**CMS Manhattan API and Design document**

DRAFT.

1. **Project description**

CMS Manhattan is sales application engine with Marketplaces functionality. J2EE Standards

1. **System Design for Amazon Cloud.**

A diagram of a computer

Description automatically generated

1. **Application Design.**

**3.1 Buy Flow Components**

BuyFlow Component diagram

The MVC diagram describes main components for buy flow process of application.

A screenshot of a computer screen

Description automatically generated

* 1. **Publish item Components.**

Publish Items Component diagram

The MVC diagram describes main components for publish data process of application.

**A screenshot of a computer screen

Description automatically generated**

1. **CMS MVC Framework**

There are 4 parts that represent CMS MVC Framework

**MVC - VEW** : implemented via JSP and XSLT .

So JSP provides data to display in XML format then XSLT can transform to HTML or JSON or ANOTHER format.

So assign VIEW developer must use **@PageController** **annotation**

**@PageController( jspName = "AccountHistory.jsp" )**

**public class AccountHistoryAction extends TemplateAction {**

**}**

And VIEW transformation via **@PageXsltView for model class.**

**@PageXsltView( jspName = "AccountHistory.jsp" , xsltName="accounthistory.xsl" , responseType= Type.XML )**

**public class AccountHistoryBean extends com.cbsinc.cms.WebControls implements java.io.Serializable {**

**}**

Then Maven build script will add the MVC component in configuration file .

**MVC – CONTROLLER**: implemented via java class.

So since developer assigned @PageController **annotation then**

Maven builds script will add the MVC component in configuration file.

**@PageController( jspName = "AccountHistory.jsp" )**

**public class AccountHistoryAction extends TemplateAction {**

**}**

**MVC-MODEL:** implemented via java class.

So since developer assigned @PageModel **annotation then**

Maven builds script will add the MVC component in configuration file.

**@PageXsltView( jspName = "AccountHistory.jsp" , xsltName="accounthistory.xsl" , responseType= Type.XML )**

**@PageModel( Id="accountHistoryBeanId" , scope=Scope.SESSION )**

**public class AccountHistoryBean extends com.cbsinc.cms.WebControls implements java.io.Serializable {**

**}**

1. **CMS MVC API**

CATALOG COMPOMENT – Product list

ProductlistAction . java is Controller

Get requests API

Default request URL - Productlist.jsp?**catalog**\_id=-2

Scroll result set cursor for request URL - Productlist.jsp?catalog\_id=-2& **offset**=10

Use “**offset**” for every request for every page .

Open a catalog by catalog\_id request URL - Productlist.jsp?catalog\_id=-101

Search via search key words by catalog\_id request URL - Productlist.jsp?**creteria1**\_id=211&**creteria2**\_id=7895&creteria3\_id=235&creteria4\_id=775&creteria5\_id=115&creteria6\_id=577&creteria1\_i7=25

Scroll result set cursor for Productlist.jsp?**creteria1**\_id=211&**creteria2**\_id=7895&creteria3\_id=235&creteria4\_id=775&creteria5\_id=115&creteria6\_id=577&creteria1\_i7=25& **offset**=10

Setup currency for items from catalog request URL - Productlist.jsp? currency\_cd=USD

Search via between dates :

Productlist.jsp?dayfrom\_id=1

Productlist.jsp?mountfrom\_id=1

Productlist.jsp?yearfrom\_id=2006

Productlist.jsp?dayto\_id=1

Productlist.jsp?mountto\_id=1

Productlist.jsp?yearto\_id=2006

POST requests

So **offset** avoid pulling big data set to avoid fill all memory

Search by text request .

<form name="searchform" action="Productlist.jsp" method="POST">

<input name="search\_value" type="text" size="20 value="">

<input name="search\_char" value=”” type="hidden">

<input name="**searchquery**" type="hidden" value="**1**" >

<input type="HIDDEN" name="**offset**" value="0">

</form>

Search via search key words

<form name="searchform" action="Productlist.jsp" method="POST">

<input name="search\_value" type="text" size="20 value="">

<input name="search\_char" value=”” type="hidden">

<input name="**searchquery**" type="hidden" value="**3**" >

<input type="HIDDEN" name="**offset**" value="0">

</form>

Search via char

<form name="searchform" action="Productlist.jsp" method="POST">

<input name="search\_value" type="text" size="20 value="">

<input name="search\_char" value=”” type="hidden">

<input name="**searchquery**" type="hidden" value="**2**" >

<input type="HIDDEN" name="**offset**" value="0">

</form>

Catalog Response in XML Format .



PRODUCT INFORMATION – Product full description

ProductInfoAction.java – Controller

Get requests API

Default request URL - ProductInfo.jsp?policy\_byproductid=982

ADD ITEM TO CART

<form name="order" action="Order.jsp" method="POST>

<input autocomplete="off" type="HIDDEN" name="position" value="23316050137317597">

<input autocomplete="off" type="HIDDEN" name="action" value="add">

<input autocomplete="off" type="number" size="6" min="1" max="90000" name="quantity" value="1>

<input class="button" autocomplete="off" type="Submit" name="Submit" alt="Add to cart" value="Add to cart>

</form>

Product information - Response in XML Format.



