CAR-TOON'S CARWASH

PROTOTYPE DESIGN AND IMPLEMENTATION REPORT

BUS 465

Term: Fall 2020

Instructor: Amin Milani Fard

Date: December 8th, 2020

D100-Team 1:

Alex MacLeod (301313902)
Jenny Ren (301278797)
Yanal Butt (301229424)

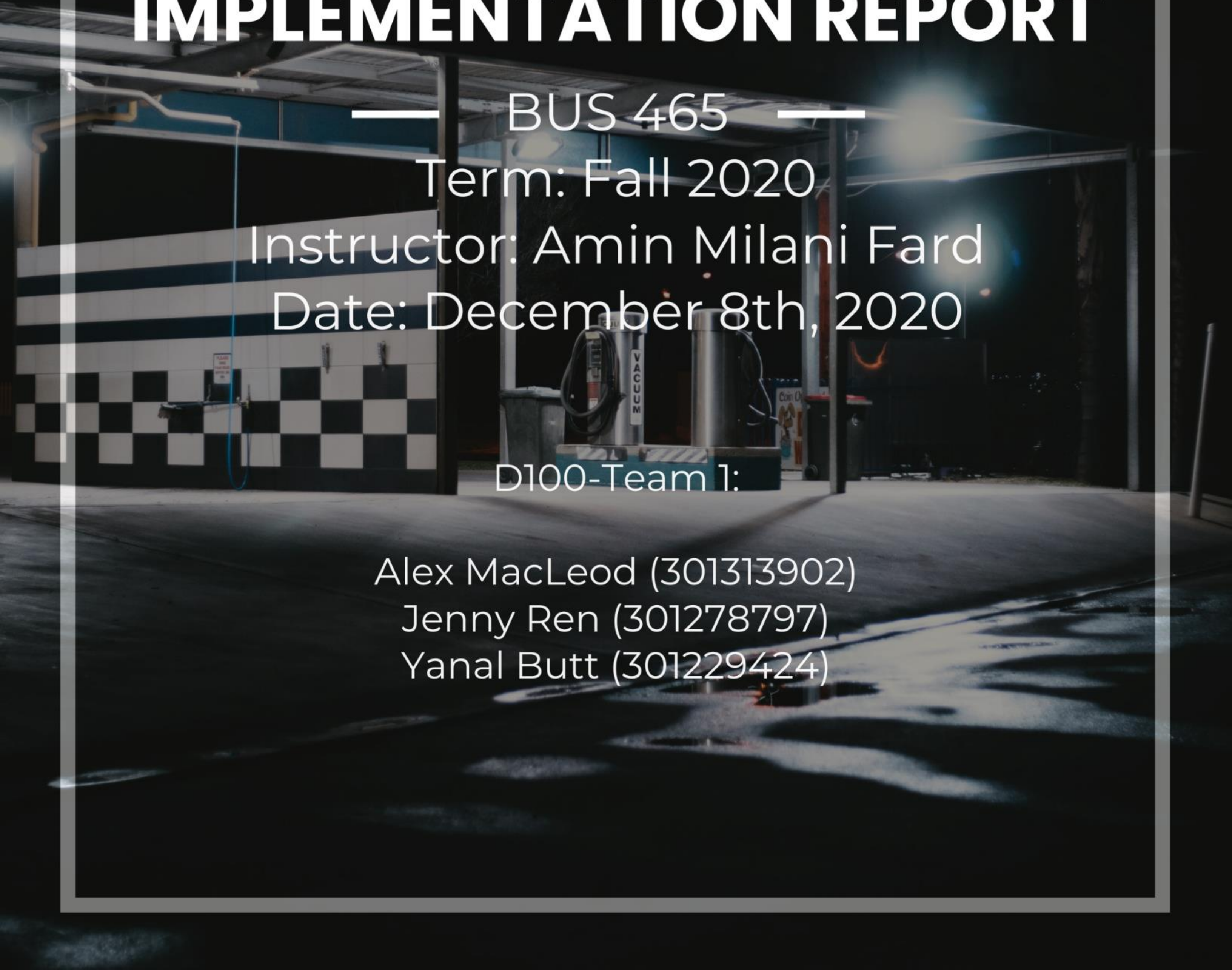


TABLE OF CONTENTS

EXECUTIVE SUMMARY

Car-Toon's Carwash is a small business based in Langley, BC that sells washes and details. Their current operations are non-computerized, with sales tracked in aggregate, payment only available through in-person methods, no customer data collected. This current system is inefficient and fails to capture and utilize data about the company's customers. This report outlines the design and implementation of a prototype information system that will allow customers to create accounts and make purchases online.

This prototype will be able to solve several issues for the business. The owner expressed that while he knows many customers personally, he has no way of tracking trends in the customer base as a whole to inform his decisions. Payment being in-person also causes frequent issues, with long lines forming to use a payment machine. Details are also difficult to book as there is no centralized system for tracking bookings and requests are taken in the form of phone messages.

By allowing customers to create accounts and make purchases online, the company will improve the customer experience by reducing wait times. Analyzing the data collected will result in the company understanding customer trends and purchasing preferences. Details will be far more organized, with booked dates stored in a database that can easily be referred to.

Seven use cases were created, which illustrate how the system will interact with customers as well as the system administrator. From these use cases, two levels of data flow diagrams were modelled. These clearly show how the new system creates and stores key data about customers and purchases. Finally, an entity relationship diagram was created showing how the data is organized for use by the system.

A prototype system was created which is capable of executing the most critical features. The prototype was built using a variety of languages, including HTML, PHP, and Javascript. It allows customers to create accounts, login, buy washes, and book details. It also allows the admin of the site to view several reports about site metrics. The prototype created is the first step towards a public facing system, and there are many future additions prescribed, including upgrades to security and the user experience.

COMPANY BACKGROUND

Car-Toon's Carwash is a company based in Langley, BC that offers hand and automatic wash services as well as details. The company is operated by the owner, Don MacLeod, and a small staff of attendants who perform all functions. Currently their operations are mostly non-computerized. Customers line up and purchase washes and add-ons in person. No customer data are collected, and purchases are not tracked to specific customers.

The current system is inefficient and fails to capitalize on several opportunities. Firstly, because customers can only pay in person, there are often long lines as people wait their turn to use the single payment machine or exchange cash. These transactions are recorded on an aggregate basis but are not linked to customers so tracking of customer preferences is not possible. The company has a website with fairly steady traffic, but because there is no option to make sales directly on the website there is a missed opportunity to capitalize on web traffic. There is also no centralized datastore for detail bookings, so it is up to staff to ensure no bookings are lost or dates are double booked.

The owner has expressed interest in modernizing several of these processes. Allowing online sales would decrease line ups and wait times and convert some of their existing web traffic to sales. Tracking customer purchases would allow greater insight into customer trends and preferences, as well as a smoother experience for customers. Handling details with a central database would avoid double bookings and reduce the burden on the detail manager to track bookings manually.

PROJECT SCOPE

Given our team’s level of experience and the limited time frame for the project, our focus has been on creating a minimum viable product to meet the core needs of the sponsor.

Key features we deemed necessary for this first prototype include:

- Customers being able to create accounts through the site
- Customers being able to log in
- Customers having their purchases and bookings logged to their account
- Customers being able to view their purchase history
- Customers being able to place a wash order online
- Customers being able to book a detail online
- Checks in place to ensure details are not double booked
- Admin being able to view reports

We have made the following assumptions when implementing these features:

- The system will only be accessible to those with accounts. In order to view the washes and details and make purchases, customers must create an account and log in.
- Payment will be implemented at a later date. Our current version has a placeholder payment page to allow users to execute all the steps to buy a wash and have it reflected in the database, but no transactions are actually processed. Use cases reference payment information entering the system, but currently there is no data model in place for storing or using this data.
- To change a detail booking after it is made, customers must contact the business via phone or email and the administrator will change it in the database themselves.

USE CASES

Use Case Name:	Create Customer Account	**ID:**	UC-1	**Priority:**	High
Actor:	User				
Description:	Process for new customers creating an account in order to use the system				
Trigger:	Customer enters system for first time and chooses create account option				
Type:	External				
Preconditions:	1. No existing accounts in the system with the same email address				

Major Steps:	**Information for Steps:**
Normal Course:	
1. User enters personal details to create new account	← First Name ← Last Name ← Email ← Phone Number ← Date of Birth ← New Password ← Confirm Password
2. System verifies details and confirms account creation	→ Account Confirmation
Alternative Course: **1.1 Required Field Blank or Incorrect (Occurs at step 1)**	
1. System notifies user of blank or incorrect field	→ Incorrect Fields Notification
2. Customer fills the field correctly	← Corrected Fields

3. Return to step 2	

Postconditions:	1. User account and details are stored in the user datastore 2. User is taken to login
Exceptions:	1. If there is already an account with the email address given in the system, the user is notified and asked to try again

Summary:			
Inputs	**Source**	**Outputs**	**Destination**
First name Last name Email Phone number Data of birth New password Confirm password Corrected fields	User	Account confirmation Incorrect fields notification	User

Use Case Name:	Login as Customer	**ID:**	UC-2	**Priority:**	High
Actor:	User				
Description:	Process for existing customers to login to their account to access the system				
Trigger:	User enters system and chooses login option				

Type:	External		
Preconditions:	1. User has created an account which is stored in the users datastore		
Major Steps:		**Information for Steps:**	
Normal Course:			
1. User enters email and password for their account		← User Email ← User Password	
2. System verifies credentials and confirms login		→ Login Confirmation	
3. System displays user's profile		→ Order History	
Alternative Courses:			
Postconditions:	1. User is given access to system with information specific to their account		
Exceptions:	1. If email and password given do not match any in the system, user will be alerted and asked to try again		
Summary:			
Inputs	**Source**	**Outputs**	**Destination**
User Email User Password	User	Login Confirmation Order History	User

Use Case Name:	Buy a Wash	**ID:**	UC-3	**Priority:**	High
Actor:	User				
Description:	Process for a user to purchase a wash online using the system				
Trigger:	User chooses the option to purchase a wash in the system				
Type:	External				
Preconditions:	1. User is logged in to the system				
Major Steps:		**Information for Steps:**			
Normal Course:					
1. System displays wash options		→ Wash Options			
2. User selects a wash		← Wash Selection			
3. System displays payment form		→ Payment Form			
4. User enters billing address and payment details		← Street Address ← Apt Number ← City ← Postal Code ← Province ← Name on Card			

	← Card Number ← Security Code ← Expiry Date
5. System displays wash receipt	→ Wash Receipt
Alternative Courses:	
Postconditions:	1. Wash purchase is logged in the ordered services datastore 2. Email of receipt with wash code is sent to user’s email address on file 3. Payment details are sent to third party processor to process transaction (outside current scope)
Exceptions:	
Summary:	

Inputs	**Source**	**Outputs**	**Destination**
Wash Selection Street Address Apt Number City Postal Code Province Name on Card Card Number Security Code Expiry Date	User	Wash Options Payment Form Wash Receipt	User

Use Case Name:	Book a Detail	**ID:**	UC-4	**Priority:**	High
Actor:	User				
Description:	Process for a user to purchase a wash online using the system				
Trigger:	User chooses the option to purchase a wash in the system				
Type:	External				
Preconditions:	1. User is logged in to the system				

