

**CPS630 Iterations III & IV Report:**  
**Smart Customer Services**  
**Team 30**  
**Anthony Tran, David Tran, Sera Wong**  
**500981445, 500890757, 500702293**

<b>Members</b>	<b>Percentage</b>
Anthony Tran	33%
David Tran	33%
Sera Wong	33%

## **Table of Contents**

<b>Part B-2: New Service</b>	<b>4</b>
<b>Part B-3: Shopping Cart</b>	<b>5</b>
<b>Part B-4: Review/Ranking</b>	<b>10</b>
<b>Part B-5: Browsers</b>	<b>11</b>
<b>Part C: SPA</b>	<b>13</b>
Webpages	13
Elements	16
SPA implementation in React	17
SPA Architecture	17
<b>Iter IV:</b>	<b>18</b>
Login and Register Validation	18
Credit Card and Payment	19

## Part B-2: New Service



**15% Off Sale Storewide!**

Our team decided to make our special service be a limited time storewide sale. When the customer is at the checkout, they will automatically have a 15% discount coupon applied to their entire shopping cart, reducing the amount that they would have to pay. It would make for an incredibly alluring experience, and would not only make a strong impact on the customer, but also increase their likelihood to return and shop with us again in the future. It dynamically displays the total savings for the customer at the top right, under the “Savings” heading. It is automatically used when the customer adds products to their shopping cart, and then the discount is displayed to them when they proceed to checkout.

### **Summary**

Subtotal: \$107.03

Shipping &

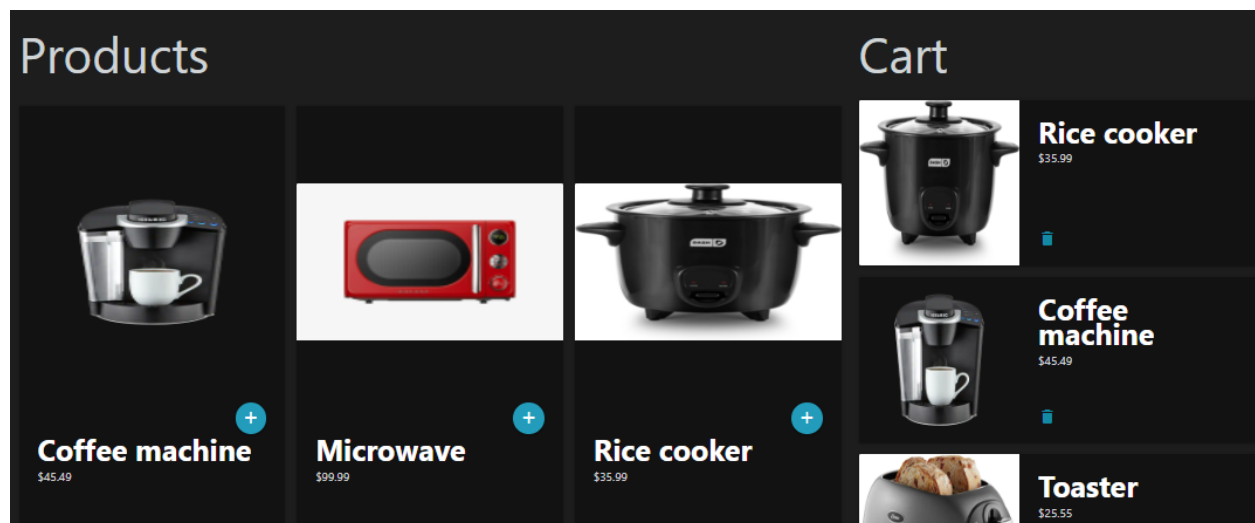
Handling: \$6.99

Taxes: \$13.91

Savings: -\$16.05

**TOTAL: \$111.88**

## Part B-3: Shopping Cart



While the user is browsing through our product catalog, they can add selected items to their cart by using a button that utilizes React's Effect hook to save it to their local storage temporarily.

```
function cartBtn(id) {  
  setCart((oldCart) => [...oldCart, id]);  
}  
  
function removeCart(id) {  
  let str = id.target.id;  
  let removeTarget = str.substring(str.length - 1);  
  const newCart = cart.filter((ele, i) => i !== removeTarget - 1);  
  setCart(newCart);  
}
```

```
useEffect(() => {  
  localStorage.setItem(LOCAL_STORAGE_KEY, JSON.stringify(cart));  
}, [cart]);
```

The user can then proceed to the checkout area where it will access the local storage and retrieve all of the various items that the user wants to purchase.

```
useEffect(() => {  
  generateProducts();  
  const storedCart = JSON.parse(localStorage.getItem(LOCAL_STORAGE_KEY));  
  if (storedCart) setCart(storedCart);  
}, []);
```

There, it will use the array it retrieved from the local storage, use axios to retrieve the various details on each item from the database.

```
function generateProducts() {
  axios({
    method: "post",
    url: "http://localhost/get_checkout.php",
    headers: { "content-type": "application/json" },
    data: { user: localStorage.getItem("username") },
  })
  .then((res) => {
    //console.log(res.data);
    setProduct(res.data.product);
    setOrigin(res.data.address);
    setBranch(res.data.branch);
    setCar(res.data.car);
    setLoading(false);
  })
  .catch((err) => {
    console.log(err);
  });
}
```

It will then map through the array, and render the details and image of each product that the user wants to buy.

