**COURSE PROJECT**

**Instructor:** Truong Ninh Thuan

**Project team:**

* Tran Quang Cuong Student.No: 13020049 K58CA
* Nguyen Thanh Trung Student.No: 13020463 K58CA
* Nguyen Vu Hoang Vuong Student.No: 13020763 K58CA

**Project requirement:** Analyze and design a documentation for any software project.

Table of contents

1. Social Network Requirements
   1. Problem statement
   2. Glossary
   3. Supplementary Specification
   4. Use-Case Model
      1. Registration
      2. Login
      3. Update User Profile
      4. Friend Manager
      5. Posting
      6. View Post
      7. Edit/Delete Post
      8. Like Post
      9. Comment
      10. View Notifications
      11. Chatting
2. Social Network Analysis
   1. Architectural Analysis
   2. Use-case analysis
      1. Use-case Realization Interaction Diagrams
         1. Registration
         2. Login
         3. Update User Profile
         4. Friend Manager
         5. Posting
         6. View Post
         7. Edit/Delete Post
         8. Like Post
         9. Comment
         10. View Notifications
         11. Chatting
      2. Use-case Realization View of Participating Classes (VOPCs)
         1. Registration
         2. Login
         3. Update User Profile
         4. Friend Manager
         5. Posting
         6. View Post
         7. Edit/Delete Post
         8. Like Post
         9. Comment
         10. View Notifications
         11. Chatting
3. Social Network Design
   1. Identify Design Elements
      1. Subsystem Context Diagram
         1. Maintain Catalog Subsystem
         2. Notification Subsystem
         3. Chat Subsystem
      2. Analysis-Class-to-Design-Element-Map
      3. Design-Element-to-Owning-Package Map
      4. Architectural Components and Their Dependencies
      5. Packages and Their Dependencies
   2. Describe the Run-time Architect
   3. Describe Distribution
   4. Use-case design
   5. Subsystem Design
   6. Class Design
      1. Describe each class or interface
         1. Item
         2. ICatalog
         3. User
         4. UserCatalog
         5. Post
         6. PostCatalog
         7. Comment
         8. CommentCatalog
         9. UserRelation
         10. UserRelationCatalog
         11. Notification
         12. FriendRequest
         13. LikeNotification
         14. CommentNotification
         15. INotificationCatalog
         16. NotificationCatalog
         17. RegistrationForm
         18. LoginForm
         19. UpdateUserProfileForm
         20. AddFriendForm
         21. AcceptFriendRequestForm
         22. UnfriendForm
         23. BlockUserForm
         24. PostViewer
         25. EditPostForm
         26. DeletePostForm
         27. LikePostForm
         28. CommentForm
         29. NotificationViewer
         30. Message
         31. IChatWindow
         32. ChatWindow
      2. Class diagram in total

SECTION 1: Social Network Requirements

**1.1 Problem statement**

Online social networking nowadays is extremely popular, attracts many users of all age, especially young ones.

Social network is the mapping and measuring of relationships and flows between people, groups, organizations, computers, URLs, and other connected information/knowledge entities. The nodes in the network are the people and groups while the links show relationships or flows between the nodes. Social network provides both a visual and a mathematical analysis of human relationships.

Social networking sites are not only for you to communicate or interact with other people globally but, this is also one effective way for business promotion. A lot of business minded people these days are now doing business online and use these social networking sites to respond to customer queries. It isn't just a social media site used to socialize with your friends but also, represents a huge pool of information from day to day living.

We are tasked with to developing a Social Network Website project where people can make friends as well as stay connected and interact with each other even if they are distance apart.

**1.2 GLOSSARY**

* ***Server***:

Where receives requests and send responses requests.

* ***User***:

A person who use our social network.

* ***Post***:

A Wall Post object works very similar to a Message object. The only difference is that the Wall Post objects are viewable to all friends of the user. Another difference is that a Wall Post object can hold comments. If a user adds a wall post to his or her own account, their status will change to the new wall post.

* ***Comment***:

A user’s writing that be under / following the post – felling about the post.

* ***Wall***:

A user’s home page - The wall is a place where the account owner can express his current feeling or thoughts with a wall status. Other users can add their thoughts by creating wall posts that show up below the status. It is basically a way for many users to communicate publicly or simply leave their thoughts about each other.

* ***Chatting***:

One of the advantages of having friends is the ability to have live communications with them via chat. When a friend is online, he/she will appear in the chat tab as an available friend to chat with. To start a chat session, simply click on the name of the friend which will begin a new chat session. The little bar at the bottom will change from chat disabled to chatting with friend name as seen in the ﬁgure below. Also, the friend receiving the chat will get a notiﬁcation that a new chat session has started. Then the two friends can chat with each other until one or both log out.

* ***Friend*** ***list***:

A management of friends. It contains all user's friends.

* ***Privacy***:

All about privacy rules of User

**1.1 SUPPLEMENTARY SPECIFICATION**

**Objectives**

The purpose of this document is to define requirements of the Social Networking Website. This Supplementary Specification lists the requirements that are not readily captured in the use cases of the use-case model. The Supplementary Specifications and the use-case model together capture a complete set of requirements on the system.

**Scope**

This Supplementary Specifications applies to the Social Networking Website, which will be developed by the ooad students.

This specification defines the non-functional requirements of the system; such as reliability, usability, performance, and supportability, as well as functional requirements that are common across a number of use cases. (The functional requirements are defined in the Use Case Specifications.)

**References**

None.

**Functionality**

The social network to be developed is a free website where people can access via a computer or a smartphone with Internet connection.

* Multiple user can access their homepage at the same time.
* Anyone can register an account. An account has a profile which contains provided user’s information. Some are private, such as email, mobile number; password, and the rest are public to other users. Changing profile is available.
* Users can share their story, their thought by writing a post.
* Only your friends can see what you have written, vote (like/dislike) and make comments. Posts are displayed in which we call a “wall” by newest post to oldest post. When one user goes to another user’s wall, that user’s wall should be appeared.
* Users can make friend easily by click Add friend to send request to their friends. Visit friends wall, chatting.
* Users can chat with each other by a chat window. The chat window keep conversations include old ones between users. When a message came, users must be received a notification about it.

**Usability**

* The website has user-interface of html5, css (bootstrap) which user can easily to understand see.

**Reliability**

* The system will be available 24 hours a day and 7 days a week, with no more than 10% down time.

**Performance**

* The system shall support up to thousands users against the central database at any time.
* The system should provide access to the user’s database with no more than a 10 second latency.
* The system must be able to complete 80% of all transactions within 2 minutes.

**Supportability**

None.

**Security**

* The system must prevent user to do something wrong with the Privacy rules.

**Design Constraints**

* The system must have Web interface in stage 1 (and maybe have Windows app in stage 2).
* System interface must be good-looking, visually, easy to understand it’s features.

**1.4 USE-CASE MODEL**

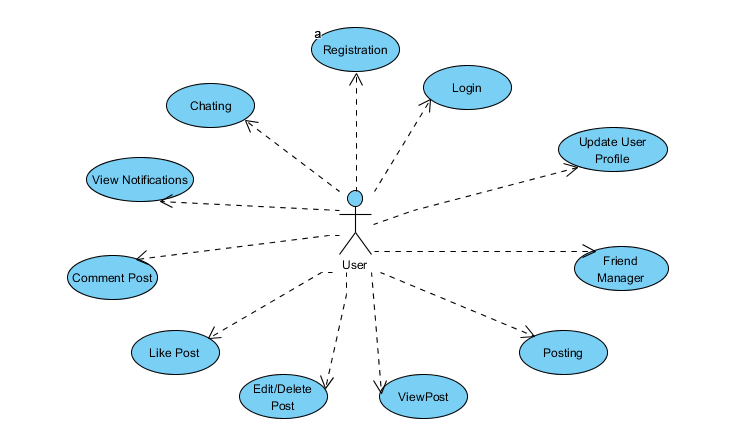
****

Figure 1 Use-case Model

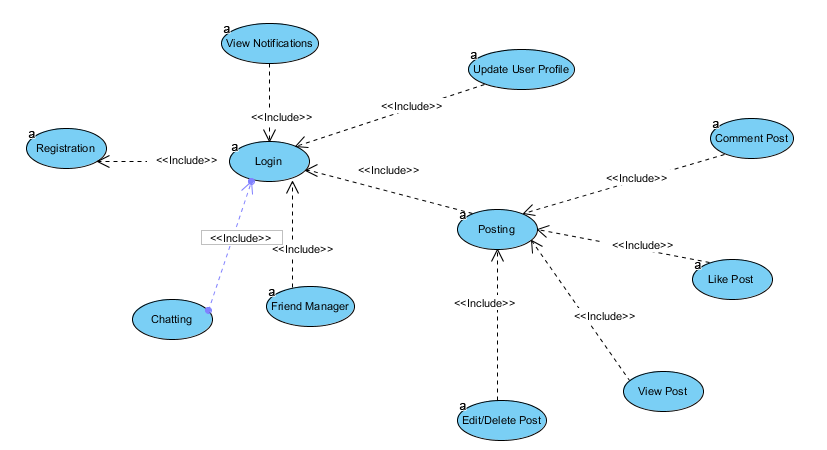
****

Figure 2- The use-case dependencies

**1.4.1 Registration**

**Brief Description**

This use case describes how a user register an account of the Social Networking Website.

**Flow of Events**

*Basic Flow*

This use case starts when the actor wishes to register an account of the SOCIAL NETWORK WEBSITE.

1. The actor clicks on Register button.
2. The system response a webpage that include a form for actor. Actor need to full fill the form and click Submit button.
3. The system validates the entered name and password and other detail information that the actor had filled the form.
4. If success (all the information that actor provide to system are valid) the actor’s account should be stored in system database and redirect actor to Login page
5. If not success (one of these information is invalid, the system will return registration page with highlight which wrong or invalid information)

*Alternative Flows*

**Invalid provided Information**

If, in the Basic Flow, the actor enters an invalid name and/or password/ other information, the system displays an error message. The actor can choose to either return to the beginning of the Basic Flow or cancel the registration, at which point the use case ends.

If one of these provided information is invalid, the system will return registration page with highlight which wrong or invalid information.

**Special Requirements**

None.

**Pre-conditions**

The system is in the registration state and has the registration screen displayed.

**Post-conditions**

If the use case was successful, the actor’s account will be stored at system database. If not, the system state is unchanged.

**Extension Points**

None.

