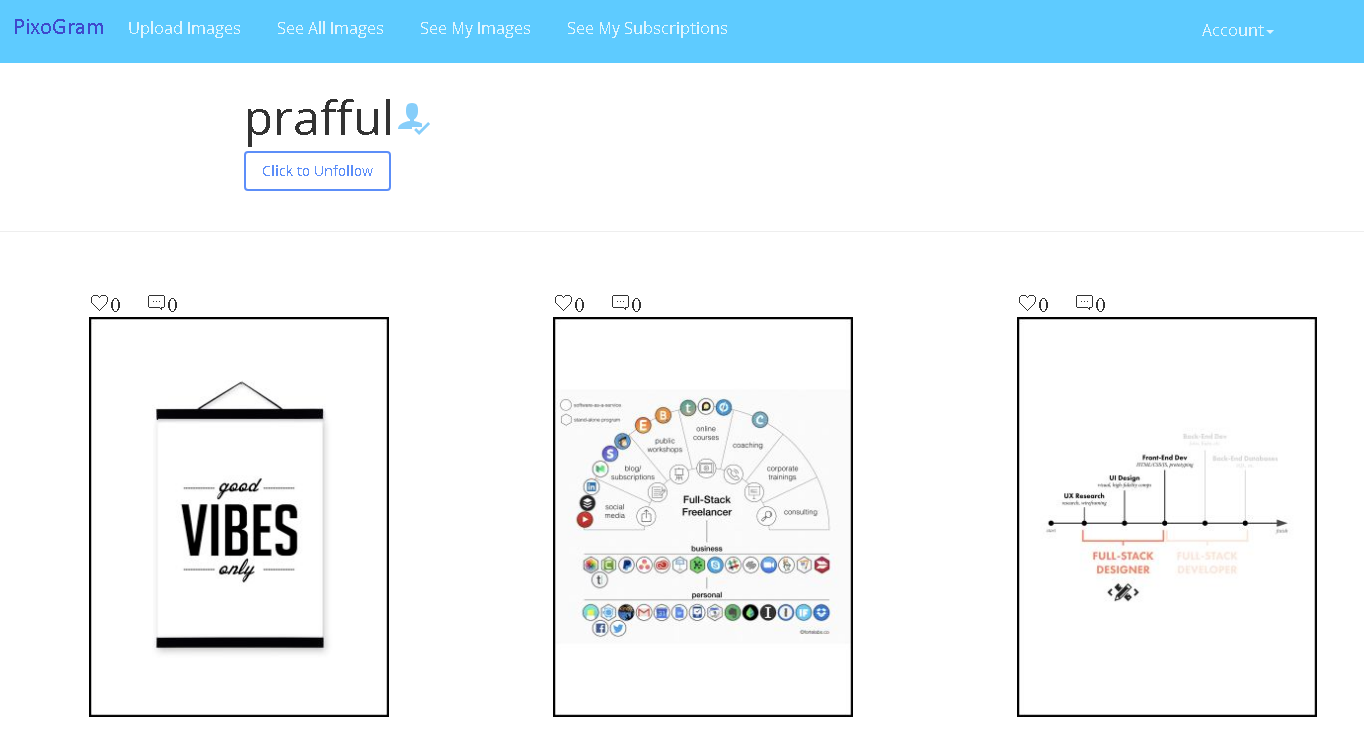
Spring Boot & ORM Microlayer

Prafful Daga  
IIHT

Micro Credential: SpringBoot & ORM Microlayer

Duration: 4 to 8 Hour



SpringBoot APP FOR SOCIAL PICTURE SHARING APPLICATION

This page is intentionally left blank.

Contents

[1 Important Instructions 4](#_Toc529802191)

[2 Business-Requirement: 5](#_Toc529802192)

[2.1 Problem Statement: 5](#_Toc529802193)

[2.2 User Interface (JSP OR Thymeleaf Pages) 6](#_Toc529802194)

[3 User-Interface (JSP Pages OR Thymeleaf) Details 7](#_Toc529802195)

[3.1 Sign In Page 7](#_Toc529802196)

[3.1.1 Sign In Page Requirement 7](#_Toc529802197)

[3.1.2 Sign-In Page Wireframe 7](#_Toc529802198)

[3.1.3 Register Page Wireframe 7](#_Toc529802199)

[3.2 Upload Media Page 8](#_Toc529802200)

[3.2.1 Upload Media Page Requirement 8](#_Toc529802201)

[3.2.2 Upload Single Media Page Wireframe 8](#_Toc529802202)

[3.2.3 Upload Multiple Media Page Wireframe 9](#_Toc529802203)

[3.3 My Media Page 10](#_Toc529802204)

[3.3.1 My Media Page Requirement 10](#_Toc529802205)

[3.3.2 My Media Page Wireframe 10](#_Toc529802206)

[3.4 Media Detail Page 11](#_Toc529802207)

[3.4.1 Media Detail Page Requirement 11](#_Toc529802208)

[3.4.2 Media Detail Page Wireframe 12](#_Toc529802209)

[3.5 Followers/Following Page 13](#_Toc529802210)

[3.5.1 Followers/Following Page Requirement 13](#_Toc529802211)

[3.5.2 Followers/Following Page Wireframe 14](#_Toc529802212)

[3.6 Account Page 15](#_Toc529802213)

[3.6.1 Activity Log/Newsfeed page 15](#_Toc529802214)

[3.6.2 Blocked Users Page 15](#_Toc529802215)

[3.6.3 Account Details Page 16](#_Toc529802216)

[3.6.4 Search Page 17](#_Toc529802217)

[4 High-Level Solution Architecture 18](#_Toc529802218)

[4.1 Package and Controller Architecture 19](#_Toc529802219)

[5 Methodology 20](#_Toc529802220)

[5.1 Agile 20](#_Toc529802221)

[5.2 Continuous Integration 20](#_Toc529802222)

[5.2.1 As you code the solution: 20](#_Toc529802223)

[5.2.2 On conclusion of application development (before going to production) 20](#_Toc529802224)

[6 Technical Specification – Spring Boot Layer Solution Development Environment 21](#_Toc529802225)

[6.1 Spring Boot Layer 21](#_Toc529802226)

[6.2 Editors 21](#_Toc529802227)

[7 Development Workflow 22](#_Toc529802228)

[8 Important Instructions 23](#_Toc529802229)

[9 Assessment Deliverables 23](#_Toc529802230)

[10 Other Full Stack Layers 24](#_Toc529802231)

[10.1 UI Layer (Not Applicable for Present Case Study) 24](#_Toc529802232)

[10.2 UX Layer (Not Applicable for Present Case Study) 24](#_Toc529802233)

[10.3 Back End Layer (Not Applicable for Present Case Study) 24](#_Toc529802234)

[10.4 Middle Tier Framework Layer (Not Applicable for Present Case Study) 24](#_Toc529802235)

[10.5 ORM & Integration Layer (Applicable for Present Case Study) 24](#_Toc529802236)

[10.6 Database Layer (Not Applicable for Present Case Study) 24](#_Toc529802237)

[10.7 Ancillary Layer(Not Applicable for Present Case Study) 24](#_Toc529802238)

[10.8 Deployment & Infra(Not Applicable for Present Case Study) 24](#_Toc529802239)

