Increment 4 Report

Plan Your Shopping

Project Group 18

Vikas Kondapalli (39)

Swatvik Gunamaneni (27)

Gopi Krishna Bodapati (8)

Introduction

Would it not be great if we were to know the total amount we would end up spending on our shopping trip beforehand? What if we had an estimate of the price for all the products we were to shop when we go out for shopping? Our project provides a solution to these scenarios. Our project gives you an estimate of the total you might end up spending when you go out for shopping even before you step out of your house.

This helps the shoppers to plan their shopping trip. As you have the total well in advance you would not end up spending too much or too little. You search for all the items you need and you search for them in our application. Then you have a list of all the stores offering the items along with their prices for those items. You get to pick the store offering the best price or the one closest to you.

The shoppers can look for all the items they need and pick the best option to buy an item and add it to the cart and continue the search for another item until they have looked for all the items on their shopping list. The final product is a list of stores offering the best price on each product with their addresses. Now the user can simply go to the particular store offering him the best price on the items he is looking for, rather than going to each and every store and making a comparison of the prices himself.

This application comes in handy to those people who cannot afford to lose time going to each and every store and compare the prices by themselves. It is also useful to those people who have a very limited budget for their expenses. They are not restricted to buying all the items from a single store as they have an idea of the best price being offered on the same item at a different place.

Progress from Increment 3

Prices

In the previous increment we could display the prices of the products on the dashboard which we have fetched using the Supermarket API. This is because of the fact that the Supermarket API in its trial version does not give the price of the items. It gives all other details like Store ID, Address, Product Name etc. but not the price of each product. The project being academic in nature, we have allotted random prices to the products.

Cart

We did not have the cart page in the previous increment. Now we have a cart page and when the user goes through the list of the items and their prices he can choose the best deal and add that item to his cart. When the user is finished searching for his items he can have a complete list of all the items along with the store offering the lowest price.

Map

We have improvised the map from the previous increment. The map now correctly points to the desired store than to the city of the store as in the previous increment. We also display the distance to the store from the current location of the user. We have the Navigator object from the Google Maps API to locate the coordinates of the user to determine his current location.

Features

Login and Registration Pages

The login page allows a user to log into the application and then has his dashboard displayed to him. The user enters his username and password and they are verified with the credentials he used during the time of registering himself with the application. When the user registers with the application his details are stored in the MongoDB database. When the user tries to log into his account, his username and password are validated against the ones stored in the database. Only when the credentials match, the user is logged into his account and gets his dashboard displayed to him.

The following is the code snippet we have used to implement the login functionality.

```
controller('login', function ($scope,$http) {
   $scope.login = function () {
          var username =  document.getElementById('username2').value;
         var password = document.getElementById('password2').value;
        $http({
              method: 'GET',
              url:
'https://api.mongolab.com/api/1/databases/vikas2/collections/users?apiKey=HxPILPvmcIj3SyZB
MlrPV38t7_BiBKUJ'
            }).then(function successCallback(response) {
   // this callback will be called asynchronously
   // when the response is available
            var users=response.data;
            //alert(users.length + " " + users[0].name);

            var i;
```

```
            var flag=0;
            for(i=0;i<users.length;i++)
            {
               if(username==users[i].name)
               {
                  if(password==users[i].password)
                  {

                     alert("login successful");
                      flag=1;
                      localStorage.setItem("username", username);
                      window.location.assign("home.html")
                      break;

                  }
               }
            }
            if(flag==0)
                alert("Incorrect username, password combination.");
            //alert(JSON.stringify(response.data[0]));
  }, function errorCallback(response) {
    // called asynchronously if an error occurs
    // or server returns response with an error status.
  });

        };

})
```

Dashboard

The dashboard is the place where the user has all his search results displayed. It is from here where the user can add items to his cart. The results of the search, the maps and all other things are displayed here.

Search

When the user selects the state in which he is and then selects his city and then chooses to displays the stores in that city, he has a list of all the stores in his city with their addresses and their Store IDs. He can pick a store and use its Store ID to search for an item in that store.
The following is the code snippet used in displaying the list of stores.

```
var mygetrequest=new ajaxRequest();
if (mygetrequest.overrideMimeType){
 mygetrequest.overrideMimeType('text/xml');
 }
mygetrequest.onreadystatechange=function(){
 if (mygetrequest.readyState==4){

  if (mygetrequest.status==200 || window.location.href.indexOf("http")==-1){
   var xmldata=mygetrequest.responseXML //retrieve result as an XML object

          myFunction(xmldata);
  }
    else
    {
       alert("An error has occured making the request")
  }
 }
```

```
}
mygetrequest.open("GET",
"http://www.SupermarketAPI.com/api.asmx/StoresByCityState?APIKEY=abb0645e0a&Selecte
dCity="+city2+"&SelectedState="+state, true);
mygetrequest.send();
function myFunction(xmlDoc) {

 storename = xmlDoc.getElementsByTagName("Storename");
 address = xmlDoc.getElementsByTagName("Address");
 city = xmlDoc.getElementsByTagName("City");
 state_global = xmlDoc.getElementsByTagName("State");
 zip = xmlDoc.getElementsByTagName("Zip");
 storeid = xmlDoc.getElementsByTagName("StoreId");
var table="<table style='width:100%'><tr><th align='left'><font
size='4'>Storename</font></th><th align='left'><font size='4'>Address</font></th><th
align='left'><font size='4'>City</font></th><th align='left'><font size='4'>State</font></th><th
align='left'><font size='4'>Zip</font></th><th align='left'><font
size='4'>StoreID</font></th></tr>";
    var str = "bbbbbbbbbbbbbbb";
for (var i = 0; i <(storename.length); i++) {

    table += "<tr><td>" +
  storename[i].childNodes[0].nodeValue + "</td>"  +
  "<td>" + address[i].childNodes[0].nodeValue + "</td>" +
   "<td>" + city[i].childNodes[0].nodeValue + "</td>" +
   "<td>" + state_global[i].childNodes[0].nodeValue + "</td>" +
    "<td>" + zip[i].childNodes[0].nodeValue + "</td>" +
     "<td>" + storeid[i].childNodes[0].nodeValue + "</td>" +
    "<td>"  +  "<button  type  =  'button'  id  =  "+  i  +  "  value  =  "+i+"  onclick
='locationStorage(this.id)'>"+showonmaps+"</button>"+ "</td>"+
  "</tr>";
```

```
}
table+="</table>";
document.getElementById("id01").innerHTML = table;




}
}
                    }
```

The following snippet of code has been used for listing the items

```
var mygetrequest=new ajaxRequest();
if (mygetrequest.overrideMimeType){
 mygetrequest.overrideMimeType('text/xml');
 }
mygetrequest.onreadystatechange=function(){
 if (mygetrequest.readyState==4){

  if (mygetrequest.status==200 || window.location.href.indexOf("http")==-1){
   var xmldata=mygetrequest.responseXML //retrieve result as an XML object

          myFunction(xmldata);
  }
    else
    {
       alert("An error has occured making the request")
  }
 }
}
```

```
mygetrequest.open("GET",
"http://www.SupermarketAPI.com/api.asmx/SearchForItem?APIKEY=abb0645e0a&StoreId="+
storeid+"&ItemName="+item, true);
mygetrequest.send();
function myFunction(xmlDoc) {

 x1 = xmlDoc.getElementsByTagName("Itemname");
 x2 = xmlDoc.getElementsByTagName("ItemDescription");
 x3 = xmlDoc.getElementsByTagName("ItemCategory");
 x4 = xmlDoc.getElementsByTagName("ItemID");
 x5 = xmlDoc.getElementsByTagName("AisleNumber");
    for( var i=0; i< (x1.length);i++)
        x6[i]=Math.floor((Math.random() * 100) + 1);
var table="<table style='width:100%'><tr><th align='left'><font
size='4'>Itemname</font></th><th align='left'><font size='4'>ItemDescription</font></th><th
align='left'><font size='4'>ItemCategory</font></th><th align='left'><font
size='4'>ItemID</font></th><th align='left'><font size='4'>AisleNumber</font></th><th
align='left'><font size='4'>Price</font></th></tr>";
    alert("hello");
for (var i = 0; i <(x1.length); i++) {
    var itemname = x1[i].childNodes[0].nodeValue;
    var itemdescription =   x2[i].childNodes[0].nodeValue;
    var itemcategory = x3[i].childNodes[0].nodeValue;
    var itemID = x4[i].childNodes[0].nodeValue;
    var itemAislenumber = x5[i].childNodes[0].nodeValue;

  table += "<tr><td>" +
  itemname + "</td>"  +
  "<td>" + itemdescription + "</td>" +
   "<td>" + itemcategory + "</td>" +
   "<td>" + itemID + "</td>" +
```

```
    "<td>" + itemAislenumber + "</td>" +
    "<td>" + x6[i]+ "</td>" +
      "<td>" + "<button type = 'button' id = "+ i + " value = "+i+" onclick =cartstorage('" + item
+"'," +x6[i]+")>"+addtocart+"</button>"+ "</td> </tr>";

}
table+="</table>";
document.getElementById("id01").innerHTML = table;
}
```

Cart

When the user finds a product at a price that he is interested in, he can add that particular item to his cart using a button. And then that particular item gets added to the user's cart. The user searches for all the items he needs and finally the cart has a complete list of all the items he is looking for. The cart stores all the items in the order of the store they are to be bought from. At the end the user gets a list of stores along with the items to be bought at that store.

```
function cartstorage(item, price){
        var flag = 0;
       alert("cart storage");
      cart[cart_count] = [];
     cart[cart_count][0] = item;
      cart[cart_count][1] = price;
       cart[cart_count][2] = storeid;
       alert(cart[cart_count][0] + "  "+ cart[cart_count][1] + " " + cart[cart_count][2]);
       if(store_list.length == 0)
       {
          store_list[0] = storeid;
          store_count= store_count+1;
       }
       else
       {
          for( var i = 0;i<store_count;i++)
          {
             if(storeid == store_list[i])
             {
                flag = 1;
                break;
             }
```

```
            }
            if(flag == 0){
                store_list[store_count]=storeid;
                store_count= store_count+1;
            }
        }
        cart_count=cart_count+1;
        alert(cart.length);

    }

function showcart()
    {
     // alert(cart.length);
        //alert(cart[0][0]+" "+cart[0][1]+cart[1][0]+" "+cart[1][1]);
        alert("store count"+ store_count);
        for (var i=0; i<cart.length; i++)
            {

                localStorage.setItem("cartdata_0"+i, cart[i][0]);
                localStorage.setItem("cartdata_1"+i, cart[i][1]);
                localStorage.setItem("cartdata_2"+i, cart[i][2]);
            }
            for( var j=0;j<store_count;j++)
            {
                localStorage.setItem("storelist"+j, store_list[j]);
            }
            localStorage.setItem("stores_count", store_count);
            localStorage.setItem("cart_length", cart.length);
            window.location.assign("cart.html");
    }
```

Map

When the user has a list of all the stores in a particular city he is also provided with a button which on selection displays the map giving the route to that store from the user's current location. The map gives both the route to the store and also the distance from the user's current location.

```
.controller('mapoutput', function ($scope,$http) {

    var map;
    var map_spec;
    var directionsDisplay = new google.maps.DirectionsRenderer({
        draggable: true
    });
    var directionsService = new google.maps.DirectionsService();
    var latitude;
    var longitude;
    var pos_geo;
    $scope.initialize = function () {
        var pos = new google.maps.LatLng(0, 0);
        var map_spec = {
            zoom: 3,
            center: pos
          };

        map = new google.maps.Map(document.getElementById('map-canvas'),
        map_spec);
     };
    $scope.findOutMap = function () {
        if (navigator.geolocation) {
```

```
  navigator.geolocation.getCurrentPosition(function(position) {
    pos_geo = {
     lat: position.coords.latitude,
     lng: position.coords.longitude
    };
   alert(pos_geo.lat + " " +pos_geo.lng);
  });
 } else {
  // Browser doesn't support Geolocation
  alert("not able to get geo location");
 }    var start = new google.maps.LatLng(pos_geo.lat, pos_geo.lng);
      alert(start);
        var     end     =          localStorage.getItem("city")+"     "+localStorage.getItem("state")+"
"+localStorage.getItem("zip");
        alert(localStorage.getItem("city")+"                          "+localStorage.getItem("state")+"
"+localStorage.getItem("zip"));
        var request = {
          origin: start,
          destination: end,
          travelMode: google.maps.TravelMode.DRIVING
                    };

        directionsService.route(request, function (response, status) {
          if (status == google.maps.DirectionsStatus.OK) {
            directionsDisplay.setMap(map);
            directionsDisplay.setDirections(response);
            console.log(status);
                                                    }
          else{
          alert("there is a problem");
          });
```

```
                var distance = new google.maps.DistanceMatrixService;
        distance.getDistanceMatrix({
   origins: [start],
   destinations: [end],
   travelMode: google.maps.TravelMode.DRIVING,
   unitSystem: google.maps.UnitSystem.METRIC,
   avoidHighways: false,
   avoidTolls: false
                                }, function(response, status) {

                                if (status !== google.maps.DistanceMatrixStatus.OK) {
                                                alert('Error was: ' + status);

}

else {
                var originList = response.originAddresses;

                var destinationList = response.destinationAddresses;

                var outputDiv = document.getElementById('output_distance');


                outputDiv.innerHTML = '';

                var results = response.rows[0].elements;
                outputDiv.innerHTML = originList + '       to       ' + destinationList +

                        ':          ' + results[0].distance.text +  '<br>';

            }
```

Existing APIs

We have used APIs like Supermarket API, MongoDB API, Google Maps API in our application.

Supermarket API

In our application we had to get the prices of products available at various stores and also details like the store's address to be used in our app. This purpose was served by the Supermarket API which is a commercial API. We have used the trial version of the Supermarket API in our project.

The only shortcoming of using the trial version of the API was that it did not give the price of the items while giving all other details like the store ID, addresses. It is open source in its trial version. We can information from over 9000 stores all over US along with many of the retail chains. This API gives information about over 1,000,000 grocery products.

Google Maps

We intend to show the route to the store the user wants to shop at from his current location. We also want to find the distance from the user's location to the store. For this purpose we have used the Google Maps API which is a desktop web mapping service from Google. We can have satellite imagery, street maps, real time traffic conditions and many other features on using Google Maps.

MongoDB API

In our project the user has to log into the application but before this he has to register himself. The registration details cannot be stored in the local storage as they have to been fetched back at a later time for validations during login. So we needed a Database that could store the user details permanently. We have the used the MongoDB database for this purpose. This APIs allows data tone stored in the cloud and its retrieval whenever needed.

Mashup

In addition to the above APIs we have a used a mash up where the details given by one API in another API. We have used the store location given by the Supermarket in the Google Maps API to have the route to the store.

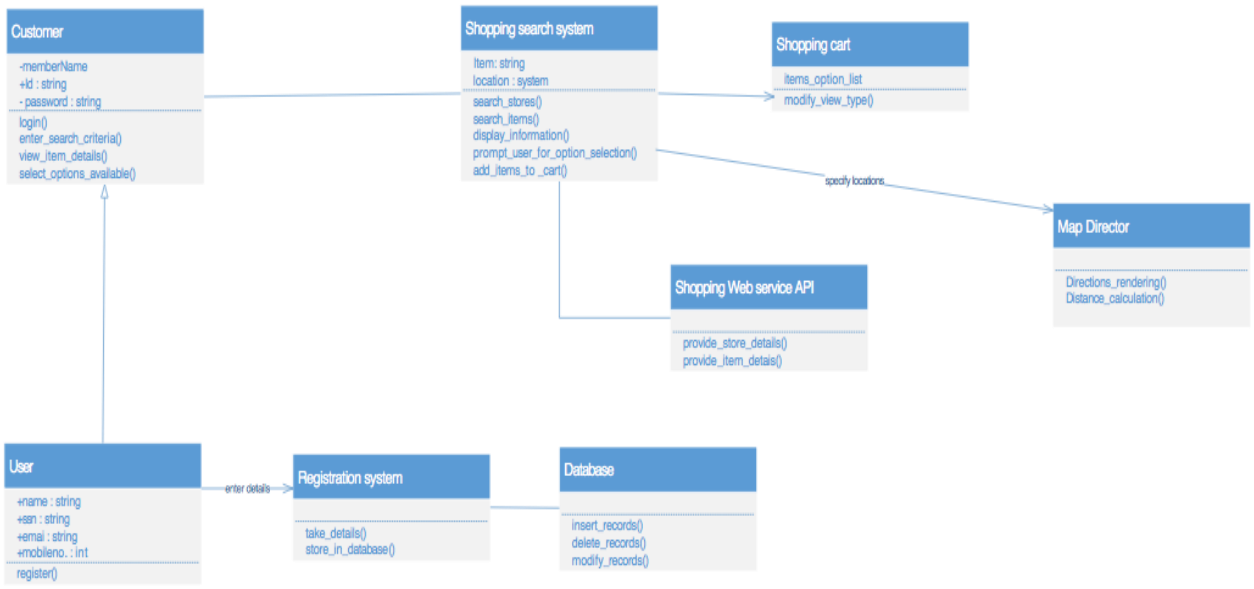
User Stories

From a user's point of view the application should the user to get himself registered in the first place. Secondly, when he tries to log into the system using the details he has already given during the registration page, he must be successfully logged in and directed to his Dashboard. At the dashboard the user must be able to select the state and city he is currently in. When the user after giving his state and city hits the display results button he must have a list of all the stores in that city with their details. When the user uses these details to search for an item he must get the price of time in that store and must be able to add the item of his choice to the cart. The cart page must segregate all the items based on the store they are to be bought from.

Detailed Design of Services

UML Diagrams

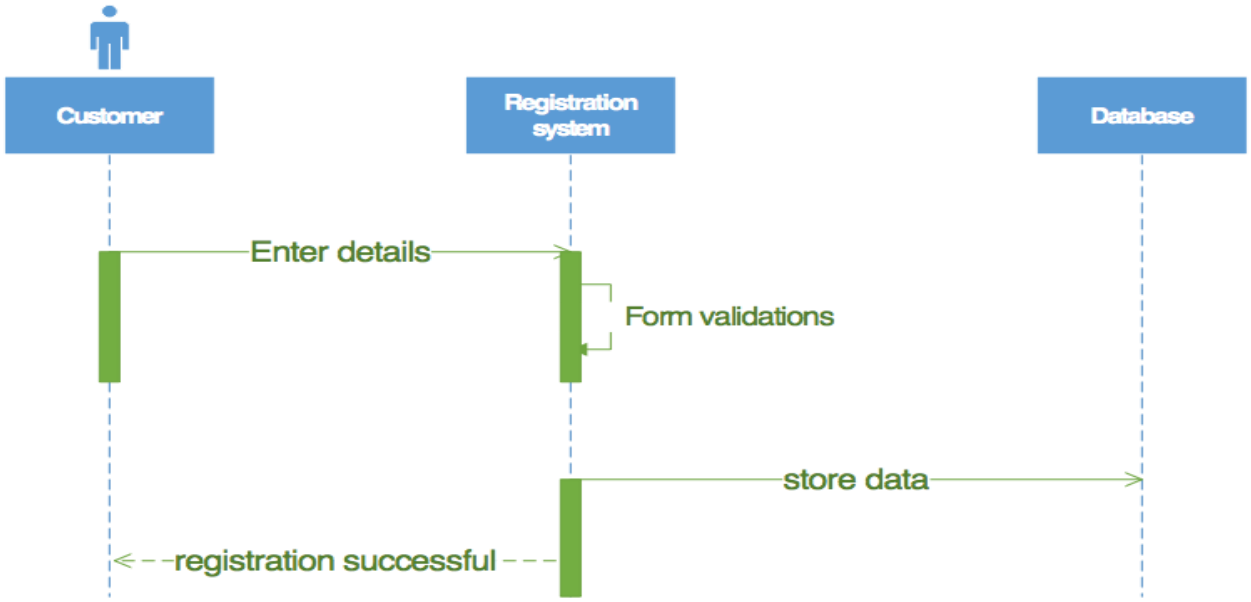
Class Diagram



In the above class diagram we can see that we have classes like customer, user, database, shopping system, shopping cart, map director, registration system. For a user to become a customer he has to get registered and this happens by the interactions between the user and registration system classes.

