**-:Third Week :-**

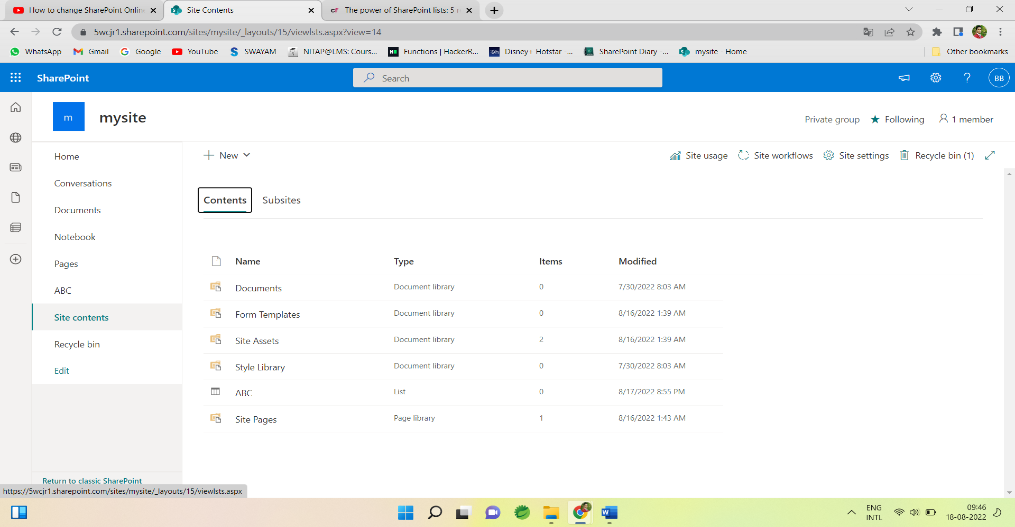
**-:Day 1st :-**

**SharePoint Predefined List Template:**

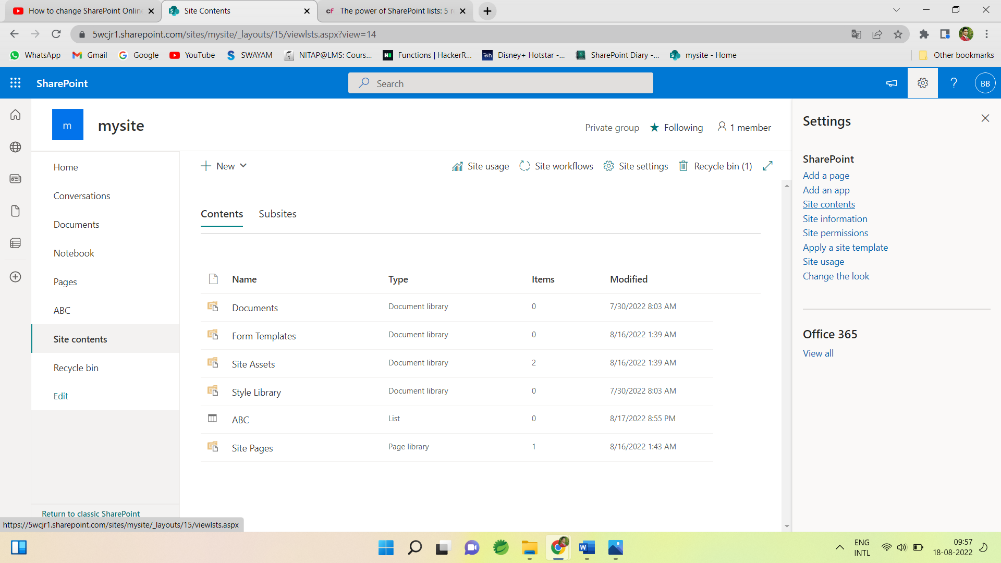
The SharePoint List Template is a great resource for developers who are looking to programmatically create new lists. Each template may represent a unique predefined set of settings, content types, columns, and views. Manually configuring a large quantity of these list settings may no longer needed when lists are created by using the appropriate SharePoint List Template.

