**A PROJECT REPORT**

**ON**

**Product Sales Dashboard**

**Submitted to**



**Project Report**

**on**

**Product Sales Dashboard**

**In the partial fulfillment of the requirement for the Degree of**

**Master of Computer Application**

**Submitted by**

**Vivek Dhakre (1405006452)**

[OBJECTIVE 1](#_Toc222596668)

[Project Scope 2](#_Toc222596669)

[Customer, Products, Billing Generation 2](#_Toc222596670)

[Report Generation System 2](#_Toc222596671)

[Out of Scope 3](#_Toc222596672)

[Goal 3](#_Toc222596673)

[Solution Concept 3](#_Toc222596674)

[User Profiles 5](#_Toc222596675)

[Usage Summary 6](#_Toc222596676)

[Requirements Summary 7](#_Toc222596677)

[Business Requirements 7](#_Toc222596678)

[User Requirements 7](#_Toc222596679)

[Operational Requirements 8](#_Toc222596680)

[System Requirements 9](#_Toc222596681)

[Success Criteria 9](#_Toc222596682)

[Sales Automation 9](#_Toc222596683)

[Design Goals and Constraints 10](#_Toc222596684)

[Performance 10](#_Toc222596685)

[Availability 10](#_Toc222596686)

[Reliability 10](#_Toc222596687)

[Scalability 10](#_Toc222596688)

[Security 10](#_Toc222596689)

[Interoperability 10](#_Toc222596690)

[Location 10](#_Toc222596691)

[Setup/Installation 10](#_Toc222596692)

[Features/Functionality 10](#_Toc222596693)

[Software and Hardware Requirement 13](#_Toc222596694)

[General Requirement for Server/Client: 13](#_Toc222596695)

[Development Tools and Technologies 13](#_Toc222596696)

[Development Tools and Technologies 14](#_Toc222596697)

[Front End 14](#_Toc222596698)

[Back End 14](#_Toc222596699)

[Development Environment 15](#_Toc222596700)

[Use Case Summary 16](#_Toc222596701)

[Manage Orders 16](#_Toc222596702)

[Description 16](#_Toc222596703)

[Business Need 16](#_Toc222596704)

[Priority 16](#_Toc222596705)

[Manage Products 16](#_Toc222596706)

[Description 16](#_Toc222596707)

[Business Need 16](#_Toc222596708)

[Priority 16](#_Toc222596709)

[Manage Contacts 16](#_Toc222596710)

[Description 16](#_Toc222596711)

[Business Need 16](#_Toc222596712)

[Priority 17](#_Toc222596713)

[Analyze Customers 17](#_Toc222596714)

[Description 17](#_Toc222596715)

[Business Need 17](#_Toc222596716)

[Priority 17](#_Toc222596717)

[Forecast Sales 17](#_Toc222596718)

[Description 17](#_Toc222596719)

[Business Need 17](#_Toc222596720)

[Priority 17](#_Toc222596721)

[Establish Sales Goals 17](#_Toc222596722)

[Description 17](#_Toc222596723)

[Business Need 17](#_Toc222596724)

[Priority 17](#_Toc222596725)

[Usage Scenarios 18](#_Toc222596726)

[Use Case ID: UC 4.7.1 Retrieve Customer Data 18](#_Toc222596727)

[Intent 18](#_Toc222596728)

[Scenario Narrative 18](#_Toc222596729)

[Assumptions/Preconditions 18](#_Toc222596730)

[Actors 18](#_Toc222596731)

[Basic Course 18](#_Toc222596732)

[Alternate Course 18](#_Toc222596733)

[Uses/Extends 18](#_Toc222596734)

[User Implementation Requests 18](#_Toc222596735)

[Frequency 18](#_Toc222596736)

[Authority 18](#_Toc222596737)

[Associated Specifications 19](#_Toc222596738)

[Issues 19](#_Toc222596739)

[Decision Points 19](#_Toc222596740)

[Future Requirements 19](#_Toc222596741)

[Modification 19](#_Toc222596742)

[Use Case Model: Retrieve Customer Data 20](#_Toc222596743)

[Use Case ID: UC 04.7.2 Manage Products 21](#_Toc222596744)

[Intent 21](#_Toc222596745)

[Scenario Narrative 21](#_Toc222596746)

[Assumptions/Preconditions 21](#_Toc222596747)

[Actors 21](#_Toc222596748)

[Basic Course: 21](#_Toc222596749)

[Alternate Course: 21](#_Toc222596750)

[Uses/Extends 21](#_Toc222596751)

[User Implementation Requests 21](#_Toc222596752)

[Frequency 21](#_Toc222596753)

[Authority 21](#_Toc222596754)

[Associated Specifications 21](#_Toc222596755)

[Issues 22](#_Toc222596756)

[Decision Points 22](#_Toc222596757)

[Future Requirements 22](#_Toc222596758)

[Modification 22](#_Toc222596759)

[Use Case: Manage Products 23](#_Toc222596760)

[Use Case ID: UC 04.7.3 Manage Orders 24](#_Toc222596761)

[Intent 24](#_Toc222596762)

[Scenario Narrative 24](#_Toc222596763)

[Assumptions/Preconditions 24](#_Toc222596764)

[Actors 24](#_Toc222596765)

[Basic Course: 24](#_Toc222596766)

[Uses/Extends 24](#_Toc222596767)

[User Implementation Requests 24](#_Toc222596768)

[Frequency 24](#_Toc222596769)

[Authority 24](#_Toc222596770)

[Associated Specifications 25](#_Toc222596771)

[Issues 25](#_Toc222596772)

[Decision Points 25](#_Toc222596773)

[Future Requirements 25](#_Toc222596774)

[Modification 25](#_Toc222596775)

[Use Case ID: UC 04.7.3 Manage Contacts 27](#_Toc222596776)

[Intent 27](#_Toc222596777)

[Scenario Narrative 27](#_Toc222596778)

[Assumptions/Preconditions 27](#_Toc222596779)

[Actors 27](#_Toc222596780)

[Basic Course: 27](#_Toc222596781)

[Uses/Extends 27](#_Toc222596782)

[User Implementation Requests 27](#_Toc222596783)

[Frequency 27](#_Toc222596784)

[Authority 27](#_Toc222596785)

[Associated Specifications 27](#_Toc222596786)

[Issues 28](#_Toc222596787)

[Decision Points 28](#_Toc222596788)

[Future Requirements 28](#_Toc222596789)

[Modification 28](#_Toc222596790)

[Solution Architecture 30](#_Toc222596792)

[Infrastructure 30](#_Toc222596793)

[Visual Design 30](#_Toc222596794)

[Conceptual Design 30](#_Toc222596795)

[Logical Design 32](#_Toc222596796)

[Physical Design 34](#_Toc222596797)

[eBilling and Invoice System Data Dictionary 35](#_Toc222596798)

[eBilling and Invoice System Data Dictionary 36](#_Toc222596799)

[Table : product\_master 36](#_Toc222596800)

[Table : bill 36](#_Toc222596801)

[Table : bill\_details 37](#_Toc222596802)

[Table : Company\_master 37](#_Toc222596803)

[Table : user\_master 38](#_Toc222596804)

[eBilling and Invoice System – Data Schema ER Diagram 39](#_Toc222596805)

[eBilling and Invoice System DFD (Data Flow Diagram) 40](#_Toc222596806)

[Form Design and Coding 41](#_Toc222596807)

[FrmSplash 41](#_Toc222596808)

[FrmLogin 43](#_Toc222596809)

[FrmMain 46](#_Toc222596810)

[FrmProducts 48](#_Toc222596811)

[FrmBill 54](#_Toc222596812)

[FrmReports 71](#_Toc222596814)

[FrmPrintBill 73](#_Toc222596815)

[ProductSummary Report 75](#_Toc222596816)

[FrmBillSummary 76](#_Toc222596817)

[FrmExportData 78](#_Toc222596819)

[FrmUtlity 84](#_Toc222596820)

[FrmCompactAndRepairDatabase 86](#_Toc222596821)

[FrmBackupDatabase 88](#_Toc222596822)

[FrmCompanyInfo 91](#_Toc222596823)

[FrmClientInfo 96](#_Toc222596824)

[FrmChangePassword 101](#_Toc222596825)

[ModGen 103](#_Toc222596826)

[ModBrowse 105](#_Toc222596827)

