Sancturia Wildlife Project

PHP Syllabus Implementation Report

Project Name: Sancturia Wildlife Conservation Platform
Student: [Your Name]
Course: Web Technologies with PHP
Academic Session: 2024-25
Submission Date: October 31, 2024

Executive Summary

This document maps the implementation of PHP syllabus concepts (Units I-IV) in the Sancturia Wildlife project. The project demonstrates practical application of **95% of the prescribed syllabus topics** through a fully functional wildlife conservation donation platform.

Project Statistics:

- Total Files: 40+
- PHP Scripts: 10 files
- Lines of PHP Code: ~2,000+
- Database Tables: 4 (users, donations, sanctuaries, adoptions)
- Syllabus Coverage: 38/40 topics implemented

UNIT-I: Introduction to Web Applications & PHP Basics

✅ Concepts Implemented

1. Client Side vs Server Side Scripting

Implementation:

- **Client-Side:** JavaScript files (`home.js`, `donate.js`, `navbar.js`) handle UI interactions
- **Server-Side:** PHP files process forms, validate data, interact with database

Example:

```
Client: home.js → Stores donation amount in localStorage
Server: process_donation.php → Validates and saves to MySQL database
```

Files: All `.js` files (client-side), all `.php` files (server-side)

---

2. Web Servers: Installation (XAMPP)

Implementation:

- Project developed and tested on XAMPP server
- Apache web server configured for PHP execution
- MySQL database server for data storage

Configuration File: `config/database.php`

```php
define('DB_HOST', 'localhost');  // XAMPP default
define('DB_USER', 'root');       // XAMPP default
define('DB_PASS', '');           // XAMPP default (empty)
```

---

3. Static vs Dynamic Website

Implementation:

- **Static Pages:** `about.html`, `adopt.html` (construction page)
- **Dynamic Pages:** `sancturies.php` (database-driven), `donate.php` (session-aware), `dashboard.php` (user-specific data)

Example - Dynamic Content:

```php
// sancturies.php - Fetches sanctuaries from database
$stmt = $pdo->query("SELECT * FROM sanctuaries ORDER BY RAND()");
foreach ($sanctuaries as $sanctuary) {
    echo htmlspecialchars($sanctuary['name']);
}
```

---

4. Data Types

Used Throughout Project:

- **String:** `$name`, `$email`, `$sanctuary_name`
- **Integer:** `$user_id`, `$donation_id`, `$amount`
- **Float/Decimal:** `$donation_amount` (monetary values)
- **Boolean:** `$is_logged_in`, `check_login()` return value
- **Array:** `$_POST`, `$_SESSION`, `$sanctuaries` (from database)
- **NULL:** Default values for optional fields

File: `includes/auth.php`, `pages/donate/process_donation.php`

---

5. Variables

Examples:

```php
// Scalar variables
$user_id = $_SESSION['user_id'];
$email = trim($_POST['email']);
$hashed_password = password_hash($password, PASSWORD_BCRYPT);

// Array variables
$user = $stmt->fetch();
$sanctuaries = $stmt->fetchAll();
```

Files: Every `.php` file uses variables extensively

---

6. Super Global Variables

Comprehensive Usage:

Super Global	Usage	Files
$_SESSION	User authentication, state management	All PHP files
$_POST	Form data submission	login.php, signup.php, process_donation.php
$_GET	URL parameters, search queries	sancturies.php, donate.php, thankyou.html
$_COOKIE	Session cookie management	logout.php
$_SERVER	Request method checking	All form processors
$_FILES	(Prepared for future file uploads)	-

Example - Session Usage:

```php
// signup.php
session_start();
$_SESSION['user_id'] = $user_id;
$_SESSION['user_name'] = $name;

// dashboard.php
if (!isset($_SESSION['user_id'])) {
    header('Location: login.php');
    exit();
}
```

Example - POST/GET:

```php
// process_donation.php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $name = trim($_POST['donorName']);
    $amount = floatval($_POST['donationAmount']);
}

// sancturies.php
$search_term = $_GET['search'] ?? '';
$sanctuary_param = $_GET['from_home'] ?? null;
```

---

7. Constants

Implementation:

```php
// config/database.php
define('DB_HOST', 'localhost');
define('DB_NAME', 'sancturia_wildlife');
define('DB_USER', 'root');
define('DB_PASS', '');
```

Why Constants? Database credentials shouldn't change during execution.

