

**Pick Pic**

**Software Design Specification**

**2017 - 1 Human ICT Software Engineering | 2017.06.02**

|  |  |
| --- | --- |
| 20155652 | KangSoYeon |
| 20154019 | LeeYeWon |
| 20151722 | MyungSeKyo |
| 20152164 | LeeWonJun |
| 20151535 | ParkGunHoo |

Table of Contents

[1.0 System architecture 4](#_Toc484191502)

[1.1 Modules and interfaces 4](#_Toc484191503)

[1.2 Data description 5](#_Toc484191504)

[1.3 Design alternatives 5](#_Toc484191505)

[1.3.1 MainActivity 5](#_Toc484191506)

[1.3.2 SearchActivity 6](#_Toc484191507)

[1.3.3 GalleryActivity 6](#_Toc484191508)

[1.3.4 Synchronizer 6](#_Toc484191509)

[1.3.5 TagDBManager 7](#_Toc484191510)

[1.3.6 Server 7](#_Toc484191511)

[1.3.7 Server - ML 7](#_Toc484191512)

[1.3.8 Server – color extractor 7](#_Toc484191513)

[1.3.9 Server – color namer 7](#_Toc484191514)

[1.4 Assumption 7](#_Toc484191515)

[2.0 Diagrams 8](#_Toc484191516)

[2.1 Class diagrams 8](#_Toc484191517)

[2.1.1 front End – Main Activity 8](#_Toc484191518)

[2.1.2 front End – Gallery Activity 9](#_Toc484191519)

[2.1.3 front End – Search Activity 10](#_Toc484191520)

[2.1.4 back End 10](#_Toc484191521)

[2.2 Sequence diagram 11](#_Toc484191522)

[3.0 Process 12](#_Toc484191523)

[3.1 Risk assessment 12](#_Toc484191524)

[3.2 Project schedule 13](#_Toc484191525)

[3.3 Team structure 13](#_Toc484191526)

[3.4 Test plan 14](#_Toc484191527)

[3.4.1 Frontend 14](#_Toc484191528)

[3.4.2 Backend 15](#_Toc484191529)

[3.5 Documentation plan 17](#_Toc484191530)

[3.6 Coding style guidelines 18](#_Toc484191531)

[Use Javadoc Standard Comments 18](#_Toc484191532)

[Write Short Methods 18](#_Toc484191533)

[Define Fields in Standard Places 18](#_Toc484191534)

[Limit Variable Scope 19](#_Toc484191535)

[Order Import Statements 19](#_Toc484191536)

[Follow Field Naming Conventions 19](#_Toc484191537)

[Use Standard Brace Style 19](#_Toc484191538)

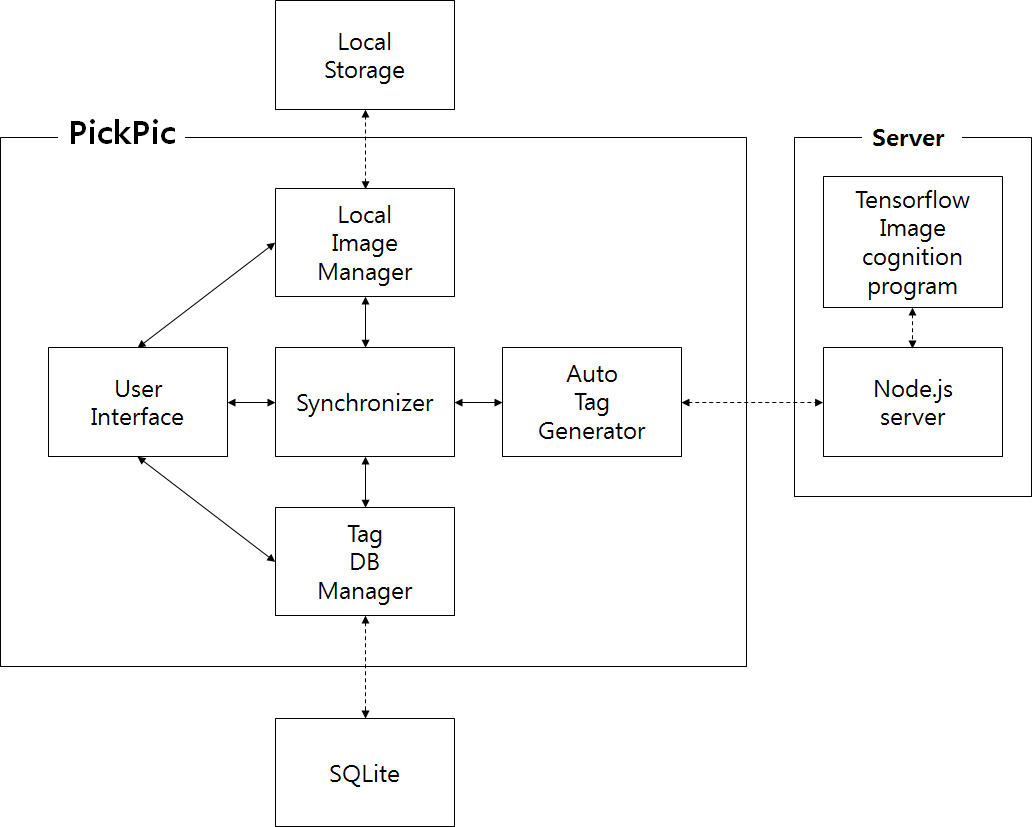
[Limit Line Length 19](#_Toc484191539)

[Use Standard Java Annotations 20](#_Toc484191540)

[Use TODO Comments 20](#_Toc484191541)

# System architecture

## 1.1 Modules and interfaces



Local Image Manager : this module provide easy way to access local storage images.

Synchronizer : this module synchronize between application DB and local storage each time it is run.

Tag DB Manager : this module manage the application DB that include tags.

Auto Tag Generator : this module is responsible for tag generation using API.

Local Storage : Internal storage space in local device.

MS Cognitive API : this API return tags when the image is transferred.

