LAMPIRAN

Source Code Aplikasi

1. *Source Code Class rtbBooking*

```
<?php
if ( !defined( 'ABSPATH' ) ) exit;
if ( !class_exists( 'rtbBooking' ) ) {
/*Class to handle a booking for Restaurant Table Bookings*/
class rtbBooking {
		public $request_processed = false;
		/*Whether or not this request was successfully saved to the database. */
		public $request_inserted = false;
		public function __construct() {}
		/*Load the booking information from a WP_Post object or an ID*/
		public function load_post( $post ) {
			if ( is_int( $post ) || is_string( $post ) ) {
				$post = get_post( $post );
			}
			if ( get_class( $post ) == 'WP_Post' && $post-
>post_type == RTB_BOOKING_POST_TYPE ) {
				$this->load_wp_post( $post );
				return true;
			} else {return false;
			}
		}
public function load_wp_post( $post ) {



// Store post for access to other data if needed by extensions
			$this->post = $post;
			$this->ID = $post->ID;
			$this->name = $post->post_title;
			$this->date = $post->post_date;



			$this->message = $post->post_content;
			$this->post_status = $post->post_status;
			$this->load_post_metadata();
			do_action( 'rtb_booking_load_post_data', $this, $post
);
		}
		public function load_post_metadata() {
			$meta_defaults = array(
			'party' => '',
			'email' => '',
			'phone' => '',
			'date_submission' => '',
			'logs' => array(),) ;
$meta_defaults = apply_filters( 'rtb_booking_metadata_defaults', $meta_defaults );
			if ( is_array( $meta = get_post_meta( $this->ID, 'rtb',
true ) ) ) {
$meta = array_merge( $meta_defaults, get_post_meta( $this->ID, 'rtb', true ) );
			} else {
			$meta = $meta_defaults;
			}
			$this->party = $meta['party'];
			$this->email = $meta['email'];
			$this->phone = $meta['phone'];
			$this->date_submission = $meta['date_submission'];
			$this->logs = $meta['logs'];
		}
		/*Prepare booking data loaded from the database for display in a
booking*/
		public function prepare_request_data() {
			// Split $date to $request_date and $request_time
			if ( empty( $this->request_date ) || empty( $this-
>request_time ) ) {
				$date = new DateTime( $this->date
);
				$this->request_date = $date-
>format( 'Y/m/d' );
				$this->request_time = $date-
>format( 'h:i A' );
			}
		}
}//so many source there
} // endif;
```

2. *Source Code Class SimplePie_Parse_Date*

```
<?php
/* Date Parser
* @package SimplePie
 * @subpackage Parsing
 */
class SimplePie_Parse_Date
{
/*Parse C99's asctime()'s date format*/
		public function date_asctime($date)
		{
			static $pcre;
			if (!$pcre)
			{
				$space = '[\x09\x20]+';
				$wday_name = $this->day_pcre;
				$mon_name = $this->month_pcre;
				$day = '([0-9]{1,2})';
				$hour = $sec = $min = '([0-9]{2})';
				$year = '([0-9]{4})';
				$terminator = '\x0A?\x00?';
$pcre = '/^' . $wday_name . $space . $mon_name . $space . $day . $space . $hour . ':' .
$min . ':' . $sec . $space . $year . $terminator . '$/i';
			}

				1: Day name
				2: Month
				3: Day
```

```
                                        4: Hour
                                        5: Minute
                                        6: Second
                                        7: Year
                                        */

                                        $month = $this-
>month[strtolower($match[2])];
                                        return
gmmktime($match[4], $match[5], $match[6], $month, $match[3], $match[7]);
                                }
                                                if
(preg_match($pcre, $date, $match))
                                {
                                        /*Capturing subpatterns:
                                        1: Day name
                                        2: Month
                                        3: Day
                                        4: Hour
                                        5: Minute
                                        6: Second
                                        7: Year*/
                                        $month = $this-
>month[strtolower($match[2])];
return gmmktime($match[4], $match[5], $match[6], $month, $match[3], $match[7]);
                                }
                                else
                                {
                                        return false;
                                }
                        }
}
```