**1.4.2 Login**

**Brief Description**

This use case describes how a user logs into the Social Networking Website.

**Flow of Events**

*Basic Flow*

This use case starts when the actor wishes to log into the SOCIAL NETWORK WEBSITE.

1. The actor enters his/her name and password.
2. The system validates the entered name and password and logs the actor into the system.

*Alternative Flows*

**Invalid Name/ Password**

If, in the Basic Flow, the actor enters an invalid name and/or password, the system displays an error message. The actor can choose to either return to the beginning of the Basic Flow or cancel the login, at which point the use case ends.

If actor enter her/his name or password that wrong over 5 times, server will response a message that “Only can login after 15 minutes later.”

**Special Requirements**

None.

**Pre-conditions**

The system is in the login state and has the login screen displayed.

**Post-conditions**

If the use case was successful, the actor is now logged into the system. If not, the system state is unchanged.

**Extension Points**

None.

**1.4.3 Update User Profile**

**Brief Description**

This use case describes how a user Update their profile for an account of the Social Networking Website.

**Flow of Events**

*Basic Flow*

This use case starts when the actor wishes to Update Profile for an account of the SOCIAL NETWORK WEBSITE.

1. The actor clicks on Edit Profile button.
2. The system response a webpage that include a form for actor. Actor need to full fill the form or edit the information that have in form and click Submit button.
3. The system requires account password to makes any changes to logged account.
4. Actor should enter password correctly.
5. The system validates the entered name and password and other detail information that the actor had filled the form.
6. If success (all the information that actor provide to system are valid) the actor’s account should be update in system database and redirect actor to Login page
7. If not success (one of these information is invalid, the system will return Edit Profile Page with highlight which wrong or invalid information)

*Alternative Flows*

**Invalid provided Information**

If, in the Basic Flow, the actor enters an invalid name and/or password/ other information, the system displays an error message. The actor can choose to either return to the beginning of the Basic Flow or cancel the Updating profile, at which point the use case ends.

If success (all the information that actor provide to system are valid) the actor’s account should be update in system database and redirect actor to Login page

If not success (one of these information is invalid, the system will return Edit Profile Page with highlight which wrong or invalid information)

**Special Requirements**

None.

**Pre-conditions**

The system is in the logged state and has the Wall screen displayed.

**Post-conditions**

If the use case was successful, the actor’s account will be updated at system database. If not, the system state is unchanged.

**Extension Points**

None.

**1.4.4 Friend Manager**

**Brief Description**

This use case describes how a user manager friend list on the Social Networking Website.

**Flow of Events**

*Basic Flow*

This use case starts when the actor wishes to **add / delete / block** another SNW user on the SOCIAL NETWORK WEBSITE.

1. Add friend:
   * 1. Actor X click “Add friend” button on the other wall or on the pop up when hover mouse other actor avatar.
     2. System send “Add friend request” to other actor and response to actor X that “Friend request sent”. Then update the database.
2. Delete friend:
   * 1. Actor X click “Delete friend” button on the other (Y) wall or on the pop up when hover mouse other actor avatar.
     2. System makes a warning that “Do you really want to delete this friend?” asked for “Yes” or “No”.
     3. If actor select Yes, System immediately delete Y from friend list of X then update the database.
3. Block friend:
   * 1. Actor X click “Block” button on the other (Y) profile.
     2. System update database a constraint that X and Y are hidden with each other.

*Alternative Flows*

**Add friend**

If, in the Basic Flow, “Add friend request” not be accept from receiver two actor will not be friend.

**Delete friend**

If in the Basic Flow, actor select “No”, nothing changes to actor friend list.

**Special Requirements**

None.

**Pre-conditions**

The system is in the logged state and has the wall screen displayed.

**Post-conditions**

If the use case was successful, the actor friend list is updated. If not, the system database is unchanged.

**Extension Points**

None.

**1.4.5 Posting**

**Brief Description**

This use case describes how a user post text on the Social Networking Website.

**Flow of Events**

*Basic Flow*

This use case starts when the actor wants to post a text on the SOCIAL NETWORK WEBSITE.

1. The actor fill the post text-box then click Post.
2. Server receive text, store into database and then send response to the actor that the text is posted.

*Alternative Flows*

**Invalid text format or Network errors**

If the post is valid that does not contains anything which due to the errors or warning of system. Text – post will be store in system database and then show on place where actor post in.

If the post contains invalid format, server will show messages to actor what invalid is.

**Special Requirements**

None.

**Pre-conditions**

The system is in the logged state.

Wall screen displayed.

**Post-conditions**

If the use case was successful, actor can view their post on wall.

**Extension Points**

None.

**1.4.6 View Post**

**Brief Description**

This use case describes how a user view a post on the Social Networking Website.

**Flow of Events**

*Basic Flow*

This use case starts when the actor wants to view a post on the SOCIAL NETWORK WEBSITE.

1. The actor Click on their name or the other user’s name to show their wall. In other way, click on homepage.
2. Server returns a list of post and event on main screen of homepage, wall.

The posts are shown shortly almost.

1. User click on “Read more…” to expand the post to view in detail.

*Alternative Flows*

**Not have permission**

If the posts do not belong to the user that was friend with the actor the posts do not show by default.

**Bad internet**

If the post’s data is too long to load or load slowly, the post should not be view by any user.

**Special Requirements**

**None**

**Pre-conditions**

The system is in the logged state.

**Post-conditions**

If the use case was successful, actor can view their post on wall.

**Extension Points**

None

**1.4.7 Edit/Delete Post**

**Brief Description**

This use case describes how a user edit/delete a post on the Social Networking Website.

**Flow of Events**

*Basic Flow*

This use case starts when the actor wants to EDIT/DELETE a post on the SOCIAL NETWORK WEBSITE.

1. The actor Click on the down arrow on the right-top corner, an option menu is shown.
2. If the actor clicks on “Delete”, that post will be delete immediately.
3. If the actor clicks on “Edit”, an edit form will be show to actor can edit post easily.

*Alternative Flows*

**Not have permission**

If the place that actor posted the post before do not allow actor can edit or delete content, actor could not change the post.

**Special Requirements**

**None**

**Pre-conditions**

The system is in the logged state.

**Post-conditions**

If the use case was successful, actor can view their edited post on wall.

System will save edit times.

**Extension Points**

None

**1.4.8 Like Post**

**Brief Description**

This use case describes how a user LIKE a post on the Social Networking Website.

**Flow of Events**

*Basic Flow*

This use case starts when the actor wants to LIKE a post on the SOCIAL NETWORK WEBSITE.

1. The actor Click on the “Like” button on the botton-left corner.
2. An icon option menu show, pick an icon and done.

*Alternative Flows*

**None**

**Special Requirements**

**None**

**Pre-conditions**

The system is in the logged state.

**Post-conditions**

If the use case was successful, actor can view Unlike button.

System will save like with the post.

**Extension Points**

None

**1.4.9 Comment Post**

**Brief Description**

This use case describes how a user COMMENT on a post on the Social Networking Website.

**Flow of Events**

*Basic Flow*

This use case starts when the actor wants to COMMENT on a post on the SOCIAL NETWORK WEBSITE.

1. The actor Click on the “Comment” button on the right of “Like” button.
2. An input text box show, actor type comment to that box. Click “Send”
3. Server save comments to the database of that post and make it visible.

*Alternative Flows*

**None**

**Special Requirements**

**None**

**Pre-conditions**

The system is in the logged state.

**Post-conditions**

If the use case was successful, actor can view comment that had been posted.

System will save comments with the post.

**Extension Points**

None

**1.4.10 View Notifications**

**Brief Description**

This use case describes how a user view notifications on the Social Networking Website.

**Flow of Events**

*Basic Flow*

This use case starts when the actor wants to view notifications on the SOCIAL NETWORK WEBSITE.

1. The actor Click on the “Notification” on the top right of homepage.
2. Server will show a list of notifications in order from latest to oldest.

*Alternative Flows*

**None**

**Special Requirements**

**None.**

**Pre-conditions**

The system is in the logged state.

**Post-conditions**

If the use case was successful, actor can view all of his/her notifications.

**Extension Points**

None

**1.4.11 Chatting**

**Brief Description**

This use case describes how a user chatting with their friends on the Social Networking Website.

**Flow of Events**

*Basic Flow*

This use case starts when the actor wants to chatting with their friend on the SOCIAL NETWORK WEBSITE.

1. The actor clicks on Name of the friend which will begin a new chat session.
2. The little bar at the bottom will change from chat disabled to chatting with friend which named by Friend's name.
3. An actor type a text into the text box then click Send or Enter. The text will be sent to server.
4. Server receive text, store into database and then send response to the other one who will be receive the messages.
5. The system validates the entered name and password and other detail information that the actor had filled the form.

*Alternative Flows*

**None**

**Special Requirements**

None.

**Pre-conditions**

The system is in the logged state.

Two actor is friend on system. Actors do not block each other.

**Post-conditions**

If the use case was successful, the actor 2 can live chatting with actor 1, system save all messages automatically by default.

**Extension Points**

None.

SECTION 2: Social Network Analysis

**2.1 Architectural Analysis**

**2.1.1 Key Abstractions**

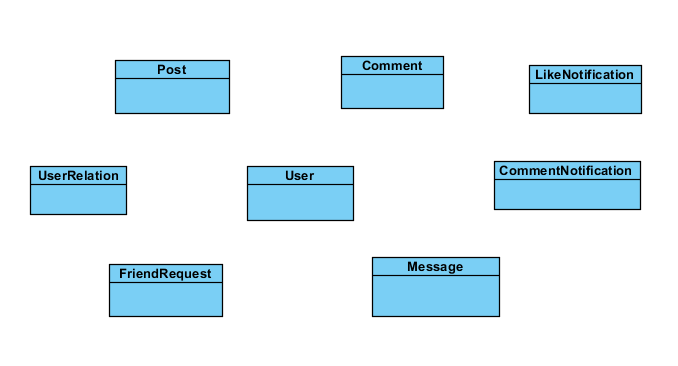


Figure 3 Key Abstractions

**2.1.1.1 Key Abstraction Definitions**

**User**: is an account of the system.

**UserRelation**: is a relation between two users: either friend relation or blocked relation.

**Post**: user write things he/she want to share with his/her friends and store it on a Post object.

**Comment**: User or his/her friends can write a comment of a post of that user

**FriendRequest**: is a notification will be sent to a user. It has two types, a friend request from other users or an accept friend request to the user.