Major Steps:	**Information for Steps:**
Normal Course:	
1. System displays detail options	→ Detail Options
2. User selects a detail	← Detail Selection
3. System displays booking calendar	→ Booking Calendar
4. User selects booking date	← Date Selection
5. System displays detail receipt	→ Detail Receipt
Alternative Courses: **1.1 Date Already Booked** **(Occurs at step 4)**	

1. System alerts user that selected date is booked and asks them to try another	→ Booked Date Alert
2. User enters valid date	← New Date Selection
3. Return to step 5	

Postconditions:	1. Detail booking is logged in the ordered services datastore 2. Email of receipt sent to user's email address on file 3. Email is sent to detail manager notifying them of booking so they can contact the customer
Exceptions:	
Summary:	

Inputs	**Source**	**Outputs**	**Destination**
Detail Selection Date Selection New Date Selection	User	Detail Options Booking Calendar Detail Receipt Booked Date Alert	User

Use Case Name:	Login as Admin	**ID:**	UC-5	**Priority:**	High
Actor:	Admin				
Description:	Process for Admin to log into the system's admin section				
Trigger:	Admin selects admin login option				
Type:	External				
Preconditions:	1. Admin account has been created in the users datastore				

Major Steps:	**Information for Steps:**
Normal Course:	
1. Admin enters email and password for admin account	← Admin Email ← Admin Password
2. System verifies credentials and confirms login	→ Login Confirmation
Alternative Courses:	

Postconditions:	1. Admin is given access to admin pages of system
Exceptions:	1. If email and password given do not match any in the system, user will be alerted and asked to try again
Summary:	

Inputs	**Source**	**Outputs**	**Destination**

Admin Email Admin Password	Admin	Login Confirmation	Admin

Use Case Name:	View Reports	**ID:**	UC-6	**Priority:**	High
Actor:	Admin				
Description:	Process for the admin viewing various reports available on the admin pages of the system				
Trigger:	Admin select option to view a report from the admin dashboard				
Type:	External				
Preconditions:	1. Admin is logged into the admin portion of the system				

Major Steps:	**Information for Steps:**
Normal Course:	
1. Admin selects one of the available reports	← Report Request
2. System displays selected report	→ Selected Report
Alternative Courses:	

Postconditions:			
Exceptions:			
Summary:			
Inputs	**Source**	**Outputs**	**Destination**
Report Request	Admin	Selected Report	Admin

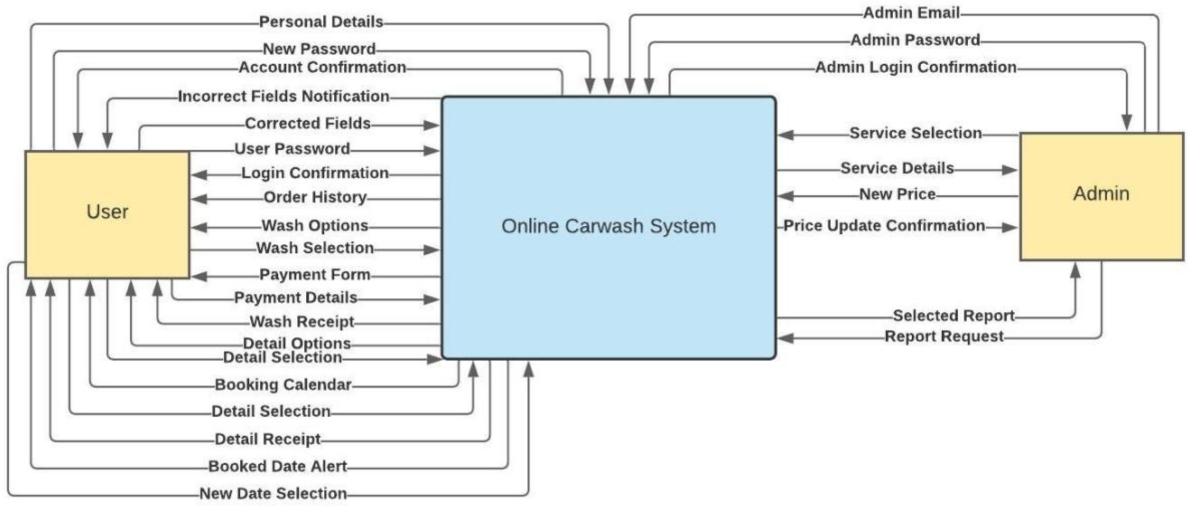
Use Case Name:	Update Pricing	**ID:**	UC-7	**Priority:**	High
Actor:	Admin				
Description:	Process for the admin to update the prices of any services in the system				
Trigger:	Admin selects option to update pricing from the admin dashboard				
Type:	External				
Preconditions:	1. Admin is logged into the admin portion of the system				
Major Steps:			**Information for Steps:**		
Normal Course:					

1. Admin selects service to update		← Service Selection	
2. System displays current service details		→ Service Details	
3. Admin enters new price		← New Price	
4. System confirms updated price		→ Price Update Confirmation	
Alternative Courses:			
Postconditions:	1. Price is updated in database and across system pages		
Exceptions:			
Summary:			
Inputs	**Source**	**Outputs**	**Destination**
Service Selection New Service Price	Admin	Selected Report Price Update Confirmation	Admin

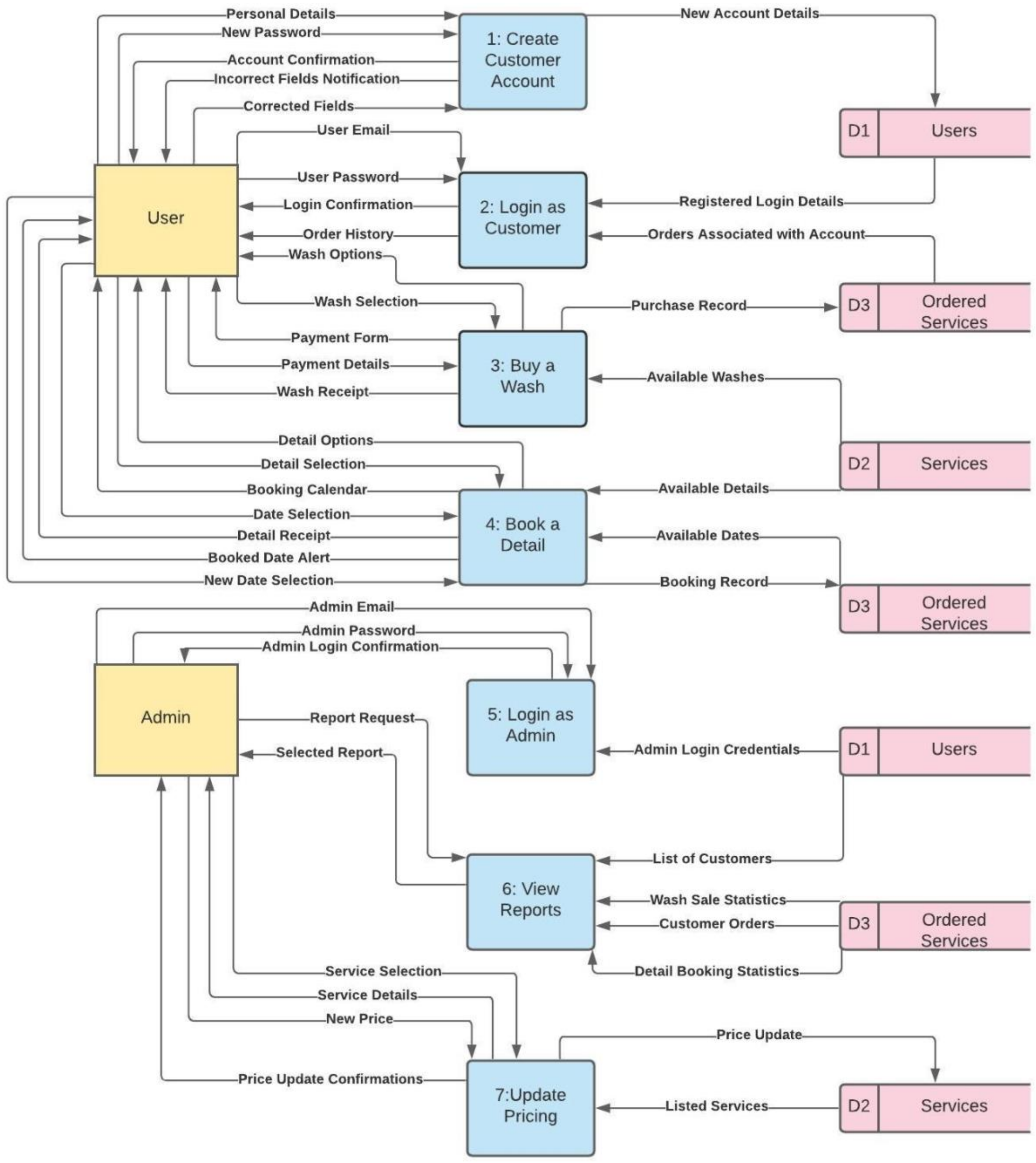
PROCESS AND DATA MODELS

The models for the new system show how it will allow for the use and storage of key data about customers. Customer put in personal information and are assigned a unique account, which is then used to track all future purchases. The ERD goes on to show how sales will be linked to customer orders on their accounts, as well as how user credentials are stored for login. The models also show how the Admin interacts with the system to alter pricing and retrieve key reports

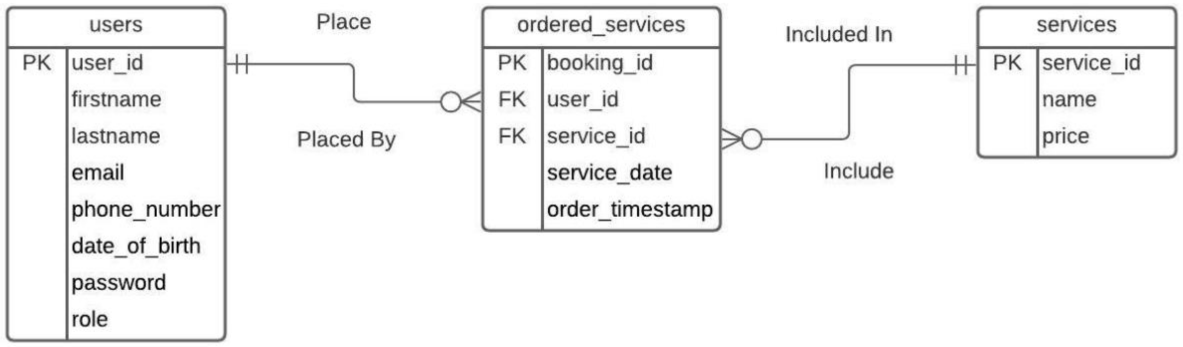
Context Diagram



Level 0 DFD


Personal Details
New Password
Account Confirmation
Incorrect Fields Notification
Corrected Fields
1: Create Customer Account
New Account Details
D1 Users
User Email
User Password
Login Confirmation
Order History
Wash Options
User
2: Login as Customer
Registered Login Details
Orders Associated with Account
Wash Selection
Purchase Record
D3 Ordered Services
Payment Form
Payment Details
Wash Receipt
3: Buy a Wash
Available Washes
Detail Options
Detail Selection
Booking Calendar
Date Selection
Detail Receipt
Booked Date Alert
New Date Selection
D2 Services
Available Details
4: Book a Detail
Available Dates
Booking Record
D3 Ordered Services
Admin Email
Admin Password
Admin Login Confirmation
Admin
5: Login as Admin
Report Request
Selected Report
Admin Login Credentials
D1 Users
List of Customers
6: View Reports
Wash Sale Statistics
Customer Orders
D3 Ordered Services
Detail Booking Statistics
Service Selection
Service Details
New Price
Price Update
Price Update Confirmations
7:Update Pricing
Listed Services
D2 Services