[ModRsToWord 107](#_Toc222596828)

[eBilling and Invoice System Test Plan 111](#_Toc222596829)

[Introduction 111](#_Toc222596830)

[Test Scope 111](#_Toc222596831)

[Test Strategy 111](#_Toc222596832)

[Preconditions 112](#_Toc222596833)

[Test Priorities 112](#_Toc222596834)

[Test Techniques 112](#_Toc222596835)

[Test Organization 113](#_Toc222596836)

[Roles and Responsibilities 113](#_Toc222596837)

[Deliverables 113](#_Toc222596838)

[Test Environment 113](#_Toc222596839)

[Hardware and Software 113](#_Toc222596840)

[Testing Automation Software 114](#_Toc222596841)

[Application Configuration 114](#_Toc222596842)

[Test Management 114](#_Toc222596843)

[Testing Schedules 114](#_Toc222596844)

[Threats to Testing 115](#_Toc222596845)

[Conclusion and future enhancement 116](#_Toc222596846)

[Bibliography 117](#_Toc222596847)

[Websites 117](#_Toc222596848)

[Books 117](#_Toc222596849)

# OBJECTIVE

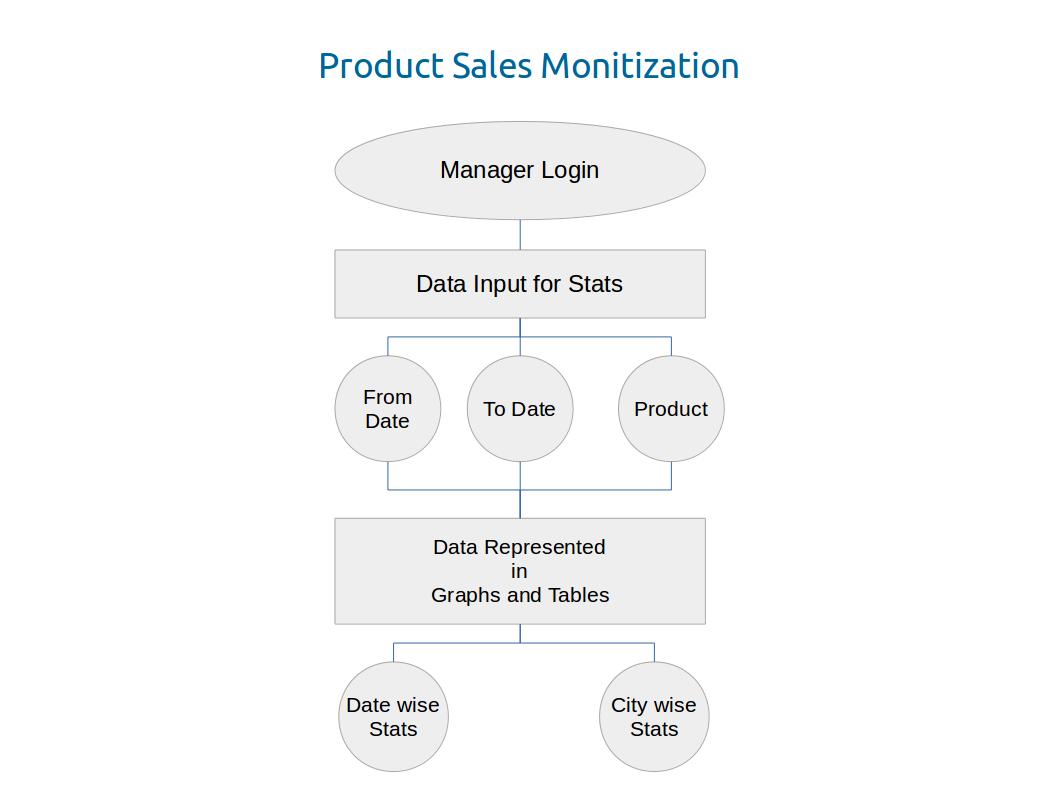
A Sales or Marketing Personnel always wants to monetize the up sale of there product. Though he/she needs an application or dashboard where that monetization can be done.

Product Sales Dashboard is basically web based application. In this dashboard we can see Specific product sale report between from date to to date. In this application you can monetize up sale stats in Chart Form. Product Sold statistics from date to to date of multiple store in column chart, You can see city wise statistic in column chart.

Sales and Marketing personnel are interested in viewing sales performance information, such as what is the sales status from selected date a day wise and also want to see statistics from selected date by city wise but group with stores.

We have decided to implement a Sales dashboard to provide specific product sales statistics across the organization with the information they need. The Sales dashboard includes a variety of chart, arranged in two or more dashboard pages.

**Walk Through**



## Usage Summary

Product Sales Dashboard will address the following use cases.

1. Monetize of sale store wise with respect to date.
2. Monetize of sale store wise with respect to city.

The complete usage scenarios will be completed during the information-gathering process. Use cases will be created and prioritized. Selected use cases will be expanded into usage scenarios and features that are derived from both use cases and the usage scenarios.

## Requirements Summary

The following preliminary lists are based on initial interviews

### Business Requirements

The goal for the application is to support an management guy to see the statistics of specific product according to the date wise with in specific time period with respect to date, store wise with respect to city .

Business requirements are discussed in the Scope section, with the following additional detail:

* Each sales representative should have a login credentials to access the application.
* Sales representatives need to login our application and after that they need to select a date interval and product of which statistics they want to monetize.
* Sales or Marketing representative don’t need to update or upload sold data.
* At the time of billing, POS will notify or update the sales record.
* This application fulfills the utility of Sales and Marketing representative to monetize the sale.
* No manual intervention required from sales guy like excel work, mis.

Sales or Marketing representative wants to improve their current ability to analyze their product sale. In particular, they want to focus on identifying their best product sales and concentrate to sale that product efficiently. To enable them to accomplish this goal, they want to extract meaningful data that easily answers the following questions:

* How to get Product Sales Data?
* What is the product sales?
* Which product sales we need to concentrate?
* Geographically, See the best product sales in the city?
* Which product sales are too much and what is the price of that product?
* Which product is getting back locked?
* Which product sold frequently based on days?

### User Requirements

User requirements are categorized by user type.

**Sales Staff (Representatives and Managers)**

* View the data in various ways, for example:
  + which product is having high sale?
  + Which product is getting back locked?
  + Which store are performing best?
  + Which store need to focus on sale?
  + In which city product have high demand?
  + In which city we need to focus for sale?
* Monetize the sale report of a store, having multiple product and multiple variant, rather than relying on the sales staff to collect the sales report.
* Identify which product is selling fast and running out, which is still in stock.
* Which store are needed focus that can help the sales representative convert an opportunity into a sale.
* Identify where promotions and programs would be the most beneficial
* Apply discounts to product orders:
  + Sales representatives can offer discounts up to 15 percent, or up to 20 percent with authorization
* Enable capture, analysis, and sharing of data about a product across the company
* Use forecasts to establish sales goals

### System Requirements

These are additional constraints from a system perspective:

* Sales data needs to be updated by the billing software or POS on runtime.
* Sales person must have an idea about the POS notification to application.
* Sales Person must know, How to operate the application.
* The applocation must be accessible by your sales or Marketing representative, who want to monetize.

## Success Criteria

To determine the success of this project, the following metrics can be quantified and used to analyze success factors.

### Sales Automation

* **Product down sale identification** – We can easily monetize back lock products, so that we can focus on that product.
* **Best store in a city** – We can modify, Which product is properly sale on which store.

# Design Goals and Constraints

## Performance

No more than a 5-percent degradation in average query response is allowed while all concurrent user are using the system.

Processor utilization should not exceed 80 percent during all concurrent users are using the system.

## Availability

Because the system is accessed by sales or marketing representative to monetize the sale of specific product according to date or city based on date interval, and their should not be any single point of failure.

## Reliability

Because of the need no single point failure, automatic notification will be required at the time of failure. In addition, existing disaster recovery and backup plans and procedures must be revised to incorporate the Product Sales Stats Dashboard.

## Scalability

Product Sales Stats Dashboard an average load of 50 concurrent users after the system is fully operational.

## Security

For the sensitive product sales data, all users will need to log on the system with their user id and password.

## Interoperability

In Version 1.0 of the Product Sales Stats Dashboard, there is a requirement for interoperability with other systems.

## Location

The Product Sales Stats Dashboard are used in the company office only.