**LikeNotification**: When a friend act like a user’s post, a LikeNotification will be created to notify user.

**CommentNotification**: When a friend write comment on user’s post, a CommentNotification will be created to notify user.

**Message**: is a string represent conversation two users.

**2.1.2 Upper-Level Components and Their Dependencies**

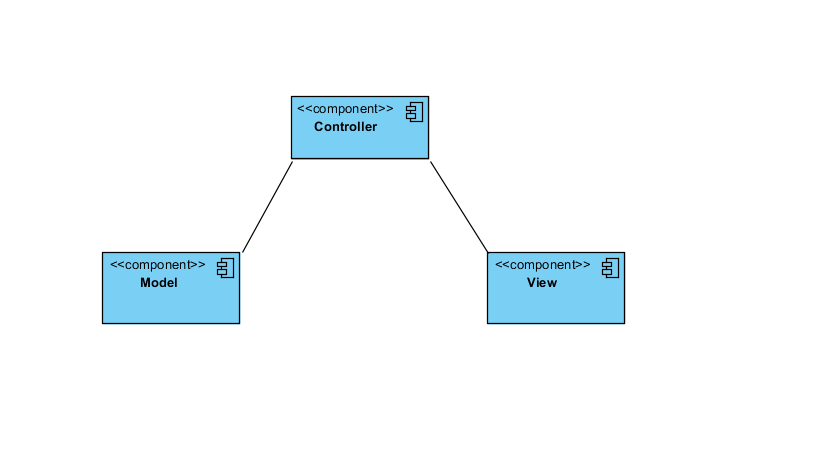


Figure 4 The upper layer architecture

**2.1.2.1 Component Definitions**

**Controller**: Keep navigation tasks received requests from user and call the accordance methods to handle them. For example, this component will receive the request from url and form to manipulate directly with database.

**Model**: This is the component that contains the static data (database), method to access, query and process with it.

**View**: Assume the information display, interaction with users, which contains all the GUI objects such as text boxes, images, e.t.c. Understand a simple way, it is a set of form or HTML files.