ERD


users
PK user_id
firstname
lastname
email
phone_number
date_of_birth
password
role
Place
Placed By
ordered_services
PK booking_id
FK user_id
FK service_id
service_date
order_timestamp
Included In
Include
services
PK service_id
name
price

TECHNICAL COMPONENTS

We used several languages and modules as well as software solutions to build the structure and functionality of our site:

- HTML: Page and content structure
- CSS: Styling of pages
- MySQL: Database design and querying
- PHP: Form processing, interfacing between website and MySQL database, user sessions
- Javascript: Input verification, alert messages, page redirects
- JQuery: Construction of data tables for data pulled from the MySQL database
- Github: Version control and file sharing
- Repl.it: Collaborative coding in HTML, CSS and Javascript
- Mockaroo: Sample data generation

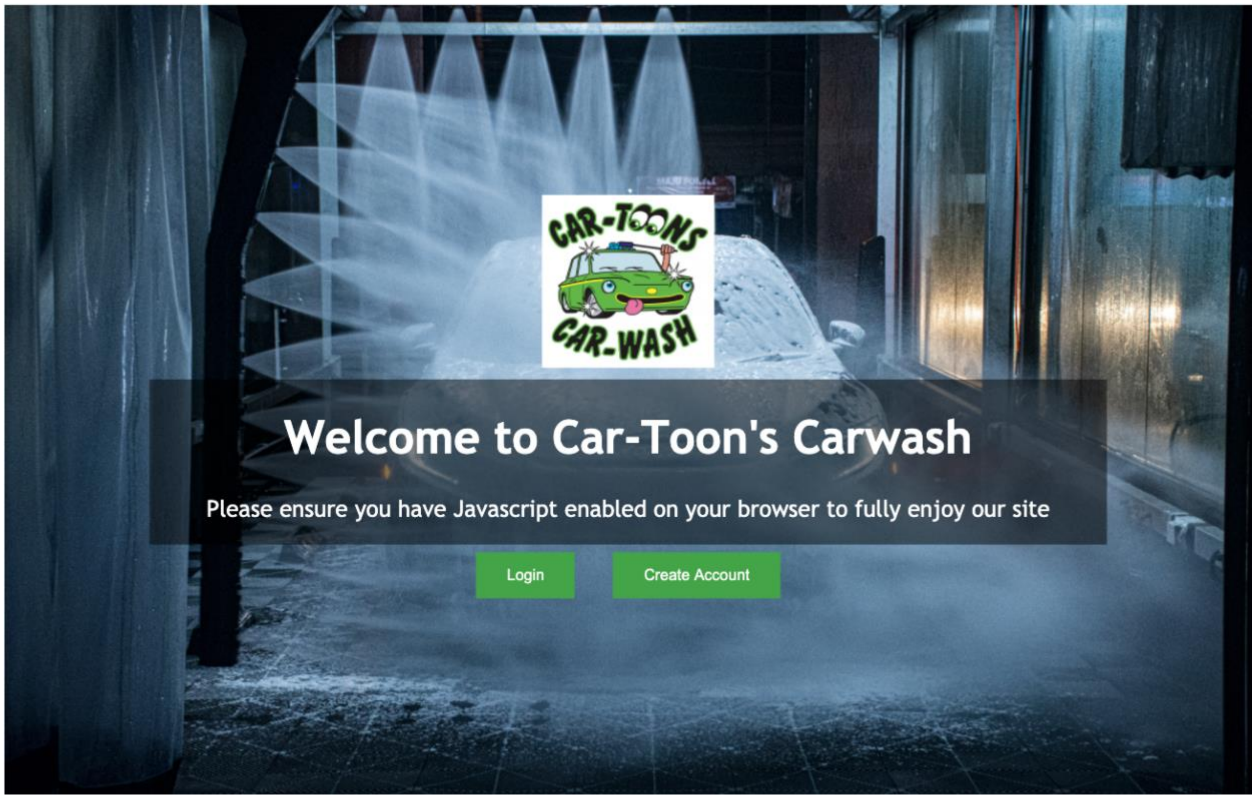
In order to implement these languages, we referenced a number of online resources:

- W3Schools: Basic programming ideas, examples and language references
- StackOverflow: Troubleshooting, ideas for implementation, code snippets
- GeekForGeeks: Help with designing certain Javascript functions
- Datatables Forum: Troubleshooting and styling tips for working with JQuery's Datatables functions
- FreeCodeCamp: Tutorials on styling
- purecss.io: Styling templates
- phpchecker.com: Syntax checking with specific errors for PHP

SYSTEM WALKTHROUGH

Landing Page

This is where all customers will start their journey. As the system is only accessible to registered users, it has links to either login or create an account. We have also included a message to remind customers to ensure Javascript is enabled, as several of our pages depend on it

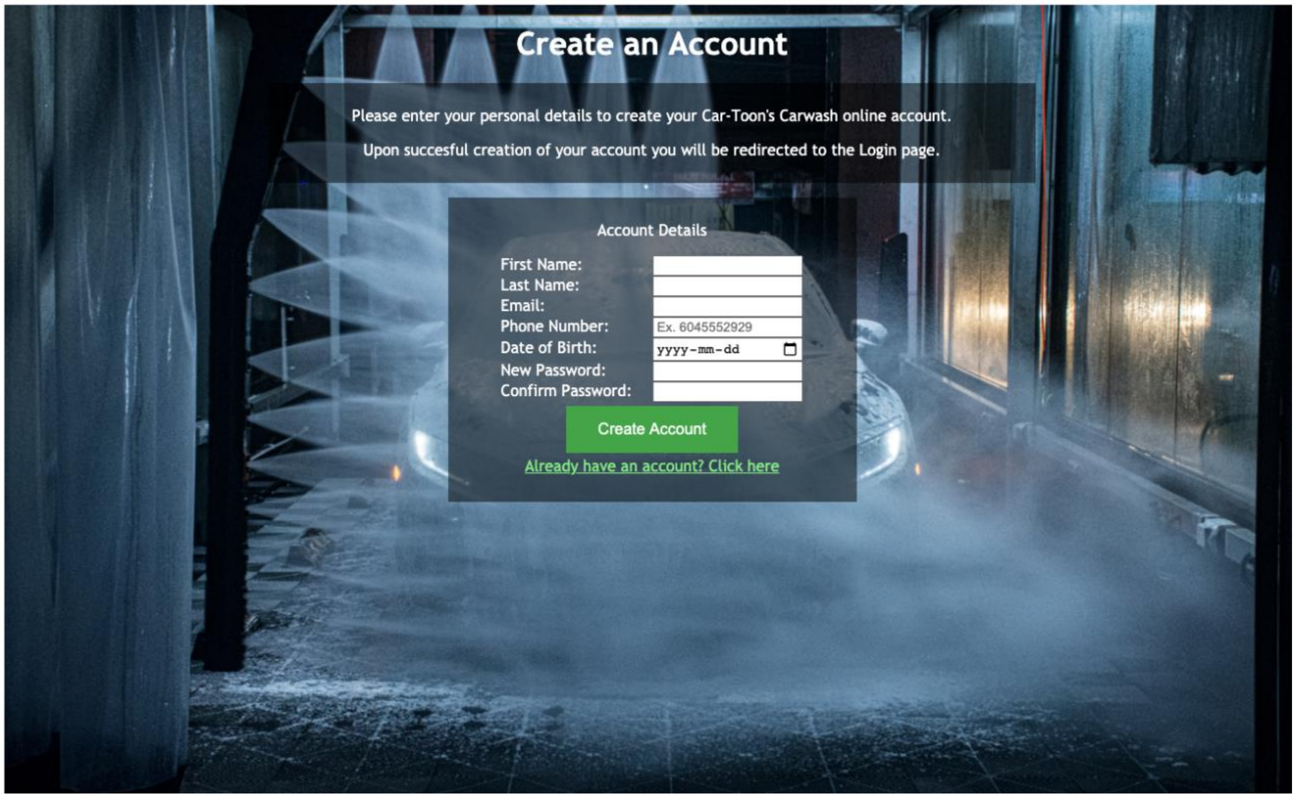


Welcome to Car-Toon's Carwash

Please ensure you have Javascript enabled on your browser to fully enjoy our site

Create Customer Account

If a customer clicks the "Create Account" button on the homepage, they are taken to the page shown below and begin the Create Customer Account use case.



Create an Account

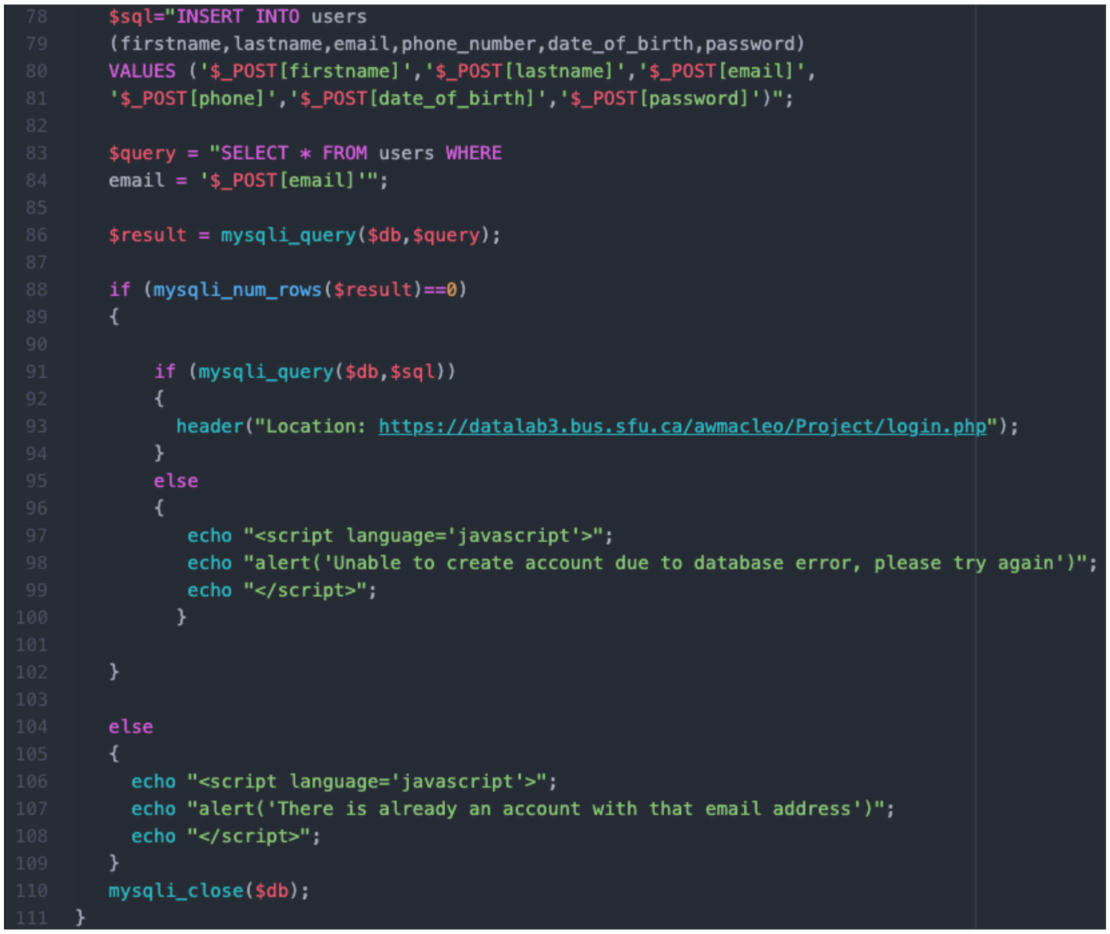
Please enter your personal details to create your Car-Toon's Carwash online account.

