

# Project Design Document

**Group Name:** Alpha\_X

**Topic :** Photo Search Engine

- |                           |                      |
|---------------------------|----------------------|
| 1. Thitirat Chitrobaree   | 63070503413 (Phukan) |
| 2. Teerasan Rattanuengkul | 63070503421 (Bung)   |
| 3. Napas Vinitnantharat   | 63070503422 (Fang)   |
| 4. Manutsawin Jeamsaksiri | 63070503442 (Sern)   |

## Summary statement of problem

Now a day taking a photo is very popular. From our research Matthew Brogie - Resply discovered that the average number of photos taken in US is about 20.2 per day. when a number of the photo is too large, it is harder for user to look up the specific photos that user want to look for.

This program will allow the user to search the photo that belong to the tag that user given with in the efficient way and allow the user to modify the tags of the photos ,The program also have a function to display a photo that user satisfy to the web browser

## Program Ability

- Allow the user to add,delete the tag from the photo that user selected.
- Allow the user to search for photo(s) with tags that match the user satisfy.
- Allow the user to search for photo(s) with tags that match the user want and do not have any tags from a second set.
- Allow the user to select the page from the list and ask to see the most three most similar , similar is calculate.
- Allow the user to select one photo and display it on the web browser.

## **Requirement list**

- The system must record all information of each photo tag detail
- The system must be working on any computer that operating on Linux
- The system must allow the user to select one of the photo and display it in the default browser.
- The system must find and display a list of all photo with the multiple tags that user given
- The system must find and display a list of all photo with the multiple tags that user given and don't have any tags from the second set.
- The system must allow the user to add, delete the tag from the photo that user selected
- The system must allow the user to select one of the photos from a list and then the system find the most three similar photos, where "similar" is calculate by the number of sharing tags
- The system must be able to handle an unlimited number of photos
- The system must allow the user to have the 20 maximum number tags in each photo
- The system will work only with English tags and photo files

## **Use case diagram**

### **All Use case name:**

1. Add or delete a tag from the photo that user selected.
2. Search for photo(s) with tags that match user need.
3. Search for photo(s) with tags that match user need with the condition that do not have any tags from a second set.
4. Find similar of the photo and display most top three similar photo(s)
5. Select the photo and display to web browser.
6. Add new photo with a new tag

**1. Use case name: Add or delete a tag.**

Actor: Any system user

Goal: Add or delete a tag from the photo that user selected.

Pre-condition: User choose Search for photo(s) with tags that match user need.

Post-condition: The tag had been successfully added or deleted. OR System display error message OR System bring user back to main menu.

Main success narrative:

1. System ask the user enter the name of the image file (or the image title)
2. System asks the user want to add or delete.
3. User select the option add/delete
4. System asks the user to confirm edit.
5. User enter confirm
6. System update the data

Alternative narrative 1:

2. User input the photo that are not in the database.
3. System go the main menu

Alternative narrative 2:

5. User input wrong option selected (not add /delete)
6. System repeat ask user to select option add or delete until it is valid

Alternative narrative 3:

6. User not confirm
7. System go to main menu without update the data

## **2. Use case choose: Search by tag**

Actor: Any system user

Goal: To find the photo that include the tag that user given

Pre-condition: User choose Search for photo(s) with tags that match user need option from menu.

Post-condition: Photo(s) had been shown to the user OR System display error message and bring user back to main menu.

Main success narrative:

1. System asks user to enter the tag of the photo that user want to find
2. User enter the information of tag
3. System search the photo matched the condition include tag display the result

Alternative narrative 1:

3. System does not find the photo that match user condition
4. System print an error message and go back to main menu

---

## **3. Use case choose: Search by tag (don't have any tags from a second set.)**

Actor: Any system user

Goal: To find the photo that include the tag that user given

Pre-condition: User choose Search for photo(s) with tags that match user need with the condition that don't have any tags from a second set option from menu.

Post-condition: Photo(s) had been shown to the user OR System display error message and bring user back to main menu.

Main success narrative:

1. System asks user to enter the tag of the photo that user want to find, and the second set of tag that user do not want to include.
2. User enter the information of tag and exception tags.
3. System search the photo matched the condition include tags and exception tags and display the result.

