



**Dharmsinh Desai University, Nadiad**  
**Faculty of Technology**  
**Department of Computer Engineering**

**B. Tech. CE Semester – IV**

**Subject: Software Engineering Principles and Practices**

**Project Title: SocialHub**

**By:**

- 1) Keval Gandevia, Roll No: CE037, ID: 19CEUEG017**
- 2) Jenil Gandhi, Roll No: CE038, ID: 19CEUON133**

**Guided by:**

- 1) Prof. Brijesh Bhatt**
- 2) Prof. Pinkal Chauhan**
- 3) Prof. Jigar Pandya**

**DHARMSINH DESAI UNIVERSITY**

**NADIAD-387001, GUJARAT**



**CERTIFICATE**

This is to certify that the project entitled as “SocialHub” is a bonafide report of the work carried out by

**1) Keval Gandevia, ID: 19CEUEG017**

**2) Jenil Gandhi, ID: 19CEUON133**

Of Department of Computer Engineering, semester IV, under the guidance and supervision of **Prof. Pinkal Chauhan** for the subject Software Engineering Principles and Practices during the academic year 2020-2021.

Project Guide  
Assistant Professor  
Prof. Pinkal Chauhan  
Department of Computer  
Engineering,  
Faculty of Technology,  
Dharmsinh Desai University,  
Nadiad  
Date: 01/04/2021

Head of the Department  
Dr. C. K Bhensdadia  
Prof. & Head,  
Department of Computer  
Engineering,  
Faculty of Technology,  
Dharmsinh Desai University,  
Nadiad  
Date: 01/04/2021

# **Contents:**

1) <a href="#">Abstract/Introduction</a> .....	4
2) <a href="#">Software Requirement Specifications</a> .....	5
3) <a href="#">Design</a> .....	12
3.1) <a href="#">Use Case Diagram</a> .....	12
3.2) <a href="#">Class Diagram</a> .....	13
3.3) <a href="#">DFD Diagram</a> .....	14
3.4) <a href="#">Structure Chart</a> .....	17
3.5) <a href="#">Sequence Diagram</a> .....	21
3.6) <a href="#">Activity Diagram</a> .....	23
3.7) <a href="#">Data Dictionary</a> .....	24
4) <a href="#">Implementation Details</a> .....	28
4.1) <a href="#">Modules</a> .....	28
4.2) <a href="#">Major Functionalities</a> .....	29
5) <a href="#">Conclusion</a> .....	33
6) <a href="#">Limitations and Future Extensions</a> .....	34
7) <a href="#">Bibliography</a> .....	35

# **1) Abstract/Introduction**

---

- ❖ SocialHub is a social media website having many functionalities for user like Login, Signup, forgot password, responsive messages for each alert and errors.
- ❖ The Home page is provided with functionality of viewing your friend's post, getting notification about events, getting follow requests.
- ❖ The current logged in user can like post, comment on post, share post, report post, view post, delete post, save post.
- ❖ User can send friend request to other users, search other users, see what's going around the world, chat with his friends.
- ❖ User would be able to modify the account settings like delete account, update account, report account, etc.
- ❖ In user's profile, user will able to see all his posts, all followers, followings, likes and comments on all posts, change profile photo.
- ❖ If user is premium user, he would be able to track his monthly gained followers, likes and comments on his posts. He would also be able to advertise new products to the targeted audience.

## 2) Software Requirement Specifications

---

### SocialHub

#### 1. Account Management

##### R.1.1 Register User:

- Description: The user needs to register himself/herself to use this website by providing the credentials mentioned in the input. The user will not be able to login until he verifies his/her account.
- Input: Choose a username, provide his/her full name, email-address and verify it and need to set a password.
- Output: The account would be created and the user will be redirected to the login page.

##### R.1.2 Registered people can Login into the website:

- Description: The user will be allowed to access the account only if he is not reported and provides the correct combination of username and password.
- Input: Enter username and password.
- Output: The user will be redirected to the home page.

##### R.1.3 Forgot Password:

- Description: The user will have a chance to change his/her password only if he clicks on the link sent to him on the email.
- Input: Open link sent in the mail and provide new password and confirm it.
- Output: Password would be changed.
- Process: Sending mail from the server to the registered email address.
- Status: If the user clicked it within 30 minutes then he can reset his password.

#### R.1.4 Change Password:

- Description: The user will be able to change his/her password from their account.
- Input: Provide the new password and confirm it.
- Output: Password would be changed.

#### R.1.5 Update Profile:

- Description: The user can update his/her profile in the profile section. Ex: update bio profile picture etc.
- Input: User selection.
- Output: The user's profile will be updated.

#### R.1.6 Report User:

- Description: The user could be reported if he/she is found to have posted explicit or vulgar content.
- Input: Click on the settings and press report a user button.
- Output: The user would be reported.

#### R.1.7 Logout:

- Description: The user will log-out from his/her account.
- Input: Click on the logout button.
- Output: The user is logged out from the website.

