**E-Shopping Zone**

Low-Level-Design (LLD) for Product-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 Product-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 7

8.0 API canvas 7

9.0 ENV Variable 8

10.0 HTTP Status Code 8

11.0 Response 8

# Document Purpose

This document describes the solution architecture for E-Shopping Zone Product-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

Product-Management Microservice ​ will perform the following

* User Should be able to get All products or get by Name, Category, Type.
* Only Admin should be able to create a new product where productId is unique.
* Only Admin should be able to Update the product.
* Only Admin Should be able to Delete the product.
* Admin also Should be able to get All products or get by Name, Category, Type.

# Design Pattern

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

# 5.0 Solution Steps

**Product**

All products:

When user want to view all the products, call reaches api gateway, Api gateway forwards call to UserProductController.getAllProducts() which calls productService.getAll() function which calls productRepo.findAll() which returns all the products stored in database.

Search by Category:

If user want search product by category, call reaches api gateway, Api gateway forwards call to UserProductController. getProductByCategory () which calls productService.getProductByCategory() function which calls productRepo.findProductByCategory () which returns all the products by category stored in database.

Search by Id:

If user want search product by category, call reaches api gateway, Api gateway forwards call to UserProductController. getProductById() which calls productService.getProductById() function which calls productRepo.findProductById() which returns all the products by id stored in database.

Search by Type:

If user want search product by category, call reaches api gateway, Api gateway forwards call to UserProductController.getProductByType() which calls productService.getProductByType() function which calls productRepo.findProductByType() which returns all the products by type stored in database.

Search by Name:

If user want search product by category, call reaches api gateway, Api gateway forwards call to UserProductController. getProductByName() which calls productService.getProductByName() function which calls productRepo.findProductByName() which returns all the products by Name stored in database.

Add Product:

If admin want to add Product, call reaches api gateway, Api gateway forwards call to AdminProductController.addProduct() which consumes Product details contain productId, productName, Type, Category, price, image, rating, description and calls productService.createProduct() function which calls productRepo.save() which creates new document in database.

Delete Product:

If admin want to delete product, call reaches api gateway, Api gateway forwards call to AdminProductController.deleteProduct() which calls productService.getProductById() and calls productService.delete() function which calls productRepo.remove() and which deletes the product stored in database.

Update Product:

If admin want to update product, call reaches api gateway, Api gateway forwards call to AdminProductController.updateProduct() which calls productService.getProductById() and calls productService.update() function which calls productRepo.save() and which update the product stored in database.

# 6.0 Classes

|  |  |  |
| --- | --- | --- |
| **#** | **Class** | **Description** |
| 1 | product | Model holds the product schema details |
| 2 | productRepo | This class deals with the data accessibility for product |
| 3 | productService | It contains the core logic for the adding, removing, updating, getting products. Which calls the productRepo class to connect to product database |
| 4 | userProductController | This class deals with the data accessibility for getting products called by user. |
| 5 | adminProductController | This class deals with creating, updating, deleting and getting product called by admin only |
| 6 | productNotFound | This class deals with exception handling, whenever wrong product is called this class is accessed. |

# 7.0 Data Model

|  |  |
| --- | --- |
| Product |  |
| id | int |
| name | string |
| image | string |
| type | string |
| price | double |
| category | string |
| description | string |

# 8.0 API canvas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Micro Service | Path | Verb | API Description | Role |
| Product-Management | /admin/addProduct | POST | Product creation | Admin |
| Product-Management | /admin/getAll | GET | To get product list | Admin |
| Product-Management | /admin/update/{id} | PUT | To update product details | Admin |
| Product-Management | /admin/delete/{id} | DELETE | To remove the product | Admin |
| Product-Management | /admin/getById/id | GET | To get details of a particular product | Admin |
| Product-Management | /admin/getByName/name | GET | To get details of a particular product | Admin |
| Product-Management | /admin/getByType/type | GET | To get details of a particular product | Admin |
| Product-Management | /admin/getByCateogory/category | GET | To get details of a particular product | Admin |
| Product-Management | /user/getById/id | GET | To get details of a particular product | User |
| Product-Management | /user/getByName/name | GET | To get details of a particular product | User |
| Product-Management | /user/getByType/type | GET | To get details of a particular product | User |
| Product-Management | /user/getByCateogory/category | GET | To get details of a particular product | User |
| Product-Management | /user/getAll | GET | To get product list | User |

# 9.0 ENV variables

server:

port: 9002

spring:

data:

mongodb:

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

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 valid details**

{

"message": "successfully"

}

status code: 201

**If invalid details**

{

“message”: “Inputs are not valid”

}

status code: 400

**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 product already exist in database**

{

“message”: “product already exist”

}

status code: 208

**If product deleted successfully**

{

“message”: “product deleted”

}

status code: 204