PIXOGRAM (FSD) v3.0

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

Table of Contents

1. Business Requirement (Pixogram)

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

1. Register as a user
2. Login as a user
3. Retrieve password
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
7. Manage Content
    a. Organize Picture in Gallery
    b. Organize Videos in Playlists
    c. Rename Pictures and Videos
    d. Edit Caption, Description, Comment
8. Social Features
    a. Use emojis in comment
    b. Like or Unlike comment, pictures and videos of other users
    c. Follow/Unfollow other users
9. Hide Pictures/Videos
10. Activity/Newsfeed
    a. View activity log of user-activity on the PixoGram
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:
    a. To implement offline image upload functionality such that user can upload content when offline. It will sync with backend when connected.

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 & 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 & 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.

2. 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.

3. UI/UX Wireframes

Angular Components

1. As per the navigation bar (each is independent page). Each page can be thought of as independent component with few child components where required:
    a. Upload Media Component
        1. -> Single Media Upload Component
        2. -> Multiple Media Upload Component
    b. My Media Component
        1. -> Media Detail Component
    c. Followers/Following Component
        1. -> Follower Page -> Follower Media Detail Component
        2. -> You Follow User Page -> You Follow Media Detail Component
    d. Account Details Component
        1. Sign In Component
            a. Blocked Accounts Component
            b. Newsfeed Component
            c. Account Update Component
            d. Search Component
            e. Logout Component
        2. Register Component

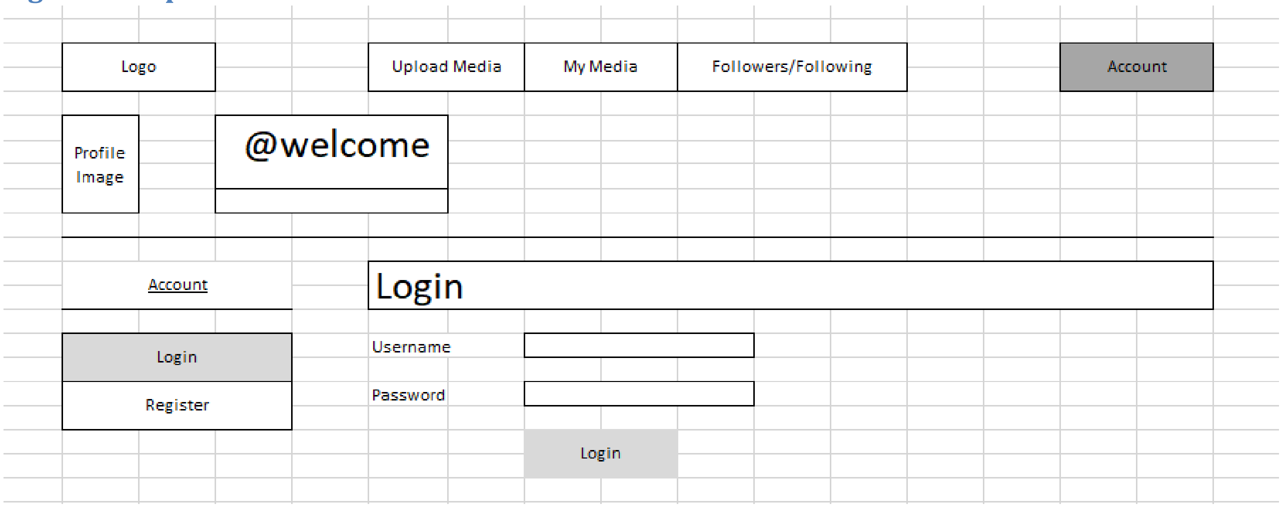
Components

Sign In Components

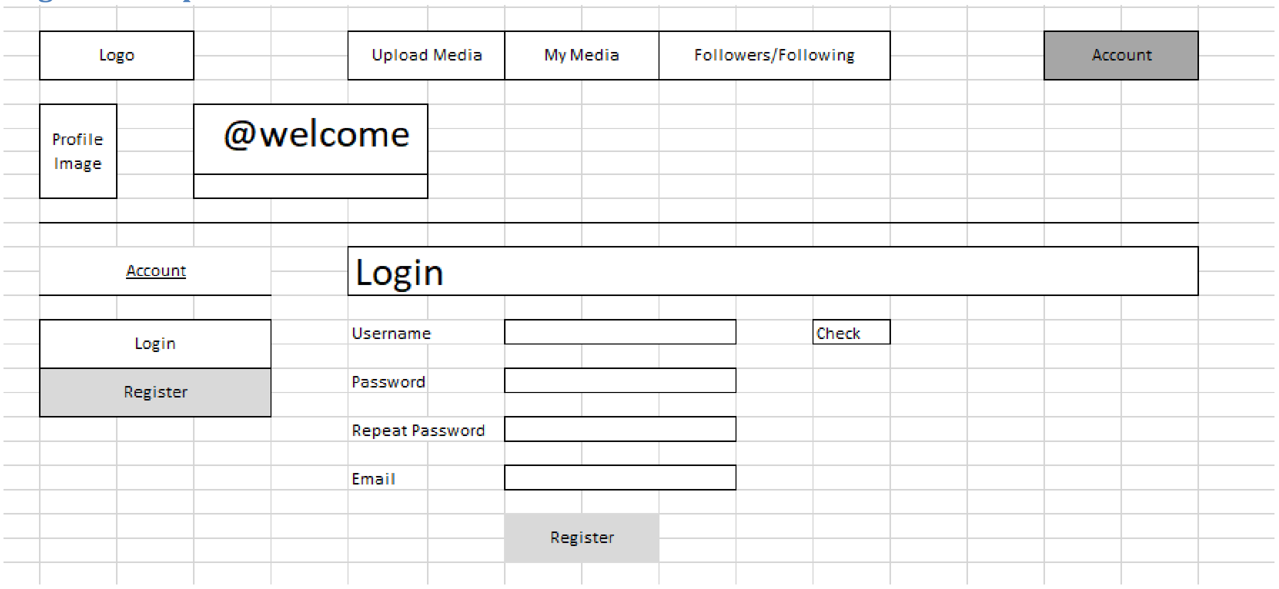
Sign In Component Requirement

1. It allows user to sign-in with registered credentials.
2. If the user is not registered, user may register before signing-in.
    a. Username.
    b. Password
    c. Email
3. Clicking on any link: Upload Media, My Media, Followers/Following will redirect users to Login component.
4. On register component, there is check button to check if username is already in use.

