Project Report - Social Media
CS 521 Object-Oriented Design and Analysis

Email

blafon@hawk.iit.edu
apimpalgaonkar@hawk.iit.edu
rseetha1@hawk.iit.edu

Presented By

Lafon Benoit(A20386408)
Pimpalgaonkar Abhinav(A20387324)
Seethamraju Rekha(A20353704)

Requirements

The intent is to build a social media platform where users can post pictures, video or gifs and follow their friends. A user can post some media (picture, video or gif) with a short message. The message can contain hashtags describing what the media is about. The media on the site will be indexed based on the hashtags that can be searched through. A user can follow another user by searching for them using a search bar functionality by their username or their name. Once a user logs in, they can see their news feed which contains posts from the users they are following and recommendations for people they might want to follow. A user can like and comment on posts and can report other posts as inappropriate if he/she feels it is so.

If a user's posts are reported as inappropriate more than 10 times he will be banned.

The administrator investigates inappropriate posts.

A user can
1. Create a new account
   a. Name
   b. Username
   c. Email
   d. Password
   e. Phone no
   f. Gender
   g. Bio
2. Disable account
3. Choose privacy settings
4. Change account
   a. Edit profile
      i. Add, update or delete a personal information
   b. Change password
   c. Authorize apps to access his/her account
   d. Subscribe to news feed in the form email
   e. Hide comments that are
      i. Inappropriate
      ii. Contain words specified(custom keywords)
5. Add, update or delete a post

6. Respond to a post from their friend
   a. Like a post
   b. Comment on a post
   c. Report a post as inappropriate
   d. Share a post
7. Search for hashtags, usernames and names
8. Request an incident report
9. Follow other users

Project - Social Media

An administrator can
1. Check user account
2. Delete a post
3. Delete a user account
4. Create an incident report
5. Handle inappropriate complaints
6. Ban a user
7. Create an incident ticket

A technical support can
1. Cancel or close an incident ticket
2. Schedule incident ticket

Actor Dictionary

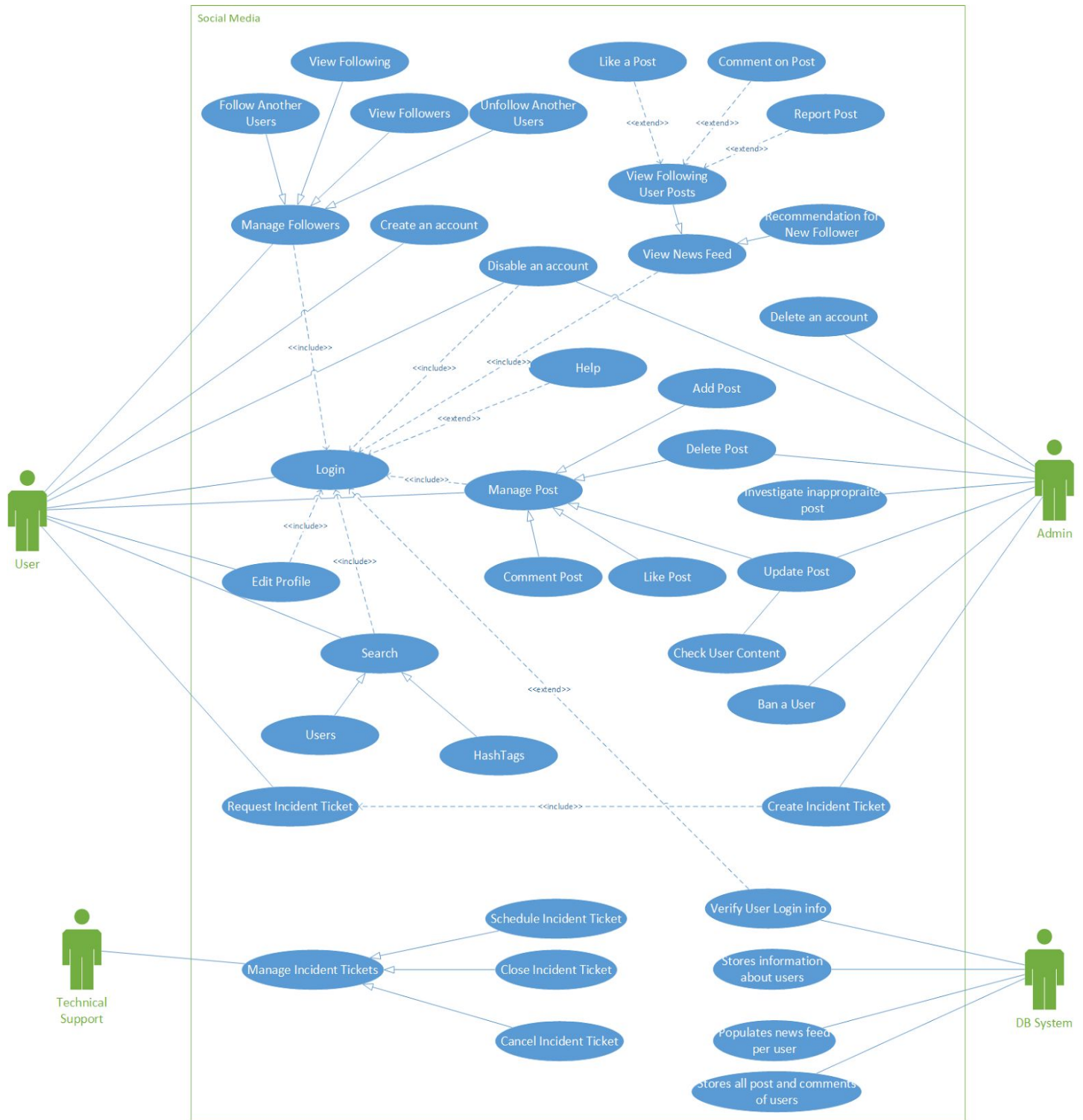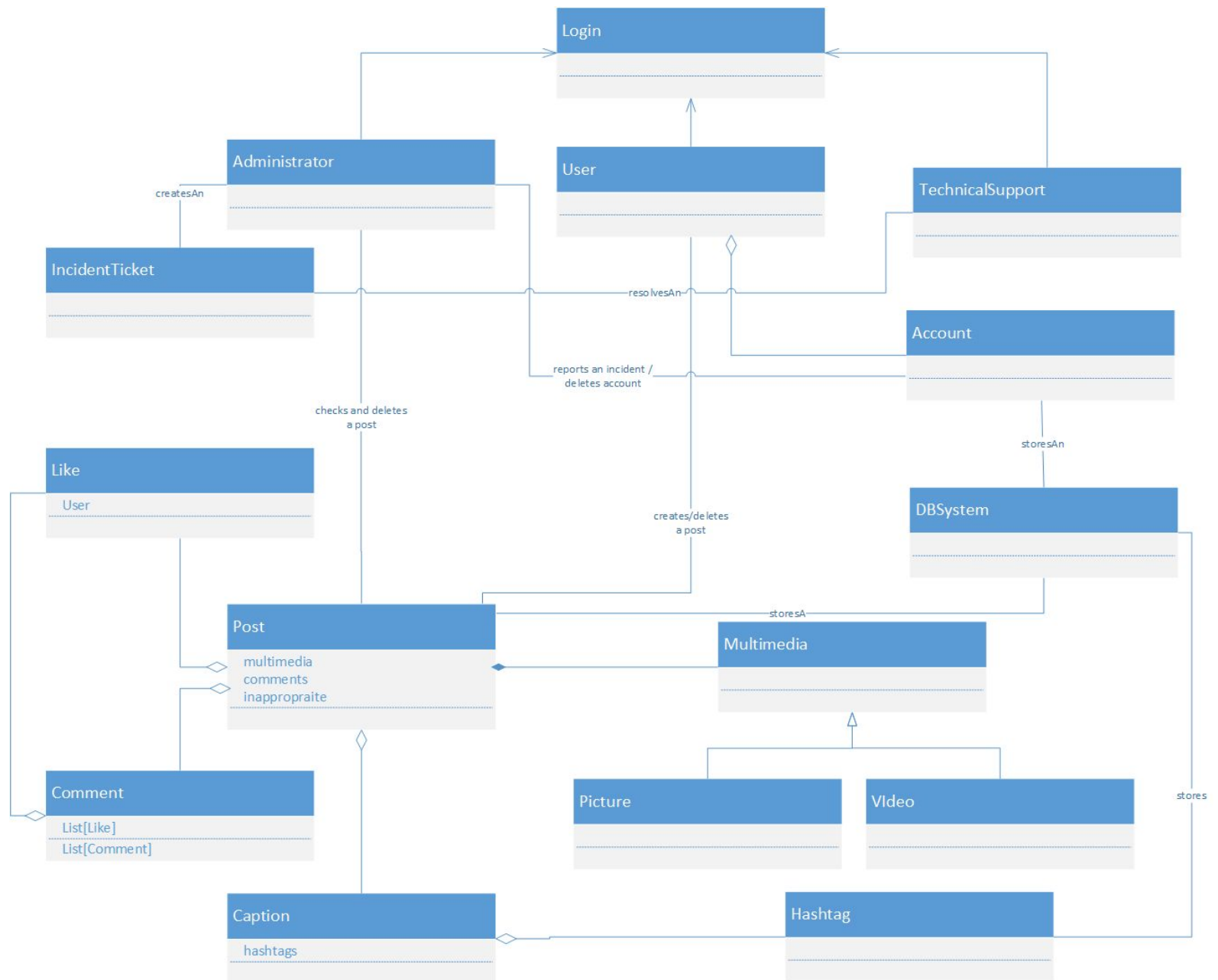| Actor	Description	Use Case(s)
User	Interacts with other users by adding new posts and, commenting and liking other user's posts	Create an account Disable an account Follow a friend Manage post Edit Profile Search for hashtags or people Request an incident ticket
Administrator	Manages user accounts by checking contents of posts reported as inappropriate using some automated tools or report made by other users. Any violation can conduct to removing user post or account or, a ban.	Check user content Delete a user post Delete a user account Ban a user Create an incident report Disable user account Investigate inappropriate post
Technical Support	Handles incident tickets	Schedule incident ticket Close incident ticket Cancel incident ticket
DB system	A database that stores information about the system	Stores information about users Populates a news feed per user Stores all the posts and comments by users Verify User login Info