# Important Instructions

1. Follow the design specifications mentioned in the case study. You are free to improvise certain specifications mentioned in the case-study. But, for each such improvisation, you should keep the concerned POC informed. **POC will get in touch with concerned team at IIHT.**
2. You should stay **motivated** to initiate such and specific communications as it may have positive influence on the evaluation scores.
3. Please make sure that your code does not have any compilation errors while submitting your case study solution.
4. The final solution **should be deployed in docker**.
5. **Running the docker image should run the Spring Boot project** at <http://localhost:portnumber> or <http://ipaddress:portnumber>.
6. **Use h2 in-memory database where database functionality is required.**
7. Implement the code using best design standards for:
   1. Variable declarations
   2. Class names
   3. Package names
   4. Code Refactoring
8. Use Spring Internationalization, Spring Validation, Spring Exception, Spring Security where required.
9. Use SL4J for logging.
10. If you wish to use template library (e.g. thymeleaf) in place of JSP for building front-end; please feel free to do so.
    1. <https://www.thymeleaf.org/>
    2. <https://www.thymeleaf.org/doc/tutorials/2.1/thymeleafspring.html>

# Business-Requirement:

## Problem Statement:

**The PixoGram (Spring Boot Web Application)** allows you to:

1. Register as a user
2. Login as a user
3. Retrieve/Change password
4. Manage your user account
5. Login/Logout to/from your account on PixoGram
6. Add Content
   1. Upload single/multiple pictures, caption and description
   2. Upload single/multiple videos, caption and description
7. Manage Content
   1. Organize Picture in Gallery
   2. Organize Videos in Playlists
   3. Rename Pictures and Videos
   4. Edit Caption, Description, Comment
8. Social Features
   1. Use emojis in comment
   2. Like or Unlike comment, pictures and videos of other users
   3. Follow/Unfollow other users
9. Edit Pictures
   1. Apply effects to pictures (sepia, grayscale, etc.)
10. Hide Pictures/Videos
11. Activity/Newsfeed
    1. View activity log of user-activity on the PixoGram
12. Offline Functionality:
    1. Certain parts of the application should be available in absence of connectivity.
    2. Relevant areas on the screen should display “Connectivity Not Available”
13. BONUS REWARDS/SCORE Feature:
    1. To implement offline image upload functionality such that user can upload content when offline. It will sync with backend when connected.

**In this micro layer you will develop Spring Boot web application using Java 8 and Spring Boot Framework with ORM (Spring Data OR Hibernate) support.**

## User Interface (JSP OR Thymeleaf Pages)

1. As per the navigation bar (each is independent page).
   1. Upload Media Page
      * 1. -> Single Media Upload Page
        2. -> Multiple Media Upload Page
   2. My Media Page
      * 1. -> Media Detail Page
   3. Followers/Following Page
      * 1. -> Follower Page -> Follower Media Detail Page
        2. -> You Follow User Page -> You Follow Media Detail Page
   4. Account Details Page
      * 1. Sign In Page
           1. Blocked Accounts Page
           2. Newsfeed Page
           3. Account Update Page
           4. Search Page
           5. Logout Page
        2. Register Page

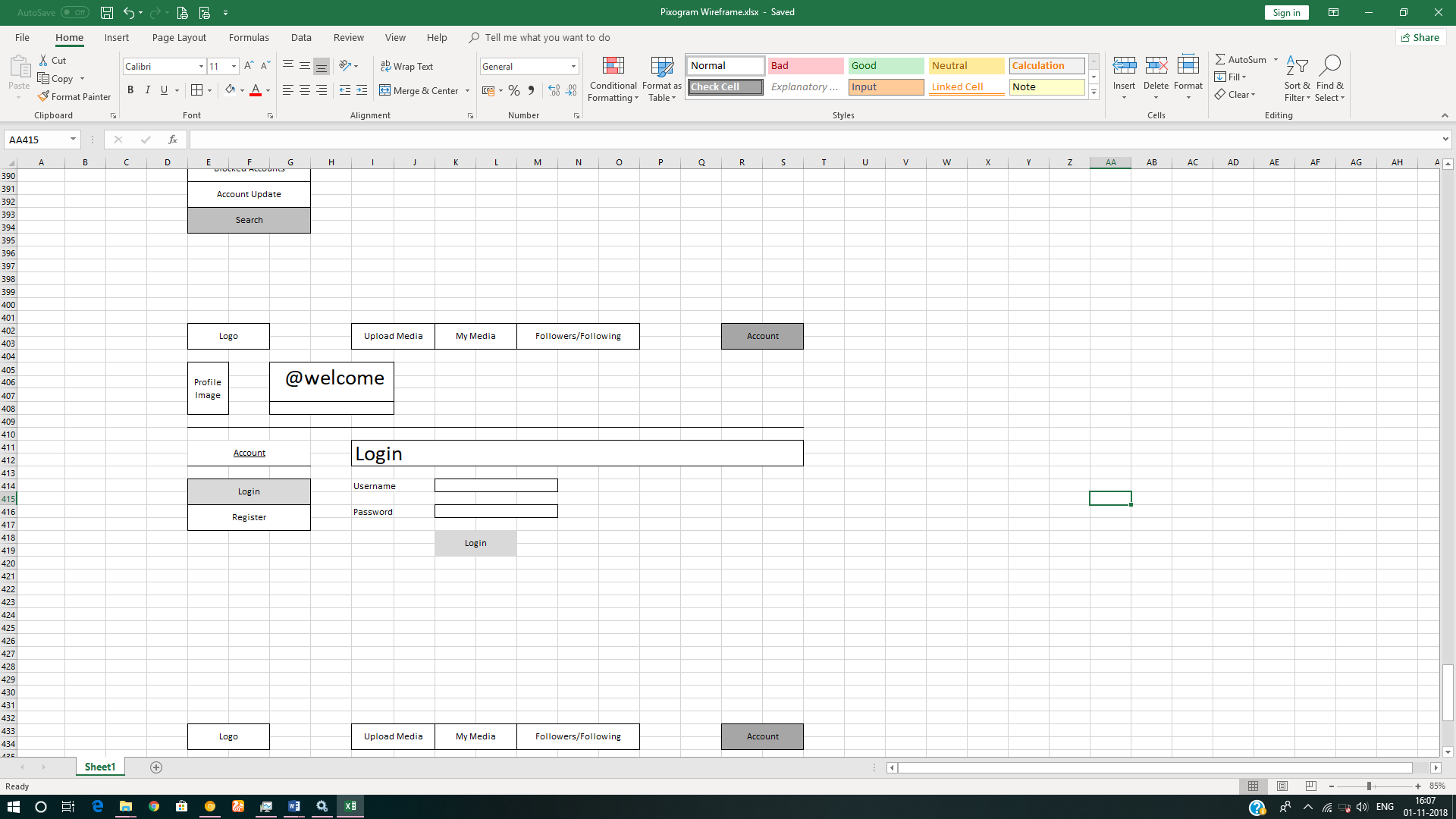
# User-Interface (JSP Pages OR Thymeleaf) Details