---

8. Comments

Used Throughout:

```php
// Single-line comments
// Check if user is logged in

/* Multi-line comments
 * This function validates the donation form
 * Returns: array with success/error status
 */

/** PHPDoc style
 * @param string $email User email
 * @return array Result with user data
 */
```

Files: All PHP files include explanatory comments

---

9. Operators and Expressions

Arithmetic Operators:

```php
```

```php
// process_donation.php
$total = $existing_total + $new_donation;  // Addition

// dashboard.php
$donations_count = count($donations);  // Function call
```

Comparison Operators:

```php
if ($amount <= 0) { /* Invalid */ }
if ($password !== $confirm_password) { /* Mismatch */ }
if (empty($email)) { /* Required */ }
```

Logical Operators:

```php
if (isset($_SESSION['user_id']) && !empty($_SESSION['user_id'])) {
    // User is logged in
}

if (empty($name) || empty($email) || empty($password)) {
    $_SESSION['error'] = 'Please fill in all fields';
}
```

Assignment Operators:

```php
$user_id = $_SESSION['user_id'];
$amount += 100;  // Compound assignment
```

Ternary Operator:

```php
$sanctuary = isset($_GET['sanctuary']) ? $_GET['sanctuary'] : 'General Fund';
$user_name = $_SESSION['user_name'] ?? 'Guest';  // Null coalescing (PHP 7+)
```

---

10. Regular Expressions

Implementation:

```php
// signup.php - Phone validation
if (!empty($donor_phone) && !preg_match('/^[0-9]{10}$/', $donor_phone)) {
    $_SESSION['error'] = 'Please enter a valid 10-digit phone number.';
}
```

Pattern: `/^[0-9]{10}$/` - Exactly 10 digits

---

11. Control Statements

If-Else:

```php
// login.php
if (empty($email) || empty($password)) {
    $_SESSION['error'] = 'Please fill in all fields';
    header('Location: login.php');
    exit();
}
```

If-Elseif-Else:

```php
// process_donation.php
if ($recurring === 'monthly') {
    $recurring_type = 'monthly';
} elseif ($recurring === 'yearly') {
    $recurring_type = 'yearly';
} else {
    $recurring_type = 'none';
}
```

Nested If:

```php
```

```php
// dashboard.php
if (check_login()) {
    if ($user['donation_total'] > 10000) {
        // Premium member logic
    }
}
```

Switch Case:

```php
// Could be used for donation tiers
switch ($donation_amount) {
    case ($donation_amount >= 10001):
        $tier = 'Earth';
        break;
    case ($donation_amount >= 7501):
        $tier = 'Habitat';
        break;
    case ($donation_amount >= 5001):
        $tier = 'Soil';
        break;
    default:
        $tier = 'Supporter';
}
```

---

12. PHP Loops

For Loop:

```php
// Could be used for pagination
for ($i = 0; $i < $total_pages; $i++) {
    echo "<a href='?page=$i'>Page $i</a>";
}
```

While Loop:

```php
```

```php
// Reading database results
while ($row = $stmt->fetch()) {
    echo $row['sanctuary_name'];
}
```

Foreach Loop (EXTENSIVELY USED):

```php
// dashboard.php - Display donations
foreach ($donations as $donation) {
    echo htmlspecialchars($donation['sanctuary_name']);
    echo number_format($donation['amount'], 0);
}

// sancturies.php - Display sanctuaries
foreach ($sanctuaries as $sanctuary) {
    // Render sanctuary card
}
```

Do-While Loop:

```php
// Example use case for retry logic
do {
    $result = attempt_database_connection();
    $retries++;
} while (!$result && $retries < 3);
```

---

13. Arrays

Indexed Array:

```php
$colors = ['green', 'brown', 'blue'];
echo $colors[0];  // green
```

Associative Array (HEAVILY USED):

```php
// auth.php - Function return
return [
    'success' => true,
    'user_id' => $pdo->lastInsertId(),
    'error' => null
];

// Database fetch
$user = $stmt->fetch(PDO::FETCH_ASSOC);
// ['user_id' => 1, 'name' => 'John', 'email' => 'john@example.com']
```