3. *Source Code Class WPCF7_Submission*

```
<?php
class WPCF7_Submission {
        private static $instance;

        private $contact_form;
        private $status = 'init';
        private $posted_data = array();
        private $uploaded_files = array();
        private $skip_mail = false;
        private $response = '';
        private $invalid_fields = array();
        private $meta = array();
        private function __construct() {}
public static function get_instance( WPCF7_ContactForm $contact_form = null ) {
                if ( empty( self::$instance ) ) {
                        if ( null == $contact_form ) {
                                return null;
                        }
                        self::$instance = new self;
                        self::$instance->contact_form =
$contact_form;
self::$instance->skip_mail = $contact_form->in_demo_mode();
                        self::$instance-
>setup_posted_data();
                        self::$instance->submit();
                                } elseif ( null !=
$contact_form ) {
                        return null;
                }
                return self::$instance;
        }
        public function get_status() {
                return $this->status;
        }
        public function is( $status ) {
                return $this->status == $status;
        }
public function get_response() {
                return $this->response;
        }
        public function get_invalid_field( $name ) {
                if ( isset( $this->invalid_fields[$name] ) ) {
                        return $this-
>invalid_fields[$name];
                } else {
                        return false;
                }
        }

        public function get_invalid_fields() {
                return $this->invalid_fields;
        }
                public function get_posted_data( $name = '' ) {
                if ( ! empty( $name ) ) {
                        if ( isset( $this-
>posted_data[$name] ) ) {
                                return $this-
>posted_data[$name];
                        } else {
                                return null;
                        }
                }
                return $this->posted_data;
        }//so many source there
}
```

4. *Source Code Class GridGallery_Galleries_Model_Galleries*

```
<?php

/*Class GridGallery_Photos_Model_Folders
 * Folders
 * @package GridGallery\Photos\Model*/
class GridGallery_Photos_Model_Folders extends Rsc_Mvc_Model
{
    /**
     * @var string
     */
    protected $table;
     /**
     * @var bool
     */
    protected $debugEnabled;

    /**
     * @var string
     */
    protected $lastError;

    /**
     * @var int
     */
    protected $insertId;

    /**
     * Constructor
     */
    public function __construct($debugEnabled = false)
    {
        parent::__construct();

        $this->table = $this->db->prefix . 'gg_folders';
        $this->debugEnabled = (bool)$debugEnabled;
    }
     public function getInsertId()
    {
        return $this->insertId;
    }
    /*Adds the new album to the database*/


public function add($title){
        $title = htmlspecialchars($title, ENT_QUOTES, get_bloginfo('charset'));
        $query = $this->getQueryBuilder()->insertInto($this->table)
            ->fields('title', 'date')
            ->values($title, date('Y-m-d H:i:s'));
        if (!$this->db->query($query->build())) {
            $this->lastError = $this->db->last_error;
            return false; }
        $this->insertId = $this->db->insert_id;
        return true; }
     public function getById($id)
    {
        $query = $this->getQueryBuilder()->select('*')
            ->from($this->table)
            ->where('id', '=', (int)$id);

        if ($folder = $this->db->get_row($query->build())) {
            return $this->extend($folder);}
        return null; }
}
```