**Create a SharePoint list using predefined templates:**

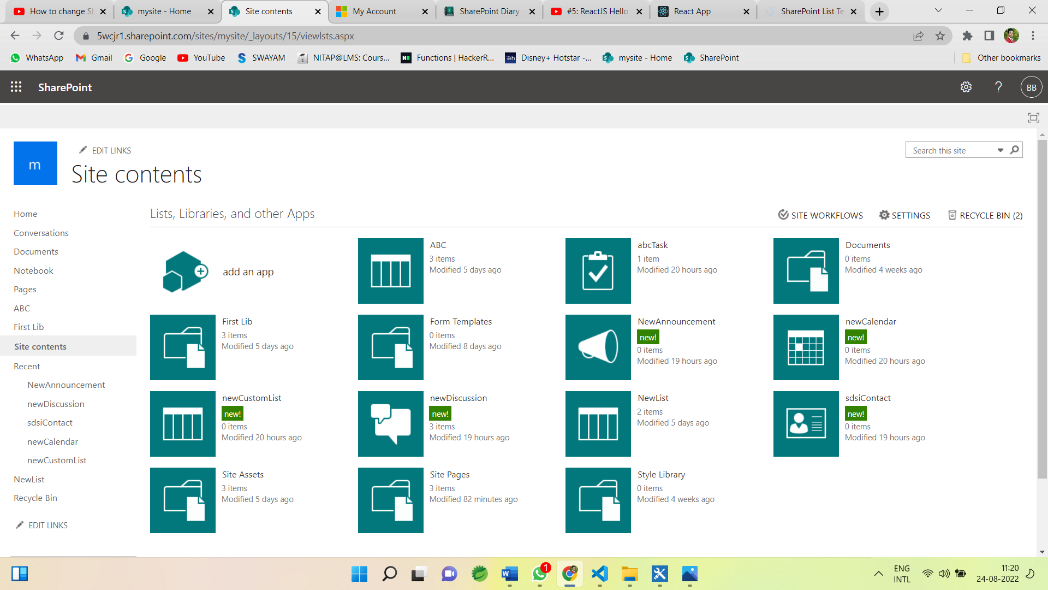
From SharePoint site home page



