**Chapter 4 System Design**

In this chapter, we will first discuss our system design based on our experiment results. And then we will describe how to implement the system in detail.

1. User Case

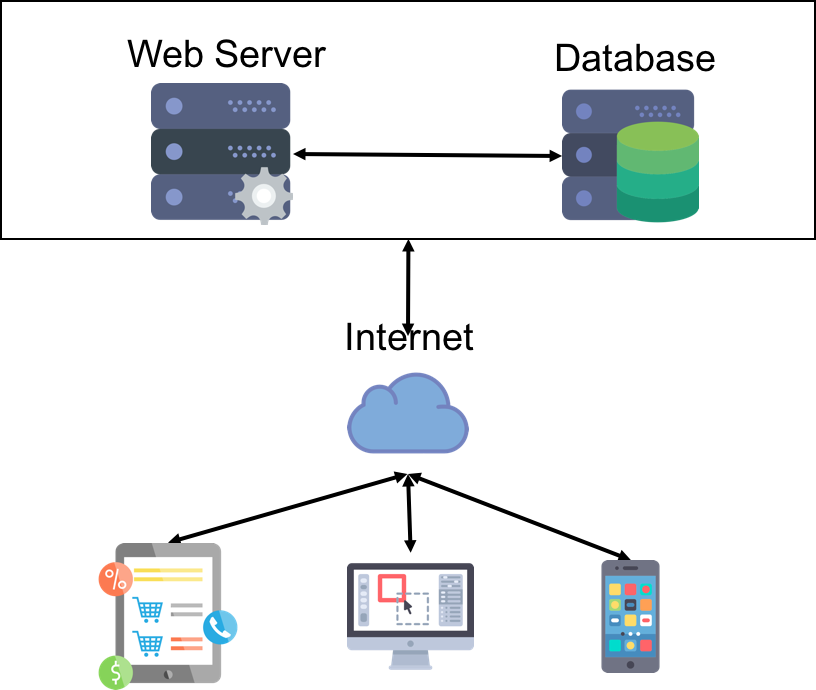
User case is the definition and description of a whole unit function of a system or subsystem without the presentation of the internal structure of a system or subsystem, so user cases can help user to understand the system better. This part will introduce the user cases of the system.

1. Import Images which recorded by user

After the day’s record was finished, user can click the choose-image button to choose the whole day’s images for importing into our system. And then click the import-button. The system will automatically process these metadata, such as faces, objects, location, creation time and so on. Upon the process of importing image is finished, the result will be show on the page. The user can view it easily.

1. View images via single cue
2. The user met somebody, and he/she cannot remember the other cues. He/She can double click the person image to view the related images.
3. The user remembers an object in his memory, and he/she cannot remember the other cues. He/She can double click the object image to view the related images.
4. The user remembers there is a thing happened somewhere, and he/she cannot remember the other cues. He/She can double click the location image to view the related images.
5. Query images via cues combination
6. The user met someone at about 2pm but he didn’t remember the specific time. He can click to choose this man as one cue and choose 14:00 as another cue, and the system will search the images between 13:30 to 14:30 which contain this person and show result.
7. The user met someone somewhere, he wants to review these pictures. He can choose this guy and this location to search images and recall what happened.
8. The user used some object or saw some object somewhere. He can choose this object and this location as cues combination and search the result.
9. The user remembered some object around 3pm, he can choose this object and 15:00 as cues. The system will find all images during 14:30 and 15:30 contain the object.
10. System Architecture

The system is designed using Browser/Server pattern. The advantage of this pattern is that we can use our system in any platforms which contains mobiles, pad and personal computer upon we deploy our system into the Internet. The system architecture can be designed as picture 4-1.



