

ENNU Life Plugin - Comprehensive Code Documentation

Plugin Structure Overview

The ENNU Life plugin is organized into a modular architecture with clear separation of concerns:

Plain Text

```
ennulifeassessments/
├── ennu-life-plugin.php           # Main plugin file & initialization
├── includes/                     # Core PHP classes
│   ├── class-admin.php          # WordPress admin interface
│   ├── class-assessment-cpt.php # Custom post types for assessments
│   ├── class-assessment-shortcodes.php # Shortcode handlers
│   ├── class-database.php       # Database operations
│   ├── class-debug-logger.php   # Logging system
│   ├── class-email-system.php   # Email functionality
│   ├── class-form-handler.php   # AJAX form processing
│   ├── class-live-content-manager.php # Dynamic content
│   ├── class-scoring-system.php # Assessment scoring
│   ├── class-template-loader.php # Template system
│   └── class-woocommerce-integration.php # E-commerce
├── assets/                      # Frontend resources
│   ├── css/                    # Stylesheets
│   └── js/                     # JavaScript files
├── templates/                  # PHP template files
└── languages/                  # Internationalization
```

Key Interconnections

Data Flow: Form submission → AJAX handler → Database → Admin display

Template System: Shortcodes → Template loader → PHP templates → Frontend display

User Management: Form handler ↔ WordPress users ↔ Admin interface



Main Plugin File: ennu-life-plugin.php



Plugin Header & Security

Plugin Header Block (Lines 2-17)

PHP

```
/**
 * Plugin Name: ENNU Life - Health Platform
 * Version: 22.4
 * ...
 */
```

- **Purpose:** WordPress plugin metadata for recognition and management
- **Key Fields:** Name, version, description, requirements, text domain
- **Interconnection:** Version number used throughout plugin for cache busting

Security Check (Lines 25-27)

PHP

```
if ( ! defined( 'ABSPATH' ) ) {
    exit( 'Direct access forbidden.' );
}
```

- **Purpose:** Prevents direct file access outside WordPress
- **Security:** Essential WordPress security practice



Constants Definition (Lines 29-48)

Plugin Constants Block

PHP

```
define( 'ENNU_LIFE_VERSION', '22.4' );
define( 'ENNU_LIFE_PLUGIN_FILE', __FILE__ );
// ... more constants
```

- **Purpose:** Global plugin configuration accessible throughout codebase
- **Key Constants:**
 - `ENNU_LIFE_VERSION` : Used for asset versioning and cache busting
 - `ENNU_LIFE_PLUGIN_PATH` : File system paths for includes
 - `ENNU_LIFE_PLUGIN_URL` : Web URLs for assets (CSS/JS)
- **Interconnection:** Used by all classes for file loading and asset enqueueing



Main Plugin Class: `ENNU_Life_Plugin`

Singleton Pattern Implementation (Lines 54-89)

PHP

```
private static $instance = null;
public static function get_instance() {
    if ( null === self::$instance ) {
        self::$instance = new self();
    }
    return self::$instance;
}
```

- **Purpose:** Ensures only one plugin instance exists
- **Benefits:** Prevents conflicts, manages resources efficiently
- **Interconnection:** Other classes access plugin via `ENNU_Life_Plugin::get_instance()`

System Requirements Check (Lines 106-125)

PHP

```
private function check_requirements() {
    // Check PHP version
    if ( version_compare( PHP_VERSION, ENNU_LIFE_MIN_PHP_VERSION, '<' ) ) {
        add_action( 'admin_notices', array( $this, 'php_version_notice' ) );
        return false;
    }
    // Check WordPress version
    // ...
}
```

- **Purpose:** Validates environment before initialization
- **Checks:** PHP version (7.4+), WordPress version (5.0+)
- **Failure Handling:** Shows admin notices, prevents plugin loading

Plugin Initialization (Lines 127-158)

Main Init Method

PHP

```
public function init() {
    if ( $this->initialized ) {
        return; // Prevent double initialization
    }

    try {
        $this->load_textdomain();    // Internationalization
        $this->load_includes();      // Load all PHP classes
        $this->init_components();    // Initialize class instances
        $this->setup_hooks();        // WordPress hooks

        $this->initialized = true;
    } catch ( Exception $e ) {
        // Error handling
    }
}
```

- **Purpose:** Orchestrates plugin startup sequence
- **Error Handling:** Try-catch prevents fatal errors

- **Interconnection:** Calls all major initialization methods

File Loading (Lines 170-190)

PHP

```
private function load_includes() {
    $includes = array(
        'includes/class-debug-logger.php',
        'includes/class-database.php',
        // ... all class files
    );

    foreach ( $includes as $include ) {
        $file_path = ENNU_LIFE_PLUGIN_PATH . $include;
        if ( ! file_exists( $file_path ) ) {
            throw new Exception( sprintf( 'Required file not found: %s',
            $include ) );
        }
        require_once $file_path;
    }
}
```

- **Purpose:** Loads all PHP class files in correct order
- **Order Matters:** Database and logger loaded first, then dependent classes
- **Error Handling:** Throws exception if files missing

Component Initialization (Lines 192-240)

Component Registry

PHP

```
private function init_components() {
    // Database (singleton)
    if ( class_exists( 'ENNU_Life_Database' ) ) {
        $this->components['database'] = ENNU_Life_Database::get_instance();
    }

    // Form handler (instance)
    if ( class_exists( 'ENNU_Life_Form_Handler' ) ) {
```

```

        $this->components['form_handler'] = new ENNU_Life_Form_Handler();
    }
    // ... more components
}

```

- **Purpose:** Creates instances of all plugin classes
- **Pattern:** Mix of singletons and regular instances
- **Interconnection:** Components stored in `$this->components` array for access
- **Key Components:**
 - **Database:** Singleton for data operations
 - **Form Handler:** AJAX form processing
 - **Admin:** WordPress admin interface
 - **Assessment Shortcodes:** Frontend display



WordPress Hooks Setup (Lines 252-280)

Hook Registration

PHP

```

private function setup_hooks() {
    // Admin hooks
    add_action( 'admin_enqueue_scripts', array( $this, 'enqueue_admin_assets' ) );

    // Frontend hooks
    add_action( 'wp_enqueue_scripts', array( $this, 'enqueue_frontend_assets' ) );

    // AJAX hooks (NO SECURITY CHECKS - Important!)
    add_action( 'wp_ajax_enu_form_submit', array( $this, 'ajax_form_submit' ) );
    add_action( 'wp_ajax_nopriv_enu_form_submit', array( $this, 'ajax_form_submit' ) );

    // User profile hooks

```

```

    add_action( 'show_user_profile', array( $this,
'show_user_assessment_fields' ) );
    add_action( 'edit_user_profile', array( $this,
'show_user_assessment_fields' ) );
}

```

- **Purpose:** Registers WordPress action/filter hooks
- **Critical:** AJAX hooks have NO security checks (removed for compatibility)
- **Interconnection:** Links WordPress events to plugin methods

Asset Management (Lines 400-480)

Frontend Asset Loading

PHP

```

public function enqueue_frontend_assets() {
    if ( ! $this->should_load_assets() ) {
        return; // Conditional loading for performance
    }

    // CSS files
    wp_enqueue_style( 'ennu-main-style', ... );
    wp_enqueue_style( 'ennu-assessment-style', ... );

    // JavaScript files
    wp_enqueue_script( 'ennu-main-script', ... );
    wp_enqueue_script( 'ennu-assessment-script', ... );

    // AJAX configuration
    wp_localize_script( 'ennu-main-script', 'ennuAjax', array(
        'ajaxurl' => admin_url( 'admin-ajax.php' ),
        'nonce' => wp_create_nonce( 'ennu_ajax_nonce' ),
        // ...
    ) );
}

```

- **Purpose:** Loads CSS/JS only when needed
- **Performance:** Conditional loading based on page content

- **Interconnection:** Provides AJAX configuration to JavaScript

Smart Asset Detection

PHP

```
private function should_load_assets() {
    // Check for ENNU shortcodes in content
    if ( $post && $this->has_enu_shortcode( $post->post_content ) ) {
        return true;
    }

    // Check page slugs
    $enu_pages = array( 'health-assessment', 'weight-loss-assessment', ...
);
    // ...
}
```

- **Purpose:** Optimizes performance by loading assets only when needed
- **Checks:** Shortcodes, page slugs, templates, meta fields



AJAX Handlers (Lines 550-620)

Form Submission Handler

PHP

```
public function ajax_form_submit() {
    // NO SECURITY CHECK - Removed for compatibility
    error_log('ENNU: Main plugin AJAX handler called - delegating to form handler');

    if ( isset( $this->components['form_handler'] ) ) {
        $this->components['form_handler']->handle_ajax_submission();
    } else {
        wp_send_json_error( array( 'message' => 'Form handler not available.'
) );
    }
}
```

- **Purpose:** Routes AJAX requests to form handler

- **Security:** NO nonce verification (removed for assessment compatibility)
- **Interconnection:** Delegates to `ENNU_Life_Form_Handler` class

Other AJAX Handlers

- **Appointment Booking:** Processes booking requests
- **Membership Calculator:** Handles cost calculations
- **Pattern:** All AJAX handlers have security checks removed

User Profile Integration (Lines 700-900)

Profile Fields Display

PHP

```
public function show_user_assessment_fields( $user ) {  
    $assessment_types = array(  
        'hair_assessment' => 'Hair Assessment',  
        'ed_treatment_assessment' => 'ED Treatment Assessment',  
        // ... all 5 assessment types  
    );  
  
    foreach ( $assessment_types as $type => $label ) {  
        $this->display_assessment_fields( $user->ID, $type, $label );  
    }  
}
```