## Sign In Page

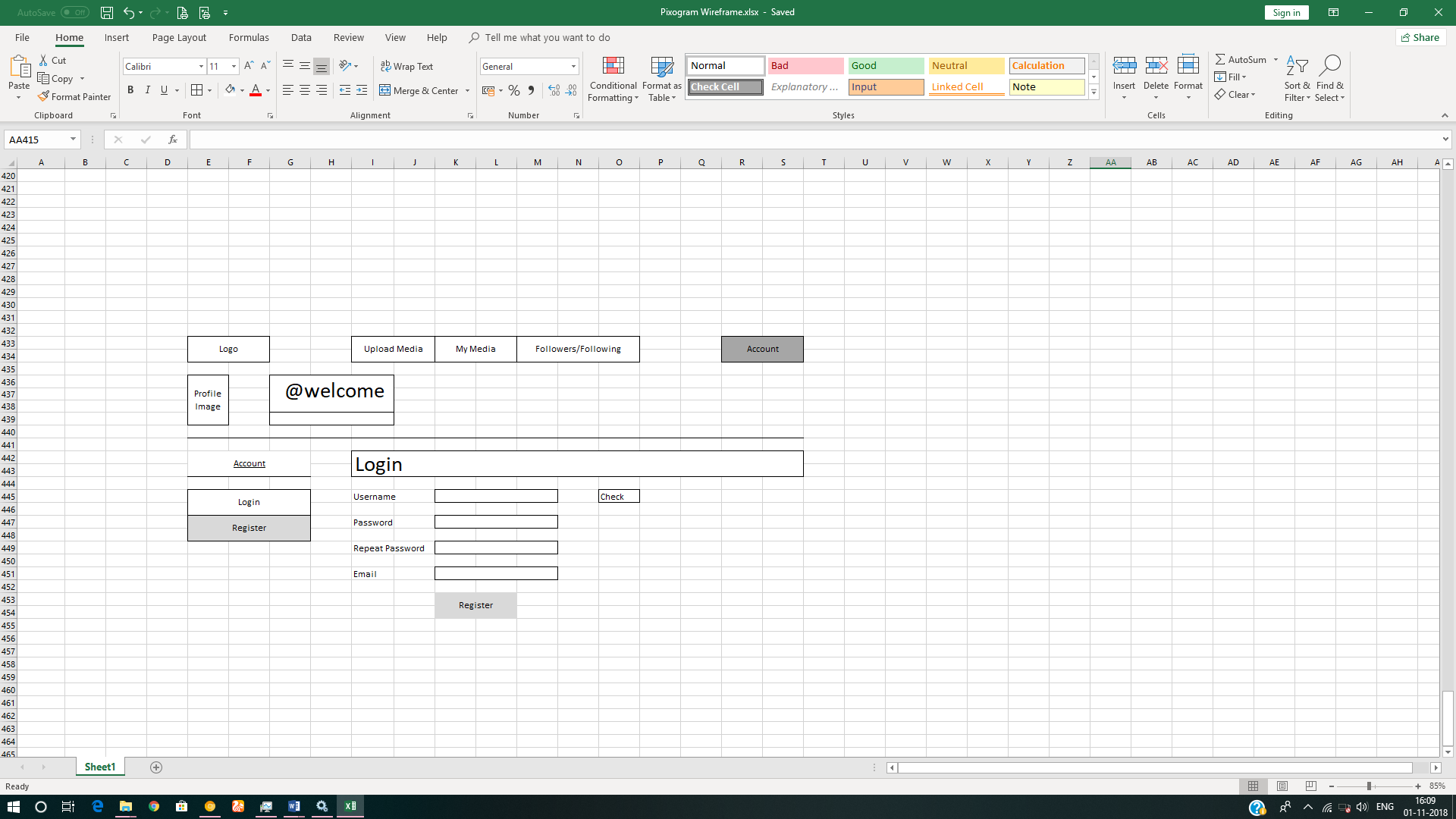
### Sign In Page Requirement

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

### Sign-In Page Wireframe



### Register Page Wireframe

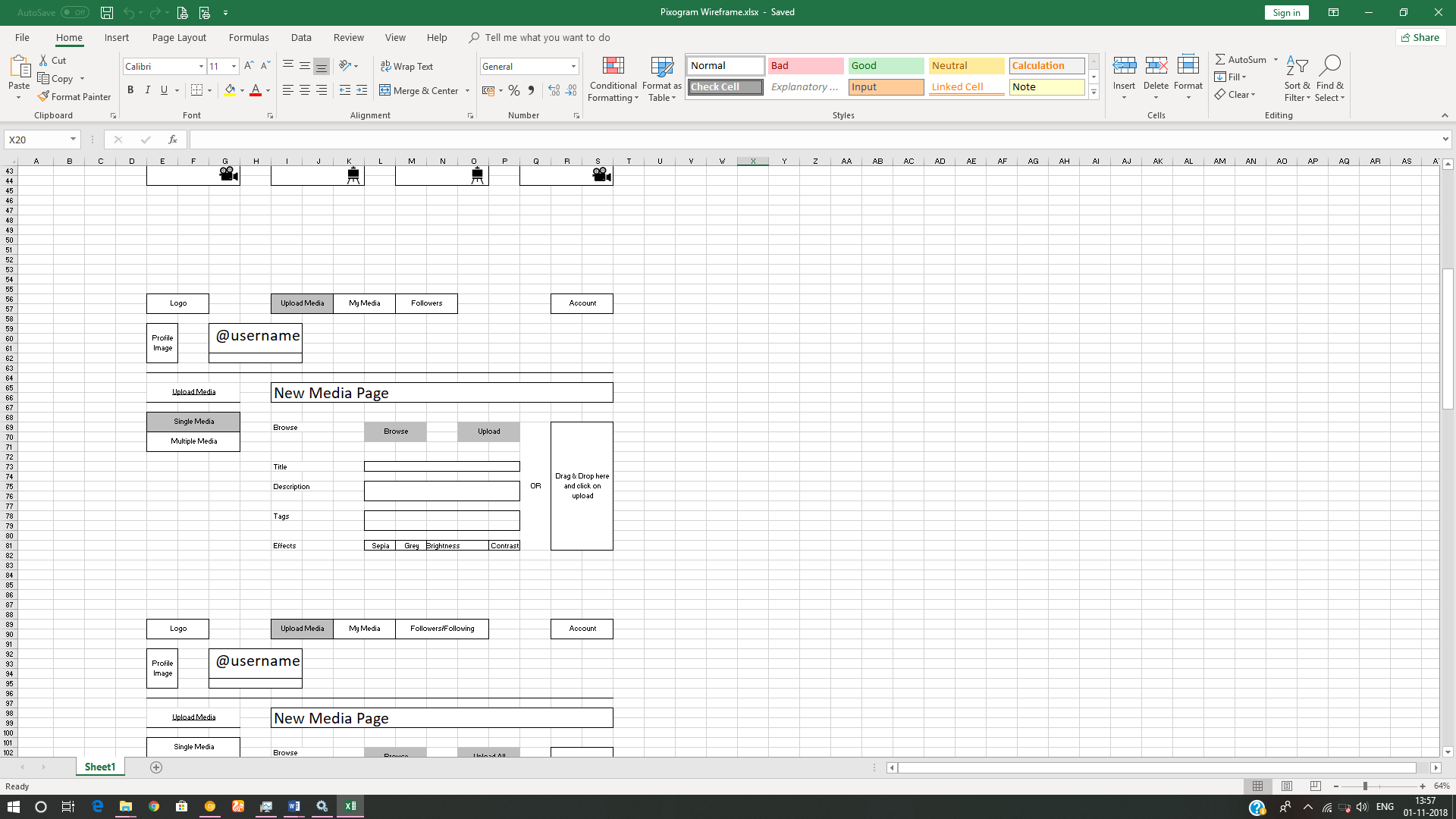


## Upload Media Page

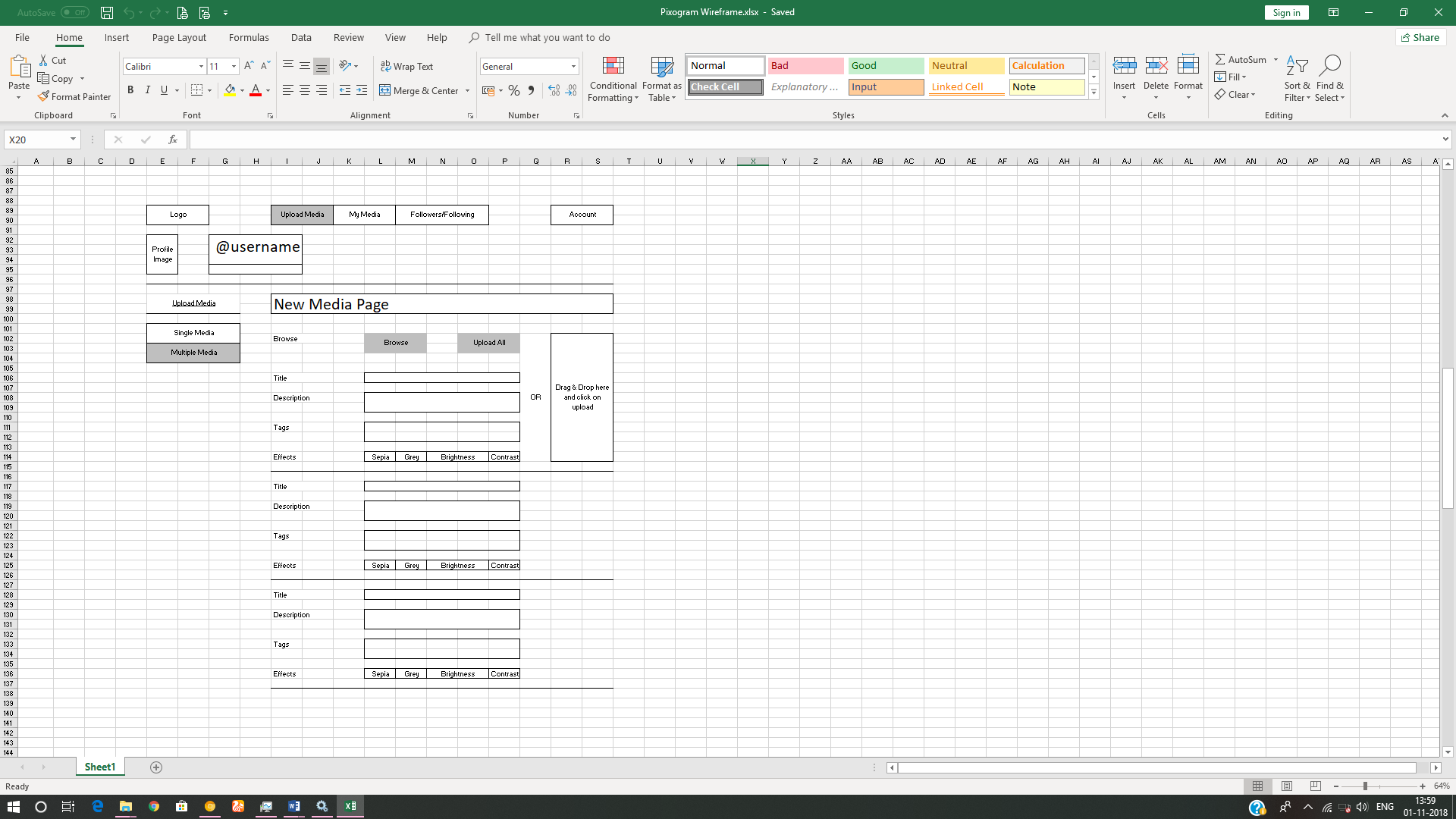
### Upload Media Page Requirement

1. It will have two sub-page
   1. Single Media Upload Page
   2. Multiple Media Upload Page
2. It allows you to upload media in two formats
   1. Images - png, jpeg, gif
   2. 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 page for single media upload. Once it is done and approved, then create the page 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:
   1. Title
   2. Description
   3. Tags
