

Cognizant Academy

truYum

## Spring Core Specification Document

Version 1.0

	Prepared By / Last Updated By	Reviewed By	Approved By
Name	Chandrasekaran Janardhanan	Vimalathithan Krishnan	Ramadevanahalli Lingachar, Shashidhara Murthy
Role	Learning Solution Designer	Learning Solution Architect	Learning Solution Lead
Signature			
Date			

# Table of Contents

<b>1.0</b>	<b>Introduction</b>	<b>4</b>
1.1	Purpose of this document	4
1.2	Definitions & Acronyms	4
1.3	Project Overview	4
1.4	Scope	4
1.5	Intended Audience	4
1.6	Hardware and Software Requirement	4
1.7	Eclipse Configuration for Tomcat Server	Error! Bookmark not defined.
1.7.1	Add Tomcat Server to Eclipse	Error! Bookmark not defined.
1.7.2	Add Tomcat library to eKart project	Error! Bookmark not defined.
<b>2.0</b>	<b>Class Diagram</b>	Error! Bookmark not defined.
2.1	Servlets	Error! Bookmark not defined.
<b>3.0</b>	<b>Design for View Menu Item List Admin (TYUC001)</b>	<b>7</b>
3.1	Sequence Diagram	Error! Bookmark not defined.
3.2	Servlet	Error! Bookmark not defined.
3.3	JSP	7
<b>4.0</b>	<b>Design for View Menu Item List Customer (TYUC002)</b>	<b>9</b>
4.1	Sequence Diagram	Error! Bookmark not defined.
4.2	Servlet	9
4.3	JSP	Error! Bookmark not defined.
<b>5.0</b>	<b>Design for Edit Menu Item (TYUC003)</b>	<b>10</b>
5.1	Sequence Diagram (Show Edit Menu Item Form)	10
5.2	Sequence Diagram (Edit Menu Item)	11
5.3	Servlet	Error! Bookmark not defined.
5.4	JSP (edit-menu-item.jsp)	Error! Bookmark not defined.
5.1	JSP (edit-menu-item-status.jsp)	13
<b>6.0</b>	<b>Design for Add Cart (TYUC004)</b>	<b>13</b>
6.1	Sequence Diagram	13
6.1	Servlet	Error! Bookmark not defined.
6.1	JSP (menu-item-list-customer.jsp)	Error! Bookmark not defined.
<b>7.0</b>	<b>Design for View Cart (TYUC005)</b>	<b>14</b>
7.1	Sequence Diagram	14
7.2	Servlet	Error! Bookmark not defined.
7.3	JSP (cart.jsp)	Error! Bookmark not defined.

7.1 JSP (cart-empty.jsp)	Error! Bookmark not defined.
<b>8.0 Design for Remove Item from Cart (TYUC006)</b>	<b>16</b>
8.1 Sequence Diagram	16
8.2 Servlet	Error! Bookmark not defined.
8.3 JSP (cart.jsp)	16
<b>9.0 Standards and Guidelines</b>	<b>17</b>
9.1 JSP	17
9.2 Servlets	Error! Bookmark not defined.
<b>10.0 Submission</b>	Error! Bookmark not defined.
10.1 Code submission instructions	Error! Bookmark not defined.
<b>11.0 Change Log</b>	<b>17</b>

# 1.0 Introduction

## 1.1 Purpose of this document

The purpose of this document is to define the server side implementation of the truYum application.

## 1.2 Definitions & Acronyms

Definition / Acronym	Description
JSP	Java Server Pages
JSTL	Java Standard Tag Library

## 1.3 Project Overview

Refer truYum-use-case-specification.docx for understanding the functionality and features.

## 1.4 Scope

1. Creation of Spring Core controller for truYum application

## 1.5 Intended Audience

- Product Owner
- Scrum Master
- Application Architect
- Project Manager
- Test Manager
- Development Team
- Testing Team

## 1.6 Hardware and Software Requirement

1. Hardware Requirement:
  - a. Developer PC with 4GB Ram
2. Software Requirement

- a. Git
- b. JDK 1.8
- c. Eclipse IDE for Enterprise Java Developers 2019-03 R
- d. Apache Tomcat 9

## 1.7 Create Spring Boot MVC project in Eclipse

### 1.7.1 Generate project using Spring Initializr

1. Go to <https://start.spring.io>
2. Select Project as Maven Project
3. Select Language as Java
4. Select the relevant latest Spring Boot version
5. Key in Group as “com.cognizant”
6. Key in artifact as “truyum”
7. Select following dependencies
  - a. Spring Web
  - b. Spring Boot DevTools
  - c. MySQL Driver
8. Click Generate and download the zip.