#### R.1.8 Delete Account:

- Description: The user can delete his/her account.
- Input: Delete account button in the profile.
- Output: Account deleted.
- Process: The user will be sent an email confirming that he/she really wants to delete the account.

→ Status: The user's account would be deleted if he/she chooses to do so or else no action would be taken.

## **2. Post Management**

### **R.2.1 Post a Photo, Location, Caption:**

→ Description: The user will be allowed to post a photo specifying the location and the caption. If the caption contains hashtags it would be identified on its own and would be reflected in the caption.

→ Input: The user will select a photo from their device and select a caption and location.

→ Output: The post would be posted and would be visible to all the friends.

### **R.2.2 Delete Post:**

→ Description: If the user feels that the post, he/she posted must be removed for whatever reason then the user has got the rights to do so.

→ Input: The user will have to go to the profile section and select the post's settings and select delete post option.

→ Output: The Post will be deleted.

### **R.2.3 Report Post:**

→ Description: Particular post could be reported by anyone and would be reviewed by the admin if it gets more than 25 reports.

→ Input: Report post button in the settings menu on the post.

→ Output: A request has been submitted to admin and proper action would be taken by the admin.

### **R.2.4 Save Post:**

→ Description: It will save the post into the local machine of the user who has saved the post.

→ Input: Save post button in the settings menu on the post.

→ Output: The post would be saved in the machine.

### R.2.5 Detect Hashtag:

→ Description: The person while writing a caption or a comment if they use a syntax with # as a prefix then that would be considered a tag on that post.

→ Input: Simple text preceded with a # sign ex #cool.

→ Output: This would be considered as a tag and not a part of the caption or comment.

### R.2.6 See Hashtag related posts:

→ Description: If the user clicks on a hashtag the user will be able to see the hashtag related posts.

→ Input: Click on the hashtag.

→ Output: Images related to a particular hashtag.

## 3. Friend Management

### R.3.1 Send Friend Request:

→ Description: The user is able to send friend requests to the other users. If the account is public then it is only possible to follow a particular user.

→ Input: Click over the add friend or follow button.

→ Output: Added the friend to the friend list.

### R.3.2 Public Account:

→ Description: The account is considered a public account if it has more than 100k followers or friends. The account needs to be verified by the admin.

→ Input: User selection.

→ Output: Verification request to admin.



### R.3.3 Get Suggestions:

- Description: The user will get suggestions for friends based on his friends.
- Input: User selection.
- Output: The user can see a list of suggestions.

### R.3.4 Search Friends:

- Description: The user can search his/her friends in the search option where he/she needs to enter the username or the name of the user to see the suggestions.
- Input: Provide the username or the name of the user.
- Output: Related searches would appear.

#### R.3.4.1 See what's trending:

- Description: The user can see trending posts.
- Input: 'Search' option selection.
- Output: The page containing trending posts would be displayed.

## **4. Content Management**

### R.4.1 Remove Vulgar Posts:

- Description: If a post is tagged by multiple users and if the admin's deep learning algorithm also finds it vulgar then the post would be subjected to be deleted.
- Input: Tag button in settings.
- Output: Post is reported.

### R.4.2 Remove Reported Users:

- Description: If the user is reported more than 25 times and the admin also finds it suspicious then the user would be sent an email regarding this and the user would be banned, and the user will be activated only if he/she deletes the content.
- Input: Tagged users.
- Output: Banning them from accessing the website.

### R.4.3 Remove Tagged Comments:

→ Description: If a comment is tagged more than 10 times then that comment is reviewed by the Natural Language processing model of the admin and if that is classified toxic then it is meant to be removed.

→ Input: Toxic comments.

→ Output: Comment would be removed by the admin.

### R.4.4 Remove Hacked Accounts:

→ Description: If the user from his mail account informs about the hacking his or her account then the account would be temporarily banned from being accessed and the user would be sent an OTP for confirming that the account is hacked and if the user replies yes then the account would be deleted.

→ Input: Mail from user's mail address.

→ Output: Delete account if hacked.

## **5. Chat Management**

### R.5.1 Send Message to his/her Friend:

→ Description: The user can send a message to his/her friends.

→ Input: User selection.

→ Output: Message window of user's choice.

### R.5.2 Get Notifications Messages:

→ Description: If anyone sends messages to the user, then will get notifications about messages.

→ Input: User selection.

→ Output: Get notifications.

### R.5.3 User can make groups with friends:

- Description: The user can make the group with his/her friends and interact with all his/her friends in this group.
- Input: Select option described within the home page to make a group.
- Output: A group will be created and a messaging facility will be started.

## **6. Miscellaneous Post Management**

### R.6.1 Share Post:

- Description: When the user shares a link of a post to another user and when the other user clicks on this link, he would be able to see the post that user1 has shared.
- Input: Share option in the post.
- Output: Link generated to be shared.
- Note: This is only applicable to the public profiles.

### R.6.2 Like Post:

- Description: The user would be able to like the post.
- Input: Press like button displayed in the post.
- Output: The post will be liked by that user.