Domain Dictionary

Name	Type	Description
User	Role	Makes posts, comments
Administrator	Role	Investigates inappropriate post complaints and makes incident reports
Tech Support	Role	Deals with incident tickets
DB Backend System	Role	A database that stores information about the system
Post	Object	A picture, gif or video that a user can share based on his/her privacy settings
Comment	Object	A text message that a user can make on any post that they can access
Privacy Settings	Object	Defines a set of rules that a defines who can see a post that a user makes
Incident ticket	Object	An entity that describes problems encountered by the customer/account specialist
Schedule incident ticket	Function	The act of scheduling a time slot to work on the incident ticket by the tech support
Close incident ticket	Function	The act of closing incident ticket after the problem described by the incident report has been solved by the tech support
Cancel incident ticket	Function	The act of cancelling the incident ticket by the tech support because the incident ticket is no longer valid for whatever reasons
Create incident ticket	Function	The act of creating an incident ticket about a problem faced by the admin/user
hashtags	Object	An entity used in the text part of the post that starts with # followed by text that is used to search for kinds of posts by the user. It is a way of segregating the images based on topics introduced by the user
usernames	Object	A unique identifier that the user chooses as his/her identity while using the product
names	Object	Word or set of words by which a person is addressed
Report a post as inappropriate	Function	Allow a user to signal an inappropriate content for it to be corrected or removed
Like a post	Function	Allow a user to quickly and simply express his/her approval for a given post
Comment on a post	Function	Let users express their opinion by commenting his/her posts or others ones
Hide comments	Process	Act of reducing visibility of a comment to other users by the publisher regardless of the reason

Hide comments that are inappropriate	Function	Allow the publisher of a post to hide a given comment considered inappropriate to other users
Hide comments that contain custom keywords	Function	Allow the publisher of a post to hide a given comment based on customized keywords to other users
Add custom keywords	Function	Allow a publishers to append keywords to their posts
Apps	Object	An application, especially as downloaded by a user to a mobile device
Authorize apps to access user accounts	Function	Ask the user permissions to access his/her activities on the social media platform
Change account	Process	Allow the user to switch between multiple account
Edit profile	Process	A user update his/her personal information
Change password	Function	A user update his/her password
Subscribe to news feed	Function	Allow a user to follow activities of another channel
Add post	Function	A user publish a new content
Delete post	Function	A user remove a content he/she has previously published
Update post	Function	A user update content he/she has previously published
Create a new account	Process	A user provide information to establish his/her new account
Disable account	Function	Prevent other users to access the disabled account anymore
Search	Process	Act of looking up for a given user or posts containing the customized hashtags
Search for hashtags	Function	Allow users to look for posts associated with customized hashtags he/she will provide
Search for usernames	Function	Allow users to look for users having the corresponding username
Search for names	Function	Allow users to look for users having the corresponding name
user account	Object	An established relationship between a user and the social media platform allowing the user to access to services proposed by this latter
Create an incident report	Function	Allow users to signal a feature not working properly
Ban a user	Function	An administrator has the privilege to delete a user account and prevent that user to create a new account using the same email address he/she used for their banned account
Check user account	Function	An administrator can access users personal information even though not visible to public
Delete a user account	Function	An administrator can remove a user account
Check inappropriate post	Function	An administrator can check every post related to a report incident for inappropriate content

Project - Social Media

Follow other users	Function	A user wishing to be notified of activities from other users
Unfollow other users	Function	A user wishing to unsubscribe to activities from other users
Follow recommendation	Object	List of users a user might want to get updates from
Follow recommendation	Function	Allow a user to interact with users they might be interested in without prior knowledge of their activities
User feed	Object	Historic of contents a user posted so far
Display User feed	Function	Allow followers to access to all previous posts of the following user

Use Case Diagram

Social Media

- View Following
- Like a Post
- Comment on Post
- Follow Another Users
- View Followers
- Unfollow Another Users
- <<extend>>
- <<extend>>
- Report Post
- <<extend>>
- View Following User Posts
- Manage Followers
- Create an account
- Recommendation for New Follower
- Disable an account
- View News Feed
- Delete an account
- <<include>>
- <<include>>
- <<include>>
- Help
- Add Post
- <<extend>>
- Login
- <<include>>
- Manage Post
- Delete Post
- Investigate inappropraite post
- <<include>>
- Edit Profile
- <<include>>
- Comment Post
- Like Post
- Update Post
- Search
- Check User Content
- Users
- HashTags
- <<extend>>
- Ban a User
- Request Incident Ticket
- <<include>>
- Create Incident Ticket
- Verify User Login info
- Schedule Incident Ticket
- Manage Incident Tickets
- Close Incident Ticket
- Stores information about users
- Cancel Incident Ticket
- Populates news feed per user
- Stores all post and comments of users

User

Admin

Technical Support

DB System

Analysis Class Diagram

Design Patterns

1.Composite Pattern
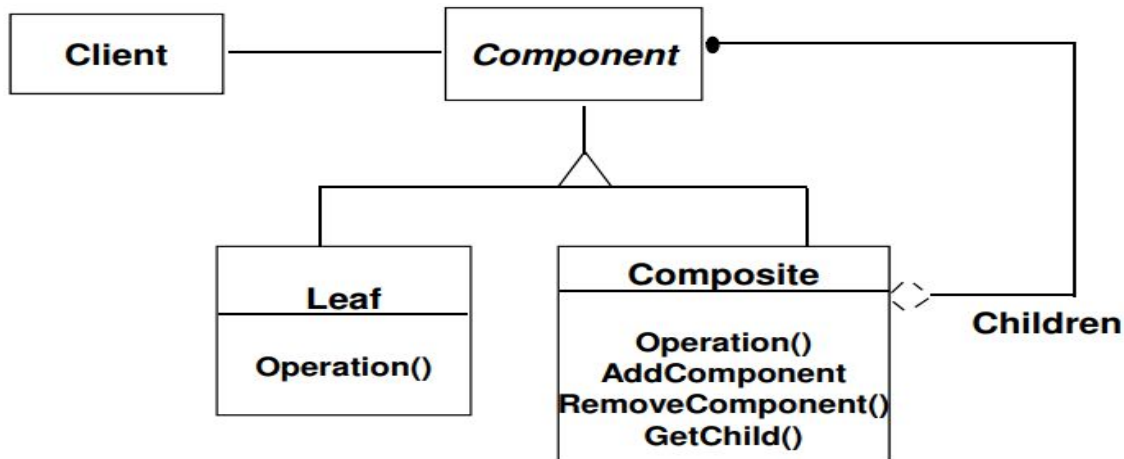
Client - Post
Leaf - CommentText
Composite - NestedComment
Component - Comment

A Post has nested comments. A comment contains at minimum some text that makes the comment. The comment may also contain other comments. This is represented in a composite pattern.