Upon succesful creation of your account you will be redirected to the Login page.

Account Details

First Name:
Last Name:
Email:
Phone Number: Ex. 6045552929
Date of Birth: yyyy-mm-dd
New Password:
Confirm Password:

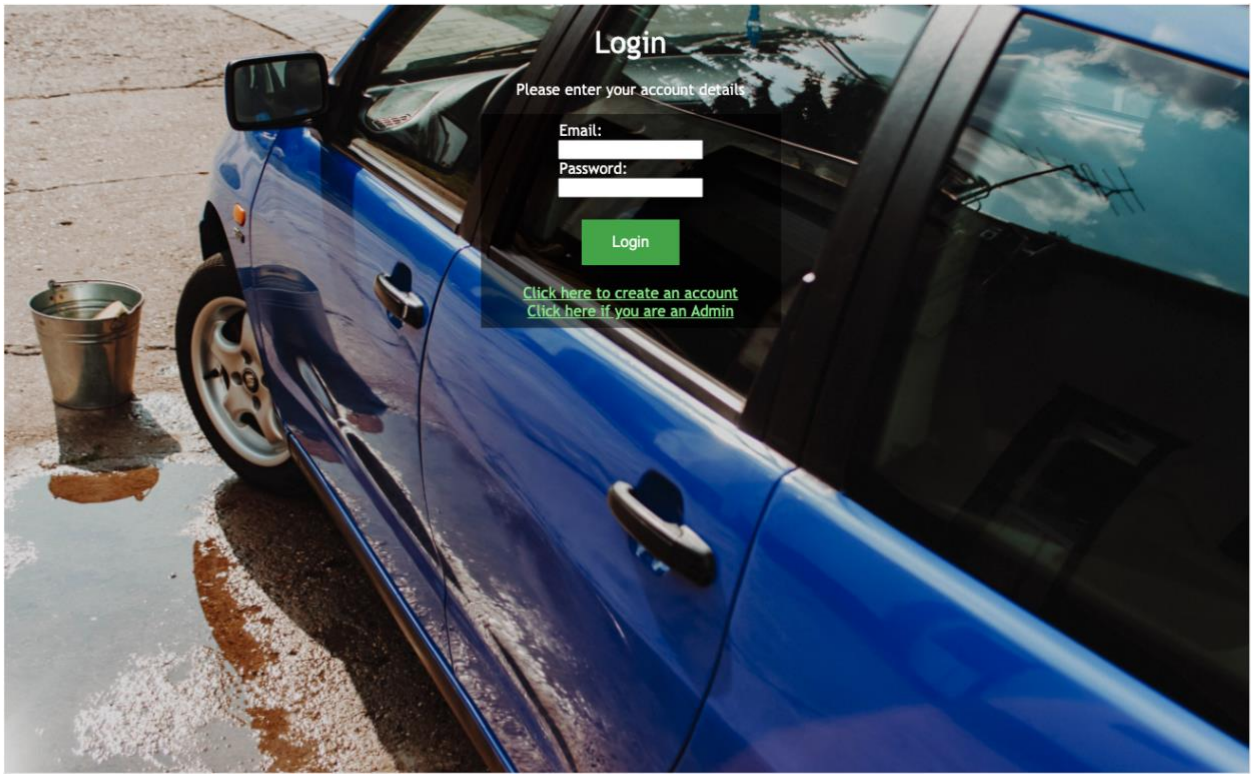
Pictured below is a critical piece of PHP for account creation. We have used nested conditionals to make sure email addresses are not duplicated and customers are made aware if their account creation fails. The outer if statement on line 88 first checks whether there are any accounts in the database with the customer's email address. If there are none, an attempt is made to insert the account information into the database. If this fails, the customer is alerted via a Javascript alert box. If there is already an account with the email entered, no attempt is made and a Javascript alert tells the user that the email is already in the system.



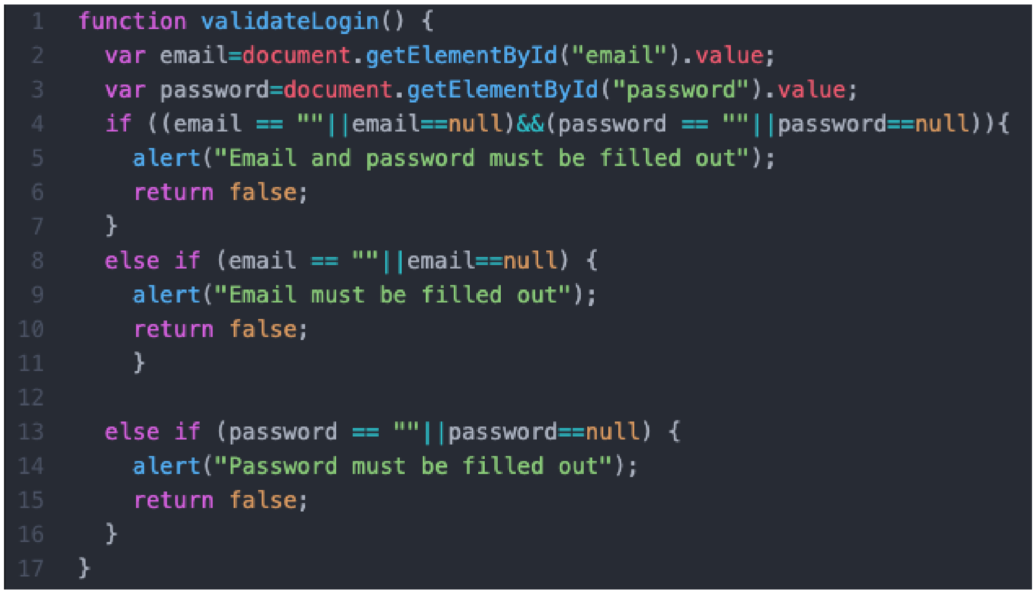
```
 78       $sql="INSERT INTO users
 79       (firstname,lastname,email,phone_number,date_of_birth,password)
 80       VALUES ('$_POST[firstname]','$_POST[lastname]','$_POST[email]',
 81       '$_POST[phone]','$_POST[date_of_birth]','$_POST[password]')";
 82
 83       $query = "SELECT * FROM users WHERE
 84       email = '$_POST[email]'";
 85
 86       $result = mysqli_query($db,$query);
 87
 88       if (mysqli_num_rows($result)==0)
 89       {
 90
 91           if (mysqli_query($db,$sql))
 92           {
 93             header("Location: https://datalab3.bus.sfu.ca/awmacleo/Project/login.php");
 94           }
 95           else
 96           {
 97              echo "<script language='javascript'>";
 98              echo "alert('Unable to create account due to database error, please try again')";
 99              echo "</script>";
100             }
101
102       }
103
104       else
105       {
106         echo "<script language='javascript'>";
107         echo "alert('There is already an account with that email address')";
108         echo "</script>";
109       }
110       mysqli_close($db);
111    }
```

Login as Customer

Once the customer has an account, they will have to login to access the rest of the site.

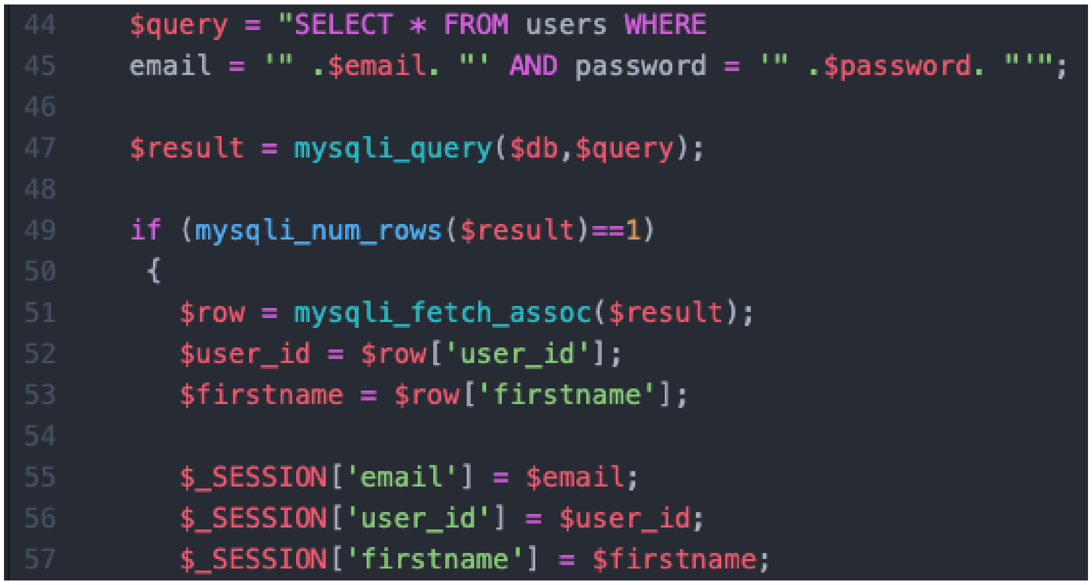

Login
Please enter your account details
Email:
Password:
Login
Click here to create an account
Click here if you are an Admin

We used a Javascript function on the login page to ensure both fields are filled out before the form is submitted.



```
function validateLogin() {
  var email=document.getElementById("email").value;
  var password=document.getElementById("password").value;
  if ((email == ""||email==null)&&(password == ""||password==null)){
    alert("Email and password must be filled out");
    return false;
  }
  else if (email == ""||email==null) {
    alert("Email must be filled out");
    return false;
    }

  else if (password == ""||password==null) {
    alert("Password must be filled out");
    return false;
  }
}
```