### R.6.3 Unlike Post:

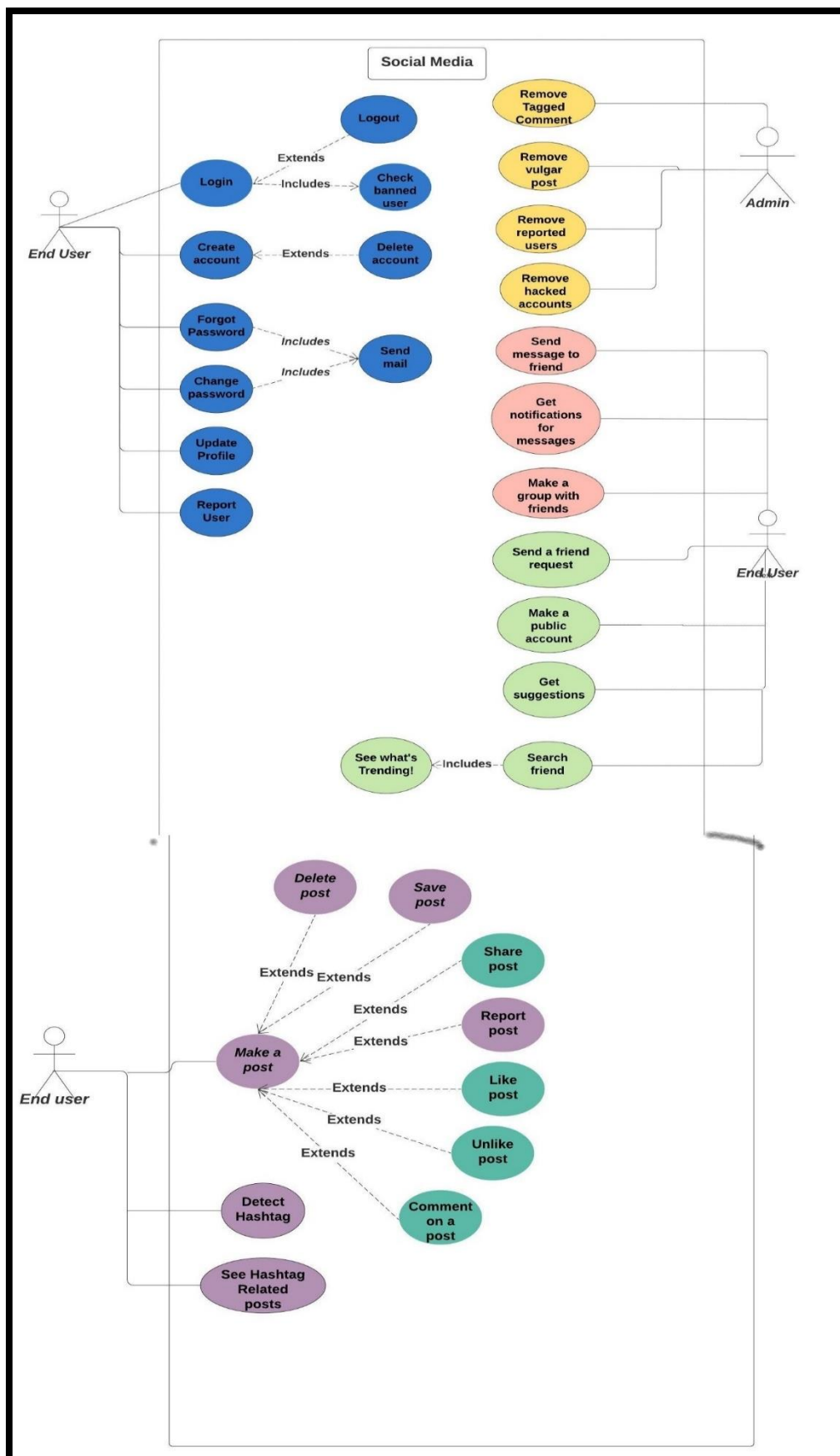
- Description: The user would be able to unlike the post.
- Input: Press like button displayed in the post.
- Output: The post will be unliked by that user.

### R.6.4 Comment on Post:

- Description: The user would be able to comment on a particular post.
- Input: Write your comment in the comment section and press send icon.
- Output: The post will be commented by that user.

