L.A. City Bureau of Engineering

**Software Design Document**

**LAPA**

Bernard Bollinger III

Phillip Han

Brian Kan

Sharon Lake

Araceli Lopez

Patricia Luz

Kaila Mayho

Version 1

September 23, 2017

**Table of Contents 2**

Table of Contents

**Revision History 3**

1. **Introduction 4**
   1. Purpose 4
   2. Document Conventions 4
   3. Intended Audience and Reading Suggestions 4
   4. System Overview 4
2. **Assumptions and Dependencies 5**
   1. Dependencies 5
   2. General Constraints 5
   3. Goals and Guidelines 5
   4. Development Method 5
   5. Choice of which specific products used 5
3. **System Architecture 6**
   1. System Structure 6
      1. DFD Level 0 6
      2. DFD Level 1 7
4. **Detailed System Design 8**
   1. Dashboard Module 8
      1. Responsibilities 8
   2. Tags Module 9
      1. Responsibilities 9
   3. Camera Module 10
      1. Responsibilities 10
   4. History Module 11
      1. Responsibilities 11
   5. Settings Module 12
      1. Responsibilities 12
5. **Detailed Lower level Component Design 13**
   1. Dashboard Class 13
   2. Tags Class 15
   3. Camera Class 16
   4. History Class 18
   5. Settings Class 17
6. **Database Design 20**

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 1. Introduction

## 1.1 Purpose

The purpose of this document is three -fold:

1. Completely define the design for the **L.A.P.A. – Section 4.0**.

b) Define modules for the **L.A.P.A. – Section 5.0**.

c) Completely define the Test Plan for the **L.A.P.A.** **– Section 6.0**.

The complete definition of  **L.A.P.A.** provides the source inputs for the development of the subsequent supporting software subsystems documents.

The scope of this document includes the following:

* All functional and nonfunctional requirements on the **L.A.P.A.** are captured.
* The objectives are organized by key **L.A.P.A.** functional units shown on the Level 1 DFD. The Level 1 DFD is shown on page

## 1.2 Document Conventions

[NEED TO BE DONE ]

## 1.3 Intended Audience and Reading Suggestions

Developers, project managers, testers and documentation writers. The SRS contains an overall description of the project as well as the specific functional requirements of the app itself.

## 1.4 System Overview

The final product is to have a functional application where multiple photos can be taken and labeled Before and After. Also the project will be able to mark the primary photo along with give the photos the appropriate categories to be organized within the database.

# 2. Assumptions and Dependencies

## 2.1 Dependencies

* The phone is running at least KitKat(4.0) Android.
* The phone is capable of internet access.
* The phone has a functioning browser.
* The phone is equipped with a camera.

## 2.2 General Constraints

* None

## 2.3 Goals and Guidelines

* A fully functional application
* Change the User Interface
* Add before and After Photo option along with categories

