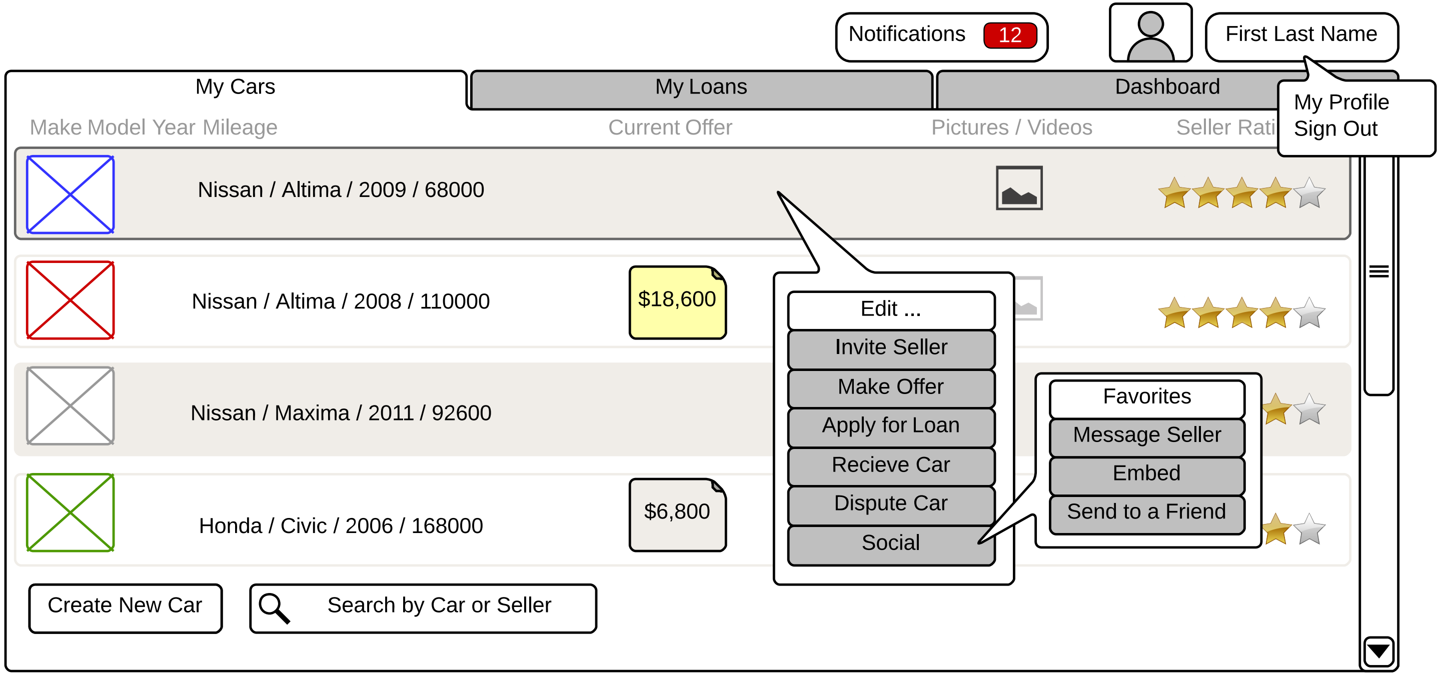
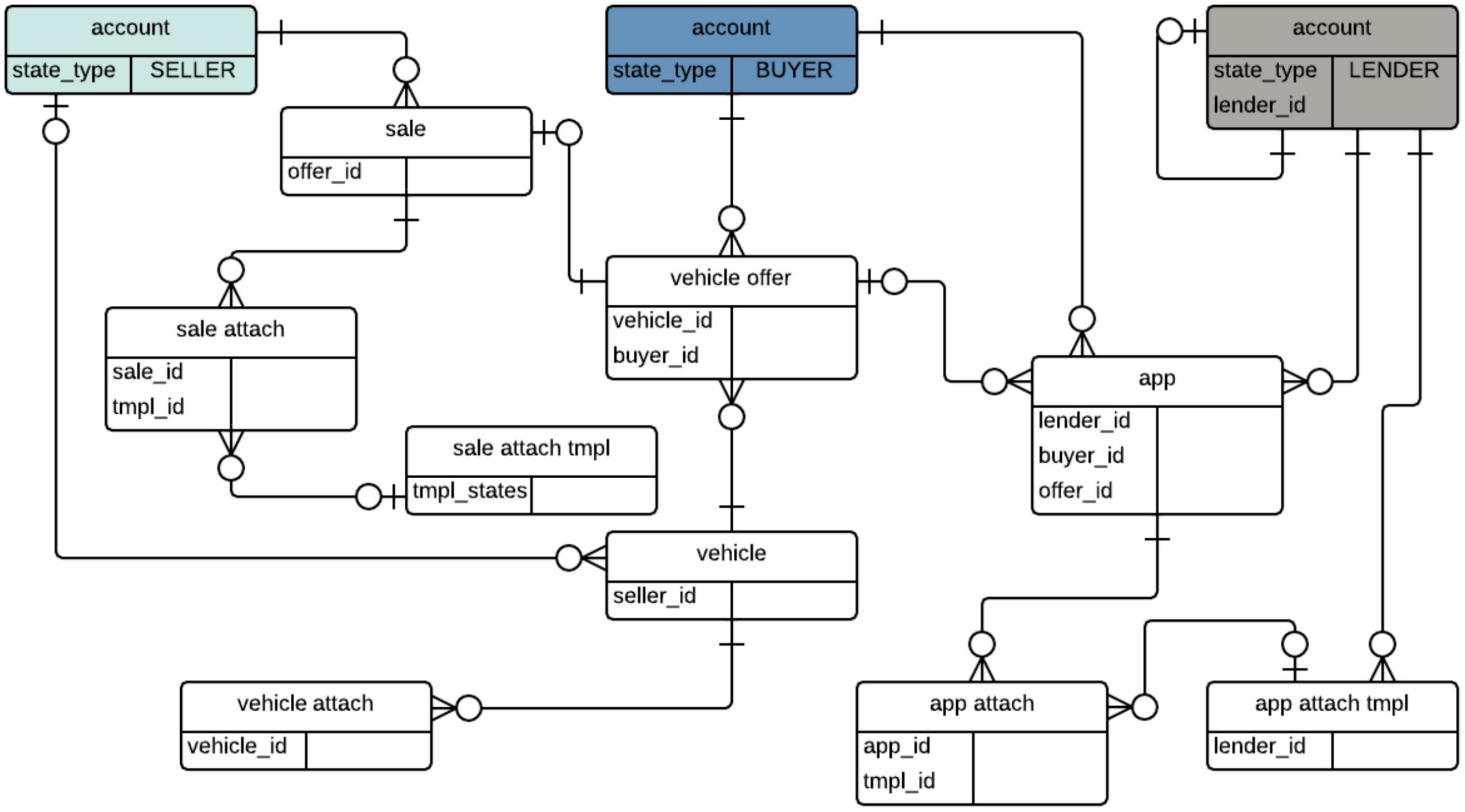
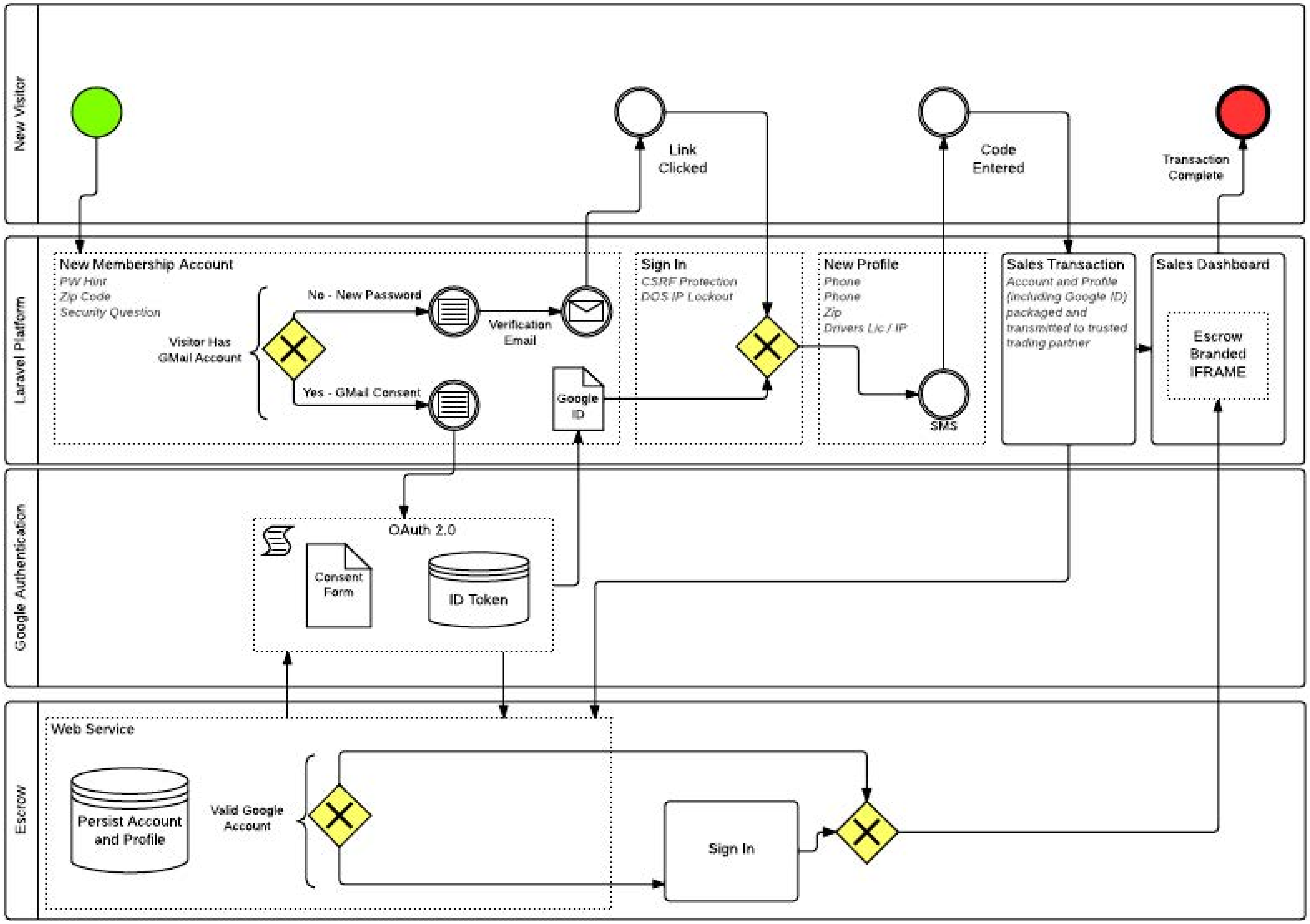