Or the Site contents page from gear icon,

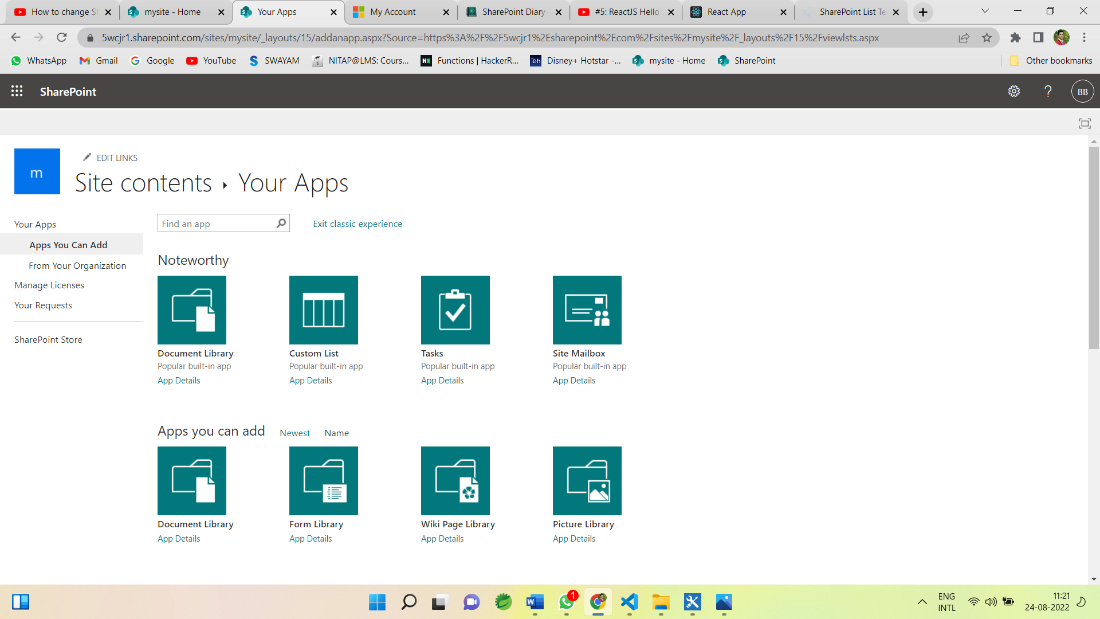


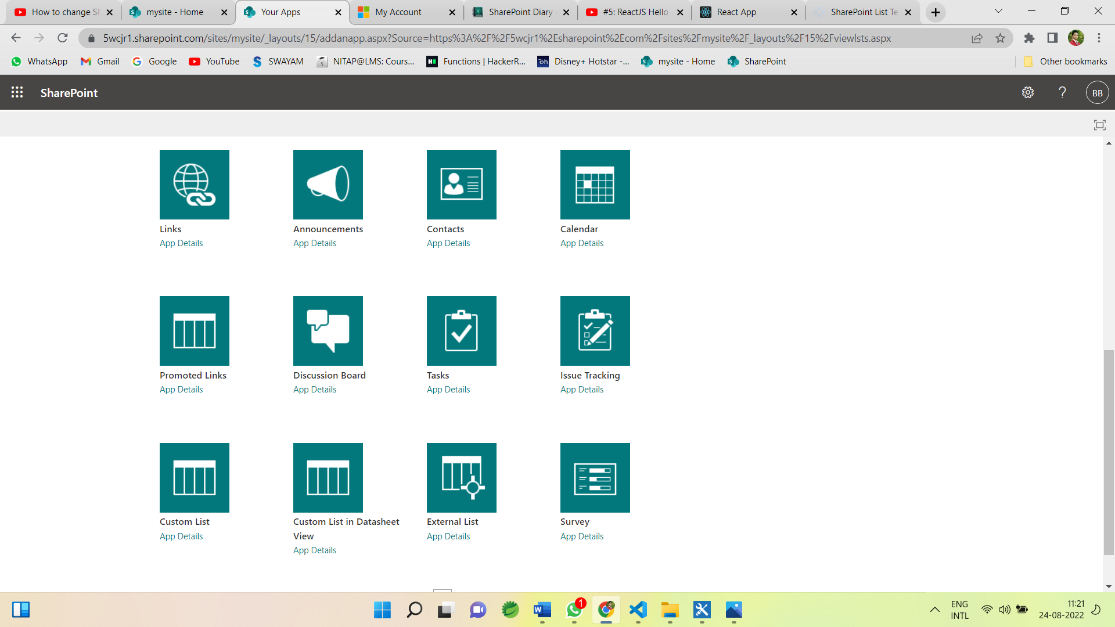
Then classic experience interface looks like