### 1.7.2 Configure generated project in Eclipse

1. Extract the truyum folder from the zip and paste it to D drive.
2. Rename the truyum folder as truyum-spring-mvc
3. Move the truyum-spring-mvc folder to Eclipse workspace folder
4. Open truyum-spring-mvc folder in command prompt and execute the maven command to download all the dependent jars required for the project:

```
mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -  
Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -  
Dhttp.proxyUser=125546
```

5. Open Eclipse
6. Follow steps below to import the maven project.

7. File > Import .. > Maven > Existing Maven Projects > Click Browse and select the truyum-spring-mvc folder > Click Finish
8. In Eclipse, right click on the “truyum-spring-mvc” project select “Maven” > “Update Project ..”
9. Copy the following files from truYum project:
  - a. CartDaoSqlImpl.java
  - b. CartDaoSqlImplTest.java
  - c. MenuItemDaoSqlImpl.java
  - d. MenuItemDaoSqlImplTest.java
  - e. Copy connection.properties to 'src/main/resources' folder
  - f. Move spring-config.xml to 'src/main/resources' folder
  - g. Copy

## 2.0 Design for View Menu Item List Admin (TYUC001)

### 2.1 MenuItemController.java

1. Create this new class in package: com.cognizant.truyum.controller
2. Create instance variable for MenuItemService and auto wire the same.
3. Include LOGGER and static variable. Refer code below

```
private static final Logger LOGGER = LoggerFactory.getLogger(MenuItemController.class);  
Imports for the above statement.
```