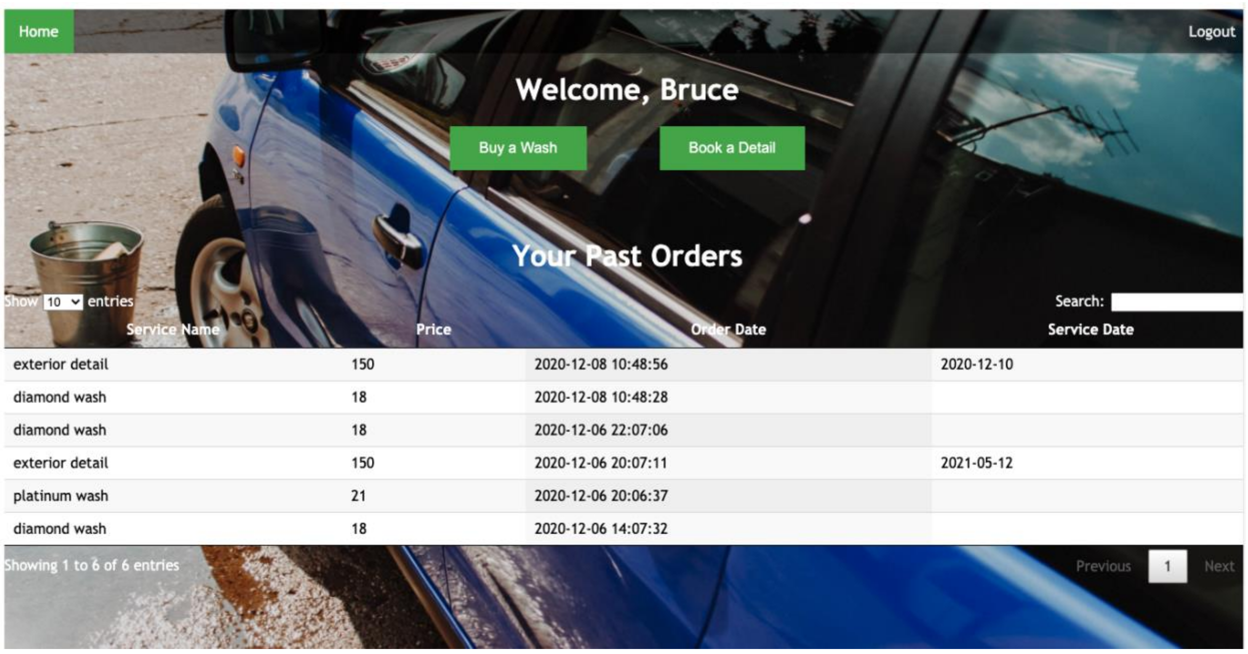
The following PHP snippet shows where critical session variables are set. When a customer logs in, the system stores their email, user id and first name use in later pages.



```php
$query = "SELECT * FROM users WHERE
email = '" .$email. "' AND password = '" .$password. "'";

$result = mysqli_query($db,$query);

if (mysqli_num_rows($result)==1)
 {
   $row = mysqli_fetch_assoc($result);
   $user_id = $row['user_id'];
   $firstname = $row['firstname'];

   $_SESSION['email'] = $email;
   $_SESSION['user_id'] = $user_id;
   $_SESSION['firstname'] = $firstname;
```

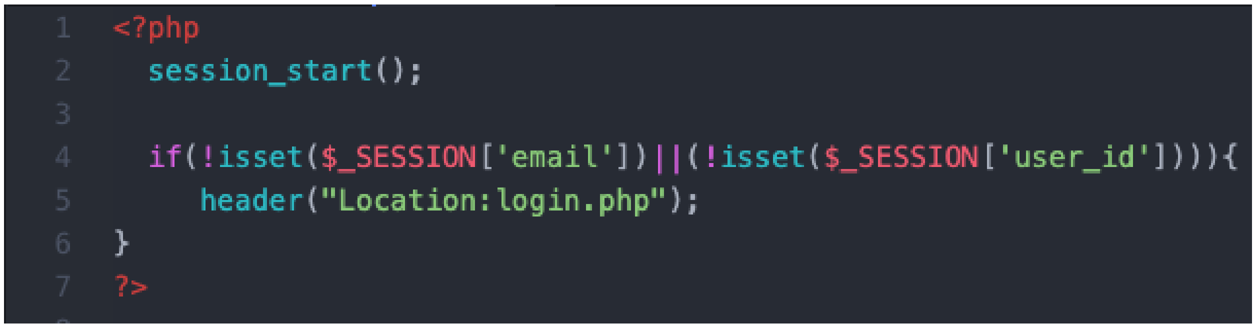
Customer Profile

Upon logging in customers are presented with their profile page. The page is personalized using the session variable for their first name that was set during login, and their order history is displayed using the JQuery Datatables function. Service date will be shown for any detail bookings, while for washes only the order date is shown as they have no set service date.



Service Name	Price	Order Date	Service Date
exterior detail	150	2020-12-08 10:48:56	2020-12-10
diamond wash	18	2020-12-08 10:48:28	
diamond wash	18	2020-12-06 22:07:06	
exterior detail	150	2020-12-06 20:07:11	2021-05-12
platinum wash	21	2020-12-06 20:06:37	
diamond wash	18	2020-12-06 14:07:32	

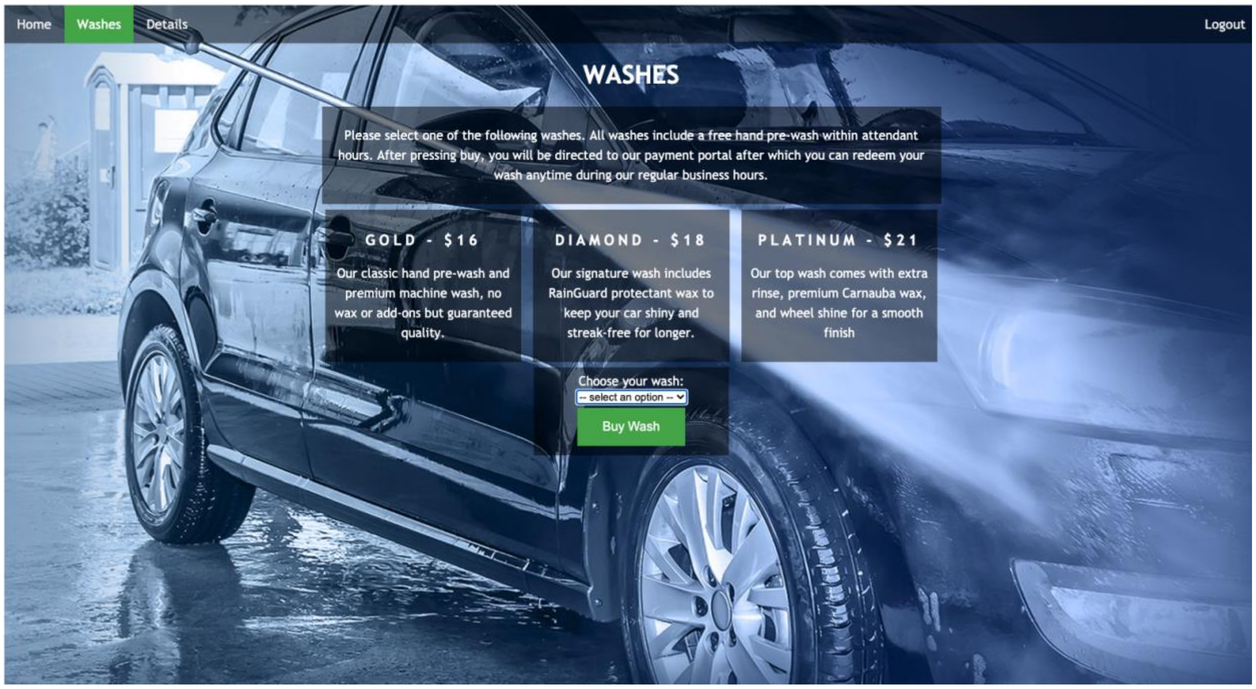
Like all pages following the login, this PHP snippet is located at the top of the page to ensure the user is logged in, and will redirect them to the login page if they manage to access the page without logging in.



```php
<?php
  session_start();

  if(!isset($_SESSION['email'])||(!isset($_SESSION['user_id']))){
     header("Location:login.php");
}
?>
```

Buy a Wash

Upon selecting the buy a wash option, the customer is presented with information on washes available. Once they select one and hit submit, they will be taken to the payment page



WASHES

Please select one of the following washes. All washes include a free hand pre-wash within attendant hours. After pressing buy, you will be directed to our payment portal after which you can redeem your wash anytime during our regular business hours.

GOLD - $16

Our classic hand pre-wash and premium machine wash, no wax or add-ons but guaranteed quality.

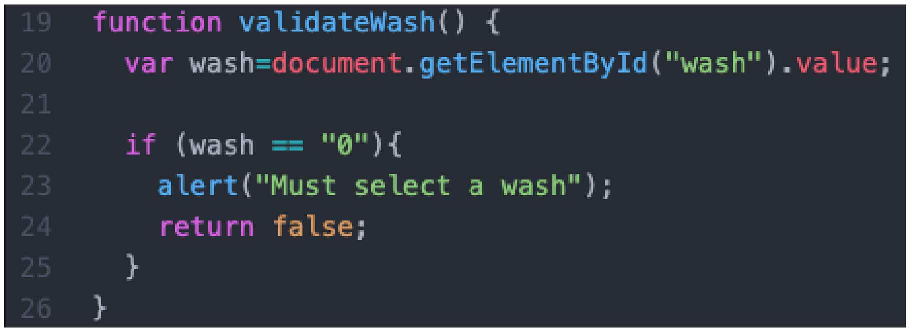
DIAMOND - $18

Our signature wash includes RainGuard protectant wax to keep your car shiny and streak-free for longer.

PLATINUM - $21

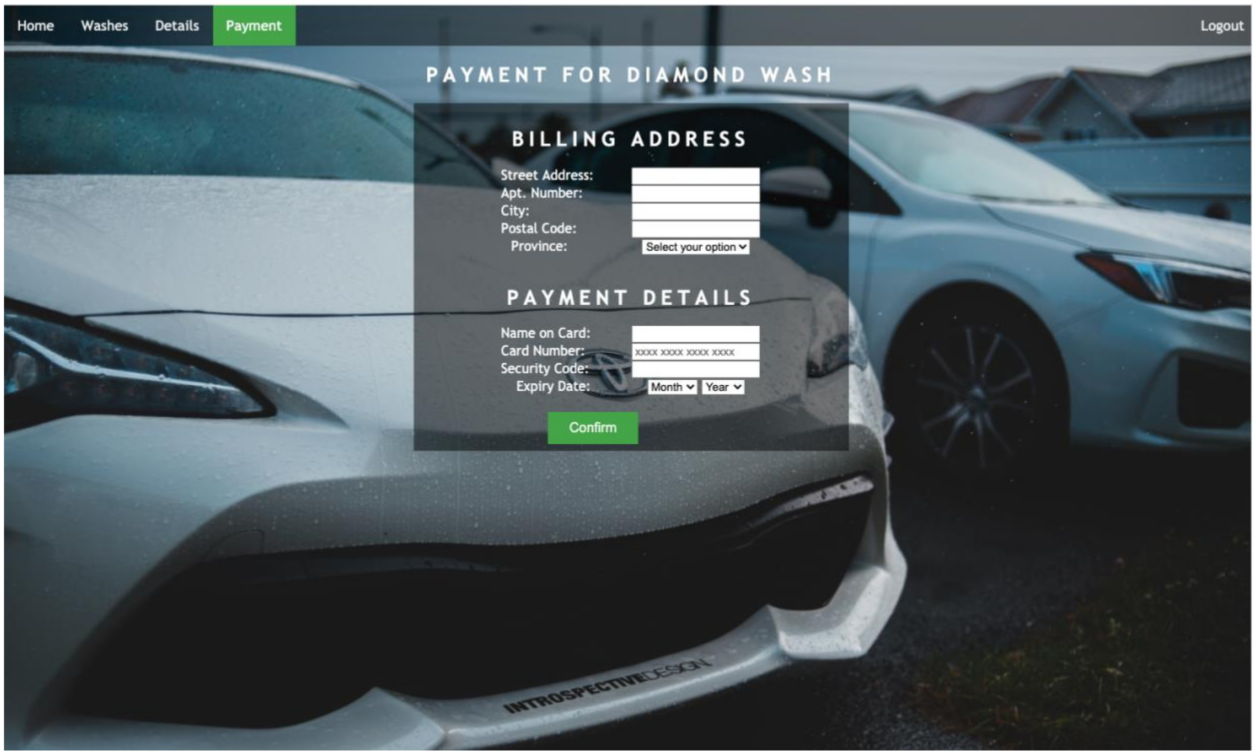
Our top wash comes with extra rinse, premium Carnauba wax, and wheel shine for a smooth finish

The following Javascript function is used to ensure that a wash is selected before the form is processed, displaying an alert box if not:



```
function validateWash() {
  var wash=document.getElementById("wash").value;

  if (wash == "0"){
    alert("Must select a wash");
    return false;
  }
}
```

This payment page is a placeholder. At this stage of the project, the system is not capable of processing transactions. The page is currently there to demonstrate the flow of the website and to ensure that purchase entries are not made in the database for washes until payment has been confirmed. In the final system this page will likely be linked to a third-party processor.



