# Project Design Document

**Group Name :**

Alpha_X

**Topic :**

Photo Search Engine

**Team :**


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

# Abstract


## Summary statement of problem

Nowadays taking a photo is very popular. From our research Matthew Brogie - Repsly discovered that the average number of photos taken in US is about 20.2 per day. When there are too many photos, it is harder for user to look up the specific photos that user wants 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

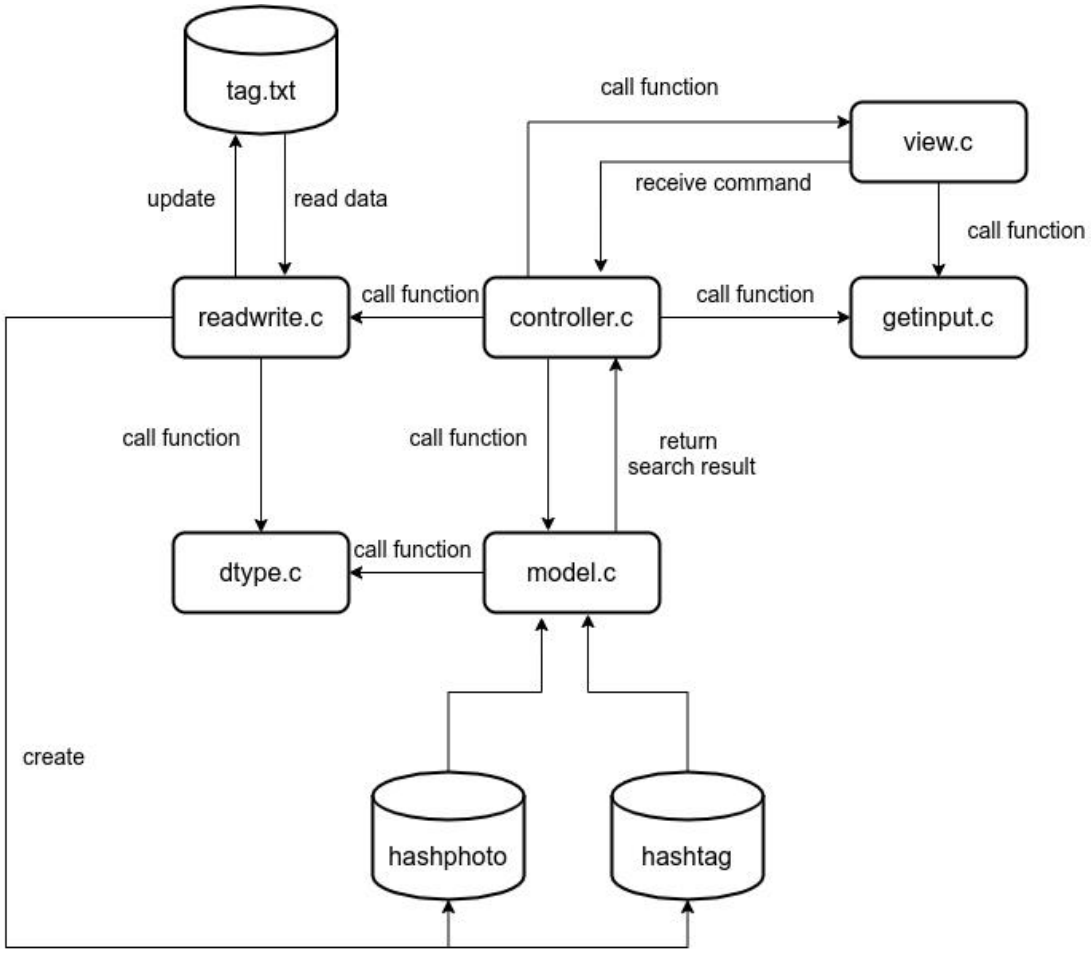
- Allow the user to search for photo(s) with tags that match the user satisfy.

- 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 photo 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.

# Architecture Diagram




**datastruct.h**

This file contain all of typedef of the program.

**readwrite.c**

This file contain all function to set up the data structure of a fie tag.txt, add the photo to the data structure and also include the function write data structure to a file.