## 2.4 Development Method

* Android Studio
* Continuing last year’s project

## 2.5 Choice of which specific products used

* Android Studio
* Azure Development
* Visual Studio

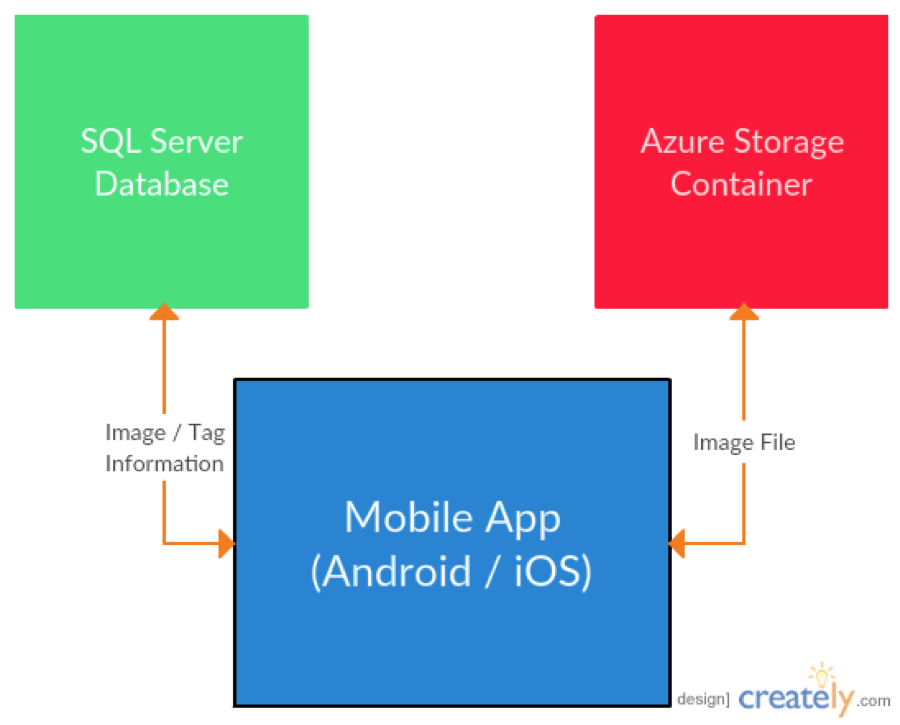
# 3. System Architecture

## 3.1 System Structure

The LAPA system is broken up into two major components: a client-side Android application and a server-side Azure database.

The client-side application is also separated into two parts: the functional component (written in Java) and the graphical component (written in XML). The functional component forms the core of LAPA. It receives user input and stores it in an XML file. The graphical component, as the name implies, is simply the graphical user interface. It provides all of the buttons, text-boxes, and other on-screen elements which allow the user to access all of the features provided by the application.

The server-side component of LAPA is comprised of Azure database which provides centralized storage for synchronized data. There server application receives serialized data from Android devices and parses it into useful information. This data is then stored in the database and subsequently synchronized to other devices in the same group.

