CPS630 Iterations I & II Report: Smart Customer Services

Team 30

Anthony Tran, David Tran, Sera Wong

500981445, 500890757, 500702293

Members	Percentage
Anthony Tran	33%
David Tran	33%
Sera Wong	33%

Project Objective

Smart Customer Services is an online kitchen appliance store that provides green delivery services to customers during the Covid-19 pandemic. It aims to give users a stress-free shopping experience with responsive user-accessible features and a simple delivery system. On the website, users will be able to create and sign into their own accounts, make purchases, manage their shopping cart and orders, learn more about the company, and write their own reviews.

Languages and Tools Used

- HTML
- CSS
- JavaScript
- JQuery
- Materialize CSS (Similar to Bootstrap)
- MySQL
- PHP
- XAMPP

Implementation of Database Maintain Mode

To prevent unwanted modifications to our database, the Database Maintain Mode pages are only accessible when the user is logged into an admin account. We have implemented four different functionalities: insert, delete, select, and update.

MySQL commands used

Establishing connection

```php
$db = mysqli_connect("localhost", "root", "", "project");
if ($db->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

mysqli_connect was used to open a connection to the database. If the connection fails, an error message will be displayed.

Displaying table names

```
$sql = "SHOW TABLES";
foreach ($db->query($sql) as $row){
  $text= ucfirst($row['Tables_in_project']);
  if ($text == "Error" || $text== "Messages"){
    continue;
  }
  echo  "<option value='$row[Tables_in_project]'> $text</option>";
}
```

To display the different tables in a dropdown menu, a query was first assigned to the variable, *$sql*. The database was then queried with *($db->query($sql) as $row)*. Then, the table names were echoed in a dropdown menu via for-loop.

Catching errors

```
} catch(Throwable $e){
    $error = "Insert failed";
    $s = "INSERT into error(error_text) values ('$error')";
    mysqli_query($db,$s);
    header("Location: ../insert.php");
}
```

If a query results in an error, the error is caught and an error message is inserted into the error table. The user is redirected to the initial page they came from, and the error message is subsequently displayed.

Queries

```
  $insert = "INSERT into car(model, availibility) values ('$car_model', '$car_aval')";
  mysqli_query($db, $insert);
```

```
$goodbye = $_POST['id_select'];
$delete = "DELETE FROM $chosen_table WHERE $chosen_keyword = $goodbye";
mysqli_query($db, $delete);
```

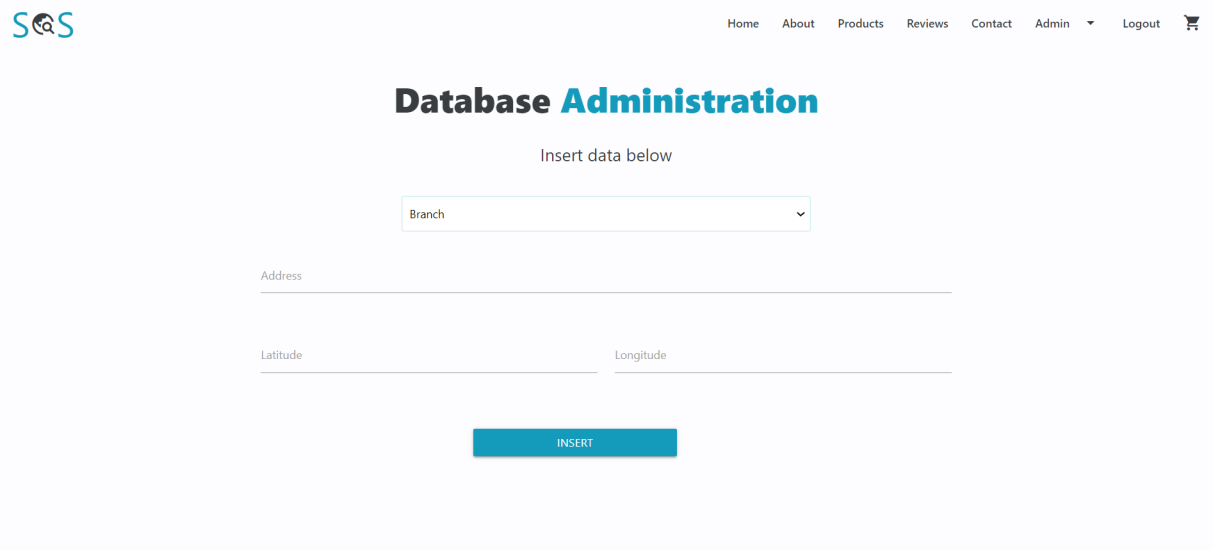
When querying the database, the query is first assigned to a variable. Then, using *mysqli_query,* the database attempts to carry out the query.

Displaying data as tables