- **Purpose:** Shows assessment data in WordPress user profiles
- **Interconnection:** Works with admin class for enhanced display
- **Data Source:** Reads from user meta fields

Field Labels System

PHP

```

private function get_assessment_field_labels( $assessment_type ) {
    $labels = array(
        'first_name' => 'First Name',
        'last_name' => 'Last Name',
        // ... standard fields
    );

    switch ( $assessment_type ) {
        case 'hair_assessment':
            $labels = array_merge( $labels, array(
                'hair_gender' => 'Gender',
                'hair_concern_type' => 'Hair Concern Type',
                // ... hair-specific fields
            ) );
            break;
        // ... other assessment types
    }
}

```

- **Purpose:** Provides human-readable labels for form fields
- **Extensible:** Easy to add new assessment types and fields

Plugin Lifecycle (Lines 282-350)

Activation Handler

PHP

```

public static function activate() {
    try {
        $instance = self::get_instance();
        $instance->create_database_tables(); // Create DB tables
        $instance->set_default_options(); // Set default settings
        $instance->cleanup_dummy_assessment_fields(); // Clean old data
        flush_rewrite_rules(); // Update permalinks
    } catch ( Exception $e ) {
        wp_die( 'Plugin activation failed...' );
    }
}

```

- **Purpose:** Sets up plugin on activation

- **Database:** Creates assessment tables
- **Safety:** Error handling prevents partial activation

Database Table Creation

PHP

```
private function create_database_tables() {  
    $sql = "CREATE TABLE {$wpdb->prefix}ennu_assessments (  
        id mediumint(9) NOT NULL AUTO_INCREMENT,  
        assessment_type varchar(255) NOT NULL,  
        assessment_data longtext NOT NULL,  
        user_id bigint(20) DEFAULT NULL,  
        results longtext NOT NULL,  
        created_at datetime DEFAULT CURRENT_TIMESTAMP,  
        PRIMARY KEY (id),  
        KEY idx_user_id (user_id),  
        KEY idx_assessment_type (assessment_type)  
    )";  
  
    dbDelta( $sql );  
}
```

- **Purpose:** Creates database table for assessment storage
- **Structure:** Stores assessment data, results, user association
- **Indexes:** Optimized for common queries

Key Interconnections

Component Communication

- **Main Plugin → Form Handler:** AJAX delegation
- **Main Plugin → Admin Class:** User profile integration
- **Main Plugin → Shortcode Class:** Assessment rendering
- **All Classes → Main Plugin:** Access via singleton pattern

Data Flow

1. **Frontend Form** → **AJAX Handler** → **Form Handler** → **Database**
2. **Database** → **Admin Class** → **User Profile Display**
3. **Shortcodes** → **Template Loader** → **Frontend Display**

Critical Dependencies

- **Database class** must load before form handler
- **Form handler** must be initialized before AJAX hooks
- **Asset loading** depends on content detection
- **User profile** integration requires admin class



Includes Classes Documentation



CRITICAL CLASS: `class-form-handler.php`

Class Declaration & Constructor (Lines 9-28)

PHP

```
class ENNU_Life_Form_Handler {
    public function __construct() {
        error_log('ENNU: Form Handler constructor called - registering AJAX
actions');

        // Standard WordPress AJAX handlers
        add_action('wp_ajax_enu_form_submit', array($this,
'handle_ajax_submission'));
        add_action('wp_ajax_nopriv_enu_form_submit', array($this,
'handle_ajax_submission'));

        // Test handler
        add_action('wp_ajax_enu_test', array($this, 'test_ajax_handler'));
```

```

        add_action('wp_ajax_nopriv_ennu_test', array($this,
'test_ajax_handler'));

        // Custom endpoint
        add_action('init', array($this, 'add_custom_endpoint'));
        add_action('template_redirect', array($this,
'handle_custom_endpoint'));
    }
}

```

- **Purpose:** Registers AJAX handlers and custom endpoints for form processing
- **Key Feature:** Dual AJAX system (WordPress + custom endpoint)
- **Interconnection:** Called by main plugin during component initialization

Custom Endpoint System (Lines 30-65)

PHP

```

public function add_custom_endpoint() {
    add_rewrite_rule('^ennu-submit/?$', 'index.php?ennu_submit=1', 'top');
    add_rewrite_tag('%ennu_submit%', '([^&]+)');
}

public function handle_custom_endpoint() {
    if (get_query_var('ennu_submit')) {
        // Set JSON headers
        header('Content-Type: application/json');
        header('Access-Control-Allow-Origin: *');
        // Process form submission
        $this->handle_ajax_submission();
        exit;
    }
}

```

- **Purpose:** Provides alternative to WordPress AJAX for compatibility
- **URL:** `/ennu-submit/` bypasses WordPress AJAX entirely
- **Headers:** CORS-enabled for cross-origin requests



MAIN FORM HANDLER (Lines 75-185) - MOST CRITICAL

PHP

```
public function handle_ajax_submission() {
    // NO SECURITY CHECKS - Line 77
    error_log('ENNU v21.7: Form submission received');

    try {
        // Get form data (Lines 80-87)
        $form_data = $_POST;
        unset($form_data['action'], $form_data['nonce']);

        // Detect assessment type (Lines 89-95)
        $assessment_type = $this->detect_assessment_type($form_data);

        // Get or create user (Lines 97-104)
        $user_id = $this->get_or_create_user($form_data);


        // Save data with standardized naming (Lines 106-145)
        foreach ($form_data as $key => $value) {
            $clean_key = sanitize_key($key);
            $clean_assessment_type = str_replace('-', '_', $assessment_type);
            $meta_key = 'enu_' . $clean_assessment_type . '_' . $clean_key;

            $result = update_user_meta($user_id, $meta_key,
            $sanitized_value);

            // Save metadata for admin display (Lines 137-141)
            update_user_meta($user_id, $meta_key . '_date',
            current_time('mysql'));
            update_user_meta($user_id, $meta_key . '_label', $field_label);
        }

        // Save completion status (Lines 147-151)
        $completion_meta_key = 'enu_' . str_replace('-', '_',
        $assessment_type) . '_completion_status';
        update_user_meta($user_id, $completion_meta_key, 'completed');

        // Send success response (Lines 160-170)
        wp_send_json_success($response_data);
    }
}
```

-  **CRITICAL:** NO security checks (Line 77) - removed for compatibility
- **Field Naming:** ennu_[assessment_type]_[field_name] pattern (Line 130)

- **Metadata:** Saves timestamps and labels for admin display
- **Error Handling:** Try-catch with detailed logging

User Management System (Lines 187-260)

PHP

```
private function get_or_create_user($form_data) {
    // Check if user is logged in (Lines 189-197)
    $user_id = get_current_user_id();
    if ($user_id) {
        $this->update_logged_in_user_profile($user_id, $form_data);
        return $user_id;
    }

    // Get email from form (Lines 199-207)
    $email = sanitize_email($form_data['email']);

    // Check existing user (Lines 215-220)
    $existing_user = get_user_by('email', $email);
    if ($existing_user) {
        return $existing_user->ID;
    }

    // Create new user (Lines 222-245)
    $auto_password = wp_generate_password(12, false);
    $user_data = array(
        'user_login' => $email,
        'user_email' => $email,
        'user_pass' => $auto_password,
        'role' => 'subscriber'
    );

    $new_user_id = wp_insert_user($user_data);

    // Send welcome email (Line 253)
    $this->send_welcome_email($new_user_id, $email, $auto_password,
    $first_name);
}
```

- **Smart Logic:** Handles logged-in users vs. new user creation
- **Auto-Password:** 12-character password without special characters

- **Welcome Email:** Automatic email with login credentials

Profile Update System (Lines 262-320)

PHP

```
private function update_logged_in_user_profile($user_id, $form_data) {
    // Update first name (Lines 267-275)
    if (isset($form_data['first_name']) && !empty($form_data['first_name']))
    {
        $new_first_name = sanitize_text_field($form_data['first_name']);
        if ($current_user->first_name !== $new_first_name) {
            wp_update_user(array('ID' => $user_id, 'first_name' =>
$new_first_name));
        }
    }

    // Email update with safety check (Lines 290-304)
    if (isset($form_data['email']) && !empty($form_data['email'])) {
        $new_email = sanitize_email($form_data['email']);
        $existing_user = get_user_by('email', $new_email);
        if (!$existing_user || $existing_user->ID === $user_id) {
            wp_update_user(array('ID' => $user_id, 'user_email' =>
$new_email));
        }
    }
}
```

- **Purpose:** Updates existing user profiles with new form data
- **Safety:** Prevents email conflicts between users
- **Logging:** Tracks all profile changes

Welcome Email System (Lines 322-375)

PHP

```
private function send_welcome_email($user_id, $email, $password, $first_name)
{
    $subject = 'Welcome to ' . $site_name . ' - Your Account Details';

    $message = "
Hello " . $first_name . ",
```


Your Account Details:

- Email: " . \$email . "
- Password: " . \$password . "
- Login URL: " . \$login_url . "

What's Next:

1. Log in to your account
2. Review your assessment results
3. Schedule a consultation

";

```
$email_sent = wp_mail($email, $subject, $message, $headers);
```

```
// Track email delivery (Lines 370-372)
```

```
update_user_meta($user_id, 'ennu_welcome_email_sent',  
current_time('mysql'));  
}
```

- **Professional Template:** Complete welcome email with instructions
- **Tracking:** Records when welcome email was sent
- **Headers:** Proper from address and content type