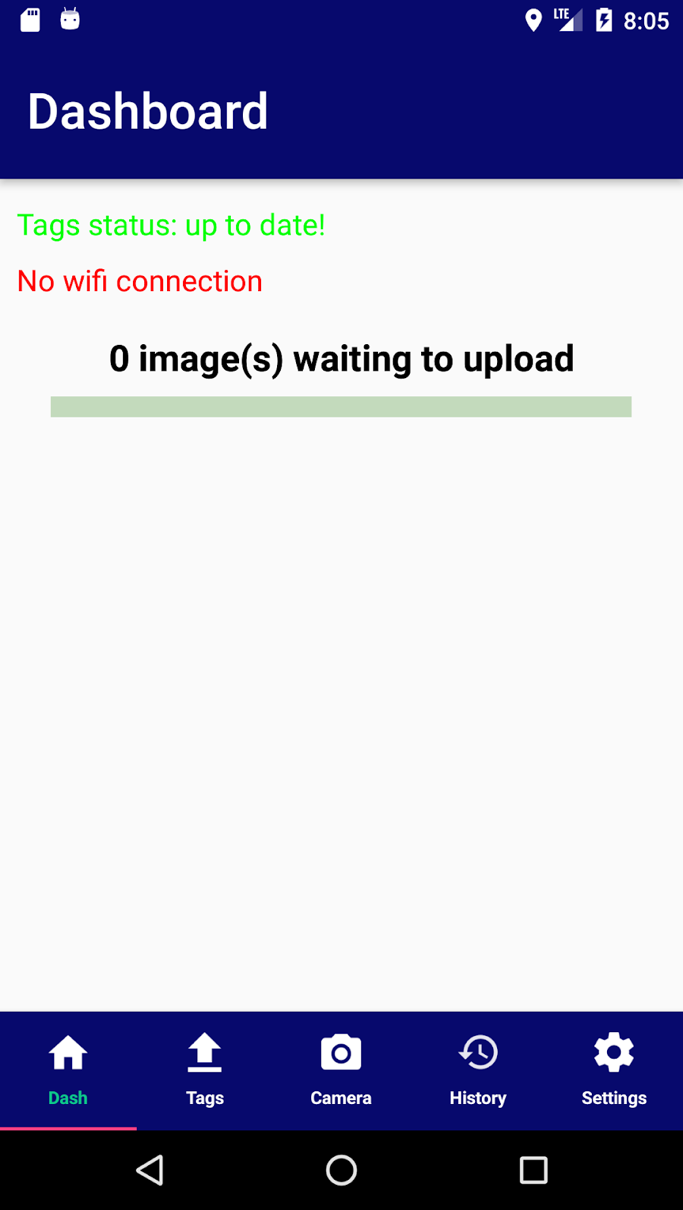


**DFD Level 0**

|  |  |
| --- | --- |
| DFD LEVEL 1DFDLevel1.png |  |

# 4. Detailed System Design

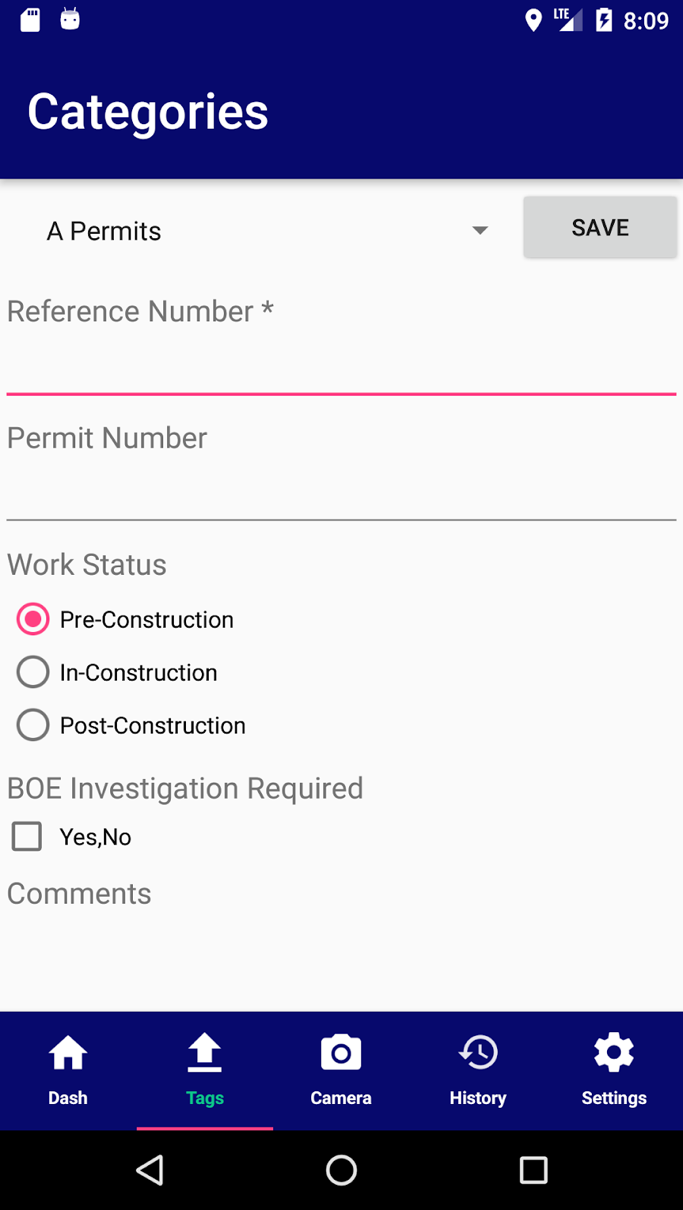
## 4.1  Dashboard Module

****

### 4.1.1 Responsibilities

The Dashboard main responsibilities is to give the user the number of photos in queue waiting  to be uploaded to the database. In the Dashboard the user is able to see the progress of the image uploads, the status of the user WI-FI connection and the tag status.