PAYMENT FOR DIAMOND WASH

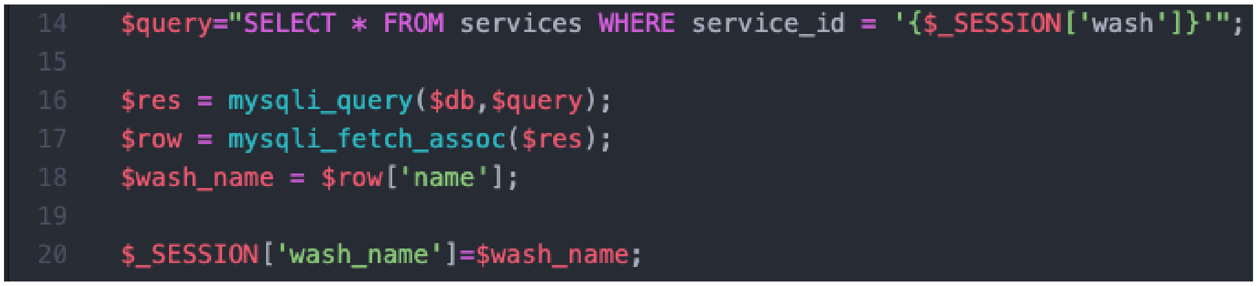
BILLING ADDRESS

Street Address:
Apt. Number:
City:
Postal Code:
Province: Select your option

PAYMENT DETAILS

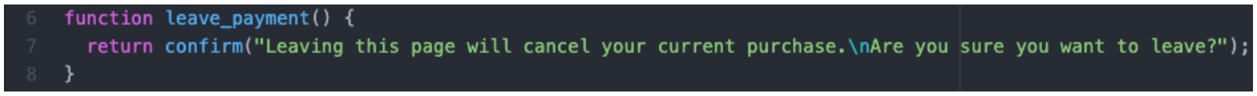
Name on Card:
Card Number: xxxx xxxx xxxx xxxx
Security Code:
Expiry Date: Month Year

This PHP is used to set the name of the wash selected on the previous page so that the text at the top of the payment page can be adjusted for whatever wash the user is purchasing.



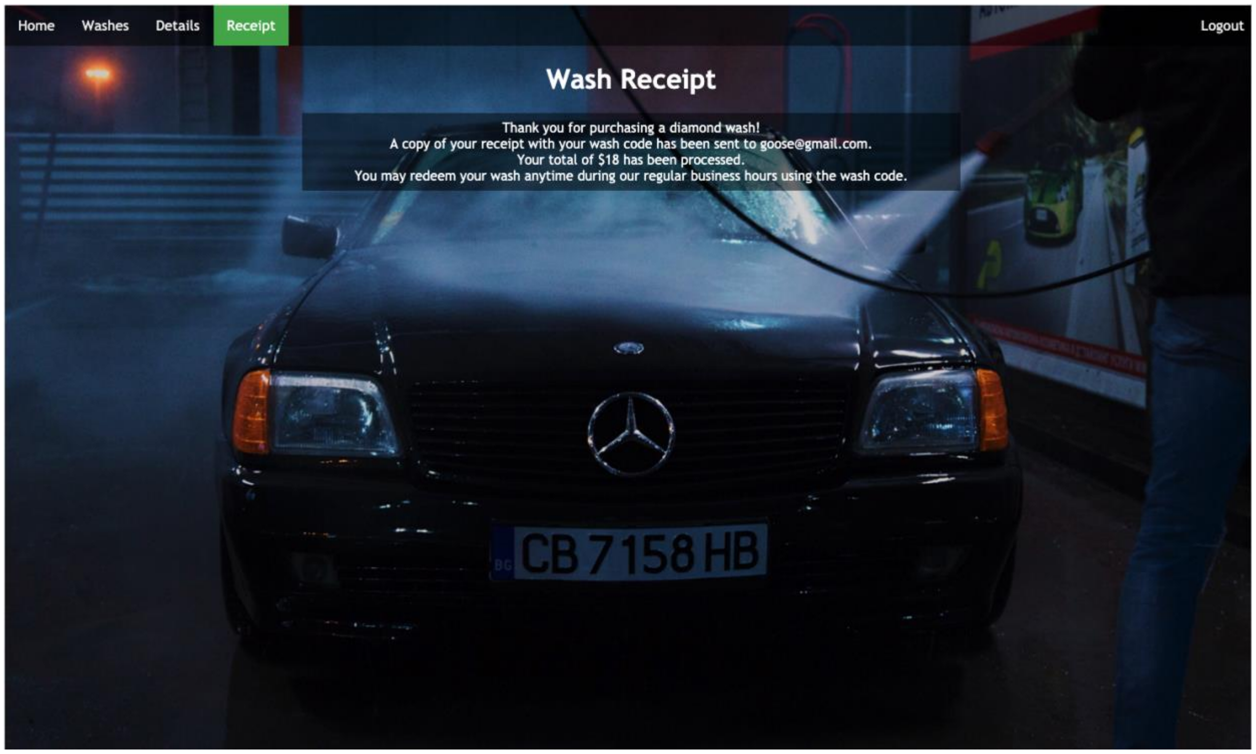
```php
$query="SELECT * FROM services WHERE service_id = '{$_SESSION['wash']}'";

$res = mysqli_query($db,$query);
$row = mysqli_fetch_assoc($res);
$wash_name = $row['name'];

$_SESSION['wash_name']=$wash_name;
```

To keep customers from mistakenly leaving the payment page before their purchase is complete, and thus erasing the current purchase, the following Javascript function asks the user to confirm if they click on any navbar links while on this page.



```javascript
function leave_payment() {
  return confirm("Leaving this page will cancel your current purchase.\nAre you sure you want to leave?");
}
```

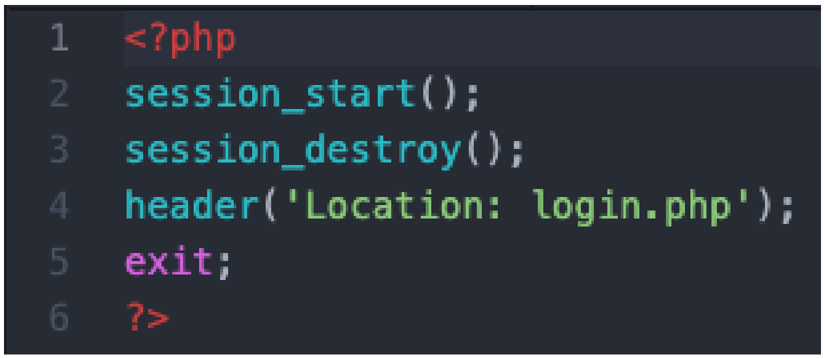
After payment is confirmed, the receipt page is displayed. Content in the receipt message is personalized to the user and their specific purchase using session variables set previously.



Wash Receipt

Thank you for purchasing a diamond wash!
A copy of your receipt with your wash code has been sent to goose@gmail.com.
Your total of $18 has been processed.
You may redeem your wash anytime during our regular business hours using the wash code.

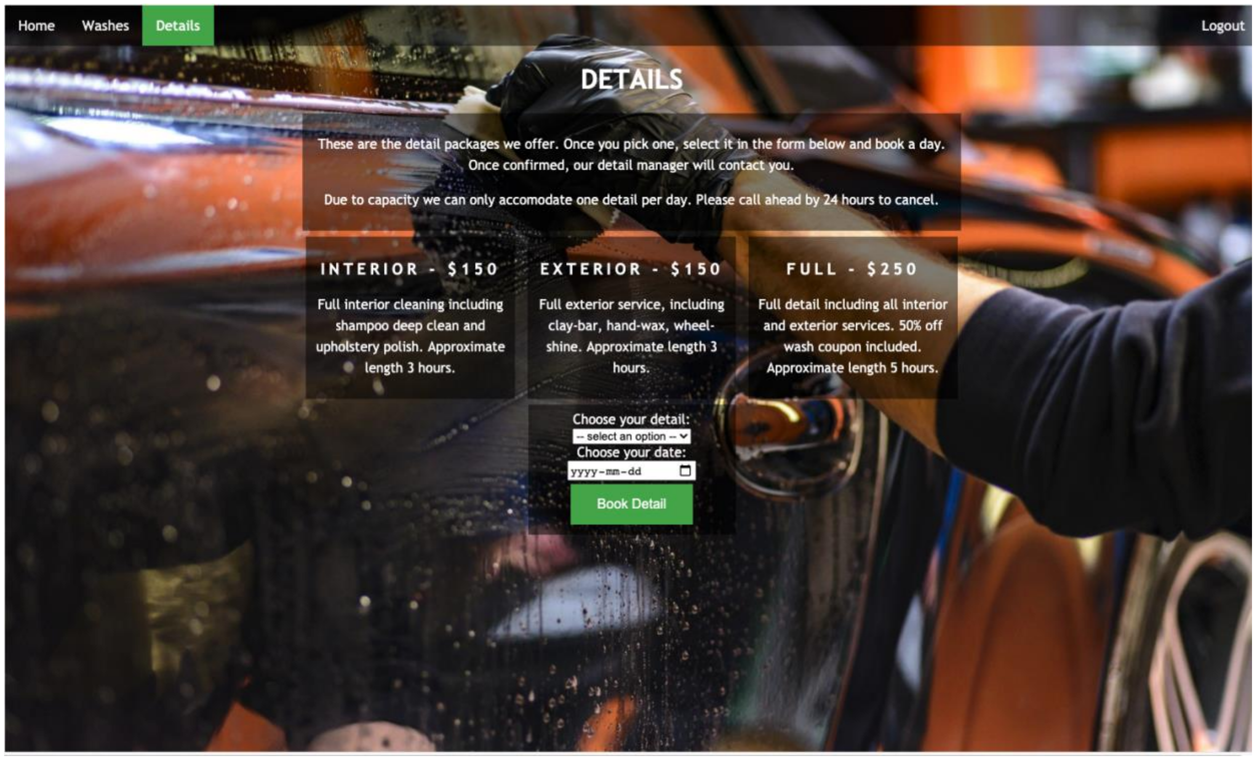
Should the user decide to logout at this point, a Javascript window will ask them to confirm. To prevent accidental logouts this has been applied to the logout button sitewide. If they do confirm, the PHP shown below will destroy their current session and redirect them back to the login page.



```php
<?php
session_start();
session_destroy();
header('Location: login.php');
exit;
?>
```

Book a Detail

If the customer chooses to book a detail instead of buy a wash, they are taken to the page below. They can pick which detail they want and then the date.



