|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | |  | | **MINISTRY OF EDUCATION AND TRAINING** | | | **FPT UNIVERSITY** | | |
| Capstone Project Document |
| Family Album Sharing System |
| |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | **Group 11** | | | **Group member** | Huỳnh Trần Minh Nhựt – Leader – SE60936  Tô Đình Tân – Member – SE60926 | | **Supervisor** | Nguyễn Huy Hùng | | **Ext. Supervisor** | N/A | | **Capstone Project code** | FAMSAM | | |
| – Ho Chi Minh City, September 2014 – |

Table of Contents

[A. INTRODUCTION 5](#_Toc405647112)

[I. PROJECT INFORMATION 5](#_Toc405647113)

[II. OVERVIEW AND SOLUTION 5](#_Toc405647114)

[B. SOFTWARE PROJECT MANAGEMENT PLAN 6](#_Toc405647115)

[I. PROBLEM DEFINITION 6](#_Toc405647116)

[1. NAME OF THIS CAPSTONE PROJECT 6](#_Toc405647117)

[2. PROBLEM ABSTRACT 6](#_Toc405647118)

[3. PROBLEM OVERVIEW 6](#_Toc405647119)

[II. PROJECT ORGANIZATION 7](#_Toc405647120)

[1. SOFTWARE PROCESS MODEL 7](#_Toc405647121)

[2. ROLES AND RESPONSIBILITIES 8](#_Toc405647122)

[3. TOOLS AND TECHNIQUES 8](#_Toc405647123)

[III. PROJECT MANAGEMENT PLAN 9](#_Toc405647124)

[1. TASKS 9](#_Toc405647125)

[2. DELIVERABLES 10](#_Toc405647126)

[3. ALL MEETING MINUTES 10](#_Toc405647127)

[C. SOFTWARE REQUIREMENTS SPECIFICATIONS 11](#_Toc405647128)

[I. USER REQUIREMENT SPECIFICATION 11](#_Toc405647129)

[1. WEBSITE USER REQUIREMENTS 11](#_Toc405647130)

[2. FAMILY OWNER REQUIREMENTS 11](#_Toc405647131)

[3. ADMINISTRATOR REQUIREMENTS 11](#_Toc405647132)

[II. SYSTEM REQUIREMENT SPECIFICATION (SPECIFIC REQUIREMENTS) 12](#_Toc405647133)

[1. SYSTEM FEATURES 12](#_Toc405647134)

[2. USE CASE DIAGRAM AND USE CASE SPECIFICATION 14](#_Toc405647135)

[3. SOFTWARE SYSTEM ATTRIBUTES 55](#_Toc405647136)

[4. ENTITY RELATIONSHIP DIAGRAM 56](#_Toc405647137)

[D. SOFTWARE DESIGN DESCRIPTION 57](#_Toc405647138)

[*I.* *DESIGN OVERVIEW* 57](#_Toc405647139)

[*II.* *SYSTEM ARCHITECTURAL DESIGN* 57](#_Toc405647140)

[1. CHOICE OF SYSTEM ARCHITECTURE 57](#_Toc405647141)

[*III.* *PACKAGE DIAGRAM* 60](#_Toc405647142)

[1. WEB CLIENT PACKAGE DIAGRAM 60](#_Toc405647143)

[2. SERVER PACKAGE DIAGRAM 61](#_Toc405647144)

[*IV.* *CLASS DIAGRAM* 62](#_Toc405647145)

[1. MANAGE STORY CLASS DIAGRAM 62](#_Toc405647146)

[2. MANAGE ALBUM CLASS DIAGRAM 63](#_Toc405647147)

[3. FAMILY CLASS DIAGRAM 64](#_Toc405647148)

[4. NOTIFICATION CLASS DIAGRAM 65](#_Toc405647149)

[*V.* *SEQUENCE DIAGRAM* 66](#_Toc405647150)

[1. LOGIN 66](#_Toc405647151)

[2. SERVER AUTHORIZE 67](#_Toc405647152)

[3. CREATE STORY 68](#_Toc405647153)

[4. INVITE A PERSON TO A FAMILY 68](#_Toc405647154)

[5. SELECT PHOTO TO UPLOAD 70](#_Toc405647155)

[6. UPLOAD PHOTO 70](#_Toc405647156)

[7. SHARING A POST (STORY/ALBUM) 72](#_Toc405647157)

[8. CREATE THUMBNAIL IMAGE 72](#_Toc405647158)

[9. NOTIFICATION 73](#_Toc405647159)

[10. CALCULATE TF VALUES 74](#_Toc405647160)

[11. RECOMMEND SERVICE 75](#_Toc405647161)

[*VI.* *USER INTERFACE DESIGN* 75](#_Toc405647162)

[1. LOGIN (SIGN IN) SCREEN 75](#_Toc405647163)

[2. REGISTER (SIGN UP) SCREEN 76](#_Toc405647164)

[3. HOMEPAGE SCREEN 77](#_Toc405647165)

[4. USER PERSONAL INFORMATION SCREEN 78](#_Toc405647166)

[5. FAMILY HOMEPAGE SCREEN 79](#_Toc405647167)

[6. CREATE STORY SCREEN 81](#_Toc405647168)

[7. CREATE ALBUM SCREEN 82](#_Toc405647169)

[8. UPLOAD PHOTO SCREEN 83](#_Toc405647170)

[9. STORY DETAIL SCREEN 83](#_Toc405647171)

[10. PHOTO DETAIL SCREEN 83](#_Toc405647172)

[VII. DATABASE DESIGN 84](#_Toc405647173)

[VIII. ALGORITHMS DESCRIPTION 87](#_Toc405647174)

[1. DETECT BLURRY IMAGE 87](#_Toc405647175)

[2. RECOMMEND SIMILAR POST (STORY/ALBUM) BASED ON USER’S LIKES 88](#_Toc405647176)

[IX. REFERENCES FOR PART D 88](#_Toc405647177)

[E. SOFTWARE TEST DOCUMENTATION 89](#_Toc405647178)

[I. INTRODUCTION 89](#_Toc405647179)

[3. SYSTEM OVERVIEW 89](#_Toc405647180)

[4. TEST APPROACH 89](#_Toc405647181)

[II. TEST PLAN 89](#_Toc405647182)

[1. FEATURES TO BE TESTED 89](#_Toc405647183)

[2. FEATURES NOT TO BE TESTED 90](#_Toc405647184)

[3. TESTING TOOLS AND ENVIRONMENT 90](#_Toc405647185)

[*III.* *TEST CASES* 92](#_Toc405647186)

[1. GUEST – COMMON FUNCTIONS 92](#_Toc405647187)

[2. LOGGED IN USER – FAMILY FUNCTIONS 93](#_Toc405647188)

[3. LOGGED IN USER – GENERAL POST (STORY/ALBUM) FUNCTIONS 93](#_Toc405647189)

[4. LOGGED IN USER – SOCIAL NETWORK FUNCTIONS 93](#_Toc405647190)

[*IV.* *CHECKLISTS* 95](#_Toc405647191)

[1. CHECKLIST OF VALIDATION 95](#_Toc405647192)

[2. SUBMISSION CHECKLIST 95](#_Toc405647193)

[*V.* *OTHER MATERIAL* 95](#_Toc405647194)

[1. TEST RESULT 95](#_Toc405647195)

[2. INCIDENT REPORT 95](#_Toc405647196)

# INTRODUCTION

## PROJECT INFORMATION

* Project name: Family Album Sharing System
* Project code: FAMSAM, this also is used to name the whole system.
* Product Type: Web Application.

## OVERVIEW AND SOLUTION

Nowadays, sharing photos through the internet has become more and more popular. At present, there are several service which provide photo album sharing feature like Facebook, Instagram, Flickr. However, they only support to share photos randomly, not fully support specific characteristics of each family. As result of the fact, users are on demands of a service for family to share their moment such as wedding, baby growing, ceremonies and so on.

Therefore, the system is created to provide functions like creating albums with stories, sharing albums with other families and users can comment on photos, albums or stories. The system will make families save their memorable moment and connect together easily.

# SOFTWARE PROJECT MANAGEMENT PLAN

## PROBLEM DEFINITION

### NAME OF THIS CAPSTONE PROJECT

Project official name is “Family Album Sharing System”. Project code is “FAMSAM”.

### PROBLEM ABSTRACT

In order to satisfy users demand for a photo sharing system, this project is provide a web application that allow user to create or join families and share their stories/photos with others.

### PROBLEM OVERVIEW

#### The Current Systems

At moment, there are several services around the world which support photo album sharing as social network:

* Facebook (facebook.com): Facebook is a huge social network, which provide photo album and story sharing feature. However, when using Facebook, people tend to have many friends which they do not even know.
* Instagram (instagram.com): Instagram helps user to take and share picture to friends and family but all its feature only focus on mobile app.
* Flickr (flickr.com): Unlike Instagram, Flickr has rich feature user interface on both website and mobile app while it does not strongly support user to create story.

There are more photo album sharing system but after all, their services are not focus on family characteristic or groups of people that have strong relationship, who want to store, manage and share their specific moment through albums and stories.

#### The Proposed System

This product is a social network that support managing and sharing photo album among families. This also provide the most convenient and effective way to manage photo albums.

For more specific, the system supports the follow features:

* **Family features**:
  + User can create family.
  + Invite other users into family.
  + User who is owner of a family can make his/her family be neighbor with other families.
* **Post (Story/Album/Photo) features:** 
  + User can create and manage album.
  + User can upload photos and add photos into album.
  + User can create and manage story.
  + User can add albums into story and edit album in the story.
  + User can create slideshow from their album or story.
* **Social Network features**
  + User can share a post (Story/Album/Photo) to other family.
  + User can follow a family.
  + User can like/comment on a post.
* **Admin management features**
  + Admin can manage user.
  + Admin can manage reports from user about bad posts (story/album/photo).
  + Admin can manage reported posts.
  + Admin can manage recommend services.
* **Additional features**
  + Uploaded photos can be detect if they are blurry for user to decide to remove before add them to the album.
  + Recommend similar story or album for user based on they likes.

#### Boundaries of the System

About client support, the system only support web client, not mobile client. The web client must best compatible with follow browsers: Chrome 39+, Firefox 33+, Internet Explorer 11+.

About recommendation system, the system only recommends story and album based on user’s likes, not based on similar content.

About image quality processing, the system only supports detect blurry image, not detect other features of bad quality images.

#### Development Environment

##### Hardware requirement:

* Ram: 2 GB.
* CPU: Intel Core i3 2.5 GHz.
* HDD: 500 GB.

##### Software requirement:

* + Windows 7 or Windows 8 with Message Queue installed.
  + .NET Framework 4.5.

## PROJECT ORGANIZATION

### SOFTWARE PROCESS MODEL

Our project use **Scrum** as project model.

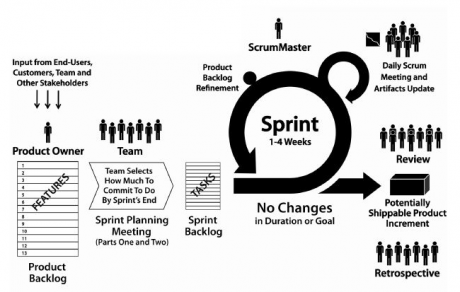


Figure 1 - Scrum Model [URL]

Due to Capstone Project characteristic, we decide each sprint takes one week. At the beginning of each sprint, we have sprint planning meeting, all team members will discuss to define sprint backlog which is suitable to be completed within a week. Daily report is operated through online conference meeting.

### ROLES AND RESPONSIBILITIES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Full name | Team Role | Scrum Team Role | Responsibilities |
| 1 | Nguyễn Huy Hùng | Supervisor | Product Owner | * Outline work in scrum backlog * Prioritize work * Answer question and deliver direction |
| 2. | Huỳnh Trần Minh Nhựt | Team Leader | Scrum Master | * Facilitate productivity – maximize team performance * Complete all individual work |
| 3. | Tô Đình Tân | Team Member | Scrum Team Member | * Commit individual product on time * Support each other to complete team work |

### TOOLS AND TECHNIQUES

##### Tools:

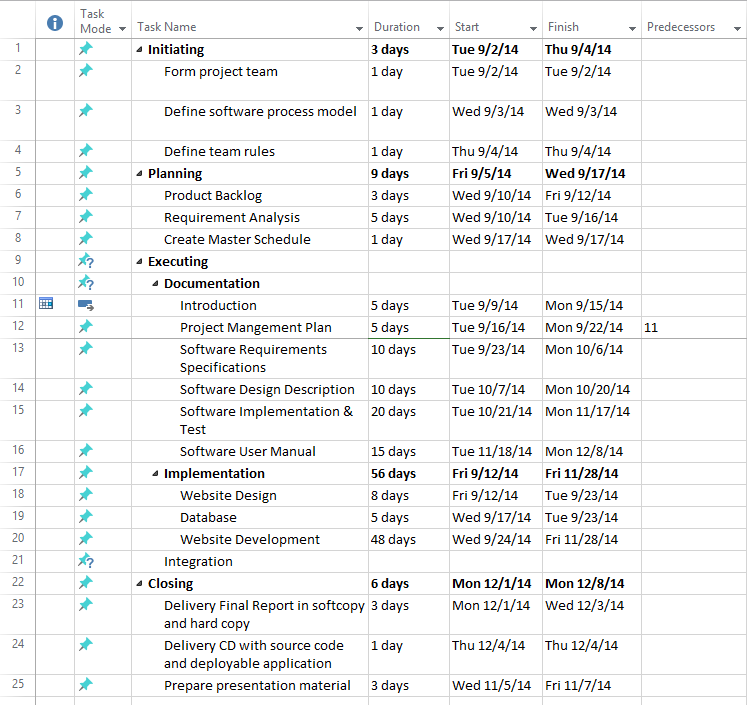
|  |  |
| --- | --- |
| IDE | Visual Studio 2013 Professional |
| SVN | Visual Studio Foundation Server |
| Software design | Microsoft Visio 2013, StarUML, Software Ideas Modeler |

##### Techniques

|  |  |
| --- | --- |
| Server side | * .NET Framework 4.5. * ASP .NET Web API 2 to provide RESTful web service API on server. * Microsoft SQL Server 2012 as database. * Entity Framework 6 as Object-Relational Mapping to access database. |
| Client side | * HTML5 * Bootstrap 3.0 for design web UI. * AngularJS 1.2 to provide MV\* javascript framework. Major responsibility of AngularJS is to support dynamic view, UI routing, server data access at client side. |

## PROJECT MANAGEMENT PLAN

### TASKS



### DELIVERABLES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Deliverable | Delivery date | Delivery location | Note |
| 1 | Introduction Document | 15/9/2014 | FU – CMS | Report No. 1 |
| 2 | Software Project Management Plan | 22/9/2014 | FU – CMS | Report No. 2 |
| 3 | Software Requirements Specification | 06/10/2014 | FU – CMS | Report No. 3 |
| 4 | Software Design Description | 10/20/2014 | FU – CMS | Report No. 4 |
| 5 | Software Implementation & Test | 17/11/2014 | FU – CMS | Report No. 5 |
| 6 | Software User Manual | 17/11/2014 | FU – CMS | Report No. 6 |
| 7 | Completed version of Report No. 1 to Report No. 6 | 08/11/2014 | Classroom  FU – CMS | Final version |
| 9 | CD contains all Source code and installable system | 08/11/2014 | Classroom | Final version |

### ALL MEETING MINUTES

Refer to: *CD:\Meeting Minutes*

# SOFTWARE REQUIREMENTS SPECIFICATIONS

## USER REQUIREMENT SPECIFICATION

The system is a social network that provides sharing albums services. On this system, users can connect with their family, and share their photos, albums or stories together. The system should provide features that detects the blur photos and recommend stories or albums to user base on their “like”.

More precisely, the system should offer the features listed below.

### WEBSITE USER REQUIREMENTS

A website user has the following abilities:

* Create new family.
* Invite other users to join family.
* Accept invitation join family from other users.
* Leave family.
* Search family.
* Follow/Unfollow family.
* Create/Edit/Remove Story in Family.
* Create/Edit/Remove Album.
* Create Slideshow for Album/Story.
* Tag neighbors in a story.
* Update user information.

### FAMILY OWNER REQUIREMENTS

A family owner has the following abilities:

* Send neighbor request to other family.
* Accept neighbor request from other family.
* Change background image of family.

### ADMINISTRATOR REQUIREMENTS

Administrators of the system has the following abilities:

* Configure recommendation system attributes
* Manage reports from users.
* Manage users’ account.

## SYSTEM REQUIREMENT SPECIFICATION (SPECIFIC REQUIREMENTS)

### SYSTEM FEATURES

This section provides a short overview of all features. Each feature is a group of similar use cases which are given by the system to satisfy user requirements.