**2.2 Use-case Analysis**

**2.2.1 Use-case Realization Interaction Diagrams**

**2.2.1.1 Registration**

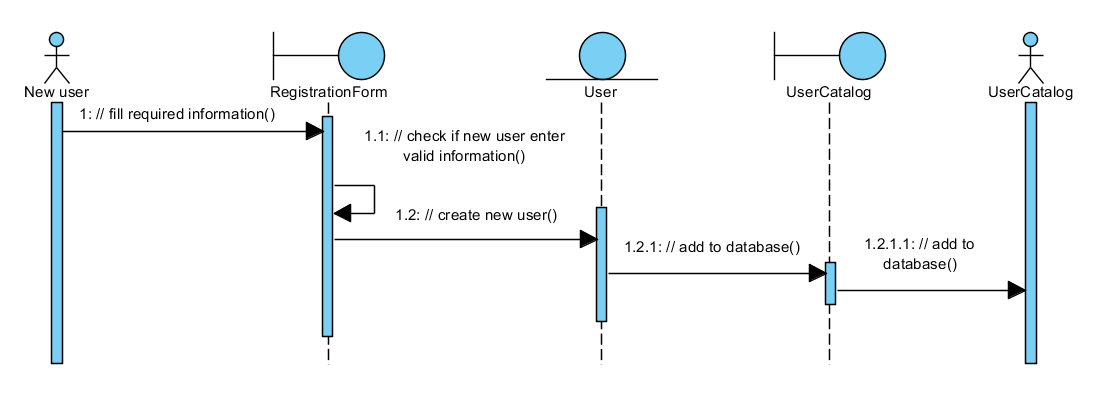


Figure 5 Registration - Basic Flow

**2.2.1.2 Login**

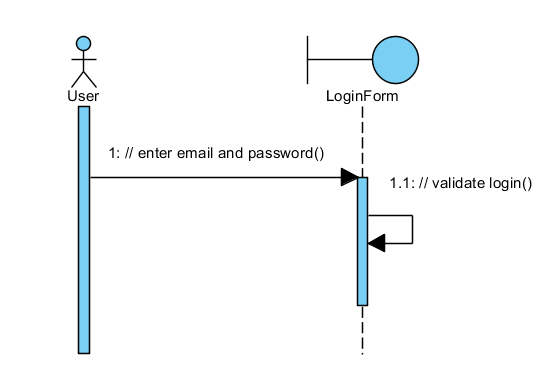


Figure 6 Login - Basic Flow

**2.2.1.3 Update User Profile**

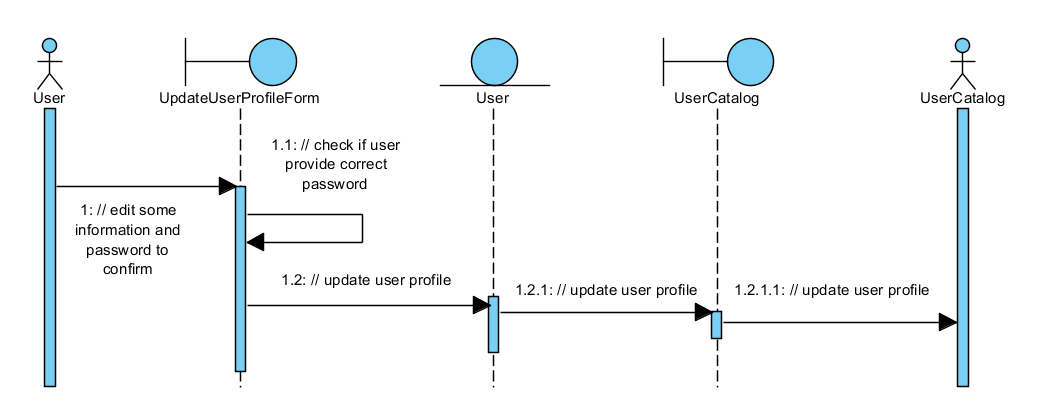


Figure 7 Update User Profile

**2.2.1.4 Friend Manager**

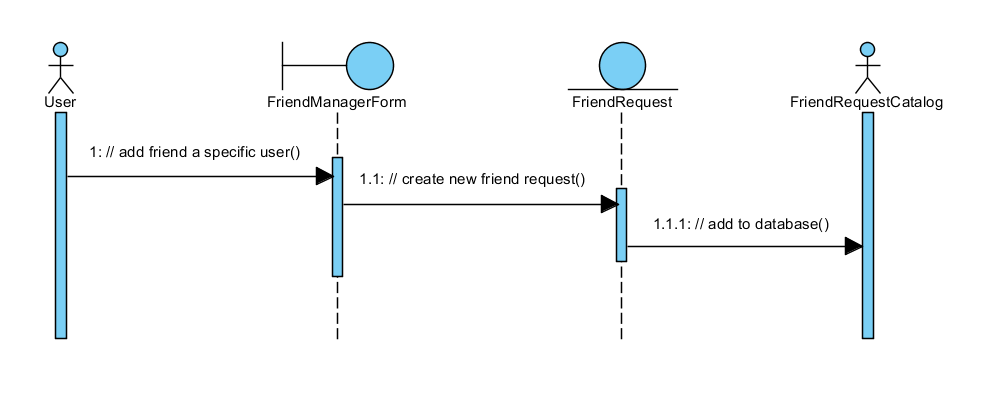


Figure 8 Add Friend - Basic Flow

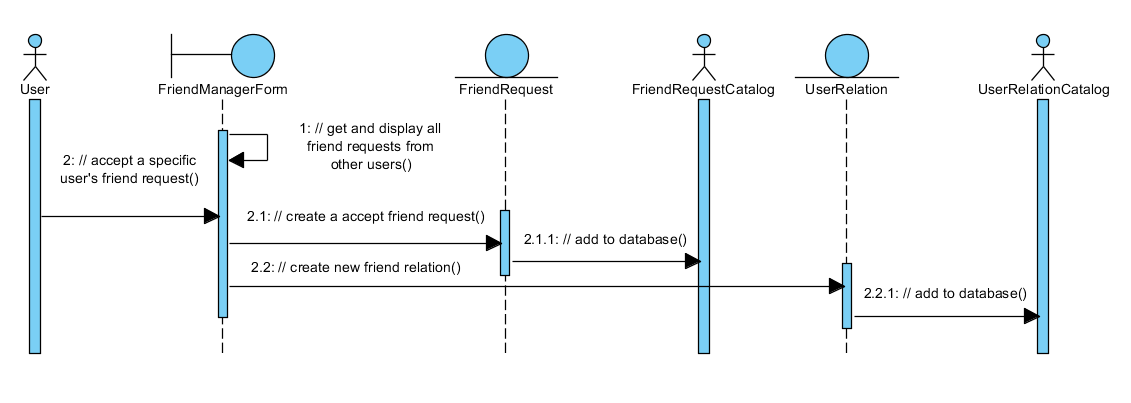


Figure 9 Accept Friend Request - Basic Flow

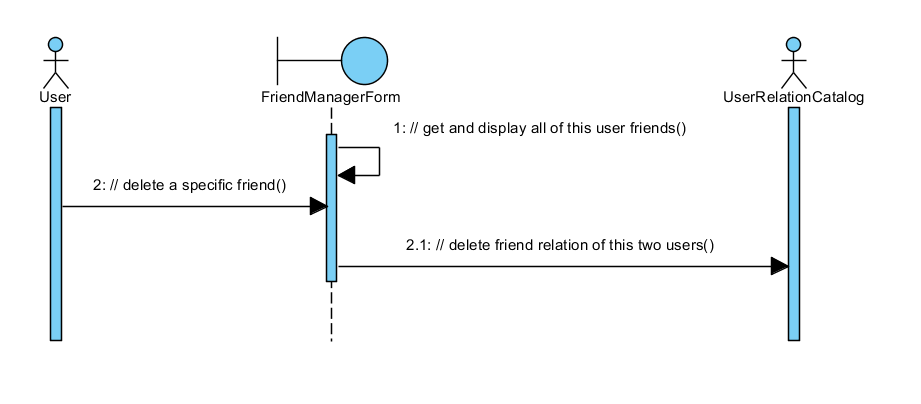


Figure 10 Unfriend – Basic Flow

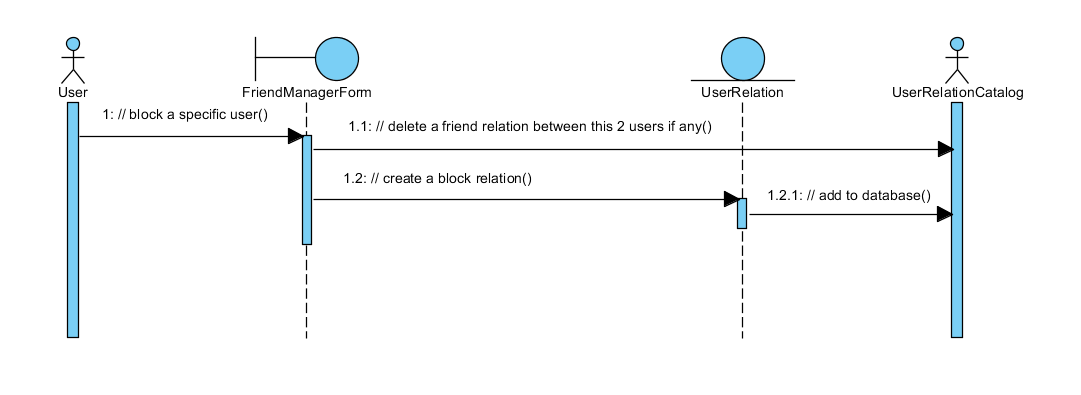


Figure 11 Block a user - Basic Flow

**2.2.1.5 Posting**

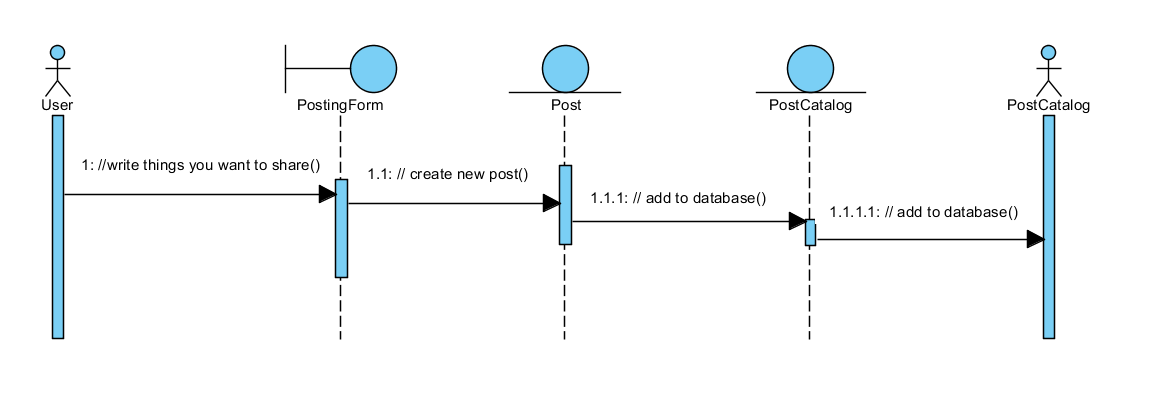


Figure 12 Posting - Basic Flow

**2.2.1.6 View Post**

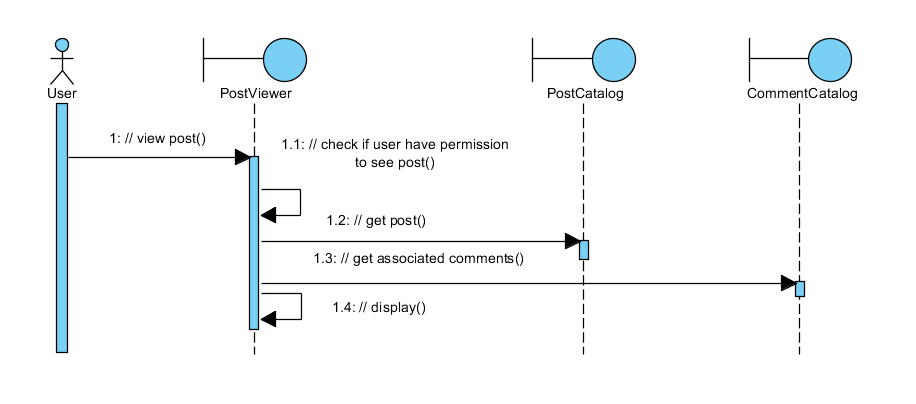


Figure 13 View Post - Basic Flow

**2.2.1.7 Edit/Delete Post**

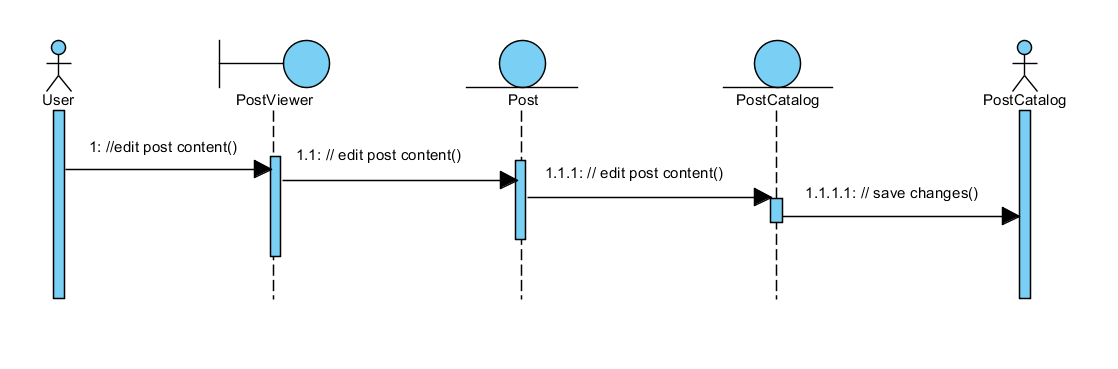


Figure 14 Edit Post - Basic Flow

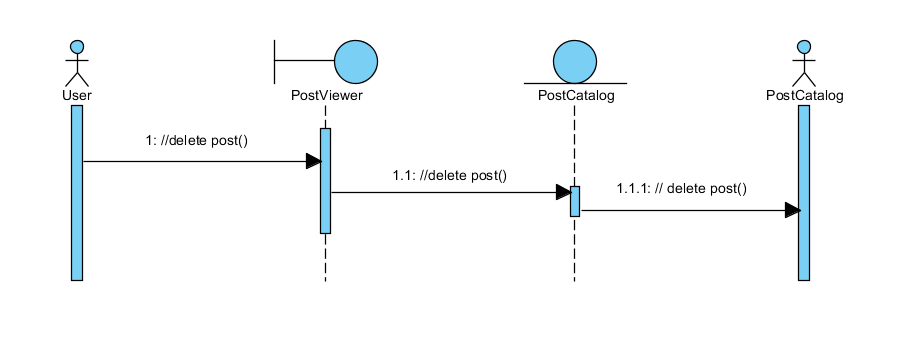


Figure 15 Delete Post - Basic Flow

**2.2.1.8 Like Post**

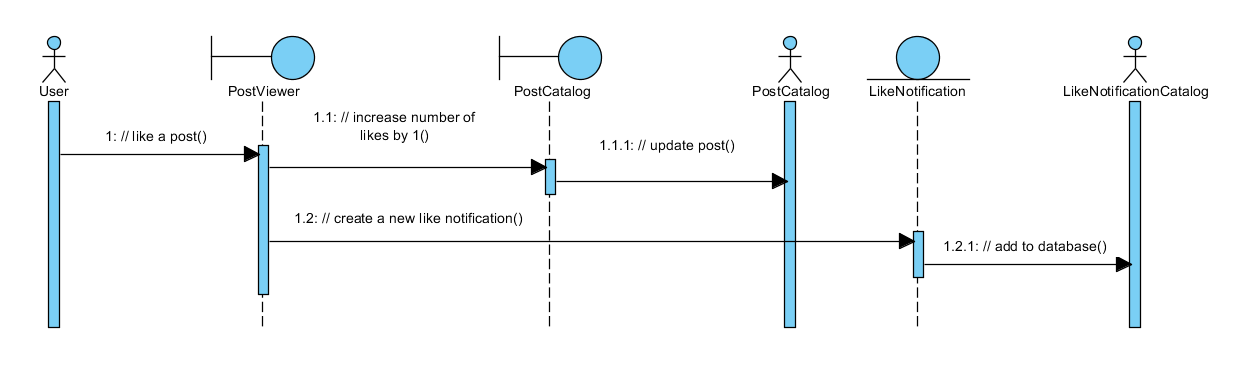


Figure 16 Like Post - Basic Flow

**2.2.1.9 Comment**

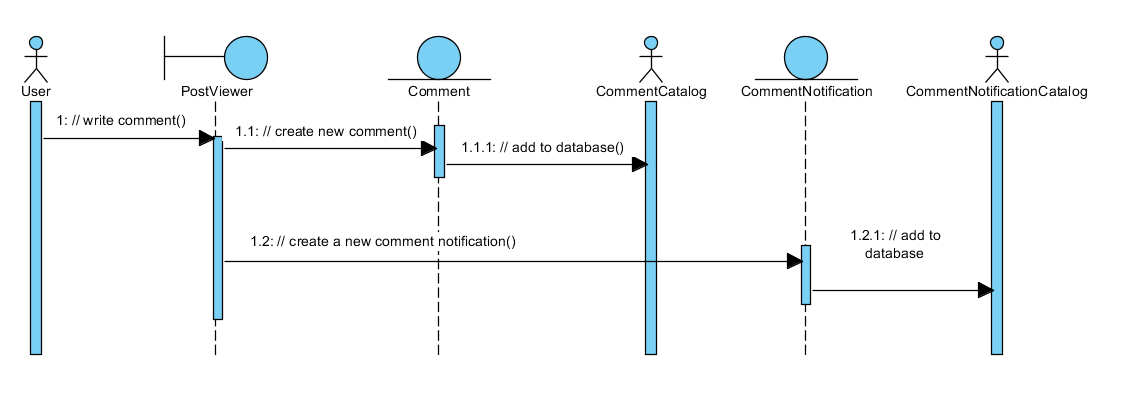


Figure 17 Comment Post - Basic Flow

**2.2.1.10 View Notifications**

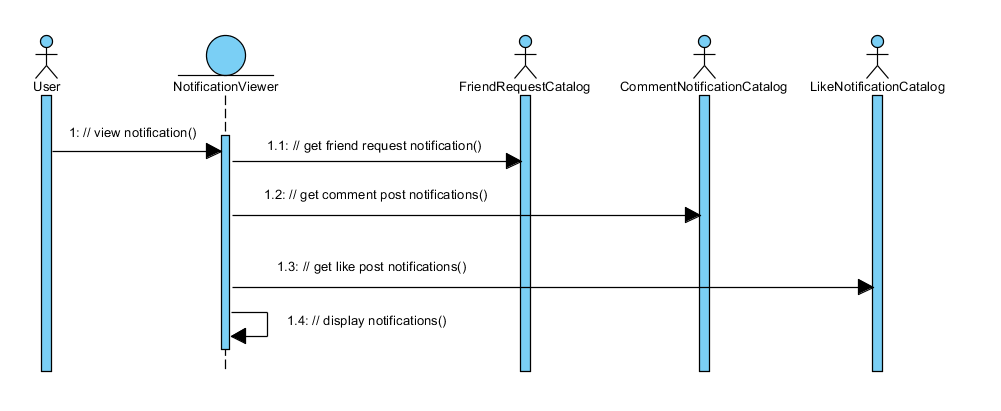


Figure 18 View Notifications - Basic Flow

**2.2.1.11 Chatting**

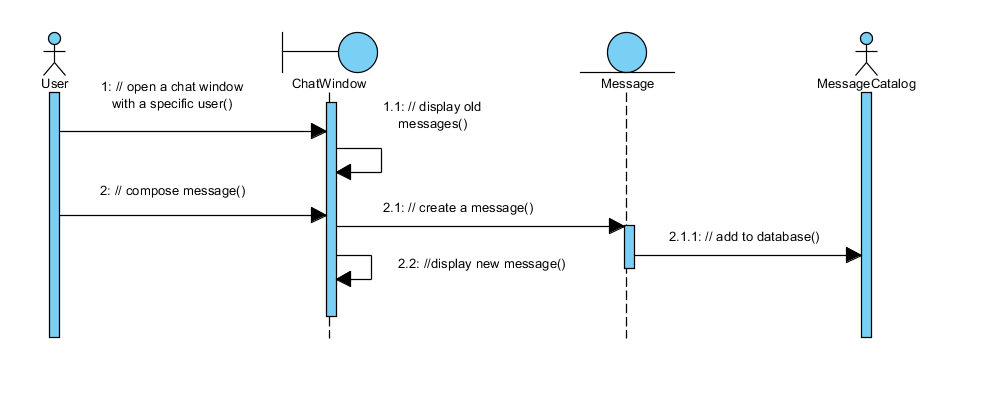


Figure 19 Chatting - Basic Flow

**2.2.2 Use-case Realization View of Participating Class (VOPCs)**

**2.2.2.1 Registration**

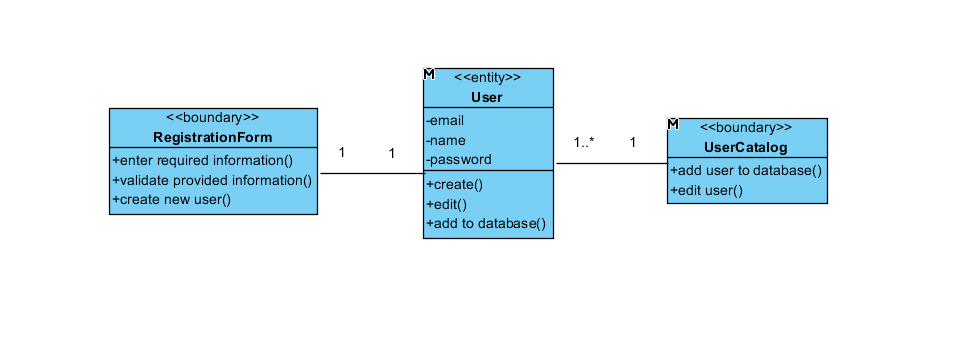


Figure 20 Registration- VOPC