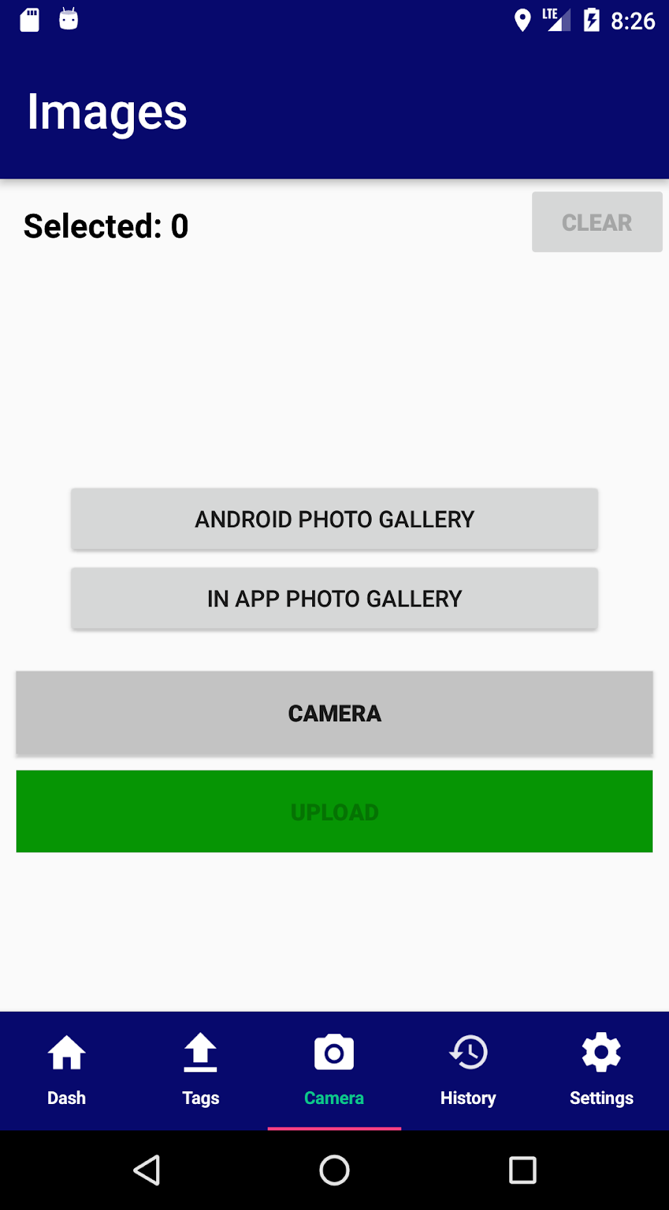
## 4.2  Tags Module

****

### 4.2.1 Responsibilities

The Tags module main responsibilities is to give the user privilege to add tags to the photos by according to the permit number they are working on. The user can also select the work status of the permit number, there are able to check the box in case if the permit needs more investigation and they have the ability to add comments to the current permit number there working on.