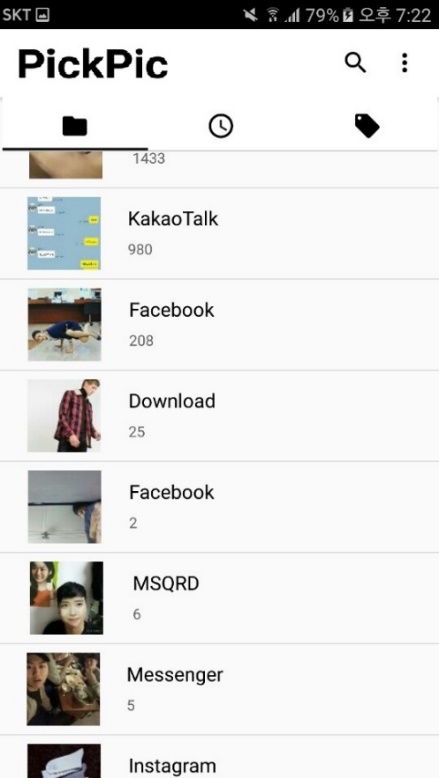
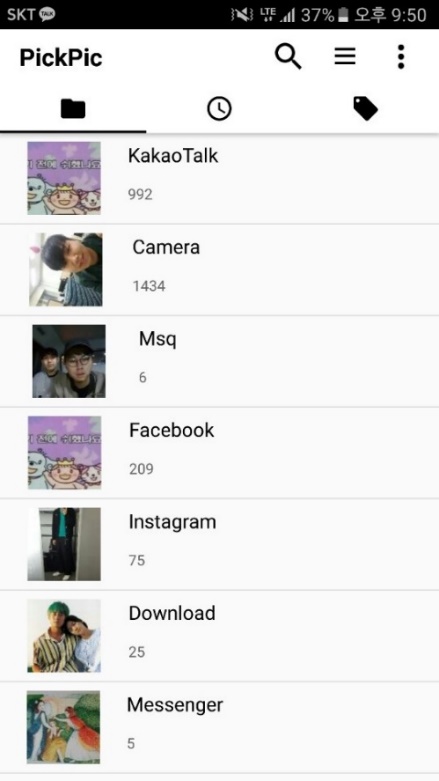
SQLite : compact DBMS in Android system.

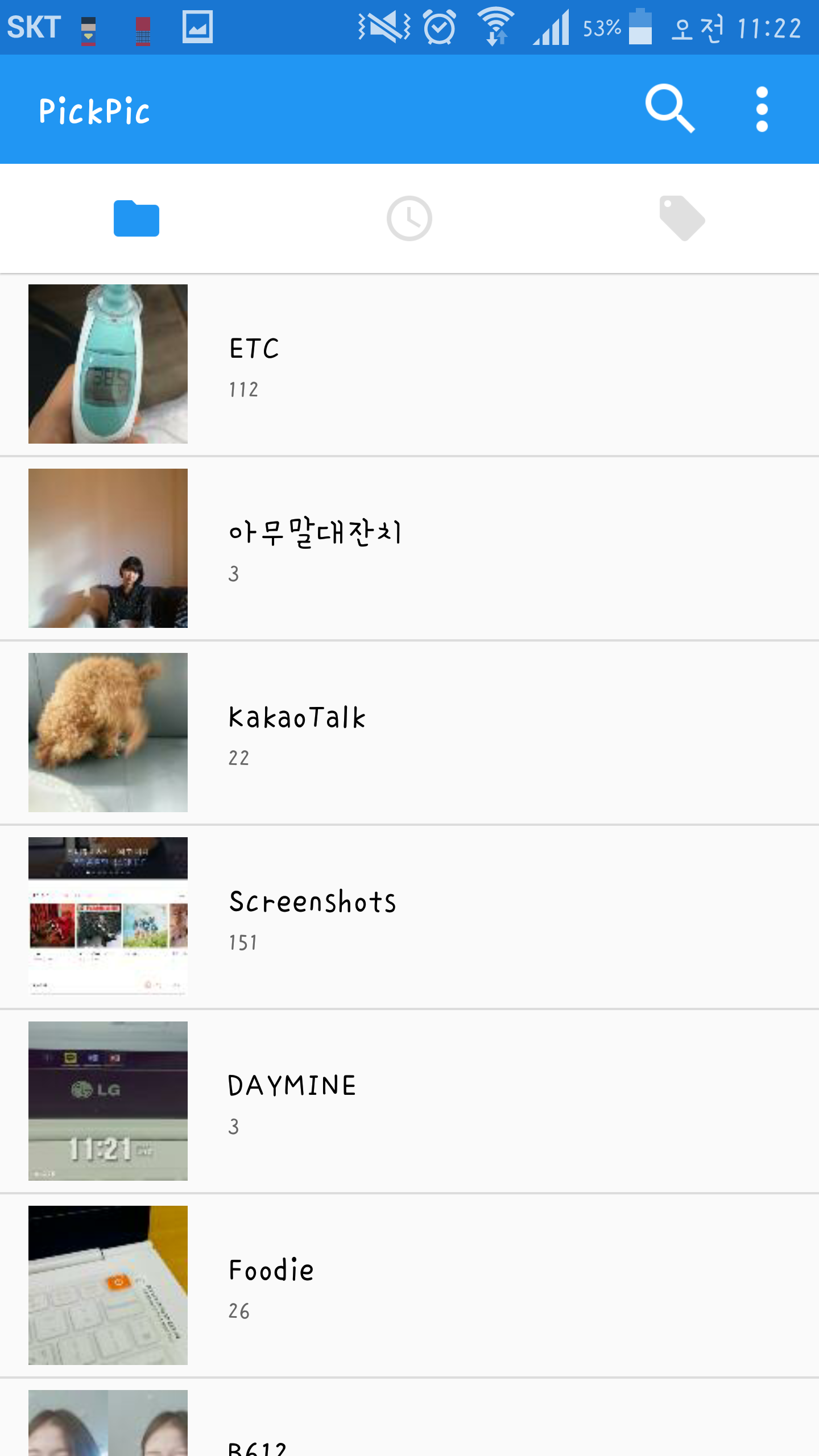
## 1.2 Data description

|  |  |  |
| --- | --- | --- |
| class | variable | note |
| DirectoryTabListViewItem | long thumbnail; | thumbnail ID of this directory |
| String name; | name of this directory |
| String id; | ID of this directory |
| int count | number of images |
| GalleryTagListItem | String tag | Tag value |
| RecommendTagListItem | String tag | Tag value |
| SearchRecyclerViewItem | TextView textView | Text that is now searched |
| ImageButton imageButton | Remove button of text |
| TagTabListViewItem | String tag | Tag value |
| GridViewItem | long thumbnail | Thumbnail ID |
| String path | Absolute path of image |

## 1.3 Design alternatives

### 1.3.1 MainActivity





We modified a bit in the early version of MainActivity. In the toolbar, we deleted a menu icon. Instead, we add additional functions in the search icon. When users click the search, then go to SearchActivity. And then performs the combined functions of two icons because of 2 reasons.