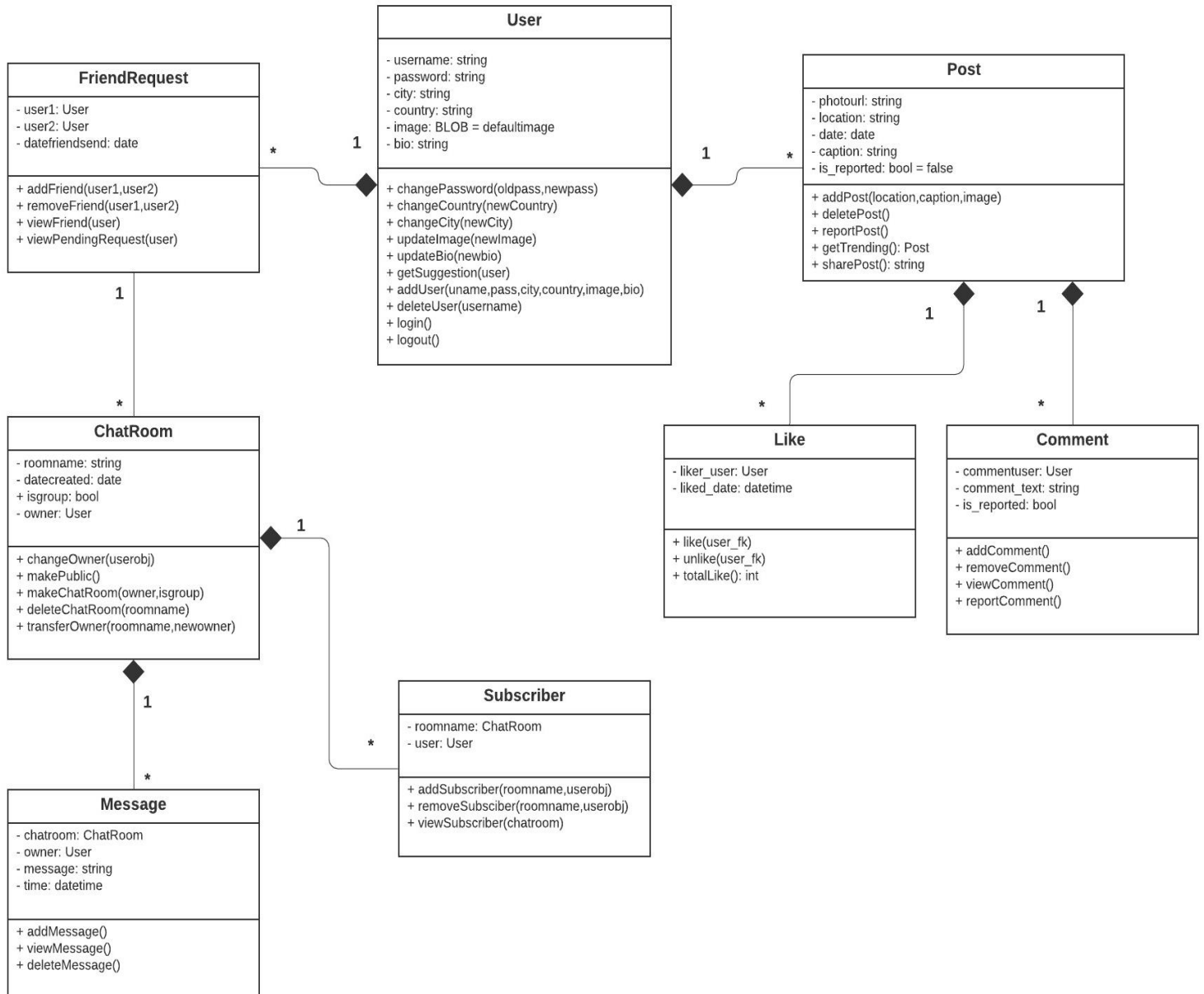
### 3) Design

#### 3.1) Use Case Diagram



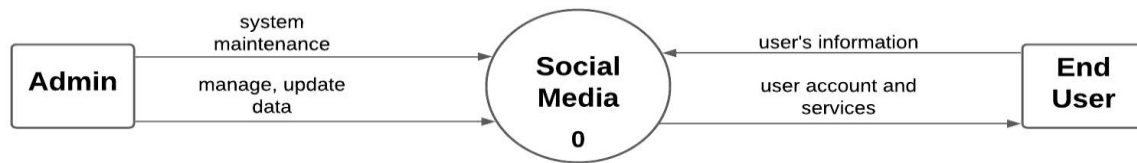
