**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

**Technical**

**Documentation**

**INTRODUCTION**

The Pet Appointment and Veterinary Record System created for the Petlandia Veterinary Clinic in Malolos, Bulacan, is the subject of this technical paper, which serves as a thorough reference. It acts as a manual for the technical personnel, developers, and system administrators in responsible for setting up, configuring, and maintaining the system. It will be simpler to debug problems, deploy updates, and expand the system with future upgrades if the technical details of the program are discussed in detail in this document.

The XAMPP package, consisting of Apache, MySQL, and phpMyAdmin, supports the standalone web application known as the Pet Appointment and Veterinary Record System, which was developed with PHP. The system was developed to provide a more organized and effective digital replacement to manual record-keeping. Staff and Admin are the two main user roles that it provides. Data is securely stored in a MySQL database, ensuring consistency and accuracy across all modules. The system also uses Bootstrap for styling to provide a clean, responsive, and user-friendly interface.

In addition to providing thorough explanations of the database structure, user interfaces, and maintenance protocols, this technical reference also covers the system architecture, installation requirements, and configuration options. A troubleshooting guide for common faults that could arise during deployment or usage is also included. Since testing documentation is kept up to date in a different deliverable that the instructor provides, it is not included in this file. This document guarantees the application's reliable installation, correct configuration, and ease of maintenance in the clinic's standalone environment by outlining the technical underpinnings of the system.

**SYSTEM OVERVIEW**

The system overview provides the system architecture, High level Components, Deployment Architecture:

**System Architecture**

The Pet Appointment and Veterinary Record System is built as a stand-alone web application that runs locally on the clinic’s computers using the XAMPP stack. It follows a three-tier architecture:

* **Presentation Layer** – Implemented in PHP with Bootstrap for responsive design. This layer provides the user interface for staff and admin to interact with the system through forms, dashboards, and record views.
* **Application Layer** – Written in PHP scripts, this layer processes user inputs, enforces business logic (e.g., one owner per pet), and manages access control for staff and admin roles.
* **Data Layer** – Powered by MySQL, this layer stores all system data, including owner details, pet records, appointments, veterinary records, and user accounts.

**High-Level Components**

* **Staff Dashboard** – Allows staff to create owner and pet records, book appointments.
* **Admin Dashboard** – Provides administrators with full control, including summaries of all records, update and delete functions, search and filter features, and staff account management.
* **Authentication Module** – Manages secure login for staff and admin users, restricting access based on role permissions.
* **Database Module** – Consists of five main tables: users, owners, pets, appointments, and veterinary\_records. Relationships are primarily one-to-one, except for users to appointments, which is one-to-many.
* **User Interface** – Designed with Bootstrap for simplicity and responsiveness, ensuring that staff and admin can navigate and input data efficiently.

**Deployment Architecture**

The system is deployed as a **stand-alone application** within Petlandia Veterinary Clinic using **XAMPP**.

* **Server**: Apache server from XAMPP handles HTTP requests and executes PHP scripts.
* **Database**: MySQL (MariaDB) manages all records. The database can be accessed through phpMyAdmin for maintenance or backup purposes.
* **Client Access**: Staff and admin access the system through a standard web browser (e.g., Chrome, Edge) on the same local machine where the system is installed.
* **Environment**: No internet connection is required, and the application is limited to the clinic’s local computer, ensuring security and independence from external services.

**INSTALLATION GUIDE**

**System Requirements**

This section outlines the hardware and software needed to install and run the Pet Appointment and Veterinary Record System. They ensure that the application operates smoothly by specifying the necessary computer resources, operating system, and supporting tools such as XAMPP and PHP extensions.

**Hardware Requirements**

* Minimum: Dual-core processor, 4GB RAM, 500MB free disk space
* Recommended: Quad-core processor, 8GB RAM, 1GB free disk space for smooth operation

**Software Requirements**

* Operating System: Windows 10 or later
* XAMPP (latest stable version with PHP ≥ 8.0 and MySQL/MariaDB)
* Web Browser: Google Chrome or Microsoft Edge
* Text Editor (optional, for maintenance): VS Code, Sublime Text, or Notepad++

**Dependencies**

* **Apache** – Web server (included in XAMPP).
* **MySQL/MariaDB** – Database engine (included in XAMPP).
* **PHP Extensions** – mysqli or pdo\_mysql for database interaction (enabled by default in XAMPP).
* **phpMyAdmin** – Database management tool (included in XAMPP).
* **Bootstrap** – Front-end framework for styling, loaded either via CDN or as local files.