Users who can interact directly with the system can be divided into 4 kinds of actors as follows:

|  |  |  |
| --- | --- | --- |
| No. | Actor | Description |
| 1 | Guest | Visitor who is not logged in to the system. |
| 2 | User | Logged in guest. |
| 3 | Family Owner | User who is owner of a family. |
| 4 | Administrator | User who has the right of administration: manage all activities happen in the system. |

Table 1: Actor list



Figure 2: Overall Use-case diagram

### USE CASE DIAGRAM AND USE CASE SPECIFICATION

* 1. **<Guest> Login**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE LOGIN SPECIFICATION | | | | | |
| Use-case No. | UC001 | Use-case Version | | 1.0 | |
| Use-case Name | Login | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Guest | | | | |
| **Summary** | This use case is about Login function. | | | | |
| **Goal** | Authenticate User. | | | | |
| **Triggers** | * Guest accessed website’s address in browser. | | | | |
| **Preconditions** | N/A. | | | | |
| **Post Conditions** | * Success: User has logged in. Redirects to User homepage. * Fail: Display error message “Login Fail! Wrong username or password”. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Access website’s address. | Display Login page:   * Username: textbox * Password: password field * Sign in: button * Sign up for free: button | | 2 | Fill username and password in textboxes |  | | 3 | User click “Sign in” button  [Alternative Scenario 1]  [Exception 1] | Display User homepage. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Sign up for free” button | Display Register page. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Username and password are not valid | Display error message: “Login Fail! Wrong username or password”. | | | | | | |
| **Relationships**  N/A | | | | |  |
| **Business Rules**   * If username or password is empty, “Sign in” button must be disabled. * If user login successful, the system creates a session for user. * User’s session will expire after 2 hours since last action of user. | | | | |  |

* 1. **<Website User> Logout**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE LOGOUT SPECIFICATION | | | | | |
| Use-case No. | UC002 | Use-case Version | | 1.0 | |
| Use-case Name | Logout | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about Logout function. | | | | |
| **Goal** | Website User can log out from system successful. | | | | |
| **Triggers** | * Website User click “Log out” link in top-bar menu. | | | | |
| **Preconditions** | * User login successful. | | | | |
| **Post Conditions** | * Success: Display Login page. * Fail: Display error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Log out” link in top-bar menu. | Display Login page. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | | | | | |
| **Relationships**  UC001 | | | | |  |
| **Business Rules**   * If user log out successful, the system will clear user’s session. | | | | |  |

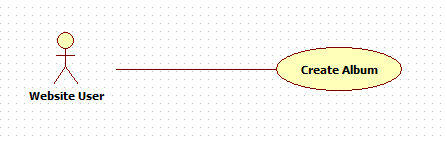
* 1. **<Website User> View Album**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE VIEW ALBUM SPECIFICATION | | | | | |
| Use-case No. | UC003 | Use-case Version | | 1.0 | |
| Use-case Name | View Album | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about view an album function. | | | | |
| **Goal** | User can view album detail. | | | | |
| **Triggers** | * Website user clicks on the name of album in navigation bar. * Or website user clicks on the name of album while viewing story detail. * Or website user clicks on the name of album while viewing family homepage. * Or website user clicks on the name of album while viewing all albums of a user. | | | | |
| **Preconditions** | User login successful. | | | | |
| **Post Conditions** | * Success: Display Album Detail page * Fail: Display “Content not found” page. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on the name of album  [Exception 1]  [Exception 2] | Show Album Detail Screen with:  + Album Title: text  + Album Owner: link  + Date Time upload  + Album Description: text  + “Like”: link  + “Edit”: link (if current user is owner of album)  + “Remove this album”: link (if current user is owner of album)  + “I don’t like this album”: link (if current user is not owner of album)  + List of photos  + “Share”: link  + List of comments  + Comment: textbox | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login Page. | | 2 | Album does not exist. | Display “Content not found” Page. | | | | | | |
| **Relationships**  UC001 | | | | |  |
| **Business Rules**   * If album exists in database, it will be shown in Album Detail page with:   + All photos thumbnail of album.  + Number of likes on album.  + All comments about album.  + All hashtags of album. | | | | |  |

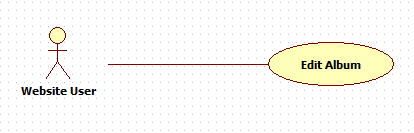
* 1. **<Website User> Create Album**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE CREATE ALBUM SPECIFICATION | | | | | |
| Use-case No. | UC004 | Use-case Version | | 1.0 | |
| Use-case Name | Create Album | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about Create new album function. | | | | |
| **Goal** | Users can create new album. | | | | |
| **Triggers** | * Website user clicks on “Create Album” link on top-bar menu. * Or website user clicks on “Create New Album” in pop-up “Add Albums” while creating story. | | | | |
| **Preconditions** | User login successful. | | | | |
| **Post Conditions** | * Success: New album is created, display album detail page. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Create Album” or “Create New Album” link | “Create Album” screen show:   * Story: read-only textbox * Choose Story: button * Title: textbox * Share with Others Family: checkbox * Families to share: textbox * Description: text-area * Hashtags: textbox * Add: button * “Add Photos”: button * Remove Selected Photos: link * “Create”: button | | 2 | Click “Choose Story” button | “Choose a story” pop-up is shown with:   * List of stories of current user. * “Cancel”: button | | 3 | Click on the name of story  [Alternative Scenario 1] | Name of chosen story is filled in Story textbox. | | 4 | Check “Share with Others Family” checkbox | Show “Families to share” textbox | | 5 | Type keyword in “Families to share” checkbox | Show list of current user’s families and neighbors that contains keyword. | | 6 | Click on family in list | Chosen family added into “Families to share” textbox | | 7 | Click “X” link on the top right of textbox “Families to share” | Textbox is removed. | | 8 | Type keyword in “Hashtags” textbox and click “Add” button | New textbox with keyword value is added below. | | 9 | Click “X” link on the top right of textbox “Hashtags” | Textbox is removed. | | 10 | Click “Add Photos” button | Refer to UC012 | | 11 | Check the checkbox at the right bottom of photo and click “Remove Selected Photos” in “Create Album” Screen | Chosen photos are removed. | | 12 | Click on photo | Show “Description” textbox. | | 13 | Type in “Description” textbox | Photo description is saved. | | 14 | Type in “Title” textbox and “Description” text-area then click “Create” button  [Exception 1]  [Exception 2] | Display Album Detail Page. | | | | | | |
| **Alternative Scenario 1**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button | Close pop-up. No story is added into Story textbox. | | | | | | |
|  | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Title or Description is empty. | Display error message “Title and Description must not be empty”. | | | | | | |
| **Relationships**  UC001, UC012 | | | | |  |
| **Business Rules**   * Title and Description must not be empty. Title must not exceed 20 characters. * If story field is not null, album will belong to that story. * If list of families user want to share album is not empty, album will be shared to families in list after it is created. Notifications will be sent to all members of each family. * Each hashtag is added (if new) and belong to album. * Photos are uploaded and belong to album. | | | | |  |

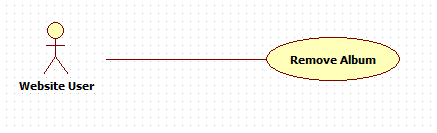
* 1. **<Website User> Edit Album**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE EDIT ALBUM SPECIFICATION | | | | | |
| Use-case No. | UC005 | Use-case Version | | 1.0 | |
| Use-case Name | Edit Album | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about edit an album function. | | | | |
| **Goal** | Users can edit their albums. | | | | |
| **Triggers** | * Website user is in Album Detail Page. * Website user clicks arrow symbol link at the right of Album Title. * Website user clicks “Edit” link. | | | | |
| **Preconditions** | User login successful and user is owner of album. | | | | |
| **Post Conditions** | * Success: The album is updated, Display album detail page * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Edit” link | “Edit Album” screen show:   * Title: textbox * Description: text-area * “Add Photos”: button * Remove Selected Photos: link * List of photos * “Save”: button | | 2 | Click “Add Photos” button | Refer to UC012 | | 3 | Check the checkbox at the right bottom of photo and click “Remove Selected Photos” in “Create Album” Screen | Checked photos are removed. | | 4 | Type in “Title” textbox and “Description” text-area then click “Save” button  [Exception 1]  [Exception 2] | Album is edited and show Album Detail Screen | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Title or Description is empty. | Display error message “Title and Description must not be empty”. | | | | | | |
| **Relationships**  UC001, UC003 | | | | |  |
| **Business Rules**   * User must be owner of album. * Title and Description must not be empty. Title must not exceed 20 characters. * Photos are uploaded and belong to album. * If photos are removed from Album, they are still in their owner’s profile. | | | | |  |

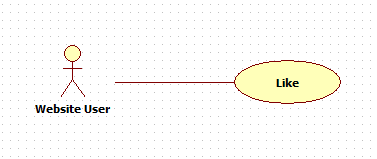
* 1. **<Website User> Remove Album**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE REMOVE ALBUM SPECIFICATION | | | | | |
| Use-case No. | UC006 | Use-case Version | | 1.0 | |
| Use-case Name | Remove Album | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about Remove an album function. | | | | |
| **Goal** | Users can remove their albums. | | | | |
| **Triggers** | * Website user is in album detail page. * Website user clicks arrow symbol link at the right of Album Title. * Website user clicks “Remove this album” link. | | | | |
| **Preconditions** | User login successful and user is owner of album. | | | | |
| **Post Conditions** | * Success: The album is removed, display user homepage * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Remove this album” link | Show “Remove Album” pop-up:   * “Remove album will not effect its photos. Are you sure to remove this album? Restore removed albums is not supported yet.”: text * “Yes”: button * “No”: button | | 2 | Click “Yes” button  [Alternative Scenario 1]  [Exception 1]  [Exception 2] | Display success message and “Ok” button. | | 3 | Click “Ok” button | Display User Homepage. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “No” button | Close pop-up. Album will not be removed. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Album does not exist | Display error message. | | | | | | |
| **Relationships**  UC001, UC003 | | | | |  |
| **Business Rules**   * User must be owner of album. * If album is removed successful:   + Remove relationship between album and stories that contain album.  + Remove relationship between photos and album.  + Remove all likes and comments of users on album.  + Remove all shared posts from album.  + Remove relationship between keywords (used in recommendation system) and album.  + Remove all reports of album. | | | | |  |

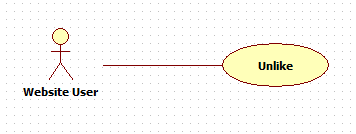
* 1. **<Website User> Like**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE LIKE SPECIFICATION | | | | | |
| Use-case No. | UC007 | Use-case Version | | 1.0 | |
| Use-case Name | Like | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about Like story, album, photo, or shared post function. | | | | |
| **Goal** | Users can like Story/album/photo/shared-post. | | | | |
| **Triggers** | * Website user is in story/album/photo detail page or user is in family homepage. | | | | |
| **Preconditions** | User login successful. | | | | |
| **Post Conditions** | * Success: Story/Album/Photo/Shared-post is liked, and number of likes increase. * Fail: Number of likes unchanged. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Like” link below story/album/photo/shared-post description.  [Exception 1]  [Exception 2] | Number of likes increase. Show “Unlike” link, hide “Like” link. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Post (Story/Album/Photo/Shared-post) does not exist | Number of likes unchanged. | | | | | | |
| **Relationships**  UC001, UC003, UC011, UC015. | | | | |  |
| **Business Rules**   * If current user is not owner of liked post (story/album/photo/shared-post), a notification is created and sent to owner of the post. | | | | |  |

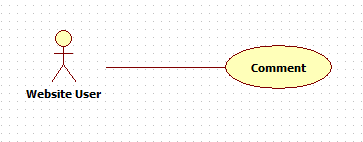
* 1. **<Website User> Unlike**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE UNLIKE SPECIFICATION | | | | | |
| Use-case No. | UC008 | Use-case Version | | 1.0 | |
| Use-case Name | Unlike | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about Unlike story, album, photo, or shared post function. | | | | |
| **Goal** | Users can unlike Story/album/photo/shared-post. | | | | |
| **Triggers** | * Website user is in story/album/photo detail page or user is in family homepage. | | | | |
| **Preconditions** | User login successful and liked Story/album/photo/shared-post already. | | | | |
| **Post Conditions** | * Success: Story/Album/Photo/Shared-post is unliked, and number of likes decrease. * Fail: Number of likes unchanged. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Unlike” link below story/album/photo/shared-post description.  [Exception 1]  [Exception 2] | Number of likes decrease. Show “Like” link, hide “Unlike” link. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Post (Story/Album/Photo/Shared-post) does not exist | Number of likes unchanged. | | | | | | |
| **Relationships**  UC001, UC003, UC007, UC011, UC015. | | | | |  |
| **Business Rules**  N/A | | | | |  |

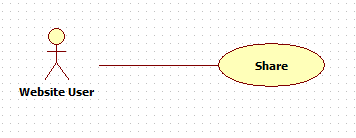
* 1. **<Website User> Comment**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE COMMENT SPECIFICATION | | | | | |
| Use-case No. | UC009 | Use-case Version | | 1.0 | |
| Use-case Name | Comment | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about comment on story, album, or photo function. | | | | |
| **Goal** | Users can comment on Story/album/photo. | | | | |
| **Triggers** | * Website user is in story/album/photo detail page. | | | | |
| **Preconditions** | User login successful. | | | | |
| **Post Conditions** | * Success: A comment is added to story/album/photo, and shown above Comment textbox * Fail: No comment is added. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Type in Comment textbox then press “Enter”  [Exception 1]  [Exception 2] | New comment is shown above Comment textbox. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Story/Album/Photo does not exist | Display “Content not found” page. | | | | | | |
| **Relationships**  UC001, UC003, UC011, UC015. | | | | |  |
| **Business Rules**   * If current user is not owner of commented post (story/album/photo), notifications are created and sent to owner of the post and users (except current user) who commented before. * Owner of the post (story/album/photo) can remove violated comments. | | | | |  |

* 1. **<Website User> Share**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE SHARE SPECIFICATION | | | | | |
| Use-case No. | UC010 | Use-case Version | | 1.0 | |
| Use-case Name | Share | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about share story, or album function. | | | | |
| **Goal** | Users can share story/album to their families or neighbors. | | | | |
| **Triggers** | * Website user is in story/album detail page. | | | | |
| **Preconditions** | User login successful. | | | | |
| **Post Conditions** | * Success: Story/Album is shared to family. * Fail: No Post is shared. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | * Click “Share” link below story description in Story Detail Screen * Click “Share” link below list photos of album in Album Detail Screen | Show “Share” pop-up:   * “Choose family” drop-down list. * “Description”: text-area * “Share”: button * “Cancel”: button | | 2 | Choose a family in drop-down list, type in text-area then click “Share”.  [Alternative Scenario 1] | Show success message and close pop-up. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button | Close pop-up. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Go to login page. | | | | | | |
| **Relationships**  UC001, UC003, UC015 | | | | |  |
| **Business Rules**   * A post (story/album) can be shared many times to user’s family. * Members (except current user) of shared family got notifications from system. | | | | |  |

* 1. **<Website User> View photo.**
     1. Use case diagram



* + 1. Use case description

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE VIEW PHOTO SPECIFICATION | | | | | |
| Use-case No. | UC011 | Use-case Version | | 1.0 | |
| Use-case Name | View Photo | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about view photo function. | | | | |
| **Goal** | User can view a photo. | | | | |
| **Triggers** | * Website user clicks on photo thumbnail. | | | | |
| **Preconditions** | User login success. | | | | |
| **Post Conditions** | * Success: Display photo detail pop-up. * Fail: No photo is shown. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on photo thumbnail.  [Exception 1]  [Exception 2] | Display Photo Detail pop-up with:  + Image  + Description: text  + Add Description: link (if photo have not description)  + Edit Description: link (if photo have description)  + I don’t like this photo: link (if current user is not owner of this photo)  + Like: link  + Comment: input text. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Photo does not exist. | Show error message. | | | | | | |
| **Relationships**  UC001 | | | | |  |
| **Business Rules**   * Display photo detail with:   + Owner of photo.  + All likes and comments on photo.  + Get full-size image, not thumbnail.  + Photo when displayed must be fit the “View photo” pop-up with original ratio. | | | | |  |
|  | | | | | |

* 1. **<Website User> Upload photo.**
     1. Use case diagram



* + 1. Use case description

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE UPLOAD PHOTO SPECIFICATION | | | | | |
| Use-case No. | UC012 | Use-case Version | | 1.0 | |
| Use-case Name | Upload Photo | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about upload photo function. | | | | |
| **Goal** | User can upload photo. | | | | |
| **Triggers** | * Website user clicks “Create Album” in top-bar menu. * Website user clicks “Add Photos” button. | | | | |
| **Preconditions** | User login success. | | | | |
| **Post Conditions** | * Success: Photos are uploaded. * Fail: Display error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Add Photos” button | Show “Upload Photos” pop-up:   * “Choose Photo”: button * “Auto Detect Bad Quality Image”: checkbox * “Remove Selected Photos”: link * “Finish”: button * “Cancel”: button | | 2 | Click “Auto Detect Bad Quality Image” checkbox. |  | | 3 | Click “Choose Photos” | Show Open File dialog | | 4 | Choose image files from local computer then click Open | Photos are listed below. All bad quality photos detected by system are checked. | | 5 | Check the checkbox at the right bottom of photo and click “Remove Selected Photos” in pop-up | Checked Photos are removed. | | 6 | Click “Finish” button  [Alternative Scenario 1]  [Exception 1] | Display uploading process percentage until complete.  Show OK button. | | 7 | Click “Ok” button | All photos in pop-up are uploaded and listed in Create Album screen. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button | Close pop-up. No photo uploaded. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | | | | | |
| **Relationships**  UC001, UC004, UC005 | | | | |  |
| **Business Rules**   * Only support upload image file types: jpeg, jpg, png. * Upload file must not be exceed 10MB. * Uploaded photos are saved in two files: full-size image and thumbnail image. * The thumbnail image must have   + The ratio and resolution is the same as the original.   + The maximum of width and height is less than or equal 200px. * Photos are uploaded sequentially. * If users checked “auto detect bad quality image” checkbox, system must detect bad photos and check them. Users can remove them before they are uploaded. | | | | |  |
|  | | | | | |

* 1. **<Website User> View story**
     1. Use case diagram



* + 1. Use case description

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE VIEW STORY SPECIFICATION | | | | | |
| Use-case No. | UC015 | Use-case Version | | 0.1 | |
| Use-case Name | View Story | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about view story function. | | | | |
| **Goal** | Users can view story. | | | | |
| **Triggers** | * Website user clicks on the name of story in navigation bar. * Or website user clicks on the name of story while viewing all stories of an user. * Or website user clicks on the name of story while viewing family homepage. | | | | |
| **Preconditions** | User login success. | | | | |
| **Post Conditions** | * Success: Display Story Detail page. * Fail: Display “Content Not Found” page. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on the name of story  [Exception 1] [Exception 2] | Show Story Detail page with:  + Story Title: text  + Family Name: link  + “Edit Title”: link (if current user is owner of story)  + “Edit Description”: link (if current user is owner of story)  + “Add Album”: link (if current user is owner of story)  + “Edit Album”: link (if current user is owner of story)  + “Remove Story”: link (if current user is owner of story)  + “I don’t like this story”: link (if current user is not owner of story)  + Name of story owner: link  + List of neighbors are tagged in story: link  + Story Description: text  + “Share”: link  + “Like”: link  + Number of likes: number  + List of albums (Title, Date Created, Description, “Like” link, photos of album)  + List of hashtags: text  + List of comments: text  + Comment: textbox | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Story does not exist | Display “Content not found” page. | | | | | | |
| **Relationships**  UC001 | | | | |  |
| **Business Rules**   * If story exists in database, it will be shown in Story Detail page with:   + All albums of story.  + Number of likes on story.  + All comments about story.  + All families are tagged in story  + All hashtags of story | | | | |  |