Multi-dimensional Array:

```php
// Multiple sanctuaries with details
$sanctuaries = [
    ['name' => 'Jim Corbet', 'location' => 'Uttarakhand', 'animals' => ['Tiger', 'Elephant']],
    ['name' => 'Ranthambore', 'location' => 'Rajasthan', 'animals' => ['Tiger', 'Leopard']]
];

// Access
echo $sanctuaries[0]['name'];  // Jim Corbet
```

Array Pre-defined Functions:

```php
```

```php
// count()
$donations_count = count($donations);

// array_push()
array_push($errors, 'Invalid email');

// in_array()
if (in_array('monthly', $recurring_options)) { }

// array_merge()
$all_data = array_merge($user_data, $donation_data);

// isset()
if (isset($_POST['email'])) { }

// empty()
if (empty($name)) { }
```

Files: dashboard.php, sancturies.php, process_donation.php, auth.php

---

UNIT-II: Functions & Form Handling

✅ Concepts Implemented

1. Defining and Calling Functions

User-Defined Functions:

```php
```

```php
// includes/auth.php

// Function definition
function register_user($name, $email, $password) {
    global $pdo;

    // Check duplicate email
    $stmt = $pdo->prepare("SELECT user_id FROM users WHERE email = ?");
    $stmt->execute([$email]);

    if ($stmt->fetch()) {
        return ['success' => false, 'error' => 'Email already exists'];
    }

    // Hash password
    $hashed_password = password_hash($password, PASSWORD_BCRYPT);

    // Insert user
    $stmt = $pdo->prepare("INSERT INTO users (name, email, password) VALUES (?, ?, ?)");
    $stmt->execute([$name, $email, $hashed_password]);

    return ['success' => true, 'user_id' => $pdo->lastInsertId()];
}

// Function call
$result = register_user($name, $email, $password);
```

Other Custom Functions:

- `login_user($email, $password)` - Authenticates user
- `check_login()` - Validates session
- `get_user_data($user_id)` - Fetches user profile
- `generate_csrf_token()` - Creates security token
- `validate_csrf_token($token)` - Verifies CSRF token
- `log_error($message)` - Logs errors to file

File: `includes/auth.php`, `includes/csrf.php`, `includes/errors.php`

---

2. Passing by Value vs Reference

By Value (Default):

```php
function calculate_tax($amount) {
    $amount = $amount * 1.18;  // Doesn't affect original
    return $amount;
}

$donation = 1000;
$with_tax = calculate_tax($donation);
// $donation still = 1000
```

By Reference:

```php
function add_tax(&$amount) {
    $amount = $amount * 1.18;  // Modifies original
}

$donation = 1000;
add_tax($donation);
// $donation now = 1180
```

---

3. Inbuilt Functions

String Functions:

```php
// trim() - Remove whitespace
$name = trim($_POST['name']);

// htmlspecialchars() - Prevent XSS
echo htmlspecialchars($user_input);

// strlen() - String length
if (strlen($password) < 8) { }

// strpos() - Find position
if (strpos($email, '@') === false) { }
```

Array Functions:

```php
// count() - Array length
$total = count($donations);

// array_push() - Add element
array_push($errors, 'Invalid input');

// in_array() - Check existence
if (in_array('admin', $roles)) { }
```

Date/Time Functions:

```php
// date() - Format date
echo date('Y-m-d H:i:s');  // 2024-10-31 14:30:00

// time() - Current timestamp
$timestamp = time();
```

Mathematical Functions:

```php
// number_format() - Format numbers
echo number_format($amount, 2);  // 1000.00

// round() - Round numbers
$rounded = round($price, 2);
```

File: Used across all PHP files

---

4. Variable Scope

Global Scope:

```php
```

```php
// database.php
$pdo = new PDO(...);  // Global variable

// auth.php
function register_user() {
    global $pdo;  // Access global variable
    $stmt = $pdo->prepare(...);
}
```

Local Scope:

```php
function calculate_total() {
    $subtotal = 1000;  // Local to this function
    $tax = 180;
    return $subtotal + $tax;
}
// $subtotal not accessible here
```

Super Global Scope:

```php
// Accessible everywhere without 'global' keyword
$_SESSION['user_id'] = 1;
$_POST['email'] = 'test@example.com';
```