## Setup/Installation

Setup and installation must not interrupt the sales staff’s daily tasks and work flow.

# Use Case Summary

## Manage Sale

### Description

This function will enable sales representative to monetize the daily, monthly sale of various product based on stores and cities .

### Business Need

This function will enable sales representative to interact and collect the sales and data directly without the intervention of any other employees.

### Priority

High

## Analyze Customers

### Description

This function allows management to analyze the product sale and find out information, such as the best product sale or worst product sale..

### Business Need

This function will allow management to determine the most profitable store and product lines, enabling better decision making in the running of the business.

### Priority

Medium

## Forecast Sales

### Description

This function allows management to plan and track production sales results, and plan sales staffing needs.

### Business Need

This function will allow management to manage the revenue streams of the organizations and control costs.

### Priority

Medium

## Establish Sales Goals

### Description

This function allows management to view current sales trends and store performance and to determine sales goals for the store.

### Business Need

This function will allow management to set goals for product sale, review the performance of stores, and track product performance.

### Priority

Medium

## Usage Scenarios

Usage scenarios that describe how sales representative will monetize the product sold stats citywise or a day wise, See following usage scenarios.

### Use Case ID: UC 001 Retrieve Product Statistics

From Date: 05/10/2017

To Date: 05/15/2017

Product Id: 1

### *Intent*

Return the selected information about a product.

### *Scenario Narrative*

**A sales representative may want to retrieve information about one product for analysis and forecasting activities.**

### *Assumptions/Preconditions*

1. Sales representative has access rights to view sales statistics.

### *Actors*

1. Sales Representative

### *Basic Course*

1. Use case begins when the sales representative decides to retrieve product sales states.
2. System prompts for the type of input that is required.
3. Sales representative selects the date interval and product to see stats.
4. System presents the appropriate product stats to the sales representative.
5. Use case ends when the stats is showed by a selected input.

### *Uses/Extends*

1. None

### *User Implementation Requests*

1. None

### *Frequency*

Quite frequent

### *Authority*

1. applicable

### *Issues*

1. None identified

### *Decision Points*

1. None

### *Future Requirements*

1. The sales representative may want to apply filters to the retrieved sales data.
2. The sales representative may want to sort the data. For example, the sales representative may want to list, which product sales is best.

# Solution Architecture

## Infrastructure

New servers must be dual processor 1.26 GHz computers, with 4 GB of RAM. The operating system software will be Ubuntu Linux, although it will work for all operating System.

Web Browser

These servers will be installed on existing network.

To help provide security of the application, both an internal and external firewall will be installed.

# 

Product Sales Dashboard Data Dictionary

***Database Name:*** *psd (product sales dashboard)*

## Table : user

**Number of columns:** 5

**Number of indexes:** 0

**Number of foreign keys:** 0

**Primary Key:** 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Columns** | **Data type** | **Allow NULLs** | **Value/range** |
| **Id** | bigint(20) | Not allowed | Primary key |
| **User\_name** | varchar(20) | Not allowed |  |
| **Password** | varchar(20) | Not allowed |  |
| **Role** | Varchar(20) | allowed |  |
| **created\_on** | Timestamp | Not allowed |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Id** | Primary key |  |

## Table : stores

**Number of columns:** 3

**Number of indexes:** 0

**Number of foreign keys:** 0

**Extended attributes:**

Primary Key: 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Columns** | **Data type** | **Allow NULLs** | **Value/range** |
| **id** | bigint(20) | Not allowed |  |
| **Name** | varchar(20) | Not allowed |  |
| **created\_on** | timestamp | Not allowed |  |

|  |  |  |
| --- | --- | --- |
| **Indexes** | **Type** | **Columns** |
| **id** | Primary key | sno |

## Table : citys

**Number of columns:** 3

**Number of indexes:** 0

**Number of foreign keys:** 0

**Extended attributes:**

Primary Key: 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Columns** | **Data type** | **Allow NULLs** | **Value/range** |
| **id** | bigint(20) | Not allowed |  |
| **Name** | varchar(20) | Not allowed |  |
| **created\_on** | timestamp | Not allowed |  |

|  |  |  |
| --- | --- | --- |
| **Indexes** | **Type** | **Columns** |
| **id** | Primary key | sno |

## Table : product

**Number of columns:** 3

**Number of indexes:** 0

**Number of foreign keys:** 0

**Extended attributes:**

Primary Key: 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Columns** | **Data type** | **Allow NULLs** | **Value/range** |
| **id** | bigint(20) | Not allowed |  |
| **Name** | varchar(20) | Not allowed |  |
| **created\_on** | timestamp | Not allowed |  |

|  |  |  |
| --- | --- | --- |
| **Indexes** | **Type** | **Columns** |
| **id** | Primary key | sno |

## Table : sold

**Number of columns:** 7

**Number of indexes:** 2

**Number of foreign keys:** 0

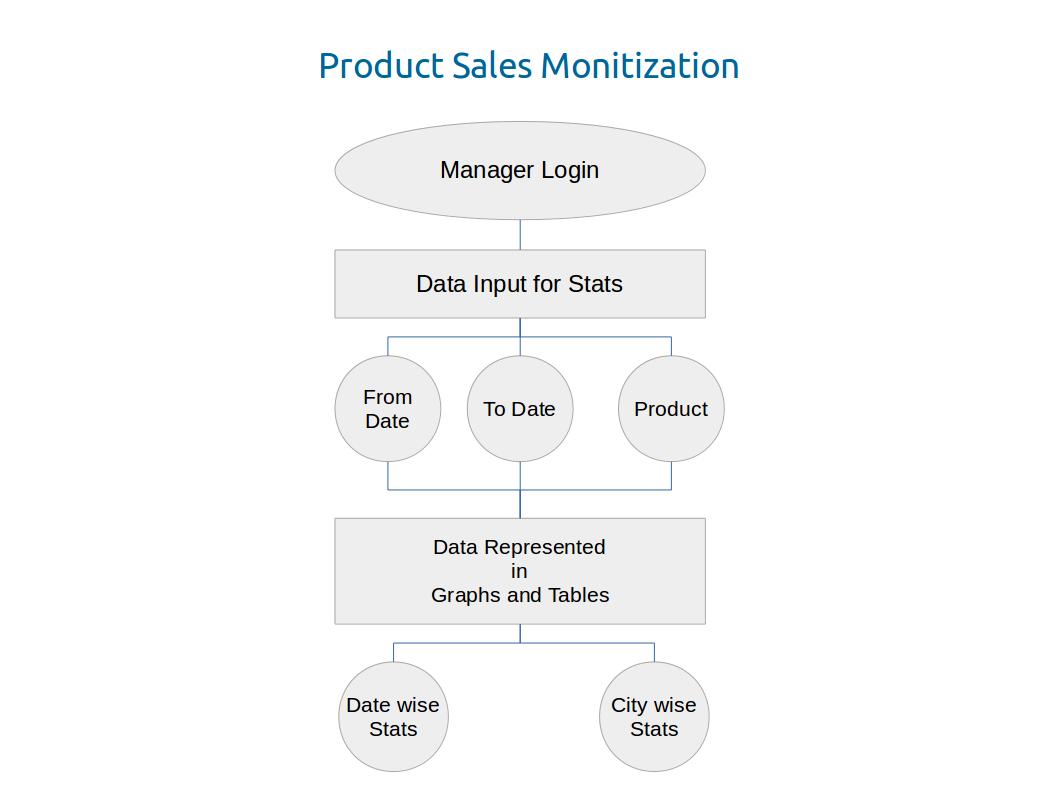
primary key: 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Columns** | **Data type** | **Allow NULLs** | **Value/range** |
| **Id** | Bigint(20) | Not allowed |  |
| **campaign\_fkey** | Bigint(20) | Not allowed |  |
| **Count** | Int(11) | Not Allowed |  |
| City | Varchar(20) | Not Allowed |  |
| Store | Varchar(20) | Not Allowed |  |
| created\_on | Timestamp | Not null |  |

|  |  |  |
| --- | --- | --- |
| **Indexes** | **Type** | **Columns** |
| **Id** | Primary key |  |
| **campaign\_fkey** | Foreign Key |  |