**2.2.2.2 Login**

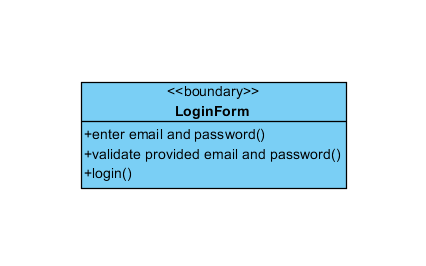


Figure 21 Login - VOPC

**2.2.2.3 Update User Profile**

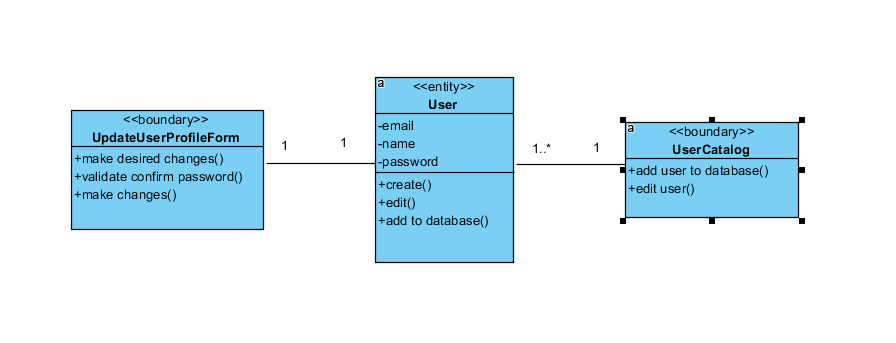


Figure 22 Update User Profile - VOPC

**2.2.2.4 Friend Manager**

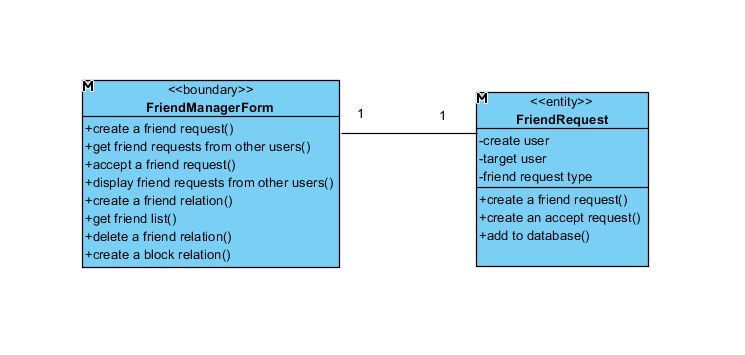


Figure 23 Add Friend - VOPC

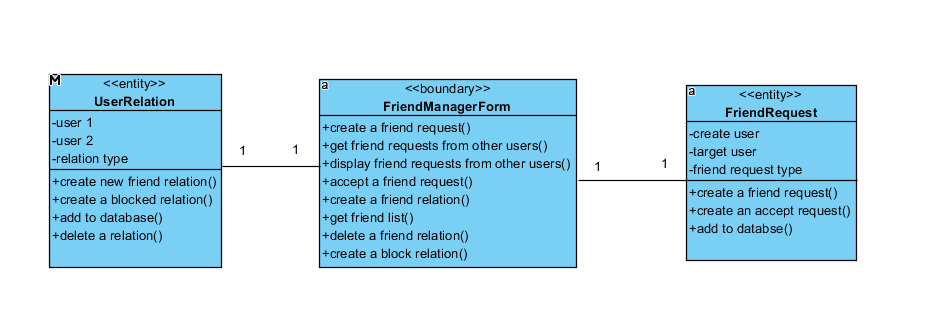


Figure 24 Accept Friend Request - VOPC

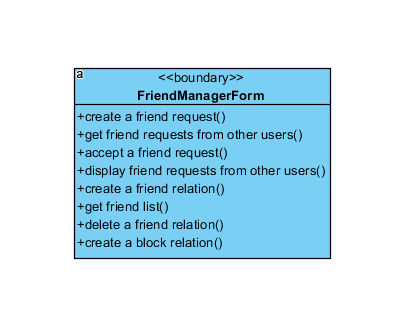


Figure 25 Unfriend - VOPC

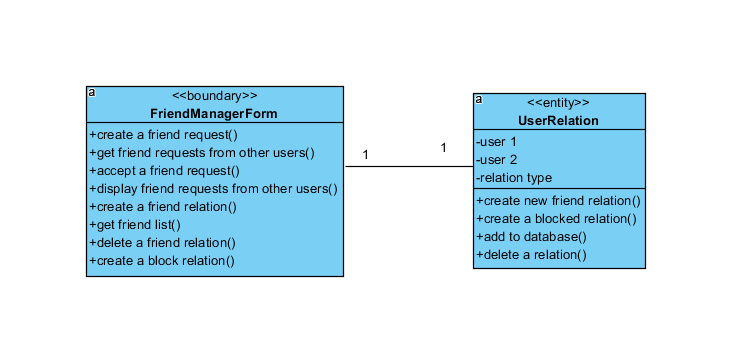


Figure 26 Block a User - VOPC

**2.2.2.5 Posting**

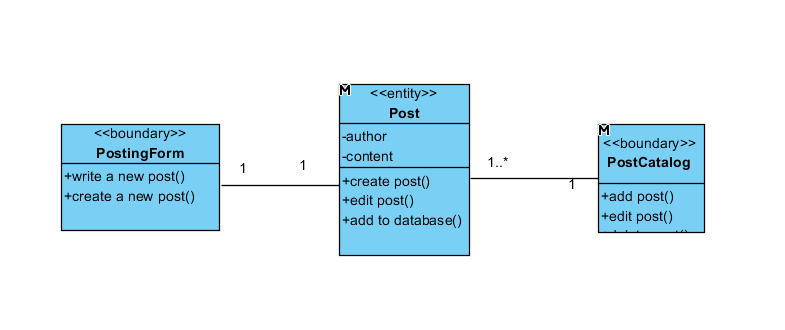


Figure 27 Posting - VOPC

**2.2.2.6 View Post**

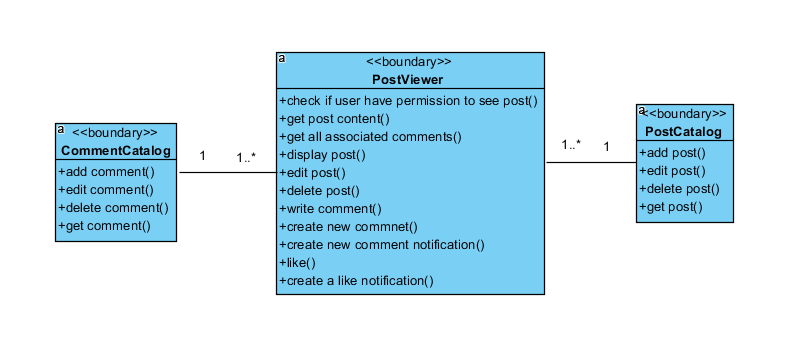


Figure 28 View Post - VOPC

**2.2.2.7 Edit/Delete Post**

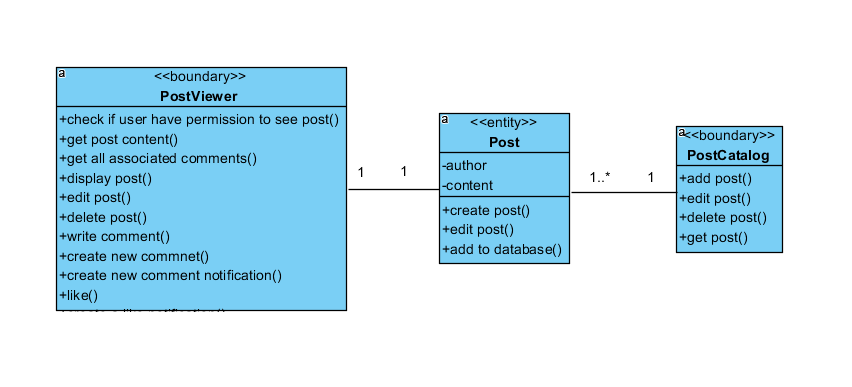


Figure 29 Edit Post - VOPC

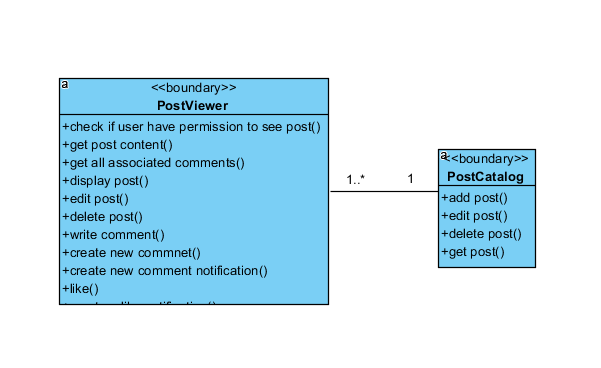


Figure 30 Delete Post - VOPC

**2.2.2.8 Like Post**

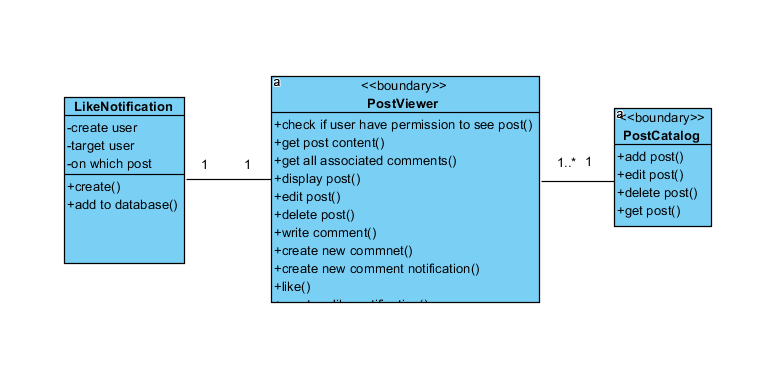


Figure 31 Like Post - VOPC

**2.2.2.9 Comment**

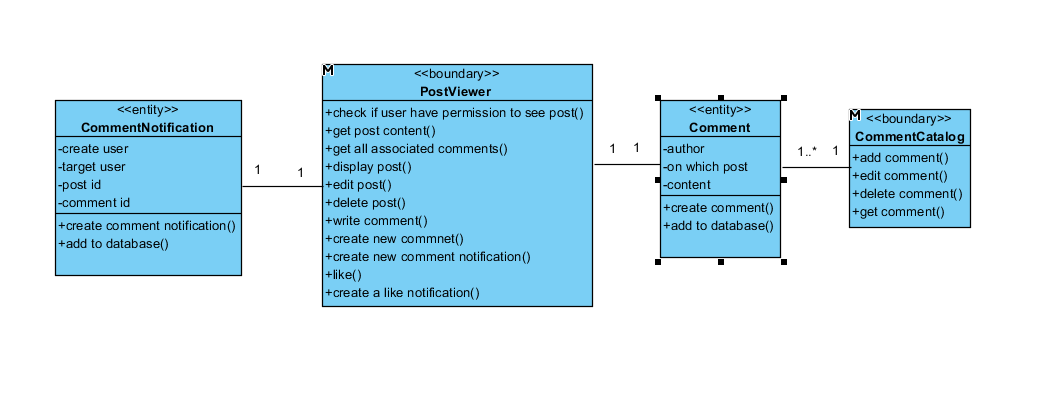


Figure 32 Comment Post - VOPC

**2.2.2.10 View Notifications**

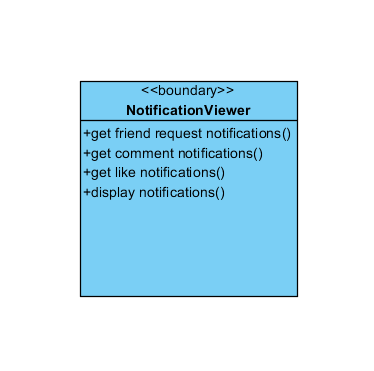


Figure 33 View Notifications - VOPC

**2.2.2.11 Chatting**

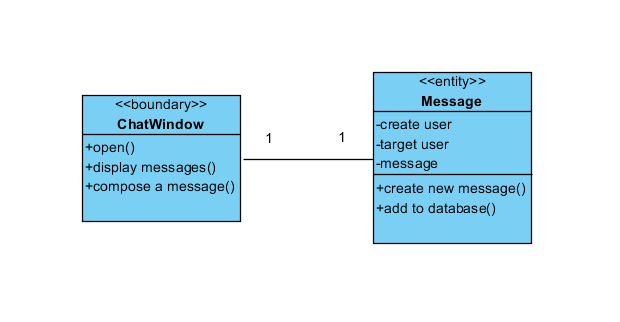


Figure 34 Chatting - VOPC

**2.2.3 Analysis-Class-To-Analysis-Mechanism Map**

|  |  |
| --- | --- |
| **Analysis Class** | **Analysis Mechanism(s)** |
| RegistrationForm | Error Detection/Handling/Reporting |
| User | Security |
| UserCatalog | Persistency, Security |
| LoginForm | Error Detection/Handling/Reporting |
| UpdateUserForm | Error Detection/Handling/Reporting |
| FriendManagerForm | None |
| FriendRequest | Security |