## 3.2) Class Diagram

### SocialHub

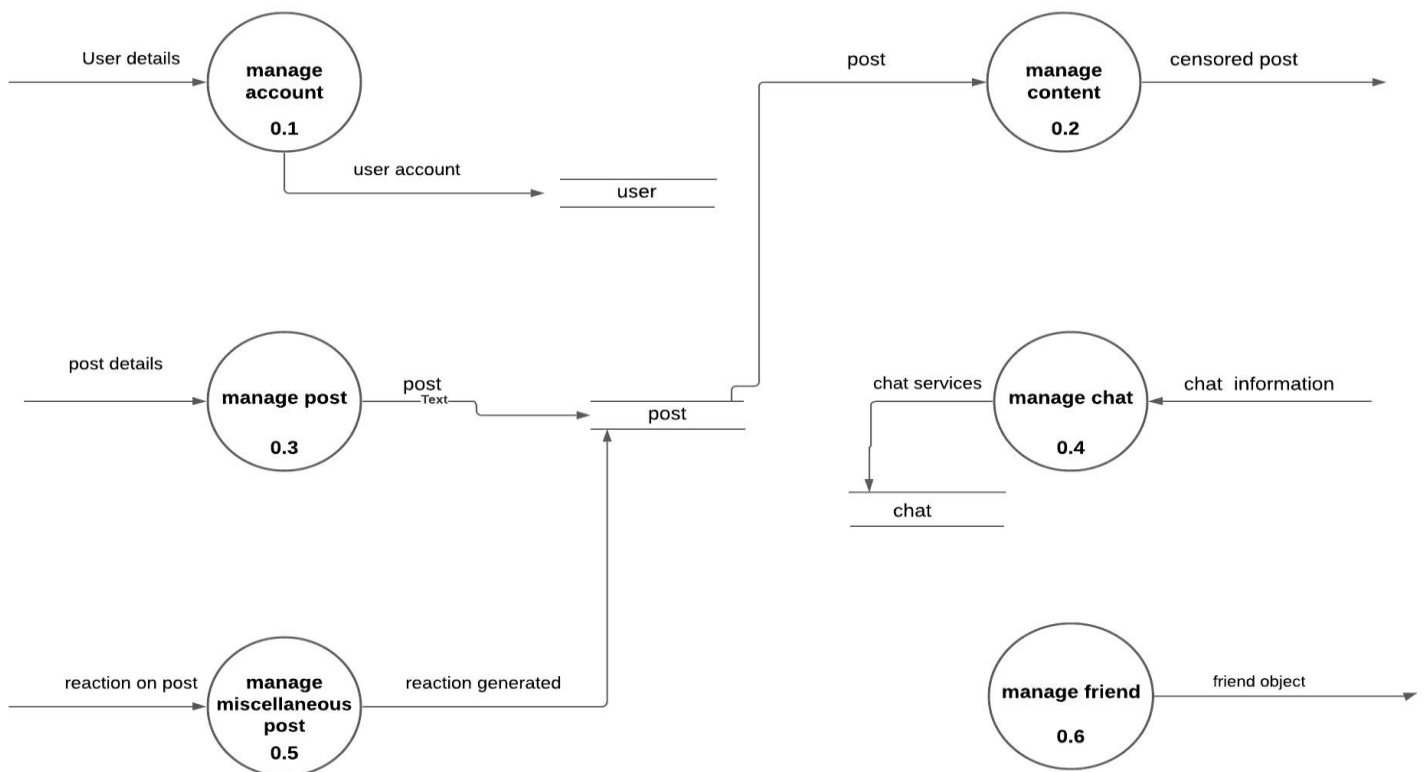


### 3.3) DFD Diagram

#### ❖ Context Diagram and Level – 1 Diagram