Sign-In Component Wireframe



Register Component Wireframe

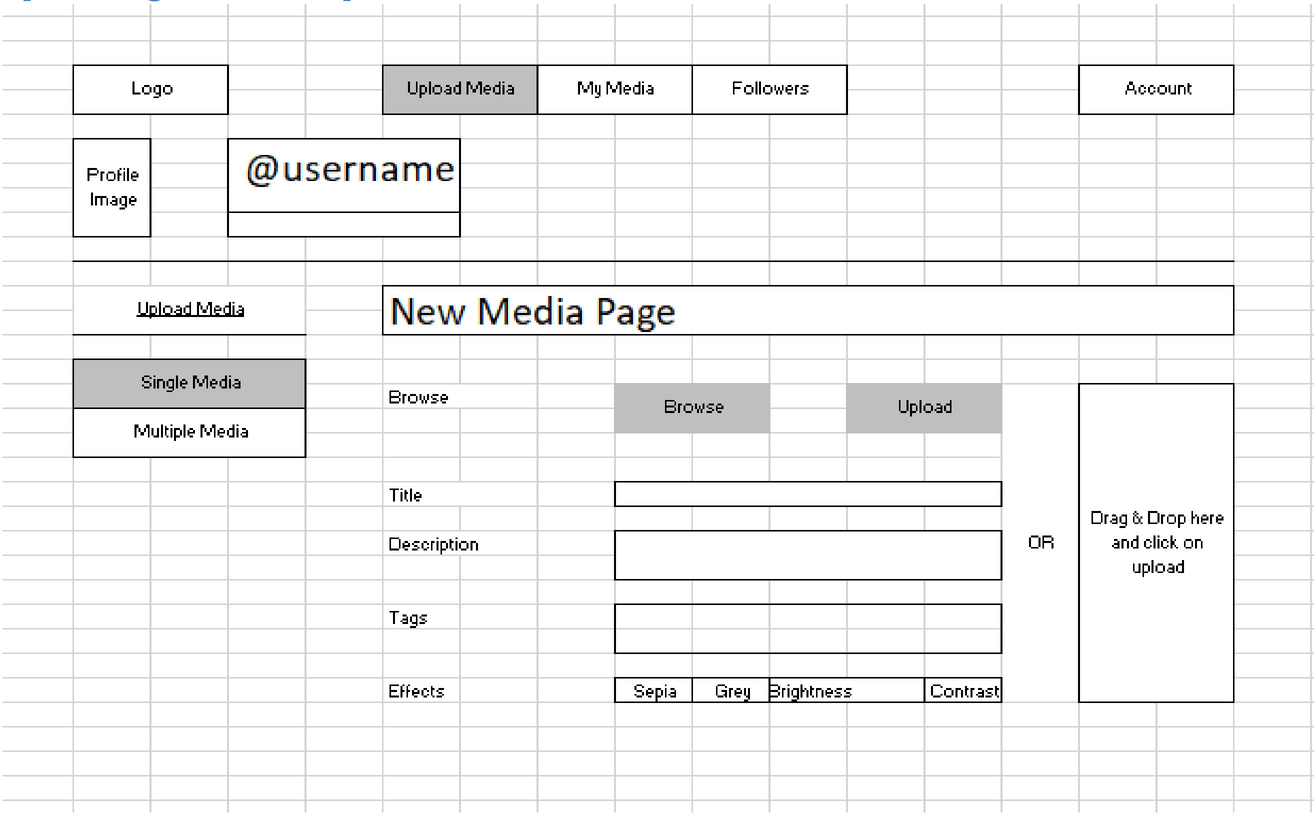


Upload Media Component

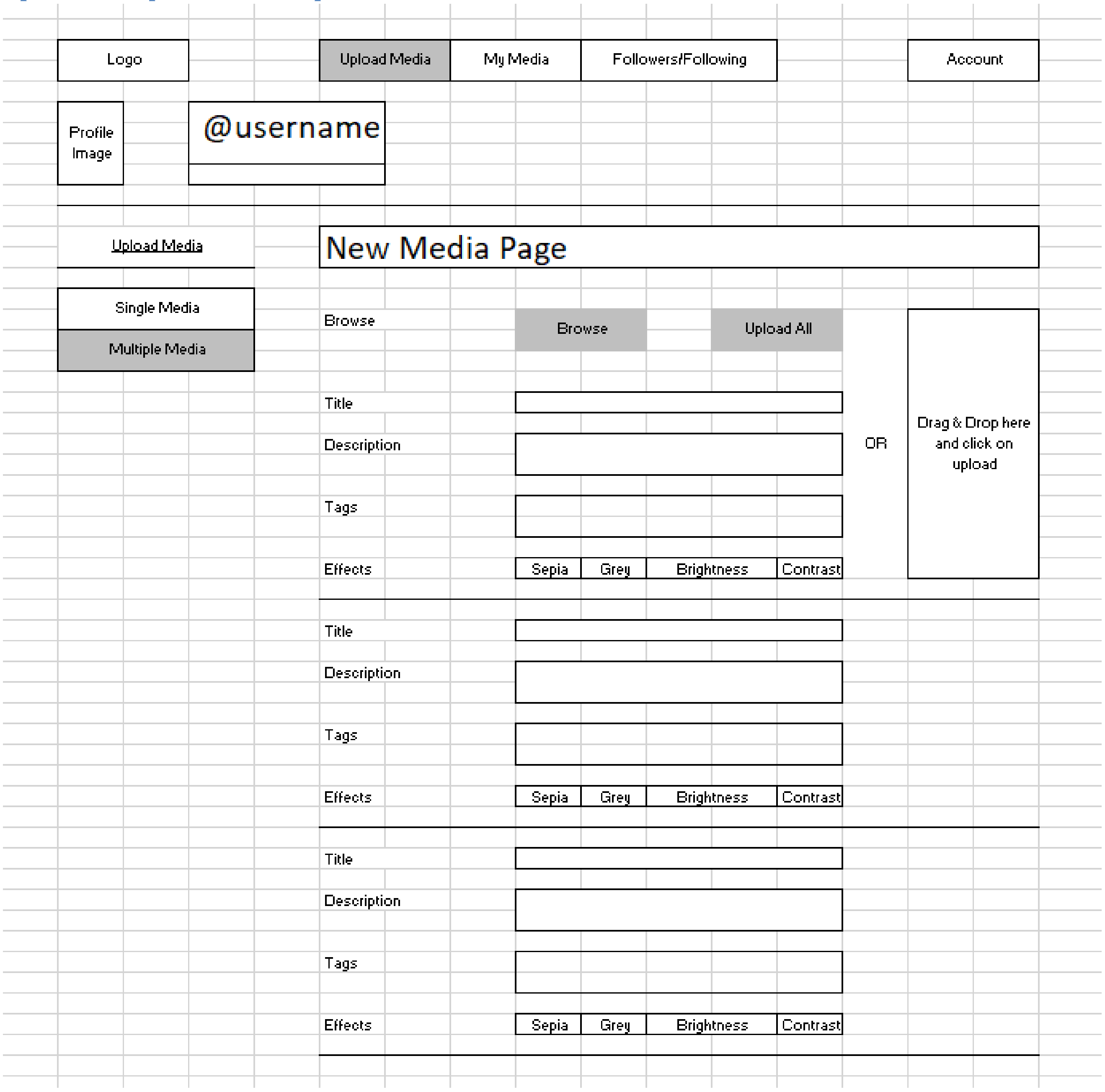
Upload Media Component Requirement

1. It will have two sub-components
    a. Single Media Upload Component
    b. Multiple Media Upload Component
2. It allows you to upload media in two formats
    a. Images - png, jpeg, gif
    b. Video – wmv, avi, mp4
3. User should be able to upload single/multiple media items using drag and drop from file explorer in the host operating system. It is recommended that you should first create the component for single media upload. Once it is done and approved, then create the component for multiple media upload.
4. The first image which you upload will be used as a default profile picture for your account.
5. In case of video being uploaded, default image should be used as a poster/thumbnail.
6. Each upload item should have following three fields:
    a. Title
    b. Description
    c. Tags
    d. Effects – sepia, greyscale, brightness, contrast etc.
        i. Should be disabled initially. Enabled only after the media is uploaded and saved.
7. User should be able to add multiple tags; each separated by comma (,)
8. User should be able to save the uploaded media item/s
9. You will post the data to json-server using Angular httpClient library.