* 1. **<Website User> Create Story**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE CREATE STORY SPECIFICATION | | | | | |
| Use-case No. | UC016 | Use-case Version | | 1.0 | |
| Use-case Name | Create Story | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about create story function. | | | | |
| **Goal** | Users can create story. | | | | |
| **Triggers** | * Website user clicks on “Create Story” link on top-bar menu. * Or website user clicks on “Create New Story” while in all stories page. | | | | |
| **Preconditions** | User login successful. | | | | |
| **Post Conditions** | * Success: New story is created, display story detail page * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Create Story” or “Create New Story” link | “Create Story” screen show:   * Family: read-only textbox * “Change Family”: button * Name: textbox * Share with: drop-down list * Tag your neighbors: textbox (show when choose option “Some Neighbors” in Share with drop-down list) * Description: text-area * Hashtags: textbox * “Add”: button * “Add New Album(s)”: link * “+Add Albums”: button * “Create Story”: button | | 2 | Click “Change Family” button. | Show “Choose a family” pop-up:   * List of families * “Cancel”: button | | 3 | Click on the name of a family  [Alternative Scenario 1] | Name of chosen family is filled in Family textbox | | 4 | Choose “Some Neighbors” option in drop-down list “Share with” | Show “Tag your neighbors” label and textbox. | | 5 | Type keyword in textbox | Show list of neighbors of current user that their names contain keyword. | | 6 | Choose one neighbor from list | Name of chosen neighbor is filled into textbox | | 7 | Type keyword in Hashtags textbox then click “Add” button | New read-only textbox with value keyword is added below. | | 8 | Click “Add New Album(s)” link or “+Add Albums” button | Show “Add Albums” pop-up:   * “Create New Album”: button * List of Albums * “Ok”: button * “Cancel”: button | | 9 | Click on checkbox at the bottom right of album and click “Ok”  [Alternative Scenario 1] | Chosen albums are added into creating story. Show “Remove Selected Albums” | | 10 | Click on checkbox at the bottom right of album and click “Remove Selected Albums” link | Chosen albums are removed from creating story. | | 11 | Fill Name textbox and Description text-area then click Create Story  [Exception 1] [Exception 2]  [Exception 3] | New Story is added and show Story Detail Page. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Create New Album” button | Display Create Album Page.  Refer to Use Case “Create Album” UC004.  Created Album is added to story. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display login page. | | 2 | Title or Description is empty | Show error message. | | 3 | Family textbox is empty | Show pop-up Choose Family. | | | | | | |
| **Relationships**  N/A | | | | |  |
| **Business Rules**   * Family textbox must not be empty. * Title and Description must not be empty. Title must not exceed 20 characters. * Story belong to family in textbox. * If “Share with” value is “Public”, every user can see this story. * If “Share with” value is “My Family Only”, just members in family can see this story. * If list of families user want to tag is not empty, they will be tagged in story and received notification from system. * Hashtag is added (if new) and belong to story. | | | | |  |

* 1. **<Website User> Edit Story Description**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE EDIT STORY SPECIFICATION | | | | | |
| Use-case No. | UC018 | Use-case Version | | 1.0 | |
| Use-case Name | Edit Story | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about edit story function. | | | | |
| **Goal** | Users can change their story title and description. | | | | |
| **Triggers** | * Website user clicks on “Edit” link on the right of story title while in story detail page. Then click “Edit” from the list options. | | | | |
| **Preconditions** | User login successful and is in story detail page. | | | | |
| **Post Conditions** | * Success: Story title and description is updated. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Edit Story”. | Display “Edit Story” pop-up:   * New Title: label. * Title: textbox with current title value. * New Description: label. * Description: Text-area with current description value. * “Ok”: button * “Cancel”: button | | 2 | Edit textbox and description then click “Ok”  [Alternative Scenario 1]  [Exception 1] [Exception 2] | Close pop-up. Story title and description is updated. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button | Close pop-up. Story title and description unchanged. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Story does not exist | Display “Content not found” page. | | | | | | |
| **Relationships**  UC001, UC015 | | | | |  |
| **Business Rules**   * Title must not be empty and not exceed 20 characters. | | | | |  |

* 1. **<Website User> Add Album to story**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE ADD ALBUM TO STORY SPECIFICATION | | | | | |
| Use-case No. | UC019 | Use-case Version | | 1.0 | |
| Use-case Name | Add album to story | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 02/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about add albums to story function. | | | | |
| **Goal** | Users can add their albums into their story. | | | | |
| **Triggers** | * Website user clicks on “Edit” link on the right of story title while in story detail page. Then click “Add Album” from the list options. | | | | |
| **Preconditions** | User login successful and is in story detail. | | | | |
| **Post Conditions** | * Success: Albums are added into story. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Add Album” link | Show “Add Albums” pop-up:   * List of albums of current user. * “Ok”: button * “Cancel”: button | | 2 | Click checkbox at the bottom right of album and click “Ok” button  [Alternative Scenario 1]  [Exception 1] [Exception 2]  [Exception 3] | Close pop-up. Chosen albums are added into story and show in story detail page. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button | Close pop-up. No album is added. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Story does not exist | Display “Content not found” page. | | 3 | Album has been in story already. | Display error message. | | | | | | |
| **Relationships**  UC001, UC015 | | | | |  |
| **Business Rules**   * Story can have many albums. * An album can belong to many stories. | | | | |  |

* 1. **<Website User> Remove album from story**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE REMOVE ALBUM FROM STORY SPECIFICATION | | | | | |
| Use-case No. | UC020 | Use-case Version | | 1.0 | |
| Use-case Name | Remove album from story | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about remove albums from story function. | | | | |
| **Goal** | Users can remove albums from their story. | | | | |
| **Triggers** | * Website user clicks on “Edit” link on the right of story title while in story detail page. Then click “Edit Album” from the list options. | | | | |
| **Preconditions** | User login successful and is in story detail page. | | | | |
| **Post Conditions** | * Success: Albums are removed from story. * Fail: Display error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Edit Album” link | Show:   * “Save Edit”: button * “Cancel”: button * “Remove this album”: link * “Down arrow”: button * “Up arrow”: button | | 2 | Click “Remove this album” beside each album. | Album is removed temporary from story. | | 3 | Click “Save Edit” button  [Alternative Scenario 1]  [Exception 1] [Exception 2] | Album is removed completely from story. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button | Recovery removed albums. No album removed. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Story does not exist | Display “Content not found” page. | | | | | | |
| **Relationships**  UC001, UC015, UC018 | | | | |  |
| **Business Rules**  N/A | | | | |  |

* 1. **<Website User> Edit album index**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE EDIT ALBUM INDEX SPECIFICATION | | | | | |
| Use-case No. | UC021 | Use-case Version | | 1.0 | |
| Use-case Name | Edit album index | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about edit the order of albums function. | | | | |
| **Goal** | Users can change the order of albums in their story. | | | | |
| **Triggers** | * Website user clicks on “Edit” link on the right of story title while in story detail page. Then click “Edit Album” from the list options. | | | | |
| **Preconditions** | User login successful and is in story detail page. | | | | |
| **Post Conditions** | * Success: Albums are reordered. * Fail: Albums index unchanged. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Edit Album” link | Show:   * “Save Edit”: button * “Cancel”: button * “Remove this album”: link * “Down arrow”: button * “Up arrow”: button | | 2 | Click “Down arrow” beside each album. | Album is moved down temporary. Album index decreased 1. | | 3 | Click “Up arrow” beside each album. | Album is moved up temporary. Album index increased 1. | | 3 | Click “Save Edit” button  [Alternative Scenario 1]  [Exception 1] [Exception 2] | Order of albums is changed completely. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button | Recovery the order of albums. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Story does not exist | Display “Content not found” page. | | | | | | |
| **Relationships**  UC001, UC015, UC018. | | | | |  |
| **Business Rules**   * First album of story must not have “Move Up” button. * Last album of story must not have “Move down” button. * By default, the albums are sorted by time added. Last album added has the first position in story. | | | | |  |

* 1. **<Website User> Remove Story**
     1. Use Case Diagram

****

* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE REMOVE STORY SPECIFICATION | | | | | |
| Use-case No. | UC022 | Use-case Version | | 1.0 | |
| Use-case Name | Remove Story | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about remove a story function. | | | | |
| **Goal** | Users can remove their story. | | | | |
| **Triggers** | * Website user clicks “Edit” link at the right of Story Title in Story detail page. * Website user clicks “Remove Story” link. | | | | |
| **Preconditions** | User login successful and is owner of story. | | | | |
| **Post Conditions** | * Success: The story is removed, display homepage * Fail: Display error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Remove Story” link | Show “Remove Story” pop-up:   * “Remove story will not effect its albums. Are you sure to remove this story? Restore removed story is not supported yet.”: text. * “Yes”: button * “No”: button | | 2 | Click “Yes” button  [Alternative Scenario 1]  [Exception 1] [Exception 2] | Show success message and “Ok” button. | | 3 | Click “Ok” button | Display Homepage. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “No” button | Close pop-up. Story will not be removed. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Story does not exist | Display “Content not found” page. | | | | | | |
| **Relationships**  UC001, UC015 | | | | |  |
| **Business Rules**   * User must be owner of story. * If story is removed successful:   + Remove relationship between albums and story.  + Remove all likes and comments of users on story.  + Remove all shared posts from story.  + Remove relationship between keywords (used in recommendation system) and story.  + Remove all reports of story. | | | | |  |

* 1. **<Website User> Create Family**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE CREATE FAMILY SPECIFICATION | | | | | |
| Use-case No. | UC023 | Use-case Version | | 1.0 | |
| Use-case Name | Create Family | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about create family function. | | | | |
| **Goal** | Users can create new family. | | | | |
| **Triggers** | * Website user clicks on “Families” tab in navigation bar then clicks on “Create New Family” button. | | | | |
| **Preconditions** | User login successful. | | | | |
| **Post Conditions** | * Success: New family is created. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Create New Family” button | Show Create New Family screen:   * Family Name: label * “Name”: Textbox * Description: label * “Description”: Text-area * “Create”: button | | 2 | Fill “Name” textbox and “Description” text-area then click “Create”  [Exception 1] [Exception 2] | Family is created and show family homepage. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Family name is empty | Display error message. | | | | | | |
| **Relationships**  UC001 | | | | |  |
| **Business Rules**   * Family name must not be empty and must not exceed 20 characters. * Allow duplicate family names. * Unlimited number of members in family. * User who created family has role “owner” of the family. * Must not add members while creating family. | | | | |  |

* 1. **<Website User> Leave Family**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE LEAVE FAMILY SPECIFICATION | | | | | |
| Use-case No. | UC024 | Use-case Version | | 1.0 | |
| Use-case Name | Leave Family | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about leave a family function. | | | | |
| **Goal** | User can leave their family. | | | | |
| **Triggers** | * Website user clicks on “Arrow” button in family homepage then clicks on “Leave this family” link. | | | | |
| **Preconditions** | User login successful and is a member of family. | | | | |
| **Post Conditions** | * Success: User left family. Display all families page. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Leave this family” link  [Exception 1] | Family is removed from list families of user. Display All Families Page. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Go to login page. | | | | | | |
| **Relationships**  UC001 | | | | |  |
| **Business Rules**   * If user is member of family, the relationship between family and user is removed. * If user is owner of family:   + If number of members is larger than two, the relationship between family and user is removed and the second member who joined in family after the owner is set to be owner of family.  + If family has one member, the relationship between family and user is removed and the family has no member. | | | | |  |

* 1. **<Website User> Follow Family**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE FOLLOW FAMILY SPECIFICATION | | | | | |
| Use-case No. | UC025 | Use-case Version | | 1.0 | |
| Use-case Name | Follow Family | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about follow family function. | | | | |
| **Goal** | Users can follow a family. | | | | |
| **Triggers** | * Website user clicks on “Follow” button in family homepage. | | | | |
| **Preconditions** | User login successful and user is not a member of family. | | | | |
| **Post Conditions** | * Success: User followed family. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Follow” button  [Exception 1] | Family is added to list follow families of current user. Button “Follow” changes to “Unfollow” | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | | | | | |
| **Relationships**  UC001 | | | | |  |
| **Business Rules**   * Display stories in followed family in user homepage. * User can follow a family when user is not a member of that family. | | | | |  |

* 1. **<Website User> Unfollow Family**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE UNFOLLOW FAMILY SPECIFICATION | | | | | |
| Use-case No. | UC026 | Use-case Version | | 1.0 | |
| Use-case Name | Unfollow Family | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about unfollow family function. | | | | |
| **Goal** | Users can unfollow a family. | | | | |
| **Triggers** | * Website user clicks on “Unfollow” button in family homepage. | | | | |
| **Preconditions** | User login successful and user followed family. | | | | |
| **Post Conditions** | * Success: User unfollowed family. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Unfollow” button  [Exception 1] | Family is removed from list follow families of current user. Button “Unfollow” changes to “Follow” | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | | | | | |
| **Relationships**  UC001, UC025 | | | | |  |
| **Business Rules**  N/A | | | | |  |

* 1. **<Website User> Invite family member**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE INVITE FAMILY MEMBER SPECIFICATION | | | | | |
| Use-case No. | UC027 | Use-case Version | | 1.0 | |
| Use-case Name | Invite family member | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about invite users to join family function. | | | | |
| **Goal** | Users can invite other users to their family. | | | | |
| **Triggers** | * Website user clicks on “Invite someone” button in family homepage. | | | | |
| **Preconditions** | User login successful and user is member of family. | | | | |
| **Post Conditions** | * Success: Users are invited to family. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Invite someone” button | Show “Invite someone to family” pop-up:   * Someone to invite: label. * “Search”: textbox. * “Search”: button. * “Send invite”: button * “Cancel”: button | | 2 | Type keyword in textbox and click “Search” | List of users which have name contains keyword are displayed. | | 3 | Click on name of a user | Show:   * Invite: label. * List of Name of user: text * “Add Someone Else”: button * Message: label * “Message”: textbox | | 4 | Click “Add Someone Else” | Show:   * Someone to invite: label. * “Search”: textbox. * “Search”: button. | | 5 | Repeat step 2 and 3 | Name of user adds to list. | | 6 | Fill in Message textbox and click “Send Invite”  [Alternative Scenario 1]  [Exception 1] | Show “Invite” pop-up:   * <Name of users> invited: text. * “Ok”: button | | 7 | Click “Ok” button | Close pop-up. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button | Close pop-up. No invite sent. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | | | | | |
| **Relationships**  UC001 | | | | |  |
| **Business Rules**   * If users have been in family already, no invitation sent. * If users are not in family, invitations are created and notifications are sent to users. | | | | |  |

* 1. **<Website User> Accept invitation to family.**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE ACCEPT INVITATION TO FAMILY SPECIFICATION | | | | | |
| Use-case No. | UC028 | Use-case Version | | 1.0 | |
| Use-case Name | Accept Invitation to family | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about accept invitation to join family function. | | | | |
| **Goal** | Users can accept invitations join family from other users. | | | | |
| **Triggers** | * Website user clicks on Number on top-bar menu then clicks “You’ve got an invitation from <name of user>” link. | | | | |
| **Preconditions** | User login successful. | | | | |
| **Post Conditions** | * Success: User joined family. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “You’ve got an invitation from <name of user>” link. | Show “Invitation” pop-up:   * Text: “<name of user> has invited you to his <name of family> family.” * Text: “From <name of user>: <message>” * “Join”: button * “Ignore”: button | | 2 | Click “Join” button  [Alternative Scenario 1]  [Exception 1] | Show “New Member” pop-up:   * Text: “You are now member of <name of family> family. Welcome!” * “Ok”: button. | | 3 | Click “Ok” button | Close pop-up. Go to family homepage. A notification is sent to inviter. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Ignore” button | Close pop-up. User did not join family. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | | | | | |
| **Relationships**  UC001, UC027 | | | | |  |
| **Business Rules**   * If user accepts invitation, user becomes a member of family. User can create story in family, get notification when other members create story in family, or share story/album to family. * Notify to other members of the family about the acceptance of current user. | | | | |  |

* 1. **<Website User> Create SlideShow**

1. Use Case Diagram

****

1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE CREATE SLIDESHOW SPECIFICATION | | | | | |
| Use-case No. | UC029 | Use-case Version | | 1.0 | |
| Use-case Name | Create SlideShow | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Website User | | | | |
| **Summary** | This use case is about create slideshow function. | | | | |
| **Goal** | Website Users can create slideshow from their albums/stories. | | | | |
| **Triggers** | * Website User clicks “Create SlideShow” button in story/album detail page. | | | | |
| **Preconditions** | User login successful. | | | | |
| **Post Conditions** | * Success: SlideShow is created successful. Display SlideShow Detail * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Create SlideShow” button | Display “Create Slideshow Options” page:   * Theme: dropdown-list * Photo Delay: textbox * Transition Delay: textbox * Show photo’s description: checkbox * Soundtrack: input field * Total Slideshow duration: label * Soundtrack duration: label * “Preview”: button * “Cancel”: button | | 2 | Choose Theme, Fill Photo Delay, Transition Delay textbox, Choose file mp3. | Display Total SlideShow Duration and Soundtrack duration | | 3 | Click “Preview” button  [Alternative Scenario 1]  [Exception 1] [Exception 2] | Display Slideshow Detail page. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button | No slideshow created. Display story/album detail page. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | 2 | Story/Album does not exist | Display “Content not found” page. | | | | | | |
| **Relationships**  UC001 | | | | |  |
| **Business Rules**   * A story/album has maximum 1 slideshow. * If user creates new slideshow for story/album, it will overwrite the previous slideshow. * Slideshow display all photos of album. * Duration of slideshow is calculated by sum of <number of photos> \* <photo delay> + <number of photos – 1> \* <transition delay> | | | | |  |