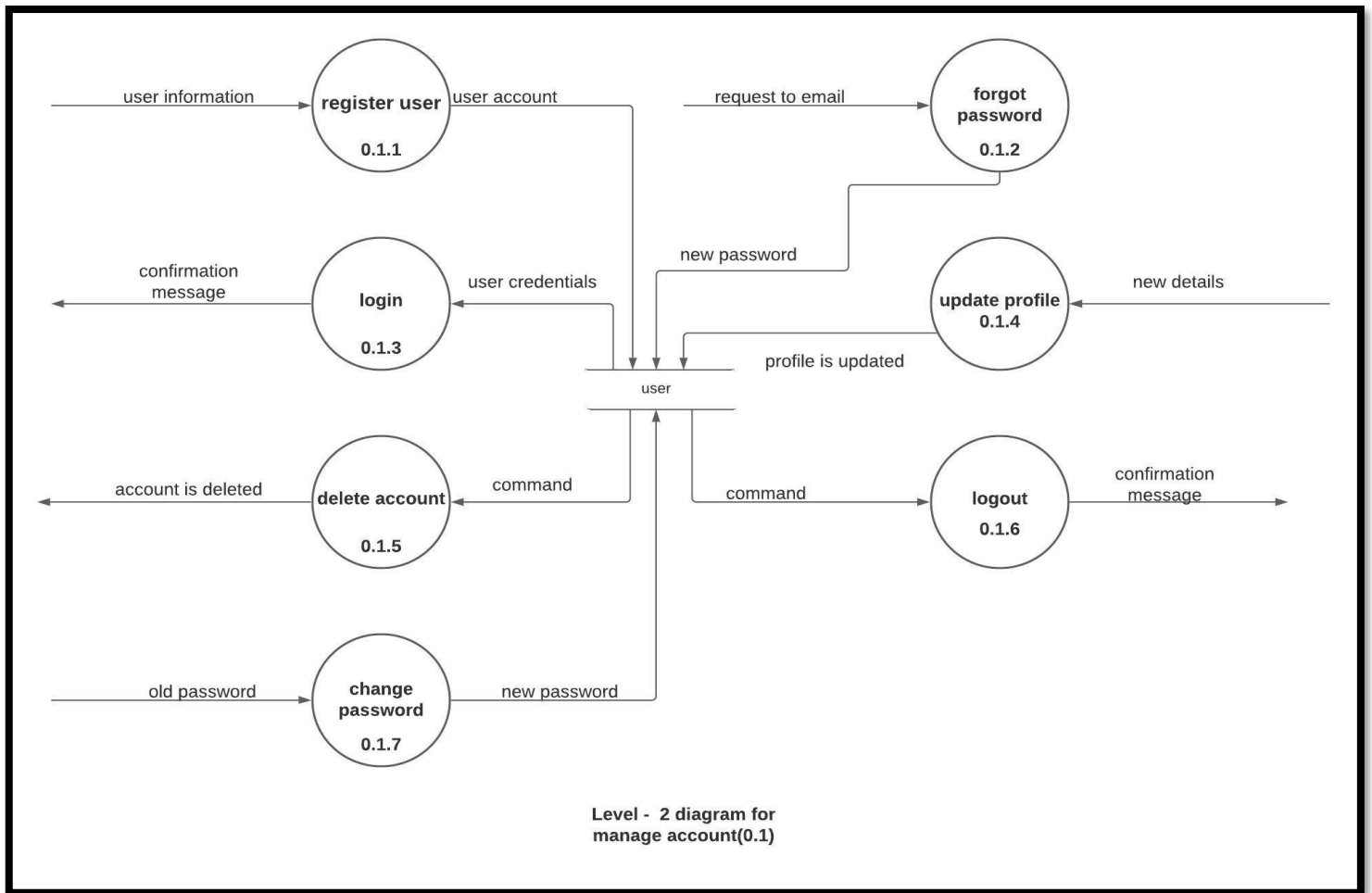


Context diagram

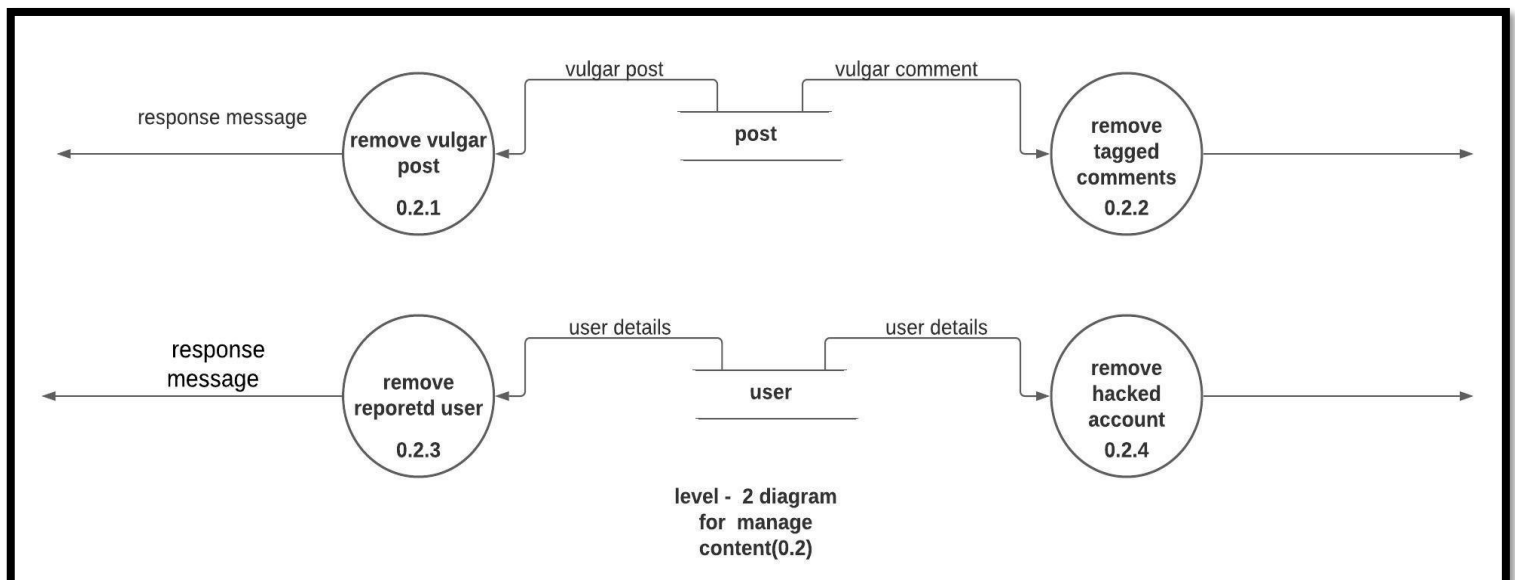


Level - 1 diagram

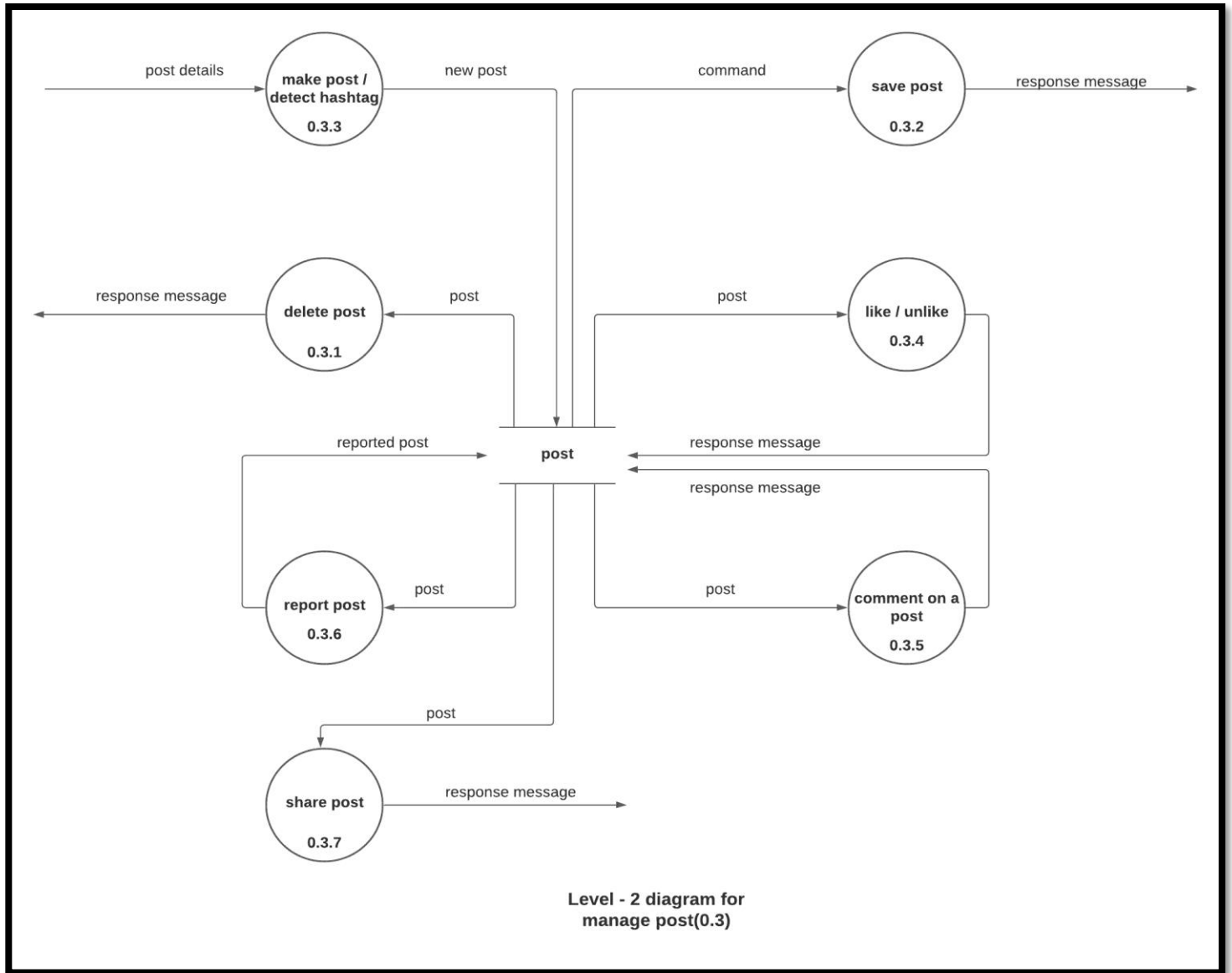