   4. Effects – sepia, greyscale, brightness, contrast etc.
      1. 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

### Upload Single Media Page Wireframe



### Upload Multiple Media Page Wireframe

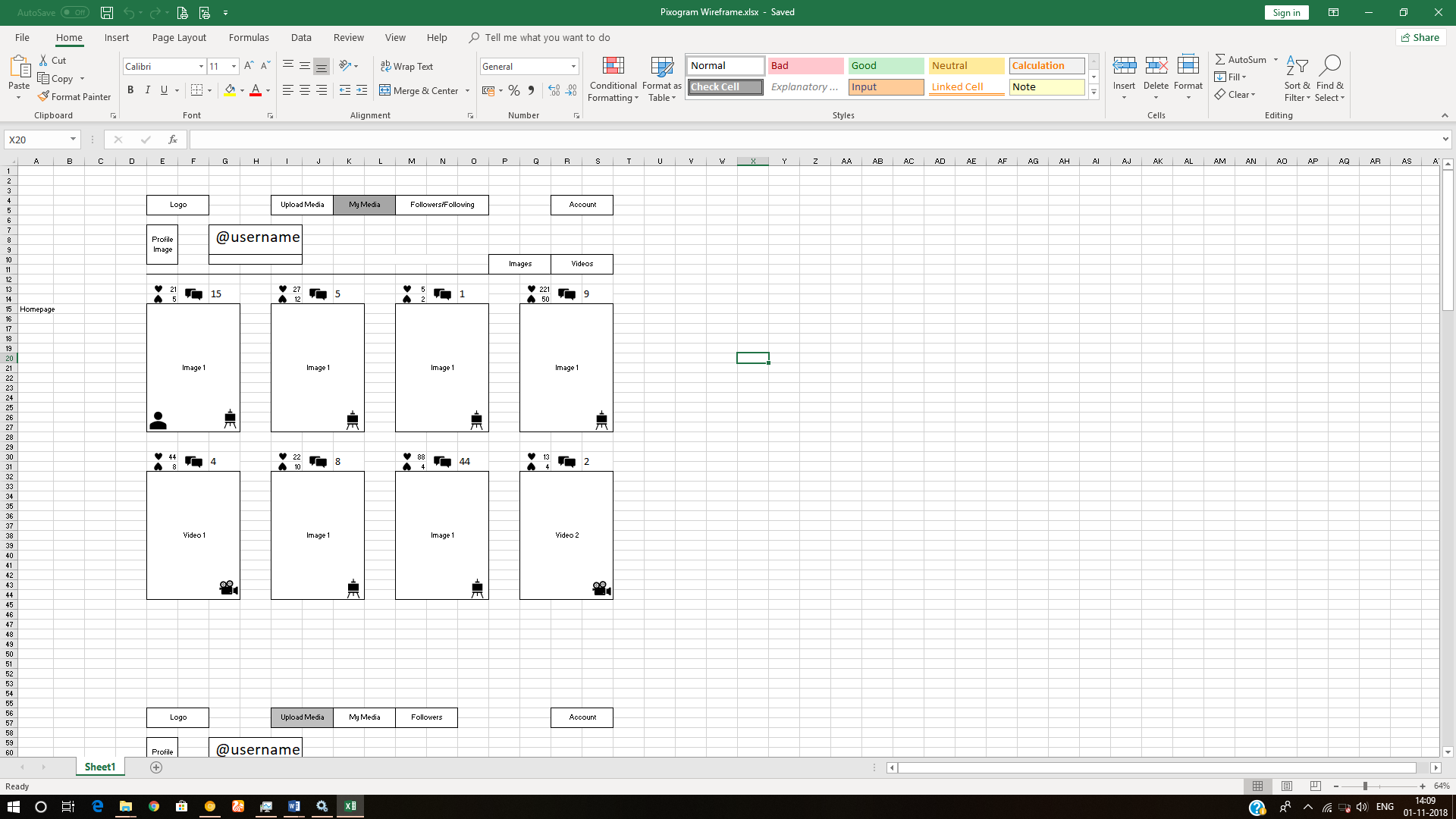


## My Media Page

### My Media Page Requirement

1. This page 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.
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:
   1. Emoji Icon + number of like. (not clickable)
   2. Emoji Icon + number of unlike. (not clickable)
   3. Emoji Icon + number of comments.
   4. Emoji Icon to specify whether it is used for default profile picture.
9. User should be able to click on the media (image/video) thumbnail to view further media details and interact with the media.

