CPS 630: Web Applications
Plan for Smart Services (P2S)

TEAM 16

Aitirja Chowdhury (500832196)
Felicia Levina (500856106)
Mehnaaz Rahman (500835150)

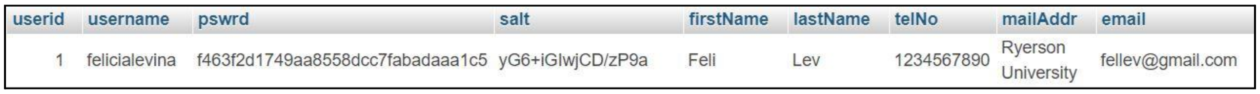
Percentage of tasks completed by:

Aitirja Chowdhury: 33%	Felicia Levina: 33%	Mehnaaz Rahman: 33%
• Previous iteration: ○ Home ○ Service B - drag and drop; front-end & back-end ○ DB maintain - front-end and view functionality • “Maintain” - uploading the page with the database retrieved from phpMyAdmin • “Review” - uploading the reviews for every service used (per item/trip) • “Compare & Select” - implementing the rating portion for ideal selection. • Sign in - front-end updates (admin / user dropdown) • “SPA” - attempted view database SPA • Technical Report • ReadMe.txt	• Previous iteration: ○ Sign in ○ Sign up back-end ○ Geolocation ○ Service A - front end & back end ○ About us, Contact us ○ Database design • “Maintain” - update functionality • “Service C” - completed back-end functionality to remove item from cart • “Service D” - implemented the new cleaning service • Separate admin login back-end • Security - user and admin password hashing • Browser Information • “SPA” - attempted main, sign in and sign up front-end page	• Previous iteration: ○ Review ○ Sign up front-end ○ Home, About us ○ Technical Report • “Maintain” - insert, delete functionalities • “Service C” - front-end and attempted back-end functionality to remove items from cart • “Compare & Select” - implementation of the comparison table • “SPA” - attempted connection of main navigation links to display information • Modifications of database tables • Technical Report

Table of Contents

Detection of Security Issues

The possible security issues detected in our program can be attributed to user passwords. User passwords were found to present a potential security risk to any user account as there was no real form of authentication. Hence to tackle this issue, all passwords were saved in the database after being encoded by the “Salting the Hash” and “Secure Hash” MD5() function. Essentially, a random “salt” was linked to each password after which the MD5() function was applied to it. When authenticating the username and password, the salt is then obtained from the database using the user id entered. The salt and the submitted password then create a salted password which is run through a one-way hash. The generated MD5() hash value and the user id are then compared with the values saved in the database. If the values compared are the same, then the login is considered to be successful. However, if they are different, an error is generated.

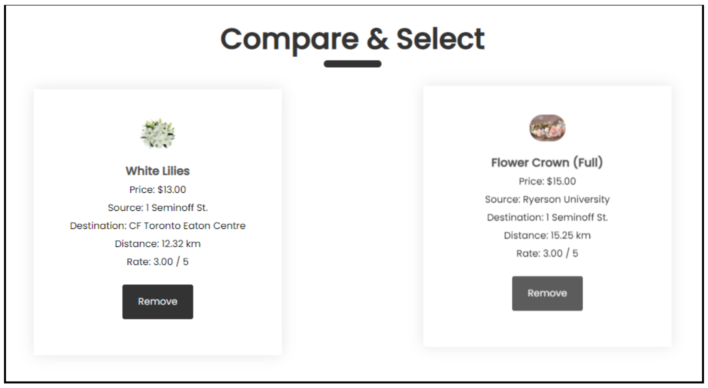


userid	username	pswrd	salt	firstName	lastName	telNo	mailAddr	email
1	felicialevina	f463f2d1749aa8558dcc7fabadaaa1c5	yG6+iGlwjCD/zP9a	Feli	Lev	1234567890	Ryerson University	fellev@gmail.com

The Shopping Cart & Service C

A user can only view the items in their shopping cart once they have signed in. Hence, the user id is bound to the shopping cart through the session that is started once the user has logged in. If the user wishes to book a trip using Service A or Service C, then the trip id that is generated and assigned for the given user id is also bound to the shopping cart. If the user wishes to deliver either a flower or coffee item using Service B or Service C, then the product id associated with the user id is also bound to the shopping cart. When the user selects the checkout button, all of these ids are retrieved to create a unique order placement distinguished by an order id. Hence in this way, the Shopping Cart works to cover the orders’ parameters.

When selecting Service C, our program allows you to choose either two Service A or two Service B. Regardless of the choice, the two Service A / two Service B are both displayed in a comparison result table as seen below:



Compare & Select

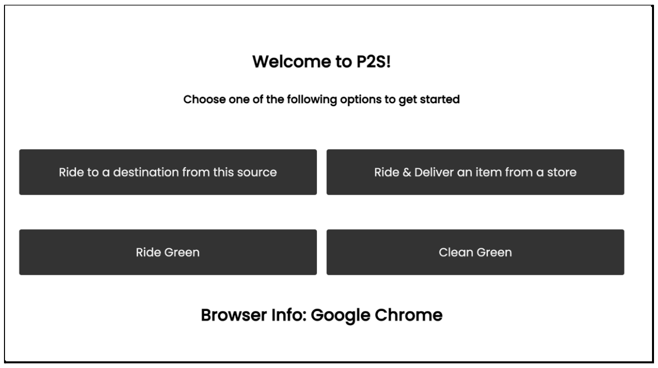
White Lilies
Price: $13.00
Source: 1 Seminoff St.
Destination: CF Toronto Eaton Centre
Distance: 12.32 km
Rate: 3.00 / 5

