

# CPS 630

## Plan for Smart Services (P2S)

Iteration 3 and 4

Group # 4 – Section 01

Jonathan Lam - 500792483

Payton Lawson - 500899870

Thomas Young-Audet – 500865018

Name	Participation (%)
Jonathan Lam	33.33
Payton Lawson	33.33
Thomas Young-Audet	33.33

# Security

The security issues we were primarily concerned with was with regards to the Integrity Security Principles. This includes data such as a user's login password and their payment information. For the former, we chose to apply a MD5 hash as well as salting the hash on the user's password. As MD5 does not allow for decryption this meant that it would be a safe method to verify the user's password against the hashed value stored in our database aka. verifying the integrity of the user's password and more importantly to prevent the decryption if the data were to be leaked. For the latter, since we need to retrieve payment information such as a credit card number, we thought it would be more suitable to encode the data using base64 encryption (text to binary).

Furthermore, our Maintain pages are set to only allow users with an Admin role to access. This means that no ordinary user can view sensitive information nor modify it.

# Ride Green

The Ride Green service allows users to select at most two of the same service into their shopping cart, compare against the two and finally select one to checkout.

For example if the user selects the service 'Ride to Your Destination', the user can select up to two vehicles and compare:

### Ride to Your Destination

Ride 1

5415 Landsborough Ave, I

5415 Landsborough Ave, I

2021-04-08

04:51 AM


Ride 2

5415 Landsborough Ave, I


5415 Landsborough Ave, I

2021-04-08

04:51 AM




2004 Toyota Sienna  
\$50




2017 Dodge Ram  
\$70

#### Your Order



2018 Fiat 500  
\$40



2020 Nissan Sentra  
\$40


Compare

Clicking on the 'Compare' button will lead the user to the next page which will allow them to view their chosen vehicles, the set parameters and any existing reviews that exist for the respective vehicles.

P25 Home About Contact Us Reviews Suggestions Browser Types of Services \* Welcome Jonathan

### Compare your choices


#### Item 1



Product name: 2018 Fiat 500  
Price: \$40  
FROM:  
5415 Landsborough Ave, Mississauga, ON L5R 3K2, Canada  
TO:  
5415 Landsborough Ave, Mississauga, ON L5R 3K2, Canada  
DATETIME:  
2021-04-08 04:51

Reviews:  
Select this item

#### Item 2





Product name: 2020 Nissan Sentra  
Price: \$40  
FROM:  
5415 Landsborough Ave, Mississauga, ON L5R 3K2, Canada  
TO:  
5415 Landsborough Ave, Mississauga, ON L5R 3K2, Canada  
DATETIME:  
2021-04-08 04:51

Reviews:  
Select this item

Finally, once the user selects which vehicle they would like to request, they are directed to the 'Order Summary' page to complete the transaction.

### Order Summary



2020 Nissan Sentra  
\$40

**FROM:**  
5415 Landsborough Ave, Mississauga, ON L5R 3X2, Canada

**TO:**  
5415 Landsborough Ave, Mississauga, ON L5R 3X2, Canada

**DATETIME:**  
2021-04-08 04:51

#### Payment Information

First Name	Last Name
<input type="text"/>	<input type="text"/>
Email	
<input type="text"/>	
Card Number	
<input type="text"/>	
Expiration Date (MM/YYYY)	CVC/CVV
<input type="text"/>	<input type="text"/>

Place Order

# Cross Browser Support

At this moment, the browsers verified to be supported include: Google Chrome, Firefox, Microsoft Edge and Safari. Testing was performed manually and in two phases: visual tests and functionality end-end tests. For visual tests, each page was viewed on each browser marking any differences and making changes to accommodate such differences. With regards to end-end testing, on each browser, we tested the functionality of our service in the following steps:

1. Register a new account
2. Login to the account
3. Select a service
4. Enter fields and ensure required fields are filled and field requirements are met before moving onto the next step
  - a. Testing of the drag and dropping of items (max. 2 if Ride Green otherwise 1)
5. Checkout and place order
6. Using our Maintain service page, ensure the order was placed and entered correctly into the database.

