SOFTWARE DESIGN FOR PhotoN WEBSITE

Anudeep Rao Perala (CS21BTECH11043)
Asli Nitej Reddy Busireddy (CS21BTECH11011)
Narsupalli Sai Vamsi (CS21BTECH11038)
Pranav Varma Pericherla (CS21BTECH11044)

Contents

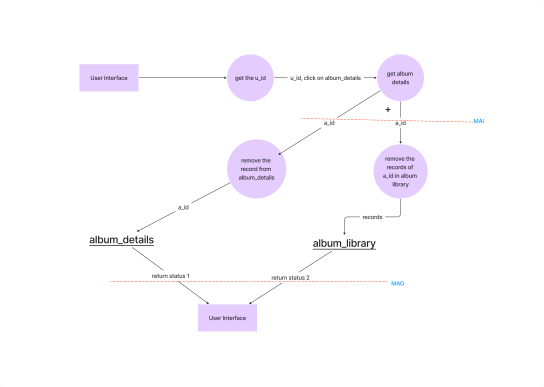
1 Overview

The goal of PhotoN is to revolutionize the way photographers and photography enthusiasts engage with digital photography, providing a versatile platform for photo storage, sharing, exploration. PhotoN tries to make the process of managing photo collection, making it easy for a photographers' work to go public, and facilitating the discovery of new photography according to taste of the users.

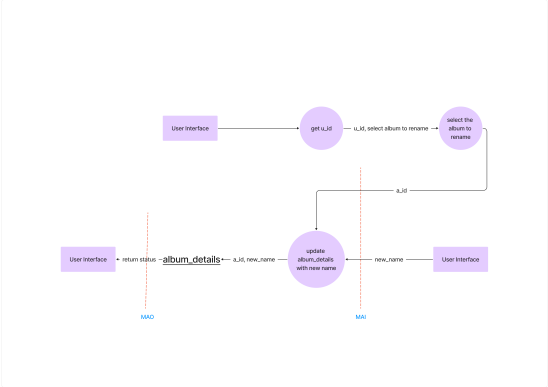
With a user friendly interface, PhotoN will offer many features like sharing options for users, a marketplace for photos viewing, a sophisticated search to navigate the the large set of photographs and to find user profiles.

The design document elaborates on the technical and architectural blueprint of PhotoN, stating about the system's structure, data flow, module integration, and interaction patterns.

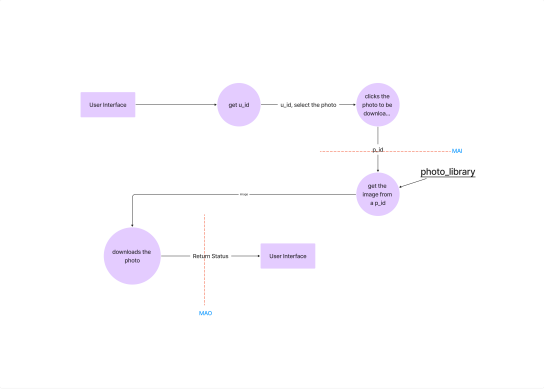
2 Data Flow Diagrams



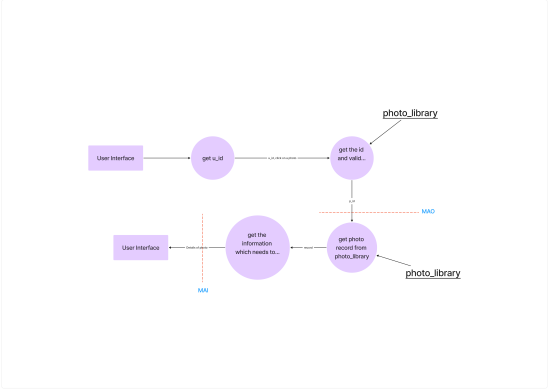
Remove an album



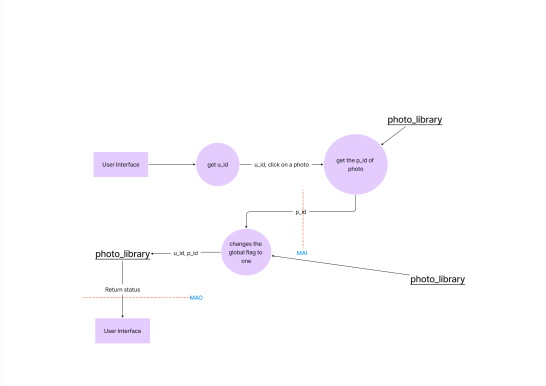
album name change



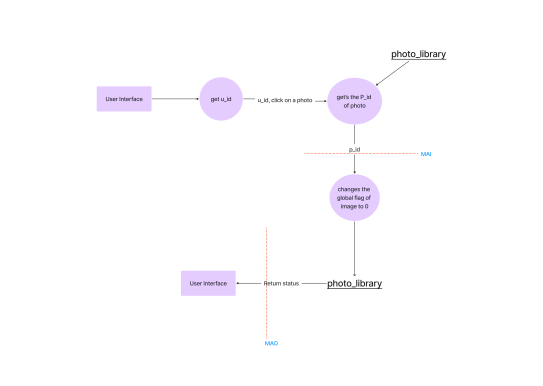
save in local device



viewing details of image



Adding photo to global



Removing photo from global

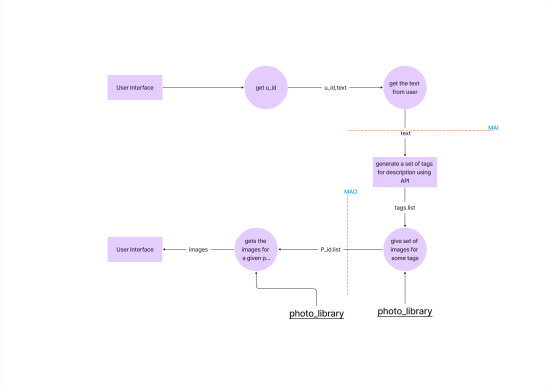
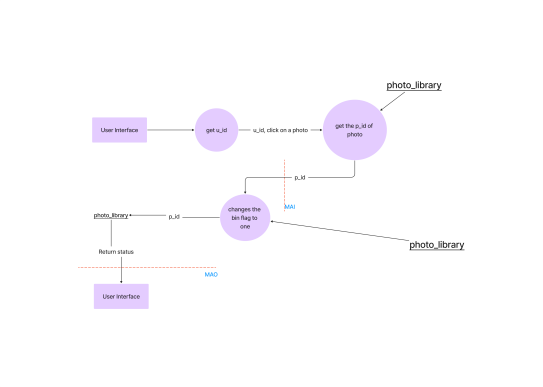
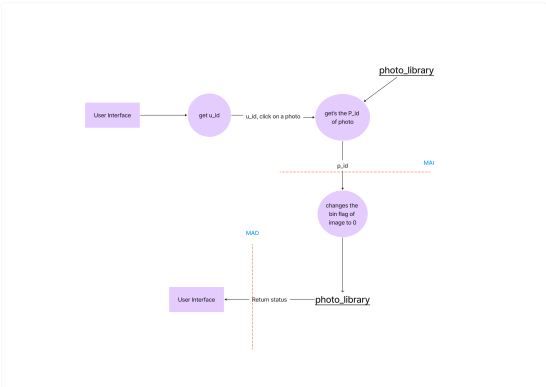


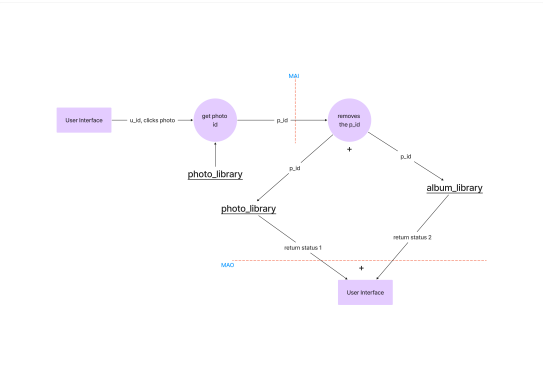
Image search



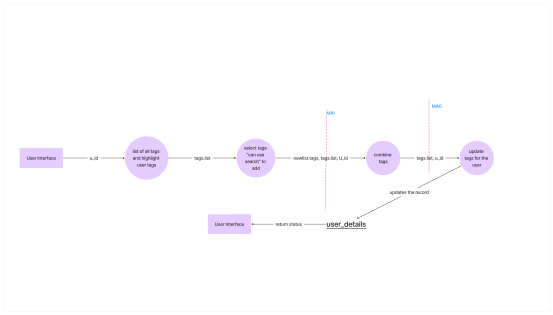
Move to bin



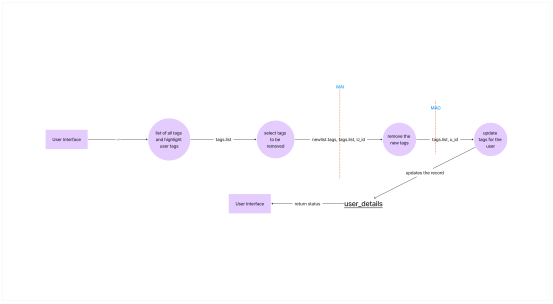
Recover from bin



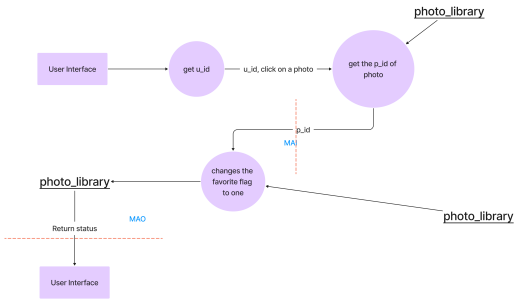
Delete from bin



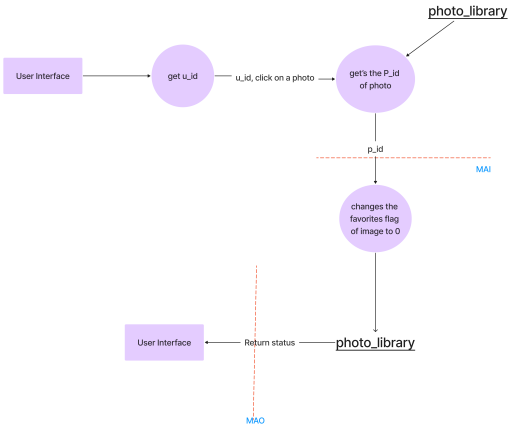
Add tags(User)



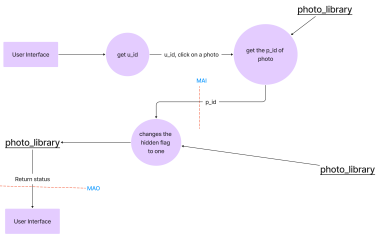
Delete tags(User)