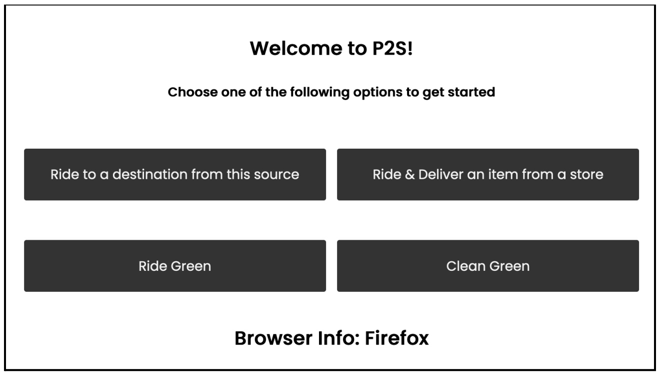
Flower Crown (Full)
Price: $15.00
Source: Ryerson University
Destination: 1 Seminoff St.
Distance: 15.25 km
Rate: 3.00 / 5

As one can see, both services display values for the same parameters. Below each service, there is also a Remove button. The user must click on the Remove button for one of the services to delete that service and be directed to the shopping cart which displays the service that was not removed. Thus by implementing it this way, we displayed two boxes with all the parameters for the comparison and only needed one shopping cart.

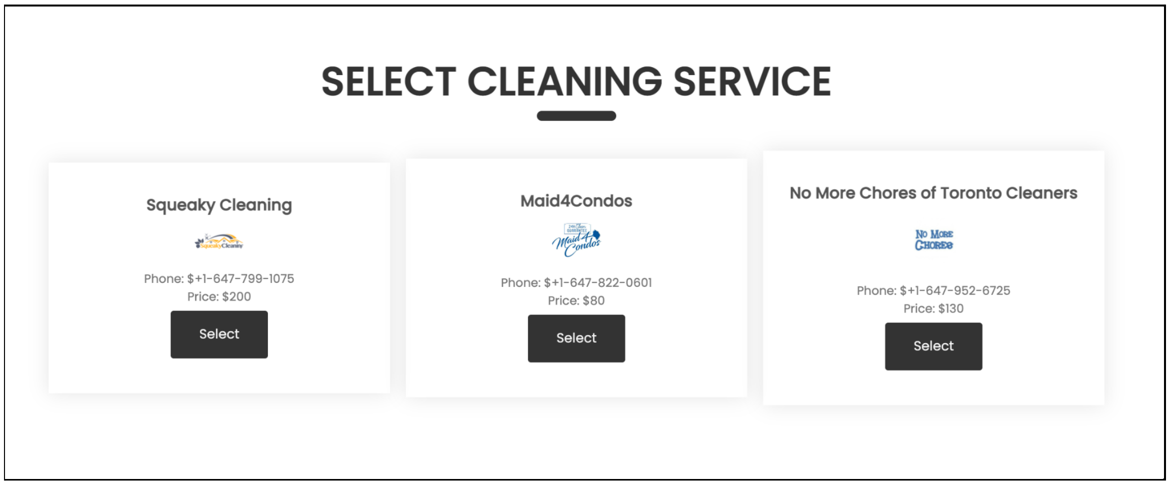
Cross-Browser Compatibility

Our program supports Chrome, Firefox, and Internet Explorer. We coded our program using simple if statements so that the layout is the same among all three browsers when accessed. Additionally, for these three browsers, the browser information corresponding to the particular browser used can also be seen on the home page.


Welcome to P2S!
Choose one of the following options to get started
Ride to a destination from this source
Ride & Deliver an item from a store
Ride Green
Clean Green
Browser Info: Google Chrome


Welcome to P2S!
Choose one of the following options to get started
Ride to a destination from this source
Ride & Deliver an item from a store
Ride Green
Clean Green
Browser Info: Firefox

Description of Service D



For our Service D, we chose to add a cleaning service as a part of our program. Given people's busy and stressful lifestyles, we decided that we would offer cleaning services at affordable prices for our new service. Called Clean Green, the idea behind this is to offer cleaning services where only eco-friendly products are used. Placing an importance on hygiene and sustainability, we partner with cleaning companies that use durable products with environmentally friendly chemicals. That said, if the user selects the Clean Green service, they can choose the cleaning service they would like based on the company and price. Following this, the user can enter values to specify their address as well as the time they would like to book their appointment. The order-to-be will then be placed in the shopping cart from where the user can checkout and place the order.

Incorporation of SPA Design

Our program consists of 46 pages (plus the style.css) which include the main functionalities of our service page. However, when designing these pages, to create a responsive website, we decided to include more pages to further divide the backend and front-end portion for the website. Using AngularJS, we chose to implement the SPA design in our program.

We created a new php which is the main page where all the SPA will be taking place. This is the page that contains the angular directive "ng-view". Since the top navigation bar is the source of getting to one page to another, the angular directive was placed right beneath the navigation bar. From here on out this will show all the pages that exist in our website.

For each of the pages, we set up the Controllers to separate the view and model part of the project. The view and model are not directly related and if there are changes that have to be made in the model (data), it will have to go through the controller that is connected to the page.