

AI Mode All Short videos Images Videos More ▾



Ask anything



i want to make a wordpress application for club membership.

To create a **WordPress club membership application**, you can build a highly functional system without writing code by **combining a core membership plugin with targeted add-ons**.  MemberPress +1

The process involves selecting your tech stack, configuring tiers, designing registration workflows, and restricting content or club features.  Gravity Forms +2

Step 1: Choose Your Core Membership Plugin

Your choice depends on your budget and whether you require recurring fees or free registration:

- [Paid Memberships Pro](#) **(Best Overall / Free Option)**: Highly flexible, offering an incredibly robust free version. It handles user registrations, paid tiers (via built-in payment gateways), and automatic role assignments seamlessly.  YouTube · IdeaSpot
- [MemberPress](#) **(Best for Paid/Premium Clubs)**: The industry standard for paid memberships, recurring subscriptions, and member management, though it requires a paid license.  WPForms +1
- [Members Plugin](#) **(Best for Simple/Internal Clubs)**: An excellent, completely free option if you just need to manage club directory access, create custom user roles, and restrict content without handling payments.  YouTube · MapilitMedia ... +1

Step 2: Configure Your Membership Tiers

Once your plugin is installed via **Plugins → Add New**:  Memberstack +1

1. Navigate to the plugin settings menu (e.g. **Memberships → Settings** or

Ask anything



resources).  Spaceship.com

Step 3: Build the Registration and Profile Forms

You need a sleek interface for users to sign up and input club-specific data (e.g., phone numbers, addresses, t-shirt sizes):  RegistrationMagic

- **Native Forms:** Tools like *Paid Memberships Pro* generate standard registration, login, and profile editing pages automatically during setup.  YouTube
- **Advanced Forms:** If you require complex custom fields, build your registration form using [Gravity Forms](#) (paired with their User Registration Add-On) or [WPForms](#), and map those fields directly to your membership database.  Gravity Forms +1

Step 4: Restrict Content and Features

Once users have roles assigned, protect your club assets:

- **Pages & Posts:** Use the restriction metabox provided by your membership plugin at the bottom of the block editor to select which tiers can view specific club pages.  Gravity Forms
- **Club Features:** To expand your "application" into a social hub, integrate plugins like [BuddyPress](#) or [ProfileGrid](#) to create member directories, group messaging, and activity feeds.  RegistrationMagic

Summary Checklist to Get Started

Task	Recommended Tool	Cost
Core Membership Engine	Paid Memberships Pro	Free (Paid add-ons available)

Ask anything



Member Directories &
Social

Home and of BuddyPress

Free / Premium Options

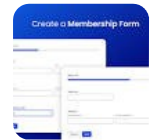
To help me tailor this setup to your exact needs, tell me a bit more about your club:

- Will memberships be **free, paid one-time, or recurring subscriptions**?
- What specific features do members need (e.g., **event calendars, discussion forums, digital member cards**)?
- Do you want to convert this website into a **mobile app** eventually?

 Gravity Forms

How to create a WordPress membership form (template ...

3. Set up your membership plugin (if you haven't done so already) Next, you'll want to set up whichever WordPress membership plugi...



 YouTube · MapilitMedia Inc.

How To Create A Membership Website On WordPress In 2026

so if this is something you're interested in learning how to do then watch this tutorial okay so here I am in a fresh install of W...



 Reddit

Developing a membership facility for WordPress. Options?

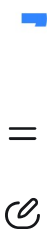
Developing a membership facility for WordPress. Options? My client wants a membership login facility for his WordPress based appli...

Show all

Ask anything



functions.php file.



WordPress natively includes a powerful user management system, metadata structure, and authentication engine. You can leverage these built-in features to manage custom club roles, registration fields, and content restrictions entirely through code.

Architecture: Create a Custom Single-File "Plugin"

Instead of installing a third-party plugin, create your own lightweight script. In your WordPress files, navigate to `/wp-content/plugins/`, create a folder named `club-membership`, and add a file inside named `club-membership.php`. Paste the custom code blocks below into this file.

Step 1: Create Custom Club Roles

WordPress comes with default roles like *Subscriber* and *Editor*. You can programmatically add your own club-specific tiers (like "Standard Member" and "VIP Member") using the `add_role()` function.

php

```
<?php
/*
Plugin Name: Custom Club Membership System
Description: Lightweight custom code to handle roles, registration, and cont
Version: 1.0
Author: Your Name
*/

// Run once when your custom plugin is activated
register_activation_hook( __FILE__, 'club_register_custom_roles' );
```

Ask anything



```
        '
    );

    // Add VIP Member
    add_role(
        'vip_member',
        'VIP Member',
        array(
            'read' => true,
        )
    );
}
```

Use code with caution.

Step 2: Add Custom Fields to the Registration Form

To collect club-specific details (like a phone number or club ID) during registration, hook into the native WordPress registration form actions.

php

```
// 1. Add fields to the registration page
add_action( 'register_form', 'club_registration_fields' );
function club_registration_fields() {
    $phone_number = ( ! empty( $_POST['phone_number'] ) ) ? sanitize_text_fi
    ?>
    <p>
        <label for="phone_number">Phone Number<br />
        <input type="text" name="phone_number" id="phone_number" class="input
    </p>
    <?php
}
```

Ask anything

+



```
$user = new WP_User( $user_id );  
$user->set_role( 'standard_member' );  
}
```

Use code with caution.

Step 3: Restrict Content with a Shortcode

To protect private club content without a visual backend interface, you can write a simple custom shortcode (`[club_gate]`). Wrap any text, link, or video in this shortcode within the [WordPress block editor](#) to hide it from non-members.

php

```
// Restrict content block via shortcode  
add_shortcode( 'club_gate', 'club_restrict_content_shortcode' );  
  
function club_restrict_content_shortcode( $atts, $content = null ) {  
    // Check if the current viewer is logged in and has a member role  
    if ( is_user_logged_in() && ( current_user_can( 'standard_member' ) || c  
        return do_shortcode( $content );  
    }  
  
    // Message shown to unauthorized guests  
    return '<div class="club-restriction-notice"><strong>Members Only:</:  
}
```

Use code with caution.

Usage in the editor: `[club_gate]Secret club details go here...[/club_gate]`

Step 4: Hide Entire Pages Programmatically

Ask anything

+



```
function club_restrict_page_access() {  
    // If viewing the page slug 'member-dashboard'  
    if ( is_page( 'member-dashboard' ) ) {  
  
        // Kick them out if not logged in or not a member  
        if ( ! is_user_logged_in() || ( ! current_user_can( 'standard_member  
            wp_redirect( wp_login_url() );  
            exit;  
        }  
    }  
}
```

Use code with caution.

Non-Plugin Implementation Summary

Architecture Layer	Core Native WordPress Hook / Function	Purpose
User Roles	<code>add_role()</code>	Creates distinct tiers (e.g., standard, VIP) in the database.
UI Forms	<code>add_action('register_form')</code>	Injects custom HTML text fields into the native <code>/wp-login.php?action=register</code> page.
Data Storage	<code>update_user_meta()</code>	Saves custom profile fields cleanly inside the <code>wp_usermeta</code> table.
Access Control	<code>add_shortcode()</code> & <code>template_redirect</code>	Evaluates user permissions on the fly and blocks content or triggers

Ask anything



external API like **Stripe** via custom PHP curl requests).

- Are you comfortable using a custom **child theme** file, or would you prefer writing this entirely as a **custom single-file script**?

=



members pay annual dues. stripe is ok

To accept annual dues using **Stripe without a WordPress plugin**, you will write a custom script that connects your WordPress users to Stripe Billing (Subscriptions) via the official Stripe Checkout API [INDEX1, INDEX2].

This setup utilizes standard WordPress user data paired with Stripe's hosted payment pages, ensuring your site remains fast, completely secure, and entirely plugin-free [INDEX1].

Architecture: How the Flow Works

1. **The Product:** You create an **Annual Subscription Product** directly inside your Stripe Dashboard.
2. **The Checkout:** When a user clicks "Pay Dues" on your site, your code redirects them safely to a **Stripe Checkout Page** [INDEX1, INDEX2].
3. **The Webhook:** Once payment succeeds, Stripe sends a secure signal (**Webhook**) back to your site [INDEX2]. Your script reads this signal, matches it to the user's WordPress profile, and changes their role to your custom member tier [INDEX2].

Ask anything



2. Go to **Product Catalog → Add Product**.

3. Name it `Annual Club Membership` and set the billing period to **Recurring → Yearly**.

4. Save the product and copy the **Price ID** (it looks like `price_1N23abc...`).

Step 2: Route Users to Stripe Checkout

Add this code block to your custom membership plugin file or your theme's `functions.php` file. It generates a secure link to Stripe's payment page [INDEX1, INDEX2].

Note: You do not need the heavy Stripe SDK library; you can make standard WordPress HTTP API calls (`wp_remote_post`).

php

```
// Define your Stripe API Keys (Replace with your actual keys from Stripe Da
define( 'STRIPE_SECRET_KEY', 'sk_live_YOUR_SECRET_KEY' );
define( 'STRIPE_PRICE_ID', 'price_YOUR_PRICE_ID' );

// Shortcode to display a "Pay Dues" button
add_shortcode( 'stripe_pay_button', 'club_render_pay_button' );

function club_render_pay_button() {
    if ( ! is_user_logged_in() ) {
        return '<p>Please <a href="' . wp_login_url() . '">log in or registe
    }

    // If they click the button, trigger the API checkout process
    if ( isset( $_POST['action'] ) && $_POST['action'] === 'club_stripe_chec
        club_initiate_stripe_checkout();
    }

    return '
```

Ask anything

+

```
function club_initiate_stripe_checkout() {
    $current_user = wp_get_current_user();

    // Call Stripe API using native WordPress HTTP framework
    $response = wp_remote_post( 'https://stripe.com', array(
        'headers' => array(
            'Authorization' => 'Bearer ' . STRIPE_SECRET_KEY,
            'Content-Type' => 'application/x-www-form-urlencoded',
        ),
        'body' => http_build_query( array(
            'success_url' => home_url( '/membership-success/' ),
            'cancel_url' => home_url( '/membership-failed/' ),
            'mode' => 'subscription',
            'customer_email' => $current_user->user_email,
            'client_reference_id' => $current_user->ID, // Pass the WordPres
            'line_items[0][price]' => STRIPE_PRICE_ID,
            'line_items[0][quantity]' => 1,
        ) ),
    ) );

    if ( ! is_wp_error( $response ) ) {
        $body = json_decode( wp_remote_retrieve_body( $response ), true );
        if ( isset( $body['url'] ) ) {
            wp_redirect( $body['url'] ); // Redirect to Stripe's hosted secu
            exit;
        }
    }
}
```

Use code with caution.