Make Model Year Mileage	Current Offer	Pictures / Videos	Seller Rati
Nissan / Altima / 2009 / 68000			★★★★☆
Nissan / Altima / 2008 / 110000	$18,600		★★★★☆
Nissan / Maxima / 2011 / 92600			★☆
Honda / Civic / 2006 / 168000	$6,800		★☆


account
state_type
SELLER
account
state_type
BUYER
account
state_type
LENDER
lender_id
sale
offer_id
vehicle offer
vehicle_id
buyer_id
app
lender_id
buyer_id
offer_id
sale attach
sale_id
tmpl_id
sale attach tmpl
tmpl_states
vehicle
seller_id
vehicle attach
vehicle_id
app attach
app_id
tmpl_id
app attach tmpl
lender_id

```php
<?php

class Notification extends \Eloquent {
    protected $fillable = [];
    protected $table = 'notification';

    public static function total_notifications(){
        //  Return the total number of notifications
        //  for the current user.  For top bar(s)

        return Notification::where('to_id', '=', Auth::user()->id)->where('active_ind', '=', 1)->where('broadcast_at',
         '=', null)->count();//
    }

    public static function notf_cy_admin($notification_caption, $notification_msg, $dislpay_class = 'info', $from =
    null) {
        //    Simple admin notify - Assumes the To is user 1 (cy_admin)

        if($from == null){
            //    If no user provided use auth
            $id = Auth::user()->id;
        }else{
            //    Try the email address
            $id = Account::where('email', '=', $from)->where('state_status', '=', Account::STATE_STATUS_ENABLED)->
            first();
            if(!$id){
                //    Try the sid
                $id = Account::where('sid', '=', $from)->where('state_status', '=', Account::STATE_STATUS_ENABLED)->
                first();
            }
            $id = $id->id;
        }
        Eloquent::unguard();
        $insert_notf = Notification::create(array(
                        'to_id' => 1,
                        'from_id' => $id,
                        'notification_caption' => $notification_caption,
                        'notification_msg' => $notification_msg,
                        'dislpay_class' => $dislpay_class
                        ));
        return true;
    }
```

```php


    public static function notf_self($notification_caption, $notification_msg, $dislpay_class = 'info', $user = null) {
        //    Simple self notify - Assumes the From is user 1 (cy_admin)

        if($user == null){
            //    If no user provided use auth
            $id = Auth::user()->id;
        }else{
            //    Try the email address
            $id = Account::where('email', '=', $user)->where('state_status', '=', Account::STATE_STATUS_ENABLED)->
            first();
            if(!$id){
                //    Try the sid
                $id = Account::where('sid', '=', $user)->where('state_status', '=', Account::STATE_STATUS_ENABLED)->
                first();
            }
            $id = $id->id;
        }

        Eloquent::unguard();
        $insert_notf = Notification::create(array(
                        'to_id' => $id,
                        'from_id' => 1,
                        'notification_caption' => $notification_caption,
                        'notification_msg' => $notification_msg,
                        'dislpay_class' => $dislpay_class
                        ));
        return true;
    }

    public static function renderNotf( $uid = null ) {
        //   Generate the markup for all active notifications for this user sorted by display class

        $mu = "";
        $Count = 0;

        $notfs = Self::where('to_id', '=', $uid)->where('active_ind', '=', 1)->where('broadcast_at', '=', null)->
        orderBy('created_at')->get();
        foreach ($notfs as $notf) {
            $notf_render =  new NotfAlert($notf);  // Laravel collection item
            $notf_render->addClass_alert($notf->dislpay_class);
            $notf_render->addClass_cap('cy_notf_cap_' . $notf->dislpay_class);
            //  Instance tiny avatar
            $avaTiny = new AvatarPanel_tiny();
            $avaTiny->setMember(Account::find( $notf->from_id ));
            $avaTiny->setFullName(Account::getFullName( $notf->from_id ));
            //
            $notf_render->addTitle_cap($avaTiny->render());
            $notf_render->addTitle_cap($notf->created_at);

            $mu .= $notf_render->renderMarkup();
            $notf_render = $avaTiny = null;
        }
        return $mu;
    }

    private static function createNotf( $uid = null ) {
        //
    }
}
```