Then select ->  add an app

Now all the predefined list templates are visible



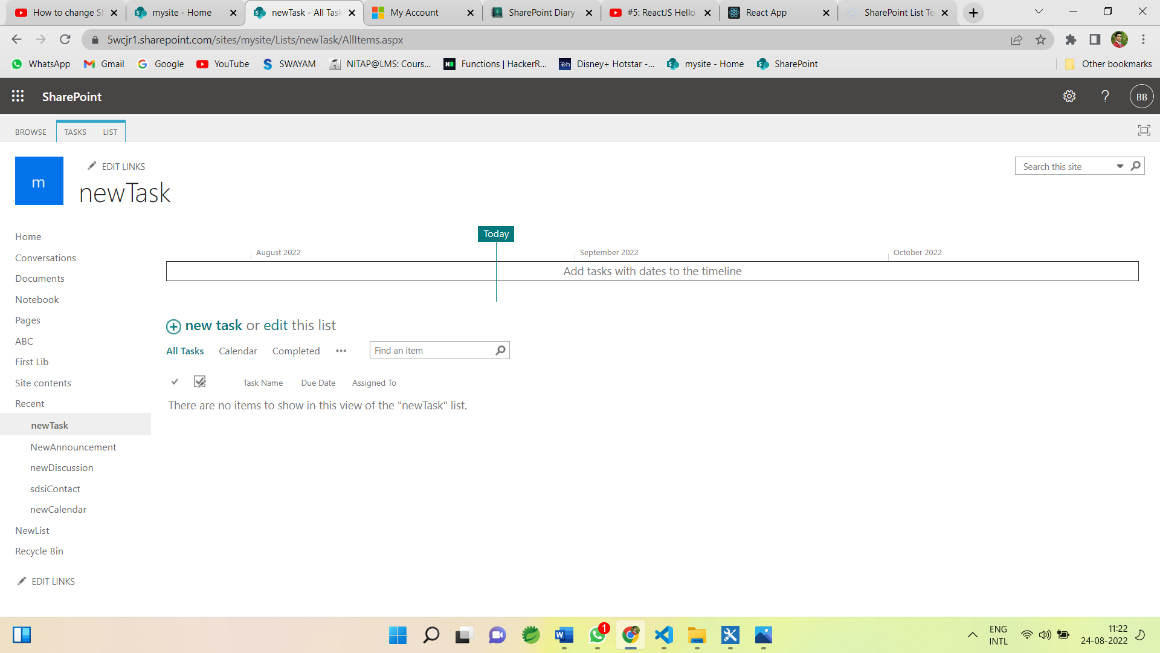


Now choose template according to the requirement.

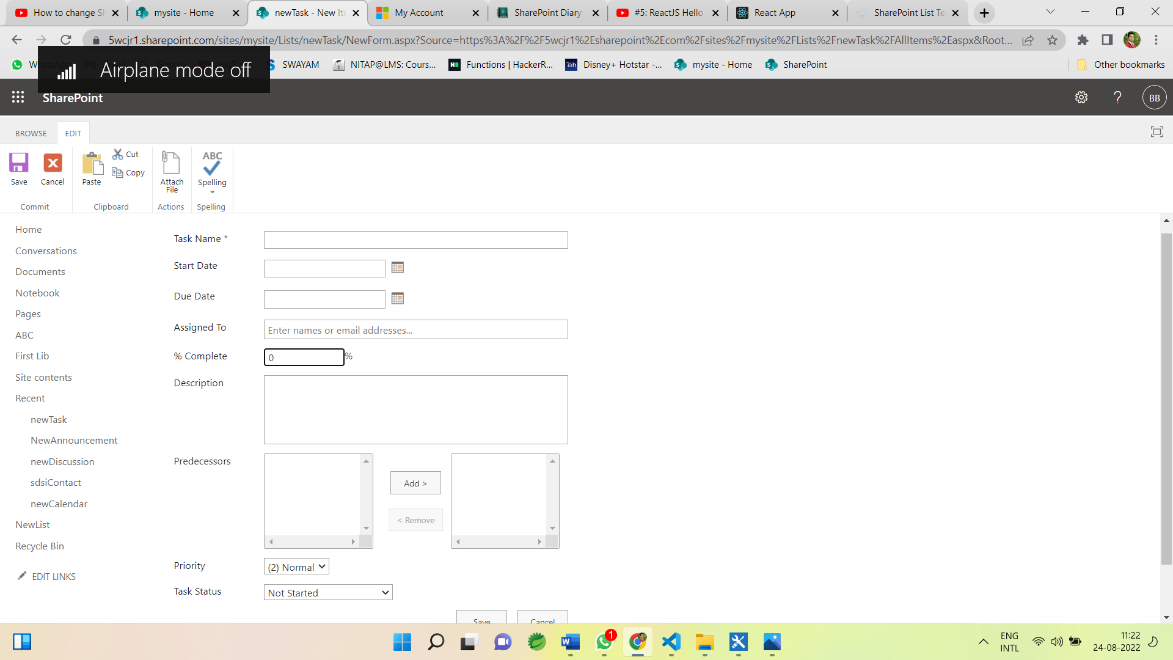
For example for Creating a task

Choose a task templete and give name

Then new task created with default field and views



Then click on + new task and its default fields open.



Now give the name and other fields.

Similarly, we can add other templates like contact, calendar, Discussion Board, Issue tracking, survey etc.