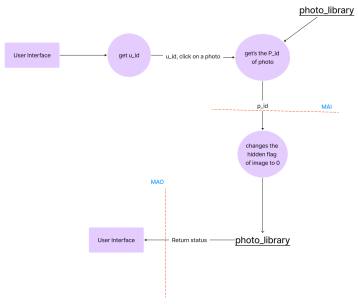
Add to favorites



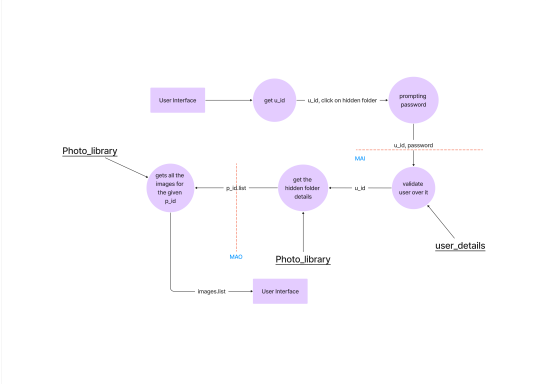
Remove from favorites



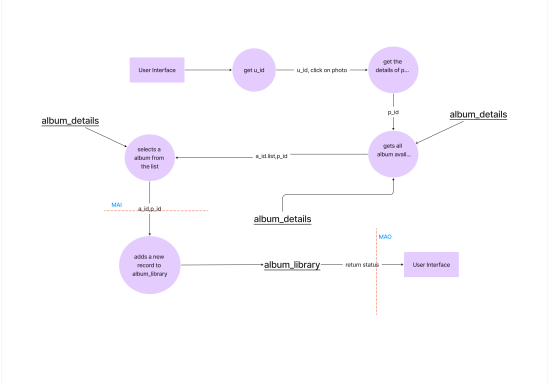
Add to hidden folder



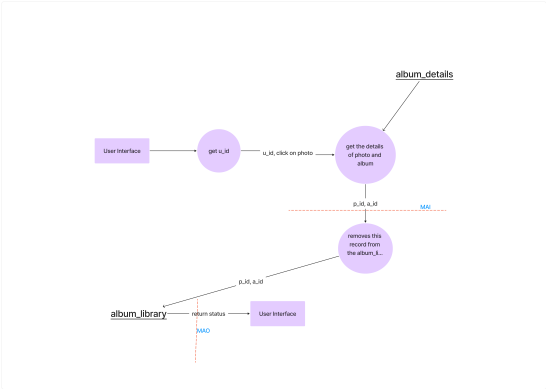
Remove from hidden folder



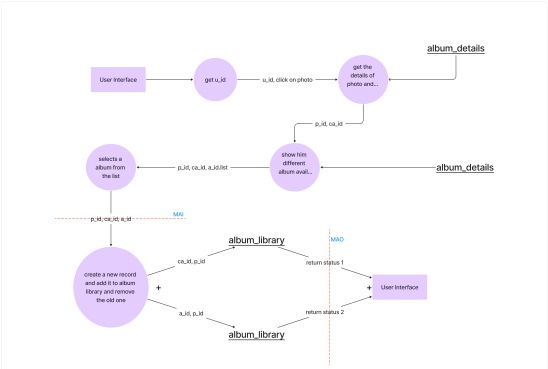
Seeing in hidden folder



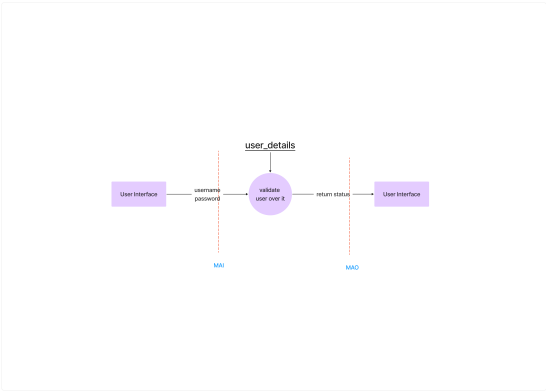
Add to album



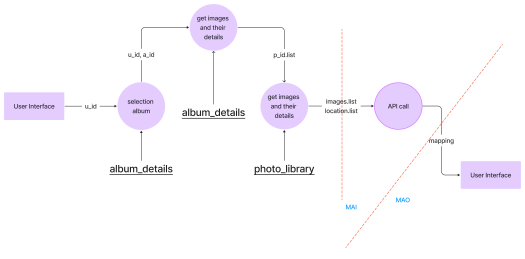
Remove from album



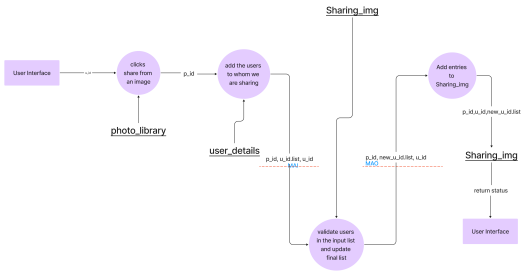
Moving between album



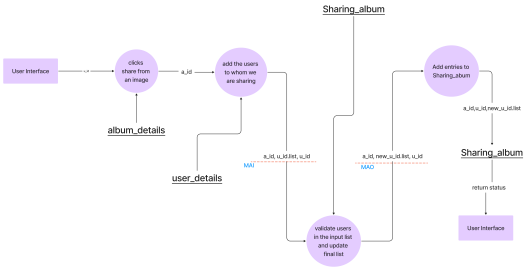
Login



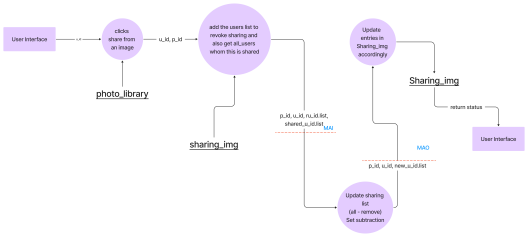
MAP API



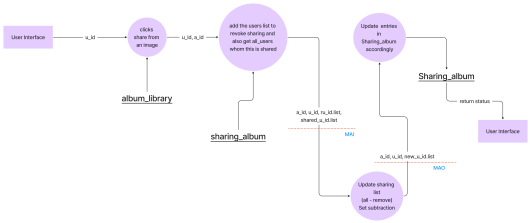
Sharing Image



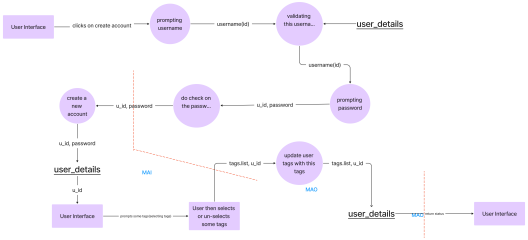
Sharing an album



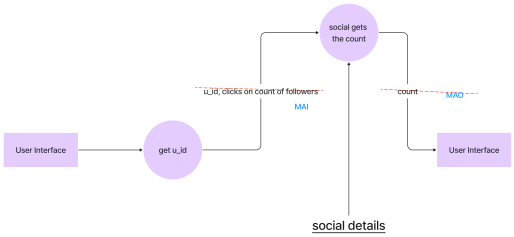
removing user's shared image permissions



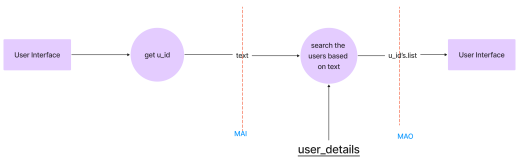
removing user's shared album permissions



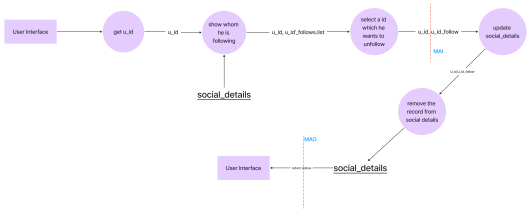
Create an account



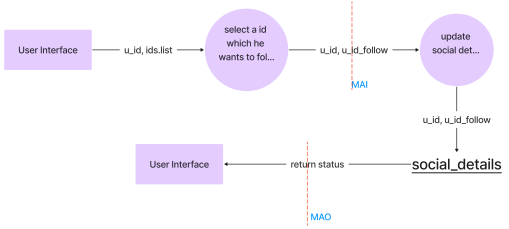
Counting followers



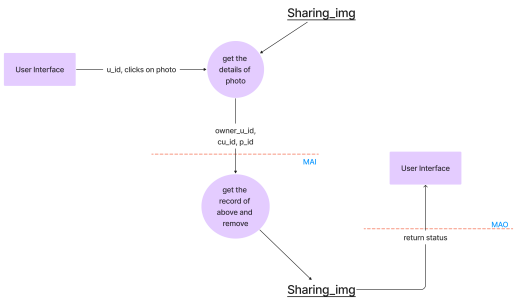
Search users



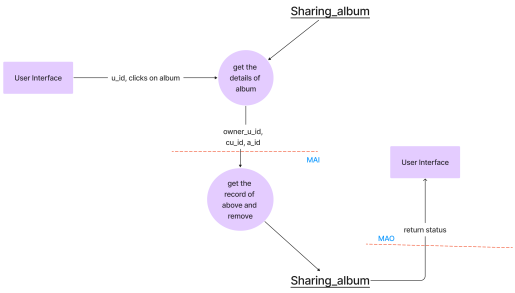
Unfollowing a user



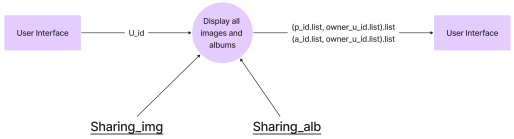
Following a user



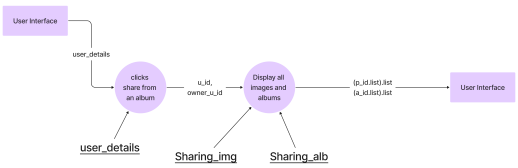
Removing the shared image



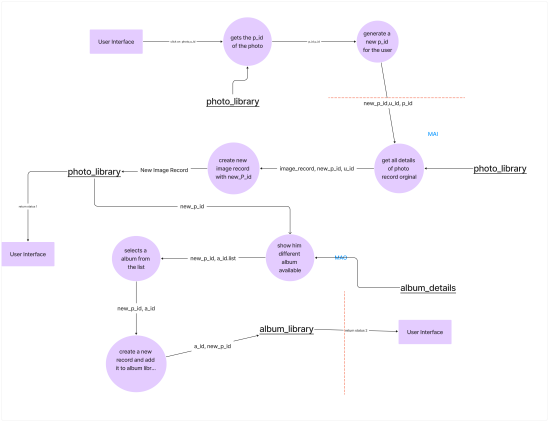
Removing the shared album



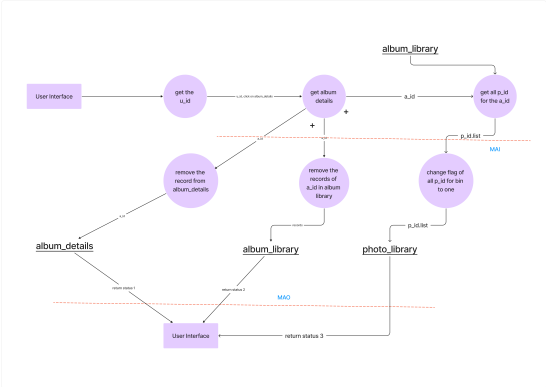
Viewing all shared



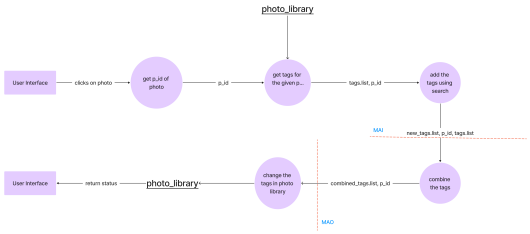
Viewing Shared for a user



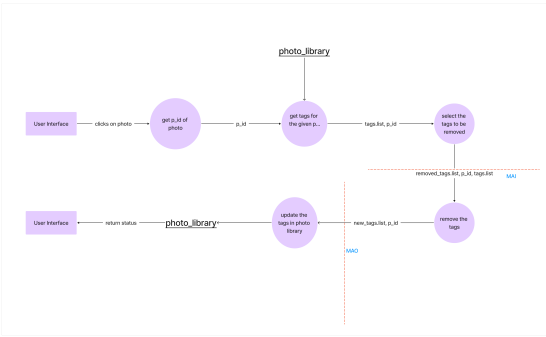
Copying image to a album



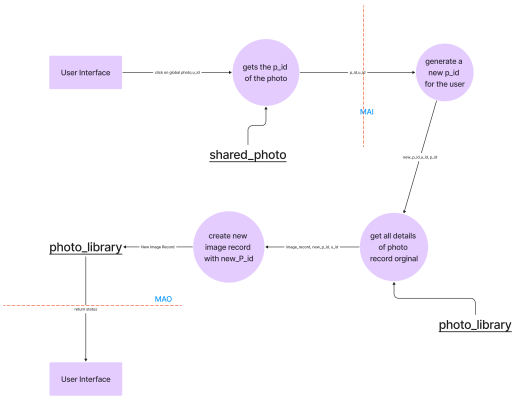
Deleting Album



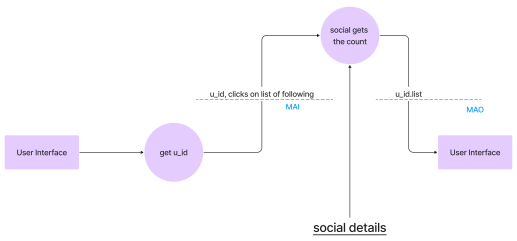
Adding new tags to photo



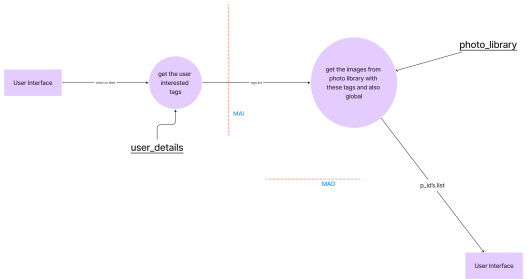
Deleting tags of photo



Saving a shared photo



Following list



Feed

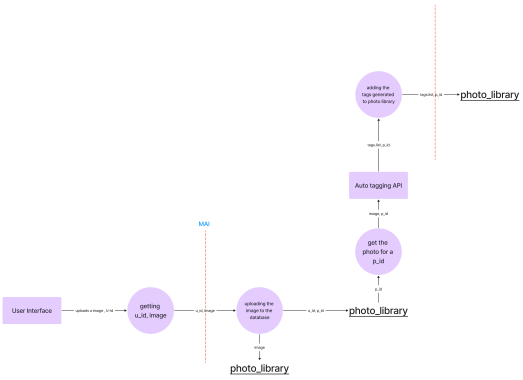
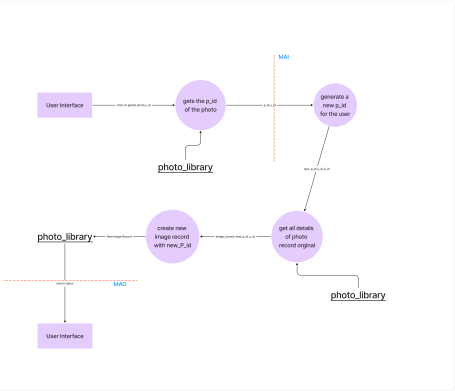
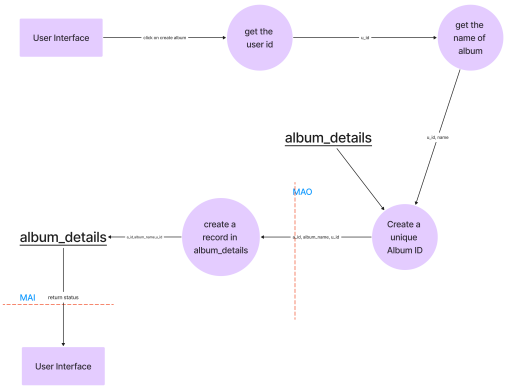