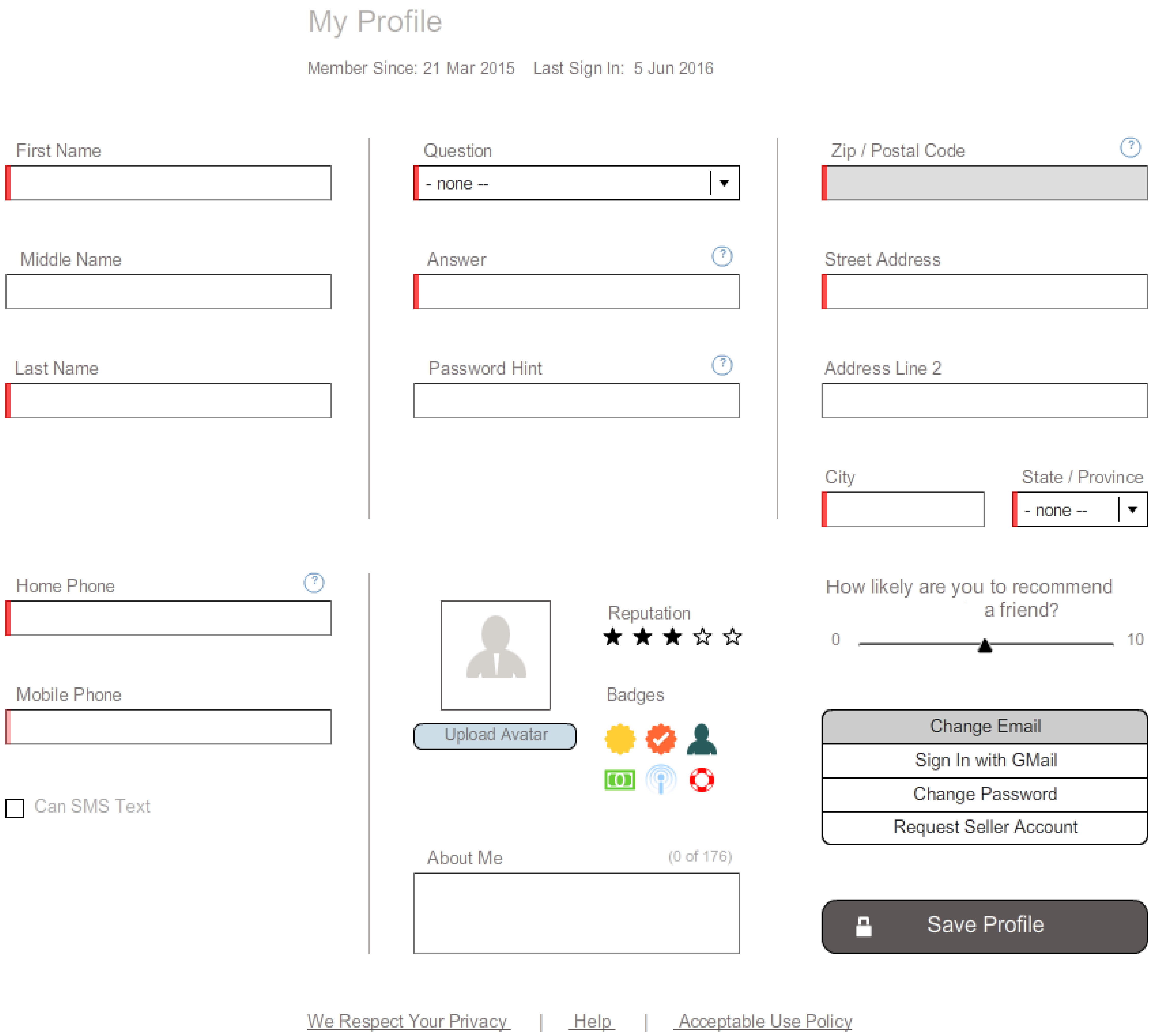

New Visitor
Laravel Platform
Google Authentication
Escrow
Link Clicked
Code Entered
Transaction Complete
New Membership Account
PW Hint
Zip Code
Security Question
Visitor Has GMail Account
No - New Password
Yes - GMail Consent
Verification Email
Google ID
Sign In
CSRF Protection
DOS IP Lockout
New Profile
Phone
Phone
Zip
Drivers Lic / IP
SMS
Sales Transaction
Account and Profile (including Google ID) packaged and transmitted to trusted trading partner
Sales Dashboard
Escrow Branded IFRAME
OAuth 2.0
Consent Form
ID Token
Web Service
Persist Account and Profile
Valid Google Account
Sign In

```php
<?php

class GuestController extends BaseController {

    /*
    |--------------------------------------------------------------------------
    |    SCK    These are the actions for users who are NOT authenticated
    |--------------------------------------------------------------------------
    */

    public function get_sign_in_via_sid($sid){
        //    If a valid sid was provided activate account and
        //    flash a welcome msg

        $account = Account::where('sid', '=', $sid)->where('state_status', '=', Account::STATE_STATUS_ENABLED)->firstOrFail();
        if ($account !== null) {
            if ($account->state_stage == Account::STATE_STAGE_INACTIVE){
                //    An accnt should only graduate to ACTIVE from INACTIVE
                Account::update_accnt_state_stage($account->id, Account::STATE_STAGE_ACTIVE);
                if (Notification::notf_self('Profile Reminder', 'Click <b>My Profile</b> to complete your CarsYours Profile.', "success", $account->sid)){
                }
                Session::push('all_msg_alerts',
                ['message' => 'Welcome '.$account->first_nm.'.&nbsp; Your membership account is now active.&nbsp; Please Sign In with your email address and password.',
                'severity' => 'success', 'icon' => '', 'content' => '']);
            }
        }

        return View::make('account.sign_in');
    }


    public function get_forgot_password_sid($sid){
        //    If a valid sid was provided within 1 hour
        //    swap geocode with tmp geocode, msg, notification,
        //    redirect to Sign In

        if(Account::forget_password_via_sid($sid)){
            if(Notification::notf_self('Password Reset', 'Your password has been reset.', 'success', $sid)){
```

```php
                Session::push('all_msg_alerts',
                ['message' => 'Please enter your new password exactly as it exists in your email message.',
                'severity' => 'success', 'icon' => '', 'content' => '']);

                Session::push('all_msg_alerts',
                ['message' => 'Note: We recommend that after you have Signed In you visit the My Profile page and create a new, more meaningful password that you can easily remember.',
                'severity' => 'secondary', 'icon' => '', 'content' => '']);
            }
            return Redirect::route('get_sign_in');
        }else{
            Session::push('all_msg_alerts',
            ['message' => 'The password reset has expired please try again.',
            'severity' => 'alert', 'icon' => '', 'content' => '']);
            return Redirect::route('get_forgot_password');
        }
    }


    public function get_forgot_password(){
        //    Enter just email
        return View::make('account.forgot_password');
    }


    public function post_forgot_password(){
        //    Given email get security question

        if(Input::get('SQ') == null) {
            //    From forgot_password_form
            $val = Validator::make(Input::all(), Account::$forgot_password_form);    //dd($val->messages());
            if($val->fails()){
                return Redirect::back()->withInput()->withErrors($val);
            }else{
                $auth = User::where('email', '=', Input::get('email'))->where('state_status', '=', Account::STATE_STATUS_ENABLED)->first();
                if($auth){
                    if($auth->state_stage == Account::STATE_STAGE_INACTIVE){
                        //  If Account state_stage = INACTIVE then alert and resend email.
                        Account::mail_new_membership_activation($auth->id);
                        Session::push('all_msg_alerts',
                        ['message' => 'Please check your email. Click on the activation link in the email message.
                        Check your spam folder',
                        'severity' => 'alert', 'icon' => '', 'content' => '']);
                        return Redirect::back()->withInput();
                    }
                    //    Put email into session
                    Session::put('forgot_password_sq', Input::get('email'));
                    Session::push('all_msg_alerts',
                    ['message' => Account::get_lkup('challenge_q_1', $auth->challenge_q_1),
                    'severity' => 'secondary', 'icon' => '', 'content' => '']);

                    return View::make('account.forgot_password_sq');
                }else{
                    Session::push('all_msg_alerts',
                    ['message' => 'Sorry, that email does not exist within CarsYours.com.',
                    'severity' => 'alert', 'icon' => '', 'content' => '']);
                    return Redirect::back()->withInput();
                }
            }
        }else{
            //    From forgot_password_sq_form
            $val = Validator::make(Input::all(), Account::$forgot_password_sq_form);
            if($val->fails()){
                return Redirect::back()->withInput()->withErrors($val);
            }else{
                $auth = User::where('email', '=', Input::get('SQ'))->where('state_status', '=', Account::
                STATE_STATUS_ENABLED)->first();
                if($auth){
                    if(strcasecmp(Input::get('security_answer'), $auth->challenge_a_1) == 0){
                        Account::mail_forgot_password($auth->id);
                        Session::push('all_msg_alerts',
