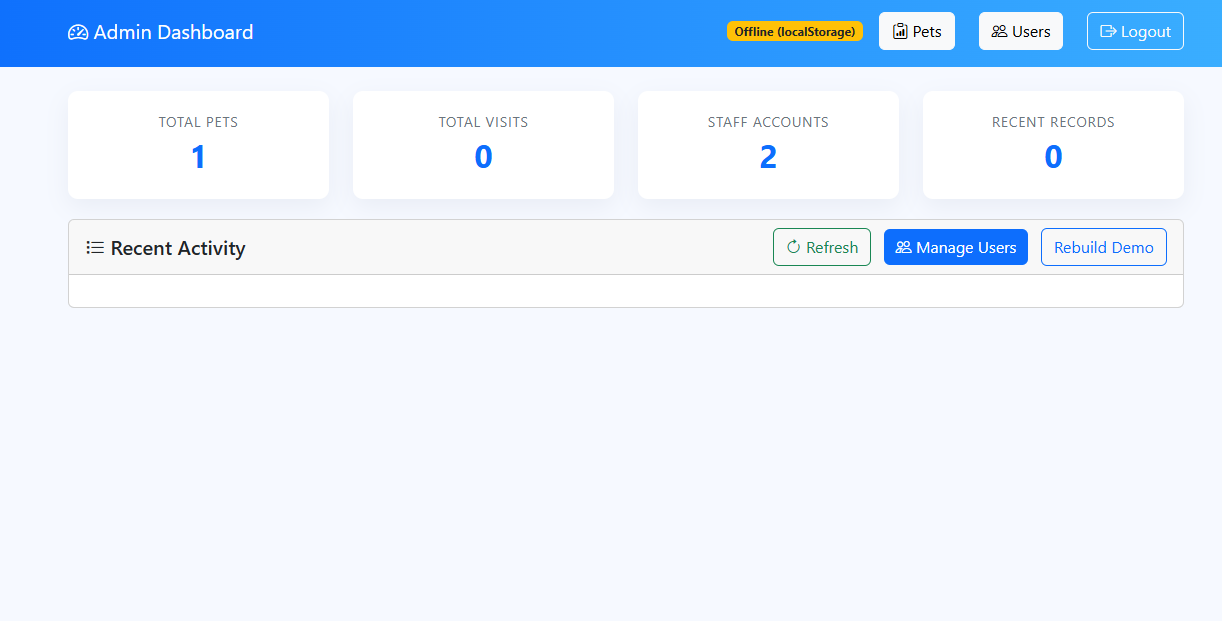
**USER INTERFACE DESIGN**

The system user interface design is created to be simple and easy to use. It allows staff to access the system quickly without confusion. At the same time, the interface provides convenient access to manage staff accounts, monitor activities, and oversee all records in the system. The design is clear and neat, making it convenient for users to perform their tasks.



