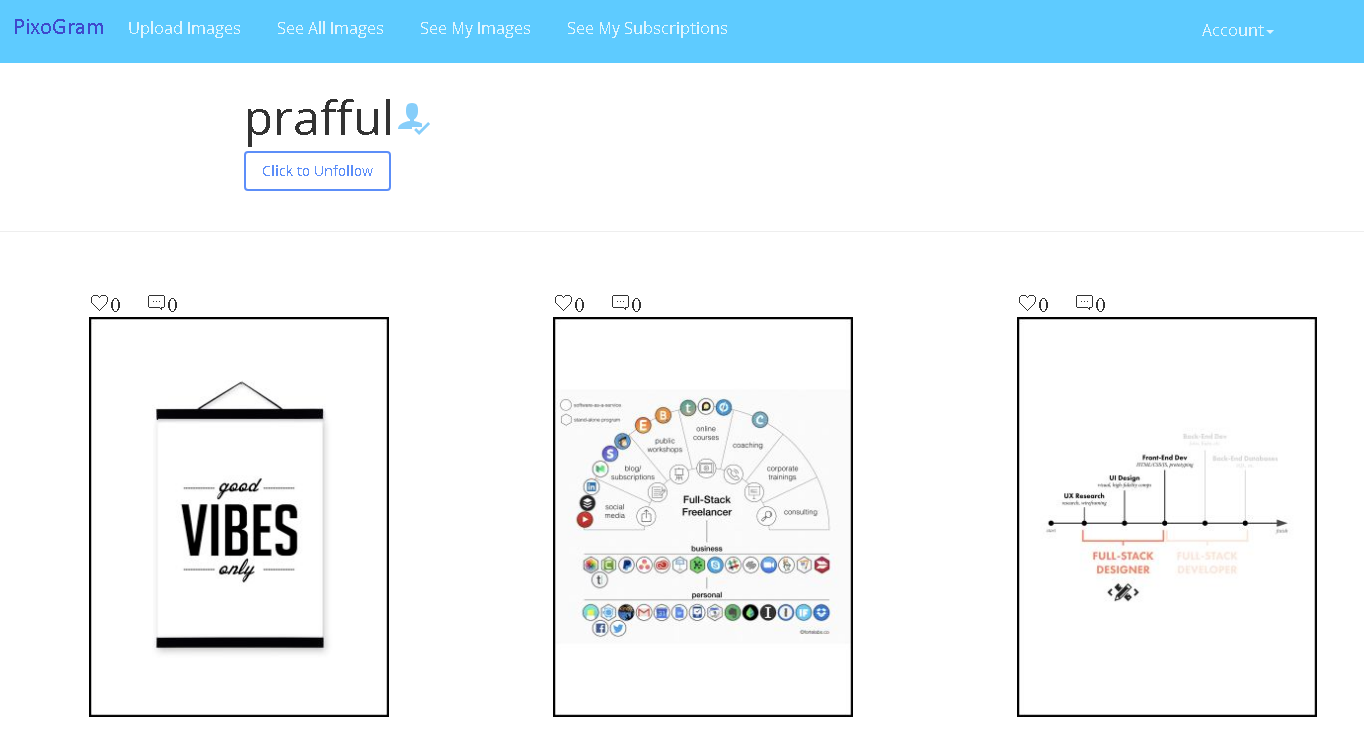
UI Microlayer

Prafful Daga  
IIHT

Micro Credential: UI Microlayer

Duration: 4 to 8 Hour



RESPONSIVE USER INTERFACE FOR SOCIAL PICTURE SHARING APPLICATION

This page is intentionally left blank.

Contents

[2 Important Instructions 3](#_Toc529799735)

[3 Business-Requirement: 4](#_Toc529799736)

[3.1 Problem Statement: 4](#_Toc529799737)

[3.2 Pages 5](#_Toc529799738)

[4 Pages (Details) 6](#_Toc529799739)

[4.1 Sign In Page 6](#_Toc529799740)

[4.1.1 Sign In Page Requirement 6](#_Toc529799741)

[4.1.2 Sign-In Page Wireframe 7](#_Toc529799742)

[4.1.3 Register Page Wireframe 7](#_Toc529799743)

[4.2 Upload Media Page 7](#_Toc529799744)

[4.2.1 Upload Media Page Requirement 7](#_Toc529799745)

[4.2.2 Upload Single Media Page Wireframe 8](#_Toc529799746)

[4.2.3 Upload Multiple Media Page Wireframe 9](#_Toc529799747)

[4.2.4 JSON Structure for Single Media File Upload 10](#_Toc529799748)

[4.2.5 JSON Structure for Multiple Media File Upload 10](#_Toc529799749)

[4.3 My Media Page 12](#_Toc529799750)

[4.3.1 My Media Page Requirement 12](#_Toc529799751)

[4.3.2 My Media Page Wireframe 13](#_Toc529799752)

[4.4 Media Detail Page 14](#_Toc529799753)

[4.4.1 Media Detail Page Requirement 14](#_Toc529799754)

[4.4.2 Media Detail Page Wireframe 15](#_Toc529799755)

[4.5 Followers/Following Page 16](#_Toc529799756)

[4.5.1 Followers/Following Page Requirement 16](#_Toc529799757)

[4.5.2 Followers/Following Page Wireframe 17](#_Toc529799758)

[4.6 Account Page 18](#_Toc529799759)

[4.6.1 Activity Log/Newsfeed Page 18](#_Toc529799760)

[4.6.2 Blocked Users Page 18](#_Toc529799761)

[4.6.3 Account Details Page 19](#_Toc529799762)

[4.6.4 Search Page 20](#_Toc529799763)

[5 Methodology 21](#_Toc529799764)

[5.1 Agile 21](#_Toc529799765)

[5.2 Continuous Integration 21](#_Toc529799766)

[5.2.1 As you code the solution: 21](#_Toc529799767)

[5.2.2 On conclusion of application development (before going to production) 21](#_Toc529799768)

[6 Technical Specification – UI Layer Solution Development Environment 21](#_Toc529799769)

[6.1 UI Layer 21](#_Toc529799770)

[6.2 Editors 21](#_Toc529799771)

[7 App UI Development Process (AGILE/SCRUM) 22](#_Toc529799772)

[8 Important Instructions 23](#_Toc529799773)

[9 Assessment Deliverables 23](#_Toc529799774)

[10 Other Full Stack Layers 24](#_Toc529799775)

[10.1 UI Layer – Just for Information (Applicable for Present Case Study) 24](#_Toc529799776)

[10.2 UX Layer (Not Applicable for Present Case Study) 24](#_Toc529799777)

[10.3 Back End Programming Layer (Not Applicable for Present Case Study) 24](#_Toc529799778)

[10.4 Middle Tier Framework Layer (Not Applicable for Present Case Study) 24](#_Toc529799779)

[10.5 ORM & Integration Layer (Not Applicable for Present Case Study) 24](#_Toc529799780)

[10.6 Database Layer (Not Applicable for Present Case Study) 24](#_Toc529799781)

[10.7 Ancillary Layer (Not Applicable for Present Case Study) 24](#_Toc529799782)

[10.8 Deployment & Infra (Not Applicable for Present Case Study) 24](#_Toc529799783)

# 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 on some http server.**
5. Implement the code using best design standards.
6. **The UI for the app should be multilingual.**
7. The **logo for the app should be in SVG format**. Use logo of your choice.
8. UI should be **responsive** across multiple devices.
9. It should be the **progressive web app** such that certain parts of the application are accessible in absence of connectivity.
10. **Dynamic functionality is not required.** Since it is UI Layer, use static content where required.