Photo Upload



Save global Photo



Rename Album

3 Structure Chart

3.1 Main Module

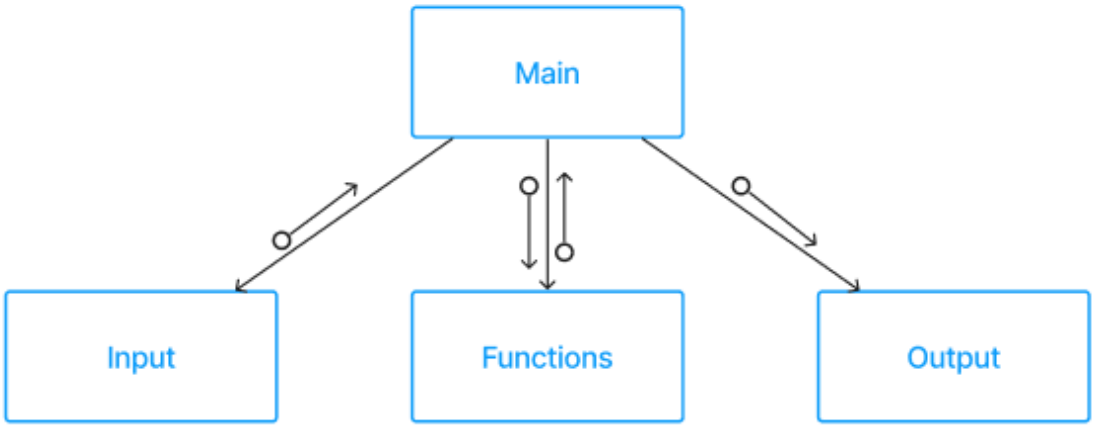
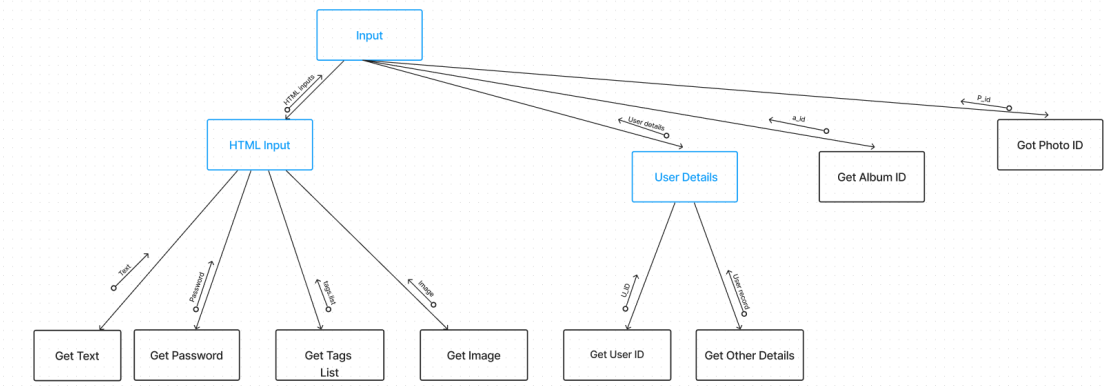
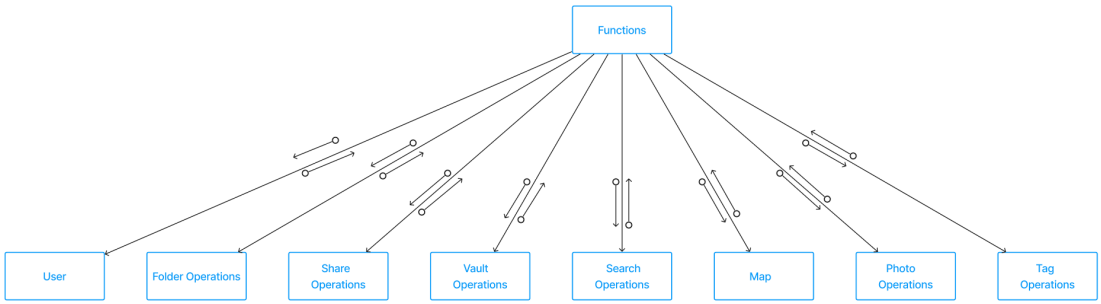


