source: mstore-api / trunk / controllers / flutter-user.php

Last change on this file was 3279132, checked in by inspireui, 11 days ago
version 4.17.5
Property svn:eol-style set to native
File size: 57.3 KB

```
<?php
require_once(__DIR__ . '/flutter-base.php');
require_once(__DIR__ . '/helpers/apple-sign-in-helper.php');
require_once(__DIR__ . '/helpers/facebook-jwt-helper.php');

class FlutterUserController extends FlutterBaseController
{

    /**
     * Endpoint namespace
     *
     * @var string
     */
    protected $namespace = 'api/flutter_user';

    public function __construct()
    {
    }

    public function register_routes()
    {
        register_rest_route($this->namespace, '/reset-password', array(
            array(
                'methods' => 'POST',
                'callback' => array($this, 'reset_password'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/notification', array(
            array(
                'methods' => 'POST',
                'callback' => array($this, 'chat_notification'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/sign_up', array(
            array(
                'methods' => 'POST',
                'callback' => array($this, 'register'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
```

```
                }
            ),
        ));
        register_rest_route($this->namespace, '/sign_up_2', array(
            array(
                'methods' => 'POST',
                'callback' => array($this, 'register'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/register', array(
            array(
                'methods' => 'POST',
                'callback' => array($this, 'register'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/generate_auth_cookie', array(
            array(
                'methods' => 'POST',
                'callback' => array($this, 'generate_auth_cookie'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/fb_connect', array(
            array(
                'methods' => 'GET',
                'callback' => array($this, 'fb_connect'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/sms_login', array(
            array(
                'methods' => 'GET',
                'callback' => array($this, 'sms_login'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/firebase_sms', array(
            array(
                'methods' => 'POST',
                'callback' => function ($request) {
                    $phone = $this->firebase_sms_verify_id_token($request);
                    if (is_wp_error($phone)) {
                        return $phone;
                    }
                    return $this->firebase_sms_login($phone);
                },
```

```
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/firebase_sms_v2', array(
            array(
                'methods' => 'POST',
                'callback' => function ($request) {
                    $phone = $this->firebase_sms_verify_id_token($request);
                    if (is_wp_error($phone)) {
                        return $phone;
                    }
                    return $this->firebase_sms_login_v2($phone);
                },
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/apple_login_2', array(
            array(
                'methods' => 'POST',
                'callback' => array($this, 'apple_login'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/google_login', array(
            array(
                'methods' => 'GET',
                'callback' => array($this, 'google_login'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/post_comment', array(
            array(
                'methods' => 'GET',
                'callback' => array($this, 'post_comment'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/get_currentuserinfo', array(
            array(
                'methods' => 'GET',
                'callback' => array($this, 'get_currentuserinfo'),
                'permission_callback' => function () {
                    return parent::checkApiPermission();
                }
            ),
        ));

        register_rest_route($this->namespace, '/get_points', array(
```

```
174             array(
175                 'methods' => 'GET',
176                 'callback' => array($this, 'get_points'),
177                 'permission_callback' => function () {
178                     return parent::checkApiPermission();
179                 }
180             ),
181         ));
182 
183         register_rest_route($this->namespace, '/update_user_profile', array(
184             array(
185                 'methods' => 'POST',
186                 'callback' => array($this, 'update_user_profile'),
187                 'permission_callback' => function () {
188                     return parent::checkApiPermission();
189                 }
190             ),
191         ));
192 
193         register_rest_route($this->namespace, '/checkout', array(
194             array(
195                 'methods' => 'POST',
196                 'callback' => array($this, 'prepare_checkout'),
197                 'permission_callback' => function () {
198                     return parent::checkApiPermission();
199                 }
200             ),
201         ));
202 
203         register_rest_route($this->namespace, '/get_currency_rates', array(
204             array(
205                 'methods' => 'GET',
206                 'callback' => array($this, 'get_currency_rates'),
207                 'permission_callback' => function () {
208                     return parent::checkApiPermission();
209                 }
210             ),
211         ));
212 
213         register_rest_route($this->namespace, '/get_countries', array(
214             array(
215                 'methods' => 'GET',
216                 'callback' => array($this, 'get_countries'),
217                 'permission_callback' => function () {
218                     return parent::checkApiPermission();
219                 }
220             ),
221         ));
222 
223         register_rest_route($this->namespace, '/get_states', array(
224             array(
225                 'methods' => 'GET',
226                 'callback' => array($this, 'get_states'),
227                 'permission_callback' => function () {
228                     return parent::checkApiPermission();
229                 }
230             ),
231         ));
232 
233         register_rest_route($this->namespace, '/check-user', array(
234             array(
235                 'methods' => 'GET',
236                 'callback' => array($this, 'check_user'),
```

```
Line
237                 'permission_callback' => function () {
238                     return parent::checkApiPermission();
239                 }
240             ),
241         ));
242 
243         register_rest_route($this->namespace, '/digits/register/check', array(
244             array(
245                 'methods' => 'POST',
246                 'callback' => array($this, 'digits_register_check'),
247                 'permission_callback' => function () {
248                     return parent::checkApiPermission();
249                 }
250             ),
251         ));
252 
253         register_rest_route($this->namespace, '/digits/register', array(
254             array(
255                 'methods' => 'POST',
256                 'callback' => array($this, 'digits_register'),
257                 'permission_callback' => function () {
258                     return parent::checkApiPermission();
259                 }
260             ),
261         ));
262 
263         register_rest_route($this->namespace, '/digits/login/check', array(
264             array(
265                 'methods' => 'POST',
266                 'callback' => array($this, 'digits_login_check'),
267                 'permission_callback' => function () {
268                     return parent::checkApiPermission();
269                 }
270             ),
271         ));
272 
273         register_rest_route($this->namespace, '/digits/login', array(
274             array(
275                 'methods' => 'POST',
276                 'callback' => array($this, 'digits_login'),
277                 'permission_callback' => function () {
278                     return parent::checkApiPermission();
279                 }
280             ),
281         ));