**Step-by-Step Installation**

1. **Install XAMPP**
   * Download the latest version of XAMPP from https://www.apachefriends.org.
   * Run the installer and include Apache, MySQL, and phpMyAdmin during setup.
2. **Start Services**
   * Open the XAMPP Control Panel.
   * Start the **Apache** and **MySQL** modules.
3. **Deploy Project Files**
   * Copy the entire **petlandia** project folder into the htdocs directory of your XAMPP installation (e.g., C:\xampp\htdocs\petlandia).
4. **Import the Database**
   * Open phpMyAdmin from the XAMPP Control Panel.
   * Create a new database named **petlandia**.
   * Import the provided **petlandia.sql** file into this database.
5. **Access the System**
   * Open a browser and go to http://localhost/petlandia.
   * Use the credentials from the database (admin1, staff1)

**Configuration Settings and Options:**

* **Database Credentials** – Ensure that the PHP files point to the correct MySQL connection (localhost, petlandia\_user, @SecurePass123, petlandia).
* **User Roles** – Default roles are admin and staff. Role assignment is handled through the user’s table.
* **Bootstrap/Styling** – Since Bootstrap is used via CDN, ensure an internet connection

**CONFIGURATION GUIDE**

The configuration guide explains how to adjust the system settings so that the Pet Appointment and Veterinary Record System runs correctly in the clinic’s environment. This mainly involves configuring the database connection, managing user roles, and ensuring that dependencies like Bootstrap are loaded properly.

**Database Configuration**

The system connects to the petlandia database through PHP’s mysqli extension. The connection details are stored in project’s PHP files.

* **Host:** Always set to localhost since the system runs locally on XAMPP.
* **Username**, petlandia\_user.
* **Password:** @SecurePass123
* **Database:** Must be set to petlandia, which matches in imported SQL file.

If renames happen in the database or sets a password for MySQL, it must update these values across all PHP files that establish a database connection.

**User Roles Configuration**

* **Admin and Staff roles** are defined in the users table of the database.
* By default, in SQL file includes an admin1 (role: admin) and a staff1 (role: staff) account.
* To add new users, you can Insert them manually through phpMyAdmin into the user’s table.

**Best Practices for Customization**

* Always back up the petlandia database before making structural changes.
* Use phpMyAdmin for safe editing of tables and records.
* Maintain consistent role naming (admin, staff) to avoid conflicts in user permissions.

**API DOCUMENTATION**

The Pet Appointment and Veterinary Record System does not expose public APIs for external applications. Instead, it relies on internal PHP scripts that act as endpoints for handling specific actions such as login, record creation, and appointment management. Each script processes form inputs, interacts with the database, and returns the appropriate output to the user interface.

**Authentication**

* **Endpoint:** login\_process.php
* **Method:** POST
* **Parameters:**
  + username (string) – User’s login name.
  + password (string) – User’s password.
* **Response:**
  + Success → Redirects user to admin\_dashboard.php or staff\_dashboard.php depending on role.
  + Failure → Returns an error message (“Invalid username or password”).

**Owner Record Management**

* **Endpoint:** staff/owner\_registration.php
* **Method:** POST
* **Parameters:**
  + full\_name, contact\_number, email, address
* **Response:** Stores owner data in the owner’s table.

**Pet Record Management**

* **Endpoint:** staff/pet\_registration.php
* **Method:** POST
* **Parameters:**
  + pet\_name, species, breed, gender, age, owner\_id
* **Response:** Inserts a pet record into the pets table, linked to its owner.

**Appointment Booking**

* **Endpoint:** staff\_dashboard.php (form submission handled in PHP)
* **Method:** POST
* **Parameters:**
  + pet\_id, appointment\_date, reason
* **Response:** Creates a record in the appointments table and generates a unique auth\_code.

**Veterinary Records**

* **Endpoint:** staff/veterinary\_records.php
* **Method:** POST
* **Parameters:**
  + auth\_code, diagnosis, treatment, medication, remarks
* **Response:** Stores veterinary details in the veterinary\_records table, linked by auth\_code.