Figure 3.1: First Factored Modules

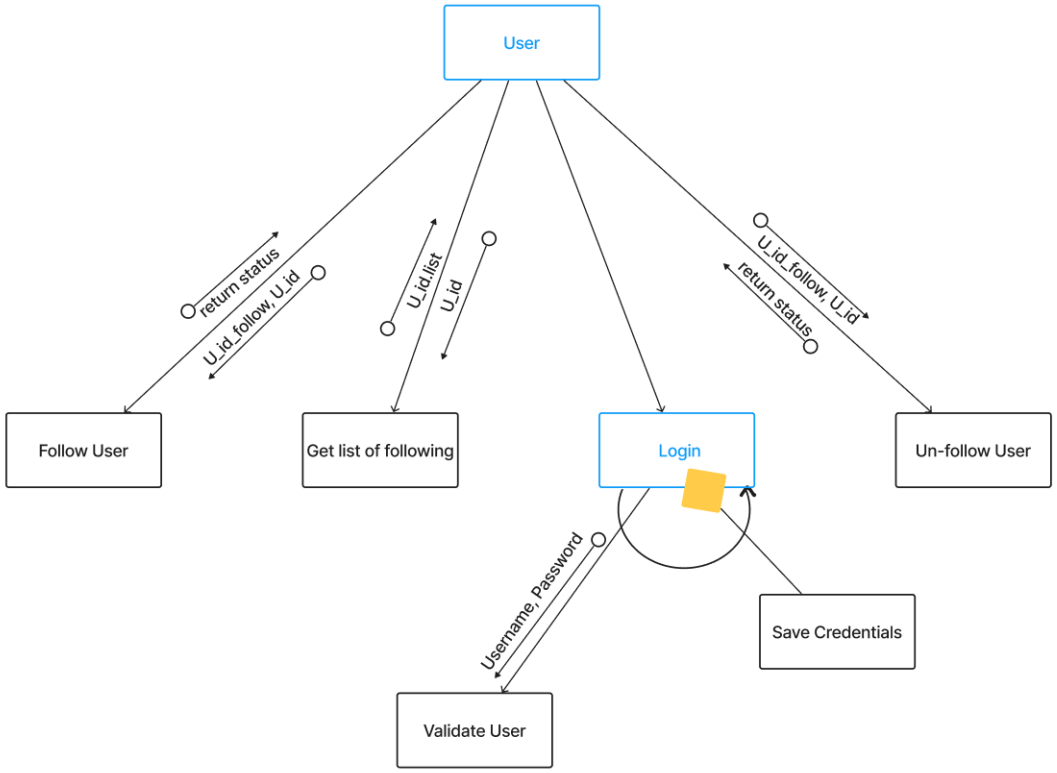
3.2 Input Module



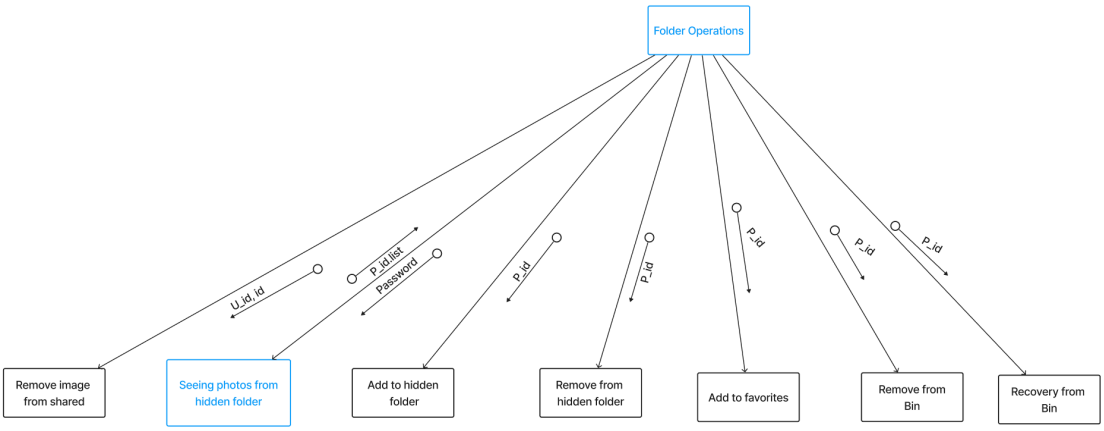
3.3 Function Module



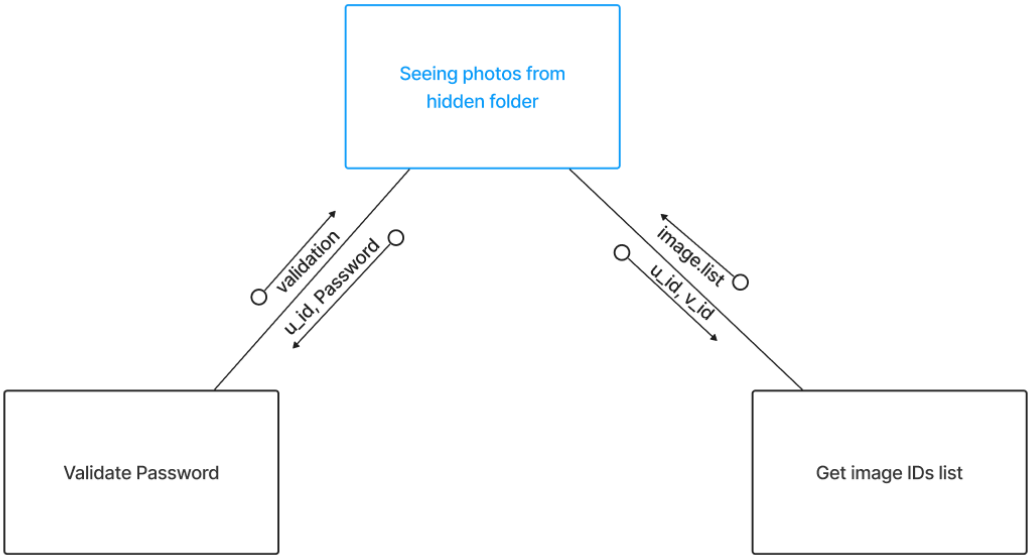
3.3.1 User



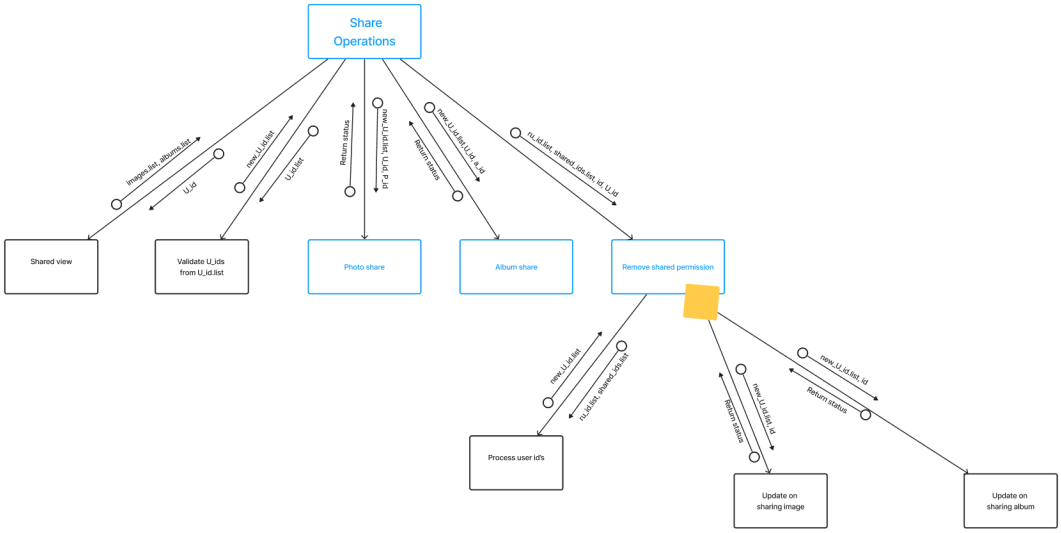
3.3.2 Folder Operations



Seeing Photos from Hidden Folder



3.3.3 Share Operations



Album Share

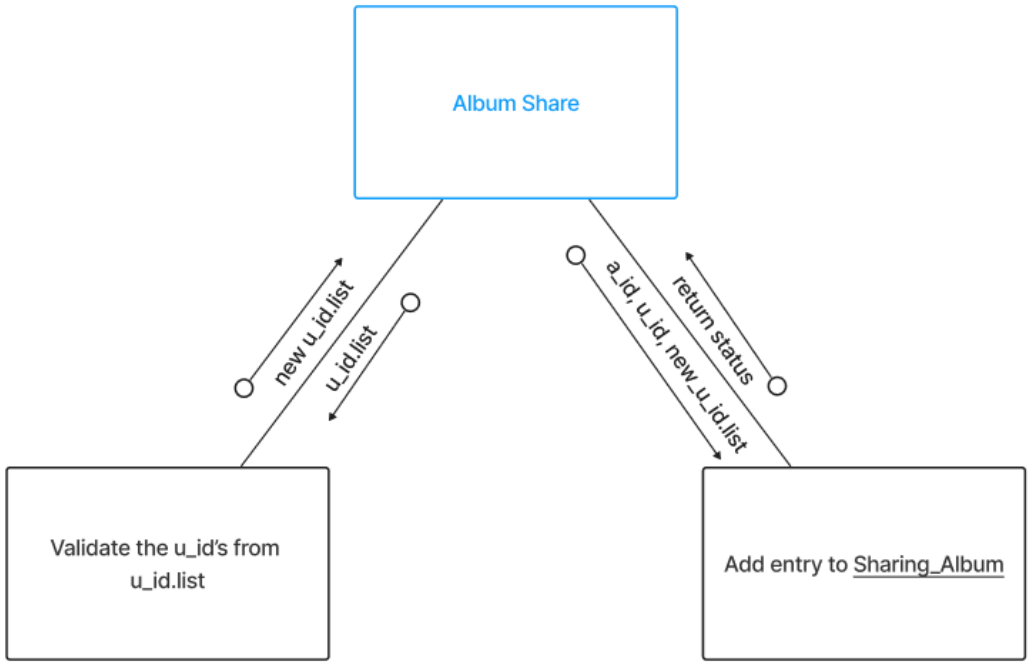
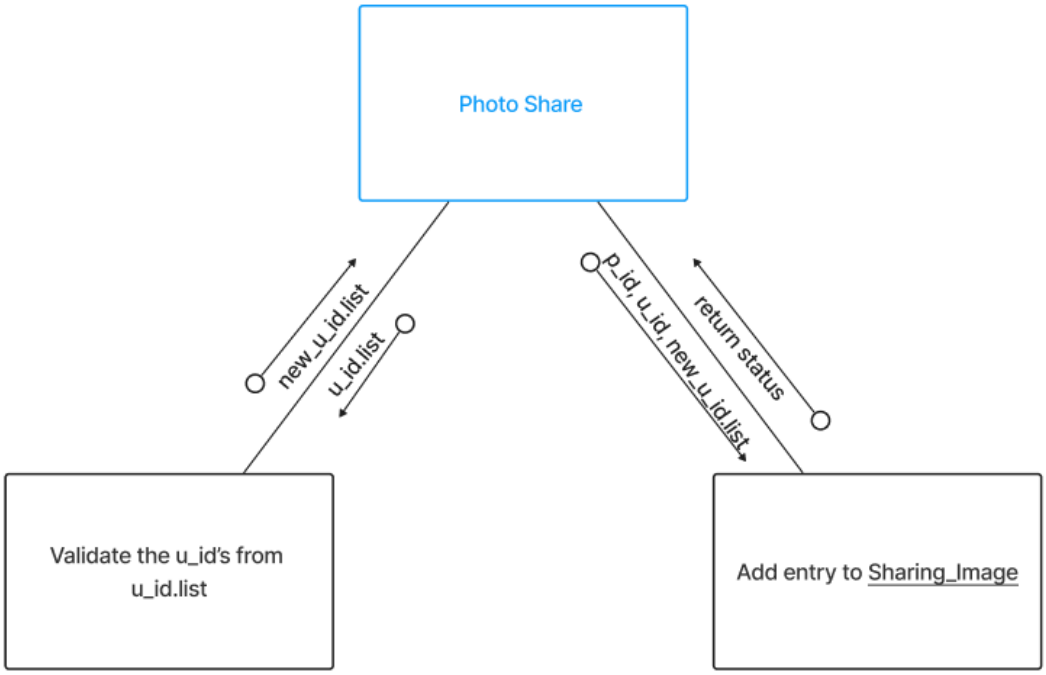
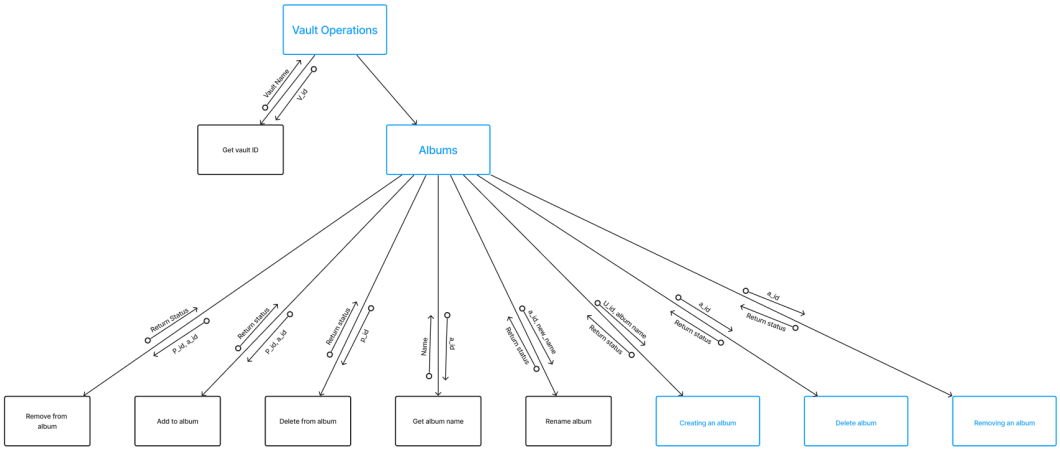