282 
283         register_rest_route($this->namespace, '/digits/send_otp', array(
284             array(
285                 'methods' => 'POST',
286                 'callback' => array($this, 'digits_send_otp'),
287                 'permission_callback' => function () {
288                     return parent::checkApiPermission();
289                 }
290             ),
291         ));
292 
293         register_rest_route($this->namespace, '/digits/resend_otp', array(
294             array(
295                 'methods' => 'POST',
296                 'callback' => array($this, 'digits_resend_otp'),
297                 'permission_callback' => function () {
298                     return parent::checkApiPermission();
299                 }
```

```
            ),
        ));

        register_rest_route($this->namespace, '/delete_account', array(
            array(
                'methods' => WP_REST_Server::DELETABLE,
                'callback' => array($this, 'delete_account'),
                'permission_callback' => array($this, 'custom_delete_item_permissions_check'),
            ),
        ));
    }


    public function check_user($request)
    {
        $phone = $request['phone'];
        $username = $request['username'];
        if (isset($phone)) {
            $args = array('meta_key' => 'registered_phone_number', 'meta_value' => $phone);
            $search_users = get_users($args);
            if (empty($search_users)) {
                return false;
            }
        }
        if (isset($username)) {
            if (strpos($username, '@')) {
                $user_data = get_user_by('email', trim(wp_unslash($username)));
            } else {
                $login = trim($username);
                $user_data = get_user_by('login', $login);
            }
            if (empty($user_data)) {
                return false;
            }
        }

        return true;
    }


    public function reset_password()
    {
        $json = file_get_contents('php://input');
        $params = json_decode($json, TRUE);
        $usernameReq = $params["user_login"];

        $errors = new WP_Error();
        if (empty($usernameReq) || !is_string($usernameReq)) {
            return parent::sendError("empty_username", "Enter a username or email address.", 400);
        } elseif (strpos($usernameReq, '@')) {
            $user_data = get_user_by('email', trim(wp_unslash($usernameReq)));
            if (empty($user_data)) {
                return parent::sendError("invalid_email", "There is no account with that username or email
address.", 404);
            }
        } else {
            $login = trim($usernameReq);
            $user_data = get_user_by('login', $login);
        }
        if (!$user_data) {
            return parent::sendError("invalid_email", "There is no account with that username or email
address.", 404);
        }
```

```
361
362         $user_login = $user_data->user_login;
363         $user_email = $user_data->user_email;
364         $key = get_password_reset_key($user_data);
365
366         if (is_wp_error($key)) {
367             return $key;
368         }
369
370         if (is_multisite()) {
371             $site_name = get_network()->site_name;
372         } else {
373             $site_name = wp_specialchars_decode(get_option('blogname'), ENT_QUOTES);
374         }
375
376         $message = __('Someone has requested a password reset for the following account:') . "\r\n\r\n";
377         $message .= sprintf(__('Site Name: %s'), $site_name) . "\r\n\r\n";
378         $message .= sprintf(__('Username: %s'), $user_login) . "\r\n\r\n";
379         $message .= __('If this was a mistake, just ignore this email and nothing will happen.') . "\r\n\r\n";
380         $message .= __('To reset your password, visit the following address:') . "\r\n\r\n";
381         $message .= network_site_url("wp-login.php?
    action=rp&key=$key&login=" . rawurlencode($user_login), 'login') . "\r\n";
382         $title = sprintf(__('[%s] Password Reset'), $site_name);
383         $title = apply_filters('retrieve_password_title', $title, $user_login, $user_data);
384         $message = apply_filters('retrieve_password_message', $message, $key, $user_login, $user_data);
385
386         wp_mail($user_email, wp_specialchars_decode($title), $message);
387
388         return new WP_REST_Response(array(
389             'status' => 'success',
390         ), 200);;
391     }
392
393     public function register()
394     {
395         if (!get_option( 'users_can_register' )) {
396             return parent::sendError("disabled_register", "Registration is not enabled.", 400);
397         }
398         $json = file_get_contents('php://input');
399         $params = json_decode($json, TRUE);
400         $usernameReq = $params["username"];
401         $emailReq = $params["email"];
402         $userPassReq = $params["user_pass"];
403         $userLoginReq = $params["user_login"];
404         $userEmailReq = $params["user_email"];
405
406         if (array_key_exists('referral_code', $params)) {
407             $referralCodeReq = $params["referral_code"];
408         }
409
410         if(isset($params['wcfm_membership_application_status'])){
411                     $wcfm_membership_application_status = $params['wcfm_membership_application_status'];
412                 }
413
414         $username = sanitize_user($usernameReq);
415         $email = sanitize_email($emailReq);
416
417         if ($username == $userEmailReq && $username == $userLoginReq) {
418             $is_email = is_email($username);
419             if ($is_email) {
420                 $email = $username;
421                 $user_name = explode("@", $email)[0];
422                 $params["user_email"] = $email;
```

```
                $params["user_login"] = $user_name;
            } else {
                $user_name = $username;
                $params["user_login"] = $user_name;
                $params["user_email"] = '';
            }
        }

        if (isset($params["seconds"])) {
            $seconds = (int)$params["seconds"];
        } else {
            $seconds = 1209600;
        }

        if (!validate_username($username)) {
            return parent::sendError("invalid_username", "Username is invalid.", 400);
        } elseif (username_exists($username)) {
            return parent::sendError("existed_username", "Username already exists.", 400);
        } else {
            if (!is_email($email)) {
                return parent::sendError("invalid_email", "E-mail address is invalid.", 400);
            } elseif (email_exists($email)) {
                return parent::sendError("existed_email", "E-mail address is already in use.", 400);
            } else {
                if (!$userPassReq) {
                    $params->user_pass = wp_generate_password();
                }


$allowed_params = array('user_login', 'user_email', 'user_pass', 'display_name', 'user_nicename', 'user_url', 'nickname', 'first_name',

'last_name', 'description', 'rich_editing', 'user_registered', 'role', 'jabber', 'aim', 'yim',
                    'comment_shortcuts', 'admin_color', 'use_ssl', 'show_admin_bar_front',
                );

                $dataRequest = $params;

                foreach ($dataRequest as $field => $value) {
                    if (in_array($field, $allowed_params)) {
                        $user[$field] = trim(sanitize_text_field($value));
                    }
                }

                $default_role = class_exists( 'WooCommerce' ) ? 'customer' : get_option('default_role');
                if( isset($params['dokan_enable_selling'])){
                    $user['role'] = 'seller';
                }else{

if (array_key_exists('role', $params) && in_array($params['role'], ['wcfm_delivery_boy', 'driver'], true)) {
                        $user['role'] = $params['role'];
                    }else{
                        $user['role'] = $default_role;
                    }
                }
                $_POST['user_role'] = $user['role'];//fix to register account with role in listeo

                //
                if (isset($referralCodeReq) && $referralCodeReq) {
                    $_COOKIE['woo_wallet_referral'] = sanitize_text_field( wp_unslash( $referralCodeReq ) );
                }

                $user_id = wp_insert_user($user);