**dtype.c**

This file contain all ADT of hash data structure and adaptive array size structure (use in find similar photo)

**model.c**

This file contain all function to search the photo from the hash and also contain a function to calcultae similar of photo

**view.c**

This file contain all function that display the user interface and receive the data as a output argument

**getinput.c**

This file contain all function that get the input from the user and validate the input from the user

**controller.c**

Thid file contain all function controller all the program by using a function from another module

# Function Descriptions

| Component | Functionality |
|---|---|
| **controller.c** | **handleAddNewPhoto -** get all the user input, validate the input then add new photo with new tag and path in the structure file.<br>**handleSearchOption -** get all the user input, validate the input then search for tags that match what user had input and print out the result.<br>**handlefindSimilar -** get the input from user, validate the input then find the similar photo.<br>**handleDisplayInBrowser -** get the input from user, validate the input then display photo in browser.<br>**handleSubMenu -** print out the sub-menu for extra option, add or delete tags, find similar and display photo in browser. Validate the option and show the function that option selected.<br>**main -** print out main menu, Call function in **readwrite.c** to read and build data structure and also do the main of the program. |
| **dtype.c** | **intialHash -** use to initialize hash table.<br>**hashfunction -** convert key to the index of the array, use as a hash function both, dictionary tag and photo.<br>**insertitem -** insert hash item to hash table.<br>**getlist -** return the linklist of the hash in the given key. |
| **getinput.c** | **checkStr -** check if the string input is valid or not.<br>**checkPathAndNamePhoto -** ckeck if path is valid or not.<br>**isunique -** check is photo is unique or not.<br>**isEnter -** check if player press enter or not.<br>**getCharacter -** get input of character.<br>**getOption -** get input of option with validation.<br>**getString -** get input of string.<br>**getNamePhoto -** get input of name of photo and check if it is all alphabhetic or spacebar or “.” or “/” and asking the name photo until it valid. |

| | |
|---|---|
| | **getAllNameTag -** get input of nametag and validate it until user press enter.<br>**getPath -** get input of path with validation.<br>**freestring -** free string in array of string. |
| **model.c** | **findPhoto -** find the photo with the given name and return the information about the photo.<br>**checktag -** check the photo have all of the tag[] or not if not return false and true for the photo include with tag[].<br>**searchByTag -** find the photo with the given tag[] that user given and return linklist photo result that have an all tag in tag[].<br>**checkexcept -** check the photo have a tag in tag[] but not in except[] tag or not.<br>**searchCondition -** find the photo with the given tag[] that user given and return linklist photo result that have an all tag in tag[].<br>**calculateSimiliar -** calculate each photo similiar with the given alltag.<br>**comparator -** use in function **qsort** use to compare the priority of two item.<br>**findSimilar -** find the similar photo with the given namephoto and return Array of the sorted photo |
| **readwrite.c** | **add_photo_2_hashphoto -** add the information of data structure for each photo into the hash table.<br>**add_photo_2_hashtag -** add the information of linked list for tag(s) in photo into the hash table.<br>**add_photo_2_masterlist -** copy the information from one data structure to the master list which will be used to compare and search in other function.<br>**createFile -** create the data base file if the program failed to open the file or the file doesn't exist.<br>**checkNULL -** check calloc operation, is it successful or not, it will display error if calloc is not successful. |

| | |
|---|---|
| | **readData -** read the data from the data base file then initialized the data structure required in the program and other necessary elements.<br>**addPhotoToStruct -** read the data from the argument given then initialized the data structure required in the program and other necessary elements.<br>**writeData -** will write the data into the output file that will be used in the other module or function in the program.<br>**freeAll -** will free all the used memory allocated to the data structure and linked list. |
| **view.c** | **clearscreen -** clear the screen and ask of enter to go the main menu.<br>**displayphoto -** display photo and information of photo on the screen.<br>**menuUI -** will display main menu on the screen and ask for option that user want to choose. Moreover, set value of choice.<br>**subMenuUI -** display submenu on the screen and ask for option that user want to choose. Moreover, set value of choice.<br>**addNewPhotoUI -** display user interface of add new photo and ask for name of photo,path and tag. Moreover, set value of namephoto, sizetag, path and tag.<br>**searchByTagUI -** display user interface of search by tag and ask for tag and ask for enter to stop. Moreover, set value of tag and sizetag in output argument.<br>**searchConUI -** display user interface of search by include tag and exclude tag and ask for include tag and exclude tag and ask for enter to stop. Moreover set value of tag,sizetag,except and sizeexcept in output argument.<br>**similarUI -** display user interface of find 3 similar photos and ask for name of photo after that display 3 similar photos. Moreover this function also store photo name.<br>**displayInBrowserUI -** display user interface of display on the browser. |