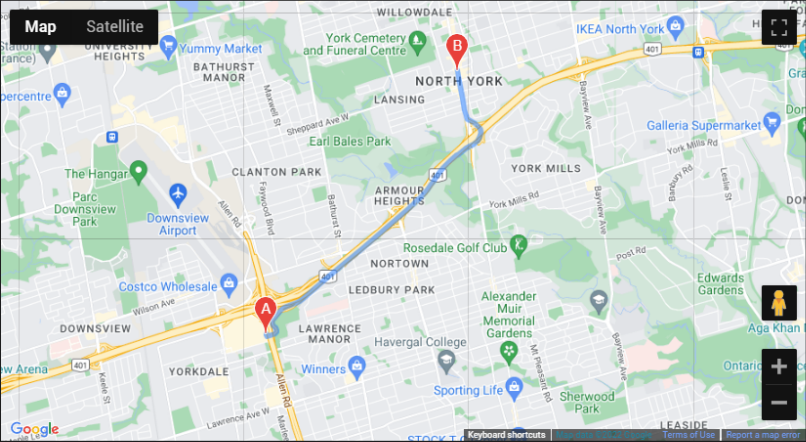
```
{cart.map((item) => {
  cartCount++;
  const name = product[item - 1]["prod_name"];
  const price = product[item - 1]["prod_price"];
  const imgurl = product[item - 1]["img_url"];

  const img = "http://localhost/CPS630/scs/src/" + imgurl;

  return (
    <div class="card horizontal small" id={"card" + cartCount}>
      <div class="card-image card-img">
        <img src={img} />
      </div>
      <div class="card-stacked">
        <div class="card-content">
          <span
            class="cardspan card-title"
            style={{ color: "black", fontWeight: "bold" }}>
            {name}
          </span>{" "}
          <p>${price}</p>
        </div>
      </div>
    </div>
  );
})}
```

Once the page and the user's shopping cart is loaded, the user is able to choose the various branch, car, and shipping method they want to utilize. The map responsively reacts to the user's choice of branch; depending on the branch they select, they will be shown the route taken in real-time using Google Map's Direction API. Similarly, depending on the shipping method chosen, the price of the user's cart will immediately change to reflect what works best for them. They are then prompted to enter in their credit card information to proceed and to place their order.

## Checkout



4841 Yonge Street

Bugatti - Available

\$6.99 - One-Day Shipping

Name on Card

Jimothy Davino

Card Number

4231-4312-6745-9123

Expiry Date

05/25


CVV

554


PLACE ORDER

## Cart


**Summary**  
Subtotal: \$107.03  
Shipping & Handling: \$6.99  
Taxes: \$13.91  
Savings: -\$16.05  
TOTAL: \$111.88



**Rice cooker**  
\$35.99

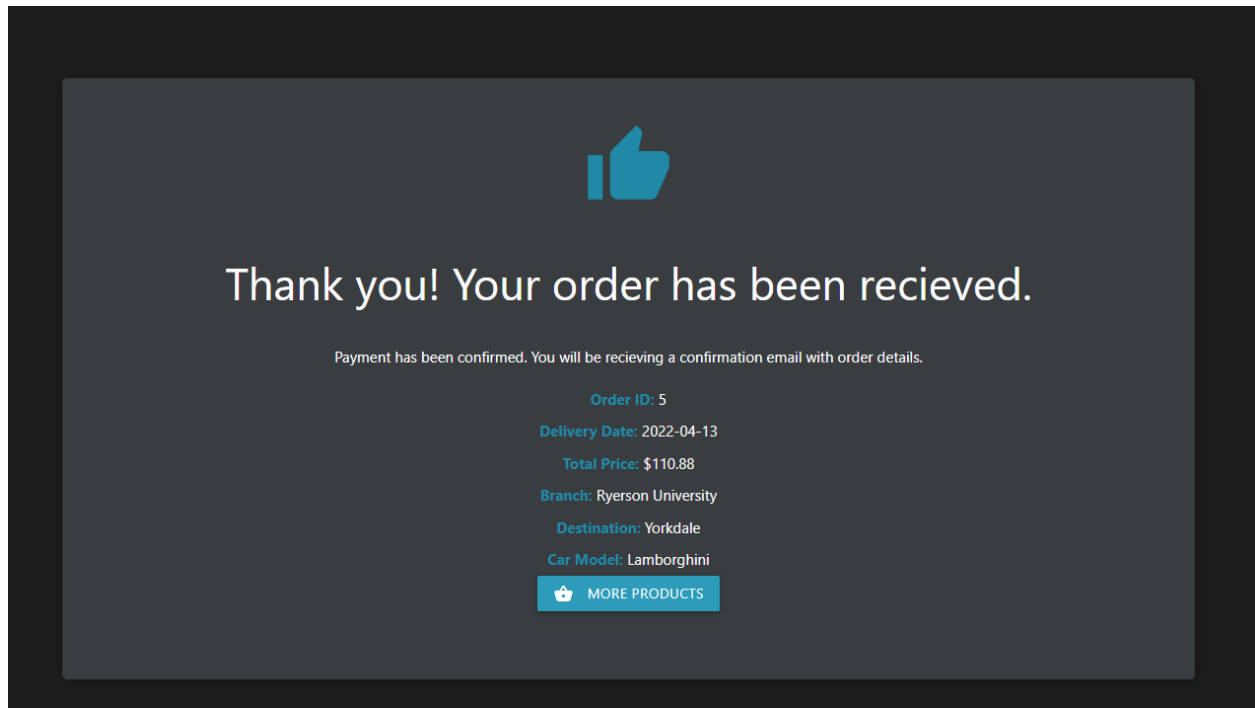


**Coffee machine**  
\$45.49



**Toaster**  
\$25.55

Once the user places their order, they are then redirected to an invoice page displaying the details of their order that they've selected at checkout. It retrieves all this data by similarly once again using the axios function, and then shows various details such as the order number, delivery date, price, branch, delivery location, and what car would be used to deliver their product. All of this data is held in a simple rectangle, where the user is given the option to continue browsing through more of our products.