```

```
Line
483                 if (is_wp_error($user_id)) {
484                     return parent::sendError($user_id->get_error_code(), $user_id->get_error_message(), 400);
485                 } elseif (isset($params["phone"])) {
486                     update_user_meta($user_id, 'billing_phone', $params["phone"]);
487                     update_user_meta($user_id, 'registered_phone_number', $params["phone"]);
488                 }
489             }
490         }
491         wp_new_user_notification($user_id, null, 'both');
492         if(isset( $wcfm_membership_application_status) &&  $wcfm_membership_application_status == 'pending'){
493             update_user_meta($user_id,'store_name', $user['display_name']);
494 
495             //fix crash when approve membership in WCFM
496             $wcfmvm_static_infos = (array) get_user_meta( $member_id, 'wcfmvm_static_infos', true );
497             $wcfmvm_static_infos['phone'] = $params["phone"] ?? '';
498             update_user_meta($user_id, 'wcfmvm_static_infos', $wcfmvm_static_infos);
499             update_user_meta($user_id, 'billing_phone', $params["phone"] ?? '');
500 
501             update_user_meta($user_id,'temp_wcfm_membership', true);
502             global $WCFMvm;
503             $WCFMvm->send_approval_reminder_admin( $user_id );
504         }
505 
506         if (isset($params['dokan_enable_selling'])) {
507             update_user_meta($user_id, 'dokan_enable_selling', 'no');
508         }
509         $cookie = generateCookieByUserId($user_id,  $seconds);
510 
511         return array(
512             "cookie" => $cookie,
513             "user_id" => $user_id,
514         );
515     }
516 
517 
518     private function get_shipping_address($userId)
519     {
520         $shipping = [];
521 
522         $shipping["first_name"] = get_user_meta($userId, 'shipping_first_name', true);
523         $shipping["last_name"] = get_user_meta($userId, 'shipping_last_name', true);
524         $shipping["company"] = get_user_meta($userId, 'shipping_company', true);
525         $shipping["address_1"] = get_user_meta($userId, 'shipping_address_1', true);
526         $shipping["address_2"] = get_user_meta($userId, 'shipping_address_2', true);
527         $shipping["city"] = get_user_meta($userId, 'shipping_city', true);
528         $shipping["state"] = get_user_meta($userId, 'shipping_state', true);
529         $shipping["postcode"] = get_user_meta($userId, 'shipping_postcode', true);
530         $shipping["country"] = get_user_meta($userId, 'shipping_country', true);
531         $shipping["email"] = get_user_meta($userId, 'shipping_email', true);
532         $shipping["phone"] = get_user_meta($userId, 'shipping_phone', true);
533 
534 if (empty($shipping["first_name"]) && empty($shipping["last_name"]) && empty($shipping["company"]) && empty($sh
ipping["address_1"]) && empty($shipping["address_2"]) && empty($shipping["city"]) && empty($shipping["state"])
&& empty($shipping["postcode"]) && empty($shipping["country"]) && empty($shipping["email"]) && empty($shipping[
"phone"])) {
535             return null;
536         }
537         return $shipping;
538     }
539 
540     private function get_billing_address($userId)
541     {
542         $billing = [];
```

```
        $billing["first_name"] = get_user_meta($userId, 'billing_first_name', true);
        $billing["last_name"] = get_user_meta($userId, 'billing_last_name', true);
        $billing["company"] = get_user_meta($userId, 'billing_company', true);
        $billing["address_1"] = get_user_meta($userId, 'billing_address_1', true);
        $billing["address_2"] = get_user_meta($userId, 'billing_address_2', true);
        $billing["city"] = get_user_meta($userId, 'billing_city', true);
        $billing["state"] = get_user_meta($userId, 'billing_state', true);
        $billing["postcode"] = get_user_meta($userId, 'billing_postcode', true);
        $billing["country"] = get_user_meta($userId, 'billing_country', true);
        $billing["email"] = get_user_meta($userId, 'billing_email', true);
        $billing["phone"] = get_user_meta($userId, 'billing_phone', true);


if (empty($billing["first_name"]) && empty($billing["last_name"]) && empty($billing["company"]) && empty($billing["address_1"]) && empty($billing["address_2"]) && empty($billing["city"]) && empty($billing["state"]) && empty($billing["postcode"]) && empty($billing["country"]) && empty($billing["email"]) && empty($billing["phone"])) {
            return null;
        }

        return $billing;
    }

    function getResponseUserInfo($user)
    {
        $shipping = $this->get_shipping_address($user->ID);
        $billing = $this->get_billing_address($user->ID);
        $avatar = get_user_meta($user->ID, 'user_avatar', true);
        if (!isset($avatar) || $avatar == "" || is_bool($avatar)) {
            $avatar = get_avatar_url($user->ID);
        } else {
            $avatar = $avatar[0];
        }
        $is_driver_available = false;
        if(is_plugin_active('delivery-drivers-for-woocommerce/delivery-drivers-for-woocommerce.php')){
                        $is_driver_available = get_user_meta( $user->ID, 'ddwc_driver_availability', true );
                }else{
            $is_driver_available = in_array('administrator',$user->roles) || in_array('wcfm_delivery_boy',$user->roles);
        }

        // Check order status change capability
        $order_status_change = false;

        // Check for Dokan
        if (is_plugin_active('dokan-lite/dokan.php') || is_plugin_active('dokan-pro/dokan-pro.php')) {
            $dokan_settings = get_option('dokan_selling');
            $order_status_change = isset($dokan_settings['order_status_change']) ?
                filter_var($dokan_settings['order_status_change'], FILTER_VALIDATE_BOOLEAN) : false;
        }

        // Check for WCFM
        if (is_plugin_active('wc-frontend-manager/wc_frontend_manager.php') && class_exists('WCFMmp')) {
            global $WCFM;
            $order_status_change = $WCFM->wcfm_vendor_support->wcfm_vendor_has_capability($user->ID, 'order_status_update');
        }

        // If user is admin, always allow order status change
        if (in_array('administrator', $user->roles)) {
            $order_status_change = true;
        }