# eBilling and Invoice System – Data Schema ER Diagram

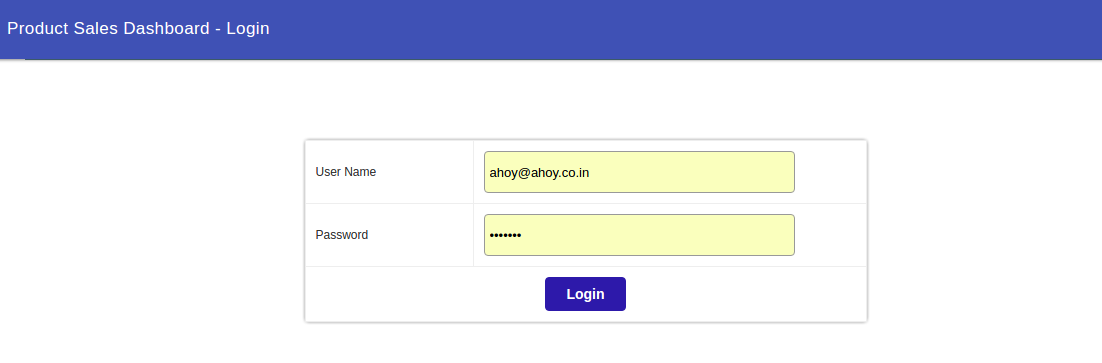
# eBilling and Invoice System DFD (Data Flow Diagram)



# Html Design and Coding

### Login Page

(Module to authenticate a user)

****

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Product Sales Dashboard

' Version 1.0.0

' Created by Vivek Dhakre

' Date : 17-May-2017

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Login Page

' Used Table : user

'Module to authenticate a user

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**<!DOCTYPE html>**

<**html lang="en" ng-app="Dashboard"**>

<**head**>

<**meta charset="UTF-8"**>

<**meta name="viewport" content="width=device-width"**>

<**title**>Product Sales Dashboard</**title**>

<**link rel="stylesheet" href="<%=**request.getContextPath()**%>/chart/css/style.css" type="text/css"**>

<**link type="text/css" rel="stylesheet" href="<%=**request.getContextPath()**%>/chart/css/material.css"**>

<**link rel="stylesheet" type="text/css" href="<%=**request.getContextPath()**%>/css/style.css"**>

<**link rel="stylesheet" type="text/css" href="<%=**request.getContextPath()**%>/css/font-awesome.css"**>

<**script src="<%=**request.getContextPath()**%>/js/jquery-1.10.2.js"**></**script**>

</**head**>

<**script type="text/javascript"**>

**$**(**document**).ready(**function** () {

**$**(**'#sbmtn'**).click(**function** () {

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

**if** (username == **''**) {

alert(**"Please Enter User Name"**);

**$**(**"#username"**).**focus**();

**return false**;

}

**var** pwd = **$**(**"#password"**).val();

**if** (pwd == **''**) {

alert(**"Please Enter Password"**);

**$**(**"#pwd"**).**focus**();

**return false**;

}

**return true**;

});

});

</**script**>

<**body**>

<**div class="mdl-layout mdl-js-layout mdl-layout--fixed-header"**>

<**header class="mdl-layout\_\_header"**>

<**div class="mdl-layout\_\_header-row"**>

*<!-- Title -->*

<**span class="mdl-layout-icon brand-logo pos-left"**></**span**>

<**span class="mdl-layout-title mdl-layout--large-screen-only"**>

<**small class="mdl-color-text--white"**>

Product Sales Dashboard - Login

</**small**></**span**>

<**div class="mdl-layout-spacer"**></**div**>

</**div**>

</**header**>

<**main class="mdl-layout\_\_content"**>

<**div class="auto-1000"**>

<**div class="main-container"**>

<**div class="report-container"**>

<**h2**></**h2**>

<**div class="select-form"**>

<**form action="loginservlet" method="post"**>

<**table**>

<**tr**>

<**td style="width**:30%;**text-align**: **left**;**"**>

<**label**>User Name</**label**></**td**>

<**td**>

<**input type="text" name="username" id="username" placeholder="Enter Username"**/>

</**td**>

</**tr**>

<**tr**>

<**td style="width**:30%;**text-align**: **left**;**"**>

<**label**>Password</**label**></**td**>

<**td**>

<**input type="password" name="password" id="password" placeholder="Enter Password"**>

</**td**>

</**tr**>

<**tr**>

<**td colspan="2"**><**input type="Submit" id="sbmtn" value="Login"**></**td**>

</**tr**>

</**table**>

</**form**>

</**div**>

</**div**>

</**div**>

</**div**>

</**main**>

</**div**>

</**body**>

</**html**>

**package** edu.mca.servlet;

**import** edu.mca.util.DbConnection;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.servlet.http.HttpSession;

**import** java.io.IOException;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

*/\*\**

*\* Created by vivek on 10/5/17.*

*\*/*

@WebServlet(**"/loginservlet"**)

**public class** LoginServlet **extends** HttpServlet {

**protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

String userName = request.getParameter(**"username"**);

String password = request.getParameter(**"password"**);

**try** {

**if** (userName != **null** && !userName.trim().equals(**""**)

&& password != **null** && !password.trim().equals(**""**)) {

Connection connection = DbConnection.*getConnection*();

PreparedStatement pst = connection.prepareStatement(**"select user\_name,password,role from users where user\_name=? and password=?"**);

pst.setString(1, userName);

pst.setString(2, password);

ResultSet rst = pst.executeQuery();

**if** (rst != **null** && rst.next()) {

HttpSession session = request.getSession();

session.setAttribute(**"user"**, rst.getString(1));

session.setAttribute(**"role"**, rst.getString(2));

response.sendRedirect(**"home"**);

} **else** {

response.sendRedirect(**"login?status=true"**);

}

} **else** {

response.sendRedirect(**"login?status=true"**);

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**package** edu.mca.servlet;

**import** edu.mca.util.DbConnection;

**import** javax.servlet.RequestDispatcher;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.servlet.http.HttpSession;

**import** java.io.IOException;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

*/\*\**

*\* Created by vivek on 10/5/17.*

*\*/*

@WebServlet(**"/home"**)

**public class** Home **extends** HttpServlet {

**protected void** process(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

**try** {

HttpSession session = request.getSession();

String user = (String) session.getAttribute(**"user"**);

String role = (String) session.getAttribute(**"role"**);

**if** (user != **null** && !**""**.equals(user.trim()) && !**"null"**.equalsIgnoreCase(user.trim())

&& role != **null** && !**""**.equals(role.trim()) && !**"null"**.equalsIgnoreCase(role.trim())) {

Connection cn = DbConnection.*getConnection*();

PreparedStatement pst = cn.prepareStatement(**"select id,name from campaign"**);

ResultSet rst = pst.executeQuery();

request.setAttribute(**"rst"**, rst);

RequestDispatcher rd = request.getRequestDispatcher(**"/WEB-INF/jsps/selectdate.jsp"**);

rd.forward(request, response);

} **else** {

response.sendRedirect(**"login?status=true&msg=Session+Expired+or+Invalid"**);

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

**protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

process(request, response);

}

**protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

process(request, response);

}

}

### Select date and Product

(for see the stats)

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

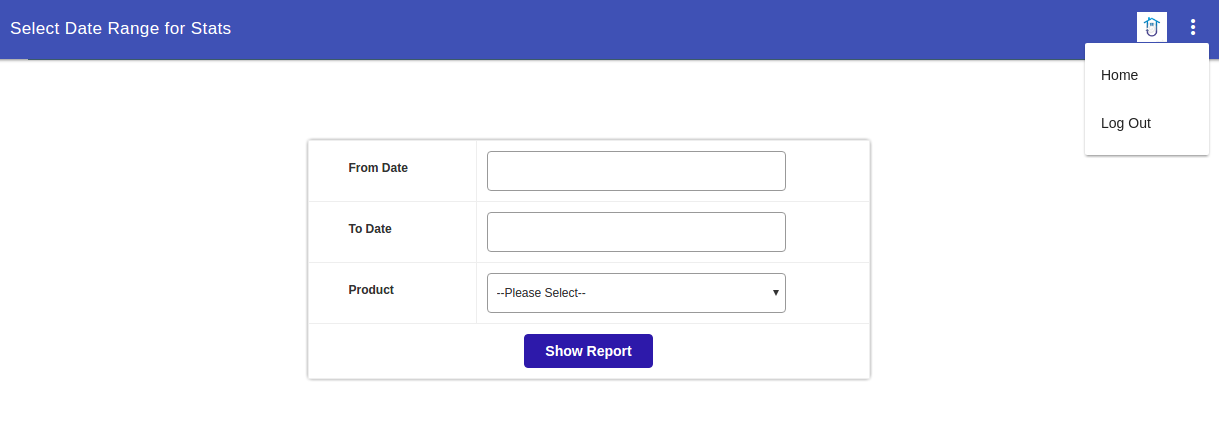
' Product Sales Report

' Version 1.0.0

' Created by Vivek Dhakre

' Date : 17-05-2017

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



<%@ **page import**="**java.sql.ResultSet**" %>

<!DOCTYPE **html**>

<**html lang="en" ng-app="Dashboard"**>

<**head**>

<**meta charset="UTF-8"**>

<**meta name="viewport" content="width=device-width"**>

<**title**>Product Sales Dashboard</**title**>

<**link rel="stylesheet" href="<%=**request.getContextPath()**%>/chart/css/style.css" type="text/css"**>