### My Media Page Wireframe

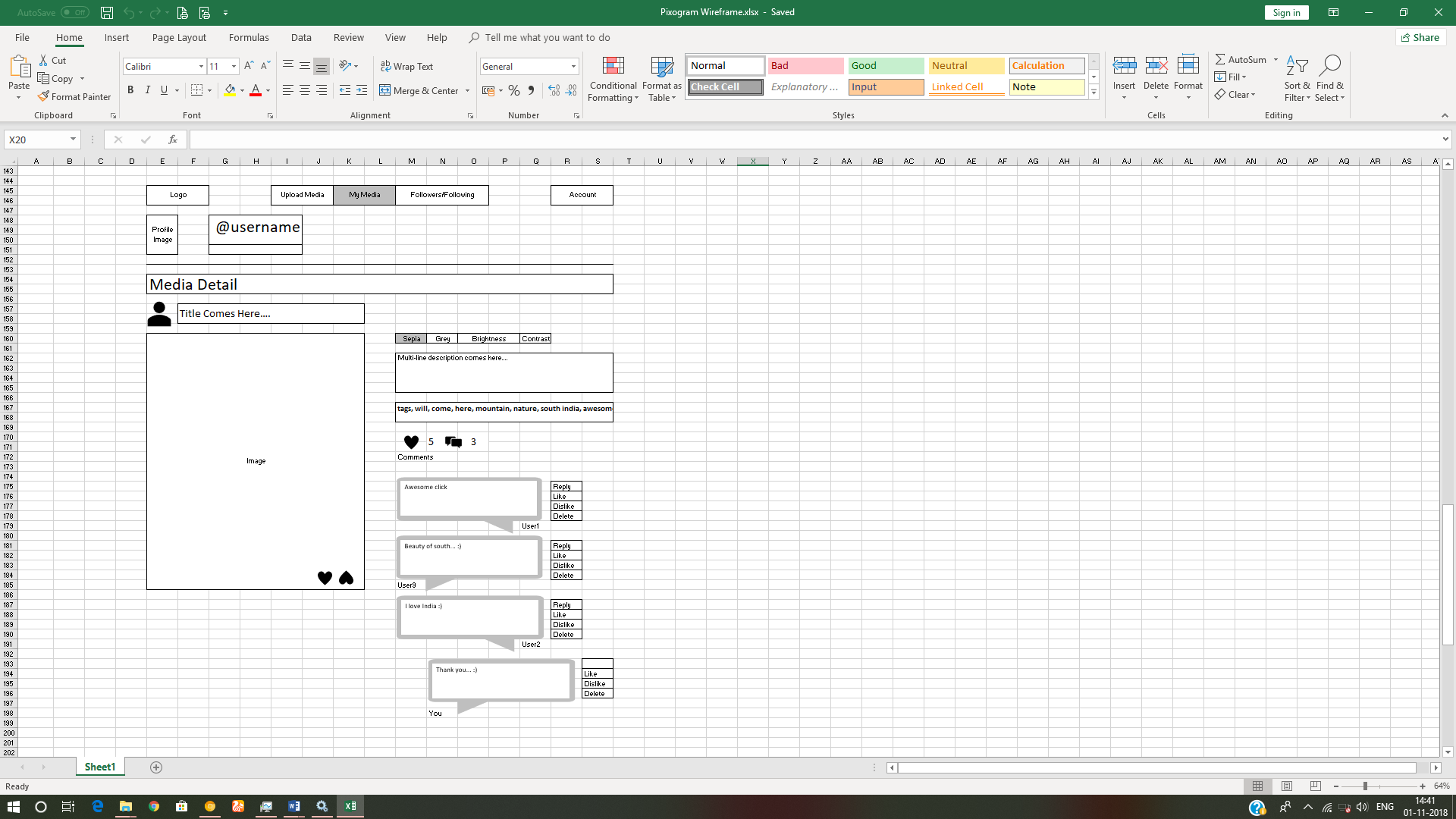


## Media Detail Page

### Media Detail Page Requirement

1. It will display your username on top along with (Follow/Unfollow) toggle button
2. If Image:
   1. Original dimension.
   2. Name of effect applied.
   3. “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:
   1. 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
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

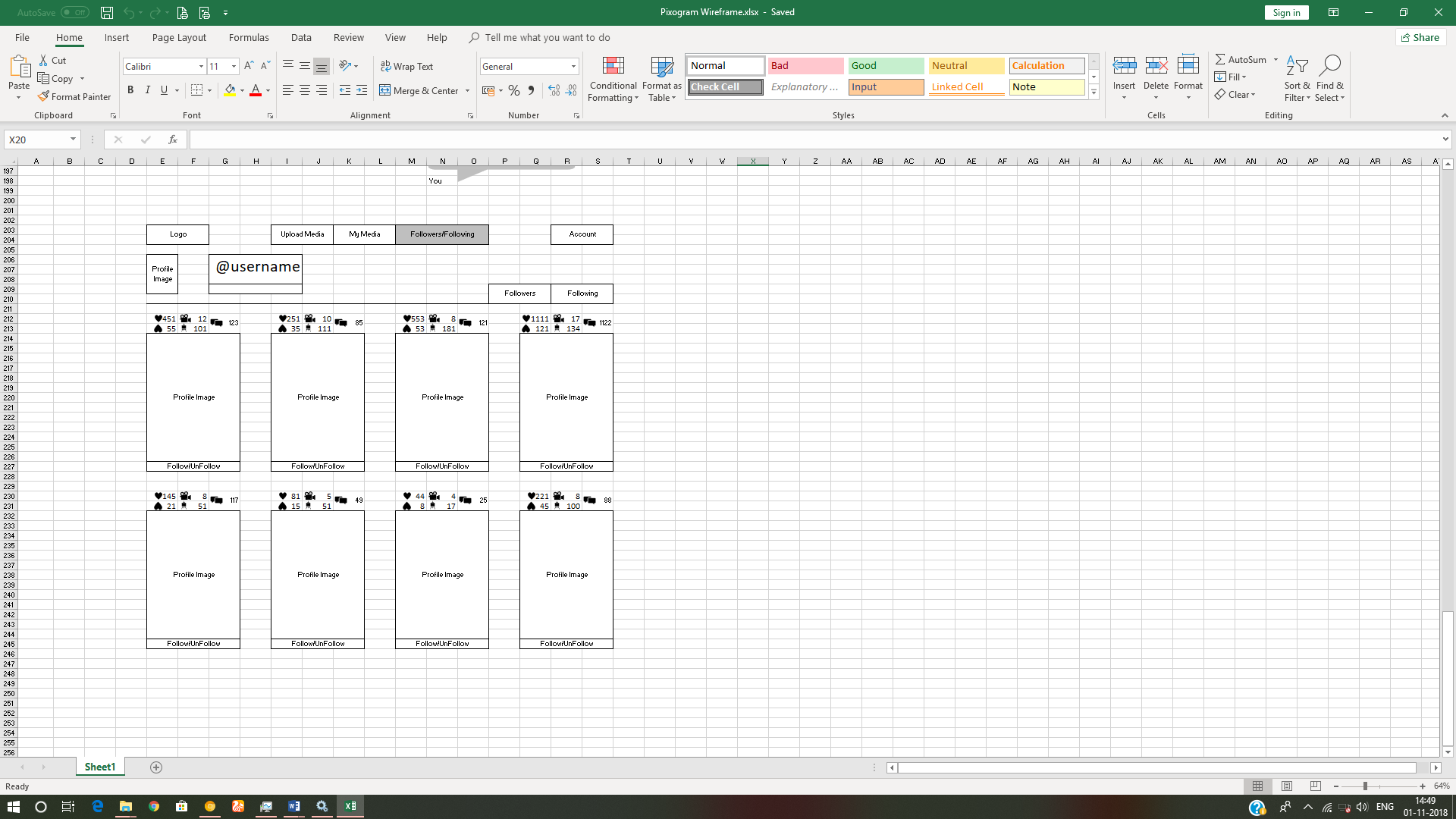


## Followers/Following Page

### Followers/Following Page 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:
   1. Emoji Icon + total number of like. (not clickable)
   2. Emoji Icon + total number of unlike. (not clickable)
   3. 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:
   1. Followers Button – Will display all followers of your pixogram account
   2. Following Button – Will display all accounts you are following
   3. By default, both buttons are enabled.
5. User can click on any user profile picture and navigate to the “My Media Page” of respective user.
6. Once on the “My Media Page” of the respective user, you can click on any media item to navigate to the respective “Media Detail Page” page.
7. Once on media detail page of the respective user for respective media:
   1. It will display username on top along with (Follow/Unfollow) toggle button
   2. If Image:
      1. Original dimension.
      2. Name of effect applied.
      3. “Make Profile Picture” button is disabled as this media does not belong to your account.
   3. If Video:
      1. 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
   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 respective user’s post.