## Part B-4: Review/Ranking

SQS

[Home](#)[About](#)[Products](#)[Reviews](#)[Contact](#)[Admin](#)[Logout](#)

### What people are saying

We've had the pleasure of delivering to over 500 customers worldwide. Here's what they've had to say.

★★★★★

I enjoy being able to shop to my needs while also being able to help the environment at the same time.

YAE MIKO

★★★★★

I am happy with the service I received. I ordered a smartphone and received it in good time at a reasonable price.

SERA WONG

★★★★★

I ordered one thing and received something completely different, I'm not mad though

SETO KAIBA

### Tell us what you think

Product

Coffee machine

Rating

1

Type your review here

SUBMIT

### View product reviews

Blender

★★★★★

I enjoyed the blender

DAVE TRAN

★★★★★

Fast delivery and works well

ANTHONY TRAN

★★★★★

Blends well I like it

SERA WONG

On the review page, users are able to view a set of general reviews culminating from the Reviews table in the database. If a user is logged into their account,



they are able to submit a review for a product they've purchased in the past, which consists of a block of text and a rating.

```
useEffect(() => {
  axios({
    method: 'post',
    url: 'http://localhost/submit_review.php',
    headers: { 'content-type': 'application/json' },
    data: submitReview
  })
  .catch(err =>{
    console.log(err);
  })
}, [submitReview])
```

Using Axios, JSON data is sent to the appropriate php file via POST.

```
$review_insert = "INSERT into review_products(preview_text, preview_rate, prod_id, user_id) values ('$text', '$rate', '$prod_id', '$user_id')";
mysqli_query($db, $review_insert);
```

The corresponding php file handles the data, and inserts it into the database under the review\_products table. The table consists of five columns for the review ID, review text, rating, product ID, and user ID. Once entered into the corresponding columns, the review is registered in the database and can be viewed.

```
useEffect(() => {
  if(chosenReview !== ""){
    axios({
      method: 'post',
      url: 'http://localhost/get_product_review.php',
      headers: { 'content-type': 'application/json' },
      data: chosenReview
    })
    .then(res =>{
      setGetProductReview(res.data)
    }).catch(err =>{
      console.log(err);
    })
  }
}, [chosenReview])
```

To view product reviews, users are able to select a product from the dropdown menu. Upon selection, the page makes a GET request using Axios and receives the JSON data, grabbed from the database.

```

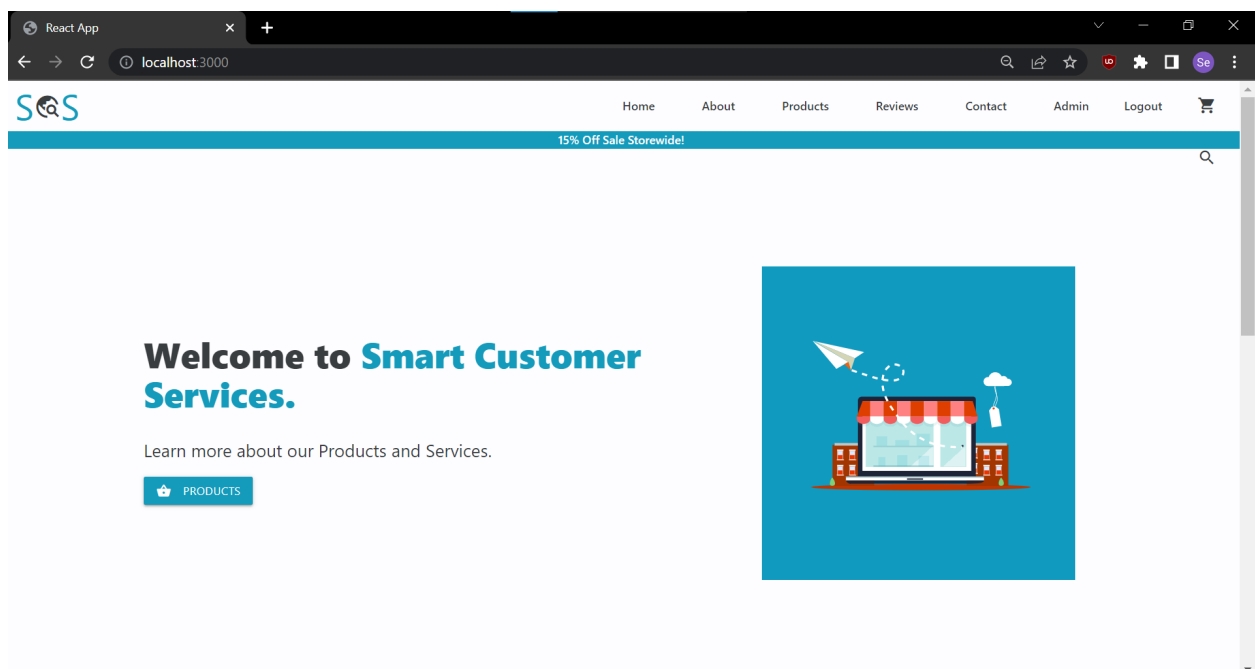
$sql= mysqli_query($db, "SELECT preview_text, preview_rate, first_name, last_name FROM users u, review_products r WHERE prod_id = '$prod_id' AND u.user_id = r.user_id");
$num = mysqli_num_rows($sql);
if($num > 0){
    while($r = mysqli_fetch_assoc($sql)) {
        $rows[] = $r;
    }
    print json_encode($rows);
}