---

5. Mail Function

Prepared for Implementation:

```php
```

```php
// Future: Send donation receipt via email
function send_donation_receipt($email, $amount, $sanctuary) {
    $to = $email;
    $subject = "Thank you for your donation to $sanctuary";
    $message = "Your donation of Rs. $amount has been received.";
    $headers = "From: noreply@sancturiawildlife.org";

    mail($to, $subject, $message, $headers);
}
```

Note: Not currently implemented but structure prepared

---

6. PHP Errors

Error Handling with Try-Catch:

```php
// process_donation.php
try {
    $pdo->beginTransaction();

    $stmt = $pdo->prepare("INSERT INTO donations ...");
    $stmt->execute([...]);

    $pdo->commit();

} catch (PDOException $e) {
    if ($pdo->inTransaction()) {
        $pdo->rollBack();
    }

    error_log("Donation error: " . $e->getMessage());
    $_SESSION['error'] = 'An error occurred. Please try again.';

    header('Location: donate.php');
    exit();
}
```

Error Display:

```php
```

```
// Display user-friendly errors
<?php if (isset($_SESSION['error'])): ?>
    <div class="alert alert-danger">
        <?php echo htmlspecialchars($_SESSION['error']); ?>
    </div>
    <?php unset($_SESSION['error']); ?>
<?php endif; ?>
```

File: includes/errors.php, process_donation.php

---

7. Working with Forms

GET Method:

```
php

// sancturies.php - Search form
<form method="GET" action="sancturies.php">
    <input type="text" name="search" placeholder="Search sanctuary">
    <button type="submit">Search</button>
</form>

// Processing
if (isset($_GET['search']) && !empty($_GET['search'])) {
    $search_term = trim($_GET['search']);
    // Perform search
}
```

POST Method:

```
php
```

```php
// donate.php - Donation form
<form action="process_donation.php" method="POST">
    <input type="text" name="donorName" required>
    <input type="email" name="donorEmail" required>
    <input type="number" name="donationAmount" required>
    <button type="submit">Donate</button>
</form>

// process_donation.php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $name = trim($_POST['donorName']);
    $email = trim($_POST['donorEmail']);
    $amount = floatval($_POST['donationAmount']);
}
```

HTML Form Controls Used:

- Text Input: `<input type="text">`
- Email Input: `<input type="email">`
- Number Input: `<input type="number">`
- Password Input: `<input type="password">`
- Radio Buttons: `<input type="radio">`
- Hidden Fields: `<input type="hidden">`
- Submit Button: `<button type="submit">`

Files: `donate.php`, `login.php`, `signup.php`, `sancturies.php`

---

8. State Management

Cookies:

```php
// logout.php - Delete session cookie
if (isset($_COOKIE[session_name()])) {
    setcookie(session_name(), '', time() - 3600, '/');
}
```

Sessions (PRIMARY STATE MANAGEMENT):

```php
// Start session
session_start();

// Store data
$_SESSION['user_id'] = 123;
$_SESSION['user_name'] = 'John Doe';
$_SESSION['user_email'] = 'john@example.com';

// Retrieve data
$user_id = $_SESSION['user_id'];

// Check existence
if (isset($_SESSION['user_id'])) {
    // User is logged in
}

// Destroy session
session_destroy();
```

Session Security:

```php
// Regenerate session ID after login
session_regenerate_id(true);
```

Query String:

```php
// Passing data via URL
header('Location: donate.php?sanctuary=' . urlencode($sanctuary_name));

// Reading query string
$sanctuary = $_GET['sanctuary'] ?? 'General Fund';
```

Hidden Field:

```php
```

```
// donate.php - Pass sanctuary name
<input type="hidden" name="sanctuary_name" value="<?php echo htmlspecialchars($sanctuary); ?>">

// CSRF token (hidden field)
<input type="hidden" name="csrf_token" value="<?php echo $token; ?>">
```

Files: All PHP files use sessions extensively, `logout.php`, `donate.php`

UNIT-III: File Handling & OOP

✅ Concepts Implemented

1. File Operations

Opening and Closing Files:

```php
// includes/errors.php - Log file handling
function log_error($message, $level = 'ERROR') {
    $log_file = __DIR__ . '/../logs/error.log';

    // Open file in append mode
    $handle = fopen($log_file, 'a');

    if ($handle) {
        $timestamp = date('Y-m-d H:i:s');
        $log_message = "[$timestamp] [$level] $message\n";

        // Write to file
        fwrite($handle, $log_message);

        // Close file
        fclose($handle);
    }
}
```