First is compaction. Having three icons can be complicated for viewing. The functions is not lost, but it combined into one icon.

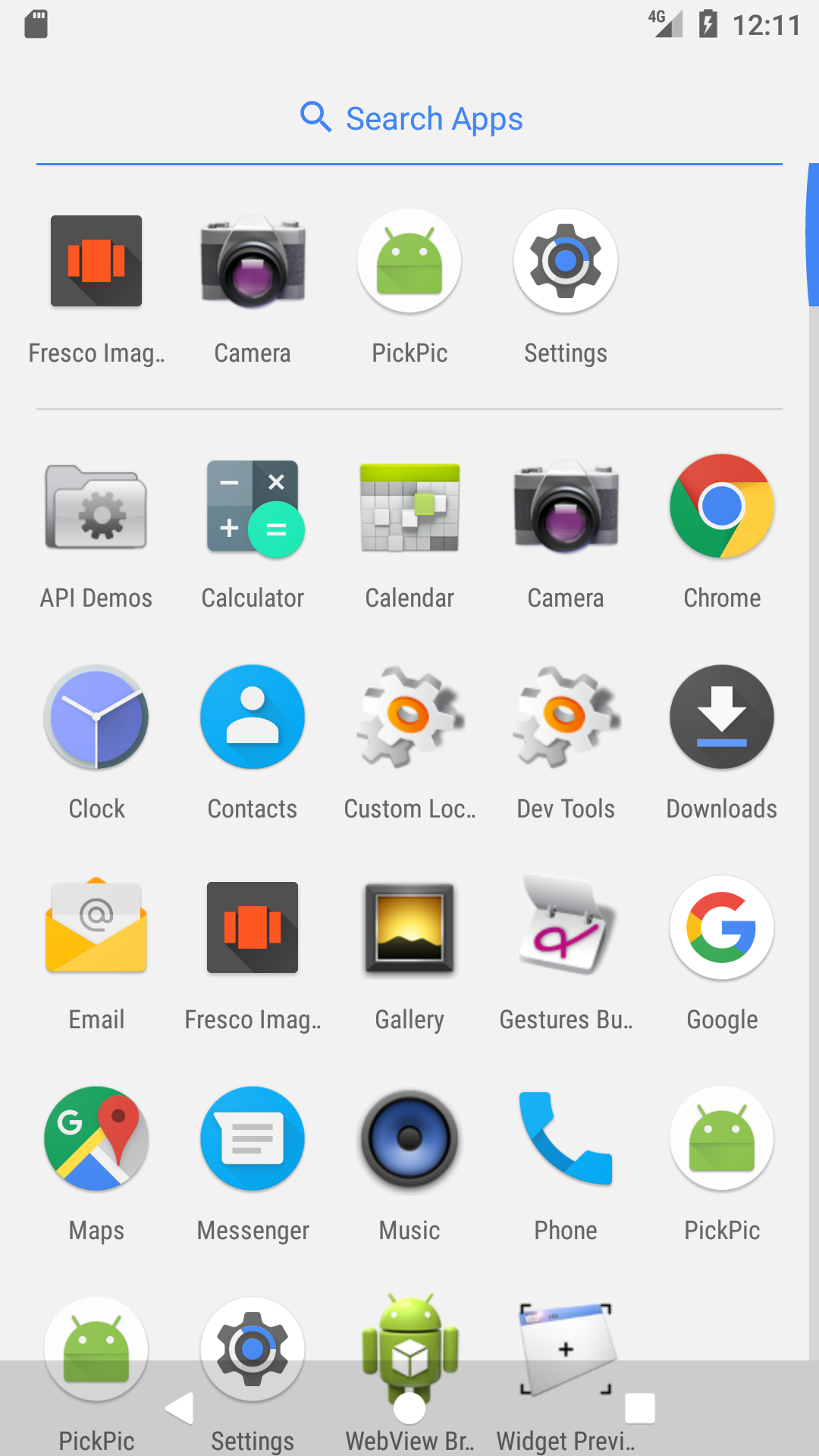
Second is easy to use. From the user’s point of view, two icons can be easier to use than three. And users mainly use search icon in the MainActivity, so menu icon is not likely to be needed for viewing.

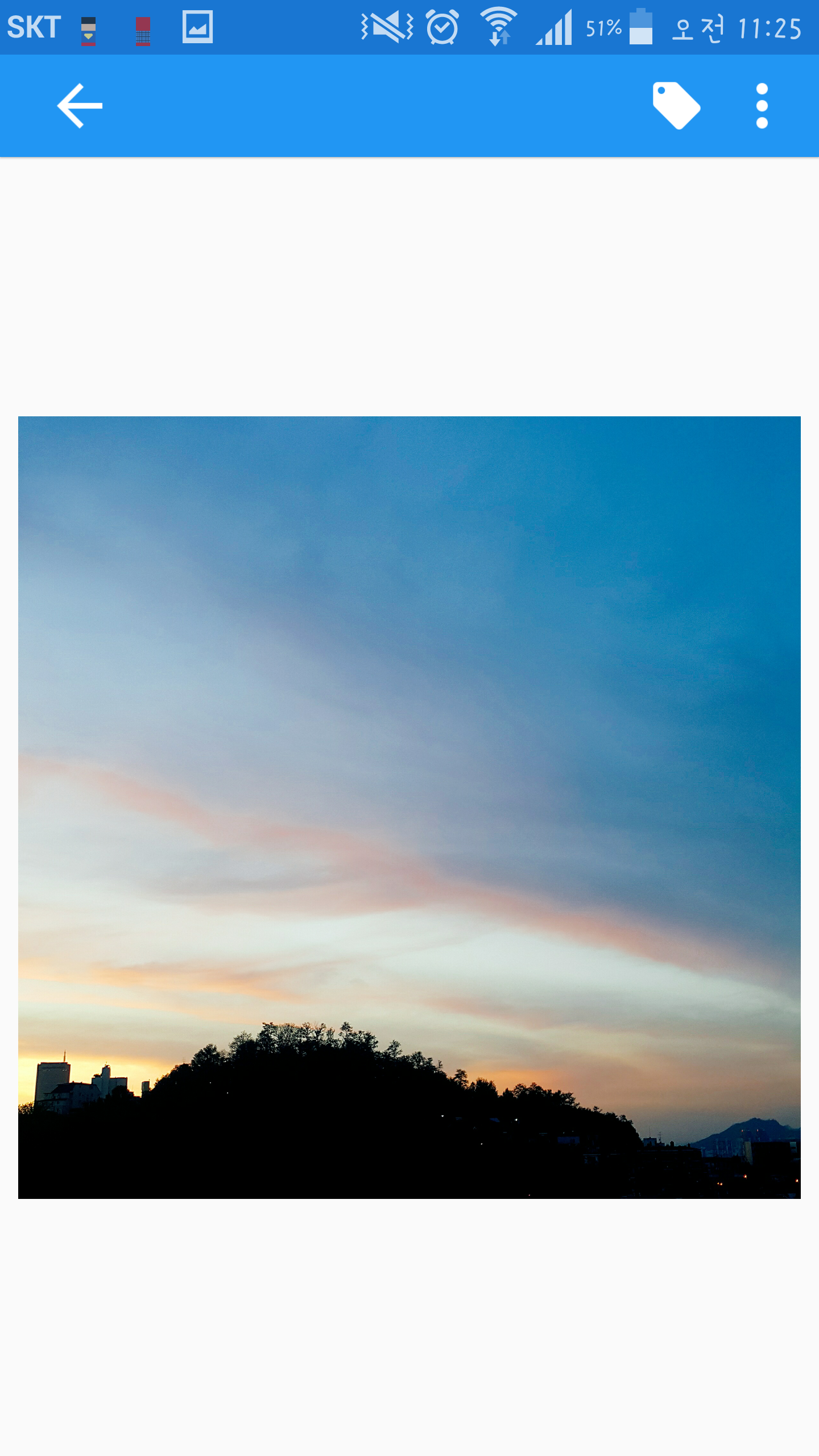
And finally we changed main dominent color black to blue, because the final application icons color is blue so for unity of color we changed color.