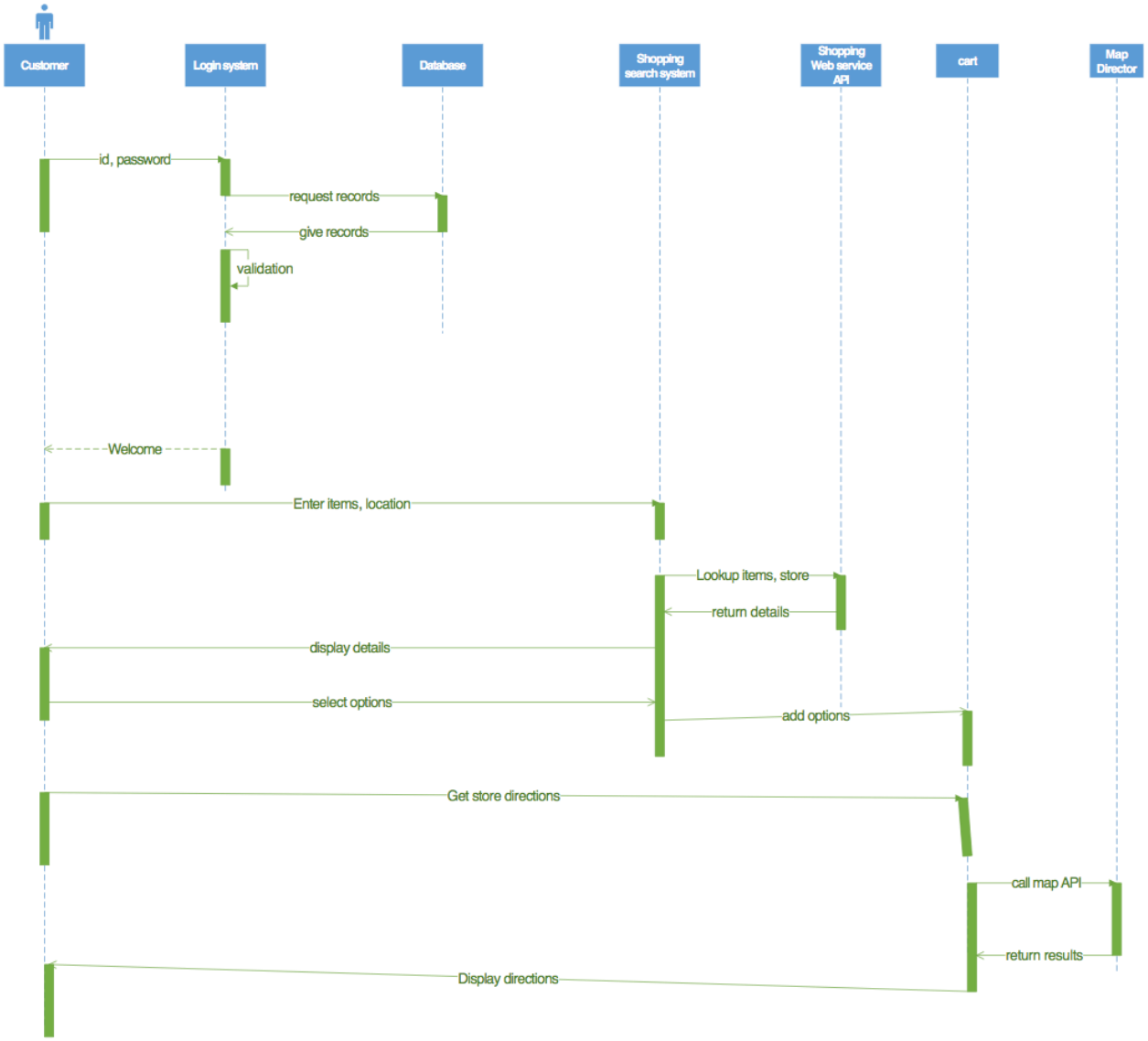
Sequence Diagram

Registration



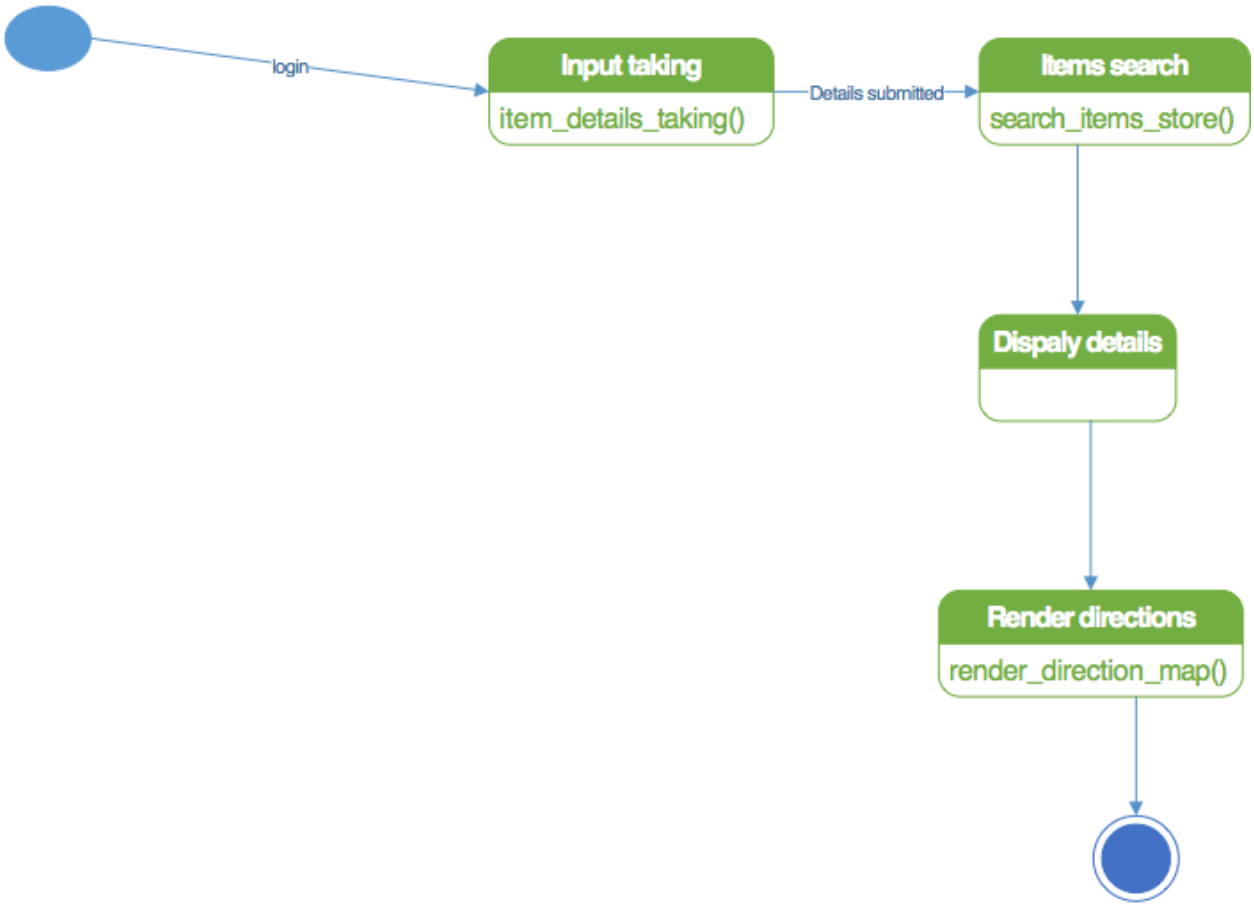
As seen in the above diagram, the customer has to first give his details which get stored in the database. These details are used later when the user wants to log into the system.

Search and display service



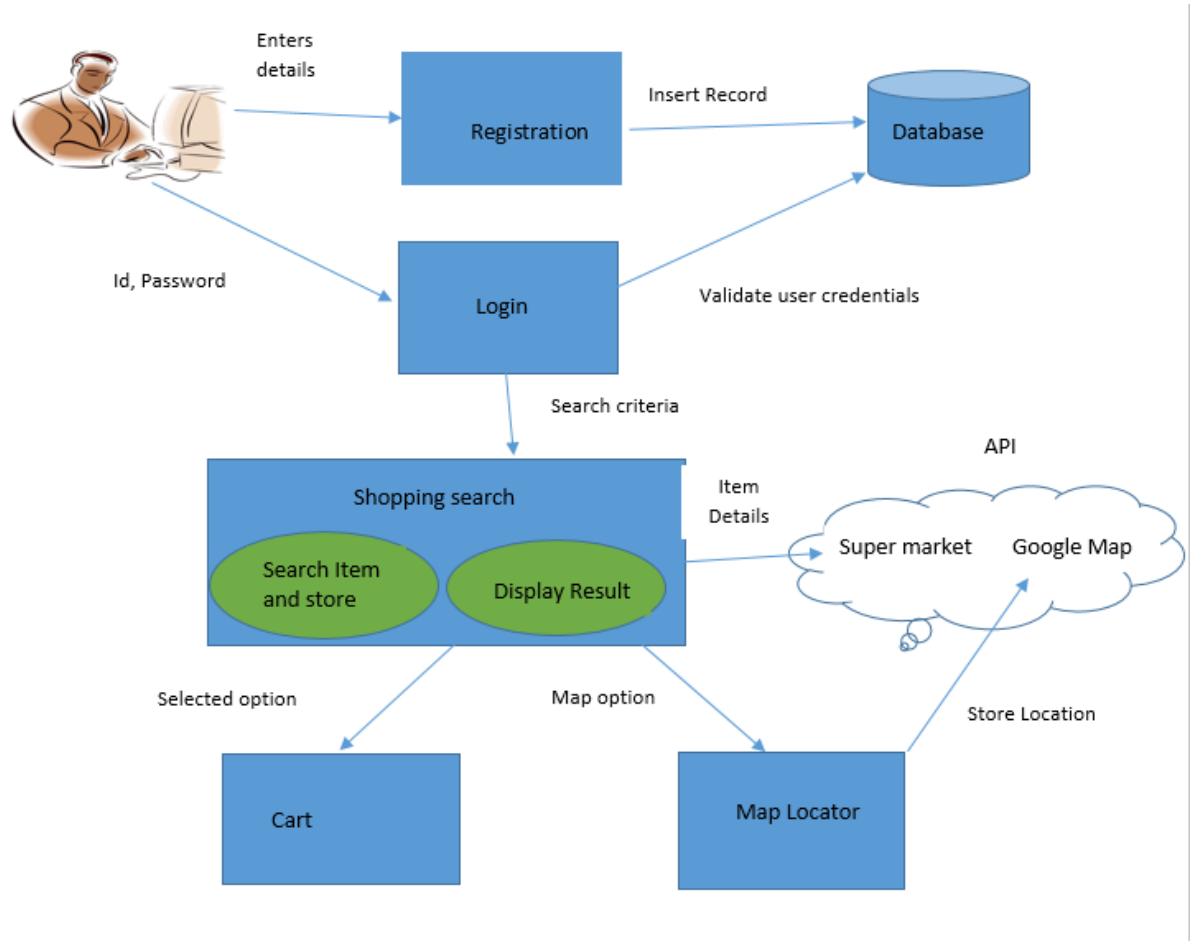
We can see from the above diagram how a user searches for a product and how the results are displayed in his dashboard.

State Diagram



The various states involved are illustrated in the above diagram.

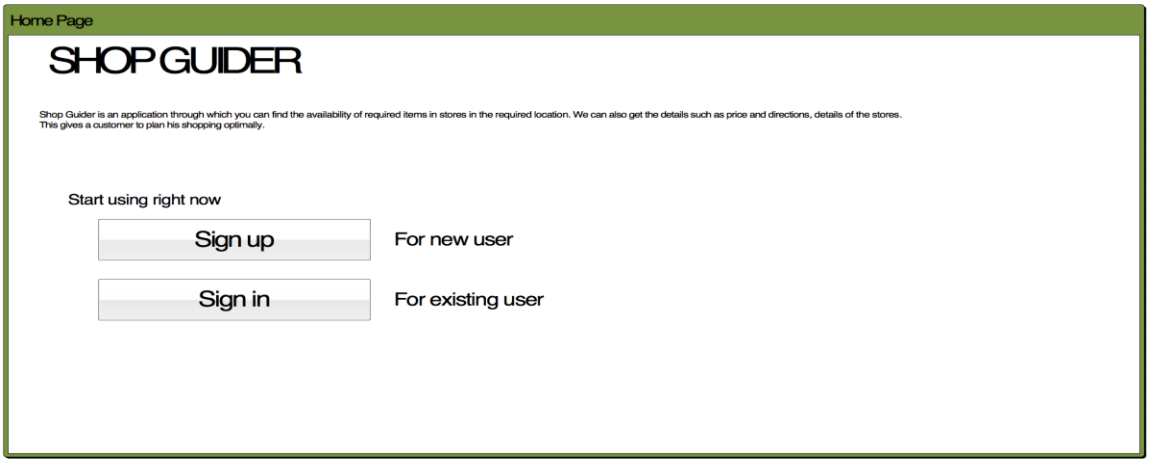
System Architecture



The above architecture depicts all the interactions taking place between various components of the system.

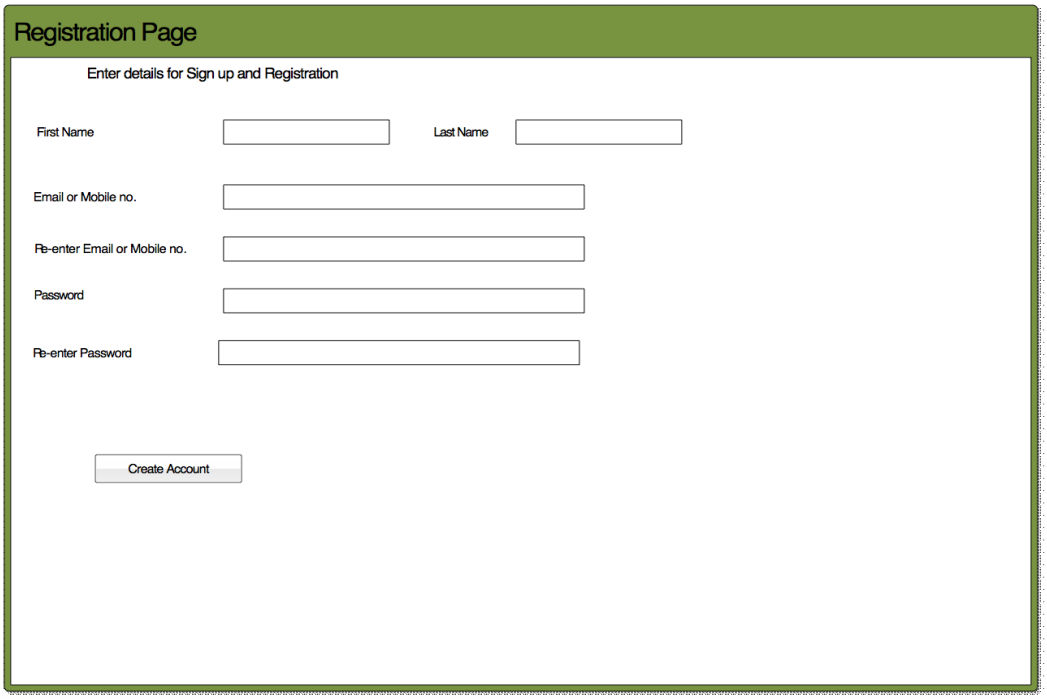
Wireframes

Home page



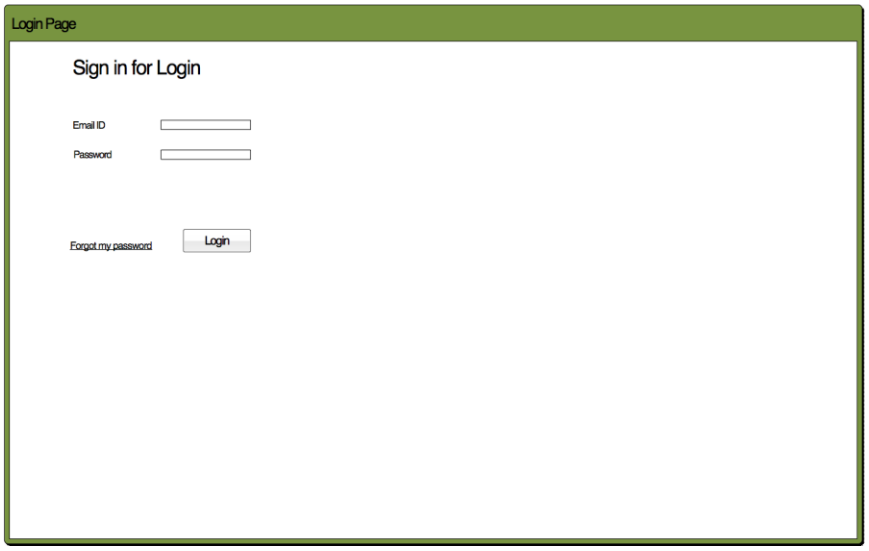
This is the home page where the user can either sign in or sign up if he is new.

Registration Page



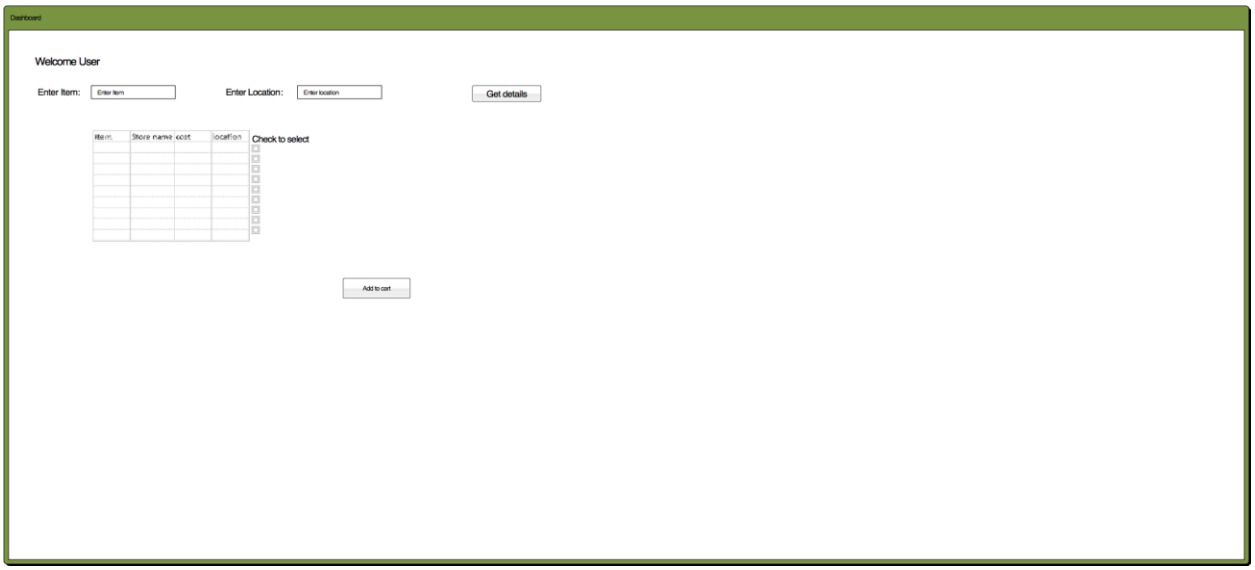
This is the page where the user registers with the system.

Login Page



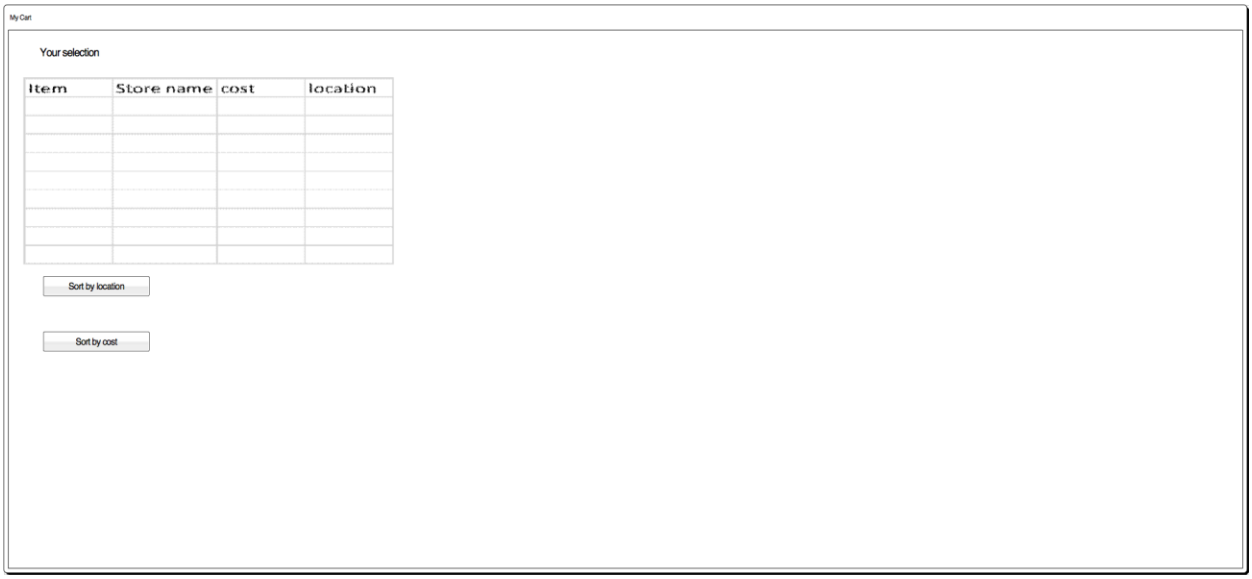
An already existing user can simply give his username and password and log into the system.

Dashboard



This is where all the activity happens. The search results get displayed.

Cart Page



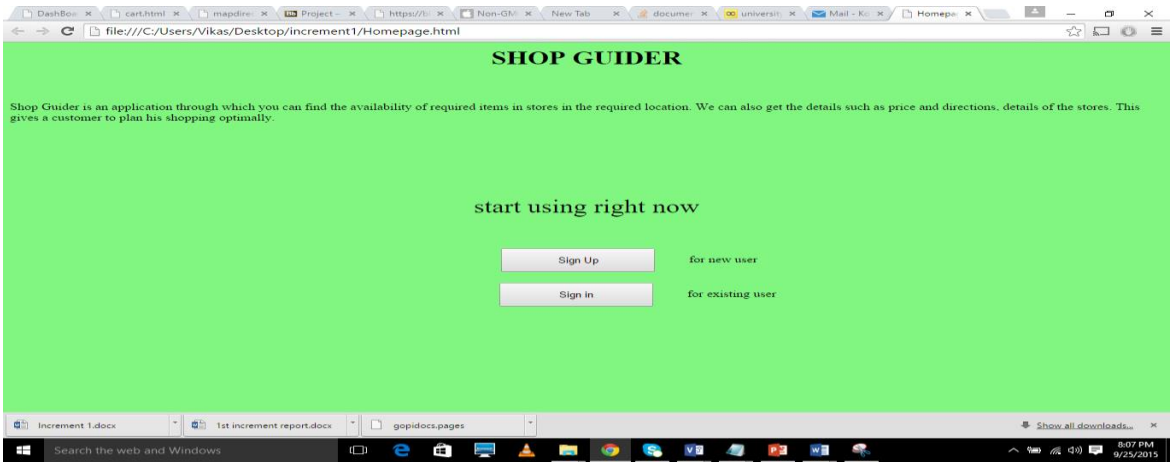
The user can add all the items he is interested in to his cart and the cart segregates them based on the store from there the user intends to buy them.

Map Locator Page

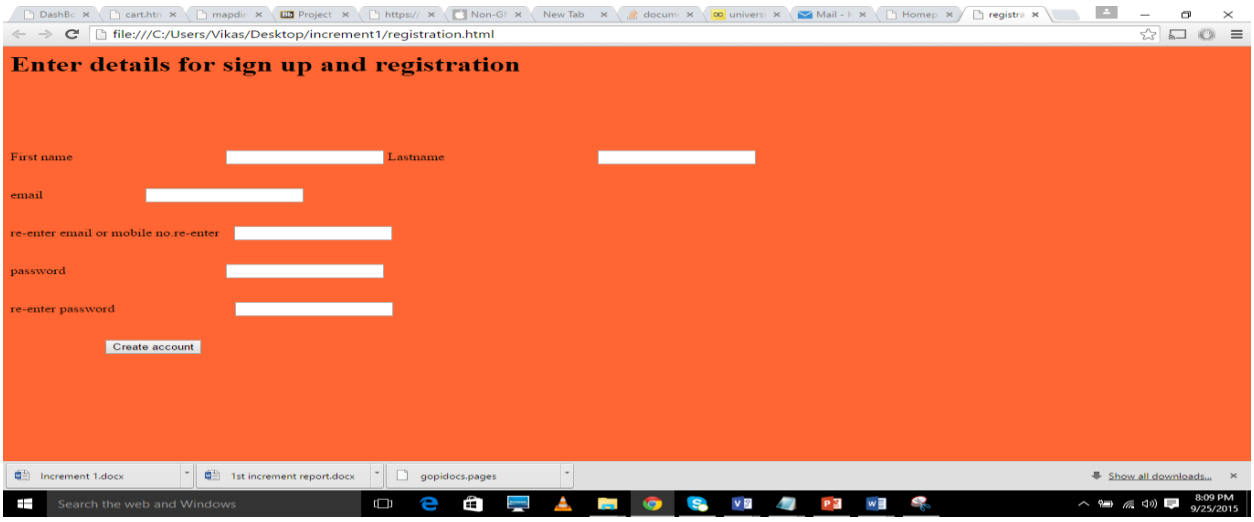
The map leading to the desired store from the user's current location is displayed.