                        ['message' => 'You will receive a password reset email.&nbsp; Click on the link in the email
                        within 1 hour to reset your password.',
                        'severity' => 'success', 'icon' => '', 'content' => '']);
                        return Redirect::route('home');
                    }else{
                        Session::push('all_msg_alerts',
                        ['message' => 'Incorrect Answer.',
                        'severity' => 'alert', 'icon' => '', 'content' => '']);
                        return Redirect::route('home');
                    }
                }
            }
```

```php
        }
    }


    public function get_sign_in(){
        //
        return View::make('account.sign_in');
    }


    public function post_sign_in(){
        //    Validate then Sign In the user
        //

        $val = Validator::make(Input::all(), Account::$sign_in_form);    //dd($val->messages());
        if($val->fails()){
            return Redirect::back()->withInput()->withErrors($val);
        }else{

            $auth = User::where('email', '=', Input::get('email'))->where('state_status', '=', Account::
            STATE_STATUS_ENABLED)->first();
            if($auth){
                if($auth->state_stage == Account::STATE_STAGE_INACTIVE){
                    //  If Account state_stage = INACTIVE then alert and resend email.
                    Account::mail_new_membership_activation($auth->id);
                    Session::push('all_msg_alerts',
                    ['message' => 'Please check your email. Click on the activation link in the email message.  Check
                    your spam folder',
                    'severity' => 'alert', 'icon' => '', 'content' => '']);
                    return Redirect::back()->withInput();
                }
                if(Hash::check(Input::get('password'), $auth->geocode)){
                    if($auth->locked_at != null){
                        //    The account is locked.  Msg user and go back.
                        if(Account::accnt_locked_at($auth->id, $auth->locked_at)){
                            if(Notification::notf_self('Account Locked', 'Your account was locked due to invalid Sign
                            In attempts.', 'warning', Input::get('email'))){
                                Session::push('all_msg_alerts',
                                ['message' => 'Your account has been temporarily locked. Please try again later.',
                                'severity' => 'warning', 'icon' => '', 'content' => '']);
                            }
                            return Redirect::back()->withInput();
                        }
                    }
                    //    Sign In this guest
                    //    UPDATE the audit_last_sign_in_at / null out sign_in_attempts
                    $update_accnt_sign_in_audit = Account::find($auth->id);
                    $update_accnt_sign_in_audit->audit_last_sign_in_at = date("Y-m-d H:i:s");
                    $update_accnt_sign_in_audit->sign_in_attempts = null;
                    $update_accnt_sign_in_audit->save();
                    //  Guest becomes Member
                    Session::flush();
                    Auth::login($auth);
                    Account::post_sign_in_session();
                    //  Contruct Msg: No of failed attempts
                    if($auth->sign_in_attempts == null) {
                        $attempts = 0;
                    }else{
                        $attempts = $auth->sign_in_attempts;
                    };
                    //  Contruct Msg: Last Login
                    if($auth->audit_last_sign_in_at == null) {
                        $last_sign_in = '';
                    }else{
                        $last_sign_in = 'Last Sign In: '.date('d/M/Y', strtotime($auth->audit_last_sign_in_at)).' ';
                    };
                    Session::push('all_msg_alerts',
                    ['message' => $last_sign_in.'Failed Sign In attempts since last Sign In: '.$attempts,
                    'severity' => 'success', 'icon' => '', 'content' => '']);
                    return View::make('dashboard.'.Session::get('app_config')[0]['app_role']);
                }else{    // invalid pw
                    Account::increment_sign_in_attempts($auth->id);
                    Session::push('all_msg_alerts',
                    ['message' => 'Invalid email / password&nbsp;',
                    'severity' => 'alert', 'icon' => '', 'content' => '']);
                    if(($auth->sign_in_attempts == 1) AND ($auth->passw_hint != null)) {
                        Session::push('all_msg_alerts',
                        ['message' => 'Password Hint: '.$auth->passw_hint,
                        'severity' => 'info', 'icon' => '', 'content' => '']);
                    };
                    if($auth->sign_in_attempts == 4) {
                        Session::push('all_msg_alerts',
                        ['message' => 'Warning: After 6 consecutive failed Sign In attempts the Account will be locked for 30 minutes',
```

```
                        'severity' => 'info', 'icon' => '', 'content' => '']);
                    };
                    return Redirect::back()->withInput();
                }
            }else{   // account row NOT found
                //    log this
                Session::push('all_msg_alerts',
                ['message' => 'Invalid email / password',
                'severity' => 'alert', 'icon' => '', 'content' => '']);
                return Redirect::back()->withInput();
            } // account row found - active
        } // Validation passed
    }
}

```



My Profile

Member Since: 21 Mar 2015 Last Sign In: 5 Jun 2016

First Name

Middle Name

Last Name

Question: - none --

Answer

Password Hint

Zip / Postal Code

Street Address

Address Line 2

City

State / Province: - none --

Home Phone

Mobile Phone

☐ Can SMS Text

Reputation ★★★☆☆

Badges

About Me (0 of 176)

How likely are you to recommend a friend?