### 1.3.2 SearchActivity

We try to use SearchView, widget that provides a user interface for the user to enter a search query and submit a request to a search provider. But it is ambiguous to use and it makes hard to read a code. So we use EditText instead of SearchView. When we want to get text from User, We use getText(); and when the recommend tag is selected we use setText();. It reduce many ambiguity and it makes our code more readable and clearly.

### 1.3.3 GalleryActivity



In Gallery Activity, our developer make our own gallery. We found open-source gallery, Fresco Image Viewer. But we decided to use our own gallery because of dependency problem.

### 1.3.4 Synchronizer

We came up with two synchronizing methods.

The First method assumes that the record order of the media store and tag database is guaranteed.

Scan from the oldest image record to the latest image record. Delete the records in the tag database until the same record as the image record of each media store. Compare all the records in the tag database, add records from the remaining media repositories to the tag database, and request tags from the server..

The second method is to synchronize with the database's query statement. Delete records that do not exist in the media store from the tag database. Insert a record that does not exist in the tag database. Requests tags for records that do not receive tags from the tag database.

### 1.3.5 TagDBManager

When mapping an image to Tag DB and storing it, we can write the key uri or use the absolute path. Using uri is convenient when loading thumbnails Using an absolute path is convenient for sending images to the server. So, when we map one image to the DB, we decided to write the absolute path as the key.

### 1.3.6 Server

We planned to use MS cognitive API but canceled it because of dependency problem. We want to make completely independent application. So, we decided to use our own server.

### 1.3.7 Server - ML

We used inception v3, but we can create object recognition models directly or use many shared object recognition models.

### 1.3.8 Server – color extractor

We used dominant-color module in node js, but we can use other color extractor modules such as [img-color-extractor](https://www.npmjs.com/package/img-color-extractor) and so on.

### 1.3.9 Server – color namer

We used color-namer's html format as a name matching module for color numbers, but there are many other options. (basic, ntc, pantone, roygbiv, x11)

## 1.4 Assumption

Client running environment = android smart phone (With Target SDK = 22, Minimum SDK = 15)

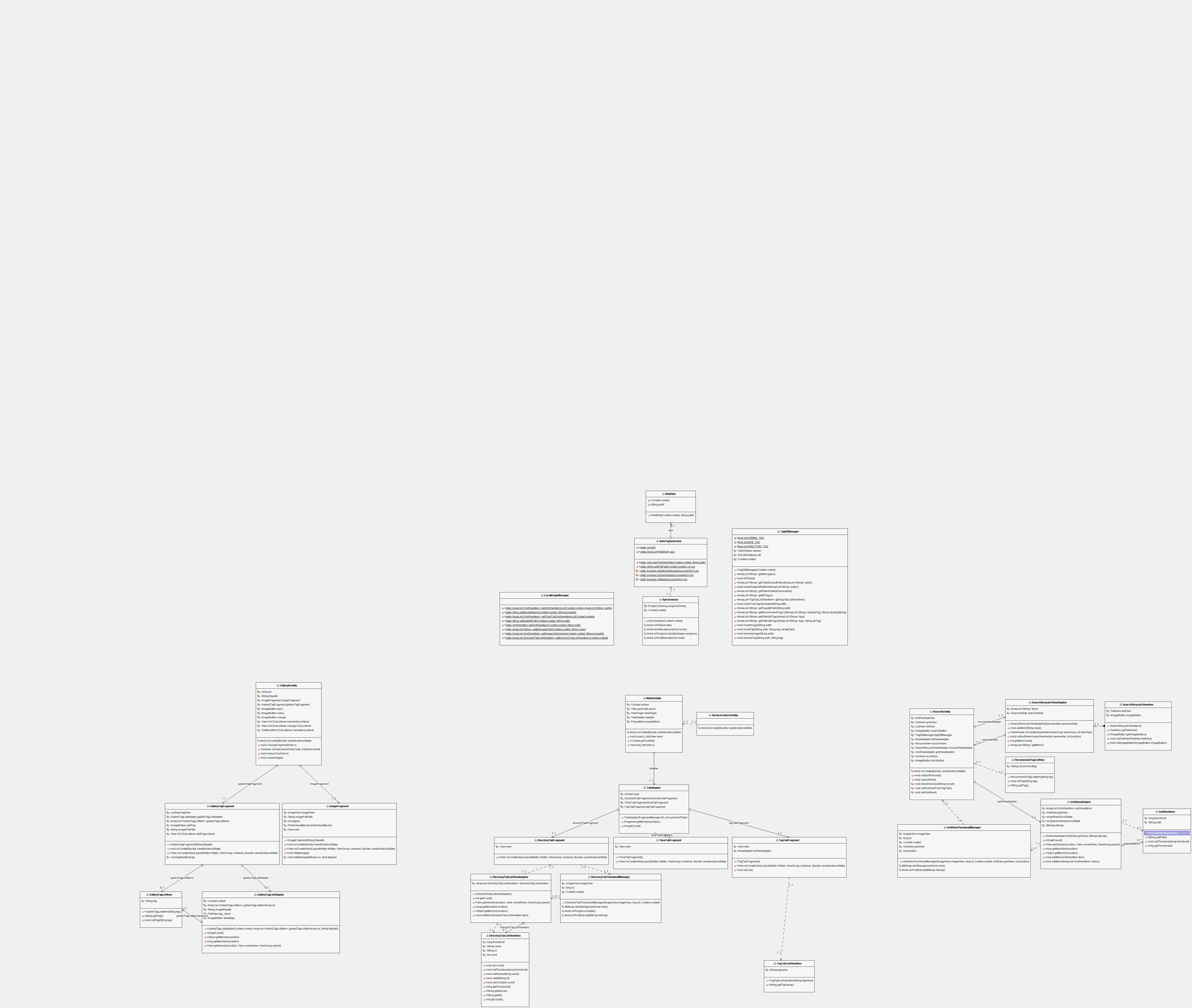
Server running environment = We support Windows OS, but we recommend Ubuntu OS.(With node js, tensorflow, tesseract-ocr, python)