* 1. **<Family Owner> Request neighbor.**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE REQUEST NEIGHBOR SPECIFICATION | | | | | |
| Use-case No. | UC030 | Use-case Version | | 1.0 | |
| Use-case Name | Request neighbor | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Family Owner | | | | |
| **Summary** | This use case is about send request neighbor to a family function. | | | | |
| **Goal** | Family owner can request neighbor to a family. | | | | |
| **Triggers** | * Family owner clicks “Invite other family” button in family homepage. | | | | |
| **Preconditions** | User login successful and user is owner of family. | | | | |
| **Post Conditions** | * Success: Request neighbor is sent to a family. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Invite other family” button | Show “Neighbor Request” pop-up:   * Family name to invite: label * “Search”: textbox * “Search”: button * “Send Invite”: button * “Cancel”: button | | 2 | Type keyword in textbox then click “Search” | Show list of families have the name contains keyword. | | 3 | Click on the name of a family from the list | Show:   * Send Request To: label. * <name of family>: text. * Message: label. * “Message”: textbox | | 4 | Click “Send Invite” button  [Alternative Scenario 1]  [Exception 1] | Show:   * “Neighbor request to test was already sent.”: text * “Ok”: button   An Request Neighbor sent to family owner. | | 5 | Click “Ok” button | Close pop-up. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button | Close pop-up. No request sent. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | | | | | |
| **Relationships**  UC001 | | | | |  |
| **Business Rules**   * If family owner sent a request neighbor to a family, a notification is created and sent to owner of family. | | | | |  |

* 1. **<Family Owner> Accept neighbor request.**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE ACCEPT NEIGHBOR REQUEST SPECIFICATION | | | | | |
| Use-case No. | UC031 | Use-case Version | | 1.0 | |
| Use-case Name | Accept neighbor request | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Family Owner | | | | |
| **Summary** | This use case is about accept request neighbor from a family function. | | | | |
| **Goal** | Family Owner can accept request neighbor from other family. | | | | |
| **Triggers** | * Family Owner clicks on Number on top-bar menu then clicks “<name of family> family has just request to be neighbor of your <name of user’s family> family” link. | | | | |
| **Preconditions** | User login successful and user is owner of family. | | | | |
| **Post Conditions** | * Success: User’s family is neighbor of invited family. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “<name of family> family has just request to be neighbor of your <name of user’s family> family” link | Show “Neighbor Request” pop-up:   * “<name of family owner> from <name of requested family> family has just requested to be neighbor of your family. From <name of family owner>: <message>”: text * “Accept”: button * “Ignore”: button | | 2 | Click “Accept” button  [Alternative Scenario 1]  [Exception 1] | Show:   * “You are now neighbor with <name of requested family> family”: text * “Ok”: button   Number of neighbors of family increase 1. | | 3 | Click “Ok” button | Close pop-up. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Ignore” button | Close pop-up. Request is ignored. | | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Go to login page. | | | | | | |
| **Relationships**  UC001, UC029 | | | | |  |
| **Business Rules**   * If family owner accept a request neighbor from a family, two families become neighbor. Notifications are created and sent to requested family members. | | | | |  |

* 1. **<Administrator> View List of Users.**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE VIEW LIST OF USERS SPECIFICATION | | | | | |
| Use-case No. | UC032 | Use-case Version | | 1.0 | |
| Use-case Name | View list of users | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Administrator | | | | |
| **Summary** | This use case is about view list of users function. | | | | |
| **Goal** | Admin can view all of users. | | | | |
| **Triggers** | * Admin click on tab “User” on the left in Admin Dashboard. | | | | |
| **Preconditions** | User login successful as an admin. | | | | |
| **Post Conditions** | * Success: Show list of users. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on tab “User”  [Exception 1] | Show Users management page:   * Table:   + No., Username, Name, Date Join, Status, Action (Ban/Unban) | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | | | | | |
| **Relationships**  N/A | | | | |  |
| **Business Rules**  N/A | | | | |  |

* 1. **<Administrator> Search User.**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE SEARCH USER SPECIFICATION | | | | | |
| Use-case No. | UC033 | Use-case Version | | 1.0 | |
| Use-case Name | Search User | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Administrator | | | | |
| **Summary** | This use case is about search users function. | | | | |
| **Goal** | Admin can search users by name. | | | | |
| **Triggers** | * Admin click on tab “User” on the left in Admin Dashboard. * Fill in textbox search and click “Search” | | | | |
| **Preconditions** | User login successful as an admin. | | | | |
| **Post Conditions** | * Success: Show all matched users. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Type keyword in textbox then click “Search” button  [Exception 1] | Show Table with list of users which have the name contain keyword. | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | | | | | |
| **Relationships**  N/A | | | | |  |
| **Business Rules**  N/A | | | | |  |

* 1. **<Administrator> Search User.**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE BAN USER SPECIFICATION | | | | | |
| Use-case No. | UC034 | Use-case Version | | 1.0 | |
| Use-case Name | Ban User | | | | |
| Author | To Dinh Tan | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Administrator | | | | |
| **Summary** | This use case is about ban a user function. | | | | |
| **Goal** | Admin can ban a user. | | | | |
| **Triggers** | * Admin click on tab “User” on the left in Admin Dashboard. * Click “Ban” link in column “Action”. | | | | |
| **Preconditions** | User login successful as an admin. | | | | |
| **Post Conditions** | * Success: User is banned. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Ban” link  [Exception 1] | Status of user change from “active” to “banned” “Ban” change to “Unban” link | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Go to login page. | | | | | | |
| **Relationships**  N/A | | | | |  |
| **Business Rules**   * If a user is banned, they can not login into system. | | | | |  |

* 1. **<Administrator> View Report.**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE VIEW REPORT SPECIFICATION | | | | | |
| Use-case No. | UC035 | Use-case Version | | 1.0 | |
| Use-case Name | View Report | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Administrator | | | | |
| **Summary** | This use case is about view all reports function. | | | | |
| **Goal** | Admin can view all reports from users. | | | | |
| **Triggers** | * Admin click on tab “Reports” on the left in Admin Dashboard. | | | | |
| **Preconditions** | User login successful as an admin. | | | | |
| **Post Conditions** | * Success: Show list of reports. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on tab “Reports”.  [Exception 1] | Show 2 table “Newest Report” and “Posts with most reports”:   * “Newest Reports”: No., Message, Post Type, Post Title, From (username), Date * “Posts with most reports”: No, Type, Title, Created By (username), Reports (number of reports about that post) | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Display Login page. | | | | | | |
| **Relationships**  N/A | | | | |  |
| **Business Rules**   * Newest report table contains all newest reports by time. * Post with most reports table contains all posts (story/album/photo) with highest number of reports. | | | | |  |

* 1. **<Administrator> Set Time of Recommend Service.**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE SET TIME OF RECOMMEND SERVICE SPECIFICATION | | | | | |
| Use-case No. | UC036 | Use-case Version | | 1.0 | |
| Use-case Name | Set Time of Recommend Service | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Administrator | | | | |
| **Summary** | This use case is about setting time of recommend service. | | | | |
| **Goal** | Admin can change time of recommend service. | | | | |
| **Triggers** | * Admin click on tab “Recommend Service” on the left in Admin Dashboard. * Choose Hour, Minute, Second and click “Save” button | | | | |
| **Preconditions** | User login successful as an admin. | | | | |
| **Post Conditions** | * Success: Time is changed. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Choose Hour, Minute, Second then click “Save” button  [Exception 1] | Show success message: “Recommend System Configuration Saved.” | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Go to login page. | | | | | | |
| **Relationships**  N/A | | | | |  |
| **Business Rules**   * Time of recommend service follows 24-hour standard. * Recommend Service run on established time. | | | | |  |

* 1. **<Administrator> Run Recommend Service.**
     1. Use Case Diagram



* + 1. Use Case Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE RUN RECOMMEND SERVICE SPECIFICATION | | | | | |
| Use-case No. | UC037 | Use-case Version | | 1.0 | |
| Use-case Name | Run Recommend Service | | | | |
| Author | Huynh Tran Minh Nhut | | | | |
| Date | 05/12/2014 | Priority | Normal | | |
| **Actor** | Administrator | | | | |
| **Summary** | This use case is about run recommend service function. | | | | |
| **Goal** | Admin can run recommend service immediately. | | | | |
| **Triggers** | * Admin click on tab “Recommend Service” on the left in Admin Dashboard. * Admin click “Do Recommend Now” button. | | | | |
| **Preconditions** | User login successful as an admin. | | | | |
| **Post Conditions** | * Success: Recommend Service is run. * Fail: Show error message. | | | | |
| **Main Success Scenario** | | | | | |
| |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Do Recommend Now” button  [Exception 1] | Show success message: “Action sent.” | | | | | | |
| **Alternative Scenario 1** | | | | | |
| N/A | | | | | |
| **Exceptions** |  | | | | |
| |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User Session Time Out | Go to login page. | | | | | | |
| **Relationships**  N/A | | | | |  |
| **Business Rules**   * Recommend Service is run immediately. * System detects and suggests posts to users. All posts and users are saved in database. | | | | |  |

### SOFTWARE SYSTEM ATTRIBUTES

#### Reliability

When user needs to log in to the website over a given period of time for using some specified services, the system should deliver those services as user’s expectation. The system must be prevented from producing unexpected situation or incorrect output. Based on the situation and users’ operation, reliability requirement can be different. However, in general, system must ensure the followings:

* Certain system functions are visible for only appreciate user, e.g. only administrator can manage users, only family owner can change background image of family.
* Give appropriate response to user operation, e.g. moving from screen to screen.
* In case of failure, system will show error message instead crash.

#### Availability

The system has to run 24/7 that means when any request at any given time is coming, the system should be available and able to respond.

#### Security

There is no specific security requirement. In general:

* Only user with valid account and appropriate role can access certain system functions.
* Prevent everyone, including system administrator, from knowing users’ passwords, except the owner.
* Check data integrity and validate input before execute recording into system.

#### Maintainability

When the system is deployed, there may be new or extra requirements. So that the system itself must be changeable that means development process must provide the ease to modify and extend. For the need to be easy to maintain, even though for new software team:

* Delivery product must have the software application packaged with all documented software requirements, software design, testing report and other material if any.
* For source code, all files and modules should have comment with short description and records of modification.
* In case of maintenance, a maintaining message should be shown to users when they enter to the website.

#### Portability

Any personal computer, which has an internet browser, is able to use this system. Therefore, the website must be developed and tested to make sure it can run smoothly with some popular web browsers, such as Internet Explorer, Mozilla Firefox and Google Chrome.

#### Performance

The system should cost the least time interval for user to finish their operation. Hence, screen layout, color block, style sheet and all web controls should be well arranged and easy to use, in order to help users save their time when interacting with the system.

### ENTITY RELATIONSHIP DIAGRAM

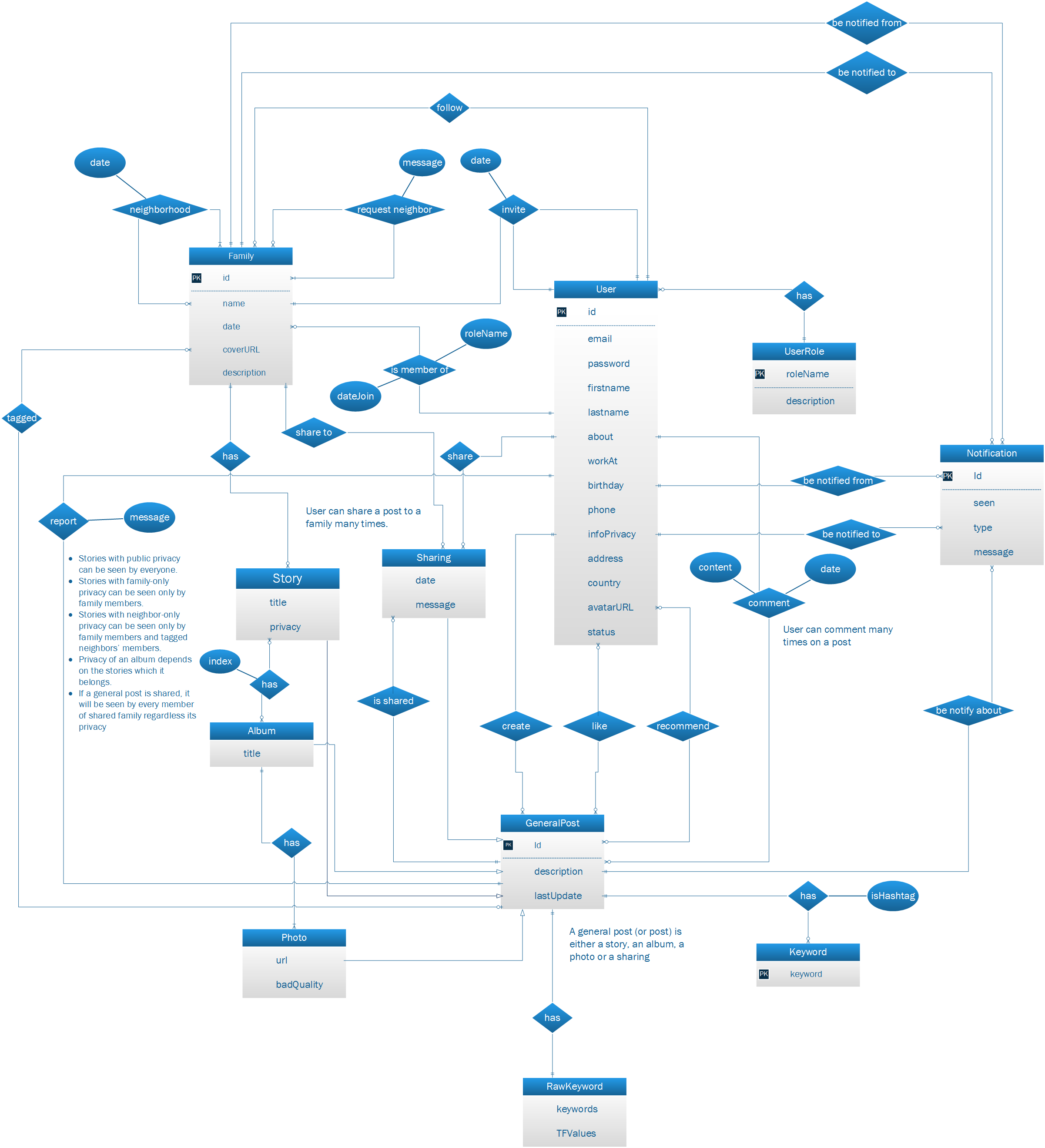


Figure 3: Entity relationship diagram

# SOFTWARE DESIGN DESCRIPTION

## DESIGN OVERVIEW

The intention of this document is to specify the design of the FAMSAM system in detail to describe the system implementation. This document will contain information about the system architecture, detailed description of components, data structures, user interface design and algorithms. This is to give the reader, a better understanding of the design process and procedure for this application. This document is also used by the developers for details about the system implementation.

Overview of document:

* The first section describes the introduction of document.
* The second section describes the system architecture.
* The third section gives the package diagrams that describe each package and relationship between them in the system.
* The fourth and fifth section show class diagrams and sequence diagram of each module to specify more details about the system implementation.
* The sixth section provides the information of user interface design.
* The seventh section gives the detailed database design.
* The eighth section describe two important algorithms being used in the system.

## SYSTEM ARCHITECTURAL DESIGN

### CHOICE OF SYSTEM ARCHITECTURE

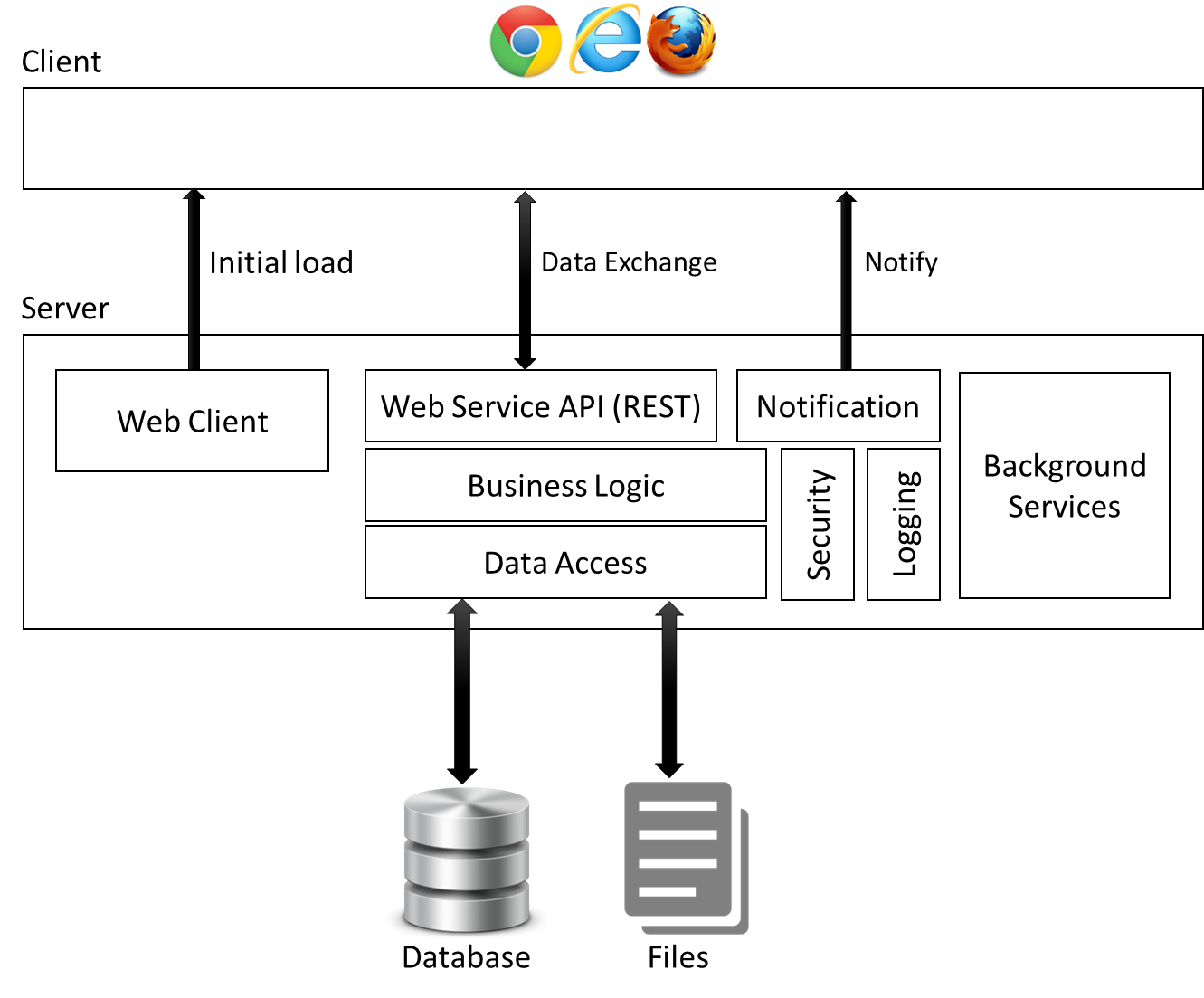
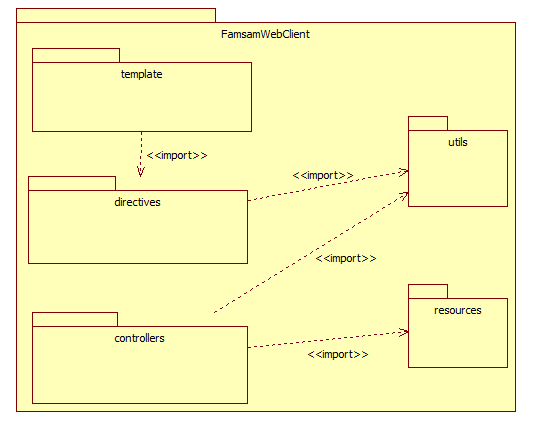


Figure 4. FAMSAM system architecture

* The aim of this architecture is to build single-page application.
* Client is presentation layer. It major responsible for UI rendering, client side routing, dynamic view, server data access. Client’s contents (presented in the diagram by Web Client) are first in server, once it initial loaded to client, no further browser refresh is needed, client-server transaction is data-only (using JSON). The Web Client is built on HTML5 and AngularJS framework.
* Server
  + Web Server API:
    - Provide an API for data exchange from client to server.
    - Provide basic security of authentication and authorization.
    - This component is implemented using ASP.NET Web API 2.
  + Have function of notifying to client using SignalR.
  + Do the business logic.
  + Data gathering and accessing from database via Entity Framework 6.
  + Database using Microsoft SQL 2012.
  + Files are simply windows file on hard drive.
* The background services contain services handled behind client-server transaction to improve system performance. In FAMSAM system they are window services.