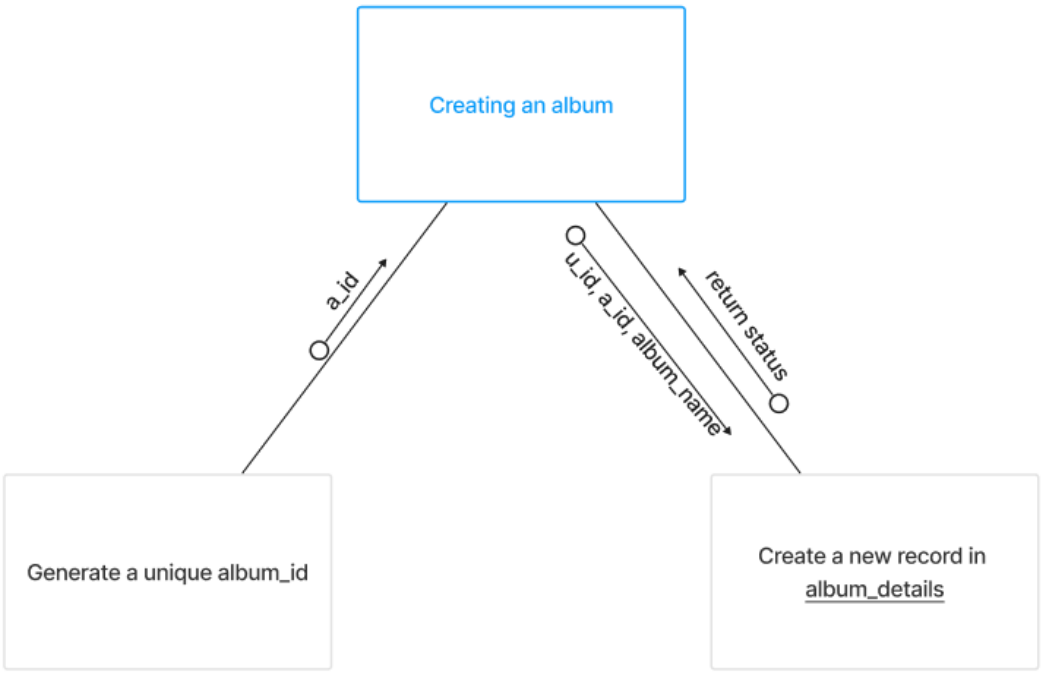
Photo Share



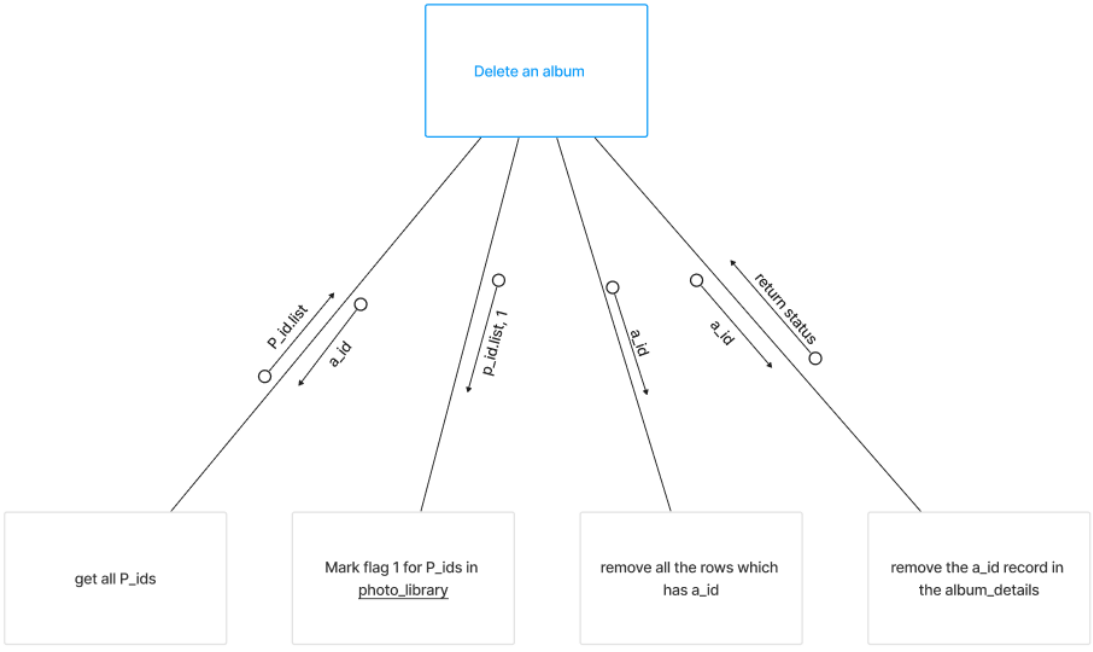
3.3.4 Vault Operations



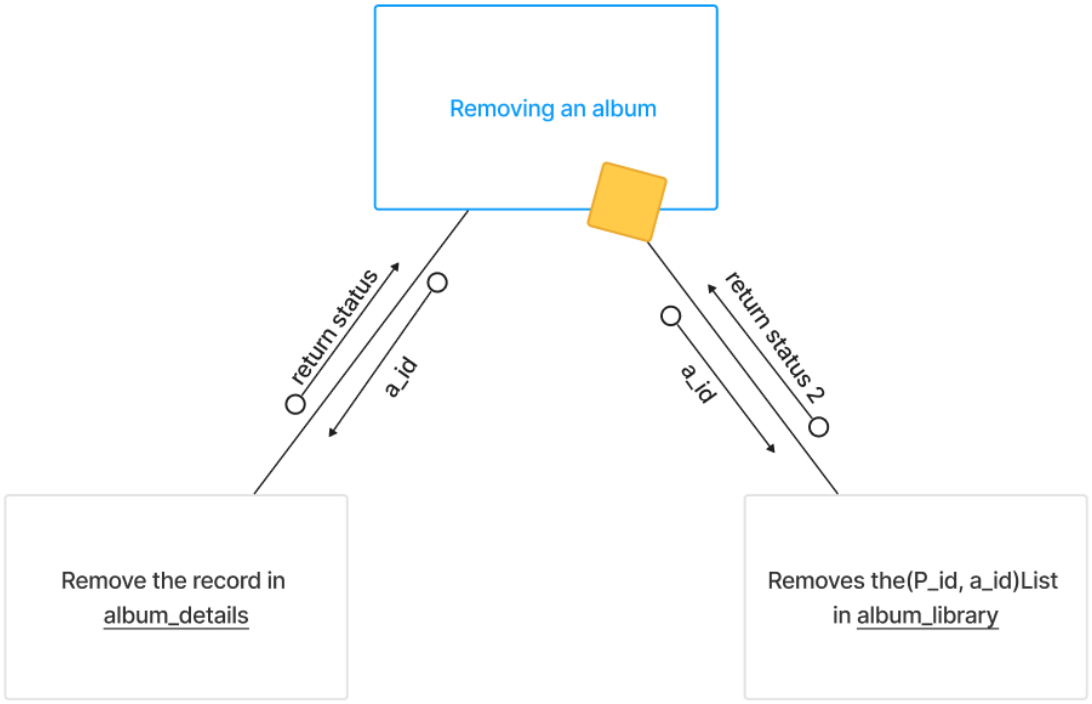
Creating an Album



Delete an Album



Removing an Album



3.3.5 Search Operations

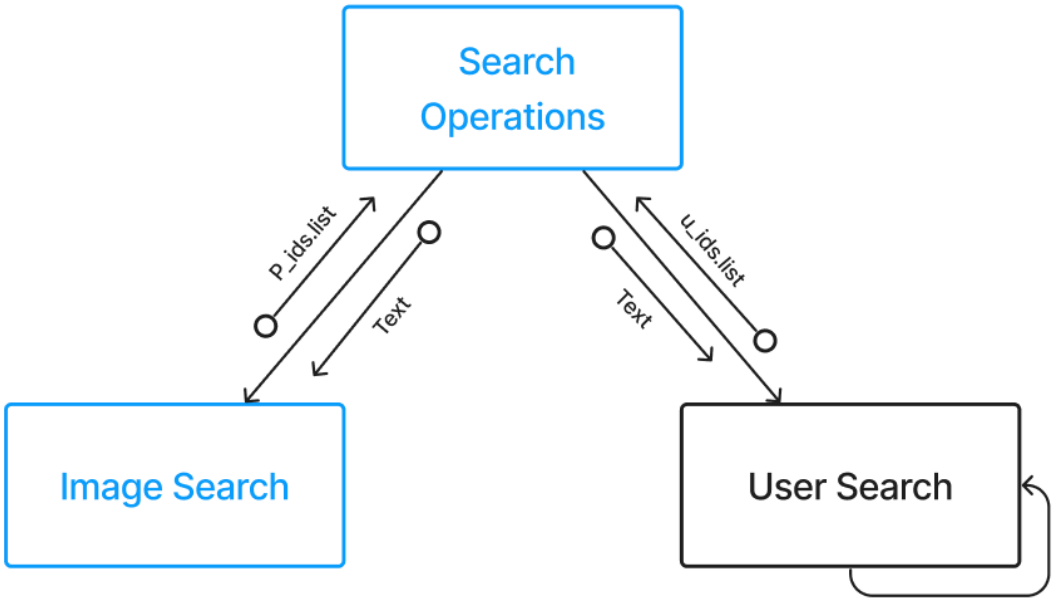
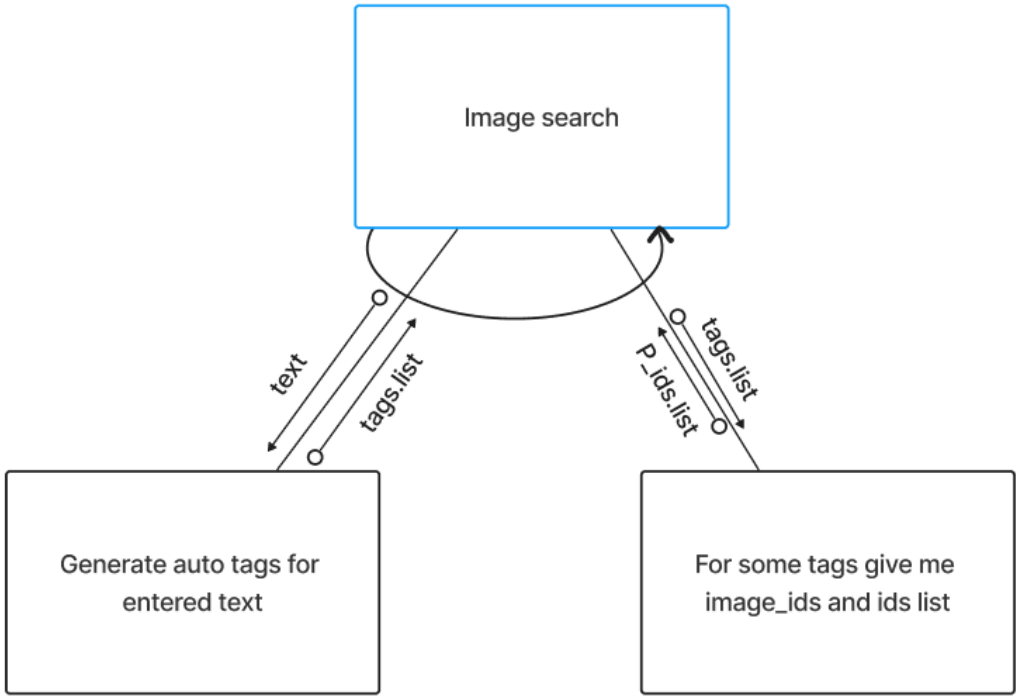
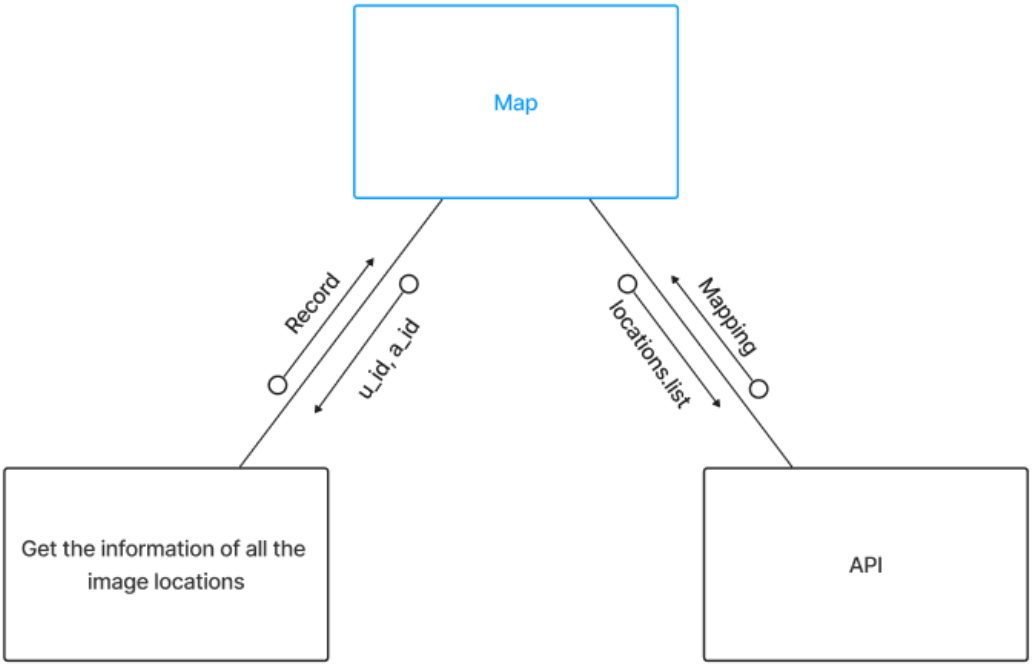