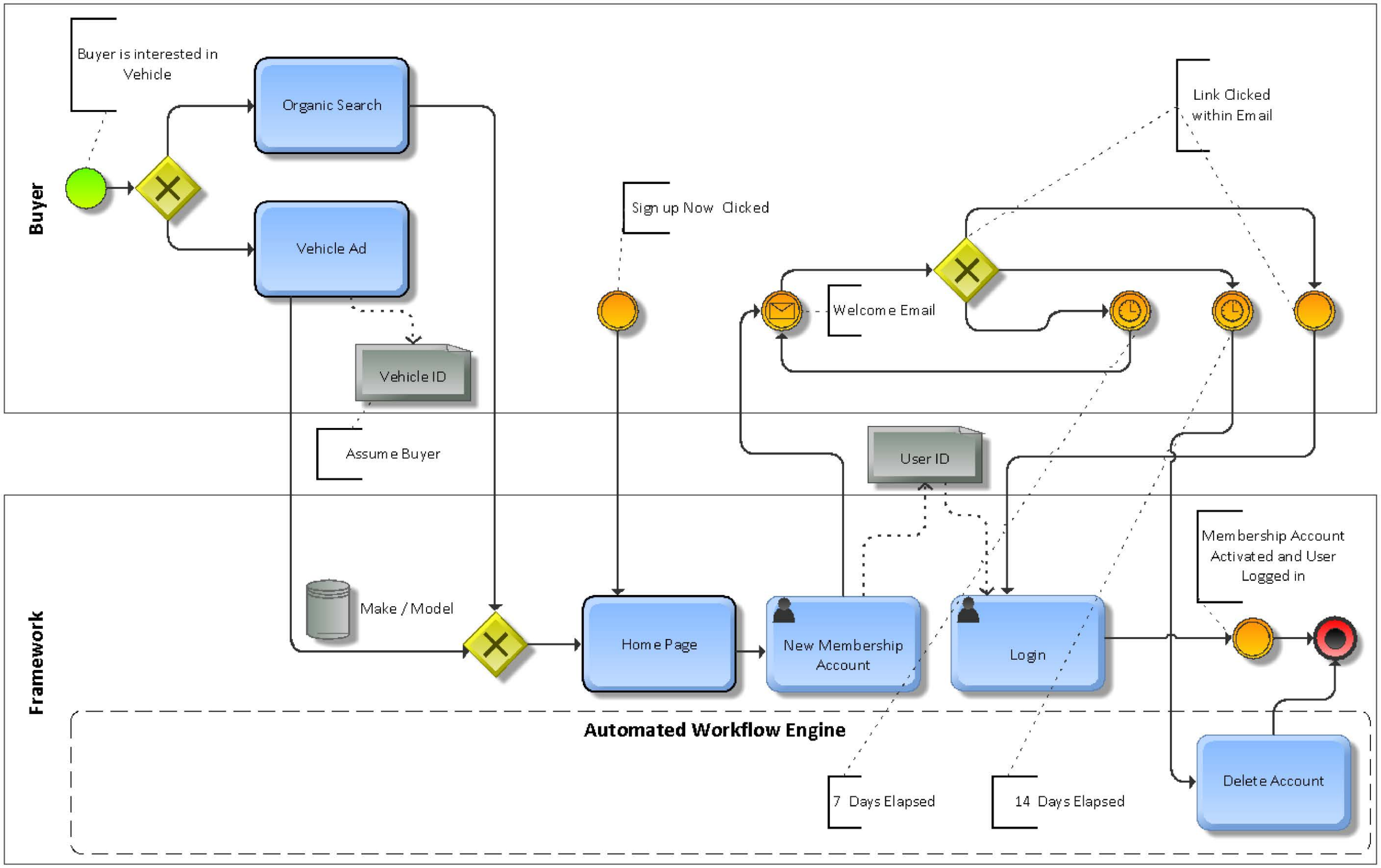
0 10

We Respect Your Privacy | Help | Acceptable Use Policy

```php
<?php

// Render in standard Bootstrap / Foundation 5 markup a single notification

Class NotfAlert {

    protected $_id;                           //  int(11) NOT NULL AUTO_INCREMENT COMMENT 'Primary Key / Auton'
    protected $_to_id;                        //  int(11) NOT NULL COMMENT 'Foreign Key: account  > ID'
    protected $_from_id;                      //  int(11) DEFAULT '1' COMMENT 'Foreign Key: account  > ID or Null'
    protected $_created_at;                   //  datetime DEFAULT NULL COMMENT 'audit: eloquent ORM'
    protected $_notification_caption;         //  varchar(32) NOT NULL COMMENT Alert Caption
    protected $_updated_at;                   //  datetime DEFAULT NULL
    protected $_notification_msg;             //  varchar(176) NOT NULL COMMENT 'Notification Message'
    protected $_notification_action;          //  varchar(176) DEFAULT NULL COMMENT 'URI or Route'
    protected $_dislpay_class;                //  varchar(32) DEFAULT 'standard' COMMENT 'standard success alert
    secondary'
    protected $_passive_ind;                  //  int(11) DEFAULT NULL COMMENT 'Display passively within UI'
    protected $_active_ind;                   //  int(11) DEFAULT '1' COMMENT 'Display as integrated dialog within UI'
    protected $_email_ind;                    //  int(11) DEFAULT NULL COMMENT 'Send as an email message'
    protected $_sms_ind;                      //  int(11) DEFAULT NULL COMMENT 'Send as a text message'
    protected $_broadcast_at;                 //  datetime DEFAULT NULL COMMENT 'Time sent or alert closed if
    applicable'
    protected $_state_status;                 //  varchar(32) NOT NULL DEFAULT 'ENABLED' COMMENT 'ENABLED,  DISABLED
    or ARCHIVED'

    protected $_mu_wrap = '<div ##ID## ##CLASS## ##ATRIB## ##TITLE##>';
    protected $_mu_wrap_close = '</div>';

    protected $_mu_a = '<a href=\"#\" ##ATRIB## ##CLASS##>';
    protected $_mu_a_close = '</a>';

    protected $_mu_class_columns = ['large-12','columns'];
    protected $_mu_atrib_columns = [];

    protected $_mu_class_close = ['close','cy_notf_cls','fa','fa-times'];
    protected $_mu_atrib_close = [];

    protected $_mu_class_alert = ['alert-box','radius'];
    protected $_mu_atrib_alert = ['data-alert'];

    protected $_mu_class_cap = ['cy_notf_cap', 'tip-left', 'text-center'];
    protected $_mu_atrib_cap = ['data-tooltip'];
```

```php
    protected $_mu_title_cap = [];

    protected $_mu_class_msg = ['cy_notf_msg'];




    public function __construct($notf = null) {
        //    Lets grab the whole collection

        if ($notf !== null) {
            $this->_id                    = $notf->id;
            $this->_to_id                     = $notf->to_id;
            $this->_from_id                   = $notf->from_id;
            $this->_created_at                = $notf->created_at;
            $this->_notification_caption      = $notf->notification_caption;
            $this->_updated_at                = $notf->updated_at;
            $this->_notification_msg          = $notf->notification_msg;
            $this->_notification_action       = $notf->notification_action;
            $this->_dislpay_class             = $notf->dislpay_class;
            $this->_passive_ind               = $notf->passive_ind;
            $this->_active_ind                = $notf->active_ind;
            $this->_email_ind                 = $notf->email_ind;
            $this->_sms_ind                   = $notf->sms_ind;
            $this->_broadcast_at              = $notf->broadcast_at;
            $this->_state_status              = $notf->state_status;
        }
    }


    public function addClass_alert($className) {
        //    Add a new class to the wrapper element

        $this->_mu_class_alert[] = $className;
    }


    public function addClass_cap($className) {
        //    Add a new class to the caption element

        $this->_mu_class_cap[] = $className;
    }


    public function addTitle_cap($Title) {
        //    Add a new title to the caption element

        $this->_mu_title_cap[] = $Title;
    }


    public function renderMarkup() {
        //    Construct the markup for this alert.  Assign DOM id based on DB Pkey

        $wrap  = str_replace("##CLASS##", 'class="' . implode(" ", $this->_mu_class_columns) . '"', $this->_mu_wrap);
        $wrap  = str_replace("##ATRIB##", implode(" ", $this->_mu_atrib_columns), $wrap);
        $wrap  = str_replace("##ID##", ' id="' . 'notf_' . $this->_id . '"', $wrap);
        $wrap  = str_replace("##TITLE##", '', $wrap);

        $cap   = str_replace("##CLASS##", 'class="' . implode(" ", $this->_mu_class_cap) . '"', $this->_mu_wrap);
        $cap   = str_replace("##ATRIB##", implode(" ", $this->_mu_atrib_cap), $cap);
        $cap   = str_replace("##ID##", '', $cap);
        $cap   = str_replace("##TITLE##", 'title="' . implode("<br>", $this->_mu_title_cap) . '"', $cap);

        $alert = str_replace("##CLASS##", 'class="' . implode(" ", $this->_mu_class_alert) . '"', $this->_mu_wrap);
        $alert = str_replace("##ATRIB##", implode(" ", $this->_mu_atrib_alert), $alert);
        $alert = str_replace("##ID##", '', $alert);
        $alert = str_replace("##TITLE##", '', $alert);

        $msg   = str_replace("##CLASS##", 'class="' . implode(" ", $this->_mu_class_msg) . '"', $this->_mu_wrap);
        $msg   = str_replace("##ATRIB##", '', $msg);
        $msg   = str_replace("##ID##", '', $msg);
        $msg   = str_replace("##TITLE##", '', $msg);

        $close = str_replace("##CLASS##", 'class="' . implode(" ", $this->_mu_class_close) . '"', $this->_mu_a);
        $close = str_replace("##ATRIB##", implode(" ", $this->_mu_atrib_close), $close);

        $mu  = $wrap . "\r\n";
            $mu .= $alert . "\r\n";

                //  Caption
                $mu .= $cap . "\r\n";
                $mu .= (string) $this->_notification_caption;
                $mu .= $this->_mu_wrap_close . "\r\n";
```

```php

                //  Close Command
                $mu .= $close;
                $mu .= $this->_mu_a_close . "\r\n";

                //  Message
                $mu .= $msg . "\r\n";
                $mu .= (string) $this->_notification_msg . "\r\n";
                $mu .= $this->_mu_wrap_close . "\r\n";

                //  Action
                if ($this->_notification_action !== null) {
                    $mu .= $this->_notification_action;
                }

            $mu .= $this->_mu_wrap_close . "\r\n";
        $mu .= $this->_mu_wrap_close . "\r\n";

        return $mu;
    }


    public function __toString() {
        //    Show markup when echo

        return $this->renderMarkup();
    }
}
```