DETAILS

These are the detail packages we offer. Once you pick one, select it in the form below and book a day. Once confirmed, our detail manager will contact you.

Due to capacity we can only accomodate one detail per day. Please call ahead by 24 hours to cancel.

INTERIOR - $150

Full interior cleaning including shampoo deep clean and upholstery polish. Approximate length 3 hours.

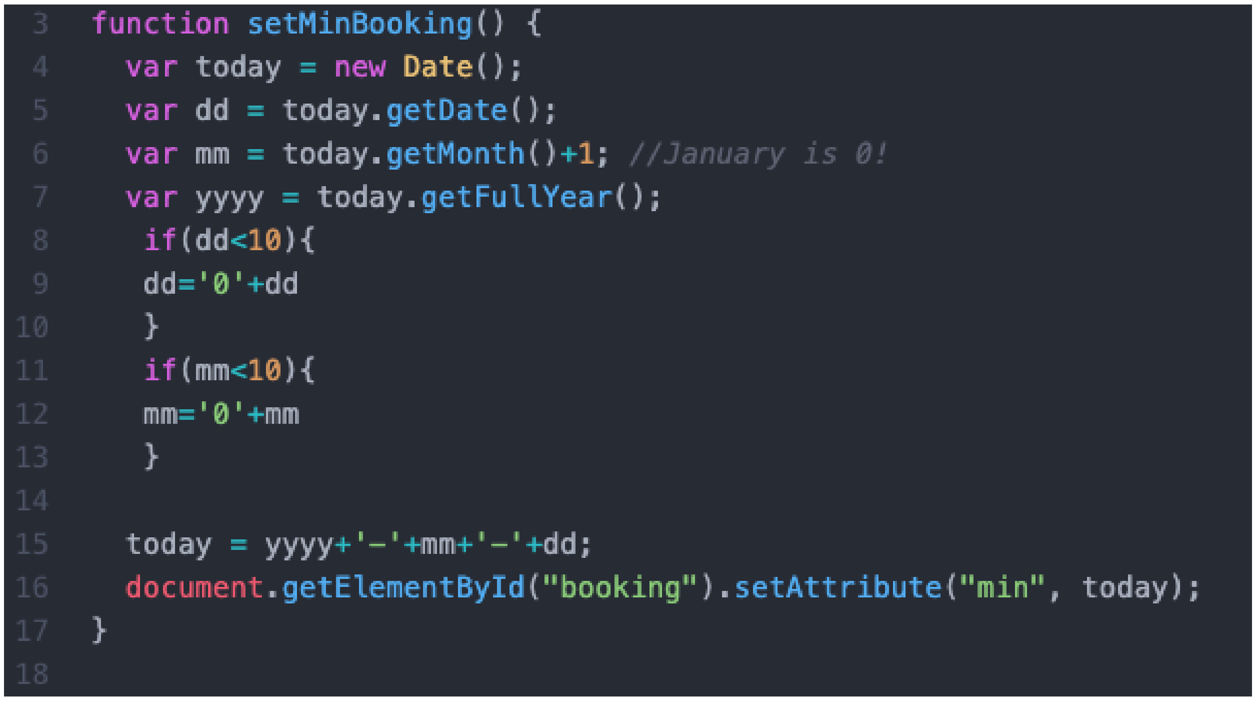
EXTERIOR - $150

Full exterior service, including clay-bar, hand-wax, wheel-shine. Approximate length 3 hours.

FULL - $250

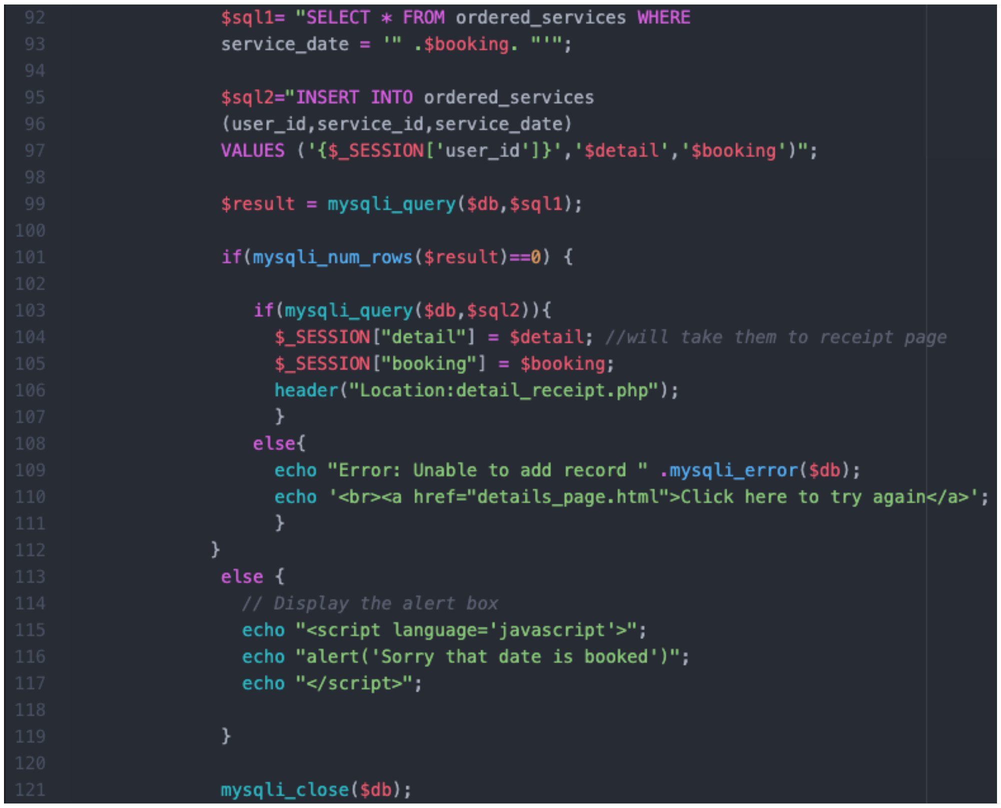
Full detail including all interior and exterior services. 50% off wash coupon included. Approximate length 5 hours.

With some help from StackOverflow, we were able to create the following Javascript function. It is applied to the date field of the booking form, and dynamically sets the minimum date to today and greys out any past dates.



```javascript
function setMinBooking() {
  var today = new Date();
  var dd = today.getDate();
  var mm = today.getMonth()+1; //January is 0!
  var yyyy = today.getFullYear();
   if(dd<10){
   dd='0'+dd
   }
   if(mm<10){
   mm='0'+mm
   }

  today = yyyy+'-'+mm+'-'+dd;
  document.getElementById("booking").setAttribute("min", today);
}

```

If the user chooses a date that is already booked (the business can only handle one detail per day) then they will be shown an alert and asked to try another date. This is handled in a similar manner to the create account page where emails are not allowed to be duplicated. Using nested conditionals, the system first checks if the selected date is booked before attempting to add the date to the database.

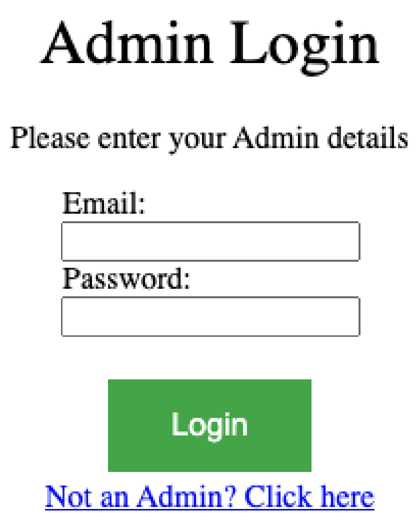


```
$sql1= "SELECT * FROM ordered_services WHERE
service_date = '" .$booking. "'";

$sql2="INSERT INTO ordered_services
(user_id,service_id,service_date)
VALUES ('{$_SESSION['user_id']}','$detail','$booking')";

$result = mysqli_query($db,$sql1);

if(mysqli_num_rows($result)==0) {

   if(mysqli_query($db,$sql2)){
     $_SESSION["detail"] = $detail; //will take them to receipt page
     $_SESSION["booking"] = $booking;
     header("Location:detail_receipt.php");
     }
   else{
     echo "Error: Unable to add record " .mysqli_error($db);
     echo '<br><a href="details_page.html">Click here to try again</a>';
     }
}
else {
  // Display the alert box
  echo "<script language='javascript'>";
  echo "alert('Sorry that date is booked')";
  echo "</script>";

}

mysqli_close($db);
```

After this the user is shown the same receipt page as above, but with different text informing them that the detail manager will be in touch and they will pay at time of pickup. Unlike washes, details are not paid for online.

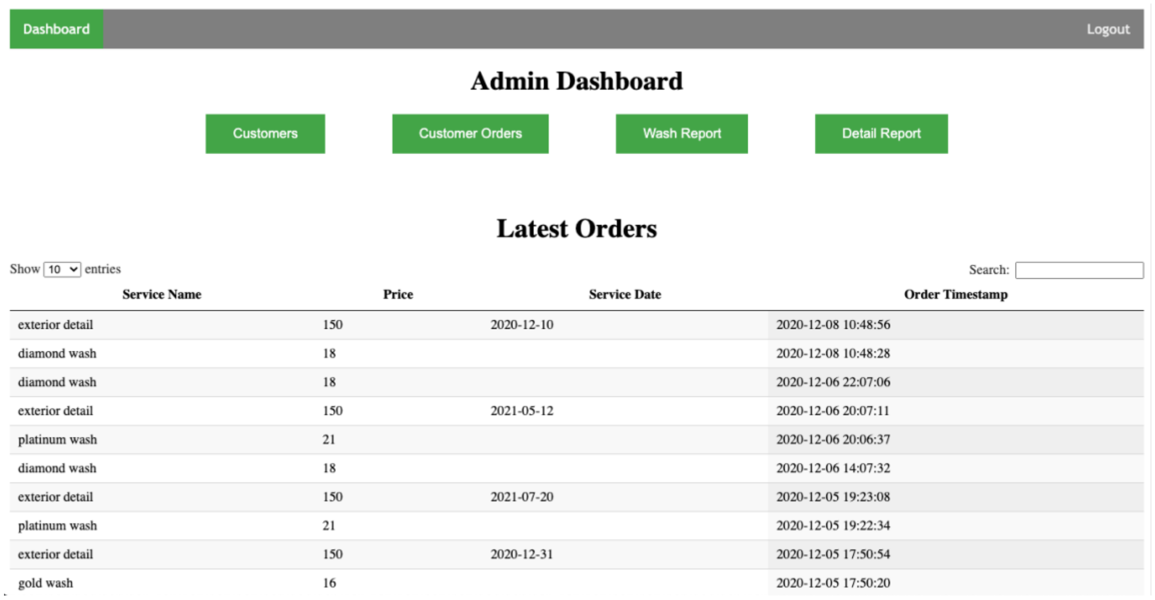
Login as Admin

The Admin logs in through a separate login page that gives them access to the admin-specific areas of the site. They are also able to login to the customer-facing portion if they need to test it, but to access any reports they must use the admin portal. This login page works much like the normal one, but it also checks to make sure that the credentials have the role value 1 in the database, which indicates an admin. Regular users are assigned the value 0 by default to differentiate the admin.