Upload Single Media Component Wireframe



Upload Multiple Media Component Wireframe



JSON Structure for Single Media File Upload

1. The following is the structure of the JSON object for single media upload component:

```
{
"id": 1,
"title": "Full Stack Freelancer",
"type": "video",
"videoposter": "poster.jpeg",
"description": "It is great to be a full stack developer!",
"tags": [
{
"id": 1,
"tag": "fsd"
},
{
"id": 2,
"tag": "freelancer"
},
{
"id": 3,
"tag": "full stack"
},
{
"id": 4,
"tag": "full stack cognizant"
}
],
"effect": "greyscale",
"filename": "freelancer_poster.jpeg",
"filetype": "image/jpeg",
"filesize": "541144",
"uploaddate": "31-08-2018",
"uploadtime": "1331",
"defaultprofile": 0,
"likes": 0,
"unlike": 0,
"shares": 0,
"numberofcomments": 0
}
//type can be 'video' or 'image'
//in case of “image”, the value of “videoposter” is “”
//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:

```
[
{
"id":1,
"title":"Full Stack Freelancer",
…
…
"uploadtime":"1331"
},
{
"id":2,
"title":"Technology Solutions",
…
…
"uploadtime":"1313"
},
{
"id":3,
"title":"Development Stack",
…
…
"uploadtime":"1111"
}
]
```

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

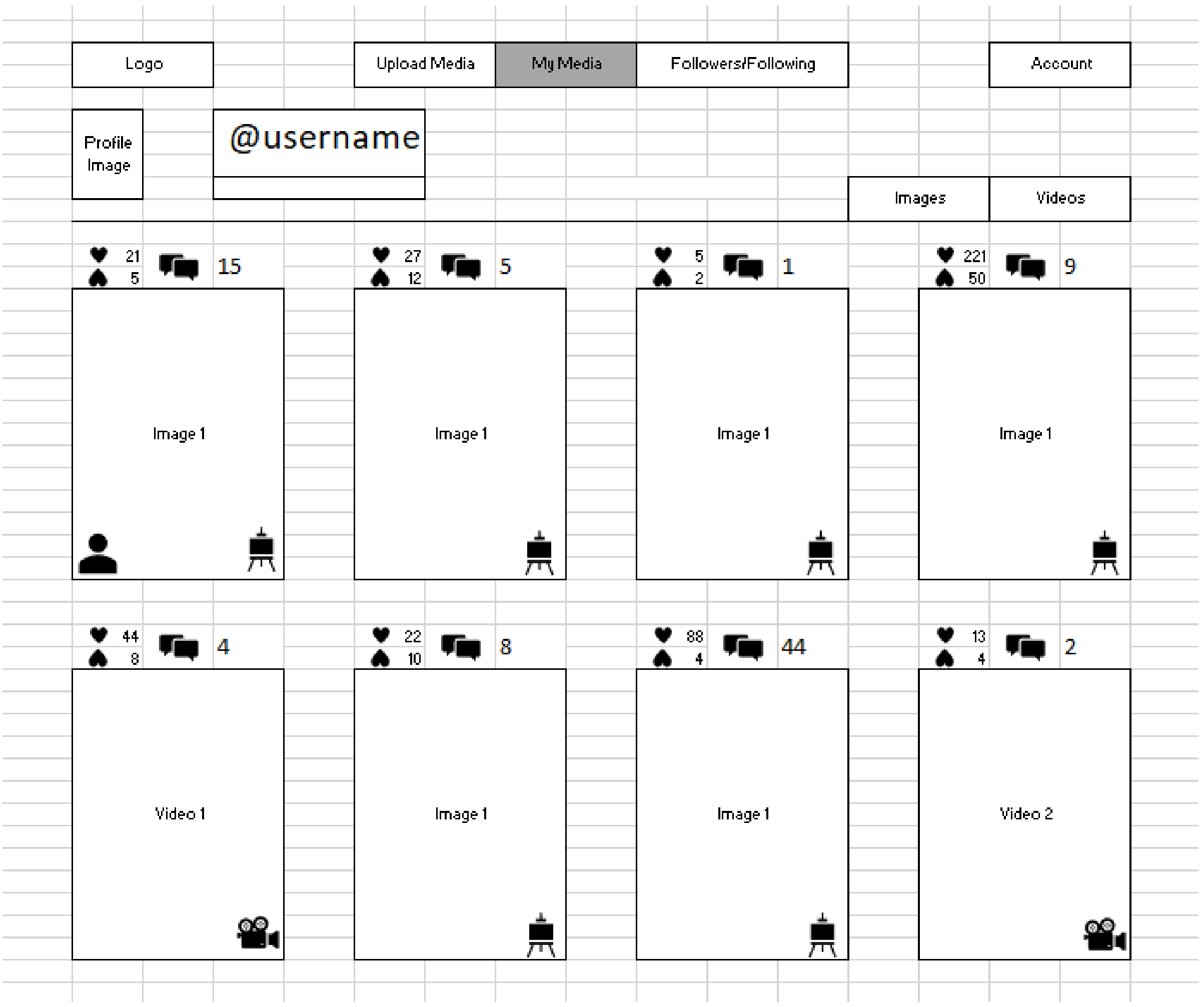
My Media component

My Media Component Requirement

1. This component contains all the media uploaded by you along with other information.
2. It will display your username on top along with (Follow/Unfollow) toggle button. Any user and click on Follow/Unfollow button to follow or unfollow you. It will be disabled for you as you are the account owner.
3. It will display all media items uploaded by you, as a user, in a grid format.
4. It will contain two more toggle button i.e. Images, Videos
5. If “Images” is activated, then only images are displayed.
6. If “Videos” is activated, then only videos are displayed.
7. By default, both are activated.
8. Each media item will be displayed in one cell of responsive grid with following information:
    a. Emoji Icon + number of like. (not clickable)
    b. Emoji Icon + number of unlike. (not clickable)
    c. Emoji Icon + number of comments.