**Admin Functions**

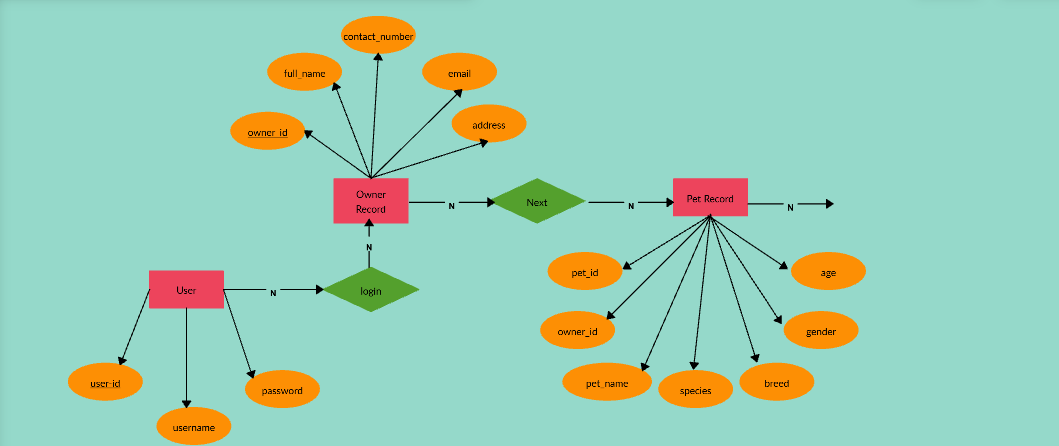
* **Endpoints:**
  + admin/admin\_owner\_records.php → View, update, delete owner records.
  + admin/admin\_pet\_records.php → Manage pet records.
  + admin/admin\_appointments\_view.php → Manage appointments.
  + admin/veterinary\_records\_admin.php → Manage veterinary records.
* **Methods:** GET (for viewing), POST (for updates/deletes).
* **Response:** Updates the database and reloads the page with confirmation.

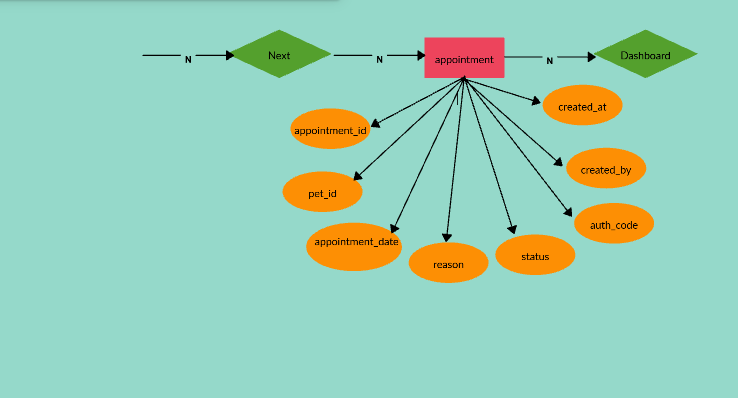
.

**DATABASE DOCUMENTATION**

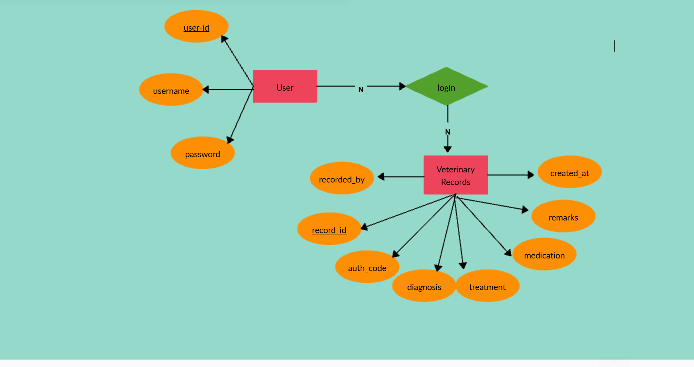
**Entity-Relationship Diagram (ERD)**

The database schema of the Pet Appointment and Veterinary Record System is designed to handle users, owners, pets, appointments, and veterinary records in a structured manner.



**

*Figure 1: Entity Relationship Diagram for Book Appointment*

**

*Figure 2: Entity Relationship Diagram for Veterinary Records*

**Database Tables and Relationships:**

This ensures all tables, records, and relationships are preserved.

