|  |
| --- |
|  |

|  |
| --- |
| **Mentor On Demand (Mid Tier) Phase5 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](#_Toc28724494)

[NOTE: Features marked as optional are not mandatory for GenCs 3](#_Toc28724495)

[Overview of Fields used in User Registration 3](#_Toc28724496)

[Overview of fields used for Add Content 4](#_Toc28724497)

[2. Design Inputs 5](#_Toc28724498)

[3. Microservices Integration and Security 5](#_Toc28724499)

[3. Spring Microservices Tools to be used 6](#_Toc28724500)

[4. JWT Authentication 6](#_Toc28724501)

[5. Architecture/Design 7](#_Toc28724502)

[6. Technical Spec – Solution Development Environment 7](#_Toc28724503)

[6.1. Front End Layer 7](#_Toc28724504)

[6.2. Middle Tier Layer 7](#_Toc28724505)

[6.3. Database & Integration Layer 7](#_Toc28724506)

[6.4. Ancillary Layer 7](#_Toc28724507)

[Controllers can be tested using Postman Tool 8](#_Toc28724508)

[6.5. Security 8](#_Toc28724509)

[6.6. Deployment & Infrastructure 8](#_Toc28724510)

[6.7. Editors 8](#_Toc28724511)

[7. Assessment Deliverables 8](#_Toc28724512)

[8. Important Instructions 8](#_Toc28724513)

# 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.

2. 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.

# Microservices Integration and Security

Assuming that you are done with developing individual Microservices in previous Phase, current Phase includes creating and integrating Zuul gateway, Eureka Server and Eureka client in each Microservice. This is shown in architecture Diagram, in next section.

Zuul Gateway(create a Zuul based Project using Spring Initilaizer or STS IDE), add required annotation. Authentication and JWT Token validation can be performed in Zuul’s Pre Filter.

Add below details to yml or property file

1. add route configurations
2. port number & url of eureka Server

Eureka Server(create a Eureka Discovery Server using Spring Initializer or STS IDE), add required annotation & port number in yaml configuration file

Add Eureka Discovery Client to all the Microservice

Now open Eureka Server Dashboard by opening and crosscheck if all Microservices are registered in the dashboard

Now start sending the requests to Zuul Gateway which further routes to a specific Microservice based on the url pattern

Develop code for Unit Testing

PostMan, to test REST end points

# Spring Microservices Tools to be used

As already specified under Full Stack Technologies Microservice Architecture need to be followed. Ensure that the Application is divided into multiple Microservices, along with database/tables each Microservice Manages. Below Spring Microservices Tools need to be used

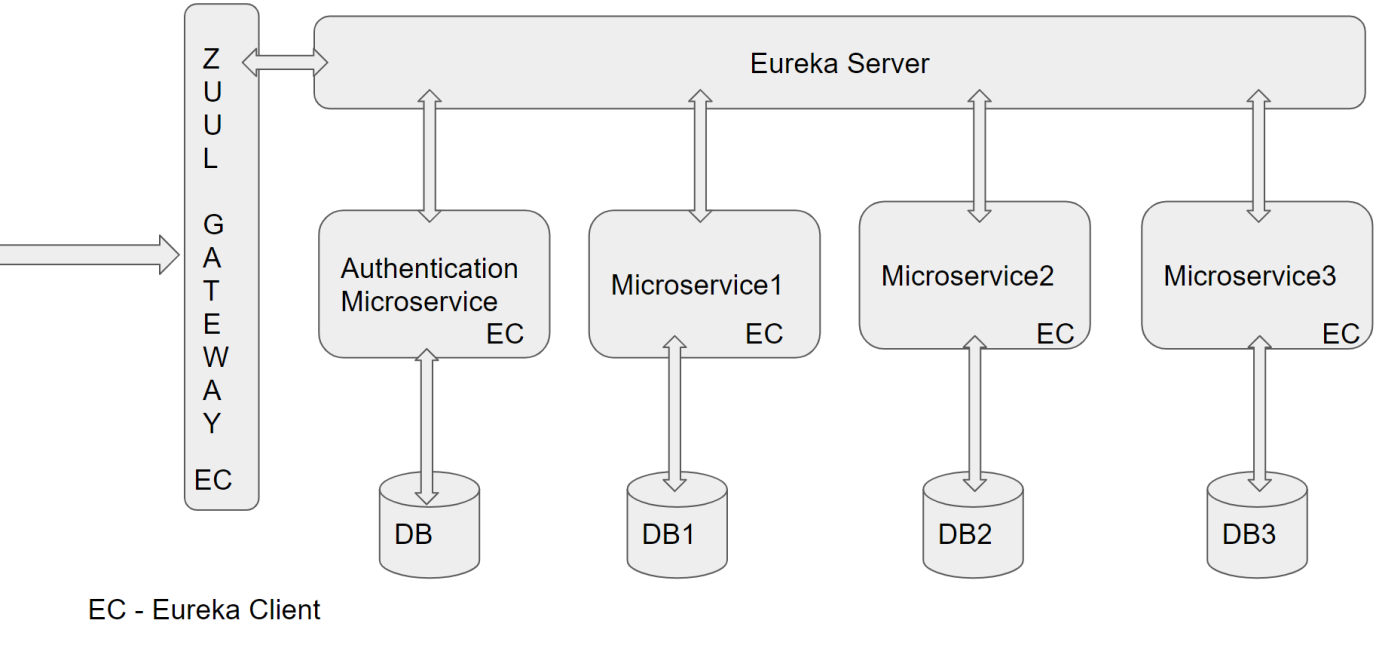
* Zuul API Gateway
* Eureka Service Registry & Discovery
* Ribbon Client side Load Balancer(optional)
* Feign Client
* Hystrix Circuit Breaker & Fault Tolerant Tool(optional)

# JWT Authentication

Create additional Microservice which takes care of authentication and role activities, and JWT Token validation. Spring Security need to be used for Authentication. On successful authentication or token validation the actual request need to be forwarded to the corresponding Microservice. Invoke authentication REST endpoints from Zuul Gateway. Use PreFilter to perform JWT Token validation by invoking REST endpoint of this Microservice.

Instead of JWT, any other security protocol such as OAuth2 can be used. Authentication data can be stored in MySQL DB or LDAP or any other data source.

# Architecture/Design



# 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 |  |

# 

# Assessment Deliverables

1. Check in the projects Zuul Gateway Project, Eureka Server Project, all Microservices with Eureka Client added
2. Screen shots showing sample invocation of Zuul Gateway end points from Post Man Tool.
3. Eureka Dashboard Screenshots with registered instances of Microservices
4. Few Steps on how to run the solution.

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