## *PACKAGE DIAGRAM*

### WEB CLIENT PACKAGE DIAGRAM



Package diagram for Web Client implementation

* “template”: HTML, CSS for states of views (a state of view contains three major properties: URL to access state, template and controller).
* “directives”: define custom HTML elements or attributes which make richer template view.
* “controllers”: contains controllers, to define code behind for each state of views.
* “resources”: contains all the elements by which controllers can communicate with server.
* “utils”: code of utilities.

### SERVER PACKAGE DIAGRAM

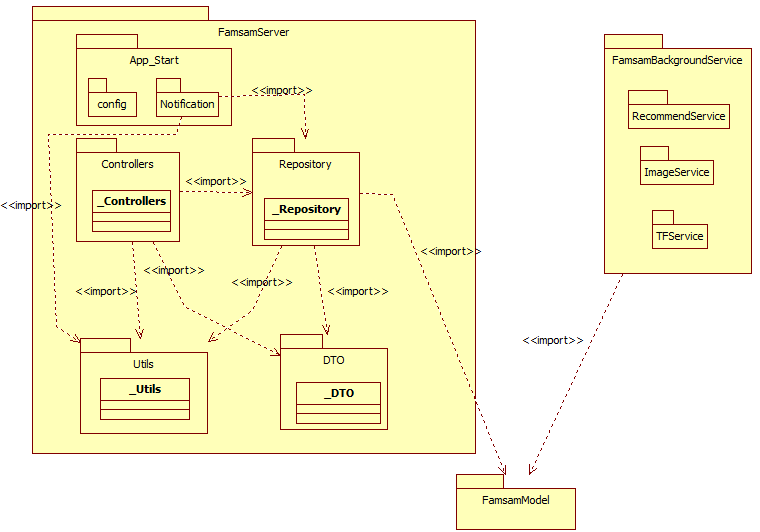
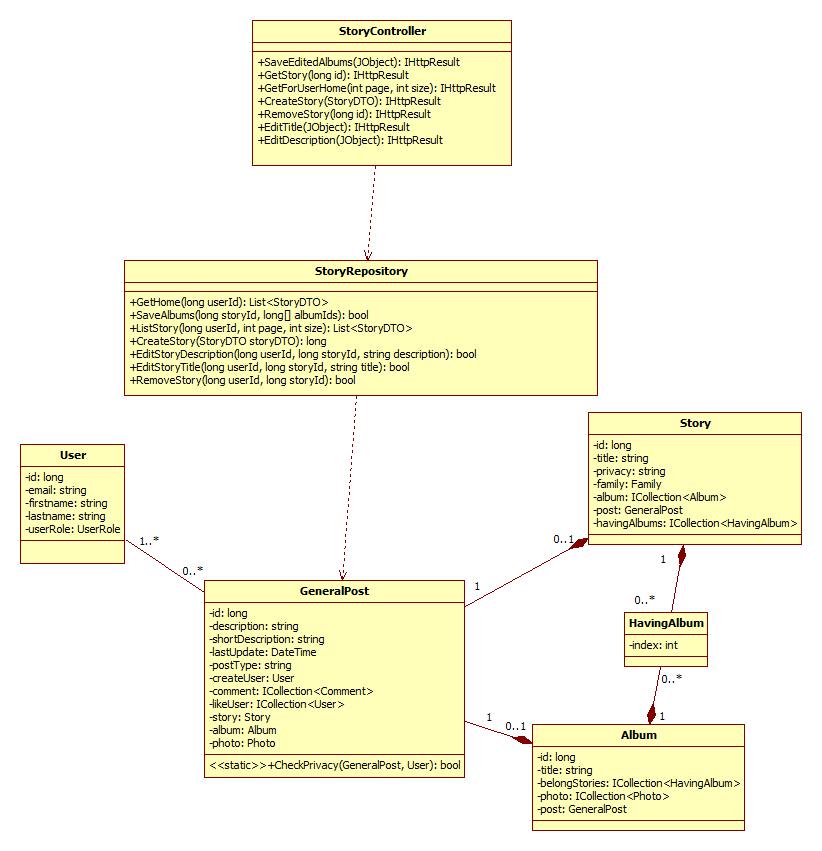


Figure 5 - Package diagram for FAMSAM Server and Background Services implementation

* FamsamServer:
  + App\_Start contains elements which are used when start server (IIS), they are:
    - Server configurations.
    - Notification process.
  + Controllers: contains ApiControllers of ASP.NET Web API.
  + Repository: has responsible for access to model and prepare DTOs using Entity Framework.
  + DTO: contains data transfer objects which are used for client-server data exchange.
  + Utils: code for utilities.
  + All classes in each sub-packages of FamsamServer have package name as suffix.
* FamsamBackgroudService: contains background services. Background services handle heavy task at some request-response of client and server. The services will handle that task at background in order to decrease time interval for user to finish their operation.
* FamsamModel: contains entities from Entity Framework Code-First. In Famsam, both repository and background services access into one database (FAMSAM’s database), hence they must use the same model.

## *CLASS DIAGRAM*

### MANAGE STORY CLASS DIAGRAM



Manage story class diagram

* **StoryController**: Defines API for client-server request and response. The controller contains functions for APIs which relate to story management.
* **StoryRepository**: Defines functions for story management. The Story Repository use Entity Framework to manipulate model relate to story management. Models are:
  + **User**: Author who create the story, this model is needed to get information of story’s creator.
  + **GeneralPost**: A story, an album or a photo is considered a post. The model **GeneralPost** is used to hold common information that each story, album, photo should have. Also each Story/Album/Photo has a require **GeneralPost***.* The CheckPrivacy() method is used to check whether a user have permission to view the **Story** or **Album** with corresponds **GeneralPost** according to its privacy. For more information about post’s privacy see [Entity Relationship Diagram at Part C, chapter II](#_ENTITY_RELATIONSHIP_DIAGRAM).
  + **Story:** The model to hold specific information of a story.
  + **HavingAlbum:** The relationship of story and album, a story can contains many album, album order in the story is specified by *index* property.
  + **Album:** The model to hold specific information of an album.

### MANAGE ALBUM CLASS DIAGRAM

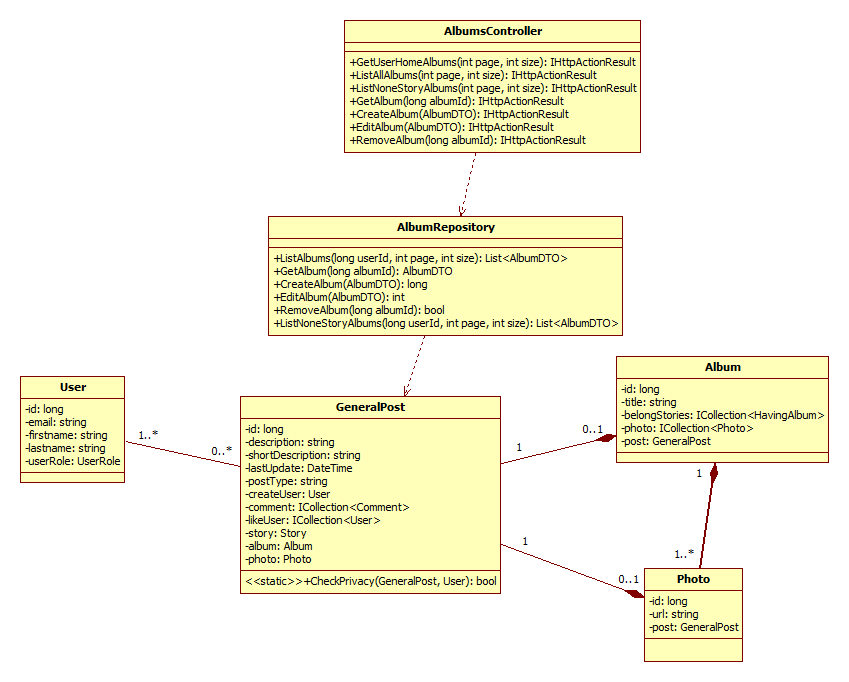


Figure 6 - Manage Album Class Diagram

* As similar to Manage Story class diagram but here we manage album, each album has photos.

### FAMILY CLASS DIAGRAM

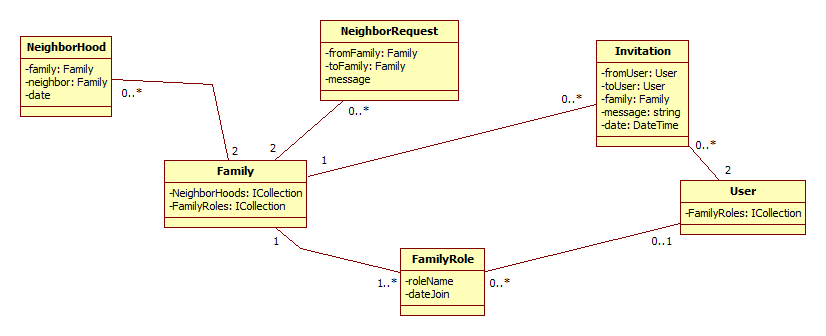
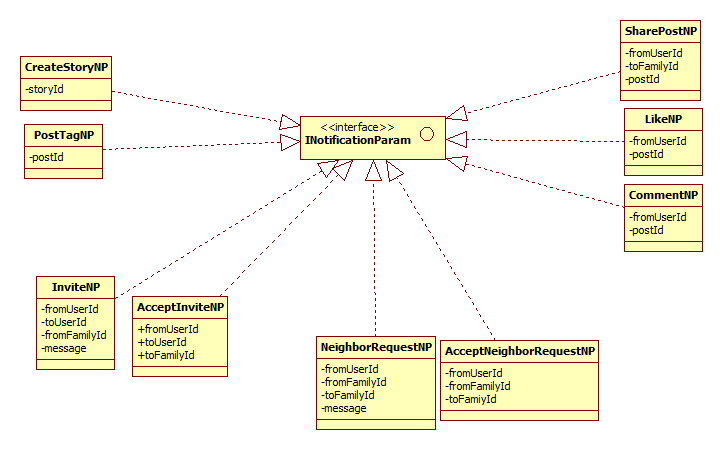


Figure 7 - Family Class Diagram

The above class diagram represent model and their relationship to implement Family features which are:

* Create family.
* Invite family member.
* Accept invitation to family.
* Request neighbor.
* Accept neighbor request.

### NOTIFICATION CLASS DIAGRAM



1. Figure 8 - Notification Class Diagram

The above class diagram is to represent concrete type of notification which are further used in Create Story Sequence diagram, Invite a person to a family Sequence diagram, Sharing a Post Sequence Diagram and Notification Sequence Diagram. See [Chapter V](#_SEQUENCE_DIAGRAM) of this part.

## *SEQUENCE DIAGRAM*

### LOGIN

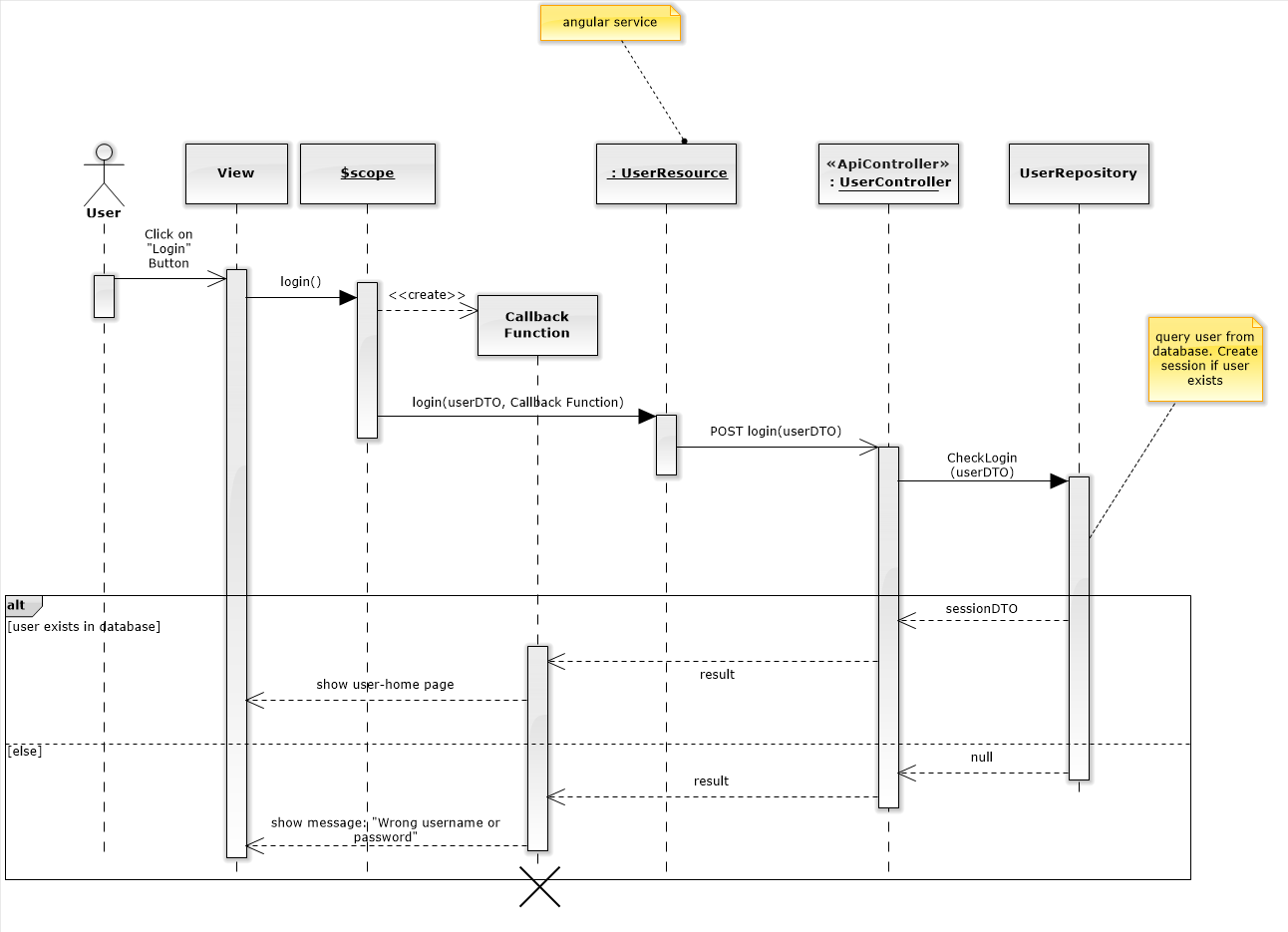


Figure 9: Login sequence diagram

Note:

* $scope is an object provided by AngularJS framework for each state of views, its properties can be bound to the view in order to make view dynamic, its methods can be bound to the view as an event handler for action controls.

### SERVER AUTHORIZE

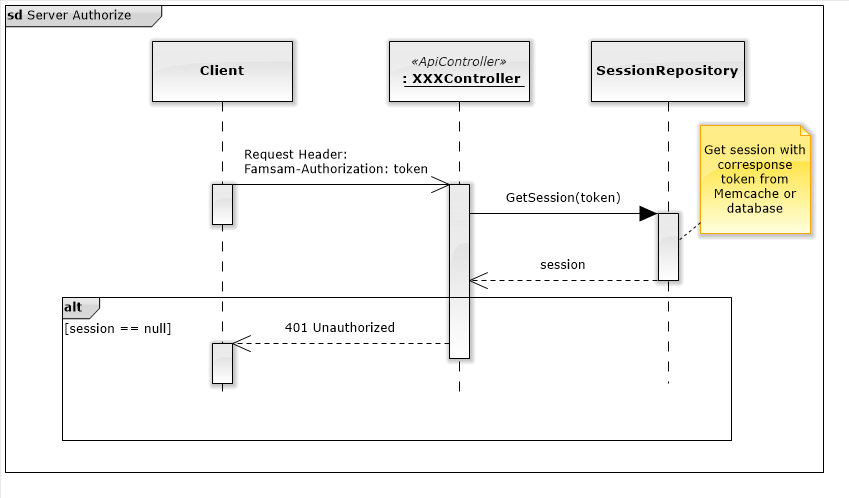


Figure 10: Server authorize sequence diagram

Note:

* XXXController: any controller in Controllers package.

