

Ajax- Search Products

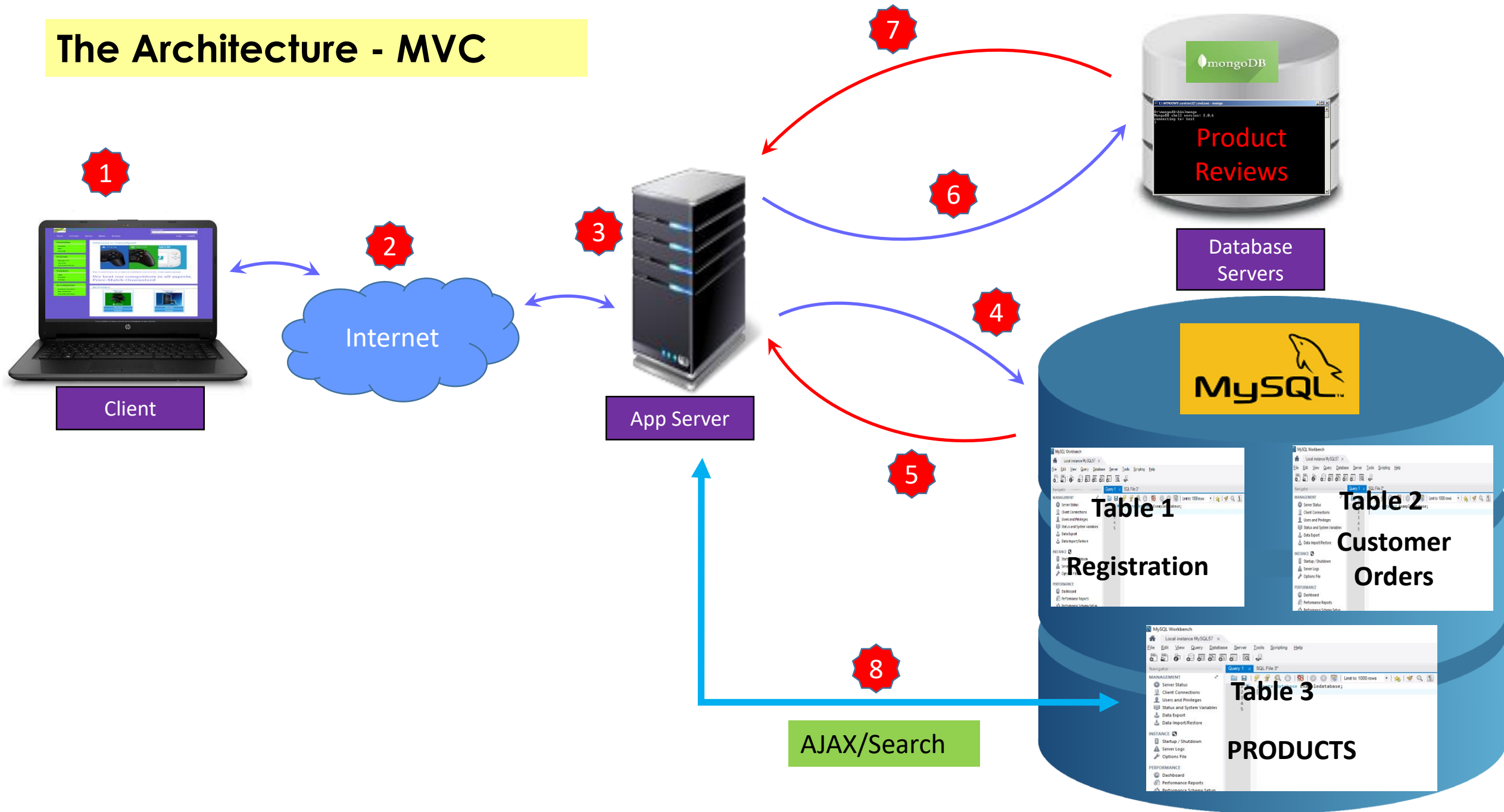
Tutorial – 5, Ajax Auto Complete Search feature