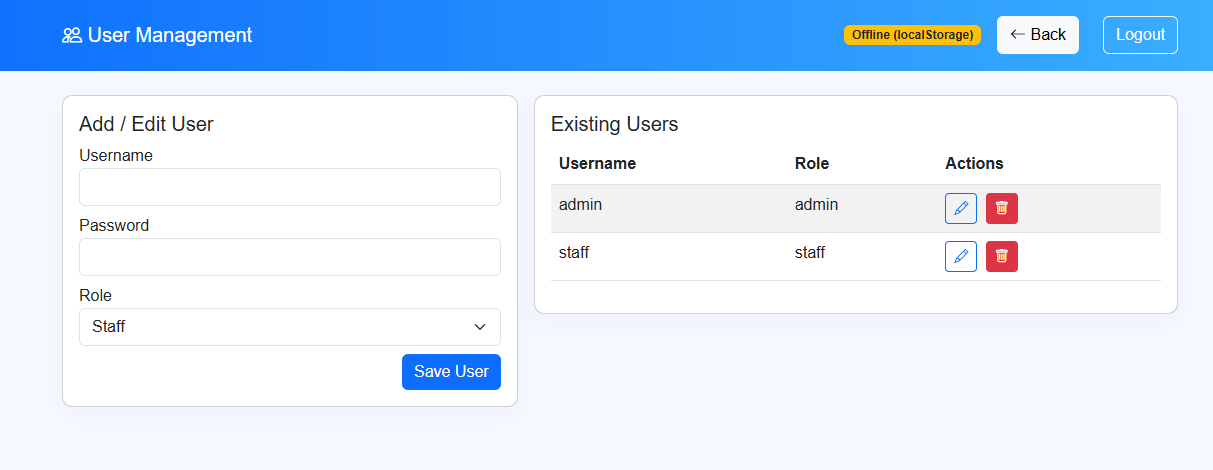
*Figure 2. Login Page*

The **Login page** is the gateway to the Pet Medical Records System. Users enter their username, password, and role (Staff) to access their dashboard. manage staff and system activity, while staff handle pet profiles and medical records.



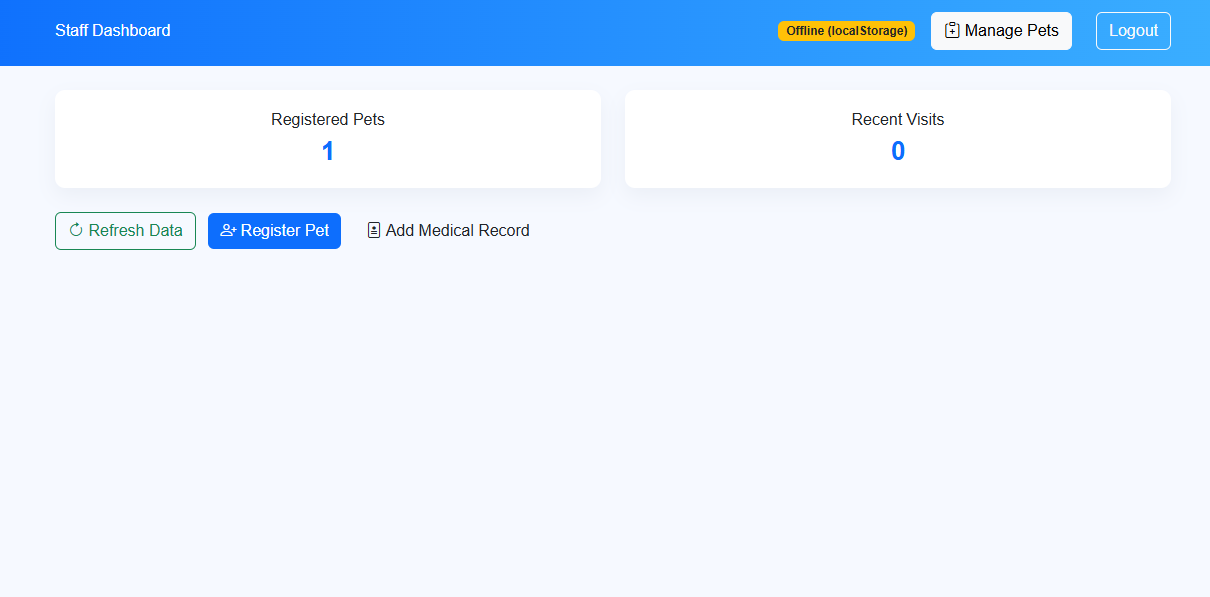
*Figure 3. Admin Dashboard*

The **Admin Dashboard** provides an overview of the system, showing total pets, visits, staff accounts, and recent records. can monitor recent activity and manage user accounts