View Reports

Once logged in, the Admin is shown their dashboard. It includes a live table with the most recent orders, as well as links to various reports.

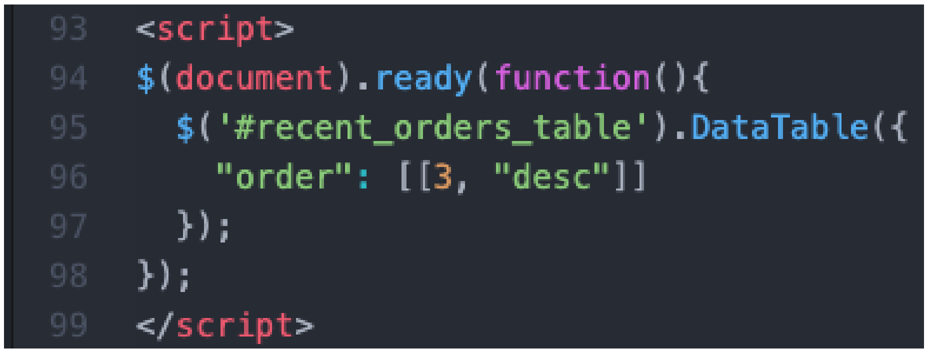


Admin Dashboard

Latest Orders

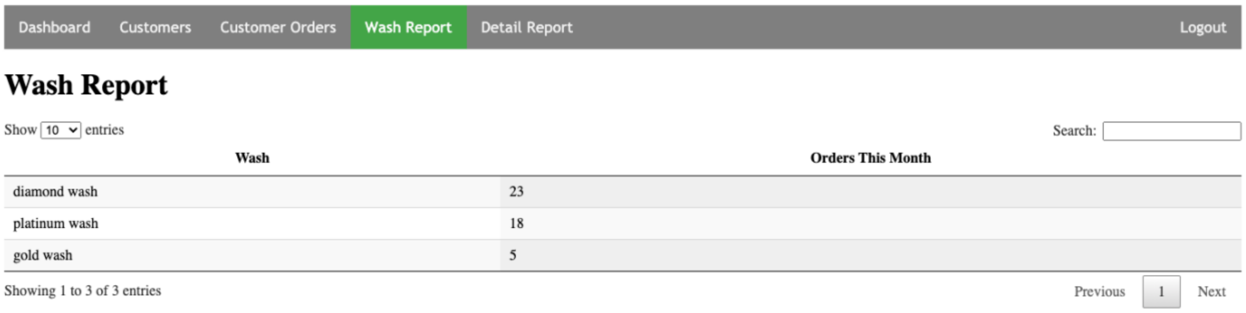
Service Name	Price	Service Date	Order Timestamp
exterior detail	150	2020-12-10	2020-12-08 10:48:56
diamond wash	18		2020-12-08 10:48:28
diamond wash	18		2020-12-06 22:07:06
exterior detail	150	2021-05-12	2020-12-06 20:07:11
platinum wash	21		2020-12-06 20:06:37
diamond wash	18		2020-12-06 14:07:32
exterior detail	150	2021-07-20	2020-12-05 19:23:08
platinum wash	21		2020-12-05 19:22:34
exterior detail	150	2020-12-31	2020-12-05 17:50:54
gold wash	16		2020-12-05 17:50:20

The following call to the Datatables function includes the order initialization parameter, instructing Datatables to order the data in descending order by the Order Timestamp column to ensure the latest orders are listed first, as this page is intended to be a live snapshot of site activity.



```
<script>
$(document).ready(function(){
  $('#recent_orders_table').DataTable({
    "order": [[3, "desc"]]
  });
});
</script>
```

Shown below is one of the reports the Admin can view, showing the sales of Washes in the most recent month, so they can quickly see how each wash is selling. This is generated using the COUNT and GROUP BY SQL functions, sent to the database through PHP, and displayed with JQuery Datatables. Similar to the above table, we used the order parameter to order the data table by washes ordered.



Wash Report

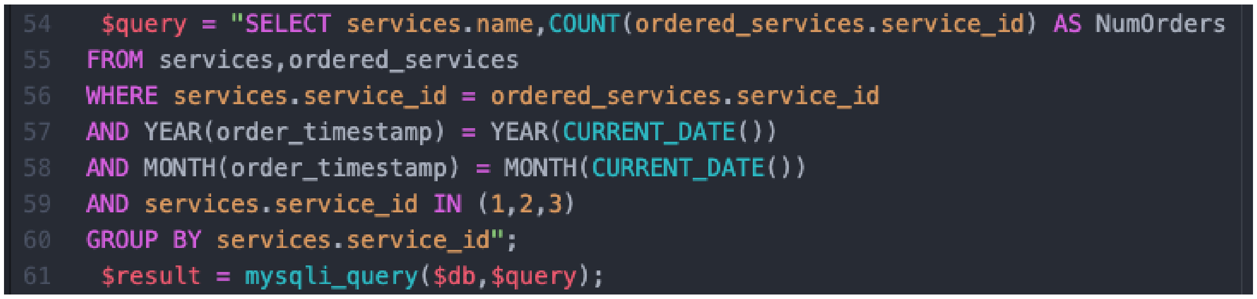
Show 10 entries

Search:

Wash	Orders This Month
diamond wash	23
platinum wash	18
gold wash	5

Showing 1 to 3 of 3 entries

Previous 1 Next



```
 $query = "SELECT services.name,COUNT(ordered_services.service_id) AS NumOrders
FROM services,ordered_services
WHERE services.service_id = ordered_services.service_id
AND YEAR(order_timestamp) = YEAR(CURRENT_DATE())
AND MONTH(order_timestamp) = MONTH(CURRENT_DATE())
AND services.service_id IN (1,2,3)
GROUP BY services.service_id";
 $result = mysqli_query($db,$query);
```

FUTURE IMPROVEMENTS

This working prototype has met the core requirements for the system, but there is more work to be done to make it ready for the public. The following are recommendations for what we believe would add value later in the development of the system.

Cancel Bookings Online:
The current version assumes any changes or cancellations to detail bookings must be done by phone or email. Later iterations should include a page where customers can cancel bookings. A simple way to implement this would be to have a page where all a customer's active bookings are listed. Next to each booking there could be a cancel button linked to a PHP file. That file would include a SQL script to search the ordered services table for a booking tied to the same unique booking ID and would remove the associated row from the database. This would automatically make that date available for new bookings due to the way the booking system searches ordered services to determine if a date is booked. Ideally there would also be a notification email automatically sent to the detail manager or system admin to notify them of the cancellation.

Timeslots:
Because the business can currently only handle a single detail per day, our system considers a date unavailable if there an entry in ordered services with that date as the service date. If the company's capacity should change, we could add a timeslot column to the ordered services table. The data in this column could be as simple as a 1 to indicate morning and a 2 to indicate afternoon. When customers book a detail, they would choose a timeslot as well as a date. Instead of just checking for rows with the same date as entered, the system would check if there was a matching date and timeslot, and the booking would only not be available if the date and timeslot were both already in the database. If there were a detail in the table with the same date but a different timeslot, the system would allow the booking.

Guest Login:
In the current iteration, users must create an account and login to use the system. It would be convenient for less-frequent customers to be able to make purchases as a guest. These purchases could be attached to a dummy account in the users table to keep track of them and customers would simply enter the necessary details for payment. For details, minimal information such as email and name would be stored so that the detail manager can get in touch with the guest to set up the detail. The ability to cancel bookings online would not be available to guest users, since they will not be able to login to view and alter their detail bookings. They would be required to use the current method of calling or emailing ahead to cancel.

Security:
Current security is minimal as the system is designed to run in a private server with sample data. Some important security additions include storing passwords as hash instead of plaintext and using PHP prepared statements for database insertion activities to prevent SQL injection attacks. Passwords can be handled by PHP's built-in encryption and decryption functions. Passwords would be encrypted upon account creation and stored as hash, then decrypted when so they can be checked against the user's input when they log in.

Input Validation:
We have built and implemented several Javascript functions to validate key user inputs. The full version of the site should have more comprehensive and user-friendly validation. JQuery's validation library could be a good option for this, as it is capable of validating formats such as credit cards and email addresses.

Shopping Cart:
We did not implement a shopping cart function at this time because wash and detail services are typically bought once a week at most and buying in a batch is not a very common need for customers. However, on a longer timeline it could be worth implementing for washes. The shopping cart could be built primarily with PHP, with washes added to an associative array when the add to cart button is clicked, and the array used to create database entries. Users should also have the option to edit quantities and remove items before checkout, which could be done by creating buttons which alter the values associated with keys in the array such as quantity.

Portability:
Currently, the website is optimized for use on modern browsers such as Google Chrome with Javascript enabled. Later versions should be adapted for a wider variety of browsers, including legacy versions. Mobile formatting should also be considered, and there are several CSS libraries that can help scale our site more effectively to different screen shapes.

Email Receipts:
While it is stated in the use cases and site that the user will be emailed receipts for washes and details, this functionality has not been implemented. This is critical to implement before public deployment as these receipts are used to redeem washes and email communication is needed to set up details. This can be done using PHP's mail() function once we have an email server set up.

Update Prices:
Use Case 7 refers to the Admin updating prices in the site. Due to time constraints this has not yet been implemented. To do so the prices listed in the site should be echoed in PHP based on the values in the services table. The Admin should be able to enter new prices into a form and have the services table edited by SQL statements to reflect them.

AJAX Form Submission:
Forms are currently submitted to the same PHP page they are defined on to allow for Javascript popups in case of errors and reduce the number of files needed. In the future, AJAX's form submission functions could be used to make this process even smoother. For example, AJAX could be used to check whether a date is already booked for a detail and submit the new booking, which would allow Javascript alerts for booked dates to appear on the same page seamlessly.

Payment:
Our current payment page is merely a placeholder to demonstrate the flow of the site. In the future we will likely partner with a third-party payment provider like Square or Stripe and use their API to create a payment portal on our site.

LESSONS LEARNED

As our team's first foray into web development, this has been a valuable learning experience. We are proud of the progress we have made, as we all started with next to no coding experience and have now managed to build a functional prototype website. What follows are some of the key learnings we feel will be valuable to our future endeavours and future cohorts of BUS 465.

- **Scope Planning**: We found it was easy to get carried away and allow scope creep to expand the project beyond what was possible. As can be seen from our future improvements section, we have a number of additional ideas we did not get to implement. A rigorous conceptualization and planning process where the entire team agrees on a set scope early in the project could reduce stress by setting clear, stable goals.
- **Frameworks**: Throughout the project, we applied the knowledge from our BUS 465 classes and labs to build the majority of our functionalities from scratch in the languages we have learned. Learning to work with frameworks like Bootstrap could cut down the amount of coding we have to do for mundane things and allow us to expand scope.
- **Libraries**: One issue we ran into was how to integrate libraries into our code. Complex libraries were difficult to integrate with the code we created ourselves and we often decided to do things ourselves. Doing more in-depth research in the planning phase into the libraries available for functionalities we need could help us save a great deal of time.