Admin.php

```php
<?php
namespace Elementor\Core\Admin;

use Elementor\Api;
use Elementor\Beta_Testers;
use Elementor\App\Modules\Onboarding\Module as Onboarding_Module;
use Elementor\Core\Base\App;
use Elementor\Core\Upgrade\Manager as Upgrade_Manager;
use Elementor\Core\Utils\Assets_Config_Provider;
use Elementor\Core\Utils\Collection;
use Elementor\Plugin;
use Elementor\Settings;
use Elementor\User;
use Elementor\Utils;
use Elementor\Core\Utils\Hints;

if ( ! defined( 'ABSPATH' ) ) {
    exit; // Exit if accessed directly.
}

class Admin extends App {

    private $menus = [];

    /**
     * Get module name.
     *
     * Retrieve the module name.
     *
     * @since 2.3.0
     * @access public
     *
     * @return string Module name.
     */
    public function get_name() {
        return 'admin';
    }

    /**
     * @since 2.2.0
     * @access public
     */
    public function maybe_redirect_to_getting_started() {
        if ( ! get_transient( 'elementor_activation_redirect' ) ) {
            return;
        }

        if ( wp_doing_ajax() ) {
            return;
        }

        delete_transient( 'elementor_activation_redirect' );

        if ( is_network_admin() || isset( $_GET['activate-multi'] ) ) {
            return;
        }

        $already_had_onboarding = get_option( Onboarding_Module::ONBOARDING_OPTION );

        // Get the latest installation from Elementor's Install log in the DB.
        $latest_install = key( Upgrade_Manager::get_installs_history() );

        if ( ! empty( $latest_install ) ) {
            $is_new_install = version_compare( $latest_install, '3.6.0-beta', '>=' );
        } else {
            // If `$latest_install` is not set, Elementor was never installed on this site.
            $is_new_install = true;
        }

        if ( $already_had_onboarding || ! $is_new_install ) {
            return;
        }

        wp_safe_redirect( admin_url( 'admin.php?page=elementor-app#onboarding' ) );

        exit;
    }

    private function register_packages() {
        $assets_config_provider = ( new Assets_Config_Provider() )
            ->set_path_resolver( function ( $name ) {
                return ELEMENTOR_ASSETS_PATH . "js/packages/{$name}/{$name}.asset.php";
            } );

        Collection::make( [ 'ui', 'icons', 'query' ] )
            ->each( function( $package ) use ( $assets_config_provider ) {
                $suffix = Utils::is_script_debug() ? '' : '.min';
                $config = $assets_config_provider->load( $package )->get( $package );

                if ( ! $config ) {
                    return;
                }

                wp_register_script(
                    $config['handle'],
                    ELEMENTOR_ASSETS_URL . "js/packages/{$package}/{$package}{$suffix}.js",
                    $config['deps'],
                    ELEMENTOR_VERSION,
                    true
                );
            } );
    }

    /**
     * Enqueue admin scripts.
     *
```

```
	 * Registers all the admin scripts and enqueues them.
	 *
	 * Fired by `admin_enqueue_scripts` action.
	 *
	 * @since 1.0.0
	 * @access public
	 */
	public function enqueue_scripts() {
		wp_register_script(
			'elementor-admin-modules',
			$this->get_js_assets_url( 'admin-modules' ),
			[],
			ELEMENTOR_VERSION,
			true
		);

		$this->register_packages();

		// Temporary solution for the admin.
		wp_register_script(
			'elementor-ai-admin',
			$this->get_js_assets_url( 'ai-admin' ),
			[
				'elementor-common',
				'elementor-v2-ui',
				'elementor-v2-icons',
			],
			ELEMENTOR_VERSION,
			true
		);

		wp_register_script(
			'elementor-admin',
			$this->get_js_assets_url( 'admin' ),
			[
				'elementor-common',
				'elementor-admin-modules',
			],
			ELEMENTOR_VERSION,
			true
		);

		wp_enqueue_script( 'elementor-admin' );

		wp_set_script_translations( 'elementor-admin', 'elementor' );

		$this->maybe_enqueue_hints();

		$this->print_config();
	}

	/**
	 * Enqueue admin styles.
	 *
	 * Registers all the admin styles and enqueues them.
	 *
	 * Fired by `admin_enqueue_scripts` action.
	 *
	 * @since 1.0.0
	 * @access public
	 */
	public function enqueue_styles() {
		$direction_suffix = is_rtl() ? '-rtl' : '';

		wp_register_style(
			'elementor-admin',
			$this->get_css_assets_url( 'admin' . $direction_suffix ),
			[
				'elementor-common',
			],
			ELEMENTOR_VERSION
		);

		wp_enqueue_style( 'elementor-admin' );

		// It's for upgrade notice.