| UserRelation | Security |
| PostingForm | None |
| Post | Security |
| PostCatalog | Persistency, Security |
| PostViewer | None |
| CommentCatalog | Persistency, Security |
| LikeNotification | Security |
| Comment | Security |
| CommentNotification | Security |
| NotificationViewer | None |
| ChatWindow | Security, Persistency |
| Message | Security |

SECTION 3: Social Network Analysis

**3.1 Identify Design Elements**

**3.1.1 Subsystem Context Diagram**

**3.1.1.1 Maintain Catalog Subsystem**

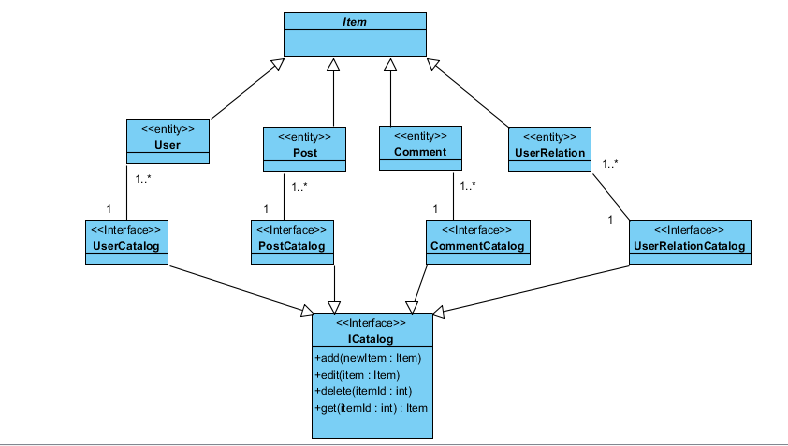


Figure 35 Maintain Catalog Subsystem

**3.1.1.2 Notification Subsystem**

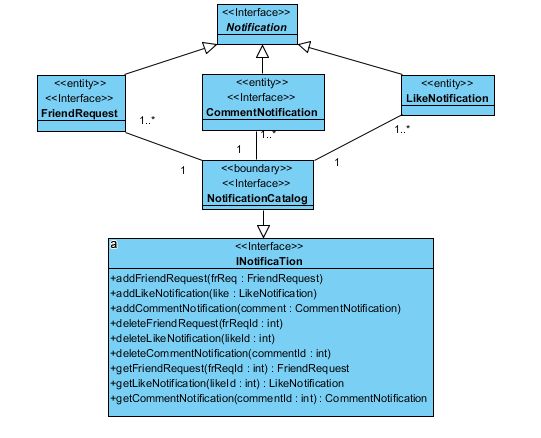


Figure 36 Notification Subsystem

**3.1.1.3 Chat Subsystem**

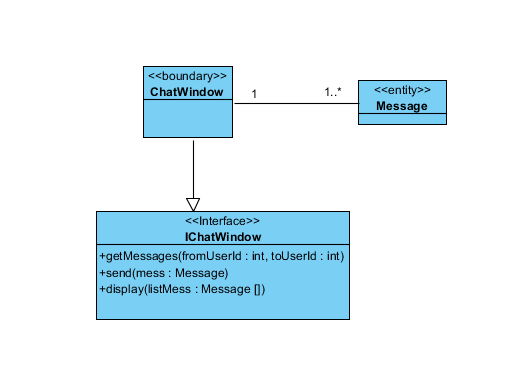


Figure 37 Chat Subsystem

**3.1.2 Analysis-Class-to-Design-Element-Map**

|  |  |
| --- | --- |
| **Analysis Class** | **Design Element** |
| RegistrationForm | RegistrationForm |
| User | User |
| UserCatalog | UserCatalog |
| LoginForm | LoginForm |
| UpdateUserForm | UpdateUserForm |
| FriendManagerForm | AddFriendForm |
|  | AcceptFriendRequestForm |
|  | UnfriendForm |
|  | BlockUserForm |
| FriendRequest | FriendRequest |
| UserRelation | UserRelation |
| PostingForm | PostingForm |
| Post | Post |
| PostCatalog | PostCatalog |
| PostViewer | PostViewer |
|  | EditPostForm |
|  | DeletePostForm |
|  | CommentForm |
|  | LikeForm |
| CommentCatalog | CommentCatalog |
| LikeNotification | LikeNotification |
| Comment | Comment |
| CommentNotification | CommentNotification |
| NotificationViewer | NotificationViewer |
|  | NotificationCatalog |
| ChatWindow | ChatWindow |
| Message | Message |