Step 3: Handle the Webhook (Activate Membership Roles)

When Stripe takes the money every year, it will contact your website via an API

Ask anything

+



```
'methods'          => 'POST',
'callback'          => 'club_handle_stripe_webhook',
'permission_callback' => '__return_true', // Stripe needs public acc
    ) );
} );

function club_handle_stripe_webhook( WP_REST_Request $request ) {
    $payload = $request->get_body();
    $event    = json_decode( $payload, true );

    // Listen specifically for a successful checkout/payment
    if ( $event && $event['type'] === 'checkout.session.completed' ) {
        $session = $event['data']['object'];

        $user_id      = intval( $session['client_reference_id'] );
        $stripe_cust_id = sanitize_text_field( $session['customer'] );
        $stripe_sub_id  = sanitize_text_field( $session['subscription'] );

        if ( $user_id ) {
            // 1. Upgrade their WordPress Role to your custom Member tier
            $user = new WP_User( $user_id );
            $user->set_role( 'standard_member' );

            // 2. Save Stripe data in the user database for tracking renewal
            update_user_meta( $user_id, 'stripe_customer_id', $stripe_cust_id );
            update_user_meta( $user_id, 'stripe_subscription_id', $stripe_sub_id );
            update_user_meta( $user_id, 'membership_status', 'active' );
        }
    }

    return new WP_REST_Response( array( 'status' => 'success' ), 200 );
}
```

Use code with caution.

Ask anything

+



- 4. Under "Select events to listen to", choose `checkout.session.completed` [INDEX2].
- 5. Click **Add endpoint**.

Custom Stripe Integration Summary

System Layer	Implementation Method	Purpose
Product Database	Stripe Dashboard	Stores your subscription product definitions, tracking pricing and annual intervals securely.
Payment Trigger	Shortcode <code>[stripe_pay_button]</code>	Renders a button that maps the logged-in user's details and triggers a redirection API request.
Secure Handshake	<code>wp_remote_post()</code>	Connects securely to Stripe's APIs without needing massive external developer SDK libraries.
Role Activation	REST API Callback (<code>/club/v1/webhook</code>)	Asynchronously updates user capabilities instantly when a recurring invoice clears [INDEX2].

To finalize your setup, let me know:

- What should happen to a user in WordPress if they **cancel their subscription** or their payment fails next year? (e.g., downgrade them back to a basic "Subscriber" role, or block them entirely?)
- Do you need an automatic workflow that **sends an initial welcome email** with

Ask anything

+

yes send an initial welcome email



To send an automated **initial welcome email** right after a new member signs up, you can hook directly into the custom registration function we created earlier.

By using WordPress's native, built-in `wp_mail()` engine, you can deliver customized HTML onboarding instructions containing login credentials, membership details, and links to your private spaces without installing any email marketing plugins.

The Implementation: Adding the Welcome Email Action

Update your custom registration code block (from Step 2 of our second response) by appending this modified hook logic. This intercepts the user creation process and automatically fires the email.

php

```
// Hook into user registration to save data and trigger the onboarding email
add_action( 'user_register', 'club_save_registration_fields_and_email' );

function club_save_registration_fields_and_email( $user_id ) {
    // 1. Process and save the custom field (Phone Number) if present
    if ( ! empty( $_POST['phone_number'] ) ) {
        update_user_meta( $user_id, 'phone_number', sanitize_text_field( $_P
    }

    // 2. Fetch the user details to build the custom onboarding email
    $user = get_userdata( $user_id );
    $user_email = $user->user_email;
    $user_login = $user->user_login;

    // Automatically set their baseline role to standard_member
```

Ask anything



```
<html>
<head>
    <title>Welcome to the Club!</title>
</head>
<body style="font-family: Arial, sans-serif; line-height: 1.6; color: #3
    <div style="max-width: 600px; margin: 0 auto; padding: 20px; border:
        <h2 style="color: #6772e5;">Welcome to Our Club Community!</h2>
        <p>Hello <strong>' . esc_html( $user_login ) . '</strong>,</p>
        <p>Thank you for creating an account on our platform! Your regis

    <div style="background: #f9f9f9; padding: 15px; border-left: 4px
        <p style="margin: 0 0 10px 0;"><strong>Your Login Credential
        <p style="margin: 0;"><strong>Username:</strong> ' . esc_html
        <p style="margin: 5px 0 0 0;"><em>Use the password you selec
    </div>

    <p><strong>⚠ Final Activation Step: Annual Dues</strong></p>
    <p>To fully unlock your member benefits, premium directories, an

    <p style="text-align: center; margin: 30px 0;">
        <a href="" . esc_url( home_url( '/pay-dues/' ) ) . '' style=
    </p>

    <hr style="border: 0; border-top: 1px solid #eee; margin: 20px 0
    <p style="font-size: 12px; color: #777;">If you have any questio
    </div>
</body>
</html>';
```

```
// 4. Set Headers for HTML Delivery
$headers[] = 'Content-Type: text/html; charset=UTF-8';
$headers[] = 'From: Club Administration <admin@yourclubdomain.com>'; //

// 5. Fire the email engine
wp_mail( $to, $subject, $message, $headers );
}
```

Ask anything





Because you are **not using plugins**, your server relies entirely on its internal PHP mail configuration (`wp_mail()`). On modern hosting providers, default PHP mail notifications frequently land in the member's spam folder, or get completely blocked by incoming filters ([Gmail](#), [Yahoo](#), etc.).

How to fix this without a plugin:

Instead of adding an SMTP plugin, configure a lightweight service at the **server system layer**. Providers like [DigitalOcean](#), [Linode](#), or [SiteGround](#) allow you to configure mail-relay tools (like **Postfix** or **ssmtp**) natively inside your server environment panel or SSH CLI using API tokens from professional transactional mail engines (such as **SendGrid** or [Mailgun](#)). This keeps your WordPress footprint entirely code-only while guaranteeing flawless email inbox placement.

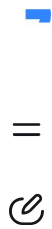
Membership Application Workflow Architecture

Trigger Event	Underlying Programmatic Hook	Resulting Automated Action
User Submits Signup Form	<code>add_action('user_register')</code>	Code maps metadata fields, configures user profiles, and executes custom PHP variables.
Profile Assembly	<code>wp_mail()</code>	Delivers a custom-themed HTML onboarding digest directly to the user's primary mailbox.
User Access Matrix	<code>[stripe_pay_button]</code> Shortcode	Renders inside the landing page linked from the onboarding email, routing them to Stripe Checkout.

Ask anything



renewal date inside their wordpress account layout:



yes to both

To implement both features without plugins, we will update the **Stripe Webhook code** to trigger a success email [INDEX2] and create a **custom shortcode** to display the membership status and renewal date inside the member dashboard.

Step 1: Send the Payment Success Email & Save Renewal Date

We need to update our existing webhook code (from Step 3 of the Stripe response) [INDEX2]. When Stripe successfully charges the user, the script will now calculate the next billing date, save it to the database, and email the member their official confirmation receipt [INDEX2].

Add or overwrite your webhook callback function with this implementation:

php

```
function club_handle_stripe_webhook( WP_REST_Request $request ) {  
    $payload = $request->get_body();  
    $event    = json_decode( $payload, true );  
  
    // Listen specifically for a successful checkout/payment  
    if ( $event && $event['type'] === 'checkout.session.completed' ) {  
        $session = $event['data']['object'];  
    }  
}
```

Ask anything



```
// 2. Calculate the renewal date (Exactly 1 year from today)
$renewal_timestamp = strtotime( '+1 year' );
$renewal_date_formatted = date( 'F j, Y', $renewal_timestamp );

// 3. Save Stripe and subscription data in user metadata
update_user_meta( $user_id, 'stripe_customer_id', $stripe_cust_i
update_user_meta( $user_id, 'stripe_subscription_id', $stripe_su
update_user_meta( $user_id, 'membership_status', 'active' );
update_user_meta( $user_id, 'membership_renewal_date', $renewal_

// 4. Send Payment Success Email
$to      = $user->user_email;
$subject = 'Payment Confirmed! Your Club Membership is Active 🎉

$message = '
<html>
<body style="font-family: Arial, sans-serif; line-height: 1.6; c
    <div style="max-width: 600px; margin: 0 auto; padding: 20px;
        <h2 style="color: #28a745;">Payment Received!</h2>
        <p>Hello <strong>' . esc_html( $user->user_login ) . '</
        <p>Great news! We have successfully processed your annua
        <p>You now have unrestricted access to all private club

        <div style="background: #f4fdf7; padding: 15px; border-1
            <p style="margin: 0;"><strong>Membership Status:</st
            <p style="margin: 5px 0 0 0;"><strong>Next Automatic
        </div>

        <p style="text-align: center; margin: 30px 0;">
            <a href="" . esc_url( home_url( '/member-dashboard/'
        </p>
        <hr style="border: 0; border-top: 1px solid #eee; margin
        <p style="font-size: 12px; color: #777;">Thank you for y
    </div>
</body>
```

Ask anything



```
,  
  
    return new WP_REST_Response( array( 'status' => 'success' ), 200 );  
}
```

Use code with caution.

Step 2: Code the Dashboard Account Layout Shortcode

To show members their renewal date dynamically, create a shortcode named `[member_dashboard]`. You can place this shortcode directly inside a standard block editor Page (e.g., a page titled `/member-dashboard/`).