# Business-Requirement:

## Problem Statement:

**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 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
    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, responsive UI of the application using HTLM5, HTML5 API, CSS3 and Bootstrap/Material.**

## Pages

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 Link
   2. Register Page

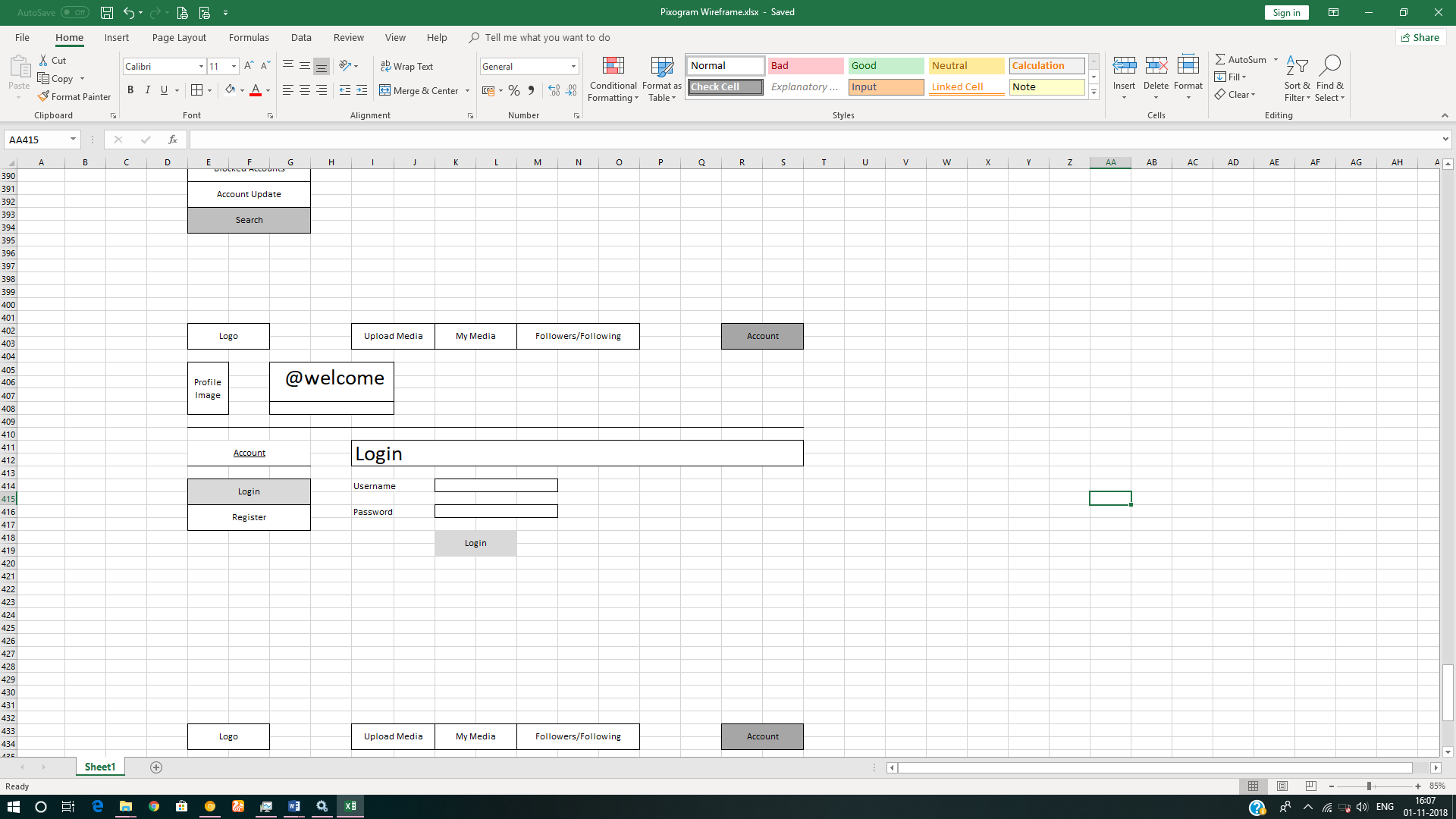
# Pages (Details)