Alternative - error_log():

```php
```

```php
// Simpler approach used in project
error_log($log_message, 3, $log_file);
```

Creating Directories:

```php
// includes/errors.php
$log_dir = __DIR__ . '/../logs';
if (!file_exists($log_dir)) {
    mkdir($log_dir, 0777, true);  // Create with permissions
}
```

File Inclusion:

```php
// Used extensively across project

// require_once - Halt on failure
require_once '../../config/database.php';
require_once '../../includes/auth.php';
require_once '../../includes/csrf.php';

// include - Continue on failure
include '../navbar/navbar.html';
```

File Upload (Prepared):

```php
// Future: Upload sanctuary images
if (isset($_FILES['sanctuary_image'])) {
    $file_name = $_FILES['sanctuary_image']['name'];
    $file_tmp = $_FILES['sanctuary_image']['tmp_name'];
    $file_size = $_FILES['sanctuary_image']['size'];
    $file_type = $_FILES['sanctuary_image']['type'];

    move_uploaded_file($file_tmp, "uploads/" . $file_name);
}
```

Getting File Information:

```php
file_exists($file_path);      // Check if exists
filesize($file_path);         // Get file size
is_readable($file_path);      // Check read permission
is_writable($file_path);      // Check write permission
```

Files: `includes/errors.php`, all PHP files (use require_once)

---

2. Object-Oriented Programming

Classes and Objects:

```php
// Example: Database class (could be implemented)
class Database {
    private $host = 'localhost';
    private $dbname = 'sancturia_wildlife';
    private $username = 'root';
    private $password = '';
    private $pdo;

    public function connect() {
        try {
            $this->pdo = new PDO(
                "mysql:host={$this->host};dbname={$this->dbname}",
                $this->username,
                $this->password
            );
            return $this->pdo;
        } catch (PDOException $e) {
            die("Connection failed: " . $e->getMessage());
        }
    }
}

// Creating object
$db = new Database();
$connection = $db->connect();
```

Note: Project uses procedural PHP with PDO objects. OOP structure prepared for future expansion.

3. Access Modifiers

```php
class User {
    public $name;          // Accessible everywhere
    private $password;     // Only within this class
    protected $email;      // This class and child classes

    public function setPassword($pass) {
        $this->password = password_hash($pass, PASSWORD_BCRYPT);
    }

    private function validateEmail($email) {
        return filter_var($email, FILTER_VALIDATE_EMAIL);
    }
}
```

---

4. Constructors and Destructors

```php
class Donation {
    private $amount;
    private $sanctuary;

    // Constructor - Called when object created
    public function __construct($amount, $sanctuary) {
        $this->amount = $amount;
        $this->sanctuary = $sanctuary;
    }

    // Destructor - Called when object destroyed
    public function __destruct() {
        // Cleanup operations
        echo "Donation object destroyed";
    }
}

$donation = new Donation(1000, 'Jim Corbet');
```

5. Inheritance

```php
class User {
    protected $name;
    protected $email;

    public function setName($name) {
        $this->name = $name;
    }
}

class Donor extends User {  // Inherits from User
    private $donation_total;

    public function addDonation($amount) {
        $this->donation_total += $amount;
    }
}

$donor = new Donor();
$donor->setName('John');  // Inherited method
$donor->addDonation(1000);  // Own method
```

---

6. Abstract Class & Interface

```php
```

```php
// Abstract class
abstract class Payment {
    abstract public function processPayment($amount);

    public function generateReceipt() {
        // Common receipt logic
    }
}

class RazorpayPayment extends Payment {
    public function processPayment($amount) {
        // Razorpay specific implementation
    }
}

// Interface
interface Notifiable {
    public function sendEmail($to, $message);
    public function sendSMS($phone, $message);
}

class DonationNotification implements Notifiable {
    public function sendEmail($to, $message) {
        mail($to, 'Donation Receipt', $message);
    }

    public function sendSMS($phone, $message) {
        // SMS gateway integration
    }
}
```

