This is an excellent way to think about building a more robust and scalable system. The PHP model you've created is a strong and functional foundation. To transition it into a professional-grade system, the key is to move from a page-by-page approach to a more **modular and data-driven architecture**.

A professional DMS would be built around a core principle of **role-based access control (RBAC)**, which you have already implemented. The next step is to make this control more granular and tie it to a comprehensive set of features for each user type.

Here is a conceptual model of how a professional DMS would be structured, based on the components you've already established.

**Core Architectural Upgrades**

Your current system uses a placeholder.php file to include different pages. A professional model would expand on this by:

* **Modularizing the Code:** Instead of a single placeholder.php, a more professional approach would use a **Model-View-Controller (MVC)** framework (like Laravel or Symfony in the PHP ecosystem). This separates your code into:
  + **Models:** Handle all database interactions (e.g., UserModel, DonationModel).
  + **Views:** Contain the HTML and UI logic (your current PHP files).
  + **Controllers:** Act as the intermediary, fetching data from the Model and passing it to the View.
* **Enhanced Database Schema:** To support the new features, your database would need a few additions.
  + **users table:** Add columns for status (e.g., 'pending', 'active'), last\_login, and a foreign key to a permissions table.
  + **permissions table:** This table would store granular actions like can\_edit\_users, can\_view\_reports, etc., with a link to each role.
  + **donations table:** Add a pledge\_id to link donations to pledges and a recurrence field for managing recurring payments.

**The Professional Dashboard Experience**

Each user role would have a dedicated, intuitive dashboard with features tailored to their specific needs.

**1. Admin Dashboard**

The admin dashboard would serve as the central command center, offering powerful tools for oversight and management. It moves beyond simple lists to provide actionable insights.

* **Executive Summary:** A landing page with a snapshot of real-time KPIs, including total donations, active campaigns, and a count of pending role requests.
* **Comprehensive Financial Reports:** Automated reports that can be filtered by date, campaign, or donor. The reports would include trends, pledge fulfillment rates, and revenue projections.
* **Advanced User Management:** An interface to easily search for, filter, and edit user accounts. Admins could manually change a user's role, deactivate an account, or send direct messages.
* **Campaign Lifecycle Management:** Tools to create new campaigns, set fundraising goals, add descriptions, and track their progress from launch to completion.

**2. Donor Dashboard**

The donor dashboard would provide a personalized experience that encourages continued engagement and transparency.

* **Personal Profile:** A clean view of their personal information, with the ability to update their address, contact details, and communication preferences.
* **Donation History & Receipts:** A searchable, chronological list of all past donations. Each entry would link to a digital receipt that can be downloaded as a PDF for tax purposes.
* **Recurring Donations:** A dedicated section to manage recurring gifts, allowing donors to change the amount, frequency, or payment method without needing to contact an admin.
* **Impact Tracking:** A visual progress bar showing how a campaign is progressing towards its goal and how their individual donation has contributed.

**3. Volunteer Dashboard**

The volunteer dashboard would function as a central hub for all volunteer-related activities, making it easy for them to get involved and track their contributions.

* **Events Calendar:** A dynamic calendar view of all upcoming volunteer opportunities, allowing them to sign up for specific shifts and add them to their personal calendar.
* **Hour Logging:** A simple form to log hours spent volunteering for different tasks or events. These hours would then be submitted for admin approval.
* **Skill Management:** Volunteers can list their specific skills (e.g., photography, grant writing, web design). This would allow the system to send them targeted notifications for tasks that match their expertise.
* **Communication Center:** A messaging board or chat system for volunteers to communicate with event coordinators and each other.

By shifting from your current model to this professional design, you would create a robust, user-friendly, and secure platform that supports your organization's mission and enhances the experience for every user.

Which of these areas are you most interested in developing further?