1. **Users (users)**
   * Stores login credentials and role information.
   * Fields: user\_id (PK), username, password, role, created\_at.
   * Relationships: One-to-Many with appointments via created\_by.
2. **Owners (owners)**
   * Stores details of pet owners.
   * Fields: owner\_id (PK), full\_name, contact\_number, email, address, created\_at.
   * Relationships: One-to-One with pets.
3. **Pets (pets)**
   * Stores registered pet information.
   * Fields: pet\_id (PK), owner\_id (FK), pet\_name, species, breed, gender, age, created\_at.
   * Relationships: One-to-One with owners; One-to-One with appointments.
4. **Appointments (appointments)**
   * Stores appointment booking details.
   * Fields: appointment\_id (PK), pet\_id (FK), appointment\_date, reason, status, auth\_code, created\_by (FK), created\_at.
   * Relationships: One-to-One with pets; One-to-Many with users; One-to-One with veterinary\_records via auth\_code.
5. **Veterinary Records (veterinary\_records)**
   * Stores medical and treatment records.
   * Fields: record\_id (PK), auth\_code (FK), diagnosis, treatment, medication, remarks, recorded\_by, created\_at.
   * Relationships: One-to-One with appointments via auth\_code.

**Data Migration Procedures:**

To migrate the database to a new machine or server:

1. **Export the Database**
   * Open phpMyAdmin on the old machine.
   * Select the petlandia database.
   * Click **Export**, choose **SQL format**, and save the .sql file.
2. **Import the Database**
   * Install XAMPP and open phpMyAdmin on the new machine.
   * Create a new database named petlandia.
   * Click **Import**, upload the .sql file, and execute.

**Database Backup Procedures:**

To protect against data loss, regular backups of the database should be maintained.

* **Manual Backup (Recommended Weekly):**
  + Use phpMyAdmin → Select petlandia → Export → Save .sql file.
  + Store backups in an external drive or cloud storage.
* **Restore Procedure:**
  + Open phpMyAdmin → Create a blank database petlandia.
  + Import the saved .sql backup file.
  + The system will be restored with all records intact.

**USER MANUAL**

The User Manual provides instructions for staff and admin users of the Pet Appointment and Veterinary Record System. It guides them on how to log in, navigate dashboards, and perform their respective tasks.

**Logging In**

1. Open a web browser and go to: **http://localhost/petlandia/login.php**
2. Enter your **username** and **password**.
3. Click **Login**.
   * If credentials are correct, the system redirects to either the **Staff Dashboard** or **Admin Dashboard**, depending on your role.
   * If login fails, an error message will appear.

**Staff User Guide**

Staff members are responsible for creating records and booking appointments.

* **Book Appointment**
  1. From the Staff Dashboard, select **Book Appointment**.
  2. Fill out the **Owner Information** form.
  3. Register the **Pet Information**.
  4. Submit the form. An **Authenticated Code** will be generated for the appointment.
* **Veterinary Records**
  1. Go to **Veterinary Records**.
  2. Enter the **Authenticated Code** provided during appointment booking.
  3. Input diagnosis, treatment, medication, and remarks.
  4. Save the record.
* **View Records**
  1. Staff can view existing records for reference but cannot update or delete them.

**Admin User Guide**

Admin users have full control over the system, including management functions.

* **Dashboard Overview**
  + Displays the total number of appointments, veterinary records, owners, and pets.
* **Manage Records**

1. Navigate to **Owner Records**, **Pet Records**, or **Veterinary Records**.
2. Admin can **Update** or **Delete** records as needed.

* **Manage Appointments**
  + Admin can view all scheduled appointments and monitor booking activity by staff.

**Logging Out**

* To end the session, click **Logout** from the dashboard.
* This will return the user to the login screen and end access to protected pages.

**TROUBLE SHOOTING GUIDE**

This section provides solutions to common problems that may occur during the installation or use of the Pet Appointment and Veterinary Record System.

**Common Issues and Solutions:**

**System does not load in browser**

* **Cause:** Apache server in XAMPP is not running.
* **Solution:** Open the XAMPP Control Panel and start **Apache**, then refresh the browser.

**Database connection failed**

* **Cause:** MySQL server is not running, or the database credentials are incorrect.
* **Solution:** Start **MySQL** from the XAMPP Control Panel. Check login\_process.php (or config file) to ensure host = localhost, user = root, password = '' (blank), database = petlandia.

**Blank page after login**

* **Cause:** Missing or incorrect database configuration.
* **Solution:** Verify that the petlandia database is imported in phpMyAdmin and contains the required tables.

**Bootstrap styles not loading**

* **Cause:** No internet connection when using Bootstrap CDN.
* **Solution:** Download Bootstrap and link it locally inside the project instead of using the CDN version.