Composite pattern in the project
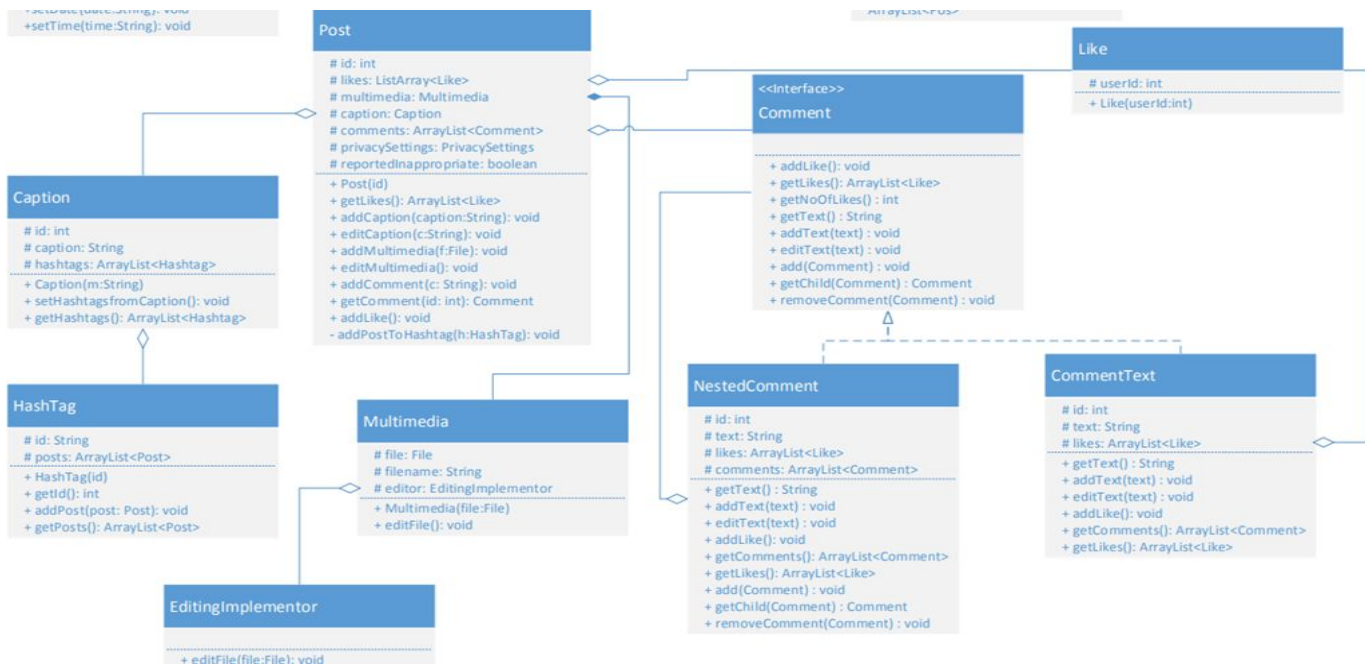
2.FOLLOW - Observer Pattern
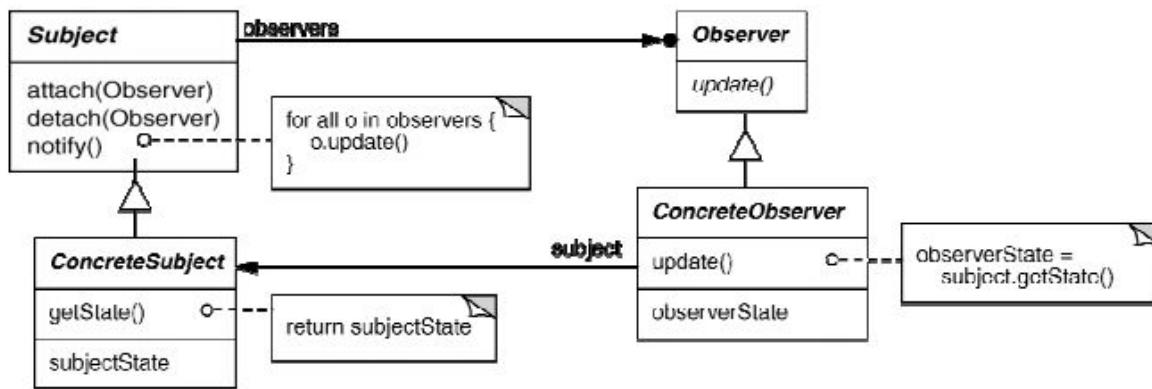
ConcreteSubject: User
ConcreteObserver: User

User *A* can follow another user *B*. By doing so, user *A* is notified each time user *B* adds or updates one of their post. Similarly, user *A* can decide at any time to unfollow user *B*. This action prevents user A to be notified of activities from user *B*.

Observer Pattern in the Project
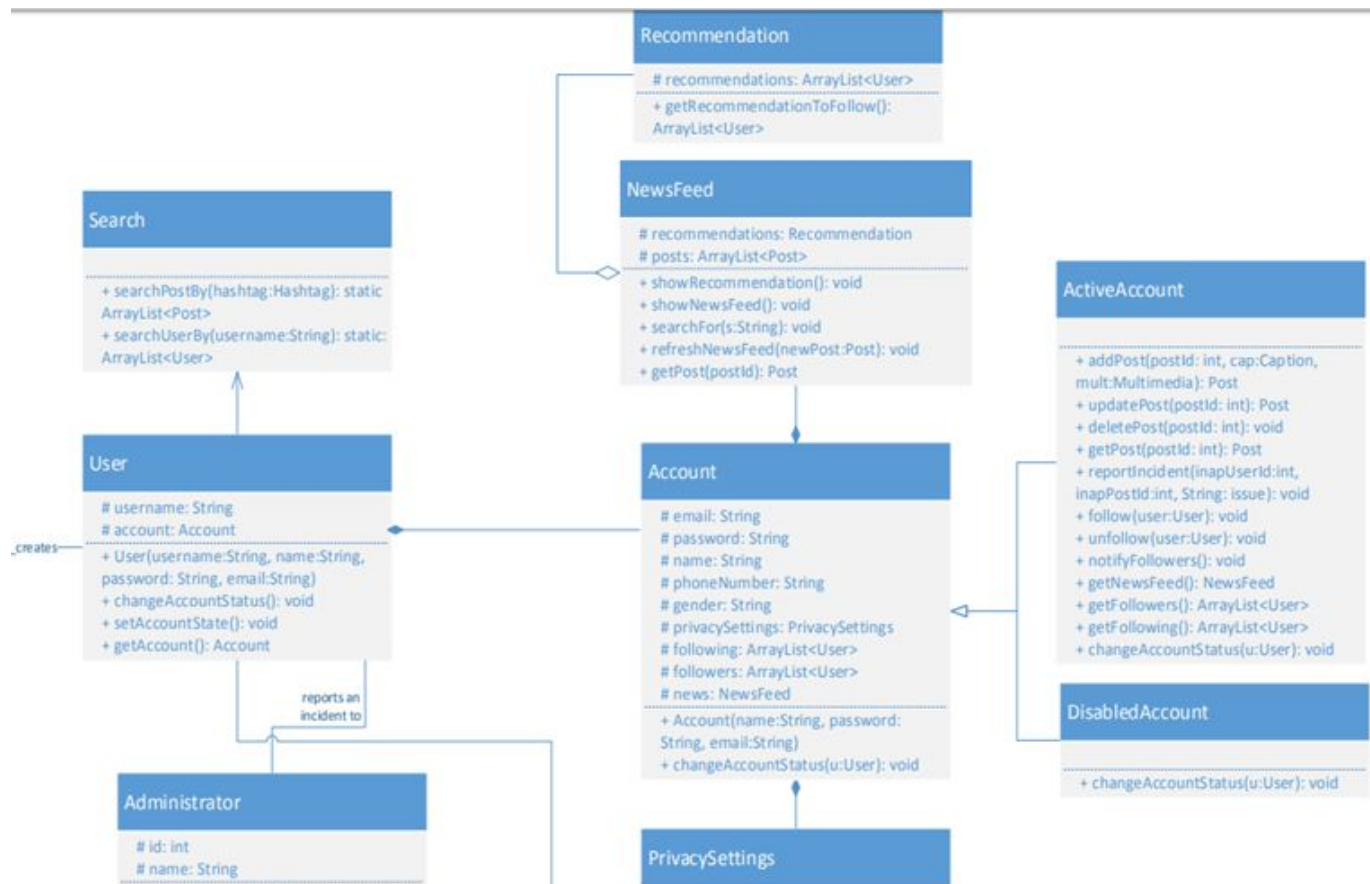
3.Editing functionality - Bridge Pattern
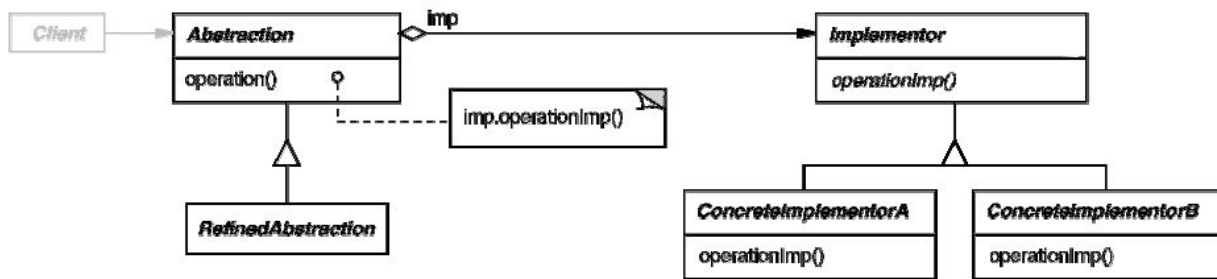
Abstraction: Multimedia
Implementor: EditingImplementor
ConcreteImplementorA: PictureEditor
ConcreteImplementorB: VideoEditor

A given multimedia can be either a picture or a video. Both formats can be edited using an editing tool. However, operations available for a picture are not the same as for a video. In that regard, two editors are implemented each specialized either in pictures or videos editing.