5. *Source Code Class FLPhotoModule*

```
<?php
/*Class GridGallery_Photos_Model_Photo
 *@package GridGallery\Photos\Model*/
class GridGallery_Photos_Model_Photos extends Rsc_Mvc_Model
{
    /**
     * @var bool
     */
    protected $debugEnabled;

    /**
     * @var string
     */
    protected $table;

    /**
     * @var int
     */
    protected $insertId;

    /**
     * @var string
     */
    protected $lastError;
        /*Returns the identifier of the last inserted photo
     * @return int*/
public function getInsertId()
    {
        return $this->insertId;
    }

    /**
     * @return string
     */
    public function getLastError()
    {
        return $this->lastError;
    }

     /* Adds the photo to the database.
     * @param int $attachmentId The identifier of the attachment to add

     * @param int $folderId The identifier of the folder (Default: 0)
     * @return bool*/
    public function add($attachmentId, $folderId = 0)
    {
        $query = $this->getQueryBuilder()->insertInto($this->table)
            ->fields('attachment_id', 'folder_id')
            ->values((int)$attachmentId, (int)$folderId);
                    if (!$this->db->query($query->build())) {
            $this->lastError = $this->db->last_error;
            return false;
```

```
    }

    $this->insertId = $this->db->insert_id;
    return true;
  }

  /*Update attachmentID for photo
   * @param $photoId
   * @param $attachmentId
   * @return false|int*/
  public function updateAttachmentId($photoId,$attachmentId){
    $query = $this->getQueryBuilder()->update($this->table)
      ->fields('attachment_id')
      ->values((int)$attachmentId)
      ->where('id','=',$photoId);

    return $this->db->query($query->build());
  }

  /**
   * Returns the photo by the id
   * @param int $id The identifier of the photo
       * @return object $photo or NULL on failure
   */
  public function getById($id)
  {
    return $this->getBy('id', (int)$id);
  }

  /*Returns the photo by the attachment id
   * @param int $attachmentId The identifier of the attachment
   * @return object $photo or NULL on failure*/
  public function getByAttachmentId($attachmentId)
  {
    $query = $this->getQueryBuilder()->select('*')
      ->from($this->table)
      ->where('attachment_id', '=', (int)$attachmentId)
      ->orderBy('id')
      ->order('DESC') ;

    if (null === $photo = $this->db->get_row($query->build(), ARRAY_A)) {
      return null;
    }
    return $this->extend($photo);
  }
    /*Returns the array of the photos
   * @return array|null*/
  public function getAll()
  {
    $query = $this->getQueryBuilder()->select('*')
      ->from($this->table);

    if ($photos = $this->db->get_results($query->build())) {
      foreach ($photos as $index => $photo) {
        $photos[$index] = $this->extend($photo);
      }
    }
    return $photos;
  }
   /** Deletes photo from the plugin by the attachment id
   * @param int $attachmentId The identifier of the attachment
   * @return bool TRUE on success, FALSE otherwise*/
  public function deleteByAttachmentId($attachmentId)
  {  $attachmentId = (int)$attachmentId;
    $photo = $this->getByAttachmentId($attachmentId);
    do_action('gg_delete_photo_attachment_id', $attachmentId);

    return $this->deleteBy('attachment_id', $attachmentId);
  }
    /*Deletes photo from the plugin by the identifier
   * @param int $id The identifier of the photo
   * @return bool TRUE of success, FALSE otherwise*/
  public function deleteById($id)
  {
    do_action('gg_delete_photo_id', $id);

    return $this->deleteBy('id', (int)$id);
  }

  public function deleteByFolderId($id)
  {
    return $this->deleteBy('folder_id', $id);
  }
}
```