CSP 595 – Enterprise Web Application

Dr. Atef Bader

Illinois Institute of Technology

The Architecture - MVC



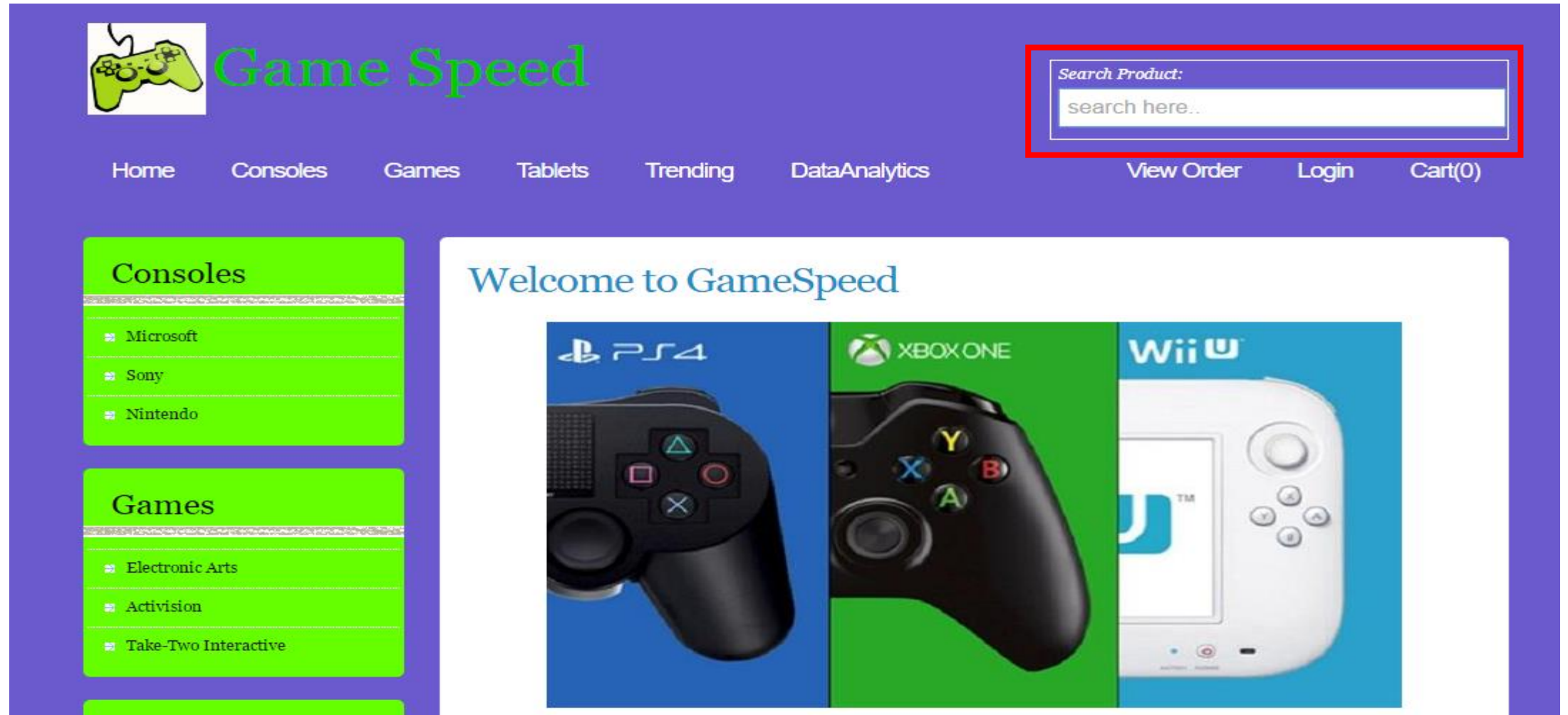
Introduction:

- ❖ **JavaScript** is the programming language of HTML and the Web.
- ❖ JavaScript is one of the 3 languages all web developers must learn:
 1. HTML to define the content of web pages
 2. CSS to specify the layout of web pages
 3. **JavaScript to program the behavior of web pages**
- ❖ To Learn More on JavaScript, visit: <http://www.w3schools.com/js/default.asp>