		// TODO: enqueue this just if needed.
		add_thickbox();
	}

	/**
	 * Print switch mode button.
	 *
	 * Adds a switch button in post edit screen (which has cpt support). To allow
	 * the user to switch from the native WordPress editor to Elementor builder.
	 *
	 * Fired by `edit_form_after_title` action.
	 *
	 * @since 1.0.0
	 * @access public
	 *
	 * @param \WP_Post $post The current post object.
	 */
	public function print_switch_mode_button( $post ) {
		// Exit if Gutenberg are active.
		if ( did_action( 'enqueue_block_editor_assets' ) ) {
			return;
		}

		$document = Plugin::$instance->documents->get( $post->ID );

		if ( ! $document || ! $document->is_editable_by_current_user() ) {
			return;
		}

		wp_nonce_field( basename( __FILE__ ), '_elementor_edit_mode_nonce' );
```

```
    ?>
    <div id="elementor-switch-mode">
        <input id="elementor-switch-mode-input" type="hidden" name="_elementor_post_mode" value="<?php echo esc_attr( $document->is_built_with_elementor()
        <button id="elementor-switch-mode-button" type="button" class="button button-primary button-hero">
            <span class="elementor-switch-mode-on">
                <i class="eicon-arrow-<?php echo ( is_rtl() ) ? 'right' : 'left'; ?>" aria-hidden="true"></i>
                <?php echo esc_html__( 'Back to WordPress Editor', 'elementor' ); ?>
            </span>
            <span class="elementor-switch-mode-off">
                <i class="eicon-elementor-square" aria-hidden="true"></i>
                <?php echo esc_html__( 'Edit with Elementor', 'elementor' ); ?>
            </span>
        </button>
    </div>
    <div id="elementor-editor">
        <a id="elementor-go-to-edit-page-link" href="<?php echo esc_url( $document->get_edit_url() ); ?>">
            <div id="elementor-editor-button" class="button button-primary button-hero">
                <i class="eicon-elementor-square" aria-hidden="true"></i>
                <?php echo esc_html__( 'Edit with Elementor', 'elementor' ); ?>
            </div>
            <div class="elementor-loader-wrapper">
                <div class="elementor-loader">
                    <div class="elementor-loader-boxes">
                        <div class="elementor-loader-box"></div>
                        <div class="elementor-loader-box"></div>
                        <div class="elementor-loader-box"></div>
                        <div class="elementor-loader-box"></div>
                    </div>
                </div>
                <div class="elementor-loading-title"><?php echo esc_html__( 'Loading', 'elementor' ); ?></div>
            </div>
        </a>
    </div>
    <?php
}

/**
 * Save post.
 *
 * Flag the post mode when the post is saved.
 *
 * Fired by `save_post` action.
 *
 * @since 1.0.0
 * @access public
 *
 * @param int $post_id Post ID.
 */
public function save_post( $post_id ) {
    if ( ! wp_verify_nonce( Utils::get_super_global_value( $_POST, '_elementor_edit_mode_nonce' ), basename( __FILE__ ) ) ) {
        return;
    }

    if ( defined( 'DOING_AUTOSAVE' ) && DOING_AUTOSAVE ) {
        return;
    }

    Plugin::$instance->documents->get( $post_id )->set_is_built_with_elementor( ! empty( $_POST['_elementor_post_mode'] ) );
}

/**
 * Add Elementor post state.
 *
 * Adds a new "Elementor" post state to the post table.
 *
 * Fired by `display_post_states` filter.
 *
 * @since 1.8.0
 * @access public
 *
 * @param array    $post_states An array of post display states.
 * @param \WP_Post $post        The current post object.
 *
 * @return array A filtered array of post display states.
 */
public function add_elementor_post_state( $post_states, $post ) {
    $document = Plugin::$instance->documents->get( $post->ID );

    if ( $document && $document->is_built_with_elementor() && $document->is_editable_by_current_user() ) {
        $post_states['elementor'] = esc_html__( 'Elementor', 'elementor' );
    }

    return $post_states;
}

/**
 * Body status classes.
 *
 * Adds CSS classes to the admin body tag.
 *
 * Fired by `admin_body_class` filter.
 *
 * @since 1.0.0
 * @access public
 *
 * @param string $classes Space-separated list of CSS classes.
 *
 * @return string Space-separated list of CSS classes.