    d. Emoji Icon to specify whether it is used for default profile picture.

   User should be able to click on the media (image/video) thumbnail to view further media details and interact with the media.

My Media Component Wireframe

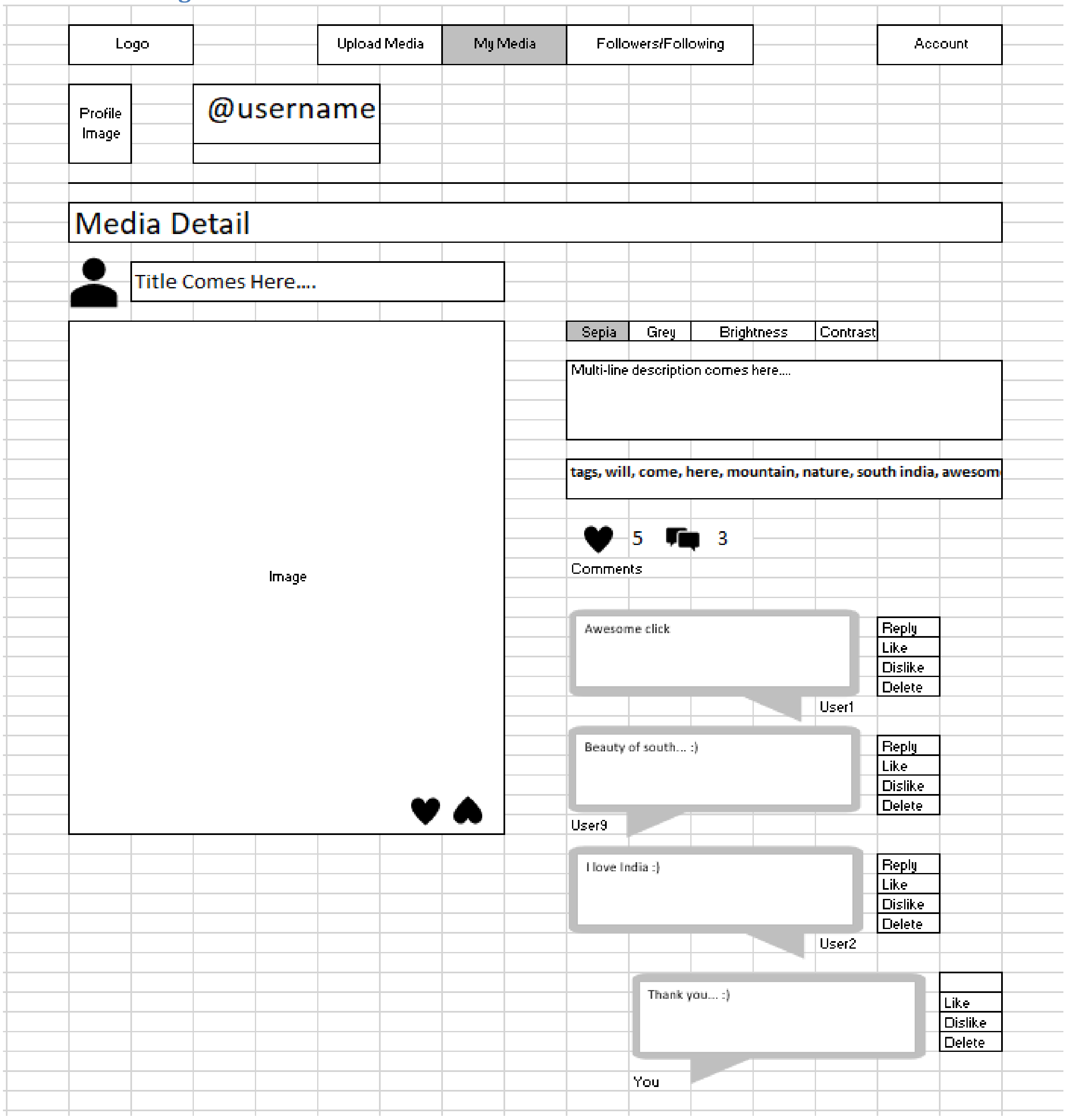


Media Detail Component

Media Detail Component Requirement

1. It will display your username on top along with (Follow/Unfollow) toggle button
2. If Image:
    a. Original dimension.
    b. Name of effect applied.
    c. “Make Profile Picture” button, clicking on which will make it a default profile picture for your account. This button is disabled when you are browsing the collection of any other user.
3. If Video:
    a. HTML5 video player
        i. default play/pause/volume button.
        ii. video player should also have custom playback progress bar.
        iii. Full screen feature
        iv. Mute/unmute feature
        v. Replay feature
        vi. Loop feature
4. Media title
5. Emoji Icon + number of like. (clickable only once)
6. Emoji Icon + number of unlike. (clickable only once)
7. Emoji Icon + number of comments.
8. Emoji Icon to specify whether it is used for default profile picture.
9. List of comments.
10. Name (hyperlink) of the user who made the comment in front of each comment
11. Link to reply to any comment which will open reply text field.
12. Text field to add new comment to your own post.

Media Detail Page Wireframe

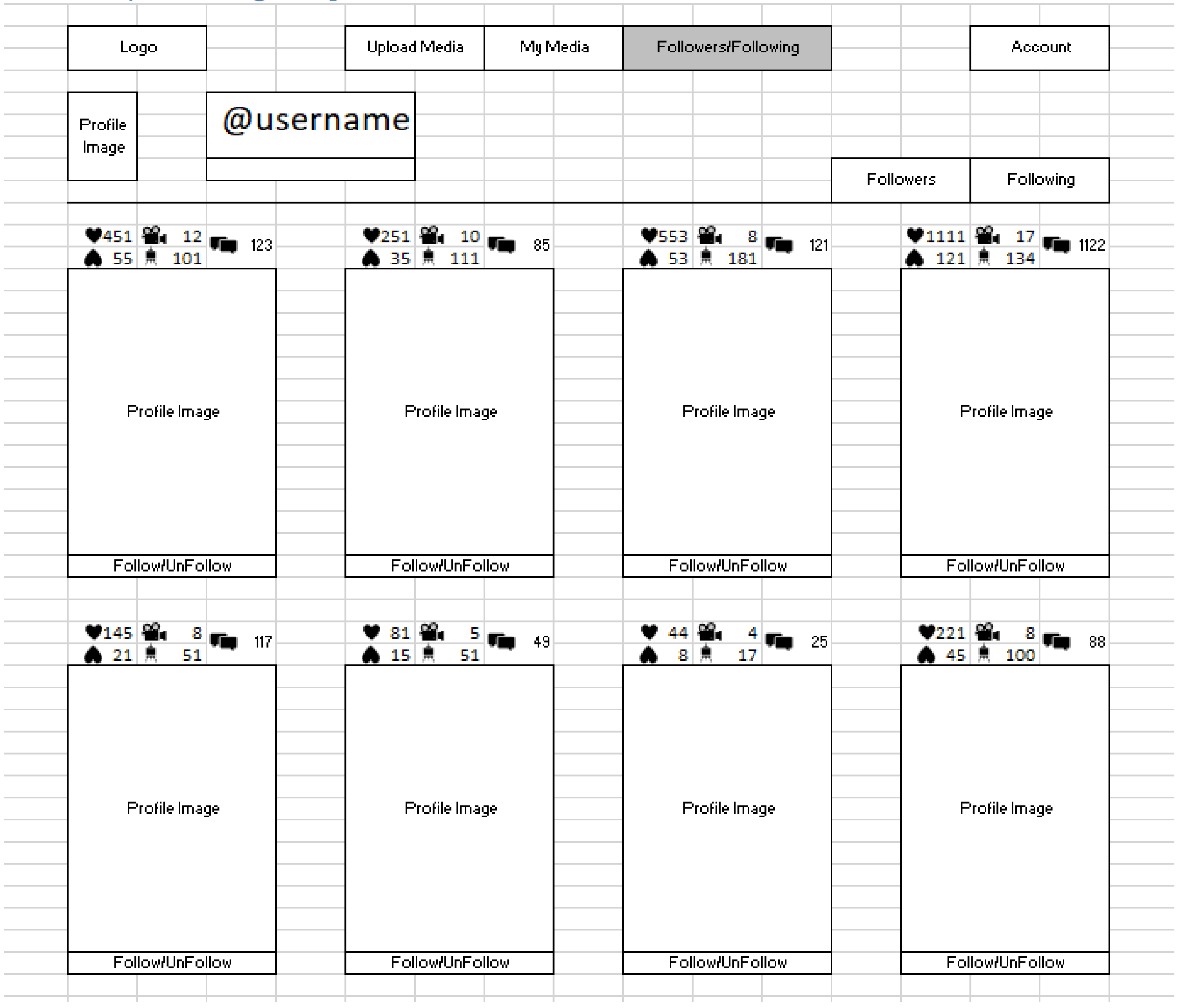


Followers/Following Component

Followers/Following Component Requirement

1. Will display the all the followers of your PixoGram account and the users you are following.
2. It will display all followers/following in grid view.
3. Each follower/following profile picture will display below information:
    a. Emoji Icon + total number of like. (not clickable)
    b. Emoji Icon + total number of unlike. (not clickable)
    c. Emoji Icon + total number of comments.
4. User may decide the profiles to be displayed on the page by clicking the buttons on top right:
    a. Followers Button – Will display all followers of your pixogram account