<**link type="text/css" rel="stylesheet" href="<%=**request.getContextPath()**%>/chart/css/material.css"**>

<**link rel="stylesheet" type="text/css" href="<%=**request.getContextPath()**%>/css/style.css"**>

<**link rel="stylesheet" type="text/css" href="<%=**request.getContextPath()**%>/css/font-awesome.css"**>

<**link rel="stylesheet" type="text/css" href="<%=**request.getContextPath()**%>/css/font-awesome.min.css"**>

<**link rel="stylesheet" type="text/css" href="<%=**request.getContextPath()**%>/css/bootstrap.min.css" media="screen"**>

<**link rel="stylesheet" type="text/css" href="<%=**request.getContextPath()**%>/css/bootstrap-datepicker.css"**

**media="screen"**>

<**script src="<%=**request.getContextPath()**%>/js/jquery-1.10.2.js"**></**script**>

<**script type="text/javascript" src="<%=**request.getContextPath()**%>/chart/js/material.js"**></**script**>

<**script type="text/javascript" src="<%=**request.getContextPath()**%>/js/bootstrap-datepicker.js"**></**script**>

<**script**>

**$**(**document**).ready(**function** () {

**$**(**'#sbmtn'**).click(**function** () {

**var** from = **$**(**"#from"**).val();

**if** (from == **''**) {

alert(**"Please Enter From Date"**);

**$**(**"#from"**).**focus**();

**return false**;

}

**var** to = **$**(**"#to"**).val();

**if** (from == **''**) {

alert(**"Please Enter To Date"**);

**$**(**"#to"**).**focus**();

**return false**;

}

**var** pid = **$**(**"#pid"**).val();

**if** (pid == **'-1'**) {

alert(**"Please Select Product"**);

**$**(**"#pid"**).**focus**();

**return false**;

}

**return true**;

});

});

**$**(**function** () {

**$**(**"#from"**).datepicker({

**endDate**: **"today"**,

**changeMonth**: **true**,

**numberOfMonths**: 1,

**autoclose**: **true**,

**todayHighlight**: **true**

}).on(**'changeDate'**, **function** () {

**var** selectedFdate = **new** Date(**$**(**this**).val());

**var** endDate = **new** Date(selectedFdate.setDate(selectedFdate.getDate() + 14));

**$**(**'#to'**).datepicker(**'setStartDate'**, **new** Date(**$**(**this**).val()));

**var** endDate = **new** Date() > endDate ? endDate : **new** Date(**$**(**this**).val());

**$**(**'#to'**).datepicker(**'setEndDate'**, endDate);

});

**$**(**"#to"**).datepicker({

**endDate**: **"today"**,

**changeMonth**: **true**,

**numberOfMonths**: 1,

**autoclose**: **true**,

**todayHighlight**: **true**

}).on(**'changeDate'**, **function** () {

**$**(**'#from'**).datepicker(**'setEndDate'**, **new** Date(**$**(**this**).val()));

**var** selectedTdate = **new** Date(**$**(**this**).val());

**var** startDate = **new** Date(selectedTdate.setDate(selectedTdate.getDate() - 14));

**$**(**'#from'**).datepicker(**'setStartDate'**, startDate);

});

});

</**script**>

</**head**>

<**body**>

<**div class="mdl-layout mdl-js-layout mdl-layout--fixed-header"**>

<**header class="mdl-layout\_\_header"**>

<**div class="mdl-layout\_\_header-row"**>

*<!-- Title -->*

<**span class="mdl-layout-title mdl-layout--large-screen-only"**>

<**small class="mdl-color-text--white"**>

Select Date Range for Stats

</**small**>

</**span**>

<**div class="mdl-layout-spacer"**></**div**>

<**span class="mdl-layout-icon brand-logo pos-left"**><**img**

**src="<%=**request.getContextPath()**%>/images/home-logo.jpg" onclick="location**.**href**=**'home'"**></**span**>

<**button id="demo-menu-lower-right"**

**class="mdl-button mdl-js-button mdl-button--icon"**>

<**i class="material-icons"**>more\_vert</**i**>

</**button**>

<**ul class="mdl-menu mdl-menu--bottom-right mdl-js-menu mdl-js-ripple-effect"**

**for="demo-menu-lower-right"**>

<**li class="mdl-menu\_\_item" onclick="location**.**href**=**'home'"**>Home</**li**>

<**li class="mdl-menu\_\_item" onclick="location**.**href**=**'logout'"**>Log Out</**li**>

</**ul**>

</**div**>

*<!-- Tabs -->*

</**header**>

<**main class="mdl-layout\_\_content"**>

<**div class="auto-1000"**>

<**div class="main-container"**>

<**div class="report-container"**>

<**h2**></**h2**>

<**div class="select-form"**>

<**form action="stats" method="post"**>

<**table**>

<**tr**>

<**td style="width**:30%;**text-align**: **left**;**"**><**label**>From Date</**label**></**td**>

<**td**><**input type="text" name="from" id="from"**

**style="background**:**#fff url**(**<%=**request.getContextPath()**%>/images/cal.png**) **no-repeat right** 50%**"**

**readonly**>

</**td**>

</**tr**>

<**tr**>

<**td style="width**:30%;**text-align**: **left**;**"**><**label**>To Date</**label**></**td**>

<**td**><**input type="text" name="to" id="to"**

**style="background**:**#fff url**(**<%=**request.getContextPath()**%>/images/cal.png**) **no-repeat right** 50%**"**

**readonly**>

</**td**>

</**tr**>

<**tr**>

<**td style="width**:30%;**text-align**: **left**;**"**><**label**>Product</**label**></**td**>

<**td**>

<**select name="pid" id="pid" style="background**: **white**;**"**>

<**option value="-1"**>--Please Select--</**option**>

**<%** ResultSet rst = (ResultSet) request.getAttribute(**"rst"**);

**while** (rst.next()) {

**%>**

<**option value="<%=**rst.getLong(1)**%>"**>**<%=**rst.getString(2)**%>**

</**option**>

**<%**}**%>**

</**select**>

</**td**>

</**tr**>

<**tr**>

<**td colspan="2"**><**input type="Submit" id="sbmtn" value="Show Report"**></**td**>

</**tr**>

</**table**>

</**form**>

</**div**>

</**div**>

</**div**>

</**div**>

</**main**>

</**div**>

</**body**>

</**html**>

**package** edu.mca.servlet;

**import** com.google.gson.Gson;

**import** edu.mca.util.DbConnection;

**import** edu.mca.util.Response;

**import** javax.servlet.RequestDispatcher;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.servlet.http.HttpSession;

**import** java.io.IOException;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.text.DateFormat;

**import** java.text.SimpleDateFormat;

**import** java.util.\*;

*/\*\**

*\* Created by vivek on 13/5/17.*

*\*/*

@WebServlet(**"/stats"**)

**public class** StatsServlet **extends** HttpServlet {

**protected void** process(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

String fdate = request.getParameter(**"from"**);

String tdate = request.getParameter(**"to"**);

String pid = request.getParameter(**"pid"**);

**try** {

HttpSession session = request.getSession();

String user = (String) session.getAttribute(**"user"**);

String role = (String) session.getAttribute(**"role"**);

**if** (user != **null** && !**""**.equals(user.trim()) && !**"null"**.equalsIgnoreCase(user.trim())

&& role != **null** && !**""**.equals(role.trim()) && !**"null"**.equalsIgnoreCase(role.trim())) {

