Sancturia Wildlife Conservation Platform

Complete Technical Documentation

Project Name: Sancturia Wildlife

Purpose: Wildlife conservation donation and sanctuary management platform

Technology Stack: PHP, MySQL, JavaScript, HTML5, CSS3, Bootstrap 5

Date: October 2024

Author: [Your Name]

Table of Contents

---

1. Project Overview

Sancturia Wildlife is a full-stack web application designed to connect wildlife conservation enthusiasts with Indian wildlife sanctuaries. The platform facilitates:

- **Direct Donations** to specific sanctuaries or general conservation funds
- **User Account Management** with authentication and authorization
- **Sanctuary Discovery** through search and browse functionality
- **Personal Dashboards** tracking individual conservation impact
- **Responsive Design** accessible across all devices

Core Functionality

- User registration and login with session management

- Donation processing with one-time and recurring options
- Sanctuary database with search/filter capabilities
- User dashboard showing donation history and statistics
- CSRF-protected forms for security

---

2. System Architecture

Technology Stack

- **Frontend:** HTML5, CSS3, JavaScript (ES6+), Bootstrap 5
- **Backend:** PHP 7.4+
- **Database:** MySQL (via PDO)
- **Session Management:** PHP Sessions
- **Security:** CSRF tokens, password hashing (bcrypt)

Application Flow

```
Client Request → Navbar (Dynamic) → Page Content → PHP Processing → Database → Response
```

---

3. File-by-File Analysis

3.1 Root Level Files

index.html

- **Purpose:** Landing page/homepage of the application
- **Key Features:**
  - Hero section with conservation message
  - Central donation widget with preset amounts (₹100, ₹500, ₹1000, ₹2000)
  - Featured sanctuary cards (Jim Corbet, Ranthambore, Chilika, Nal Sarovar)
  - Recurring donation options (monthly/yearly)
- **Interactions:**
  - Loads `navbar.html` dynamically via `navbar.js`

- Uses `home.js` for donation button interactions
- Stores selected sanctuary names in `localStorage`
- Redirects to `sancturies.html` on sanctuary card clicks
- Links to `donate.php` for donation processing

- **Styling:** `homeStyle.css`

homeStyle.css

- **Purpose:** Styles for homepage
- **Key Styles:**
  - Full-screen jungle background image (`.jungleBg`)
  - Hero section with large typography (Sofia font, 60px)
  - Donation section with glassmorphism effect (backdrop-filter blur)
  - Sanctuary image cards with hover animations (scale 82%)
  - Custom radio buttons styled with goldenrod theme
  - Responsive breakpoints (1550px, 1199px, 991px, 872px, 768px, 576px)
- **Responsive Design:**
  - Hides columns progressively on smaller screens
  - Adjusts donation section width (30dvw → 95vw)
  - Scales navbar for tablet/mobile views

home.js

- **Purpose:** Homepage interaction logic
- **Functionality:**
  - Handles preset donation button clicks
  - Updates donation input placeholder dynamically
  - Stores donation amount in `localStorage` as `DonationPreset`
  - Handles sanctuary card clicks
  - Stores sanctuary name in `localStorage` as `SanctuaryName`
  - Enables radio button toggling (click to uncheck)

    - Enter key support for donation amount input
- **localStorage Keys:**
    - `DonationPreset`: Selected donation amount
    - `SanctuaryName`: Selected sanctuary for donation

.htaccess

- **Purpose:** Apache server configuration
- **Security Features:**
    - Prevents directory listing (`Options -Indexes`)
    - Hides dot files (`.env`, `.git`, etc.)
    - Protects `config/*` and `includes/*` directories
    - Sets security headers (XSS protection, content-type sniffing prevention, frame options)
    - Configures PHP settings (upload limits, execution time)
- **Error Handling:**
    - Custom 404 and 500 error pages
    - UTF-8 charset enforcement

3.2 Pages Directory

/pages/sanctuaries/sancturies.php

- **Purpose:** Displays all wildlife sanctuaries with search functionality
- **Database Interaction:**
    - Fetches all sanctuaries via `SELECT * FROM sanctuaries ORDER BY RAND()`
    - Searches sanctuaries by name using `LIKE` query with prepared statements
    - Handles both URL parameter search (`?search=`) and localStorage-based search
- **Key Features:**
    - Modal popup for searched sanctuary details
    - "Donate Now" buttons storing sanctuary name in localStorage
    - Grid layout of sanctuary cards with images, descriptions, website links
    - Displays location and sanctuary information

- **Interactions:**
  - Receives `from_home` parameter from homepage sanctuary clicks
  - Redirects to `donate.php` with sanctuary parameter on donation click
  - Uses `navbar.js` for navigation
- **Styling:** `sancturiesStyle.css`
- **Security:** Uses `htmlspecialchars()` for XSS prevention

/pages/sanctuaries/sancturiesStyle.css

- **Purpose:** Styles for sanctuary listing page
- **Key Features:**
  - Background image with fixed positioning
  - Sanctuary grid layout (auto-fit, minmax 270px)
  - Card-based design with hover effects
  - Search bar styling with focus states
  - Modal styling with animations (slideDown keyframe)
  - Orange "Donate" buttons (btn-donate)
- **Responsive Design:**
  - Adjusts navbar scale at 1200px, 990px, 768px breakpoints

