|  |
| --- |
|  |

|  |
| --- |
| **eMart (Mid-Tier Phase 3) v4.0** |
| Case Study |
|  |
| This document covers Software Requirements of eMart, along with list of Technologies to be used to develop this Software System, and also includes some details on the Architecture |
|  |
| **IIHT** |
| **1/10/2019** |
|  |

Table of Contents

[1. Business Requirement(eMart) 2](#_Toc28991760)

[1.1. Roles 2](#_Toc28991761)

[1.1.1. Buyer Use Cases. 2](#_Toc28991762)

[1.1.2. Seller Use Cases 2](#_Toc28991763)

[1.1.3. Admin Use Cases(optional) 3](#_Toc28991764)

[1.1.4. Data fields 3](#_Toc28991765)

[2. Design Inputs 5](#_Toc28991766)

[3. Development of individual Microservices 6](#_Toc28991767)

[4. Architecture Diagram 7](#_Toc28991768)

[5. Full Stack Technologies 8](#_Toc28991769)

[6. Database Tables 8](#_Toc28991770)

[7. Technical Spec – Solution Development Environment 8](#_Toc28991771)

[7.1. Front End Layer 8](#_Toc28991772)

[7.2. Middle Tier Layer 9](#_Toc28991773)

[7.3. Database & Integration Layer 9](#_Toc28991774)

[7.4. Ancillary Layer 9](#_Toc28991775)

[7.5. Security 9](#_Toc28991776)

[7.6. Deployment & Infrastructure 9](#_Toc28991777)

[7.7. Editors 9](#_Toc28991778)

[8. Assessment Deliverables 10](#_Toc28991779)

[9. Important Instructions 10](#_Toc28991780)

# Business Requirement(eMart)

eMart is a online eCommerce portal used to search and buy an item online. Buyer can add items to the cart, checkout and perform other operations. Admin can block/unblock Seller, Buyer, selling items. Below are eMart Features in detail.

## Roles

Below are the different roles, which need to be supported by above Software System.

#1. Buyer

#2. Seller

#3. Admin(optional)

Below are the Use Cases which need to be supported by each of above Roles

### Buyer Use Cases.

Login/Logout

Signup

Should be able to search an Item.

Once list of items are displayed, it should be possible to filter items based on Price, Manufacturer, etc...

When a specific item is selected by Buyer, complete details of item need to be displayed along with picture(one or more) and (list of)specifications which depends on Category and Sub Category of the item. Add to Cart button need to be provided.

It should be possible to open the Cart and checkout. Tax amount need to be displayed.

Items in the Cart can be deleted

It should be possible to apply discount, before checking out.

It should be possible to view History of purchases, along with Item, numbers

Integration with any Payment Gateway(optional)

### Seller Use Cases

Login/Logout

Signup with details such as Company/individual name, email id, Postal Address, GSTIN, Bank details

Add an item to be sold along with the number of items(in Stock)

Should be able to view Inventory(items sold and remaining)

Should be able to generate report like number of items sold and which items, over certain period.

Tax Calculation need to be performed.(optional)

### Admin Use Cases(optional)

Login/Logout

Block/Unblock a specific Seller

Block/Unblock a specific item sold by a Seller

Block/Unblock a specific Buyer

Add/remove Categories, Sub Category along with GST

Add/update discounts - discount code, start date, end date

View daily Turnover Category wise

**NOTE: Shipping/Delivery Tracking related Use Cases are not in the scope, it is assumed that Seller need to manage Shipping/Delivery Tracking offline.**

### Data fields

Below are data fields, for your reference

**Buyer:** Buyer’s login and profile details

id

username

password

emailid

mobile number

created datetime

**Seller:** Seller’s login and Selling company details

id

username

password

companyname

GSTIN

brief about company

postal\_address

website

emailid

contact number

**Category:** List of Categories, for example Electronic, Fashion, etc…

category\_id

category\_name

brief\_details

**Subcategory:** Sub Category of each Category. For example Electronic Category can have Mobile, TV, Laptop, etc… as Sub categories

subcategory\_id

subcategory\_name

category\_id

brief\_details

GST %

For example Mobiles, TV, etc... can be Sub Catgeories in Electronic Category

**Items:** Selling Item details

id

category\_id

subcategory\_id

price

item\_name

description

stock\_number

remarks

**Purchase History:**

Id

Buyer\_id

Seller\_id

Transaction\_id

Item\_id

Number\_of\_items

Date\_time

remarks

**Transactions:** Transactions performed during Checkout, etc…

id

user\_id

seller\_id

transaction\_type(Eg. debit or credit)

date\_time

remarks

**Discounts:** Discount details

Id

Discount\_code

percentage

start\_date

end\_date

description

# Design Inputs

Next sections in this document provides inputs on designing the solution for above requirements.

Design inputs provided in this document are just for your reference purpose, Associates can make changes or additions to the Design, based on their analysis.