➤ **AJAX : Asynchronous JavaScript And XML.**

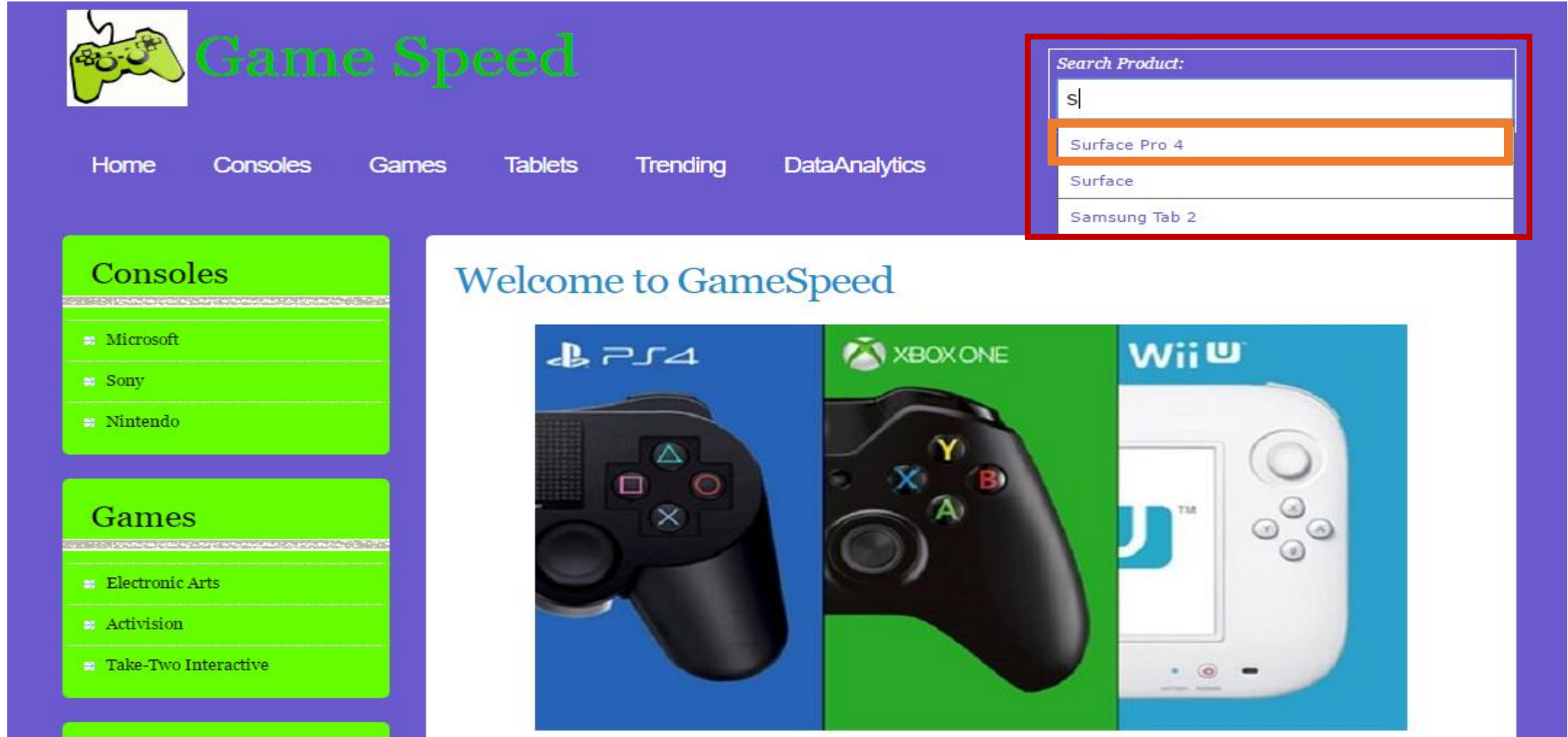
- AJAX allows **web pages to be updated asynchronously by exchanging data with a web server behind the scenes.**
- Advantages: Updates a web page without reloading the page , Request data from a server - after the page has loaded, Receive data from a server - after the page has loaded, Sends data to a server - in the background
- To Learn More on AJAX, visit: http://www.w3schools.com/xml/ajax_intro.asp

Sample Example For Search Box



Search Product Functionality

- After entering letter in the search box all the products starting with that letter should be displayed.
- In the Below Example , Customer enters the letter 'S' and the Suggestions from product are shown , From the suggestions list user clicks the Surface Pro 4.



The screenshot displays the GameSpeed website interface. At the top left is a logo featuring a green game controller and the text "Game Speed" in green. Below the logo is a navigation bar with links: Home, Consoles, Games, Tablets, Trending, and DataAnalytics. On the right side, a search box is highlighted with a red border. It contains the text "Search Product:" and a search input field with the letter "s". Below the input field, a list of suggestions is shown, with "Surface Pro 4" highlighted by an orange border. The suggestions list also includes "Surface" and "Samsung Tab 2". On the left side of the page, there are two green sections: "Consoles" and "Games". The "Consoles" section lists Microsoft, Sony, and Nintendo. The "Games" section lists Electronic Arts, Activision, and Take-Two Interactive. The main content area features a welcome message "Welcome to GameSpeed" and three large images of gaming consoles: PS4, Xbox One, and Wii U.