**-: Day 2nd :-**

**SharePoint Designer 2013:**

SharePoint Designer 2013 is the tool of choice for the rapid development of SharePoint applications. It is the tool of choice for the rapid development of SharePoint applications. Using SharePoint Designer, advanced users and developers alike can rapidly create SharePoint solutions in response to business needs.

Using this tool we can edit, customise the site we can write or edit code.

**SharePoint Workflow:**

* List Workflow
* Reusable Workflow
* Site Workflow

**Workflow Triger option:**

Trigger condition takes the format of an expression and must evaluate to either true or False. If the trigger condition is true then the flow will run, else it will ignore the trigger event.

Triger options are:

* Manually
* Automatic when condition satisfy.
* When an item is created and
* When item is changed.

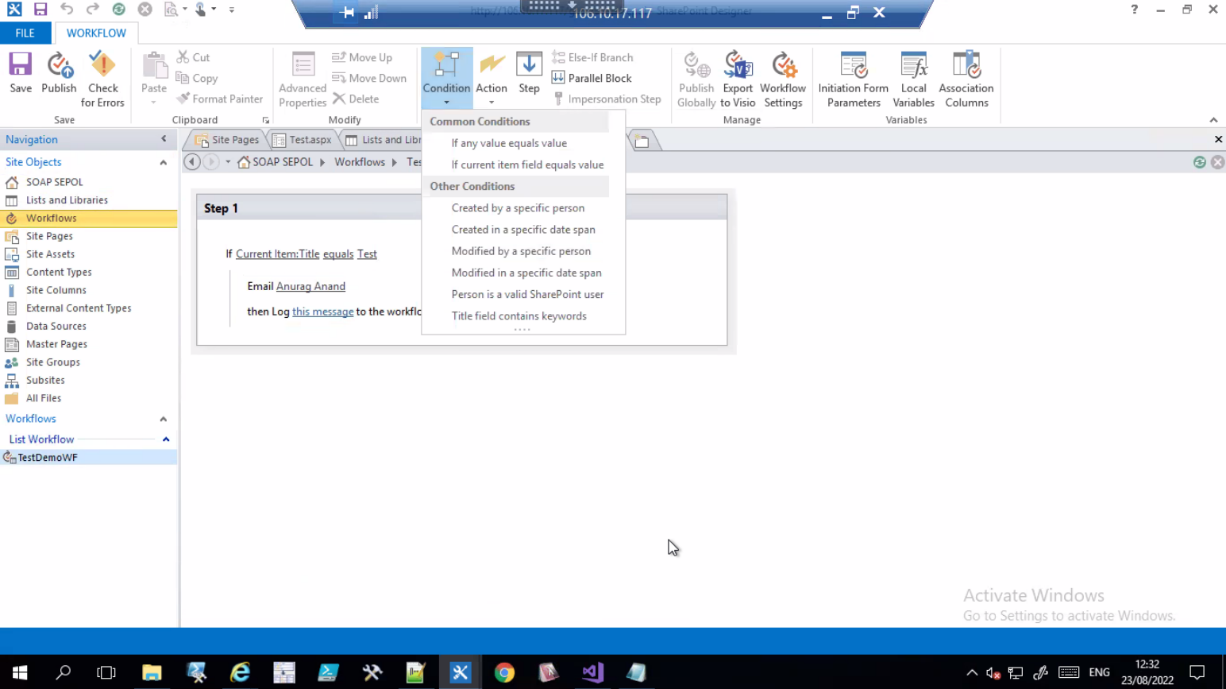
For example, if we use a trigger called ‘When an item is created or modified”, the flow will trigger with each update.

**Conditions of workflow:**

Workflow will be executed if the given condition satisfies then it executes the action inside the condition.