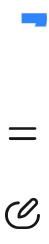
php

```
// Shortcode to display dynamic membership dashboard metrics  
add_shortcode( 'member_dashboard', 'club_render_member_dashboard' );  
  
function club_render_member_dashboard() {  
    if ( ! is_user_logged_in() ) {  
        return '<p>Please <a href="' . wp_login_url() . '">log in</a> to vie  
    }  
  
    $user_id      = get_current_user_id();  
    $user         = wp_get_current_user();  
    $status       = get_user_meta( $user_id, 'membership_status', true );  
    $renewal_time = get_user_meta( $user_id, 'membership_renewal_date', true  
  
    // Default template architecture for non-paying users  
    if ( empty( $status ) || $status !== 'active' ) {  
        return '  
        <div style="padding: 20px; background: #fff3cd; border: 1px solid #f
```

Ask anything

+





```
// UI display for verified active premium members
return '
<div style="background: #ffffff; border: 1px solid #e3e8ee; padding: 25px
  <h3 style="margin-top:0; color:#32325d;">Welcome back, ' . esc_html(
  <p>You are logged into your private club access portal.</p>

  <table style="width:100%; border-collapse: collapse; margin-top: 15px
    <tr style="border-bottom: 1px solid #e3e8ee;">
      <td style="padding: 10px 0; font-weight: bold; color: #6b7c9
      <td style="padding: 10px 0; text-align: right; font-weight:
    </tr>
    <tr style="border-bottom: 1px solid #e3e8ee;">
      <td style="padding: 10px 0; font-weight: bold; color: #6b7c9
      <td style="padding: 10px 0; text-align: right;"><span style=
    </tr>
    <tr>
      <td style="padding: 10px 0; font-weight: bold; color: #6b7c9
      <td style="padding: 10px 0; text-align: right; font-weight:
    </tr>
  </table>
</div>';
}
```

Use code with caution.

Dashboard & Notification Flow Summary

Operational Phase	Programmatic Event Engine	User-Facing Action / Output
Instant	Stripe API Payload	Updates internal WordPress

Ask anything



Dynamic Profile Display

Custom shortcode
[member_dashboard]

Displays personalized data fields, account roles, and renewal details inside standard theme templates.



With the membership metrics successfully saved and displayed, would you like to:

- Build an additional feature allowing users to **cancel their subscription** directly from this dashboard page by making an outbound API deletion call to Stripe?
- Restrict specific menu tabs or hide specific sidebar navigational elements depending on whether their layout status reads **active** or **unpaid**?

restrict on status

To restrict sections of your website based on a member's payment status, you can write two custom functions. The first function creates a **content wrapper shortcode** ([member_only]) for pages and posts, while the second function programmatically **hides navigation links** from your theme's main menu if a user is unpaid or logged out.

Step 1: Create a Content Gate Shortcode

Add this shortcode to your custom membership plugin file. It checks the user's `membership_status` metadata field that Stripe updates. If the status isn't `active`, the content hidden inside the shortcode will be replaced with a lock notice.

Ask anything



```
// 1. Kick out completely logged-out visitors
if ( ! is_user_logged_in() ) {
    return '<div style="padding:15px; background:#f8d7da; border:1px solid #f8d7da; border-radius:5px; text-align:center;">
        <strong>🔒 Members Only:</strong> Please <a href="' . wp_login_url() . '>log in</a>
    </div>';
}

$user_id = get_current_user_id();
$status = get_user_meta( $user_id, 'membership_status', true );

// 2. Allow administrators to see everything, and verify paying members
if ( current_user_can( 'administrator' ) || $status === 'active' ) {
    return do_shortcode( $content );
}

// 3. Fallback message shown to registered users who have not paid their dues
return '<div style="padding:15px; background:#fff3cd; border:1px solid #ffee58; border-radius:5px; text-align:center;">
    <strong>🔒 Access Restricted:</strong> This section is reserved for active members.
</div>';
}
```

Use code with caution.

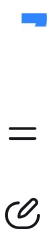
Usage in your block editor: `[member_only]` This secret text or file download is only visible to active payers. `[/member_only]`

Step 2: Dynamically Hide Theme Navigation Menu Items

You can cleanly filter your theme's custom menu links on the fly using WordPress's native `wp_nav_menu_objects` hook. If an item is labeled "Member Portal" or "Club Directory," it will disappear entirely from the navigation bar unless the visitor is an active, paying member.

Ask anything





```
$is_active_member = ( is_user_logged_in() && ( $status === 'active' || c

// Define the exact titles of the menu items you want to restrict
$restricted_menu_titles = array( 'Member Portal', 'Club Directory', 'Mem

foreach ( $sorted_menu_items as $key => $item ) {
    // If the item title matches a restricted name and they aren't paid
    if ( in_array( $item->title, $restricted_menu_titles ) && ! $is_acti
        unset( $sorted_menu_items[$key] );
    }
}

return $sorted_menu_items;
}
```

Use code with caution.

Security Execution Matrix

Access Target	Native Hook / Execution	Unpaid / Guest Behavior	Paid Active Behavior
In-Page Elements	<code>add_shortcode('member_only')</code>	Strips target block content and injects custom warning banners.	Compiles and renders all text, forms, or downloads seamlessly.
Main Navigation	<code>add_filter('wp_nav_menu_objects')</code>	Completely unsets target links from array so they never	Leaves menu tree structurally complete for

Ask anything

+



stripe payment
gateways.

theme
template
pipelines.



Now that your entire frontend site restricts itself automatically based on payment status, let me know:

- Do you want to build the **Stripe subscription cancellation button** directly into their profile dashboard page so members can manage their own billing status?
- Should we add a system that automatically downgrades their status to **"expired"** if Stripe fires a webhook saying a payment failed or a card was declined?

i want the dues amount based on role

To scale your annual dues based on different custom roles (for example, charging a **Standard Member** less than a **VIP Member**), we will modify the shortcode to check the user's assigned WordPress role before generating the checkout session.

This lets you use a single "Pay Dues" button that dynamically routes the user to the correct **Stripe Price ID** based on their club tier [INDEX1, INDEX2].

Step 1: Create Your Tiered Products in Stripe

1. Log into your **Stripe Dashboard**.

Ask anything



Step 2. Update Your Custom Script Configuration

We will map your WordPress roles directly to your corresponding Stripe Price IDs. Replace your existing button shortcode function with this dynamic mapping system:

php

```
// Define your Stripe Secret Key
define( 'STRIPE_SECRET_KEY', 'sk_live_YOUR_SECRET_KEY' );

// Shortcode to display a dynamic "Pay Dues" button matching their role
add_shortcode( 'stripe_pay_button', 'club_render_role_based_pay_button' );

function club_render_role_based_pay_button() {
    if ( ! is_user_logged_in() ) {
        return '<p>Please <a href="' . wp_login_url() . '">log in or registe
    }

    // Trigger the API checkout process if the button is clicked
    if ( isset( $_POST['action'] ) && $_POST['action'] === 'club_stripe_chec
        club_initiate_role_based_stripe_checkout();
    }

    // Get the current logged-in user to see what tier they belong to
    $current_user = wp_get_current_user();
    $button_text = 'Pay Annual Dues';

    // Tailor the text overlay on the button to match their group tier
    if ( in_array( 'vip_member', $current_user->roles ) ) {
        $button_text = 'Pay VIP Annual Dues';
    } elseif ( in_array( 'standard_member', $current_user->roles ) ) {
        $button_text = 'Pay Standard Annual Dues';
    }

    return '

```

Ask anything

+



```
function club_initiate_role_based_stripe_checkout() {
    $current_user = wp_get_current_user();

    // Default fallback Price ID
    $stripe_price_id = 'price_STANDARD_PRICE_ID_HERE';

    // Programmatically swap the Stripe Price ID depending on their active w
    if ( in_array( 'vip_member', $current_user->roles ) ) {
        $stripe_price_id = 'price_VIP_PRICE_ID_HERE'; // Replace with your l
    } elseif ( in_array( 'standard_member', $current_user->roles ) ) {
        $stripe_price_id = 'price_STANDARD_PRICE_ID_HERE'; // Replace with y
    }

    // Fire the secure HTTP API call directly to Stripe Checkout
    $response = wp_remote_post( 'https://stripe.com', array(
        'headers' => array(
            'Authorization' => 'Bearer ' . STRIPE_SECRET_KEY,
            'Content-Type' => 'application/x-www-form-urlencoded',
        ),
        'body' => http_build_query( array(
            'success_url' => home_url( '/membership-success/' ),
            'cancel_url' => home_url( '/membership-failed/' ),
            'mode' => 'subscription',
            'customer_email' => $current_user->user_email,
            'client_reference_id' => $current_user->ID,
            'line_items[price]' => $stripe_price_id,
            'line_items[quantity]' => 1,
        ) ),
    ) );

    if ( ! is_wp_error( $response ) ) {
        $body = json_decode( wp_remote_retrieve_body( $response ), true );
        if ( isset( $body['url'] ) ) {
            wp_redirect( $body['url'] );
            exit;
        }
    }
}
```

Ask anything



Step 3. Webhook Verification

Because we pass the WordPress User ID inside the `client_reference_id` property, our existing webhook structure remains completely compatible [INDEX2]. When Stripe fires the successful payment webhook back to your site, it checks the metadata, updates the `membership_status` database flag to `active`, and schedules their renewal date without overriding their custom assigned role [INDEX2].

Tiered Membership Mapping Summary

WordPress Club Role	Evaluated Stripe Price ID	Checkout Interface Style	System Database Action
Standard Member	<code>price_STANDARD_PRICE_ID</code>	Displays "Pay Standard Annual Dues" on submission.	Activates the core <code>membership_status</code> flag to <code>active</code> .
VIP Member	<code>price_VIP_PRICE_ID</code>	Displays "Pay VIP Annual Dues" on submission.	Activates premium status and unlocks high-tier gated shortcodes.
Custom Tiers	Custom mapped mapping variables	Dynamically adapts button texts matching custom fields.	Saves the relative checkout transaction values seamlessly [INDEX2].

Now that dues map dynamically to roles, let me know:

- Do users choose their preferred membership role **during the initial registration**

Ask anything



the annual dues cycle is the same for everyone, jan 1 to dec 31

=



To enforce a uniform **calendar-year dues cycle (January 1 to December 31) for everyone**, we need to calculate a **prorated amount** for members who join mid-year.

Stripe Checkout does not natively calculate prorations for fixed calendar cycles on the fly, so your custom WordPress code must calculate the remaining days in the current year, compute the prorated cost, and pass a dynamic line item price directly to Stripe.

When **January 1** arrives, Stripe will automatically bill them the full annual rate for their respective role going forward.