Assessment Type Detection (Lines 420-490) - CRITICAL LOGIC

PHP

```
private function detect_assessment_type($form_data) {  
    // Explicit type check (Lines 422-425)  
    if (isset($form_data['assessment_type'])) {  
        return sanitize_text_field($form_data['assessment_type']);  
    }  
  
    // Pattern matching (Lines 430-470)  
    $all_keys_lower = strtolower(implode(' ', array_keys($form_data)));  
  
    // Hair Assessment (Lines 433-441)  
    if (strpos($all_keys_lower, 'hair') !== false ||  
        strpos($all_keys_lower, 'scalp') !== false ||  
        isset($form_data['question_hair_1'])) {  
        return 'hair_assessment';  
    }  
  
    // ED Treatment (Lines 443-450)
```

```

        if (strpos($all_keys_lower, 'ed_treatment') !== false ||
            strpos($all_keys_lower, 'erectile') !== false) {
            return 'ed_treatment_assessment';
        }

        // Fallback to referrer URL (Lines 475-482)
        if (isset($_SERVER['HTTP_REFERER'])) {
            $referrer = strtolower($_SERVER['HTTP_REFERER']);
            if (strpos($referrer, 'hair') !== false) return 'hair_assessment';
        }
    }
}

```

- **Multi-Method Detection:** Form fields, keywords, URL referrer
- **Keyword Matching:** Searches field names for assessment-specific terms
- **Fallback System:** Uses HTTP referrer if form data unclear

Data Verification System (Lines 377-418)

PHP

```

private function verify_user_meta_save($user_id, $assessment_type) {
    $all_meta = get_user_meta($user_id);
    $assessment_meta = array();

    // Filter assessment-specific fields (Lines 382-388)
    foreach ($all_meta as $key => $value) {
        if (strpos($key, 'ennu_' . $clean_assessment_type) === 0) {
            $assessment_meta[$key] = $value;
        }
    }

    // Log verification results (Lines 390-395)
    error_log('ENNU: Found ' . count($assessment_meta) . ' assessment meta
fields');
    foreach ($assessment_meta as $key => $value) {
        error_log('ENNU: ' . $key . ' = ' . $val);
    }
}

```

- **Purpose:** Confirms data was saved to database
- **Debugging:** Logs all saved fields for troubleshooting

- **Interconnection:** Used by admin class to verify field existence

Key Interconnections

- **Main Plugin** → **Form Handler:** AJAX delegation (Line 77)
- **Form Handler** → **Admin Class:** Field naming pattern compatibility
- **Form Handler** → **Database:** User meta storage with standardized keys
- **JavaScript** → **Form Handler:** AJAX submissions and custom endpoint

CRITICAL CLASS: `class-admin.php`

Class Declaration & Constructor (Lines 9-24)

PHP

```
class ENNU_Admin {
    public function __construct() {
        // Admin menu removed per user request (Lines 15-16)
        // add_action( 'admin_menu', array( $this, 'add_admin_menu' ), 10 );
        // add_action( 'admin_init', array( $this, 'settings_init' ), 10 );

        add_action( 'wp_ajax_enu_admin_action', array( $this,
'handle_admin_ajax' ) );

        // User profile integration (Lines 19-23)
        add_action( 'show_user_profile', array( $this,
'show_user_assessment_fields' ), 20 );
        add_action( 'edit_user_profile', array( $this,
'show_user_assessment_fields' ), 20 );
        add_action( 'personal_options_update', array( $this,
'save_user_assessment_fields' ), 10 );
        add_action( 'edit_user_profile_update', array( $this,
'save_user_assessment_fields' ));
    }
}
```

- **Purpose:** Manages WordPress admin interface and user profile integration

- **Key Feature:** Admin menu disabled per user request (Lines 15-16)
- **Priority:** User profile hooks use priority 20 to load after other plugins

USER PROFILE DISPLAY (Lines 175-230) - MOST CRITICAL

PHP

```
public function show_user_assessment_fields($user) {
    if (!current_user_can('edit_user', $user->ID)) {
        return; // Security check
    }

    // Debug: Check for assessment data (Lines 181-190)
    $all_meta = get_user_meta($user->ID);
    $has_assessment_data = false;

    foreach ($all_meta as $key => $value) {
        if (strpos($key, 'ennu_') === 0) {
            $has_assessment_data = true;
            break;
        }
    }

    // Assessment types definition (Lines 210-217)
    $assessment_types = array(
        'hair_assessment' => 'Hair Assessment',
        'ed_treatment_assessment' => 'ED Treatment Assessment',
        'weight_loss_assessment' => 'Weight Loss Assessment',
        'health_assessment' => 'Health Assessment',
        'skin_assessment' => 'Skin Assessment'
    );

    // Display each assessment type (Lines 219-222)
    foreach ($assessment_types as $type => $label) {
        $this->display_assessment_fields($user->ID, $type, $label);
    }
}
```

- **Security:** Capability check before displaying fields
- **Debug Mode:** Shows all user meta keys when no data found
- **Assessment Types:** Hardcoded list of 5 core assessments

- **Interconnection:** Calls `display_assessment_fields()` for each type

FIELD DISPLAY LOGIC (Lines 232-345) - ENHANCED DEBUGGING

PHP

```
private function display_assessment_fields($user_id, $assessment_type,
$assessment_label) {
    // Field naming pattern (Lines 234-237)
    $clean_assessment_type = str_replace('-', '_', $assessment_type);
    $meta_prefix = 'ennu_' . $clean_assessment_type . '_';
    $all_meta = get_user_meta($user_id);

    // Enhanced debugging (Lines 239-253)
    error_log("ENNU Admin Debug: Looking for fields with prefix: " .
$meta_prefix);
    error_log("ENNU Admin Debug: Total user meta fields: " .
count($all_meta));

    $ennu_fields = array();
    foreach ($all_meta as $key => $value) {
        if (strpos($key, 'ennu_') === 0) {
            $ennu_fields[$key] = $value[0];
        }
    }
    error_log("ENNU Admin Debug: Found ENNU fields: " .
print_r(array_keys($ennu_fields), true));

    // Field filtering (Lines 255-270)
    $assessment_fields = array();
    foreach ($all_meta as $key => $value) {
        if (strpos($key, $meta_prefix) === 0 &&
            !strpos($key, '_date') &&
            !strpos($key, '_label') &&
            !strpos($key, '_completion_status') &&
            !strpos($key, '_submission_date') &&
            !strpos($key, '_version')) {

            $field_key = str_replace($meta_prefix, '', $key);
            $field_value = is_array($value) ? $value[0] : $value;


            // Skip empty values (Lines 264-266)
            if (empty($field_value)) {
                continue;
            }

            $assessment_fields[$field_key] = array(
```

```

        'value' => $field_value,
        'date' => get_user_meta($user_id, $key . '_date', true),
        'label' => get_user_meta($user_id, $key . '_label', true),
        'meta_key' => $key
    );
}
}
}

```

-  **CRITICAL:** Field naming must match form handler pattern
- **Filtering:** Excludes metadata fields (_date, _label, etc.)
- **Empty Values:** Skips fields with no data to reduce clutter
- **Debugging:** Comprehensive logging for troubleshooting

✅ SUCCESS DISPLAY (Lines 277-310)

PHP

```

if (!empty($assessment_fields)) {
    echo '<tr><th colspan="2"><h4>' . esc_html($assessment_label) . '</h4>
</th></tr>';

    // Show completion status first (Lines 280-293)
    $completion_status = get_user_meta($user_id, $meta_prefix .
'completion_status', true);
    $submission_date = get_user_meta($user_id, $meta_prefix .
'submission_date', true);

    if ($completion_status || $submission_date) {
        echo '<tr>';
        echo '<th><label>Assessment Status</label></th>';
        echo '<td>';
        if ($completion_status) {
            echo '<strong>' . esc_html(ucfirst($completion_status)) .
'</strong>';
        }
        if ($submission_date) {
            echo ' - Completed: ' . date('M j, Y g:i A',
strtotime($submission_date));
        }
        echo '</td>';
        echo '</tr>';
    }
}

```

```

    }

    // Display individual fields (Lines 295-308)
    foreach ($assessment_fields as $field_key => $field_data) {
        $label = !empty($field_data['label']) ? $field_data['label'] :
ucwords(str_replace('_', ' ', $field_key));
        $date = !empty($field_data['date']) ? ' (Updated: ' . date('M j, Y
g:i A', strtotime($field_data['date'])) . '): ' : '';

        echo '<tr>';
        echo '<th><label for="' . esc_attr($field_data['meta_key']) . '">' .
esc_html($label) . ' (ID: ' . esc_html($field_key) . '): ' . $date . '</label>
</th>';
        echo '<td>';
        echo '<input type="text" id="' . esc_attr($field_data['meta_key']) .
'" name="' . esc_attr($field_data['meta_key']) . '" value="' .
esc_attr($field_data['value']) . '" class="regular-text" readonly />';
        echo '</td>';
        echo '</tr>';
    }
}

```

- **Status First:** Shows completion status and date prominently
- **Field Display:** Read-only inputs with actual values
- **Labels:** Uses saved labels or generates from field names
- **Field IDs:** Shows database field names for debugging

DEBUG DISPLAY (Lines 312-343)

PHP

```

} else {
    // Show debugging info when no fields are found
    echo '<tr><th colspan="2"><h4>' . esc_html($assessment_label) . '</h4>
</th></tr>';
    echo '<tr>';
    echo '<td colspan="2">';
    echo '<p><em>No assessment data found for this assessment type.</em>
</p>';
    echo '<p><strong>Debug Info:</strong> Looking for fields with prefix:
<code>' . esc_html($meta_prefix) . '</code></p>';
}