Connection cn = DbConnection.*getConnection*();

PreparedStatement preparedStatement = cn.prepareStatement(**"select name from Stores"**);

ResultSet storeRst = preparedStatement.executeQuery();

List<String> storeList = **new** ArrayList<>();

**while** (storeRst.next()) {

storeList.add(storeRst.getString(1));

}

DateFormat format = **new** SimpleDateFormat(**"MM/dd/yyyy"**);

DateFormat format1 = **new** SimpleDateFormat(**"yyyy-MM-dd"**);

DateFormat format2 = **new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**);

DateFormat format3 = **new** SimpleDateFormat(**"dd-MMM-yy"**);

Date fd1 = format.parse(fdate);

Date td1 = format.parse(tdate);

Date nextD = **null**;

**int** i = 0;

Map<String, String> dateWiseMap = **new** LinkedHashMap<>();

Map<String, Integer> cityWiseMap = **new** LinkedHashMap<>();

Gson gson = **new** Gson();

List<String> datewiseKeys = **new** LinkedList<>();

List<Object> datewiseValues = **new** LinkedList<>();

Map<String, List<Integer>> mp = **new** HashMap<>();

Map<String, List<Response>> dateWiseTableMap = **new** LinkedHashMap<>();

**do** {

Calendar c = Calendar.*getInstance*();

c.setTime(fd1);

c.add(Calendar.***DATE***, i++);

nextD = c.getTime();

PreparedStatement pst = cn.prepareStatement(**"select store,** *sum***(count),***count***(***\****) from sold where created\_on between ? and ? group by store"**);

pst.setTimestamp(1, **new** java.sql.Timestamp(format2.parse(format2.format(format2.parse(format1.format(nextD) + **" 00:00:00"**))).getTime()));

pst.setTimestamp(2, **new** java.sql.Timestamp(format2.parse(format2.format(format2.parse(format1.format(nextD) + **" 23:59:59"**))).getTime()));

ResultSet rst = pst.executeQuery();

Map<String, Integer> map = **new** LinkedHashMap<>();

List<Response> responses = **new** ArrayList<>();

**while** (rst.next()) {

map.put(rst.getString(1), rst.getInt(2));

responses.add(**new** Response(rst.getString(1), rst.getInt(2), rst.getInt(3)));

}

storeList.forEach(s -> {

**try** {

**if** (mp.containsKey(s)) {

List<Integer> lst = mp.get(s);

**if** (map.containsKey(s)) {

lst.add(map.get(s));

} **else** {

lst.add(0);

}

mp.put(s, lst);

} **else** {

List<Integer> lst = **new** LinkedList<>();

**if** (map.containsKey(s)) {

lst.add(map.get(s));

} **else** {

lst.add(0);

}

mp.put(s, lst);

}

} **catch** (Exception e) {

}

});

datewiseKeys.add(format3.format(nextD));

**if** (responses != **null** && responses.size() > 0)

dateWiseTableMap.put(format3.format(nextD), responses);

} **while** (!td1.equals(nextD));

mp.forEach((m, n) -> {

Map<String, Object> map1 = **new** LinkedHashMap<>();

map1.put(**"name"**, m);

map1.put(**"data"**, n);

datewiseValues.add(map1);

});

request.setAttribute(**"datewiseKeys"**, gson.toJson(datewiseKeys));

request.setAttribute(**"datewiseValues"**, gson.toJson(datewiseValues));

request.setAttribute(**"dateWiseTableMap"**, dateWiseTableMap);

PreparedStatement preparedStatement1 = cn.prepareStatement(**"select distinct city from sold where campaign\_fkey=?"**);

preparedStatement1.setLong(1, Long.*valueOf*(pid.trim()));

ResultSet citiesRst = preparedStatement1.executeQuery();

List<String> citywiseKeys = **new** LinkedList<>();

List<Object> citywiseValues = **new** LinkedList<>();

Map<String, List<Response>> cityWiseTableMap = **new** LinkedHashMap<>();

Map<String, List<Integer>> mp1 = **new** HashMap<>();

**while** (citiesRst.next()) {

PreparedStatement pst = cn.prepareStatement(**"select store,***sum***(count),***count***(***\****) from sold where campaign\_fkey=? and city=? and created\_on between ? and ? group by store"**);

pst.setLong(1, Long.*valueOf*(pid));

pst.setString(2, citiesRst.getString(1));

pst.setTimestamp(3, **new** java.sql.Timestamp(format2.parse(format2.format(format2.parse(format1.format(fd1) + **" 00:00:00"**))).getTime()));

pst.setTimestamp(4, **new** java.sql.Timestamp(format2.parse(format2.format(format2.parse(format1.format(td1) + **" 23:59:59"**))).getTime()));

ResultSet rst = pst.executeQuery();

Map<String, Integer> map = **new** LinkedHashMap<>();

List<Response> responses = **new** ArrayList<>();

**while** (rst.next()) {

map.put(rst.getString(1), rst.getInt(2));

responses.add(**new** Response(rst.getString(1), rst.getInt(2), rst.getInt(3)));

}

storeList.forEach(s -> {

**if** (mp1.containsKey(s)) {

List<Integer> lst = mp1.get(s);

**if** (map.containsKey(s)) {

lst.add(map.get(s));

} **else** {

lst.add(0);

}

mp1.put(s, lst);

} **else** {

List<Integer> lst = **new** LinkedList<>();

**if** (map.containsKey(s)) {

lst.add(map.get(s));

} **else** {

lst.add(0);

}

mp1.put(s, lst);

}

});

citywiseKeys.add(citiesRst.getString(1));

**if** (responses != **null** && responses.size() > 0)

cityWiseTableMap.put(citiesRst.getString(1), responses);

}

mp1.forEach((m, n) -> {

Map<String, Object> map1 = **new** LinkedHashMap<>();

map1.put(**"name"**, m);

map1.put(**"data"**, n);

citywiseValues.add(map1);

});

request.setAttribute(**"citywiseKeys"**, gson.toJson(citywiseKeys));

request.setAttribute(**"citywiseValues"**, gson.toJson(citywiseValues));

request.setAttribute(**"cityWiseTableMap"**, cityWiseTableMap);

*// Get campaign Name*

PreparedStatement pst1 = cn.prepareStatement(**"select name from campaign where id=?"**);

pst1.setLong(1, Long.*valueOf*(pid));

ResultSet rst = pst1.executeQuery();

**if** (rst.next()) request.setAttribute(**"campaignName"**, rst.getString(1));

RequestDispatcher rd = request.getRequestDispatcher(**"/WEB-INF/jsps/stats.jsp"**);

rd.forward(request, response);

} **else** {

response.sendRedirect(**"login?status=true&msg=Session+Expired+or+Invalid"**);

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

**protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

process(request, response);

}

**protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

process(request, response);

}

}

### Stats View Page

(Product Stats day and city wise)

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Product Sales Report

' Version 1.0.0

' Created by Vivek Dhakre

' Date : 27-May-2017

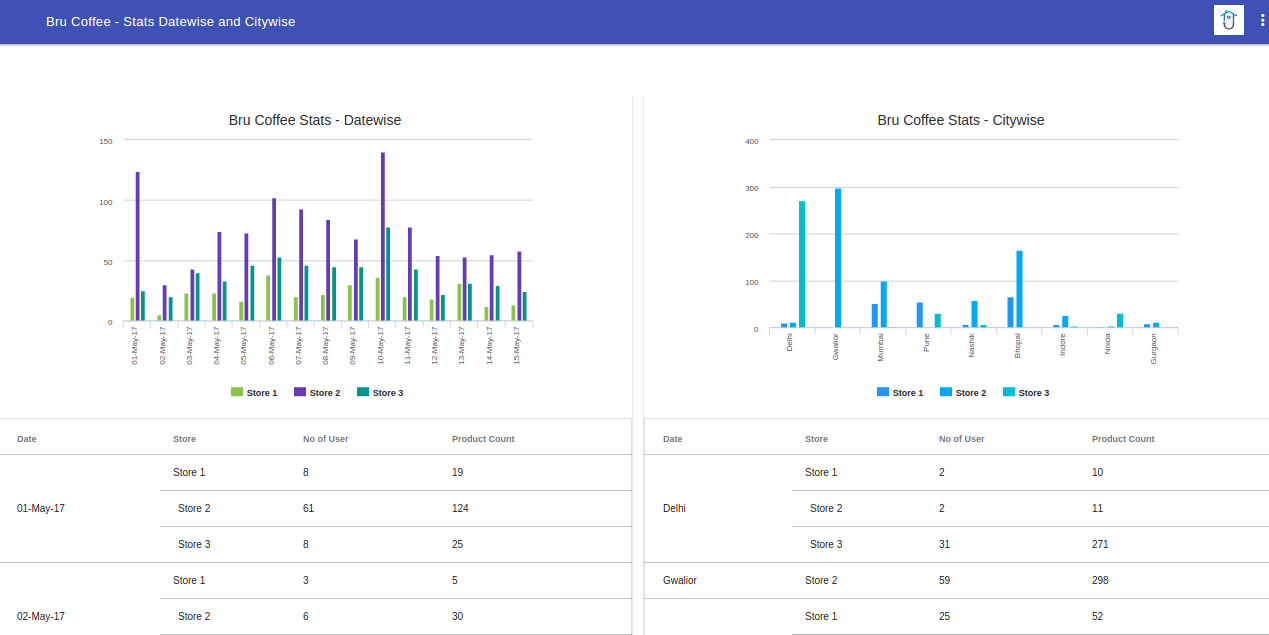
'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Main Screen