# Data Structure Documentation

## 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

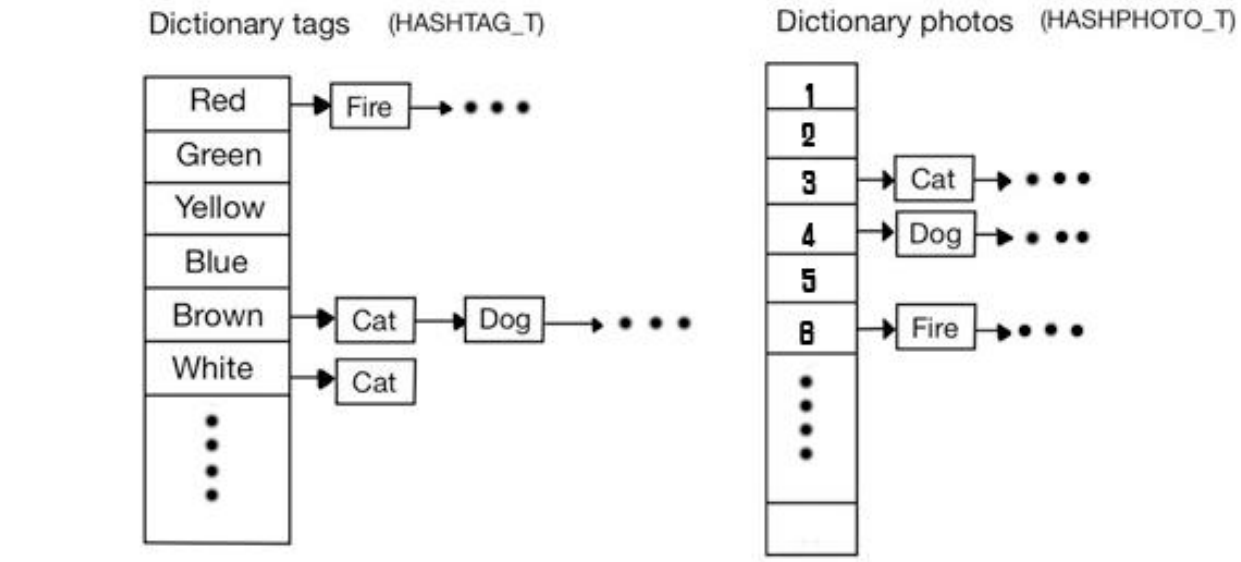
which is a data structure that hold all the information about photo and data that will be used for calculation. In master List consist of PHOTO_T structure that linked with (LIST_TAG_T* alltag) to store all the tag that photo had.

### 2. LIST_TAG_T

which is a link-list of tag photo that hold all 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 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).


Dictionary tags (HASHTAG_T)
Red
Green
Yellow
Blue
Brown
White
Fire
Cat
Dog
Cat
Dictionary photos (HASHPHOTO_T)
1
2
3
4
5
6
Cat
Dog
Fire

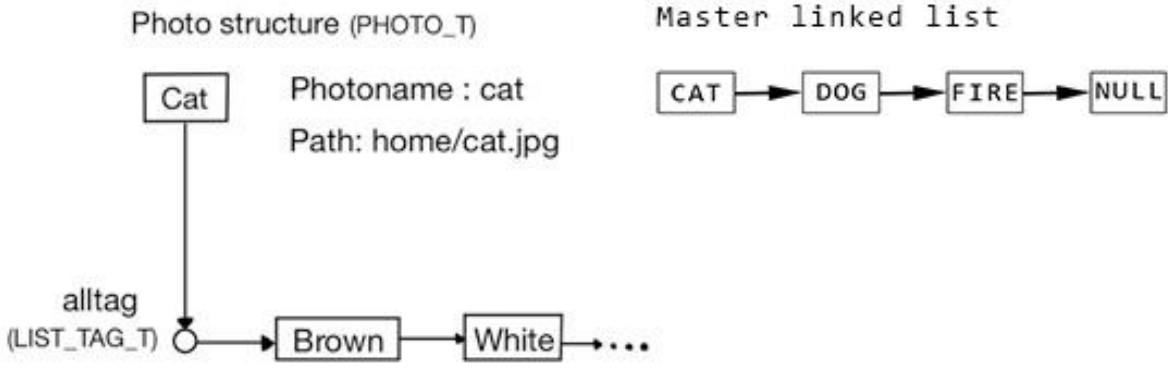


Photo structure (PHOTO_T)
Cat
Photoname : cat
Path: home/cat.jpg
alltag
(LIST_TAG_T)
Brown
White
Master linked list
CAT
DOG
FIRE
NULL

## 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

Re

# 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. 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.

### 3. Check is the photo have all tag in tag[] but not in except[].

**checkexcept** (photo,name,tag[],excepts[],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 and checktag(photo,name tags[],sizetag) is true.

Set result to true.

Return result.

## 4. 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.

## 5. 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[], except[] 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[] but not in except[].

                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. Find similar photo.

**Program ask the user to enter a name photo.**

Arraysort = []

data = findphoto(name photo).

alltag = data->alltag.

Loop tag in alltag that photo have

    Loop each photo P in linklist of hashtag[tag]

        If photo P state not 1 (not already in arraysort) and P is not photo user given

            Calculate similar photo P and set the value P->count

            Add the photo to arraysort

            Set P->state to 1 (already in arraysort)

    End loop

End loop

Quick sort the arraysort

Return arraysort

# Top Level Flowchart