### Followers/Following Page Wireframe



## Account Page

It will consist of 4 sub-pages

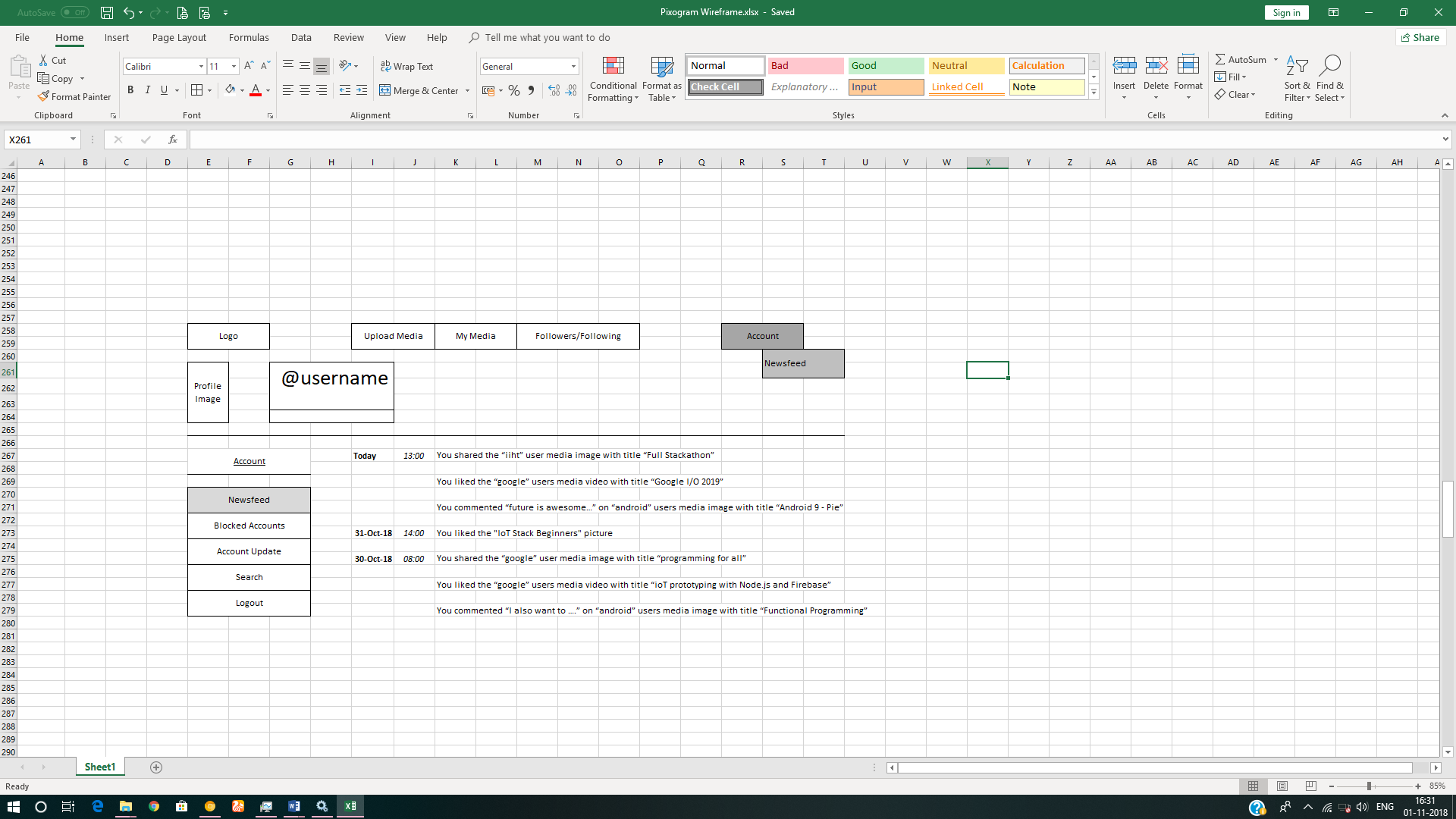
* 1. Account Details sub-page
  2. Activity Log/Newsfeed sub-page
  3. Blocked Users sub-page
  4. Search sub-page