    b. Following Button – Will display all accounts you are following
    c. By default, both buttons are enabled.
5. User can click on any user profile picture and navigate to the “My Media Component” of respective user.
6. Once on the “My Media Component” of the respective user, you can click on any media item to navigate to the respective “Media Detail Component” page.
7. Once on media detail component of the respective user for respective media:
    a. It will display username on top along with (Follow/Unfollow) toggle button
    b. If Image:
        i. Original dimension.
        ii. Name of effect applied.
        iii. “Make Profile Picture” button is disabled as this media does not belong to your account.
    c. If Video:
        i. HTML5 video player
            1. Default play/pause/volume button.
            2. video player should also have custom playback progress bar.
            3. Full screen feature
            4. Mute/unmute feature
            5. Replay feature
            6. Loop feature
    d. Media title
    e. Emoji Icon + number of like. (clickable only once)
    f. Emoji Icon + number of unlike. (clickable only once)
    g. Emoji Icon + number of comments.
    h. Emoji Icon to specify whether it is used for default profile picture.
    i. List of comments
    j. Name (hyperlink) of the user who made the comment in front of each comment
    k. Link to reply to any comment which will open reply text field.
    l. Text field to add new comment to respective user’s post.

Followers/Following Component Wireframe



Account Component

It will consist of 5 sub-component

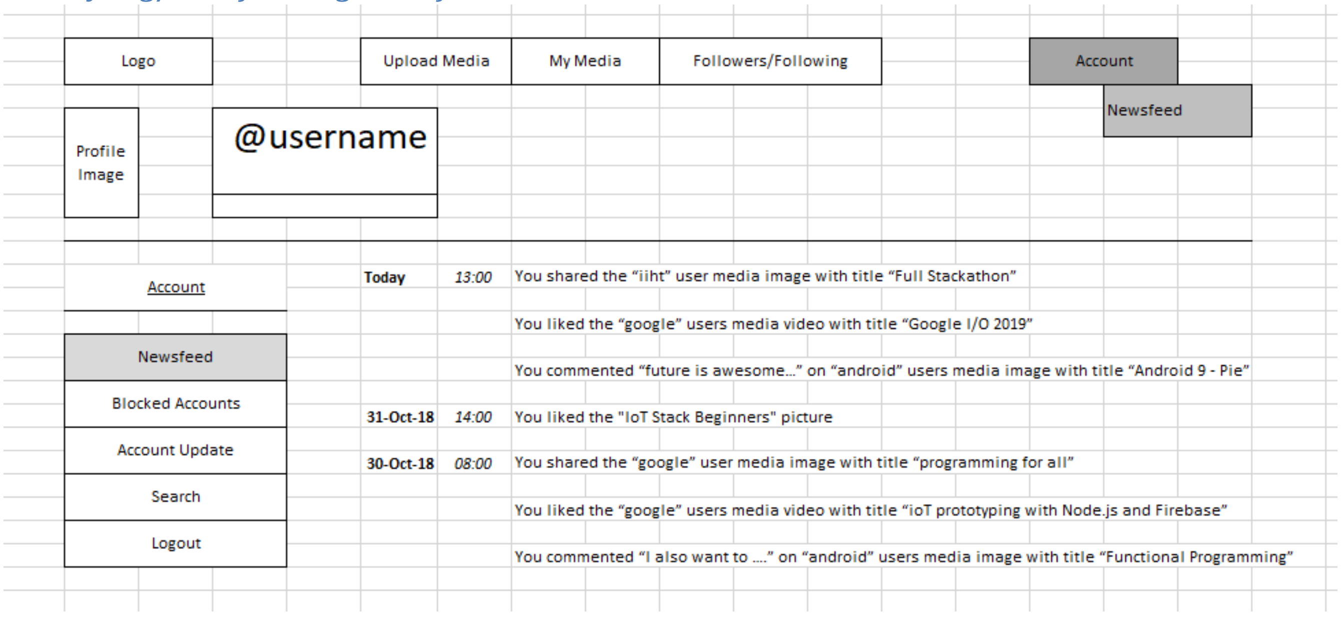
m. Account Details sub-component
n. Activity Log/Newsfeed sub-component
o. Blocked Users sub-component
p. Search sub-component
q. Logout sub-component

Activity Log/Newsfeed Component

Activity Log/Newsfeed Component Requirement

1. Will display the log of all the activity user does on the “PixoGram” app till date.
    a. E.g.
        i. You shared the “iiht” user media image with title “Full Stackathon”