```

```
Line
601         return array(
602             "id" => $user->ID,
603             "username" => $user->user_login,
604             "nicename" => $user->user_nicename,
605             "email" => $user->user_email,
606             "url" => $user->user_url,
607             "registered" => $user->user_registered,
608             "displayname" => $user->display_name,
609             "firstname" => $user->user_firstname,
610             "lastname" => $user->last_name,
611             "nickname" => $user->nickname,
612             "description" => $user->user_description,
613             "capabilities" => $user->wp_capabilities,
614             "role" => $user->roles,
615             "shipping" => $shipping,
616             "billing" => $billing,
617             "avatar" => $avatar,
618             "is_driver_available" => $is_driver_available,
619             "dokan_enable_selling" => $user->dokan_enable_selling,
620             "order_status_change" => (bool)$order_status_change
621         );
622     }
623 
624     public function generate_auth_cookie()
625     {
626         $json = file_get_contents('php://input');
627         $params = json_decode($json, TRUE);
628         if (!isset($params["username"]) || !isset($params["password"])) {
629             return parent::sendError("invalid_login", "Invalid params", 400);
630         }
631         $username = $params["username"];
632         $password = $params["password"];
633 
634 
635         if (isset($params["seconds"])) {
636             $seconds = (int)$params["seconds"];
637         } else {
638             $seconds = 1209600;
639         }
640         $_POST['action'] = 'listeoajaxlogin'; //fix to return json if login error in listeo
641         $user = wp_authenticate($username, $password);
642 
643         if (is_wp_error($user)) {
644             return parent::sendError($user->get_error_code(), "Invalid username/email and/or password.", 401);
645         }
646 
647         $cookie = generateCookieByUserId($user->ID, $seconds);
648 
649         return array(
650             "cookie" => $cookie,
651             "cookie_name" => LOGGED_IN_COOKIE,
652             "user" => $this->getResponseUserInfo($user),
653         );
654     }
655 
656     function createSocialAccount($email, $name, $firstName, $lastName, $userName)
657     {
658         $email_exists = email_exists($email);
659         if ($email_exists) {
660             $user = get_user_by('email', $email);
661             $user_id = $user->ID;
662         } else {
663             $i = 0;
```

```
            while (username_exists($userName)) {
                $i++;
                $userName = strtolower($userName) . '.' . $i;
            }
            $random_password = wp_generate_password($length = 12, $include_standard_special_chars = false);
            $userdata = array(
                'user_login' => $userName,
                'user_email' => $email,
                'user_pass' => $random_password,
                'display_name' => $name,
                'first_name' => $firstName,
                'last_name' => $lastName);
            $user_id = wp_insert_user($userdata);
        }

        $cookie = generateCookieByUserId($user_id);
        $user = get_userdata($user_id);

        $response['wp_user_id'] = $user_id;
        $response['cookie'] = $cookie;
        $response['user_login'] = $user->user_login;
        $response['user'] = $this->getResponseUserInfo($user);
        return $response;
    }

    public function fb_connect($request)
    {
        $fields = 'id,name,first_name,last_name,email';
        $enable_ssl = true;
        $access_token = $request["access_token"];
        if (!isset($access_token)) {
            return parent::sendError("invalid_login", "You must include a 'access_token' variable. Get the valid access_token for this app from Facebook API.", 400);
        }

        $result = [];

        // If token is an AuthenticationToken (in case of limited login for
        // iOS), validate the JWT and return the payload
        $jwt = FacebookJWTHelper::validateJWT($access_token);

        if ($jwt['success']) {
            $decodedPayload = $jwt['decoded']['payload'];
            $result["email"] = $decodedPayload->email;
            $result["name"] = $decodedPayload->name;
            $result["first_name"] = $decodedPayload->given_name;
            $result["last_name"] = $decodedPayload->family_name;
        } else {
            $url = 'https://graph.facebook.com/me/?fields=' . $fields . '&access_token=' . $access_token;
            $payload = wp_remote_retrieve_body(wp_remote_get($url));
            $result = json_decode($payload, true);
        }

        if (isset($result["email"])) {
            $user_name = strtolower($result['first_name'] . '.' . $result['last_name']);
            return $this->createSocialAccount($result["email"], $result['name'], $result['first_name'], $result['last_name'], $user_name);
        } else {
            return parent::sendError("invalid_login", "Your 'access_token' did not return email of the user. Without 'email' user can't be logged in or registered. Get user email extended permission while joining the Facebook app.", 400);
        }
    }
```

```
    public function sms_login($request)
    {
        $access_token = $request["access_token"];
        if (!isset($access_token)) {
            return parent::sendError("invalid_login", "You must include a 'access_token' variable. Get the valid access_token for this app from Facebook API.", 400);
        }
        $url = 'https://graph.accountkit.com/v1.3/me/?access_token=' . $access_token;

        $WP_Http_Curl = new WP_Http_Curl();
        $result = $WP_Http_Curl->request($url, array(
            'method' => 'GET',
            'timeout' => 5,
            'redirection' => 5,
            'httpversion' => '1.0',
            'blocking' => true,
            'headers' => array(),
            'body' => null,
            'cookies' => array(),
        ));

        $result = json_decode($result, true);

        if (isset($result["phone"])) {
            $user_name = $result["phone"]["number"];
            $user_email = $result["phone"]["number"] . "@flutter.io";
            return $this->createSocialAccount($user_email, $user_name, $user_name, "", $user_name);
        } else {
            return parent::sendError("invalid_login", "Your 'access_token' did not return email of the user. Without 'email' user can't be logged in or registered. Get user email extended permission while joining the Facebook app.", 400);
        }
        return $response;

    }

    private function firebase_sms_verify_id_token($request)
    {
        $json = file_get_contents('php://input');
        $params = json_decode($json, TRUE);

        $id_token = $params["id_token"];
        if (!isset($id_token)) {
            return parent::sendError("invalid_login", "id_token is required", 400);
        }

        $helper = new FirebasePhoneAuthHelper();
        $result = $helper->verify_id_token($id_token);

        if (is_wp_error($result)) {
            return $result;
        }
        if ($result == false) {
            return parent::sendError("invalid_login", "id_token is invalid.", 400);
        }
        return $result;
    }

    private function firebase_sms_login($phone)
    {
        if (!isset($phone)) {
            return parent::sendError("invalid_login", "You must include a 'phone' variable.", 400);
        }
```

```
784 $domain = $_SERVER['SERVER_NAME'] == 'default_server' ? $_SERVER['HTTP_HOST'] : $_SERVER['SERVER_NAME'];
785         if (count(explode(".", $domain)) == 1) {
