|  |
| --- |
|  |

|  |
| --- |
| **Pixogram (Deployment Phase6) v4.0** |
| Case Study |
|  |
| This document covers Software Requirements of Pixogram, 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(Pixogram) 2](#_Toc28558779)

[NOTE: Features marked as optional, above are not mandatory for GenCs 3](#_Toc28558780)

[Overview of Fields used in User Registration 3](#_Toc28558781)

[Overview of fields used for Add Content 4](#_Toc28558782)

[2. Design Inputs 5](#_Toc28558783)

[3. Jenkins CI/CD 5](#_Toc28558784)

[4. Configure Jenkins and Docker for the Project 6](#_Toc28558785)

[6. Perform CI/CD 6](#_Toc28558786)

[7. Diagram 7](#_Toc28558787)

[8. Other Design constraints 7](#_Toc28558788)

[9. Assessment Deliverables 7](#_Toc28558789)

[5. Full Stack Technologies 7](#_Toc28558790)

[6. Spring Microservices Tools to be used 8](#_Toc28558791)

[7. Technical Spec – Solution Development Environment 8](#_Toc28558792)

[7.1. Front End Layer 8](#_Toc28558793)

[7.2. Middle Tier Layer 8](#_Toc28558794)

[7.3. Database & Integration Layer 8](#_Toc28558795)

[7.4. Ancillary Layer 8](#_Toc28558796)

[7.5. Security 9](#_Toc28558797)

[7.6. Deployment & Infrastructure 9](#_Toc28558798)

[7.7. Editors 9](#_Toc28558799)

[10. Important Instructions 9](#_Toc28558800)

# Business Requirement(Pixogram)

Pixogram is a Social Media portal, which lets Users upload, add effects to Pictures and other Media. Users can manage the Gallery created out of uploaded Media Content. Media can be shared with the Followers. Followers can comment or like the Shared Images.

Below are the features which need to be supported by Pixogram:

The Pixogram (Single Page Picture Sharing Application) allows you to:

1. Register as a user

2. Login as a user

3. Retrieve password(when Forgot)

4. Manage your (user) account

5. Login/Logout to/from your account on Pixogram

6. Add Media Content

a. Upload single/multiple pictures, caption and description

b. Upload single/multiple videos, caption and description(optional)

7. Manage Content

a. Organize Picture in Gallery

b. Organize Videos in Playlists(optional)

c. Rename Pictures

d. Edit Caption, Description, Comment

8. Social Features

a. Use emojis in comment(optional)

b. Like or Unlike, comment, pictures and videos(optional) of other users

c. Follow/Unfollow other users

9. Hide Pictures/Videos

10. Activity/Newsfeed

a. View activity log of user-activity(i..e posted Images) on the PixoGram. That means a User’s Newsfeed shows all the Images posted by Users whom current User follows, in reverse chronological order.

11. Offline Functionality(optional):

a. Certain parts of the application should be available in absence of connectivity.

b. Relevant areas on the screen should display “Connectivity Not Available”

12. BONUS REWARDS/SCORE Feature(optional)

a. To implement offline image upload functionality such that user can upload content when offline. It will sync with backend when connected.

### NOTE: Features marked as optional are not mandatory for GenCs

### Overview of Fields used in User Registration

The application will consist of 7 fields. Given below are the fields and validation guidelines (as used in creation of UI. Some of the guidelines given for the fields in this section may not be applicable to the Java layer).