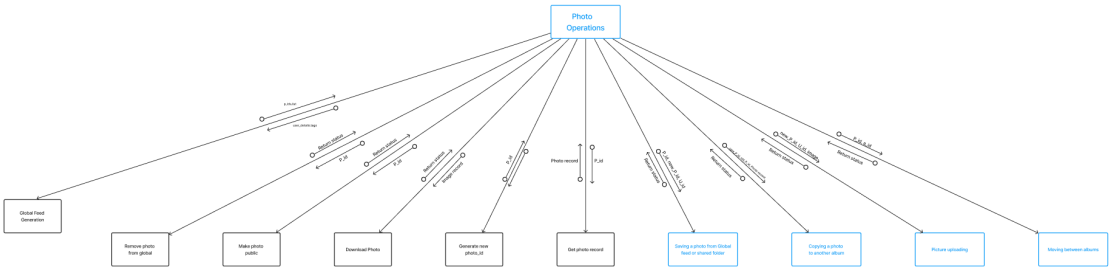
Image Search



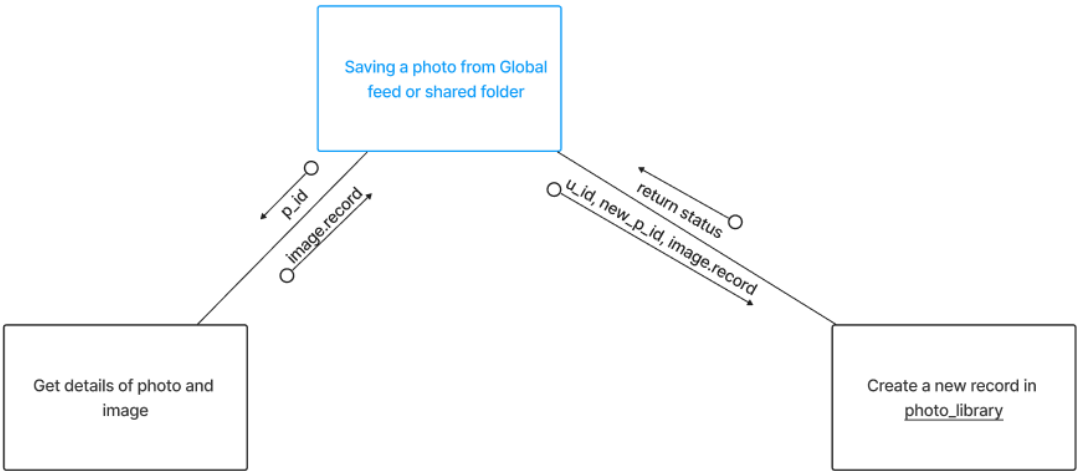
3.3.6 Map



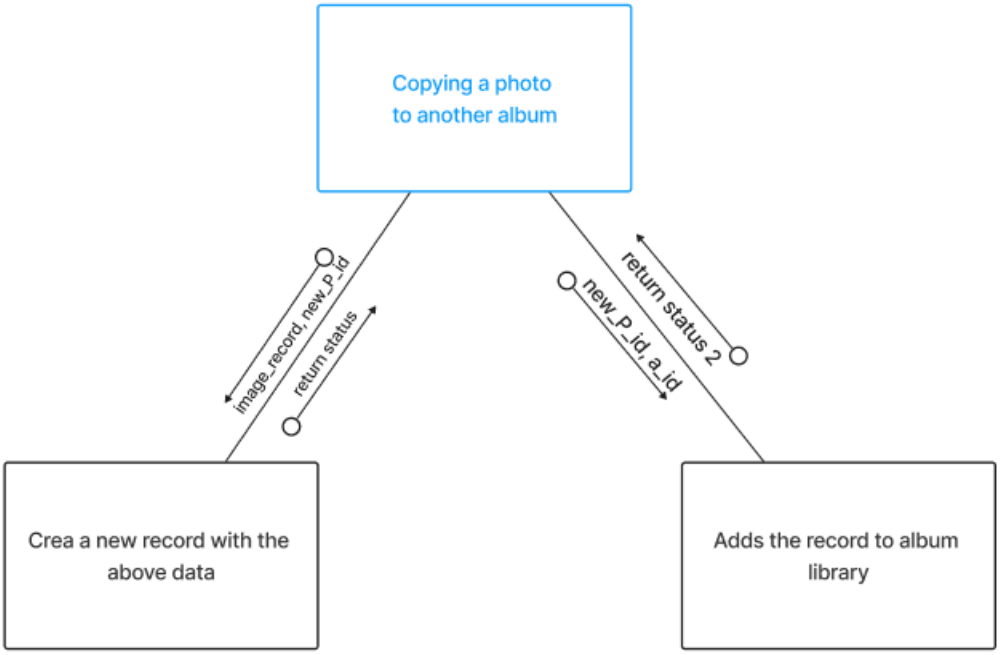
3.3.7 Photo Operations



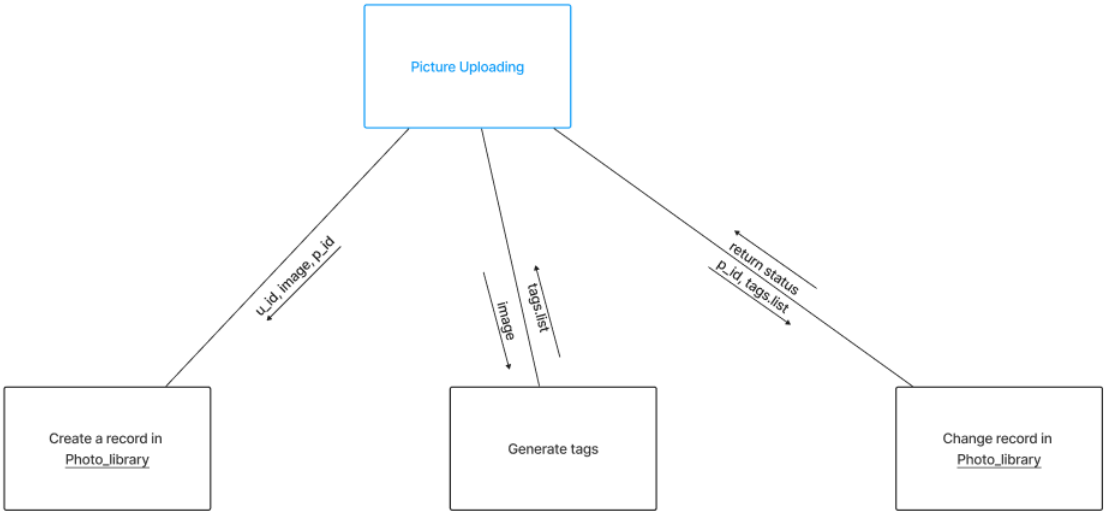
Saving a photo from Global feed or shared folder



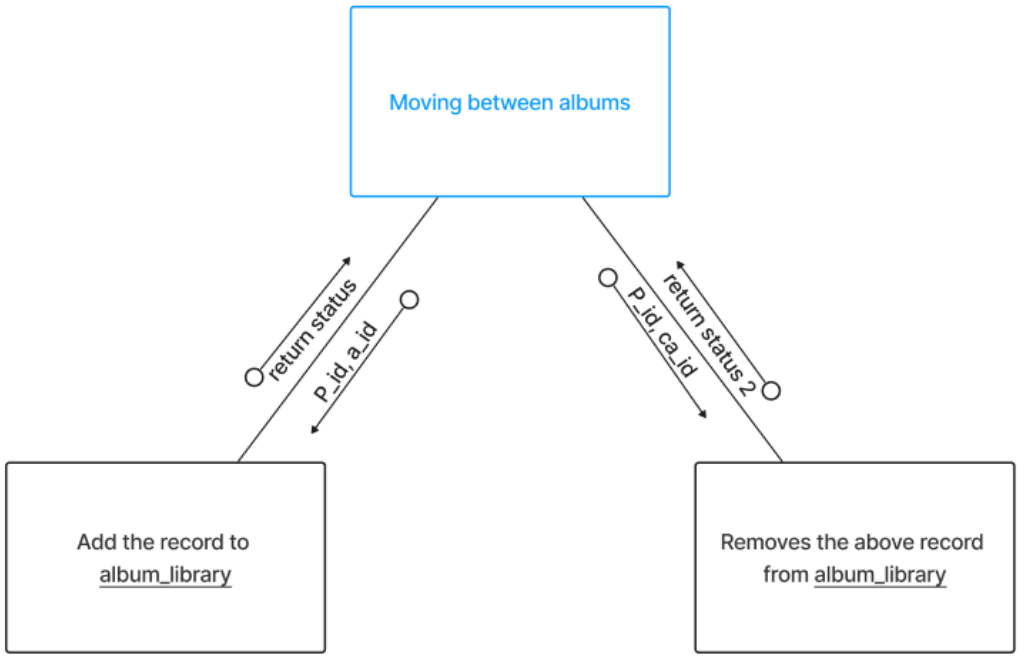
Copying a photo to another album



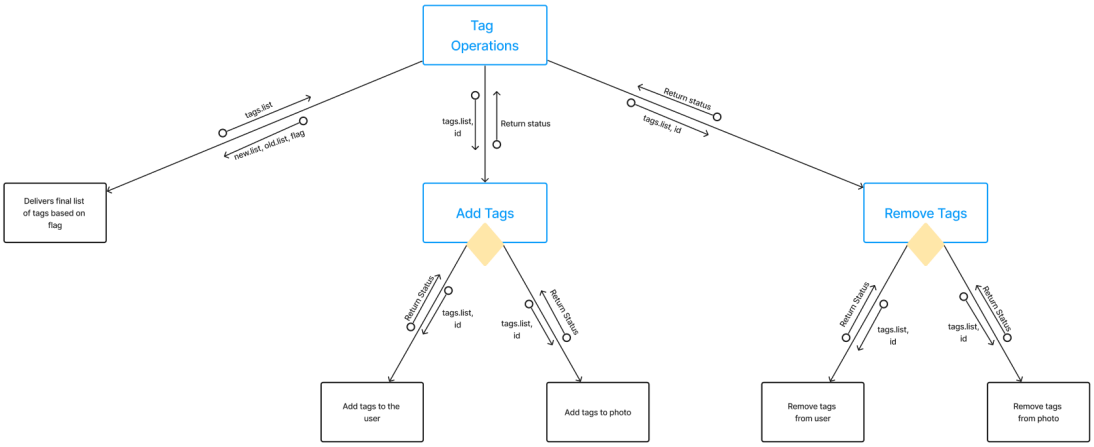
Picture uploading



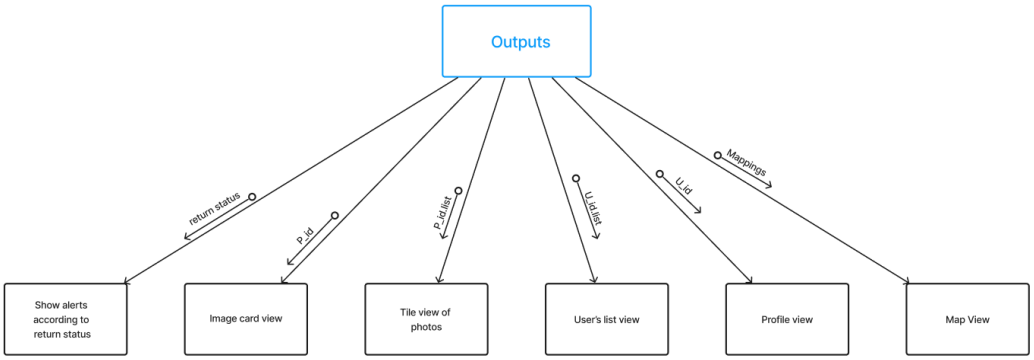
Moving between albums



3.3.8 Tag Operations



3.4 Output Module



4 Design Analysis

4.1 Information About Modules

Name of the module	**Module Type**	**Type of cohesion**	**Justification**	**Estimated Size of Module(in LOC)**	**Degree of Coupling**
Main	Coordinated Module	Sequential Cohesion	As always the sequence of execution between the modules is input, functions, output and as proceed the modules require input from previous modules	1025	Low
Input	Coordinate Module	Logical Cohesion	This module is handles all input related for the application	90	High
Functions	Transform Module	Logical Cohesion	Contains functions that transform data and performs required tasks	785	High
Output	Output Module	Logical Cohesion	Solely dedicated to formatting and presenting output data	150	High
HTML Input	Input Module	Logical Cohesion	This handles all HTML inputs, which is a single, specific function.	50	High
Get Text	Input Module	Functional Cohesion	Handles text inputs exclusively, representing a single cohesive functionality	10	Low

Name of the module	**Module Type**	**Type of cohesion**	**Justification**	**Estimated Size of Module(in LOC)**	**Degree of Coupling**
Get Password	Input Module	Functional Cohesion	Dedicated to password input handling, only does that single job	10	Low
Get Tags List	Input Module	Functional Cohesion	Responsible for retrieving a list of tags	15	Low
Get Image	Input Module	Functional Cohesion	Focuses on image data retrieval	15	Low
User Details	Composite Module	Communica--tional Cohesion	Groups related functionalities that manage different aspects of user details	20	Moderate
Get User ID	Input Module	Functional Cohesion	Fetches user ID based on provided input data of **On-Click**	10	Low
Get Other Details	Input Module	Functional Cohesion	Collects additional user details, shows a clear purpose	10	Low
Get Album ID	Input Module	Functional Cohesion	Retrieves an album ID, based on provided input data of **On-Click**	10	Low
Get Photo ID	Input Module	Functional Cohesion	Aimed at getting the photo ID, based on provided input data of **On-Click**	10	Low
Show Alerts	Output Module	Functional Cohesion	Each function is centered around a singular task, which is to show alerts based on different return statuses	20	Low

Name of the module	**Module Type**	**Type of cohesion**	**Justification**	**Estimated Size of Module(in LOC)**	**Degree of Coupling**
Image Card View	Output Module	Functional Cohesion	This module generates a view for images in a card format, a single, specific task	10	Low
Tile View of Photos	Output Module	Functional Cohesion	The module's sole purpose is to present photos in a tile view	30	Low
User's List View	Output Module	Functional Cohesion	It is designed to display a list view of users which is for only that task	30	Low
Profile View	Output Module	Functional Cohesion	This module is focused on showing the profile details which is a single task	30	Low
Map View	Output Module	Functional Cohesion	This module's role is to provide a mapping view	30	Low
User	Composite Module	Logical Cohesion	The User module serves as the overall category that logically includes all user-related operations	65	High
Folder Operations	Composite Module	Logical Cohesion	All the operations related to folders are in this module	85	High
Shared Operations	Composite Module	Logical Cohesion	It encapsulates all sharing-related functionalities, which are related but separate operations	120	High
Vault Operations	Composite Module	Logical Cohesion	It logically groups together various vault related operations	135	High

Name of the module	**Module Type**	**Type of cohesion**	**Justification**	**Estimated Size of Module(in LOC)**	**Degree of Coupling**
Search Operations	Composite Module	Logical Cohesion	It logically groups together various search-related functionalities	90	High
Map	Composite Module	Procedural cohesion	First metadata is extracted and then mapping is done	50	Moderate
Photo Operations	Transform Module	Logical Cohesion	Has all the operations related to photos	150	High
Tag Operations	Composite Module	Logical Cohesion	This is a broader category that includes different tag-related functionalities, grouped logically	90	High
Delivers final list of tags based on flag	Transform Module	Functional Cohesion	This module has a single task of delivering a processed list of tags	30	Low
Add tags	Composite Module	Logical Cohesion	As includes both adding tags to photos and users	30	High
Remove Tags	Composite Module	Logical Cohesion	As includes both removing tags to photos and users	30	High
Add tags to the user	Transform Module	Functional Cohesion	Its singular task is to add tags to a user	15	Low
Add tags to photo	Transform Module	Functional Cohesion	Its singular task is to add tags to a photo	15	Low
Remove tags from user	Transform Module	Functional Cohesion	Its singular task is to removing tags from a user	15	Low
Remove tags from photo	Transform Module	Functional Cohesion	Solely focused on the task of removing tags from a photo	15	Low

Name of the module	**Module Type**	**Type of cohesion**	**Justification**	**Estimated Size of Module(in LOC)**	**Degree of Coupling**
Remove photo from global	Transform Module	Functional Cohesion	Dedicated to a single operation of removing a photo from the global feed	10	Low
Make photo public	Transform Module	Functional Cohesion	Handles the specific task of changing a photo's status to public	10	Low
Download Photo	Transform Module	Functional Cohesion	Focuses on just downloading the photo	10	Low
Generate new photo_id	Input Module	Functional Cohesion	Creates a new identifier id for a photo	5	Low
Get photo record	Transform Module	Functional Cohesion	Retrieves the photo's data on inputting photo ID	10	Low
Saving a photo from Global feed or shared folder	Composite Module	Sequential Cohesion	First the detail of the photo is obtained then saving is done	20	Low
Copying a photo to another album	Composite Module	Sequential Cohesion	New pid is created and then record is copied	25	Low
Picture uploading	Composite Module	Sequential Cohesion	Required steps are done sequentially	30	Low
Moving between albums	Composite Module	Sequential Cohesion	Required steps are done sequentially	30	Low
Get details of photo and image	Transform Module	Functional Cohesion	It's responsible for retrieving data of a photo	10	Low