' Used Table : Stores, City, Campaign,Sold

'Module to show Statistics

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



<%@ **page import**="**edu.mca.util.Response**" %>

<%@ **page import**="**java.util.List**" %>

<%@ **page import**="**java.util.Map**" %>

<!DOCTYPE **html**>

<**html lang="en" ng-app="Dashboard"**>

<**head**>

<**meta charset="UTF-8"**>

<**meta name="viewport" content="width=device-width"**>

<**title**>Product Sales Dashboard</**title**>

<**link rel="stylesheet" href="<%=**request.getContextPath()**%>/chart/css/style.css" type="text/css"**>

<**link type="text/css" rel="stylesheet" href="<%=**request.getContextPath()**%>/chart/css/material.css"**>

<**script type="text/javascript" src="<%=**request.getContextPath()**%>/chart/js/material.js"**></**script**>

<**script src="<%=**request.getContextPath()**%>/chart/js/jquery-2.2.2.min.js"**></**script**>

</**head**>

<**body**>

*<!-- Simple header with scrollable tabs. -->*

<**div class="mdl-layout mdl-js-layout mdl-layout--fixed-header"**>

<**header class="mdl-layout\_\_header"**>

<**div class="mdl-layout\_\_header-row"**>

*<!-- Title -->*

<**span class="mdl-layout-title mdl-layout--large-screen-only"**>

<**small class="mdl-color-text--white"**>

**${**campaignName**}** - Stats Datewise and Citywise

</**small**>

</**span**>

<**div class="mdl-layout-spacer"**></**div**>

<**span class="mdl-layout-icon brand-logo pos-left"**><**img**

**src="<%=**request.getContextPath()**%>/images/home-logo.jpg" onclick="location**.**href**=**'home'"**></**span**>

<**button id="demo-menu-lower-right"**

**class="mdl-button mdl-js-button mdl-button--icon"**>

<**i class="material-icons"**>more\_vert</**i**>

</**button**>

<**ul class="mdl-menu mdl-menu--bottom-right mdl-js-menu mdl-js-ripple-effect"**

**for="demo-menu-lower-right"**>

<**li class="mdl-menu\_\_item" onclick="location**.**href**=**'logout'"**>Log Out</**li**>

</**ul**>

</**div**>

*<!-- Tabs -->*

</**header**>

*<!-- <div class="mdl-layout\_\_drawer">*

*<span class="mdl-layout-title">Title</span>*

*<nav class="mdl-navigation">*

*<a class="mdl-navigation\_\_link" href="index.html">Campaign Name</a>*

*</nav>*

*</div> -->*

<**main class="mdl-layout\_\_content"**>

<**section ng-controller="CampaignController"**>

<**div class="mdl-grid" style="margin-top**: 50**px**;**"**>

<**div class="mdl-cell mdl-cell--6-col mdl-cell--6-col-desktop mdl-cell--12-col-phone mdl-cell--4-col-tablet demo-card-square mdl-card mdl-shadow--2dp"**>

<**div class="mdl-card\_\_supporting-text"**>

*<!--<div id="TajMahalCouponGenerated" style="width: 100%; min-width: 250px; max-width: 600px; margin: 0 auto"></div>-->*

<**div id="datewise\_stats"**

**style="width**: 100%; **min-width**: 250**px**; **max-width**: 600**px**; **margin**: 0 **auto"**></**div**>

</**div**>

<**table class="mdl-data-table mdl-js-data-table" width="100%"**>

<**thead**>

<**tr**>

<**th class="mdl-data-table\_\_cell--non-numeric"**>Date</**th**>

<**th class="mdl-data-table\_\_cell--non-numeric"**>Store</**th**>

<**th class="mdl-data-table\_\_cell--non-numeric"**>No of User</**th**>

<**th class="mdl-data-table\_\_cell--non-numeric"**>Product Count</**th**>

</**tr**>

</**thead**>

<**tbody**>

**<%**

Map<String, List<Response>> dateWiseTableMap = (Map<String, List<Response>>) request.getAttribute(**"dateWiseTableMap"**);

**for** (Map.Entry<String, List<Response>> map : dateWiseTableMap.entrySet()) {

**%>**

<**tr**>

<**td class="mdl-data-table\_\_cell--non-numeric"**

**rowspan="<%=**map.getValue().size()**%>"**>**<%=**map.getKey()**%>**

</**td**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(0).getStoreName()**%>**

</**td**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(0).getCount()**%>**

</**td**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(0).getSum()**%>**

</**td**>

</**tr**>

**<%**

**for** (**int** i = 1; i < map.getValue().size(); i++) {

**%>**

<**tr**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(i).getStoreName()**%>**

</**td**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(i).getCount()**%>**

</**td**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(i).getSum()**%>**

</**td**>

</**tr**>

**<%**}**%>**

**<%**

}

**%>**

</**tbody**>

</**table**>

</**div**>

<**div class="mdl-cell mdl-cell--6-col mdl-cell--6-col-desktop mdl-cell--12-col-phone mdl-cell--4-col-tablet demo-card-square mdl-card mdl-shadow--2dp"**>

<**div class="mdl-card\_\_supporting-text"**>

<**div id="citywise\_stats"**

**style="width**: 100%; **min-width**: 250**px**; **max-width**: 600**px**; **height**: 400**px**; **margin**: 0 **auto"**></**div**>

</**div**>

<**table class="mdl-data-table mdl-js-data-table" width="100%"**>

<**thead**>

<**tr**>

<**th class="mdl-data-table\_\_cell--non-numeric"**>Date</**th**>

<**th class="mdl-data-table\_\_cell--non-numeric"**>Store</**th**>

<**th class="mdl-data-table\_\_cell--non-numeric"**>No of User</**th**>

<**th class="mdl-data-table\_\_cell--non-numeric"**>Product Count</**th**>

</**tr**>

</**thead**>

<**tbody**>

**<%**

Map<String, List<Response>> cityWiseTableMap = (Map<String, List<Response>>) request.getAttribute(**"cityWiseTableMap"**);

**for** (Map.Entry<String, List<Response>> map : cityWiseTableMap.entrySet()) {

**%>**

<**tr**>

<**td class="mdl-data-table\_\_cell--non-numeric"**

**rowspan="<%=**map.getValue().size()**%>"**>**<%=**map.getKey()**%>**

</**td**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(0).getStoreName()**%>**

</**td**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(0).getCount()**%>**

</**td**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(0).getSum()**%>**

</**td**>

</**tr**>

**<%**

**for** (**int** i = 1; i < map.getValue().size(); i++) {

**%>**

<**tr**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(i).getStoreName()**%>**

</**td**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(i).getCount()**%>**

</**td**>

<**td class="mdl-data-table\_\_cell--non-numeric"**>**<%=**map.getValue().get(i).getSum()**%>**

</**td**>

</**tr**>

**<%**}**%>**

**<%**

}

**%>**

</**tbody**>

</**table**>

</**div**>

</**div**>

</**section**>

</**main**>

</**div**>

<**script src="<%=**request.getContextPath()**%>/chart/js/highcharts.js"**></**script**>

<**script src="<%=**request.getContextPath()**%>/chart/js/modules/data.js"**></**script**>