1. First Name:

a. Should allow alphabets only

2. Last Name:

a. Should allow alphabets only

3. Username

a. Should allow mix of alphabets and number

b. Username must not start with number

c. Length of username should be between 8 to 12

4. Email

a. Must allow email in valid email format

b. Must not allow two @ symbols

5. Password

a. Must be alphanumeric

b. Should allow only following special characters- . # % $ !

c. Length of password should be between 8 to 12

d. Should contain at-least one capital alphabet

6. Confirm Password

a. Should be like the above password

b. Same validation rules should apply

7. Upload profile picture

a. Upload the profile picture. Picture should be of dimension 200x200 before upload

Spreadsheet Wireframe: Empty form (Do not create in project. FYI only.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| First Name | : |  | | |
|  |  |  |  |  |
| Last Name | : |  | | |
|  |  |  |  |  |
| User Name | : |  | | |
|  |  |  |  |  |
| Email | : |  | | |
|  |  |  |  |  |
| Date Of Birth | : |  | | |
|  |  |  |  |  |
| Password | : |  | | |
|  |  |  |  |  |
| Confirm Password | : |  | | |
|  |  |  |  |  |
| Profile Picture | : |  | Browse |  |
|  |  |  |  |  |
|  | Submit |  | Reset |  |

### Overview of fields used for Add Content

There are two scenarios for content input:

1. Single Image Input

a. Title – can be alphanumeric. The length should not go beyond 80 characters.

b. Description – can be alphanumeric. The length should not go beyond 144 characters.

c. Image name – can be alphanumeric. You must supply full image name (e.g. imagesample.jpg)

d. Date – It should take current date and time using Date object.

e. The program will response with success or failure depending on whether image was saved in the database or not.

f. If success, program will end.

g. If failure, program will re-start.

1. Multiple Image Input

a. Title – can be alphanumeric. The length should not go beyond 80 characters.

b. Description – can be alphanumeric. The length should not go beyond 144 characters.

c. Image name – can be alphanumeric. You must supply multiple image names separate by comma “,” (e.g. imagesample1.jpg, imagesample2.jpg etc)

d. Date – It should take current date and time using Date object.

e. The program will response with success or failure depending on whether multiple images was saved in the database or not. Here, each image saved will have same title and description as input above.

f. If success, program will end.

g. If failure, program will re-start.

# 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). It should run the unit tests created in ”Maven, GIT, Junit, Tomcat Micro Layer for the Project”
* From Jenkins, invoke Docker commands to perform, below
* Creation of Docker Image(docker build . )
* Create and run Docker Container(docker run <image\_id>)