```php
try{
    $selected_table = $_POST['tables'];
    $get_col = "SELECT `COLUMN_NAME` FROM `INFORMATION_SCHEMA`.`COLUMNS` WHERE `TABLE_SCHEMA`='project' AND `TABLE_NAME`='$selected_table'";
          echo "<div>
            <table class='responsive-table striped centered'>
                <thead>
                  <tr>";
                    foreach ($db->query($get_col) as $row){
                        echo "<th>" . $row['COLUMN_NAME'] . "</th>";
                    }
                  echo "</tr>
                </thead>";
                $sql = "SELECT * FROM $selected_table";
                echo "<tbody>";
                foreach ($db->query($sql) as $row){
                  echo "<tr>";
                  foreach ($db->query($get_col) as $crow){
                    echo "<td>" . $row[$crow['COLUMN_NAME']] . "</td>";
                  }
                  echo "</tr>";
                }
                echo "</tbody>
            </table></div>";
} catch(Throwable $e){
    echo "<p id='error'>Table could not be displayed</p>";
}
```

First, the database is queried to get the name of the columns from the selected database table. Then, in a for loop using (*$db -> query($get_col) as $row),* the columns are displayed as the head of each column for the printed table. Next, the database is queried to get the data in the tables. Then, in a double for loop, the data is displayed row by row in the printed table.

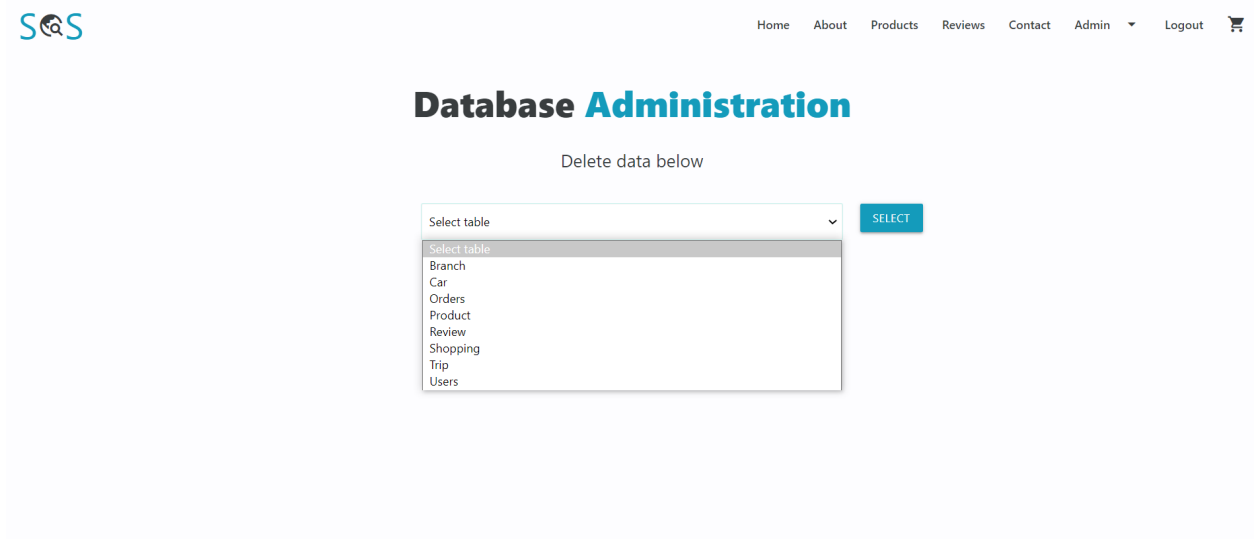
Insert



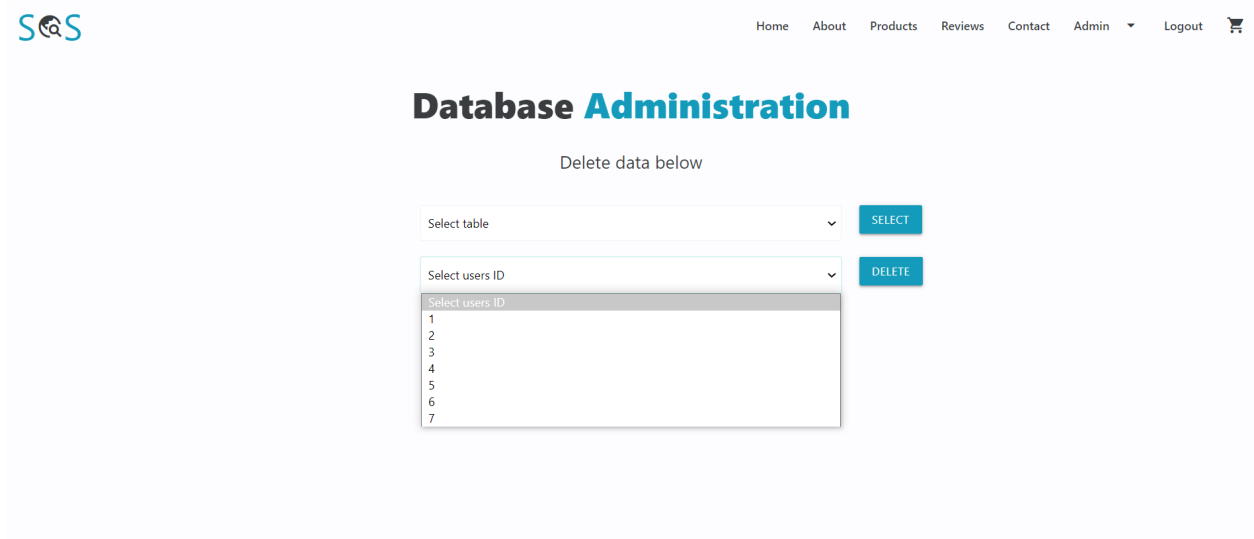
The user is prompted to select one of the tables in the database. Depending on which table the user selects, the form changes to adapt to what information is needed to insert data. The page performs basic checks to ensure that the type of data the user inputs

adheres to the type of data the table needs. If the insertion fails, an error message is displayed. Otherwise, a success message is displayed and the data is inserted into the database.