786             $domain = "flutter.io";
787         }
788         $user_name = $phone;
789         $user_email = $phone . "@" . $domain;
790         return $this->createSocialAccount($user_email, $user_name, $user_name, "", $user_name);
791     }
792
793     private function firebase_sms_login_v2($phone)
794     {
795         if (!isset($phone)) {
796             return parent::sendError("invalid_login", "You must include a 'phone' variable.", 400);
797         }
798
799         if (isset($phone)) {
800             $args = array('meta_key' => 'registered_phone_number', 'meta_value' => $phone);
801             $search_users = get_users($args);
802             if (empty($search_users)) {
803 $domain = $_SERVER['SERVER_NAME'] == 'default_server' ? $_SERVER['HTTP_HOST'] : $_SERVER['SERVER_NAME'];
804                 if (count(explode(".", $domain)) == 1) {
805                     $domain = "flutter.io";
806                 }
807                 $user_name = $phone;
808                 $user_email = $phone . "@" . $domain;
809                 $user = get_user_by('email', $user_email);
810                 if (!$user) {
811                     return parent::sendError("invalid_login", "User does not exist", 400);
812                 }
813                 $cookie = generateCookieByUserId($user->ID);
814                 $response['wp_user_id'] = $user->ID;
815                 $response['cookie'] = $cookie;
816                 $response['user_login'] = $user->user_login;
817                 $response['user'] = $this->getResponseUserInfo($user);
818                 return $response;
819             }
820             if (count($search_users) > 1) {
821                 return parent::sendError("invalid_login", "Too many users with the same phone number", 400);
822             }
823             $user = $search_users[0];
824             $cookie = generateCookieByUserId($user->ID);
825             $response['wp_user_id'] = $user->ID;
826             $response['cookie'] = $cookie;
827             $response['user_login'] = $user->user_login;
828             $response['user'] = $this->getResponseUserInfo($user);
829             return $response;
830         }
831         return parent::sendError("invalid_login", "Unknown Error", 400);
832     }
833
834
835     function jwtDecode($token)
836     {
837         $splitToken = explode(".", $token);
838         $payloadBase64 = $splitToken[1]; // Payload is always the index 1
839         $decodedPayload = json_decode(urldecode(base64_decode($payloadBase64)), true);
840         return $decodedPayload;
841     }
842
843     public function apple_login($request)
844     {
```

```
        $json = file_get_contents('php://input');
        $params = json_decode($json, TRUE);
        $authorization_code = $params["authorization_code"];
        $firstName = $params["first_name"];
        $lastName = $params["last_name"];
        $teamId = $params["team_id"];
        $bundleId = $params["bundle_id"];
        if(!FlutterAppleSignInUtils::is_file_existed()){
            return parent::sendError("invalid_login", "You need to upload AuthKey_XXXX.p8 file to MStore Api plugin", 400);
        }
        $token = AppleSignInHelper::generate_token($bundleId,$teamId,$authorization_code);
        if($token == false || is_wp_error($token)){
            return is_wp_error($token) ? $token : parent::sendError("invalid_login",  "Invalid authorization_code", 400);
        }
        $decoded = $this->jwtDecode($token);
        $user_email = $decoded["email"];
        if (!isset($user_email)) {
            return parent::sendError("invalid_login", "Can't get the email to create account.", 400);
        }
        $display_name = explode("@", $user_email)[0];
        if(isset($firstName) && isset($lastName) && !empty($firstName)){
            $display_name = $firstName.' '.$lastName;
        }else{
            $firstName =  $display_name;
            $lastName = "";
        }
        $user_name = $display_name;

        return $this->createSocialAccount($user_email, $display_name, $firstName, $lastName, $user_name);
    }

    public function google_login($request)
    {
        $access_token = $request["access_token"];
        if (!isset($access_token)) {
            return parent::sendError("invalid_login", "You must include a 'access_token' variable. Get the valid access_token for this app from Google API.", 400);
        }

        $url = 'https://www.googleapis.com/oauth2/v1/userinfo?alt=json&access_token=' . $access_token;

        $result = wp_remote_retrieve_body(wp_remote_get($url));

        $result = json_decode($result, true);
        if (isset($result["email"])) {
            $firstName = $result["given_name"];
            $lastName = $result["family_name"];
            $email = $result["email"];
            $display_name = $firstName . " " . $lastName;
            $user_name = $firstName . "." . $lastName;
            return $this->createSocialAccount($email, $display_name, $firstName, $lastName, $user_name);
        } else {
            return parent::sendError("invalid_login", "Your 'token' did not return email of the user. Without 'email' user can't be logged in or registered. Get user email extended permission while joining the Google app.", 400);
        }
    }

    /*
     * Post commment function
     */
    public function post_comment($request)
```

```
    {
        $cookie = $request["cookie"];
        $user_id = validateCookieLogin($cookie);
        if (is_wp_error($user_id)) {
            return $user_id;
        }
        if (!$request["post_id"]) {
            return parent::sendError("invalid_data", "No post specified. Include 'post_id' var in your request.", 400);
        } elseif (!$request["content"]) {
            return parent::sendError("invalid_data", "Please include 'content' var in your request.", 400);
        }

        $comment_approved = 0;
        $user_info = get_userdata($user_id);
        $time = current_time('mysql');

$agent = filter_has_var(INPUT_SERVER, 'HTTP_USER_AGENT') ? filter_input(INPUT_SERVER, 'HTTP_USER_AGENT') : 'Mozilla';

$ips = filter_has_var(INPUT_SERVER, 'REMOTE_ADDR') ? filter_input(INPUT_SERVER, 'REMOTE_ADDR') : '127.0.0.1';
        $data = array(
            'comment_post_ID' => $request["post_id"],
            'comment_author' => $user_info->user_login,
            'comment_author_email' => $user_info->user_email,
            'comment_author_url' => $user_info->user_url,
            'comment_content' => $request["content"],
            'comment_type' => '',
            'comment_parent' => 0,
            'user_id' => $user_info->ID,
            'comment_author_IP' => $ips,
            'comment_agent' => $agent,
            'comment_date' => $time,
            'comment_approved' => $comment_approved,
        );
        //print_r($data);
        $comment_id = wp_insert_comment($data);
        //add metafields
        $meta = json_decode(stripcslashes($request["meta"]), true);
        //extra function
        add_comment_meta($comment_id, 'rating', $meta['rating']);
        add_comment_meta($comment_id, 'verified', 0);

        return array(
            "comment_id" => $comment_id,
        );
    }

    public function get_currentuserinfo($request)
    {
        $cookie = $request["cookie"];
        if (isset($request["token"])) {
            $cookie = urldecode(base64_decode($request["token"]));
        }
        $user_id = validateCookieLogin($cookie);
        if (is_wp_error($user_id)) {
            return $user_id;
        }
        $user = get_userdata($user_id);
        return array(
            "user" => $this->getResponseUserInfo($user)
        );
    }