Mockups

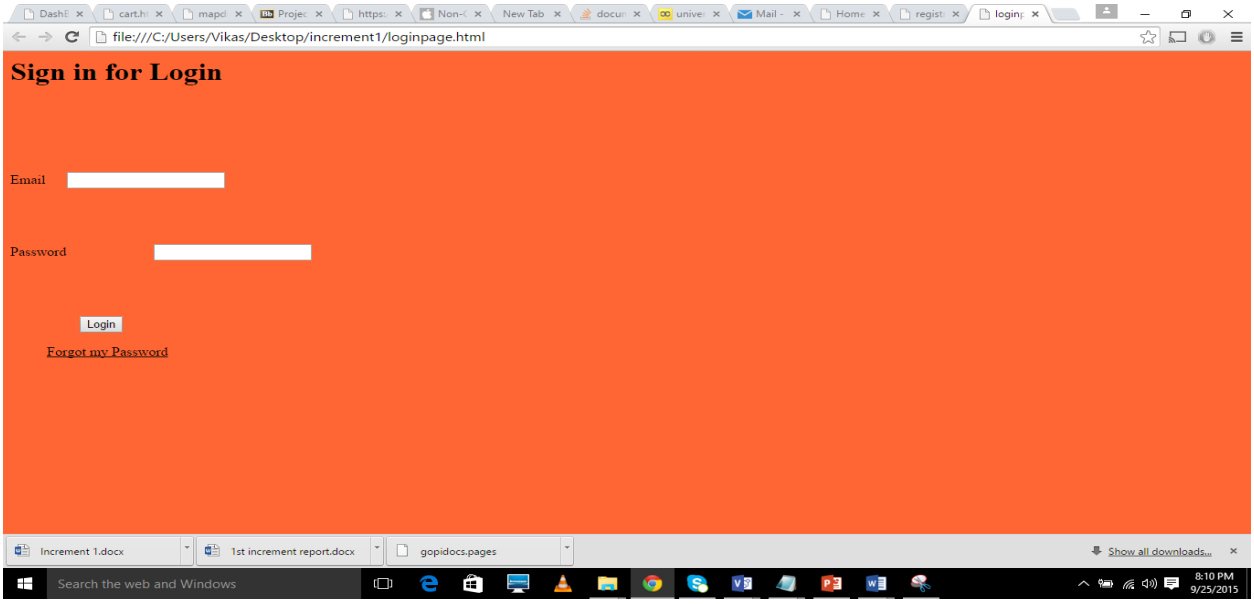
Home page



Registration Page

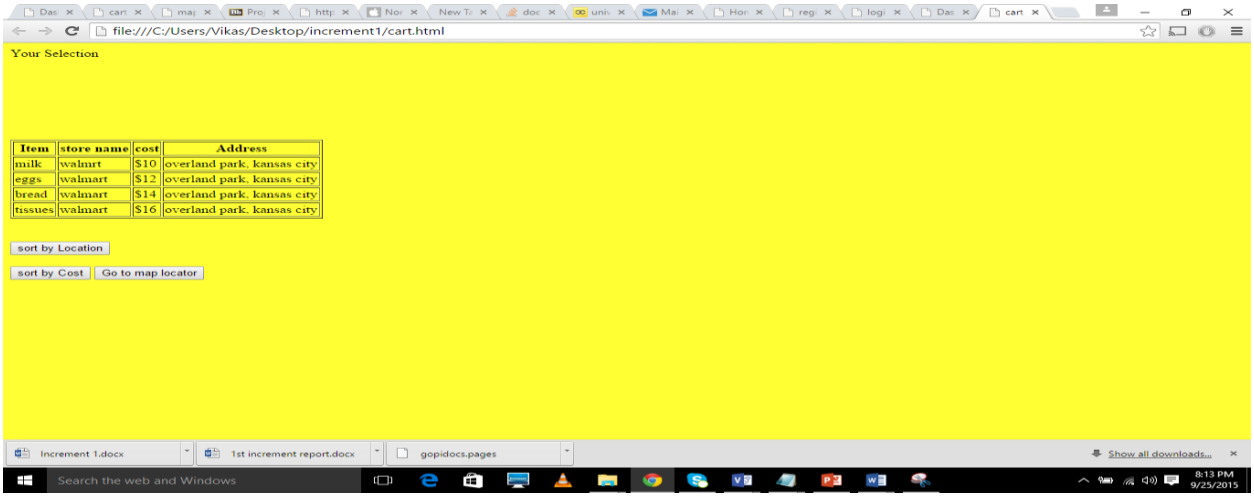


Login Page

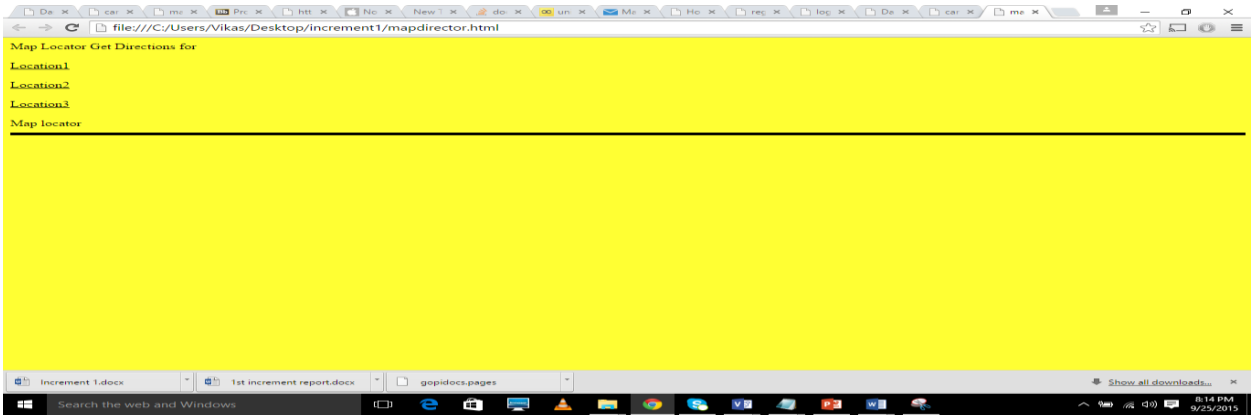


Dashboard page

Cart Page



Map Locator Page

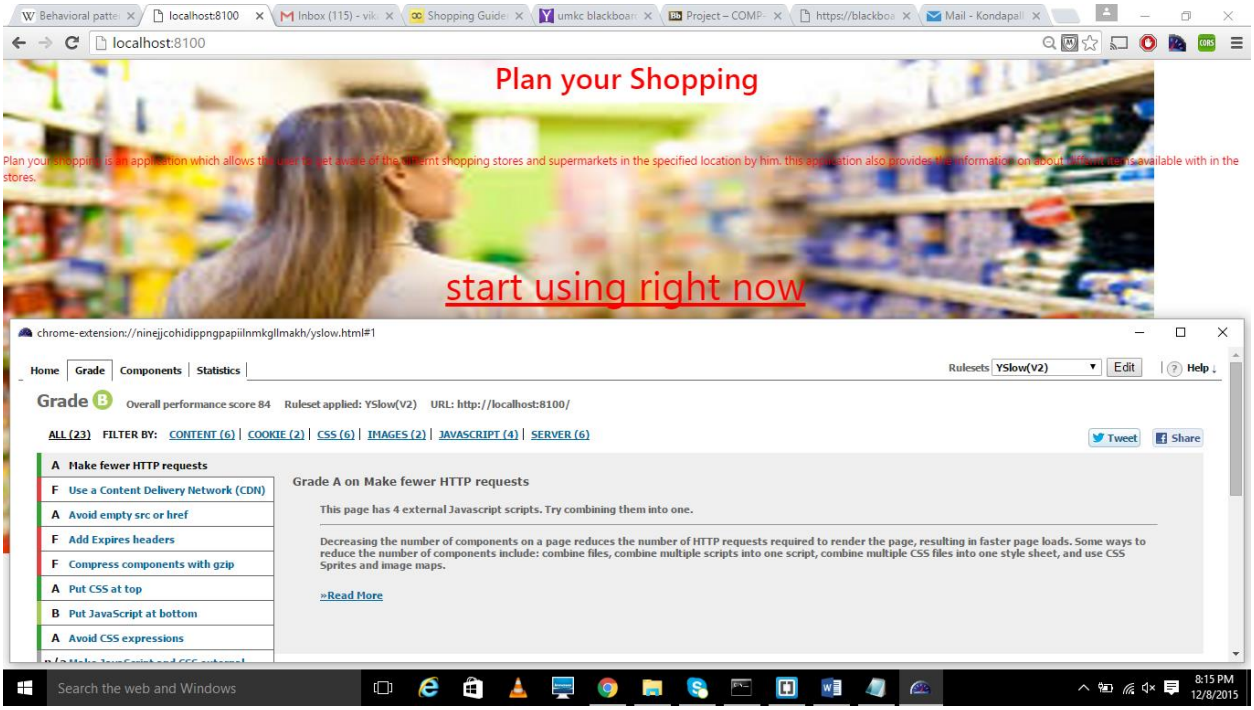


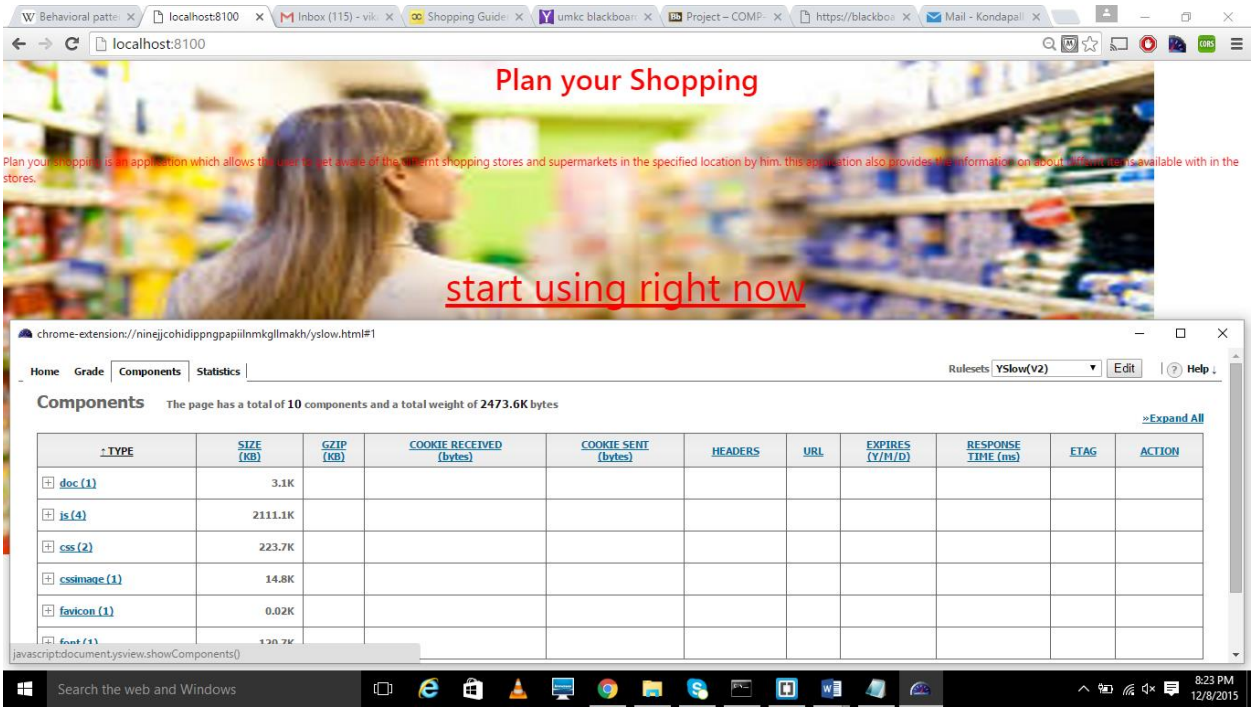
Testing

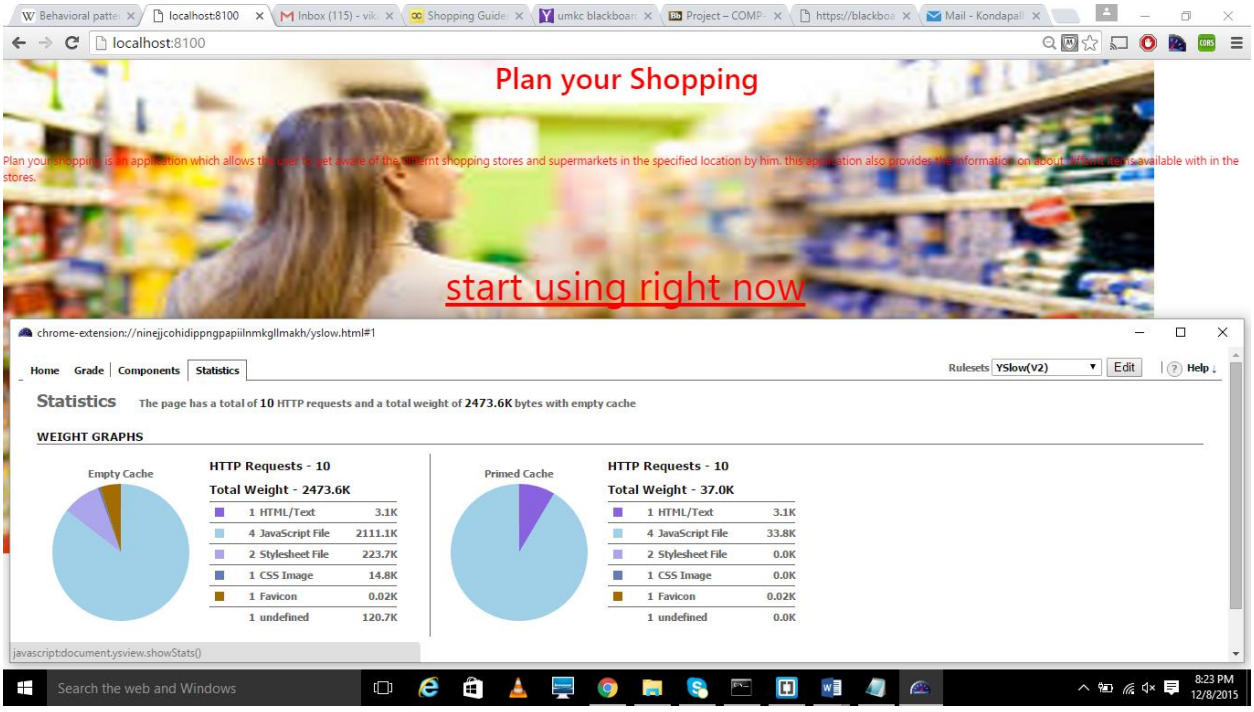
Performance Testing

YSlow tool has been used for doing the performance testing of the application.

This tool has graded the Application with a grade of B in terms of performance.



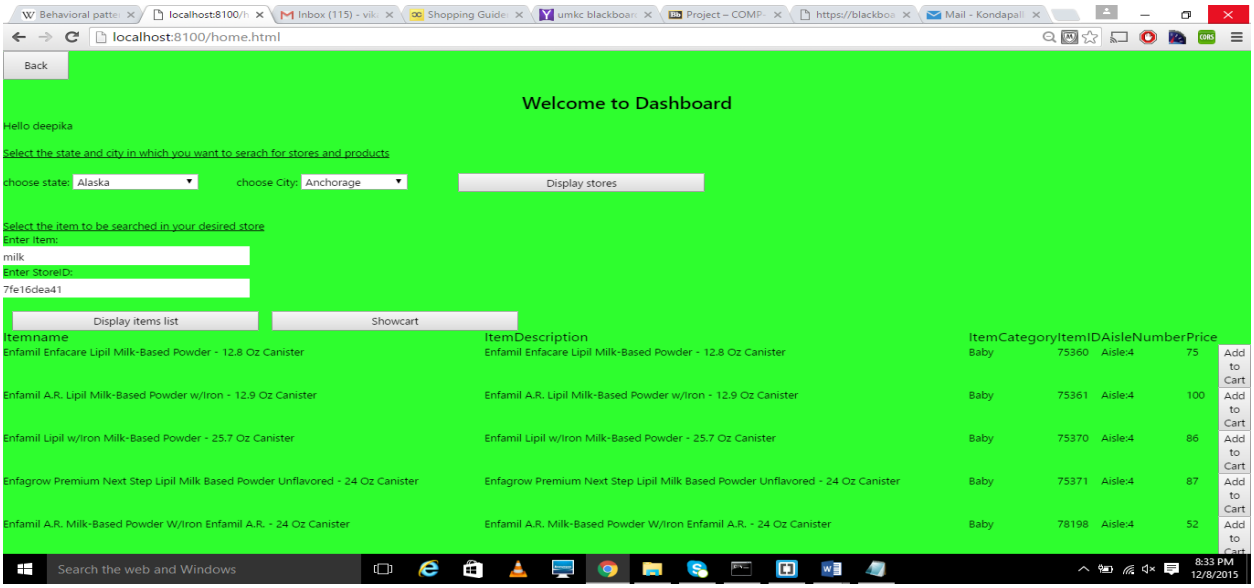




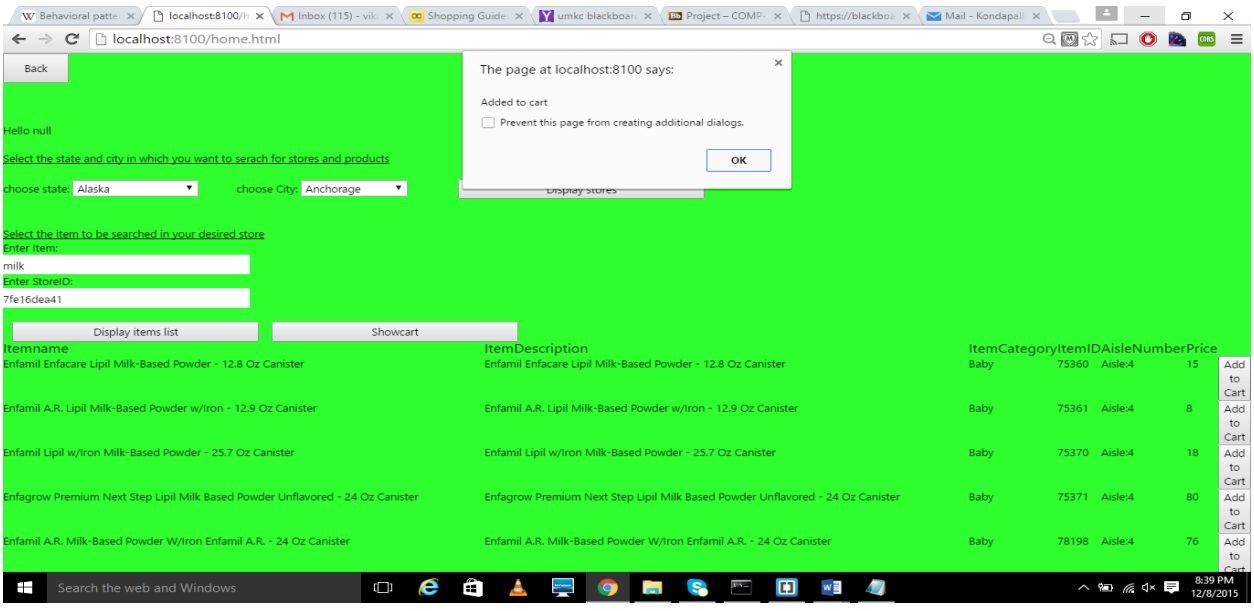
The above screen shots show the various statistics that effected the performance of the application.

Unit Testing

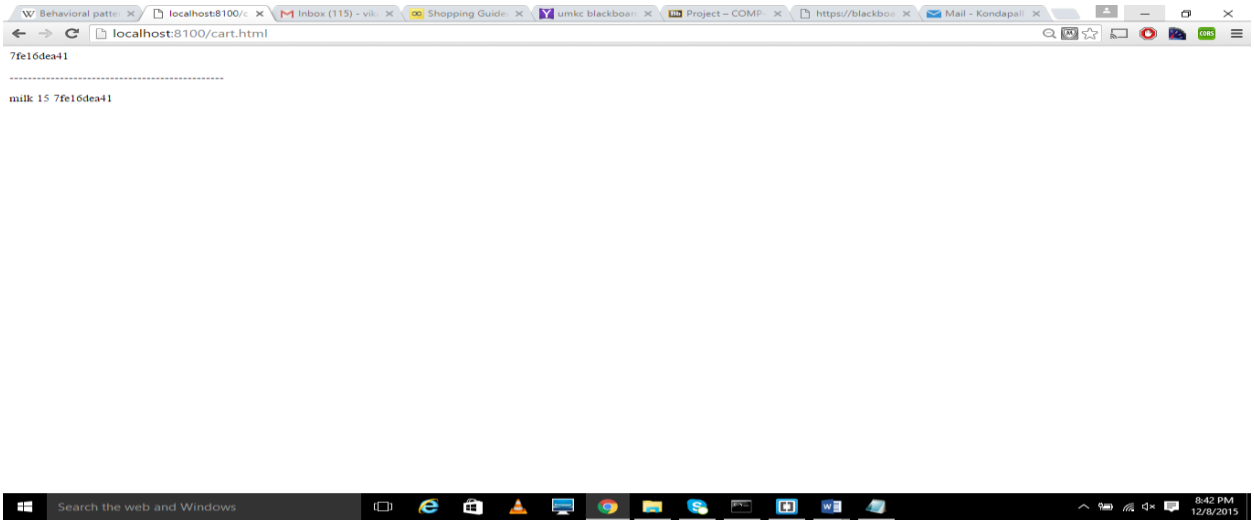
Testing the functionality of the Cart page and its integration with the Dashboard page



Upon clicking the Add to cart button the corresponding item should be added to the Cart page.