/pages/donate/donate.php

- **Purpose:** Main donation page with form
- **Authentication Check:**
  - Detects if user is logged in via `$_SESSION['user_id']`
  - Shows login prompt if not authenticated
  - Preserves form data in session for post-login redirect
- **Form Fields:**
  - Donor name, email, phone (10-digit validation)
  - Donation amount (number input)
  - Recurring options (monthly/yearly radio buttons)
  - Hidden sanctuary name field

- **Features:**
  - Displays sanctuary name from URL parameter or localStorage
  - Welcome back message for logged-in users
  - Three donation tiers with animated icons (Soil, Habitat, Earth)
  - Video animations on tier hover
- **Interactions:**
  - Submits to `process_donation.php` via POST
  - Includes CSRF token via `csrf_field()`
  - Redirects to login with return URL if not authenticated
  - Uses `navbar.js` and `donate.js`
- **Styling:** `donateStyle.css`

/pages/donate/donateStyle.css

- **Purpose:** Styles for donation page
- **Key Features:**
  - Video background (`.bgVid` with object-fit cover)
  - Grid layout for main content (auto-fit, minmax 450px)
  - Tier cards with unique background colors (brown, teal, green)
  - Video icons that scale on hover (50% → 120%)
  - Form styling with goldenrod border theme
  - Glassmorphism effects on donation section
- **Responsive Design:**
  - Stacks grid vertically on screens < 920px
  - Removes right border on mobile for donation section

/pages/donate/donate.js

- **Purpose:** Donation page interactivity
- **Functionality:**
  - Loads donation preset from localStorage on page load
  - Tier card hover animations (plays video, scales icon)

  - Mouse out stops video playback
  - "Donate Now" button triggers simultaneous animation of all three tier icons
  - Resets video playback to start on animation trigger
- **Video Management:**
  - Controls loop property dynamically
  - Manages currentTime for synchronized playback
  - Scales videos (100% → 130% → 100%)

/pages/donate/process_donation.php

- **Purpose:** Backend donation processing script
- **Process Flow:**
  1. Validates CSRF token
  2. Sanitizes and validates form inputs (name, email, phone, amount)
  3. Retrieves user_id from session if logged in
  4. Inserts donation into database with transaction
  5. Updates user's total donation amount
  6. Redirects to `thankyou.html` with donation details
- **Validation:**
  - Email format validation (`FILTER_VALIDATE_EMAIL`)
  - 10-digit phone number regex
  - Positive amount validation
  - Empty field checks
- **Database Operations:**
  - INSERT into `donations` table
  - UPDATE `users` table donation_total
  - Uses PDO transactions for data integrity
- **Error Handling:**
  - Rolls back transaction on failure
  - Logs errors via `error_log()`

    - Sets user-friendly error messages in session
- **Security:** Prepared statements, input sanitization

/pages/donate/thankyou.html

- **Purpose:** Donation confirmation/receipt page
- **Features:**
    - Animated checkmark icon with scaling effect
    - Displays donation details from URL parameters
    - Shows transaction ID, date/time, amount
    - Donor information (name, email, phone)
    - Sanctuary name and recurring type
    - Print receipt button
    - Return to home link
- **JavaScript Functionality:**
    - Parses URL parameters to populate page
    - Generates transaction ID format: `TXN{YEAR}{MONTH}{DAY}{ID}`
    - Formats date with locale options
    - Localizes currency (₹ symbol with Indian formatting)
- **Design:**
    - Green gradient hero section
    - Card-based info display
    - Animated fade-in effects (slideDown, fadeInUp, scaleIn)
    - Disclaimer that it's a prototype (no actual transaction)

/pages/login-signup/signup.php

- **Purpose:** User registration page
- **Backend Processing:**
    - Validates CSRF token
    - Checks for empty fields
    - Verifies password match (password === confirm_password)

    - Enforces 8-character minimum password
    - Calls `register_user()` function from `auth.php`
    - Creates session and redirects to dashboard on success
- **Form Fields:**
    - Full name
    - Email address
    - Password
    - Confirm password
- **Security:**
    - Password hashing via bcrypt
    - Session regeneration on successful registration
    - Duplicate email check
- **Error Display:**
    - Shows errors from session via alert-danger
    - Clears error after display
- **Styling:** `sign-style.css`
- **Design:** Right-aligned form card with background overlay

/pages/login-signup/login.php

- **Purpose:** User authentication page
- **Backend Processing:**
    - Validates CSRF token
    - Checks for empty fields
    - Calls `login_user()` function from `auth.php`
    - Verifies password hash
    - Updates last_login timestamp
    - Creates session and redirects appropriately
- **Special Features:**
    - Captures redirect parameter from donate page (`?redirect=donate`)

  - Stores donation intent in session (`donation_sanctuary`)
  - Redirects back to donate page after login with sanctuary parameter
- **Redirect Logic:**

```
User tries to donate → Not logged in → Login page → After login → Donate page (with sanctuary)
```

- **Security:**
  - Password verification via `password_verify()`
  - Session regeneration on successful login
- **Styling:** `sign-style.css`

/pages/login-signup/sign-style.css

- **Purpose:** Unified styles for login and signup pages
- **Key Features:**
  - Full-screen background with overlay gradient
  - Right-aligned form card with glassmorphism (rgba + backdrop-filter)
  - Green-themed buttons (gradient: 🟩 `#22c55e` → 🟩 `#16a34a`)
  - Rounded input fields with focus states (green shadow)
  - Animation: slideInRight on page load
- **Form Styling:**
  - Label font size 0.875rem, weight 600
  - Input padding 0.875rem, border-radius 12px
  - Button hover lifts with translateY(-1px)
  - Alert messages with colored backgrounds
- **Responsive Design:**
  - Centers form on mobile < 768px
  - Reduces padding on small screens
  - Adjusts font sizes for mobile