Name of the module	**Module Type**	**Type of cohesion**	**Justification**	**Estimated Size of Module(in LOC)**	**Degree of Coupling**
Create a new record in photo_library	Transform Module	Functional Cohesion	This module transforms the state by creating a new record	10	Low
Create a new record with the above data	Transform Module	Functional Cohesion	Performs the operation of creating a new data record	15	Low
Adds the record to album library	Transform Module	Functional Cohesion	Handles the operation of adding a new record to the album library	10	Low
Create a record in Photo_library	Transform Module	Functional Cohesion	This module is responsible for creating a new record	10	Low
Generate tags	Transform Module	Functional Cohesion	It processes an image to generate tags	10	Low
Change record in Photo_library	Transform Module	Functional Cohesion	Alters an existing record	10	Low
Add the record to album_library	Transform Module	Functional Cohesion	Adds a new record to the album_library	15	Low
Removes the above record from album_library	Transform Module	Functional Cohesion	Removes a record from the album_library	15	Low

Name of the module	**Module Type**	**Type of cohesion**	**Justification**	**Estimated Size of Module(in LOC)**	**Degree of Coupling**
Image Search	Transform Module	Sequential Cohesion	It is focused on the task of searching within images after generating tags	60	Low
User Search	Transform Module	Functional Cohesion	It exclusively handles the search operation for user profiles	30	Low
Generate auto tags for entered text	Transform Module	Functional Cohesion	This module transforms input text into a list of tags	40	Low
For some tags give me image_ids and ids list	Transform Module	Functional Cohesion	It takes a list of tags and transforms it into corresponding list of image IDs with the tags	20	Low
Get vault ID	Transform Module	Functional Cohesion	Retrieves the ID for a vault, which is a single, focused input task related to the Vault Operations	10	Low
Albums	Composite Module	Logical Cohesion	This module groups various album-related operations, which are logically related but distinct functions	125	High
Remove from album	Transform Module	Functional Cohesion	Dedicated to the single function of removing a photo from an album	10	Low
Add to album	Transform Module	Functional Cohesion	Focused on the specific function of adding a photo to an album	10	Low

Name of the module	**Module Type**	**Type of cohesion**	**Justification**	**Estimated Size of Module(in LOC)**	**Degree of Coupling**
Delete from album	Transform Module	Functional Cohesion	Handles the single task of moving a photo to bin indicating a singular purpose	15	Low
Get album name	Transform Module	Functional Cohesion	Retrieves the name of an album	10	Low
Rename album	Transform Module	Functional Cohesion	Performs the transformation of changing the album's name	10	Low
Creating an Album	Transform Module	Sequential Cohesion	Involves the action of creating a new album by generating a new album id	20	Low
Delete Album	Transform Module	Procedural Cohesion	As always all the photo ID's from the albums is retrieved and then the corresponding photo is moved to bin	35	Moderate
Removing an Album	Transform Module	Procedural Cohesion	As first the album is removed from the storage and then the further deletion of album data is done	15	Moderate
Generate a unique album_id	Input Module	Functional Cohesion	It performs the single task of generating a unique ID for an album	10	Low
Create a new record in album_details	Transform Module	Functional Cohesion	It handles the creation of a new album record	10	Low
bat get all p_ids	Transform Module	Functional Cohesion	Retrieves all photo IDs from the album	10	Low

Name of the module	**Module Type**	**Type of cohesion**	**Justification**	**Estimated Size of Module(in LOC)**	**Degree of Coupling**
Mark flag 1 for p_id's in photo_library	Transform Module	Functional Cohesion	Marks photos as flagged in the library	10	Low
remove all the rows which has a_id	Transform Module	Functional Cohesion	Removes all album-related records from the database	10	Low
remove the a_id record in the album_details	Transform Module	Functional Cohesion	Deletes the album's metadata from the database	5	Low
Remove the record in album_details	Transform Module	Functional Cohesion	This module is responsible for a single operation removing an album's record from the album details	5	Low
Removes the(p_id, a_id)List in album_library	Transform Module	Functional Cohesion	Dedicated to the specific task of removing a list of photo IDs associated with an album from the album library	10	Low
Get the information of all the image locations	Transform Module	Functional Cohesion	It is focused solely on the task of retrieving location information for all the photos in the album	20	Low
API	Transform Module	Functional Cohesion	It likely serves as an interface for external systems to interact with the map services and giving the mapping outputs	30	Low

Name of the module	**Module Type**	**Type of cohesion**	**Justification**	**Estimated Size of Module(in LOC)**	**Degree of Coupling**
Shared view	Transform Module	Functional Cohesion	Provides a view of shared items	20	Low
Validate U_ids from U_id.list	Transform Module	Functional Cohesion	Handles validation of the input user IDs	10	Low
Photo share	Transform Module	Sequential Cohesion	Input user ID's are validated and then photo sharing is done	30	Low
Album share	Transform Module	Sequential Cohesion	Input user ID's are validated and then album sharing is done	30	Low
Remove shared permission	Transform Module	Sequential Cohesion	Input user ID's are validated and then album permission is done	30	Low
Validate the u_id's from u_id.list	Transform Module	Functional Cohesion	This operation transforms the state of the photo by sharing it	15	Low
Add entry to Sharing_Image	Transform Module	Functional Cohesion	This operation transforms the state of the photo by sharing it	15	Low
Validate the u_id's from u_id.list	Transform Module	Functional Cohesion	This operation transforms the state of the album by sharing it	15	Low
Add entry to Sharing_Album	Transform Module	Functional Cohesion	This operation transforms the state of the album by sharing it	15	Low
Process user id's	Transform Module	Functional Cohesion	verifies the user id's for removing the sharing permission	10	Low

Name of the module	**Module Type**	**Type of cohesion**	**Justification**	**Estimated Size of Module(in LOC)**	**Degree of Coupling**
Update on sharing image	Transform Module	Functional Cohesion	Update is done on the image	10	Low
Update on sharing album	Transform Module	Functional Cohesion	Similar to updating the sharing status of an image, it focuses on albums	10	Low
Follow User	Transform Module	Functional Cohesion	Executes the specific follow function for the user	20	Low
Unfollow User	Transform Module	Functional Cohesion	Executes the specific unfollow function for the user	20	Low
Get list of following	Transform Module	Functional Cohesion	Retrieves the list of users that a particular user is following	15	Low
Login	Transform Module	Sequential Cohesion	The password, username is first validated and then they are saved	10	Low
Validate User	Transform Module	Functional Cohesion	Validates user login details	5	Low
Save Credentials	Transform Module	Functional Cohesion	saves login credentials of the user	5	Low
Remove image from shared	Transform Module	Functional Cohesion	Removes a photo from shared library	5	Low
Seeing photos from hidden folder	Transform Module	Sequential Cohesion	This submodule is involved in retrieving and displaying photos from a hidden folder after validation	20	Low

Name of the module	Module Type	Type of cohesion	Justification	Estimated Size of Module(in LOC)	Degree of Coupling
Add to hidden folder	Transform Module	Functional Cohesion	Adds photos to a hidden folder	10	Low
Remove from hidden folder	Transform Module	Functional Cohesion	Removes photos from a hidden folder	10	Low
Add to favorites	Transform Module	Functional Cohesion	Involves marking a photo or album as a favorite	10	Low
Remove from Bin	Transform Module	Functional Cohesion	This function handles the removal of photos from a Bin	15	Low
Recovery from Bin	Transform Module	Functional Cohesion	Used for restoring items that have been previously moved to the Bin	15	Low
Validate Password	Transform Module	Functional Cohesion	Performs the single task of validating a password	10	Low
Get image IDs list	Transform Module	Functional Cohesion	Retrieves a list of image IDs, which are in the hidden folder	10	Low

4.2 Number of Types of Modules count

Name of the Module	Count
Input Modules	11
Output Modules	7
Coordinate Modules	1
Transform Modules	69
Composite Modules	15

4.3 Top-3 Fan in and Fan Out

Name of the Module	Fan In	Fan Out
Photo operations Module	1	9
Albums	1	8
Functions Module	1	7

4.4 Complex and Error prone

The Mean of all the Dc's of the modules : 14.52
The Standard Deviation of all the Dc's of the modules : 83.36

4.4.1 Error Prone Modules

Condition: $Dc > Mean + StdDev$

Name of the Module	Reason
Function Module	Since the Dc for the Function Module is 786, which is much higher than 97.88(i.e., mean + std dev) this module is classified as error-prone. It exceeds the threshold for being considered normal or even complex. This might be because of the fact that as function is the module ie., the logical cohesion module for all the functions hence it is possible to have more errors

4.4.2 Complex Modules

Condition: $Mean < Dc < Mean + StdDev$

Name of the Module	Reason
Tag Operations Module	14.52 < Dc(=18) < 97.88
User Module	14.52 < Dc(=20) < 97.88
Folder Operations Module	14.52 < Dc(=35) < 97.88
Vault Operations Module	14.52 < Dc(=47) < 97.88
Albums Module	14.52 < Dc(=48) < 97.88
Input Module	14.52 < Dc(=52) < 97.88
Photo Operations Module	14.52 < Dc(=63) < 97.88
Share Operations Module	14.52 < Dc(=77) < 97.88

4.5 Total LOC

Name of the Module	LOC
Main Module	1025
Input Module	90
Functions Module	785
Output Module	150

5 Design Specification

We have 3 main modules in our applicaiton they are:

- input.js: This module contains all the functions that are related to input handling.
- functions.js: This module contains all the functions that are the business logic for this application.
- output.js: This module contains all the functions that are responsible for displaying the output to the user.

5.1 Main Structures in the Application

5.2 input.js

```
// Outer function for handling HTML input

// Input for this would be different DOM elements
function HTMLInput() {
  // Inner function to get text input
  function getText() {
    // Code to get text input
  }

  // Inner function to get password input
  function getPassword() {
    // Code to get password input
  }

  // Inner function to get a list of tags
  function getTagsList() {
    // Code to get tags list
  }

  // Inner function to get image
  function getImage() {
    // Code to get image input
  }

  // Returning the inner functions so they can be used outside of
     HTMLInput
```

```
26    return { getText, getPassword, getTagsList, getImage };
27  }
28
29  // Outer function for handling user details
30  function UserDetails() {
31    // Inner function to get user ID
32    function getUserID() {
33      // Code to get user ID
34    }
35
36    // Inner function to get other user details
37    function getOtherDetails() {
38      // Code to get other details
39    }
40    // Returning the inner functions so they can be used outside of
          UserDetails
41    return { getUserID, getOtherDetails };
42  }
43
44  // function for handling the retrieval of a album ID
45  function GetAlbumID() {
46
47      return a_id ;
48  }
49
50  // Outer function for confirming retrieval of the photo ID
51  function GotPhotoID() {
52
53      return p_id ;
54  }
```

5.3 functions.js

```
1
2   // User-related operations
3   function userOperations(u_id_follow, u_id, username, password) {
4     // Define user-related inner functions here, using the arguments
          as needed
5
6   return {
7       // Return inner functions here
8     };
9   }
10
11  // Folder-related operations
12  function folderOperations(u_id, id, password, p_id) {
13    // Define folder-related inner functions here
14
```

```
  return {
    // Return inner functions here
  };
}

// Share-related operations
function shareOperations(u_id, u_id.list, p_id, a_id, ru_id.list,
    shared_ids.list) {
  // Define share-related inner functions here, using the
     arguments as needed


  return {
    // Return inner functions here
  };
}

// Vault-related operations
function vaultOperations(v_id, p_id, a_id, new_name, album_name,
    u_id) {
  // Define vault-related inner functions here, using the
     arguments as needed

  return {
    // Return inner functions here
  };
}

// Search-related operations
function searchOperations(text) {
  // Define search-related inner functions here

  return {
    // Return inner functions here
  };
}

// Map-related operations
function mapOperations(u_id, a_id) {
  // Define map-related inner functions here

  return {
    // Return inner functions here
  };
}

// Photo-related operations
function photoOperations(p_id, img_record, u_id) {
  // Define photo-related inner functions here, using the
```

```
      arguments as needed
  // ...
  return {
    // Return inner functions here
  };
}

// Tag-related operations
function tagOperations(new.list, old.list, flag, id) {
  // Define tag-related inner functions here, using the arguments
     as needed

  return {
    // Return inner functions here
  };
}
```

5.3.1 userOperations

```
// Function to handle following a user
function followUser(u_id_follow_u_id) {
    // Code to follow user
    return return_status ;
}

// Function to get the list of users that the current user is
   following
function getListOfFollowing(u_id_list) {
    // Code to get list of following
    return []; // Should return an array of user IDs
}

// Function to handle unfollowing a user
function unfollowUser(u_id_unfollow_u_id) {
    // Code to unfollow user
    return return_status ;

}

// Login module
function login() {

  // Inner function to validate a user's login credentials
  function validateUser(username, password) {
    // Code to validate user
    return return_status ;
  }

  // Inner function to save the user's credentials
```

```
  function saveCredentials(username, password) {
    // Code to save credentials
    return return_status ;
  }

  // Returning the inner functions to be used outside of login
  return { validateUser, saveCredentials };
}
```