Delete

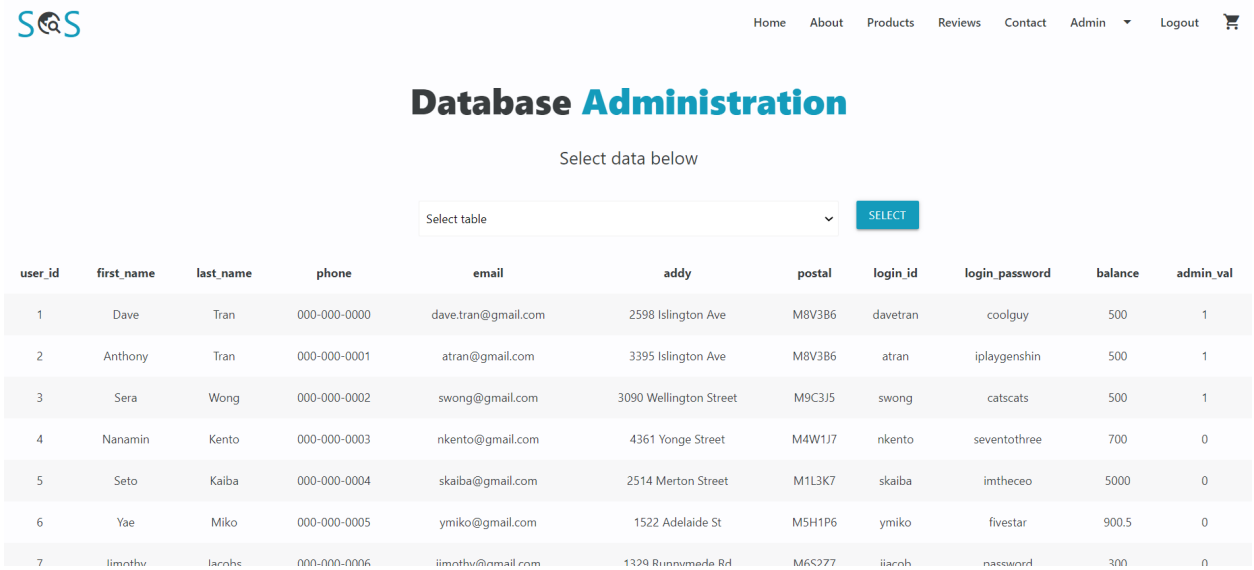


In regards to the delete function, the user is first prompted to select a branch from a dropdown menu.



Once selected, the page queries the database, and displays the IDs of the table in another dropdown menu. Once the user selects the ID of the specific entry they wish to delete, it is then deleted from the database.

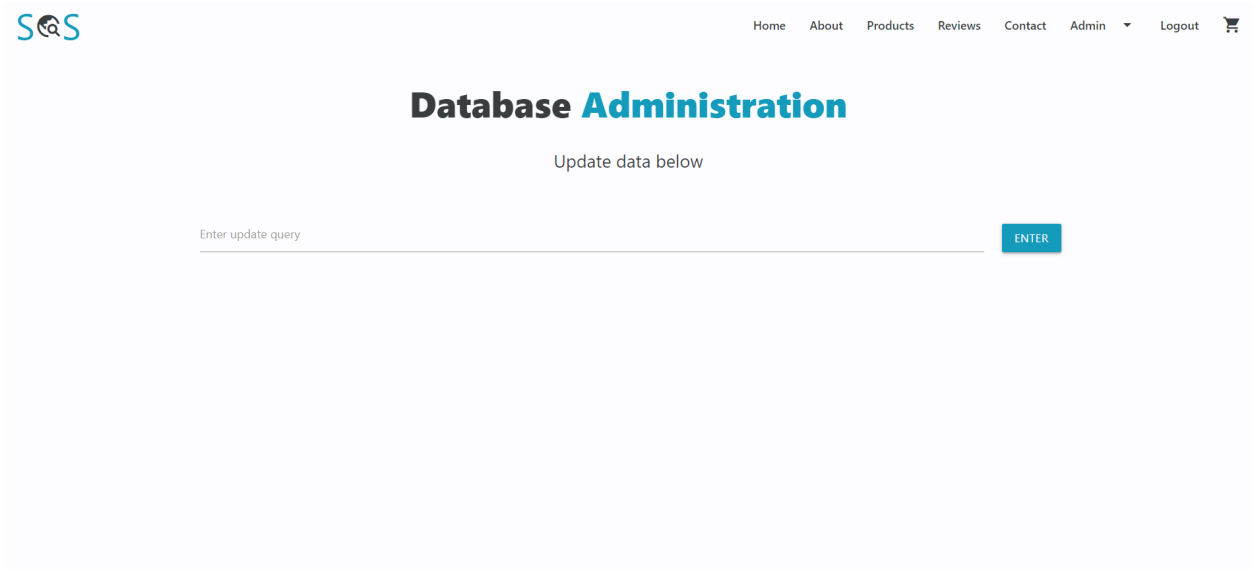
Select



user_id	first_name	last_name	phone	email	addy	postal	login_id	login_password	balance	admin_val
1	Dave	Tran	000-000-0000	dave.tran@gmail.com	2598 Islington Ave	M8V3B6	davetran	coolguy	500	1
2	Anthony	Tran	000-000-0001	atran@gmail.com	3395 Islington Ave	M8V3B6	atran	iplaygenshin	500	1
3	Sera	Wong	000-000-0002	swong@gmail.com	3090 Wellington Street	M9C3J5	swong	catscats	500	1
4	Nanamin	Kento	000-000-0003	nkento@gmail.com	4361 Yonge Street	M4W1J7	nkento	seventothree	700	0
5	Seto	Kaiba	000-000-0004	skaiba@gmail.com	2514 Merton Street	M1L3K7	skaiba	imtheceo	5000	0
6	Yae	Miko	000-000-0005	ymiko@gmail.com	1522 Adelaide St	M5H1P6	ymiko	fivestar	900.5	0
7	jimothy	jacobs	000-000-0006	jimothy@gmail.com	1329 Runnymede Rd	M6S2Z7	jjacob	password	300	0

When users wish to select and display data, they have the option to choose which branch they wish to display. Once their selection is made, the database is queried, and all data is displayed in a table format.

Update



In the update page, the user is able to enter a query that may modify existing data. If the update is successful, a message will inform the user; otherwise, an error message will appear.

Design and Layout of User Interface

General design details

The website was designed with a very minimalistic approach by making use of a limited color palette, as well as minimal amounts of elements on a single page. The color palette consisted of a dark grey color (#393D3F), a contrasting white (#FDFDFF), and a light blue (#149BBB) for highlights. Every