/pages/login-signup/logout.php

- **Purpose:** Terminates user session
- **Process:**
  1. Starts session
  2. Clears all session variables (`$_SESSION = array()`)
  3. Destroys session cookie
  4. Calls `session_destroy()`
  5. Redirects to homepage with success parameter
- **Security:** Ensures complete session cleanup

/pages/navbar/navbar.html

- **Purpose:** Reusable navigation component
- **Structure:**
  - Bootstrap navbar with brand logo
  - Navigation links (Home, About, Donate, Sanctuaries, Adopt)
  - Auth buttons container (dynamically populated)
  - Collapsible for mobile responsiveness
- **Dynamic Content:**
  - Auth buttons populated by `navbar.js` based on login status
  - Logo changes (black/white) based on page theme
  - Button styles adapt to home vs other pages
- **Links:**
  - All use absolute paths from root (`/index.html`, `/pages/donate/donate.php`)

/pages/navbar/navbar.js

- **Purpose:** Dynamic navbar loading and authentication state management
- **Functionality:**
  1. **Load Navbar:**
     - Determines correct path based on current location
     - Fetches `navbar.html` via Fetch API
     - Injects into `#navbar-placeholder`

   2. **Apply Theme:**
      - Detects homepage via `isHomePage()` function
      - Home: White logo, light buttons, dark navbar
      - Other pages: Black logo, dark buttons, light navbar
   3. **Check Login Status:**
      - Fetches `/pages/auth/check_session.php`
      - Parses JSON response
      - Updates auth buttons container with:
        - Logged in: Username, Dashboard, Logout buttons
        - Not logged in: Sign Up, Login buttons
   4. **Highlight Active Page:**
      - Compares current URL with nav link hrefs
      - Applies solid button style to active page
- **Error Handling:**
  - Fallback navbar if fetch fails
  - Console logging for debugging
- **Helper Function:**
  - `escapeHtml()`: Prevents XSS in displayed usernames

/pages/dashboard/dashboard.php

- **Purpose:** User personal dashboard
- **Authentication:**
  - Requires login via `check_login()`
  - Redirects to login page if not authenticated
- **Data Fetched:**
  - User profile via `get_user_data($user_id)`
  - Recent 5 donations: `SELECT * FROM donations WHERE user_id = ? ORDER BY donation_date DESC LIMIT 5`
  - Recent 3 adoptions: `SELECT * FROM adoptions WHERE user_id = ? ORDER BY adoption_date DESC LIMIT 3`

    - 3 random recommended sanctuaries

- **Statistics Displayed:**
    - Total donated (₹)
    - Animals adopted (count)
    - Number of donations made
    - Member since date
- **UI Components:**
    - Sidebar navigation with logo, nav items, user profile, logout
    - Top bar with breadcrumb and date display
    - Welcome banner
    - Statistics grid (4 cards)
    - Content grid (donations, adoptions)
    - Recommended sanctuaries section
- **Empty States:**
    - Shows "No donations yet" with action button
    - Shows "Coming Soon" for adoptions
- **Styling:** `dashboard-style.css`

/pages/dashboard/dashboard-style.css

- **Purpose:** Clean minimal dashboard design
- **Design System:**
    - CSS Variables for theming (primary: `#22c55e`, bg: `#fafafa`)
    - Inter font family
    - Sidebar width: 250px fixed
- **Components:**
    - **Sidebar:** Fixed left, dark green background, flex column
    - **Nav Items:** Hover background change, active state with primary color
    - **User Profile:** Small avatar (32px), name/email display
    - **Logout:** Red themed (`#ef4444`)

    - **Main Content:** Left margin equal to sidebar width
    - **Stat Cards:** Grid layout, hover border color change
    - **List Items:** Background on hover, flexbox layout
- **Responsive:**
    - Hides sidebar on mobile (< 768px)
    - Sets sidebar width to 0
    - Removes main content left margin
    - Stacks grids into single column

/pages/about/about.html

- **Purpose:** About/information page
- **Content Sections:**
    - Our Mission: Platform purpose and goals
    - Our Vision: Future aspirations for wildlife
    - Our Story: Founded in 2024, conservation passion
    - How We Work: Explains donation process
    - Join Us: Call to action
- **Interactions:**
    - Loads navbar dynamically
    - Links to social media (placeholders)
- **Styling:** `aboutStyle.css`

/pages/about/aboutStyle.css

- **Purpose:** Styles for about page
- **Key Features:**
    - Fixed background image (`.aboutBg`)
    - Green header (🟩 `#4CAF50`)
    - Content in centered container (max-width 70vw)
    - Cream background with backdrop blur (rgba(255, 244, 221, 0.897))
    - Sofia font for headings

    - HR separators between sections

- **Responsive:**
    - Reduces padding on mobile
    - Adjusts heading font size

/pages/adopt/adopt.html

- **Purpose:** Animal adoption feature (under construction)
- **Content:**
    - Construction icon (🚧)
    - Explanation of upcoming feature
    - Description: Symbolic animal adoption with naming
    - Back to home button
- **Styling:** Inline styles
- **Background:** Construction-themed image with yellow accent