# New Service

Our new service is titled *Suggestions*. As a provider of goods and services, our primary goal is to serve and meet the needs of customers. Oftentimes the interests of these customers aren't met in such cases where certain vendors are not available through us. As a result, we have created a simple service page that allows clients to suggest their favourite or commonly used vendors to our platform. This will also allow us to gauge their interests of the clients and further improve and tailor our services to them. An example of the page is shown below:

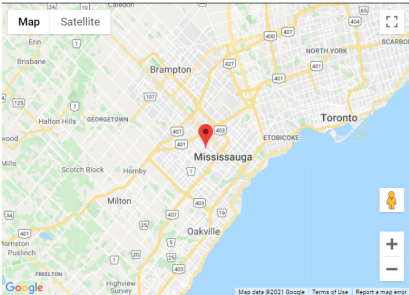
P2S

[Home](#) [About](#) [Contact Us](#) [Reviews](#) [Suggestions](#) [Browser](#) [Types of Services](#)

Welcome Jonathan

## Suggest a Place!

Found a vendor not on our list? Fret not! Suggest it to us and we will get the ball rolling!



Business Name:

Address:

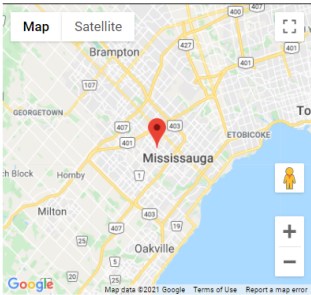
Enter a location

CPS 630 - Plan for Smart Services

This page allows the user to enter a company name as well as the address which will be sent to our suggestions table in the database for further review. This page also implements Google Maps JavaScript API as well as Places API which further lets the user to view their suggestions on the map.

## Suggest a Place!

Found a vendor not on our list? Fret not! Suggest it to us and we will get the ball rolling!



Business Name:

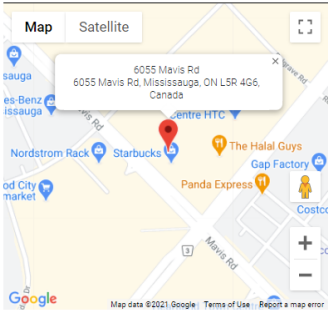
Address:

6055 Mavis Road, Mississauga, ON L5R 4G6, Canada

powered by Google

## Suggest a Place!

Found a vendor not on our list? Fret not! Suggest it to us and we will get the ball rolling!



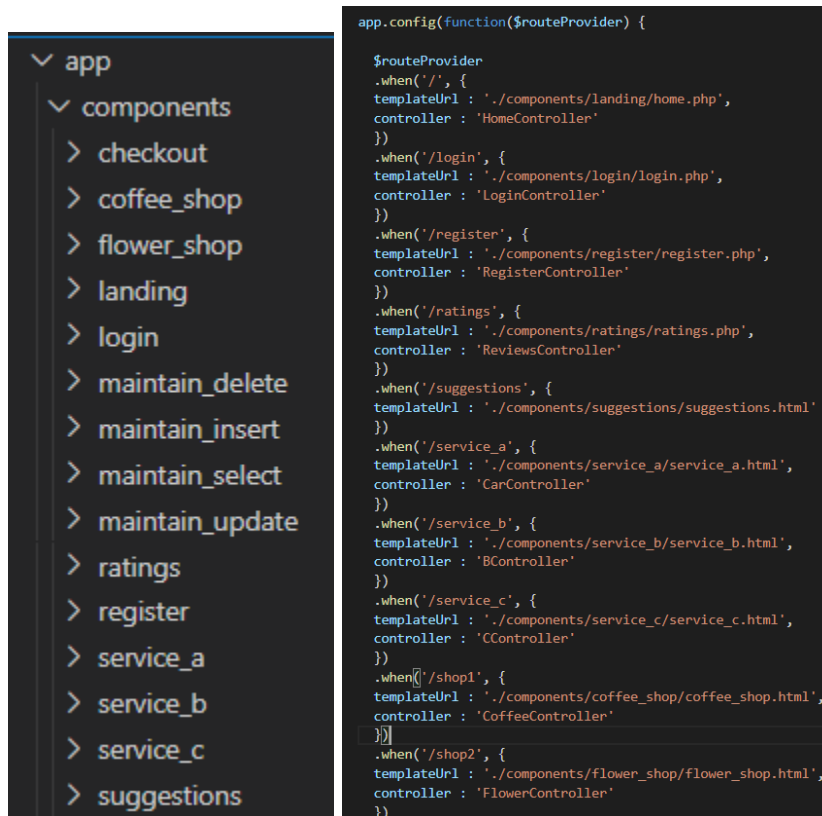
Business Name:

Address:

Submit

# SPA Design

The Single Page Application consists of 16 separate pages all routed by a single application using AngularJS. An example snippet and folder structure is shown below:



The main application has a primary controller and using the \$rootScope, we can control certain components from being shown on the screen depending on the current and next routes. For example:

```
app.run(function($rootScope) {  
    $rootScope.$on("$locationChangeStart", function(event, next, current) {  
        if (next.endsWith('#!/') || next.endsWith('/#about') || next.endsWith('/#contact') || next.endsWith('/#top')) {  
            $('#order_query').addClass("d-flex")  
            $('#order_query').show();  
        } else {  
            $('#order_query').removeClass("d-flex")  
            $('#order_query').hide();  
        }  
    });  
});
```

Other controllers are used to make HTTP Requests with our PHP server such as retrieving a list of supported vendors:

```
app.controller('CController', function($scope, $http) {
    $scope.message = 'Hello from C';
    $http.get('../server/get_shops.php')
        .then( function (response) {$scope.shops = response.data.records;})
    });
```

Using a controller, we can also retrieve/organize information embedded in the HTML such as form information and further POST it to the PHP server such as when creating new orders:

```
app.controller('CarController', function($scope, $http, $timeout, cartService, checkoutInfoService) {
    $scope.message = 'Hello from cars';
    // default post header
    $http.defaults.headers.post['Content-Type'] = 'application/x-www-form-urlencoded;charset=utf-8';
    //get cars
    $http.post('../server/get_cars.php')
        .then( function (response) {$scope.cars = response.data.records;})

    $timeout(function() {
        setDraggable();
    }, 1000); // 1 seconds
    $scope.submitCart = function() {
        $scope.checkoutInfo = {
            'start_address' : document.getElementById('start_address').value,
            'end_address' : document.getElementById('end_address').value,
            'date' : document.getElementById('date').value,
            'time' : document.getElementById('time').value,
        }
        $scope.productInfo = {
            'product_info' : document.getElementById("cart").childNodes[3].children[1].children[0].innerHTML,
            'product_price' : document.getElementById("cart").childNodes[3].children[1].children[1].innerHTML,
            'product_img' : document.getElementById("cart").childNodes[3].children[0].src,
            'product_id' : document.getElementById("cart").childNodes[3].id,
            'category' : "Car"
        }
        cartService.clearProducts();
        checkoutInfoService.clearInfo();
        cartService.addProduct($scope.productInfo);
        checkoutInfoService.addcheckoutInfo($scope.checkoutInfo);
        window.location.href = "#!checkout"
    };
});
```



Finally, with AngularJS we can create Services like the Factory Recipes to share code and information across views within our apps. An example is as follows:

```
app.factory('checkoutInfoService', function() {
  var checkoutInfo = [];

  var addcheckoutInfo = function(newObj) {
    checkoutInfo.push(newObj);
  };

  var getcheckoutInfo = function(){
    return checkoutInfo;
  };
  var getNumOfCheckoutInfo = function(){
    return checkoutInfo.length;
  };
  var clearInfo = function() {
    checkoutInfo = [];
  }
  var clearFirst = function() {
    checkoutInfo.shift();
  }
  var clearSecond = function() {
    checkoutInfo.pop();
  }
  return {
    addcheckoutInfo: addcheckoutInfo,
    getNumOfCheckoutInfo: getNumOfCheckoutInfo,
    clearInfo: clearInfo,
    clearFirst: clearFirst,
    clearSecond: clearSecond,
    getcheckoutInfo: getcheckoutInfo
  };
});
```

With regards to MVC integration using AngularJS, the Views are routed by the main application and the controllers associated with each route is used to handle user input, retrieve information and make GET/POST requests to the database. For example, when a page is loaded, the controller retrieves necessary information from the database and uses the \$scope object to update the View. A simple example would be as follows:

```
//get products
$http.post('../server/get_products.php', 'category=flowershop').then( function (response) {
  $scope.products = response.data.records;
});
```