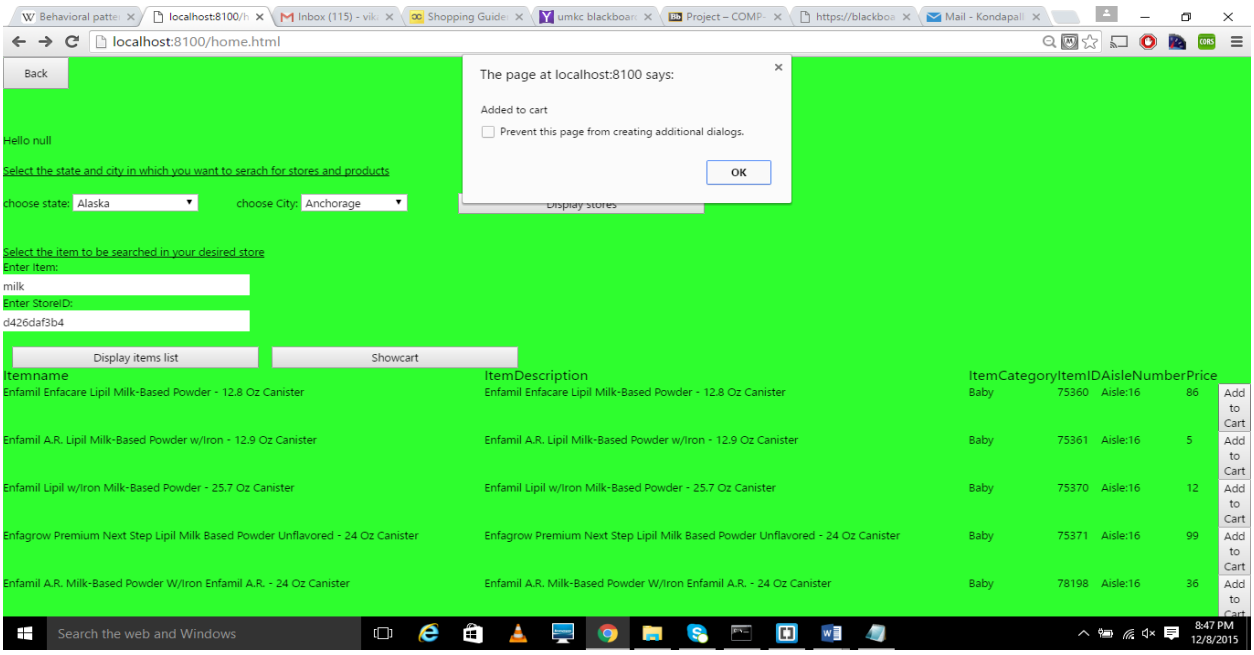
Cart page



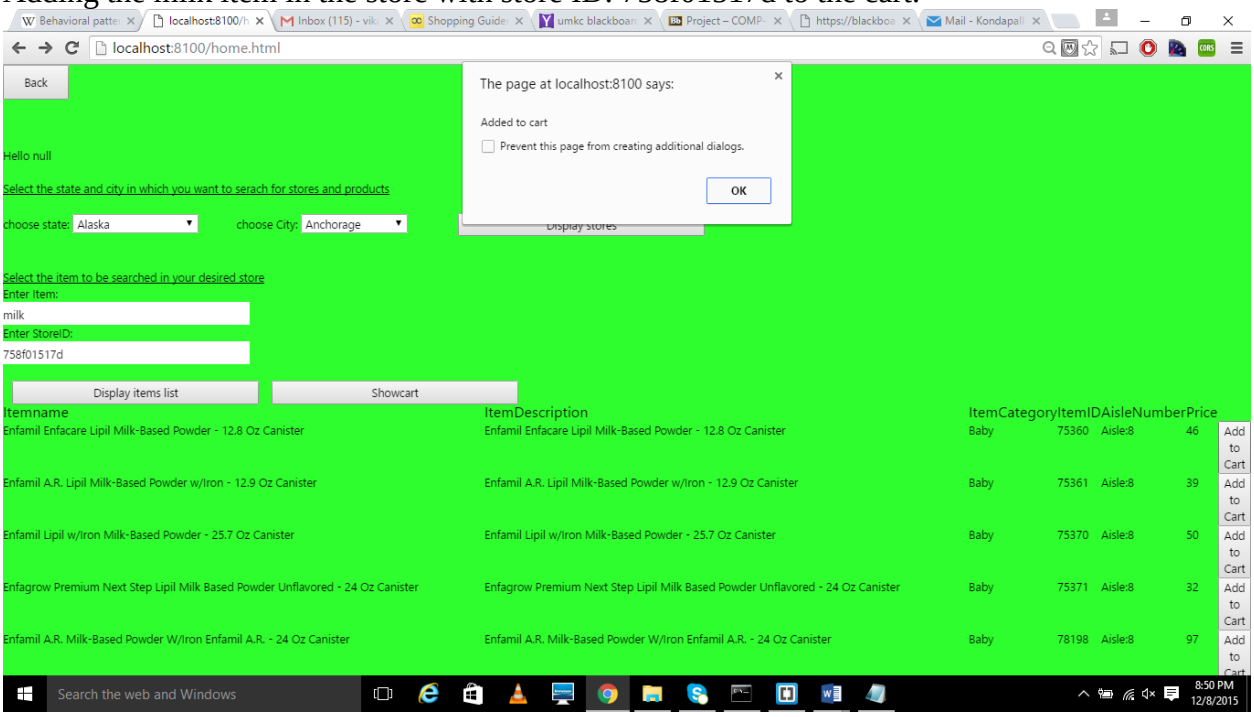
So upon clicking the Add to Cart button item has been added to Cart page. So the adding to cart page has been tested and is working correctly.

If items from multiple stores are added to the cart then the items should be sorted store wise in the cart page.

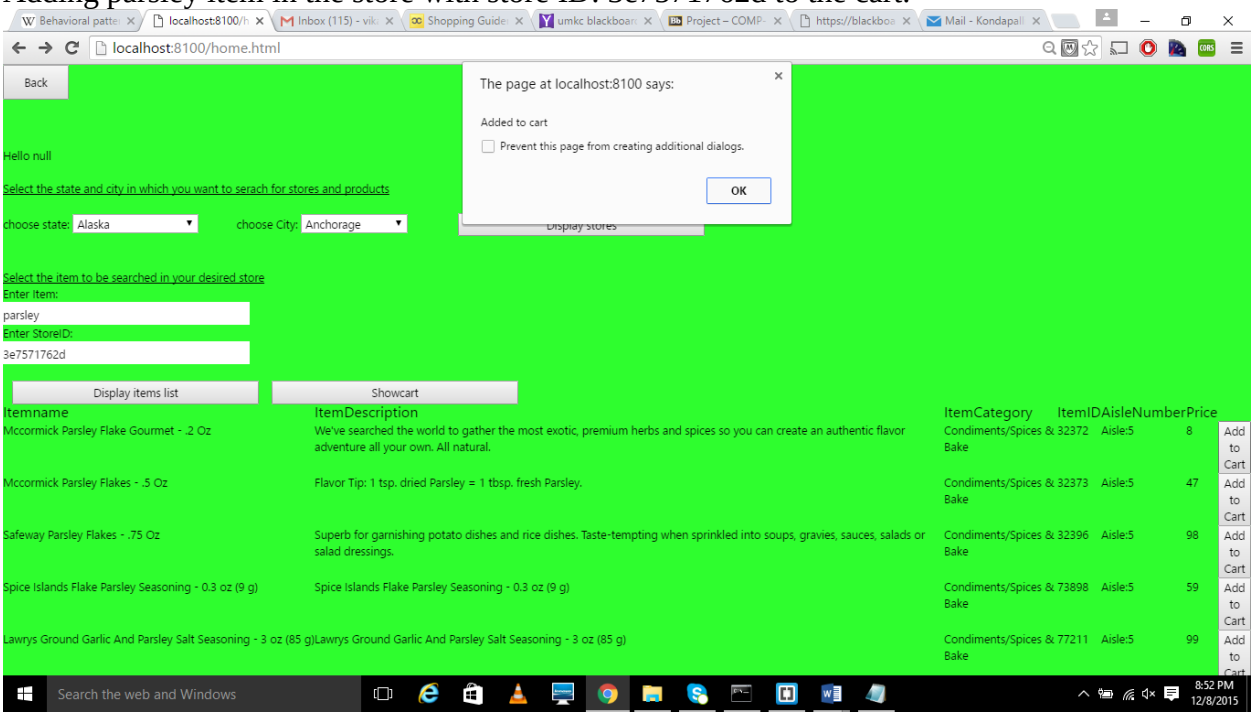
Adding the milk item in the store with store ID d426daf3b4 to the cart.



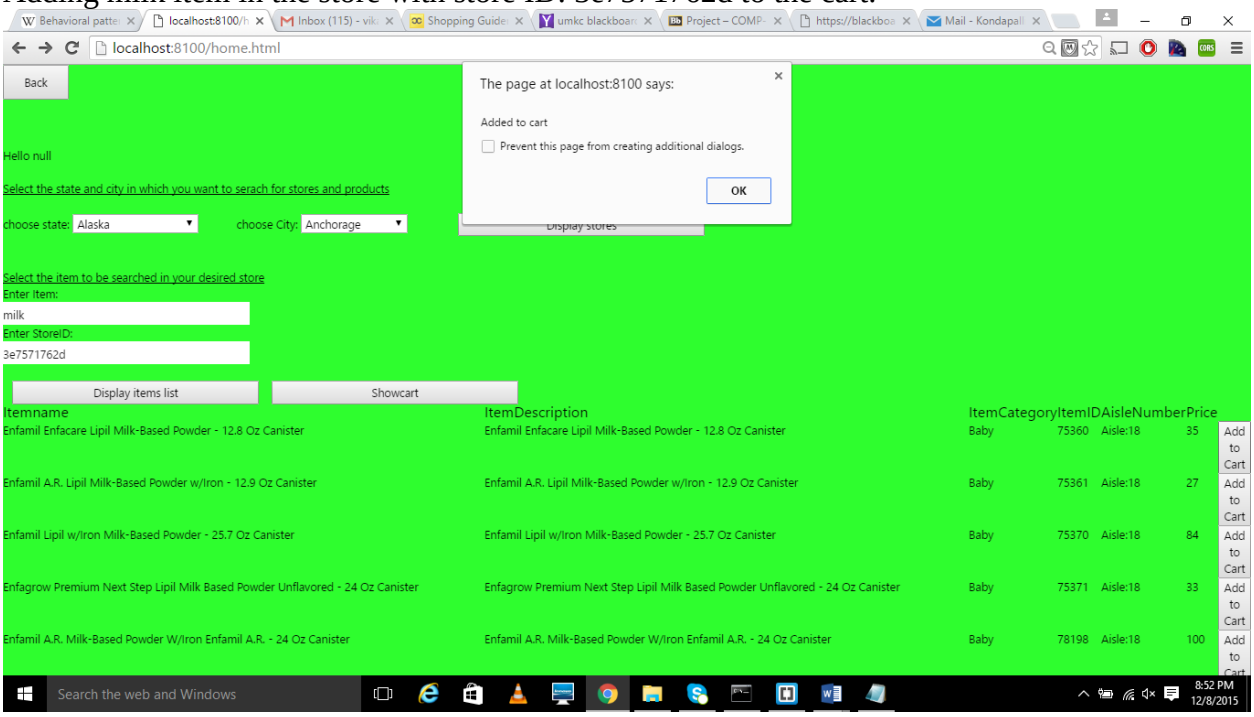
Adding the milk item in the store with store ID: 758f01517d to the cart.



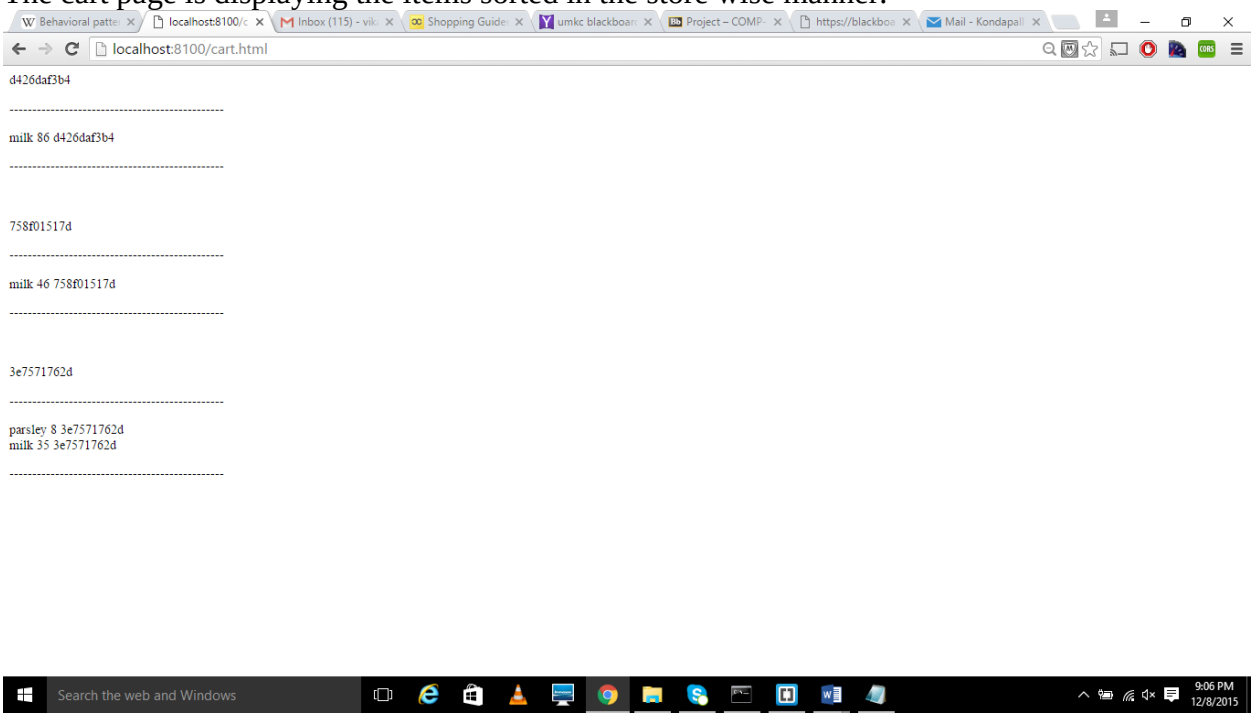
Adding parsley item in the store with store ID: 3e7571762d to the cart.



Adding milk item in the store with store ID: 3e7571762d to the cart.



The cart page is displaying the items sorted in the store wise manner.

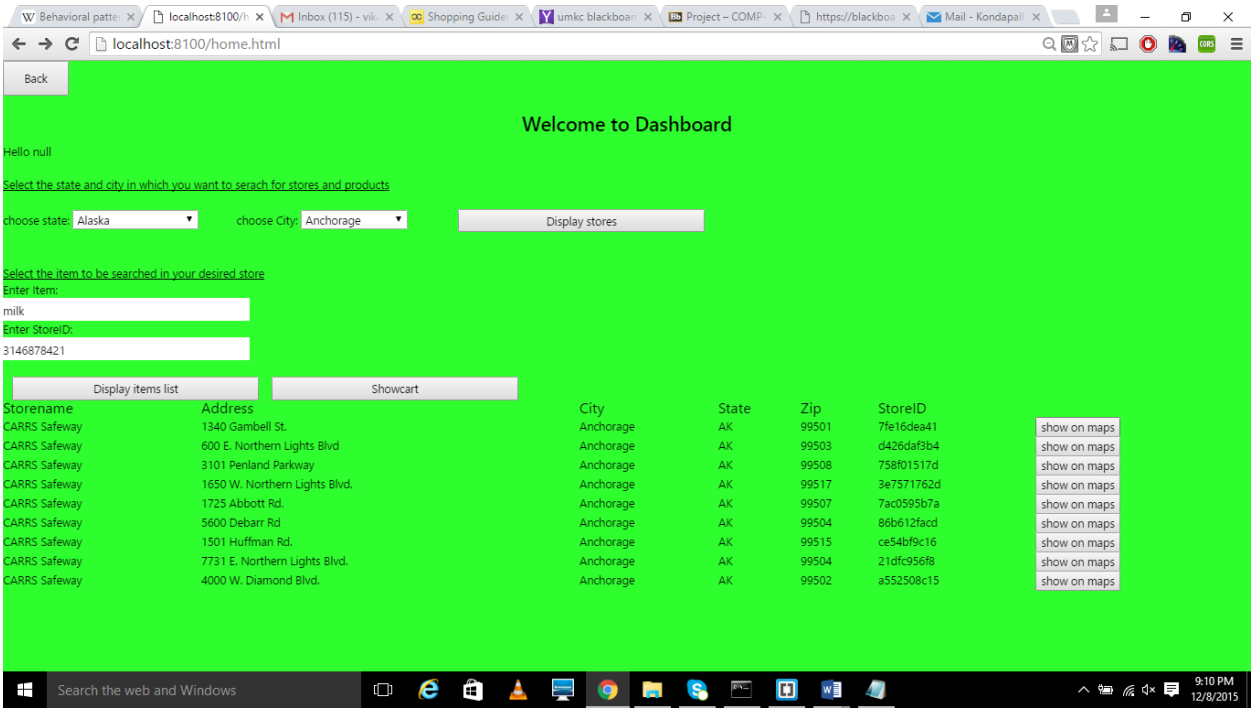


So the adding to cart functionality and store wise sorting of the items is tested and is working correctly.

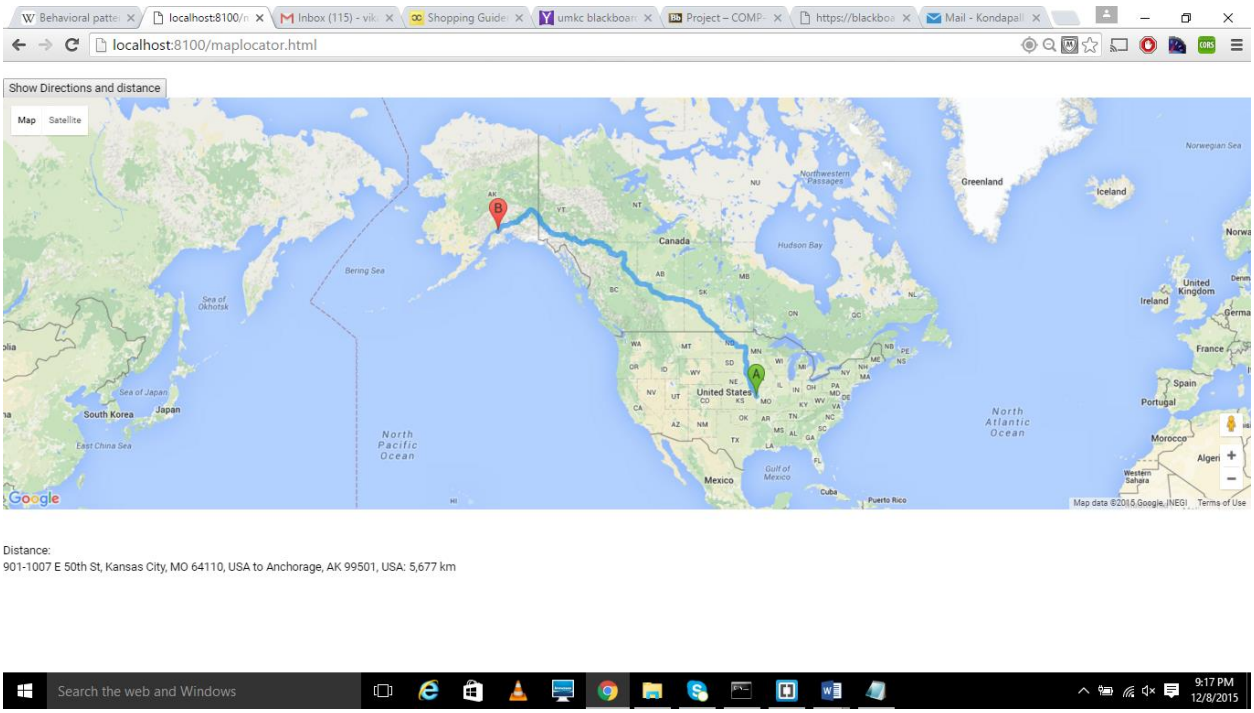
Testing the Map location page functionality

All the stores in the specified location are displayed. Upon clicking the show on Maps corresponding to a particular store only that particular store directions and distance should be displayed in Map locator page.

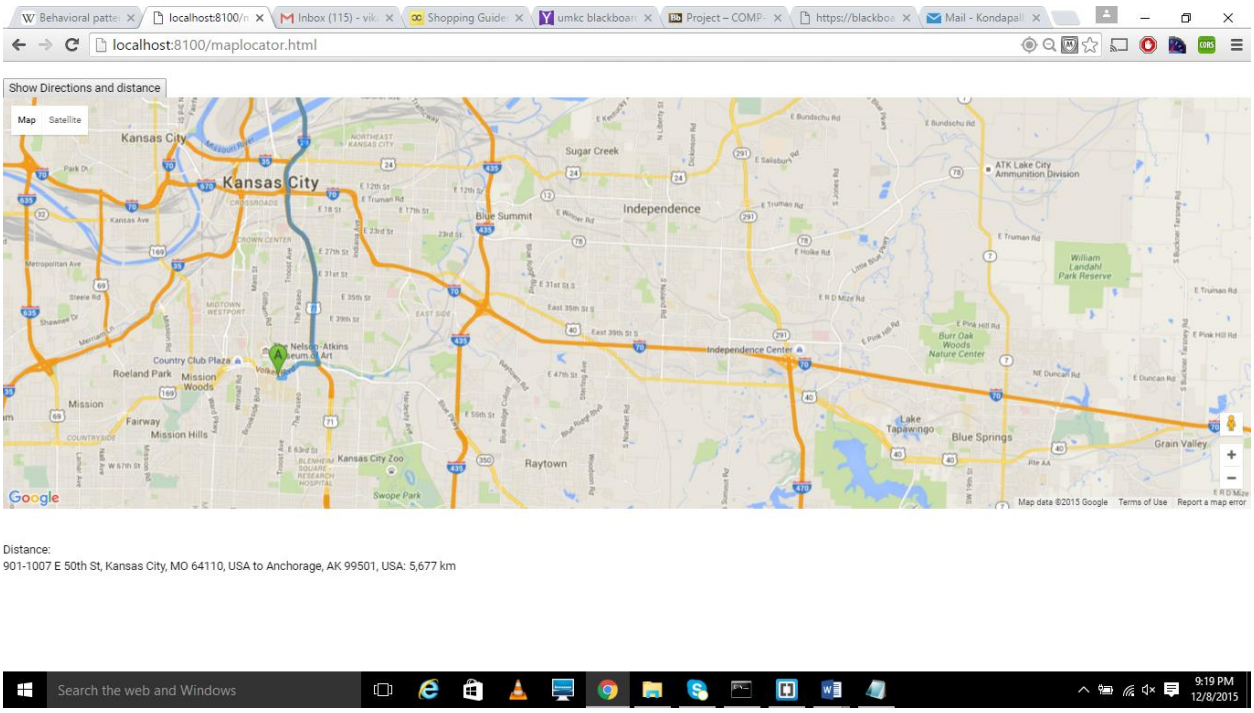
The show on Maps button corresponding to CARS safeway store with store ID: 3146878421 is clicked.



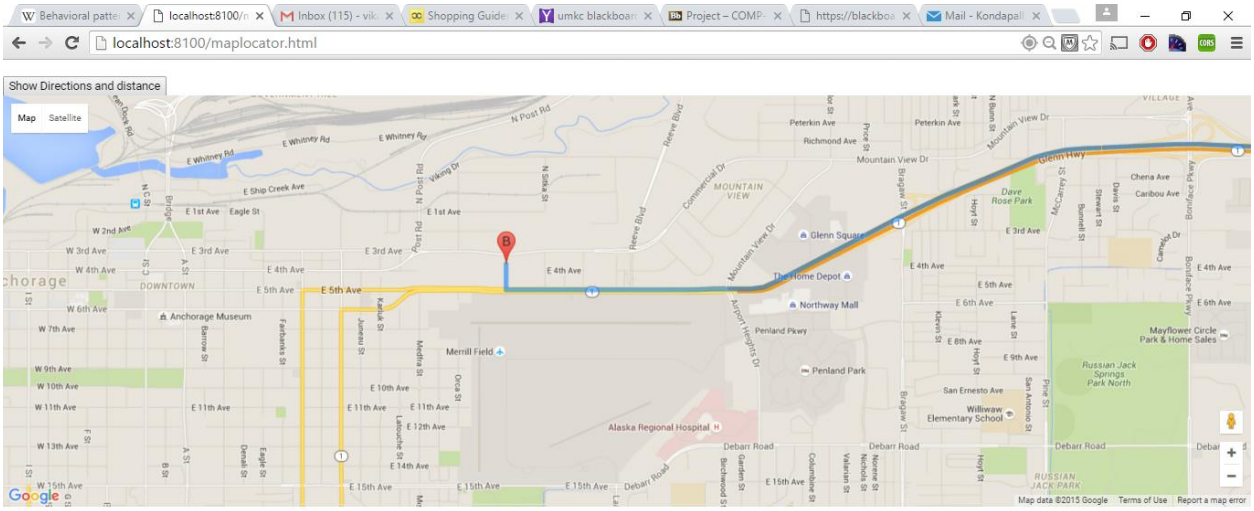
The directions and distance from current user location to the selected store location is displayed.



Current Geo location of the user.