### Activity Log/Newsfeed page

#### Activity Log/Newsfeed Page Requirement

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

#### Activity Log/Newsfeed Page Wireframe

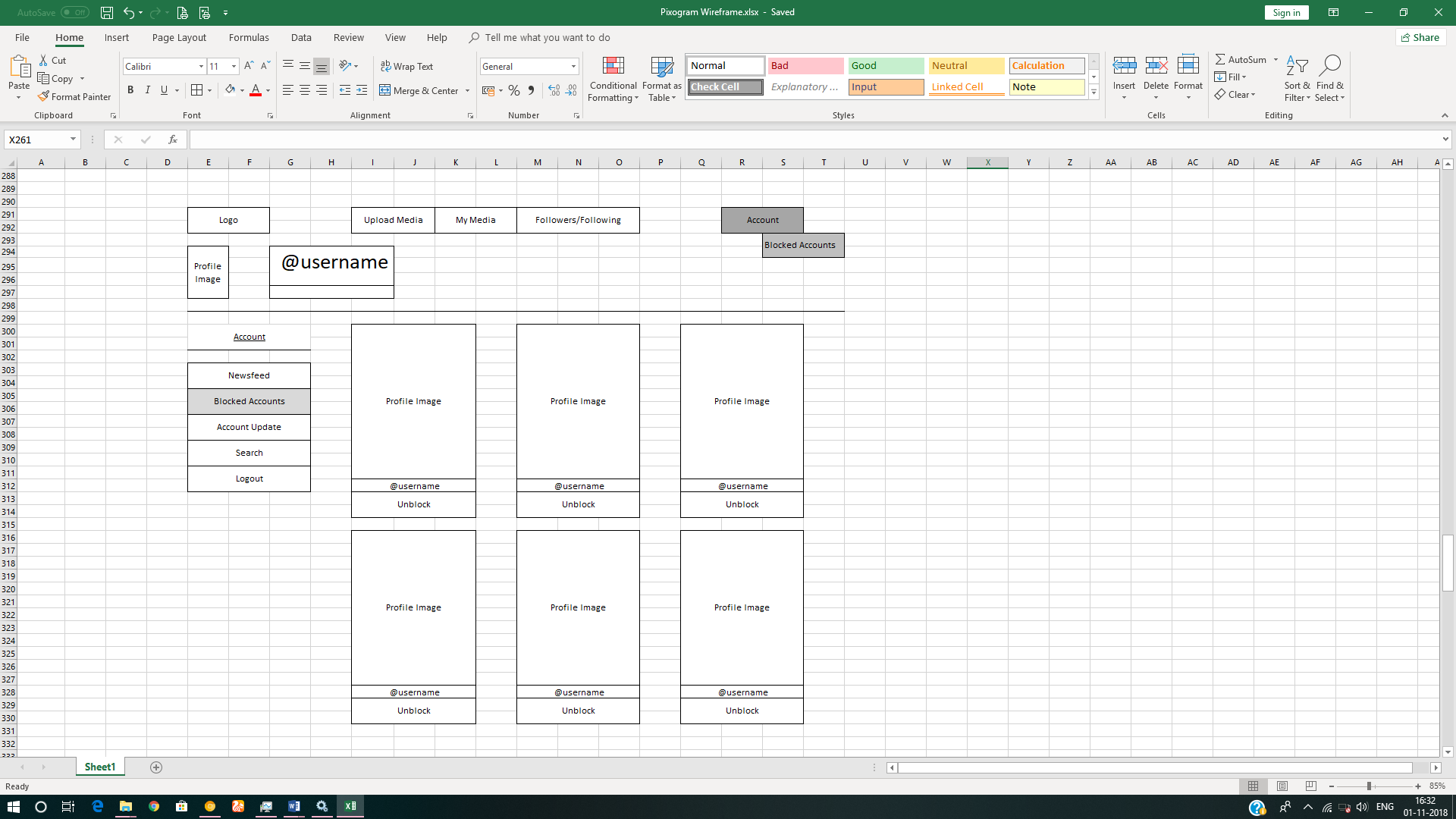


### Blocked Users Page

#### Blocked Users Page Requirement

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

#### Blocked User Page Wireframe

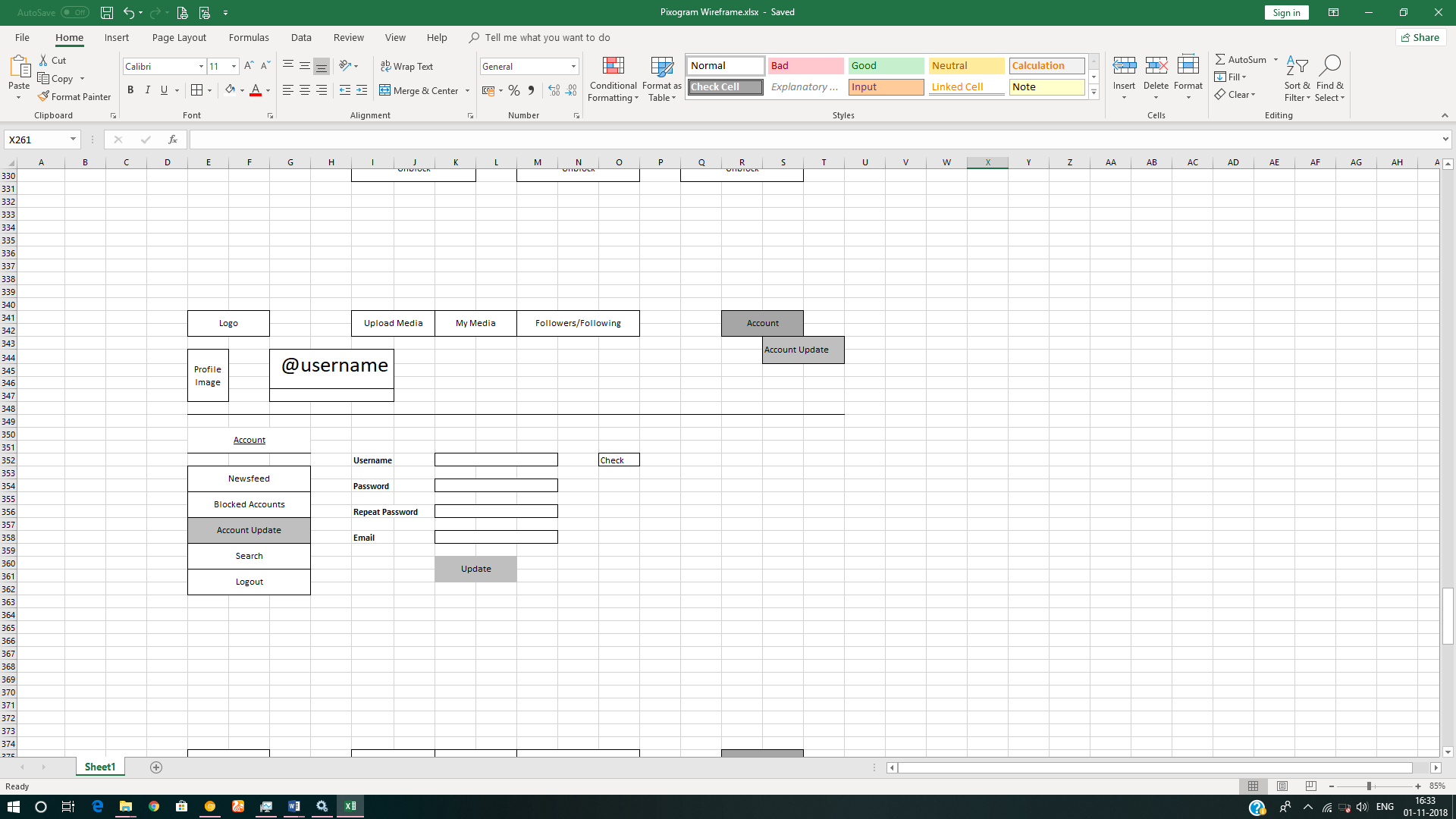


### Account Details Page

#### Account Details Page 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 Page Wireframe

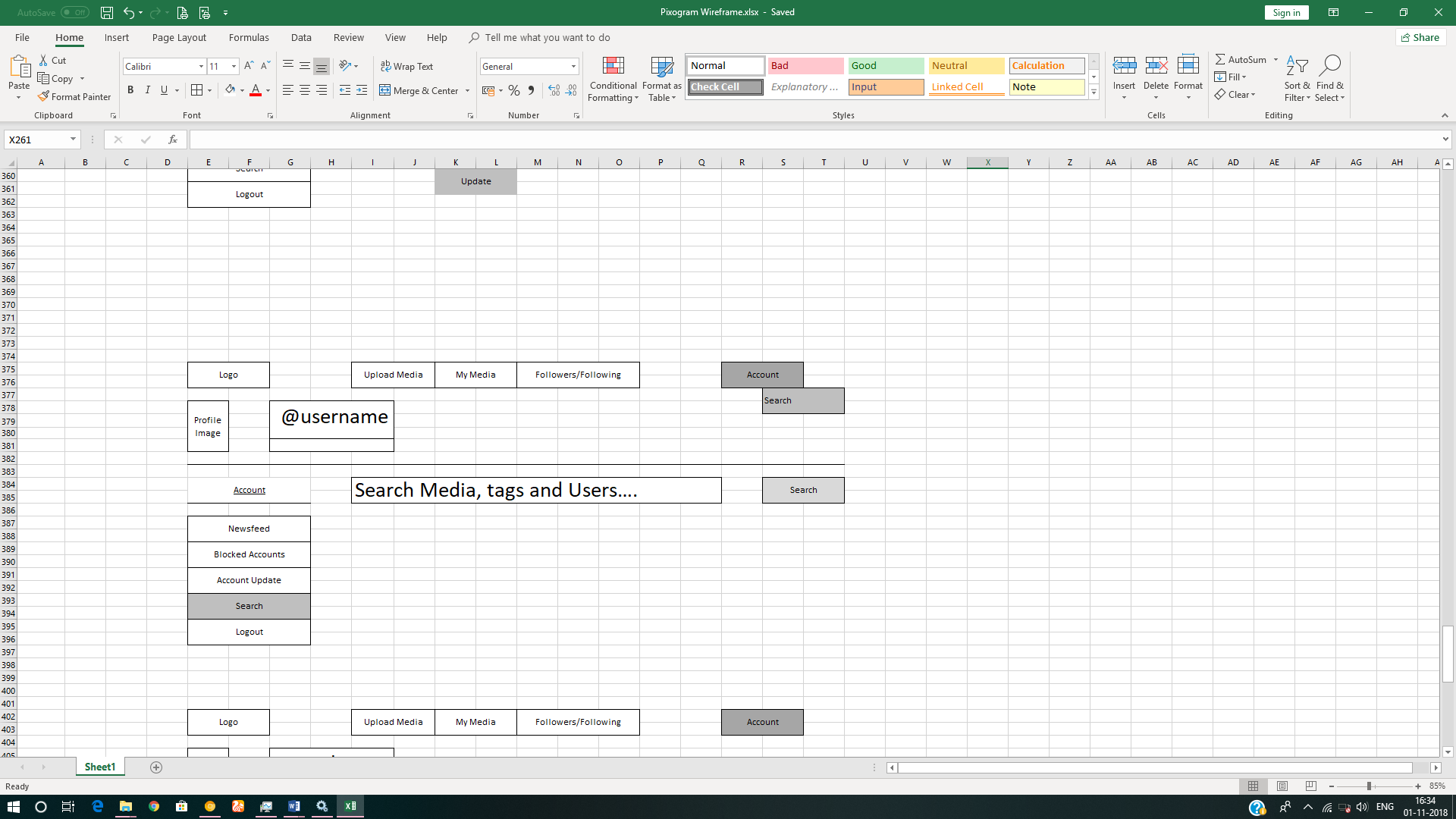


### Search Page

#### Search Page Requirement

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

#### Search Page Wireframe



# High-Level Solution Architecture

Spring Boot

Dispatcher Servlet

Deployment Descriptor

HTTP Request

Handler Adapter

Repository   
(Spring Data)

Execute Business Logic

Service   
(Business Layer)

Resolve View

View Resolver

HTTP Response

View

View Name

Controller

Model

Database

Entity

## Package and Controller Architecture

1. com.social.imageApp.controllers
   1. com.social.imageApp.upload.controllers
      1. UploadSingleMediaController
      2. UploadMultipleMediaControllers
   2. com.social.imageApp.mymedia.controllers
      1. MyMediaController
      2. MediaDetailController
   3. com.social.imageApp.follow.controllers
      1. FollowController
   4. com.social.imageApp.account.controllers
      1. SignInController
      2. RegisterController
      3. BlockedController
      4. NewsfeedController
      5. AccountUpdateController
      6. SearchController
      7. LogoutController
2. Each controller will depend on its respective service; which in turn will depend on respective DAO layer.
3. All entities should be created in respective com.social.imageApp.entities package
4. All POJO’s should be created in respective com.social.imageApp.model package
5. **Use h2 in-memory database where database functionality is required.**
6. Implement the code using best design standards for:
   1. Variable declarations
   2. Class names
   3. Package names
   4. Code Refactoring
7. Use POJOs, Entities, Spring Internationalization, Spring Validation, Spring Exception, Spring Security where required.
8. Use SL4J for logging.

# Methodology

## Agile

1. Mentor will ask you about daily progress as you start implementing Spring Boot Layer.
2. Communicate with your mentor via email; ideally daily as you develop the Spring framework Layer.
3. Scope of discussion with your mentor:
   1. Q/A
   2. New Ideas and New feature implementations
   3. Any development related challenges
   4. Skill Gaps
   5. Another pointers key to Spring Development

## Continuous Integration

### As you code the solution:

1. Check in your code - ideally every hour in the internal GIT repository at 172.18.2.18

### On conclusion of application development (before going to production)

1. Set up Jenkins on the cloud and integrate with internal GIT repository.
2. Setup Jenkins to create docker image which can deploy the final solution.
3. Save the Jenkins/build report’s as a part of final assessment deliverable.

# Technical Specification – Spring Boot Layer Solution Development Environment

## Spring Boot Layer

|  |  |
| --- | --- |
| **Framework(s)/SDK/Libraries** | **Version** |
| Spring Boot, Hibernate, Spring Data | - |
| Maven | - |
| GIT Basics | - |
| Jenkins Basics | - |
| Docker Basics | - |

## Editors

|  |  |
| --- | --- |
| **Name** | **Version** |
| STS | - |

# Development Workflow

1. You must follow following process while creating Java classes
   1. \*\*Approval includes:
      1. Communicating with mentor via email to showcase the progress.