Step 1: Set Up Your Products in Stripe

Instead of using standard recurring Price IDs, you will configure your products with a **Base Price** in your Stripe Dashboard Product Catalog (e.g., Standard at \$100/year, VIP at \$300/year). Your code will dynamically generate a custom invoice amount for their first partial year based on when they click the button.

Step 2: Dynamic Calendar-Year Proration Script

Replace your previous checkout function with this math framework. It handles the calendar logic completely without any third-party plugins.

php

```
define( 'STRIPE_SECRET_KEY', 'sk_live_YOUR_SECRET_KEY' );
```

Ask anything



‘

+

```
// Generate a dynamic line item for the remainder of the calendar
'line_items[0][price_data][currency]'      => 'usd',
'line_items[0][price_data][product_data][name]' => 'Club Dues Re
'line_items[0][price_data][unit_amount]'    => $prorated_price,
'line_items[0][quantity]'                  => 1,
    ) ),
    ) );

if ( ! is_wp_error( $response ) ) {
    $body = json_decode( wp_remote_retrieve_body( $response ), true );
    if ( isset( $body['url'] ) ) {
        wp_redirect( $body['url'] );
        exit;
    }
}
```

Use code with caution.

Step 3: Webhook Alignment for Fixed Expirations

Because everyone's membership expires on exactly December 31, update your Webhook function logic to hardcode the expiration timestamp to the last second of the current year rather than using `+1 year`.

php

```
// Inside your club_handle_stripe_webhook() function, update the timestamp 1
$current_year = date( 'Y' );
$dec_31_timestamp = strtotime( "December 31, $current_year 23:59:59" );

// Save the fixed expiration calendar matrix down to their profile
update_user_meta( $user_id, 'membership_status', 'active' );
update_user_meta( $user_id, 'membership_renewal_date', $dec_31_timestamp );
```

Ask anything

+



To collect the next year's full dues starting January 1 without manual intervention, you can write a simple custom **cron script** that runs once daily. On January 1, it checks for active users, flags their account status as `unpaid`, and locks down their portal access shortcodes until they press the checkout button again for the new year.

php

```
// Schedule a native background cleanup routine to run daily
if ( ! wp_next_scheduled( 'club_daily_expiry_check' ) ) {
    wp_schedule_event( time(), 'daily', 'club_daily_expiry_check' );
}

add_action( 'club_daily_expiry_check', 'club_execute_calendar_rollover' );

function club_execute_calendar_rollover() {
    $current_time = current_time( 'timestamp' );

    // Find all users who are currently marked as active
    $members = get_users( array(
        'meta_key'    => 'membership_status',
        'meta_value' => 'active'
    ) );

    foreach ( $members as $member ) {
        $expiration = get_user_meta( $member->ID, 'membership_renewal_date',

        // If the calendar year crossed midnight on Dec 31, automatically re
        if ( ! empty( $expiration ) && $current_time > intval( $expiration )
            update_user_meta( $member->ID, 'membership_status', 'unpaid' );
        }
    }
}
```

Ask anything

+



	Scenario Matrix	System Mathematical Formula	Resulting User Experience
■			
=	Joining January 1	days	Charged 100% of the selected role's dues; profile set to expire Dec 31.
🔗	Joining July 1	days	Charged roughly 50% of full annual fees for the exact days remaining.
	Crossing Jan 1 Midnight	Current Time > Dec 31 Expiration	System cron runs background sweep, marking metadata as <code>unpaid</code> and locking navigation.

no need to prorate. joining at anytime in the year is the full annual

To enforce a **flat annual fee regardless of when a member joins** within the calendar year, we can remove the complex proration mathematics. Whether someone registers on January 2nd or October 22nd [INDEX2], they will pay the full annual fee for their role, and their membership will still strictly expire on **December 31st** of that current year.

Step 1: Set Up Recurring Products in Stripe

Because everyone pays a fixed amount, we can go back to using Stripe's native **Recurring Subscription** model [INDEX1, INDEX2].

Ask anything



Step 2: Route Users to the Full Fee Checkout

We will update our button script to pull the exact fixed Price ID based on the user's role. It passes this clean data straight to Stripe Checkout [INDEX1, INDEX2].

php

```
define( 'STRIPE_SECRET_KEY', 'sk_live_YOUR_SECRET_KEY' );

function club_initiate_fixed_calendar_checkout() {
    $current_user = wp_get_current_user();

    // Map your custom WordPress roles directly to your fixed Stripe Price I
    $stripe_price_id = 'price_STANDARD_FALLBACK_ID';

    if ( in_array( 'vip_member', $current_user->roles ) ) {
        $stripe_price_id = 'price_VIP_PRICE_ID_HERE';
    } elseif ( in_array( 'standard_member', $current_user->roles ) ) {
        $stripe_price_id = 'price_STANDARD_PRICE_ID_HERE';
    }

    // Launch Stripe Checkout with the full annual price
    $response = wp_remote_post( 'https://stripe.com', array(
        'headers' => array(
            'Authorization' => 'Bearer ' . STRIPE_SECRET_KEY,
            'Content-Type'  => 'application/x-www-form-urlencoded',
        ),
        'body' => http_build_query( array(
            'success_url'      => home_url( '/membership-success/' ),
            'cancel_url'      => home_url( '/membership-failed/' ),
            'mode'             => 'subscription', // Core subscription t
            'customer_email'  => $current_user->user_email,
            'client_reference_id' => $current_user->ID,
            'line_items[price]' => $stripe_price_id,
            'line_items[quantity]' => 1,
        )
    )
);
```

Ask anything

+



```
}
}
}
```

Use code with caution.

Step 3: Set the Hard Expiration on the Webhook Handshake

When Stripe confirms a successful checkout, your webhook will immediately calculate the absolute end of the *current* calendar year and save that timestamp to the user's database profile [INDEX2].

php

```
// Inside your webhook processing callback:
if ( $event && $event['type'] === 'checkout.session.completed' ) {
    $session = $event['data']['object'];
    $user_id = intval( $session['client_reference_id'] );

    if ( $user_id ) {
        $user = new WP_User( $user_id );
        $user->set_role( 'standard_member' );

        // Hard set expiration to December 31st of the current year at midni
        $current_year = date( 'Y' );
        $dec_31_timestamp = strtotime( "December 31, $current_year 23:59:59"

        // Save metadata fields down to WordPress
        update_user_meta( $user_id, 'membership_status', 'active' );
        update_user_meta( $user_id, 'membership_renewal_date', $dec_31_times
        update_user_meta( $user_id, 'stripe_subscription_id', sanitize_text_

    }
}
```

Ask anything

+



on **December 31st**, it identifies all expired profiles, marks their `membership_status` metadata as `unpaid` , and locks them out of your menu structures and page content until they pay the full dues for the brand-new year.

=



Fixed Calendar Billing Summary Matrix

Signup Window	Fee Structure Applied	Access Duration Granted	System Expiration Lock
January 1st	100% Full Annual Rate	12 full months	December 31st (Midnight)
Mid-Year Signups	100% Full Annual Rate	Remaining months left in year	December 31st (Midnight)
New Year Day (Jan 1)	100% Full Annual Rate	Profile status flips to <code>unpaid</code>	Portal automatically gated until new payment

Now that the simplified flat-fee calendar cycle is locked down, let me know:

- Would you like to create an automated template that **emails a reminder notice** to all members on December 15th warning them that their access expires on December 31st?
- Do you need assistance creating a custom **registration layout page template** to securely capture their initial profile sign-ups before they click the pay button?

Ask anything



Step 1: Schedule the December 15th Expiration Reminder Email

We can tap into our existing daily background cron engine

(`club_daily_expiry_check`) that we built earlier. The script will look at the current date every day. When it detects that the date is **December 15th**, it will fetch all active members and send them a beautifully formatted HTML warning email.

Add this logic directly into your custom plugin script file:

php

```
// Hook into our daily cron operation to check for the renewal reminder date
add_action( 'club_daily_expiry_check', 'club_send_december_membership_remind
```

```
function club_send_december_membership_reminders() {
```

```
    // Check if today is exactly December 15th
```

```
    if ( date( 'm-d' ) === '12-15' ) {
```

```
        // Fetch all users currently flagged as active paying members
```

```
        $active_members = get_users( array(
```

```
            'meta_key'    => 'membership_status',
```

```
            'meta_value' => 'active'
```

```
        ) );
```

```
        foreach ( $active_members as $member ) {
```

```
            $to      = $member->user_email;
```

```
            $subject = '📅 Upcoming Club Membership Renewal Notice';
```

```
            $message = '
```

```
            <html>
```

```
            <body style="font-family: Arial, sans-serif; line-height: 1.6; c
```

```
                <div style="max-width: 600px; margin: 0 auto; padding: 20px;
```

```
                <h3 style="color: #d9534f;">Membership Expiration Notice
```

Ask anything

+





```
<p>To ensure your access to the member directories, disc

<p style="text-align: center; margin: 30px 0;">
    <a href="" . esc_url( home_url( '/member-dashboard/'
</p>
<hr style="border: 0; border-top: 1px solid #eee; margin
<p style="font-size: 12px; color: #777;">Thank you for a
</div>
</body>
</html>';

$headers[] = 'Content-Type: text/html; charset=UTF-8';
$headers[] = 'From: Club Administration <admin@yourclubdomain.co

wp_mail( $to, $subject, $message, $headers );

    }
}
}
```

Use code with caution.

Step 2: Build the Frontend Registration Form Page Layout

Instead of sending prospective members to the default, ugly WordPress `/wp-login.php?action=register` backend screen, we will create a **custom page shortcode** named `[club_registration_form]`.

You can place this shortcode on any standard frontend WordPress page (e.g., a page with the URL slug `/join-the-club/`). This handles form design, data sanitization, input error handling, and security validation entirely via code.