Store exact location



Distance:
901-1007 E 50th St, Kansas City, MO 64110, USA to Anchorage, AK 99501, USA: 5,677 km

Implementation

Server Side Implementation

In this Application the server side implementation is mostly dealing with APIs. The APIs we are using in this project are residing in the remote server. The Application deployed in the client side call the APIs on the server side, then the API on the server side responds to the request and processes the request according to the manner it was specified in this request. This application uses the APIs as the REST services to attain the required functionality. So APIs processing the request received by the client is the main server side implementation of this application.The APIs server side implementation in this project are “supermarket” API, “MongoDB” API and the “Google Maps” API.

MongoDB API is used to connect to the Mongo Lab Database existing in the remote location. This Mongo Lab Database is used to store the user details upon registration and also fetch the details of the user for validation during the login procedure. The MongoDB store the data in the form of key-value pairs, where for each of the user one document is created where all his details like username, password, mobile number, Email ID are stored in the form of key value pairs. This MongoDB is accessed from our Application using the Mongo DB drivers. All the objects required for the data fetching and data storing are sent by the server to the client.

Supermarket API is used to fetch the details of various stores in the specified location (state, city) and all the required items with in that store. Supermarket API stores all the stores, items details in the XML format. Client calls this API using the XMLHTTPREQUEST() method where the URL of the required page holding the required item or store details are residing. The Supermarket API residing on the server side responds to the request made by the client getting the required details in XML format and sending those details to the client as an XML object.

Google Maps API is used for getting the user current Geo Location, location of the store and locating both of them on the Google Maps. Google Maps API on the server side responds to the request made by the client by sending the objects with specified parameters to the client. The client then executes them. So server side implementation here is responding to the calls made by

the client by processing the request in the specified manner and returning the required executable objects to the client. This Google Maps API also returns the distance to be travelled by the user from his current location to the store location

Client Side Implementation

The major part of execution and implementation for this application is done on the client side only. In the client side the various pages fetch data from the APIs and perform the processing of the data to achieve the required functionality. The various pages in this Application are Login page, Registration page, Home page, Cart page, Map Locator page.

Registration Page

The functionality of the Registration page is mostly of the previous increment only, except that the GUI of the page has been made more user friendly. The Registration page accepts the required details from the user and registers him in the Database. The Registration page uses the MongoDB API to connect to the Mongo Lab Database in the remote location for this purpose. Then the user becomes authorized to start accessing this application.

Login page

The Login page also has almost of the same functionality of the previous increment, but the GUI of the page has been made more user friendly. The Login page asks the user to enter his username and password, then it validates the user for the authorization, if he enters the correct username and password combination then he will be directed to the home page where the core functionality of the Application exists.

Home page (stores, items search)

In the Home page, the Application allow the user to select the State and the city in the USA in which he/she wants to search for the Stores. Then the stores available in the State, city selected by the user are displayed as a list. In this list all stores along with complete address are displayed. This details of this list are displayed by using the SuperMarket API. In this list there

also a button called Show on Maps against each store which upon clicking which the exact store location is displayed on the Google Maps. In Map the route from the user current location to the Store location is rendered and also the distance to be travelled by the user is displayed.
For this functionality of getting the user current location and locating store on the Maps Google Maps API has been used.

In this list the user selects any one particular store and enters the required item to be searched in that particular store. Then all the related items available in that particular store are displayed as a list. In this list there is button called add to cart against each of the item in the list upon clicking the corresponding item will be added into the user cart.

Cart Page

The Cart page consists of the list of items selected by the user along their prices. The items are sorted in a Store wise manner. So all the items selected by a user in a particular store are of one segment, and all the items selected by the user in another store are of another segment and so on. This allows the user to make comparison of prices of various items he has selected in each of the store.

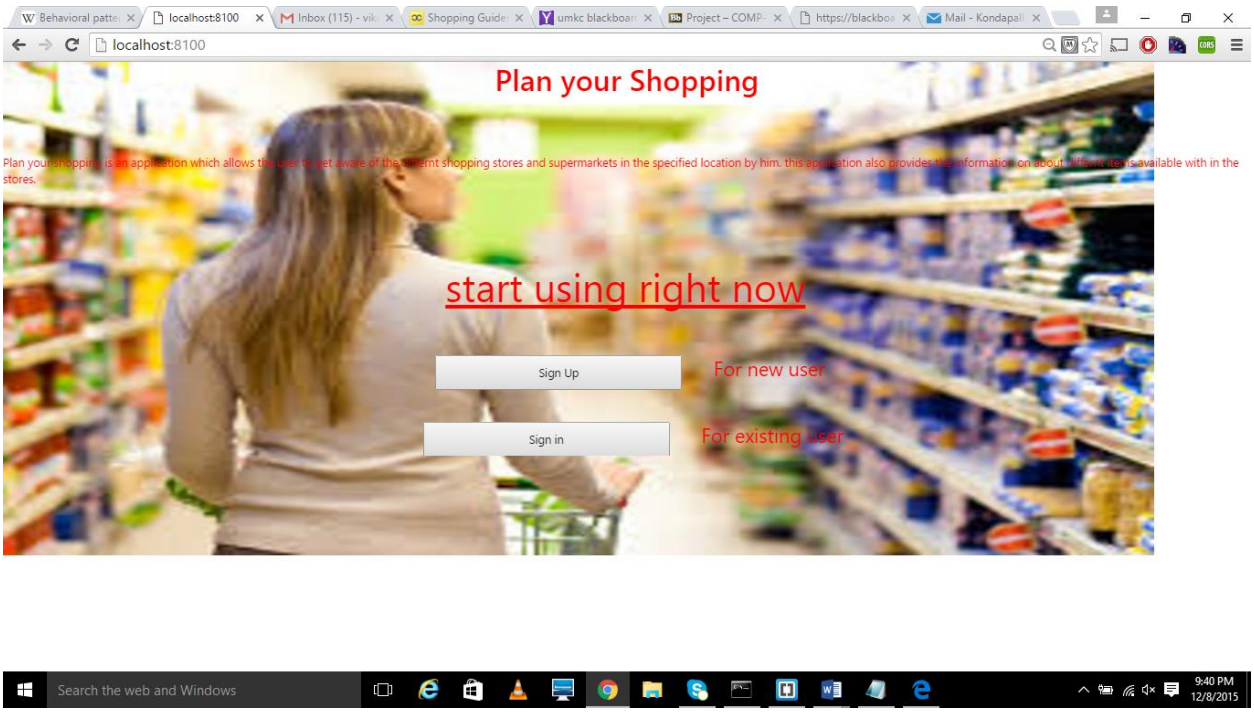
Map Location page

Map location page detects the user current Geolocation and renders the route to be travelled from his current location to the selected store location. In this increment the exact location of the store along with complete address is located on the Map. So the two end points of the route on the Map are user current Geo Location and the store exact location. And also the distance to be travelled by the user to reach the required store using the route is displayed. Google Maps API has been used for this purpose.

Screenshots of the Application

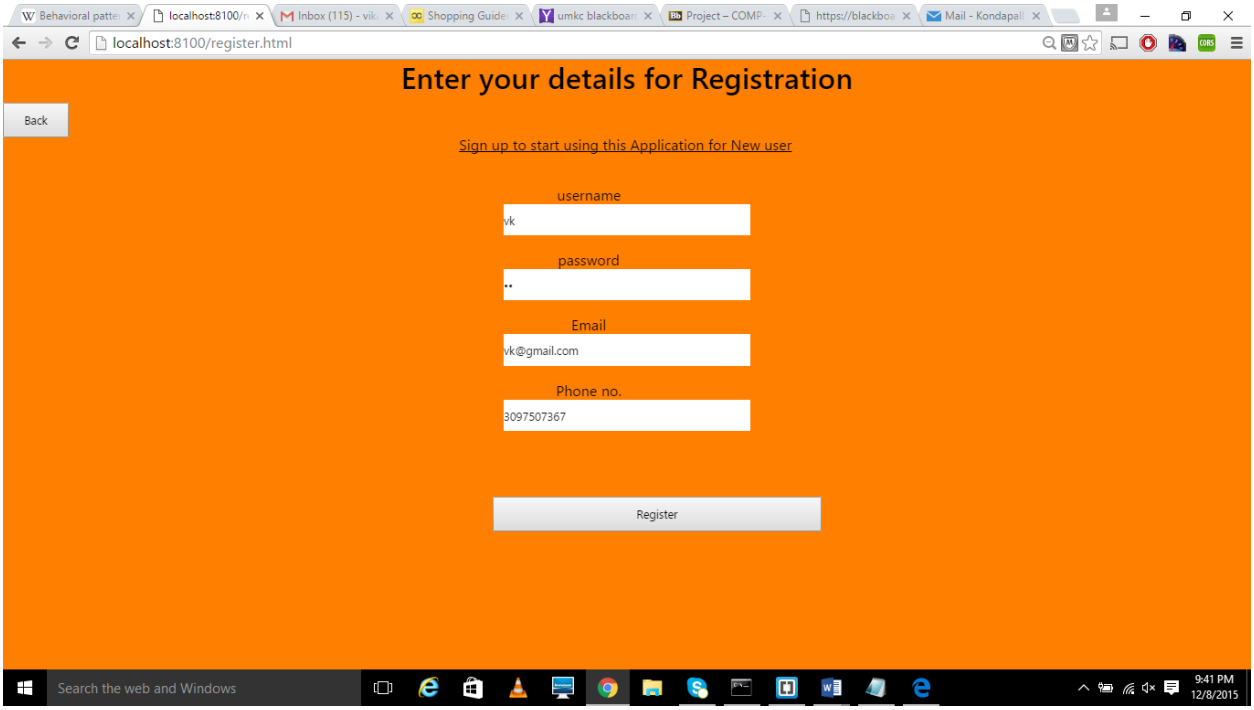
Index page

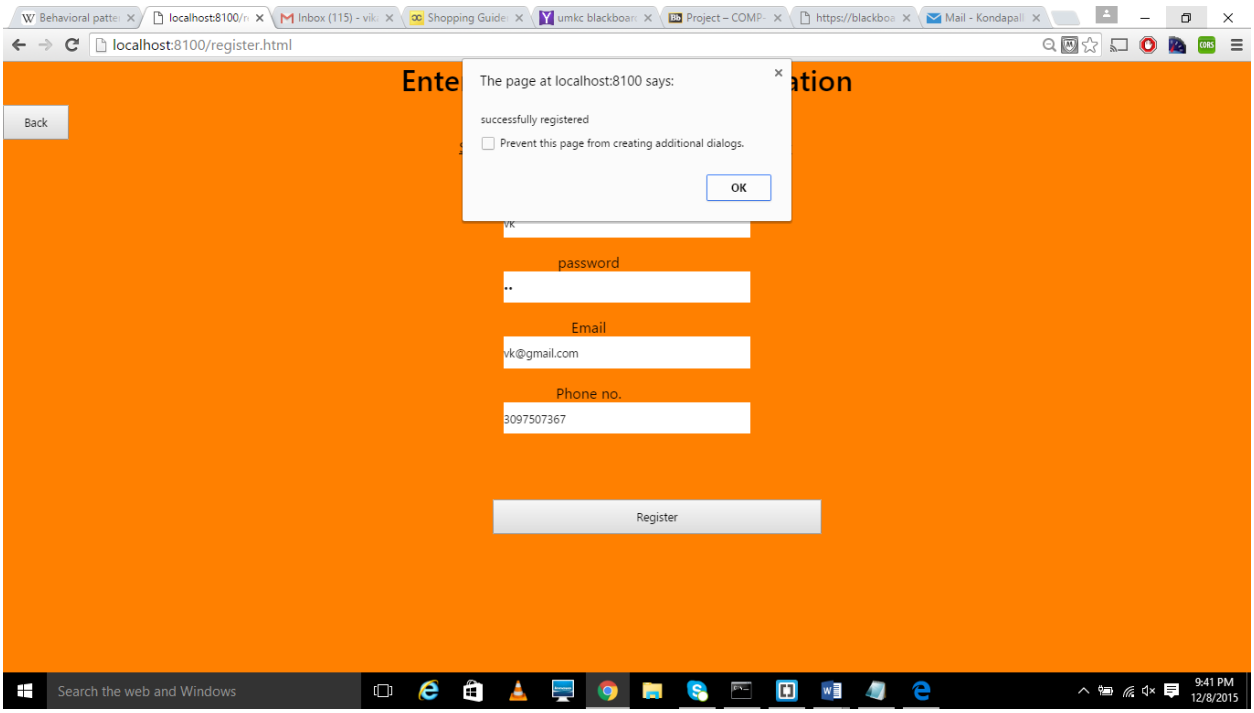
Starting page of the application. It contains basic information about the application and Sign Up, Sign In transitions to Registration and Login pages



Registration page

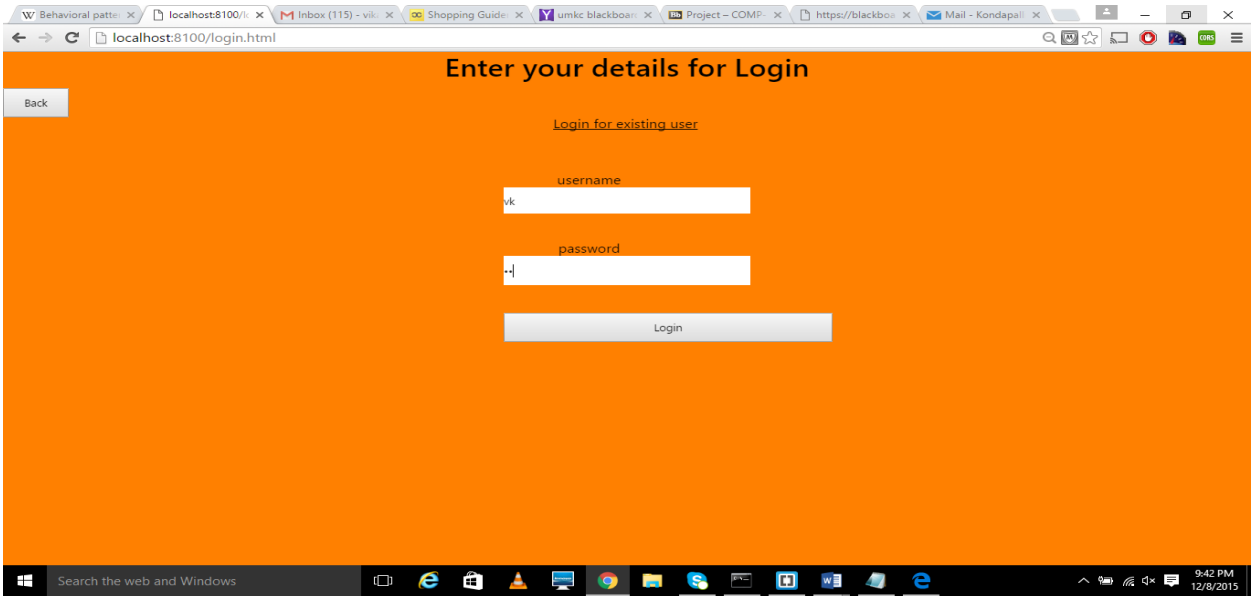
In Registration page user enters his details and gets himself registered in to the Application Database.

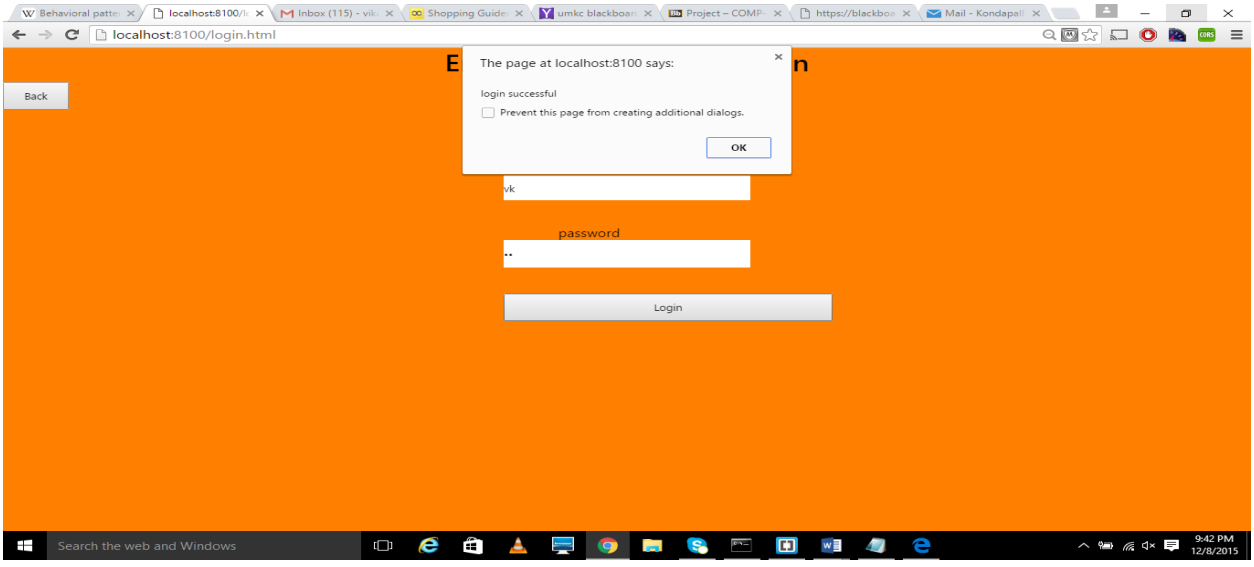




Login page

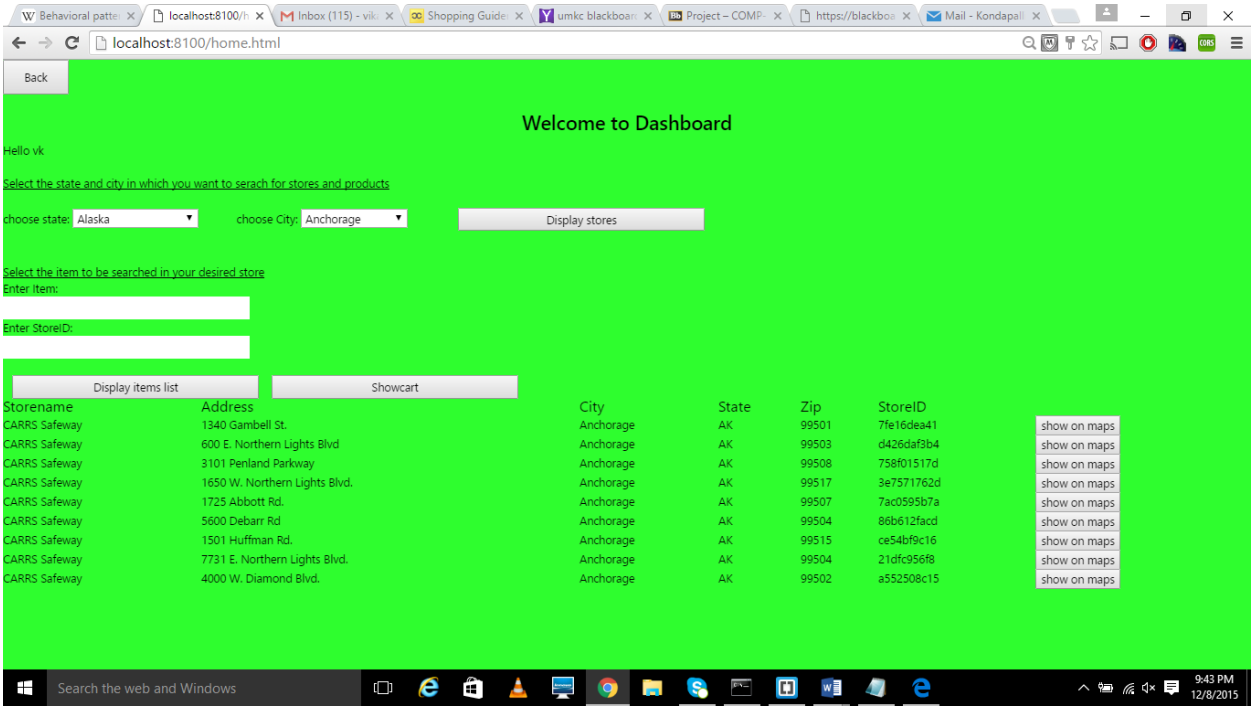
In Login page the user enters his Username and password and clicks login button upon which the entered credentials are validated and if they are found to be correct then the user is welcomed with a login Successful message and redirected to the Home (Dashboard page).



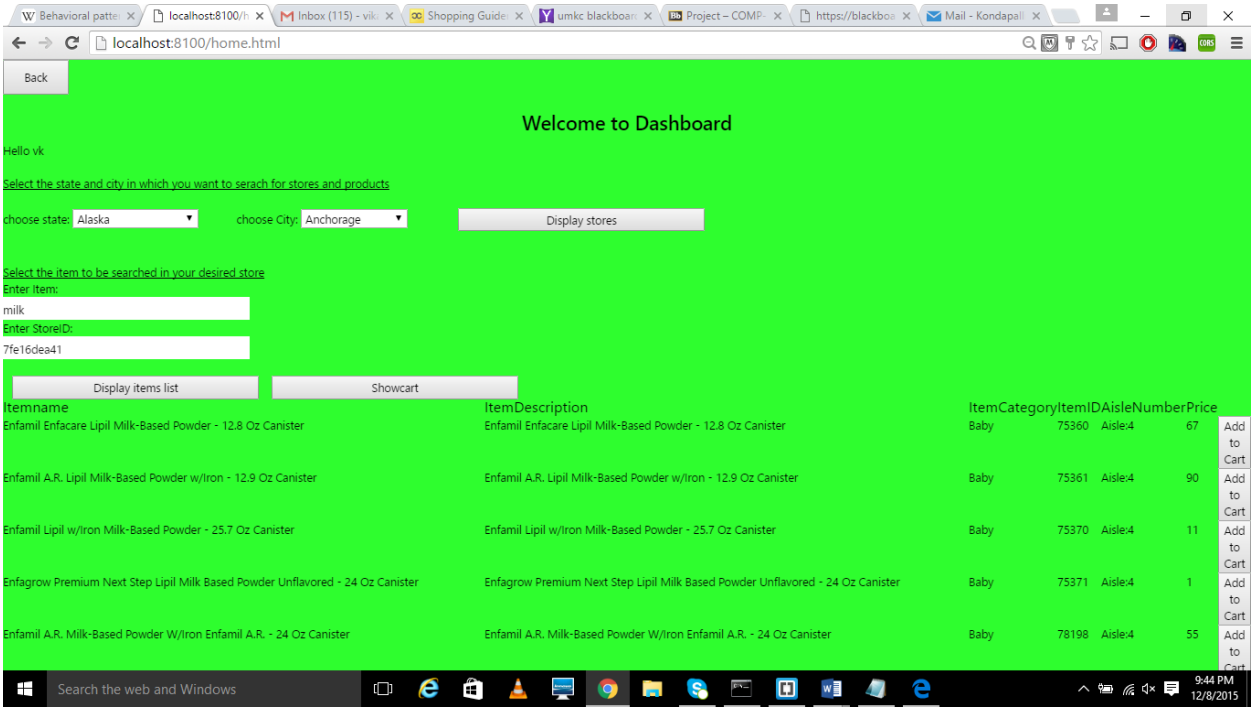


Dashboard page

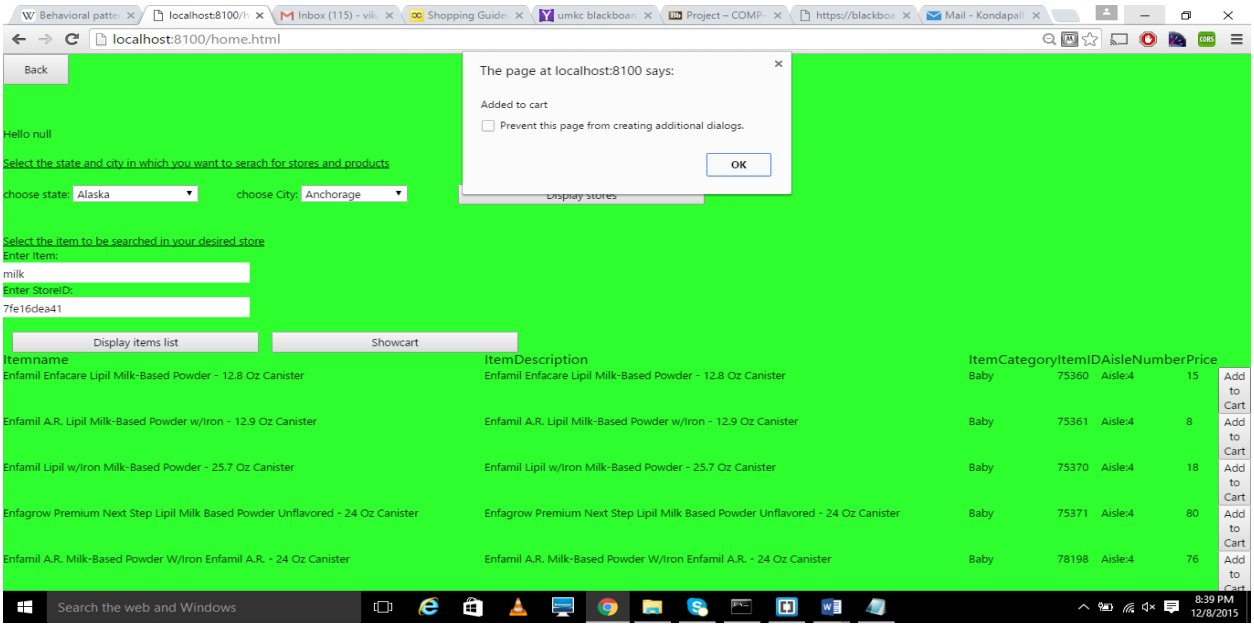
In Dashboard page the user will be prompted to selected a state with in USA and a particular city to be searched in that state for list of stores. Upon specifying these details and clicking on the Display stores button, list of stores in that location will be displayed.