```

```

// Show all ENNU fields for debugging (Lines 320-335)
$all_ennu_fields = array();
foreach ($all_meta as $key => $value) {
    if (strpos($key, 'ennu_') === 0) {
        $all_ennu_fields[] = $key . ' = ' . (is_array($value) ? $value[0]
: $value);
    }
}

if (!empty($all_ennu_fields)) {
    echo '<details style="margin: 10px 0;">';
    echo '<summary>Debug: All ENNU fields found (click to expand)
</summary>';
    echo '<ul style="max-height: 200px; overflow-y: auto; background:
#f9f9f9; padding: 10px; margin: 10px 0;">';
    foreach ($all_ennu_fields as $field_info) {
        echo '<li><code>' . esc_html($field_info) . '</code></li>';
    }
    echo '</ul>';
    echo '</details>';
} else {
    echo '<p><strong>No ENNU fields found at all.</strong> This suggests
the assessment data was not saved properly.</p>';
}
}

```

- **Purpose:** Comprehensive debugging when no fields found
- **Search Pattern:** Shows exact prefix being searched
- **Field Inventory:** Expandable list of all ENNU fields in database
- **Diagnosis:** Helps identify if data saving or retrieval is the issue

AJAX Handler (Lines 355-395)

PHP

```

public function handle_admin_ajax() {
    try {
        // Security verification (Lines 357-361)
        $nonce = $_POST['ennu_nonce'] ?? $_POST['nonce'] ?? '';
        if ( empty( $nonce ) || ! wp_verify_nonce( $nonce, 'ennu_admin_nonce'
) ) {
            throw new Exception( 'Security verification failed' );
        }
    }
}

```



```

    }

    // Capability check (Lines 363-366)
    if ( ! current_user_can( 'manage_options' ) ) {
        throw new Exception( 'Insufficient permissions' );
    }

    // Action processing (Lines 374-385)
    switch ( $action ) {
        case 'export_data':
            $this->export_assessment_data();
            break;
        case 'delete_submission':
            $this->delete_submission();
            break;
        case 'clear_cache':
            $this->clear_plugin_cache();
            break;
        default:
            throw new Exception( 'Unknown action: ' . $action );
    }
} catch ( Exception $e ) {
    wp_send_json_error( array( 'message' => $e->getMessage() ) );
}
}

```

- **Security:** Full nonce verification and capability checks
- **Error Handling:** Try-catch with proper JSON responses
- **Actions:** Export, delete, cache clearing functionality

Key Interconnections

- **Form Handler** → **Admin Class:** Field naming pattern compatibility (`enmu_[type]_[field]`)
- **Admin Class** → **WordPress:** User profile integration hooks
- **Admin Class** → **Database:** Direct user meta queries for field display
- **JavaScript** → **Admin Class:** AJAX requests for admin actions

Critical Dependencies

- **Field Naming:** Must match form handler pattern exactly
 - **User Capabilities:** Requires `edit_user` permission for profile display
 - **Meta Data:** Depends on form handler saving metadata (`_date`, `_label`)
 - **Assessment Types:** Hardcoded list must match form handler detection
-



CRITICAL CLASS: `class-assessment-shortcodes.php`

Class Declaration & Properties (Lines 23-40)

PHP

```
final class ENNU_Assessment_Shortcodes {  
    /**  
     * Assessment configurations  
     * @var array  
     */  
    private $assessments = array();  
  
    /**  
     * Template cache  
     * @var array  
     */  
    private $template_cache = array();  
  
    public function __construct() {  
        $this->init_assessments();  
        $this->register_shortcodes();  
        $this->setup_hooks();  
    }  
}
```

- **Purpose:** Handles all frontend assessment display via WordPress shortcodes
- **Caching:** Template cache for performance optimization
- **Final Class:** Prevents inheritance for security



ASSESSMENT CONFIGURATIONS (Lines 47-105) - CRITICAL DATA

PHP

```
private function init_assessments() {
    $this->assessments = array(
        'hair_assessment' => array(
            'title' => __( 'Hair Assessment', 'ennu-life' ),
            'description' => __( 'Comprehensive hair health evaluation',
'ennu-life' ),
            'questions' => 10,
            'theme_color' => '#667eea',
            'icon_set' => 'hair'
        ),
        'ed_treatment_assessment' => array(
            'title' => __( 'ED Treatment Assessment', 'ennu-life' ),
            'description' => __( 'Confidential ED treatment evaluation',
'ennu-life' ),
            'questions' => 11,
            'theme_color' => '#f093fb',
            'icon_set' => 'medical'
        ),
        // ... 8 more assessment types
    );
}
```

- **Purpose:** Central configuration for all assessment types
- **Key Data:** Title, description, question count, theme colors, icons
- **Internationalization:** All strings wrapped in `__()` for translation
- **Total Types:** 10 assessment types defined (Lines 49-103)

SHORTCODE REGISTRATION (Lines 112-140)

PHP

```
private function register_shortcodes() {
    // Core assessments only (Lines 114-122)
    $score_assessments = array(
        'hair_assessment' => 'ennu-hair-assessment',
        'ed_treatment_assessment' => 'ennu-ed-treatment-assessment',
        'weight_loss_assessment' => 'ennu-weight-loss-assessment',
        'health_assessment' => 'ennu-health-assessment',
        'skin_assessment' => 'ennu-skin-assessment'
    );
}
```


```

        foreach ( $core_assessments as $assessment_key => $shortcode_tag ) {
            add_shortcode( $shortcode_tag, array( $this,
'render_assessment_shortcode' ) );
            error_log( "ENNU: Registered shortcode [{$shortcode_tag}] for
{$assessment_key}" );
        }

        // Results and thank you pages (Lines 130-140)
        add_shortcode( 'ennu-assessment-results', array( $this,
'render_results_page' ) );

        $thank_you_shortcodes = array(
            'ennu-hair-results' => 'hair_assessment',
            'ennu-ed-results' => 'ed_treatment_assessment',
            // ... more result pages
        );
    }
}

```

-  **CRITICAL:** Only 5 core assessments registered as shortcodes
- **Naming Pattern:** ennu-[assessment]-assessment format
- **Logging:** Each registration logged for debugging
- **Result Pages:** Separate shortcodes for thank you pages

MAIN RENDER METHOD (Lines 175-215) - MOST CRITICAL

PHP

```

public function render_assessment_shortcode( $atts, $content = '', $tag = ''
) {
    // Extract assessment type from tag (Line 177)
    $assessment_type = str_replace( array( 'ennu-', '-' ), array( '', '_' ),
$tag );

    // Validate assessment type (Lines 179-182)
    if ( ! isset( $this->assessments[ $assessment_type ] ) ) {
        return $this->render_error_message( __( 'Invalid assessment type.',
'ennu-life' ) );
    }

    // Parse attributes (Lines 184-190)
    $atts = shortcode_atts( array(

```

```

        'theme' => 'default',
        'show_progress' => 'true',
        'auto_advance' => 'true',
        'cache' => 'true'
    ), $atts, $tag );

    // Check cache (Lines 192-196)
    $cache_key = md5( $assessment_type . serialize( $atts ) );
    if ( $atts['cache'] === 'true' && isset( $this->template_cache[
$cache_key ] ) ) {
        return $this->template_cache[ $cache_key ];
    }

    try {
        // Render assessment (Line 200)
        $output = $this->render_assessment( $assessment_type, $atts );

        // Cache output (Lines 202-205)
        if ( $atts['cache'] === 'true' ) {
            $this->template_cache[ $cache_key ] = $output;
        }

        return $output;
    } catch ( Exception $e ) {
        return $this->render_error_message( __( 'Assessment temporarily
unavailable.', 'ennu-life' ) );
    }
}

```

- **Tag Parsing:** Converts shortcode tag to assessment type
- **Validation:** Ensures assessment type exists in configuration
- **Caching:** MD5-based template caching for performance
- **Error Handling:** Try-catch with user-friendly error messages

TEMPLATE SYSTEM (Lines 217-260)

PHP

```

private function render_assessment( $assessment_type, $atts ) {
    $config = $this->assessments[ $assessment_type ];
    $current_user = wp_get_current_user();

```

```

// Start output buffering (Line 222)
ob_start();

// Include assessment template (Lines 224-229)
$template_file = $this->get_assessment_template( $assessment_type );
if ( file_exists( $template_file ) ) {
    include $template_file;
} else {
    echo $this->render_default_assessment( $assessment_type, $config,
    $atts );
}

return ob_get_clean();
}

private function get_assessment_template( $assessment_type ) {
    $template_name = 'assessment-' . str_replace( '_', '-', $assessment_type
    ) . '.php';

    // Check theme directory first (Lines 243-247)
    $theme_template = get_stylesheet_directory() . '/ennu-life/' .
    $template_name;
    if ( file_exists( $theme_template ) ) {
        return $theme_template;
    }

    // Check plugin templates directory (Lines 249-253)
    $plugin_template = ENNU_LIFE_PLUGIN_PATH . 'templates/' . $template_name;
    if ( file_exists( $plugin_template ) ) {
        return $plugin_template;
    }

    return '';
}

```

- **Template Hierarchy:** Theme templates override plugin templates
- **Output Buffering:** Clean HTML generation
- **Fallback:** Default template if custom template not found