## Sign In Page

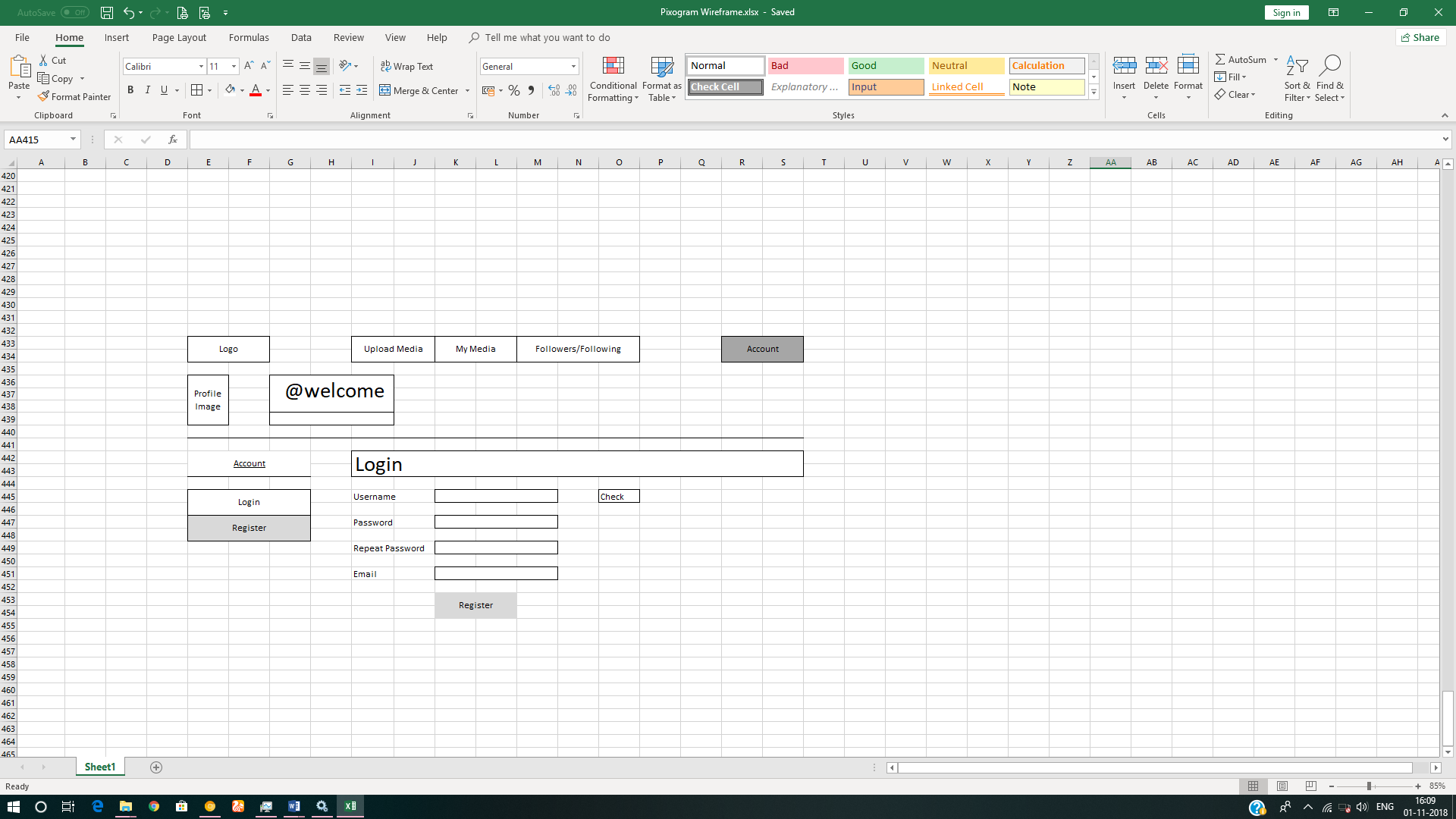
### Sign In Page Requirement

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 page, if the user is not logged-in.
4. On register page, there is check button to check if username is already in use.