Bridge Pattern in the Project
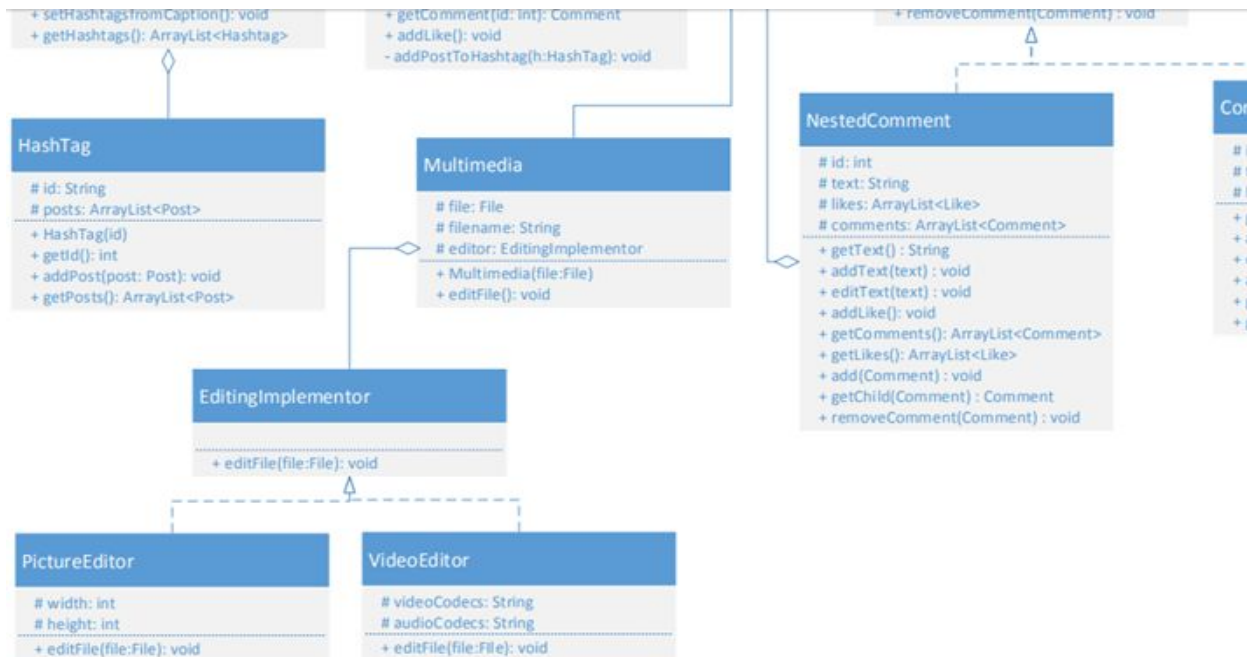
4.Account Status : State Design Pattern
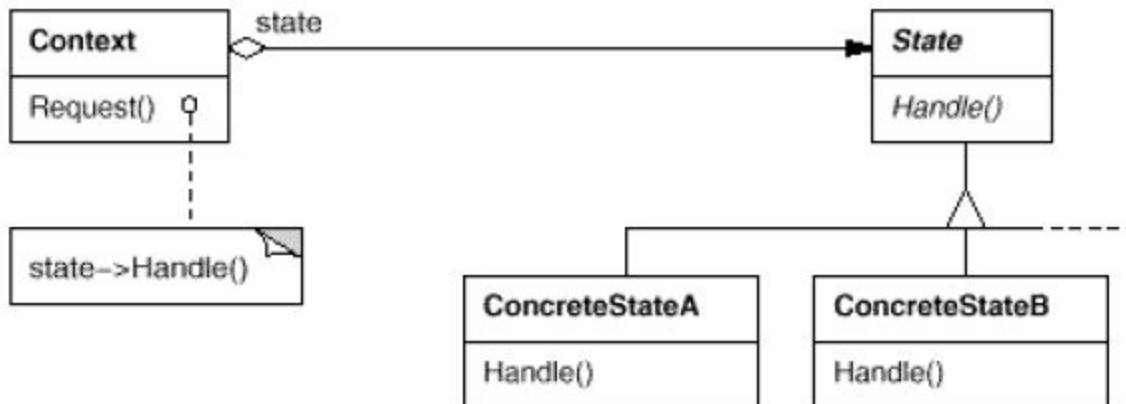
Context: User
State: Account
ConcreteStateA: ActiveAccount
ConcreteStateB: DisabledAccount

An account has two different states: either active or disabled. A user having a disabled account is reduced to basic functionalities proposed by the social platform. Those are mainly views and settings access. The user cannot add or update posts or, interact with other users. Moreover, their news feed is no more accessible by other users. However, the disabled account can be reactivated by the user at any time.
An active account allows users an access to every functionality available on the platform. That is, interacting and being visible publicly by other users.

State Pattern in the Project
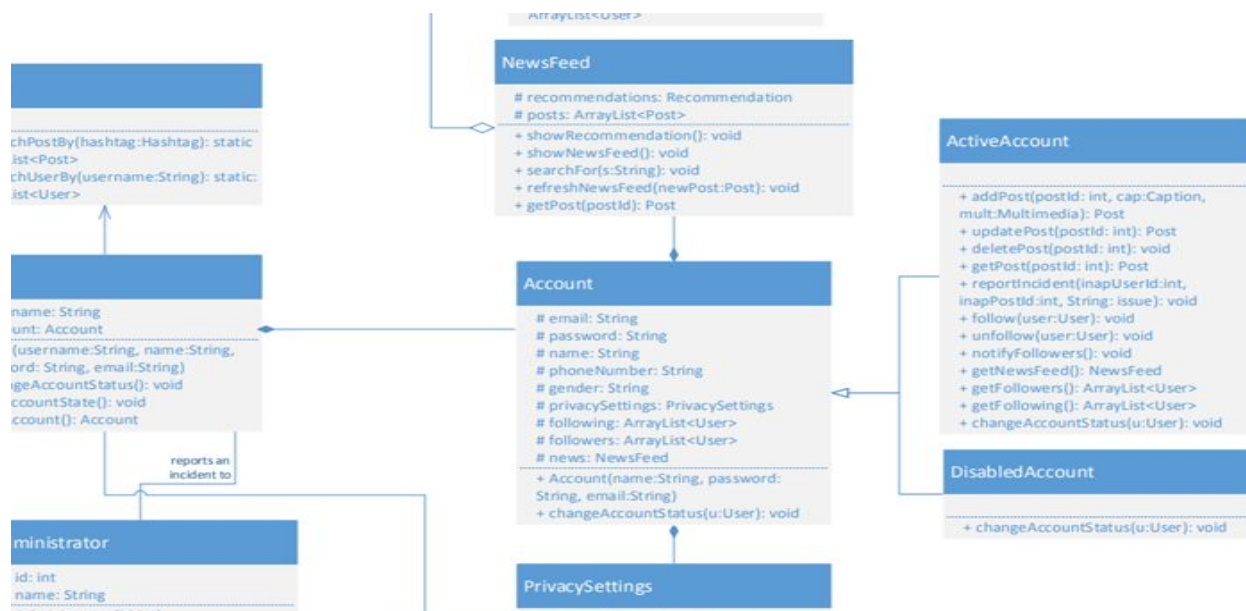
5.Search : Strategy Design Pattern
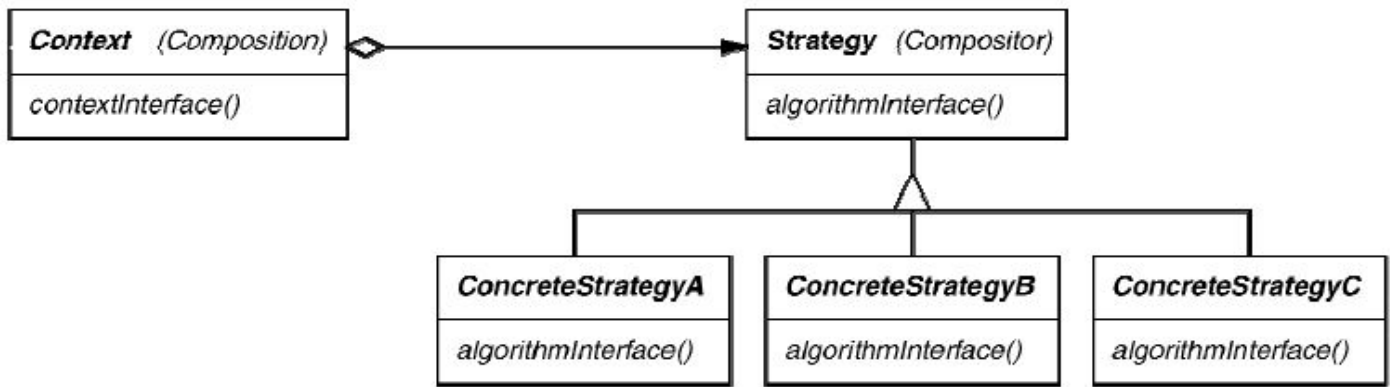
Context: SearchFramework
Search: Strategy
ConcreteStrategyA: SearchHashTag
ConcreteStrategyB: SearchUser