```

```
Line
964     /**
965      * Get Point Reward by User ID
966      *
967      * @return void
968      */
969     function get_points($request)
970     {
971         global $wc_points_rewards;
972         $user_id = (int)$request['user_id'];
973         $current_page = (int)$request['page'];
974 
975         $points_balance = WC_Points_Rewards_Manager::get_users_points($user_id);
976         $points_label = $wc_points_rewards->get_points_label($points_balance);
977         $count = apply_filters('wc_points_rewards_my_account_points_events', 5, $user_id);
978         $current_page = empty($current_page) ? 1 : absint($current_page);
979 
980         $args = array(
981             'calc_found_rows' => true,
982             'orderby' => array(
983                 'field' => 'date',
984                 'order' => 'DESC',
985             ),
986             'per_page' => $count,
987             'paged' => $current_page,
988             'user' => $user_id,
989         );
990         $total_rows = WC_Points_Rewards_Points_Log::$found_rows;
991         $events = WC_Points_Rewards_Points_Log::get_points_log_entries($args);
992 
993         return array(
994             'points_balance' => $points_balance,
995             'points_label' => $points_label,
996             'total_rows' => $total_rows,
997             'page' => $current_page,
998             'count' => $count,
999             'events' => $events
1000        );
1001    }
1002
1003    function update_user_profile()
1004    {
1005        global $json_api;
1006        $json = file_get_contents('php://input');
1007        $params = json_decode($json);
1008        $cookie = $params->cookie;
1009        $user_id = validateCookieLogin($cookie);
1010        if (is_wp_error($user_id)) {
1011            return $user_id;
1012        }
1013
1014        $user_update = array('ID' => $user_id);
1015        if (isset($params->user_pass)) {
1016            $user_update['user_pass'] = $params->user_pass;
1017        }
1018        if (isset($params->user_nicename)) {
1019            $user_update['user_nicename'] = $params->user_nicename;
1020        }
1021        if (isset($params->user_email)) {
1022            $user_update['user_email'] = $params->user_email;
1023        }
1024        if (isset($params->user_url)) {
1025            $user_update['user_url'] = $params->user_url;
1026        }
```

```
Line
1027        if (isset($params->display_name)) {
1028            $user_update['display_name'] = $params->display_name;
1029        }
1030        if (isset($params->first_name)) {
1031            $user_update['first_name'] = $params->first_name;
1032            update_user_meta($user_id, 'shipping_first_name', $params->first_name, '');
1033            update_user_meta($user_id, 'billing_first_name', $params->first_name, '');
1034        }
1035        if (isset($params->last_name)) {
1036            $user_update['last_name'] = $params->last_name;
1037            update_user_meta($user_id, 'shipping_last_name', $params->last_name, '');
1038            update_user_meta($user_id, 'billing_last_name', $params->last_name, '');
1039        }
1040        if (isset($params->phone)) {
1041            update_user_meta($user_id, 'shipping_phone', $params->phone, '');
1042            update_user_meta($user_id, 'billing_phone', $params->phone, '');
1043        }
1044        if (isset($params->shipping_company)) {
1045            update_user_meta($user_id, 'shipping_company', $params->shipping_company, '');
1046            update_user_meta($user_id, 'billing_company', $params->shipping_company, '');
1047        }
1048        if (isset($params->shipping_state)) {
1049            update_user_meta($user_id, 'shipping_state', $params->shipping_state, '');
1050            update_user_meta($user_id, 'billing_state', $params->shipping_state, '');
1051        }
1052        if (isset($params->shipping_address_1)) {
1053            update_user_meta($user_id, 'shipping_address_1', $params->shipping_address_1, '');
1054            update_user_meta($user_id, 'billing_address_1', $params->shipping_address_1, '');
1055        }
1056        if (isset($params->shipping_address_2)) {
1057            update_user_meta($user_id, 'shipping_address_2', $params->shipping_address_2, '');
1058            update_user_meta($user_id, 'billing_address_2', $params->shipping_address_2, '');
1059        }
1060        if (isset($params->shipping_city)) {
1061            update_user_meta($user_id, 'shipping_city', $params->shipping_city, '');
1062            update_user_meta($user_id, 'billing_city', $params->shipping_city, '');
1063        }
1064        if (isset($params->shipping_country)) {
1065            update_user_meta($user_id, 'shipping_country', $params->shipping_country, '');
1066            update_user_meta($user_id, 'billing_country', $params->shipping_country, '');
1067        }
1068        if (isset($params->shipping_postcode)) {
1069            update_user_meta($user_id, 'shipping_postcode', $params->shipping_postcode, '');
1070            update_user_meta($user_id, 'billing_postcode', $params->shipping_postcode, '');
1071        }
1072        if (isset($params->meta_data) && is_array($params->meta_data)) {
1073            foreach ($params->meta_data as $item) {
1074                update_user_meta($user_id, $item->key, $item->value, '');
1075            }
1076        }
1077
1078        if (isset($params->avatar)) {
1079            $count = 1;
1080            try {
1081                $attachment_id = upload_image_from_mobile($params->avatar, $count, $user_id);
1082                $url = wp_get_attachment_image_src($attachment_id);
1083                update_user_meta($user_id, 'user_avatar', $url, '');
1084             } catch (Exception $e) {
1085                return new WP_Error("invalid_avatar", $e->getMessage(), array('status' => 400));
1086            }
1087        }
1088
1089
```

```
1090        $user_data = wp_update_user($user_update);
1091
1092        if (is_wp_error($user_data)) {
1093            // There was an error; possibly this user doesn't exist.
1094            echo 'Error.';
1095        }
1096        $user = get_userdata($user_id);
1097
1098        if (isset($params->deviceToken)) {
1099            if (isset($params->is_manager) && $params->is_manager) {
1100                update_user_meta($user_id, "mstore_manager_device_token", $params->deviceToken);
1101            } else if (isset($params->is_delivery) && $params->is_delivery) {
1102                update_user_meta($user_id, "mstore_delivery_device_token", $params->deviceToken);
1103            }
1104            if (!isset($params->is_delivery) && !isset($params->is_manager)) {
1105                update_user_meta($user_id, "mstore_device_token", $params->deviceToken);
1106            }
1107            if(in_array('wcfm_delivery_boy', (array)$user->roles) || in_array('driver',(array)$user->roles)){
1108                update_user_meta($user_id, "mstore_delivery_device_token", $params->deviceToken);
1109            }
1110        }
1111
1112        return $this->getResponseUserInfo($user);
1113    }
1114
1115    function prepare_checkout()
1116    {
1117        global $json_api;
1118        $json = file_get_contents('php://input');
1119        $params = json_decode($json);
1120        $order = $params->order;
1121        if (!isset($order)) {
1122            return parent::sendError("invalid_checkout", "You must include a 'order' var in your
request", 400);
1123        }
1124        global $wpdb;
1125        $table_name = $wpdb->prefix . "mstore_checkout";
1126
1127        $code = md5(mt_rand() . strtotime("now"));
1128        $success = $wpdb->insert($table_name, array(
1129                'code' => $code,
1130                'order' => $order
1131            )
1132        );
1133        if ($success) {
1134            return $code;
1135        } else {
1136            return parent::sendError("error_insert_database", "Can't insert to database", 400);
1137        }
1138    }
1139
1140    public function get_currency_rates()
1141    {
1142        global $woocommerce_wpml;
1143
1144        if (!empty($woocommerce_wpml->multi_currency) && !empty($woocommerce_wpml-
>settings['currencies_order'])) {
1145            return $woocommerce_wpml->settings['currency_options'];
1146        }
1147        return parent::send_invalid_plugin_error("WooCommerce WPML hasn't been installed yet.");
1148    }
1149
1150    public function get_countries()
```

```
    {
        $wc_countries = new WC_Countries();
        $array = $wc_countries->get_countries();
        $keys = array_keys($array);
        $countries = array();
        for ($i = 0; $i < count($keys); $i++) {
            $countries[] = ["code" => $keys[$i], "name" => $array[$keys[$i]]];
        }
        return $countries;
    }

    public function get_states($request)
    {
        $wc_countries = new WC_Countries();
        $array = $wc_countries->get_states($request["country_code"]);
        if ($array) {
            $keys = array_keys($array);
            $states = array();
            for ($i = 0; $i < count($keys); $i++) {
                $states[] = ["code" => $keys[$i], "name" => $array[$keys[$i]]];
            }
            return $states;
        } else {
            return [];
        }
    }

    function chat_notification()
    {
        $json = file_get_contents('php://input');
        $params = json_decode($json, TRUE);
        $token = $params['token'];
        if (isset($token)) {
            $cookie = urldecode(base64_decode($token));
        } else {
            return parent::sendError("unauthorized", "You are not allowed to do this", 401);
        }
        $user_id = validateCookieLogin($cookie);
        if (is_wp_error($user_id)) {
            return $user_id;
        }
        $receiver_email = $params['receiver'];
        $sender_name = $params['sender'];
        if (is_email($sender_name)) {
            $sender = get_user_by('email', $sender_name);
            $sender_name = $sender->display_name;
        }
        $receiver = get_user_by('email', $receiver_email);

        if (!$receiver) {
            return parent::sendError("invalid_user", "User does not exist in this world. Please re-check user's existence with the Creator :)", 401);
        }

        $message = $params['message'];

        pushNotificationForUser($receiver->ID, $sender_name, $message);
        if (!is_plugin_active('onesignal-free-web-push-notifications/onesignal.php')) {//fix duplicate notification if onesignal
            pushNotificationForVendor($receiver->ID, $sender_name, $message);
        }
    }