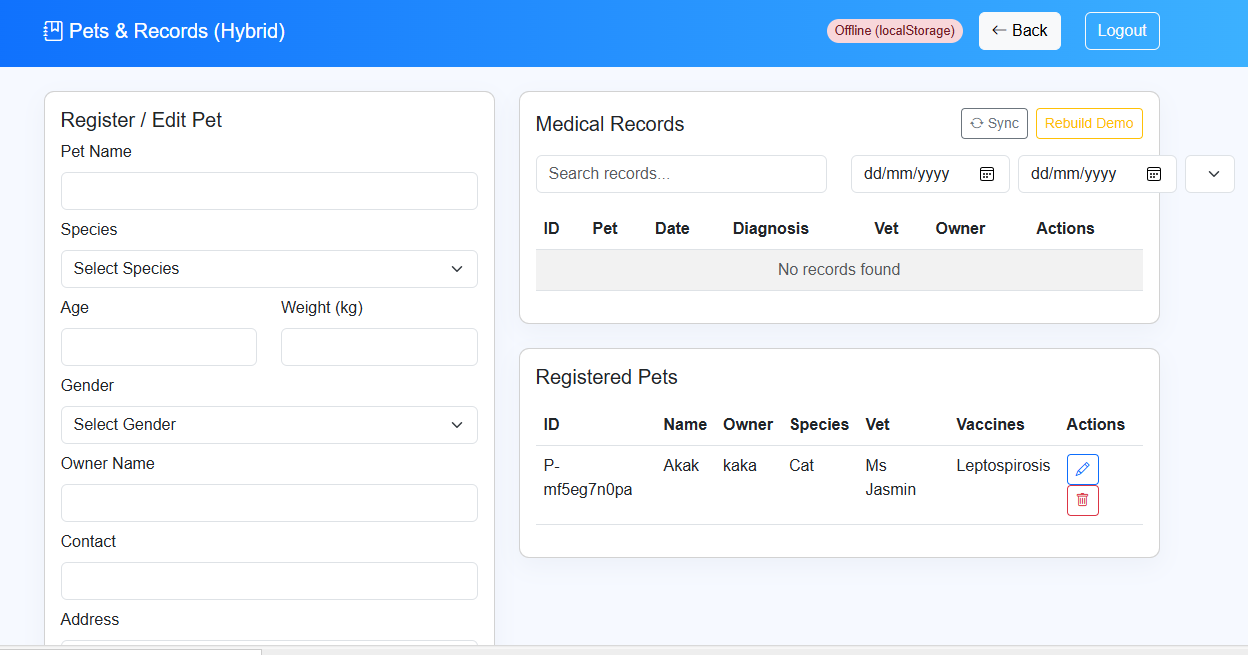
*Figure 4. User Management Page*

**User Management** page, can add new users, edit existing account details, assign roles (such as staff), and delete users when necessary.



*Figure 5. Staff Dashboard*

The **Staff Dashboard** provides staff members with tools to manage pet records and staff can view the number of registered pets and recent clinic visits.



*Figure 6. Pet Records*

The **Pet Records** allows staff to register and manage pet information along with their medical histories. Users can add or edit pet details such as name, species, age, weight, gender, and owner information. The medical records panel enables staff to record diagnoses, treatments, vaccinations, and attending veterinarians.

**COMPONENT DESIGN**

The Pet Medical Records System is organized into several main modules, each focusing on a specific function but working together to create a smooth workflow for veterinary staff.

* Login Module – Provides a secure login form where staff enter their username and password before gaining access to the system.
* Pet Profile Module – Allows staff to add and edit details about pets and their owners, including basic information such as name, species, breed, and age.
* Medical Records Module – Handles the recording and viewing of medical history, including check-ups, treatments, and vaccinations.
* Reporting Module – Generates medical history summaries for selected pets, which can be displayed or printed.

**Interface Specification**

Each module provides its own interface to allow staff to interact with the system. The Login Module presents a form with fields for username and password, verifying the input against stored records before granting access. The Pet Profile Module provides input fields for pet and owner information, along with buttons for saving and updating records, and a search function for retrieving existing profiles. The Medical Records Module contains forms where staff can enter details about the type of medical record, the date, notes, and dosage information when applicable, while also displaying the list of past records for the selected pet. The Reporting Module offers a simple interface where staff can choose a pet and immediately generate a full medical history report, either for on-screen viewing or printing.

