|  |
| --- |
|  |

|  |
| --- |
| **eMart (Deployment Phase6) 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** |
| **2/7/2019** |
|  |

Table of Contents

[1. Business Requirement(eMart) 2](#_Toc28992047)

[1.1. Roles 2](#_Toc28992048)

[1.1.1. Buyer Use Cases. 2](#_Toc28992049)

[1.1.2. Seller Use Cases 2](#_Toc28992050)

[1.1.3. Admin Use Cases(optional) 3](#_Toc28992051)

[1.1.4. Data Fields 3](#_Toc28992052)

[2. Design Inputs 6](#_Toc28992053)

[3. Jenkins CI/CD 6](#_Toc28992054)

[4. Configure Jenkins and Docker for the Project 6](#_Toc28992055)

[5. Dockerization 7](#_Toc28992056)

[6. Perform CI/CD 8](#_Toc28992057)

[7. Diagram 8](#_Toc28992058)

[8. Deliverables of this Phase 8](#_Toc28992059)

[9. Deliverables from all Phases 8](#_Toc28992060)

[10. Final Deliverables Folder Structure 9](#_Toc28992061)

[11. Full Stack Technologies 9](#_Toc28992062)

[12. Technical Spec – Solution Development Environment 10](#_Toc28992063)

[Front End Layer 10](#_Toc28992064)

[Middle Tier Layer 10](#_Toc28992065)

[Database & Integration Layer 10](#_Toc28992066)

[Ancillary Layer 10](#_Toc28992067)

[Security 11](#_Toc28992068)

[Deployment & Infrastructure 11](#_Toc28992069)

[Editors 11](#_Toc28992070)

[13. Important Instructions 11](#_Toc28992071)

# 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. Based on your analysis you may add/remove Data fields.

**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.

# Jenkins CI/CD

**Jenkins CI/CD:** As already known Jenkins is popular tool to perform CI/CD. When the code is pushed to GIT, build need to be automatically fired and deployed. If possible create a Docker image and run the Container on Docker Host

**Deployment on Cloud(optional):** Any of the Microservices or Front End can be deployed on any Cloud(AWS, Azure, etc…) of your choice.

Jenkinsfile with pipeline, having below steps to build, test and deploy

* + Pull code from Git repository
  + Check Code Quality(using SonarQube)
  + Perform Build
  + Containerize(Create Docker Images)
  + Run the Docker Images and start Containers

# Configure Jenkins and Docker for the Project

* Import the project (as discussed above) in Spring Tool Suite and configure it locally to run it as Spring Boot App.
* You may need to configure MySQL credentials and database name.
* Execute the project locally and access the app at http://localhost:portnumber
* Once, it is working fine in local development environment; Configure CI/CD in Jenkins, along with Dockerization
* Push the app source in internal GIT server. Internal GIT is 172.18.2.18 which can be accessed from IIHT VMs only.
* Configure Jenkins locally to pull the source from internal GIT repository
* Jenkins should build the project and create the deployable (war/jar). Should run Unit Tests.
* From Jenkins, invoke Docker commands to perform, below
* Creation of Docker Image(docker build . )
* Create and run Docker Container(docker run <image\_id>)

