Course Name: Web Applications
Course Code: CPS630
Project Title: Plan for Smart Services (P2S)
Team#: Section 2, #4

Team Members

Name: Joonho Myung
Email: joonho.myung@ryerson.ca
Student Id: 500845049

Name: Allen You
Email: allen.you@ryerson.ca
Student Id: 500833035

Name: Austin Cheung
Email: ahcheung@ryerson.ca
Student Id: 500810590

Table of Contents

Work Distribution :

Work Percentages

Member	Iteration 1&2	Iteration 3&4	%if you want
Joonho Myung	Creating table databases Create logo Managed ride&deliver page	Browser detector SPA design architecture Managed Reviews page	33
Allen You	Created initial set up for each page of html Created fake data values for user to select Php and javascript stuff	Sign up and Sign in page. User database . User Interface quality of life improvement on home page. Admin page and admin User Interface Check out page with invoice database.	33
Austin Cheung	Handled database values Implemented map features Implemented service page UI	Added some css elements to the home page Updated the two previous services Implemented the ridegreen services Implemented the comparison of the users choices Implemented user reviews on the compare pages	34

Security Issues :

Password Security :

Password protection and card protection were security issues present in the program. To fix these issues we encrypted the password with md5. Since we cannot reverse md5 encryption when changing the password we cannot check what the user set the password to, only change the password.

+ Options

	userid	fname	lname	email	pass	pnum	addy
Edit Copy Delete	1	admin	admin	admin	21232f297a57a5a743894a0e4a801fc3	admin	admin
Edit Copy Delete	2	Jane	Doe	janedoe@gmail.com	202cb962ac59075b964b07152d234b70	123456789	i live somewhere
Edit Copy Delete	3			admin	21232f297a57a5a743894a0e4a801fc3		

Payment Security :

The initial plan for a secure payment page is that the user has to re-enter their payment information every time so that the linked account (if breached) would not be able to make additional purchases since we did not let the user have a saved card for checkout. Another implementation is that we can drop unnecessary card information saved in the database after the payment has gone through such that if our database was compromised then there would be partial information on each card. Such as the invoice information, last 4 digits of the card for confirmation.

2	Jane	Doe	48.59
4	John	Doe	265.55
2	Jane	Doe	56.50

2	Jane Doe	123123123231132	January	2121
2	Jane Doe	123123123123	123	123
2	Jane Doe	123123123	213123123	213123213

Once a confirmation of the order is implemented the values on the right are dropped, and then the last 4 digits of the card number would be added to the invoice on the left.

Shopping Cart :

Each order has their own database and orders are inputted into their corresponding database. When an order from the original two services is added to the database; however, when an order is created using service c a separate table is created. A comparison table is created and data is pulled from these tables when the user chooses which service they want. The data is pulled from the comparison table and inputted into the corresponding order table. The comparison tables are then dropped. All orders are stored in the order tables in the database, but only orders corresponding to the user's email will be displayed in the shopping cart. This is to make sure that no other orders besides the ones made by the users will appear in the shopping cart.

Web Browser Support :

- We tried on 3 different browsers, Chrome, Microsoft Edge and Firefox.

New Service : Rent A Car

The new service we added (Service D) wasRent a Car. It allows users to rent a car from our existing catalogue. This service is very similar to Ride to Destination, where the user chooses what vehicle they want, how long they want said vehicle and when they need it. This was a very basic service to implement since most of the code was already created in Ride to Destination, only the variables were changed in order to better suit the service.

SPA Design :

- Most web pages are connected with the SPA. We get a separate page when the user fails to login correctly. When a user fails to login, they are transferred to sign in page and that page does not have a head bar which customers can navigate to other options.

- However, when customers login successfully, they are transferred back to the main page.

- Within our testing.php, we have navigation bar on the top and view elements of other files under that navigation bar using <div ng-view>

- Our group used AngularJS using a separate app.js file to implement SPA.
    - Using $routeProvider and .when, we have provided routing service

```
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
    .when("/", {
        templateUrl : "main.php"
    })
    .when("/database", {
        templateUrl : "cart.php"
    })
    .when("/signup", {
        templateUrl : "signup.php"
    })
```

- SPA architecture work based on the Model-View-Controller system.
    - For Model, it's a file that i'm going to run with SPA
    - These files are controlled from the main page using AngularJS app.js. This file has routing options to all the webpages that we are going to connect.
    - These files are selected and finally viewed in the main page using the ng-view function.