Game Speed

Home Consoles Games Tablets Trending DataAnalytics

Search Product:

s|

Surface Pro 4

Surface

Samsung Tab 2

Consoles

- Microsoft
- Sony
- Nintendo

Games

- Electronic Arts
- Activision
- Take-Two Interactive

Welcome to GameSpeed

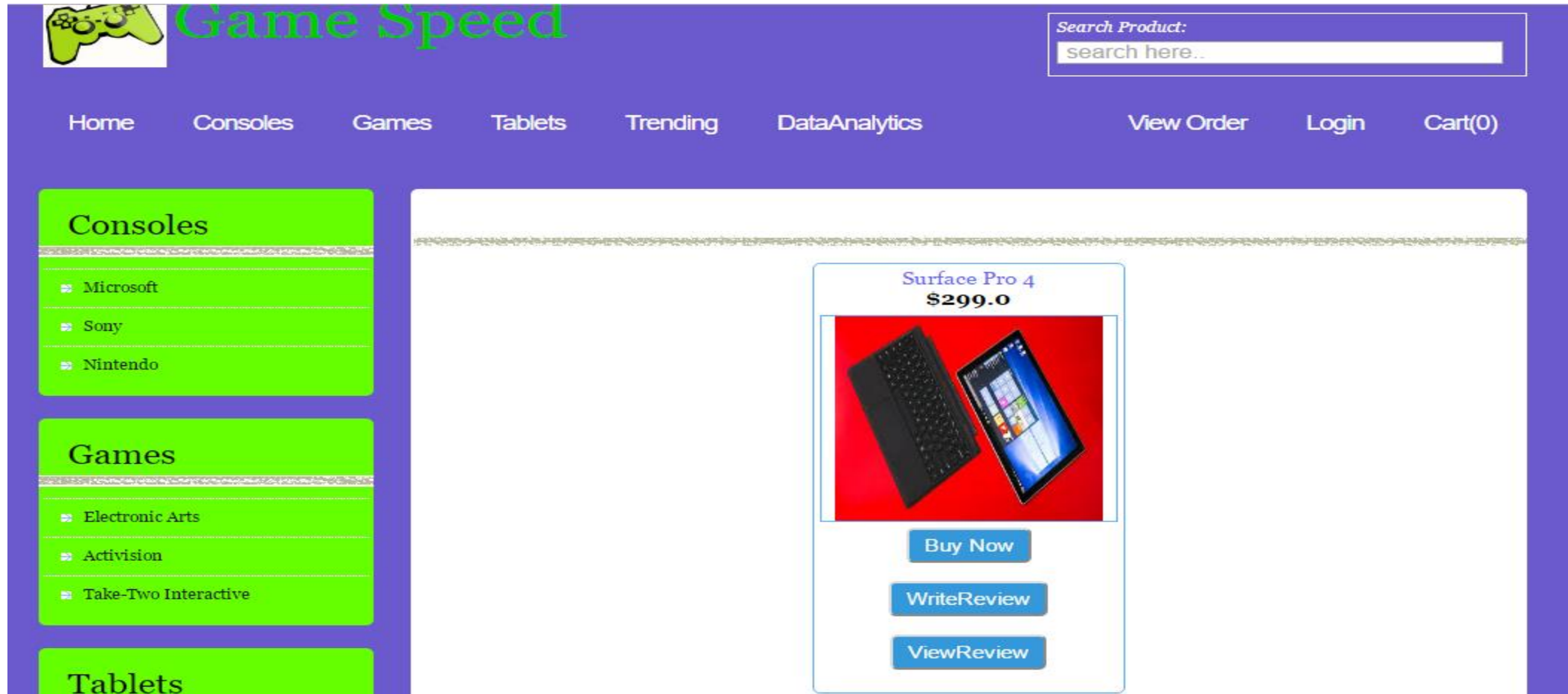
PS4

XBOX ONE

Wii U

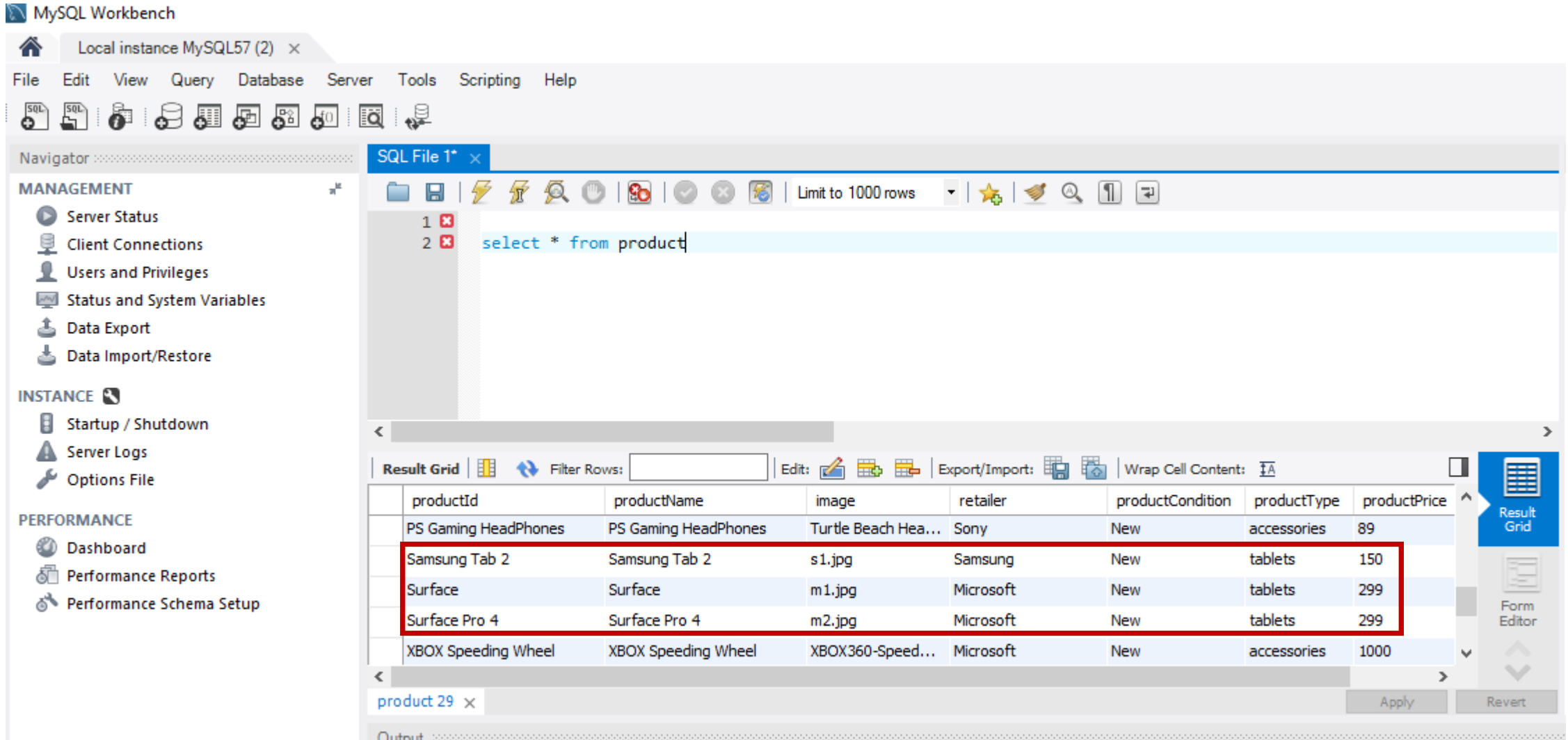
View Product

On clicking a particular product displayed in search box, Page should be redirected to display that product detail



Products Database

- All the products details are stored in product table in MySQL.
- Products Stored can be checked by executing select Query in database using MySQL workbench.



The screenshot displays the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar contains sections for MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore), INSTANCE (Startup / Shutdown, Server Logs, Options File), and PERFORMANCE (Dashboard, Performance Reports, Performance Schema Setup). The main workspace shows a SQL query in a file named 'SQL File 1*':

```
1  
2 select * from product
```