page contains a navbar at the top of the page and a footer at the bottom with the respective contrasting background color based on the background of the rest of the page. Font Sizes for every page were restricted to 0.5-3 rem. The UI's of some of the pages on the site will change based on whether or not the user is logged in. Such as the Review page, Navbar, and Products Page. Users will not be able to access the products page until the user is logged in and they will be redirected to the login page if they try to access the products page without authentication. All pages are also responsive to the users' screen size.

Navbar - Not logged in

The navigation bar changes depending on whether the user is logged in as an admin, a user, or is not logged in. When the user is not logged in, they do not have access to the admin page or their shopping cart. The links at the top lead to different pages of the website.

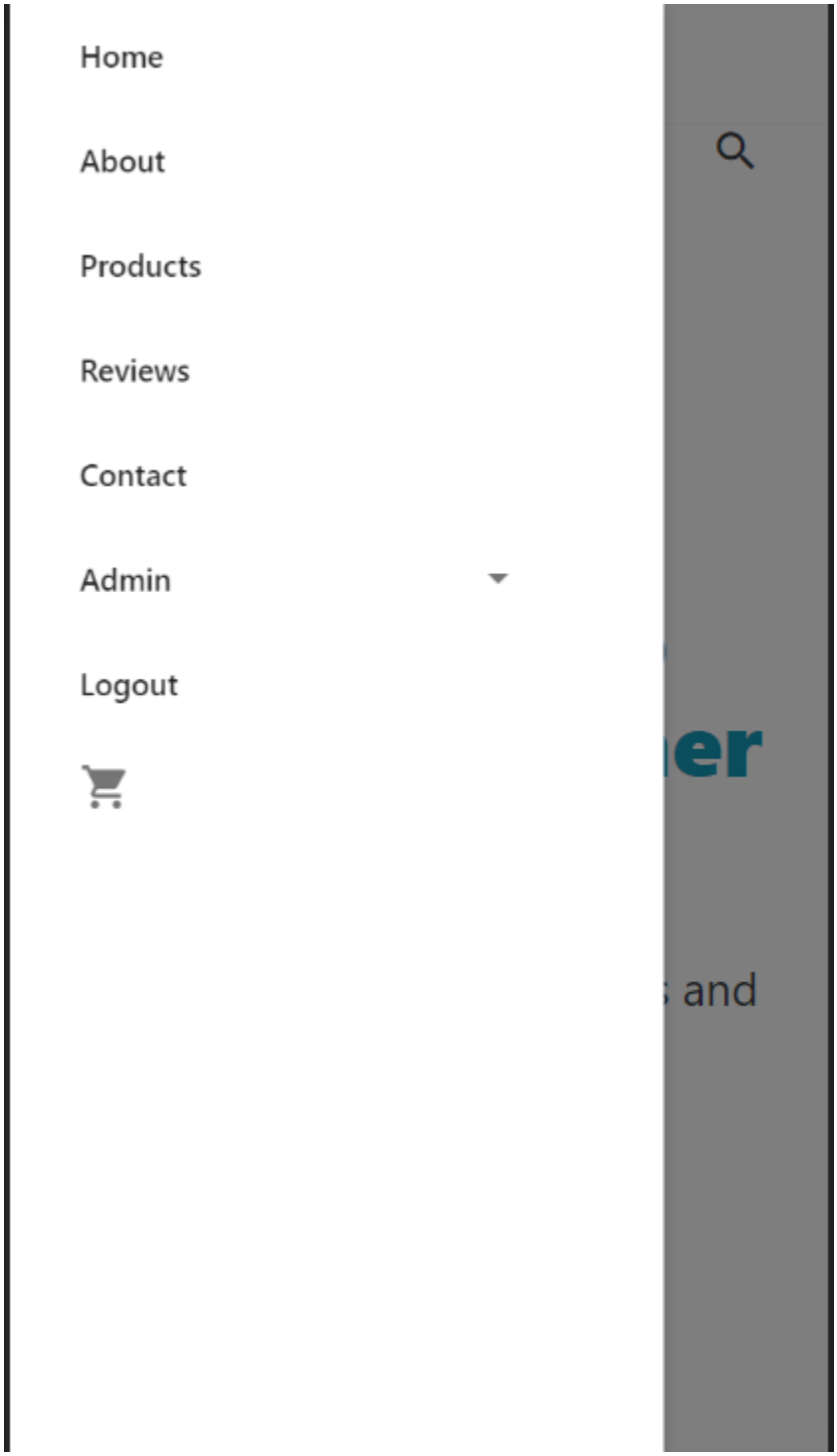
Navbar - User login

When the user is logged in, they have access to the shopping cart page.

Navbar - Admin login

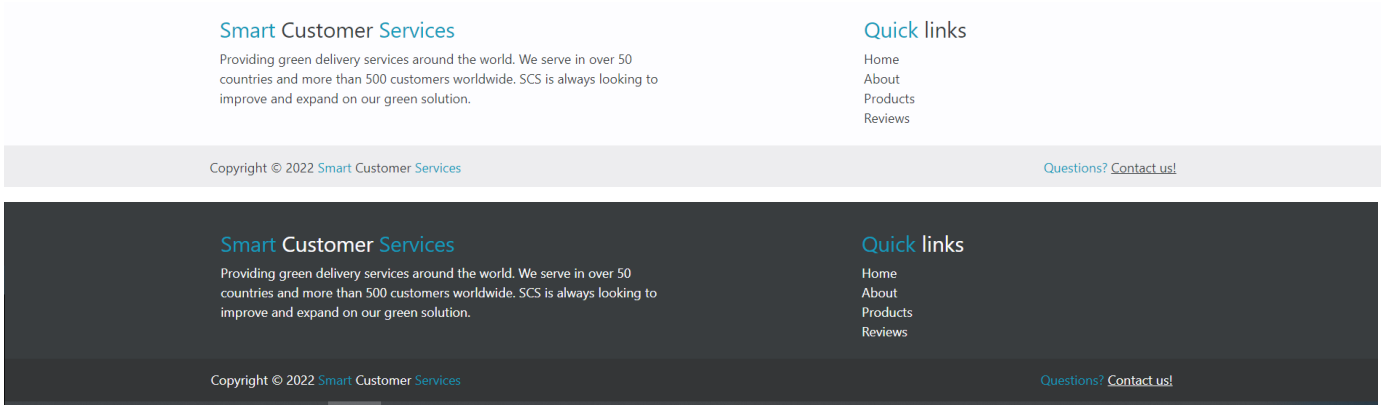
When the user is logged in as an admin, they have access to the shopping cart page, as well as the admin pages. Clicking on the admin tab reveals a dropdown menu in which the user is able to select different functionalities of the database maintain mode.

Navbar - Mobile



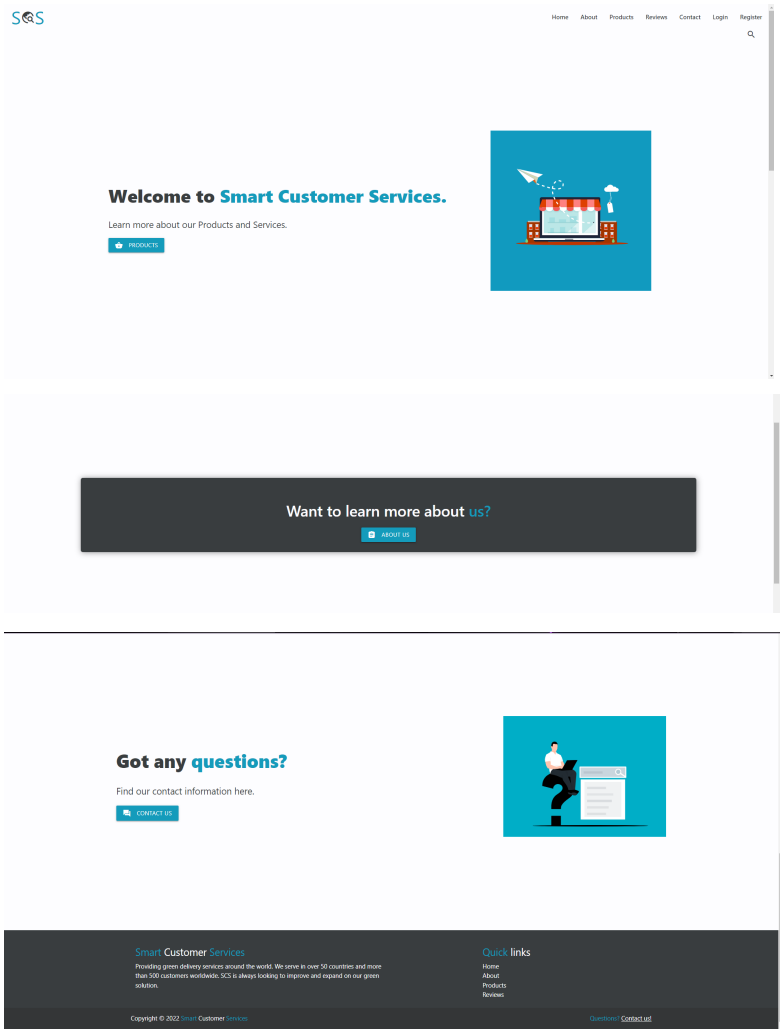
The same functionalities apply to the mobile version of the navbar. Users are able to tap a hamburger icon located at the top left. The navbar then slides out from the left.

Footer



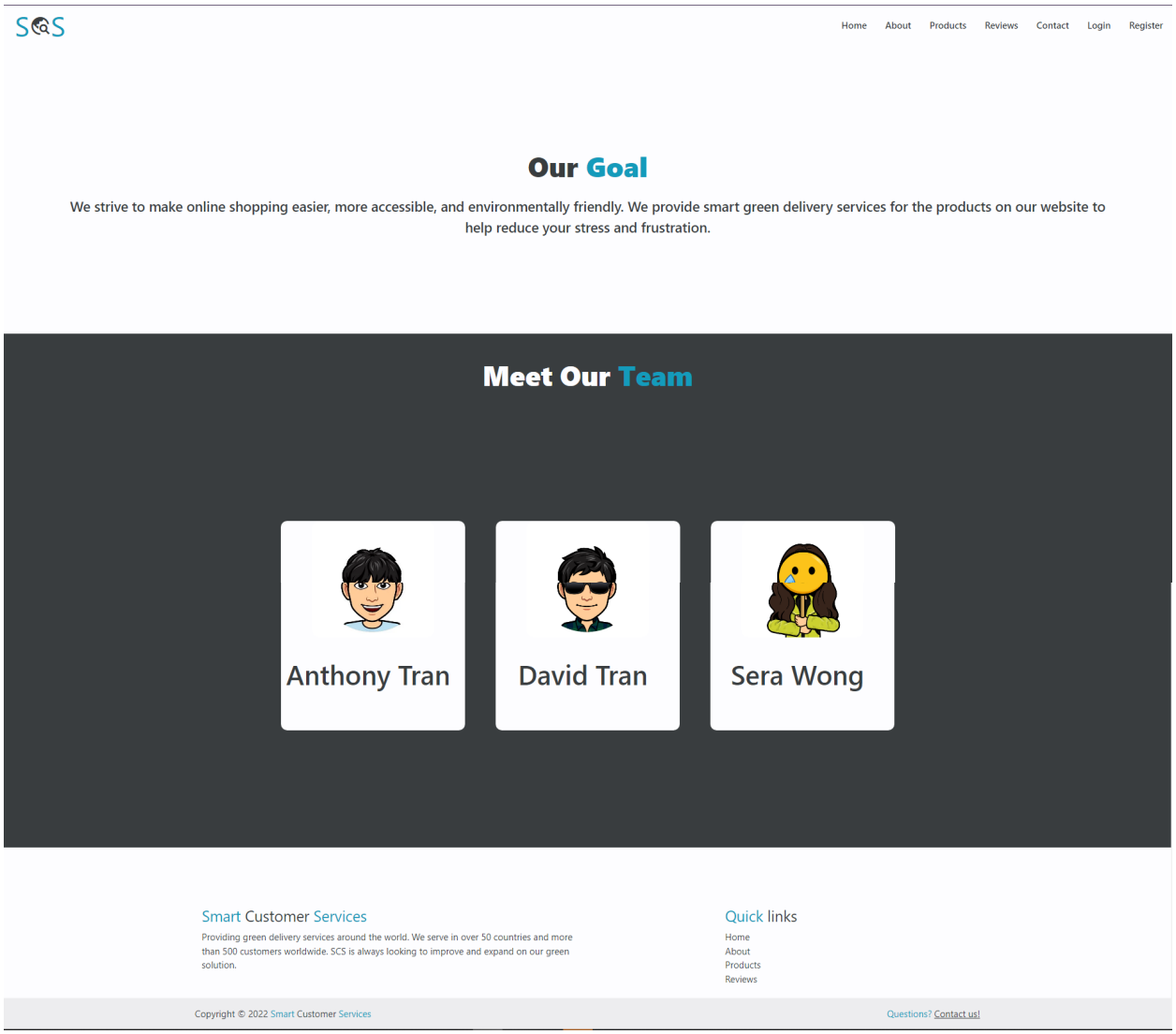
Depending on the page colour scheme, there are two versions of the footer: grey and white. Both provide the same functionality. Under “Smart Customer Services”, there is a short blurb describing the goal of the organization. Clicking on any of the links under “Quick links” will take the user to that respective page. The “contact us” located at the bottom right will take the user to the contact page.