### Sign-In Page Wireframe



### Register Page Wireframe

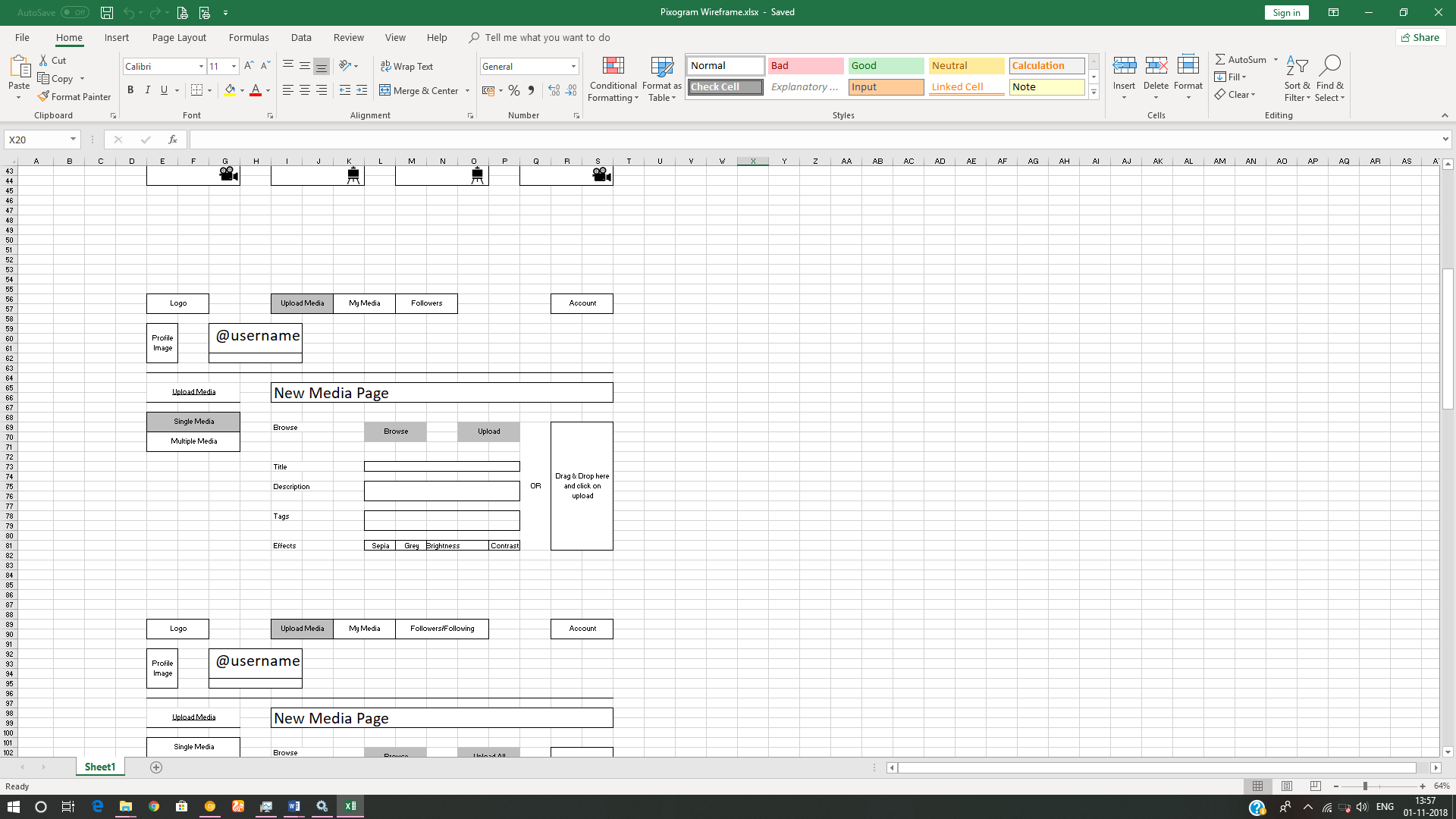


## Upload Media Page

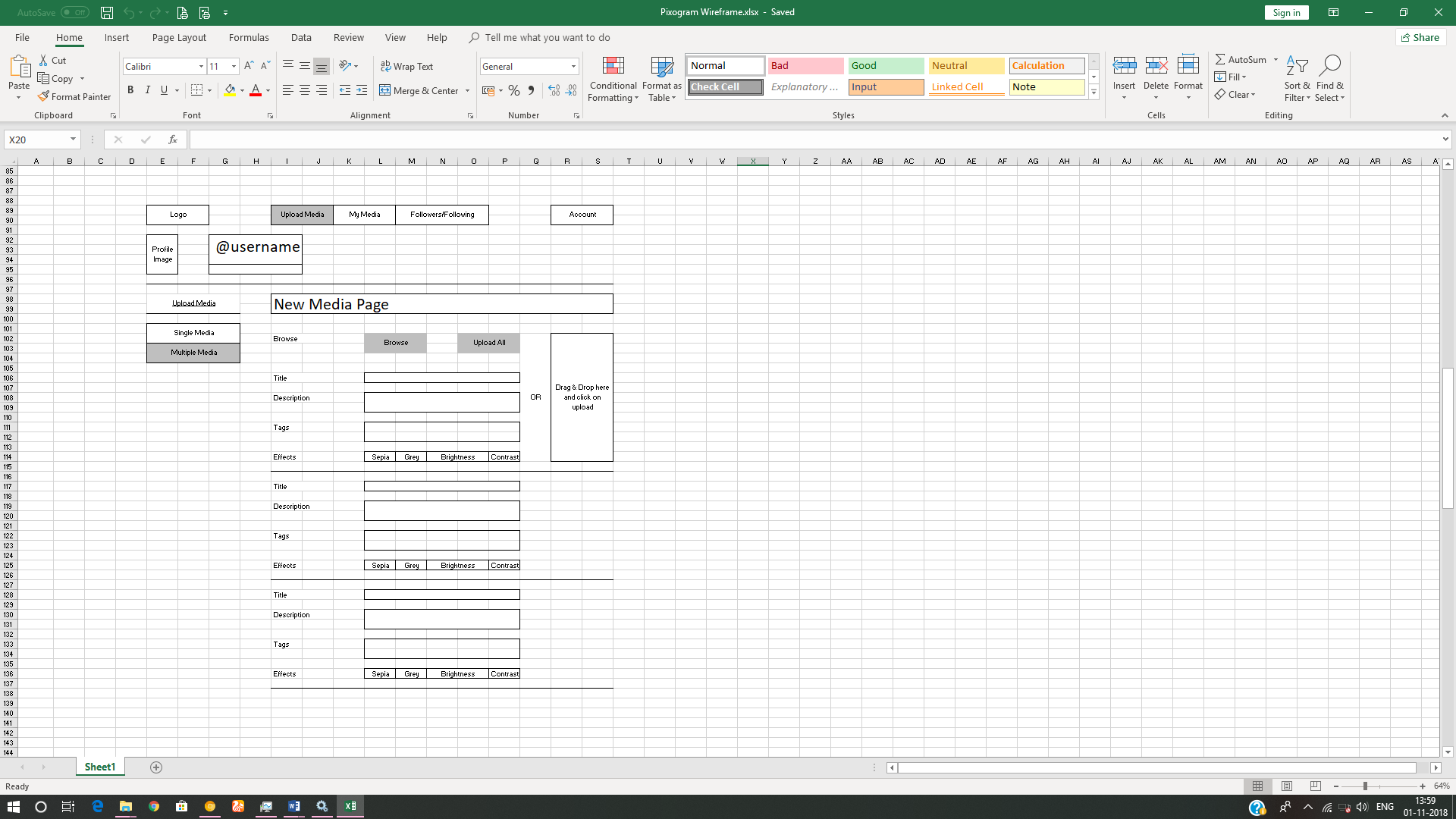
### Upload Media Page Requirement

1. It will have two sub-pages
   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 systemin respective pages. It is recommended that you should first create the UI for single media upload. Once it is done and approved, then create the UI for multiple media upload.
4. By default, 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
9. As you are not developing the UX (Angular or React); you must use HTML5 Web Storage API to save data for single media upload page