 */
public function body_status_classes( $classes ) {
    global $pagenow;

    if ( in_array( $pagenow, [ 'post.php', 'post-new.php' ], true ) && Utils::is_post_support() ) {
        $post = get_post();

        $document = Plugin::$instance->documents->get( $post->ID );
        $mode_class = $document && $document->is_built_with_elementor() ? 'elementor-editor-active' : 'elementor-editor-inactive';

        $classes .= ' ' . $mode_class;
    }

    return $classes;
}

/**
 * Plugin action links.
 *
 * Adds action links to the plugin list table
 *
 * Fired by `plugin_action_links` filter.
 *
 * @since 1.0.0
 * @access public
 *
 * @param array $links An array of plugin action links.
 *
 * @return array An array of plugin action links.
 */
public function plugin_action_links( $links ) {
    $settings_link = sprintf( '<a href="%1$s">%2$s</a>', admin_url( 'admin.php?page=' . Settings::PAGE_ID ), esc_html__( 'Settings', 'elementor' ) );

    array_unshift( $links, $settings_link );

    $go_pro_text = esc_html__( 'Get Elementor Pro', 'elementor' );
    if ( Utils::is_sale_time() ) {
        $go_pro_text = esc_html__( 'Discounted Upgrades Now!', 'elementor' );
    }

    $links['go_pro'] = sprintf( '<a href="%1$s" target="_blank" class="elementor-plugins-gopro">%2$s</a>', 'https://go.elementor.com/go-pro-wp-plugins/', $

    return $links;
}

/**
 * Plugin row meta.
 *
 * Adds row meta links to the plugin list table
 *
 * Fired by `plugin_row_meta` filter.
 *
 * @since 1.1.4
 * @access public
 *
 * @param array  $plugin_meta An array of the plugin's metadata, including
 *                            the version, author, author URI, and plugin URI.
 * @param string $plugin_file Path to the plugin file, relative to the plugins
 *                            directory.
 *
 * @return array An array of plugin row meta links.
 */
public function plugin_row_meta( $plugin_meta, $plugin_file ) {
    if ( ELEMENTOR_PLUGIN_BASE === $plugin_file ) {
        $row_meta = [
            'docs' => '<a href="https://go.elementor.com/docs-admin-plugins/" aria-label="' . esc_attr( esc_html__( 'View Elementor Documentation', 'elemen
            'ideo' => '<a href="https://go.elementor.com/yt-admin-plugins/" aria-label="' . esc_attr( esc_html__( 'View Elementor Video Tutorials', 'elemen
        ];

        $plugin_meta = array_merge( $plugin_meta, $row_meta );
    }

    return $plugin_meta;
}

/**
 * Admin footer text.
 *
 * Modifies the "Thank you" text displayed in the admin footer.
 *
 * Fired by `admin_footer_text` filter.
 *
 * @since 1.0.0
 * @access public
 *
 * @param string $footer_text The content that will be printed.
 *
 * @return string The content that will be printed.
 */
public function admin_footer_text( $footer_text ) {
    $current_screen = get_current_screen();
    $is_elementor_screen = ( $current_screen && false !== strpos( $current_screen->id, 'elementor' ) );

    if ( $is_elementor_screen ) {
        $footer_text = sprintf(
            /* translators: 1: Elementor, 2: Link to plugin review */
            __( 'Enjoyed %1$s? Please leave us a %2$s rating. We really appreciate your support!', 'elementor' ),
            '<strong>' . esc_html__( 'Elementor', 'elementor' ) . '</strong>',
            '<a href="https://go.elementor.com/admin-review/" target="_blank">&#9733;&#9733;&#9733;&#9733;&#9733;</a>'
        );
    }

    return $footer_text;
}

/**
 * Register dashboard widgets.
 *
 * Adds a new Elementor widgets to WordPress dashboard.
 *
 * Fired by `wp_dashboard_setup` action.
 *
 * @since 1.9.0
```

```
	 * @access public
	 */
	public function register_dashboard_widgets() {
		wp_add_dashboard_widget( 'e-dashboard-overview', esc_html__( 'Elementor Overview', 'elementor' ), [ $this, 'elementor_dashboard_overview_widget' ] );

		// Move our widget to top.
		global $wp_meta_boxes;

		$dashboard = $wp_meta_boxes['dashboard']['normal']['core'];
		$ours = [
			'e-dashboard-overview' => $dashboard['e-dashboard-overview'],
		];

		$wp_meta_boxes['dashboard']['normal']['core'] = array_merge( $ours, $dashboard ); // phpcs:ignore WordPress.WP.GlobalVariablesOverride.Prohibited
	}

	/**
	 * Displays the Elementor dashboard widget.