DEFAULT TEMPLATE RENDERING (Lines 262-380) - FRONTEND HTML

PHP

```

private function render_default_assessment( $assessment_type, $config, $atts
) {
    $current_user = wp_get_current_user();
    $nonce = wp_create_nonce( 'ennu_assessment_' . $assessment_type );

    ob_start();
    ?>
    <div class="ennu-assessment ennu-modern-assessment ennu-<?php echo
esc_attr( $assessment_type ); ?>"
        data-assessment="<?php echo esc_attr( $assessment_type ); ?>"
        data-theme="<?php echo esc_attr( $atts['theme'] ); ?>"

        <!-- Assessment Header (Lines 270-285) -->
        <div class="assessment-header">
            <h1 class="assessment-title"><?php echo esc_html(
$config['title'] ); ?></h1>
            <p class="assessment-description"><?php echo esc_html(
$config['description'] ); ?></p>

            <?php if ( $atts['show_progress'] === 'true' ) : ?>
            <div class="progress-container">
                <div class="progress-bar">
                    <div class="ennu-progress-fill" data-progress="0"></div>
                </div>
                <div class="progress-text">
                    <span>Question <span id="currentStep">1</span> of <span
id="totalSteps"><?php echo esc_html( $config['questions'] ); ?></span></span>
                </div>
            </div>
            <?php endif; ?>
        </div>

        <!-- Assessment Form (Lines 287-295) -->
        <form class="assessment-form" data-assessment="<?php echo esc_attr(
$assessment_type ); ?>">
            <?php wp_nonce_field( 'ennu_assessment_' . $assessment_type,
'assessment_nonce' ); ?>
            <input type="hidden" name="action"
value="ennu_submit_assessment">
            <input type="hidden" name="assessment_type" value="<?php echo
esc_attr( $assessment_type ); ?>">

            <!-- Questions Container (Lines 297-300) -->
            <div class="questions-container">
                <?php echo $this->render_assessment_questions(
$assessment_type, $config ); ?>
            </div>

```

- **Data Attributes:** Assessment type and theme for JavaScript
- **Progress Bar:** Conditional display based on attributes
- **Security:** WordPress nonce for form submission
- **Hidden Fields:** Action and assessment type for AJAX processing



CONTACT FORM (Lines 302-350) - AUTO-POPULATION

PHP

```
<!-- Contact Information (Final Step) -->
<div class="question contact-question" data-question="<?php echo esc_attr(
$config['questions'] + 1 ); ?>">
    <h2><?php esc_html_e( 'Get Your Personalized Results', 'ennu-life' ); ?>
</h2>

    <div class="contact-fields">
        <div class="field-group">
            <label for="contact_name">Full Name <span class="required">*</span></label>
            <input type="text"
                id="contact_name"
                name="contact_name"
                value="<?php echo esc_attr( $current_user->display_name );
?>"
                required
                autocomplete="name">
        </div>

        <div class="field-group">
            <label for="contact_email">Email Address <span class="required">*</span></label>
            <input type="email"
                id="contact_email"
                name="contact_email"
                value="<?php echo esc_attr( $current_user->user_email ); ?
>"
                required
                autocomplete="email">
        </div>

        <div class="field-group">
            <label for="contact_phone">Phone Number</label>
            <input type="tel">
```




```

        id="contact_phone"
        name="contact_phone"
        autocomplete="tel">

    </div>
</div>

<button type="submit" class="submit-assessment-btn">
    <span class="btn-text">Get My Results</span>
    <span class="btn-loading" style="display: none;">Processing...</span>
</button>
</div>

```

-  **AUTO-POPULATION:** Pre-fills name and email for logged-in users
- **Accessibility:** Proper labels and autocomplete attributes
- **Validation:** Required fields marked with asterisk
- **Loading States:** Button shows processing state during submission

INLINE STYLES (Lines 380-500) - DYNAMIC THEMING

PHP

```

<style>
.ennu-assessment {
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
    background: #ffffff;
    border-radius: 12px;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
}

.assessment-title {
    font-size: 2.5rem;
    font-weight: 700;
    color: <?php echo esc_attr( $config['theme_color'] ); ?>;
    margin-bottom: 10px;
}

.progress-fill {
    background: linear-gradient(90deg, <?php echo esc_attr(
$config['theme_color'] ); ?>, <?php echo esc_attr( $this-
>adjust_color_brightness( $config['theme_color'], 20 ) ); ?>);

```

```
        transition: width 0.3s ease;
    }

    .options-grid[data-columns="2"] { grid-template-columns: repeat(2, 1fr); }
    .options-grid[data-columns="3"] { grid-template-columns: repeat(3, 1fr); }

    @media (max-width: 768px) {
        .options-grid {
            grid-template-columns: 1fr !important;
        }
    }
</style>
```

- **Dynamic Colors:** Uses assessment-specific theme colors
- **Responsive Design:** Mobile-first grid layouts
- **Modern CSS:** Flexbox, grid, transitions, gradients
- **Performance:** Inline styles to avoid additional HTTP requests

Key Interconnections

- **Main Plugin → Shortcode Class:** Shortcode registration and rendering
- **Shortcode Class → Form Handler:** Form submission via AJAX
- **Shortcode Class → JavaScript:** Data attributes for frontend behavior
- **Shortcode Class → Templates:** Template hierarchy system

Critical Dependencies

- **Assessment Config:** Must match form handler detection patterns
 - **Template Files:** Fallback to default if custom templates missing
 - **JavaScript:** Requires `ennu-assessment-modern.js` for functionality
 - **CSS:** Requires `ennu-assessment-modern.css` for styling
-



OTHER CLASSES: Quick Overview

class-database.php (Lines 1-150)

PHP

```
class ENNU_Life_Database {
    private static $instance = null;

    public static function get_instance() {
        if ( null === self::$instance ) {
            self::$instance = new self();
        }
        return self::$instance;
    }

    public function create_tables() {
        // Creates ennu_assessments table
        // Handles database schema updates
    }
}
```

- **Purpose:** Database operations and table management
- **Pattern:** Singleton for single database connection
- **Key Methods:** `create_tables()` , `get_assessments()` , `save_assessment()`

class-assessment-cpt.php (Lines 1-200)

PHP

```
class ENNU_Assessment_CPT {
    public function __construct() {
        add_action( 'init', array( $this, 'register_post_type' ) );
        add_action( 'add_meta_boxes', array( $this, 'add_meta_boxes' ) );
    }

    public function register_post_type() {
        register_post_type( 'enuu_assessment', array(
            'public' => true,
            'supports' => array( 'title', 'editor', 'custom-fields' ),
            'menu_icon' => 'dashicons-clipboard'
        ) );
    }
}
```

```
}  
}
```

- **Purpose:** Custom post type for assessment management
- **Features:** Admin interface for creating assessments
- **Meta Boxes:** Custom fields for assessment configuration

class-debug-logger.php (Lines 1-100)

PHP

```
class ENNU_Debug_Logger {  
    private static $log_file;  
  
    public static function log( $message, $level = 'INFO' ) {  
        if ( ! WP_DEBUG ) {  
            return;  
        }  
  
        $timestamp = current_time( 'Y-m-d H:i:s' );  
        $log_entry = "[{$timestamp}] [{$level}] {$message}" . PHP_EOL;  
  
        error_log( $log_entry, 3, self::get_log_file() );  
    }  
}
```

- **Purpose:** Enhanced logging system for debugging
- **Conditional:** Only logs when WP_DEBUG is enabled
- **File Logging:** Writes to dedicated log file

Assets Documentation

 **CRITICAL FILE:** ennu-assessment-modern.js

File Header & Initialization (Lines 1-25)

JavaScript

```
/**
 * ENNU Life Modern Assessment JavaScript
 * Browser Compatibility: ES6+ (Chrome 51+, Firefox 54+, Safari 10+, Edge
15+)
 */

(function($) {
    'use strict';

    // Modern Assessment Controller
    window.ENNUModernAssessment = {
        currentStep: 1,
        totalSteps: 0,
        assessmentData: {},
        autoProgressDelay: 1500, // 1.5 seconds
        autoProgressTimer: null,

        // Initialize the modern assessment
        init: function() {
            this.bindEvents();
            this.initializeAssessment();
            this.updateProgress();
            console.log('ENNU Modern Assessment initialized');
        }
    };
})(jQuery);
```

- **Purpose:** Main JavaScript controller for assessment frontend
- **Global Object:** `window.ENNUModernAssessment` for external access
- **Auto-Progress:** 1.5-second delay for automatic question progression
- **Data Storage:** `assessmentData` object stores all user responses

EVENT BINDING (Lines 27-40) - CRITICAL SETUP

JavaScript

```

bindEvents: function() {
    // Answer option selection (Line 29)
    $(document).on('click', '.ennu-answer-option',
this.handleAnswerSelection.bind(this));

    // Navigation buttons (Lines 31-33)
    $(document).on('click', '.nav-btn.next-btn', this.nextStep.bind(this));
    $(document).on('click', '.nav-btn.prev-btn', this.prevStep.bind(this));
    $(document).on('click', '.submit-assessment-btn',
this.submitAssessment.bind(this));

    // DOB dropdown changes (Line 35)
    $(document).on('change', '.dob-dropdown',
this.handleDOBChange.bind(this));

    // Keyboard navigation (Line 37)
    $(document).on('keydown', this.handleKeyboardNavigation.bind(this));
}

```

- **Event Delegation:** Uses `$(document).on()` for dynamic content
- **Binding:** All methods bound to maintain `this` context
- **Key Events:** Click, change, keydown for complete interaction

ASSESSMENT INITIALIZATION (Lines 42-65)

JavaScript

```

initializeAssessment: function() {
    console.log('Initializing assessment...');

    // Count total steps (Lines 45-47)
    this.totalSteps = $('.ennu-question-step').length;
    console.log('Total steps found:', this.totalSteps);

    // Reset to first step (Line 49)
    this.currentStep = 1;

    // Ensure all steps are hidden except first (Lines 51-54)
    $('.ennu-question-step').removeClass('active').hide();
    this.showStep(1);

    // Update step counter (Lines 59-60)
    $('#currentStep').text(this.currentStep);
}

```

```
$('#totalSteps').text(this.totalSteps);  
}
```

- **Step Counting:** Finds all `.ennu-question-step` elements
- **State Reset:** Ensures clean initialization state
- **UI Updates:** Updates progress indicators immediately