10. The JSON data for single media upload page will saved in Web Storage on click of Save/Upload button

### Upload Single Media Page Wireframe



### Upload Multiple Media Page Wireframe



### JSON Structure for Single Media File Upload

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

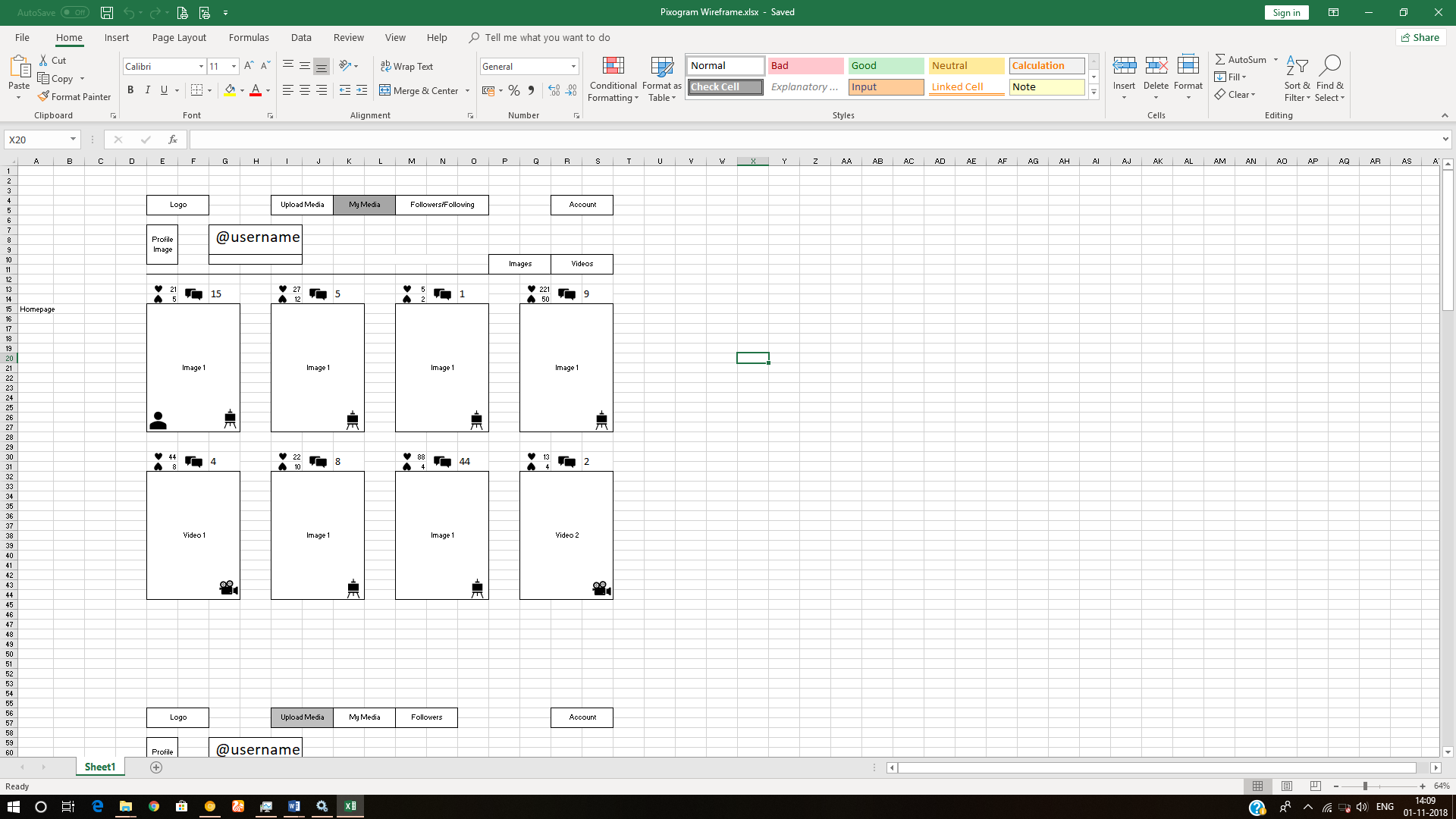
1. The following is the structure of the JSON object for multiple media upload page:
2. [
3. {
4. "id":1,
5. "title":"Full Stack Freelancer",
6. …
7. …
8. "uploadtime":"1331"
9. },
10. {
11. "id":2,
12. "title":"Technology Solutions",
13. …
14. …
15. "uploadtime":"1313"
16. },
17. {
18. "id":3,
19. "title":"Development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