# Development of individual Microservices

This specific Phase is to design/develop individual Microservices. Analyze the requirement and divide Mid Tier functionality into multiple Microservices. Based on the eMart requirements, below can be possible Microservices, along with possible REST controllers

1. User Microservice – login, logout, signup
2. Buyer Microservice – Search Items, Filter Items, addCartItems, deleteCartItems, checkout
3. Seller Microservice – addItems, deleteItems, updateItemStock, viewStock, viewBalanceAndTax
4. Admin Microservice(optional) – blockBuyer, blockSeller, blockCategory, blockSubCategory, blockItem

Each of above Microservices needs to comprise below functionality, which need to be developed

1. REST Controllers
2. Service Layer
3. Entity & Model classes, including appropriate relationship (like One-One, Many-One, etc…) between Entity Classes. (Entity and Model classes have been developed in the Previous Phase)
4. In case specific Entity or Model classes are required across multiple Microservices, it is recommended to maintain separate copy of Entity or Model classes for each Microservice, for flexibility reasons.
5. Microservice interaction with corresponding DB tables or Databases it owns.
6. It is possible that one Microservice need to interact with other Microservice(using RestTemplate or FeignClient)
7. Repository class which implements JPA or CrudRepository, if RDBMS is used
8. Usage of Custom Queries using @Query where ever custom functionality required
9. Feign Client can be used to invoke one Microservice, from another Microservice
10. Send Email Notifications to Users/Mentors in Use Cases wherever appropriate.
11. Use Postman to test the Microservices by directly passing requests to each REST end Point, of each Microservice
12. Unit Testing code can be developed using JUnit, Mockito, and perform Unit Testing

# Architecture Diagram

Multiple Microservices interacting with Database each one owns.



Architecture of a Single Microservice with REST Controller, Service, Model & Entity Classes and Repository classes



# Full Stack Technologies

The technologies included in Full Stack are not limited to following but may consist of:

* UI Layer (HTML5, CSS3, Bootstrap 4, JavaScript, Jquery, Angular 4/6)
* Middleware Restful API (Spring Boot Restful & MicroServices, JAX-RS, Spring MVC)
* Database Persistence ( Hibernate)
* Database layer (MySQL or MongoDB)
* Ancillary skills (GIT, Jenkins(CI/CD), Docker, Maven) etc.

To complete this case study, you should be comfortable with basic single page web application concepts including REST and CRUD. You may use angular-cli to create your template project. All web pages need to be responsive.

Ref1: https://cli.angular.io/

Ref2: <https://github.com/angular/angular-cli>

# Database Tables

Below are list of Database Tables, which can be used by corresponding Microservice. Though, ideally each Microservice need to use separate database, it should be fine to place all below DB Tables in a single database

|  |  |
| --- | --- |
| Table Name | Purpose |
| Buyer | Stores Buyer’s login and profile details |
| Seller | Stores Seller’s login and Selling company details |
| Category | Stores list of Categories, for example Electronic, Fashion, etc… |
| Subcategory | Stores Sub Category of each Category. For example Electronic Category can have Mobile, TV, Laptop, etc… as Sub categories |
| Items | Stores Selling Item details, (added by Seller) |
| PurchaseHistory | Stores list of Items purchased by specific Buyer |
| Transactions | Transactions performed during Checkout, etc… |
| Discounts | Discount code and related details |

Refer Entity classes to identify Columns in each of the DB Table.

# Technical Spec – Solution Development Environment

## Front End Layer

|  |  |
| --- | --- |
| **Framework(s)/SDK/Libraries** | **Version** |
| Angular with TypeScript | 4/6 |
| Bootstrap | 3.0 or above |
| CSS | 3 |
| HTML | 5 |
| JavaScript | 1.8 or above |
| JQuery | 1.3 |

## Middle Tier Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Java Stack | Spring Boot | 1.5 or above |
| Spring MVC | 4.0 or above |
| JDK | 1.7 or above |
| Maven | 3.x or above |

## Database & Integration Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Java Stack | Hibernate | 4.0 or above |
| JAX-RS Jersey/ Spring Restful |  |
| MySQL | 5.7.19 |
| MongoDB | MongoDB | 3.4 |
| NoSQL |  |

## Ancillary Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Source Code Management Tool | GIT | 2.14.2 |
| Build Tool/JAVA Stack | Maven | 3.x |
| Testing Tool/JAVA Stack | JUnit/Mockito | 4.x |
| Testing Tool/JAVA Stack | Spring Test | 4.x |
| Controllers can be tested using Postman Tool | | |

## Security

|  |  |
| --- | --- |
| **Name** | **Version** |
| Spring Boot Security |  |
| JWT |  |

## Deployment & Infrastructure

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Docker | - |  |
| Apache Tomcat | - |  |
| Jenkins(CI/CD) | - |  |
| Node | - |  |

## Editors

|  |  |
| --- | --- |
| **Name** | **Version** |
| STS(Spring Tool Suite) |  |
| Visual Studio Code |  |

Agile/Scrum Software development Model can be used

# 

# Assessment Deliverables

1. Checked in(to internal GIT) Source code of all individual Microservices
2. Screen shots of Usage of Post Man tool to test each End Point of all Microservices
3. Few Steps on how to run the solution.
4. Unit Testing code need to be included

# Important Instructions

1. Consider using below Java features
2. Lambda Expressions
3. Collection Streams
4. Generics
5. Sample Design provided is just for reference, Associates can make changes over it or follow their own Design.
6. Based on your current work, alternate Technologies can be used, for example ReactJS instead of Angular, etc…, however prior approval from the Mentor is required.
7. Please make sure that your code does not have any compilation errors while submitting your case study solution.
8. The final solution should be a zipped code having solution. Solution code will be used to perform Static code evaluation.
9. Implement the code using best design standards/family Design Patterns.
10. Use Internationalization for all the labels and messages in Rest API Development.
11. Do not use System out statements or console.log for logging in Rest API and FrontEnd respectively. Use appropriate logging methods for logging statements/variable/return values.
12. If you are using Spring Restful or Jersey JAX-RS to develop Rest API, then use Maven to build the project and create WAR file.
13. Write web service which takes input and return required details from database.
14. Use JSON format to transfer the results.

For any further queries you can contact fullstack@iiht.com