	 *
	 * Fired by `wp_add_dashboard_widget` function.
	 *
	 * @since 1.9.0
	 * @access public
	 */
	public function elementor_dashboard_overview_widget() {
		?>
		<div class="e-dashboard-overview e-dashboard-widget">
			<?php
			self::elementor_dashboard_overview_header();
			self::elementor_dashboard_overview_recently_edited();
			self::elementor_dashboard_overview_news_updates();
			self::elementor_dashboard_overview_footer();
			?>
		</div>
		<?php
	}

	/**
	 * Displays the Elementor dashboard widget - header section.
	 * Fired by `elementor_dashboard_overview_widget` function.
	 *
	 * @param bool $show_versions
	 * @param bool $is_create_post_enabled
	 *
	 * @return void
	 * @since 3.12.0
	 */
	public static function elementor_dashboard_overview_header( bool $show_versions = true, bool $is_create_post_enabled = true ) {
		if ( User::is_current_user_can_edit_post_type( 'page' ) ) {
			$create_new_label = esc_html__( 'Create New Page', 'elementor' );
			$create_new_post_type = 'page';
		} elseif ( User::is_current_user_can_edit_post_type( 'post' ) ) {
			$create_new_label = esc_html__( 'Create New Post', 'elementor' );
			$create_new_post_type = 'post';
		}
		?>
		<div class="e-overview__header">
			<?php if ( $show_versions ) { ?>
				<div class="e-overview__logo">
					<div class="e-logo-wrapper"><i class="eicon-elementor"></i></div>
				</div>
				<div class="e-overview__versions">
					<span class="e-overview__version"><?php echo esc_html__( 'Elementor', 'elementor' ); ?> v<?php echo ELEMENTOR_VERSION; // phpcs:ignore Word
					<?php
					/**
					 * Elementor dashboard widget after the version.
					 * Fires after Elementor version display in the dashboard widget.
					 *
					 * @since 1.9.0
					 */
					do_action( 'elementor/admin/dashboard_overview_widget/after_version' );
					?>
				</div>
			<?php } ?>
			<?php if ( ! empty( $create_new_post_type ) && $is_create_post_enabled ) { ?>
				<div class="e-overview__create">
					<a href="<?php echo esc_url( Plugin::$instance->documents->get_create_new_post_url( $create_new_post_type ) ); ?>" class="button"><span ari
				</div>
			<?php } ?>
		</div>
		<?php
	}

	/**
	 * Displays the Elementor dashboard widget - recently edited section.
	 * Fired by `elementor_dashboard_overview_widget` function.
	 *
	 * @param array $args
	 * @param bool $show_heading
	 *
	 * @return void
	 * @since 3.12.0
	 */
	public static function elementor_dashboard_overview_recently_edited( array $args = [], bool $show_heading = true ) {
		$recently_edited_query = Utils::get_recently_edited_posts_query( $args );

		if ( $recently_edited_query->have_posts() ) { ?>
			<div class="e-overview__recently-edited">
				<?php if ( $show_heading ) { ?>
					<h3 class="e-heading e-divider_bottom"><?php echo esc_html__( 'Recently Edited', 'elementor' ); ?></h3>
				<?php } ?>
				<ul class="e-overview__posts">
					<?php
					while ( $recently_edited_query->have_posts() ) {
						$recently_edited_query->the_post();
						$document = Plugin::$instance->documents->get( get_the_ID() );
```

```
                    $date = date_i18n( _x( 'M jS', 'Dashboard Overview Widget Recently Date', 'elementor' ), get_the_modified_time( 'U' ) );
                    ?>
                    <li class="e-overview__post">
                        <a href="<?php echo esc_url( $document->get_edit_url() ); ?>" class="e-overview__post-link"><?php echo esc_html( get_the_title() );
                            <span class="dashicons dashicons-edit"></span></a>
                        <span><?php echo $date; // phpcs:ignore WordPress.Security.EscapeOutput.OutputNotEscaped ?>, <?php the_modified_time(); ?></span>
                    </li>
                <?php } ?>
            </ul>
        </div>
    <?php }
}

/**
 * Displays the Elementor dashboard widget - news and updates section.
 * Fired by `elementor_dashboard_overview_widget` function.