**Dependency Management**

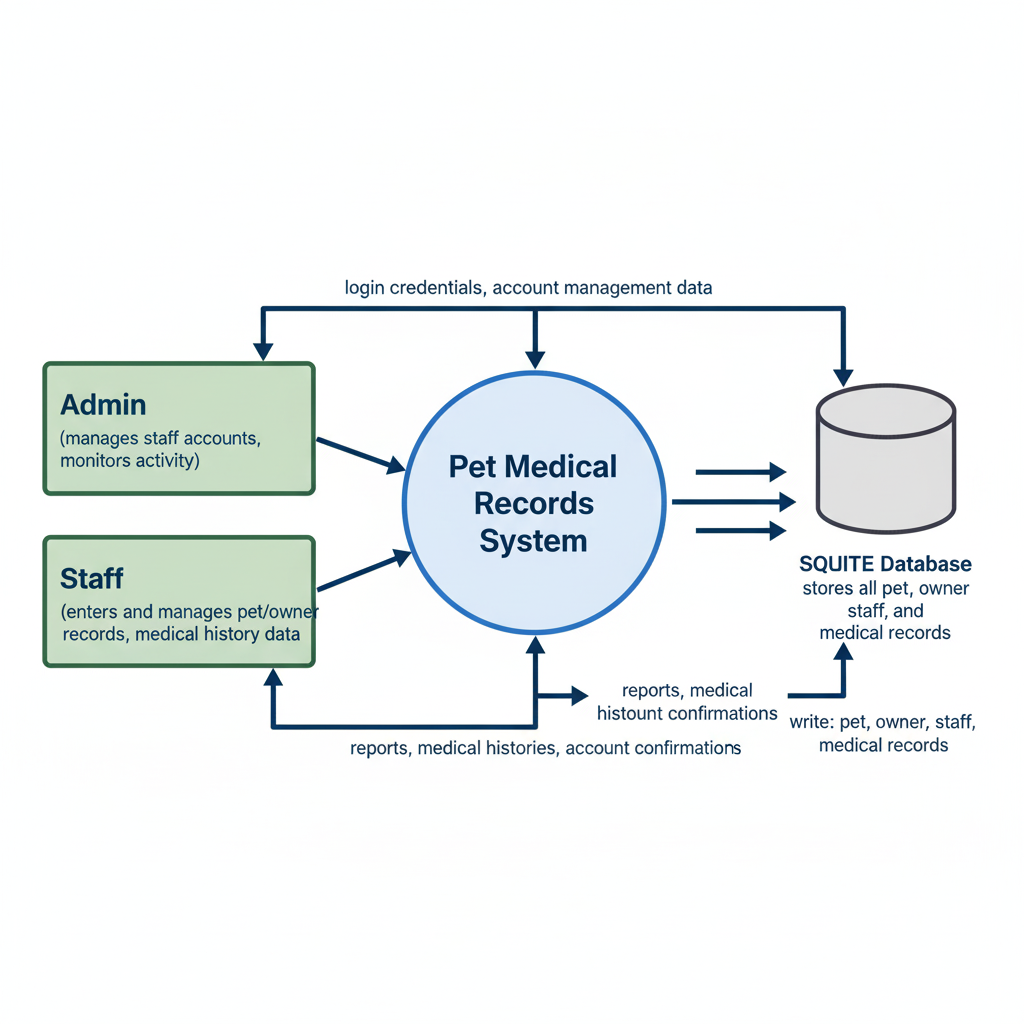
The modules in the system are designed to work together while remaining modular. The Login Module must be completed first before any other module can be accessed. The Pet Profile Module is required before medical records can be added, since each record must be linked to a pet. The Medical Records Module relies on the Pet Profile Module to ensure that all entries are associated with the correct pet, while the Reporting Module depends on the information saved in the Medical Records Module to produce accurate summaries.

**DATA FLOW DIAGRAMS**

These diagrams illustrate the interaction between users, system processes, and the database.