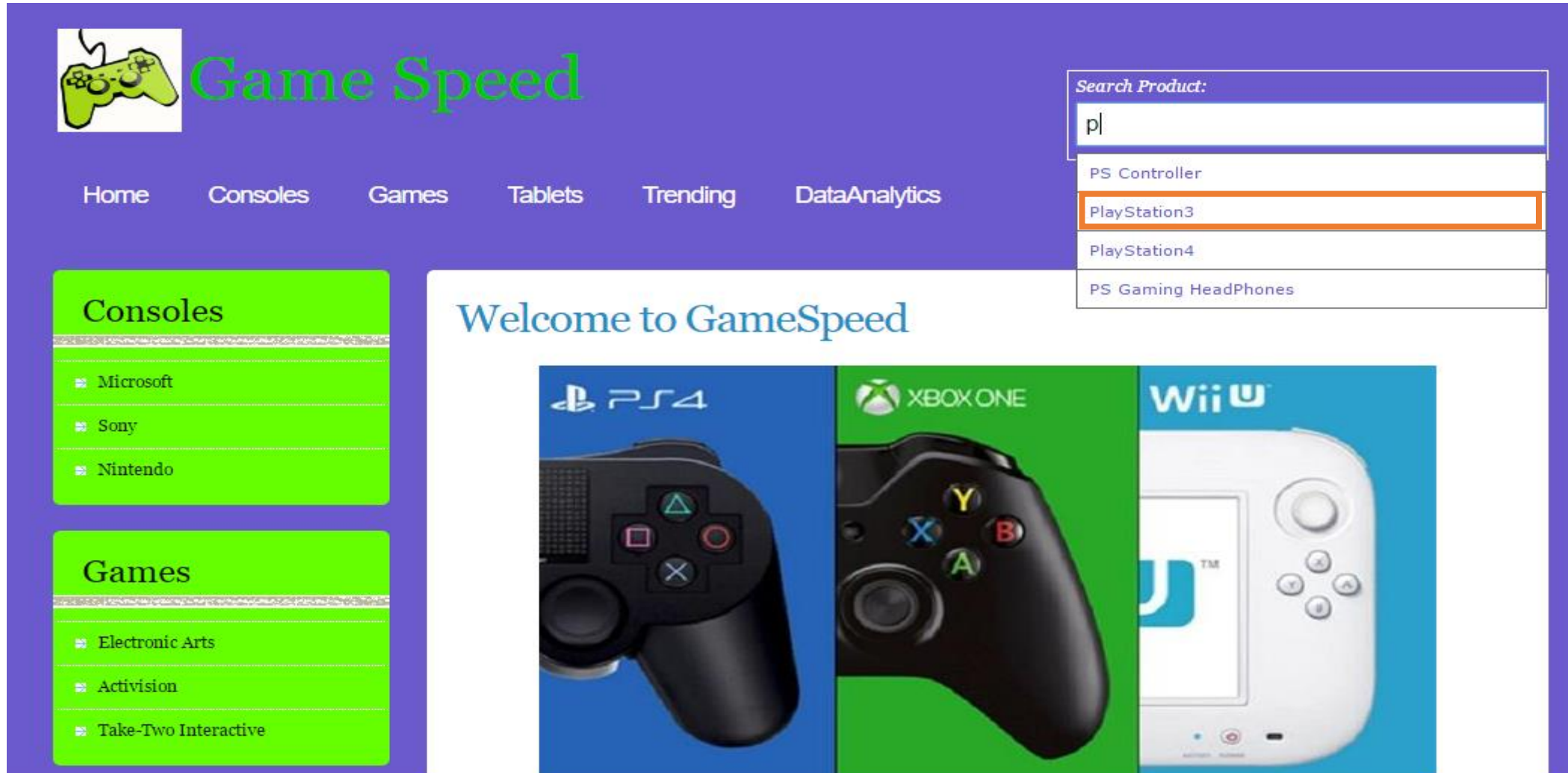
Below the query editor, the 'Result Grid' is visible, showing the results of the query. The grid has columns: productId, productName, image, retailer, productCondition, productType, and productPrice. The results are as follows:

productId	productName	image	retailer	productCondition	productType	productPrice
PS Gaming HeadPhones	PS Gaming HeadPhones	Turtle Beach Hea...	Sony	New	accessories	89
Samsung Tab 2	Samsung Tab 2	s1.jpg	Samsung	New	tablets	150
Surface	Surface	m1.jpg	Microsoft	New	tablets	299
Surface Pro 4	Surface Pro 4	m2.jpg	Microsoft	New	tablets	299
XBOX Speeding Wheel	XBOX Speeding Wheel	XBOX360-Speed...	Microsoft	New	accessories	1000

The 'Result Grid' tab is selected, and the 'product 29' tab is active at the bottom. The 'Apply' and 'Revert' buttons are visible at the bottom right.

One more search entry for search box


- After entering letter in the search box all the products starting with that letter will be displayed.
- In the Below Example , Customer enters the letter 'p' and the Suggestions from products are shown , From the suggestion list customer clicks the PlayStation 3.



The screenshot displays the GameSpeed website interface. At the top left is a logo featuring a green game controller and the text "Game Speed". Below the logo is a navigation menu with links: Home, Consoles, Games, Tablets, Trending, and DataAnalytics. On the right side, there is a search bar labeled "Search Product:" containing the letter "p". Below the search bar, a list of suggestions is shown: "PS Controller", "PlayStation3" (highlighted with an orange border), "PlayStation4", and "PS Gaming HeadPhones". On the left side of the main content area, there are two green boxes. The first box is titled "Consoles" and lists "Microsoft", "Sony", and "Nintendo". The second box is titled "Games" and lists "Electronic Arts", "Activision", and "Take-Two Interactive". The main content area features a "Welcome to GameSpeed" message above three images of gaming consoles: a PS4 controller, an Xbox One controller, and a Wii U console.