ANSWER SELECTION HANDLER (Lines 67-105) - MOST CRITICAL

JavaScript


```
handleAnswerSelection: function(e) {  
    e.preventDefault();  
  
    var $option = $(e.currentTarget);  
    var $question = $option.closest('.ennu-question-step');  
    var questionType = $question.data('question-type') || 'single';  
    var questionKey = $question.data('question-key');  
    var answerValue = $option.data('value');  
  
    // Handle different question types (Lines 76-95)  
    if (questionType === 'single') {  
        // Single choice - clear other selections first  
        $question.find('.ennu-answer-option').removeClass('selected');  
  
        // Select this option  
        $option.addClass('selected');  
        $option.find('input[type="radio"]').prop('checked', true);  
  
        // Store answer  
        this.assessmentData[questionKey] = answerValue;  
  
        // Auto-progress after delay  
        this.startAutoProgress();  
    } else if (questionType === 'multiple') {  
        // Multiple choice - toggle selection  
        $option.toggleClass('selected');  
  
        // Collect all selected values  
        var selectedValues = [];  
        $question.find('.ennu-answer-option.selected').each(function() {  
            selectedValues.push($(this).data('value'));  
        });  
    }  
}
```

```

        this.assessmentData[questionKey] = selectedValues;
        this.showNavigationButtons();
    }

    // Add selection animation (Line 103)
    this.animateSelection($option);
}

```

-  **CRITICAL:** Handles both single and multiple choice questions
- **Data Storage:** Saves answers to `assessmentData` object
- **Auto-Progress:** Single choice triggers automatic progression
- **Visual Feedback:** Adds/removes `selected` class and animations

AUTO-PROGRESSION SYSTEM (Lines 107-135)

JavaScript

```

startAutoProgress: function() {
    // Clear existing timer (Lines 109-112)
    if (this.autoProgressTimer) {
        clearTimeout(this.autoProgressTimer);
    }

    // Show auto-progress indicator (Line 114)
    this.showAutoProgressIndicator();

    // Set timer for auto-progression (Lines 116-119)
    this.autoProgressTimer = setTimeout(() => {
        this.nextStep();
    }, this.autoProgressDelay);
}

showAutoProgressIndicator: function() {
    var $indicator = $('enu-auto-progress');
    if ($indicator.length === 0) {
        $indicator = $(`
            <div class="enu-auto-progress">
                <div class="enu-auto-progress-text">Moving to next
question...</div>
                <div class="enu-auto-progress-bar">
                    <div class="enu-auto-progress-fill"></div>

```



```

        </div>
    </div>
    `);
    $('.ennu-question-step.active').append($indicator);
}
$indicator.show();
}

```

- **Timer Management:** Clears existing timers to prevent conflicts
- **Visual Indicator:** Shows progress bar during auto-progression
- **User Feedback:** Clear messaging about automatic advancement

STEP NAVIGATION (Lines 180-220) - CORE FUNCTIONALITY

JavaScript

```

nextStep: function() {
    console.log('nextStep called. Current step:', this.currentStep, 'Total
steps:', this.totalSteps);

    // Validate current step before progression (Lines 183-187)
    if (!this.validateCurrentStep()) {
        console.log('Validation failed, cannot proceed to next step');
        return;
    }

    if (this.currentStep < this.totalSteps) {
        this.hideAutoProgressIndicator();
        this.currentStep++;
        console.log('Moving to step:', this.currentStep);
        this.showStep(this.currentStep);
        this.updateProgress();
    } else {
        console.log('Assessment complete, submitting...');
        this.submitAssessment();
    }
}

showStep: function(stepNumber) {
    console.log('showStep called with stepNumber:', stepNumber);

    // Hide all steps first (Line 300)
    $('.ennu-question-step').removeClass('active').hide();
}

```

```

// Find target step - multiple selectors for compatibility (Lines 302-308)
var $targetStep = $('`enu-question-step[data-step="${stepNumber}"]`');

// Fallback: if data-step selector fails, use nth-child
if ($targetStep.length === 0) {
    $targetStep = $('.enu-question-step').eq(stepNumber - 1);
    console.log('Using fallback selector for step:', stepNumber);
}

if ($targetStep.length > 0) {
    // Show target step with animation (Line 313)
    $targetStep.addClass('active').fadeIn(300);

    // Check for contact form auto-population (Lines 315-320)
    var $contactForm = $targetStep.find('.contact-info-form');
    if ($contactForm.length > 0) {
        this.handleContactFormStep($targetStep);
    }

    this.updateNavigation();
}
}

```

- **Validation:** Ensures current step is complete before advancing
- **Dual Selectors:** Uses data attributes with fallback to nth-child
- **Animation:** Smooth fade transitions between steps
- **Contact Form:** Special handling for auto-populated contact fields

VALIDATION SYSTEM (Lines 222-280)

JavaScript

```

validateCurrentStep: function() {
    var $currentStep = $(".enu-question-step.active");

    // Check for DOB dropdowns first (Lines 225-240)
    var $dobDropdowns = $currentStep.find('.dob-dropdown');
    if ($dobDropdowns.length > 0) {
        var $monthSelect = $currentStep.find('.dob-month');
        var $daySelect = $currentStep.find('.dob-day');
        var $yearSelect = $currentStep.find('.dob-year');
    }
}

```

```

    var month = $monthSelect.val();
    var day = $daySelect.val();
    var year = $yearSelect.val();

    if (month && day && year) {
        return true; // DOB is complete
    } else {
        this.showValidationError("Please complete your date of birth
before continuing.");
        return false;
    }
}

// Check for contact information form (Lines 242-270)
var $contactForm = $currentStep.find('.contact-info-form');
if ($contactForm.length > 0) {
    var $requiredFields = $contactForm.find('input[required]');
    var allFieldsFilled = true;

    $requiredFields.each(function() {
        var value = $(this).val().trim();
        if (!value) {
            allFieldsFilled = false;
            return false; // Break out of each loop
        }
    });

    if (allFieldsFilled) {
        return true; // Contact form is complete
    } else {
        this.showValidationError("Please fill in all required fields");
        return false;
    }
}

// Check for radio button selections (Lines 272-277)
var $selectedOption = $currentStep.find("input[type=\"radio\"]:checked");
if ($selectedOption.length === 0) {
    this.showValidationError("Please select an answer before
continuing.");
    return false;
}

return true;
}

```

- **Multi-Type Validation:** DOB dropdowns, contact forms, radio buttons
- **Required Fields:** Checks all required inputs in contact forms
- **Error Messages:** User-friendly validation feedback
- **Early Return:** Stops validation on first failure

CONTACT FORM AUTO-POPULATION (Lines 325-390)

JavaScript

```
handleContactFormStep: function($step) {
    var $contactForm = $step.find('.contact-info-form');
    var $autoPopulatedFields = $contactForm.find('input[data-auto-populated="true"]');

    if ($autoPopulatedFields.length > 0) {
        console.log('Found', $autoPopulatedFields.length, 'auto-populated fields');

        // Check if all required fields are filled (Lines 332-342)
        var $requiredFields = $contactForm.find('input[required]');
        var allFieldsFilled = true;

        $requiredFields.each(function() {
            var value = $(this).val().trim();
            if (!value) {
                allFieldsFilled = false;
                return false;
            }
        });

        if (allFieldsFilled) {
            console.log('All contact fields are pre-filled and valid');

            // Add visual indicator (Lines 346-351)
            $autoPopulatedFields.each(function() {
                $(this).css({
                    'border-color': '#28a745',
                    'background-color': '#f8fff9'
                });
            });


            // Show helpful message (Lines 353-357)
            var $notice = $contactForm.find('.user-info-notice');
```

```

        if ($notice.length > 0) {
            $notice.append('<p>All fields are ready - you can proceed
immediately</p>');
        }
    }
}

// Add real-time validation (Lines 365-368)
$contactForm.find('input').on('input blur', function() {
    ENNUModernAssessment.validateContactField($(this));
});
}

```

-  **AUTO-POPULATION:** Detects and handles pre-filled fields for logged-in users
- **Visual Feedback:** Green borders for valid auto-populated fields
- **Real-time Validation:** Input/blur events for immediate feedback
- **User Experience:** Helpful messages about pre-filled data



FORM SUBMISSION (Lines 550-620) - MOST CRITICAL

JavaScript

```

submitAssessment: function() {
    console.log('ENNU v21.0: Starting assessment submission...');

    // Show loading state (Line 554)
    this.showLoadingState();

    // Collect contact form data (Line 556)
    this.collectContactFormData();

    // Get assessment type (Lines 558-560)
    var assessmentType = this.getAssessmentTypeHardcoded();
    console.log('ENNU: Assessment type for submission:', assessmentType);

    // Create POST data (Lines 567-575)
    var postData = {
        action: 'ennu_form_submit',
        assessment_type: assessmentType
    };

    // Add all assessment data to post data

```


```

    for (var key in this.assessmentData) {
        postData[key] = this.assessmentData[key];
    }

    // Submit via AJAX with fallback (Lines 580-600)
    $.ajax({
        url: '/wp-admin/admin-ajax.php',
        type: 'POST',
        data: postData,
        success: this.handleSubmissionSuccess.bind(this),
        error: function(xhr, status, error) {
            console.log('ENNU: WordPress AJAX failed, trying custom
endpoint...');

            // Try custom endpoint as fallback
            $.ajax({
                url: '/ennu-submit/',
                type: 'POST',
                data: postData,
                success: this.handleSubmissionSuccess.bind(this),
                error: this.handleSubmissionError.bind(this)
            }).bind(this));
        }.bind(this)
    });
}

```

-  **CRITICAL:** Main submission handler with dual AJAX system
- **Data Collection:** Calls comprehensive data collection method
- **Assessment Type:** Hardcoded detection from URL
- **Fallback System:** WordPress AJAX → Custom endpoint if failed
- **Error Handling:** Comprehensive error catching and logging