Homepage



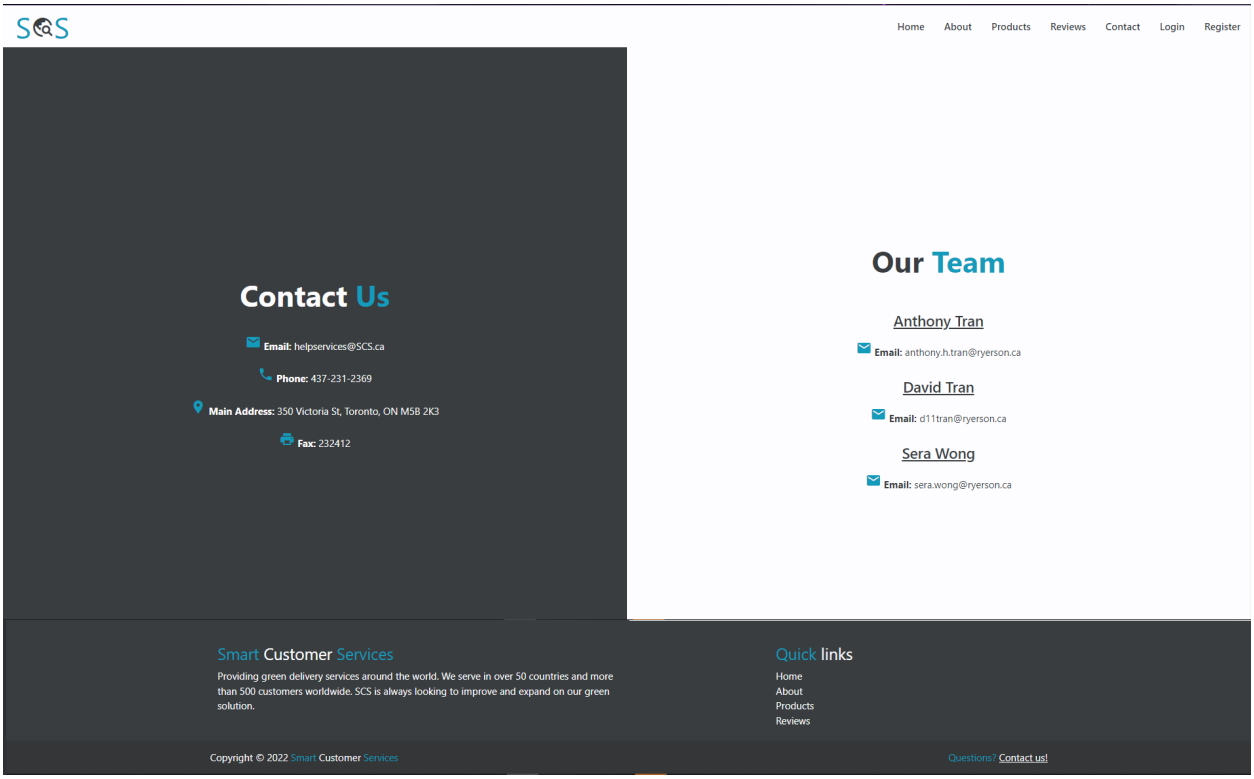
The Homepage doesn't have any functionality other than the buttons that redirect to other pages on the site. It contains a couple of basic images and containers and just serves as a starting point for users.

About Page



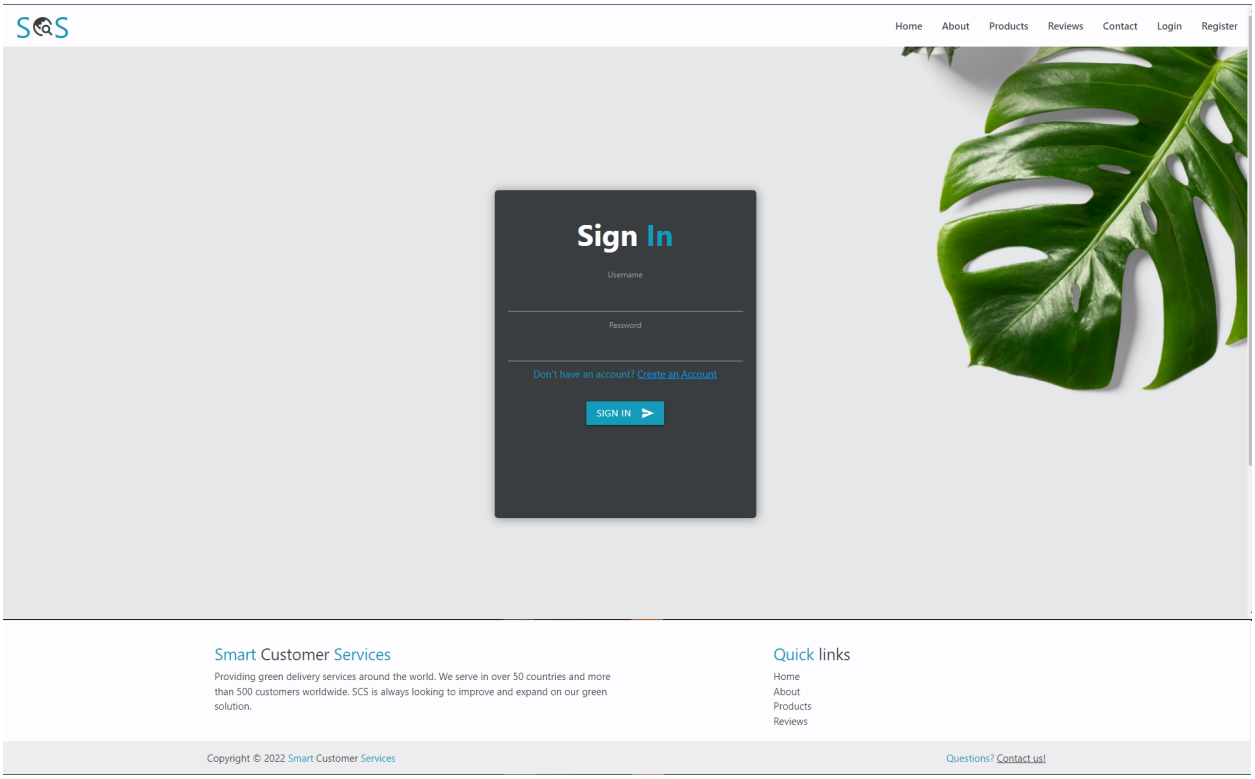
The footer on the about page is white to contrast the dark grey background above it. It provides a short description of the company goal and image introductions of the team behind the website.

Contact Us



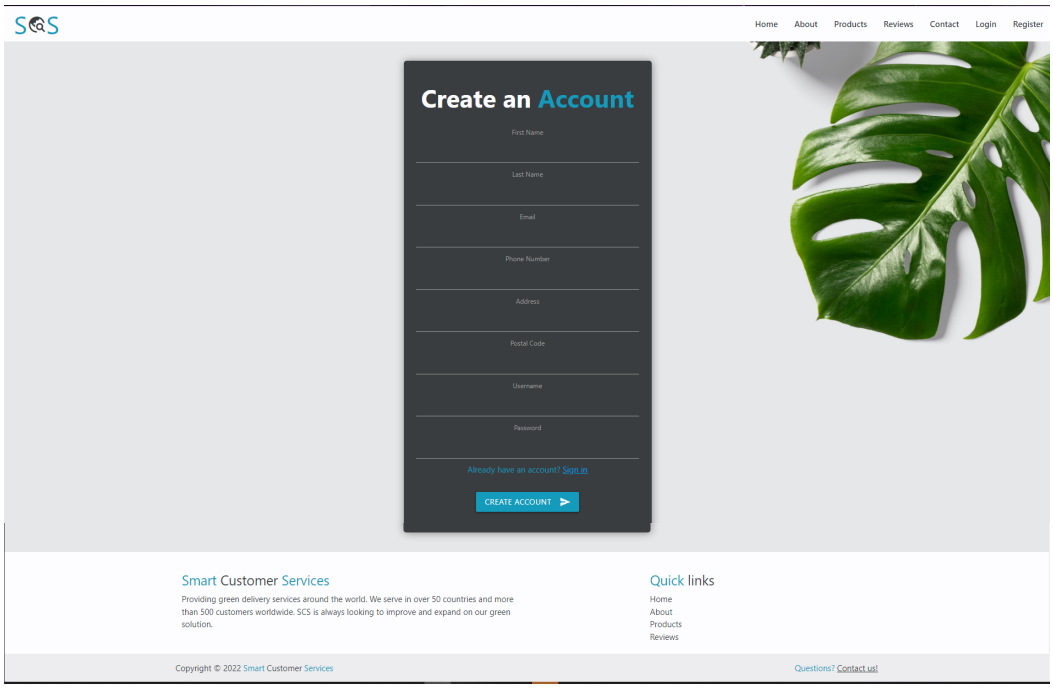
The contact us page is divided into 2 different sections. The left section contains general contact information, while the right section contains contact information for the team members.

Login Page



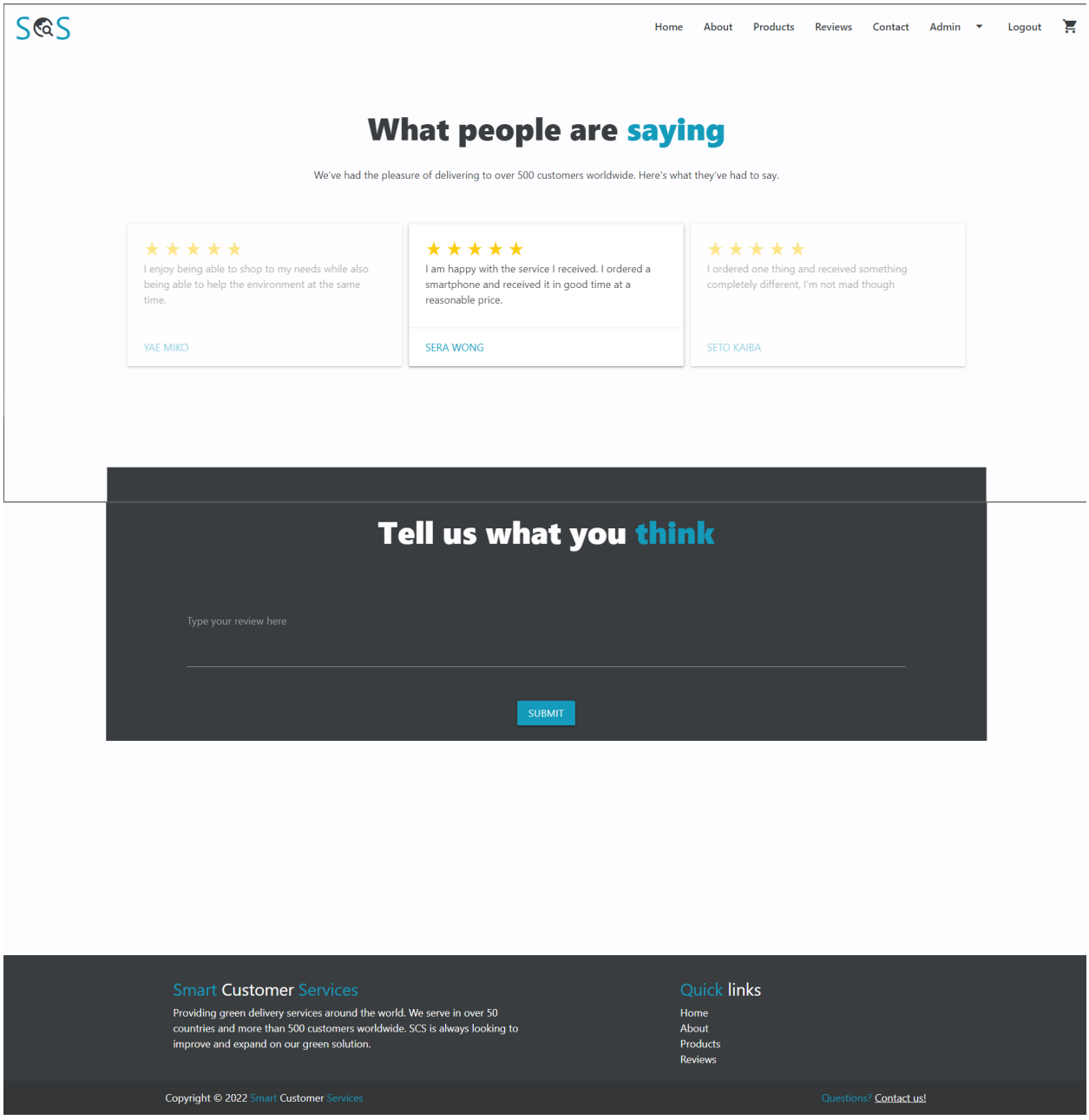
The Login Page has a dark grey container on top of a white background with a plant on the top-right corner. Users will be able to enter their login information, and if they do not have an account, there is a link for them to click that redirects them to the Register page. Once the user Logs in, they get redirected to the homepage, and the navbar updates accordingly.

Register Page



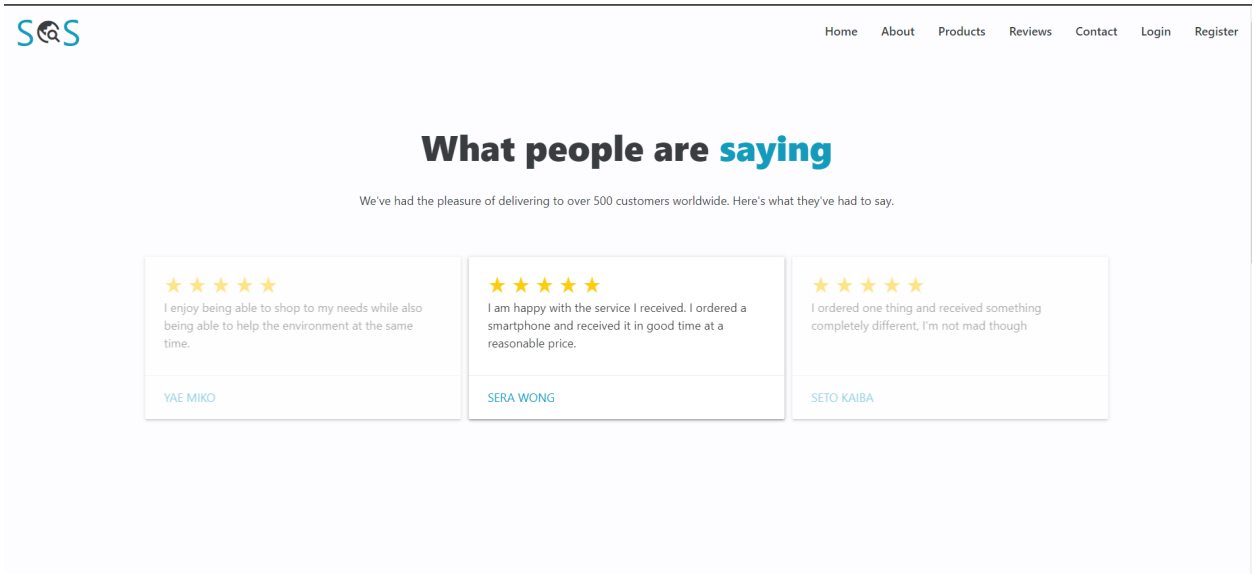
The Register page is similar to the login page. It contains more input boxes in order to create an account, and if the user already has an account, they can click the link to redirect to the login page. Once the user hits the create account button, they are automatically logged into their account and redirected to the home page. The navbar is also updated accordingly.

Review page - logged in



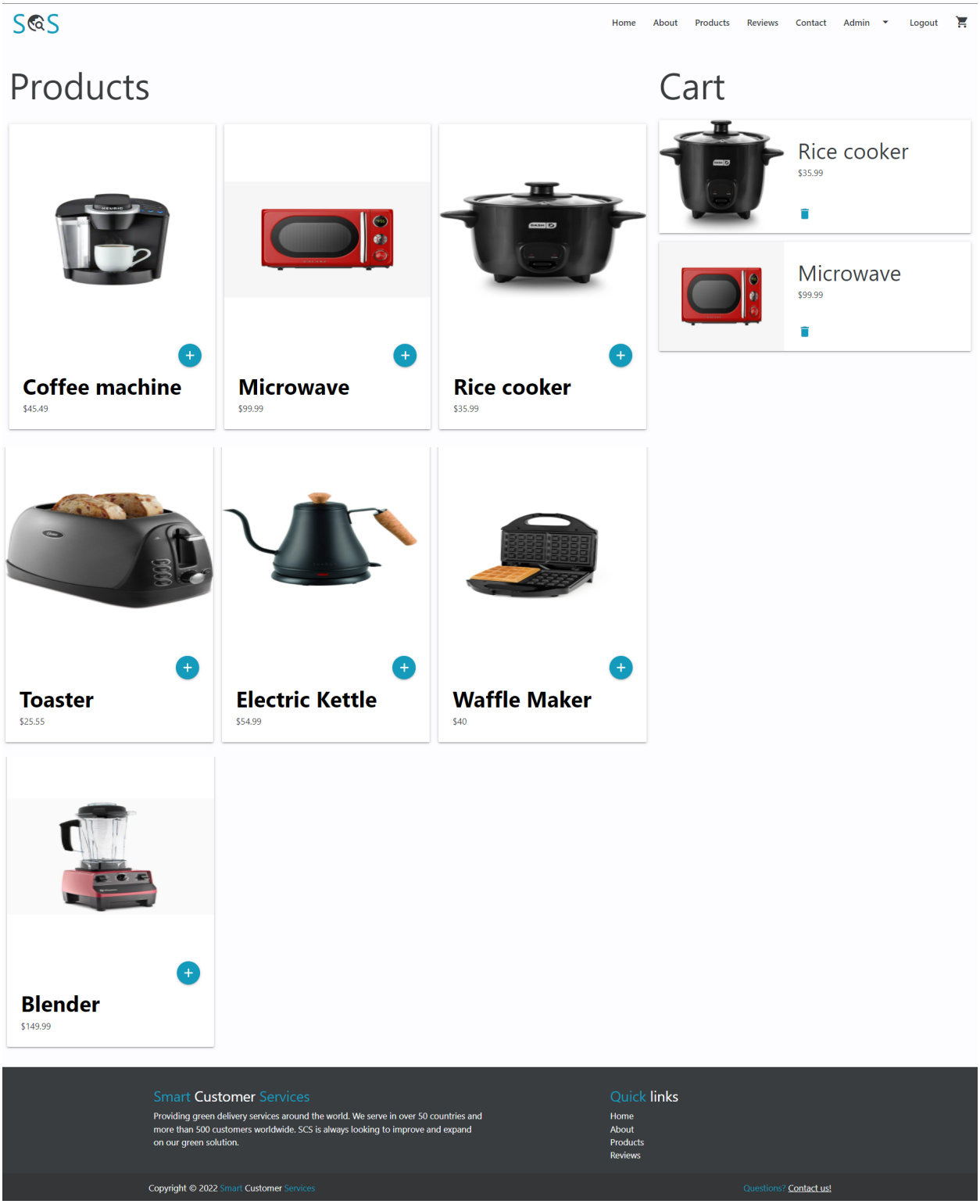
The review page changes based on if the user is logged in or out. The carousel at the top showcases several reviews drawn from the database. If the user is logged in, they see an option to submit a review, which is then inserted accordingly in the database.