 *
 * @param int $limit_feed
 * @param bool $show_heading
 *
 * @return void
 * @since 3.12.0
 * @access public
 */
public static function elementor_dashboard_overview_news_updates( int $limit_feed = 0, bool $show_heading = true ) {
    $elementor_feed = Api::get_feed_data();
    if ( $limit_feed > 0 ) {
        $elementor_feed = array_slice( $elementor_feed, 0, $limit_feed );
    }

    if ( ! empty( $elementor_feed ) ) { ?>
        <div class="e-overview__feed">
            <?php if ( $show_heading ) { ?>
                <h3 class="e-heading e-divider_bottom"><?php echo esc_html__( 'News & Updates', 'elementor' ); ?></h3>
            <?php } ?>
            <ul class="e-overview__posts">
                <?php foreach ( $elementor_feed as $feed_item ) { ?>
                    <li class="e-overview__post">
                        <a href="<?php echo esc_url( $feed_item['url'] ); ?>" class="e-overview__post-link" target="_blank">
                            <?php if ( ! empty( $feed_item['badge'] ) ) { ?>
                                <span class="e-overview__badge"><?php echo esc_html( $feed_item['badge'] ); ?></span>
                            <?php } ?>
                            <?php echo esc_html( $feed_item['title'] ); ?>
                        </a>
                        <p class="e-overview__post-description"><?php echo esc_html( $feed_item['excerpt'] ); ?></p>
                    </li>
                <?php } ?>
            </ul>
        </div>
    <?php }
}

/**
 * Displays the Elementor dashboard widget - footer section.
 * Fired by `elementor_dashboard_overview_widget` function.
 *
 * @since 3.12.0
 */
public static function elementor_dashboard_overview_footer() {
    ?>
    <div class="e-overview__footer e-divider_top">
        <ul>
            <?php foreach ( self::static_get_dashboard_overview_widget_footer_actions() as $action_id => $action ) { ?>
                <li class="e-overview__<?php echo esc_attr( $action_id ); ?>">
                    <a href="<?php echo esc_url( $action['link'] ); ?>" target="_blank"><?php echo esc_html( $action['title'] ); ?>
                        <span class="screen-reader-text"><?php echo esc_html__( '(opens in a new window)', 'elementor' ); ?></span>
                        <span aria-hidden="true" class="dashicons dashicons-external"></span></a>
                </li>
            <?php } ?>
        </ul>
    </div>
    <?php
}

/**
 * Get elementor dashboard overview widget footer actions.
 *
 * Retrieves the footer action links displayed in elementor dashboard widget.
 *
 * @since 3.12.0
 * @access public
 */
public static function static_get_dashboard_overview_widget_footer_actions() {
    $base_actions = [
        'blog' => [
            'title' => esc_html__( 'Blog', 'elementor' ),
            'link' => 'https://go.elementor.com/overview-widget-blog/',
        ],
        'help' => [
            'title' => esc_html__( 'Help', 'elementor' ),
            'link' => 'https://go.elementor.com/overview-widget-docs/',
        ],
    ];

    $additions_actions = [];
    $additions_actions['ai'] = [
        'title' => esc_html__( 'Build Smart with AI', 'elementor' ),
        'link' => 'https://go.elementor.com/overview-widget-ai/',
    ];
    $additions_actions['go-pro'] = [
        'title' => esc_html__( 'Upgrade', 'elementor' ),
        'link' => 'https://go.elementor.com/go-pro-wp-overview-widget/',
    ];

    /**
```

```php
		 * Dashboard widget footer actions.
		 *
		 * Filters the additions actions displayed in Elementor dashboard widget.
		 *
		 * Developers can add new action links to Elementor dashboard widget
		 * footer using this filter.
		 *
		 * @since 1.9.0
		 *
		 * @param array $additions_actions Elementor dashboard widget footer actions.
		 */
		$additions_actions = apply_filters( 'elementor/admin/dashboard_overview_widget/footer_actions', $additions_actions );

		$actions = $base_actions + $additions_actions;

		return $actions;
	}

	/**
	 * Get elementor dashboard overview widget footer actions.
	 *
	 * Retrieves the footer action links displayed in elementor dashboard widget.
	 *
	 * @since 1.9.0
	 * @access private
	 */
	private function get_dashboard_overview_widget_footer_actions() {
		return self::static_get_dashboard_overview_widget_footer_actions();
	}

	/**
	 * Admin action new post.
	 *
	 * When a new post action is fired the title is set to 'Elementor' and the post ID.
	 *
	 * Fired by `admin_action_elementor_new_post` action.
	 *
	 * @since 1.9.0
	 * @access public
	 */
	public function admin_action_new_post() {
		check_admin_referer( 'elementor_action_new_post' );

		$post_type = Utils::get_super_global_value( $_GET, 'post_type' ) ?? 'post';

		if ( ! User::is_current_user_can_edit_post_type( $post_type ) ) {
			return;
		}

		if ( empty( $_GET['template_type'] ) ) {
			$type = 'post';
		} else {
			$type = sanitize_text_field( wp_unslash( $_GET['template_type'] ) );
		}

		$post_data = Utils::get_super_global_value( $_GET, 'post_data' ) ?? [];

		$post_data = $this->filter_post_data( $post_data );

		/**
		 * Create new post meta data.