#### ❖ Level – 2 Diagram for manage account (0.1)



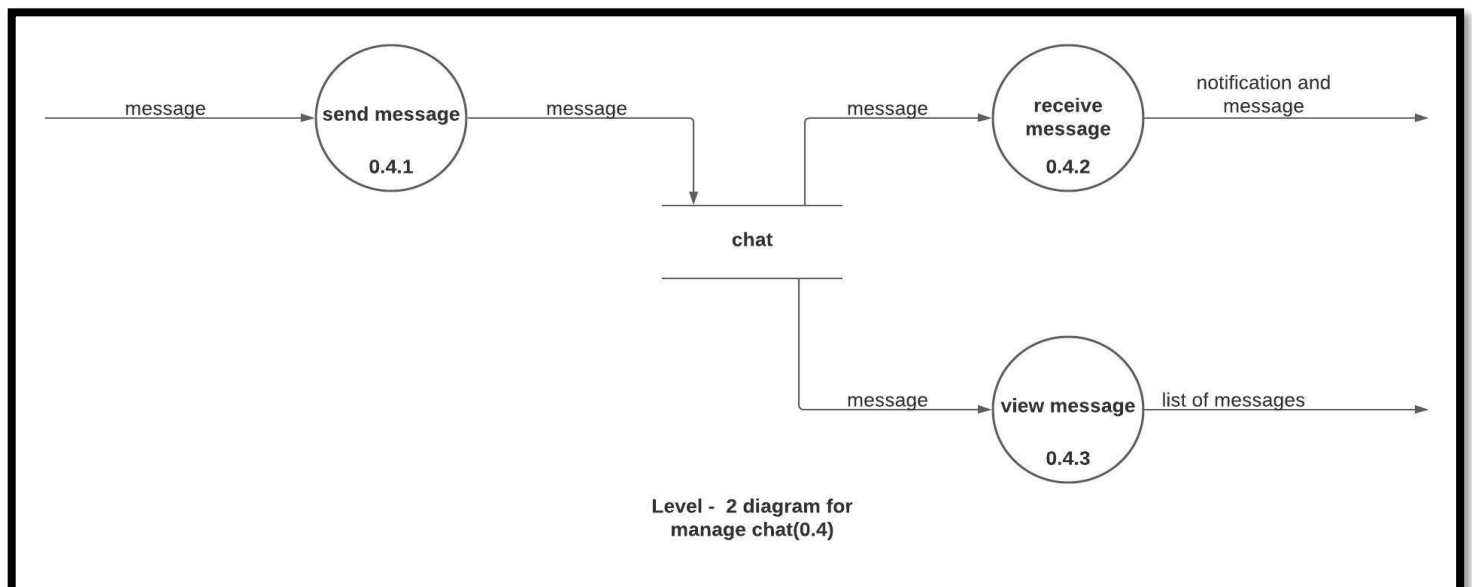
❖ Level – 2 Diagram for manage content (0.2)