---

3.3 Backend/Server Files

/config/database.php

- **Purpose:** Database connection configuration
- **Configuration:**
    - Host: localhost
    - Database: sancturia_wildlife
    - User: root
    - Password: (empty for XAMPP)
- **PDO Options:**
    - `ERRMODE_EXCEPTION`: Throws exceptions on errors
    - `FETCH_ASSOC`: Returns associative arrays
    - `EMULATE_PREPARES: false`: Uses real prepared statements
    - `PERSISTENT: true`: Reuses connections

- **Error Handling:**
  - Dies with connection error message on failure

- **Used By:** All PHP scripts requiring database access

/includes/auth.php

- **Purpose:** Authentication and user management functions

- **Functions:**
  1. **register_user($name, $email, $password):**
     - Checks for duplicate email
     - Hashes password with bcrypt
     - Inserts new user into database
     - Returns success/error array
  2. **login_user($email, $password):**
     - Fetches user by email
     - Verifies password hash
     - Updates last_login timestamp
     - Returns user data or error
  3. **check_login():**
     - Validates session user_id exists
     - Returns boolean
  4. **get_user_data($user_id):**
     - Fetches user profile including donation_total, adoptions_count
     - Returns associative array
  5. **logout_user():**
     - Clears and destroys session
     - Returns true

- **Security:**
  - Uses `PASSWORD_BCRYPT` for hashing
  - Prepared statements prevent SQL injection

    - Try-catch blocks for error handling
- **Dependencies:** `database.php`

/includes/csrf.php

- **Purpose:** CSRF token generation and validation
- **Functions:**
  1. **generate_csrf_token():**
     - Creates 32-byte random token via `random_bytes()`
     - Converts to hexadecimal string
     - Stores in session
     - Returns token
  2. **validate_csrf_token($token):**
     - Compares submitted token with session token
     - Uses `hash_equals()` for timing-safe comparison
     - Returns boolean
  3. **csrf_field():**
     - Generates hidden input field with token
     - Includes `htmlspecialchars()` for XSS protection
     - Returns HTML string
- **Usage:**
  - Included in all forms (signup, login, donate)
  - Validated on POST request processing
- **Security:** Prevents Cross-Site Request Forgery attacks

/includes/errors.php

- **Purpose:** Error logging and display utilities
- **Functions:**
  1. **log_error($message, $level):**
     - Creates `/logs` directory if not exists
     - Appends timestamped error to `error.log`

        - Format: `[YYYY-MM-DD HH:MM:SS] [LEVEL] Message`

    2. **show_error($message):**
        - Displays Bootstrap alert-danger div
        - Escapes HTML to prevent XSS

- **Usage:** Called in process_donation.php and other error-prone scripts

/pages/auth/check_session.php

- **Purpose:** AJAX endpoint for login status
- **Functionality:**
    - Returns JSON response
    - Checks `$_SESSION['user_id']` existence
    - If logged in: Returns user_id, user_name, user_email
    - If not logged in: Returns `logged_in: false`
- **Headers:**
    - Content-Type: application/json
    - Cache-Control: no-cache (prevents caching)
    - Expires: Past date
- **Used By:** `navbar.js` for dynamic button display

---

4. Inter-File Relationships

Dependency Map

```
index.html
  ├── homeStyle.css
  ├── home.js
  │   └── localStorage → sancturies.php
  └── navbar.js
      └── navbar.html
          └── check_session.php

sancturies.php
  ├── sancturiesStyle.css
  ├── database.php
  ├── navbar.js
  └── localStorage → donate.php

donate.php
  ├── donateStyle.css
  ├── donate.js
  ├── csrf.php
  ├── navbar.js
  └── process_donation.php
      ├── database.php
      ├── csrf.php
      └── thankyou.html

login.php
  ├── sign-style.css
  ├── csrf.php
  ├── auth.php
  │   └── database.php
  └── → dashboard.php OR donate.php

signup.php
  ├── sign-style.css
  ├── csrf.php
  ├── auth.php
  │   └── database.php
  └── → dashboard.php

dashboard.php
  ├── dashboard-style.css
  ├── auth.php
  │   └── database.php
  └── check_login() → login.php

about.html
  ├── aboutStyle.css
  └── navbar.js

adopt.html
  └── (under construction)

navbar.js (loaded on all pages)
  ├── navbar.html
  └── check_session.php
```

Data Flow Diagram

```
User Action → Frontend (HTML/JS) → Backend (PHP) → Database (MySQL) → Response

Example: Making a Donation
1. User clicks sanctuary on index.html
2. home.js stores sanctuary name in localStorage
3. User redirected to sancturies.php
4. User clicks "Donate Now"
5. donate.php checks if logged in
6. If not: redirect to login.php
7. After login: redirect back to donate.php
8. User fills form, submits
9. process_donation.php validates data
10. INSERT into donations table
11. UPDATE users table
12. Redirect to thankyou.html
```

Session Data Flow

```
signup.php: Create session → $_SESSION['user_id']
login.php: Create session → $_SESSION['user_id']
check_session.php: Read session → JSON response
navbar.js: Parse JSON → Display username/logout
dashboard.php: Read session → Fetch user data
logout.php: Destroy session → Redirect home
```

localStorage Usage

```
home.js:
  - DonationPreset → Amount selected
  - SanctuaryName → Sanctuary clicked

donate.js:
  - Reads DonationPreset → Populates placeholder

sancturies.php:
  - Reads SanctuaryName → Searches sanctuary

donate.php:
  - Reads DonateSanctuary → Pre-fills sanctuary
```