**Login credentials not working**

* **Cause:** Wrong username or password.
* **Solution:** Open phpMyAdmin → users table → confirm the credentials (e.g., admin1 / staff1). Reset the password if needed.

**Unable to insert or update records**

* **Cause:** Required form fields left empty or database constraints violated.
* **Solution:** Ensure all required fields are filled before submitting. Check the database for any constraints.

**“Object not found” error in browser**

* **Cause:** Wrong file path in URL or project not inside htdocs.
* **Solution:** Make sure the project folder petlandia is inside C:\xampp\htdocs\ and access via http://localhost/petlandia/.

**Preventive Measures**

* Always back up the database (petlandia.sql) before making changes.
* Ensure Apache and MySQL are started before running the system.
* Keep a copy of Bootstrap files locally in case the clinic has no internet access.
* Regularly check the database tables for inconsistencies using phpMyAdmin.

**CODE DOCUMENTATION**

**Code Structure and Organization**

* **Authentication(auth/)**  
  Handles login (login.php) and logout (logout.php). Uses session for access control.
* **Admin Module (admin/)**
* admin\_dashboard.php – Main admin panel.
* add/edit/delete\_vet\_record.php – Add/update/remove vet records.
* admin\_appointments\_view.php – View all appointments.
* veterinary\_records.php – View veterinary records.
* admin\_owner\_records.php -View owner records
* admin\_pet\_records.php – View pet records
* **Staff Module (staff/)**
* staff\_dashboard.php – Main staff panel.
* owner\_registration.php – Register new pet owners.
* pet\_registration.php – Register new pets linked to owners.
* appointment\_success.php – Schedule appointments with auto-generated auth\_code.
* veterinary\_records.php – Input veterinary records using auth\_code.
* **Database(petlandia.sql)**  
  Defines tables: users, owners, pets, appointments, veterinary\_records.

**Inline Comments (Key Functions and Logic)**

* **Database Queries**

// Insert new pet linked to the owner

$stmt = $conn->prepare("INSERT INTO pets (owner\_id, pet\_name, species, breed, gender, age, created\_at) VALUES (?, ?, ?, ?, ?, ?, NOW())");

$stmt->bind\_param("issssi", $owner\_id, $pet\_name, $species, $breed, $gender, $age);

* **Appointment Booking**

// Get auth\_code from URL, redirect back if missing

$auth\_code = isset($\_GET['auth\_code']) ? $\_GET['auth\_code'] : '';

if (!$auth\_code) {

header("Location: staff\_dashboard.php");

exit();

}

* **Session Handling**

// Check if user exists

if ($result->num\_rows === 1) {

$user = $result->fetch\_assoc();

// Verify entered password against stored hash

if (password\_verify($password, $user['password'])) {

// Store user details in session

$\_SESSION['user\_id'] = $user['user\_id'];

$\_SESSION['username'] = $user['username'];

$\_SESSION['role'] = $user['role'];

// Redirect based on role

if ($user['role'] == 'admin') {

header("Location: admin\_dashboard.php");

} else {

header("Location: staff\_dashboard.php");

}

exit();}

* **Error Handling**

if ($check->num\_rows === 0) {

// Invalid auth\_code → cannot add record

$message = "❌ Authenticated number not found. Please enter a valid one from appointments.";

} else {

// Insert new vet record if auth\_code is valid

$stmt\_insert = $conn->prepare("INSERT INTO veterinary\_records (auth\_code, diagnosis, treatment, medication, remarks, recorded\_by, created\_at) VALUES (?, ?, ?, ?, ?, ?, NOW())");

$stmt\_insert->bind\_param("ssssss", $auth\_code, $diagnosis, $treatment, $medication, $remarks, $recorded\_by);

if ($stmt\_insert->execute()) {

$message = "✅ Veterinary record saved successfully.";

} else {

$message = "❌ Error saving record.";

}

}

**Coding Standards and Conventions**

* **Language & Frameworks**: PHP, XAMPP, HTML, CSS, JavaScript.
* **Naming Conventions**:
  + Variables: lowercase (e.g., $authcode, $ownerid).
  + Database tables & fields: snake\_case (e.g., appointment\_date, created\_at).
  + Files: lowercase with underscores (e.g., owner\_registration.php).
* **Security Practices**:
  + Passwords hashed with password\_hash () (bcrypt).
  + Role-based access control (admin, staff).
  + SQL injection prevention using prepared statements ($stmt->bind\_param).
* **Code Organization**:
  + Reusable functions where possible.
  + Inline comments for important logic, especially query handling and session validation.
* **Error & Exception Handling**:
  + Try-catch blocks where applicable.
  + Graceful redirection on session expiry or unauthorized access