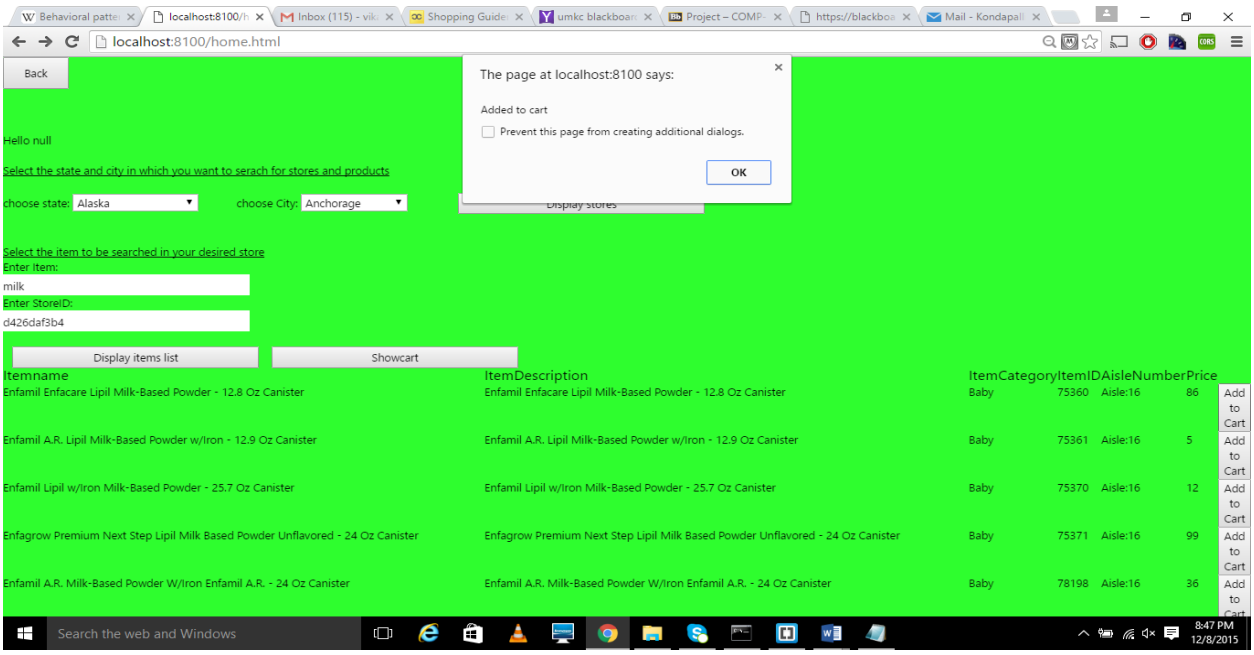
The user then enters an item to be searched in a particular store by entering its store ID and item name and clicks on Display items list button. Then all the related in that store are displayed.

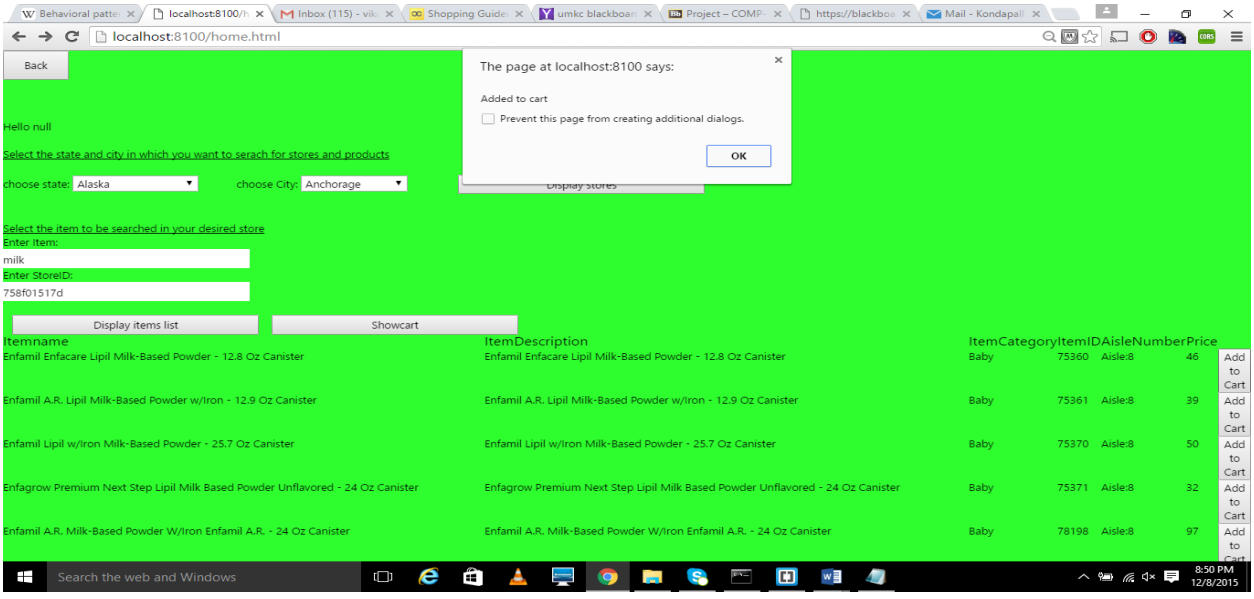


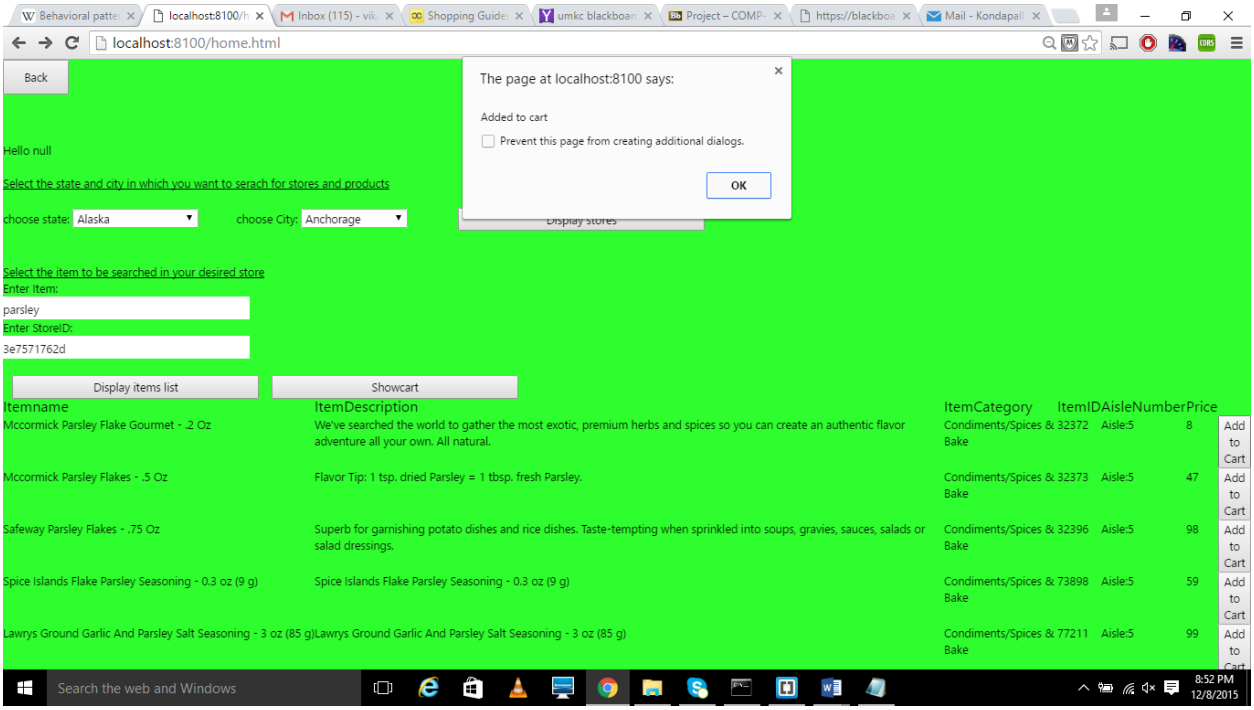
Each item has an Add to Cart button against its record in the list upon clicking which the corresponding item will be added to cart.

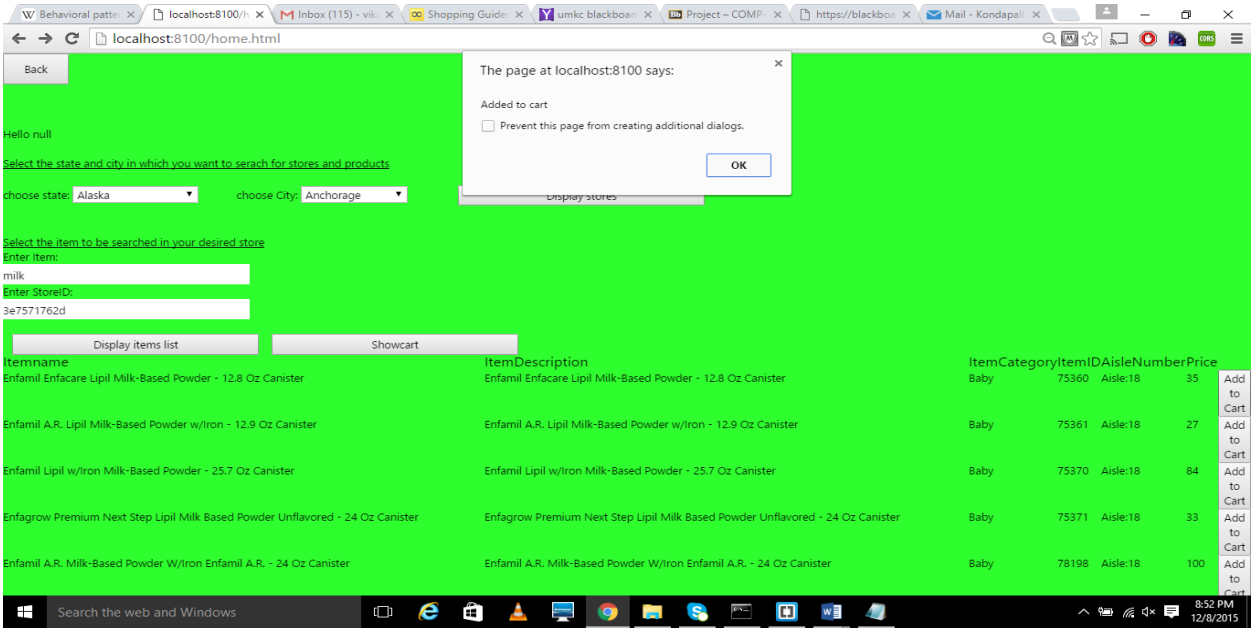


Items from different store are added to the cart



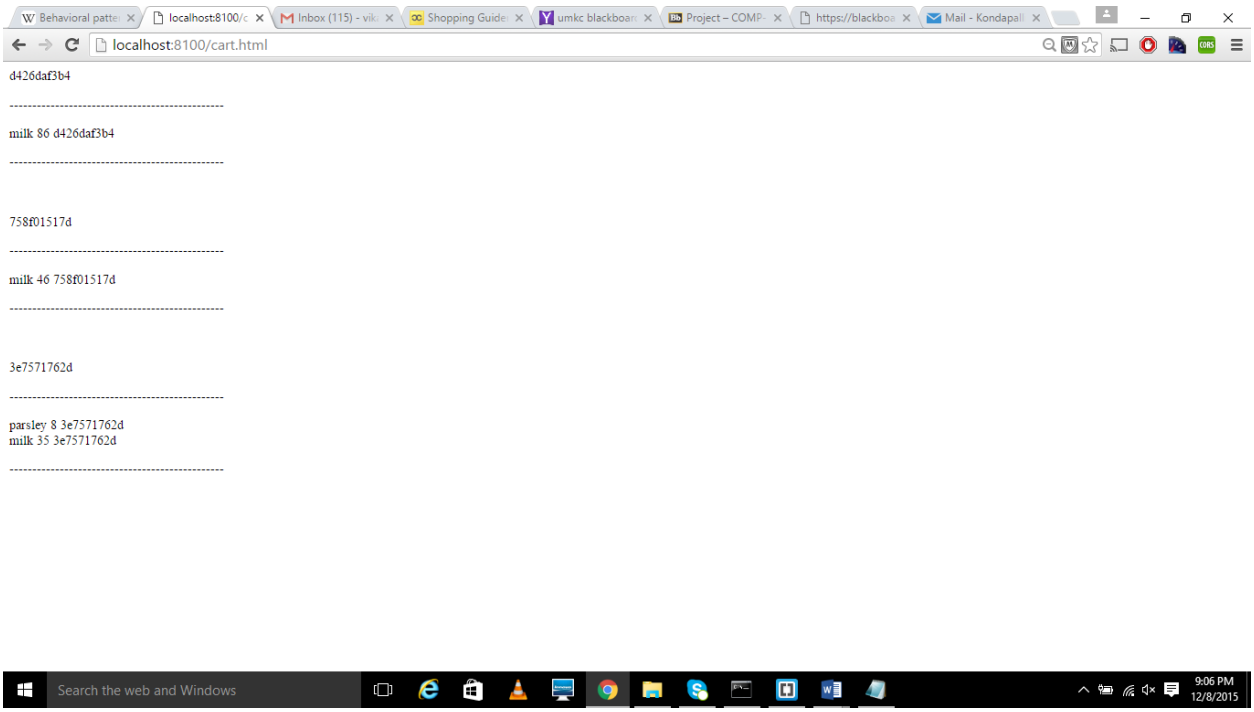



localhost:8100/home.html
Back
The page at localhost:8100 says:
Added to cart
Prevent this page from creating additional dialogs.
OK
Hello null
Select the state and city in which you want to serach for stores and products
choose state: Alaska
choose City: Anchorage
Display stores
Select the item to be searched in your desired store
Enter Item:
parsley
Enter StoreID:
3e7571762d
Display items list
Showcart
Itemname
ItemDescription
ItemCategory
ItemID
AisleNumber
Price
Mccormick Parsley Flake Gourmet - .2 Oz
We've searched the world to gather the most exotic, premium herbs and spices so you can create an authentic flavor adventure all your own. All natural.
Condiments/Spices & Bake
32372
Aisle:5
8
Add to Cart
Mccormick Parsley Flakes - .5 Oz
Flavor Tip: 1 tsp. dried Parsley = 1 tbsp. fresh Parsley.
Condiments/Spices & Bake
32373
Aisle:5
47
Add to Cart
Safeway Parsley Flakes - .75 Oz
Superb for garnishing potato dishes and rice dishes. Taste-tempting when sprinkled into soups, gravies, sauces, salads or salad dressings.
Condiments/Spices & Bake
32396
Aisle:5
98
Add to Cart
Spice Islands Flake Parsley Seasoning - 0.3 oz (9 g)
Spice Islands Flake Parsley Seasoning - 0.3 oz (9 g)
Condiments/Spices & Bake
73898
Aisle:5
59
Add to Cart
Lawrys Ground Garlic And Parsley Salt Seasoning - 3 oz (85 g)
Lawrys Ground Garlic And Parsley Salt Seasoning - 3 oz (85 g)
Condiments/Spices & Bake
77211
Aisle:5
99
Add to Cart
Search the web and Windows
8:52 PM
12/8/2015


localhost:8100/home.html
Back
The page at localhost:8100 says:
Added to cart
Prevent this page from creating additional dialogs.
OK
Hello null
Select the state and city in which you want to serach for stores and products
choose state: Alaska
choose City: Anchorage
Display stores
Select the item to be searched in your desired store
Enter Item:
milk
Enter StoreID:
3e7571762d
Display items list
Showcart
Itemname
ItemDescription
ItemCategory
ItemID
AisleNumber
Price
Enfamil Enfacare Lipil Milk-Based Powder - 12.8 Oz Canister
Enfamil Enfacare Lipil Milk-Based Powder - 12.8 Oz Canister
Baby
75360
Aisle:18
35
Add to Cart
Enfamil A.R. Lipil Milk-Based Powder w/Iron - 12.9 Oz Canister
Enfamil A.R. Lipil Milk-Based Powder w/Iron - 12.9 Oz Canister
Baby
75361
Aisle:18
27
Add to Cart
Enfamil Lipil w/Iron Milk-Based Powder - 25.7 Oz Canister
Enfamil Lipil w/Iron Milk-Based Powder - 25.7 Oz Canister
Baby
75370
Aisle:18
84
Add to Cart
Enfagrow Premium Next Step Lipil Milk Based Powder Unflavored - 24 Oz Canister
Enfagrow Premium Next Step Lipil Milk Based Powder Unflavored - 24 Oz Canister
Baby
75371
Aisle:18
33
Add to Cart
Enfamil A.R. Milk-Based Powder W/Iron Enfamil A.R. - 24 Oz Canister
Enfamil A.R. Milk-Based Powder W/Iron Enfamil A.R. - 24 Oz Canister
Baby
78198
Aisle:18
100
Add to Cart
Search the web and Windows
8:52 PM
12/8/2015

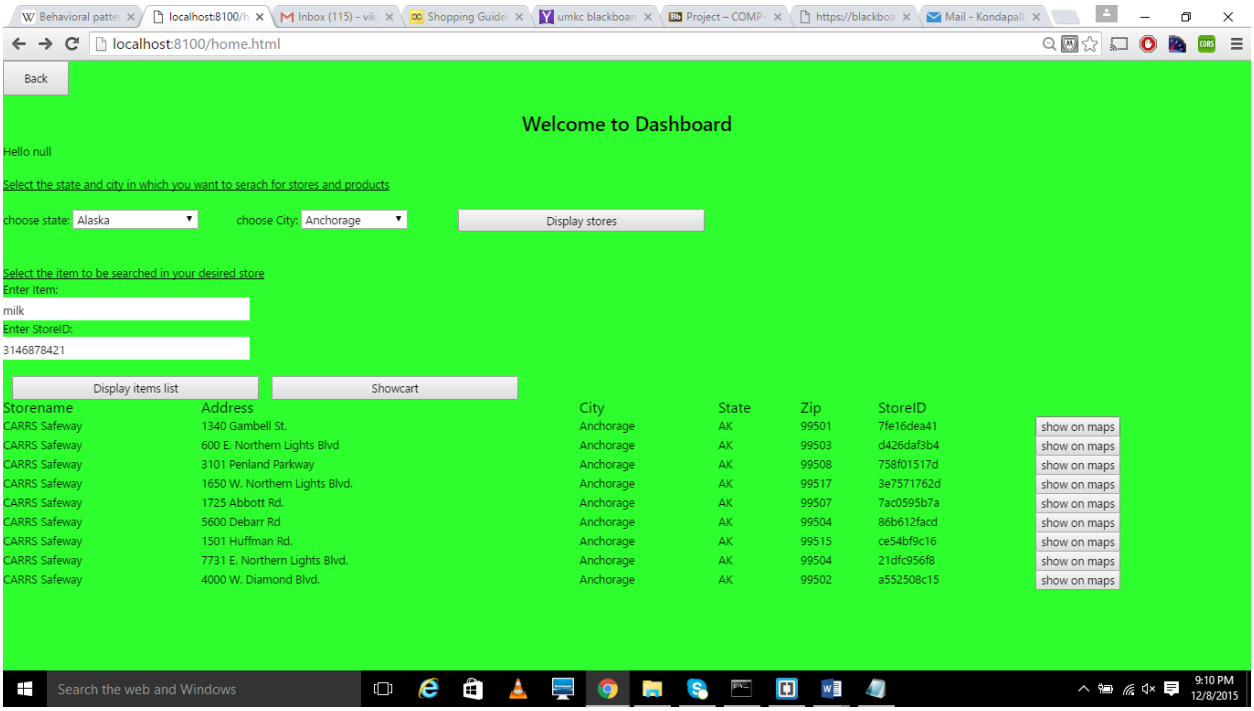
Cart page

All the items which were added to cart in the Dashboard page are displayed in the Cart page in the store wise sorted manner.