Ask anything



```
return '<p>You are already registered! Head over to your <a href=""
}

$errors = array();

// Process form data when submitted
if ( isset( $_POST['club_submit_registration'] ) ) {
    // Verify WordPress security token (Nonce verification)
    if ( ! isset( $_POST['club_reg_nonce'] ) || ! wp_verify_nonce( $_POS
        $errors[] = 'Security check failed. Please refresh and try again
    } else {
        $username      = sanitize_user( $_POST['club_username'] );
        $email          = sanitize_email( $_POST['club_email'] );
        $password       = $_POST['club_password'];
        $phone          = sanitize_text_field( $_POST['club_phone'] );
        $chosen_role    = sanitize_text_field( $_POST['club_role'] );

        // Validate Form Inputs
        if ( empty( $username ) || empty( $email ) || empty( $password ) )
            $errors[] = 'Please fill out all required fields (Username,
        }
        if ( username_exists( $username ) ) {
            $errors[] = 'That username is already taken.';
        }
        if ( ! is_email( $email ) ) {
            $errors[] = 'Please enter a valid email address.';
        }
        if ( email_exists( $email ) ) {
            $errors[] = 'That email address is already registered.';
        }
        if ( strlen( $password ) < 8 ) {
            $errors[] = 'Password must be at least 8 characters long.';
        }
        // Enforce registration selection constraints
        if ( ! in_array( $chosen_role, array( 'standard_member', 'vip_me
            $chosen_role = 'standard_member';
        }
    }
}
```

Ask anything



```
// Update user fields
$user = new WP_User( $user_id );
$user->set_role( $chosen_role );

// Save custom metadata (Phone Number)
update_user_meta( $user_id, 'phone_number', $phone );
update_user_meta( $user_id, 'membership_status', 'unpaid' );

// Log the user into WordPress automatically
$creds = array(
    'user_login'    => $username,
    'user_password' => $password,
    'remember'      => true
);
wp_signon( $creds, false );

// Send them straight to the payment area
wp_redirect( home_url( '/pay-dues/' ) );
exit;
} else {
    $errors[] = $user_id->get_error_message();
}
}
}

// Build the Form UI Output
$output = '<div class="club-registration-card" style="max-width: 450px;
$output .= '<h3 style="margin-top:0; color:#32325d; text-align:center;">

// Output visual error message blocks if found
if ( ! empty( $errors ) ) {
    $output .= '<div style="background:#f8d7da; color:#721c24; padding:1
    foreach ( $errors as $error ) {
        $output .= '<p style="margin:4px 0;">✖ ' . esc_html( $error )
    }
}
```

Ask anything

+





```
$output .= '<div style="margin-bottom:15px;"><label style="display:block
$output .= '<div style="margin-bottom:15px;"><label style="display:block
$output .= '<div style="margin-bottom:15px;"><label style="display:block

$output .= '<div style="margin-bottom:25px;"><label style="display:block
$output .= '<select name="club_role" style="width:100%; padding:10px; bo
$output .= '<option value="standard_member">Standard Member Tier</option
$output .= '<option value="vip_member">VIP Member Tier</option>';
$output .= '</select></div>';

$output .= '<button type="submit" name="club_submit_registration" style=
$output .= '</form>';
$output .= '</div>';

return $output;
}
```

Use code with caution.

Core System Execution Blocks Conclusion

Functional

Target	Trigger Action Engine	System Execution Outcome
Registration Form	[club_registration_form]	Validates input security, dynamically updates wp_users , and routes users straight to the Stripe payout layer.
December 15th Ping	Daily Scheduled Cron Hook	Scans the user meta matrix table and safely builds custom transactional email warnings.

Ask anything

+



Your completely custom, plugin-free membership site architecture is fully structured!
To complete the workflow, let me know:

- Do you want to build a **Stripe subscription cancellation button** directly onto the user's account dashboard panel?
- Do you need assistance mapping out custom **CSS stylesheet classes** to change the colors of these registration and payment form elements?

i need an administrative form to review and edit member information

To allow you or your club managers to review and update member profiles from the front end of your site without accessing the [WordPress admin dashboard](#), you can build a secure **Administrative Member Directory and Editing Panel** using a custom shortcode.

This script creates an **admin-only portal** where you can view a list of all registered members, filter by their membership status, and click an "Edit" button to modify their name, phone number, role, or payment status on the fly.

🛡️ Step 1: Code the Admin Dashboard Shortcode

Add this complete code block to your custom plugin script file. It handles the list view, secure form submissions, database updates, and permission gatekeeping (ensuring **only Administrators** can access the data).

Ask anything



```

if ( ! current_user_can( 'administrator' ) ) {
    return '<div style="padding:15px; background:#f8d7da; color:#721c24;
}

$output = '<div class="club-admin-panel" style="max-width:900px; margin:

// 2. Handle Processing Updates when the edit form is saved
if ( isset( $_POST['club_admin_update_member'] ) ) {
    if ( ! isset( $_POST['club_admin_nonce'] ) || ! wp_verify_nonce( $_P
        $output .= '<div style="background:#f8d7da; padding:10px; margin
    } else {
        $target_user_id = intval( $_POST['target_user_id'] );
        $new_phone      = sanitize_text_field( $_POST['member_phone'] );
        $new_status     = sanitize_text_field( $_POST['member_status'] );
        $new_role       = sanitize_text_field( $_POST['member_role'] );

        // Save text data to database fields
        update_user_meta( $target_user_id, 'phone_number', $new_phone );
        update_user_meta( $target_user_id, 'membership_status', $new_sta

        // Update user role if changed
        $target_user = new WP_User( $target_user_id );
        if ( ! in_array( $new_role, $target_user->roles ) && in_array( $
            $target_user->set_role( $new_role );
        }

        $output .= '<div style="background:#d4edda; color:#155724; paddi
    }
}

// 3. VIEW MODE B: Single Member Detailed Editing Layout
if ( isset( $_GET['edit_member'] ) ) {
    $edit_id = intval( $_GET['edit_member'] );
    $user_data = get_userdata( $edit_id );

    if ( ! $user_data ) {

```

Ask anything



```
$output .= '<p><a href="' . esc_url( remove_query_arg('edit_member'))

$output .= '<form method="POST" style="background:#fff; border:1px s
$output .= wp_nonce_field( 'club_admin_edit_action', 'club_admin_non
$output .= '<input type="hidden" name="target_user_id" value="' . $e

$output .= '<div style="margin-bottom:15px;"><label style="display:b
$output .= '<div style="margin-bottom:15px;"><label style="display:b

// Status Field Selector
$output .= '<div style="margin-bottom:15px;"><label style="display:b
$output .= '<select name="member_status" style="width:100%; padding:
$output .= '<option value="active" ' . selected($status, 'active', f
$output .= '<option value="unpaid" ' . selected($status, 'unpaid', f
$output .= '<option value="expired" ' . selected($status, 'expired',
$output .= '</select></div>';

// Role Field Selector
$output .= '<div style="margin-bottom:20px;"><label style="display:b
$output .= '<select name="member_role" style="width:100%; padding:10
$output .= '<option value="standard_member" ' . selected($current_ro
$output .= '<option value="vip_member" ' . selected($current_role, '
$output .= '<option value="subscriber" ' . selected($current_role, '
$output .= '</select></div>';

$output .= '<button type="submit" name="club_admin_update_member" st
$output .= '</form>';
```

```
} else {
    // 4. VIEW MODE A: Broad Directory Database List View Table
    $output .= '<h2 style="color:#32325d; margin-bottom:20px;"> 📋 Club I

    // Query users across our system components
    $all_members = get_users( array( 'orderby' => 'registered', 'order'

    $output .= '<table style="width:100%; border-collapse:collapse; back
```

Ask anything



```
foreach ( $all_members as $member ) {
    $member_status = get_user_meta( $member->ID, 'membership_status'
    $display_status = !empty($member_status) ? ucfirst($member_statu
    $status_color = ($member_status === 'active') ? '#00875a' : '#d9
    $role_display = !empty($member->roles) ? ucwords(str_replace('_'

    // Generate row data points
    $output .= '<tr style="border-bottom:1px solid #e3e8ee;">';
    $output .= '<td style="padding:12px; font-weight:bold; color:#32
    $output .= '<td style="padding:12px; color:#6b7c93;">' . esc_html
    $output .= '<td style="padding:12px;">' . esc_html($role_display
    $output .= '<td style="padding:12px; font-weight:bold; color:' .

    // Direct Link to switch into detailed editing view mode
    $edit_url = add_query_arg( 'edit_member', $member->ID );
    $output .= '<td style="padding:12px; text-align:center;"><a href
    $output .= '</tr>';
}

$output .= '</table>';
}

$output .= '</div>';
return $output;
}
```

Use code with caution.

Step 2: Display the Panel Natively

1. Log into your WordPress admin backend panel.
2. Go to **Pages → Add New**

Ask anything



member or anonymous browser randomly guesses this page URL, they will simply see the secure "Access Denied" error message, leaving your backend profile data entirely uncompromised.



Admin Operations Functional Table

Console UI Layer	Internal Action Trigger	Functional Execution Target	Security Veri
Directory Board	Default shortcode compile	Renders comprehensive registry arrays of usernames, emails, and active roles.	Checked via <code>current_user</code>
User Query Switch	<code>\$_GET['edit_member']</code>	Hydrates custom text input boxes with live <code>wp_usermeta</code> data strings.	Token identity
Database Commit	<code>\$_POST['club_admin_update_member']</code>	Runs sanitization filters, parses structures, and calls <code>update_user_meta()</code> .	Validated via fields

Now that you can review and update all your members directly from the front end, let me know:

- Do you want to add an option to the list view that allows you to **export the entire membership list to a CSV file** for spreadsheets?

Ask anything

+



impose probationary condition on new member

To implement a **probationary period** for new members without using plugins, we will modify the frontend registration form and the content gating functions.

New registrations will automatically receive a `membership_status` of **probationary** [INDEX2]. They will remain under this restriction for a fixed timeframe (e.g., 90 days), after which our background cron system will automatically elevate them to full `active` status.

Step 1: Update the Frontend Registration Form logic

We need to update the form handler (`[club_registration_form]`) to automatically calculate the end date of their probation period (e.g., **90 days from today**) and store it in their database profile metadata [INDEX2].

Replace the processing code block inside your registration script where `wp_create_user` is called:

php

```
// Insert this block inside the existing [club_registration_form] function u
if ( ! is_wp_error( $user_id ) ) {
    $user = new WP_User( $user_id );
    $user->set_role( $chosen_role );

    // Calculate Probation Window (90 Days from the moment of signup)
    $probation_length = 90 * DAY_IN_SECONDS;
    $probation_end_timestamp = current_time( 'timestamp' ) + $probation_leng

    // Save custom metadata down to the user's database entry
```

Ask anything

+



```
wp_redirect( home_url( '/pay-dues/' ) );
exit;
}
```

Use code with caution.