Buyer
Framework
Buyer is interested in Vehicle
Organic Search
Vehicle Ad
Vehicle ID
Assume Buyer
Sign up Now Clicked
Welcome Email
Link Clicked within Email
User ID
Make / Model
Home Page
New Membership Account
Login
Membership Account Activated and User Logged in
Automated Workflow Engine
7 Days Elapsed
14 Days Elapsed
Delete Account

```php
<?php

class Account extends \Eloquent {
    protected $fillable = [];
    protected $table = 'account';

    //    state_type column values
    const STATE_TYPE_BUYER = 'BUYER';                   // Member who intends to buy a vehicle
    const STATE_TYPE_SELLER = 'SELLER';                 // Member who intends to sell a vehicle
    const STATE_TYPE_BUYER_SELLER = 'BUYER_SELLER';     // Member who intends to buy and sell a vehicle
    const STATE_TYPE_LENDER = 'LENDER';                 // Loan Officer / Portfolio Manager
    const STATE_TYPE_AUTO_TECH = 'AUTO_TECH';           // Auto Technician
    const STATE_TYPE_SERVICE_REP = 'SERVICE_REP';       // Customer Service
    const STATE_TYPE_ADMIN = 'ADMIN';                   // System Administration

    //    state_stage column values
    const STATE_STAGE_INACTIVE = 'INACTIVE';            // Has not logged in - UID token in query string has not been Activated ( or 6 failed Sign In attempts)
    const STATE_STAGE_ACTIVE = 'ACTIVE';                // Has logged in - UID token in query string has been Activated
    const STATE_STAGE_PROFILED = 'PROFILED';            // Has a valid profile (check zip)

    //    state_status column values
    const STATE_STATUS_ENABLED = 'ENABLED';             //
    const STATE_STATUS_DISABLED = 'DISABLED';           //
    const STATE_STATUS_ARCHIVED = 'ARCHIVED';           //

    const SMS_PHONE_NUMBER = "99999999999999";
// TODO Add below logic to Vehicle_offer
private static $verbsCar = [
    'new car' =>             '|BUYER|SELLER|BUYER_SELLER|',
    'car make / model' =>    '|BUYER|SELLER|BUYER_SELLER|',
    'buyer name' =>          '|SELLER|BUYER_SELLER|',
    'seller name' =>         '|BUYER|BUYER_SELLER|',
    'edit' =>                '|BUYER|SELLER|BUYER_SELLER|',
    'favorites' =>           '|BUYER|SELLER|BUYER_SELLER|',
    'embed / qr code' =>     '|BUYER|SELLER|BUYER_SELLER|',
    'certify vehicle' =>     '|SELLER|BUYER_SELLER|',
    'send to a friend' =>    '|BUYER|SELLER|BUYER_SELLER|',
    'invite seller' =>       '|BUYER|BUYER_SELLER|',
    'invite buyer' =>        '|SELLER|BUYER_SELLER|',
    'validate vehicle' =>    '|BUYER|SELLER|BUYER_SELLER|',
    'make offer' =>          '|BUYER|SELLER|BUYER_SELLER|',
```

```php
    'commit price' =>        '|BUYER|SELLER|BUYER_SELLER|',
    'apply for loan' =>      '|BUYER|BUYER_SELLER|',
    'view seller profile' => '|BUYER|BUYER_SELLER|',
    'message seller' =>      '|BUYER|BUYER_SELLER|',
    'message all buyers' =>  '|SELLER|BUYER_SELLER|',
    'sell' =>                '|SELLER|BUYER_SELLER|',
    'receive possession' =>  '|BUYER|BUYER_SELLER|',
    'obsolete' =>            '|SELLER|BUYER_SELLER|',
    'dispute condition' =>   '|BUYER|BUYER_SELLER|',
    'not interested' =>      '|BUYER|BUYER_SELLER|',
    'view photos' =>         '|BUYER|SELLER|BUYER_SELLER|',
    'upload attachement' =>  '|SELLER|BUYER_SELLER|',
    'for sale sign' =>       '|SELLER|BUYER_SELLER|'
];


public static function verbsCar($uid){
    //    Expose phase gate verb matrix for self entity

    $rtrn = null;
    if($uid != null){
        $entity = Self::where('id', '=', $uid)->where('state_status', '=', Self::STATE_STATUS_ENABLED)->first();
        if($entity != null){
            $verbs = [];
            foreach(Self::$verbsCar as $verb_nm => $verb_state_type){
                if(strpos( $verb_state_type, ('|' . $entity->state_type . '|') ) !== false){
                    $verbs[$verb_nm] = 1;
                }
            }
            $rtrn = $verbs;
        }
    }
    return $rtrn;
}

    //    In order of severity - alert color step
    public static $rules_new_membership_form = array(
        'site_terms' => 'required|accepted',
        'email' => 'required|unique:account|email|same:verify_email|max:96',
        'password' => 'required|min:8|alpha_num|same:verify_password|max:176',
        'password_hint' => 'different:password|max:176',
        'security_answer' => 'required|different:password|max:64',
        'state_type' => 'required|max:32',
        'first_name' => 'required|max:128',
        'last_name' => 'required|max:128',
        'zip_code' => 'required|max:10',
        'security_question' => 'required|max:32'
    );
    public static $sign_in_form = array(
        'email' => 'required|email|max:96',
        'password' => 'required'
    );
    public static $forgot_password_form = array(
        'email' => 'required|email|max:96'
    );
    public static $forgot_password_sq_form = array(
        'security_answer' => 'required|max:64'
    );
    public static $post_accnt_change_password = array(
        'old_password' => 'required|min:8|alpha_num|max:176',
        'new_password' => 'required|min:8|alpha_num|same:verify_password|max:176',
        'password_hint' => 'different:new_password|max:176'
    );
    public static $post_accnt_change_email = array(
        'password' => 'required|min:8|alpha_num|max:176',
        'email' => 'required|unique:account|email|same:verify_email|max:96'
    );
    public static $rules_my_profile_form = array(
        'first_name' => 'required|max:128',
        'last_name' => 'required|max:128',
        'street_1' => 'required|max:176',
        'zip_code' => 'required|max:10',
        'security_answer' => 'required|max:64'
    );


    public static function create_new_inactive_accnt(){
        //    Create the account row and related activity row

        Eloquent::unguard();
        DB::beginTransaction();
            //    INSERT row with an INACTIVE state_stage
            $insert_new_member_accnt = Account::create(array(
                'state_type' => Input::get('state_type'),
```