# Diagrams

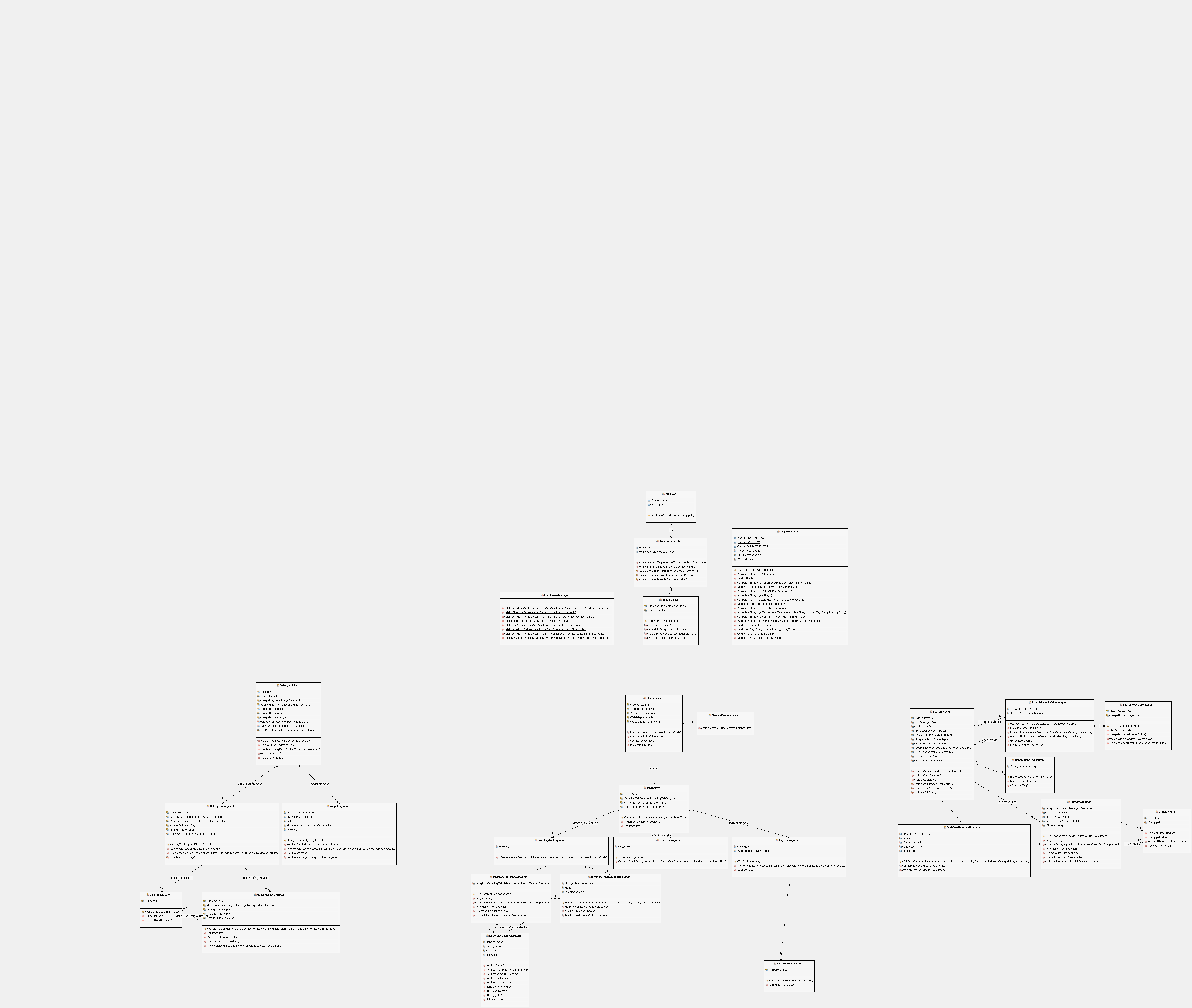
## 2.1 Class diagrams

### C:\Users\Arsene holmes\AppData\Local\Microsoft\Windows\INetCacheContent.Word\g.png2.1.1 front End – Main Activity

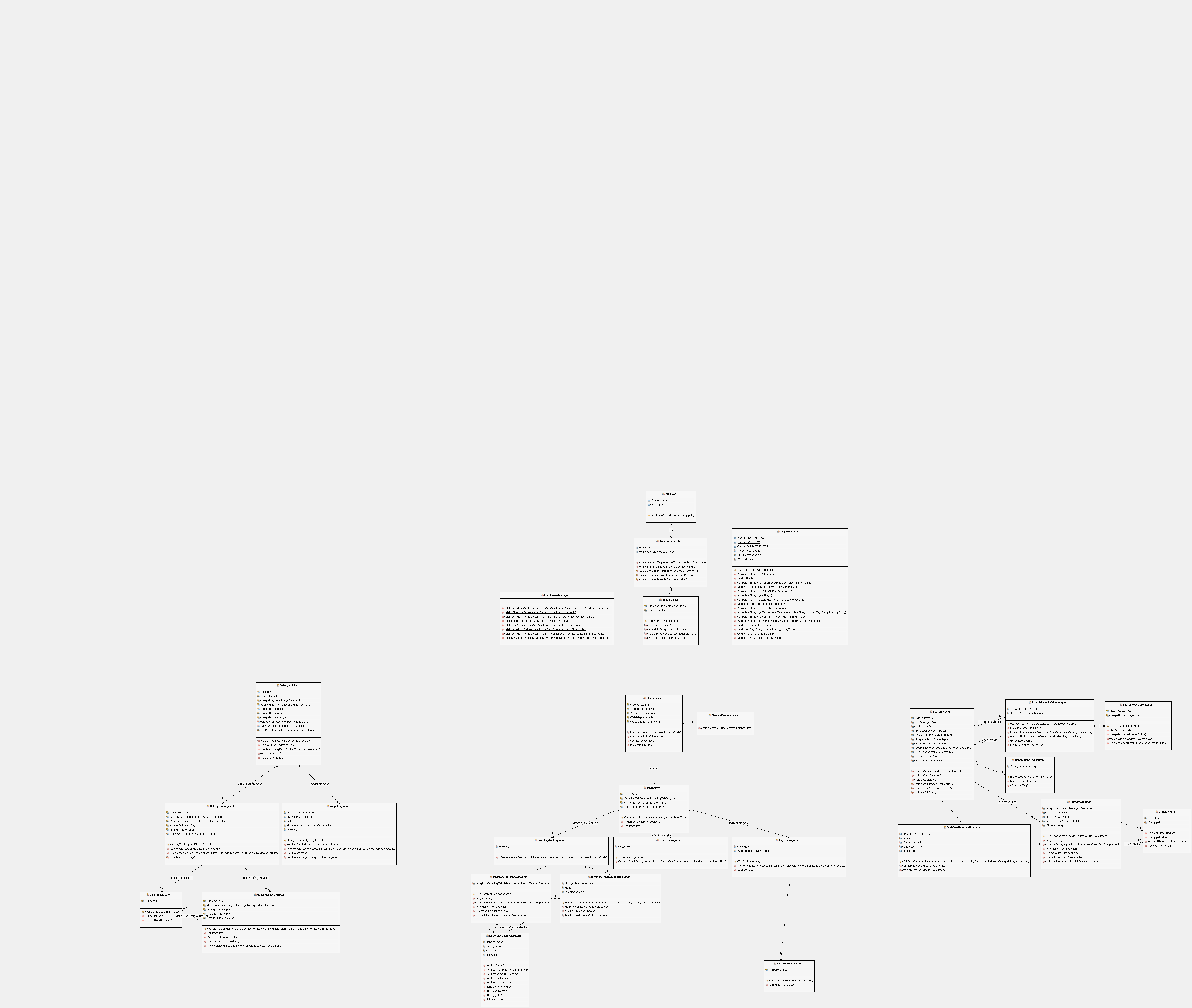
### 2.1.2 front End – Gallery Activity



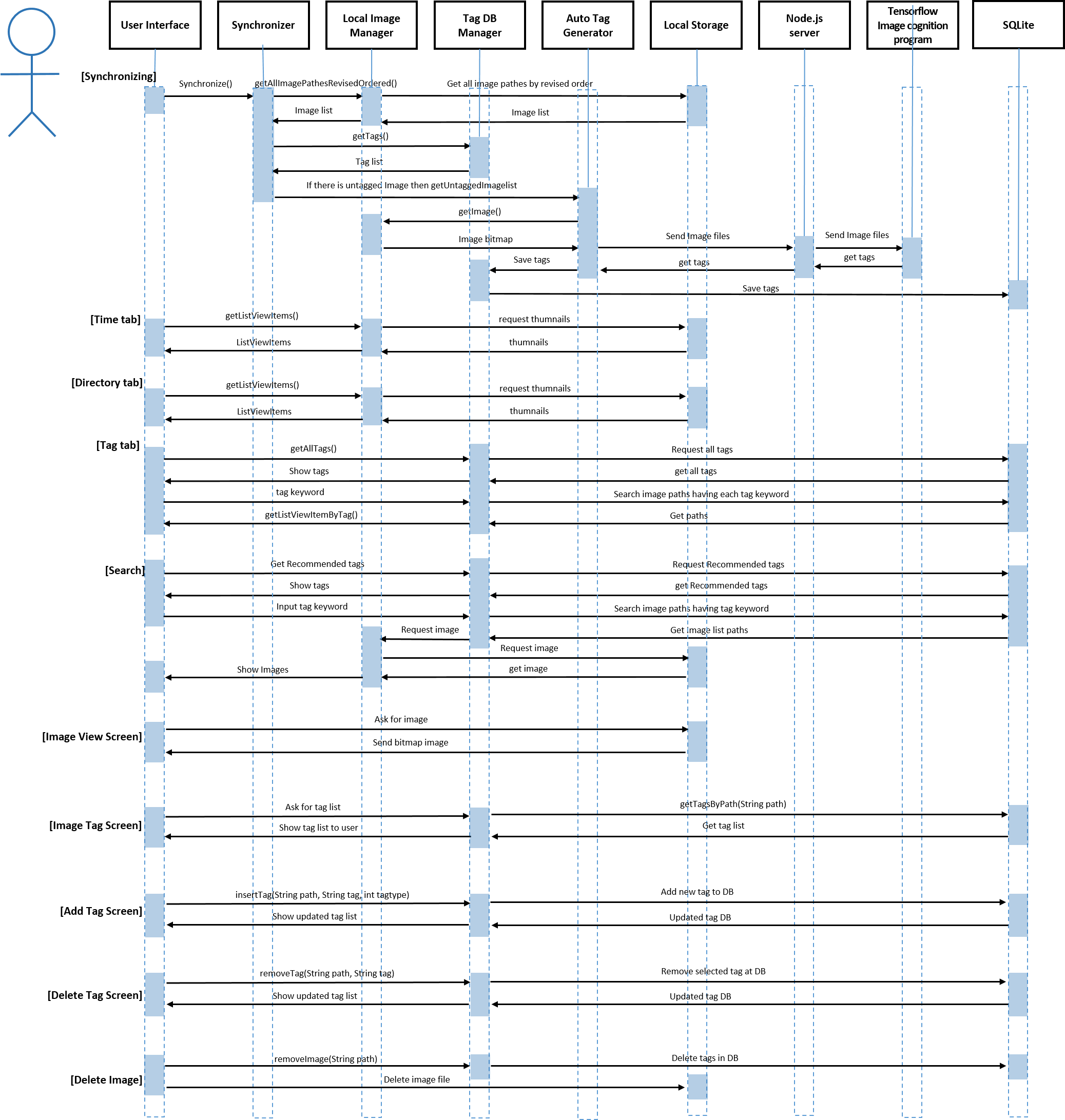
### 2.1.3 front End – Search Activity



### 2.1.4 back End



## 2.2 Sequence diagram

****

# Process

## 3.1 Risk assessment

Likelihood : frequent / infrequent , Severity : low / medium / high

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No | Activity | Hazard | Risk for whom | Likelihood | Severity | Possible measures |
| 1 | User takes lots of image. | The application requires synchronizing which needs some time, so users could feel uncomfortable. | user | frequent | low |  |
| 2 | The application needs synchronizing. | The application needs to connected to server when the application get tags. In this process, data fee will cost too much unless user uses Wi-Fi. | user | frequent | medium | Recommend using Wi-Fi. |
| 3 | User want to add more tags to an image. | Only 30 tags can be tagged because we limited it. So if there exist 30 tags, no more tags can be added. | user | infrequent | low | Recommend deleting unnecessary tags. |
| 4 | User fail to find expected image. | The tag user searched cannot tagged in an image. So, user cannot find the image expected. | user | infrequent | low | Recommend search different tag. |
| 5 | User is not satisfied with the tag received automatically | We develop our own server and the server learned image cognitive ability through machine learning. But due to time and cost problem, sufficient acquisition and learning are difficult. The learning model may not satisfy the user. | developer | infrequent | high | The server can learn the tensor flow model flexibly. Solve problem by earning and getting the data. |
| 6 | Too many user needs synchronize at the same time. | The performance of the currently running server computer is poor, so image analysis that can be processed at once is limited. | developer | infrequent | high | Have better server |

## 3.2 Project schedule

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Functions | | | 1st  (03.21~  04.01) | | 2nd  (04.02~  04.17) | | 3rd  (04.17~  04.26) | | 4th  (04.27~  05.05) | | 5th  (05.06~  05.13) | | 6th  (05.14~  05.21) | | 7th  (05.22~  05.29) | | 8th  (05.30~  06.02) | |