        ii. You liked the “google” users media video with title “Google I/O 2019”
        iii. You commented “future is awesome…” on “android” users media image with title “Android 9 - Pie”

Activity Log/Newsfeed Page Wireframe



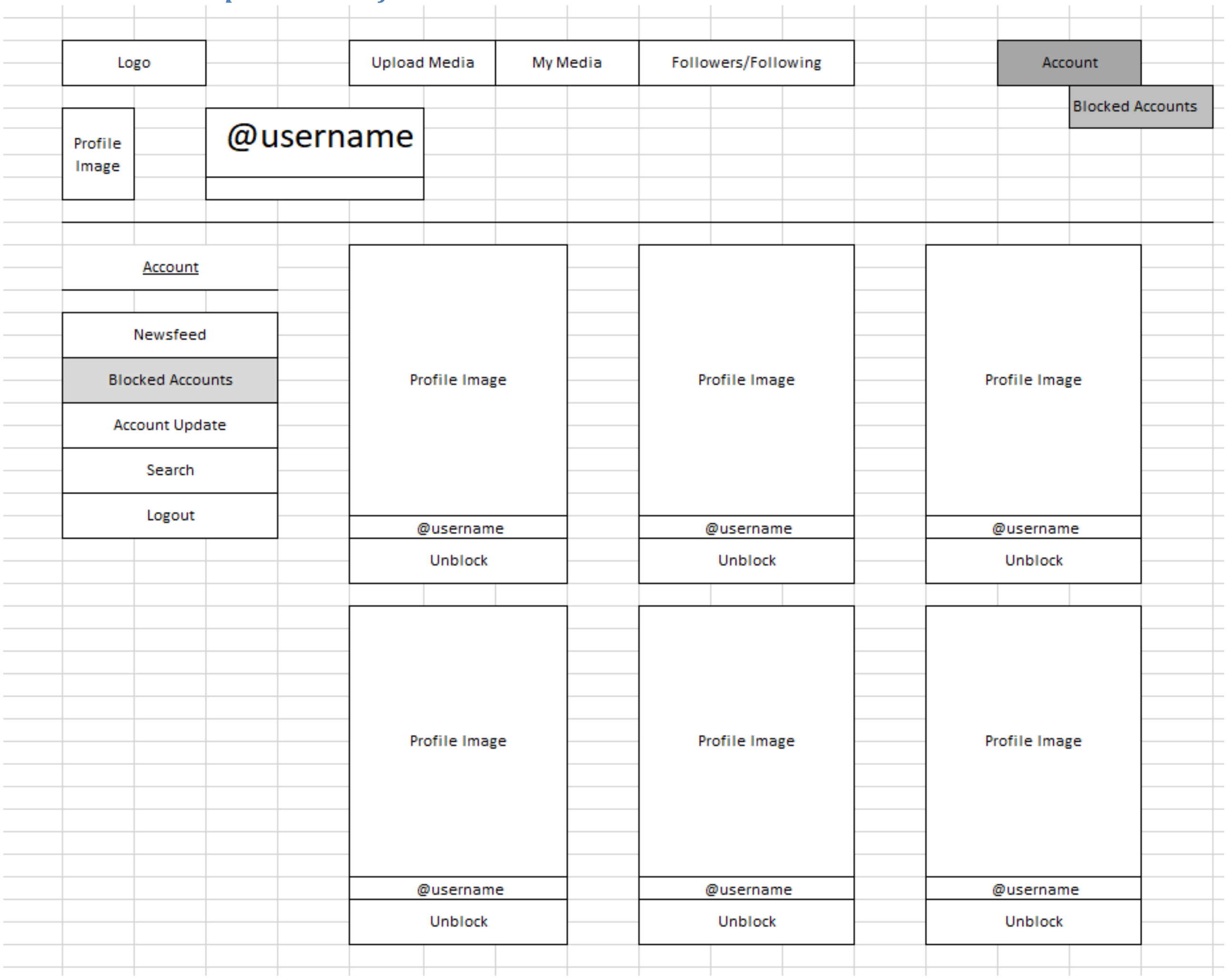
Today	13:00	You shared the “iiht” user media image with title “Full Stackathon”
		You liked the “google” users media video with title “Google I/O 2019”
		You commented “future is awesome…” on “android” users media image with title “Android 9 - Pie”
31-Oct-18	14:00	You liked the "IoT Stack Beginners" picture
30-Oct-18	08:00	You shared the “google” user media image with title “programming for all”
		You liked the “google” users media video with title “ioT prototyping with Node.js and Firebase”
		You commented “I also want to ….” on “android” users media image with title “Functional Programming”

Blocked Users Component

Blocked Users Component Requirement

2. It displays the profile of the accounts who are blocked by you.
3. Blocked accounts cannot view your account on PixoGram.

Blocked User Component Wireframe

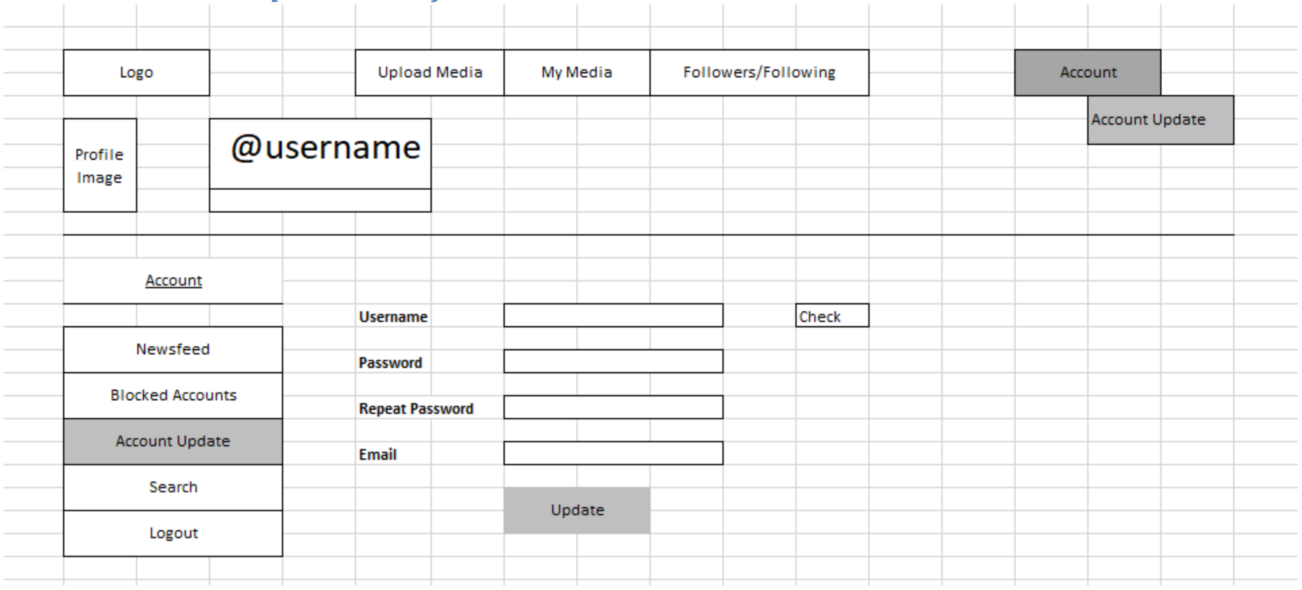


Account Details Component

Account Details Component Requirement

1. It allows you to change the username. Before changing, you need to check if the username is available.
2. You can update email and password.
3. Password validation will follow the same rule as that of password in user registration module.

Account Details Component Wireframe

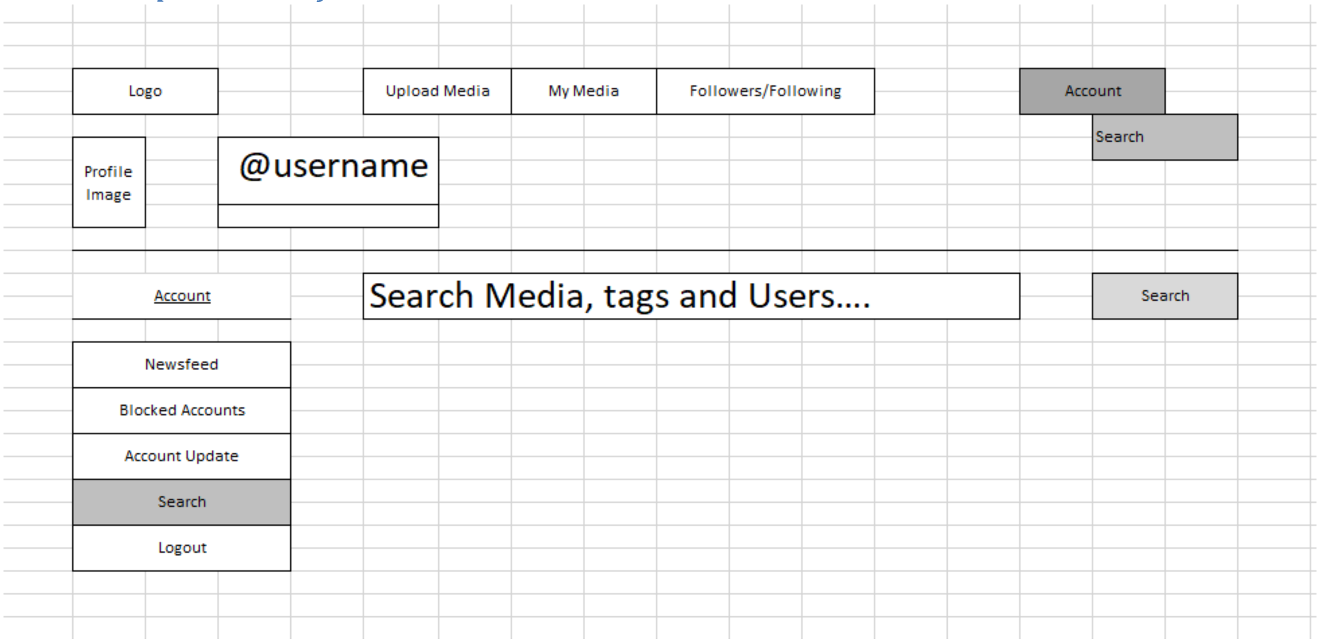


Search Component

Search Component Requirement

1. User should be able to search content via tags, media title, media description and usernames

Search Component Wireframe



4. Entity Classes – Mid Tier

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

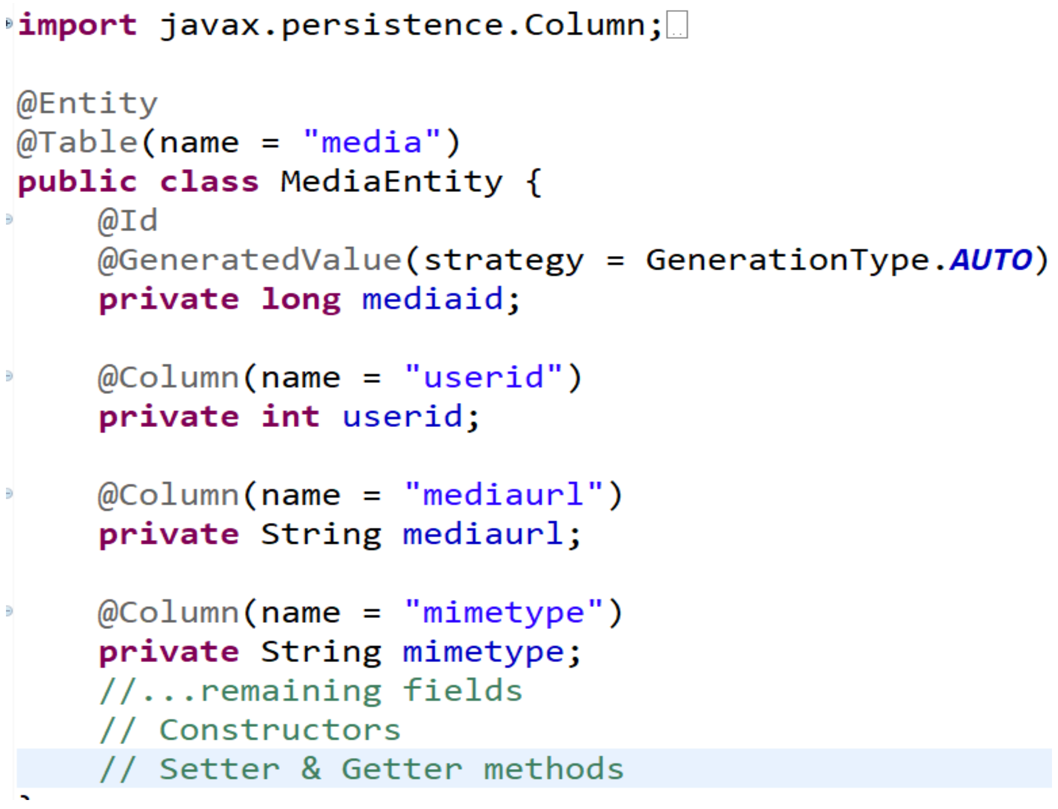
1. Identify all Entity Classes. An Entity class is the one which is mapped to underlying DB Table
2. Identify relationship(such as One to One, One to Many, Many to One, Many to Many) between Entity classes, if required
3. Develop the source code of Entity classes

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