### CREATE STORY

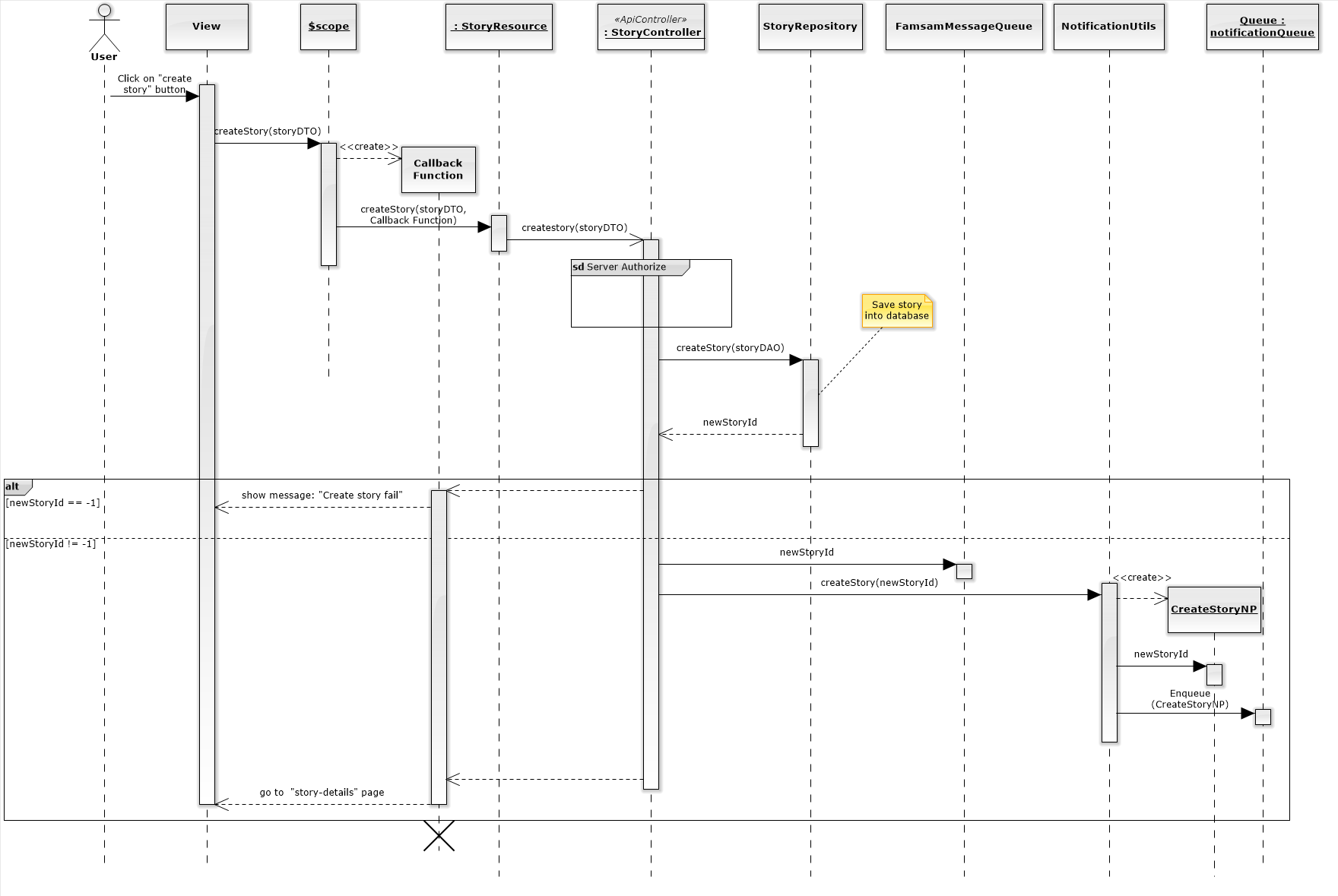


Figure 11: Create Story sequence diagram

### INVITE A PERSON TO A FAMILY

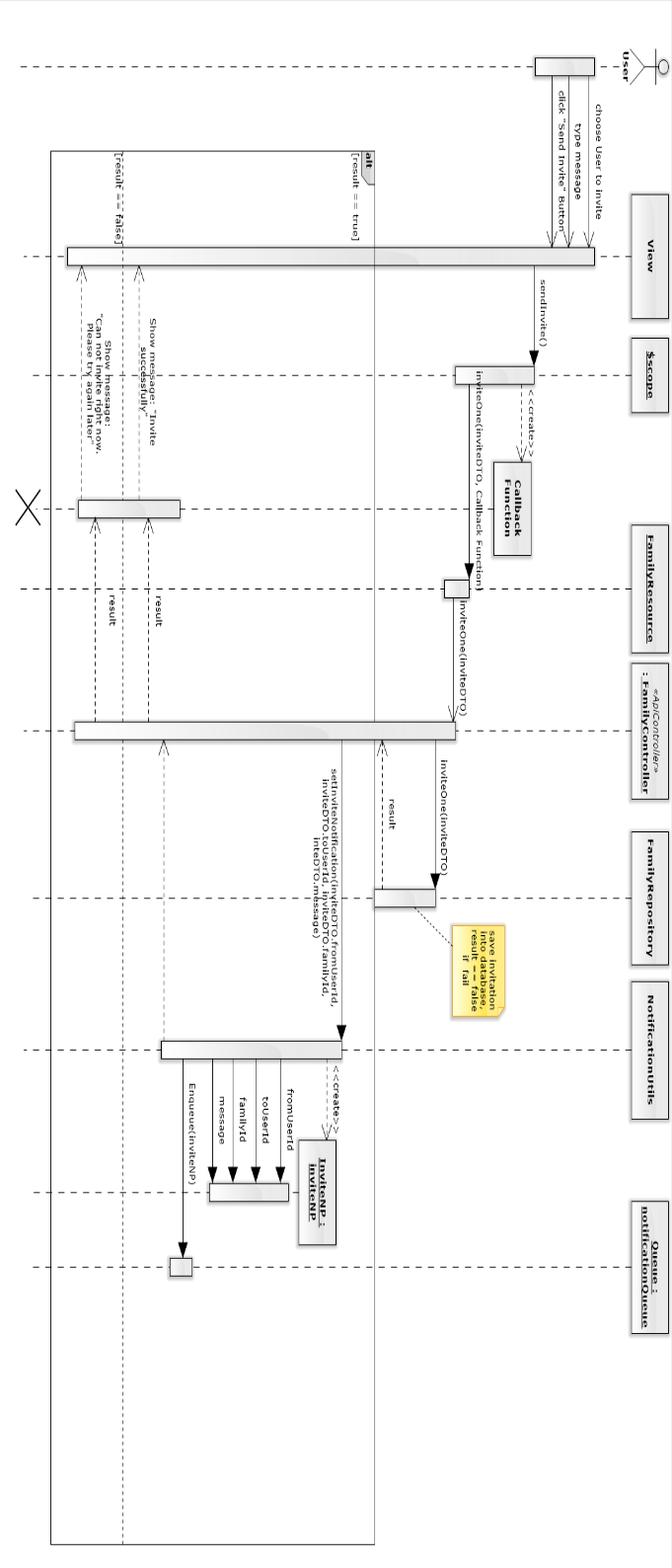


Figure 12: Invite a person to family sequence diagram

### SELECT PHOTO TO UPLOAD

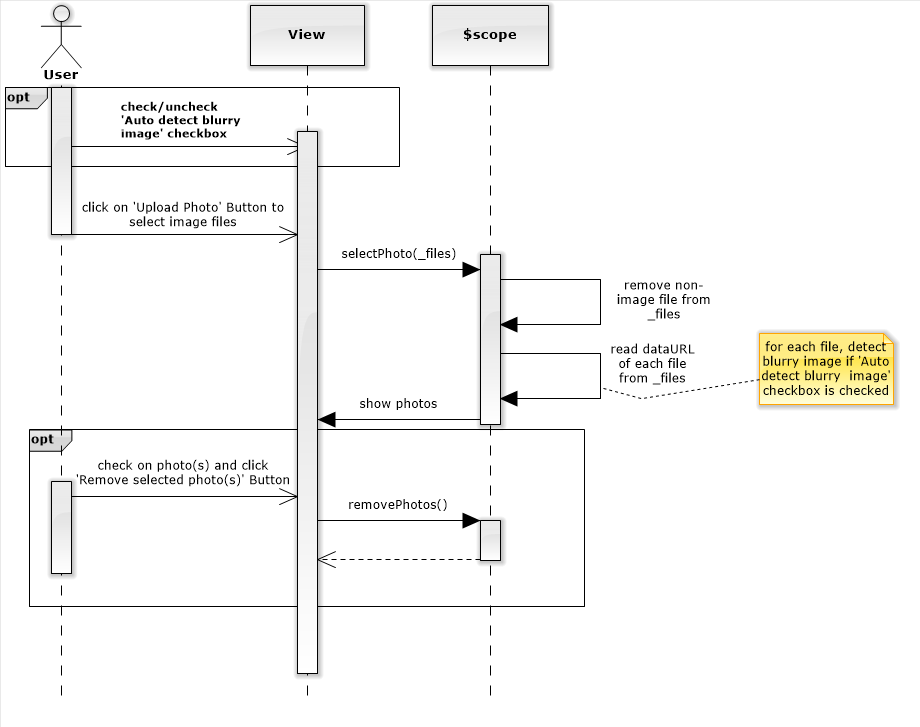


Figure 13: Select photo to upload sequence diagram

### UPLOAD PHOTO

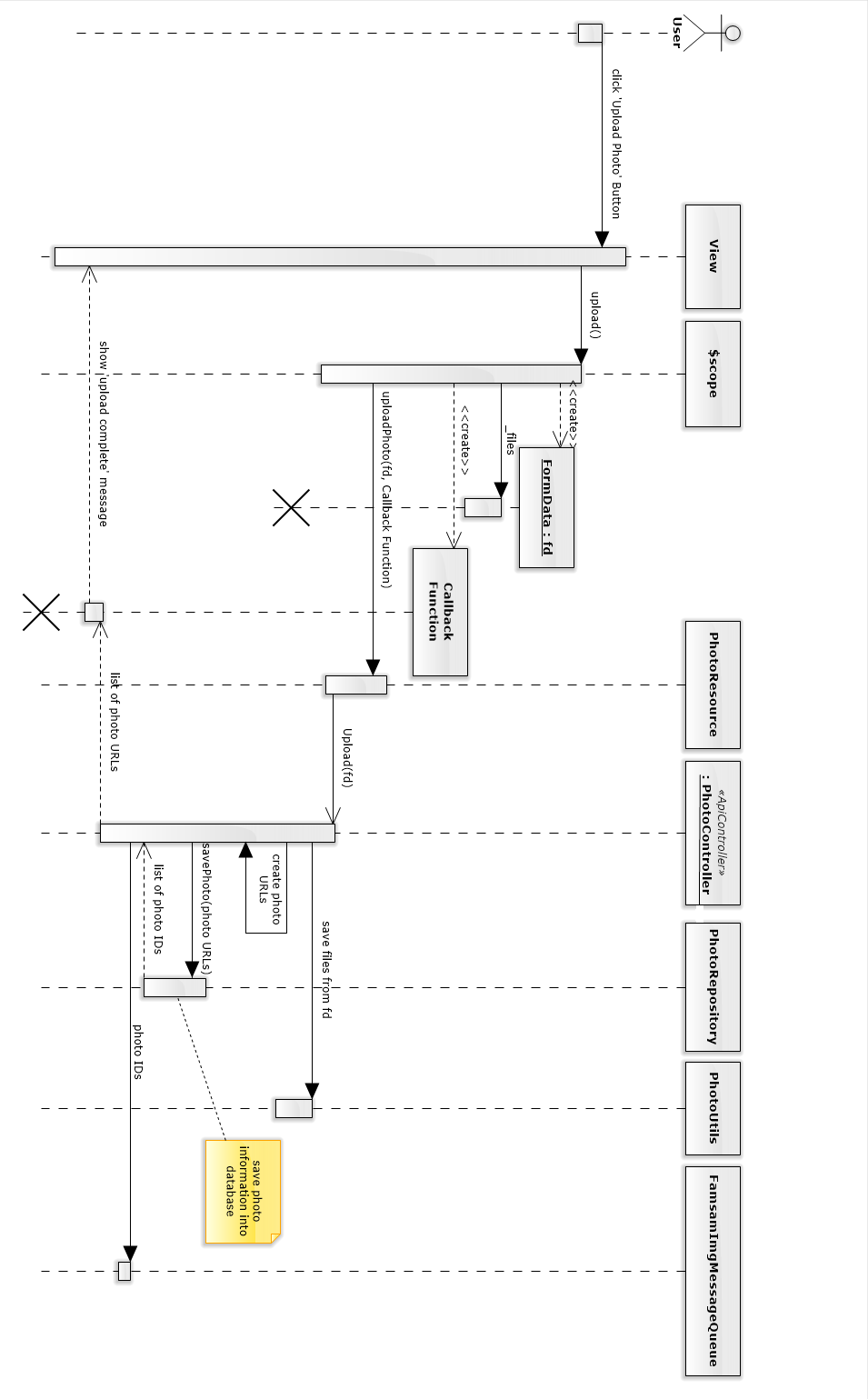


Figure 14: Upload photo sequence diagram

### SHARING A POST (STORY/ALBUM)

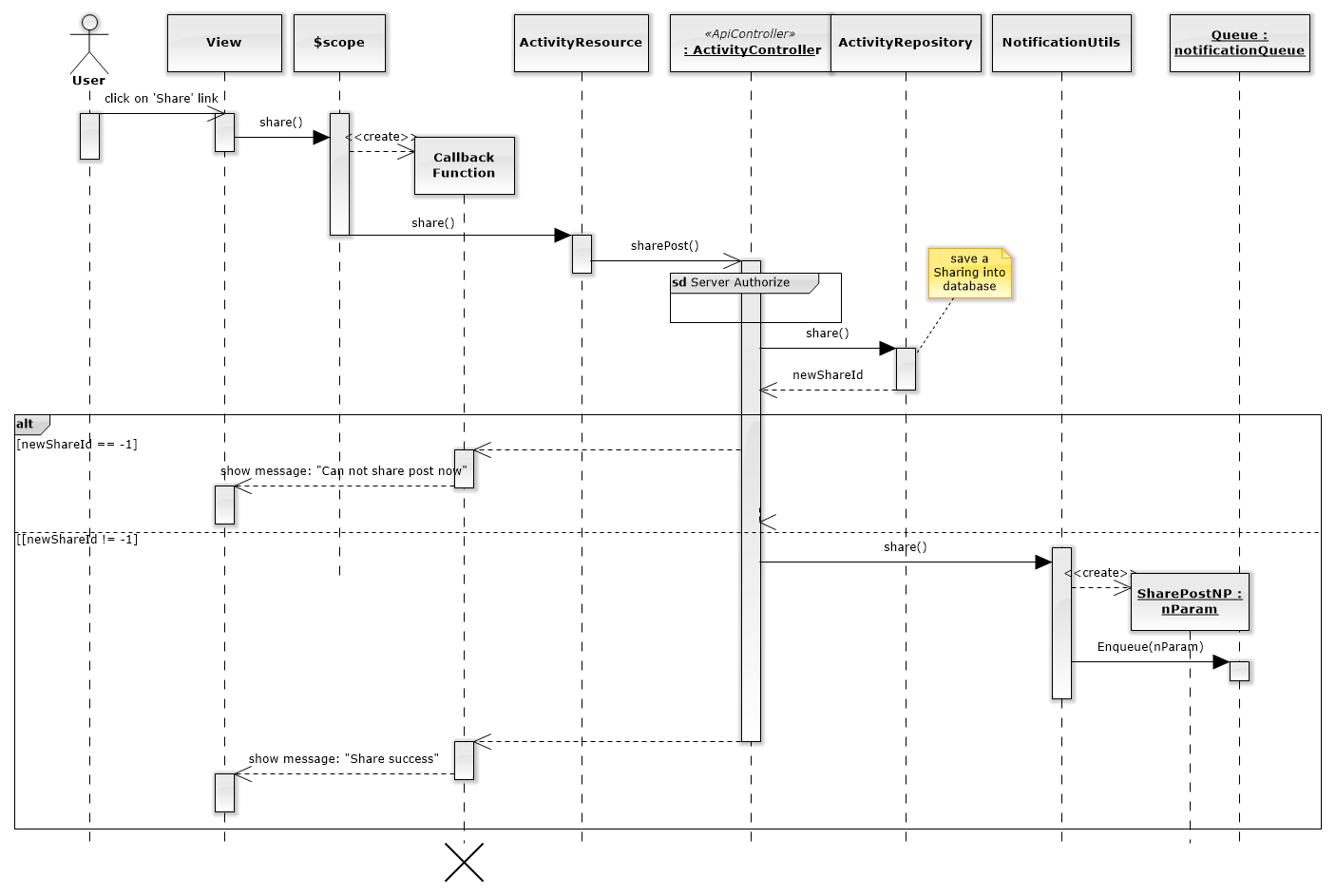


Figure 15: Sharing sequence diagram

### CREATE THUMBNAIL IMAGE

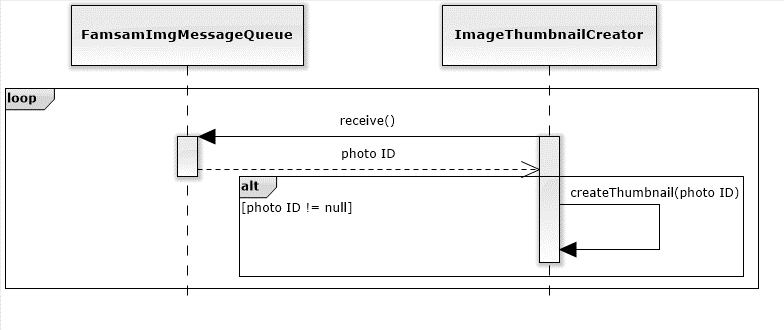


Figure 16: Create thumbnail image sequence diagram

### NOTIFICATION

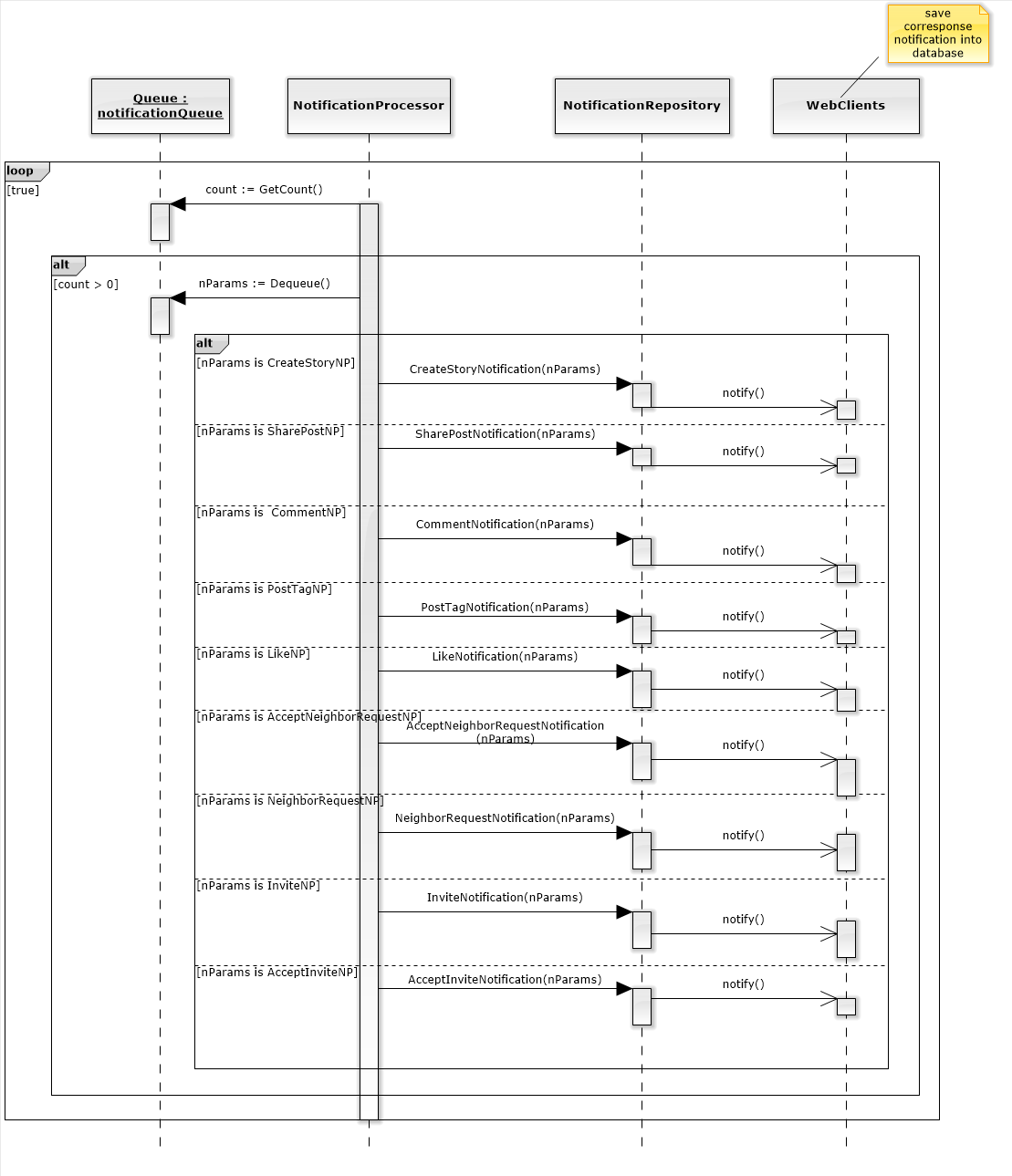


Figure 17: Notification sequence diagram

### CALCULATE TF VALUES

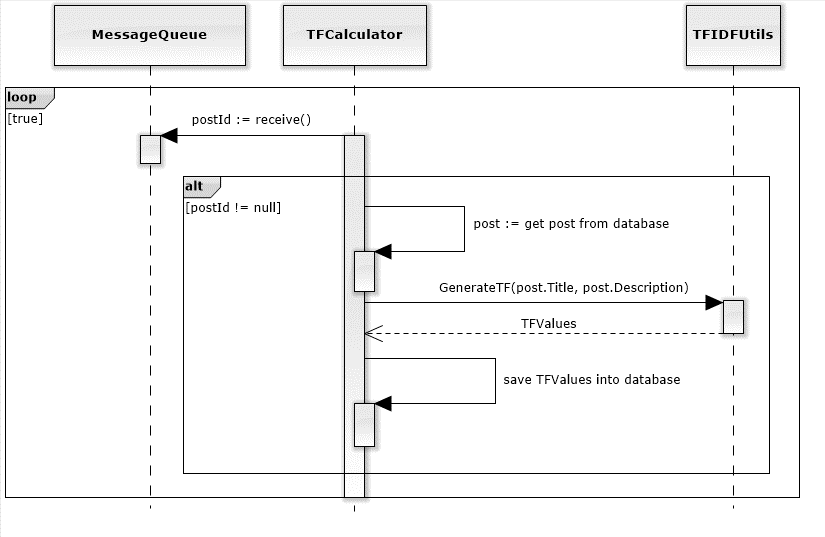
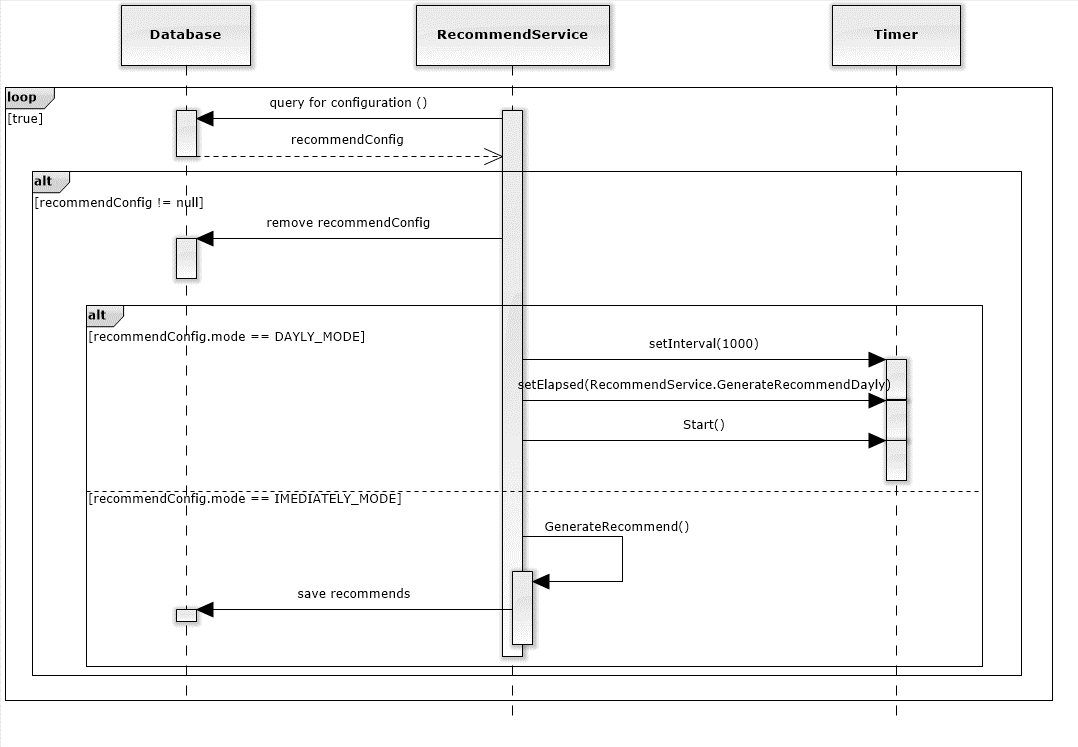


Figure 18: Calculate TF Values sequence diagram

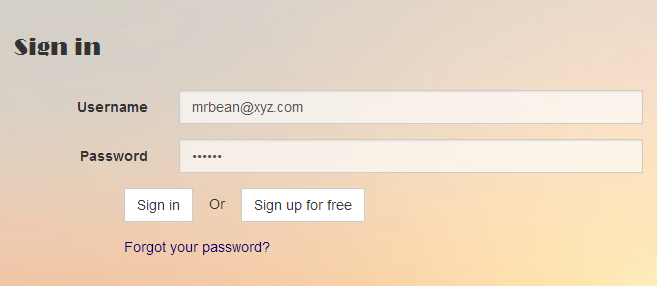
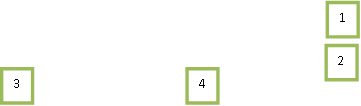
### RECOMMEND SERVICE



## *USER INTERFACE DESIGN*

### LOGIN (SIGN IN) SCREEN

* 1. Screen Images



* 1. Objects and Actions

Data Fields

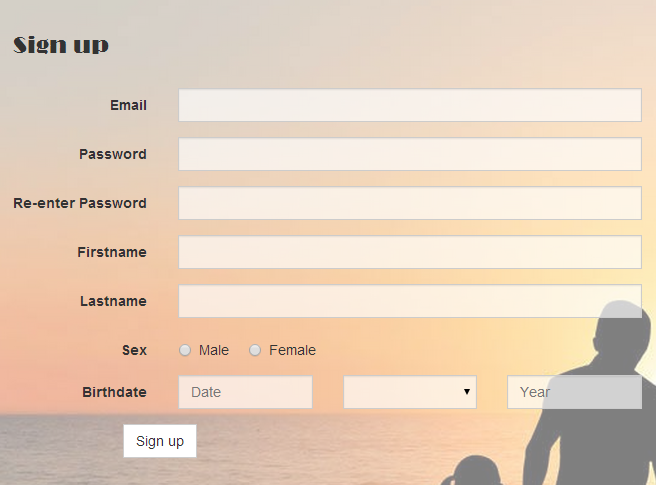
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Mandatory** | **Data type** | **Max length** | **Description** |
| 1 | Username | text | Y | nvarchar | 255 | User email to login |
| 2 | Password | password | Y | nvarchar | 255 | Password to login |

Data Actions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Action** | **Description** | **Validation** |
| 3 | Sign in | Button | On click | Sign in (login) |  |
| 4 | Sign up for free | Button | On click | Go to register (sign up) page |  |

### REGISTER (SIGN UP) SCREEN

* 1. Screen Images



1

2

3

4

5

6

7

8

9

10

* 1. Objects and Actions

Data Fields

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Mandatory** | **Data type** | **Max length** | **Description** |
| 1 | Email | text | Y | nvarchar | 255 | User email to sign up |
| 2 | Password | password | Y | nvarchar | 255 | Password to sign up |
| 3 | Re-enter Password | password | Y | nvarchar | 255 | Confirm password to login |
| 4 | Firstname | text | Y | nvarchar | - | First name of user |
| 5 | Lastname | text | Y | nvarchar | - | Last name of user |
| 6 | Sex | radio | Y | bit | - | Indicate wheather user is male or female. |
| 7 | Birthday | number | Y | int | - | Day of birth of user |
| 8 | Birthmonth | number | Y | int | - | Month of birth of user |
| 9 | Birthyear | number | Y | int | - | Year of birth of user |

Data Actions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Action** | **Description** | **Validation** |
| 10 | Sign up | Button | On click | Sign up (register) |  |

### HOMEPAGE SCREEN

* 1. Screen Images

3

2

1

4



* 1. Objects and Actions

Data Fields

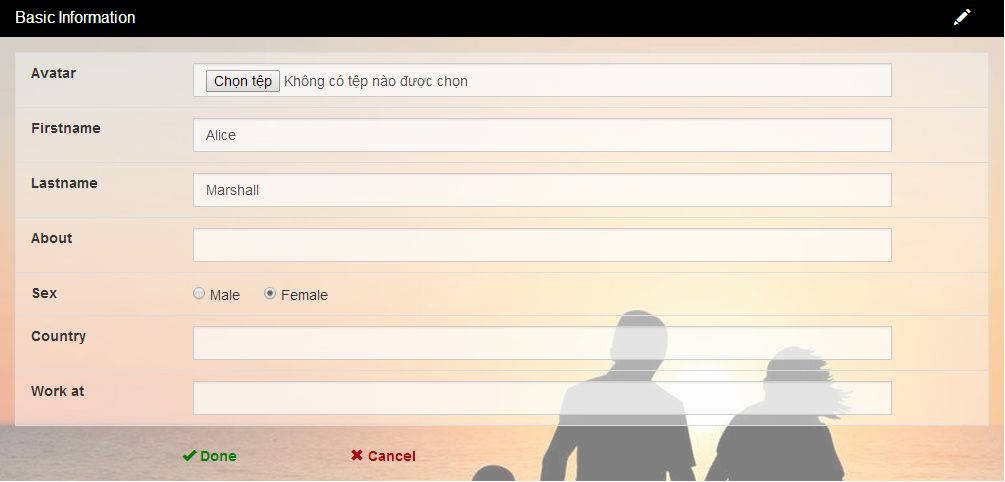