5. Database Schema

Tables Used

users

```sql
- user_id (INT, PRIMARY KEY, AUTO_INCREMENT)
- name (VARCHAR)
- email (VARCHAR, UNIQUE)
- password (VARCHAR) -- bcrypt hashed
- donation_total (DECIMAL) -- aggregated donation amount
- adoptions_count (INT) -- number of adoptions
- last_login (DATETIME)
- created_at (TIMESTAMP)
```

donations

```sql
- donation_id (INT, PRIMARY KEY, AUTO_INCREMENT)
- user_id (INT, FOREIGN KEY) -- nullable for guest donations
- donor_name (VARCHAR)
- donor_email (VARCHAR)
- donor_phone (VARCHAR)
- amount (DECIMAL)
- sanctuary_name (VARCHAR)
- recurring_type (ENUM: 'none', 'monthly', 'yearly')
- donation_date (DATETIME)
```

sanctuaries

```sql
- sanctuary_id (INT, PRIMARY KEY, AUTO_INCREMENT)
- name (VARCHAR)
- location (VARCHAR)
- description (TEXT)
- image_path (VARCHAR)
- website_url (VARCHAR)
```

adoptions

```sql
- adoption_id (INT, PRIMARY KEY, AUTO_INCREMENT)
- user_id (INT, FOREIGN KEY)
- animal_name (VARCHAR)
- animal_type (VARCHAR)
- adoption_date (DATETIME)
```

---

6. Security Features

1. CSRF Protection

- **Implementation:** `csrf.php`
- **Token Generation:** 32-byte random via `random_bytes()`
- **Validation:** Timing-safe comparison via `hash_equals()`
- **Forms Protected:** Login, signup, donation

2. Password Security

- **Hashing Algorithm:** bcrypt (PASSWORD_BCRYPT)
- **Verification:** `password_verify()` function
- **No Plain Text Storage:** Passwords never stored unhashed

3. SQL Injection Prevention

- **Method:** PDO prepared statements
- **Parameterization:** All user inputs bound as parameters
- **Example:** `$stmt->execute([$user_id, $name, $email])`

4. XSS Prevention

- **Output Encoding:** `htmlspecialchars()` on all user-generated content
- **Escaping:** `escapeHtml()` function in navbar.js
- **Content Security:** No inline script execution from user input

- **Regeneration:** `session_regenerate_id(true)` after login/signup
- **Cookie Destruction:** Removes session cookies on logout
- **Timeout:** Sessions expire after inactivity

6. Input Validation

- **Email:** `FILTER_VALIDATE_EMAIL` filter
- **Phone:** Regex pattern `/^[0-9]{10}$/`
- **Amount:** Positive number validation
- **Required Fields:** Empty string checks

7. Access Control

- **Authentication Check:** `check_login()` before dashboard access
- **Redirect:** Unauthenticated users sent to login page
- **Authorization:** Users can only view their own data

8. Server Configuration (.htaccess)

- **Directory Listing:** Disabled
- **File Protection:** Hides .htaccess, .env, config/, includes/
- **Headers:** X-XSS-Protection, X-Content-Type-Options, X-Frame-Options
- **Upload Limits:** Configured via PHP directives

---

7. Complete Directory Structure

```
SANCTURIA_WILDLIFE/
│
├── index.html                  # Homepage/landing page
├── homeStyle.css                 # Homepage styles
├── home.js                    # Homepage interactions
├── .htaccess                  # Apache configuration
├── README.md                     # Project documentation
├── test_db.php                 # Database connection test
│
├── Assets_TBU/                   # Static assets
```

```
│   ├── Background Images/            # Background images and logos
│   │   ├── HomeJungleBg.jpeg
│   │   ├── SancturiesBg.jpeg
│   │   ├── signup-loginBg.jpeg
│   │   ├── About Page Bg.jpg
│   │   ├── ConstructionBg.jpeg
│   │   ├── Sancturia Logo Black.png
│   │   ├── Sancturia Logo White.png
│   │   ├── Sancturia Logo Green.png
│   │   └── Preview Thumbnail.jpg
│   ├── Animal Images/               # Sanctuary animal photos
│   │   ├── Horndeer.jpg
│   │   ├── Leopard.jpg
│   │   ├── Parrot.jpg
│   │   └── Tucan.jpg
│   └── Logos Icons/                 # Animated icons
│       ├── leavesAnim.mp4
│       ├── elephantAnim.mp4
│       └── ecologyAnim.mp4
│
├── api/                             # (Future API endpoints)
│
├── config/                          # Configuration files
│   └── database.php                 # Database connection & PDO setup
│
├── includes/                        # Reusable PHP functions
│   ├── auth.php                     # Authentication functions
│   ├── csrf.php                     # CSRF token management
│   └── errors.php                   # Error logging utilities
│
├── Docx/                            # Documentation files
│
├── node_modules/                    # npm dependencies (if any)
│
├── References/                      # Reference materials
│
├── scripts/                         # Utility scripts
│
├── Showcase Images/                 # Screenshots for presentation
│
└── pages/                           # Application pages
    │
    ├── about/                       # About page
    │   ├── about.html               # About content
    │   └── aboutStyle.css            # About page styles
    │
    ├── adopt/                        # Adoption feature (under construction)
    │   └── adopt.html                # Construction notice page
    │
    ├── auth/                         # Authentication endpoints
    │   └── check_session.php         # AJAX session check endpoint
    │
    ├── dashboard/                    # User dashboard
    │   ├── dashboard.php             # Dashboard page (requires login)
    │   └── dashboard-style.css       # Dashboard styles
    │
    ├── donate/                       # Donation system
    │   ├── donate.php                # Donation form page
    │   ├── donateStyle.css           # Donation page styles
    │   ├── donate.js                 # Donation interactions
    │   ├── process_donation.php      # Backend donation processor
    │   └── thankyou.html             # Donation confirmation page
    │
    ├── login-signup/                 # User authentication
    │   ├── login.php                 # Login form & processing
    │   ├── signup.php                # Registration form & processing
    │   ├── logout.php                # Logout handler
    │   └── sign-style.css            # Login/signup unified styles
    │
    ├── navbar/                       # Navigation component
    │   ├── navbar.html               # Navbar HTML template
    │   └── navbar.js                 # Navbar loader & theme manager
    │
    └── sanctuaries/                  # Sanctuary listing
        ├── sancturies.php            # Sanctuary search & display
        └── sancturiesStyle.css       # Sanctuary page styles
```

