**Property of Sears Holdings India**

Purpose

This document is meant for understanding of the technical aspects of the implementation of the project.

**MEMBER OFFER SERVICE**

DETAILED TECHNICAL DOCUMENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Editor** | **Reviewer** | **Approver** | **Comments/Notes** |
| 0.0 baseline | Chakradhar DVS | Krish Narukulla | Krish Narukulla |  |

Contents

[1. Architecture 3](#_Toc466395097)

[2. User Interface (UI) 4](#_Toc466395098)

[2.1 Functional Overview 5](#_Toc466395099)

[2.1.0 Dashboard/Landing page 5](#_Toc466395100)

[2.1.1 View metrics in graphs and plots 6](#_Toc466395101)

[2.1.2 Create a campaign 7](#_Toc466395102)

[2.2 Code level overview 9](#_Toc466395103)

[2.2.0 Dashboard 9](#_Toc466395104)

[2.2.1 Metrics on dashboard 10](#_Toc466395105)

[2.2.2 Campaign Creation 11](#_Toc466395106)

[2.2.3 Validations 12](#_Toc466395107)

[2.2.3.0 Mandatory Validation: 12](#_Toc466395108)

[2.2.3.1 Min max validations: 12](#_Toc466395109)

[2.2.3.2 Length validation: 12](#_Toc466395110)

[2.2.3.3 Date validation: 13](#_Toc466395111)

[2.2.3.4 Currency validation: 13](#_Toc466395112)

# **Architecture**

The architecture adopted for MOS is 3 tier architecture.

# **User Interface (UI)**

* The User Interface (UI) layer for MOS is developed using HTML , Javascript and related libraries.

The details as below:

|  |  |
| --- | --- |
| Layout & Styling | HTML,CSS, Bootstrap |
| Scripting | Javascript,JQuery |
| Additional plugins | DataTable, Jqplot |

* The hierarchy/directory structure is as below:

Figure showing directory structure

## **Functional Overview**

* The purpose of the application is to facilitate the merchant /person logged in to view, create and analyze campaigns which drive the business flow.
* The flow here starts with the user loading the application/website with URL provided.
* The user can view existing campaign and analyze them deeper using the graphs/plots which will be explained in the next sections going forward.
* The User can also create campaigns which will call the relevant services. This triggers the service which send mails to all the customers mapped to the campaign with appropriate mails with an activation link.
* Once the activation link is clicked by the customer, the offers under the campaign will be activated.

### **2.1.0 Dashboard/Landing page**



Figure 1

* This is the landing page where the logged in user can view, create and analyze the campaigns.
* The user can see the campaigns in a table.
* The user can view the campaigns already present in the system by searching using a matching text and also sorting according to a column.
* The table also provides a feature of pagination where the user can choose to view a certain number of records/campaigns in a single page.
* The user can also navigate between pages of data by clicking on “previous” and “next” or page numbers present at the bottom of the table.
* The space on the right side of table is blank until a user clicks on a specific row in the table to view the metrics in diagrammatic view.
* The user can also create a new campaign by clicking on the “create campaign” button present on the top of the page just above the table.