Step 2: Update the Gated Content Shortcode

We will update our content protection shortcode ([member_only]) to handle probationary members. This allows you to hide certain sensitive areas (like voting pages, board directories, or premium downloads) from probationary members even if they have already paid their dues.

php

```
function club_restrict_by_payment_status( $atts, $content = null ) {
    if ( ! is_user_logged_in() ) {
        return '<div style="padding:15px; background:#f8d7da; color:#721c24;
    }

    $user_id = get_current_user_id();
    $status = get_user_meta( $user_id, 'membership_status', true );

    if ( current_user_can( 'administrator' ) || $status === 'active' ) {
        return do_shortcode( $content );
    }

    // Explicit notice displayed to paid users who are still inside their pr
    if ( $status === 'probationary' ) {
        $end_date = get_user_meta( $user_id, 'probation_end_date', true );
        $formatted_end_date = date( 'F j, Y', intval( $end_date ) );

        return '<div style="padding:15px; background:#e2e3e5; border:1px sol
```

Ask anything

+



Use code with caution.

Step 3: Automate the Probation Graduation via Cron

To ensure you do not have to lift a finger to upgrade users, expand your daily background cron function (`club_daily_expiry_check`) to check for expired probation dates and elevate those users automatically [INDEX2].

php

```
// Append this processing logic directly inside your existing daily cron fun
add_action( 'club_daily_expiry_check', 'club_check_probation_maturation' );

function club_check_probation_maturation() {
    $current_time = current_time( 'timestamp' );

    // Find all users locked into probationary status
    $probationary_users = get_users( array(
        'meta_key'    => 'membership_status',
        'meta_value' => 'probationary'
    ) );

    foreach ( $probationary_users as $user ) {
        $probation_end = get_user_meta( $user->ID, 'probation_end_date', true );

        // If their probation timeline has passed, instantly migrate them to
        if ( ! empty( $probation_end ) && $current_time >= intval( $probatio
            update_user_meta( $user->ID, 'membership_status', 'active' );

        // Optional: Send a congratulatory email confirming full members
        $to      = $user->user_email;
        $subject = '🎉 Congratulations! Your Full Club Membership is Ac
        $message = '<p>Hello ' . esc_html( $user->display_name ) . ',</p><
        $headers[] = 'Content-Type: text/html; charset=UTF-8';
```

Ask anything

+





Step 4: Include Probation Fields on the Admin Form

Update the single member editing section of your **Administrative Form** (`[club_admin_console]`) so that you can view their probation expiration timestamp or manually clear a member from probation early.

php

// Add this drop-down option inside your administrative editor select menu f
\$output .= '<option value="probationary" ' . selected(\$status, 'probationary

Use code with caution.

Probation Lifecycle Matrix

Operational Phase	Applied Metadata Value	Frontend Capabilities	Automation Trigger
New Signups	<code>status = probationary</code>	Displays customized warning bars; hides premium modules.	Code automatically appends target timestamps during form completion.
Admin Inspection	Viewable via admin dashboard list	Administrators can override, extend, or clear the probation status early via standard form dropdowns.	Executed securely using custom REST/ POST database fields.
Maturation Target	Flips instantly to <code>active</code>	Restores absolute page capabilities and removes	Daily server backaround cron

Ask anything

+



- Should the 90-day probation period start **immediately when they register**, or only **after Stripe successfully processes their payment**?
- Do you want to restrict specific pages entirely (causing a redirect) rather than showing a message for probationary accounts?



AI responses may include mistakes. For legal advice, consult a professional. [Learn more](#)

not a probationary period, but a probational condition

To impose a **probationary condition** rather than a passing time period, we will change the system from a time-based model to an **action-based condition**.

A probationary condition means a new member remains restricted until an administrator manually log onto the site, reviews their profile parameters, and explicitly verifies that they have met the club's requirement (e.g., attending an introductory meeting, submitting physical background paperwork, or completing an orientation).

Step 1: Initialize New Registrations under Condition Check

We will update our custom registration code so that when someone registers, they are instantly flagged as `probationary` and their mandatory condition field is marked as `Pending Review`.

Replace your user registration processing code with this logic:

Ask anything



```
// 1. Establish the explicit probationary condition flag
update_user_meta( $user_id, 'membership_status', 'probationary' );
update_user_meta( $user_id, 'phone_number', $phone );

// 2. Add a clear, readable description tracking what condition needs fu
update_user_meta( $user_id, 'probation_condition_status', 'Pending Fulfi
update_user_meta( $user_id, 'probation_condition_notes', 'Awaiting admin

// Automatically log in and push to payment pages
$creds = array( 'user_login' => $username, 'user_password' => $password,
wp_signon( $creds, false );

wp_redirect( home_url( '/pay-dues/' ) );
exit;
}
```

Use code with caution.

Step 2: Update the Gate to Explain the Specific Condition

We will modify the content-blocking shortcode (`[member_only]`) so it doesn't mention a countdown timer or a date. Instead, it informs the member that their account is waiting for a specific rule condition to be approved by an administrator.

php

```
function club_restrict_by_payment_status( $atts, $content = null ) {
    if ( ! is_user_logged_in() ) {
        return '<div style="padding:15px; background:#f8d7da; color:#721c24;
    }

    $user_id = get_current_user_id();
```

Ask anything

+



```
11 ( $status == 'probationary' ) {  
    $condition_notes = get_user_meta( $user_id, 'probation_condition_not  
  
    return '<div style="padding:20px; background:#f4f6f8; border-left:4p  
        <strong>Conditional Membership Status:</strong> Your account  
        <span style="font-size:14px; color:#555;"><strong>Current Requir  
        Once our administrative desk verifies this condition is fulfill  
    </div>';  
}  
  
return '<div style="padding:15px; background:#fff3cd; color:#856404; bor  
>
```

Use code with caution.

Step 3: Add the Condition Controls to Your Admin Form

To make this functional, we will add fields to your **Administrative Frontend Form** ([club_admin_console]). This allows managers to view a member's condition details, write internal evaluation notes, and mark the condition as **"Fulfilled"** to instantly upgrade the user to standard status.

Locate the field input generation area inside the individual editing layout view of your admin script and add these fields:

php

```
// Insert this block inside your administrative edit screen HTML code genera  
$condition_status = get_user_meta( $edit_id, 'probation_condition_status', t  
$condition_notes = get_user_meta( $edit_id, 'probation_condition_notes', tr  
  
$output .= '<hr style="border:0; border-top:1px solid #e3e8ee; margin:25px 0  
$output .= '<h4 style="color:#32325d;">🔧 Probationary Condition Assessment·
```

Ask anything

+



```
$output .= '<div style="margin-bottom:20px;"><label style="display:block; fo
$output .= '<textarea name="probation_condition_notes" rows="3" style="width
$output .= '</div>';
```

Use code with caution.

Next, add the backend code to process and save these new inputs when the administrator submits the edit form:

php

```
// Insert this logic inside your admin save handler logic block:
if ( isset( $_POST['club_admin_update_member'] ) ) {
    $target_user_id      = intval( $_POST['target_user_id'] );
    $updated_cond_status = sanitize_text_field( $_POST['probation_condition_
    $updated_cond_notes  = sanitize_text_field( $_POST['probation_condition_

    update_user_meta( $target_user_id, 'probation_condition_status', $update
    update_user_meta( $target_user_id, 'probation_condition_notes', $updated

    // If the administrator changes the condition tracker status to fulfille
    if ( $updated_cond_status === 'Fulfilled' ) {
        update_user_meta( $target_user_id, 'membership_status', 'active' );
    } else {
        update_user_meta( $target_user_id, 'membership_status', 'probationar
    }
}
```

Use code with caution.

Ask anything



 = 

- Would you like the system to **automatically send a custom email alert to the administrator** the moment a new user registers, reminding you that a new account needs its conditions reviewed?
- Do you want the condition requirements to be different depending on whether the user signs up for the **Standard Member** or **VIP Member** role?

Ask anything

+



without installing plugins, you can create a simple front-end card interface using HTML and standard CSS print styles.

By leveraging a custom shortcode (`[membership_card]`), your code will dynamically pull the member's name, role, phone number, and subscription renewal date straight from the WordPress database. It will then generate a sleek card directly on their user dashboard layout.

Step 1: Code the Membership Card Interface

Add this code block into your custom plugin script file. It handles the display rules and embeds a lightweight **Print Card** button that isolatedly triggers the browser's native print framework.

php

```
// Shortcode to display a printable membership card
add_shortcode( 'membership_card', 'club_render_printable_membership_card' );

function club_render_printable_membership_card() {
    // 1. Kick out logged-out visitors
    if ( ! is_user_logged_in() ) {
        return '<p>Please <a href="' . wp_login_url() . '">log in</a> to vie
    }

    $user_id = get_current_user_id();
    $user     = wp_get_current_user();
    $status   = get_user_meta( $user_id, 'membership_status', true );

    // 2. Prevent unpaid users from generating cards
    if ( $status !== 'active' && $status !== 'probationary' && ! current_use
        return '<p style="color:#d9534f; font-weight:bold;">⚠ Access Restr:
    }
}
```

Ask anything

+



```
// 4. HTML Structure + CSS Injection (includes print optimization styles)
ob_start();
?>
<style>
    /* Screen View Styles */
    .club-card-container {
        max-width: 400px;
        margin: 20px auto;
        text-align: center;
    }
    .membership-card-box {
        width: 350px;
        height: 200px;
        margin: 0 auto 15px auto;
        background: linear-gradient(135deg, #1e3c72 0%, #2a5298 100%);
        color: #ffffff;
        border-radius: 12px;
        padding: 20px;
        text-align: left;
        position: relative;
        box-shadow: 0 10px 20px rgba(0,0,0,0.15);
        font-family: 'Helvetica Neue', Arial, sans-serif;
        box-sizing: border-box;
    }
    .card-header {
        font-size: 18px;
        font-weight: bold;
        letter-spacing: 1px;
        border-bottom: 1px solid rgba(255,255,255,0.2);
        padding-bottom: 8px;
        margin-bottom: 15px;
        text-transform: uppercase;
    }
    .card-name {
        font-size: 20px;
        font-weight: bold;
        margin-bottom: 5px;
    }
```