The search functionality allows a user to look for a post or a user.
Two strategies are available: either a user looks up using hashtags or usernames.

Strategy Pattern in the Project

Project - Social Media

DESIGN CLASS DIAGRAM

SearchHashTag

+ searchPostBy(hashtag:Hashtag): static
ArrayList<Post>

SearchUser

+ searchUserBy(username:String): static:
ArrayList<User>

**<<Interface>>
Search**

text: String

CreateSearchTag

+ makeSearch(text:String): Search
+ tagOrUser(text:String): Search

CreateSearchUser

+ makeSearch(text:String): Search
+ tagOrUser(text:String): Search

**<<Interface>>
SearchFramework**

+ makeSearch(text:String): Search
+ tagOrUser(text:String): Search

Recommendation

recommendations: ArrayList<User>

+ getRecommendationToFollow():
ArrayList<User>

NewsFeed

recommendations: Recommendation
posts: ArrayList<Post>

+ showRecommendation(): void
+ showNewsFeed(): void
+ searchFor(s:String): void
+ refreshNewsFeed(newPost:Post): void

ActiveAccount

+ addPost(postId: int, cap:Caption,
mult:Multimedia): void
+ updatePost(postId: int): void
+ deletePost(postId: int): void
+ getPost(postId: int): Post
+ reportIncident(inapUserId:int,
inapPostId:int, String: issue): void
+ follow(user:User): void
+ unfollow(user:User): void
+ notifyFollowers(): void
+ getNewsFeed(): NewsFeed
+ getFollowers(): ArrayList<User>
+ getFollowing(): ArrayList<User>
+ addFollower(user: User): void
+ deleteFollower(user: User): void
+ changeAccountStatus(u:User): void
+ incInappRank(): void

Login

+ signUp(username:String, name:String,
password: String, email:String): static
void
- createUserAccount(): static User
+ signIn(username: String,
password:String): static Boolean

User

username: String
account: Account

+ User(username:String, name:String,
password: String, email:String)
+ changeAccountStatus(): void
+ setAccountState(): void
+ getAccount(): Account

Account

email: String
password: String
name: String
phoneNumber: String
gender: String
privacySettings: PrivacySettings
posts: Stack[Post]
followers: ArrayList<User>
inappRank: int

+ Account(name:String, password:
String, email:String)
+ changeAccountStatus(u:User): void

DisabledAccount

+ changeAccountStatus(u:User): void

TechnicalSupport

id: int
name: String

+ TechnicalSupport(id:int)
+ closeIncidentTicket(incidentId): void
+ cancelIncidentTicket(incidentId): void
+ searchIncidentTicket(int id,in
inapuserid,inappostid,string issue): void

Administrator

id: int
name: String

+ Administrator(id:int)
+ createIncidentTicket(incidentId:int,
inapUser:int, inapPostId:int,
issue:String): IncidentTicket
+ deletePost(post: Post): void
+ updatePost(post: Post): void
+ deleteUser(userId): void
+browseInappropriatePosts():
ArrayList<Post>

PrivacySettings

allowed: ArrayList<User>
banned: ArrayList<User>

+ allowUser(u:User): void
+ banUser(u: User): void

IncidentTicket

id: int
inapUserId: int
inapPostId: int
issue: String
assignedTechnicianId: int
date: String
time: String