# Dockerization

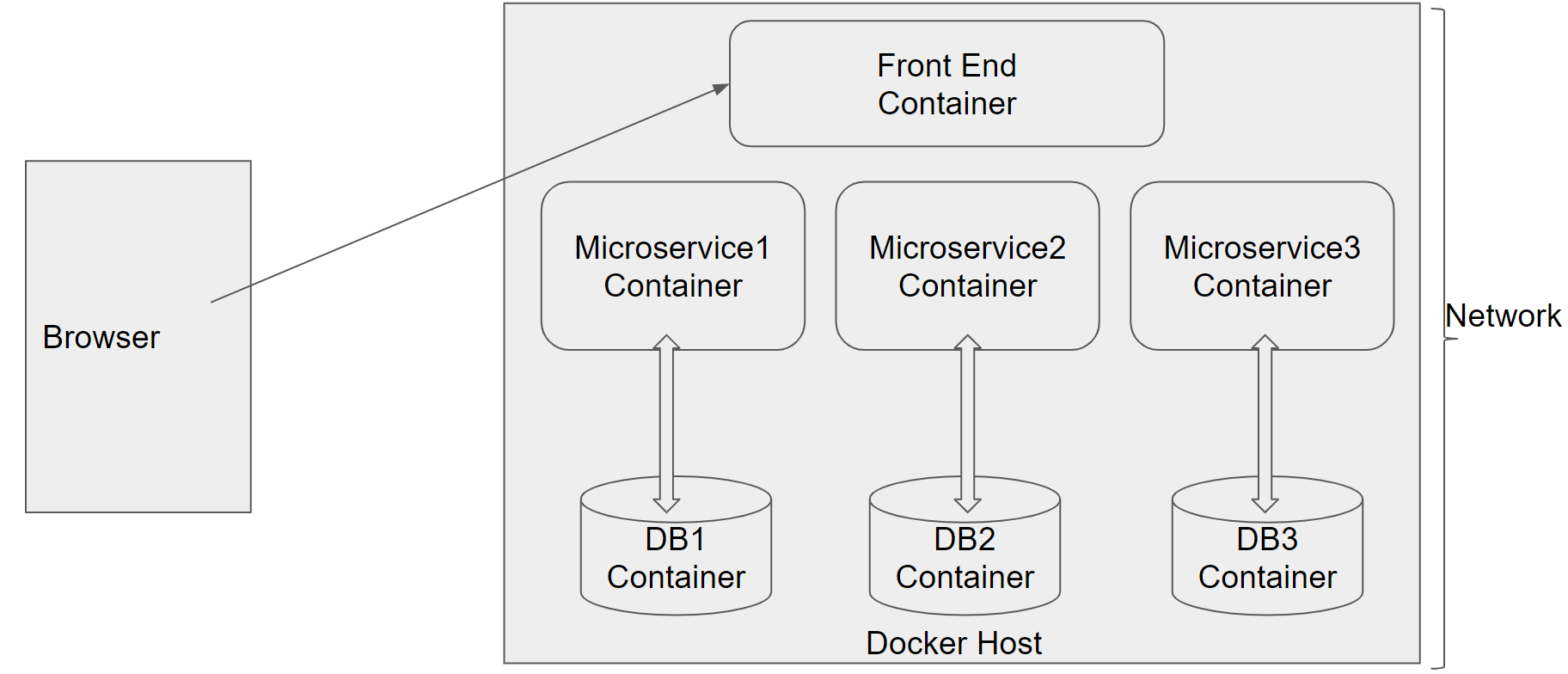
It is mandatory that all standalone applications of the Project such as below need to be Dockerized and deployed on Docker Host.

* All Microservices
* Database Server
* Angular Front End

Each need to have a corresponding Dockerfile, and a Docker container. MySQL DB Image is prepcreated and need to be pulled from hub.docker.com

To create and deploy all containers with ease, either docker-compose or docker commands can be used.

Deployment of Microservices, Database Servers and Front End on Docker Host can be visually represented as below.



To Setup Docker Client on your VM please refer <https://github.com/vskreddy652/Genc_BatchB/blob/master/Docker%20Remote%20Host%20Access%20Steps%20(3).docx>