Ask anything



```
position: absolute;
bottom: 20px;
right: 20px;
background: rgba(255,255,255,0.2);
padding: 4px 10px;
border-radius: 20px;
font-size: 11px;
font-weight: bold;
text-transform: uppercase;
}
.print-card-btn {
background: #6772e5;
color: #fff;
border: 0;
padding: 10px 20px;
border-radius: 5px;
font-weight: bold;
cursor: pointer;
font-size: 14px;
transition: background 0.2s;
}
.print-card-btn:hover { background: #5561e2; }

/* Print Media Styles: Hides EVERYTHING on the site except the card
@media print {
body * {
visibility: hidden;
}
.club-card-container, .club-card-container * {
visibility: visible;
}
.club-card-container {
position: absolute;
left: 0;
top: 0;
width: 100%;
```

Ask anything



```
        print-color-adjust: exact;
    }
}
</style>

<div class="club-card-container">
    <!-- The Visual Membership Card Box -->
    <div class="membership-card-box">
        <div class="card-header">🏠 Global Club Network</div>
        <div class="card-name"><?php echo esc_html( $user->display_name
        <div class="card-detail"><strong>Email:</strong> <?php echo esc_
        <?php if ( ! empty( $phone ) ) : ?>
            <div class="card-detail"><strong>Phone:</strong> <?php echo
        <?php endif; ?>
        <div class="card-detail" style="margin-top: 15px;"><strong>Valid

        <div class="card-badge"><?php echo esc_html( $role_display ); ?>
    </div>

    <!-- Print Action Trigger Button -->
    <button class="print-card-btn" onclick="window.print();">🖨️ Print M
</div>
<?php
return ob_get_clean();
}
```

Use code with caution.

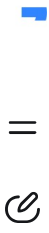
Step 2: Deploy to the Dashboard

You can bundle this feature cleanly into your existing architecture. Open up the standard WordPress dashboard page where your `[member_dashboard]` code runs,

Ask anything



Membership Card Operation Matrix



Operational Component	Frontend Output Pipeline	Print Mode Filter Behavior	Database Dependency
Card Card Display	Standard responsive div wrapper block	Styled using custom background gradients on web pages.	References user core variables (<code>display_name</code> , <code>user_email</code>).
Data Synchronization	Shortcode parser compilation	Pulls active dates, tiers, and administrative condition meta strings on page refresh.	Targets table row variables inside the <code>wp_usermeta</code> fields.
Print Output Hook	JavaScript <code>window.print()</code> trigger	Global <code>@media print</code> layout css instantly hides sidebars, headers, and footer components.	Formats properties cleanly within default desktop/mobile print engines.

Now that members can view and print their cards, let me know:

- Do you want to **integrate a dynamic QR code** onto the membership card that links directly back to their frontend admin profile page for quick in-person scanning?
- Would you like to **style different colors for the card background** depending on whether they are a **Standard Member** (e.g., corporate blue) or a **VIP Member** (e.g., premium gold)?

Ask anything



-  To unify all the custom, plugin-free features you requested, I have assembled them into a single, cohesive file. This file functions as a lightweight, custom WordPress plugin.
- 
-  This single file handles **custom roles, frontend registration, tiered calendar-year Stripe billing, automated welcome/success emails, December 15th expiration notifications, probationary condition tracking, frontend restrictions, administrative edit panels, and printable digital membership cards.**

The Complete Single-File Code

Create a file named `club-membership-core.php` inside your `/wp-content/plugins/` directory (or paste this into a new folder at `/wp-content/plugins/club-membership/club-membership.php`) and paste the following code block:

php

```
<?php
/*
Plugin Name: Custom Club Membership Core System
Description: Complete plugin-free architecture handling frontend registratio
Version: 1.5
Author: Custom Club Dev
*/

if ( ! defined( 'ABSPATH' ) ) { exit; } // Prevent direct access

// Global Configurations (Update with your live keys)
define( 'CLUB_STRIPE_SECRET_KEY', 'sk_live_YOUR_SECRET_KEY' );
define( 'CLUB_STANDARD_PRICE_ID', 'price_STANDARD_PRICE_ID_HERE' );
define( 'CLUB_VIP_PRICE_ID', 'price_VIP_PRICE_ID_HERE' );

/**
```

Ask anything

+



```
add_shortcode( 'club_registration_form', 'club_render_frontend_registration'
function club_render_frontend_registration() {
    if ( is_user_logged_in() ) {
        return '<p>You are already registered! Head over to your <a href=""

$errors = array();
if ( isset( $_POST['club_submit_registration'] ) ) {
    if ( ! isset( $_POST['club_reg_nonce'] ) || ! wp_verify_nonce( $_POS
        $errors[] = 'Security check failed. Please refresh and try again
    } else {
        $username      = sanitize_user( $_POST['club_username'] );
        $email          = sanitize_email( $_POST['club_email'] );
        $password       = $_POST['club_password'];
        $phone          = sanitize_text_field( $_POST['club_phone'] );
        $chosen_role    = sanitize_text_field( $_POST['club_role'] );

        if ( empty( $username ) || empty( $email ) || empty( $password ) )
        if ( username_exists( $username ) ) { $errors[] = 'That username
        if ( ! is_email( $email ) ) { $errors[] = 'Please enter a valid
        if ( email_exists( $email ) ) { $errors[] = 'That email is alrea
        if ( strlen( $password ) < 8 ) { $errors[] = 'Password must be a
        if ( ! in_array( $chosen_role, array( 'standard_member', 'vip_me

        if ( empty( $errors ) ) {
            $user_id = wp_create_user( $username, $password, $email );
            if ( ! is_wp_error( $user_id ) ) {
                $user = new WP_User( $user_id );
                $user->set_role( $chosen_role );

                // Set standard onboarding probationary condition variab
                update_user_meta( $user_id, 'phone_number', $phone );
                update_user_meta( $user_id, 'membership_status', 'probat
                update_user_meta( $user_id, 'probation_condition_status'
                update_user_meta( $user_id, 'probation_condition_notes',
```

Ask anything

+



```

        } else { $errors[] = $user_id->get_error_message(); }
    }
}

ob_start();
?>
<div class="club-registration-card" style="max-width: 450px; margin: 0 auto; padding: 10px; border: 1px solid #ccc; border-radius: 10px;">
    <h3 style="margin-top:0; color:#32325d; text-align:center;">Create Your Account
    <?php if ( ! empty( $errors ) ) : ?>
        <div style="background:#f8d7da; color:#721c24; padding:12px; border:1px solid #f5c6cb; margin-bottom:10px;">
            <?php foreach ( $errors as $error ) { echo ' <p style="margin:0;">
                <strong>Error</strong>: <span style="color:#721c24;">
                    <?php echo $error;
                </span>
            </p>
            </div>
        <?php endif; ?>
        <form method="POST">
            <?php echo wp_nonce_field( 'club_user_registration', 'club_reg_nonce' );
            <div style="margin-bottom:15px;"><label style="display:block; margin-bottom:5px;">Full Name
            <input type="text" name="club_full_name" value="" style="width:100%; padding:5px; border:1px solid #ccc;" />
            <div style="margin-bottom:15px;"><label style="display:block; margin-bottom:5px;">Email Address
            <input type="email" name="club_email" value="" style="width:100%; padding:5px; border:1px solid #ccc;" />
            <div style="margin-bottom:15px;"><label style="display:block; margin-bottom:5px;">Phone Number
            <input type="text" name="club_phone" value="" style="width:100%; padding:5px; border:1px solid #ccc;" />
            <div style="margin-bottom:15px;"><label style="display:block; margin-bottom:5px;">Password
            <input type="password" name="club_password" value="" style="width:100%; padding:5px; border:1px solid #ccc;" />
            <div style="margin-bottom:25px;"><label style="display:block; margin-bottom:5px;">Select Member Tier
            <select name="club_role" style="width:100%; padding:10px; border:1px solid #ccc;">
                <option value="standard_member">Standard Member Tier</option>
                <option value="vip_member">VIP Member Tier</option>
            </select>
            </div>
            <button type="submit" name="club_submit_registration" style="width:100%; padding:10px; background-color:#32325d; color:white; border:none;">
                Register
            </form>
        </div>
        <?php
        return ob_get_clean();
    }

function club_send_initial_welcome_email( $username, $email ) {
    $subject = 'Welcome to the Club! 🚀 Your Account Details';
    $message = '<html><body style="font-family:Arial,sans-serif; color:#333; background-color:#f9f9f9; padding:20px 0;">
        <div style="background-color:white; border:1px solid #ddd; border-radius:10px; padding:20px; margin:10px auto; width:90%; text-align:center;">
            <h2 style="margin:0;">Welcome to the Club!
            <h3 style="margin:5px 0 0 0;">Your Account Details
            <table border="1" style="margin:10px auto; width:80%; border-collapse:collapse;">
                <thead>
                    <tr>
                        <th style="padding:5px;">Field
                        <th style="padding:5px;">Value
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td style="padding:5px;">Username
                        <td style="padding:5px;">
                            <strong>
                                <?php echo $username;
                            </strong>
                        </td>
                    </tr>
                    <tr>
                        <td style="padding:5px;">Email
                        <td style="padding:5px;">
                            <strong>
                                <?php echo $email;
                            </strong>
                        </td>
                    </tr>
                </tbody>
            </table>
            <div style="margin-top:10px;">
                <strong>Thank you for joining! We are excited to have you on board.
            </div>
        </div>
    </body></html>';
    wp_mail( $email, $subject, $message );
}