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;
```

4. Include a method for handling the request to display the menu item list for admin. Refer details in table below:

**public String showMenuItemListAdmin(ModelMap model) throws SystemException**


- a. Annotate the method with @GetMapping with the URL as “/show-menu-list-admin”
- b. Invoke **LOGGER.info(“Start”)** in the first line of the method to log the method start.
- c. Invoke **getMenuItemListAdmin()** from **menuItemService** and add it to **model**.
- d. Invoke **LOGGER.end(“End”)** to indicate the method completion in the log.
- e. Return the view “menu-item-list-admin”

### 2.2 menu-item-list-admin.jsp

1. Copy the HTML create for this screen and convert it into jsp. Refer details below

HTML	Renamed JSP	Description
WebContent/menu-item-list-admin.html	src/main/webapp/WEB-INF/view/menu-item-list-admin.jsp	List the menu items for admin

2. Make the content dynamic based on the details provided in the screen layout.
3. Copy necessary CSS files and images from the HTML implementation and include them under the /view folder.

truYum


Menu

menulist

Menu Items

Name

Sandwich

Burger

Pizza

French Fries

Chocolate Brownie

menulist.title

Price

Rs. 99.00

Rs. 129.00

Rs. 149.00

Rs. 57.00

Rs. 32.00

menulist.price

Active

Yes

Yes

Yes

No

Yes

menulist.active

Date of Launch

15/03/2017

23/12/2017

21/08/2018

02/07/2017

02/11/2022

menulist.dateOfLaunch

Category

Main Course

Main Course

Main Course

Starters

Dessert

menulist.category

Free Delivery

Yes

No

No

Yes

Yes

menulist.freeDelivery

Action

Edit

Edit

Edit

Edit

Edit

Use JSTL formatting tags

/show-menu-list-admin?menuItem={menuItem}

/show-menu-list-admin

Copyright © 2019

## 2.3 MenuItemDaoCollectionImpl.java

To make the truYum application work end to end, we will use the SQL implementation of Dao, so that data is persisted in the database. Remove the name in @Component annotation. This will be implemented in MenuItemDaoSqlImpl.

## 2.4 MenuItemDaoSqlImpl.java

To make the truYum application work end to end, we will use the SQL implementation of Dao. Incorporate following changes to get this done:

1. Include @Component with name as "menuItemDao". This ensures that SQL implementation is loaded instead of the Collection implementation.
2. Incorporate LOGGER and include Start and End log for each method in this class.

After all the above changes try accessing the web page using the URL <http://localhost:8080/show-menu-item-list-admin>.



## 3.0 Design for View Menu Item List Customer (TYUC002)

### 3.1 MenuItemController.java

1. Include a method for showing the menu item list for customers.

**public String showMenuItemListCustomer(ModelMap model) throws SystemException**

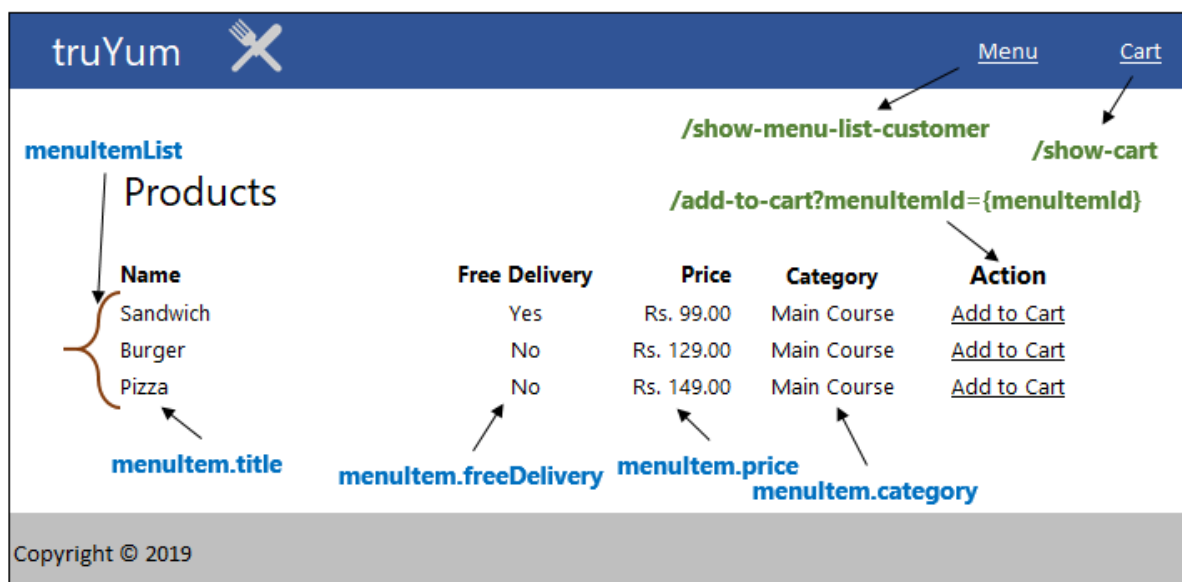
- a. Annotate the method with @GetMapping with the URL as "/show-menu-list-customer"
- b. Invoke LOGGER.info("Start") in the first line of the method to log the method start.
- c. Invoke getMenuItemListCustomer() from menuItemService and add it to model.
- d. Invoke LOGGER.end("End") to indicate the method completion in the log.
- e. Return the view "menu-item-list-customer"

### 3.2 menu-item-list-customer.jsp

1. Copy the HTML create for this screen and convert it into jsp. Refer details below

HTML	Renamed JSP	Description
WebContent/menu-item-list-customer.html	src/main/webapp/WEB-INF/view/menu-item-list-customer.jsp	List the menu items for customer

2. Make the content dynamic based on the details provided in the screen layout.



## 4.0 Design for Edit Menu Item (TYUC003)

This use case is addressed in two steps.

1. User clicks on Edit link in Menu Item List screen of admin. This retrieves the specific Menu Item details and show the Edit Menu Item form with values populated in the form fields based on the values derived from the database. Let us call this scenario as Show Edit Menu Item Form.