+IncidentTicket(id:int, inapUser:int,
inapPostId:int, issue:String)
+setId(int id): void
+setTechnician(techId:int): void
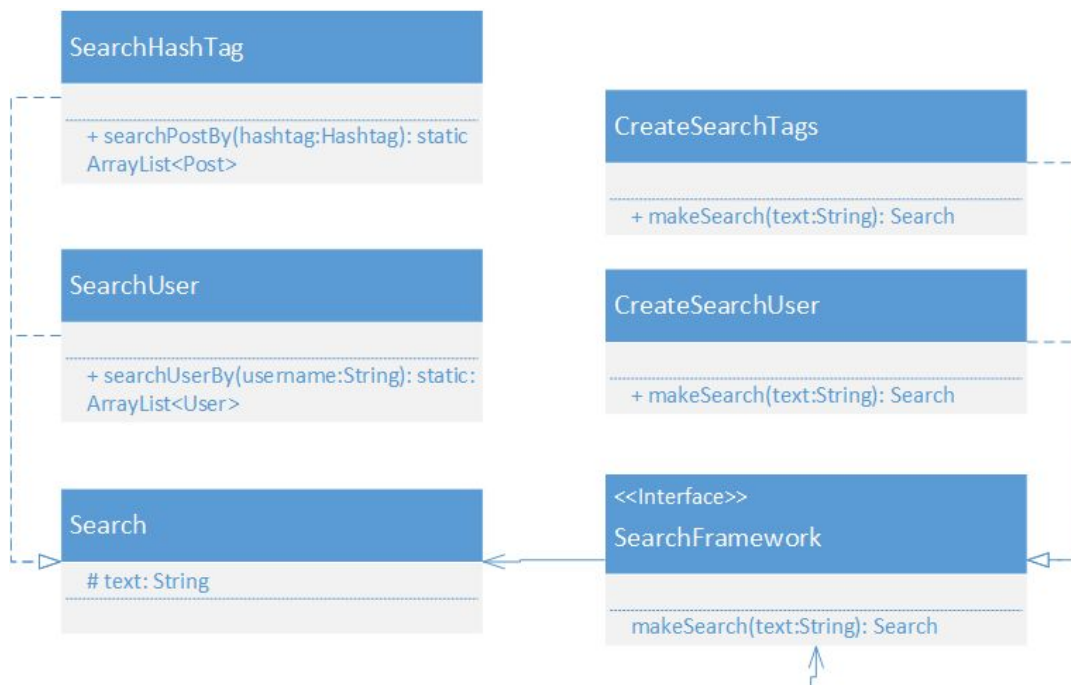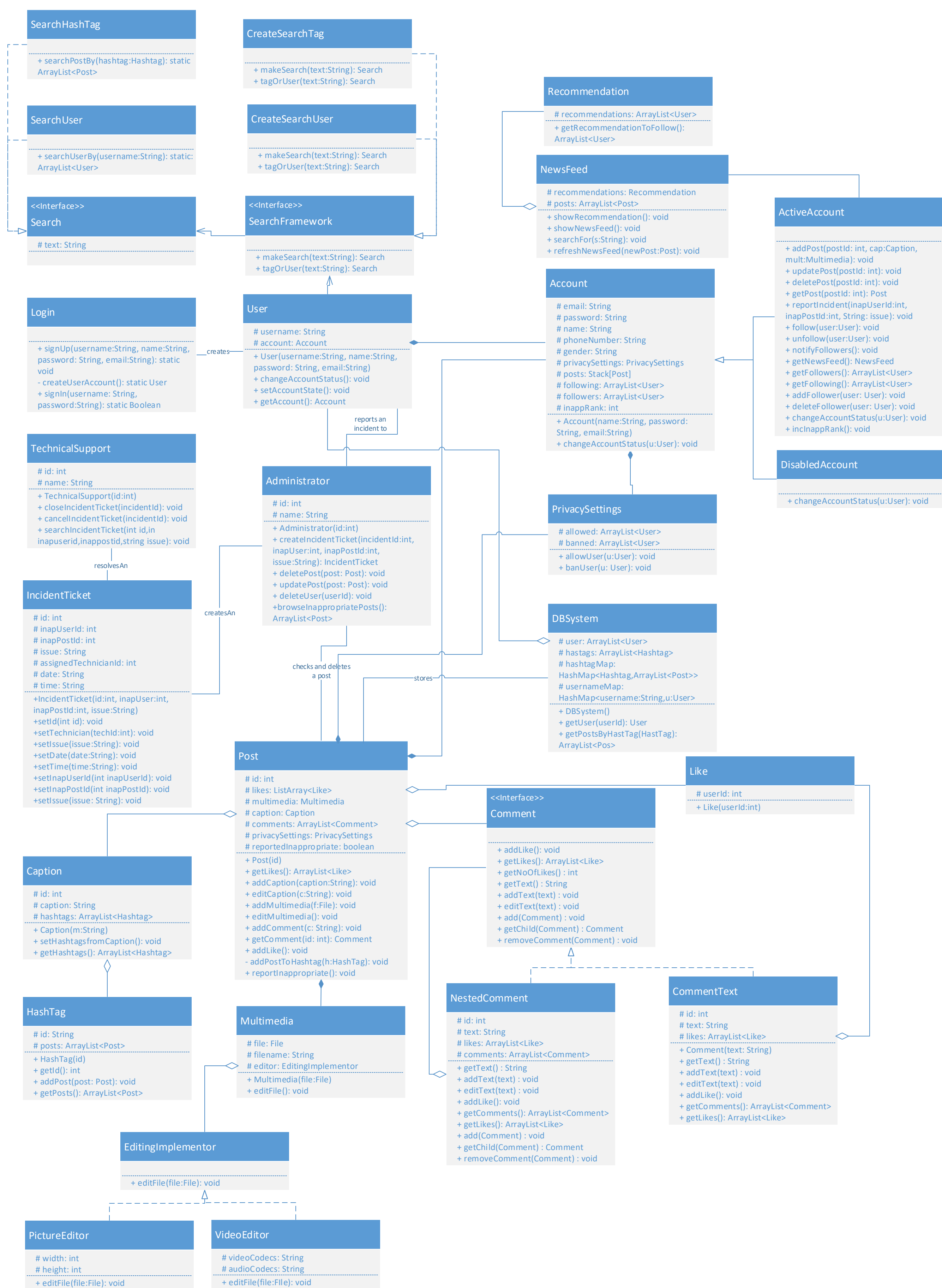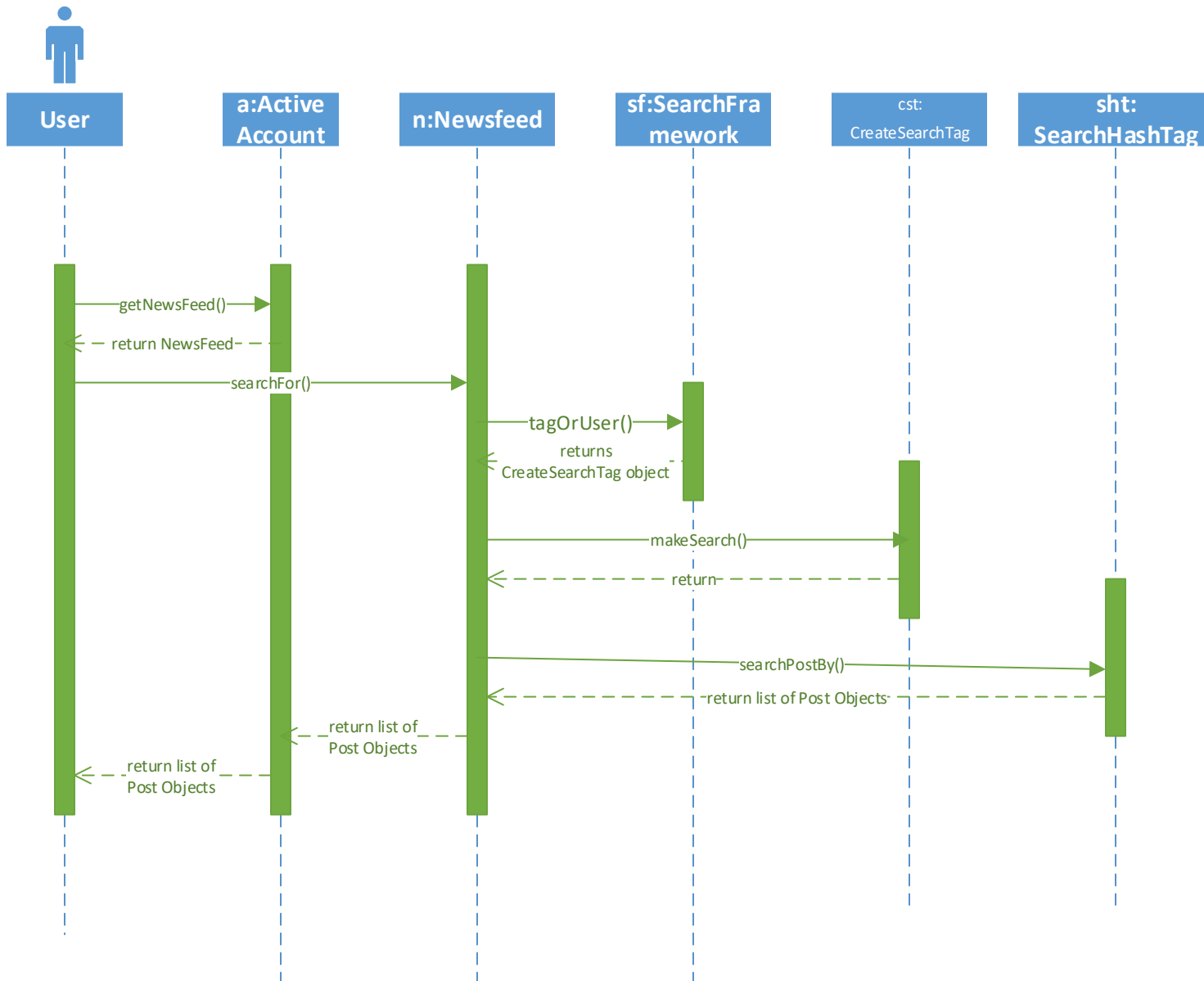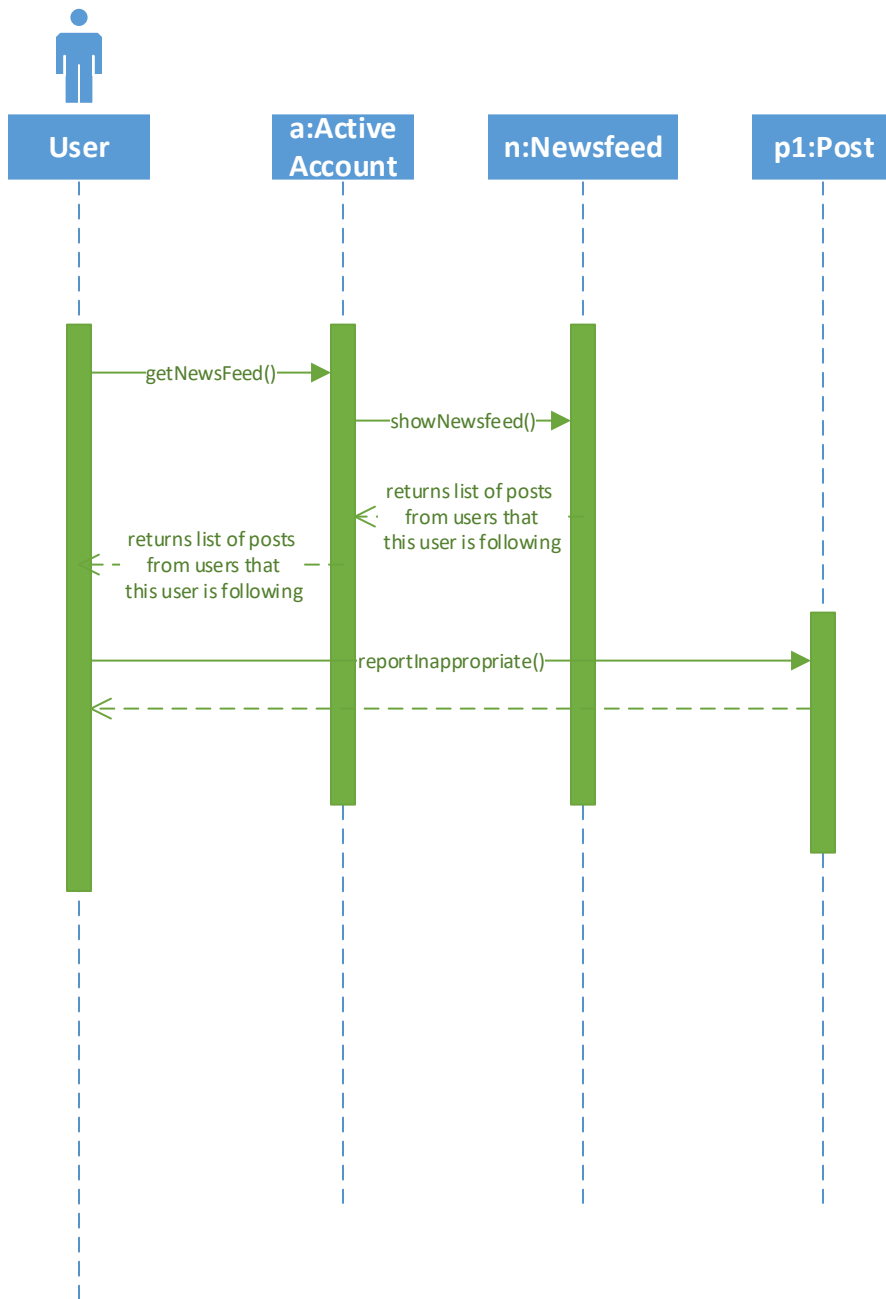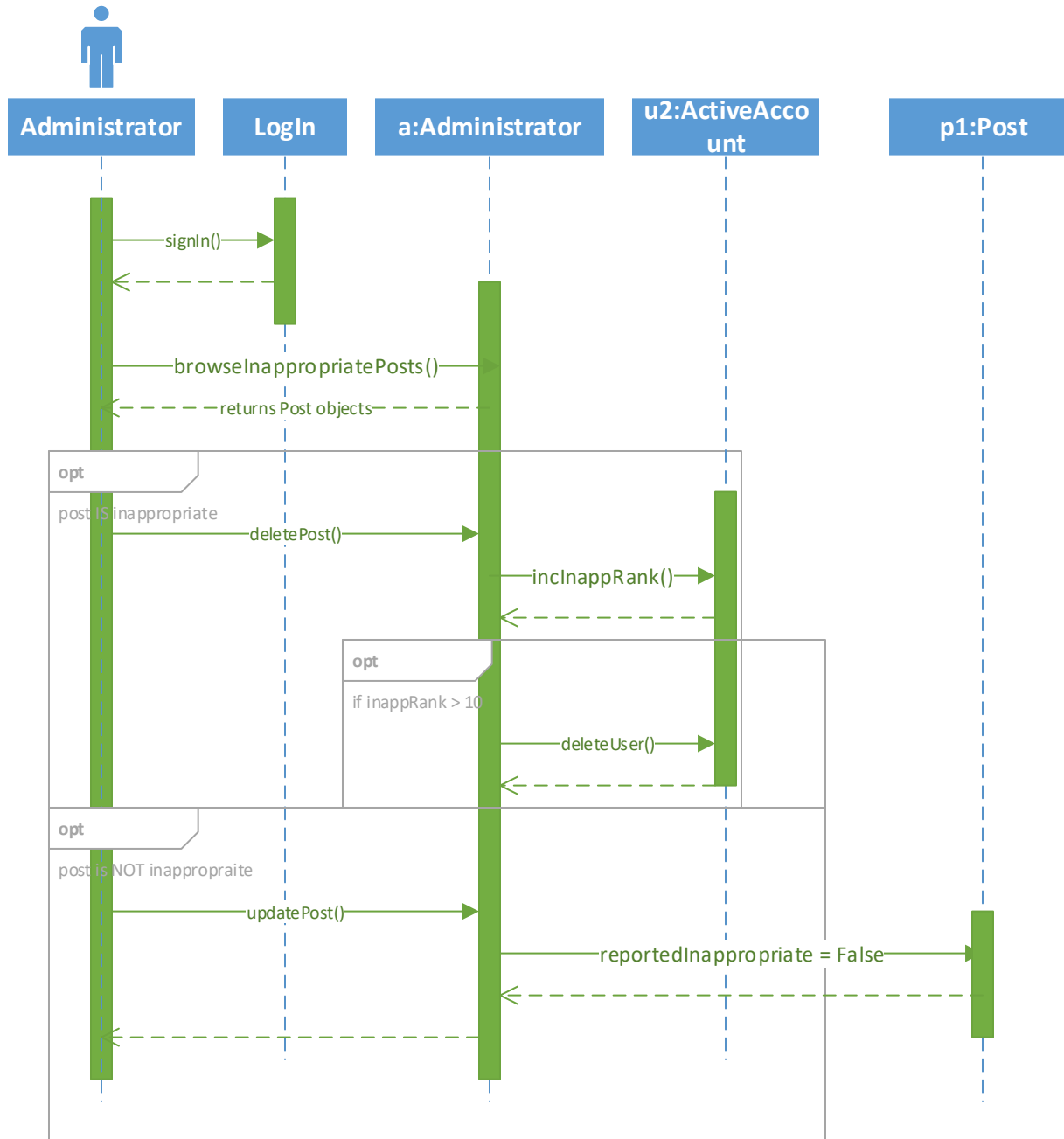+setIssue(issue:String): void
+setDate(date:String): void
+setTime(time:String): void
+setInapUserId(int inapUserId): void
+setInapPostId(int inapPostId): void
+setIssue(issue: String): void