Review page - logged out



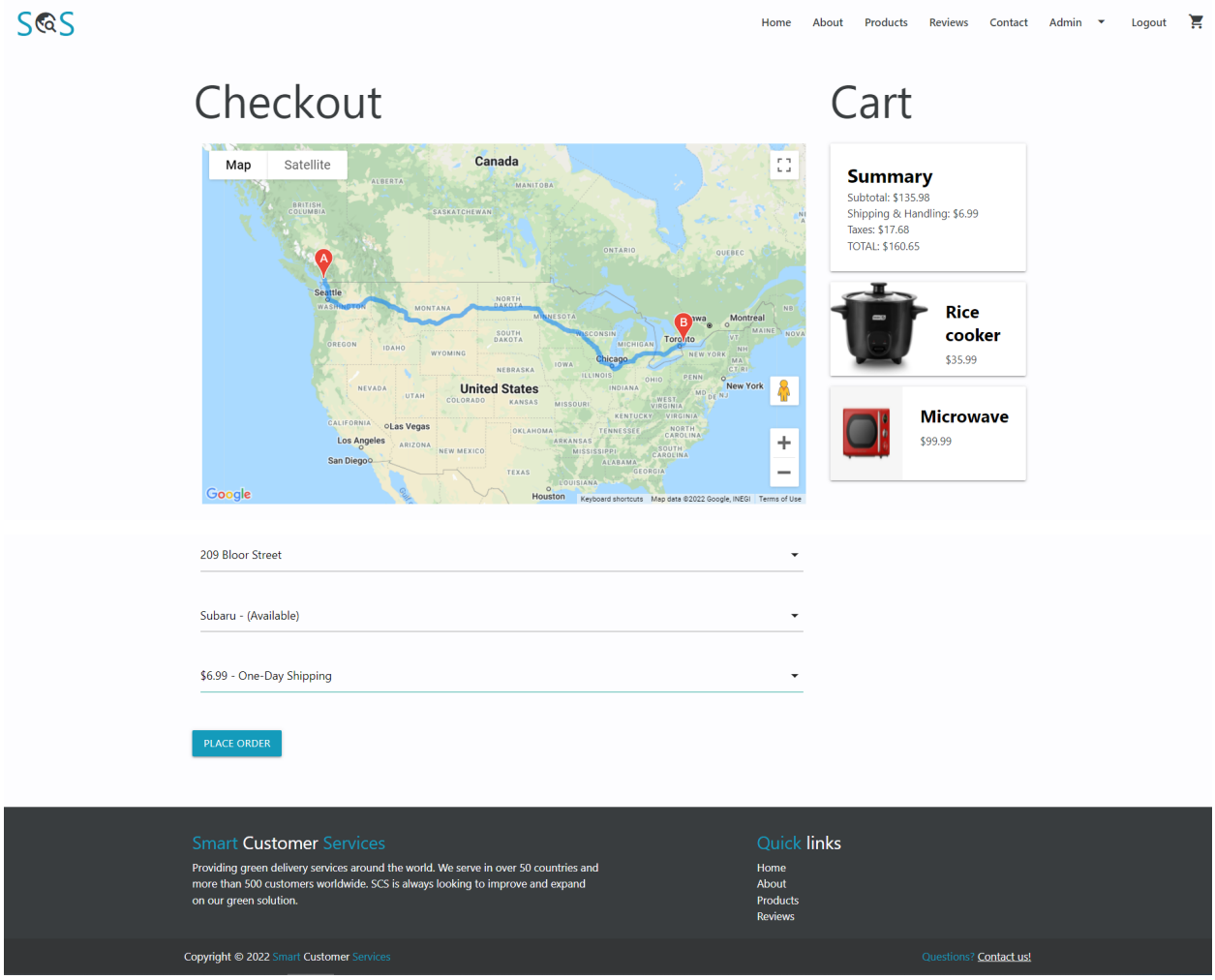
When the user is logged out, they do not have the ability to submit a review. They are able to view the carousel.

Products


SCS
Home
About
Products
Reviews
Contact
Admin
Logout
Products
Cart
Coffee machine
$45.49
Microwave
$99.99
Rice cooker
$35.99
Toaster
$25.55
Electric Kettle
$54.99
Waffle Maker
$40
Blender
$149.99
Rice cooker
$35.99
Microwave
$99.99
Smart Customer Services
Providing green delivery services around the world. We serve in over 50 countries and more than 500 customers worldwide. SCS is always looking to improve and expand on our green solution.
Quick links
Home
About
Products
Reviews
Copyright © 2022 Smart Customer Services
Questions? Contact us!

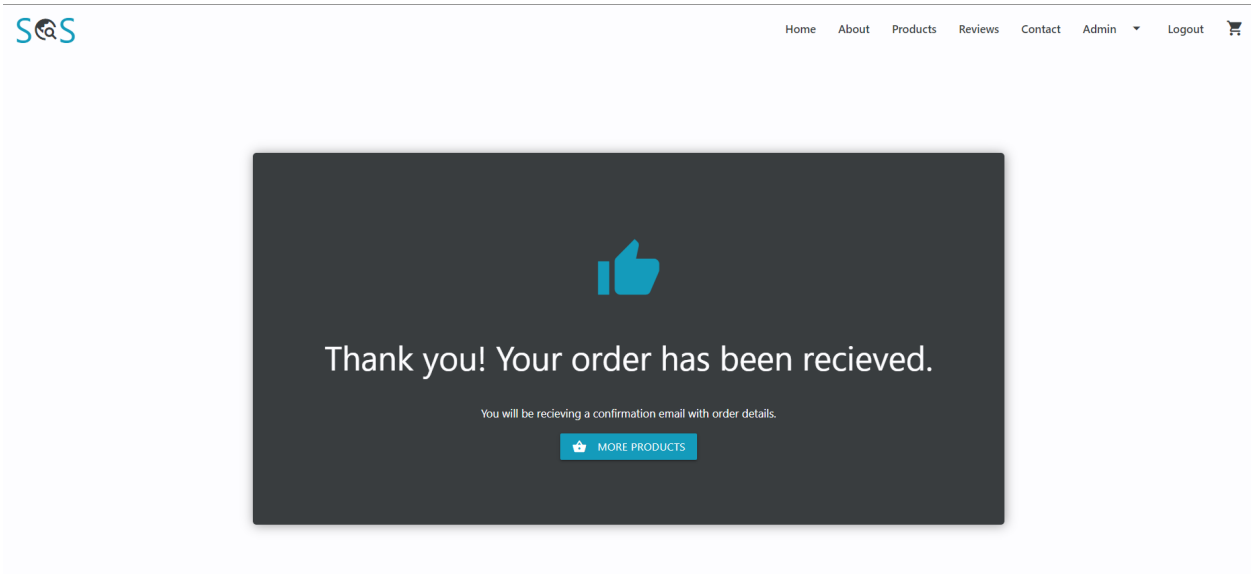
On the products page, users are able to drag and drop products into the shopping cart, located on the left. Users also have the option of clicking the plus button on each product to add the respective item to their shopping cart. Users are able to delete items from their cart by clicking on the trash can icon, located underneath the picture of the product. If users navigate to a different page and return, the items in their shopping cart will still be recorded.

Checkout



Here, users are able to see the items in their cart. The user is prompted to select a branch, from a dropdown menu of branch addresses. Once selected, the map provides a path from the user's address to the branch of their choosing. The user is also prompted to select a car for delivery. The dropdown list provides the cars that are available, drawing information from the database. Finally, the user is prompted to select their shipping option. They are able to choose from free, one day, or two day shipping. The total cost of their items is showcased on the right, under "Summary". Users are able to see a picture of the items they selected.