2. User updates the values in the form and clicks Save button. This saves the values entered in the form for the respective menu item in the database. Let us call this scenario as Edit Menu Item.

### 4.1 Show Edit Menu Item Form

#### 4.1.1 MenuItemController.java

Include a method for showing the menu item list for customers.

**public String showEditMenuItem()**

- a. Annotate the method with `@GetMapping` with the URL as `"/show-edit-menu-item"`
- b. Invoke `LOGGER.info("Start")` in the first line of the method to log the method start.
- c. Get the `menuItemId` from the URL using `@RequestParam`
- d. Invoke `getMenuItem()` passing the `menuItemId` obtained from the URL.
- e. Add the attribute `"menuItem"` with the object reference obtained from `getMenuItem()` method


- f. Invoke `LOGGER.end("End")` to indicate the method completion in the log.
- g. Return the view "edit-menu-item"

### 4.1.2 edit-menu-item.jsp

1. Copy the HTML create for this screen and convert it into jsp. Refer details below

HTML	Renamed JSP	Description
WebContent/edit-menu-item.html	src/main/webapp/WEB-INF/view/edit-menu-item.jsp	Shows the edit menu item form for the admin to edit.

2. Make the content dynamic based on the details provided in the screen layout.

**truYum**  [Menu](#)

**Edit Menu Item**

**Name**  *product.title*

**Price (Rs.)**  *menuItem.price*

**Active** ☒ Yes ☐ No

**Date of Launch**  *menuItem.dateOfLaunch*

☒ **Free Delivery** *menuItem.freeDelivery*

**Category**  *menuItem.category*

**Save**

**Use JSTL formatting tags**

**Call JavaScript Function validateMenuItemForm()**

Copyright © 2019

## 4.2 Edit Menu Item

### 4.2.1 MenuItem.java

Include validation for each attribute along with validation error message.

## 4.2.2 MenuItemController.java

Include a method for saving the menu item. The post method submission of the edit menu form will reach this method.

**public String editMenuItem(@Valid MenuItem menuItem, BindingResult bindingResult)**

- a. Annotate the method with @PostMapping with the URL as "/edit-menu-item"
- b. Invoke LOGGER.info("Start") in the first line of the method to log the method start.
- c. Validation failures must load the edit-menu-item view.
- d. Invoke modifyMenuItem() passing the menuItem obtained from the parameter.
- e. Invoke LOGGER.end("End") to indicate the method completion in the log.
- f. Return the view "edit-menu-item-status"

## 4.2.3 edit-menu-item.jsp

1. Refer screen layout for displaying error messages.

The screenshot shows the 'Edit Menu Item' form in the truYum application. The form is titled 'Edit Menu Item' and is located under the 'Menu' tab. It contains the following fields and elements:

- Name:** A text input field containing 'Sandwich'. Below it, there are two error messages: 'Title is required' and 'Title should have 2 to 65 characters'.
- Price (Rs.):** A text input field containing '97'. Below it, there are two error messages: 'Price is required' and 'Price has to be a number'.
- Active:** A radio button group with 'Yes' selected and 'No' unselected.
- Date of Launch:** A text input field containing '27/04/2022'. Below it, there is an error message: 'Launch Date required'.
- Category:** A dropdown menu with 'Main Course' selected. The dropdown list shows 'Starters', 'Main Course', 'Dessert', and 'Drinks'.
- Free Delivery:** A checkbox that is checked.
- Save:** A blue button labeled 'Save'.

At the bottom of the form, there is a footer that reads 'Copyright © 2019'.

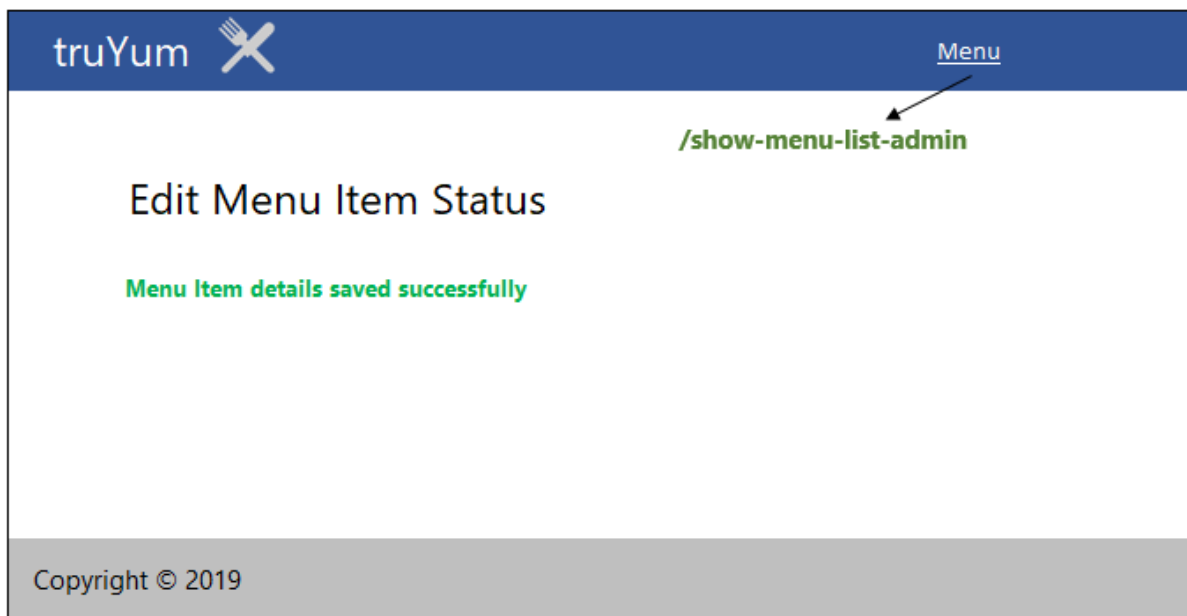
### Category Dropdown Handling

1. Set the value from menuItem.category as the first item in the drop down.

- Let the other items be added directly in the JSP with hard coded values.

#### 4.2.4 edit-menu-item-status.jsp

HTML	Renamed JSP	Description
WebContent/edit-menu-item-status.html	src/main/webapp/WEB-INF/view/edit-menu-item-status.jsp	Display the status of modifying menu item.



## 5.0 Design for Add Cart (TYUC004)

### 5.1 CartController.java

Implement this class similar to that of MenuItemController.java.