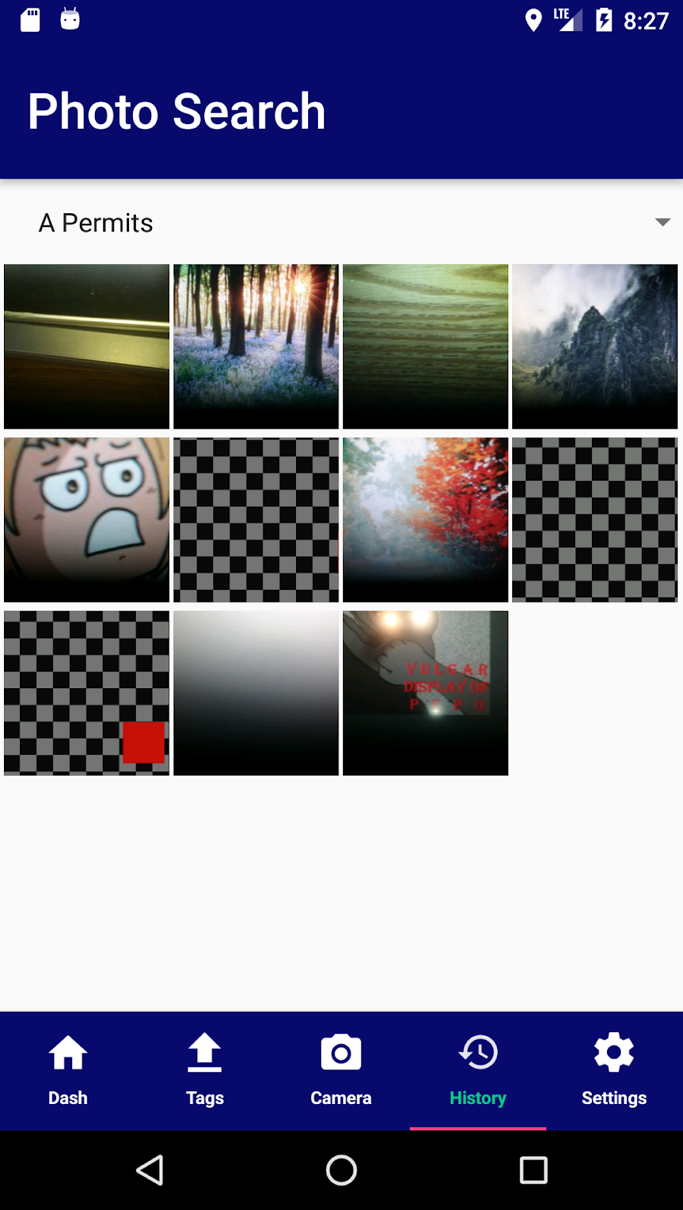
## 4.3  Camera Module

****

### 4.3.1 Responsibilities

The Camera module main responsibilities is to allow the user to take pictures from their own device and uploading it to the database with the tags. The user is also allowed to choose pictures from their device gallery and in app photo gallery.

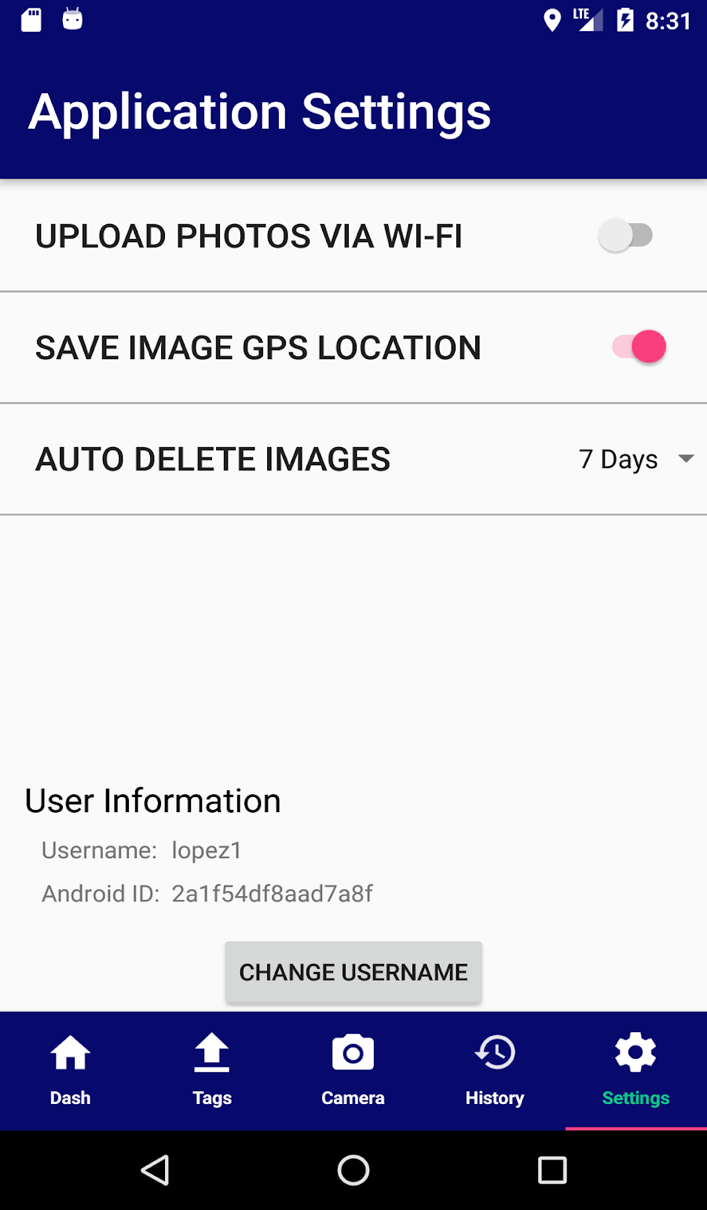
## 4.4  History Module

****

### 4.4.1 Responsibilities

The History module main responsibilities is to allow the user to search for past photos by the categories they are looking for.