```
import javax.persistence.Column;

@Entity
@Table(name = "media")
public class MediaEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long mediaid;

    @Column(name = "userid")
    private int userid;

    @Column(name = "mediaurl")
    private String mediaurl;

    @Column(name = "mimetype")
    private String mimetype;
    //...remaining fields
    // Constructors
    // Setter & Getter methods
```

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

5. Model Classes

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

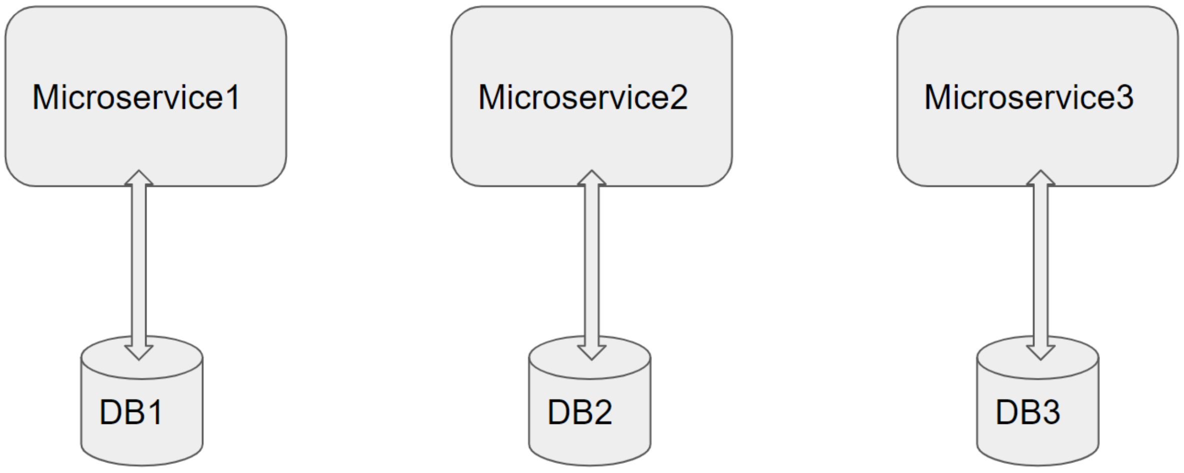
1. REST APIs and Angular Client,
2. REST Controller and Service Layer
3. Service Layer and Repository Layer

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

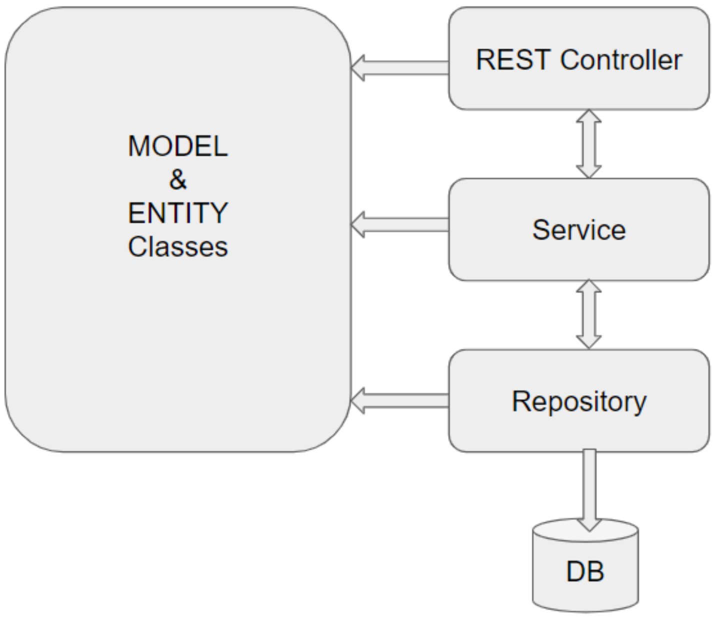
Model classes are just normal POJO classes with data members, constructors, setter/getter methods

6. Architecture Diagram

Class diagram



Architecture of a Single Microservice with REST Controller, Service, Model & Entity Classes and Repository classes



7. 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,
2. Miscellaneous Microservice:
   postMedia
   get newsfeed
   blockUser
   follow/unfollow
   get followers, etc...
3. 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

8. 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.

9. 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

Use PostMan, to test REST end points

---

10. Spring Microservices Tools to be used

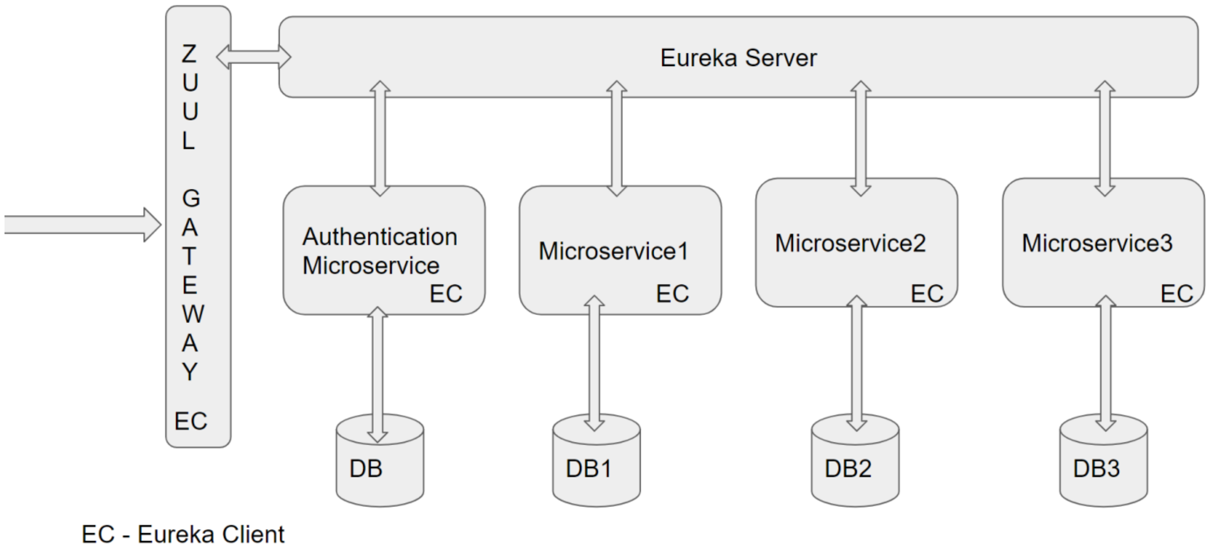
As already specified under Full Stack Technologies Microservice Architecture need to be followed. Ensure that the Application is divided into multiple Microservices, along with database/tables each Microservice Manages. Below Spring Microservices Tools need to be used

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

11. 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.

12. Architecture/Design



13. DevOps Activity

This phase includes performing below Activities

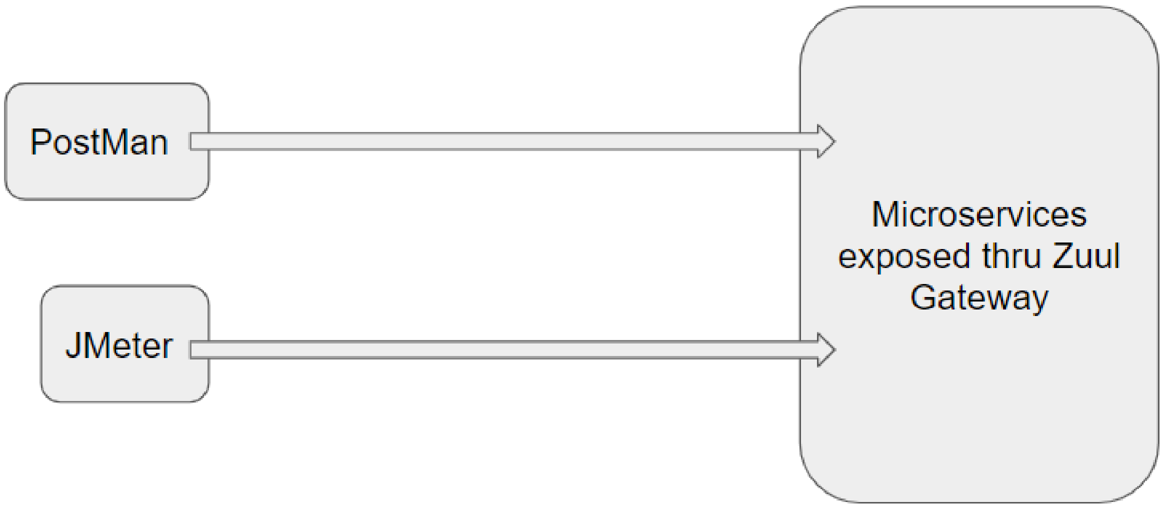
Dockerization: Dockerize atleast Front End or any one Microservice of Mid Tier. Provide Dockerfile and the docker commands used to create image and run Container. Share Screen shots of running Docker.

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

JMeter: As already known JMeter is used to perform Performance or Load Testing. Create a JMeter Test Case, which invokes a REST End point, with multiple threads. Check in jmx file and share the report generated after Performance Testing. Repeat this for atleast two REST end points.

Code Coverage: Code coverage is a Quality Metric to check if sufficient number of Test Cases are created. EclEmma tool can be used as Code Coverage Tool. Code Coverage can be performed on any one Microservice.Ensure that Code Coverage need to be atleast 80%

14. Diagram



15. 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 created and deployed. If possible create a Docker image and run the Container on Docker Host