**3.1.3 Design-Element-to-Owning-Package Map**

|  |  |
| --- | --- |
| Design Elements | “Owning” Package |
| RegistrationForm | Controller::GUI controller |
| LoginForm |
| UpdateUserForm |
| AddFriendForm |
| AcceptFriendRequestForm |
| UnfriendForm |
| BlockUserForm |
| PostingForm |
| PostViewer |
| EditPostForm |
| DeletePostForm |
| CommentForm |
| LikeForm |
| NotificationViewer |
| ICatalog | Controller::Subsystem |
| UserCatalog |
| PostCatalog |
| CommentCatalog |
| UserRelationCatalog |
| INotificationCatalog |
| NotificationCatalog |
| IChatWindow |
| Chatwindow |
| User | Controller::Elements |
| Post |
| Comment |
| UserRelation |
| FriendRequest |
| LikeNotification |
| CommentNotification |
| Message |

**3.1.4 Architectural Components and Their Dependencies**

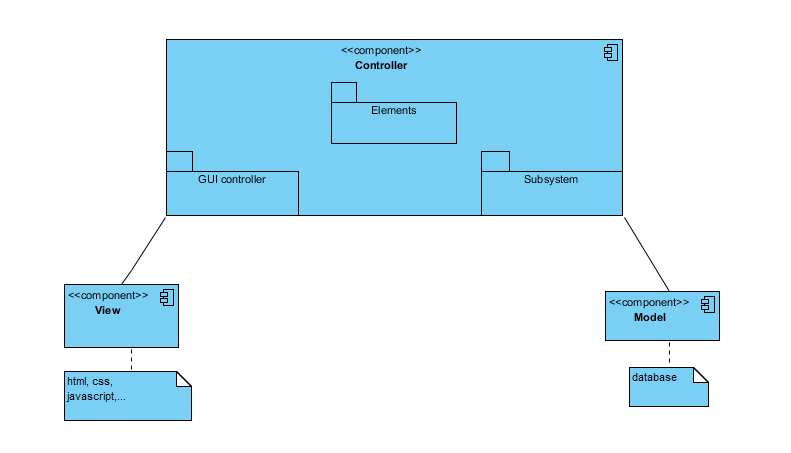


Figure 38 Architecture Components

**3.1.5 Packages and Their Dependencies**

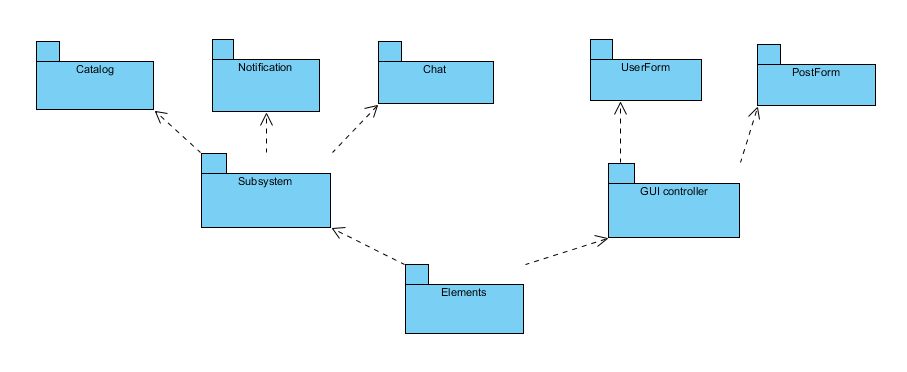


Figure 39 Packages and Their Dependencies

**Catalog**: contains the interface ICatalog, UserCatalog, PostCatalog, CommentCatalog, UserRelationCatalog

**Notification**: contains the interface INotificationCatalog, NotificationCatalog

**Chat**: contains the interface IChatWindow, ChatWindow

**Subsystem**: import Catalog, Notification and Chat packages

**UserForm**: contains RegistrationForm, LoginForm, UpdateUserForm, AddFriendForm, AcceptFriendRequestForm, UnfriendForm, BlockUserForm, NotificationViewer

**PostForm**: contains PostingForm, PostViewer, EditPostForm, DeletePostForm, CommentForm, LikeForm.

**GUI controller**: contains ChatWindow and import UserForm and PostForm

**Elements**: import all above packages and some other elements.

**3.2 Describe the Run-time Architecture**

**3.3 Describe Distribution**

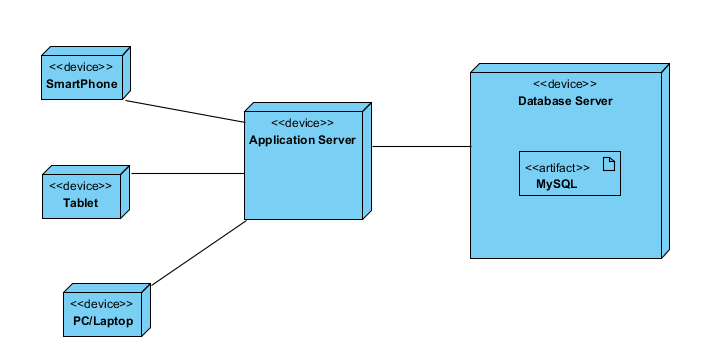
****

Figure 40 Deployment Model

**3.4 Use-case Design**

This part describes the use cases in the consideration of every mechanism that were defined before.

None.

**3.5 Subsystem Design**

None.

**3.6 Class Design**

This part describes all class structure and their description also total class diagram and their dependency.