| Design System Architecture | | Idea Develop |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Service Scenario |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UI Design |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Use-case |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Scrum | Select | Scrum Meeting |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Develop | Coding |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Refactoring |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Review | Client Meeting |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Project closure | | Documentation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| All members  Front-End Developer  Back-End Developer | | | | | | | | | | | | | | | | | | | |

**April 30** Make a prototype of the application.

1. Save the tags returned from the Server
2. Coding base backend function

**May 30** Complete detail functions

1. Sorting images – by tags, by dates, by directory
2. Make application’s own gallery
3. Share function
4. Add tag directly by user
5. Searching function
6. Make tutorial for user

**June 2** Complete final application

## 3.3 Team structure

Backend : 이원준, 명세교 (Pair programming)

Frontend : 강소연(Search Activity), 이예원(GalleryActivity) ,박건후(MainActivity)

## 3.4 Test plan

### 3.4.1 Frontend

|  |  |  |
| --- | --- | --- |
| Activity | Action | Function Specification |
| Main | DirectoryTab | If you run the application at first time or click directory tab, then directories are shown in order of latest. If you click one of the directories, then current activity will go to ‘SearchActivity’. |
| DateTab | If you click or swap to date tab, then pictures in your device are shown in order of latest. If you click one of the thumbnails, then current activity will go to ‘GalleryActivity’. |
| TagTab | If you click or swap to tag tab, then all tags in your pictures are shown in order of frequency. If you click one of the tags, then current activity will go to ‘SearchActivity’. |
| SearchIcon | If you click a search icon, then current activity will go to ‘SearchActivity’. |
| SettingIcon | If you click a setting icon, then popup menu will be shown. |