DBSystem

user: ArrayList<User>
hastags: ArrayList<Hashtag>
hashtagMap:
HashMap<Hashtag,ArrayList<Post>>
usernameMap:
HashMap<username:String,u:User>

+ DBSystem()
+ getUser(userId): User
+ getPostsByHastTag(HastTag):
ArrayList<Pos>

Post

id: int
likes: ListArray<Like>
multimedia: Multimedia
caption: Caption
comments: ArrayList<Comment>
privacySettings: PrivacySettings
reportedInappropriate: boolean

+ Post(id)
+ getLikes(): ArrayList<Like>
+ addCaption(caption:String): void
+ editCaption(c:String): void
+ addMultimedia(File): void
+ editMultimedia(): void
+ addComment(c: String): void
+ getComment(id: int): Comment
+ addLike(): void
- addPostToHashtag(h:HashTag): void
+ reportinappropriate(): void

Caption

id: int
caption: String
hashtags: ArrayList<Hashtag>

+ Caption(m:String)
+ setHashtagsfromCaption(): void
+ getHashtags(): ArrayList<Hashtag>

Comment <<Interface>>

+ addLike(): void
+ getLikes(): ArrayList<Like>
+ getNoOfLikes() : int
+ getText() : String
+ addText(text) : void
+ editText(text) : void
+ add(Comment) : void
+ getChild(Comment) : Comment
+ removeComment(Comment) : void

Like

userId: int

+ Like(userId:int)

HashTag

id: String
posts: ArrayList<Post>

+ HashTag(id)
+ getId(): int
+ addPost(post: Post): void
+ getPosts(): ArrayList<Post>

Multimedia

file: File
filename: String
editor: EditingImplementor

+ Multimedia(file:File)
+ editFile(): void

NestedComment

id: int
text: String
likes: ArrayList<Like>
comments: ArrayList<Comment>

+ getText() : String
+ addText(text) : void
+ editText(text) : void
+ addLike(): void
+ getComments(): ArrayList<Comment>
+ getLikes(): ArrayList<Like>
+ add(Comment) : void
+ getChild(Comment) : Comment
+ removeComment(Comment) : void

CommentText

id: int
text: String
likes: ArrayList<Like>

+ Comment(text: String)
+ getText() : String
+ addText(text) : void
+ editText(text) : void
+ addLike(): void
+ getComments(): ArrayList<Comment>
+ getLikes(): ArrayList<Like>

EditingImplementor

+ editFile(file:File): void

PictureEditor

width: int
height: int

+ editFile(file:File): void

VideoEditor

videoCodecs: String
audioCodecs: String

+ editFile(file:File): void

creates
reports an
incident to
resolvesAn
createsAn
checks and deletes
a post
stores

