We provided a couple of test cases to our First Demo. Essential we wanted to make sure there was an established connection to the Database. Our first goal was to implement a Read and Update from the database and a way to create a PHP session for the user. These include:

1. index.html and Login.php

   To make sure the user would provide the correct credentials when logging in we made sure to first have the Login.php be able to provide a correct return of the user's credentials. In order to do that we have to provide a post to send the users data. We confirmed this by logging the information into the Console Log via Console.log(“”). Once we thought everything was providing the correct functions we decided to have the page redirect the user to splashpage.php.

   This was the provided Test Case we wanted to implement for this page:

Test: TC01
Function Tested: Login(String, String): String

Call Function Pass	User gains Access to Administrative Rights
Call Function Fail	A String containing null is returned and the user is redirected to the login page

2. Splashpage.php

This was similar to the login.php as we wanted to make sure the SQL server was making communication to the webpage. This was fairly standard as the webpage would provide all items in the database on the webpage.

3. Search.php

We wanted the Search.php page to display all items and be able to create a session that could store a kart where a user could add items and remove them. This was a fairly standard way to test this as the items would be displayed at the top of this page including there quantity. We did this webpage to add items but not all items can be currently added and we are looking at a way to fix it. Also the search function is not properly working so we will create a way for the page to process all the items through PHP instead of a SQL search function.

These were the provided Test Cases we wanted to implement for this page:

Test: TC03
Function Tested: AddItem(String, String): ItemList

Call Function Pass	User is notified that the item was correctly added to the Database and the webpage will reload with an update item list array
Call Function Fail	User is notified that the item was not correctly added to the Database and will send back a warning message

Test: TC05
Function Tested: RemoveItem(String, String): ItemList

Call Function Pass	User is notified that the item was correctly removed from the Database and the webpage will reload with an update item list array
Call Function Fail	User is notified that the item was not correctly removed from the Database

Test: TC06
Function Tested: GetItemList(): ItemList

Call Function Pass	User gets a webpage with all items loaded
Call Function Fail	User is notified that the webpage can not currently load any items from the database

4. Warehouse.html & getWarehouse.php

We wanted only to be able to read from the SQL server and create a dynamic GUI based on the size of a selected warehouse. At first we wanted to make sure there was communication amount the two pages with a Console.log() function. Then we wanted to make sure the page would dynamically align and create a map that had similar dimension to the size of the warehouse provided by getWarehouse.php. We soon found out that the while the map would display correctly on some browsers some other browsers it would not. We have decided from this point to make sure that the whole website will display properly on all mainstream browsers.