6. *Source Code Class* sapAdminPage_2_0

```
<?php
/**Register, display and save a settings page in the WordPress admin menu.
 *@package Simple Admin Pages*/

class sapAdminPage_2_0 {

        public $title;
        public $menu_title;
        public $description; // optional description for this page
        public $capability; // user permissions needed to edit this panel
        public $id; // id of this page
        public $sections = array(); // array of sections to display on this page
        public $show_button = true; // whether or not to show the Save
Changes button

        public $setup_function = 'add_options_page'; // WP function to register
the page


        /**Initialize the page*/
        public function __construct( $args ) {

                // Parse the values passed
                $this->parse_args( $args );
        }

        /**Parse the arguments passed in the construction and assign them to
         * internal variables.*/
        private function parse_args( $args ) {
                foreach ( $args as $key => $val ) {
                        switch ( $key ) {
                                case 'id' :

$this->{$key} = esc_attr( $val );
                                default :

$this->{$key} = $val;}
                        }
        }
        /**Modify the capability required to save settings on this page*/
```

```
		public function modify_required_capability( $cap ) {
			return $this->capability;}

		/**Add the page to the appropriate menu slot.
		 * @note The default will be to post to the options page, but other
classes
		 *                                                    should override
this function.*/
		public function add_admin_menu() {
			call_user_func( $this->setup_function, $this->title,
$this->menu_title, $this->capability, $this->id, array( $this, 'display_admin_menu' ) );}

		/**Add a section to the page*/
		public function add_section( $section ) {
			if ( !$section ) {
				return;}
			$this->sections[ $section->id ] = $section;}

		/**Register the settings and sanitization callbacks for each setting*/
		public function register_admin_menu() {
			foreach ( $this->sections as $section ) {
				$section->add_settings_section();
				foreach ( $section->settings as
$setting ) {
					$setting-
>add_settings_field( $section->id );}
			}
			register_setting( $this->id, $this->id, array( $this,
'sanitize_callback' ) );
			// Modify capability required to save the settings if it's
not
			// the default `manage_options`
			if ( !empty( $this->capability ) && $this->capability
!== 'manage_options') {
				add_filter(
'option_page_capability_' . $this->id, array( $this, 'modify_required_capability' ) );}
		}
		/**Loop through the settings and sanitize the data*/
		public function sanitize_callback( $value ) {
			if ( empty( $_POST['_wp_http_referer'] ) ) {
				return $value;}
			// Get the current page/tab so we only update those
settings
			parse_str( $_POST['_wp_http_referer'], $referrer );
			$current_page = $this->get_current_page( $referrer );
			// Use a new empty value so only values for settings
that were added are
			// passed to the db.
			$new_value = array();
			foreach ( $this->sections as $section ) {
				foreach ( $section->settings as
$setting ) {
					if ( $setting->tab
== $current_page ) {

		$setting_value = isset( $value[$setting->id] ) ? $value[$setting->id] : '';

		$new_value[$setting->id] = $setting->sanitize_callback_wrapper(
$setting_value );}
				}
			}
			// Pull in the existing values so we never overwrite
values that were
			// on a different tab
			$old_value = get_option( $this->id );
			if ( is_array( $old_value ) ) {
				return array_merge( $old_value,
$new_value );
			} else {
				return $new_value;}
		}
		/**Get the current page/tab being viewed*/
		public function get_current_page( $request ) {
			if ( !empty( $request['tab'] ) ) {
				return $request['tab'];
			} elseif ( !empty( $this->default_tab ) ) {
				return $this->default_tab;
			} else {
				return $this->id;}
		}
		/**Output the settings passed to this page*/
		public function display_admin_menu() {
			if ( !$this->title && !count( $this->settings ) ) {
				return;}
			if ( !current_user_can( $this->capability ) ) {
				wp_die( __('You do not have
sufficient permissions to access this page.') );}
			$current_page = $this->get_current_page( $_GET );
			?>
				<div class="wrap">
					<?php $this-
>display_page_title(); ?>
					<?php if ( isset(
$this->default_tab ) ) : ?>
					<h2 class="nav-
tab-wrapper">
					<?php
					foreach( $this-
>sections as $section ) {

		if ( isset( $section->is_tab ) && $section->is_tab === true ) {

		$tab_url = add_query_arg(

			array(

				'settings-updated' => false,

				'tab' => $section->id)

		);

		$active = $current_page == $section->id ? ' nav-tab-active' : '';

		echo '<a href="' . esc_url( $tab_url ) . '" title="' . esc_attr( $section-
>title ) . '" class="nav-tab' . $active . '">';

			echo esc_html( $section->title );

		echo '</a>';}
					}
					?>
					</h2>
					<?php endif; ?>
					<form
method="post" action="options.php">

		<?php settings_fields( $this->id ); ?>

		<?php do_settings_sections( $current_page ); ?>

		<?php if ( $this->show_button ) { submit_button(); } ?>
					</form>
				</div>
			<?php}
		/**Output the title of the page*/
		public function display_page_title() {
			if ( empty( $this->title ) ) {
				return;}
			?>
				<h1><?php echo $this->title;
?></h1>
			<?php
		}
}
```