**Context Diagram Level (0)**

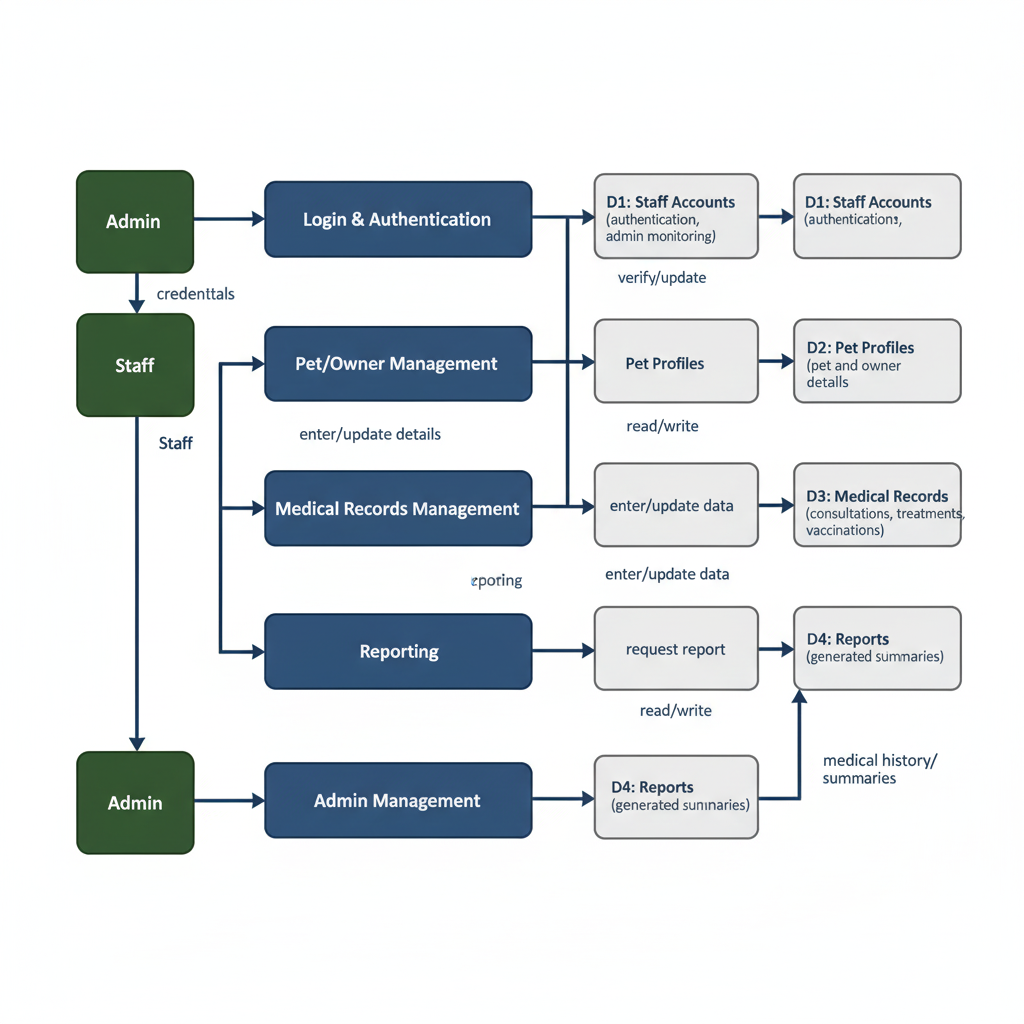
At the highest level, the Pet Medical Records System can be shown as a single process that interacts with external entities.



*Figure 7. Context Diagram Level*

**LEVEL 1 Diagram**

The system process is broken down into smaller modules.



*Figure 8. Main Process Diagram*

**SECURITY DESIGN**

The system prioritizes safeguarding sensitive information, including pet medical histories, owner details, and staff records. Its main consideration is to prevent unauthorized access and ensure that only verified users can interact with the system. Security measures focus on maintaining data confidentiality, integrity, and trust among staff.

**Authentication and Authorization Mechanism**

The system uses role-based authentication, where users must log in with valid credentials before accessing any module. staff are given higher-level access to manage accounts and monitor activity, while staff have restricted access limited to recording and updating pet and medical data.

**Data Encryption and Protection Measures**

Passwords are securely stored to prevent unauthorized use, ensuring the protection of both staff and data.

**PERFORMANCE DESIGN**

This outline designed to maintain speed, stability, and reliability while supporting daily clinic operations. It explains the system’s performance requirements, the strategies used to optimize efficiency, and the testing methods applied to ensure that the system performs well under different conditions.

**Performance Requirements and Objectives**

The system is required to support up to 100 concurrent users without downtime or noticeable delays. Pages must load within a few seconds, and database queries should return results in real time to ensure smooth clinic operations.