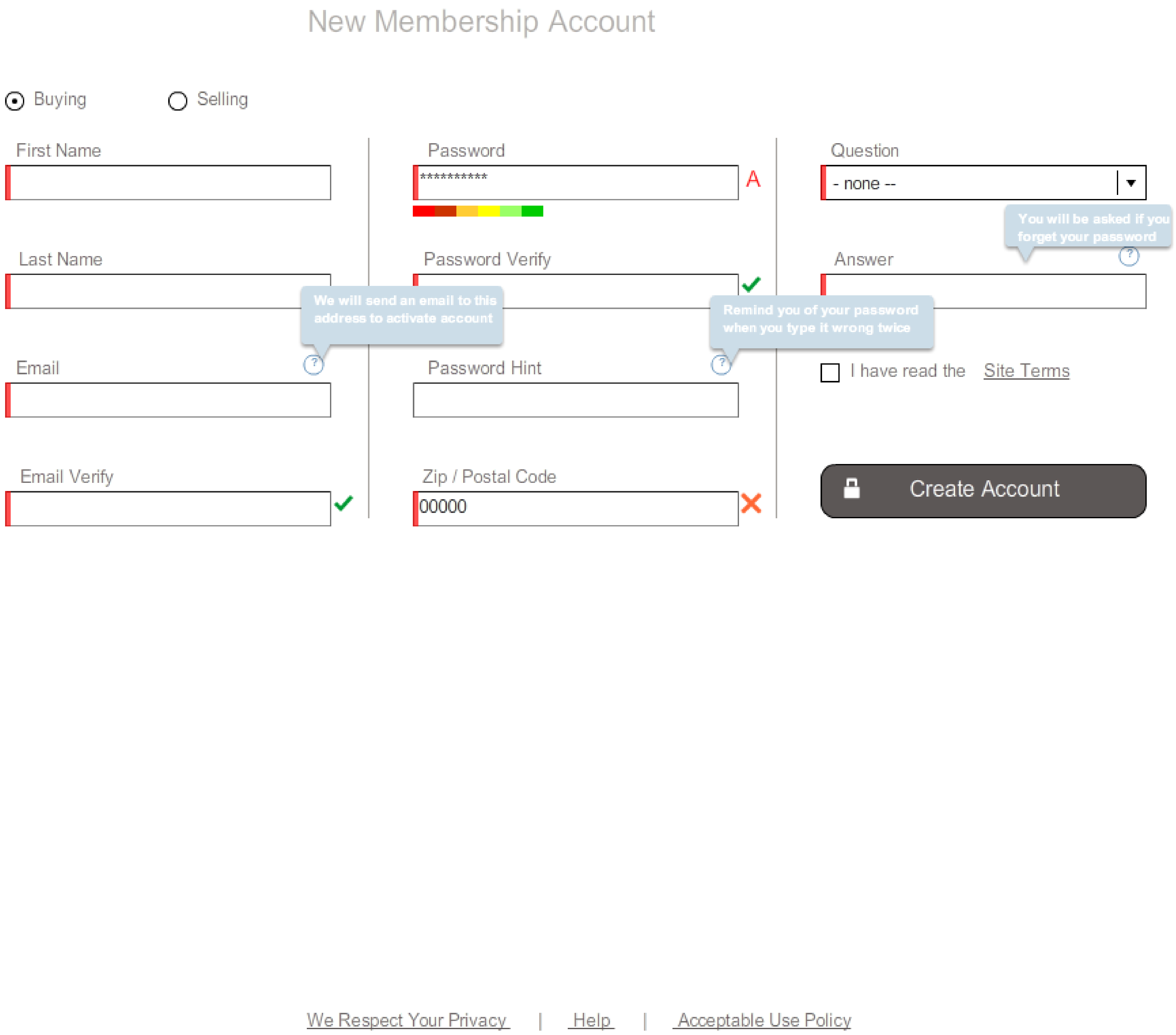
```
                'state_stage' => Account::STATE_STAGE_INACTIVE,
                'email' => Input::get('email'),
                'email_lnk_tmp' => Hash::make(Input::get('email')),
                'geocode' => Hash::make(Input::get('password')),
                'passw_hint' => Input::get('password_hint'),
                'challenge_q_1' => Input::get('security_question'),
                'challenge_a_1' => Input::get('security_answer'),
                'first_nm' => ucfirst(Input::get('first_name')),
                'last_nm' => ucfirst(Input::get('last_name')),
                'zip' => Input::get('zip_code'),
                'profile_avatar' => 'avatar_' . rand(1, 10) . '.png',
                'state_status' => Account::STATE_STATUS_ENABLED
                ));
            //    INSERT actv
            $insert_accnt_actv = Account_activity::create(array(
                'parent_id' => $insert_new_member_accnt->id,
                'state_stage' => $insert_new_member_accnt->state_stage
                ));
            //    UPDATE row with sid
            $update_new_member_accnt = Account::find($insert_new_member_accnt->id);
            $update_new_member_accnt->sid = Account::spice($insert_new_member_accnt->id, 243);
            $update_new_member_accnt->save();
        DB::commit();
        Account::zip_pop_new_inactive_accnt($update_new_member_accnt->id);
        return $insert_new_member_accnt->id;
    }


    public static function update_profiled_accnt(){
        //    Update the account row and related activity row
        //    from My Profile

        Eloquent::unguard();
        $is_profiled = null;
        $update_profiled_accnt = Account::find(Auth::user()->id);
        $update_profiled_accnt->first_nm           = Input::get('first_name');
        $update_profiled_accnt->middle_nm          = Input::get('middle_nm');
        $update_profiled_accnt->last_nm            = Input::get('last_name');
        $update_profiled_accnt->phone_home         = Input::get('phone_home');
        $update_profiled_accnt->phone_mobile       = Input::get('phone_mobile');
        $update_profiled_accnt->phone_sms_verified = Input::get('phone_sms_verified');
        $update_profiled_accnt->challenge_q_1      = Input::get('challenge_q_1');
```

```php
        $update_profiled_accnt->challenge_a_1      = Input::get('security_answer');
        $update_profiled_accnt->passw_hint         = Input::get('password_hint');
        $update_profiled_accnt->profile_nps        = Input::get('profile_nps');
        $update_profiled_accnt->zip                = Input::get('zip_code');
        $update_profiled_accnt->street_1           = Input::get('street_1');
        $update_profiled_accnt->street_2           = Input::get('street_2');
        $update_profiled_accnt->city               = Input::get('city');
        $update_profiled_accnt->state_province     = Input::get('state_province');
        $update_profiled_accnt->county_nm          = Input::get('county_nm');
        $update_profiled_accnt->profile_avatar     = Input::get('profile_avatar');
        $update_profiled_accnt->profile_desc       = Input::get('profile_desc');
        if($update_profiled_accnt->state_stage == Self::STATE_STAGE_PROFILED){
            $is_profiled = true;
        }
        $update_profiled_accnt->state_stage        = Self::STATE_STAGE_PROFILED;
        $update_profiled_accnt->save();
        //    INSERT actv
        if (!$is_profiled) {
            $insert_accnt_actv = Account_activity::create(array(
                'parent_id' => $update_profiled_accnt->id,
                'state_stage' => $update_profiled_accnt->state_stage
                ));
        }
    }


    public static function zip_pop_new_inactive_accnt($uid){
        //    Populate the accnt county, city and st based on zip meta

        $accnt = Account::find($uid);
        if($accnt){
            $met_zip = Met_zip_lkup::where('zipcode', '=', substr($accnt->zip, 0, 5))->where('primary', '=', 1)->first();
            if($met_zip){
                $accnt->county_nm = $met_zip->county_nm;
                $accnt->city = $met_zip->city;
                $accnt->state_province = $met_zip->state_province;
                $accnt->save();
            }else{
            }
        }else{
            //dd(DB::getQueryLog());
```

```php
        }
    }


    public static function mail_new_membership_activation($uid){
        //    Send an email containing a link that can activate the account
        $recipient = Account::find($uid);
        Mail::send('emails.account.new_membership_activation',
            array('first_nm' => $recipient->first_nm,
                'last_nm' => $recipient->last_nm,
                'url' => URL::Route('get_sign_in_via_sid',$recipient->sid)),
            function($message) use ($recipient) {
        $message->to($recipient->email, $recipient->first_nm.' '.$recipient->last_nm)->subject($recipient->first_nm.' Welcome to CarsYours.com!');
        });
    }


    public static function mail_forgot_password($uid){
        //    Generate new random password, hash it, store it, date it, email it
        //    with a turnaround link
        $new_gc = 'cy'.substr(md5(rand()), 0, 8);
        $recipient = Account::find($uid);
        $recipient->geocode_lnk_tmp = Hash::make($new_gc);
        $recipient->geocode_lnk_tmp_at = date("Y-m-d H:i:s");
        $recipient->save();
        //
        Mail::send('emails.account.forgot_password',
            array('first_nm' => $recipient->first_nm,
                'last_nm' => $recipient->last_nm,
                'geocode' => $new_gc,
                'url' => URL::Route('get_forgot_password_sid',$recipient->sid)),
            function($message) use ($recipient) {
        $message->to($recipient->email, $recipient->first_nm.' '.$recipient->last_nm)->subject($recipient->first_nm.' CarsYours.com Password Reset');
        });
    }


    public static function forget_password_via_sid($sid){
        //    Determine if the turnaround is over 1 hour late (false)
        //    If not (true) swap geocode with temp geocode
        $accnt = Account::where('sid', '=', $sid)->first();
        if($accnt) {
            $to_time = strtotime(date("Y-m-d H:i:s"));