### **2.1.1 View metrics in graphs and plots**

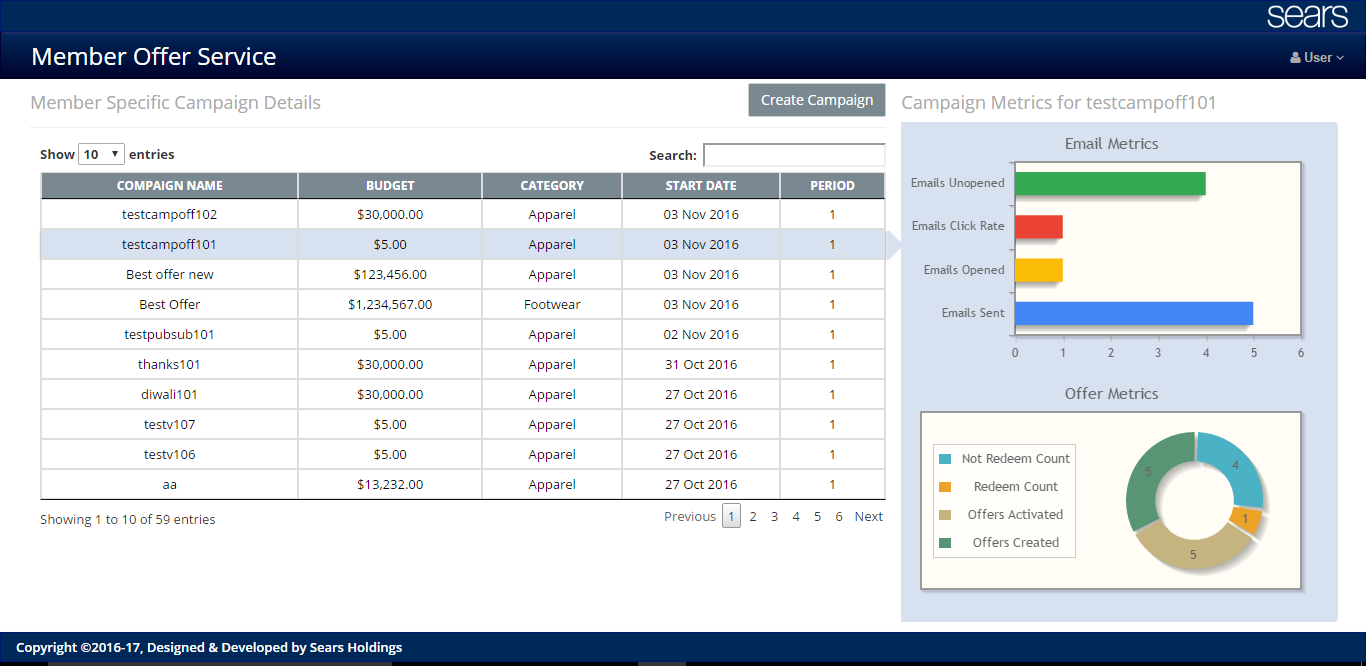


Figure 2

* This is the same dashboard page displaying the campaign specific metrics in diagrams.
* When a user clicks on a specific campaign in the table, the corresponding metrics are displayed in form of graphs/charts.
* The pictorial representation helps in better understanding of the campaign usage and other statistics.
* The graphs support some additional features when a user hovers mouse over them.