Confirmation



After the user selects “place order” on the checkout page, the user is directed to a confirmation page where they are informed that their order has been successfully received. The order is then added accordingly to the database.

Database Design and Structure

All tables can be referenced by their id variable.

Orders

Variable	DataType
order_id	PK INT
date_issued	DATE
date_completed	DATE
order_price	FLOAT
payment_code	INT
user_id	INT REFERENCES Users(user_id)
trip_id	INT REFERENCES Trip(trip_id)
receipt_id	INT REFERENCES Shopping(receipt_id)
branch_id	INT REFERENCES Branch(branch_id)

Used to store data regarding customer orders. Also links the order with a user, receipt, and branch.

Users

Variable	DataType
user_id	PK INT
first_name	VARCHAR
last_name	VARCHAR
phone	VARCHAR
email	VARCHAR
addy	VARCHAR
postal	VARCHAR
login_id	VARCHAR
login_password	VARCHAR

balance	FLOAT
admin_val	BOOLEAN

Stores info about users that register with the site.

Review

Variable	DataType
review_id	PK INT
review_text	VARCHAR
user_id	INT REFERENCES Users(user_id)

Stores reviews that may be showcased on the review page.

Branch

Variable	DataType
branch_id	PK INT
branch_addy	VARCHAR
lat	FLOAT
lon	FLOAT

Stores branch information, mainly pertaining to the address of the branch.

Shopping

Variable	DataType
receipt_id	PK INT
shopping_price	FLOAT
branch_id	INT REFERENCES Branch(branch_id)

Stores receipts for users after they place orders.

Product

Variable	DataType
prod_id	PK INT
prod_name	VARCHAR

prod_price	FLOAT
img_url	VARCHAR

Stores product information.

Car

Variable	DataType
car_id	model
model	VARCHAR
availability	VARCHAR

Stores the types of cars for trips, and their availability.

Trip

Variable	DataType
trip_id	PK INT
destination_code	INT
trip_price	FLOAT
distance	FLOAT
branch_id	INT REFERENCES Branch(branch_id)
car_id	INT REFERENCES Car(car_id)

Stores information about the trip and delivery when orders are placed.