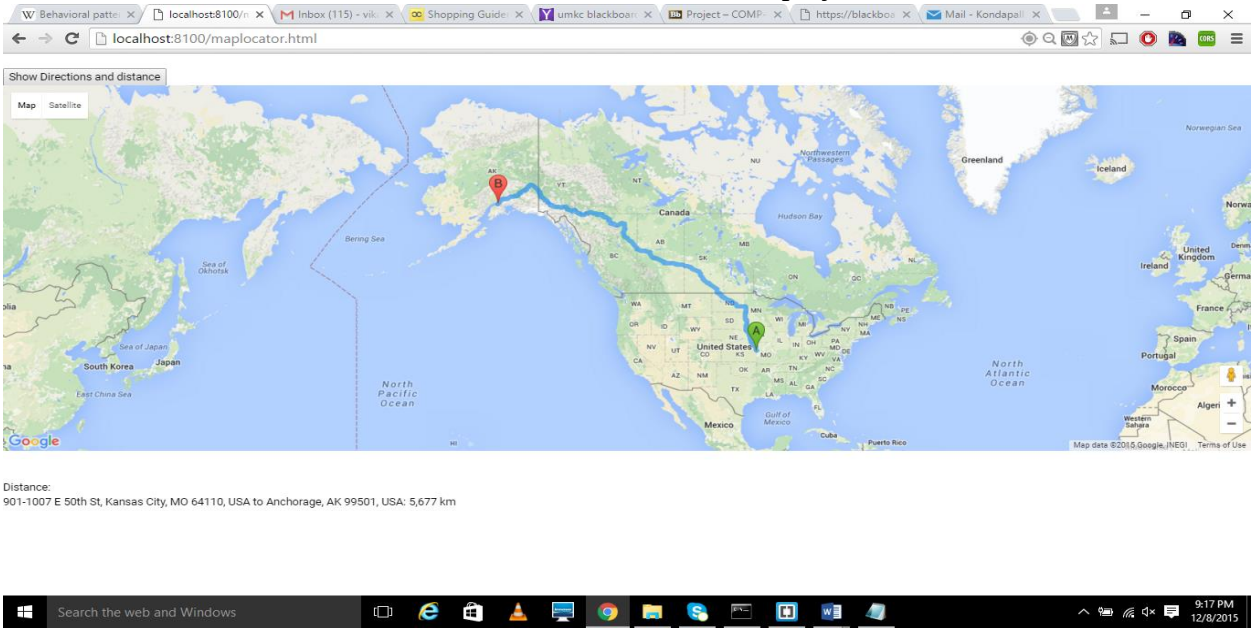
Map location

In the stores list each store has a Show on maps button against its record upon clicking which the directions and distsnce from user current Geo location to the store location are displayed on the Google Maps.



Map location page

User location to store location directions and distance are displayed



Project Management

Project Management for this project has been done by using Kanban tool. The progress up to third increment has been mentioned in the previous report. This is the final increment of the project. The following features have been added to the Application to make it a complete Application. In doing so the following task in the Development and testing have been done.

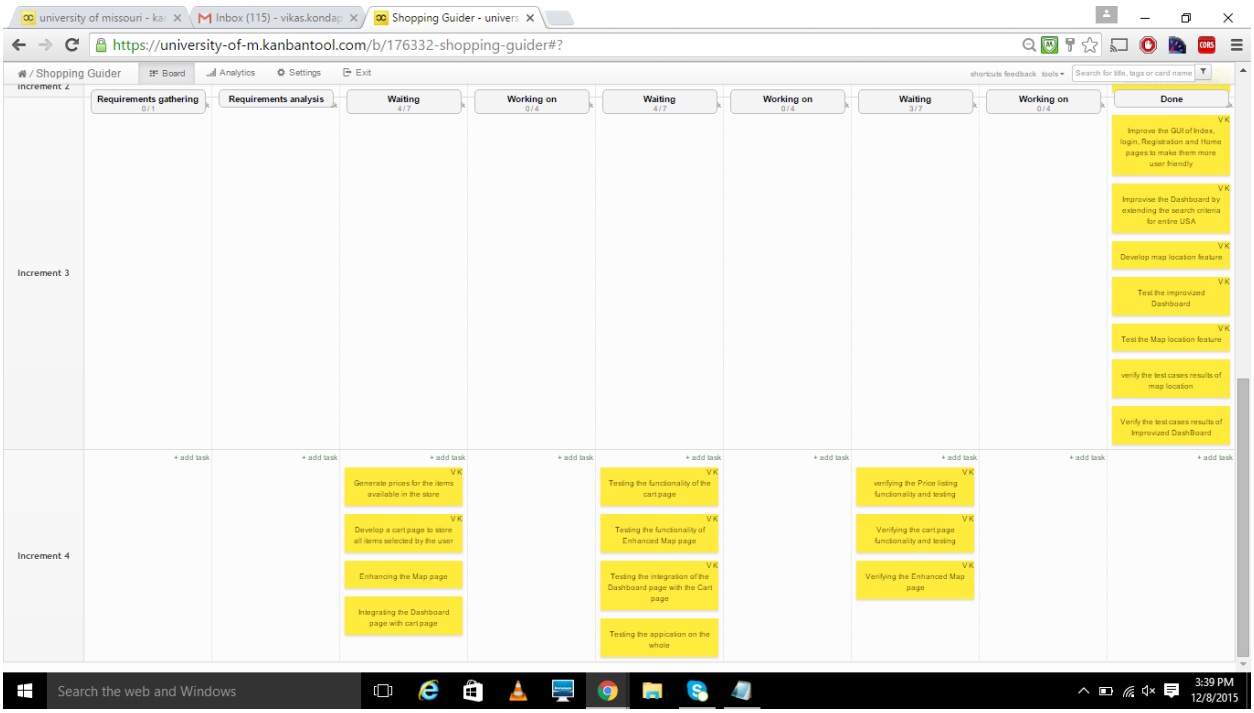
- ## Development

1. Generate prices for the items available in the stores.
2. Develop cart page to store the items selected by the user.
3. Enhancing the Map page to detect the user Geo location and render the route to store exact location on the map page and also display the distance to be travelled.
4. Integrating the Dashboard page with the Cart page

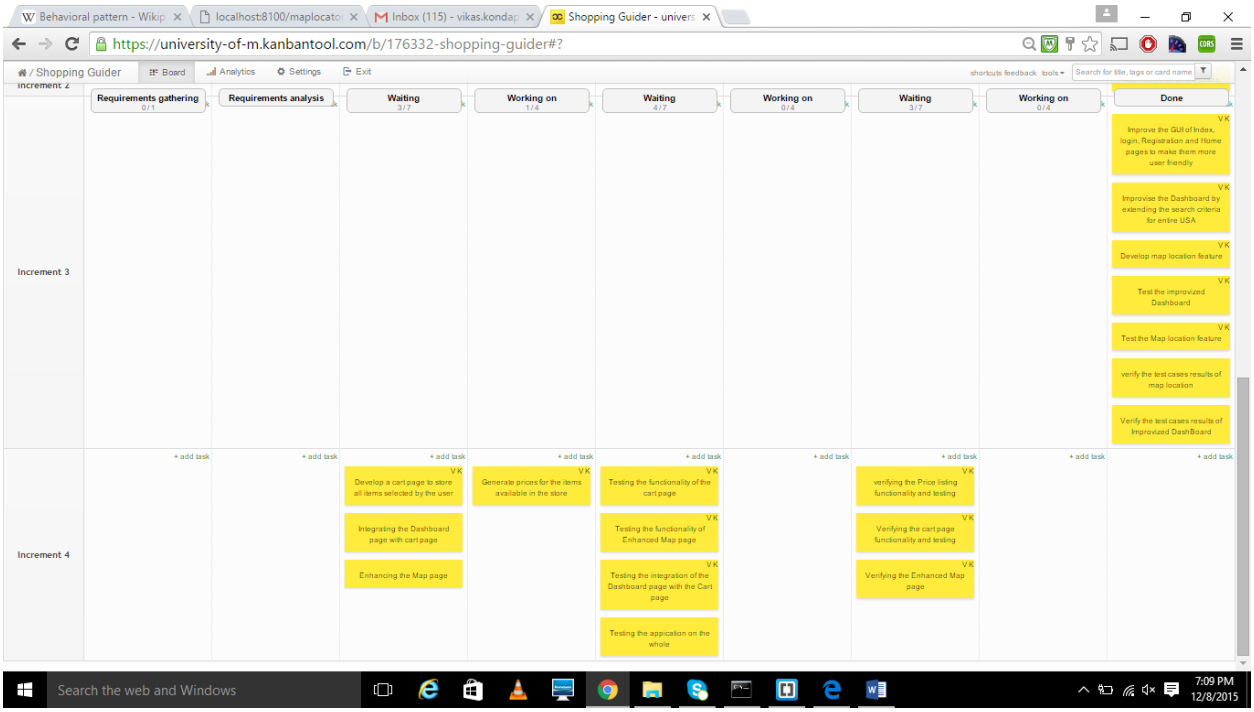
- ## Testing

1. Testing the functionality of the Enhanced Map Page.
2. Testing the functionality of the Cart page.
3. Testing the integration of the Cart page with the Dashboard page (Home page).
4. Testing the Application on the whole.

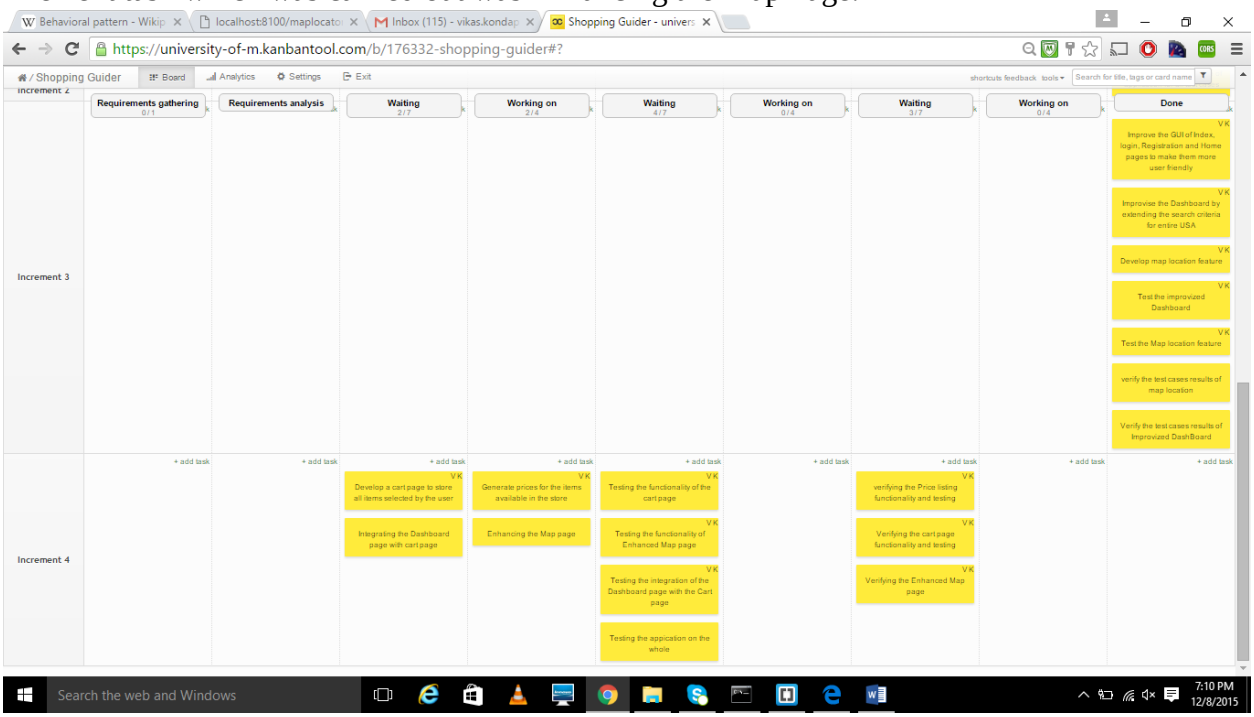
Initially all the tasks were in the waiting phase.



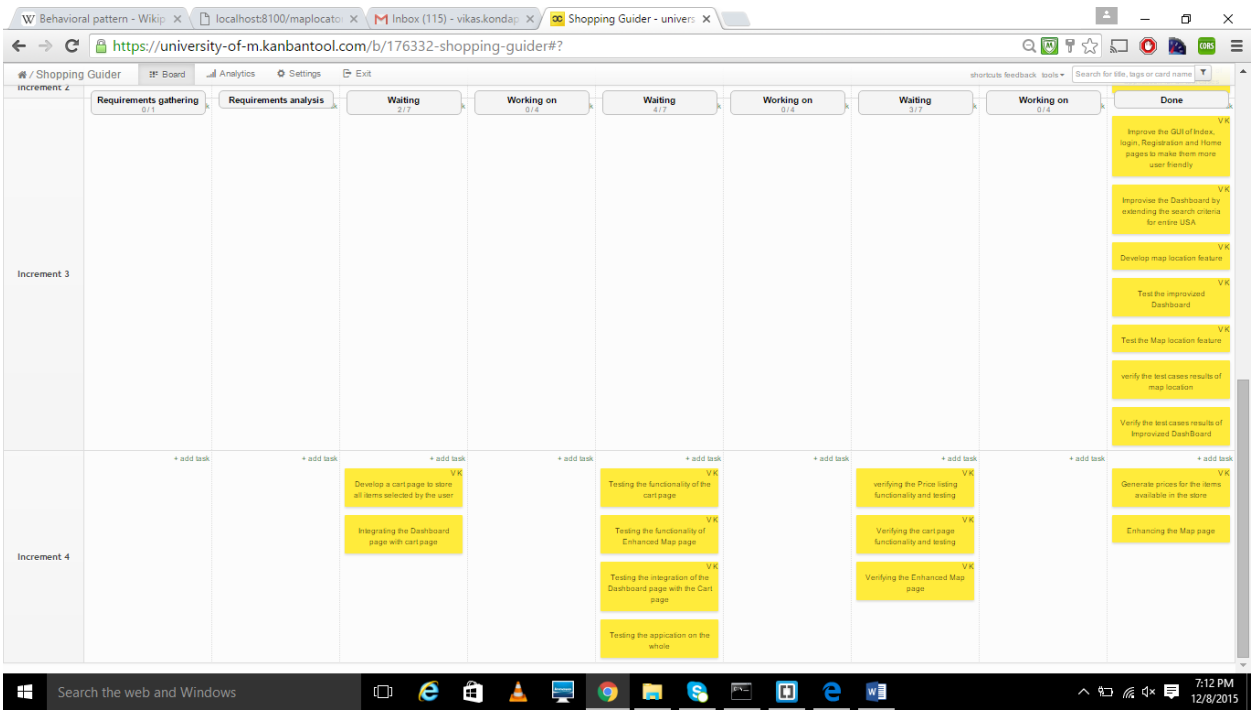
The first task which was carried was generating the prices for each of the item available with in the store.



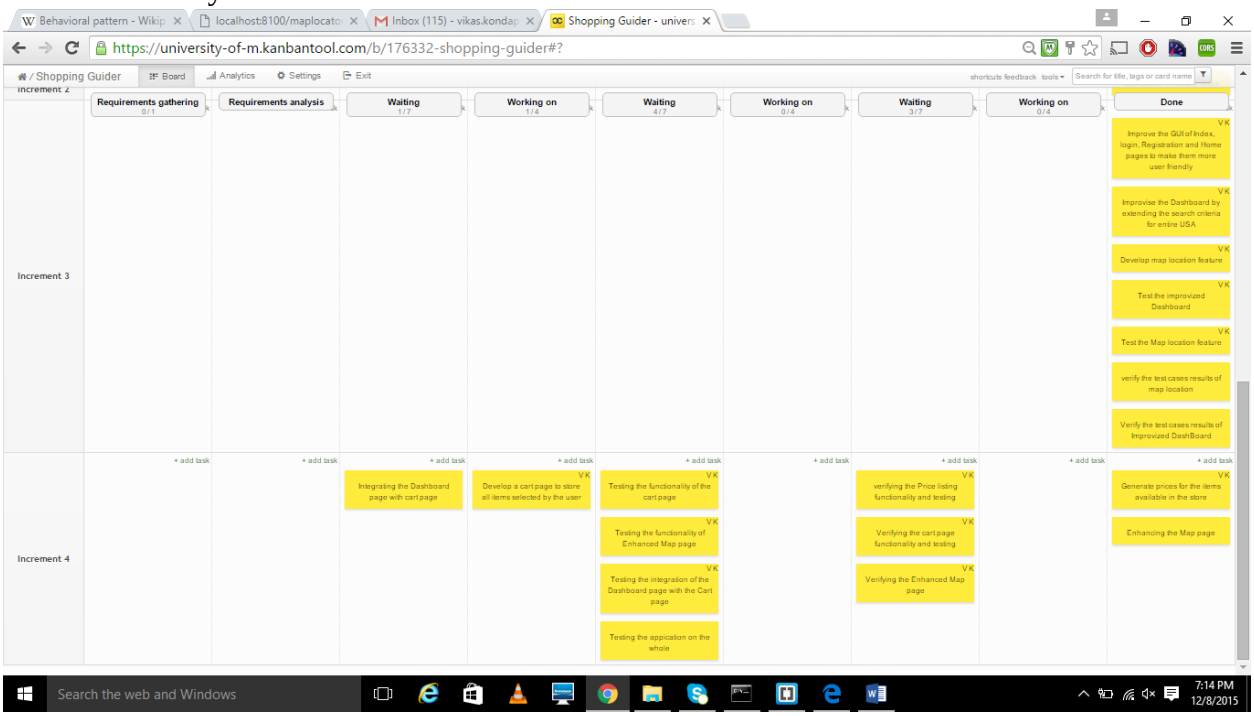
The next task which was carried out was Enhancing the Map Page.



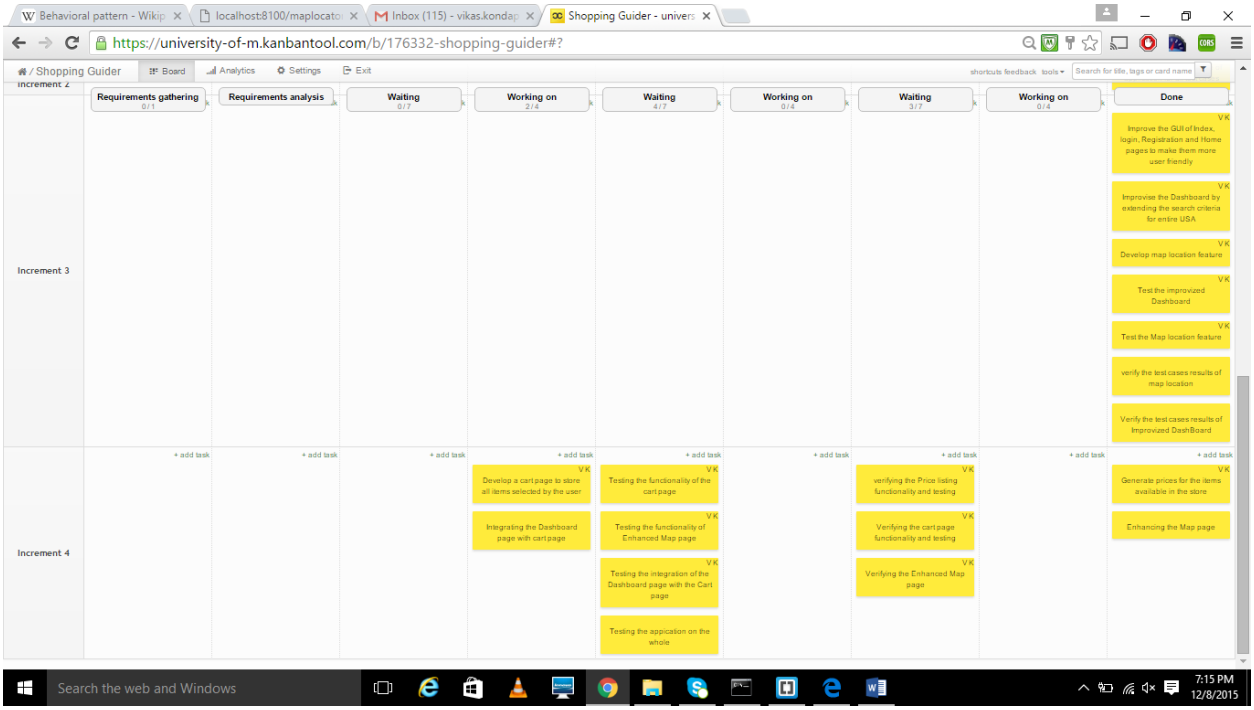
Both the above tasks were successfully completed and moved to done phase.



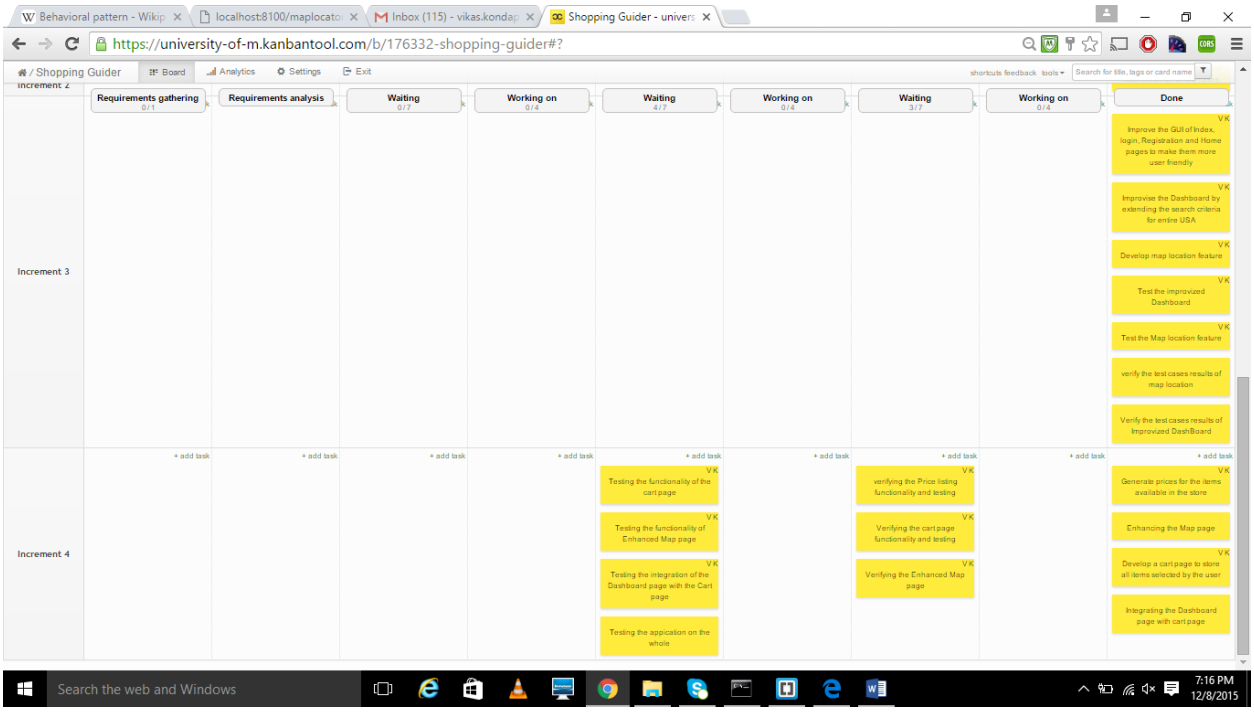
The next task which was carried out was developing a Cart page to store the details of the items selected by the user.