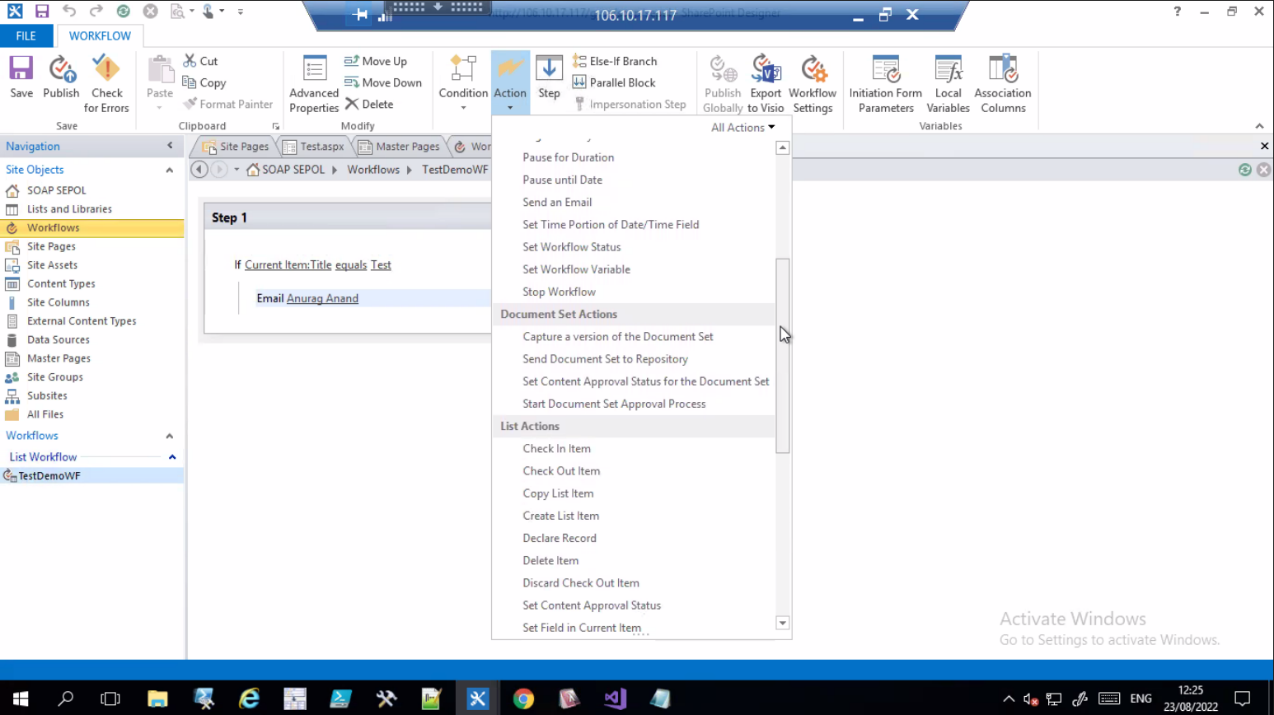
* Any value equals value
* Current item field equals value
* Created by specific person
* Created by specific date span etc.

We can apply workflow condition by choosing the suitable condition from condition tab:



Then Specify the task/action to be execute if the given condition satisfies from the action tab like:

* Pause for duration
* Pause until date
* Send an Email
* Set Workflow status
* Set Workflow variable
* Stop Workflow
* Check in Item
* Check Out Item etc.



**-: Day 3rd :-**

Today we had seen the Create operation of CRUD operation on SharePoint by coding.

**Create a HTML Form:**

Here we have a SharePoint list as “ComapnyInfoList” which has below columns:

* Username
* Password
* EmailID and
* gender

**HTML Code for form creation**

-----------------------------------------------------------------------------------------

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://onlysharepoint2013.sharepoint.com/sites/Raju/SiteAssets/Preetihtml/UserInfoJs.js"></script>

<meta charset="utf-8" />

<title>My Html</title>

</head>

<body>

<h2>Information Form</h2>

Username:<br>

<input type="text" name="Username" id="usrname" />

<br><br>

Password:<br>

<input type="password" name="password" id="usrpwd" />

<br><br>

EmailID:<br>

<input type="email" name="EmailID" id="usrid" />

<br><br>

Gender:<br>

<div id="gend">

<input type="radio" name="gender" value="Male" id="gen0" /> Male

<input type="radio" name="gender" value="Female" id="gen1" /> Female

</div>

<br><br>

<input type="submit" value="Submit" id="btnSubmit" />

<br>

<p id="pTitle"></p>

</body>

</html>

**JSOM Code for inserting data**

---------------------------------------------------------------------------------------------------