```

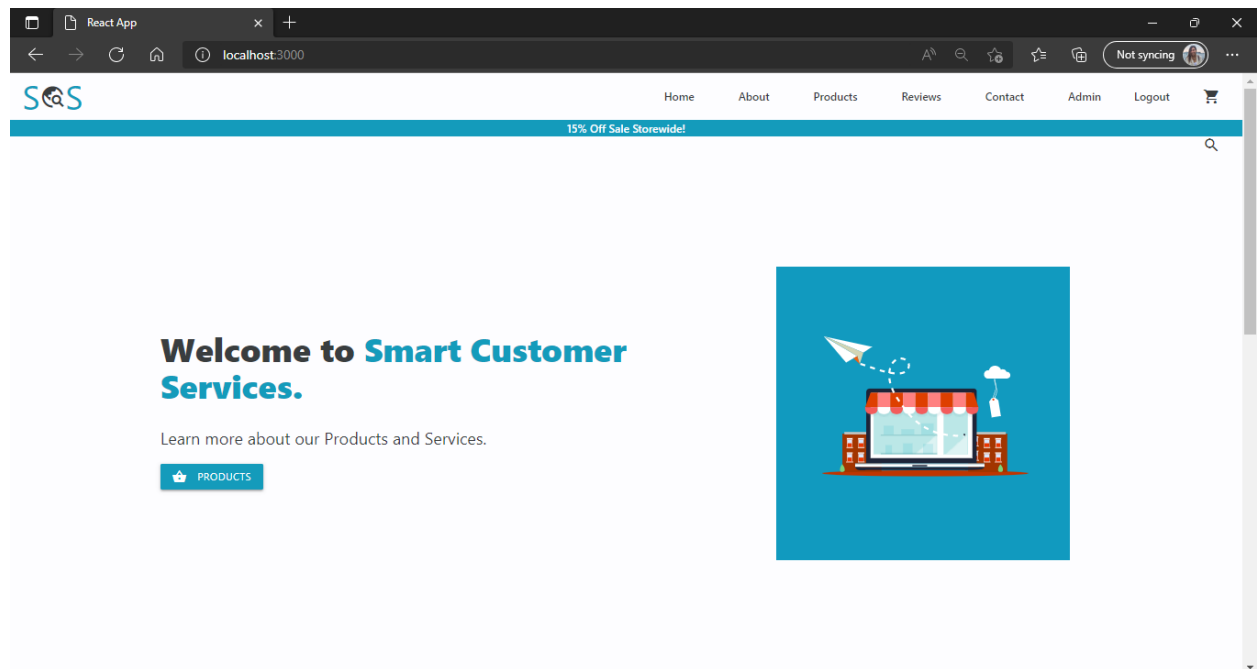
Once the data is received, the page displays the data in a carousel that the user can interact with.

## Part B-5: Browsers

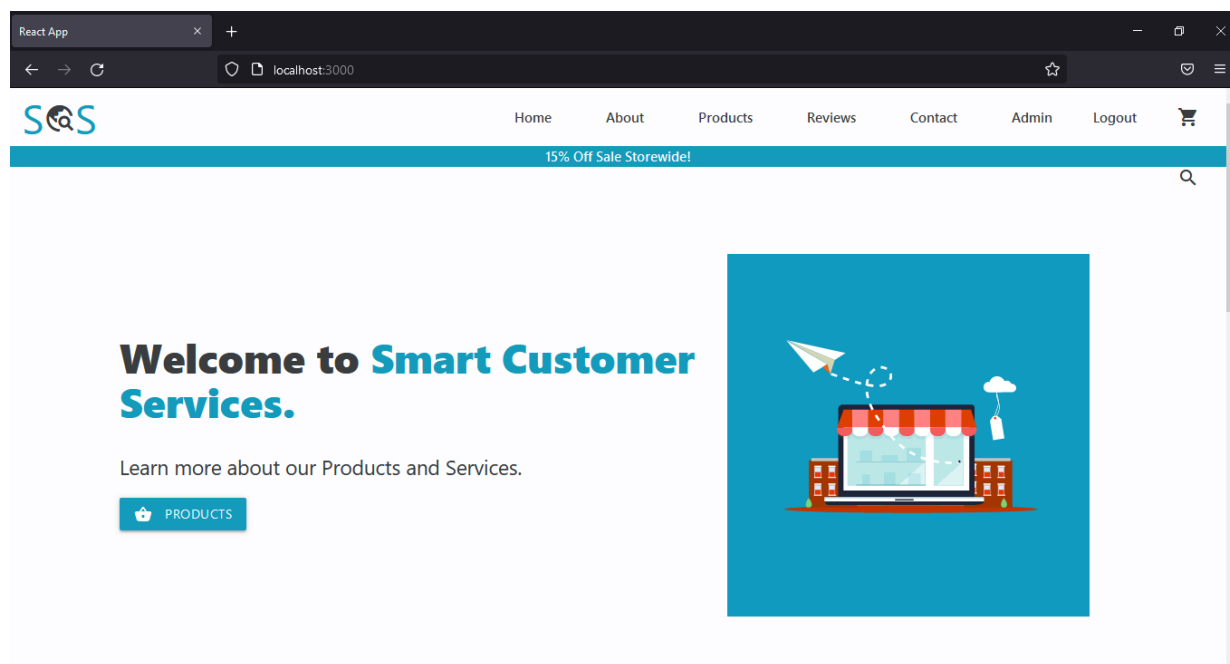
Chrome:



Edge:



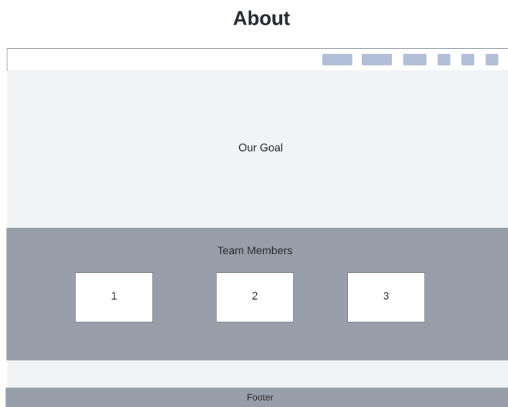
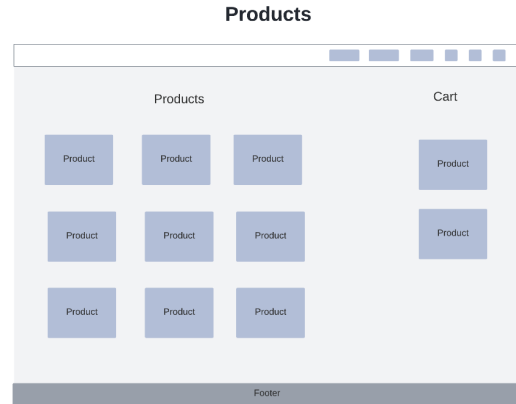
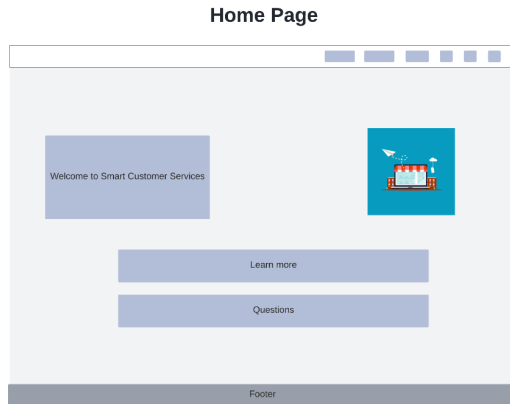
Firefox



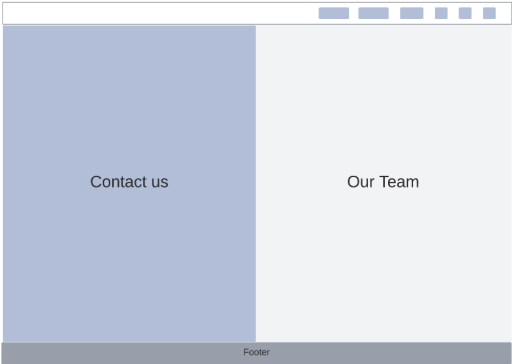
## Part C: SPA

### Webpages

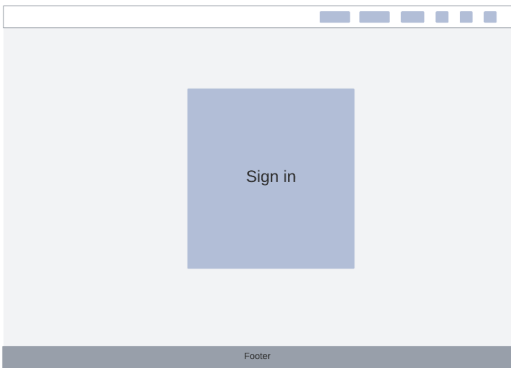
A rough draft of our UI layout for each page is as follows:



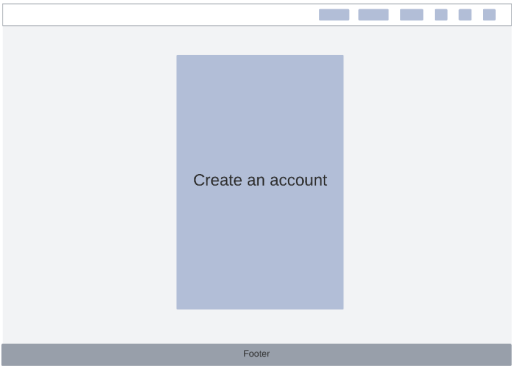
Contact



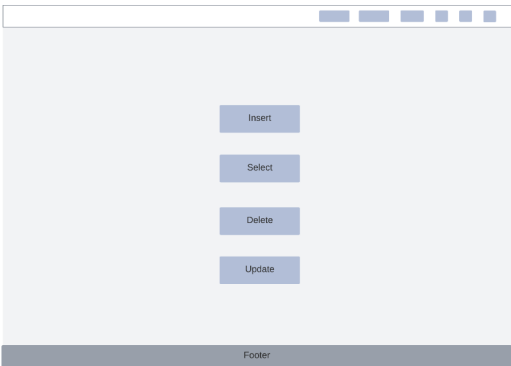
Login



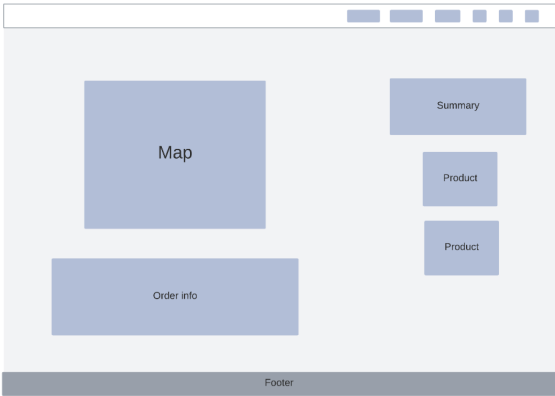
Register



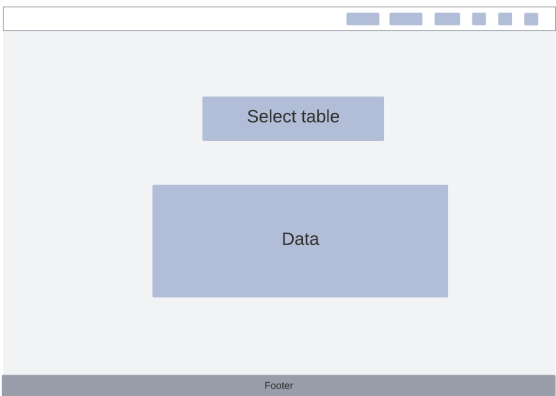
Admin



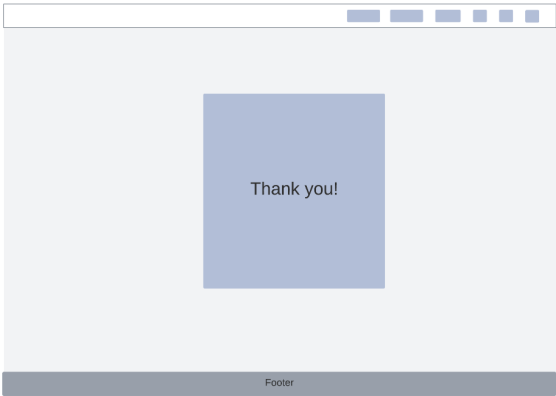
Shopping Cart



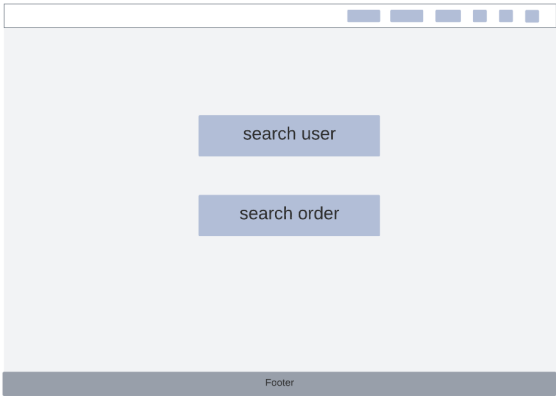
Select



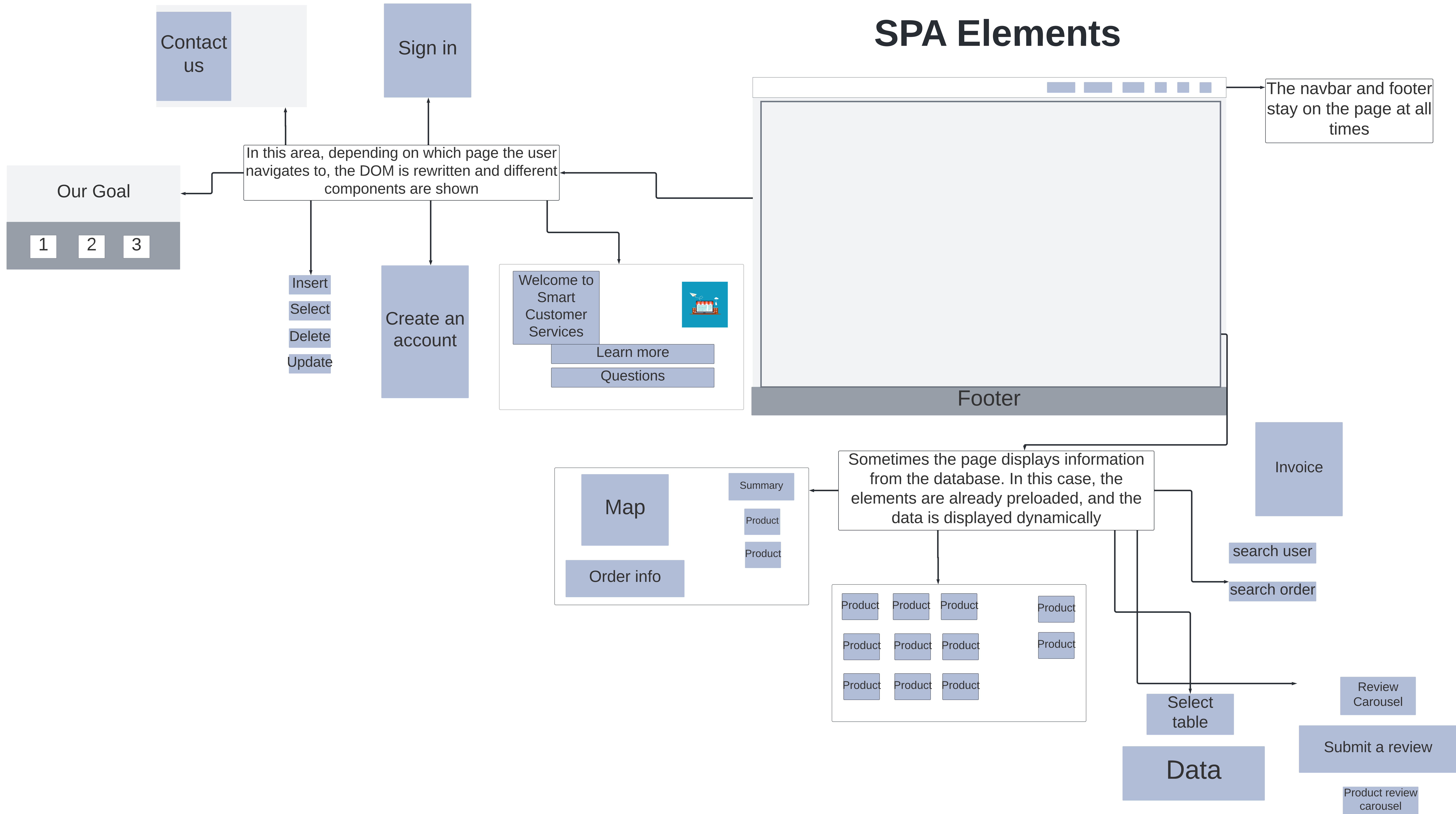
Invoice



Search



# SPA Elements



## SPA implementation in React

SPA is implemented into our React project using a library called React Router DOM. It is imported into the main App.js file like so:

```
import {BrowserRouter, Routes, Route} from 'react-router-dom';
```

The App.js file then gets set up like this:

```
function App() {  
  return (  
    <BrowserRouter>  
      <div className="App">  
        <Navbar/>  
        <Routes>  
          <Route exact path="/" element={<Home/>}></Route>  
          <Route path="/about" element={<About/>}></Route>  
          <Route path="/products" element={<Products/>}></Route>  
          <Route path="/contact" element={<Contact/>}></Route>  
          <Route path="/review" element={<Review/>}></Route>  
          <Route path="/invoice" element={<InvoiceComplete/>}></Route>  
          <Route path="/shoppingcart" element={<ShoppingCart/>}></Route>  
          <Route path="/search" element={<Search/>}></Route>  
          <Route path="/login" element={<Login/>}></Route>  
          <Route path="/register" element={<Register/>}></Route>  
          <Route path="/select" element={<Select/>}></Route>  
          <Route path="/admin" element={<Admin/>}></Route>  
        </Routes>  
        <Footer/>  
      </div>  
    </BrowserRouter>  
  );  
}  
  
export default App;
```