View Product

On clicking a particular product displayed in search box, Page should be redirected to display that product detail



Game Speed

Search Product:

[Home](#) [Consoles](#) [Games](#) [Tablets](#) [Trending](#) [DataAnalytics](#) [View Order](#) [Login](#) [Cart\(0\)](#)

Consoles


- Microsoft
- Sony
- Nintendo

Games

- Electronic Arts
- Activision
- Take-Two Interactive

PlayStation3

\$349.0



[Buy Now](#)

[WriteReview](#)

Adding Search Box in html

```
<body onload="init()">
  <script type="text/javascript" src="javascript.js"></script>
  <div name="autofillform">
    <input type="text" name="searchId" value="" class="input" id="searchId"
      onkeyup="doCompletion()"
      placeholder="search here.." style="padding: 5px; font-size: 16px;" />
    <div id="auto-row">
      <table id="complete-table" class="gridtable" style="position: absolute; width:
        315px;"></table>
    </div>
  </div>
</body>
```

Include the
javascript file in
html

Call the Javascript
function to process
the search text

JavaScript for Search

```
function init() {  
    completeField = document.getElementById("searchId");  
    completeTable = document.getElementById("complete-table");  
    autoRow = document.getElementById("auto-row");  
}  
function doCompletion() {  
    var url = "autocomplete?action=complete&searchId =" + escape(searchId.value);  
    req = initRequest();  
    req.open("GET", url, true);  
    req.onreadystatechange = callback;  
}  
function initRequest() {  
    if (window.XMLHttpRequest) {  
        if (navigator.userAgent.indexOf('MSIE') != -1) {  
            isIE = true;  
        }  
        return new XMLHttpRequest();  
    } else if (window.ActiveXObject) {  
        isIE = true;  
        return new ActiveXObject("Microsoft.XMLHTTP");  
    }  
}
```

init() function
Getting fields from
html in javascript

We are sending searchId as
parameter that is the text we
entered in search box

Direct the webpage to
autocomplete servlet

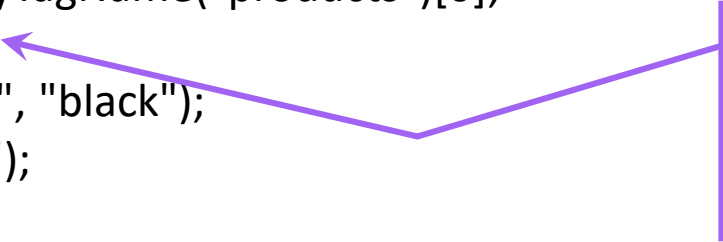
Creat Javascript Activex
Object

```
function appendProduct(productName,productId) {  
    var row;  
    var cell;  
    var linkElement;  
  
    if (isIE) {  
        completeTable.style.display = 'block';  
        row = completeTable.insertRow(completeTable.rows.length);  
        cell = row.insertCell(0);  
    } else {  
        completeTable.style.display = 'table';  
        row = document.createElement("tr");  
        cell = document.createElement("td");  
        row.appendChild(cell);  
        completeTable.appendChild(row);  
    }  
    cell.className = "popupCell";  
    linkElement = document.createElement("a");  
    linkElement.className = "popupItem";  
    linkElement.setAttribute("href", "autocomplete?action=lookup&searchId=" + productId);  
    linkElement.appendChild(document.createTextNode(productName));  
    cell.appendChild(linkElement);  
}
```

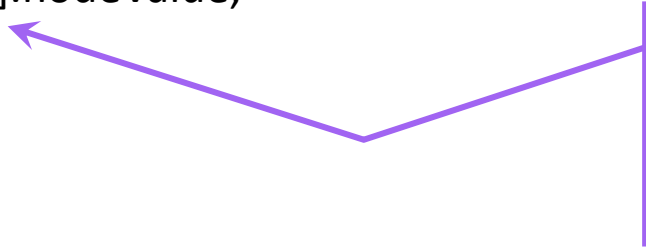
This function appends the products starting with searchId to web page

Append all products in table row and display them below the search box

```
function parseMessages(responseXML) {  
    // no matches returned  
    if (responseXML == null) {  
        return false;  
    } else {  
        var products = responseXML.getElementsByTagName("products")[0];  
        if (products.childNodes.length > 0) {  
            completeTable.setAttribute("bordercolor", "black");  
            completeTable.setAttribute("border", "1");  
  
            for (loop = 0; loop < products.childNodes.length; loop++) {  
                var product = products.childNodes[loop];  
                var productName = product.getElementsByTagName("productName")[0];  
                var productId = product.getElementsByTagName("id")[0];  
                appendProduct(productName.childNodes[0].nodeValue,  
                    productId.childNodes[0].nodeValue);  
            }  
        }  
    }  
}
```