		 *
		 * Filters the meta data of any new post created.
		 *
		 * @since 2.0.0
		 *
		 * @param array $meta Post meta data.
		 */
		$meta = [];

		if ( isset( $_GET['meta'] ) && is_array( $_GET['meta'] ) ) {
			$meta = array_map( 'sanitize_text_field', wp_unslash( $_GET['meta'] ) );
		}

		$meta = apply_filters( 'elementor/admin/create_new_post/meta', $meta );

		$post_data['post_type'] = $post_type;

		$document = Plugin::$instance->documents->create( $type, $post_data, $meta );

		if ( is_wp_error( $document ) ) {
			wp_die( $document ); // phpcs:ignore WordPress.Security.EscapeOutput.OutputNotEscaped
		}

		wp_safe_redirect( $document->get_edit_url() );

		die;
	}

	private function get_allowed_fields_for_role() {
		$allowed_fields = [
			'post_title',
			'post_content',
			'post_excerpt',
			'post_category',
			'post_type',
			'tags_input',
		];

		if ( current_user_can( 'publish_posts' ) ) {
			$allowed_fields[] = 'post_status';
		}

		if ( current_user_can( 'edit_others_posts' ) ) {
			$allowed_fields[] = 'post_author';
		}
		return $allowed_fields;
	}

	private function filter_post_data( $post_data ) {
		$allowed_fields = $this->get_allowed_fields_for_role();
		return array_filter(
			$post_data,
			function( $key ) use ( $allowed_fields ) {
				return in_array( $key, $allowed_fields, true );
			},
			ARRAY_FILTER_USE_KEY
		);
	}
	/**
	 * @since 2.3.0
	 * @access public
	 */
	public function add_new_template_template() {
		Plugin::$instance->common->add_template( ELEMENTOR_PATH . 'includes/admin-templates/new-template.php' );
	}

	public function add_new_floating_elements_template() {
		Plugin::$instance->common->add_template( ELEMENTOR_PATH . 'includes/admin-templates/new-floating-elements.php' );
	}

	public function enqueue_new_floating_elements_scripts() {
		$suffix = Utils::is_script_debug() ? '' : '.min';

		wp_enqueue_script(
			'elementor-floating-elements-modal',
			ELEMENTOR_ASSETS_URL . 'js/floating-elements-modal' . $suffix . '.js',
			[],
			ELEMENTOR_VERSION,
			true
		);

		wp_set_script_translations( 'elementor-floating-elements-modal', 'elementor' );
	}

	/**
	 * @access public
	 */
	public function enqueue_new_template_scripts() {
		$suffix = Utils::is_script_debug() ? '' : '.min';

		wp_enqueue_script(
			'elementor-new-template',
			ELEMENTOR_ASSETS_URL . 'js/new-template' . $suffix . '.js',
			[],
			ELEMENTOR_VERSION,
			true
		);

		wp_set_script_translations( 'elementor-new-template', 'elementor' );
	}

	/**
	 * @since 2.6.0
	 * @access public
	 */
	public function add_beta_tester_template() {
		Plugin::$instance->common->add_template( ELEMENTOR_PATH . 'includes/admin-templates/beta-tester.php' );
	}

	/**
	 * @access public
	 */
	public function enqueue_beta_tester_scripts() {
		$suffix = Utils::is_script_debug() ? '' : '.min';

		wp_enqueue_script(
			'elementor-beta-tester',
			ELEMENTOR_ASSETS_URL . 'js/beta-tester' . $suffix . '.js',
			[],
			ELEMENTOR_VERSION,
			true
		);

		wp_set_script_translations( 'elementor-beta-tester', 'elementor' );
	}

	public function init_floating_elements() {
		$screens = [
			'elementor_library_page_e-floating-buttons' => true,
			'edit-e-floating-buttons' => true,
		];

		if ( ! isset( $screens[ get_current_screen()->id ] ) ) {
			return;
		}

		add_action( 'admin_head', [ $this, 'add_new_floating_elements_template' ] );
		add_action( 'admin_enqueue_scripts', [ $this, 'enqueue_new_floating_elements_scripts' ] );
	}

	/**
	 * @access public
	 */
	public function init_new_template() {
		if ( 'edit-elementor_library' !== get_current_screen()->id ) {
			return;
		}

		// Allow plugins to add their templates on admin_head.
