**E-Shopping Zone**

Low-Level-Design (LLD) for Cart-Management Microservice

**Date: 09/07/2022**

**Current Document Version: 1.0**

DOCUMENT APPROVAL

**Approvers of this document**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Department** | **Role** | **Signature** | **Date** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Document Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Version #** | **Author** | **Date** | **Description** |
| 1.0 | Naga Manindra | 09/07/2022 | LLD for Cart-Management Microservice |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1.0 Document Purpose 4

2.0 Intended Audience 4

3.0 Project Background 4

4.0 Design Pattern 5

5.0 Solution Steps 5

6.0 Classes 6

7.0 Data Model 6

8.0 API canvas 6

9.0 ENV Variable 7

10.0 HTTP Status Code 7

11.0 Response 7

# Document Purpose

This document describes the solution architecture for E-Shopping Zone Cart-Management Microservice.

# Intended Audience

This document is intended as a reference for the following roles and stakeholders who are interested in the E-Shopping Zone Microservice technical architecture.

|  |  |
| --- | --- |
| Role | Nature of Engagement in WB Classics Portal Technical Architecture |
| Product Owners/SME | Key stakeholder to ensure that the architecture is aligned with business goals. |
| Business Analysts | Business analysts are one of the stakeholders who are informed with the key architectural decisions. |
| Enterprise Architects | To enforce E-Shopping Zone Platform Architecture is aligned to business goals and architecture, architectural guidelines. |
| Solution Architects | To ensure solution design and architecture is aligned to business requirements, architectural guidelines. |
| Developers | Use Technical Architecture Document as the guiding document for detail design and implantation approach to align with E-Shopping Zone Microservice |

# Project Background, Objective(s)

## Project Background

E-Shopping Zone is an ecommerce website, E-commerce is the activity of buying or selling of products on online services or over the internet. There are two roles one is merchant where he sells the products and other was the customers. Here customer can create account and login to their active account. Users can browse various types of products. Products on this site will be displayed category vies.

## Objective

Cart-Management Microservice ​ will perform the following

* User should be able perform CRUD operations in this microservice.
* User should be able to add item to the cart by productId and cartId, where cartId is username.
* User should be able to get the cart.
* User should be able to update the cart.
* User should be able to delete the item in cart.
* User should be able to delete the cart.

# Design Pattern

|  |  |  |
| --- | --- | --- |
| # | Name | Description |
| 1 | API | Using HTTP requests, we will use the respective action to trigger various operations |

# 5.0 Solution Steps

**Cart**

Adding to Cart:

When user clicks on add to cart, call reaches api gateway, Api gateway forwards call to cartController.addCart() which gets userName, productId and calls cartService.createCart() function which calls cartRepo.save() to save to the cart database.

Show Cart:

When user clicks on cart, call reaches api gateway, Api gateway forwards call to cartController.getCartByUserId() which calls cartService.getCart() function which calls cartRepo.findByCartId() to show all the added products in cart database.

Edit Cart:

When user make changes in cart, call reaches api gateway, Api gateway forwards call to cartController.updateItemsInCart() which gets userName, productId and calls cartService.updateCart() in which sub function getById() gets the cart details to update and it calls cartRepo.save() to save changes to cart database.

Remove Cart item:

When user click on delete in cart, call reaches api gateway, Api gateway forwards call to cartController.deleteItemFromCart() which gets userId, productId and calls cartService.deleteCart() in which sub function getById() gets the cart details to delete and it calls cartRepo.save() to remove item from cart database.

Remove Cart:

When user click on delete in cart, call reaches api gateway, Api gateway forwards call to cartController.deleteCart() which gets cartId and calls cartService.deleteCart() in which sub function getById() gets the cart details to delete and it calls cartRepo.remove() to remove item from cart database.

# 6.0 Classes

|  |  |  |
| --- | --- | --- |
| **#** | **Class** | **Description** |
| 1 | cart | Model holds the cart schema details |
| 2 | cartRepo | This class deals with the data accessibility for cart. |
| 3 | cartService | It contains the core logic for the adding, removing, updating, getting cart. Which calls the cartRepo class to connect to cart database |
| 4 | cartController | This class deals with the data accessibility for cart creation, editing, removing, displaying and getting data from product and user. |

# 7.0 Data Model

|  |  |
| --- | --- |
| Cart |  |
| id | int |
| totalPrice | double |
| items | List<Items> |

|  |  |
| --- | --- |
| Items |  |
| product | Product |
| subTotal | double |
| quantity | int |

|  |  |
| --- | --- |
| Product |  |
| productName | string |
| productId | int |
| price | double |

# 8.0 API canvas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Micro Service | Path | Verb | API Description | Role |
| Cart-Management | /user/additem/{cartId}/{productId} | POST | Cart creation | User |
| Cart-Management | /user/getcart/{cartId} | GET | To get cart list | user |
| Cart-Management | /user/updateitem/{id}/{productId}/  {quantity} | PUT | To update cart details | User |
| Cart-Management | /user/deleteitem/{cartId}/{productId} | DELETE | To remove the cart | User |

# 9.0 ENV variables

server:

port: 9007

spring:

data:

mongodb:

uri: mongodb+srv://manindra08:Naga7550@cluster0.uwtgi.mongodb.net/user

cloud:

config:

uri: http://localhost:8888

# 10.0 HTTP status code

200 - Request succeeded

201 – Created

204 – No Content

208 - Already reported

400 – Inputs are invalid

404 –Page Not found

502 – Bad gateway

**11.0 Response**

**If item added successfully**

{

"message": "successfully"

}

status code: 201

**If cart updated successfully**

{

"message": "successfully"

}

status code: 200

**If server encounters unexpected error**

{

“message”: “Internal server error”

}

status code: 500

**If product not in database**

{

“message”: “product not found”

}

status code: 500

**If item deleted successfully**

{

“message”: “item deleted”

}

status code: 200

**If cart deleted successfully**

{

“message”: “cart deleted”

}

status code: 204