## 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 can 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:
   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

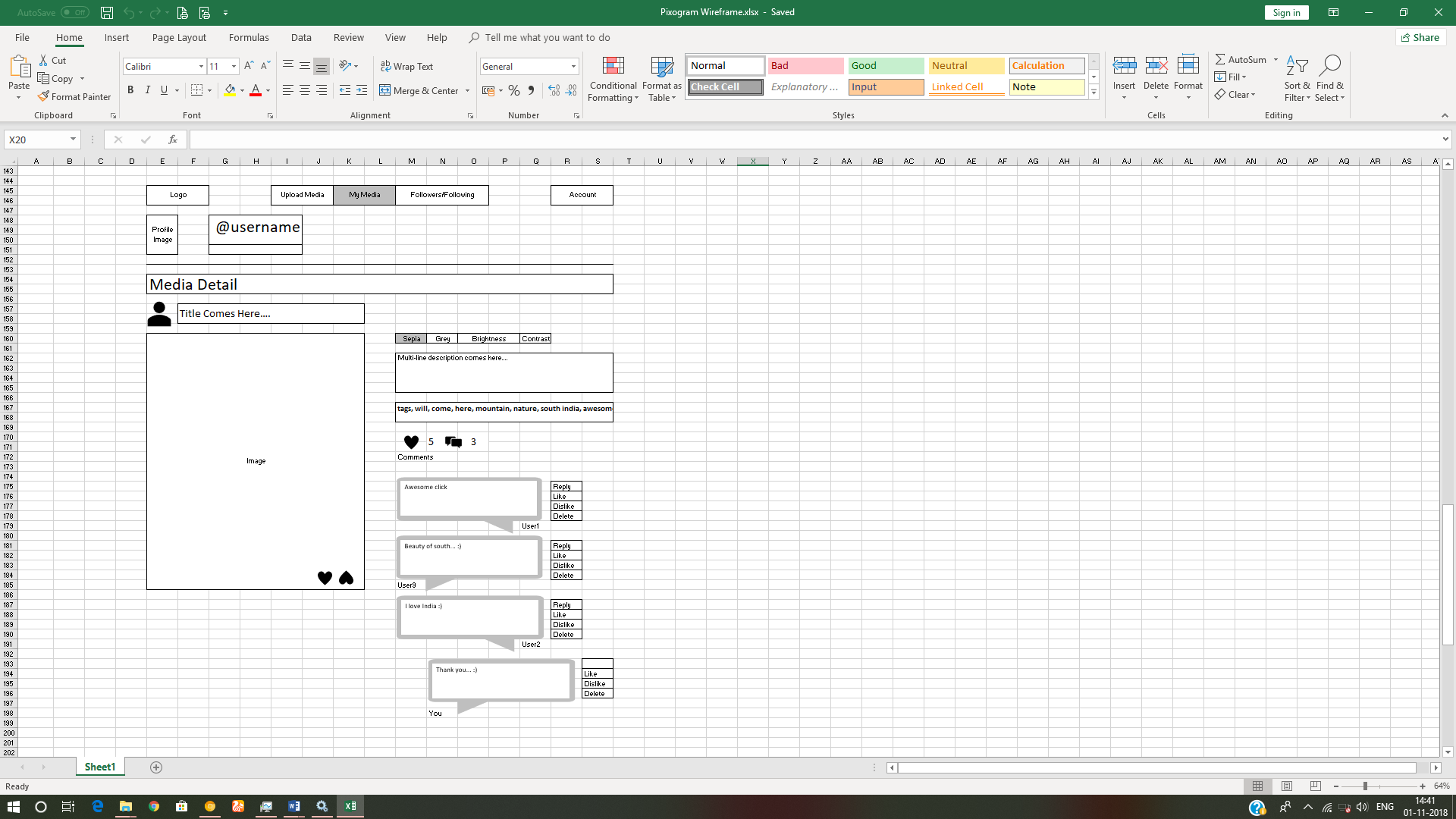


## Media Detail Page

### 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

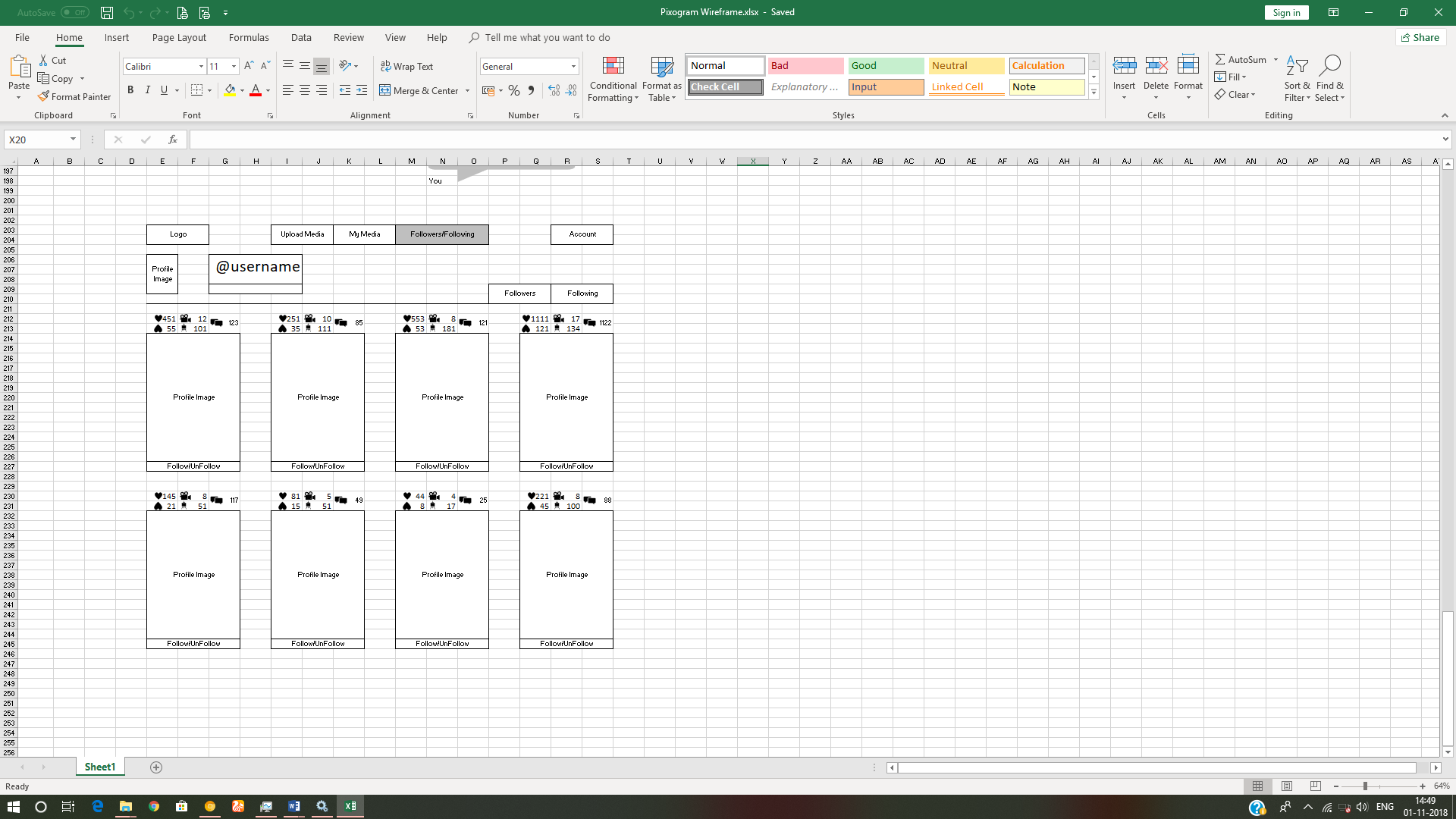


## Followers/Following Page

### 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.
   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

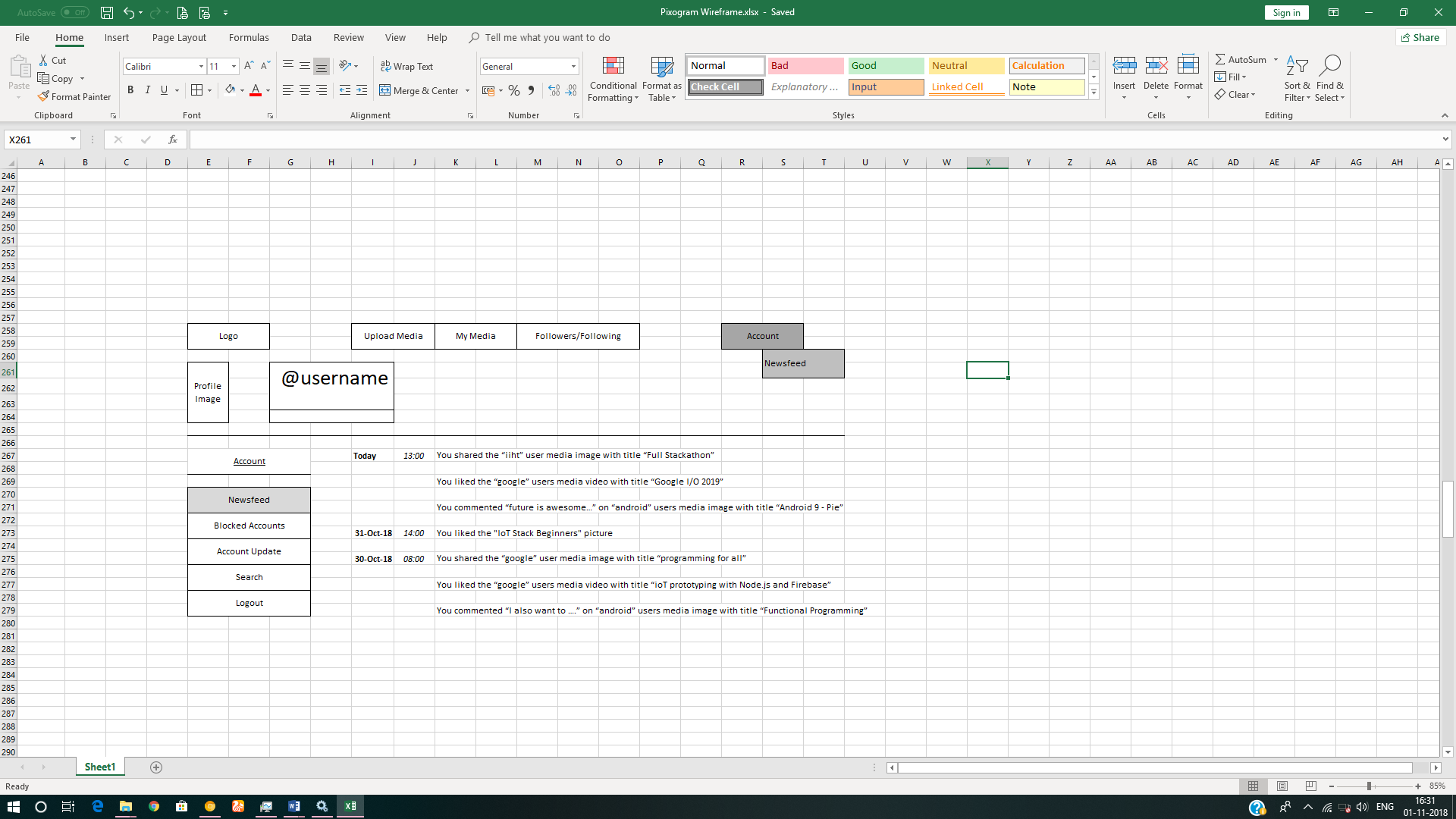
* 1. Account Details
  2. Activity Log/Newsfeed
  3. Blocked Users
  4. Search
  5. Logout link