| UISettingFrame | If you click ‘Synchronize’, then synchronize will manually run. Else if you click ‘Delete all tags’, then all tags in your application and pictures will be deleted. Else if you click ‘How to use’, then current activity will go to ‘ManualActivity’ (manual images). Else if you click ‘Service center’, then current activity will go to ‘ServiceCenterActivity’(developers’ contact and application’s web site address). |
| SearchActivity | Automatic completion | If you start texting for search, the recommended tag list is gone and automatic completion list is replaced it. If there is no similar tag, empty tag recommend list will shown. If last or first Text is similar then show the tag list. |
| Search | When searched tag is found in all tag list, show the image which have the tag and when searched tag is not found in all tag list show nothing |
| Search repeatedly | Search again in searched list and there are images that have both tags common than show the images that have both tags common. But if not, show nothing |
| Delete searched tag | There is few tags above and click X button right after the tag than the tag will be deleted. When there is no tag above anymore than return to search view. |
| Gallery | Change Fragment | C:\Users\Arsene holmes\AppData\Local\Microsoft\Windows\INetCacheContent.Word\그림1.pngWhen click button, Convert to GalleryTagFragment if current screen is ImageFragment. Convert to ImageFragment if current screen is Gallery Tag Fragment. |
| Click option menu | User can see two menus, ‘Rotate’ and ‘Share”. When clicked share button, user can see the applications that can share. When clicked rotate button, user can see the image rotates clockwise. |
| Rotate image | If the image is  1) square, it rotates.  2) horizontally long or vertically long, it rotates and resize to fit the screen. |
| Click back button | Return to previous page. If current screen is GalleryTagFragment, return to ImageFragment. If current screen is ImageFragment, return to Gridview. |
| Add tag | In GalleryTagFragment, user can add a new tag by clicking add button. User can see the popup window that asking what tag user want to add. If user clicks ADD button, new tag is added and appears in the list. If user clicks CANCEL button, nothing is changed |
| Delete tag | In GalleryTagFragment, user can delete tag that he/she doesn’t like. Also, user can see the popup window that asking really want to delete. If user clicks OK button, the tag is deleted and disappears from the list. If user clicks CANCEL, nothing is changed. |

.