# Dockerization

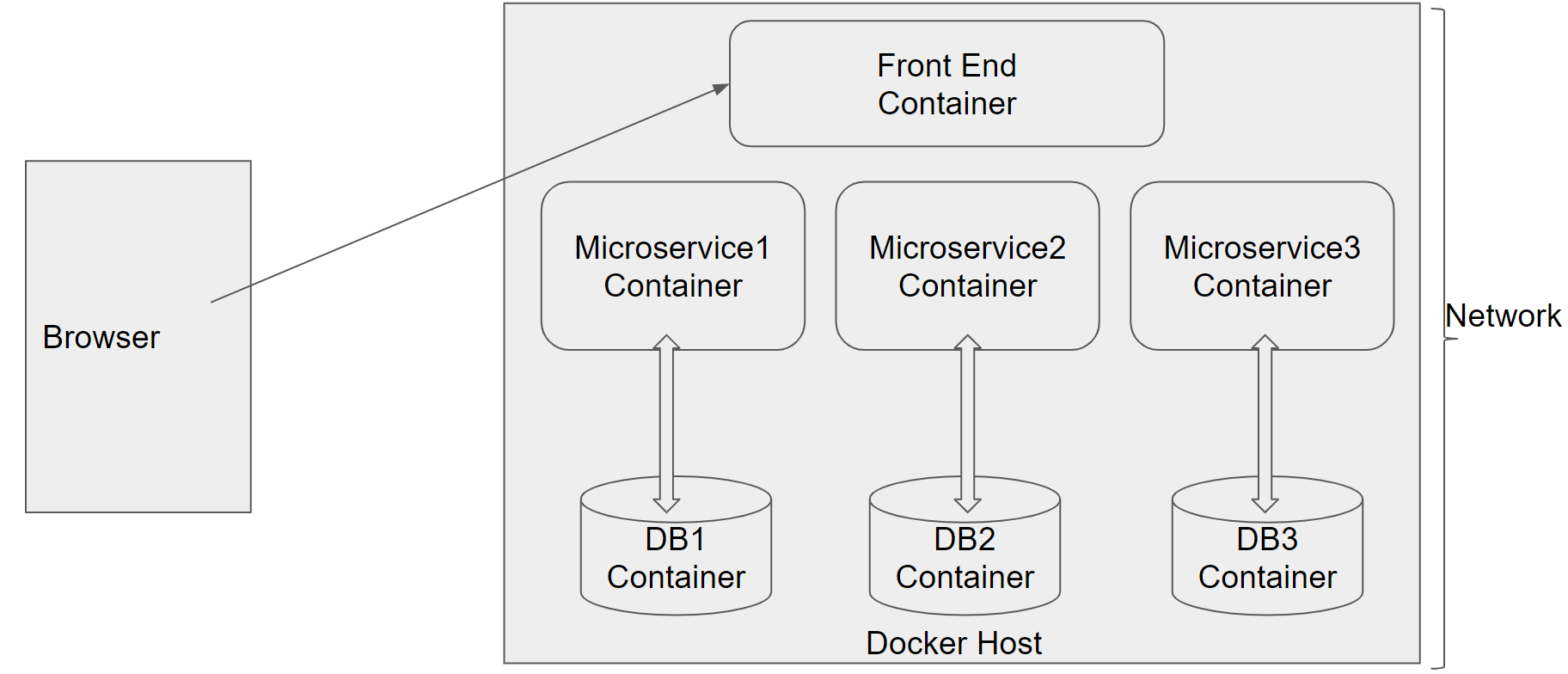
It is mandatory that all standalone applications of the Project such as below need to be Dockerized and deployed on Docker Host.

* 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)

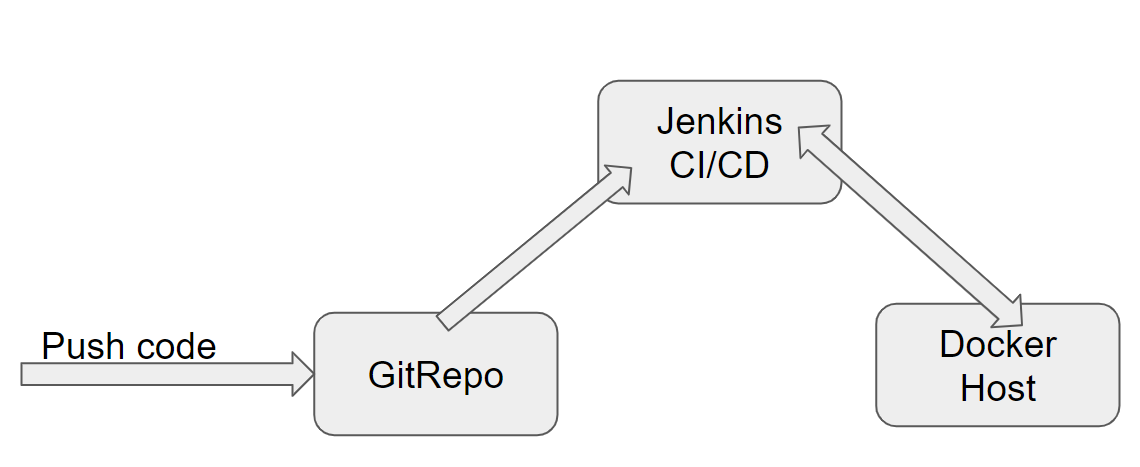
2. Make it sure that project is running locally in development environment without errors.

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.