```

Ask anything



```
add_shortcode( 'stripe_pay_button', 'club_render_role_based_pay_button' );
function club_render_role_based_pay_button() {
    if ( ! is_user_logged_in() ) { return '<p>Please <a href="" . wp_login_u
    if ( isset( $_POST['action'] ) && $_POST['action'] === 'club_stripe_chec

    $current_user = wp_get_current_user();
    $button_text = 'Pay Full Annual Dues';
    if ( in_array( 'vip_member', $current_user->roles ) ) { $button_text = '
    elseif ( in_array( 'standard_member', $current_user->roles ) ) { $button

    return '<form method="POST"><input type="hidden" name="action" value="cl
}

function club_initiate_fixed_calendar_checkout() {
    $current_user = wp_get_current_user();
    $stripe_price_id = CLUB_STANDARD_PRICE_ID;
    if ( in_array( 'vip_member', $current_user->roles ) ) { $stripe_price_id

    $response = wp_remote_post( 'https://stripe.com', array(
        'headers' => array( 'Authorization' => 'Bearer ' . CLUB_STRIPE_SECRE
        'body' => http_build_query( array(
            'success_url' => home_url( '/membership-success/' ),
            'cancel_url' => home_url( '/membership-failed/' ),
            'mode' => 'subscription',
            'customer_email' => $current_user->user_email,
            'client_reference_id' => $current_user->ID,
            'line_items[0][price]' => $stripe_price_id,
            'line_items[0][quantity]' => 1,
        ) ),
    ) );

    if ( ! is_wp_error( $response ) ) {
        $body = json_decode( wp_remote_retrieve_body( $response ), true );
        if ( isset( $body['url'] ) ) { wp_redirect( $body['url'] ); exit; }
    }
}
```

Ask anything



Use code with caution.

```
$event = json_decode( $payload, true );

if ( $event && $event['type'] === 'checkout.session.completed' ) {
    $session = $event['data']['object'];
    $user_id = intval( $session['client_reference_id'] );

    if ( $user_id ) {
        $user = get_userdata( $user_id );

        // Fixed end-of-year absolute expiration stamp configuration matrix
        $dec_31_timestamp = strtotime( "December 31, " . date('Y') . " 23:59:59" );

        // Retain probationary condition if they haven't met administrative checklist targets yet
        $current_status = get_user_meta( $user_id, 'membership_status', true );
        if ( $current_status !== 'probationary' ) {
            update_user_meta( $user_id, 'membership_status', 'active' );
        }

        update_user_meta( $user_id, 'membership_renewal_date', $dec_31_timestamp );
        update_user_meta( $user_id, 'stripe_subscription_id', sanitize_text_field(
            $session['subscription'] ) );

        // Send standard receipt transaction confirmation template
        club_send_payment_success_email( $user, date( 'F j, Y', $dec_31_timestamp ) );
    }
}

return new WP_REST_Response( array( 'status' => 'success' ), 200 );
}

function club_send_payment_success_email( $user, $formatted_date ) {
    $subject = 'Payment Confirmed! Your Club Membership Status Update 🎉';
```

Ask anything

+



/**

• 3. SECURITY GATEWAYS AND FRONTEND LOCK RULES

*/

add_shortcode('member_only', 'club_restrict_by_payment_and_condition');

function club_restrict_by_payment_and_condition(\$atts, \$content = null) {

if (! is_user_logged_in()) {

return '  **Members Only:** Please Log In to view this content.';

}

\$user_id = get_current_user_id();

\$status = get_user_meta(\$user_id, 'membership_status', true);if (

current_user_can('administrator') || \$status === 'active') { return do_shortcode(

\$content); }if (\$status === 'probationary') {

\$notes = get_user_meta(\$user_id, 'probation_condition_notes', true);

return '  **Conditional Probation:** Your profile is pending condition review.

Missing Target: ' . esc_html(\$notes) . '";

}return '  Access Restricted. Please submit your Annual Dues.';

}

add_filter('wp_nav_menu_objects', 'club_restrict_nav_menu_links', 10, 2);

function club_restrict_nav_menu_links(\$sorted_menu_items, \$args) {

\$user_id = get_current_user_id();

\$status = get_user_meta(\$user_id, 'membership_status', true);

\$is_clear = (is_user_logged_in() && (\$status === 'active' || current_user_can('administrator')));

\$restricted_menu_titles = array('Member Portal', 'Club Directory', 'Member Dashboard', 'Exclusive Events');

foreach (\$sorted_menu_items as \$key => \$item) {

if (in_array(\$item->title, \$restricted_menu_titles) && ! \$is_clear) { unset(\$sorted_menu_items[\$key]); }

}

return \$sorted menu items;

Ask anything

+





```
function club_render_member_dashboard() {
    if ( ! is_user_logged_in() ) { return 'Please log in.'; }
    $user_id = get_current_user_id(); $user = wp_get_current_user();
    $status = get_user_meta( $user_id, 'membership_status', true );
    $renewal_time = get_user_meta( $user_id, 'membership_renewal_date', true
    );

    if ( empty( $status ) || ( $status !== 'active' && $status !== 'probationary' ) ) {
        return 'Welcome, '.esc_html($user->display_name). '!Your dues are unpaid.
        Process Payment';
    }
    $renewal_date = !empty($renewal_time) ? date( 'F j, Y', intval( $renewal_time ) ) :
    'Dec 31, ' . date('Y');
    return 'Welcome back, '.esc_html($user->display_name). '! 🙌
    Status'.ucfirst($status).'Valid Until'.esc_html($renewal_date).';
    }
}
```

```
add_shortcode( 'membership_card', 'club_render_printable_membership_card' );
function club_render_printable_membership_card() {
    if ( ! is_user_logged_in() ) { return ''; }
    $user_id = get_current_user_id(); $user = wp_get_current_user();
    $status = get_user_meta( $user_id, 'membership_status', true );
    if ( $status !== 'active' && $status !== 'probationary' && ! current_user_can(
    'administrator' ) ) { return ''; }

    $role_display = ucwords( str_replace( '_', ' ', $user->roles[0] ) );
    $renewal_time = get_user_meta( $user_id, 'membership_renewal_date', true );
    $expiry_date = ! empty( $renewal_time ) ? date( 'F j, Y', intval( $renewal_time ) ) :
    'December 31, ' . date('Y');

    ob_start();
    ?>
```



OFFICIAL CLUB CARD

Ask anything



```

container * { visibility: visible; } .club-card-container { position: absolute; left: 0; top:
0; width: 100%; } button { display: none !important; } }

```

```
<?php
```

```
return ob_get_clean();
```

```
}
```

```
/**
```

• 5. ADMIN CONSOLE PORTAL LAYER

```
*/
```

```
add_shortcode( 'club_admin_console', 'club_render_admin_member_console'
);
```

```
function club_render_admin_member_console() {
```

```
if ( ! current_user_can( 'administrator' ) ) { return '🔒 Restricted Area.'; }
```

```
$output = '';
```

```
if ( isset( $_POST['club_admin_update_member'] ) ) {
```

```
if ( isset( $_POST['club_admin_nonce'] ) && wp_verify_nonce(
```

```
$_POST['club_admin_nonce'], 'club_admin_edit_action' ) ) {
```

```
$tid = intval( $_POST['target_user_id'] );
```

```
update_user_meta( $tid, 'phone_number', sanitize_text_field(
$_POST['member_phone'] ) );
```

```
update_user_meta( $tid, 'probation_condition_status', sanitize_text_field(
$_POST['probation_condition_status'] ) );
```

```
update_user_meta( $tid, 'probation_condition_notes', sanitize_text_field(
$_POST['probation_condition_notes'] ) ); $new_status = sanitize_text_field(
$_POST['member_status'] );
```

```
if ( $_POST['probation_condition_status'] === 'Fulfilled' && $new_status ===
'probationary' ) { $new_status = 'active'; }
```

```
update_user_meta( $tid, 'membership_status', $new_status ); $t_user = new
WP_User( $tid );
```

```
$t_user->set_role( sanitize_text_field( $_POST['member_role'] ) );
```

```
$output .= '✅ Member records updated.';
```

```
}
```

Ask anything

+



```
. = '  Edit Member: ' . esc_html( $ud->user_login ) . '← Back';
$output .= " . wp_nonce_field( 'club_admin_edit_action', 'club_admin_nonce',
true, false ) . " ";
$output .= 'Phone Number';
$output .= 'Status<option value="active"
'.selected($status,'active',false).>Active<option value="probationary"
'.selected($status,'probationary',false).>Probationary<option value="unpaid"
'.selected($status,'unpaid',false).>Unpaid';
$output .= 'System Role<option value="standard_member" '.selected($ud-
>roles[0],'standard_member',false).>Standard Member<option
value="vip_member" '.selected($ud->roles[0],'vip_member',false).>VIP Member';
$output .= 'Condition Tracking<option value="Pending Fulfillment"
'.selected($c_status,'Pending Fulfillment',false).>Pending<option value="Fulfilled"
'.selected($c_status,'Fulfilled',false).>Fulfilled (Auto-activates)';
$output .= 'Condition Criteria'.esc_textarea($c_notes).";
$output .= 'Save Changes';
} else {
$output .= '  Club Directory Dashboard';
$users = get_users();
$output .= 'UserRoleBilling StatusAction';
foreach ( $users as $m ) {
$m_status = get_user_meta( $m->ID, 'membership_status', true );
$output .= ".esc_html($m->user_login)".esc_html($m-
>roles[0]).".esc_html(ucfirst($m_status)).'Edit';
}
$output .= " ";
}
return $output . " ";
}
```

/**

- 6. RECURRING BACKGROUND CRON AUTOMATION

*/

Ask anything



// A. Perform Midnight December 31st Rollover Lockout Sweep

```
$active_members = get_users( array( 'meta_key' => 'membership_status',  
'meta_value' => 'active' ) );  
foreach ( $active_members as $m ) {  
$expiration = get_user_meta( $m->ID, 'membership_renewal_date', true );  
if ( ! empty( $expiration ) && $now > intval( $expiration ) ) { update_user_meta( $m->ID, 'membership_status', 'unpaid' ); }  
}
```

// B. Send December 15th Renewal Reminder Alerts

```
if ( date( 'm-d' ) === '12-15' ) {  
foreach ( $active_members as $m ) {  
$subject = '📅 Upcoming Annual Membership Renewal Notice';  
$message = 'Hello '.esc_html($m->display_name).',Your annual cycle access ends on  
December 31st. Please log into your dashboard to complete next year's dues  
selection early.';  
wp_mail( $m->user_email, $subject, $message, array('Content-Type: text/html;  
charset=UTF-8') );  
}  
}
```

Custom Package Page Setup Checklist

Once you activate this custom code plugin file via your main WordPress **Plu

Frontend Page Title	Target URL Slug	Paste Shortcode Code Marker	Acce
:---	:---	:---	:---
Join Our Club	`/join-the-club/`	`[club_registration_form]`	Publi

Ask anything



* Do you need assistance setting up your local **WordPress development envir
* Would you like me to map out the exact **Stripe Dashboard Webhook signing
</FollowUp>

=



Ask anything