5.3.2 folderOperations

```

// Function for removing an image from a shared folder
function removeImageFromShared(u_id, p_id) {
  // Code to remove an image from shared folder

}

// Function for adding an image to a hidden folder
function addToHiddenFolder(p_id) {
  // Code to add a photo to hidden folder

}

// Function for removing an image from a hidden folder
function removeFromHiddenFolder(p_id) {
  // Code to remove from hidden folder

}

// Function for adding an image to favorites
function addToFavorites(p_id) {
  // Code to add to favorites

}

// Function for removing an image from the bin
function removeFromBin(p_id) {
  // Code to remove from bin

}

// Function for recovering an image from the bin
function recoverFromBin(p_id) {
  // Code for recovery from bin

}

// Function for seeing photos from a hidden folder
```

```
function seeingPhotosFromHiddenFolder(u_id, v_id) {
  // Inner function to validate password
  function validatePassword(u_id, password) {
    // Credentials validation logic here

  }

  // Inner function to get image IDs list
  function getImageIDsList(u_id, v_id) {
    // Code to get image list here

  }

  // Returning inner functions
  return { validatePassword, getImageIDsList };
}
```

5.3.3 shareOperations

```
function shareOperations() {
  // Shows shared images/albums to a user
  function sharedView(images_list_albums_list, u_id) {
    // Code to show shared images/albums

  }

  // Function to validate user IDs from a list
  function validateUserIdsFromList(u_id_list) {
    // Code to validate user IDs

  }

  // Function to share a photo
  function photoShare(u_id.list) {
    // Inner function to validate the u_id's from u_id.list
    function validateUids(u_id.list) {
      // Code to validate user IDs for photo sharing
    }

    // Inner function to add an entry to sharing the photo
    function addEntryToSharingPhoto(p_id, u_id,new_u_id.list) {
      // Logic to add an entry for a shared photo

      return return_status;
    }

    // Returning inner functions
    return { validateUids, addEntryToSharingPhoto };
  }
```

```
31
32    // Function to share an album
33    function albumShare(u_id.list) {
34      // Inner function to validate the u_id's from u_id.list
35      function validateUids(u_id.list) {
36        // Code to validate user IDs for album sharing
37
38      }
39
40      // Inner function to add an entry to sharing the album
41      function addEntryToSharingAlbum(a_id, u_id, new_u_id.list) {
42        // Code to add an entry for a shared album
43        return return_status;
44      }
45
46      // Returning inner functions
47      return { validateUids, addEntryToSharingAlbum };
48    }
49
50    // Function to remove shared permissions
51    function removeSharedPermission(ru_i.list,shared_ids.list, id,
          u_id) {
52
53    // Function to Process user id
54    function processUserIDS(ru_id.list, shared_ids.list) {
55      // Code to Process remover list and current shared list
56    }
57
58    // Function to update on sharing an image
59    function updateOnSharingImage(new_u_id.list, p_id) {
60      // Code to update sharing permissions for an image
61    }
62
63    // Function to update on sharing an album
64    function updateOnSharingAlbum(new_u_id.list, a_id) {
65      // Code to update sharing permissions for an album
66
67    }
68      return {processUserIDS, updateOnSharingImage,
            updateOnSharingAlbum}
69    }
70
71
72    // Returning functions for use in other parts of the application
73    return {
74      sharedView,
75      validateUserIdsFromList,
76      photoShare,
77      albumShare,
```

```
78      removeSharedPermission
79    };
80  }
```

5.3.4 vaultOperations

```
1
2     // Inner function to get vault ID
3     function getVaultID(vaultName) {
4       // Code to get vault ID
5       return v_id; // Should return the vault ID
6     }
7
8     // Function that handles all the album operations
9     function albums(p_id, a_id, u_id, album_name, new_name) {
10      function removeFromAlbum(a_id) {
11          // Code to remove (photo_id, album_id) records from
                album_library
12          return status; // Return true if operation was successful
13      }
14
15      // Function that handles adding a photo to album
16      function addToAlbum(p_id, a_id){
17          // Code for adding the photo to album
18
19          return status ;
20      }
21
22      // Removes a photo from the album
23      function deleteFromAlbum(p_id){
24          // Code for deleting a photo from album
25
26          return status
27      }
28
29      // Function to get an Album Name
30      function getAlbumName(a_id){
31          // Code to get the album name based on a_id
32
33          return album_name ;
34      }
35
36      // Function rename an Album
37      function renameAlbum(a_id, new_name){
38          // Code rename an album
39
40          return status ;
41      }
42
```

```
43      function creatigAnAlbum(u_id, album_name){
44
45          // Function to generate a unique id for the new album
46          function generateUnqID(){
47
48              // code for generating the unique id
49              return unq_id ;
50          }
51
52          // Function for creating new record in album_details
53          function createNewRecord(u_id, a_id, album_name){
54              // code for creating new record
55              return status ;
56          }
57
58          return {generateUnqID, createNewRecord}
59      }
60
61      function deleteAlbum(a_id){
62          // Function to generate all the photo id's in the album
63          function getAllPIDS(a_id){
64
65              return [] ; // Arrays of pids
66          }
67
68          // Function used for marking photo's flag to 1
69          function markFlag1(p_id.list){
70
71          }
72
73      // function for removing all the rows having a_id
74          function removeAllRowsHavingA_id(a_id){
75
76          return status
77          }
78
79      // Function for removing a_idin album details
80          function removeAIDInAlbumDetails(a_id){
81              return status ;
82          }
83
84          return {getAllPIDS, markFlag1, removeAllRowsHavingA_id,
                removeAIDInAlbumDetails}
85      }
86
87
88      // Function used for removing an album
89      function removingAnAlbum(a_id){
90
```

```
91          // Function for removing record in album_details
92          function removeRecordAlbumDetails(a_id){
93              // Code for removing the a_id details
94
95              return status
96          }
97
98          // Function for removing (p_id, a_id) in album library
99          function removeP_id_A_id(a_id){
100             // Code for removing (p_id,a_id) from album_library
101
102             return status
103         }
104
105         return {removeRecordAlbumDetails, removeP_id_A_id}
106     }
107
108     return {removeFromAlbum, addToAlbum, deleteFromAlbum,
            getAlbumName, renameAlbum, creatigAnAlbum, deleteAlbum,
            removingAnAlbum}
109   }
```

5.3.5 searchOperations

```
1  // Function used to search for users
2  function searchUser(text) {
3      // Code for auto completeion and suggestion
4
5      return [] ; // returns list of users whenever searched
6  }
7
8  // Function used to search for photos
9  function imageSearch(text) {
10     // Function used to generate auto tags for a  given text
11     function generateAutoTags(text){
12         // Code for Auto Generation
13
14         return [] ; // returns array of tags auto generated
15     }
16
17     // Function that outputs
18     function giveImages(tags.list){
19         // Code for retrieving images with the input tags array
20
21         return [] ; // returns array of images ID's
22     }
23     return {generateAutoTags, giveImages}
24 }
```

5.3.6 mapOperations

```
// Function used to get the location metadata of all the photos
   from an image
function getMetadataInfo(u_id, a_id){
    // Code for processing all the data and then returns all the
       locations metadata

    return []; // This is the list of location metadata
}

// Function used to get Mappings data using API
function api(locations.list){

    // Code for API call, data storage, and return

    return [] ; // This is the list of mappings
}
```

5.3.7 photoOperations

```
  // Function to remove a photo from global visibility
  function removePhotoFromGlobal(p_id) {
    // Code to remove photo
    return 'return_status'; // return status after operation
  }

  // Function to make a photo public
  function makePhotoPublic(p_id) {
    // Code to make photo public
    return 'return_status'; // return status after operation
  }

  // Function to download a photo
  function downloadPhoto(p_id) {
    // Code to download photo
    return 'return_status'; // return status after operation
  }

  // Function to generate a new photo ID
  function generateNewPhotoID() {
    // Code to generate new ID
    return 'new_p_id'; // return new photo ID
  }

  // Function to get a photo record
  function getPhotoRecord(p_id) {
    // Code to get photo details
```

```
    return 'photo_record'; // return photo record/details
  }

// Function to save a photo from the global feed or shared folder
function savePhotoFromGlobalOrShared(p_id, img_record, u_id) {
  // Inner function to get details of photo and image
  function getPhotoAndImageDetails(p_id) {
    // Code to get photo and image details
  }

  // Inner function to create a new record in photo library
  function createPhotoRecord(u_id, img_record) {
    // Code to create new photo record
  }

  return {getPhotoAndImageDetails, createPhotoRecord}
}

// Function to copy a photo to another album
function copyPhotoToAnotherAlbum(p_id, a_id) {
  // Inner function to create a new record with photo data
  function createNewRecordWithPhotoData(p_id, a_id) {
    // Code to create new record
  }

  // Inner function to add the record to album library
  function addRecordToAlbumLibrary(record) {
    // Code to add record to album library
  }

  return {createNewRecordWithPhotoData, addRecordToAlbumLibrary}
}

// Function to handle picture uploading
function pictureUploading(u_id, image) {
  // Inner function to create a record in photo library
  function createPhotoLibraryRecord(u_id, image) {
    // Code to create photo library record
  }

  // Inner function to generate tags for the image
  function generateImageTags(image) {
    // Code to generate tags
  }

  // Inner function to change the record in photo library
  function updatePhotoLibrary(record, tags) {
    // Code to update photo library
  }
```

```
77
78    return {createPhotoLibraryRecord, generateImageTags,
          updatePhotoLibrary}
79
80  }
81
82  // Function to move a photo between albums
83  function moveBetweenAlbums(p_id, fromAlbumId, toAlbumId) {
84    // Inner function to add the record to the new album library
85    function addToNewAlbum(p_id, toAlbumId) {
86      // Code to add to new album
87    }
88
89    // Inner function to remove the record from the old album
          library
90    function removeFromOldAlbum(p_id, fromAlbumId) {
91      // Code to remove from old album
92    }
93
94    return {addToNewAlbum, removeFromOldAlbum}
95  }
```

5.3.8 tagOperations

```
1     // function to add tags
2     function AddTags(tagsList, id) {
3       // Inner function to add tags to the user
4       function addUserTags(tagsList, id) {
5         // Logic to add tags to a user's profile
6         return 'return status for adding tags to user';
7       }
8
9       // Inner function to add tags to photo
10      function addPhotoTags(tagsList, id) {
11        // Logic to add tags to a photo's metadata
12        return 'return status for adding tags to photo';
13      }
14
15      // Determine the type of tag addition based on id type and
            call the appropriate function
16      if (id === 'u_id') {
17        return addUserTags(tagsList, id);
18      } else if (id === 'p_id') {
19        return addPhotoTags(tagsList, id);
20      } else {
21        return 'return status for unknown id type';
22      }
23    }
24
```

```
25    // Inner function to remove tags
26    function RemoveTags(tagsList, id) {
27      // Inner function to remove tags from the user
28      function removeUserTags(tagsList, id) {
29        // Logic to remove tags from a user's profile
30        return 'return status for removing tags from user';
31      }
32
33      // Inner function to remove tags from photo
34      function removePhotoTags(tagsList, id) {
35        // Logic to remove tags from a photo's metadata
36        return 'return status for removing tags from photo';
37      }
38
39      // Determine the type of tag removal based on id type and call
            the appropriate function
40      if (id === 'u_id') {
41        return removeUserTags(tagsList, id);
42      } else if (id === 'p_id') {
43        return removePhotoTags(tagsList, id);
44      } else {
45        return 'return status for unknown id type';
46      }
47    }
48
49  }
```

5.4 output.js

```
1   const OutputsModule = {
2     // Function to show alerts based on a return status
3     showAlerts: function(returnStatus) {
4       if (returnStatus === 'success') {
5         // Show success alert
6       } else {
7         // Show error or other status alert
8       }
9     },
10
11    // Function to display images in a card view format
12    imageCardView: function(p_id) {
13      // Logic to display an image in card view using the photo ID
14    },
15
16    // Function to display images in a tile view format
17    tileViewOfPhotos: function(p_id_list) {
18      // Logic to display multiple images in tile view using a list
            of photo IDs
```

```
  },

  // Function to display a list view of users
  usersListView: function(u_id_list) {
    // Logic to display a list of users using a list of user IDs
  },

  // Function to display user profile
  profileView: function(u_id) {
    // Logic to display user's profile using the user ID
  },

  // Function to display a map with certain mappings
  mapView: function(mappings) {
    // Logic to display a map with locations or other mapping data
  }
};
```