### **2.1.2 Create a campaign**

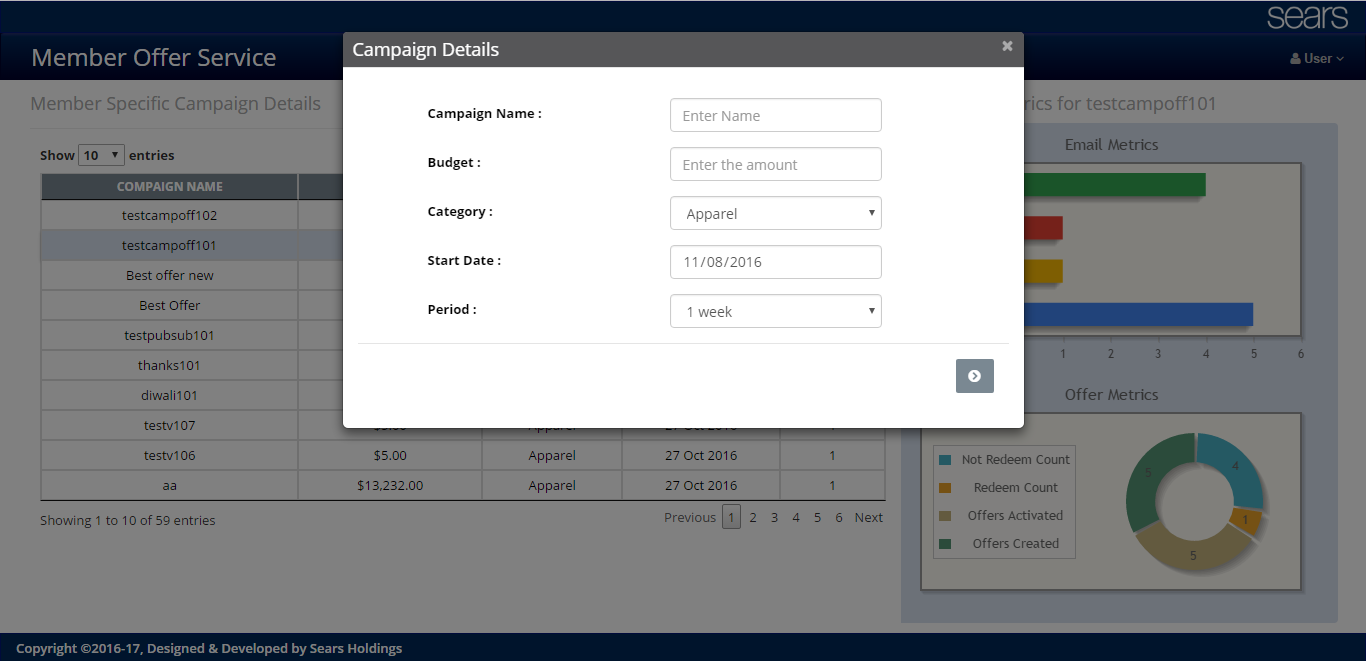


Figure 3

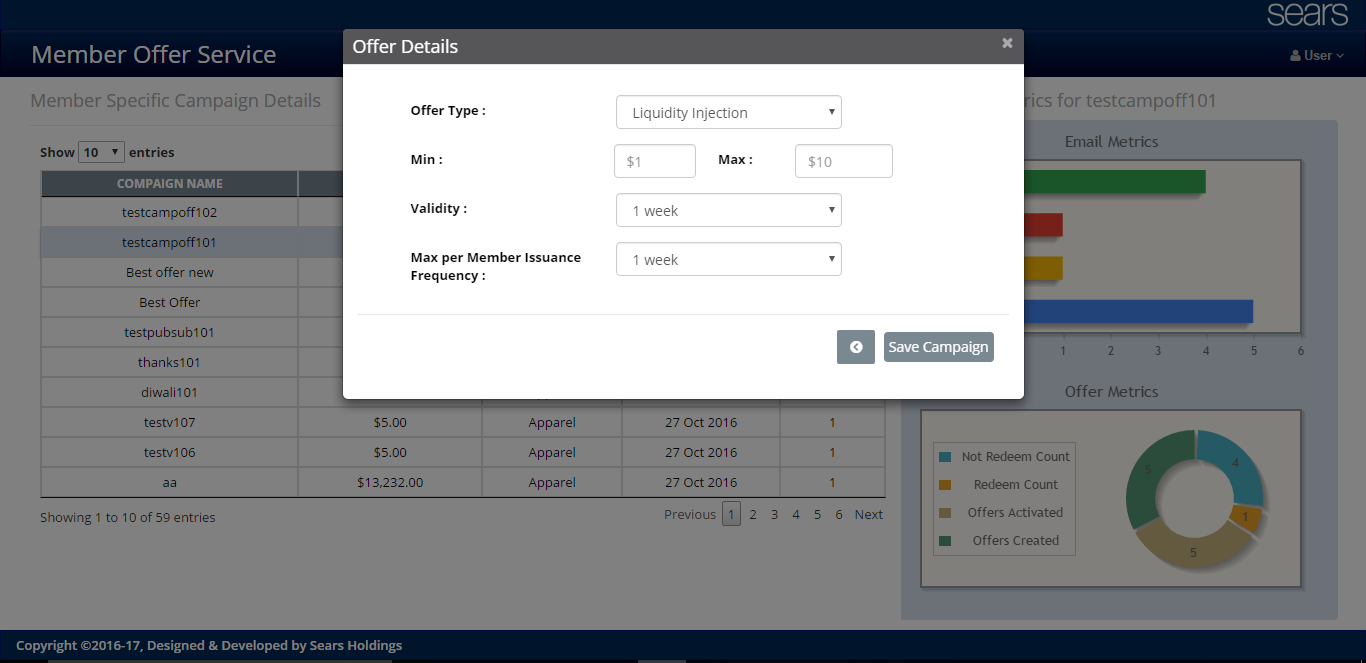


Figure 4

* Figure 3 and Figure 4 show the process of creating a campaign by clicking on the “create campaign” button on the dashboard/landing page.
* Table below shows the details of the inputs for creating a campaign.
* The user can navigate between the input screens any number of time until saving a campaign.
* Clicking on “save campaign” button on the last modal/screen creates the campaign and notifies the user with appropriate success or failure message.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Example |
| **Campaign details** |  | | |
|  |  | | |
| Campaign name | Yes | String | “Season ends” |
| Budget | Yes | Number | “money to spend in dollars” $3000 |
| Category | Yes | String | Apparel , Footwear |
| Period | Yes | Number | In terms of weeks |
| **Offer details** | | | | |
| Offer type | yes | String | Liquidity injection |
| Min value | yes | Number | 0-9 |
| Max value | yes | Number | 1-10 |
| Validity | Yes | Number | In terms of weeks |
| Member issuance/week | Yes | Number | In terms of weeks |

## **Code Level Overview**

* Here , in this section we will make an attempt to explain the underlying logic which is the backbone to make the application work seamless without any issues to the user.
* The purpose of this section is to familiarize the developer of the application with the core functionalities which are responsible for interacting with the user, interacting with the backend services and updating the user with the updated data.
* As mentioned in the previous sections, the person reading this section is expected to be having a good understanding of the architecture, relevant technologies, plugins used and the functional overview of the application.

### **2.2.0 Dashboard**

* As soon as the user loads the application, the dashboard page is loaded.
* When this page loads, an ajax call is made to the services to fetch data to be displayed in the table. The details are in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Method | Parameters | Required fields | Response |
| Campaigns  (<https://syw-offers-services-qa-dot-syw-offers.appspot.com/campaigns>) | GET | Null | Null | JSON array with campaign and offer details |

* The server responds with appropriate data and success/failure status.