(n/a)

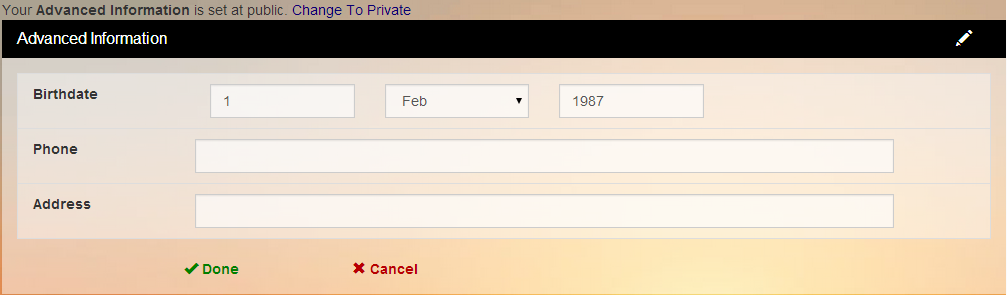
Data Actions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Action** | **Description** | **Validation** |
| 1 | Family Name | link | On click | Forward to family home page |  |
| 2 | Story Name | link | On click | Forward to story details page |  |
| 3 | Album Title | link | On click | Forward to album details page |  |
| 4 | Album Image | image | On click | Forward to album details page |  |

### USER PERSONAL INFORMATION SCREEN

* 1. Screen Images





* 1. Objects and Actions

Data fields

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Mandatory** | **Data type** | **Max length** | **Description** |
| 1 | Avatar | file | N | file | - | Avatar of user |
| 2 | Firstname | text | Y | nvarchar | 255 | First name of user |
| 3 | Lastname | text | Y | nvarchar | 255 | Last name of user |
| 4 | About | text | N | nvarchar | - | Description about user |
| 5 | Sex | checkbox | Y | bit | - | Indicate whether user is male or female. |
| 6 | Country | text | N | nvarchar | 20 | Country of user |
| 7 | Work at | text | N | nvarchar | 20 | Company’s name of user |
| 8 | Birthday | number | Y | int | - | Day of birth of user |
| 9 | Birthmonth | number | Y | int | - | Day of month of user |
| 10 | Birthyear | number | Y | int | - | Day of month of user |
| 11 | Phone | text | N | nvarchar | 20 | Phone number of user |
| 12 | Address | text | N | nvarchar | 255 | Address of user |

Data Actions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Action** | **Description** | **Validation** |
| 13 | Done basic info | button | On click | Submit basic info data to server | Inputted data must be valid |
| 14 | Cancel basic info | button | On click | Cancel edit basic info |  |
| 15 | Change To Private | link | On click | Submit private privacy of user’s advance info to server |  |
| 16 | Change To Public | link | On click | Submit public privacy of user’s advance info to server |  |
| 17 | Done Advance information | button | On click | Submit advance info data to server | Inputted data must be valid |
| 18 | Cancel basic info | button | On click | Cancel edit basic info |  |

### FAMILY HOMEPAGE SCREEN

* 1. Screen Images



1

2

3

4

4

5

6

5



* 1. Object and Actions

Data fields

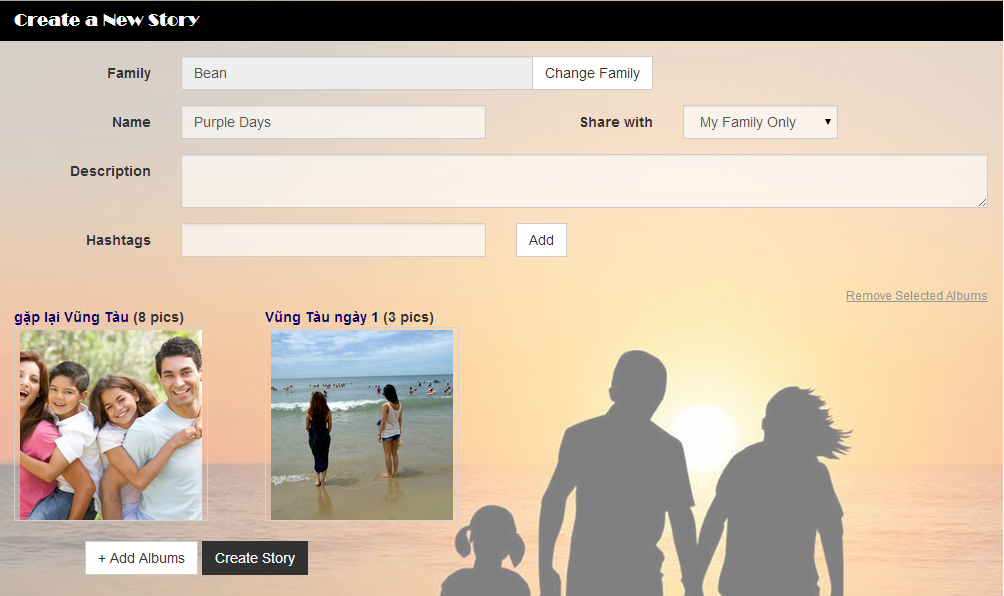
N/A

Data actions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Action** | **Description** | **Validation** |
| 1 | Invite Someone | Button | On click | Open ‘invite someone to family’ pop-up | - |
| 2 | Invite Neighbor | Button | On click | Open ‘invite neighbor’ pop-up | - |
| 3 | Change family background | Button | On click | To chang family background | - |
| 4 | User’s name | Link | On Click | forward to user’s personal info | - |
| 5 | Post title | Link | On click | forward to post (story/album) details page | - |
| 6 | Like | Link | On click | like the post (story/album) | - |