```

```
Line
1212    function mstore_digrest_set_variables()
1213    {
1214        $json = file_get_contents('php://input');
1215        $params = json_decode($json, TRUE);
1216
1217        $_POST['digits'] = 1;
1218
1219        if (dig_isWhatsAppEnabled() && $params['whatsapp'] == true) {
1220            $_POST['whatsapp'] = 1;
1221        }
1222
1223        if (isset($params['type'])) {
1224            $type = $params['type'];
1225            if ($type == 'login') {
1226                $_REQUEST['login'] = 1;
1227            }
1228            if ($type == 'register') {
1229                $_REQUEST['login'] = 2;
1230            } else if ($type == 'resetpass') {
1231                $_REQUEST['login'] = 3;
1232            } else if ($type == 'update') {
1233                $_REQUEST['login'] = 11;
1234            }
1235        }else{
1236            $_REQUEST['login'] = 2;
1237        }
1238
1239        if (isset($params['mobile'])) {
1240            $_POST['digits_reg_mail'] = $params['mobile'];
1241        }
1242        if (isset($params['email'])) {
1243            $_POST['dig_reg_mail'] = $params['email'];
1244        }
1245        if (isset($params['username'])) {
1246            $_POST['digits_reg_username'] = $params['username'];
1247            $_POST['digits_reg_name'] = $params['username'];
1248        }
1249        if (isset($params['name'])) {
1250            $_POST['digits_reg_name'] = $params['name'];
1251        }
1252        if (isset($params['last_name'])) {
1253            $_POST['digits_reg_lastname'] = $params['last_name'];
1254        }
1255        if (isset($params['country_code'])) {
1256            $_POST['digregcode'] = $params['country_code'];
1257        }
1258        if (isset($params['otp'])) {
1259            $_POST['dig_otp'] = $params['otp'];
1260        }
1261        $_POST['ftoken'] = $params['ftoken'];
1262        $_REQUEST['ftoken'] = $params['ftoken'];
1263
1264        $_REQUEST['csrf'] = wp_create_nonce('crsf-otp');
1265        $_POST['csrf'] = wp_create_nonce('crsf-otp');
1266
1267        $_POST['dig_nounce'] = wp_create_nonce('dig_form');
1268        $_POST['crsf-otp'] = wp_create_nonce('crsf-otp');
1269
1270        if (isset($params['password'])) {
1271            $_POST['digits_reg_password'] = $params['password'];
1272        }else{
1273            $_POST['digits_reg_password'] = wp_generate_password();
1274        }
```

```
        $reg_custom_fields = stripslashes(base64_decode(get_option("dig_reg_custom_field_data", "e30=")));
        $reg_custom_fields = json_decode($reg_custom_fields, true);
        foreach ($reg_custom_fields as $key => $values) {
            $required = $values['required'];
            if($required == 1){
                $meta_key = cust_dig_filter_string($values['meta_key']);
                $post_index = 'digits_reg_' . $meta_key;
                $_POST[$post_index] = '1';
            }

        }
        $_REQUEST['json'] = 1;

        if (isset($params['referral_code'])) {
            $_COOKIE['woo_wallet_referral'] = sanitize_text_field( wp_unslash( $params['referral_code'] ) );
        }
    }

    function digits_register_check()
    {
        if(!function_exists('digits_create_user')) {
            return parent::sendError("plugin_not_found", "Please install  the  DIGITS: Wordpress Mobile Number Signup and Login  plugin", 400);
        }

        $json = file_get_contents('php://input');
        $params = json_decode($json, TRUE);

        if(empty($params['email'])){
            return parent::sendError("invalid_email", 'Email is required', 400);
        }
        if (!empty($params['email']) && email_exists($params['email'])) {
            return parent::sendError("existed_email", 'Email already in use!', 400);
        }

        if(empty($params['username'])){
            return parent::sendError("invalid_username", 'Username is required', 400);
        }
        if (!empty($params['username']) && username_exists($params['username'])) {
            return parent::sendError("existed_username", 'Username already in use!', 400);
        }

        if(empty($params['country_code'])){
            return parent::sendError("invalid_country_code", 'Country code is required', 400);
        }

        if(empty($params['mobile'])){
            return parent::sendError("invalid_mobile", 'Mobile is required', 400);
        }

        $mob = $params['country_code'].$params['mobile'];
        $mobuser = getUserFromPhone($mob);
        if ($mobuser != null  || username_exists($mob)) {
            return parent::sendError("existed_mobile", 'Mobile Number already in use!', 400);
        }

        return  true;
    }

    function digits_register()
    {
        if(!function_exists('digits_create_user')) {
```

```
Line
1337            return parent::sendError("plugin_not_found", "Please install  the  DIGITS: Wordpress Mobile Number Signup and Login  plugin", 400);
1338        }
1339
1340        define( 'REST_REQUEST', true );
1341        $this->mstore_digrest_set_variables();
1342        $userId = null;
1343        add_filter('digits_user_created_response', function($data, $user_id){
1344            $data['user_id'] = $user_id;
1345            return $data;
1346        },10, 2);
1347        $data = digits_create_user();
1348        define( 'REST_REQUEST', false );
1349        remove_filter('digits_user_created_response','__return_false', 10);
1350
1351        if ($data['success'] === false) {
1352            return parent::sendError("invalid_data", explode("<br />",  $data['data']['msg'])[0], 400);
1353        } else {
1354            $user_id = $data['user_id'];
1355            $cookie = generateCookieByUserId($user_id);