* Once the data is received successfully form the server, the UI takes the responsibility of displaying the data in tabular form with the help of “DataTable” plugin.
* The plugin takes the data in the form of JSON and displays it according to the customization we define.
* The code for this is in “campaign.js” and the other relevant dependencies are mentioned in home.html as follows.

|  |
| --- |
| <script src="https://cdn.datatables.net/1.10.12/js/jquery.dataTables.min.js" type="text/javascript"></script>  <script src="default/js/plugins/jquery.cookie.min.js" type="text/javascript"></script> |

### **Metrics on dashboard**

* As mentioned earlier, the metrics will show up when the user wishes to see them by clicking on the relevant campaign in the table.
* As soon as a user clicks on a specific campaign row, UI captures the selected campaign details and makes a service call to fetch the corresponding metrics data. The details of the service call as below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Method | Parmas | Required fields | Response |
| getMetrics  Example:  (<https://syw-offers-services-qa-dot-syw-offers.appspot.com/getMetrics?campaign_id=testcampoff102>) | GET | campaign\_id | YES | JSON object of email\_metrics and offer\_metrics |

* The service responds with a JSON containing email and offer metrics.
* This data is converted to 2- dimensional array and fed to the jqplot plugin.
* Jqplot is an open source chart plugin which needs JQuery.
* In order to work with specific kind of a graph of jqplot, we need to import the corresponding dependencies into the project folder and refer them from home.html as below:

|  |
| --- |
| <script type="text/javascript" src="default/js/jqplot/jquery.jqplot.js"></script>  <script type="text/javascript" src="default/js/jqplot/plugins/jqplot.barRenderer.js"></script>  <script type="text/javascript" src="default/js/jqplot/plugins/jqplot.donutRenderer.js"></script>  <script type="text/javascript" src="default/js/jqplot/plugins/jqplot.categoryAxisRenderer.js"></script>  <script type="text/javascript" src="default/js/jqplot/plugins/jqplot.canvasAxisLabelRenderer.js"></script>  <script type="text/javascript" src="default/js/jqplot/plugins/jqplot.canvasTextRenderer.js"></script>  <script type="text/javascript" src="default/js/jqplot/plugins/jqplot.highlighter.js"></script> |