ASSESSMENT TYPE DETECTION (Lines 650-680)

JavaScript

```

getAssessmentTypeHardcoded: function() {
    var url = window.location.href.toLowerCase();
    console.log('ENNU: Detecting assessment type from URL:', url);

    // Simple hardcoded detection (Lines 654-670)

```

```

    if (url.indexOf('hair') !== -1) {
        console.log('ENNU: Detected HAIR assessment');
        return 'hair_assessment';
    }
    if (url.indexOf('ed') !== -1 || url.indexOf('erectile') !== -1) {
        console.log('ENNU: Detected ED assessment');
        return 'ed_treatment_assessment';
    }
    if (url.indexOf('weight') !== -1) {
        console.log('ENNU: Detected WEIGHT LOSS assessment');
        return 'weight_loss_assessment';
    }
    if (url.indexOf('skin') !== -1) {
        console.log('ENNU: Detected SKIN assessment');
        return 'skin_assessment';
    }
    if (url.indexOf('health') !== -1) {
        console.log('ENNU: Detected HEALTH assessment');
        return 'health_assessment';
    }

    console.log('ENNU: Could not detect assessment type, defaulting to hair_assessment');
    return 'hair_assessment'; // Default fallback
}

```

- **URL-Based Detection:** Matches form handler detection logic
- **Keyword Matching:** Simple string search in URL
- **Fallback:** Defaults to hair_assessment if no match
- **Logging:** Comprehensive console logging for debugging



COMPREHENSIVE DATA COLLECTION (Lines 850-950)

JavaScript

```

collectContactFormData: function() {
    console.log('ENNU: Starting comprehensive data collection...');

    var fieldsCollected = 0;
    var fieldsSkipped = 0;

    // Method 1: Contact form fields (Lines 857-875)

```

```

var $contactForm = $('#contact-info-form');
if ($contactForm.length > 0) {
    $contactForm.find('input, select, textarea').each((index, element) =>
    {
        var $field = $(element);
        var fieldName = $field.attr('name');
        var fieldValue = $field.val();

        if (fieldName && fieldValue && fieldValue.trim() !== '') {
            this.assessmentData[fieldName] = fieldValue.trim();
            fieldsCollected++;
            console.log('ENNU: Collected contact field:', fieldName, '=',
fieldValue.trim());
        }
    });
}

// Method 2: All assessment steps (Lines 877-920)
$('#ennu-question-step').each(function(stepIndex) {
    var $step = $(this);
    var stepNumber = stepIndex + 1;

    // Collect radio button selections
    $step.find('input[type="radio"]:checked').each(function() {
        var fieldName = $(this).attr('name');
        var fieldValue = $(this).val();

        if (fieldName && fieldValue) {
            ENNUModernAssessment.assessmentData[fieldName] = fieldValue;
            fieldsCollected++;
            console.log('ENNU: Collected radio field (step ' + stepNumber
+ '):', fieldName, '=', fieldValue);
        }
    });

    // Collect text inputs, selects, checkboxes...
});

// Method 3: DOB fields (Lines 922-935)
var $dobMonth = $('#dob-month');
var $dobDay = $('#dob-day');
var $dobYear = $('#dob-year');


if ($dobMonth.length && $dobDay.length && $dobYear.length) {
    var month = $dobMonth.val();
    var day = $dobDay.val();
    var year = $dobYear.val();
}

```



```
        if (month && day && year) {
            this.assessmentData['date_of_birth'] = month + '/' + day + '/' +
year;
            fieldsCollected += 4;
        }
    }

    console.log('ENNU: Fields collected:', fieldsCollected);
    console.log('ENNU: Total assessment data fields:',
Object.keys(this.assessmentData).length);
}
```

-  **COMPREHENSIVE:** 4 different collection methods for complete data capture
- **Field Types:** Radio, text, select, checkbox, DOB dropdowns
- **Step-by-Step:** Iterates through all assessment steps
- **Validation:** Checks for required fields before submission
- **Logging:** Detailed logging for debugging data collection issues

Key Interconnections

- **PHP Form Handler** ← **JavaScript:** AJAX submission with matching field names
- **Shortcode Class** → **JavaScript:** Data attributes for configuration
- **CSS** ← **JavaScript:** Dynamic class manipulation for styling
- **WordPress AJAX** ← **JavaScript:** Standard WordPress AJAX with custom fallback

Critical Dependencies

- **jQuery:** Required for DOM manipulation and AJAX
 - **CSS Classes:** Depends on specific CSS classes for functionality
 - **HTML Structure:** Requires specific data attributes and element structure
 - **Form Handler:** Must match field naming patterns for data saving
-



CRITICAL FILE: ennu-assessment-modern.css

Container & Layout (Lines 6-18)

CSS

```
.ennu-modern-assessment {
  font-family: 'Inter', -apple-system, BlinkMacSystemFont, 'Segoe UI',
  Roboto, sans-serif;
  background: linear-gradient(135deg, #f8fafc 0%, #e2e8f0 50%, #f1f5f9
  100%);
  min-height: 100vh;
  color: #1e293b;
  padding: 2rem 0;
}

.ennu-assessment-container {
  max-width: 900px;
  margin: 0 auto;
  padding: 0 1.5rem;
}
```

- **Modern Typography:** Inter font with system font fallbacks
- **Gradient Background:** Subtle 135-degree gradient for visual appeal
- **Responsive Container:** 900px max-width with auto centering



Progress Bar System (Lines 20-55)

CSS

```
.ennu-progress-bar {
  width: 100%;
  height: 8px;
  background: #e2e8f0;
  border-radius: 4px;
  overflow: hidden;
  margin-bottom: 1rem;
}

.ennu-progress-fill {
  height: 100%;
  background: linear-gradient(135deg, #3b82f6 0%, #1d4ed8 100%);
}
```

```

    transition: width 0.8s cubic-bezier(0.4, 0, 0.2, 1);
  }

  .ennu-progress-fill::after {
    content: '';
    background: linear-gradient(90deg, transparent, rgba(255,255,255,0.3),
transparent);
    animation: shimmer 2s infinite;
  }

  @keyframes shimmer {
    0% { transform: translateX(-100%); }
    100% { transform: translateX(100%); }
  }

```

- **Smooth Animation:** 0.8s cubic-bezier transition for progress updates
- **Shimmer Effect:** Continuous shimmer animation on progress bar
- **Blue Gradient:** Professional blue gradient for progress indication

Question Cards (Lines 57-85)

CSS

```

.ennu-question-card {
  background: white;
  border-radius: 20px;
  padding: 3rem;
  box-shadow: 0 10px 25px -5px rgba(0, 0, 0, 0.1);
  opacity: 0;
  animation: quickFadeIn 0.3s ease-out forwards;
}

@keyframes quickFadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}

.ennu-question-title {
  font-size: 2rem;
  font-weight: 800;
  color: #1e293b;
  text-align: center;
}

```

```
    line-height: 1.2;
}
```

- **Card Design:** White background with 20px border radius
- **Fade Animation:** 0.3s fade-in for smooth question transitions
- **Typography:** Bold 2rem titles with proper line height

Answer Options Grid (Lines 87-115)

CSS

```
.ennu-answer-options {
  display: grid;
  gap: 1.5rem;
  margin-bottom: 2rem;
}

.ennu-answer-options.grid-2 {
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
}

.ennu-answer-options.grid-3 {
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
}

.ennu-answer-options.grid-4 {
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
}

/* Hide all radio buttons and checkboxes */
input[type="radio"],
input[type="checkbox"] {
  display: none !important;
  visibility: hidden !important;
  opacity: 0 !important;
  position: absolute !important;
  left: -9999px !important;
}
```

- **Responsive Grid:** Auto-fit columns with minimum widths
- **Grid Variants:** 2, 3, 4 column layouts with different min-widths

- 🚨 **CRITICAL:** Completely hides native form inputs for custom styling

✨ Answer Option Cards (Lines 125-150)

CSS

```
.ennu-answer-option {
  background: #f8fafc;
  border: 2px solid #e2e8f0;
  border-radius: 16px;
  padding: 2rem;
  cursor: pointer;
  transition: all 0.15s ease;
  text-align: center;
}

.ennu-answer-option:hover {
  border-color: #3b82f6;
  background: #eff6ff;
  transform: translateY(-2px);
  box-shadow: 0 8px 25px -5px rgba(59, 130, 246, 0.15);
}

.ennu-answer-option.selected {
  border-color: #3b82f6;
  background: linear-gradient(135deg, #3b82f6 0%, #1d4ed8 100%);
  color: white;
  transform: scale(1.02);
}
```

- **Interactive States:** Hover, selected, and default states
- **Smooth Transitions:** 0.15s transitions for all properties
- **Visual Feedback:** Transform, shadow, and color changes
- **Selected State:** Blue gradient background with white text

📱 Responsive Design (Lines 200-250)

CSS

```
@media (max-width: 768px) {
  .ennu-answer-options {
```

```
        grid-template-columns: 1fr !important;
    }

    .ennu-question-card {
        padding: 2rem 1.5rem;
        border-radius: 16px;
    }

    .ennu-question-title {
        font-size: 1.5rem;
    }

    .ennu-answer-option {
        padding: 1.5rem;
    }
}

@media (max-width: 480px) {
    .ennu-assessment-container {
        padding: 0 1rem;
    }

    .ennu-question-card {
        padding: 1.5rem 1rem;
    }
}
```

- **Mobile-First:** Single column layout on mobile devices
- **Reduced Padding:** Smaller padding for mobile screens
- **Typography Scaling:** Smaller font sizes for mobile readability