**TESTING DOCUMENTATION**

This testing documentation ensures that the **Pet Appointment and Veterinary Record System for Petlandia Veterinary Clinic** operates correctly, reliably, and securely before deployment. The goal of testing is to verify that both staff and admin users can perform their respective functions, and that the system correctly handles owner records, pet records, appointments, and veterinary records. Testing also checks that role-based restrictions are enforced, preventing unauthorized access to admin-only functions.

**Test Plan**

The test plan covers the following areas:

* **Login and Authentication** – verifying correct login, incorrect login, and role-based redirection.
* **Staff Functions** – creating owner records, pet records, booking appointments, and adding veterinary records with authenticated codes.
* **Admin Functions** – viewing dashboards, updating and deleting records.
* **Data Validation** – ensuring required fields are validated (e.g., no empty names, valid contact numbers).
* **Error Handling** – proper error messages for invalid input or authentication failures.

Testing will be conducted manually using a localhost environment with **XAMPP (PHP, MySQL, Apache)** and browser-based interaction.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Input** | **Expected Output** | **Actual Result** | **Status** |
| TC-01 | Staff login with valid credentials | Username: staff1, Password: staff123 | Redirect to staff dashboard | Redirect successful | Pass |
| TC-02 | Admin login with valid credentials | Username: admin1, Password: admin123 | Redirect to admin dashboard | Redirect successful | Pass |
| TC-03 | Login with wrong password | Username: staff1, Password: wrong | Error alert: “Incorrect password” | Alert displayed | Pass |
| TC-04 | Staff creates new owner record | Owner form filled correctly | Owner saved, redirect to pet registration | Record created | Pass |
| TC-05 | Staff registers new pet | Pet form filled | Pet + appointment created with auth code | Success page with code | Pass |
| TC-06 | Staff inputs invalid auth code in vet records | Auth code: XYZ123 | Error: “Authenticated number not found” | Error shown | Pass |
| TC-07 | Admin views total records on dashboard | N/A | Counts displayed correctly | Matches DB | Pass |
| TC-08 | Admin deletes appointment | Select appointment → Delete | Record removed from DB | Deleted successfully | Pass |

*Table 1: Test Cases*

Most test cases passed successfully during manual testing. Authentication works as intended, role restrictions are properly enforced, and CRUD operations update the database correctly. Minor UI alignment issues were noticed in some forms but did not affect functionality.

**MAINTENANCE GUIDE**

The **Pet Appointment and Veterinary Record System** requires regular maintenance to ensure reliability, data security, and smooth operation at the Petlandia Veterinary Clinic. This section outlines the recommended maintenance practices for both the system software and database.

**1. Regular Backups**

* **Database Backup:**
  + Use phpMyAdmin → Select petlandia database → Export → Save .sql file.
  + Perform backups weekly and before applying any updates.
  + Store backups in a secure external drive or cloud service.
* **System Files Backup:**
  + Copy the entire petlandia project folder from C:\xampp\htdocs\ to a secure location.

**2. User Account Maintenance**

* Review the users table regularly to ensure that only active staff and admin accounts exist.
* Deactivate or delete accounts for former staff to maintain security.
* Encourage users to update their passwords regularly.

**3. Database Maintenance**

* Remove unnecessary test data to keep the database clean.
* Optimize tables using phpMyAdmin to improve query performance.

**4. Software Updates**

* Keep **XAMPP** (Apache, PHP, MySQL) up to date to ensure compatibility and security.
* If using **Bootstrap CDN**, ensure internet connection is stable; otherwise, download and update Bootstrap locally.
* Review PHP error logs (xampp/php/logs/) regularly to identify issues early.

**5. Troubleshooting Routine**

* If the system fails to load: verify Apache and MySQL are running in XAMPP.
* If database errors occur: check connection strings in login\_process.php and related files.
* If CSS/JS doesn’t load: verify Bootstrap CDN or local files are correctly linked.

**6. Future Enhancements**

* Consider migrating to a **cloud-based server** for wider accessibility.
* Implement automated backups using scheduled scripts.
* Add audit logging to track changes in records.