* The bar chart used to display email metrics requires
  + jquery.jqplot.js
  + jqplot.barRenderer.js
  + jqplot.categoryAxisRenerer.js
  + jqplot.canvasAxisLabelRenderer.js
  + jqplot.canvasTextRenderer.js
  + jqplot.highlighter.js
* The donut plot to display offer metirics requires
  + jquery.jqplot.js
  + jqplot.donutRenderer.js
* All the relevant logic to render jqplot graphs is present in “CampaignGraphs.js”

### **2.2.2 Campaign Creation**

* Campaign creation is one of the main purposes the application is built for.
* When a user clicks on create campaign button , a series of modals are invoked which contain relevant input fields.
* The inputs and boundary conditions for them are mentioned in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Example |
| **Campaign details** |  | | |
|  |  | | |
| Campaign name | Yes | String | “Season ends” |
| Budget | Yes | Number | “money to spend in dollars” $3000 |
| Category | Yes | String | Apparel , Footwear |
| Period | Yes | Number | In terms of weeks |
| **Offer details** | | | | |
| Offer type | yes | String | Liquidity injection |
| Min value | yes | Number | 0-9 |
| Max value | yes | Number | 1-10 |
| Validity | Yes | Number | In terms of weeks |
| Member issuance/week | Yes | Number | In terms of weeks |

* All the necessary validations are taken care of to ensure the user enters the correct data.
* All the logic/ functionality for the validations is present in “utils.js”.
* Once all the validations are passed and the user clicks on “save campaign” button, the UI makes a call to service with all the relevant campaign related data entered by the user. The details are as below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Method | Parmas | Required fields | Response |
| saveCampaign | GET | campaign\_name, money(budget), start\_date, end\_date, period, offer\_type, min\_value, max\_value, validity, member\_issuance | All fields are required | Success |

* Once the service responds with a status, a corresponding message is displayed to the user and the dashboard table is refreshed to reflect the newly created campaign.

### **Validations**

* In order to make a smooth glitch free experience and as part of QA, some validations are required in this application.
* The validations required are in a separate file called “utils.js” under js folder.
* These are implemented and separated from the main logic so that these are generic and can be used wherever necessary throughout the application.
* The main validations present are:

### 2.2.3.0 Mandatory Validation:

* + - This validation is done each time a user navigates between modals.
    - To make this validation, the input fields need to have a class as “mandatory”.
    - The signature of this function is :
      * ***checkMandatory(selector)***
* The selector is the class of a modal or a div under which the validations have to be done on the input fields.
* Returns “true” if the validation passed and “false” if the validations fail.

### 2.2.3.1 Min max validations:

* + This validation is done when the min and max values are to be compared against each other and also against boundaries according to business logic.
  + The signature is
    - ***validations(element,comparand,operation)***
  + Here, the “element” is the id of the element, “comparand” is the id of the other input field for comparison and “operation” takes 0 or 1 based on greater than or lesser than operation.

### 2.2.3.2 Length validation:

* + This validation is used when the system needs to restrict the user from entering values of length greater than mentioned in the business logic.
  + The signature is as follows:
    - ***positiveCurrency(element)***
* The “element” is the id of the input element for which the length of input has to be validated.

### 2.2.3.3 Date validation:

* This validation is used for various date validations.
* The signature is as follows:
  + ***dateValidation(element)***
* The “element” is the id of the input element which has to be validated.

### 2.2.3.4 Currency validation:

* This validation is introduced to make sure that the currency fields take only positive values.
* Method signature is as follows:
  + positiveCurrency(element)
* The “element” is the id of the input element which has to be validated.