**3.6.1 Describe each class or interface**

**3.6.1.1 Item**

Just an empty abstract class

**3.6.1.2 ICatalog**

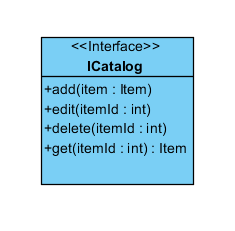


Figure 41 ICatalog

**3.6.1.3 User**

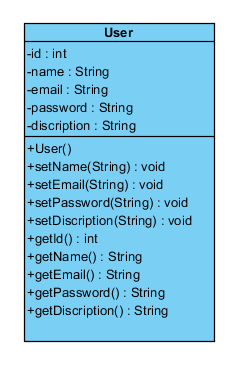


Figure 42 User

**3.6.1.4 UserCatalog**

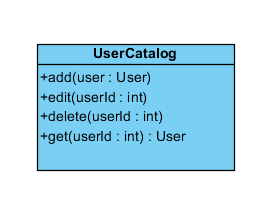


Figure 43 UserCatalog

**3.6.1.5 Post**

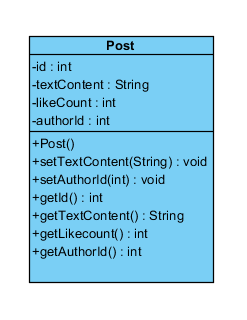


Figure 44 Post

**3.6.1.6 PostCatalog**

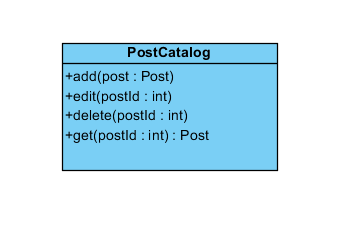


Figure 45 PostCatalog

**3.6.1.7 Comment**

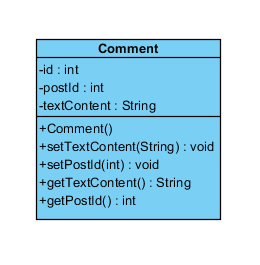


Figure 46 Comment

**3.6.1.8 CommentCatalog**

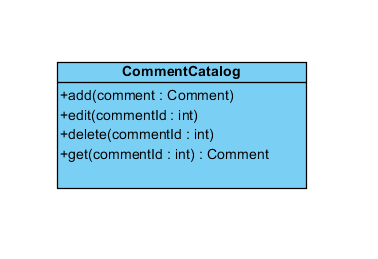


Figure 47 CommentCatalog

**3.6.1.9 UserRelation**

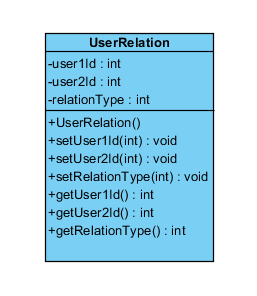


Figure 48 UserRelation

**3.6.1.10 UserRelationCatalog**

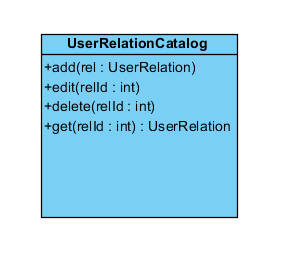


Figure 49 UserRelationCatalog

**3.6.1.11 Notification**

An abstract class

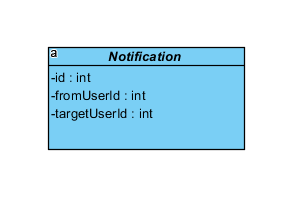


Figure 50 Notification

**3.6.1.12 FriendRequest**

Implement abstract class Notification

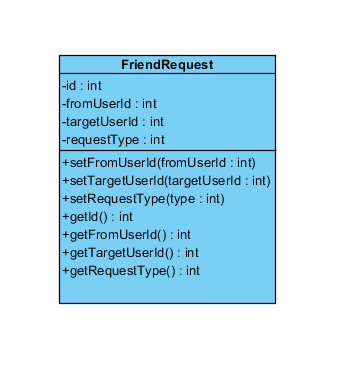


Figure 51 FriendRequest

**3.6.1.13 LikeNotification**

Implement abstract class Notification

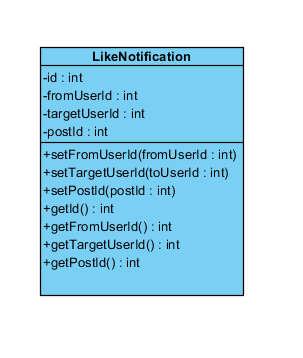


Figure 52 LikeNotification

**3.6.1.14 CommentNotification**

Implement abstract class Notification

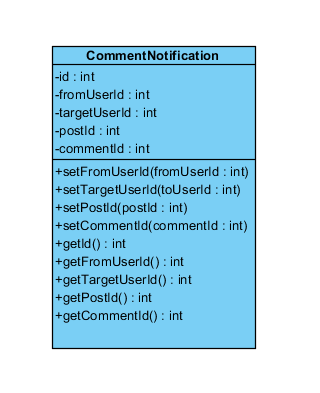


Figure 53 CommentNotification

**3.6.1.15 INotificationCatalog**

An interface

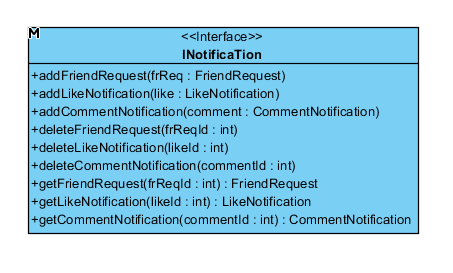


Figure 54 INotification

**3.6.1.16 NotificationCatalog**

Implement interface INotification

**3.6.1.17 RegistrationForm**

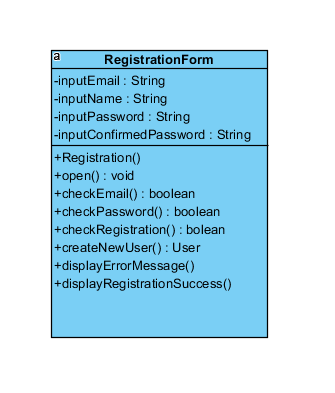


Figure 55 RegistrationForm

**3.6.1.18 LoginForm**

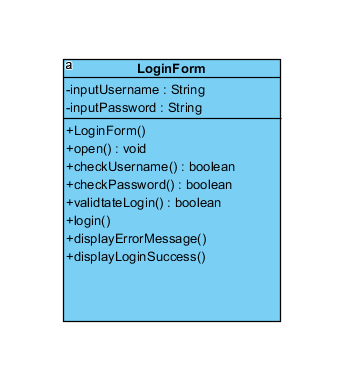


Figure 56 LoginForm

**3.6.1.19 UpdateUserProfileForm**

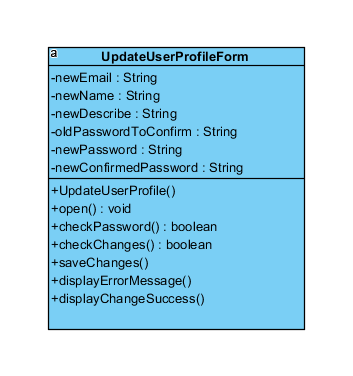


Figure 57 UpdateUserProfileForm

**3.6.1.20 AddFriendForm**

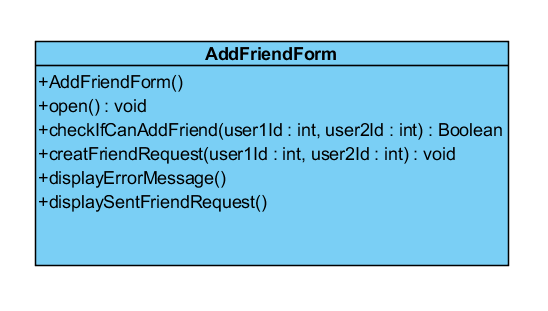


Figure 58 AddFriendForm

**3.6.1.21 AcceptFriendRequestForm**

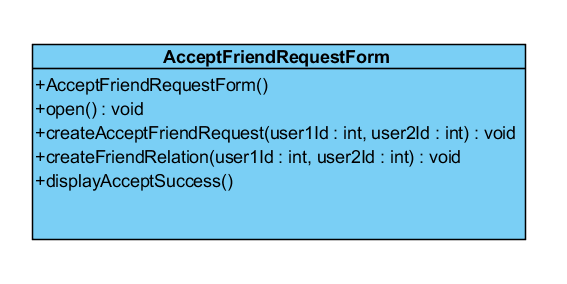


Figure 59 AcceptFriendForm

**3.6.1.22 UnfriendForm**

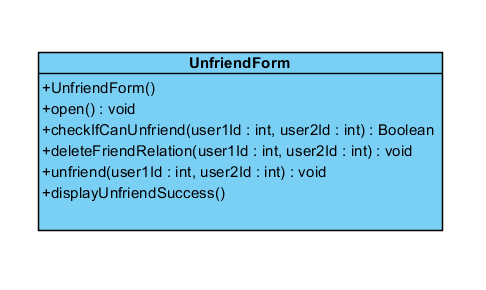


Figure 60 UnfriendForm

**3.6.1.23 BlockUserForm**

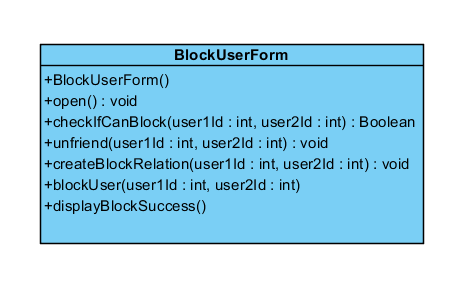


Figure 61 BlockUserForm

**3.6.1.24 PostViewer**

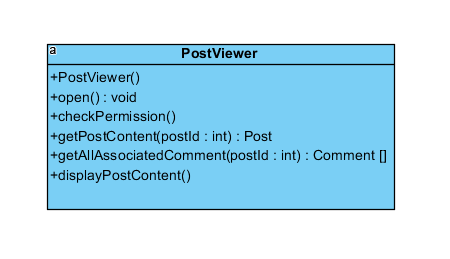


Figure 62 PostViewer

**3.6.1.25 EditPostForm**

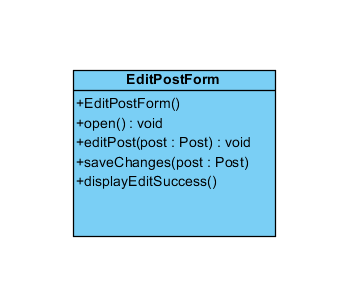


Figure 63 EditPostForm

**3.6.1.26 DeletePostForm**

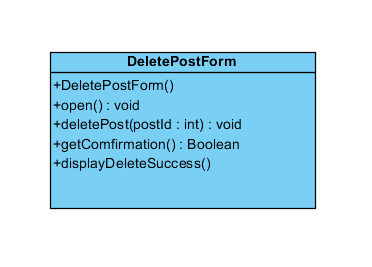


Figure 64 DeletePostForm

**3.6.1.27 LikePostForm**

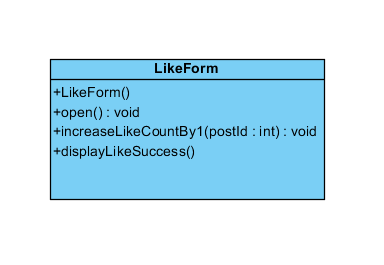


Figure 65 LikeForm

**3.6.1.28 CommentForm**

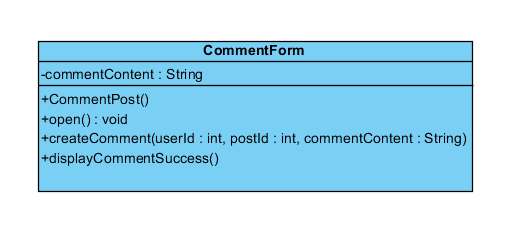


Figure 66 CommentForm

**3.6.1.29 NotificationViewer**

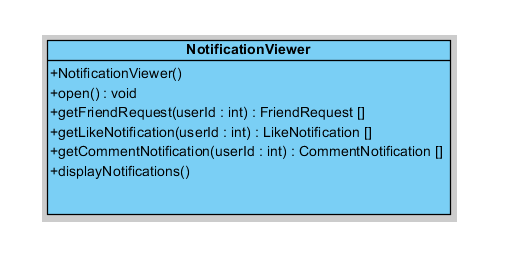


Figure 67 NotificationViewer

**3.6.1.30 Message**

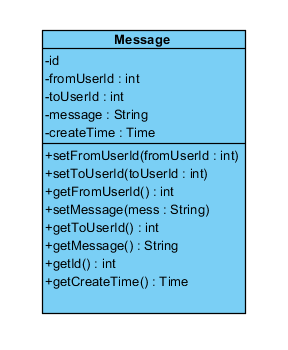


Figure 68 Message

**3.6.1.31 IChatWindow**

An interface



Figure 69 IChatWindow

**3.6.1.32 ChatWindow**

Implement interface IChatWindow

**3.6.2 Class diagram in total**

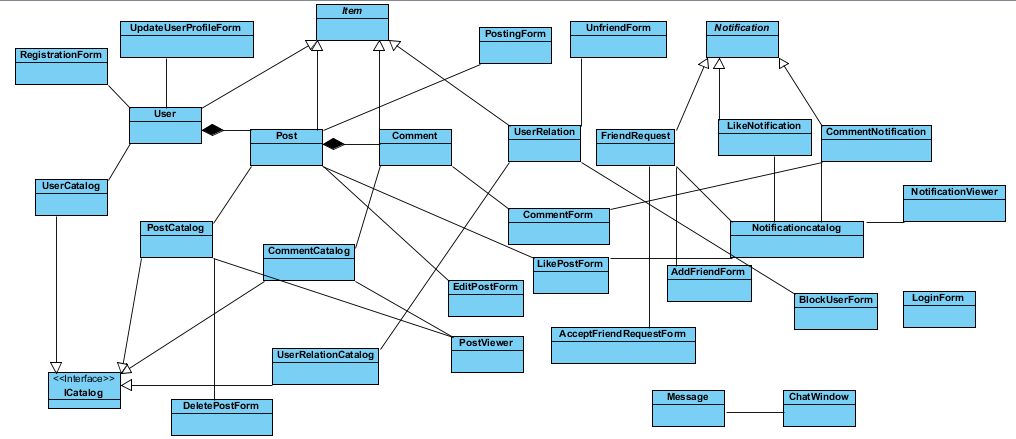
****

Figure Class Diagram in Total