The entire site is surrounded by the BrowserRouter tag which links the UI to the page URL. The Route tags connect a certain component to a specific URL so whenever the page contains a certain part of the URL path, the relevant component will be mounted to the page. In some of the components, it imports react-router hooks called Link, and useNavigate. Link tags are buttons that help change the URL to a specific pathname without refreshing the page. Link tags are used like so:

```
<Link to="/"><a>Home</a></Link>
```

UseNavigate allows you to change the path in the URL on a trigger rather than a button press. It is done by initiating the useNavigate function and then calling the function with the pathname. Like so:

```
const navigate = useNavigate();
```

```
navigate("/login");
```

## SPA Architecture

In our program, the navbar and the footer will be on the page at all times. And based on the path in the URL, the content of the page between the navbar and



footer will be filled by the component according to the pathname. Some of the components are static, meaning that all of the contents are made through HTML and CSS, such as our homepage and our about page. Other pages like our review and products pages are generated dynamically using PHP and the MySQL database. They communicate with the frontend react using axios which sends POST request information to the server. The server also sends JSON encoding back to the front end which can be manipulated using JavaScript to display on the front end. In both our static and dynamic pages, all assets, including JavaScript and CSS, are loaded onto a single page. When performing actions on our webpages, or navigating to a different page, no round trips are necessary to load subsequent assets, and the page is not reloaded; instead, the DOM is rewritten.

## **Iter IV:**

### Login and Register Validation

For the login and register, the security features that were added were the use of the md5 hash and adding a random salt to the password. We made sure to encrypt the passwords in the users table so users who have access to the database will not be able to view other user's passwords.