Alternative narrative 1:

3. System does not find the photo that match user condition.
  4. System print there no match and go back to main menu.
-

#### **4.Use case name: Search for similar**

Actor: Any System User

Goal: Find the most 3 similar photo and display on the screen

Pre-condition: User select view the photo

Post-condition: 3 Photos (or less) had been shown to the user OR System display error message and bring user back to main menu OR System bring user back to main menu.

Main success narrative:

- 1.System ask the user “Do you want to find the similar photo?”.
- 2.User confirm yes.
- 3.System ask user to enter photo from the list result of a previous search.
- 4.User enter the photo.
- 5.System use memory in database to find out 3 similar photos.
- 6.System display 3 similar photos on the screen.

Alternative narrative 1:

- 2.User not confirm.
- 3.System go to main menu.

Alternative narrative 2:

- 4.User enter the photo that not in the database.
  - 5.System print error message and go back to main menu.
-

## **5. Use case name: Select and display to web browser**

Actor: Any System User.

Goal: display the photo that user want to the web browser.

Pre-condition: System had displayed the search result

Post-condition: The photo had been shown to the user on web browser OR System display error message then ask user to enter valid photo (repeating process) OR System bring user back to main menu.

Main success narrative:

1. System ask the user to select one of the photos from the displayed list
2. User enter the photo
3. System display the photo to web browser
4. System go

Alternative narrative 1:

2. User not select photo(s)
3. System go to the main menu

Alternative narrative 2:

4. User enter the photo that not in the list of the result or not in the database
5. System display error message and system repeat ask the user to enter the valid photo

**6. Use case name: Add new photo with a new tags**

Actor: Any System User.

Goal: Add new photo with a new tag.

Pre-condition: User select option to add new photo with new tags

Post-condition: The photo and tag that user added updated in file OR System bring user back to main menu.

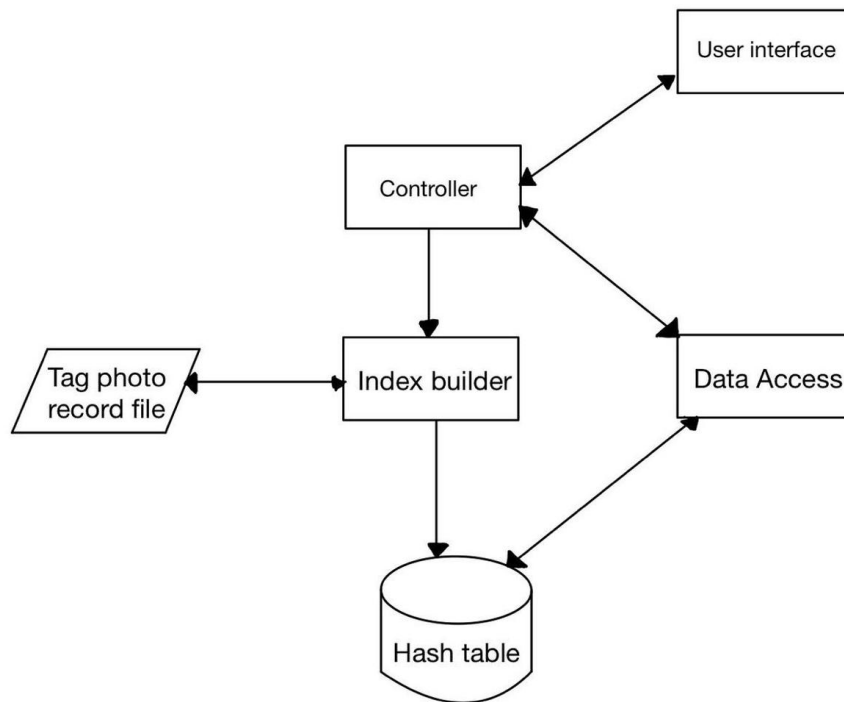
Main success narrative:

1. System ask user to input photo and tags.
2. User in put photo file and tags.
3. System ask for confirmation.
4. User confirm.
5. System update the new photo with new tags on file

Alternative narrative 1:

4. User not confirm.
5. System bring user back to main menu.

## Architecture diagram



**Controller:** Receive the command from the user and also display the data

- use data access module to search the photo and display the result
- ask the user option what user want to do (search, add/delete tag option)
- use index builder to update the tag of the photo to tag photo record file when user add or delete the tags

**Index builder:**

- setup the data structure at the beginning of the program
- update the tag of the photo to tag photo record file

**User interface:** Graphic of the program

- display main menu page
- display search by tag page
- display search by tag with the condition page
- display the image to the web browser

**Data Access:** Search the data of the database

- Search the photo with the tag that user given from the hash table and return the data to controller in term of linked list



- Search the photo with the tag and condition exception and return the data to controller in term of linked list

## Data structures used

### Photo Structure

```
typedef struct _listtag
{
    /*linklist of tags*/
    char* nametag; /*name of the tag*/
    struct _listtag *next;
}LIST_TAG_T;

typedef struct _photo
{
    char namephoto[256];/*name of the photo*/
    char path[512];/*path of photo*/
    int numtag; /*number of the tag*/
    int count; /*use for calculate the similar*/
    int state; /*use as a flag of calculation,use in search
tag and find similar
Block duplicate in list result
true => already be the result of the search
False => not already the result of the search*/

    LIST_TAG_T* alltag; /*array of all tag*/

    struct _photo *nextResult;/*use for search return as a
list*/
```

```
    struct _photo *next; /*next photo (use in masterlist) */  
}PHOTO_T;
```

\*mark in PHOTO\_T state is use for block duplicate result when add to the list result

### Hash of the tagTable and photoTable Item

```
typedef struct _hashitem  
{  
    PHOTO_T* photo;  
    struct _hashitem *next;  
}HASHITEM_T;
```

In this program the data structure has been separate to three main part which include

#### 1. PHOTO\_T

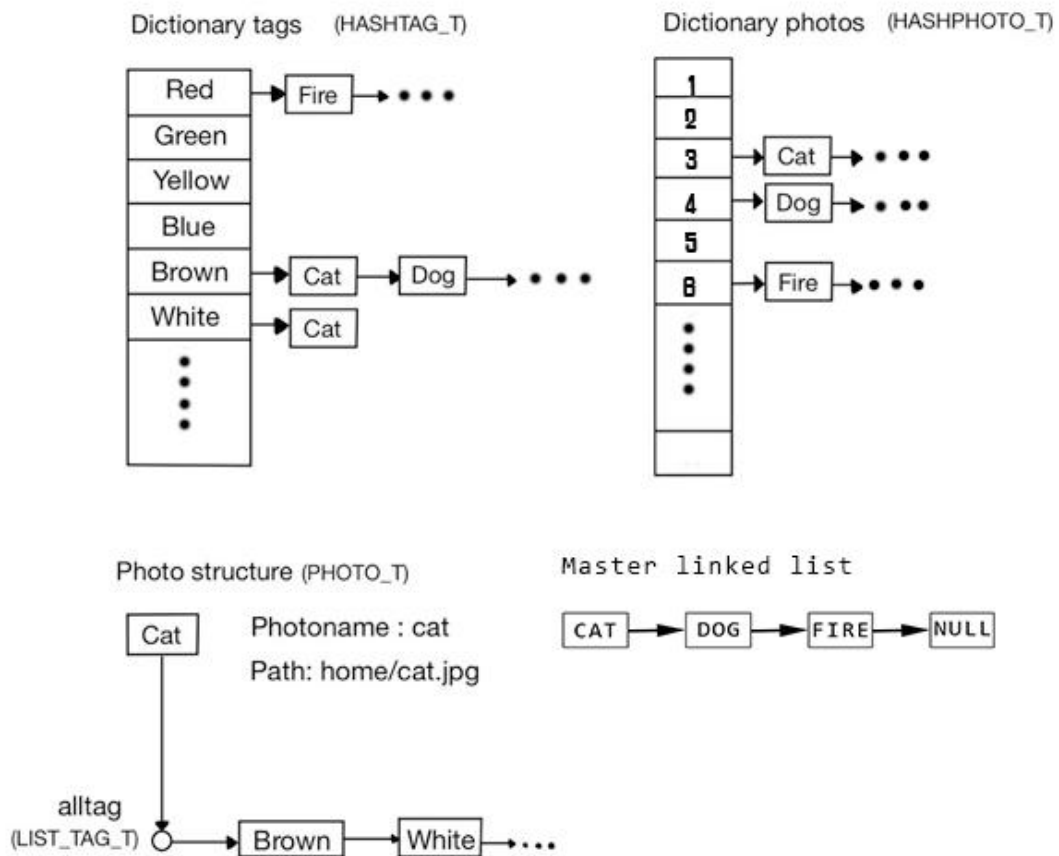
which is a data structure that hold all of the information about photo and data that will be used for calculation. In master List consist of PHOTO\_T structure that linked with (LIST\_TAG\_T\* alltag) to store all of the tag that photo had.