$(document).ready(function () {

$("#btnSubmit").click(function () {

insertitemtolist();

});

});

function insertitemtolist() {

var username = $("#usrname").val();

var password = $("#usrpwd").val();

var emailId = $("#usrid").val();

var gender = $("input[name='gender']:checked").val();

var clientContext = new SP.ClientContext.get\_current();

var oList = clientContext.get\_web().get\_lists().getByTitle('CompanyInfoList');

var item = new SP.ListItemCreationInformation();

var oListItem = oList.addItem(item);

oListItem.set\_item('Title', username);

oListItem.set\_item('Password', password);

oListItem.set\_item('EmailID', emailId);

oListItem.set\_item('Gender', gender);

oListItem.update();

clientContext.load(oListItem);

clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceeded), Function.createDelegate(this, this.onQueryFailed));

}

function onQuerySucceeded(sender, args) {

$("#pTitle").html("successfully executed");

}

function onQueryFailed(sender, args) {

alert('request failed ' + args.get\_message() + '\n' + args.get\_stackTrace());

}

----------------------------------------------------------------------------------------------------

**-: Day 4th :-**

Today we had seen the Get/Read, Update and Delete operation of CRUD operation on SharePoint by coding.

**JSOM Code for updating data in CompanyInfoList**

---------------------------------------------------------------------------------------------------

$(document).ready(function () {

ExecuteOrDelayUntilScriptLoaded(retrieveListItems, "sp.js");

$("#btnUpdate").click(function () {

updateListItemByID();

});

});

var masterListItem;

function retrieveListItems() {

var id = GetParameterValues('MyID');

getitemsbyID(id);

}

function getitemsbyID(itemID) {

var clientContext = new SP.ClientContext.get\_current();

var masterlist = clientContext.get\_web().get\_lists().getByTitle('CmpanyInfoList');

masterListItem = masterlist.getItemById(itemID);

clientContext.load(masterListItem);

clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceeded),

Function.createDelegate(this, this.onQueryFailed));

}

function onQuerySucceeded() {

$("#usrname").val(masterListItem.get\_item('Title'));

$("#usrpwd").val(masterListItem.get\_item('Password'));

$("#usrid").val(masterListItem.get\_item('EmailID'));

if (masterListItem.get\_item('Gender') == "Male") {

$("#gen0").attr('checked', 'checked');

}

else if (masterListItem.get\_item('Gender') == "Female") {

$("#gen1").attr('checked', 'checked');

}

}

function onQueryFailed(sender, args) {

alert('Request failed. \nError: ' + args.get\_message() + '\nStackTrace: ' + args.get\_stackTrace());

}

function GetParameterValues(param) {

var url = window.location.href.slice(window.location.href.indexOf('?') + 1).split('&');

for (var i = 0; i < url.length; i++) {

var urlparam = url[i].split('=');

if (urlparam[0] == param) {

return urlparam[1];

}

}

}

function updateListItemByID() {

var id = GetParameterValues('MyID');

var username = $("#usrname").val();

var password = $("#usrpwd").val();

var emailId = $("#usrid").val();

var gender = $("input[name='gender']:checked").val();

var clientContext = new SP.ClientContext.get\_current();

var oList = clientContext.get\_web().get\_lists().getByTitle('CmpanyInfoList');

var oListItem = oList.getItemById(id);

oListItem.set\_item('Title', username);

oListItem.set\_item('Password', password);

oListItem.set\_item('EmailID', emailId);

oListItem.set\_item('Gender', gender);

oListItem.update();

clientContext.executeQueryAsync(Function.createDelegate(this, this.success), Function.createDelegate(this, this.failue));

}

function success() {

alert('Item updated!');

}

function failue(sender, args) {

alert('Request failed. ' + args.get\_message() + '\n' + args.get\_stackTrace());

}

**Create HTML File for displaying the data:**

**-----------------------------------------------------------------------------------------**

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script> ----> // $ in Jsom

<script src="https://onlysharepoint2013.sharepoint.com/sites/Raju/SiteAssets/Preetihtml/UserInfoJs.js"></script> ---> SPServices in Jsom