SEQUENCE DIAGRAMS

Search for HashTag

Report Post as Inappropriate

User   a:Active Account   n:Newsfeed   p1:Post

getNewsFeed()

showNewsfeed()

returns list of posts from users that this user is following
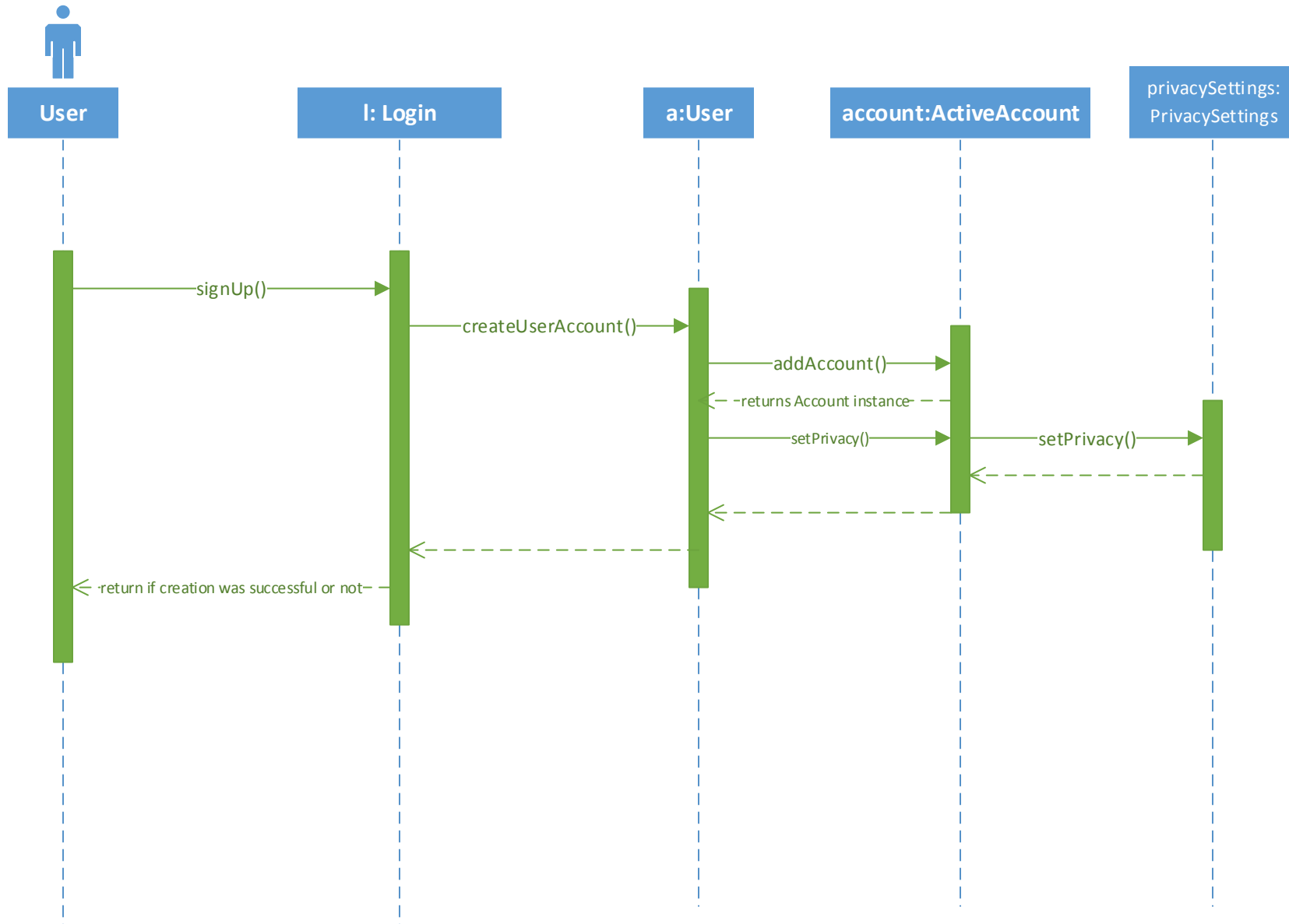
returns list of posts from users that this user is following

reportInappropriate()

Admin addressing
Inappropriate Posts

Administrator | LogIn | a:Administrator | u2:ActiveAccount | p1:Post

signIn()

browseInappropriatePosts()
returns Post objects

opt
[post is inappropriate]

deletePost()

incInappRank()

opt
[if inappRank > 10]

deleteUser()

opt
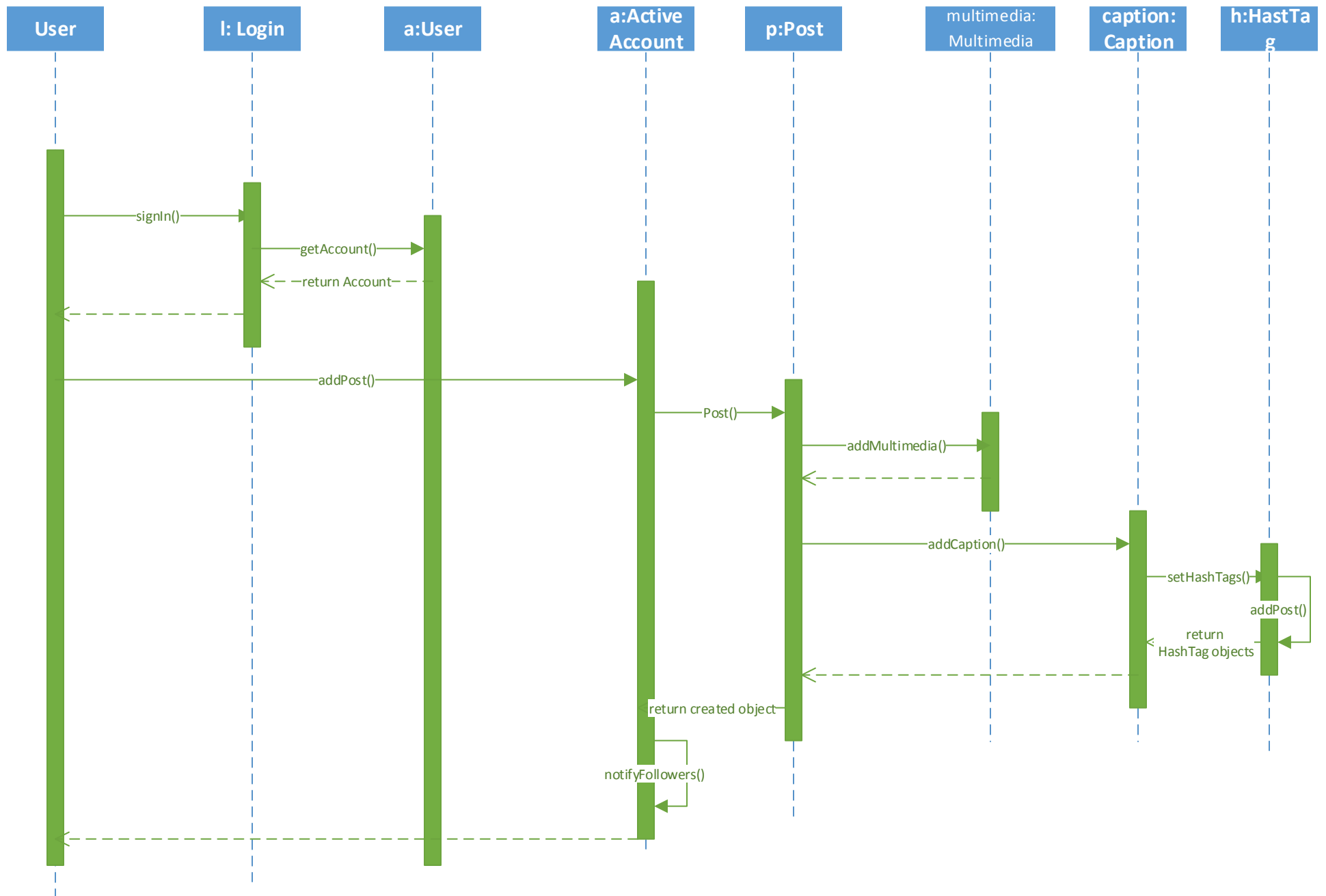[post is NOT inappropraite]

updatePost()

reportedInappropriate = False

Follow user from recommendation

User Sign Up

Post Something

User	l: Login	a:User	a:Active Account	p:Post	multimedia: Multimedia	caption: Caption	h:HastTag

signIn()

getAccount()

return Account

addPost()

Post()

addMultimedia()

addCaption()

setHashTags()

addPost()

return HashTag objects

return created object

notifyFollowers()

Check NewsFeed

User

l: Login

a:User

a:Active Account

n:Newsfeed

signIn()

getAccount()

return Account

getNewsFeed()

getNewsFeed()

showNewsfeed()

returns list of posts from users that this user is following

returns list of posts from users that this user is following

returns list of posts from users that this user is following

Interact with a Post from NewsFeed – Like a Post, Comment on a Post and Comment on a Comment

User | n:Newsfeed | p1:Post | ct:CommentText | nc:NestedComment

getNewsFeed()

returns list of posts from users that this user is following

addLike()

addComment()

CommnetText()

creates a CommentText object

getComment()

returns list of Comment objects

addLike()

addComment()

Refresh NewsFeed

```
User        n:NewsFeed

  |──refreshNewsFeed()──►|
  |                      |
  |           showRecommendation()
  |                      |
  |             showNewsFeed()
  |                      |
  |◄ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ |
```

Incident ticket assign and close

Incident ticket assign and cancel

Technical Support

I:IncidentTicket

searchIncidentTicket()

IncidentTicket()

closeIncidentTicket()