#### 2. LIST\_TAG\_T

which is a link-list of tag photo that hold all of the tag that photo had

#### 3. HASHITEM\_T

This hash table has been used for to create a two-hash table. one is for Search by tag and also calculate the similar of the photo (dictionary of the tags) and other one is for look up the from data with the given name of the photo (dictionary photos).



## Tag photo record file

The file that record the all of the tag photo will represent in a text file as below.

---

[name of photo] [number of the tag] [path]

[name tag 1][name tag2] .....[name tag N]

---

### Example:

Cat;2;home/cat.jpg

Brown;white

Dog,2;picture/dog.jpg

Brown;Black

Fire;1;picture/dog.jpg

Red

---

## Pseudocode

### 1. Find the photo information with the name photo

**findphoto** (name photo) :

Set Key to Hashfunciton(name)

Loop the linklist in hashphoto[key]

If the namephoto is match with photo->namephoto

Set result to that photo

Return result

### 2. Add the tag

Program ask a name of the photo and tags[]

data = **findphoto** (name photo)

Add data->numtag with size of tags[]

if data is not NULL

head = data->alltag

For tag in tags[]

Allocate temp(LIST\_TAG) and assign tag to temp->nametag

temp->next = head->next

Head->next = temp

### 3. Delete the tag

Program ask a name of the photo and tags[]

data = findphoto(photo)

For tag in alltag[]

If tag in data

Delete the tag from the list tag

#### 4. Check is the photo have the given tag or not

**checktag**(photo, name tags[], sizetag):

Set result = false

Set count to 0

For tag in name tags[]

if tag in photo

    Increase Count by 1

If sizetag is equal to count

    Set result to true

Return result

#### 5. Search for photo(s) with tags that match user need

Program ask a name tags[] from user

Set listresult = NULL

For tag in tags[]

    set key = hashfunction(tag->nametag)

    listphoto = dicttag[key]

    while listphoto is not NULL

        if the photo is not in listresult (photo->state is false )

            if all of listphoto tag is in tags[]

                add the photo to the listresult(linklist)

                set the photo have in listresult (photo->state to true)

        listphoto = listphoto->next

```
While loop listResult
    Set all photo->state in list to false
Return listResult
```

## 6. Search for photo(s) with tags that match user need with the condition that don't have any tags from a second set

Program ask a name tags[] and excepts[] from user

Set listresult = NULL

```
Listresult = searchbytag(tags[])
While loop listResult
    If photo have a tag in excepts[]
        Delete the photo from listresult
    Go to next photo
Return listresult
```

## 7. Find similar

Program ask the user to enter a name photo

data = findphoto(name photo)

tag = data->alltag

**initial:**

```
while tag is not null
    set key = hashfunction(tag->nametag)
    listphoto = dicttag[key]
    Loop the list of photo
        set photo->count equal to 0
        Set photo is not in the listresult (photo->state is
false)
```

```

        listphoto = listphoto->next
    tag = tag->next
calculate similar:
while tag is not null
    set key = hashfunction(tag->nametag)
    listphoto = dicttag[key]
    Loop the list of photo
        increase photo->count by one
        listphoto = listphoto->next
    tag = tag->next

```

#### **add to priority queue:**

```

while tag is not null
    set key = hashfunction(tag->nametag)
    listphoto = dicttag[key]
    Loop the list of photo
        if photo->state is false
            Add the photo to priority queue
            let photo->state be true
        listphoto = listphoto->next
    tag = tag->next
Pop priority queue 3 time and print the result

```

## Overview flowchart