## 4.5  Settings Module

****

### 4.5.1 Responsibilities

The Settings module main responsibilities is to allow the user to change their settings preference for example they are able to toggle on and off in case they want to upload photos via WI-FI only. They can also change the settings for the location of where the image was taken to allow the device to get the latitude and longitude. They can also modify if they want the image to be deleted after a certain day. Lastly, they can change their username right at the bottom of the module.

# Detailed Lower level Component Design

## Dashboard Class

Dashboard Class is the main page in the LAPA application. It shows the progress of the current tags being uploaded as well as the photos associated with it. It also display the status of the WIFI connection.

### PROCESSING NARRATIVE (PSPEC)

When the user first login to the LAPA application, a new object is either created or retrieved from the database.

### INTERFACE DESCRIPTION

DASHBOARD

|  |
| --- |
| onCreateView( Inflater: LayoutInflater, container: ViewGroup, savedInstancesState: Bundle ) |
| onCreateContextMenu() |
| onCreateOptionsMenu() |
| onDestroy() |
| showOption( id: int ) |
| hideOption( id: int ) |
| onContextItemSelected( item: MenuItem ) : boolean |
| IsConnectedViaWifi() : boolean |
| deleteImage( s: String) |
| pullAzureData() |

### PROCESSING DETAIL

Since Dashboard Class is mainly used for data storage/retrieval… It also consists of processes that communicate with the server.

#### DESIGN CLASS HIERARCHY

The Dashboard class has no parent or child classes.

#### RESTRICTIONS/LIMITATIONS

Since the Dashboard class is self-contained, there are no practical restrictions.

#### PERFORMANCE ISSUES

Considering that there is only one user object associated for each device, there are no performance-related issues associated with this class. The only potential issue would occur when this class attempts to communicate with the LAPA server. If the device cannot communicate with the server, then the methods responsible for communicating with the server will not be carried out until connection is restored.

#### DESIGN CONSTRAINTS

The major constraint for this class is that there may only be one instance of user per handset. This is because there is only one user associated with the device. In addition, each user ID must be unique, or else interactions with the server will be highly problematic.

#### PROCESSING DETAIL FOR EACH OPERATION

* onCreateView( inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle )
* onCreateContextMenu()
* onCreateOptionsMenu()
* onDestroy()
* showOption( id: int )
  + This method shows the Menu list to the user either to viewTags, viewInfo, or delete.
* hideOption( id: int )
  + This method hides the Menu list to the user.
* onContextItemSelected( item: MenuItem ) : boolean
* isConnectedViaWifi(): boolean
  + This method is responsible for determining whether the device is connected via WIFI or not.
* deleteImage( s: String )
  + This method is responsible for deleting selected Images that contains Tags.
* pullAzureData()
  + This method is responsible for pulling data from the database.

## TAGS CLASS

The Tags class enables the user to choose images to be uploaded either from the phone gallery or the in-app gallery. It also allows the user to add tags for each image selected as well as assign fields such as categories.

### PROCESSING NARRATIVE (PSPEC)

When a user selects an image and add tags to it… It is responsible for storing all the information about the image/s selected/uploaded.

### INTERFACE DESCRIPTION

TAGS

|  |
| --- |
| onCreateView( Inflater: LayoutInflater, container: ViewGroup, savedInstancesState: Bundle ) |
| SendFields() |
| onAttach( context: Context ) |
| showOption( id: int ) |
| hideOption (id: int) |
| clearTagsAndSelectedImages() |
| generateCatView() |

### PROCESSING DETAIL

#### DESIGN CLASS HIERARCHY

The Tags class has no parent or child classes.

#### RESTRICTIONS/LIMITATIONS

#### PERFORMANCE ISSUES

#### DESIGN CONSTRAINTS

#### PROCESSING DETAIL FOR EACH OPERATION

* onCreateView( inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle )
* SendFields()
  + This method is responsible for transporting data between Fragments(classes).
* onAttach( context: Context )
* showOption( id: int )
  + This method shows the Menu list to the user either to viewTags, viewInfo, or delete.
* hideOption( id: int )
  + This method hides the Menu list to the user.
* clearTagsAndSelectedImages()
* generateCatView()

## CAMERA CLASS

The Camera Class allows the user to take images from their phones with the ability to tag each image right after it was taken.