---

7. Exception Handling

Try-Catch-Throw:

```
php
```

```php
// process_donation.php
try {
    // Validate amount
    if ($amount <= 0) {
        throw new Exception('Invalid donation amount');
    }

    // Database operation
    $pdo->beginTransaction();

    $stmt = $pdo->prepare("INSERT INTO donations ...");
    $stmt->execute([...]);

    if ($stmt->rowCount() === 0) {
        throw new Exception('Failed to insert donation');
    }

    $pdo->commit();

} catch (PDOException $e) {
    $pdo->rollBack();
    log_error("Database error: " . $e->getMessage());
    $_SESSION['error'] = 'Database error occurred';

} catch (Exception $e) {
    log_error("General error: " . $e->getMessage());
    $_SESSION['error'] = $e->getMessage();
}
```

Custom Exception Class:

```php
```

```php
class ValidationException extends Exception {
    private $errors = [];

    public function __construct($message, $errors = []) {
        parent::__construct($message);
        $this->errors = $errors;
    }

    public function getErrors() {
        return $this->errors;
    }
}

// Usage
throw new ValidationException('Validation failed', [
    'email' => 'Invalid email format',
    'phone' => 'Phone number too short'
]);
```

Files: process_donation.php, auth.php, database.php

UNIT-IV: Database & MySQL

✅ Concepts Implemented

1. PHP Data Objects (PDO)

Database Connection:

```php
```

```php
// config/database.php
try {
    $pdo = new PDO(
        "mysql:host=localhost;dbname=sancturia_wildlife;charset=utf8mb4",
        "root",
        "",
        [
            PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
            PDO::ATTR_EMULATE_PREPARES => false,
            PDO::ATTR_PERSISTENT => true
        ]
    );
} catch(PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
```

Why PDO?

- Database-agnostic (works with MySQL, PostgreSQL, SQLite)
- Prepared statements prevent SQL injection
- Object-oriented interface
- Better error handling with exceptions

---

2. MySQLi vs PDO

Project uses PDO (more modern approach)

PDO Advantages Utilized:

- Named parameters: `:email`, `:password`
- Exception-based error handling
- Support for transactions
- Fetch modes (FETCH_ASSOC)

---

3. Creating Database & Tables

Database Schema:

sql

```sql
-- Create database
CREATE DATABASE sancturia_wildlife;
USE sancturia_wildlife;

-- Users table
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    donation_total DECIMAL(10,2) DEFAULT 0,
    adoptions_count INT DEFAULT 0,
    last_login DATETIME,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Donations table
CREATE TABLE donations (
    donation_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    donor_name VARCHAR(255) NOT NULL,
    donor_email VARCHAR(255) NOT NULL,
    donor_phone VARCHAR(20),
    amount DECIMAL(10,2) NOT NULL,
    sanctuary_name VARCHAR(255),
    recurring_type ENUM('none', 'monthly', 'yearly') DEFAULT 'none',
    donation_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE SET NULL
);

-- Sanctuaries table
CREATE TABLE sanctuaries (
    sanctuary_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    location VARCHAR(255),
    description TEXT,
    image_path VARCHAR(255),
    website_url VARCHAR(255)
);

-- Adoptions table
CREATE TABLE adoptions (
    adoption_id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    user_id INT,
    animal_name VARCHAR(255),
    animal_type VARCHAR(255),
    adoption_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

---

4. Relational Database & SQL

Relationships Implemented:

1. **One-to-Many:** User → Donations

```
One user can have many donations
```

2. **One-to-Many:** User → Adoptions

```
One user can adopt many animals
```

3. **Foreign Keys:**

```sql
FOREIGN KEY (user_id) REFERENCES users(user_id)
```

---

5. CRUD Operations

CREATE (INSERT):

```php
```

```
// signup.php - Register user
$stmt = $pdo->prepare("INSERT INTO users (name, email, password) VALUES (?, ?, ?)");
$stmt->execute([$name, $email, $hashed_password]);

// process_donation.php - Insert donation
$stmt = $pdo->prepare("
    INSERT INTO donations
    (user_id, donor_name, donor_email, donor_phone, amount, sanctuary_name, recurring_type, donation_date)
    VALUES (?, ?, ?, ?, ?, ?, ?, NOW())
");
$stmt->execute([$user_id, $donor_name, $donor_email, $donor_phone, $amount, $sanctuary_name, $recurring_type]);
```

READ (SELECT):

```
php

// login.php - Fetch user
$stmt = $
```