<**script src="<%=**request.getContextPath()**%>/chart/js/modules/drilldown.js"**></**script**>

<**script type="text/javascript"**>

**$**(**function** () {

**var** datewiseKeys = **<%=**request.getAttribute(**"datewiseKeys"**)**%>**;

**var** datewiseValues = **<%=**request.getAttribute(**"datewiseValues"**)**%>**;

**$**(**'#datewise\_stats'**).highcharts({

**chart**: {

**type**: **'column'**

},

**title**: {

**text**: **'${**campaignName**}** **Stats - Datewise'**

},

**credits**: {

**enabled**: **false**,

**text**: **''**,

**href**: **''**

},

**colors**: [**'#8BC34A'**, **'#673AB7'**, **'#009688'**, **'#FF1744'**, **'#3F51B5'**, **'#2196F3'**, **'#03A9F4'**, **'#00BCD4'**, **'#009688'**, **'#4CAF50'**, **'#8BC34A'**, **'#CDDC39'**, **'#FFEB3B'**, **'#FFC107'**, **'#FF9800'**, **'#FF5722'**, **'#795548'**, **'#607D8B'**],

**xAxis**: {

**categories**: datewiseKeys,

**labels**: {

**rotation**: -90

},

**crosshair**: **true**

},

**yAxis**: {

**min**: 0,

**title**: {

**text**: **''**

}

},

**tooltip**: {

**headerFormat**: **'<span style="font-size:10px">{point.key}</span><table>'**,

**pointFormat**: **'<tr><td style="color:{series.color};padding:0">{series.name}: </td>'** +

**'<td style="padding:0"><b>{point.y}</b></td></tr>'**,

**footerFormat**: **'</table>'**,

**shared**: **true**,

**useHTML**: **true**

},

**plotOptions**: {

**column**: {

**pointPadding**: 0.2,

**borderWidth**: 0

}

},

**series**: datewiseValues

});

});

**$**(**function** () {

**var** citywiseKeys = **<%=**request.getAttribute(**"citywiseKeys"**)**%>**;

**var** citywiseValues = **<%=**request.getAttribute(**"citywiseValues"**)**%>**;

**$**(**'#citywise\_stats'**).highcharts({

**chart**: {

**type**: **'column'**

},

**title**: {

**text**: **'${**campaignName**}** **Stats - Citywise'**

},

**credits**: {

**enabled**: **false**,

**text**: **''**,

**href**: **''**

},

**colors**: [**'#2196F3'**, **'#03A9F4'**, **'#00BCD4'**, **'#009688'**, **'#4CAF50'**, **'#8BC34A'**, **'#CDDC39'**, **'#FFEB3B'**, **'#FFC107'**, **'#FF9800'**, **'#FF5722'**, **'#795548'**, **'#607D8B'**],

**xAxis**: {

**categories**: citywiseKeys,

**labels**: {

**rotation**: -90

},

**crosshair**: **true**

},

**yAxis**: {

**min**: 0,

**title**: {

**text**: **''**

}

},

**tooltip**: {

**headerFormat**: **'<span style="font-size:10px">{point.key}</span><table>'**,

**pointFormat**: **'<tr><td style="color:{series.color};padding:0">{series.name}: </td>'** +

**'<td style="padding:0"><b>{point.y}</b></td></tr>'**,

**footerFormat**: **'</table>'**,

**shared**: **true**,

**useHTML**: **true**

},

**plotOptions**: {

**column**: {

**pointPadding**: 0.2,

**borderWidth**: 0

}

},

**series**: citywiseValues

});

});

</**script**>

</**body**>

</**html**>

# Product Sales Dashboard Test Plan

## Introduction

This document describes the user acceptance test plan for the Product Sales Dashboard. The complete test strategy for the Product Sales Dashboard is to perform the following kinds of tests, in sequence:

1. **Component testing** of each component that makes up the Statistics of various Product
2. **Integration testing** of the Product Sales Dashboard, to ensure the correct working of its components
3. **Validation testing** of the Product Sales Dashboard, to ensure that it works correctly in a pseudo-live environment
4. **User acceptance testing** of the Product Sales Dashboard, to ensure that its function is acceptable to its Sales or Marketing representative.

Acceptance testing is the last set of tests to be performed before the application goes officially live.

## Test Scope

The scope of the user acceptance testing covers:

* Version 1 of the Product Sales Dashboard
* Sale or Marketing representative functionality defined by a set of use cases
* Sales or Marketing representative-facing functionality defined by a set of use cases

The aim of the testing is to determine how well the application meets its functional requirements from the perspective of the Sales or Marketing representative, and to identify any issues so they can be resolved. Also, the testing serves to compile a set of test data and results that can be used during subsequent test cycles, to test for non-regression of the software in later releases or after the application is in maintenance.

Working practices might vary from user to user and are considered outside the scope of the testing.

## Test Strategy

The basis of user acceptance testing is that other tests were completed successfully, so the application and its required infrastructure are considered to be stable and reliable. Acceptance testing concentrates on the application from the Sales or Marketing perspective, that is, how the application is used and whether it meets the necessary quality criteria.

Change requests will be sent to the development team as the actionable documentation. Change criteria will be determined by the Test team and the Development team prior to the beginning of testing. For instance, criteria may include *impact to desired functionality, amount of code impacted by proposed change,* and *design required by proposed change*. The tester will evaluate the criteria. The test lead will determine Change Required or not. Once a bug has been determined as Change Required, the bug report will be translated into a Change Request and passed on to development.

The customer of the acceptance testing is the System Users, Supervisor and Adminstrator for Product Sales Dashboard. The progress of the acceptance testing will be reported to the customer, together with any issues that are discovered and their planned resolutions. Sign-off of the tests, and therefore the acceptance of the application, will be performed by the customer or a selected representative.

## Preconditions

The following items are required before testing can take place:

* A complete and coherent functional specification of the Product Sales Dashboard as use cases and usage scenarios
* A complete and validation-tested release of Product Sales Dashboard, delivered according to the delivery plan
* An agreed-upon procedure for dealing with any anomalies that are discovered during the testing process
* A set of test specifications describing how each functional area of the Product Sales Dashboard is to be acceptance tested
* An implemented test environment for the testing
* Sufficient, suitable resources to carry out the testing
* Available standards for the acceptance testing

## Test Priorities

During testing of the Product Sales Dashboard, the following qualities will be tested in order of priority:

* Functionality—whether the required functions are available and working as expected
* Usability—how user-friendly and intuitive the Product Sales Dashboard is
* Security—how well-protected and guaranteed corporate and user data is
* Performance—whether the response times are within acceptable limits
* Customization—how straightforward it is to use the application in new, unpredicted ways

## Test Techniques

The following techniques will be applied:

* Scripted tests—sequences of user interactions (based on the use case and usage scenarios) using predefined data sets against predicted results
* Unscripted tests—based on scripted tests, the tester tries to modify the scenarios to explore what-if possibilities
* Penetration tests—scripted tests to attempt unauthorized entry into the system
* Usability checklists—tests to determine the complexity of interactions
* Performance statistics—generation of performance information to check against desired performance criteria

## Test Organization

## Roles and Responsibilities

The following roles are defined:

* QA lead/test manager—responsible for planning and ensuring the smooth running of the test process
* Tester—carries out the tests according to the test plan, and then reports the results
* Product manager—ensures that the tests are carried out successfully from a user perspective
* Project sponsor/client—acts as main stakeholder, and ensures that the needs of the customer community as a whole are considered
* Test support—provides technical assistance, such as test environment configuration, and non-technical assistance, such as methodological support

Weekly team meetings will be held involving the test manager, testers, and product managers. At these meetings, the progress of the testing process will be reported, any issues will be discussed, and actions will be agreed upon.

# Conclusion and future enhancement

This project was developed to fulfill Sales or Marketing representative requirement; however there are lots of scope to improve the performance of the Product Sales Report in the area of user interface, database performance, and query processing time. Etc.

So there are many things for future enhancement of this project. The future enhancements that are possible in the project are as follows.

* Linking and integration of any legacy system for accounting.
* Integration with travel agent through Web Services
* Connection to third-party OLAP applications
* Electronic Data Interchange (EDI) system between banks, other credit verification agency and their vendors
* In the area of data security and system security.
* Provide more online tips and help.
* To optimize the query which is embedded in the system.