ORDER – Shopping cart



ORDER LIST – Purchase History



AUTHORIZATION - Registration and Authorization page

<form action="Authorization.jsp" method="post" id="form-login" class="form-login">

<input autocomplete="off" type="TEXT" name="Login" value="user">

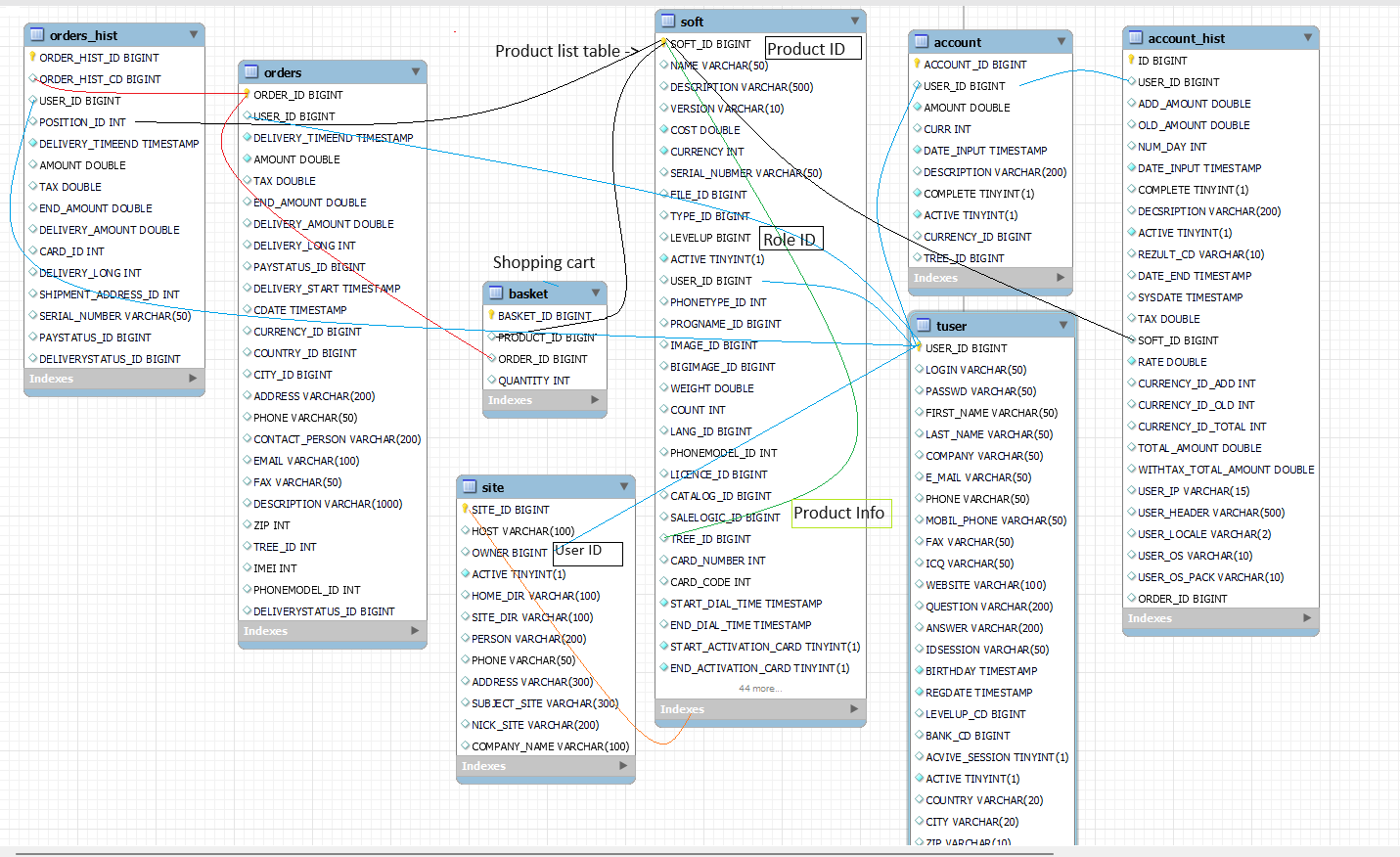
<input autocomplete="off" type="PASSWORD" name="Passwd1">

<input class="button" type="submit" value="Enter" name="Submit"></div>

</form>



1. **Buyflow Database schema**



Notes:

Table named as soft is product table.

soft.portlettype\_id – It is Product info description type on product.

soft.portlettype\_id = 1 It is column 1 in product info

soft.portlettype\_id = 2 It is column 2 in product info

soft.portlettype\_id = 3 It is review message for product

soft.portlettype\_id = 4 It is description in product tab block

soft.portlettype\_id = 5 It is attached file

soft.tree\_id . it is product Id as reference to find other related for product info which above.

soft.siteId = User Site ID or User Wallet ID . it is the same idea.

soft.type\_id It is product visibility area So User site or Market place as global post .

1. **Product content Database schema ( CMS )**

So, the data model show how product have connections to images and files and catalog to display in used interface (GUI).

When do search you can find product via criteria ( search key words or product attributes ) which nested as tree structure or DOM structure .

So those search key words, or product attributes could be used as suggestion for search by text for product description fields since it is part of product content.

Next a search for key words or product attributes can be used as linked drop down list as filter for product list via product attributes since those product attributes have linked tree or DOM structure .

Any product must be displayed or found by soft\_id filed for product info page.

When you display search you must display small image but for product info big one.

