|  |
| --- |
|  |

|  |
| --- |
| **Pixogram (Mid-Tier Phase 3) 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** |
| **1/10/2019** |
|  |

Table of Contents

[1. Business Requirement(Pixogram) 2](#_Toc28724874)

[NOTE: Features marked as optional are not mandatory for GenCs 3](#_Toc28724875)

[Overview of Fields used in User Registration 3](#_Toc28724876)

[Overview of fields used for Add Content 4](#_Toc28724877)

[2. Design Inputs 5](#_Toc28724878)

[3. Development of individual Microservices 5](#_Toc28724879)

[4. Architecture Diagram 6](#_Toc28724880)

[5. Full Stack Technologies 7](#_Toc28724881)

[6. Database Tables 8](#_Toc28724882)

[7. Technical Spec – Solution Development Environment 8](#_Toc28724883)

[7.1. Front End Layer 8](#_Toc28724884)

[7.2. Middle Tier Layer 8](#_Toc28724885)

[7.3. Database & Integration Layer 9](#_Toc28724886)

[7.4. Ancillary Layer 9](#_Toc28724887)

[7.5. Security 9](#_Toc28724888)

[7.6. Deployment & Infrastructure 9](#_Toc28724889)

[7.7. Editors 9](#_Toc28724890)

[8. Assessment Deliverables 9](#_Toc28724891)

[9. Important Instructions 10](#_Toc28724892)

# 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.

# Development of individual Microservices

This specific Phase is to design/develop individual Microservices. Analyze the requirement and divide back end functionality into multiple Microservices. Based on the Pixogram requirements, below can be possible Microservices

1. Media Microservice:

upload Single Media,

upload multiple Media,

my Media,

update Media

hide/show Media

search my Media,

add comment,

get comments,

1. Miscellaneous Microservice:

postMedia

get newsfeed

blockUser

follow/unfollow

get followers, etc...

1. User Microservice:

login

signup

user details update,

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

1. Each Microservice is a Spring Boot Rest application by specifying required spring boot starter packages in pom.xml os by using Spring Initializr
2. REST Controllers, with the appropriate REST End points to perform corresponding CRUD operations. Along with End Points which are exposed to Angular Client, you may need additional End point(s) for interaction between Microservices
3. As known, each Microservice is a self-sufficient and standalone application, and owns data stored in specific DB tables or databases.
4. Services – Service Layer
5. Entity & Model classes, including appropriate relationship (like One-One, Many-One, etc…) between Entity Classes, if required. (Entity and Model classes have been developed in the Previous Phases)
6. 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.
7. Microservice interaction with corresponding DB tables or Databases it owns.
8. It is possible that one Microservice need to interact with other Microservice(using RestTemplate or FeignClient)
9. Repository class which implements JPA or CrudRepository, if RDBMS is used
10. Usage of Custom Queries in JPA or CrudRepository using @Query where ever custom functionality required
11. Feign Client can be used to invoke one Microservice, from another Microservice
12. Use Postman to test the Microservices by directly passing requests to each REST end Point, of each Microservice
13. 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, for actual fields refer corresponding Entity classes. 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 |
| User | stores User related details |
| Media | stores Media related data(including Media details like Title, caption, hidden, etc…), owned by each User |
| Newsfeed | stores all the posts made by the Users |
| Followers | stores followers details of each User |
| Comments | stores comments of the posted Media Items |
| BlockedAccounts | stores list of blocked accounts of each User |

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 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