      2. Progress must be shared with mentor as each Spring Boot App Development layer is crossed.
      3. It is recommended to avoid moving to next stage until feedback from the mentor is received.
      4. Pushing assets/code to GIT repository.
      5. Last three stages do not need any approval as they will test your Business Logic implementation skills.
   2. \*Business Logic Plan
      1. Declaring methods you plan to use for data validation, database io, data formatting or any other utility classes/methods.
      2. Documenting methods you plan to use for data validation, database io, data formatting or any other utility classes/methods.
   3. \*\*\*Business Logic includes:
      1. Data Validation
      2. Data Formatting before Database IO
      3. Checking proper input from JSP pages

# Important Instructions

1. Follow the design specifications mentioned in the case study. You are free to improvise certain specifications mentioned in the case-study. But, for each such improvisation, you should keep the concerned POC informed. **POC will get in touch with concerned team at IIHT.**
2. You should stay **motivated** to initiate such and specific communications as it may have positive influence on the evaluation scores.
3. Please make sure that your code does not have any compilation errors while submitting your case study solution.
4. The final solution **should be deployed in docker**.
5. **Running the docker image should run the Spring Boot project** at <http://localhost:portnumber> or <http://ipaddress:portnumber>.
6. **Use h2 in-memory database where database functionality is required.**
7. Implement the code using best design standards for:
   1. Variable declarations
   2. Class names
   3. Package names
   4. Code Refactoring
8. Use POJOs, Entities, Spring Internationalization, Spring Validation, Spring Exception, Spring Security where required.
9. Use SL4J for logging.

# Assessment Deliverables

1. Docker Image which can be executed to deploy the project at <http://localhost:portnumber> or <http://ipaddress:portnumber>.
2. Dump of command “git log”
3. Jenkins report about number of times it pulled the code from GIT
4. Docker File used to create docker image from Jenkins.
5. Working POM.xml for the project.
6. Steps on how to execute the project in STS or Eclipse.

# Other Full Stack Layers

## UI Layer (Not Applicable for Present Case Study)

|  |  |
| --- | --- |
| HTML5 | - |
| CSS3 | - |
| Bootstrap/Material | - |
| Typescript | - |

## UX Layer (Not Applicable for Present Case Study)

|  |  |
| --- | --- |
| Angular | 6 |
| Javascript & JQuery | - |
| Typescript | - |

## Back End Layer (Not Applicable for Present Case Study)

|  |  |
| --- | --- |
| JDK | 1.8 or above |

## Middle Tier Framework Layer (Not Applicable for Present Case Study)

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Spring Framework | Spring MVC | 5.0 or above |

## ORM & Integration Layer (Applicable for Present Case Study)

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Spring | Spring Boot | 2.0 |
| Java JPA | Hibernate | 5.0 or above |
| Spring Data | 2.0 |

## Database Layer (Not Applicable for Present Case Study)

|  |  |  |
| --- | --- | --- |
| MySQL | MySQL | 7.x + |

## Ancillary Layer(Not Applicable for Present Case Study)

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Source Code Management Tool | GIT | 2.18 |
| Build Tool/JAVA Stack | Maven | 3.5.x |
| Testing Tool/JAVA Stack | Junit/Spring Test | 4.x/5.x |
| Javascript Dependency Management Tool | NPM | 6.x.x |

## Deployment & Infra(Not Applicable for Present Case Study)

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Docker | - | 17.06.2 |
| Apache Tomcat | - | 9.0 |
| Jenkins | - | 2.121.2 |