            $from_time = strtotime($accnt->geocode_lnk_tmp_at);
            if (59 > (round(abs($to_time - $from_time) / 60,2))){
                $accnt->geocode = $accnt->geocode_lnk_tmp;
                $accnt->geocode_lnk_tmp = null;
                $accnt->geocode_lnk_tmp_at = null;
                $accnt->save();
                return true;
            }
        }else{
            Return false;
        }
    }


    public static function update_accnt_email($uid, $new_email){
        //    Update email given uid

        $update_accnt_email = Account::find($uid);
        $update_accnt_email->email = $new_email;
        $update_accnt_email->save();
    }


    public static function update_accnt_geocode($uid, $new_geocode, $passw_hint){
        //    After post forgot password turnaround email

        $update_accnt_geocode = Account::find($uid);
        $update_accnt_geocode->geocode = Hash::make($new_geocode);
        $update_accnt_geocode->passw_hint = $passw_hint;
        $update_accnt_geocode->save();
    }


    public static function update_accnt_state_stage($uid, $state_stage) {
        //    UPDATE the Account > state_stage column and INSERT new Activity
        Eloquent::unguard();
        DB::beginTransaction();
            $update_accnt_state_stage = Account::find($uid);
```

```
            $update_accnt_state_stage->state_stage = $state_stage;
            $update_accnt_state_stage->save();
            //    INSERT actv
            $insert_accnt_actv = Account_activity::create(array(
                'parent_id' => $update_accnt_state_stage->id,
                'state_stage' => $update_accnt_state_stage->state_stage
                ));
        DB::commit();
    }


    public static function increment_sign_in_attempts($accnt_id) {
        //    Decrement the attempts counter
        //    Enforce the lockout after six rule
        $update_accnt_sign_in_audit = Account::find($accnt_id);
        $update_accnt_sign_in_audit->sign_in_attempts = $update_accnt_sign_in_audit->sign_in_attempts + 1;
        if($update_accnt_sign_in_audit->sign_in_attempts == 6){
            $update_accnt_sign_in_audit->locked_at = date("Y-m-d H:i:s");
        }
        $update_accnt_sign_in_audit->save();
    }

    public static function accnt_locked_at($accnt_id, $locked_at) {
        //    If 30 min have elapsed reset locked_at

        $to_time = strtotime(date("Y-m-d H:i:s"));
        $from_time = strtotime($locked_at);
        if (30 < (round(abs($to_time - $from_time) / 60,2))){
            $update_accnt_locked_at = Account::find($accnt_id);
            $update_accnt_locked_at->locked_at = null;
            $update_accnt_locked_at->save();
            return false;
        }else{
            return true;
        }
    }

    public static function get_lkup($column_nm, $lookup_code) {
        //    Get the descriptive value from the lookup given
        //    the column and code

        $lkup = Account_lkup::where('column_nm', '=', $column_nm)->where('lookup_code', '=', $lookup_code)->first();
        if($lkup) {
            return $lkup->lookup_desc;
        }else{
            Return $lookup_code.' not found';
        }
    }

    public static function getMobileVerify($phone_mobile) {
        //    SMS a 4 digit verification turnaround
        //
//TODO Create notification
        if ($phone_mobile) {
            $turnaround = rand(1000, 9999);
            $SMSclient = new Services_Twilio("AC245f9ebeo87hv7rh8ta9c26b802cf3b", "ac6c457cedi8v7husydrvtrrybda87cc");

            $SMSclient->account->messages->create(array(
                'To' => $phone_mobile,
                'From' => Self::SMS_PHONE_NUMBER,
                'Body' => "From CarsYours.com: " . $turnaround,
            ));

            return $turnaround;
        }
    }


    public static function getFullName($uid = 1) {
        //    Return the full name of a member given an id

        $accnt = Self::where('id', '=', $uid)->where('state_status', '=', Self::STATE_STATUS_ENABLED)->first();
        $fullname  = $accnt->first_nm;
        if ($accnt->middle_nm) {
            $fullname .= ' ' . $accnt->middle_nm[0] . '.';
        }
        $fullname .= ' ' . $accnt->last_nm;

        return $fullname;
    }


    public static function post_sign_in_session() {
        //    Role based app config stored in Session
```

```

        $app_role = "MEMBER";
        switch (Auth::user()->state_type) {
            case Account::STATE_TYPE_BUYER:
                $app_role = "MEMBER";
                break;
            case Account::STATE_TYPE_SELLER:
                $app_role = "MEMBER";
                break;
            case Account::STATE_TYPE_BUYER_SELLER:
                $app_role = "MEMBER";
                break;
            case Account::STATE_TYPE_LENDER:
                $app_role = "LENDER";
                break;
            case Account::STATE_TYPE_AUTO_TECH:
                $app_role = "AUTO_TECH";
                break;
            case Account::STATE_TYPE_SERVICE_REP:
                $app_role = "ADMIN";
                break;
            case Account::STATE_TYPE_ADMIN:
                $app_role = "ADMIN";
                break;
        }
        Session::push('app_config', ['app_role' => $app_role]);
    }

    public static function spice($in, $key){
        $oIntelNum = new Helpers\IntelNum();
        return $oIntelNum->v_fmbc($in, 61020);
    }

    public static function unspice($in, $key){
        return 12029;
    }

}
```



New Membership Account

[✓ Buying] Selling

First Name

Last Name

Email

We will send an email to this address to activate account

Email Verify

Password

****

A

Password Verify

Password Hint

Remind you of your password when you type it wrong twice

Zip / Postal Code

00000

Question

- none --

You will be asked if you forget your password

Answer

I have read the Site Terms

Create Account

We Respect Your Privacy | Help | Acceptable Use Policy

Sign In

Email

Password

Sign In

OR

Sign In with GMail

Forgot Password? | Sign Up Now

We Respect Your Privacy | Help | Acceptable Use Policy

Change Email

An Email Message has been sent to your new Address.
Please check your email.

Email ?

Email Verify ✓

This email address must not already exist.

Save Email