Key Interconnections

- **JavaScript** → **CSS:** Class manipulation for state changes
 - **Shortcode HTML** → **CSS:** Specific class structure required
 - **Progress System** → **CSS:** Width manipulation for progress bar
 - **Animation System** → **CSS:** CSS animations triggered by JavaScript
-

COMPREHENSIVE INTERCONNECTIONS & CRITICAL INSIGHTS

PLUGIN ARCHITECTURE OVERVIEW

Core Data Flow

Plain Text

Frontend Assessment → JavaScript Collection → AJAX Submission → Form Handler
→ Database Storage → Admin Display

Critical File Dependencies

1. **ennu-life-plugin.php** (Lines 170-190) → Loads all classes in specific order
2. **class-form-handler.php** (Lines 130-145) → Saves data with **ennu_[type]_[field]** pattern
3. **class-admin.php** (Lines 234-270) → Retrieves data using same naming pattern
4. **ennu-assessment-modern.js** (Lines 850-950) → Collects data for submission
5. **class-assessment-shortcodes.php** (Lines 262-380) → Renders frontend HTML

CRITICAL INTERCONNECTION POINTS


1. Field Naming Pattern (MOST CRITICAL)

PHP

```
// Form Handler (Line 130)
$meta_key = 'ennu_' . $clean_assessment_type . '_' . $clean_key;

// Admin Class (Line 236)
$meta_prefix = 'ennu_' . $clean_assessment_type . '_';
```

```
// JavaScript (Line 658)
return 'hair_assessment'; // Must match PHP detection
```

-  **CRITICAL:** All three components must use identical naming patterns
- **Failure Point:** Mismatch causes admin display to show "Assessment response (read-only)"

2. Assessment Type Detection Chain

JavaScript

```
// JavaScript (Lines 654-670)
if (url.indexOf('hair') !== -1) return 'hair_assessment';

// Form Handler (Lines 433-441)
if (strpos($all_keys_lower, 'hair') !== false) return 'hair_assessment';

// Admin Class (Lines 214-217)
'hair_assessment' => 'Hair Assessment'
```

- **Consistency:** All three use same detection logic
- **Fallback:** JavaScript defaults to 'hair_assessment', PHP to 'general_assessment'

3. AJAX Security Chain

PHP

```
// Main Plugin (Line 77) - NO SECURITY CHECKS
add_action( 'wp_ajax_nopriv_enu_form_submit', array( $this,
'ajax_form_submit' ) );

// Form Handler (Line 77) - NO SECURITY CHECKS
// Remove all security checks for maximum compatibility

// JavaScript (Lines 580-600) - Dual AJAX System
$.ajax({ url: '/wp-admin/admin-ajax.php' }) // Primary
$.ajax({ url: '/enu-submit/' }) // Fallback
```

-  **SECURITY REMOVED:** All nonce checks disabled for compatibility

- **Dual System:** WordPress AJAX with custom endpoint fallback

DATA FLOW ANALYSIS

Frontend to Backend Flow

1. **User Interaction** (CSS Lines 125-150) → Visual feedback on option selection
2. **JavaScript Collection** (JS Lines 67-105) → Stores in `assessmentData` object
3. **Validation** (JS Lines 222-280) → Ensures required fields completed
4. **Submission** (JS Lines 550-620) → AJAX to form handler
5. **Processing** (PHP Lines 106-145) → Saves to user meta with standardized keys
6. **Response** (PHP Lines 160-170) → Success response with redirect URL

Backend to Frontend Display

1. **Admin Request** (Admin Lines 175-230) → User profile page load
2. **Field Retrieval** (Admin Lines 234-270) → Searches for `ennu_[type]_` prefix
3. **Data Processing** (Admin Lines 277-310) → Formats for display
4. **HTML Output** (Admin Lines 295-308) → Read-only input fields
5. **Debug Info** (Admin Lines 312-343) → Troubleshooting information

FRONTEND ARCHITECTURE

Component Interaction

Plain Text

Shortcode → HTML Template → CSS Styling → JavaScript Behavior → AJAX Submission

Critical CSS-JS Interactions

- `.ennu-answer-option` (CSS Line 125) ↔ `handleAnswerSelection` (JS Line 67)
- `.selected` class (CSS Line 145) ↔ `addClass('selected')` (JS Line 82)
- `.ennu-progress-fill` (CSS Line 32) ↔ `updateProgress()` (JS Line 500)

Auto-Population System

PHP

```
// Shortcode (Lines 315-325)
value="<?php echo esc_attr( $current_user->display_name ); ?>"

// JavaScript (Lines 325-390)
handleContactFormStep($targetStep) // Detects auto-populated fields

// Form Handler (Lines 189-197)
if ($user_id) { $this->update_logged_in_user_profile($user_id, $form_data); }
```



CONFIGURATION DEPENDENCIES

Assessment Type Registry

PHP

```
// Shortcode Class (Lines 49-103) - 10 assessment types defined
// Form Handler (Lines 433-490) - Detection patterns for 5 core types
// Admin Class (Lines 210-217) - Display names for 5 core types
// JavaScript (Lines 654-670) - URL detection for 5 core types
```

- **Mismatch Risk:** Shortcode defines 10 types, but only 5 are fully supported

WordPress Integration Points

PHP

```
// Main Plugin (Lines 252-280) - Hook registration
add_action( 'wp_enqueue_scripts', array( $this, 'enqueue_frontend_assets' ) )
```

```
);  
add_action( 'show_user_profile', array( $this, 'show_user_assessment_fields'  
) );  
  
// Asset Loading (Lines 400-480) - Conditional loading  
if ( ! $this->should_load_assets() ) return;
```



CRITICAL FAILURE POINTS

1. Field Naming Mismatch

- **Symptom:** Admin shows "Assessment response (read-only)"
- **Cause:** Form handler saves as `ennu_hair_assessment_email` , admin looks for `ennu_hair_assessment_email`
- **Fix:** Ensure identical naming patterns (Lines documented above)

2. JavaScript Initialization Failure

- **Symptom:** Assessment questions don't display
- **Cause:** Script runs before HTML is ready
- **Fix:** `$(document).ready()` wrapper (JS Line 1000)

3. AJAX Submission Failure

- **Symptom:** Form submission fails silently
- **Cause:** WordPress AJAX endpoint issues
- **Fix:** Dual AJAX system with custom endpoint fallback (JS Lines 580-600)

4. Assessment Type Detection Failure

- **Symptom:** Data saved under wrong assessment type
- **Cause:** URL doesn't contain expected keywords

- **Fix:** Hardcoded detection with fallback (JS Lines 650-680)

PERFORMANCE OPTIMIZATIONS

Asset Loading Strategy

PHP

```
// Main Plugin (Lines 450-480)
private function should_load_assets() {
    // Only load on pages with ENNU shortcodes
    if ( $this->has_enu_shortcode( $post->post_content ) ) return true;
}
```

Template Caching

PHP

```
// Shortcode Class (Lines 192-196)
$cache_key = md5( $assessment_type . serialize( $atts ) );
if ( isset( $this->template_cache[ $cache_key ] ) ) return $this->template_cache[ $cache_key ];
```

Database Optimization

PHP

```
// Form Handler (Lines 147-151) - Metadata for admin display
update_user_meta($user_id, $meta_key . '_date', current_time('mysql'));
update_user_meta($user_id, $meta_key . '_label', $field_label);
```

DEBUGGING STRATEGIES

Console Logging Chain

1. **JavaScript:** `console.log('ENNU: ...')` throughout submission process

2. **PHP:** `error_log('ENNU: ...')` in form handler and admin class
3. **Admin Debug:** Enhanced debugging in v22.4 (Admin Lines 239-253)

Common Debug Scenarios

- **No Questions Display:** Check JavaScript console for initialization errors
- **Submission Fails:** Check network tab for AJAX response
- **Admin Shows No Data:** Check error logs for field naming issues
- **Wrong Assessment Type:** Check URL detection logic



MAINTENANCE CHECKLIST

When Adding New Assessment Types

1. Update shortcode registration (Shortcode Lines 114-122)
2. Add detection pattern (Form Handler Lines 433-490)
3. Add admin display name (Admin Lines 210-217)
4. Add JavaScript URL detection (JS Lines 654-670)
5. Test complete data flow

When Modifying Field Names

1. Update form HTML structure
2. Verify JavaScript collection (JS Lines 850-950)
3. Test form handler saving (PHP Lines 106-145)
4. Confirm admin display (Admin Lines 234-270)



PLUGIN STRENGTHS

Robust Error Handling

- Try-catch blocks throughout JavaScript and PHP
- Fallback systems for AJAX and template loading
- Comprehensive logging for debugging

User Experience

- Auto-progression for smooth flow
- Auto-population for logged-in users
- Responsive design for all devices
- Visual feedback for all interactions

WordPress Integration

- Proper hook usage and security practices
- Template hierarchy support
- Asset optimization and conditional loading
- User profile integration

Scalability

- Modular class structure
- Template caching system
- Database optimization
- Extensible assessment type system



CONCLUSION

This ENNU Life plugin is a sophisticated assessment system with **5 critical files** working in perfect harmony:

1. **ennu-life-plugin.php** - Orchestrates everything
2. **class-form-handler.php** - Processes and saves data
3. **class-admin.php** - Displays data in WordPress admin
4. **ennu-assessment-modern.js** - Handles frontend interactions
5. **class-assessment-shortcodes.php** - Renders frontend HTML

The **most critical interconnection** is the field naming pattern that must be identical across all components. The plugin uses a **dual AJAX system** for maximum compatibility and has **comprehensive debugging** built-in.

Security has been intentionally removed from AJAX handlers for compatibility, and the system includes **auto-population for logged-in users** and **responsive design** for all devices.

Understanding these interconnections is essential for maintaining and extending the plugin successfully.