Get the products obtained
as response from auto
complete servlet



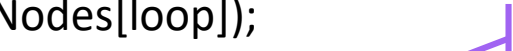
Pass the product id and
product name to
appendProduct()
function

```
function callback() {  
    clearTable();  
    if (req.readyState == 4) {  
        if (req.status == 200) {  
            parseMessages(req.responseXML);  
        }  
    }  
}
```



Parse the values given by
auto complete servlet on
response callback

```
function clearTable() {  
    if (completeTable.getElementsByTagName("tr").length > 0) {  
        completeTable.style.display = 'none';  
        for (loop = completeTable.childNodes.length - 1; loop >= 0 ; loop--) {  
            completeTable.removeChild(completeTable.childNodes[loop]);  
        }  
    }  
}
```



Clears the table below
search box and removes
all products from it

Auto Complete Servlet Code

```
try
{
    StringBuffer sb = new StringBuffer();
    boolean namesAdded = false;
    if (action.equals("complete"))
    {
        if (!searchId.equals(""))
        {
            AjaxUtility a=new AjaxUtility();
            sb=a.readdata(searchId);
            if(sb!=null || !sb.equals(""))
            {
                namesAdded=true;
            }
            if (namesAdded)
            {
                response.setContentType("text/xml");
                response.getWriter().write("<products>" + sb.toString() + "</products >");
            }
        }
    }
}
```

Calling AjaxUtility class
readData() Function to
get products starting
with searchId

Sending the string
buffer as response
in xml format

Ajax Utility Function- getData()

getData() function used to get the products from database and store in hashmap

```
public static HashMap<String,Product> getData()
```

```
{    HashMap<String,Product> hm=new HashMap<String,Product>();
```

```
    try
```

```
    {        getConnection();
```

```
        Statement stmt=conn.createStatement();
```

```
        String selectCustomerQuery="select * from product";
```

```
        ResultSet rs = stmt.executeQuery(selectCustomerQuery);
```

```
        while(rs.next())
```

```
        {
```

```
            Product p = new Product(rs.getString("productId"), rs.getString("productName"));
```

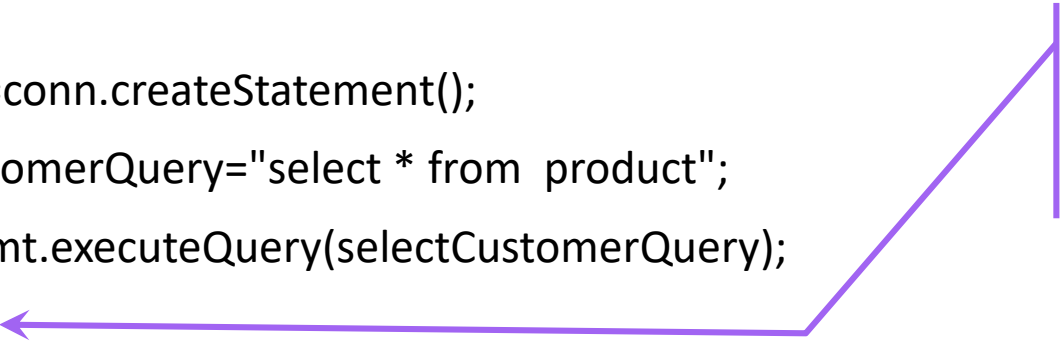
```
            hm.put(rs.getString("productId"), p);
```

```
        }
```

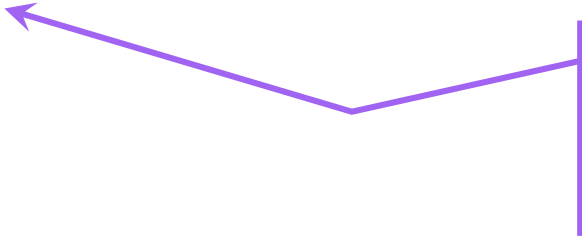
```
    }
```

```
    return hm;
```

```
}
```



Iterate through result set to get each product record



Get the data from table and store in hashmap

Ajax Utility Function- readData()

readData() function used to get the products starting with letter typed from hashmap into string buffer

```
public StringBuffer readdata(String searchId)
{
    HashMap<String,Product> data;
        data=getData();
    Iterator it = data.entrySet().iterator();
    while (it.hasNext())
    {
        Map.Entry pi = (Map.Entry)it.next();
        Product p=(Product)pi.getValue();
        if (p.getName().toLowerCase().startsWith(searchId))
        {
            sb.append("<product>");
            sb.append("<id>" + p.getId() + "</id>");
            sb.append("<productName>" + p.getName() + "</ productName >");
            sb.append("</ product >");
        }
    }
    return sb;
}
```

Calling get data function to get the product data into hashmap

Check if hashmap contains any product starting with letter stored in searchId

Append the product details in xml tag format

Questions ??