### CREATE STORY SCREEN

* 1. Screen Images



11

10

7

9

8

6

5

4

3

2

1

Figure 19 - Create Story Screen

* 1. Object and Actions

Data Fields

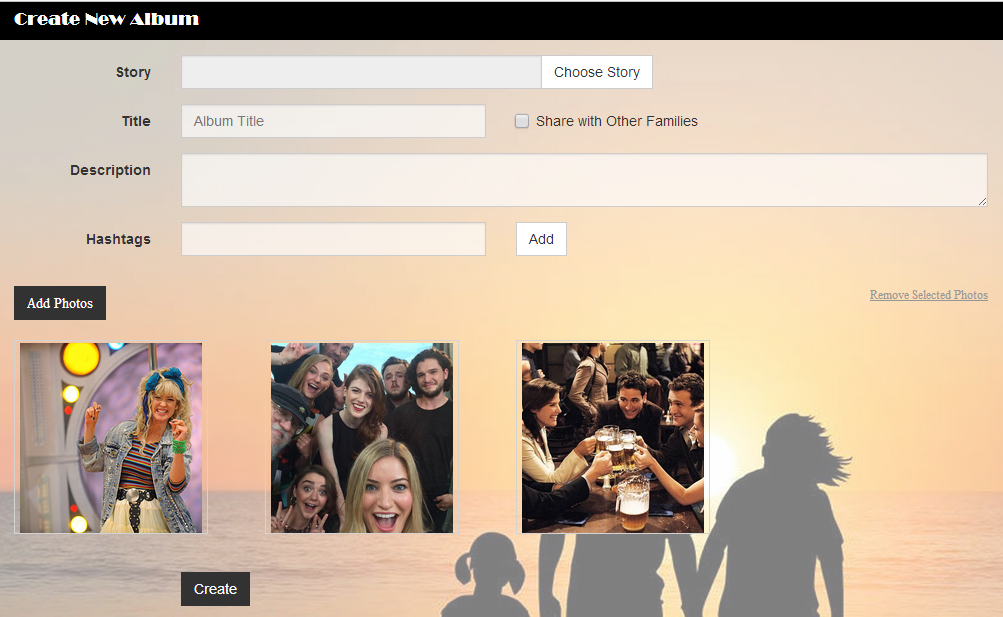
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Mandatory** | **Data type** | **Max length** | **Description** |
| 1 | Family Name | Text | Y | nvarchar | 20 | Family Name of the story |
| 3 | Story Name | Text | Y | nvarchar | 20 | Name (Title) of the story |
| 4 | Share with | Select | Y | nvarchar | 10 | Privacy of the story |
| 5 | Description | Text | N | nvarchar | 255 | Description of the story |
| 6 | HashTag | Text | N | nvarchar | 100 | Hash tag of the story |

Data Actions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Action** | **Description** | **Validation** |
| 2 | Change Family | Button | On click | Show pop-up to choose story’s family | - |
| 7 | Add HashTag | Button | On click | Add typed hashtag into current story | - |
| 8 | Add Albums | Button | On click | Show pop-up to choose or create album | - |
| 9 | Create Story | Button | On click | Create Story | - |
| 10 | Remove selected albums | Link | On click | Remove selected album | - |
| 11 | Select album | Image | On click | Click to select album | - |

### CREATE ALBUM SCREEN

* 1. Screen Images



11

10

9

8

7

6

5

4

2

1

3

* 1. Object and Actions

Data Fields

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Mandatory** | **Data type** | **Max length** | **Description** |
| 1 | Story name | Text | N | nvarchar | 20 | Story name which contains the album |
| 3 | Album Title | Text | Y | nvarchar | 20 | Title of the album |
| 5 | Description | Text | N | nvarchar | 255 | Description of the album |
| 6 | HashTag | Text | N | nvarchar | 100 | Hash tag of the story |

Data Actions

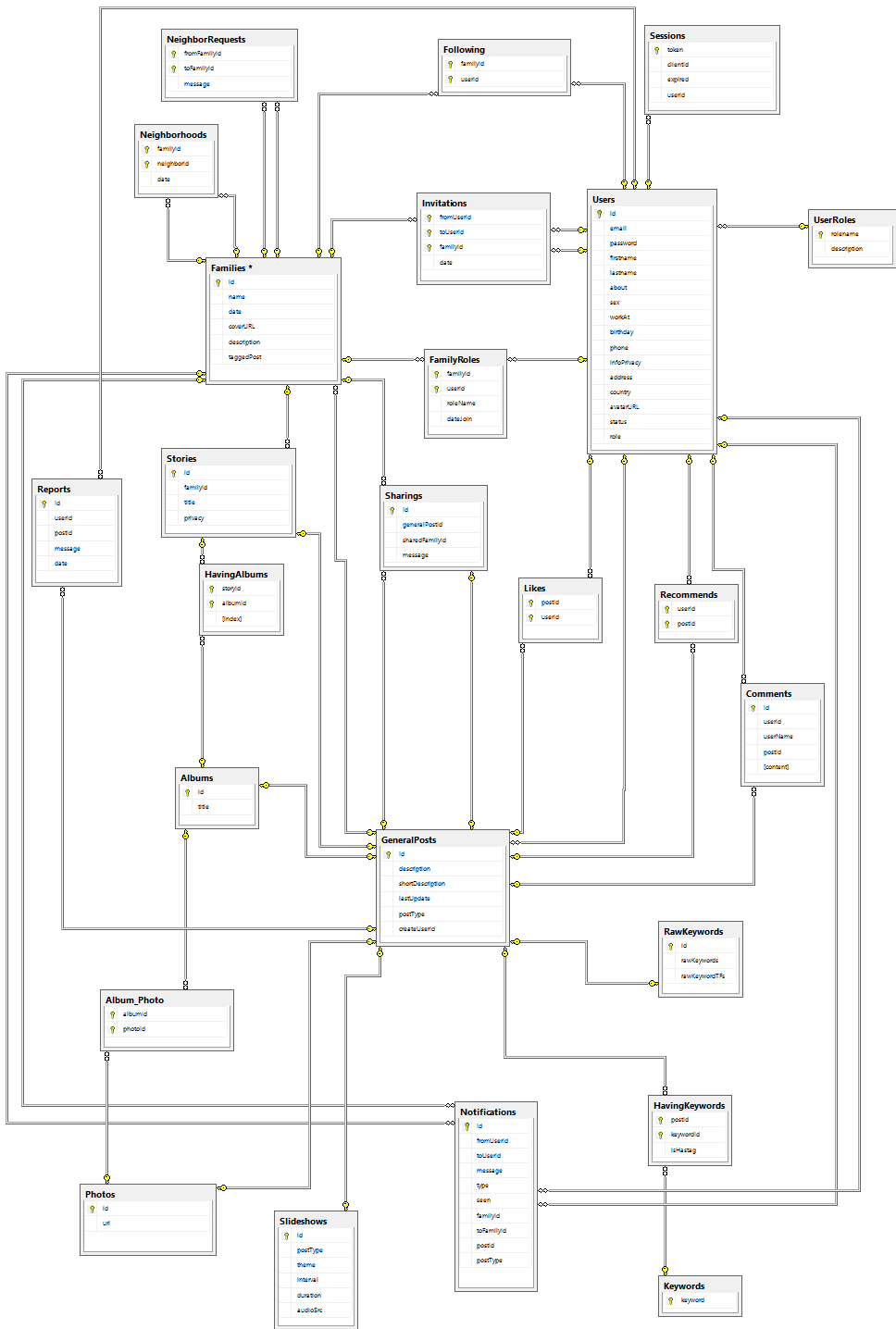
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field No** | **Field name** | **Type** | **Action** | **Description** | **Validation** |
| 2 | Choose Story | Button | On click | Show pop-up to choose story for the album | - |
| 4 | Share with other families | Checkbox | On click | Indicate whether the album is shared with other families or not | - |
| 7 | Add HashTag | Button | On click | Add typed hashtag into current album | - |
| 8 | Add Photo | Button | On click | Show pop-up to upload photo | - |
| 9 | Remove selected photos | Link | On click | Remove selected photos | - |
| 10 | Select photo | Image | On click | Click to select photo | - |
| 11 | Create | Button | On click | Submit form to create album | - |

### UPLOAD PHOTO SCREEN

### STORY DETAIL SCREEN

### PHOTO DETAIL SCREEN

## DATABASE DESIGN



**User Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Fields | Data Type | Allow Null | Reference Table |
| PK | id | bigint | N |  |
|  | email | nvarchar(255) | N |  |
|  | password |  |  |  |
|  | firstname |  |  |  |
|  | lastname |  |  |  |
|  | about |  |  |  |
|  | sex |  |  |  |
|  | workAt |  |  |  |
|  | birthday |  |  |  |
|  | phone |  |  |  |
|  | infoPrivacy |  |  |  |
|  | address |  |  |  |
|  | country |  |  |  |
|  | avatarURL |  |  |  |
|  | status |  |  |  |
| FK | role |  |  | UserRole |

**Family Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Fields | Data Type | Allow Null | Reference Table |
| PK | id | bigint | N |  |
|  | name |  |  |  |
|  | date | datetime |  |  |
|  | coverURL | nvarchar(255) |  |  |
|  | description | nvarchar(100) |  |  |
| FK | taggedPost | bigint |  | GeneralPost |

**GeneralPost Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Fields | Data Type | Allow Null | Reference Table |
| PK | id | bigint | N |  |
|  | description | nvarchar(max) | Y |  |
|  | shortDescription | nvarchar(203) | Y |  |
|  | lastUpdate | datetime | N |  |
|  | postType | nvarchar(20) | N |  |
| FK | createUserId | bigint | N | User |

Note:

* postType must be one of these string: “Story”, “Album”, “Photo”, “Sharing”

**Story Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Fields | Data Type | Allow Null | Reference Table |
| PK, FK | id | bigint | N | GeneralPost |
| FK | familyId | bigint | N |  |
|  | title | nvarchar(100) | N |  |
|  | privacy | nvarchar(20) | N |  |

Note:

* Story privacy must be one of these string: “public”, “family only”, “neighbor only”

**Album Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Fields | Data Type | Allow Null | Reference Table |
| PK, FK | Id | bigint | N | GeneralPost |
|  | title | nvarchar(100) |  |  |

Photo Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Fields | Data Type | Allow Null | Reference Table |
| PK, FK | Id | bigint | N | GeneralPost |
|  | url | nvarchar(255) | N |  |

## ALGORITHMS DESCRIPTION

### DETECT BLURRY IMAGE

The paper of Marziliano P. et al (2002) present a no-reference blur metric for images based on the analysis of the spread of the edges in the image. Based on its result, below is the description of the algorithm implemented in FAMSAM system:

Step 1. Grayscale image

Step 2. Use vertical Sobel filter to filter image grayscale into edges. The output of Sobel filter is an image in which, if pixel value is greater or equal 20, the pixel is detected as edge.

Step 3. Scan pixels by rows. For each row, if a pixel is detected as edge, calculate edge width at that pixel position (See example below).

Step 4. Calculate the average of all edge widths of all pixels which is detected as edges, we call the average HORIZONTAL\_BLUR.

Step 5. Do similar to step 2 with horizontal Sobel filter.

Step 6. Do similar to step 3 but this time we scan pixels by columns.

Step 7. Do similar to step 4 then we call the average VERTICAL\_BLUR.

Step 8. If either HORIZONTAL\_BLUR or VERTICAL\_BLUR is greater than or equal 7.3, the image is blurry.

*Example of calculate edge width at an edge pixel in a pixel row of image:*

Figure 19 illustrate an example of pixel value at positions in a pixel row of an image. The edge width at an edge pixel is the distance of the nearest local maxima and minima on two side of the current pixel. For example P1 is a pixel detected as edge, the nearest local maxima and minima on two side of P1 is P2 and P2’, so edge width at P1 is the distance of P2 and P2’.

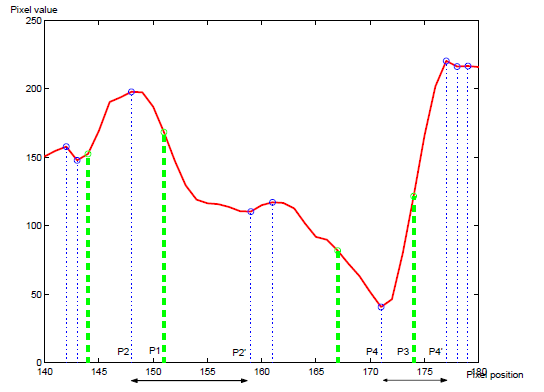


Figure 20 - Example of a pixel row with pixel value and position (Marziliano P. et al, 2002)

### RECOMMEND SIMILAR POST (STORY/ALBUM) BASED ON USER’S LIKES

1. TF
2. IDF
3. TF.IDF
4. Post profile
5. User profile

## REFERENCES FOR PART D

* 1. Marziliano P, Dufaux F, Winkler S, Ebrahimi T (2002) ‘*A no-reference perceptual blur metric.*’ International Conference on Image Processing, Rochester, NY.
  2. Leskovec, J., Rajaraman, A., Ullman, J. (2014) ‘*Mining of Massive Datasets*’, [Online], Available: <http://www.mmds.org/> [Nov 2014].

# SOFTWARE TEST DOCUMENTATION

## INTRODUCTION

### SYSTEM OVERVIEW

This document describes the approach and methodologies used to plan, organize and manage the testing of FAMSAM system. It describes the implementation details of test cases and technical details of how the product features should work.

This document also provides the checklists of validation and submission as well as the test results.

### TEST APPROACH

Test goals: the testing aims to identify the remaining bugs in the system before releasing and confirm the consistency in design and implementation.

Test type: black box testing.

Test level: system and integration test.

## TEST PLAN

The following part describes the features to be tested or not to be tested, testing tools and test environment.

### FEATURES TO BE TESTED

There are four areas to be focused on during the test:

1. **Guest – Common functions**

Function for user who first visit the website.

* Login.
* Register.

1. **Logged in User – Family functions**

Functions for logged user to manage his/her family and make neighbor:

* Create family.
* Invite a person to family.
* Make neighbor with other family.
* View family homepage.

1. **Logged in User – General Post (Story/Album) functions**

Functions for logged user to manage story, album:

* Create album
* Edit album
* Remove album
* Create story
* Edit story title
* Edit story description
* Add album to story
* Remove album from story
* Change order of album in story
* Remove story

1. **Logged in User – Social network functions**

Functions for logged user to do their activity of social network:

* Share story/album
* Like story/album/photo
* Comment on story/album/photo
* Follow family

### FEATURES NOT TO BE TESTED

* Admin management.
* Report.
* Create slideshow for story/album.

### TESTING TOOLS AND ENVIRONMENT

Testing tools: no special testing tools

Testing environment:

* Operating system: Microsoft Window 7
* Browser: Google Chrome 36+

## *TEST CASES*

### GUEST – COMMON FUNCTIONS

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Pre-Conditions** | **Test Case Procedure** | **Expected Output** | **Result** | **Test Date** | **Note** |
| Module Register | | | | | | | |
| [Guest-1] | Test empty field email and password | none | 1.Click "Sign up for free" button at login page  2. Fill all information in register form as follow:  +Email:  +Password:  +Re-enter Password:  +Firstname: “abc”  +Lastname: “abc”  +Sex: Male  +Birthdate: 1/2/1987 | “Sign up” button is disabled | Untested | 1-12-2014 | NhutHTM |
| [Guest-2] | Test register when success | there’s no User with email=[alice@yahoo.com](mailto:alice@yahoo.com) in database | 1.Click "Sign up for free" button at login page  2. Fill all information in register form as follow:  +Email: alice@yahoo.com  +Password: “Password1”  +Re-enter Password: “Password1”  +Firstname: “Alice”  +Lastname: “Marshall”  +Sex: Female  +Birthdate: 1/2/1987  3. Click on “Sign up” button | - Redirect to login page.  - Show message: “Sign up success, you can now login”.  - Username input is filled with “alice@yahoo.com” | Untested | 1-12-2014 | NhutHTM |
| [Guest-3] | Test register with existed email | there’s already existed a User with email=[alice@yahoo.com](mailto:alice@yahoo.com) in database | 1. Click “Sign up for free” button at login page.  2. Fill all information in register form as follow:  + Email: [alice@yahoo.com](mailto:alice@yahoo.com)  + Password: “Password1”  + Re-enter Password: “Password1”  + Firstname: “Alice”  + Lastname: “Marshall”  + Sex: Female  + Birthdate: 1/2/1987 | A message showed bellow Email input with content: “This email has already existed. Please try another” | Untested | 1-12-2014 | NhutHTM |
| Module Login | | | | | | | |
| [Guest-4] | Test login success | a User with username=[alice@yahoo.com](mailto:alice@yahoo.com) and password=”Password1” is created in database | 1.Fill login form with username=”alice@yahoo.com” and password=”Password1”  2. Click “Sign in” button | Forward to user home page | Untested | 1-12-1987 | NhutHTM |
| [Guest-5] | Test wrong username and password | database has no user with username=[alice@yahoo.com](mailto:alice@yahoo.com) and password=”Password1” | 1.Fill login form with username=”alice@yahoo.com” and password=”Password1”  2. Click “Sign in” button | Show Message: “Wrong Username or Password” | Untested | 1-12-1987 | NhutHTM |

### LOGGED IN USER – FAMILY FUNCTIONS

### LOGGED IN USER – GENERAL POST (STORY/ALBUM) FUNCTIONS

### LOGGED IN USER – SOCIAL NETWORK FUNCTIONS

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Pre-Conditions** | **Test Case Procedure** | **Expected Output** | **Result** | **Test Date** | **Note** |
| Register | | | | | | | |
|  |  |  |  |  |  |  |  |

## *CHECKLISTS*

### CHECKLIST OF VALIDATION

|  |  |  |
| --- | --- | --- |
| **No.** | **Question** | **Yes/No** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table 2: Checklist of validation

### SUBMISSION CHECKLIST

N/A

## *OTHER MATERIAL*

### TEST RESULT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module** | **Pass** | **Fail** | **Untested** | **Number of test cases** |
| Total | 73 | 0 | 0 | 73 |
| Guest – Register | 3 | 0 | 0 | 3 |

Table 3: Test result

### INCIDENT REPORT

N/A