Include CartService as instance variable and auto wire the same.

Incorporate logging.

Include method addToCart() with following aspects:

- URL '/add-to-cart'
- Get Mapping with menuItemId as parameter in the URL
- Invoke respective addToCart service method to add the item into the cart of user with id 1.
- Add attribute "addCartStatus" to modelMap with boolean value true, so that necessary message can be included in the JSP to denote that item added to cart successfully.


## 5.2 CartDaoSqlImpl.java

Change auto wiring of dao in service to this class implementation.

Incorporate logging to each method

## 5.3 menu-item-list-customer.jsp

Include message using <c:if> based on the addCartStatus value.

truYum 		<a href="#">Menu</a> <a href="#">Cart</a>	
Menu Items		<b>Display this message when addCartStatus is true</b> ↓ <b>Item added to Cart Successfully</b>	
Name	Free Delivery	Price	Action
Sandwich	Yes	Rs. 99.00	<a href="#">Add to Cart</a>
Burger	No	Rs. 129.00	<a href="#">Add to Cart</a>
Pizza	No	Rs. 149.00	<a href="#">Add to Cart</a>
Copyright © 2019			

# 6.0 Design for View Cart (TYUC005)

## 6.1 CartController.java

1. URL: /show-cart
2. Mapping: GET
3. Method: showCart()
4. Parameter: userId
5. Use getAllCartItems() method to get all cart items passing the userId.
6. Add "cart" into modelMap with Cart object instance returned by getAllCartItems() method and show the view 'cart'
7. If getAllCartItems() throws CartEmptyException show the view to cart-empty in the catch block of this exception

## 6.2 cart.jsp

truYum

[Menu](#)[Cart](#)

cart.menuitemList var="menuitem"

Cart

Name

Sandwich

Burger

Chocolate Brownie

menuitem.title

Free Delivery

Yes

No

No

Total

menuitem.freeDelivery

Rs. 99.00

Rs. 129.00

Rs. 32.00

Rs. 260.00

Price

Delete

Delete

Delete

menuitem.price

/remove-cart-item?menuitemId={product.id}

/show-menu-list-customer

/show-cart

Copyright © 2019

## 6.1 cart-empty.jsp

truYum

[Menu](#)[Cart](#)

Cart

No items in cart. Use 'Add to Cart' option in [Menu Item List](#).

/show-menu-list-customer

/show-cart

Copyright © 2019

## 7.0 Design for Remove Item from Cart (TYUC006)

### 7.1 CartController.java

8. URL: /remove-cart
9. Mapping: GET
10. Method: removeCart()
11. Parameters: menuItemId and userId
12. Use removeCart() passing the menuItemId and userId
13. Use getAllCartItems() method to get all cart items passing the userId.
14. If getAllCartItems() throws CartEmptyException show the view to cart-empty in the catch block of this exception
15. Add "removeCartItemStatus" into modelMap with value "true" and show the view 'cart' if CartEmptyException is now thrown
16. Incorporate logging

### 7.1 cart.jsp

truYum

Menu Cart

**Cart**

Display this message when removeCartItemStatus is true

Item removed from Cart successfully

Name	Free Delivery	Price	
Sandwich	Yes	Rs. 99.00	<a href="#">Delete</a>
Burger	No	Rs. 129.00	<a href="#">Delete</a>
<b>Total</b>		<b>Rs. 228.00</b>	

Copyright © 2019

For other field values refer menu-item-list-customer.jsp specification in View Menu Item List Customer.



## 8.0 Standards and Guidelines

### 8.1 JSP

1. Dynamic content should be always generated using JSTL
2. Usage of Scriptlets should be completely avoided
3. Indentation should be inline with the standards defined for HTML
4. Use formatting JSTL tags for displaying currency and dates

### 8.2 Controllers, Service and Dao

1. All coding standards applicable for Java are applicable here
2. Implement logging with each method having Start and End logs at info level.
3. Log all method parameters, values returned by a method at debug level. This will be very useful when debugging error.

## 9.0 Change Log

	Changes Made			
V1.0.0	Initial baseline created on <dd-Mon-yy> by <Name of Author>			
Vx.y.z	<Please refer the configuration control tool / change item status form if the details of changes are maintained separately. If not, the template given below needs to be followed>			
	Section No.	Changed By	Effective Date	Changes Effected