<meta charset="utf-8" />

<title>ALL LIST ITEM</title>

</head>

<body>

<p id="disp"></p>

</body>

</html>

**JSOM Code for Reading/Getting the data From list “CompanyInfoList”**

**----------------------------------------------------------------------------------------**

$(document).ready(function () {

getdatafromlist();

});

function getdatafromlist(){

var clientContext = new SP.ClientContext.get\_current();

var oList = clientContext.get\_web().get\_lists().getByTitle('CompanyInfoList');

var oListItems=oList.getItems(SP.CamlQuery.createAllItemsQuery()); ------>to etrieve all list items we use CamlQuery

clientContext.load(oListItem);

clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceeded),

Function.createDelegate(this, this.onQueryFailed));

}

function onQuerySucceeded(sender, args) {

var newtitle='';

var listItemEnum=oListItems.getEnumerator(); -----> data is in the list enumerator will fetch one by one

while(listItemEnum.moveNext()){ ------>As long as the Enumerator will find next item this is will return true

var oListItem=listItemEnum.get\_current().get\_item('title');

newtitle+=oListItem;

}

$("#disp").append(newtitle);

}

function onQueryFailed(sender, args) {

alert('request failed ' + args.get\_message() + '\n' + args.get\_stackTrace());

}

**JSOM Code for Deleting data From List “CompanyInfoList”**

**----------------------------------------------------------------------------------------**

$(document).ready(function () {

deleteitem();

});

function deleteitem() {

var dltItem=$("#dlttext").val();

var clientContext = new SP.ClientContext.get\_current();

var oList = clientContext.get\_web().get\_lists().getByTitle('CompanyInfoList');

oListItem=oList.getItemById(dltItem); -------->Get the element by given ID

oListItem.deleteObject();

clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceeded),

Function.createDelegate(this, this.onQueryFailed));

}

function onQuerySucceeded(sender, args) {

$("#dltmsg").html("successfully deleted");

}

function onQueryFailed(sender, args) {

alert('request failed ' + args.get\_message() + '\n' + args.get\_stackTrace());

}

**SharePoint Branding:**

SharePoint Branding is the process of changing the look and feel of the SharePoint user interface using custom master pages, style sheets, images, JavaScript, jQuery and so on.  
  
  
  
A few of the branded sites are shown below.  
  
  
  
**Advantages of Branding**  
  
A good branding leads to:

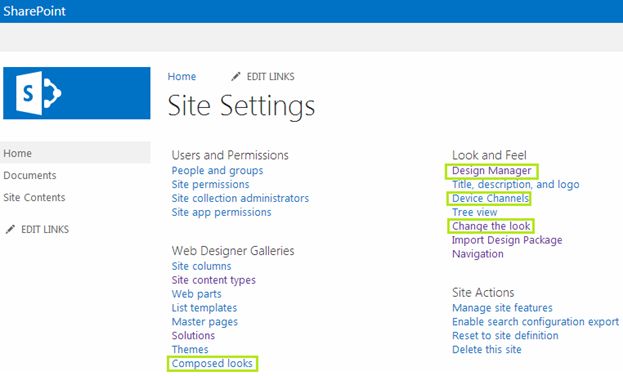
1. Better content presentation
2. Better navigation
3. Better look and feel
4. Enable responsive web design
5. Hide SharePoint default look

**Skills needed for Branding**

* HTML,
* CSS
* Basic Scripting skills and
* SharePoint Branding knowledge is an essential skill to integrate the custom branding artifacts.

**SharePoint Branding knowledge includes:**

* Master Page location
* Master Page elements like Site Settings, Quick Launch and so on.
* Composed looks

We can access these pages from Site Actions > Site Settings menu.  
  
  
  
**Note:**The Publishing feature needs to be activated for getting the Design Manager feature.  
  
**Branding Elements in SharePoint 2013**  
  
The following are the branding elements in SharePoint 2013:

1. Master Pages
2. Preview Files
3. CSS Files
4. Color Palettes
5. Font Palettes
6. Images