		add_action( 'admin_head', [ $this, 'add_new_template_template' ] );
		add_action( 'admin_enqueue_scripts', [ $this, 'enqueue_new_template_scripts' ] );
	}

	public function version_update_warning( $current_version, $new_version ) {
		$current_version_minor_part = explode( '.', $current_version )[1];
		$new_version_minor_part = explode( '.', $new_version )[1];

		if ( $current_version_minor_part === $new_version_minor_part ) {
			return;
		}
		?>
		<hr class="e-major-update-warning__separator" />
		<div class="e-major-update-warning">
			<div class="e-major-update-warning__icon">
				<i class="eicon-info-circle"></i>
			</div>
			<div>
				<div class="e-major-update-warning__title">
					<?php echo esc_html__( 'Heads up, Please backup before upgrade!', 'elementor' ); ?>
				</div>
				<div class="e-major-update-warning__message">
					<?php
					printf(
						/* translators: %1$s Link open tag, %2$s: Link close tag. */
						esc_html__( 'The latest update includes some substantial changes across different areas of the plugin. We highly recommend you %1$sback
						'<a href="https://go.elementor.com/wp-dash-update-backup/">',
						'</a>'
					);
					?>
				</div>
			</div>
		</div>
		<?php
	}

	/**
	 * @access public
	 */
	public function init_beta_tester( $current_screen ) {
		if ( ( 'toplevel_page_elementor' === $current_screen->base ) || 'elementor_page_elementor-tools' === $current_screen->id ) {
			add_action( 'admin_head', [ $this, 'add_beta_tester_template' ] );
			add_action( 'admin_enqueue_scripts', [ $this, 'enqueue_beta_tester_scripts' ] );
		}
	}

	/**
	 * Admin constructor.
	 *
	 * Initializing Elementor in WordPress admin.
	 *
	 * @since 1.0.0
	 * @access public
	 */
	public function __construct() {
		Plugin::$instance->init_common();

		$this->add_component( 'feedback', new Feedback() );
		$this->add_component( 'admin-notices', new Admin_Notices() );

		add_action( 'admin_init', [ $this, 'maybe_redirect_to_getting_started' ] );

		add_action( 'admin_enqueue_scripts', [ $this, 'enqueue_scripts' ] );
		add_action( 'admin_enqueue_scripts', [ $this, 'enqueue_styles' ] );

		add_action( 'edit_form_after_title', [ $this, 'print_switch_mode_button' ] );
		add_action( 'save_post', [ $this, 'save_post' ] );

		add_filter( 'display_post_states', [ $this, 'add_elementor_post_state' ], 10, 2 );

		add_filter( 'plugin_action_links_' . ELEMENTOR_PLUGIN_BASE, [ $this, 'plugin_action_links' ] );
		add_filter( 'plugin_row_meta', [ $this, 'plugin_row_meta' ], 10, 2 );

		add_filter( 'admin_body_class', [ $this, 'body_status_classes' ] );
		add_filter( 'admin_footer_text', [ $this, 'admin_footer_text' ] );

		// Register Dashboard Widgets.
		add_action( 'wp_dashboard_setup', [ $this, 'register_dashboard_widgets' ] );

		// Admin Actions
		add_action( 'admin_action_elementor_new_post', [ $this, 'admin_action_new_post' ] );

		add_action( 'current_screen', [ $this, 'init_new_template' ] );
		add_action( 'current_screen', [ $this, 'init_floating_elements' ] );
		add_action( 'current_screen', [ $this, 'init_beta_tester' ] );

		add_action( 'in_plugin_update_message-' . ELEMENTOR_PLUGIN_BASE, function( $plugin_data ) {
			$this->version_update_warning( ELEMENTOR_VERSION, $plugin_data['new_version'] );
		} );

		add_action( 'elementor/ajax/register_actions', [ $this, 'register_ajax_hints' ] );
	}

	/**
	 * @since 2.3.0
	 * @access protected
	 */
	protected function get_init_settings() {
		$beta_tester_email = get_user_meta( get_current_user_id(), User::BETA_TESTER_META_KEY, true );
		$elementor_beta = get_option( 'elementor_beta', 'no' );
		$all_introductions = User::get_introduction_meta();
		$beta_tester_signup_dismissed = array_key_exists( Beta_Testers::BETA_TESTER_SIGNUP, $all_introductions );

		$settings = [
			'home_url' => home_url(),
			'settings_url' => Settings::get_url(),
			'user' => [
                'introduction' => User::get_introduction_meta(),
                'restrictions' => Plugin::$instance->role_manager->get_user_restrictions_array(),
                'is_administrator' => current_user_can( 'manage_options' ),
            ],
            'beta_tester' => [
                'beta_tester_signup' => Beta_Testers::BETA_TESTER_SIGNUP,
                'has_email' => $beta_tester_email,
                'option_enabled' => 'no' !== $elementor_beta,
                'signup_dismissed' => $beta_tester_signup_dismissed,
            ],
            'experiments' => $this->get_experiments(),
        ];

        /**
         * Localize settings.