### PROCESSING NARRATIVE (PSPEC)

The user is given an option to (a) capture an image using the in-app camera or device camera, and (b) to select an image from the in-app gallery or device gallery.

### INTERFACE DESCRIPTION

CAMERA

|  |
| --- |
| onCreateView( Inflater: LayoutInflater, container: ViewGroup, savedInstancesState: Bundle ) |
| hasCamera() |
| launchCamera( v: View ) |
| onActivityResult( requestCode: int, resulCode: int, data: Intent ) |
| createImageFile() |
| clearTags() |
| clearTags2() |
| refreshDash() |
| recData |

### PROCESSING DETAIL

#### DESIGN CLASS HIERARCHY

#### RESTRICTIONS/LIMITATIONS

#### PERFORMANCE ISSUES

#### DESIGN CONSTRAINTS

#### PROCESSING DETAIL FOR EACH OPERATION

* onCreateView( inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle )
* hasCamera()
  + This method ensures that the device system has a camera feature.
* launchCamera( v: View )
  + This method launches the camera from the application.
* onActivityResult( requestCode: int, resultCode: int, data: Intent )
  + This method gives the user an option to either capture an image using a camera or select an image from the gallery.
* createImageFile(): File
  + This method stores the image taken.
* clearTags()
  + This method prompts the user if s/he wants to delete all the tags associated to the image.
* clearTags2()
  + This method clears all the listed tags that used to be associated to the image taken.
* refreshDash()
  + This method reloads the Dashboard and sync all the current information as well as if there are images that are currently being uploaded.
* recData()
  + This method is responsible for storing all the information to the server-side of LAPA application which is the Azure database.

## HISTORY CLASS

The History Class shows the users the list of uploaded images. It also enable the user to refine their search by filtering it base from fields (categories). It also pulls the images from the Azure database.

### PROCESSING NARRATIVE (PSPEC)

### INTERFACE DESCRIPTION

HISTORY

|  |
| --- |
| onCreateView( Inflater: LayoutInflater, container: ViewGroup, savedInstancesState: Bundle ) |

### PROCESSING DETAIL

#### DESIGN CLASS HIERARCHY

The History Class has no parent or child classes.

#### RESTRICTIONS/LIMITATIONS

Since the History Class is self-contained, there are no practical restrictions.

#### PERFORMANCE ISSUES

Since the History class mainly displays the list of images uploaded in the past, it will only do aggregation of information. The only potential issue would occur when this class attempts to communicate with the LAPA server. If the device cannot communicate with the server, then it will be impossible to retrieve the list of images taken.

#### DESIGN CONSTRAINTS

#### PROCESSING DETAIL FOR EACH OPERATION

* onCreateView( inflater: LayoutInflater, container: ViewGroup, savedInstancesState: Bundle )

## SETTINGS CLASS

The Settings class gives the user an option to customize the uploading process as well as assigning automatic deletion of images from the in-app gallery after it was uploaded. It also allows the user the option of uploading images via WIFI only, as well as turning on the GPS location which automatically tagged for each image captured. In addition, the Settings class allows the user to logout from the device.

### PROCESSING NARRATIVE (PSPEC)

### INTERFACE DESCRIPTION

Settings

|  |
| --- |
| onCreateView( Inflater: LayoutInflater, container: ViewGroup, savedInstancesState: Bundle ) |
| getDays( i: Intent ) : String |
| getDays() : String |

### PROCESSING DETAIL

#### DESIGN CLASS HIERARCHY

The Settings class has no parent or child classes.

#### RESTRICTIONS/LIMITATIONS

Since the Settings class is self-contained, there are no practical restrictions.

#### PERFORMANCE ISSUES

The Settings class does not have performance problems.

#### DESIGN CONSTRAINTS

#### PROCESSING DETAIL FOR EACH OPERATION

* onCreateView( Inflater: LayoutInflater, container: ViewGroup, savedInstancesState: Bundle )

# 6. Database Design