### 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

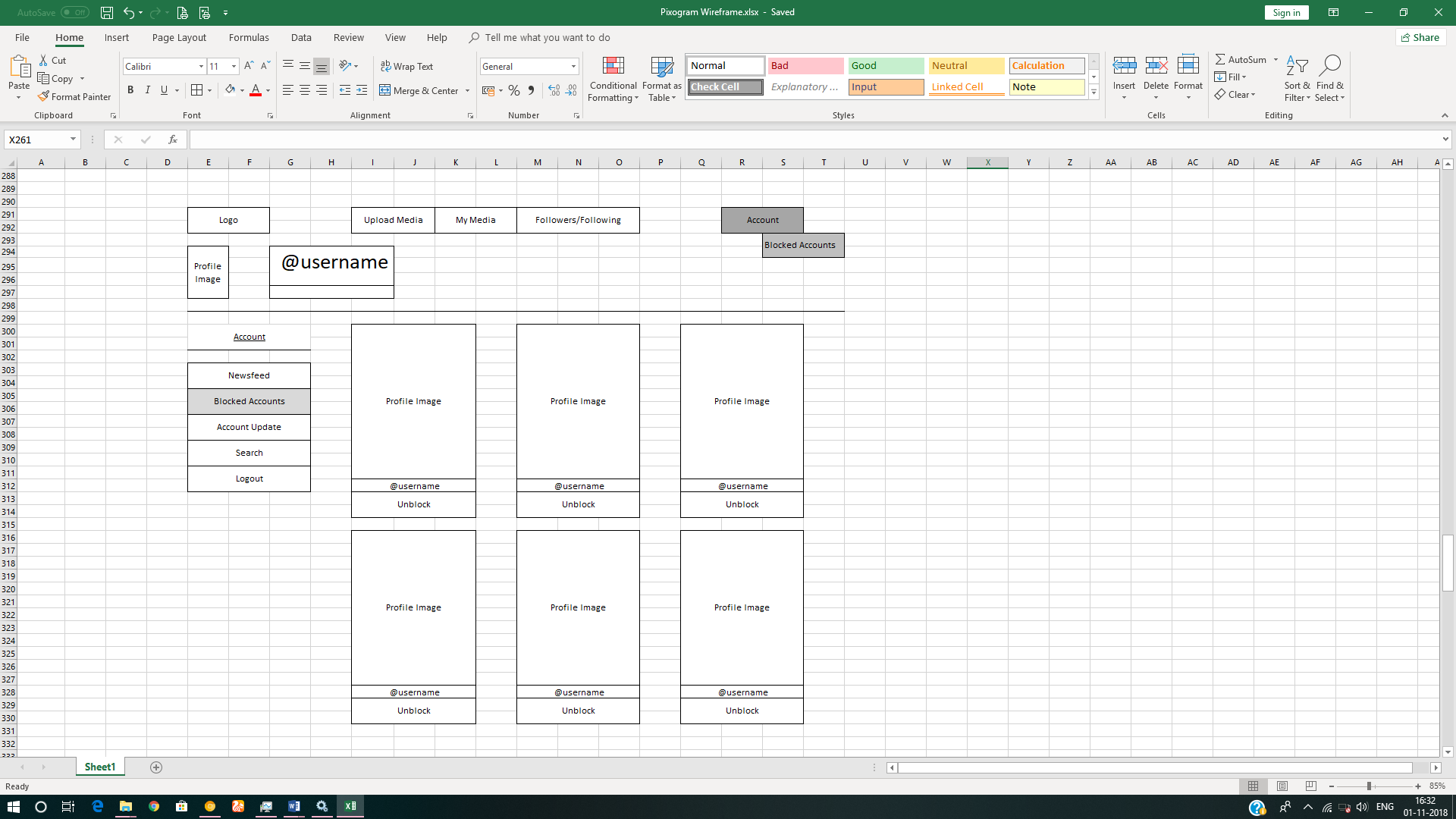


### Blocked Users Page

#### 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

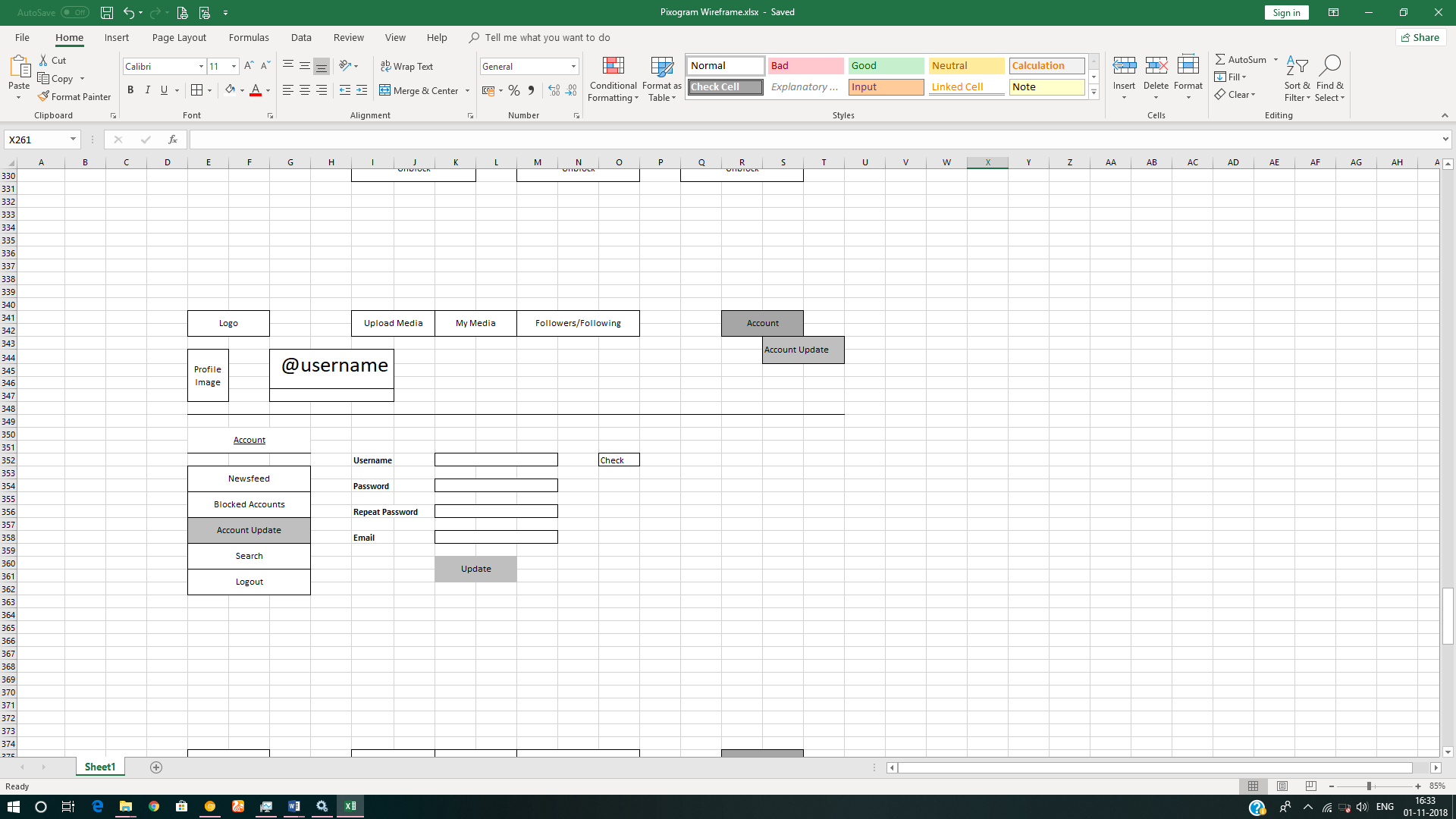


### Account Details Page

#### 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

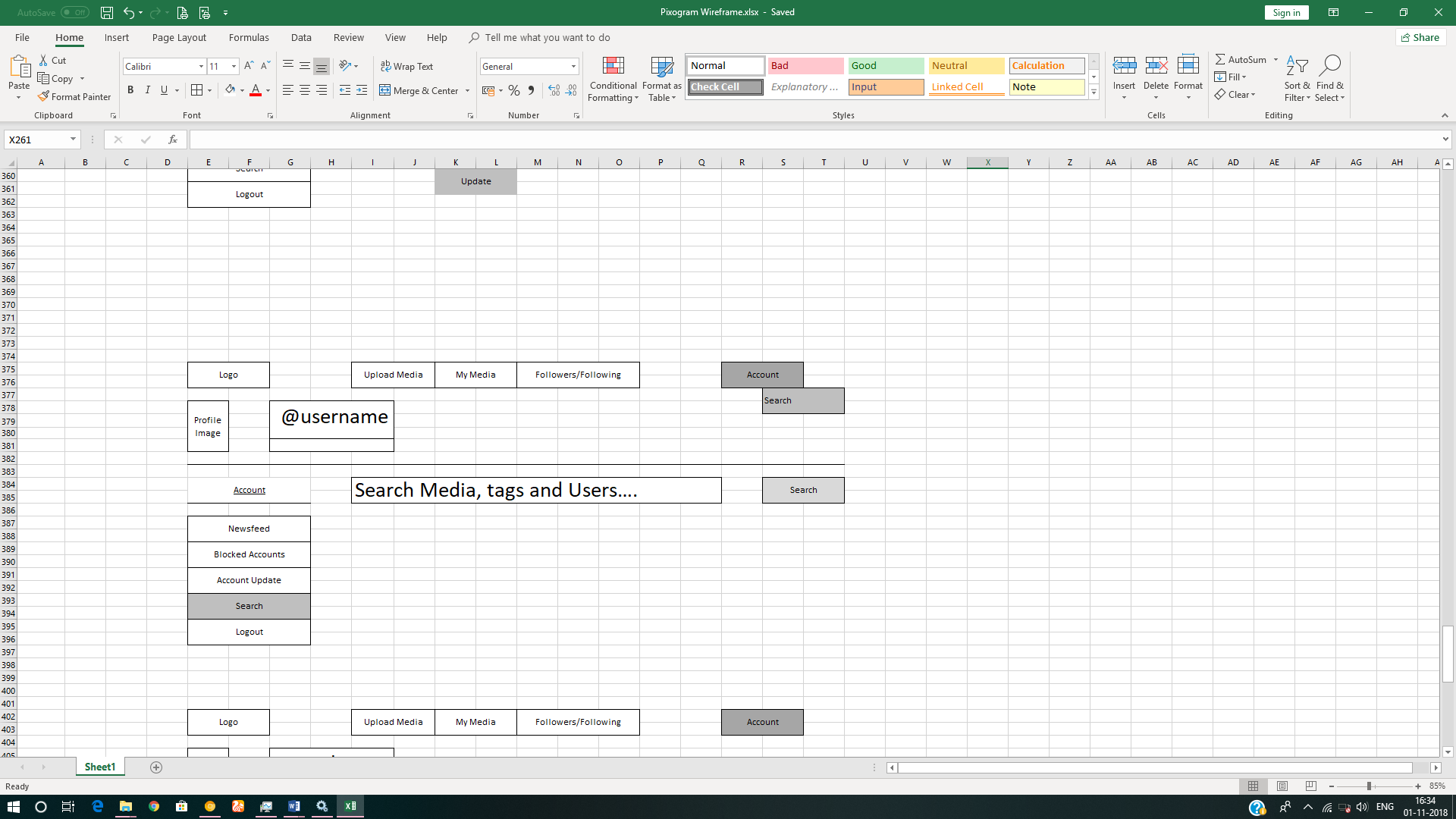


### Search Page

#### Search Page Requirement

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

#### Search Page Wireframe



# Methodology

## Agile

1. Mentor will ask you about daily progress as you start implementing responsive UI.
2. Communicate with your mentor via email; ideally daily as you develop the UI.
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 UI 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 localhost/cloud and integrate with cloud 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 – UI Layer Solution Development Environment

## UI Layer

|  |  |
| --- | --- |
| **Framework(s)/SDK/Libraries** | **Version** |
| HTML5 | - |
| CSS3 | - |
| Bootstrap/Material | - |
| GIT Basics | - |
| Jenkins Basics | - |
| Docker Basics | - |

## Editors

|  |  |
| --- | --- |
| **Name** | **Version** |
| Visual Studio Code | - |

# App UI Development Process (AGILE/SCRUM)

1. You must follow following process while creating UI
   1. \*\*Approval includes:
      1. Communicating with mentor via email to showcase the progress.
      2. Progress must be shared with mentor as each UI developer stage 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 UI implementation skills.
   2. Sketch:
      1. Use pen & paper to sketch the wireframe
      2. Capture it with camera
      3. Email to mentor
   3. Wireframe:
      1. Use spread sheet to prototype wireframe
   4. Bootstrap/Material Components Identification