         *
         * Filters the initial localize settings in the admin.
         *
         * WordPress has it's own way to pass localized data from PHP (backend) to
         * JS (frontend). Elementor uses this method to pass localize data in the
         * admin. This hook can be used to add more localized settings in addition
         * to the initial Elementor settings.
         *
         * @since 2.3.0
         *
         * @param array $settings Initial localize settings.
         */
        $settings = apply_filters( 'elementor/admin/localize_settings', $settings );

        return $settings;
    }

    private function get_experiments() {
        return ( new Collection( Plugin::$instance->experiments->get_features() ) )
            ->map( function ( $experiment_data ) {
                $dependencies = $experiment_data['dependencies'] ?? [];

                $dependencies = ( new Collection( $dependencies ) )
                    ->map( function ( $dependency ) {
                        return $dependency->get_name();
                    } )->all();

                return [
                    'name' => $experiment_data['name'],
                    'title' => $experiment_data['title'] ?? $experiment_data['name'],
                    'state' => $experiment_data['state'],
                    'default' => $experiment_data['default'],
                    'dependencies' => $dependencies,
                    'messages' => $experiment_data['messages'] ?? [],
                ];
            } )->all();
    }

    private function maybe_enqueue_hints() {
        if ( ! Hints::should_display_hint( 'image-optimization' ) ) {
            return;
        }

        wp_register_script(
            'media-hints',
            $this->get_js_assets_url( 'media-hints' ),
            [],
            ELEMENTOR_VERSION,
            true
        );

        $content = sprintf("%1\$s <a class='e-btn-1' href='%2\$s' target='_blank'>%3\$s</a>!",
            __( 'Optimize your images to enhance site performance by using Image Optimizer.', 'elementor' ),
            Hints::get_plugin_action_url( 'image-optimization' ),
            ( Hints::is_plugin_installed( 'image-optimization' ) ? __( 'Activate', 'elementor' ) : __( 'Install', 'elementor' ) ) . ' ' . __( 'Image Optimizer'
        );

        $dismissible = 'image_optimizer_hint';

        wp_localize_script( 'media-hints', 'elementorAdminHints', [
            'mediaHint' => [
                'display' => true,
                'type' => 'info',
                'content' => $content,
                'icon' => true,
                'dismissible' => $dismissible,
                'dismiss' => __( 'Dismiss this notice.', 'elementor' ),
                'button_event' => $dismissible,
                'button_data' => base64_encode(
                    wp_json_encode( [
                        'action_url' => Hints::get_plugin_action_url( 'image-optimization' ),
                    ] ),
                ),
            ],
        ] );

        wp_enqueue_script( 'media-hints' );
    }

    public function register_ajax_hints( $ajax_manager ) {
        $ajax_manager->register_ajax_action( 'elementor_image_optimization_campaign', [ $this, 'ajax_set_image_optimization_campaign' ] );
        $ajax_manager->register_ajax_action( 'elementor_core_site_mailer_campaign', [ $this, 'ajax_site_mailer_campaign' ] );
    }

    public function ajax_set_image_optimization_campaign( $request ) {
        if ( ! current_user_can( 'install_plugins' ) ) {
            return;
        }

        if ( empty( $request['source'] ) ) {
            return;
```

```php
        }

        $campaign_data = [
            'source' => sanitize_key( $request['source'] ),
            'campaign' => 'io-plg',
            'medium' => 'wp-dash',
        ];

        set_transient( 'elementor_image_optimization_campaign', $campaign_data, 30 * DAY_IN_SECONDS );
    }

    public function ajax_site_mailer_campaign( $request ) {
        if ( ! current_user_can( 'install_plugins' ) ) {
            return;
        }

        if ( empty( $request['source'] ) ) {
            return;
        }

        $campaign_data = [
            'source' => sanitize_key( $request['source'] ),
            'campaign' => 'sm-plg',
            'medium' => 'wp-dash',
        ];

        set_transient( 'elementor_site_mailer_campaign', $campaign_data, 30 * DAY_IN_SECONDS );
    }
}
```