**Workflow**

**1. User Registration and Role Assignment Flow**

This process controls how a new user enters the system and gets assigned their initial role, with a built-in approval step for higher-privileged roles.

graph TD

A[New User Registration] --> B{Selects Requested Role?};

B -->|Yes| C[User's requested\_role\_id is set];

C --> D[User is assigned default role\_id (e.g., Donor)];

D --> E[User is logged in and redirected to Donor Dashboard];

B -->|No| F[User is assigned default role\_id (e.g., Donor)];

F --> E;

E --> G[Admin Dashboard];

G --> H{Admin reviews new user requests};

H --> I[Admin approves request];

I --> J[User's role\_id is updated in database];

J --> K[User's requested\_role\_id is cleared];

H --> L[Admin rejects request];

L --> K;

**2. Administrator Flow**

The admin journey begins with a secure login and leads to a powerful, centralized dashboard.

**A. Login & Initial Access**

1. Admin logs in with their credentials.
2. login.php authenticates the user and sets $\_SESSION['user\_role'] = 'admin'.
3. placeholder.php checks the session role and loads the admin\_dashboard.php file.

**B. Core Dashboard Functionality**

* **User Management:**
  + Admin navigates to manage\_users.php.
  + The page displays a list of all users.
  + Admin can filter, search, and edit user profiles, manually changing their role or status.
* **Role Request Management:**
  + Admin navigates to the approve\_roles.php page.
  + The page queries the database for users where requested\_role\_id IS NOT NULL.
  + For each request, the admin can click "Approve" or "Reject".
  + The corresponding PHP script updates the user's role\_id to the requested\_role\_id and then sets requested\_role\_id to NULL.
* **Campaign & Event Management:**
  + Admin can create, edit, and close fundraising campaigns.
  + Admin can create and manage events, including setting dates, locations, and volunteer slots.
* **Reporting:**
  + Admin accesses the donations\_report.php to view financial analytics. The report is generated from the donations and users tables, providing insights into total giving, donor retention, and campaign performance.

**3. Donor Flow**

The donor experience is focused on transparency, personalization, and easy access to their giving history.

**A. Login & Dashboard View**

1. Donor logs in.
2. login.php authenticates and sets $\_SESSION['user\_role'] = 'donor'.
3. placeholder.php loads the donor\_dashboard.php view.
4. The donor dashboard displays a summary of their recent activity and total contributions.

**B. Key Actions**

* **Making a Donation:**
  + The donor navigates to add\_donations.php.
  + They fill out a secure form for a one-time or recurring donation.
  + The form submits to a processing script, which adds a new record to the donations table, including the user\_id and campaign information.
* **Viewing Donation History:**
  + The donor clicks on "My Donations" in the navigation bar.
  + The system queries the database for all records in the donations table where user\_id = $\_SESSION['user\_id'].
  + A list of their past donations, including amount, date, and campaign, is displayed. Each item has an option to download a digital receipt.

**4. Volunteer Flow**

The volunteer process streamlines event participation and allows for easy tracking of their contributions.

**A. Login & Dashboard View**

1. Volunteer logs in.
2. login.php authenticates and sets $\_SESSION['user\_role'] = 'volunteer'.
3. placeholder.php loads the volunteer\_dashboard.php view, which features a prominent events calendar.

**B. Key Actions**

* **Signing Up for Events:**
  + The volunteer views the events\_calendar.php and clicks on a specific event.
  + A detailed event page loads with a "Sign Up" button.
  + Clicking the button adds the volunteer's user\_id to the event\_participants table in the database.
* **Submitting Volunteer Hours:**
  + The volunteer navigates to an "Hour Log" page.
  + They select the event and enter the hours they've worked.
  + The form submits to a script that adds a record to an volunteer\_hours table, which includes user\_id, event\_id, and hours\_logged. These records can be flagged for admin approval.

This complete flow provides a robust and scalable model for a professional DMS, addressing the needs of each user role from initial registration to ongoing engagement.