1356            $user = get_userdata($user_id);
1357
1358            $response['wp_user_id'] = $user_id;
1359            $response['cookie'] = $cookie;
1360            $response['user_login'] = $user->user_login;
1361            $response['user'] = $this->getResponseUserInfo($user);
1362            return $response;
1363        }
1364    }
1365
1366    function digits_login_check()
1367    {
1368        if(!function_exists('digits_create_user')) {
1369            return parent::sendError("plugin_not_found", "Please install  the  DIGITS: Wordpress Mobile Number Signup and Login  plugin", 400);
1370        }
1371
1372        $json = file_get_contents('php://input');
1373        $params = json_decode($json, TRUE);
1374
1375        if(empty($params['country_code'])){
1376            return parent::sendError("invalid_country_code", 'Country code is required', 400);
1377        }
1378
1379        if(empty($params['mobile'])){
1380            return parent::sendError("invalid_mobile", 'Mobile is required', 400);
1381        }
1382
1383        $mob = $params['country_code'].$params['mobile'];
1384        $mobuser = getUserFromPhone($mob);
1385        if ($mobuser == null) {
1386            return parent::sendError("not_existed_mobile", 'Phone number is not registered!', 400);
1387        }
1388
1389        return  true;
1390    }
1391
1392    function digits_login()
1393    {
1394        if(!function_exists('dig_validateMobileNumber')) {
1395            return parent::sendError("plugin_not_found", "Please install  the  DIGITS: Wordpress Mobile Number Signup and Login  plugin", 400);
1396        }
1397
```

```
1398        $this->mstore_digrest_set_variables();
1399
1400        $otp = $_POST['dig_otp'];
1401    $validateMob = dig_validateMobileNumber($_POST['digregcode'], $_POST['digits_reg_mail'], $otp, null, 1, null, false);
1402
1403        if ($validateMob['success'] === false) {
1404            return parent::sendError("invalid_data",$validateMob['msg'], 400);
1405        }
1406
1407        $user = getUserFromPhone($validateMob['countrycode'] . $validateMob['mobile']);
1408        $cookie = generateCookieByUserId($user->ID);
1409        $response['wp_user_id'] = $user->ID;
1410        $response['cookie'] = $cookie;
1411        $response['user_login'] = $user->user_login;
1412        $response['user'] = $this->getResponseUserInfo($user);
1413        return $response;
1414    }
1415
1416    function digits_send_otp()
1417    {
1418        if(!function_exists('digits_create_user')) {
1419            return parent::sendError("plugin_not_found", "Please install  the  DIGITS: Wordpress Mobile Number Signup and Login  plugin", 400);
1420        }
1421
1422        $json = file_get_contents('php://input');
1423        $params = json_decode($json, TRUE);
1424
1425        if(empty($params['country_code'])){
1426            return parent::sendError("invalid_country_code", 'Country code is required', 400);
1427        }
1428
1429        if(empty($params['mobile'])){
1430            return parent::sendError("invalid_mobile", 'Mobile is required', 400);
1431        }
1432
1433        $_REQUEST['countrycode'] =  $params['country_code'];
1434        $_REQUEST['mobileNo'] =  $params['mobile'];
1435        $_REQUEST['type'] =  $params['type'];
1436
1437        $this->mstore_digrest_set_variables();
1438
1439
1440        $_REQUEST['csrf'] = wp_create_nonce('dig_form');
1441        $_POST['csrf'] = wp_create_nonce('dig_form');
1442
1443        do_action('wp_ajax_nopriv_digits_check_mob');
1444    }
1445
1446    function digits_resend_otp()
1447    {
1448        if(!function_exists('digits_resendotp')) {
1449            return parent::sendError("plugin_not_found", "Please install  the  DIGITS: Wordpress Mobile Number Signup and Login  plugin", 400);
1450        }
1451
1452        $json = file_get_contents('php://input');
1453        $params = json_decode($json, TRUE);
1454
1455        if(empty($params['country_code'])){
1456            return parent::sendError("invalid_country_code", 'Country code is required', 400);
1457        }
```

```
1458
1459        if(empty($params['mobile'])){
1460            return parent::sendError("invalid_mobile", 'Mobile is required', 400);
1461        }
1462
1463        $_REQUEST['countrycode'] =  $params['country_code'];
1464        $_REQUEST['mobileNo'] =  $params['mobile'];
1465        $_REQUEST['type'] =  $params['type'];
1466
1467        $this->mstore_digrest_set_variables();
1468
1469
1470        $_REQUEST['csrf'] = wp_create_nonce('dig_form');
1471        $_POST['csrf'] = wp_create_nonce('dig_form');
1472
1473        digits_resendotp();
1474    }
1475
1476
1477    function custom_delete_item_permissions_check($request)
1478    {
1479        $cookie = get_header_user_cookie($request->get_header("User-Cookie"));
1480        if (isset($cookie) && $cookie != null && parent::checkApiPermission()) {
1481            $user_id = validateCookieLogin($cookie);
1482            if (is_wp_error($user_id)) {
1483                return false;
1484            }
1485            $request["id"] = $user_id;
1486            return true;
1487        } else {
1488            return false;
1489        }
1490    }
1491
1492    function delete_account($request)
1493    {
1494        if(checkWhiteListAccounts($request["id"])){
1495            return parent::sendError("invalid_account", "This account can't delete", 400);
1496        }else{
1497            require_once(ABSPATH.'wp-admin/includes/user.php');
1498            return wp_delete_user($request["id"]);
1499        }
1500    }
1501}
```

Developers

Get Involved

Learn

Showcase

Plugins

Themes

Patterns

WordCamp

WordPress.TV

BuddyPress

bbPress

WordPress.com

Matt

Privacy

Public Code