This Phase also includes completion of Integration of Front end with Mid Tier.

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

16. 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. Pl. ask your mentor for the Internal GIT server URL.
- 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>)

17. Perform CI/CD

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

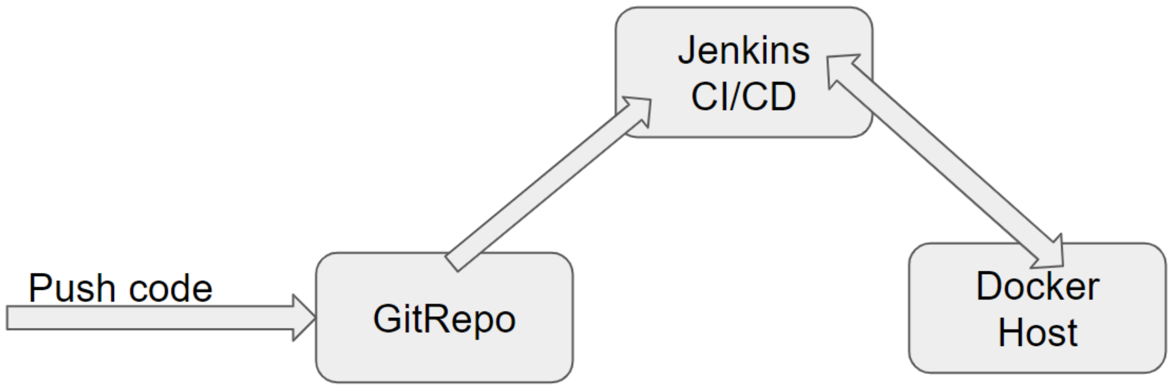
2. Make it sure that project is running locally in development environment without errors.

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

18. Diagram



19. 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

20. Technical Spec – Solution Development Environment

20.1. Front End Layer

Framework(s)/SDK/Libraries	Version
Angular with TypeScript	6 or above
Bootstrap	3.0 or above
CSS	3
HTML	5
JavaScript	1.8 or above
JQuery	1.3

20.2. Middle Tier Layer

Technology	Framework(s)/SDK/Libraries	Version
Java Stack	Spring Boot	2.x
	Spring MVC	4.0 or above
	JDK	1.7 or above
	Maven	3.x or above

20.3. 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

20.4. 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		

20.5. Security

Name	Version
Spring Boot Security	
JWT	

20.6. Deployment & Infrastructure

Technology	Framework(s)/SDK/Libraries	Version
Docker	-	
Apache Tomcat	-	
Jenkins(CI/CD)	-	
Node	-	

20.7. Editors

Name	Version
STS(Spring Tool Suite)	
Visual Studio Code	

Agile/Scrum Software development Model can be used

21. Assessment Deliverables

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

1. FrontEnd Source code
2. Mid Tier Source code of all Microservices
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
8. Jenkinsfile or Jenkins UI ScreenShot
9. URL where the Project is deployed

22. Important Instructions

1. Consider using below Java features
   a. Lambda Expressions
   b. Collection Streams
   c. Generics
2. Sample Design provided is just for reference, Associates can make changes over it or follow their own Design.
3. 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.
4. Please make sure that your code does not have any compilation errors while submitting your case study solution.
5. The final solution should be a zipped code having solution. Solution code will be used to perform Static code evaluation.
6. Implement the code using best design standards/family Design Patterns.
7. Use Internationalization for all the labels and messages in Rest API Development.
8. Do not use System out statements or console.log for logging in Rest API and FrontEnd respectively. Use appropriate logging framework(such as SLF4J) for logging statements.
9. 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.
10. Write web service which takes input and return required details from database.
11. Use JSON format to transfer the results.

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