Error

Variable	DataType
error_text	PK VARCHAR

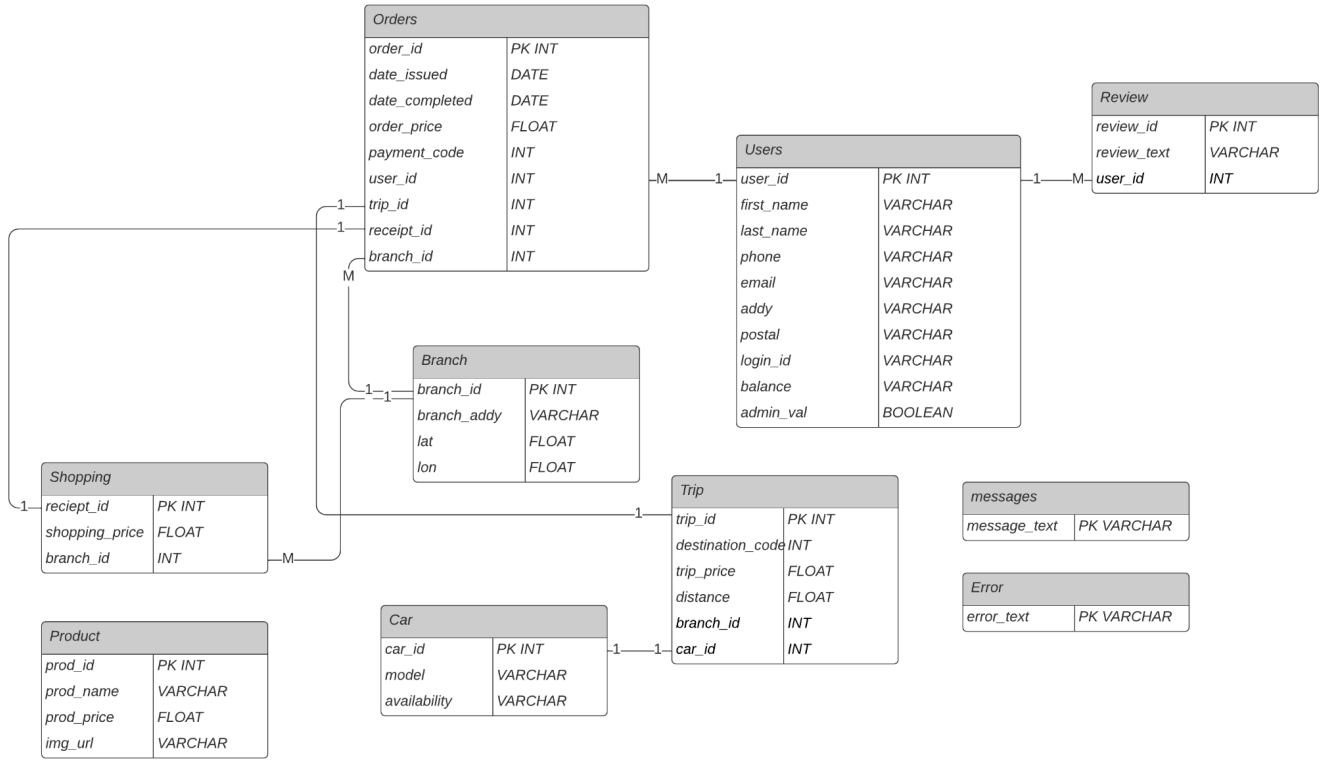
Stores error messages.

Messages

Variable	DataType
message_text	PK VARCHAR

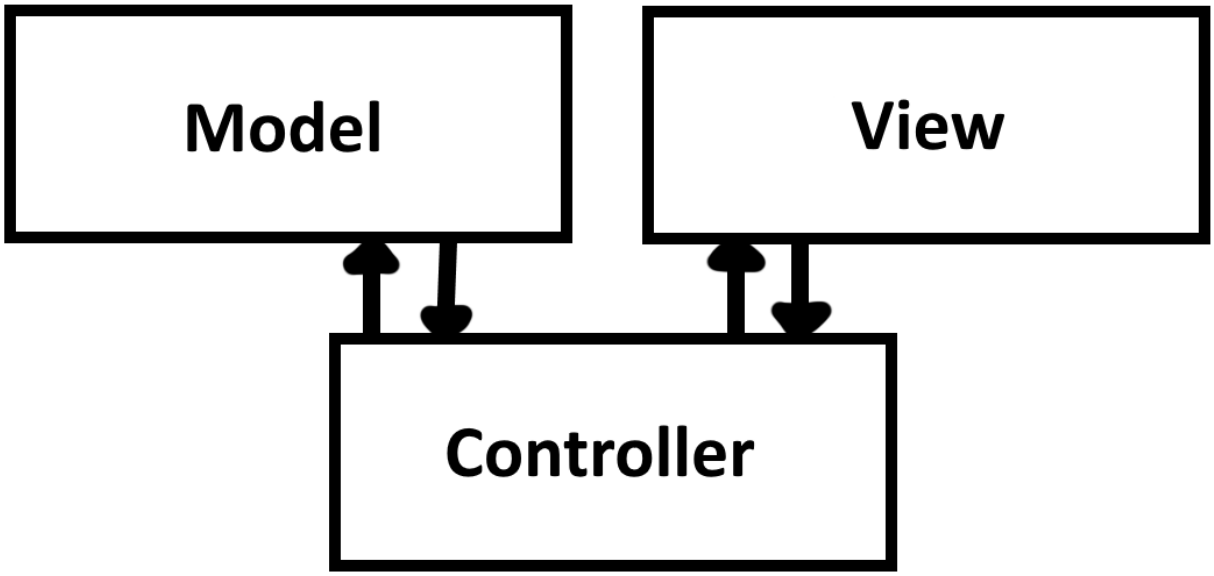
Stores page messages.

Database Schema Diagram



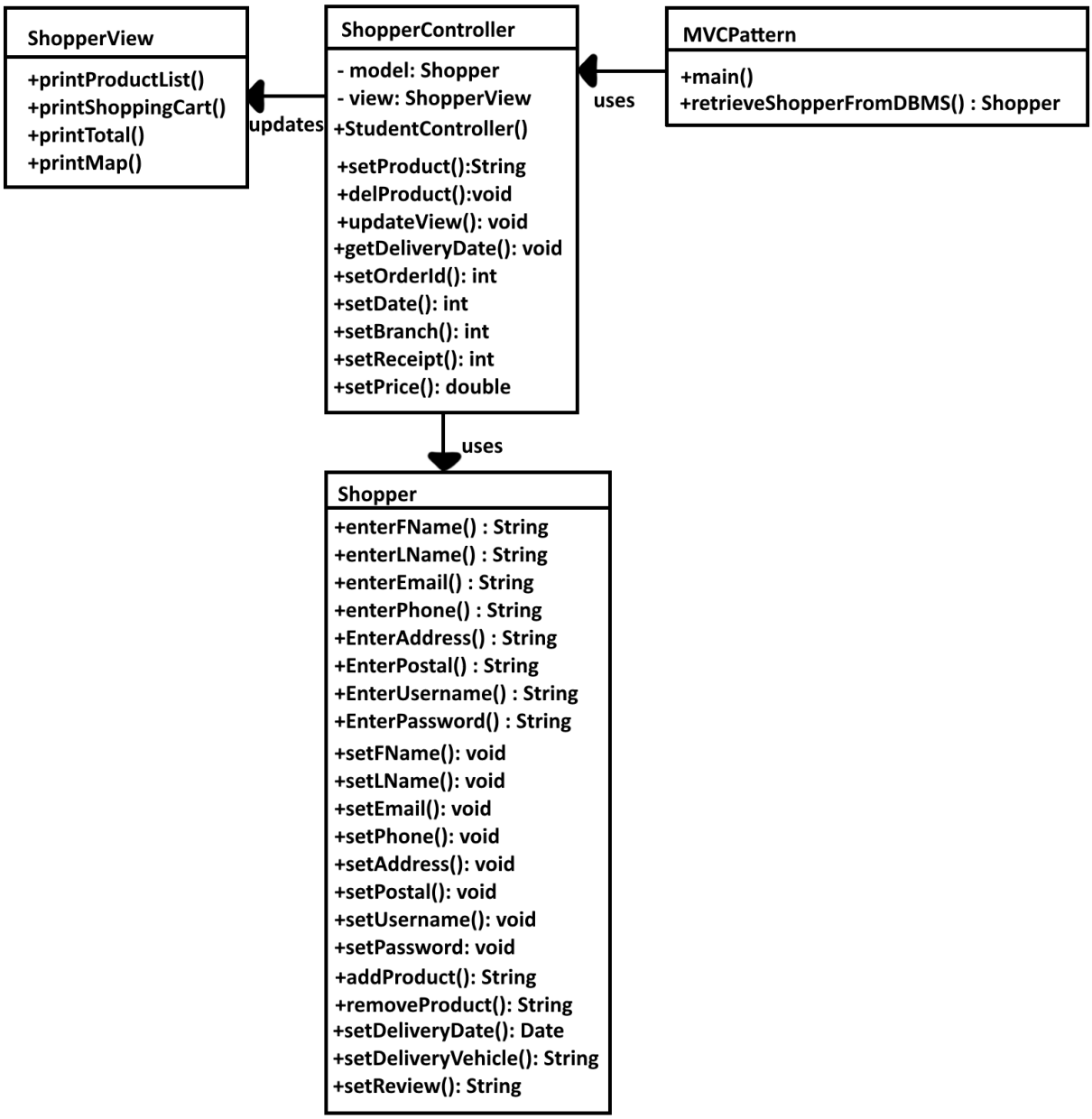
MVC Architecture

We designed our web application strictly following the MVC architecture design pattern guidelines. In order to accomplish this feat, we separated the model, view, and controller from each other in order to separate the functionality from the design of our shop.



We broke down the three components so that when the user interacts with the view of the page, they are actually commanding the controller to dynamically edit the view in real-time as well as manipulate, set, and use the model's data. This is done in such a way that the controller is actively mediating between both the model and the view in order to independently isolate each aspect of our application's architecture.

The architecture of our web application looks like the following:



One can easily see how we implemented the MVC architecture in order to produce a clean design. When the user first registers to the website, they enter in various details about themselves like their name, address, and phone number. The controller then takes this information and sets it in our database because it is the model, the data of our design. When the user decides to go to the page where all our products are displayed, the controller loops through our model database and uses the data in order to update and display the products, the user's shopping cart, and the delivery map. The user can interact with the view to drag and drop the products they want into their shopping cart, and as a result, the controller will update both the model and the view, displaying the user their newly updated cart along with the total cost of their purchase.