- **Total Files:** 40+
- **PHP Files:** 10
- **HTML Files:** 7

- **CSS Files:** 7
- **JavaScript Files:** 4
- **Configuration Files:** 3

Summary

Project Statistics

Key Achievements

1. **Full-Stack Implementation:** Complete frontend-backend integration
2. **Security-First Design:** CSRF protection, password hashing, SQL injection prevention
3. **Responsive Design:** Mobile-first approach with Bootstrap 5
4. **User Experience:** Smooth navigation, persistent state via sessions and localStorage
5. **Modular Architecture:** Reusable components (navbar, auth functions)
6. **Database Normalization:** Proper relational structure with foreign keys

Authentication System

Files Involved: `login.php`, `signup.php`, `logout.php`, `auth.php`, `csrf.php`, `check_session.php`

Flow:

1. User registers via `signup.php` → Validates input → Hashes password → Inserts into database → Creates session
2. User logs in via `login.php` → Verifies credentials → Creates session → Redirects to dashboard
3. Session persists across pages → Checked by `check_session.php` → Updates navbar display
4. User logs out via `logout.php` → Destroys session → Redirects to homepage

Security Measures:

- CSRF tokens on all forms
- Password hashing (bcrypt)
- Session regeneration on auth events
- Prepared statements for database queries

Donation System

Files Involved: `donate.php`, `donate.js`, `process_donation.php`, `thankyou.html`, `home.js`

User Journey:

1. **Homepage:** User selects preset donation amount OR enters custom amount
2. **Sanctuary Selection:** User clicks sanctuary card → Name stored in localStorage
3. **Donate Page:** User lands on donate.php with sanctuary pre-filled
4. **Authentication Check:** If not logged in → Redirected to login with return URL
5. **Form Submission:** User enters details → Submits with CSRF token
6. **Processing:** Backend validates → Inserts into database → Updates user totals
7. **Confirmation:** Redirected to thankyou.html with transaction details

Data Persistence:

- localStorage: DonationPreset, SanctuaryName, DonateSanctuary
- Session: donation_sanctuary, redirect_after_login
- Database: donations table, users.donation_total

Validation:

- Client-side: HTML5 required, pattern, min/max
- Server-side: Empty checks, email format, phone regex, positive amount

---

Sanctuary Management

Files Involved: `sancturies.php`, `sancturiesStyle.css`, `database.php`

Features:

1. **Listing:** Displays all sanctuaries in grid layout with images
2. **Search:** Form-based search by sanctuary name
3. **Modal Display:** Searched sanctuary appears in overlay modal
4. **Homepage Integration:** Receives sanctuary name from homepage clicks
5. **Donation Integration:** "Donate Now" buttons pass sanctuary to donate page

Database Queries:

```php
// List all sanctuaries (randomized)
SELECT * FROM sanctuaries ORDER BY RAND()

// Search by name
SELECT * FROM sanctuaries WHERE name LIKE ? LIMIT 1
```

User Interactions:

- Click sanctuary card → Modal opens with details
- Click "Get Info Here" → Opens sanctuary website in new tab
- Click "Donate Now" → Redirects to donate page with sanctuary parameter
- Click backdrop or close → Modal disappears, URL cleaned

Dashboard Analytics

Files Involved: `dashboard.php`, `dashboard-style.css`, `auth.php`

Protected Route:

- Requires authentication via `check_login()`
- Redirects to login if user_id not in session

Data Displayed:

1. **User Profile:** Name, email, avatar
2. **Statistics Cards:**
   - Total donated (aggregated from donations)
   - Animals adopted (from adoptions table)
   - Number of donations made
   - Member since date (user.created_at)
3. **Recent Donations:** Last 5 donations with sanctuary names and amounts
4. **Recent Adoptions:** Last 3 adoptions with animal details
5. **Recommended Sanctuaries:** 3 random sanctuaries with explore links

Sidebar Navigation:

- Dashboard (current)
- Home, About, Donate, Sanctuaries, Adopt
- User profile display
- Logout button

Empty States:

- No donations: Shows "Make a Donation" link
- No adoptions: Shows "Coming Soon" message

Navigation System

Files Involved: `navbar.html`, `navbar.js`, `check_session.php`

Dynamic Loading:

1. Each page has `<div id="navbar-placeholder"></div>`
2. `navbar.js` loads on page load
3. Determines correct path (root vs /pages/)
4. Fetches `navbar.html` via Fetch API
5. Injects HTML into placeholder

Theme Adaptation:

- **Homepage Detection:** Checks if current page is index.html or root
- **Home Theme:** White logo, light buttons (btn-outline-light), dark navbar
- **Other Pages Theme:** Black logo, dark buttons (btn-outline-dark), light navbar

Authentication Display:

1. Fetches `check_session.php` endpoint
2. Parses JSON response
3. If logged in:
    - Shows username with icon
    - Shows "Dashboard" button
    - Shows "Logout" button
4. If not logged in:
    - Shows "Sign Up" button
    - Shows "Login" button

Active Page Highlighting:

- Compares current URL with nav link hrefs
- Applies solid button style (btn-light or btn-dark) to active page

---

Technical Implementation Details

Session Management

Workflow:

```php
// Starting session (all PHP files)
session_start();

// Setting session data (signup.php, login.php)
$_SESSION['user_id'] = $user_id;
$_SESSION['user_name'] = $name;
$_SESSION['user_email'] = $email;

// Reading session data (dashboard.php)
$user_id = $_SESSION['user_id'];

// Regenerating session (security measure)
session_regenerate_id(true);

// Destroying session (logout.php)
session_destroy();
```

Security Considerations:

- Session IDs regenerated after login/signup to prevent fixation attacks
- Session cookies destroyed on logout
- Sessions timeout after inactivity (PHP default)

Database Connection (PDO)

Configuration:

```php
```

```
$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
    ]
);
```

Why PDO?

- Prepared statements prevent SQL injection
- Supports multiple databases (MySQL, PostgreSQL, SQLite)
- Exception-based error handling
- Persistent connections improve performance

Query Examples:

```php
// Prepared statement with named parameters
$stmt = $pdo->prepare("SELECT * FROM users WHERE email = ?");
$stmt->execute([$email]);
$user = $stmt->fetch();

// INSERT with multiple parameters
$stmt = $pdo->prepare("INSERT INTO donations (user_id, amount, sanctuary_name) VALUES (?, ?, ?)");
$stmt->execute([$user_id, $amount, $sanctuary]);

// Transaction for data integrity
$pdo->beginTransaction();
// ... multiple queries ...
$pdo->commit();
// Or rollback on error
$pdo->rollBack();
```

CSRF Protection Mechanism

Token Generation:

```php
function generate_csrf_token() {
    if (!isset($_SESSION['csrf_token'])) {
        $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
    }
    return $_SESSION['csrf_token'];
}
```

Token Embedding:

```php
// In forms
<?php echo csrf_field(); ?>
// Outputs: <input type="hidden" name="csrf_token" value="...">
```

Token Validation:

```php
function validate_csrf_token($token) {
    if (!isset($_SESSION['csrf_token'])) {
        return false;
    }
    return hash_equals($_SESSION['csrf_token'], $token);
}

// In processing scripts
if (!validate_csrf_token($_POST['csrf_token'] ?? '')) {
    die('CSRF validation failed');
}
```

Why hash_equals?

- Prevents timing attacks
- Compares strings in constant time
- More secure than `===` for cryptographic values

Password Hashing

Registration:

```php
$hashed_password = password_hash($password, PASSWORD_BCRYPT);
// Stores: $2y$10$randomsalt.hashedpassword
```

Login Verification:

```php
if (password_verify($password, $user['password'])) {
    // Password correct
} else {
    // Password incorrect
}
```

Security Benefits:

- Bcrypt algorithm automatically salts passwords
- Computationally expensive (slows brute-force attacks)
- Future-proof (can rehash with stronger algorithms)

LocalStorage vs Session Storage

Why localStorage?

- Persists across browser sessions
- Survives page refreshes
- Shared across all tabs of same origin

Usage in This Project:

```javascript
```

```
// Storing donation preference
localStorage.setItem('DonationPreset', '1000');

// Reading on another page
let amount = localStorage.getItem('DonationPreset');

// Removing after use
localStorage.removeItem('SanctuaryName');

// Clearing all
localStorage.clear();
```

Alternative: sessionStorage

- Only persists for current tab
- Clears when tab closes
- More secure for sensitive data

Bootstrap 5 Integration

Grid System:

```html
<!-- Responsive columns -->
<div class="row">
  <div class="col-xl-2 col-lg-3 col-md-4 col-sm-12">
    <!-- Content -->
  </div>
</div>
```

Breakpoints Used:

- xl: ≥1200px (extra large desktops)
- lg: ≥992px (large desktops)
- md: ≥768px (tablets)
- sm: ≥576px (large phones)

Utility Classes:

- Spacing: `mt-2`, `mb-3`, `p-4`
- Display: `d-flex`, `d-none`, `d-md-inline`
- Text: `text-center`, `text-light`, `fw-bold`
- Buttons: `btn`, `btn-warning`, `btn-outline-dark`

Responsive Design Strategy

Mobile-First Approach:

1. Base styles for mobile (320px-576px)
2. Tablet styles (576px-992px)
3. Desktop styles (992px+)

Techniques Used:

- CSS Grid with `auto-fit` and `minmax()`
- Flexbox for alignment
- Media queries at strategic breakpoints
- Bootstrap grid system
- Viewport units (vw, vh, dvw, dvh)

Example:

```css
```

```
/* Mobile: Stack vertically */
.main-content {
  grid-template-columns: 1fr;
}

/* Tablet: 2 columns */
@media (min-width: 768px) {
  .main-content {
    grid-template-columns: repeat(2, 1fr);
  }
}

/* Desktop: Auto-fit based on content */
@media (min-width: 992px) {
  .main-content {
    grid-template-columns: repeat(auto-fit, minmax(450px, 1fr));
  }
}
```

