|  |
| --- |
|  |

|  |
| --- |
| **Pixogram (Mid-Tier Java Phase) 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](#_Toc28724619)

[NOTE: Features marked as optional are not mandatory for GenCs 3](#_Toc28724620)

[Overview of Fields used in User Registration 3](#_Toc28724621)

[Overview of fields used for Add Content 4](#_Toc28724622)

[2. Design Inputs 5](#_Toc28724623)

[3. Entity Classes 5](#_Toc28724624)

[4. Model Classes 7](#_Toc28724625)

[JSON Structure for Single Media File Upload 7](#_Toc28724626)

[JSON Structure for Multiple Media File Upload 8](#_Toc28724627)

[5. Other Core Java Implementation 9](#_Toc28724628)

[6. Architecture Diagram(just for reference) 9](#_Toc28724629)

[7. Full Stack Technologies 10](#_Toc28724630)

[8. Technical Spec – Solution Development Environment 11](#_Toc28724631)

[8.1. Front End Layer 11](#_Toc28724632)

[8.2. Middle Tier Layer 11](#_Toc28724633)

[8.3. Database & Integration Layer 11](#_Toc28724634)

[8.4. Ancillary Layer 11](#_Toc28724635)

[Controllers can be tested using Postman Tool 11](#_Toc28724636)

[8.5. Security 11](#_Toc28724637)

[8.6. Deployment & Infrastructure 11](#_Toc28724638)

[8.7. Editors 12](#_Toc28724639)

[9. Assessment Deliverables 12](#_Toc28724640)

[10. Important Instructions 12](#_Toc28724641)

# 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.

# Entity Classes

Below are the activities which need to be performed as part of this

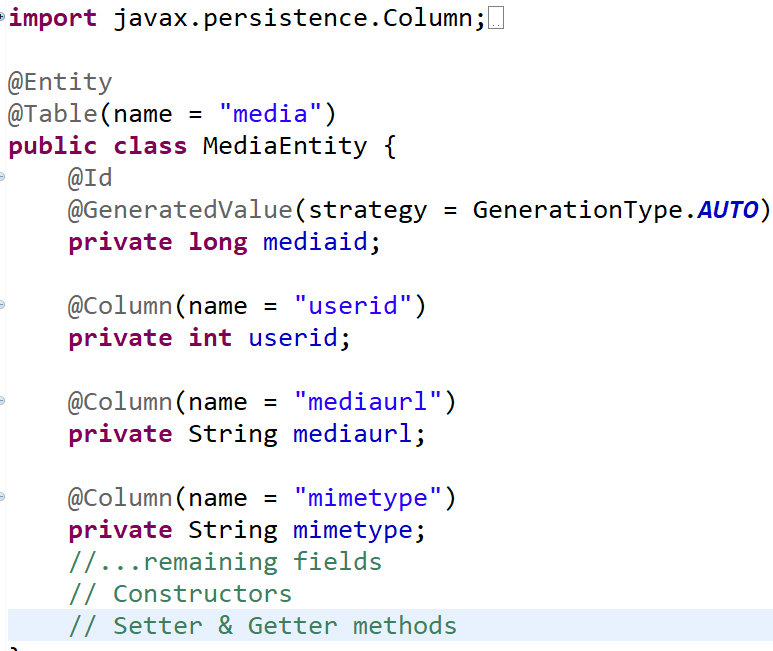
1. Identify all Entity Classes and its fields. An Entity class is the one which is mapped to underlying DB Table
2. Identify relationship(such as One to One, One to Many, Many to One, Many to Many) between Entity classes, if required
3. Develop the source code of Entity classes

Below are sample Entity Classes

**MediaEntity Class:** Indicates a Media Item which belongs to a User. Below can be fields of MediaEntity Class

1. Userid
2. MediaId
3. MediaURL
4. MimeType
5. MediaTitle
6. MediaCaption
7. UploadedDateTime
8. Hide

Snapshot of Entity class below



**CommentsEntity Class:** Indicates comments on a specific NewsFeed post by various Users. Below are the fields

1. NewsFeedId
2. Comment
3. UserId
4. DateTime

**FollowersEntity Class:** Indicates Users whom an User follows. Below are the fields

1. UserId
2. FollowsUserId

**UsersEntity Class:** Indicates User profile and login details

1. UserId
2. UserName
3. Password
4. Confirmed
5. ProfilePictureURL
6. CreatedDateTime

**NewsFeedEntity class:** Indicates NewsFeeds posted by various Users

1. MediaId
2. UserId
3. PostedDateTime

**BlockedAccountsEntity class:** Indicates all other User Accounts blocked by an User

1. userId
2. BlockedUserId

# Model Classes

Model Classes are the classes which are required to transfer the data between

1. REST APIs and Angular Client,
2. REST Controller and Service Layer
3. Service Layer and Repository Layer

As part of this Phase identify all Model classes, and develop source code for the same.

Model classes are just normal POJO classes with data members, constructors, setter/getter methods

Below sample JSON structures might be useful to come up with some of the Model classes

### JSON Structure for Single Media File Upload

1. The following is the structure of the JSON object for single media upload component:
   1. {
   2. "id": 1,
   3. "title": "Full Stack Freelancer",
   4. "type": "video",
   5. "videoposter": "poster.jpeg",
   6. "description": "It is great to be a full stack developer!",
   7. "tags": [
   8. {
   9. "id": 1,
   10. "tag": "fsd"
   11. },
   12. {
   13. "id": 2,
   14. "tag": "freelancer"
   15. },
   16. {
   17. "id": 3,
   18. "tag": "full stack"
   19. },
   20. {
   21. "id": 4,
   22. "tag": "full stack cognizant"
   23. }
   24. ],
   25. "effect": "greyscale",
   26. "filename": "freelancer\_poster.jpeg",
   27. "filetype": "image/jpeg",
   28. "filesize": "541144",
   29. "uploaddate": "31-08-2018",
   30. "uploadtime": "1331",
   31. "defaultprofile": 0,
   32. "likes": 0,
   33. "unlike": 0,
   34. "shares": 0,
   35. "numberofcomments": 0
   36. }
   37. *//type can be 'video' or 'image'*
   38. *//in case of “image”, the value of “videoposter” is “”*
   39. *//in case of “video”, the value of “defaultprofile” is 0*
2. You may change the JSON object structure as per your programming needs.

### JSON Structure for Multiple Media File Upload

1. The following is the structure of the JSON object for multiple media upload component:
2. [
3. {
4. "id":1,
5. "title":"Full Stack Freelancer",
6. …
7. …
8. "uploadtime":"1331"
9. },
10. {
11. "id":2,
12. "title":"Technology Solutions",
13. …
14. …
15. "uploadtime":"1313"
16. },
17. {
18. "id":3,
19. "title":"Development Stack",
20. …
21. …
22. "uploadtime":"1111"
23. }
24. ]

You may change the JSON object structure as per your programming needs.

# Other Core Java Implementation

This Phase also comprises development of other Core Java source code required for the Project.

# Architecture Diagram(just for reference)

Below Microservice Architecture would be required in next Phases



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>

# 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. Source code of Compiled Entity Classes
2. Source code of Compiled Model Classes

# 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