### 3.4.2 Backend

|  |  |  |
| --- | --- | --- |
| Class | Function | Function Specification |
| AutoTagGenerator | autoTagGenerate | If server returns tags, tag is generated successfully.  Else if server doesn’t returns tags, tag is not generated.  Take the path and send it to the server to get the tag and confirm it.  When no tag is generated, there should be some problem  1) The server has timed out.  2) No internet connection  3) Unsupported file format. |
| LocalImageManager | getGridViewItemList | If There are images in the context, return gridview list. Else if There are no images in the context, return nothing |
| getBucketName | If bucket id is correct, get directory name. If bucket id is incorrect, return empty string. |
| getTimeTabGirdViewItemList | If there are images in the context, return gridview item list. Else if there are no images in the context, return nothing. |
| getDateByPath | If there is an image with that path, return date. Else if there is no image with that path, return empty string. |
| getGridViewItem | If there is an image with that path, return gridview. Else if there is no image with, return nothing. |
| getAllImagePath | If directory has images, returns the path of all images in the context as an ArrayList<String>. Else if directory has no images, return empty list |
| getImagesInDirectory | If each directory has images, get images for each directory. Else if directory which have no images, does not considered. |
| getDirectoryTabListViewItem | If directory which has some images, show all directory that has image in listview. Else if directory which have no image, does not considered. |
| Synchronizer | Synchronizing | When app is started compare local images and tagDBManager tags. If there are images in local but following tag is not in TagDBManager then Get the images from local and add DB the following tags.. If all images in local are in TagDBManager then compare and do nothing.. If there are tags in TagDBManager but following images are not in local then find the unpaired tags from tagDBManager and delete it. |
| TagDBManager | getAllImages | If image paths are in TagDBManager than All Image paths in TagDBManager is gotten . If no image path in TagDBManager. Than empty arraylist is gotten. |
| initTable | Erase the Table in TagDBManager and rebuild |
| getToBeErasedPath | When there are images deleted but the path is remain in TagDBManager than delete remain image path. |
| insertImagesifnotExist | There are new images in local that does not exist in TagDBManager than Insert image path to TagDBManger. If there are no new images in local that does not exist in TagDBManager this function is not working. |
| getPathsNotAutoGenerated | If there are images which don’t have following tags than Return the image paths which don’t have following tags. If there are no images which don’t have following tags than return empty arraylist. |
| getAllTags | If there are some tags in TagDBManager than Return repetition removal tags in string arraylist.. If No tags in TagDBManager then return empty string arraylist. |
| getTagTabListViewItem | If there are some tags in TagDBManager then return repetition removal tags in listviewitem arraylist. If no tags in TagDBManager then return empty listviewitem arraylist. |
| makeTrueTagGenerated | If we want to know the image has tags return image has tags or not. |
| getTagsByPath | If there are images with following tags then get tags in String ArrayList. If there is no tags in the images then return empty string Arraylist. |
| getRecommendTagList | Get text input from user. If the text is similar with saved tags then return the tags that similar with users input text. If the text is not similar with saved tags then return no tags |
| getPathsByTags | If there are tags you want to find then return path string arraylist that include tags. If there is no paths following tags then return empty string arraylist. |
| insertImage | There are images you want to insert then insert image path to TagDBManager |
| insertTag | There are tags you want to insert then insert tags to TagDBManager |
| removeImage | There are images you want to remove then remove image path in TagDBManager |
| removeTag | There are tags you want to remove then remove tags in TagDBManager |

## 3.5 Documentation plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Documents | Purpose | Audience | Output media | Writer | Start | Completion |
| Project proposal | Propose the project that would be a term project | All students in class and the professor | ppt | SoYeon, Yewon | 170313 | 170314 |
| Software requirement specification | Establish a solid definition of the project | the Professor | docx | SoYeon, Yewon, GunHoo, SeKyo, WonJun | 170328 | 170405 |
| Software design specification | Describe all the details about the project | the Professor | docx | SoYeon, Yewon, GunHoo, SeKyo, WonJun | 170414 | 170426 |
| Interim presentation | Present the summary of requirements and planning | All students in class and the professor | ppt | SoYeon, Yewon | 170414 | 170426 |
| Zero feature | Describe the detail about the initial version of the application | the Professor | docx | SoYeon, Yewon, GunHoo, SeKyo, WonJun | 170427 | 170512 |
| Beta | Describe the detail about the revised version of the application | the Professor | docx | SoYeon, Yewon, GunHoo, SeKyo, WonJun | 170513 | 170602 |
| Final presentation | Present final version of the application | All students in class and the professor | ppt | SoYeon, Yewon | 170513 | 170602 |
| Individual report  (#1 ~ #3) | Show individual progress in the project | the Professor | docx | individual team member | every week | |

## 3.6 Coding style guidelines

**Use Javadoc Standard Comments**

Every file should have a copyright statement at the top, followed by package and import statements (each block separated by a blank line) and finally the class or interface declaration. In the Javadoc comments, describe what the class or interface does.

Every class and nontrivial public method you write must contain a Javadoc comment with at least one sentence describing what the class or method does. This sentence should start with a third person descriptive verb.

### Write Short Methods

When feasible, keep methods small and focused. We recognize that long methods are sometimes appropriate, so no hard limit is placed on method length. If a method exceeds 40 lines or so, think about whether it can be broken up without harming the structure of the program.

### Define Fields in Standard Places

Define fields either at the top of the file or immediately before the methods that use them.

### Limit Variable Scope

Keep the scope of local variables to a minimum. By doing so, you increase the readability and maintainability of your code and reduce the likelihood of error. Each variable should be declared in the innermost block that encloses all uses of the variable.

Local variables should be declared at the point they are first used. Nearly every local variable declaration should contain an initializer. If you don't yet have enough information to initialize a variable sensibly, postpone the declaration until you do.

### Order Import Statements

The ordering of import statements is:

1. Android imports
2. Imports from third parties (com, junit, net, org)
3. java and javax

### Follow Field Naming Conventions

* Non-public, non-static field names start with m.
* Static field names start with s.
* Other fields start with a lower case letter.
* Public static final fields (constants) are ALL\_CAPS\_WITH\_UNDERSCORES

### Use Standard Brace Style

Braces do not go on their own line; they go on the same line as the code before them

We require braces around the statements for a conditional. Exception: If the entire conditional (the condition and the body) fit on one line, you may (but are not obligated to) put it all on one line.

### Limit Line Length

Each line of text in your code should be at most 100 characters long. While much discussion has surrounded this rule, the decision remains that 100 characters is the maximum with the following exceptions:

* If a comment line contains an example command or a literal URL longer than 100 characters, that line may be longer than 100 characters for ease of cut and paste.
* Import lines can go over the limit because humans rarely see them (this also simplifies tool writing).

### Use Standard Java Annotations

Annotations should precede other modifiers for the same language element. Simple marker annotations (e.g. @Override) can be listed on the same line with the language element. If there are multiple annotations, or parameterized annotations, they should each be listed one-per-line in alphabetical order.

### Use TODO Comments

Use TODO comments for code that is temporary, a short-term solution, or good-enough but not perfect. TODOs should include the string TODO in all caps, followed by a colon:

(reference : https://source.android.com/source/code-style)