**Strategies for Optimizing System Performance**

The system uses database indexing to improve query speed and server-side pagination to handle large datasets efficiently. Caching is applied to frequently accessed data, such as reports and medical histories, to reduce repetitive queries. The frontend is optimized by using clean and lightweight HTML, CSS, and JavaScript code, ensuring fast loading times. For scalability, the architecture allows future upgrades, including migration to more advanced databases or integration of load balancing if needed.

**Testing Plan**

The testing process includes load testing to simulate multiple users accessing the system at the same time, ensuring stability under normal use. Stress testing is applied to check how the system behaves with heavy data volumes, while recovery testing verifies that the system can quickly restore operations after crashes or failures.

**ERROR HANDLING AND LOGGING**

The system includes basic error handling and logging to make sure the application runs smoothly and is easy to use. Errors are handled in a simple way so that staff can quickly understand the problem and fix it without confusion.

**Error Handling Mechanisms and Strategies**

Login Validation: The system checks username, password, and role before giving access. If they do not match, an error message will appear.

**Logging Requirements and Specifications**

Every login attempt, whether successful or failed, is recorded in a local log file.

**Error Codes and Messages**

E001 – Invalid Credentials for Selected Role: Shown when the username or password does not match the role selected r Staff).

**THIRD-PARTY INTEGRATIONS**

The Pet Medical Records System is designed as a standalone desktop application and does not require integration with external services or third-party APIs. All core functions, such as pet profile management, medical records, and reporting are handled internally within the application and its SQLite database.

Although no third-party integrations are included in the current design, the system is structured in a way that allows future enhancements if needed. For example, future versions could connect to external reporting tools, cloud storage services for backups, or veterinary-related APIs for vaccine reference lists. However, these are outside the current project scope and are not part of the initial deployment.

**DEPLOYMENT PLAN**

The deployment involves setting up the application on a local server within the veterinary clinic. The process includes installing required software, configuring the database, and deploying the frontend and backend components so that the system can be accessed through web browsers by staff. Once deployed, user accounts will be created, and initial test data will be loaded to verify proper functionality before full use.

**Hardware and Software Requirements**

To run smoothly, the system requires a standard desktop or laptop with the following minimum specifications:

* Processor: at least Intel i5 processor or equivalent
* Memory: 8 GB RAM
* Storage: 500GB
* Operating System: Windows or Linux-based OS
* Frontend: HTML, CSS, JavaScript
* Backend: Node.js or Python for server logic and APIs
* Database: SQLite for local storage of records

**Configuration Management and Version Control**

Version control will be managed using Git to track changes in the system’s source code and allow collaborative development. Configuration files, including database connection settings and user access roles, will be stored securely and documented for consistency. Backup procedures will be established to ensure recovery in case of system errors, and updates will follow a controlled release cycle where new features or fixes are tested before deployment to the live environment.

**MAINTENANCE AND SUPPORT**

This section outlines the guidelines for maintaining system performance, the procedures for applying updates and fixes, and the escalation process for addressing technical issues.

**Guidelines for System Maintenance and Support**

The system requires regular maintenance to ensure continued reliability and efficiency. Maintenance includes monitoring system performance, checking database integrity, and performing regular backups of medical records. Staff should also follow basic usage guidelines, such as proper shutdown of the system and safeguarding login credentials, to avoid unnecessary errors or data loss.

**Procedures for Handling Software Updates, Patches, and Bug Fixes**

All software updates, including patches and bug fixes, will be released by the development team as needed. Updates will be tested in a controlled environment before deployment to avoid disruptions in clinic operations. Once validated, the updated version of the application will replace the previous installation, while keeping a backup of existing files and databases.

**Escalation Process for Resolving Issues**

In case of technical issues, staff will first attempt to resolve minor problems. If the issue persists, it will be reported to the support team through designated communication channels. Critical issues, such as system crashes or data corruption, will be prioritized and escalated immediately to the development team for urgent resolution. This tiered escalation process ensures that problems are addressed promptly while minimizing disruption to daily clinic operations.

**REVISION HISTORY**

**APPENDIX**