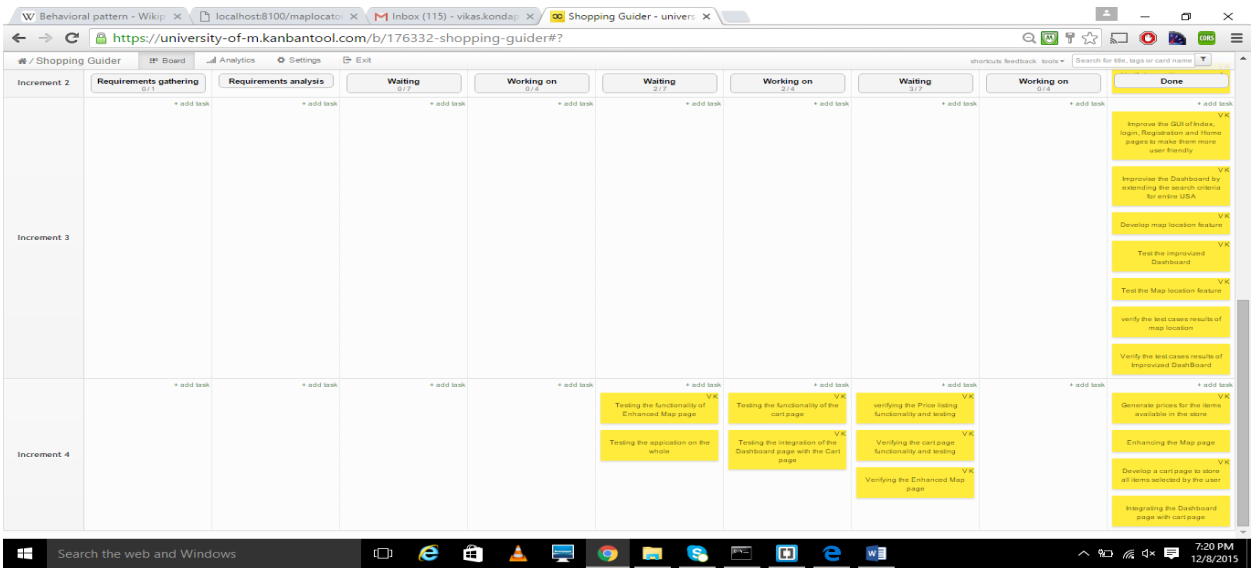
The associated task with the development of Cart page was integrating the Cart page with the Dashboard page.



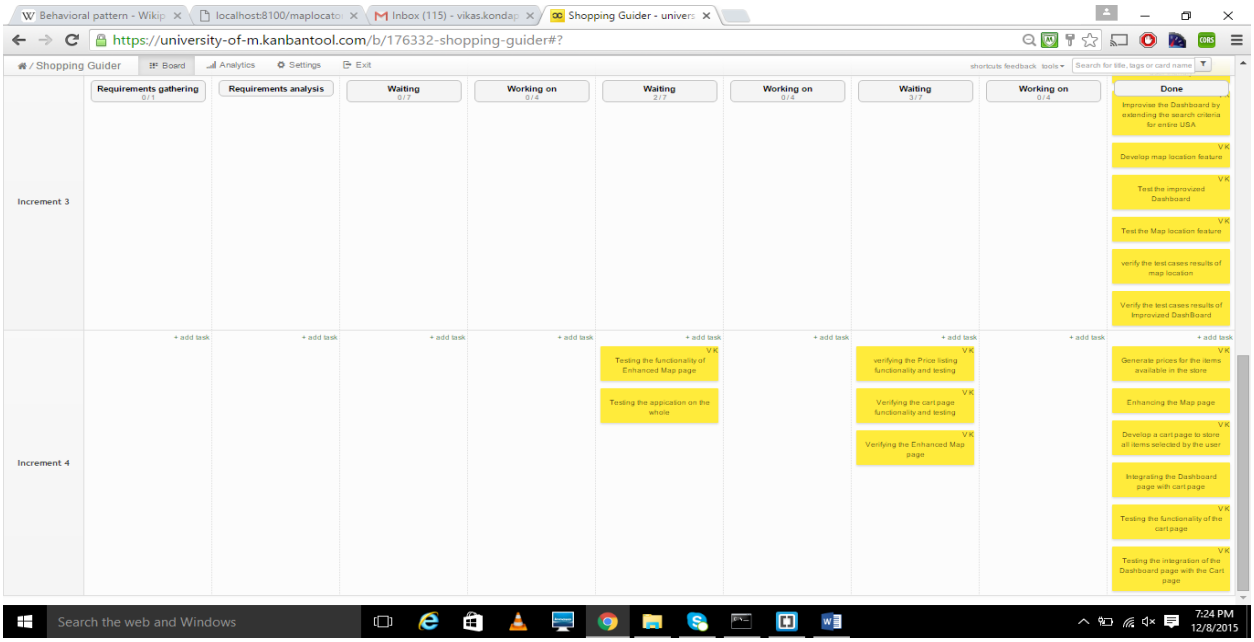
Both the above task were completed successfully and moved to done phase.



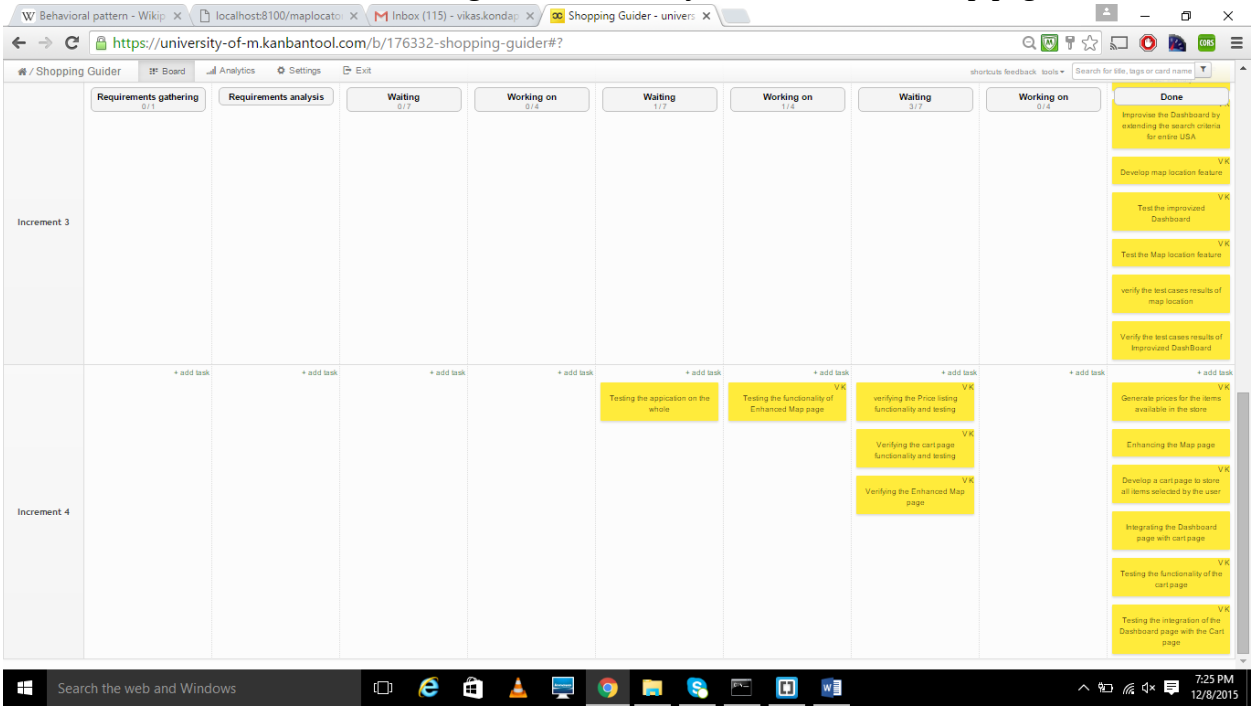
In the testing process the task which were carried out were,
The next tasks which were carried out were testing the functionality of the Cart page and testing the integration of the Cart page with the Dashboard page.



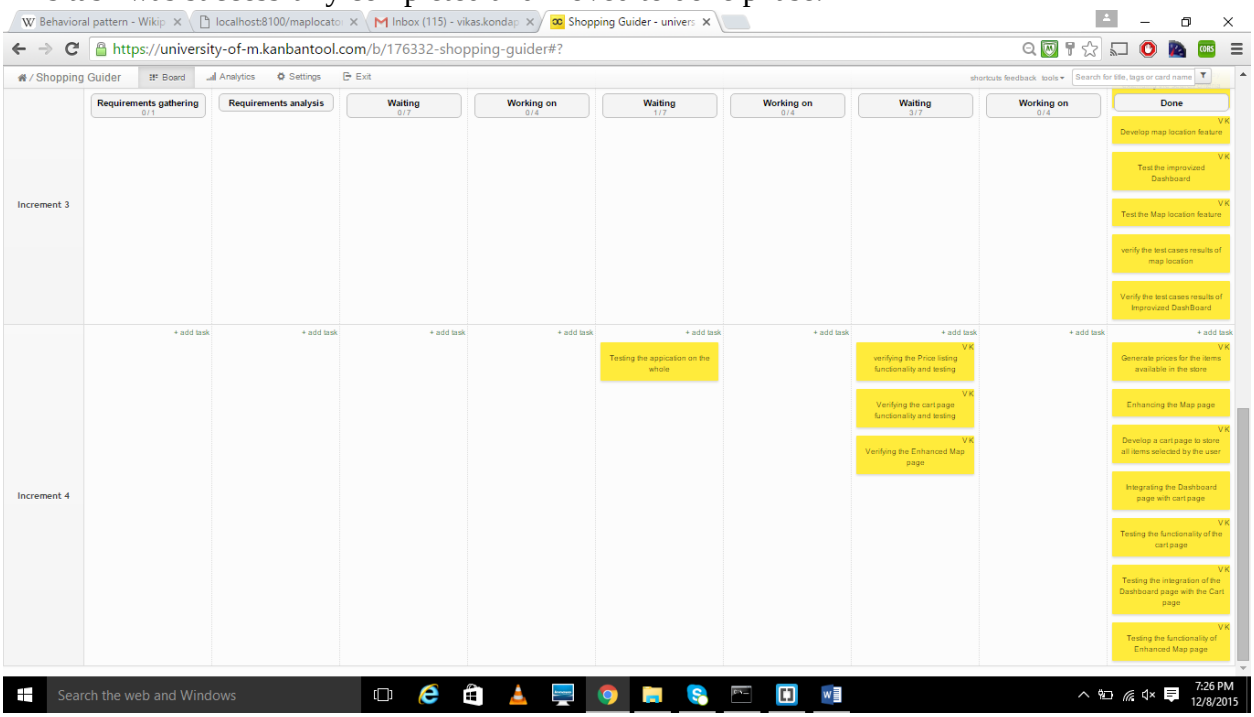
Both these tasks were successfully completed and moved to done phase.



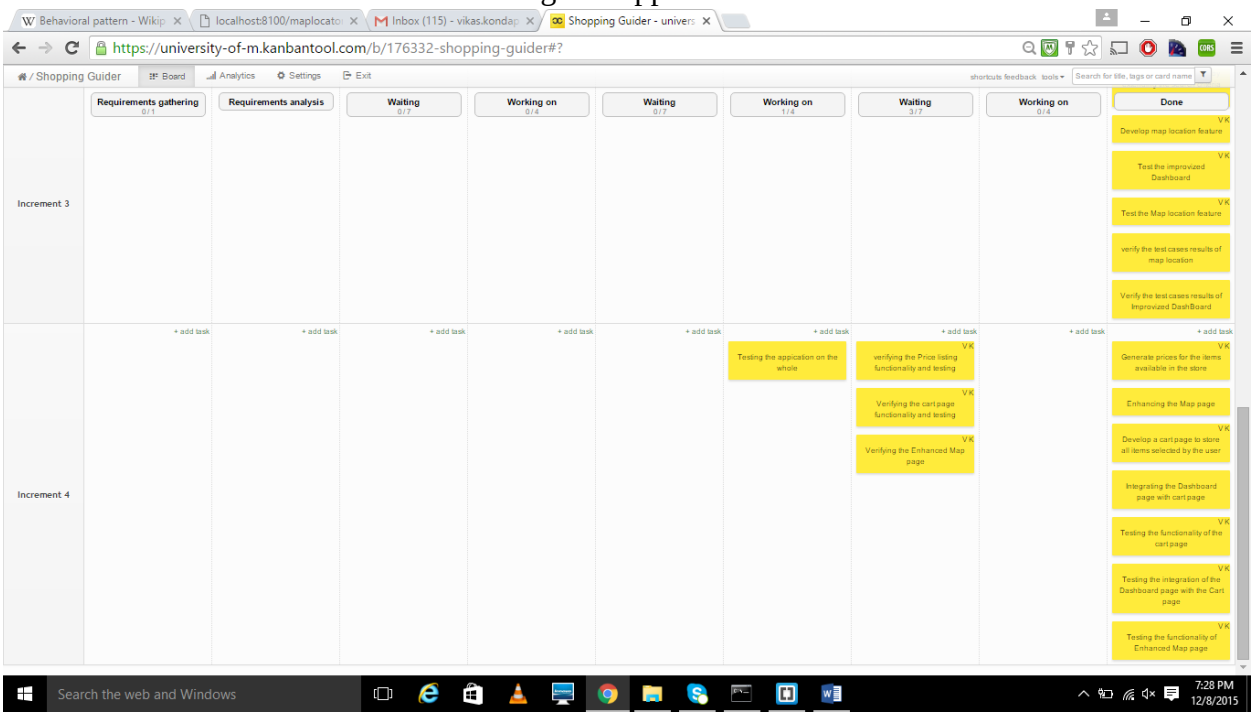
The next task carried out was Testing the functionality of the Enhanced Map page.



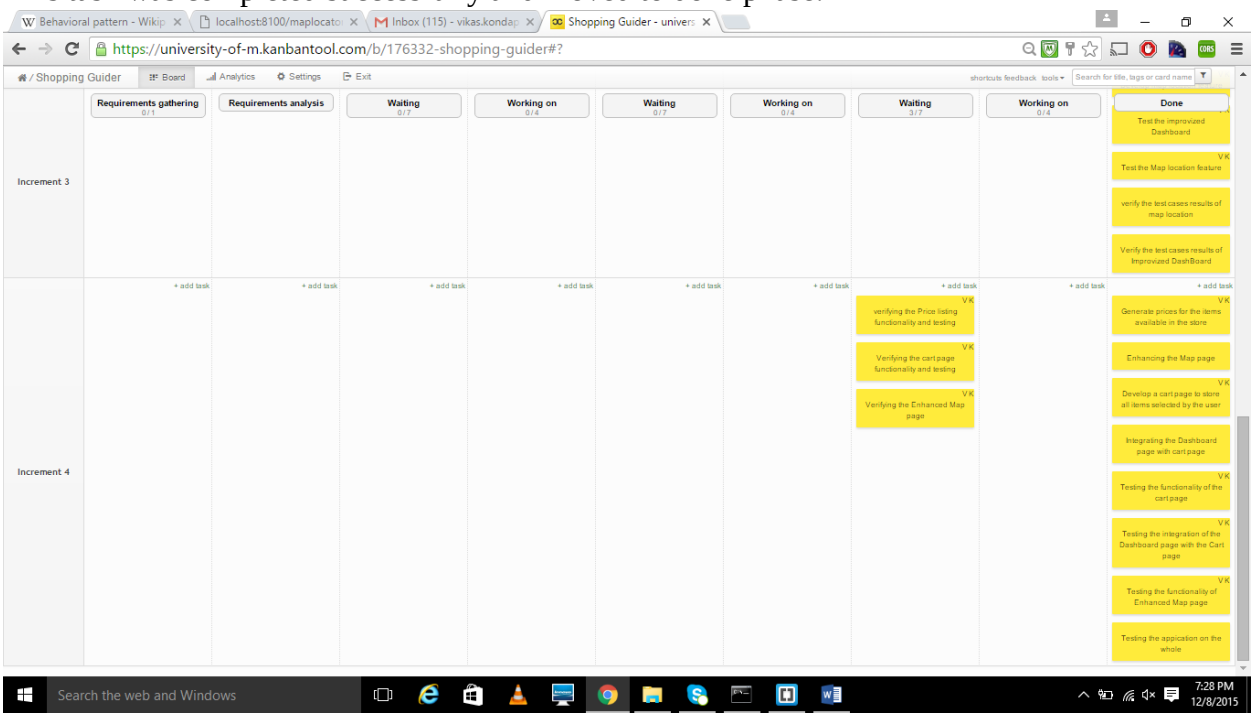
This task was successfully completed and moved to done phase.



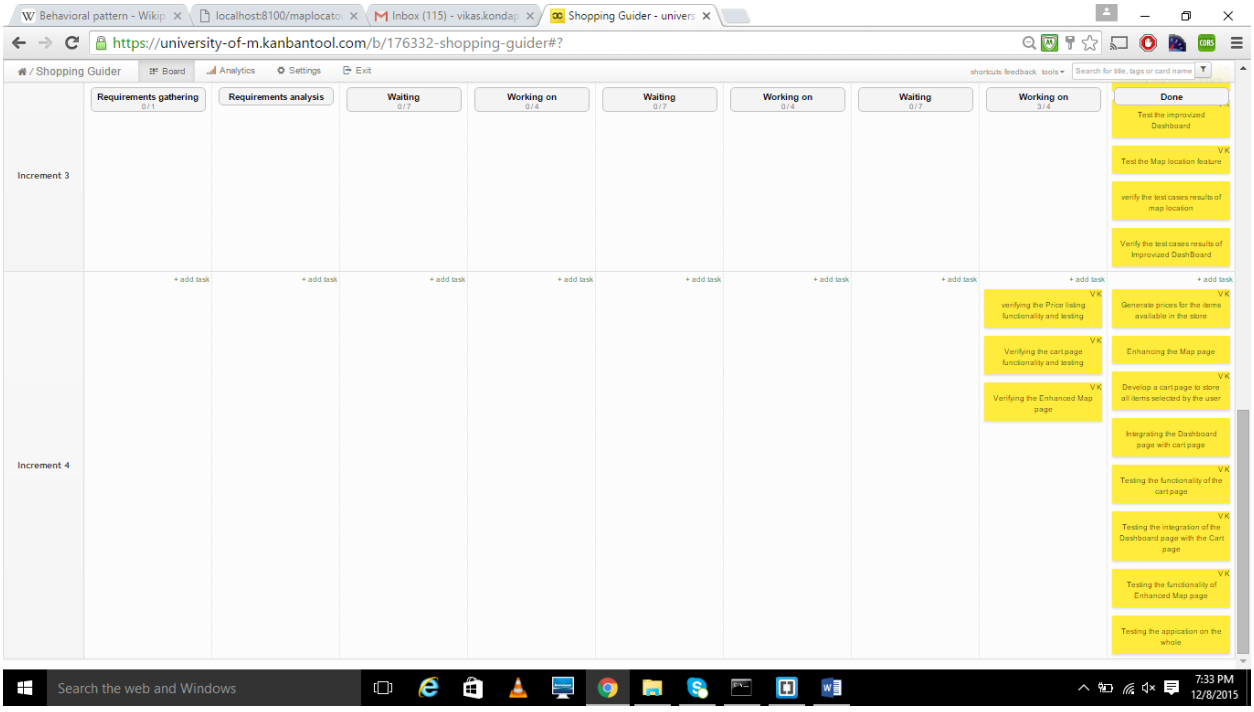
The next which was carried out was testing the Application on the whole.



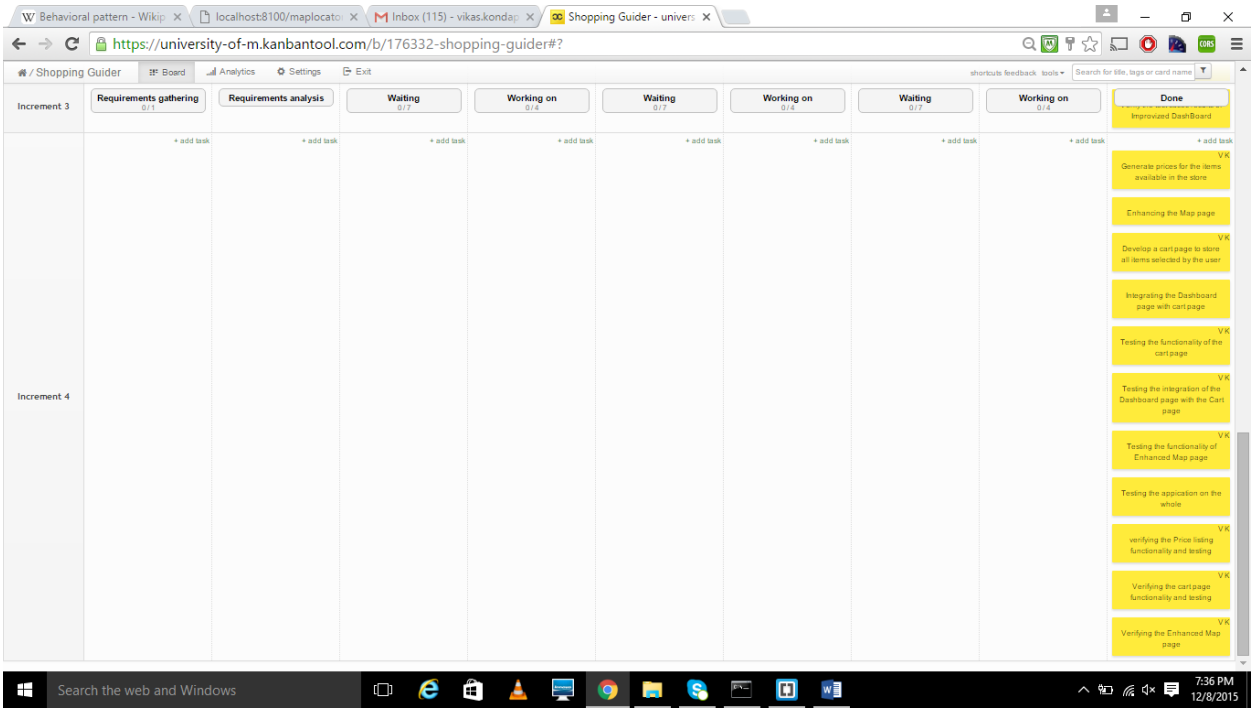
This task was completed successfully and moved to done phase.



The next tasks which were carried out were verifying the functionality and testing of the price generation, cart page and the Enhanced map page.



All of these tasks were successfully completed and moved to done phase.



All the task were completed.

Work Completed

Description

1. Developed a cart page to store the details of the items selected by the user.
2. Integrated the Cart page with the Dashboard page.
3. Generated the prices of the items available with in the stores.
4. Enhanced the Map page to render directions from current user location to store location and also display the distance to be travelled.

Responsibilities

1. Vikas Kondapalli: Development of the cart page, integrating the cart page with the Dashboard page, testing, documentation.
2. Gopi Krishna Bodapati: Generating the random prices of the items in the store, enhancing the GUI of the application, testing, documentation.
3. Swatvik Gunamaneni: Enhancing the Map locator page and integrating it with the Dashboard page, Testing, documentation.

Time taken

- Development of dashboard page and integrating it with Dashboard page: 150 hours
- Generating the prices of the items : 10 hours
- Enhancing the Map locator page : 20 hours
- Testing : 20 hours
- Documentation: 20 hours

Contributions

- Vikas Kondapalli : 33.33%

- Gopi Krishna Bodapati: 33.33%

- Swatvik Gunamaneni: 33.33%

Link for Kanban tool (project management tool)

https://university-of-m.kanbantool.com/b/176332-shopping-guider#?

Username and password are required for accessing it.

Bibliography

Link for Supermarket API used in this project

http://supermarketapi.com/Default.aspx