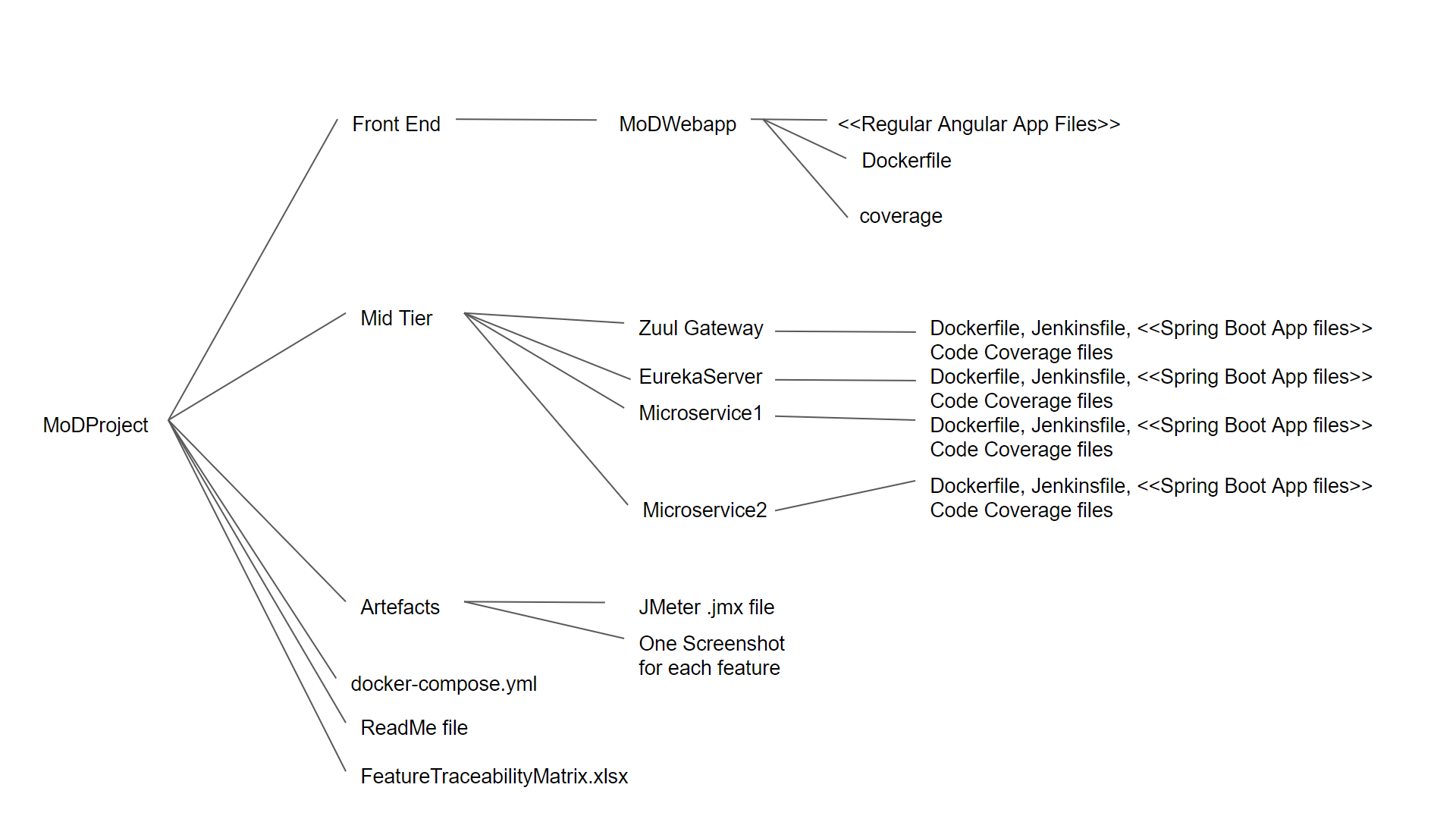
<https://github.com/vskreddy652/Genc_BatchB/blob/master/TraceabilityMatrix.xlsx>

# 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 covers 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, 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