Picture 4-1

From the picture, we can see that after we finished to develop our web server and database, and the deploy it on the Internet. We can reach our system in any platforms. It’s very convenience for user to use. The Browser/Server pattern also has other advantages are listed aa below:

1. Operate anywhere without installing any specialized software
2. The expansion of the system is very strong
3. The maintenance and the logic of upgrade is very simple

For the first point, the reason is that we using our system via a browser, so if there is a browser in a device and our system is deployed on the Internet, the user can reach the website easily. As a result, the user could use our system without installing any specialized software. For the second point, we said the expansion is very strong because if we want a new feature in our system, we only need to update our web server. For the third point, the reason is similar with the second point. We have no client to maintenance. So we could spend less time on maintenance and upgrade. We just need to focus these on our web server.

We need to notice our web server in our system architecture. This part not only includes ourselves server, but also it incudes the services from Microsoft and Google. We integrate the services from Microsoft and Google into our system. And the services include Computer Vision, Face Recognize, Bing Image Search and Google Maps Server. We use these services to handle the face, object and location in user’s images.

1. System Design
2. System Framework Design
3. System Front-End Design
4. System Database Design
5. System Implementation
   1. Development Environment
   2. API Introduction
6. Face Recognition API from Microsoft
7. Create a face list

***Request URL:***

http://westus.api.cognitive.microsoft.com/face/v1.0/facelists/{faceListId}

***Request Parameters:***

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| faceListId | String | Id of the created face list. |
| name | String | Name of the created face list. |

The function of this API is that creating a face list which stores the faces we recognized from user’s images. When we use this API, we need to provide a list id like ‘Memory\_Face\_List\_Id’and a list name like ‘Memory\_Face\_List\_Name’. And the API will return a response to show the result of creating the face list. If the response status is 200, that means we create the face list successfully.

1. Get a face list

***Request URL:***

[http://westus.api.cognitive.microsoft.com/face/v1.0/facelists/{faceListId}](http://westus.api.cognitive.microsoft.com/face/v1.0/facelists/%7BfaceListId%7D)

***Request Parameters:***

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| faceListId | String | Id of the created face list. |

***Response Body:***

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| faceListId | String | Id of the created face list. |
| name | String | Name of the created face list. |
| persistedFaces | Array | Faces in the face list. |

The function of this API is that we can get the face list we created using face list id. If we used the above API to create a face list which id is “Memory\_Face\_List\_Id”, we will use this face list id to get the face list. If we send this request, we will get a face list which contains all the faces we recognize from the user’s images.

1. Add a face into face list

Request URL:

[http://westus.api.cognitive.microsoft.com/face/v1.0/facelists/{faceListId}/persistedFaces[?targetFace](http://westus.api.cognitive.microsoft.com/face/v1.0/facelists/%7BfaceListId%7D/persistedFaces%5B?targetFace)]

Request Parameters:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| faceListId | String | The id of the face list we created. |
| targetFace | String | The rectangle area of the face on the image. |

Request Body:

The bytes of the image

Response Body:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| persistedFaceId | String | The unique id of the face in the face list. |

The function of this API is that adding a face we detected from user’s image into the face list which we created using above API. When we use this API to add a face into the face list, we need to provide the id of the face list which we created before and the area of the face location on the image. Then we will get a result of the action, the result contains the face id which was stored in the face list which we created and we can get the face in the next time.

1. Detect faces from image

***Request URL:***

http://westus.api.cognitive.microsoft.com/face/v1.0/detect[?returnFaceId]

***Request Parameters:***

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| returnFaceId | boolean | Return face ids of the detected faces or not. |

***Request Body:***

The bytes of the image.

***Response Body:***

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| faceId | String | Unique face id of the detected face. |
| faceRectangle | Object | A rectangle of the face location on image. |

The function of this API is that detecting the faces on the image which user imported via our system. When we use this API, we need to convert user’s image to bytes and set the ‘returnFaceId’field to true. Then we will get the face unique id and the location of the face on the image.

1. Find the similar faces with the face
2. External API from Google
3. Internal API
   1. To be decided