Error Handling Architecture

Client-Side:

- HTML5 validation (required, pattern, type)
- JavaScript validation before submission
- User-friendly error messages

Server-Side:

```
php
```

```
try {
    // Database operation
    $stmt = $pdo->prepare("...");
    $stmt->execute([...]);
} catch (PDOException $e) {
    // Log error
    log_error("Database error: " . $e->getMessage());

    // Show user-friendly message
    $_SESSION['error'] = 'An error occurred. Please try again.';

    // Redirect back
    header('Location: form.php');
    exit();
}
```

Error Display:

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

Future Enhancements

Planned Features

1. **Email Verification:** Confirm email addresses during registration
2. **Password Reset:** Forgot password functionality
3. **Payment Gateway:** Integrate Razorpay/Stripe for real transactions
4. **Animal Adoption:** Complete the adoption feature with database
5. **Admin Panel:** Manage sanctuaries, view all donations
6. **Donation Receipts:** Generate PDF receipts with 80G tax benefits
7. **Social Sharing:** Share conservation achievements on social media

8. **Donation Analytics:** Charts showing impact over time
9. **Recurring Payments:** Automated monthly/yearly donations
10. **Newsletter:** Email updates about conservation efforts

Technical Improvements

1. **API Development:** RESTful API for mobile app
2. **Caching:** Redis for session storage and query caching
3. **CDN:** Serve static assets from CDN
4. **Image Optimization:** WebP format, lazy loading
5. **SEO Optimization:** Meta tags, sitemap, schema markup
6. **Performance:** Minify CSS/JS, enable gzip compression
7. **Testing:** PHPUnit tests for backend, Jest for frontend
8. **CI/CD:** Automated deployment pipeline
9. **Monitoring:** Error tracking with Sentry
10. **Accessibility:** WCAG 2.1 AA compliance

Deployment Guide

Prerequisites

- Apache/Nginx web server
- PHP 7.4 or higher
- MySQL 5.7 or higher
- mod_rewrite enabled (Apache)

Installation Steps

1. **Clone Repository:**

```bash
git clone <repository-url>
cd sancturia_wildlife
```

2. **Configure Database:**

```sql
CREATE DATABASE sancturia_wildlife;
USE sancturia_wildlife;
-- Run schema.sql to create tables
```

3. **Update Configuration:**

```php
// config/database.php
define('DB_HOST', 'your_host');
define('DB_NAME', 'your_database');
define('DB_USER', 'your_username');
define('DB_PASS', 'your_password');
```

4. **Set Permissions:**

```bash
chmod 755 pages/
chmod 644 .htaccess
mkdir logs
chmod 777 logs/
```

5. **Configure Virtual Host:**

```apache
<VirtualHost *:80>
  DocumentRoot "/path/to/sancturia_wildlife"
  ServerName sancturia.local

  <Directory "/path/to/sancturia_wildlife">
    AllowOverride All
    Require all granted
  </Directory>
</VirtualHost>
```

6. **Test Installation:**
   - Navigate to http://localhost/test_db.php

- Verify database connection
- Test user registration and login

Testing Checklist

Functional Testing

- [ ] User registration with valid/invalid data
- [ ] User login with correct/incorrect credentials
- [ ] Logout functionality
- [ ] Donation form submission
- [ ] Sanctuary search functionality
- [ ] Dashboard data display
- [ ] Navigation links work correctly
- [ ] Responsive design on mobile/tablet/desktop

Security Testing

- [ ] CSRF token validation
- [ ] SQL injection attempts blocked
- [ ] XSS attempts sanitized
- [ ] Password hashing working
- [ ] Session hijacking prevention
- [ ] Unauthorized access to dashboard blocked
- [ ] File upload restrictions (if applicable)

Performance Testing

- [ ] Page load times < 3 seconds
- [ ] Database queries optimized
- [ ] Image sizes optimized
- [ ] CSS/JS minified for production

Conclusion

The Sancturia Wildlife platform successfully demonstrates a complete full-stack web application with:

✅ **Secure Authentication** using sessions, password hashing, and CSRF protection
✅ **Dynamic Content** with database-driven sanctuary listings

✅ **User Dashboard** showing personalized donation analytics
✅ **Responsive Design** working across all device sizes
✅ **Modular Architecture** with reusable components and functions
✅ **Error Handling** with logging and user-friendly messages
✅ **Best Practices** including prepared statements, input validation, and output encoding

Learning Outcomes

- PHP backend development with PDO
- MySQL database design and queries
- Session management and authentication
- CSRF and XSS prevention techniques
- Responsive web design with Bootstrap
- JavaScript DOM manipulation and Fetch API
- Project structure and file organization

Technologies Mastered

- **Frontend:** HTML5, CSS3 (Flexbox, Grid), JavaScript ES6+, Bootstrap 5
- **Backend:** PHP 7.4+, PDO, Session Management
- **Database:** MySQL, SQL queries, Prepared statements
- **Security:** CSRF tokens, Password hashing (bcrypt), Input validation
- **Tools:** Apache, .htaccess, Git version control

Project Status: ✅ Fully Functional Prototype
Code Quality: Production-ready with security best practices
Documentation: Complete technical documentation provided

Prepared by: [Your Name]
Date: October 31, 2024
Institution: [Your Institution Name]

This documentation covers all 40+ files in the Sancturia Wildlife project, explaining their purpose, functionality, and relationships with other components. The platform is ready for demonstration and further development.