# Perform CI/CD

1. Make few changes in the project (source code)

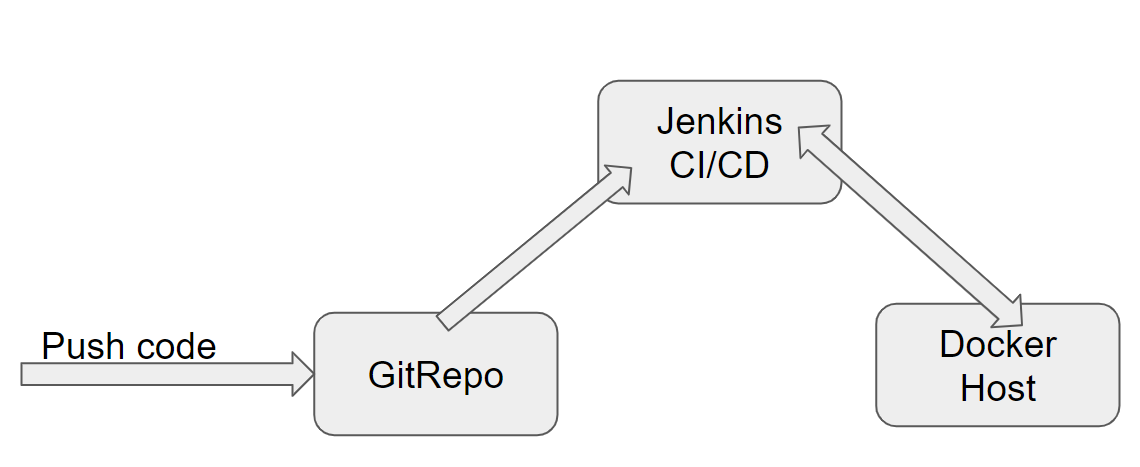
2. Make it sure that project is running locally in development environment without errors.

3. If it running locally without errors, push the changes to the internal GIT repository which was connected

4. If Project was Setup properly, Jenkins will automatically pull the code updates from internal GIT repo and build and deploy the project with updated code.

5. Now, when you visit http://localhost; you should see the changes in the browser window

# Diagram



# Deliverables of this Phase

1. Jenkinsfile, Jenkins screenshots
2. Dockerfile and Docker commands used and Screen shots of usage of Docker

# Deliverables from all Phases

Below deliverables need to be checked in(to internal GIT or github)

1. FrontEnd Source code, Unit Tested using Jasmine
2. Mid Tier Source code of all Microservices, Unit Tested using Spring Test
3. Screen shots of Usage of Post Man tool to test each End Point of all Microservices
4. Few Steps on how to run the solution.
5. Test code of Angular and Mid Tier need to be included
6. Jmeter’s JMX file to test atleast one REST End point, and Screenshot of report
7. Dockerfile & docker-compose
8. Jenkinsfile or Jenkins UI ScreenShot
9. URL where the Project is deployed

Below Traceability Matrix need to be filled by Associates and need to be placed in Root Folder, while delivering the Project.

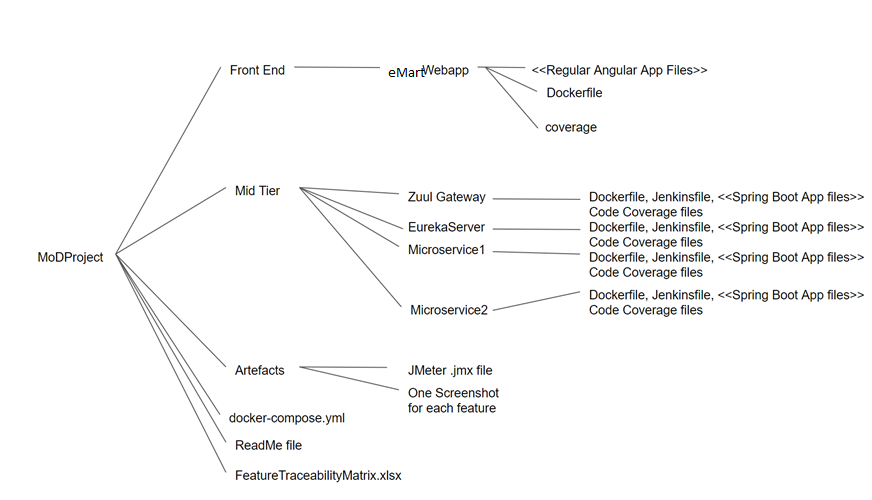
<https://github.com/vskreddy652/Genc_BatchB/blob/master/TraceabilityMatrix.xlsx>

# Final Deliverables Folder Structure

Below delivery structure includes Deliveries from all other Previous phases as well

It is mandatory to follow below folder structure for the Project delivered by Associates.

(If any specific Items not specified in the below sample folder structure, the same can be included in the appropriate related folders)



ReadMe file should include instructions on

1. how to run the project and check output
2. url at which the project is deployed and can be accessed
3. any other specific inputs for Evaluation

# 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

# 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

# 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