For register, we used a function called generateRandomSalt which generates an encoded random 12-byte value using the command "base64\_encode(random\_bytes(12))". Shown here:

```
function generateRandomSalt(){  
    return base64_encode(random_bytes(12));  
}
```

We then call the generateRandomSalt function and set the salt value to \$salt. Then we hash the password obtained from POST attached to the salt.


```
$postal = $_POST[ 'postal' ];  
$salt = generateRandomSalt();  
$password = md5($password.$salt);
```

We then insert the salt and the hashed salted password into the database.

```
$reg = "INSERT into users(first_name, last_name, phone, email, addy,  
postal, login_id, login_salt, login_password, balance, admin_val) values
```

```
(' $fname', '$lname', '$phone', '$email', '$addr', '$postal', '$username', '$salt', '$password', 0, false)";
```

This required us to add a new column to the database called “login\_salt”.



	user_id	first_name	last_name	phone	email	addy	postal	login_id	login_salt	login_password	balance	admin_val
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	Dave	Tran	000-000-0000	dave.tran@gmail.com	Jane Finch Mall	M8V3B6	davetran	vW4SJvFJ9hkUCP	ffa4fb0d0044e3f0384d75e881fd5e1c	500	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	Anthony	Tran	000-000-0001	atran@gmail.com	Yorkdale	M8V3B6	atran	5EA3f0orN4Ztwzf	fe6b3f9b3c01b91d3278d241f833e7eb	500	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	Sera	Wong	000-000-0002	swong@gmail.com	Jane Finch Mall	M9C3J5	swong	b3f5KyIPaFMEfMyB	86c9a1badb4cd57b4ee58884ba9e1f7	500	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	Nanamin	Kento	000-000-0003	nkento@gmail.com	4361 Yonge Street	M4W1J7	nkento	bHxMogQ6o6rihwuO	ffefb6e0063e8d342cfab1ab4ebcef1c	700	0
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	Seto	Kaiba	000-000-0004	skaiba@gmail.com	2514 Merton Street	M1L3K7	skaiba	AhOdbwsf6BcKF6kx	cbf720f7a3d7943c02af71e495143791	500	0
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	Yae	Miko	000-000-0005	ymiko@gmail.com	1522 Adelaide St	M5H1P6	ymiko	j+BAsC+TjHCi9Vlt	6fedbd16b169161705dbdf61a409837	900.5	0

For login, we queried the salt based on the username submitted and md5 hashed the password the user has entered attached with the salt queried from the database. We then check if the hashed password is the same as the hashed password in the database. If the values are the same, then the user gets logged in.

```
$result = $db -> query("SELECT login_salt FROM users WHERE login_id = '$username'");

$num = mysqli_num_rows($result);
$row = mysqli_fetch_row($result);
if ($num == 0){
    $error = "Invalid Username or Password.";
    echo json_encode(array("sent" => false, "session" => ""));
}else{
    $salt = $row[0];
    $password = md5($password.$salt);

$result = $db->query("SELECT * FROM users WHERE login_id = '$username' AND login_password = '$password'");

$num = mysqli_num_rows($result);

if ($num == 0){
    $error = "Invalid Username or Password.";
    echo json_encode(array("sent" => false, "session" => ""));
}else{
    $_SESSION['username'] = $_POST['username'];
    echo json_encode(array("sent" => true, "session" => $_SESSION['username']));
}
mysqli_close($db);
?>
```

## Credit Card and Payment

When the user enters in their credit card information and place their order, we will salt it once again using the MD5 hashing algorithm and store their order details into our database. This is once again to ensure the privacy and security of our users, and to prevent attacks from external hackers.

```

$salt = generateRandomSalt();
$salted_card_name = md5($card_name.$salt);
$salted_card_num = md5($num.$salt);
$salted_expiry = md5($expiry.$salt);
$salted_cvv = md5($cvv.$salt);

$insertOrder = "INSERT into orders(date_issued, date_completed, order_price, payment_code,
  user_id, trip_id, receipt_id, branch_id, card_name, card_num, expiry, cvv, salt)
VALUES (CURDATE(), CURDATE()+ INTERVAL $trip_time DAY, $total, 1, $user_id[0],
  $trip_id, $receipt_id, $branch_id[0], '$salted_card_name', '$salted_card_num',
  '$salted_expiry', '$salted_cvv', '$salt')";

```

Once in our database, the information is securely salted, but we can verify the credentials of anyone attempting to login.

card_name	card_num	expiry	cvv	salt
0d4876364b363418651146f2cb5e5aa4	1234123412341234	55ad2cb62cc371bd8db7d02fa0d8be73	cA0p6zBOwJJ5sLeHcA0p6zBOwJJ5sLeH	vW4SjfvFJf9hkUCP
0d4876364b363418651146f2cb5e5aa4	69694206969442006969420696944200	0d4876364b363418651146f2cb5e5aa4	0d4876364b363418651146f2cb5e5aa4	AhOdbwsf6BcKF6kx
0d4876364b363418651146f2cb5e5aa4	41234567746524536969420696944200	cA0p6z32BOwJJ5sLeHdse32dd25512	3JJ5sLeHd12643e231a32d12643d2534	bHxMoqO6o6rlhwuO
9b2551264381f6e90e95f5996feb098b	9b2551264381f6e90e95f5996feb098b	0d4876364b363418651146f2cb5e5aa4	9b2551264381f6e90e95f5996feb098b	cA0p6zBOwJJ5sLeH
b748a73455da3e87e5f91a108de93a11	e0b5f89186b653a52401785b8fd5e379	760eba476a7702a34db10c24363ad132	55ad2cb62cc371bd8db7d02fa0d8be73	/SIYf4ITxeFbD8nD