Cognizant Academy

truYum

FSE Angular 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

1.0 Introduction

1.1 Purpose of this document

The purpose of this document is to define the user interface specification for truYum project.

1.2 Definitions & Acronyms

Definition / Acronym	**Description**
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet
RWD	Responsive Web Design
UX Design	User Experience Design

1.3 Project Overview

Refer truyum-user-stores.xlsx for understanding the functionality and features.

1.4 Scope

Develop truYum application using Angular framework

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 Desktop PC with 8GB RAM

2. Software Requirement
    a. Notepad ++ for editing HTML and CSS
    b. Chrome Browser

2.0 Forking and Cloning

1. **Fork project:** https://code.cognizant.com/genc-fse-java/truyum
2. Change to “webapp” directory
3. Run npm install

3.0 TYUS003, TYUC001, TYUC002 & TYUC007 – Active Menu Item List

Create New FoodItemComponent (app-food-item-info)

1. Create food related components inside “food” folder
2. Create a component under “food\item-info” for showing the food menu item
3. Create a food item interface with appropriate types
4. Render the food item information on template from the food item object (hard code data to be displayed)

Create New FoodMenuComponent (app-food-menu)

1. Create a component under “food\menu” for showing list of food item
2. Hardcode the sample list of food items in the component
3. Render the food items in the view

Update app-food-item to accept food item object

1. Make the food item object to be input property (@Input)

Create New FoodService service

1. Create new service called FoodService inside “food” folder
2. Move the food item list from app-food-menu component into FoodService
3. Create a helper method (getFoodItems) with active flag and launchDate as parameter, and return food items from service (later will be fetched from REST service)

4. Make sure service is injected at the root level

Update app-food-menu component to use service

1. Inject FoodService into app-food-menu component
2. On initializing the component (ngOnInit), get the food items from service and update the list inside component with the copy.

4.0 TYUS004 – Search Menu item by name

Create New FoodSearchComponent (app-food-search)

1. Create a component for searching food item inside “food\search” folder
2. Update view template to have search input field for food item name
3. Create an event handler method in app-food-search component, which will get triggered on “input” event of above input field

Update FoodService with method to return food items for a food name

1. Create a method, that takes name and return food items filtered matching the name of items from hardcoded collection (later this will invoke REST service)

Implement search using Subject in RxJS

1. Create a subject property in FoodService, which will be notified on the search text input from component
2. Listen (subscribe) for the subject event inside app-food-menu component’s ngOnInit method, and invoke new method added above in FoodService to fetch filtered food items.

5.0 TYUC005 – View menu items in Cart with total price

Create New CartComponent (app-shopping-cart)

1. Create separate folder “shopping” to have all cart related components
2. Create a component (shopping\cart) inside this folder
3. Update view template to render cart items with price and total

Create New CartService

1. Create new CartService inside “shopping\cart” folder
2. Create cart interface with cart items and the total property with appropriate type
1. Create temporary cart items property with some default data (to be removed on implementing next user story)
3. Create method to return cart items for the user logged in (later the REST service will be invoked)
4. Create method to calculate total price from items in cart
5. Create an event emitter “cartUpdated” to emit cart updated event

Integrate component to service

1. Inject CartService into the component
2. Inside ngOnInit of component, call the method in service to return cart items

6.0 TYUC004 – Add menu item to Cart

Update FoodService to add food item in cart

1. Add method (addToCart) to accept food item id and number of items to be added to cart, update the data in service with the input item (need to lookup food item from Food Service, and then update cart with item)
2. Update the cart total with the latest items in the cart object
3. Fire “cartUpdated” event in Cart service

Update FoodItemComponent to dispay “Add to Cart” feature

4. Update app-food-item component’s template with button for “Add to Cart”
5. Create outbound event (@Output) in app-food-item component to emit on clicking “Add to Cart” with the food item id

Update FoodMenuComponent to handle add to cart event

1. Add event handler to the event fired from app-food-item component
2. Inject CartService into app-food-menu component
3. Invoke the add item to cart in the Cart service
4. Navigate to CartComponent (routing to be implemented) programmatically

7.0 TYUC005 – Remove menu item from Cart

Update FoodService to remove food item in cart

1. Add method (removeFromCart) to accept food item id and number of items to be removed from cart, update the data in service with the input item
2. Update the cart total with the latest items in the cart object
3. Fire “cartUpdated” event in Cart service

Update app-shopping-cart to remove food item in cart

1. Add button for each item in app-shopping-cart component template
2. Create event handler to handle click event of above event, which passes item id
3. Invoke removeFromCart method in CartService

8.0 TYUC003 – Edit menu item

Update FoodService to edit food item

1. Add method (updateFoodItem) to accept food item object
2. Update the temporary food item list in component (later will be used to invoke REST service)

Create New FoodItemEditComponent (app-food-item-edit)

1. Create new component FoodItemEditComponent under “food\item-edit” folder
2. Update view template with form to update item properties
3. Validate all the properties per requirement
4. Add submit form handler and invoke edit food item service

Update app-food-item-info component

1. Add “Edit” link to app-food-item component for the food item with routing link passing item id.
2. Add isAdmin boolean property in component
3. On init of component, set this boolean to true or false based on the user role, who just logged in
4. Show or Hide “Edit” button, based on the above boolean property
5. Also configure the route to be authorized using Auth Gaurd

9.0 TYUS006 – Signup user

Create New UserService

1. Create a folder “site” under root folder “app”
2. Create new service “UserService” with temporary user list property
3. Add method to add user object to the list (later this method will invoke REST service)
4. Add method to get the user (later this method will invoke REST service)

Create New Component (app-signup)

1. Create SignUpComponent with selector “app-signup”
2. Update view template with form
3. Bind the form with component by reactive forms
4. Implement all validations
5. Inject UserService into component
6. Add form submit event handler and invoke add user method in UserService

10.0 TYUS007 – User Login

Create New AuthService

1. Create new service “AuthService” under “site” folder
2. Add loggedInUser property
3. Add method to authenticate user (later this method will invoke REST service)
4. Update loggedInUser object based on REST service response (mocked)

Create New Component (app-login)

1. Create new LoginComponent under “site\login” folder
1. Update template with login form
2. Bind the login form with template driven form
3. Implement all validations
4. Inject AuthService into component
5. Add form submit event handler and invoke authenticate method
6. On successful login, redirect to home page

11.0 TYUS008 – User Logout

Update AuthService

1. Add logout method to AuthService
2. Update method to set the loggedInUser to null (later will access REST service to kill the session in server)

Update AppComponent

3. Update site layout to have Log out link
4. Add method to AppComponent to return boolean, based on the loggedInUser object in AuthService
5. Show or Hide the Logout link, based on the above method return value
6. Add event handler in AppComponent on clicking the log out link
7. Invoke the logOut method in AuthService and redirect user to home page

12.0 TYUS005 – Redirect to Login with invalid access privilege

Create AuthGaurdService

1. Create AuthGaurdService implementing CanActivate under “site” folder
2. Implement canActivate method to check loggedInUser from AuthService is null, if not null, let the request proceed, otherwise redirect to login page with error message

Configure Routes

1. Update routes to be authorized with the above guard service

13.0 TYUS001, TYUS002 – Responsive to multiple devices

Configure project to use UI template

1. Update index.html with the template header and footer
2. Update the components template with appropriate elements from UI template

14.0 Submission

14.1 Code submission instructions

Use git add, commit and push commands to upload your code into remote GItLab repository.

During commit, give the commit message as “bootstrap”.

15.0 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**