      1. Identify the bootstrap/material components from the online component example library which you will use to develop the responsive UI.

# 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 on some http server.**
5. Implement the code using best design standards.
6. **The UI for the app should be multilingual.**
7. The **logo for the app should be in SVG format**. Use logo of your choice.
8. UI should be **responsive** across multiple devices.
9. It should be the **progressive web app** such that certain parts of the application are accessible in absence of connectivity.
10. **Dynamic functionality is not required.** Since it is UI Layer, use static content where required.

# Assessment Deliverables

1. Please make sure that your code does not have any compilation/console errors while being deployed using docker.
2. The final solution should consist of:
   1. Docker Image which can be executed to launch the http which in turn will launch UI front-end in default browser.
   2. Dump of command “git log”
   3. Jenkins report about number of times it pulled the code from GIT to create a production/minified version of UI solution.
   4. Docker File used to create docker image from Jenkins.
   5. Word document with step by step instructions on:
      1. how to run/deploy the project

# Other Full Stack Layers

## UI Layer – Just for Information (Applicable for Present Case Study)

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

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

|  |  |
| --- | --- |
| Angular or | 6 |
| React | 16 |
| Javascript & JQuery | - |
| Typescript | - |

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

|  |  |
| --- | --- |
| Java SE | 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 (Not 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 |