❖ Level – 2 Diagram for manage post (0.3)

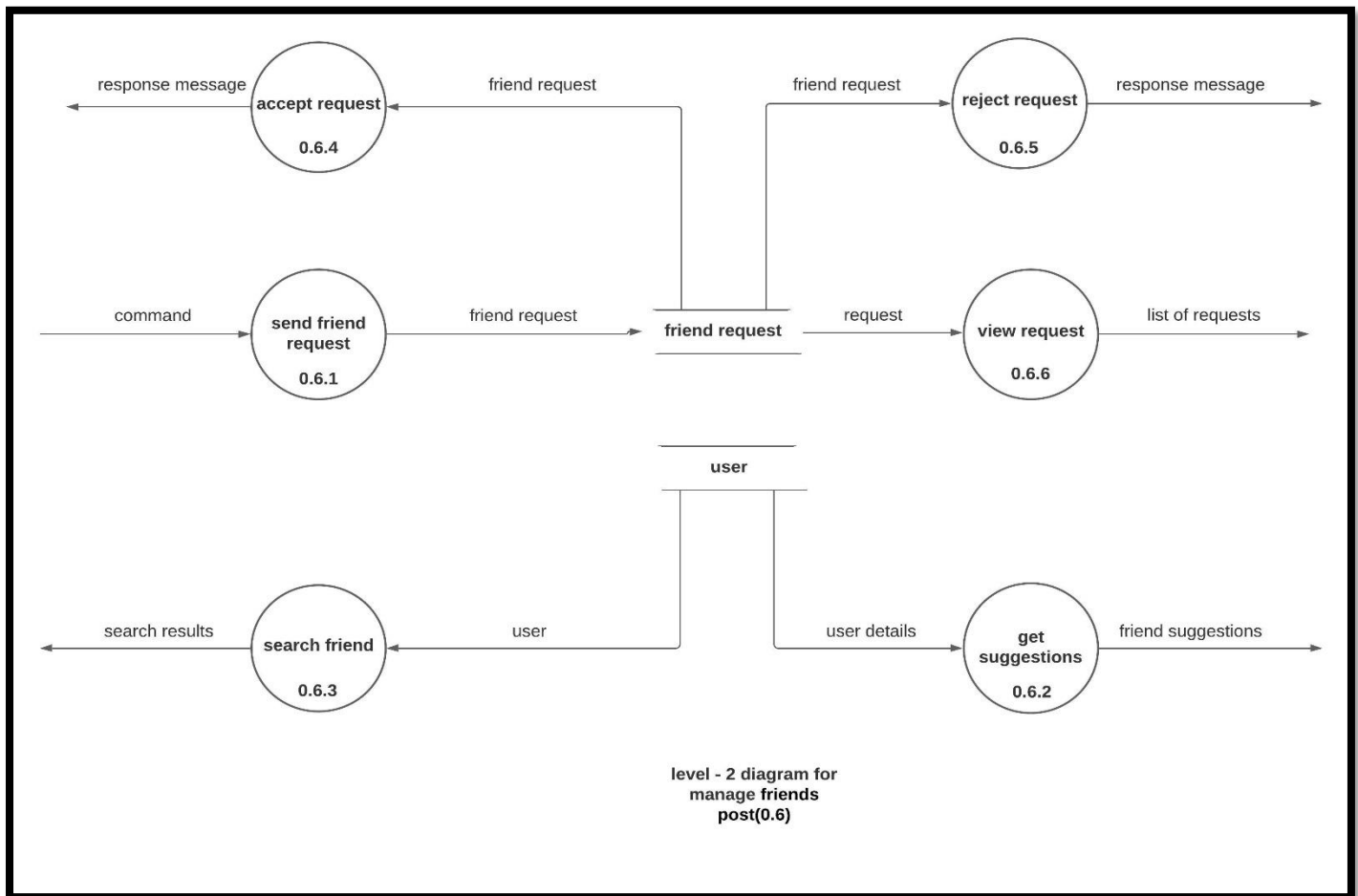


❖ Level – 2 Diagram for manage chat (0.4)



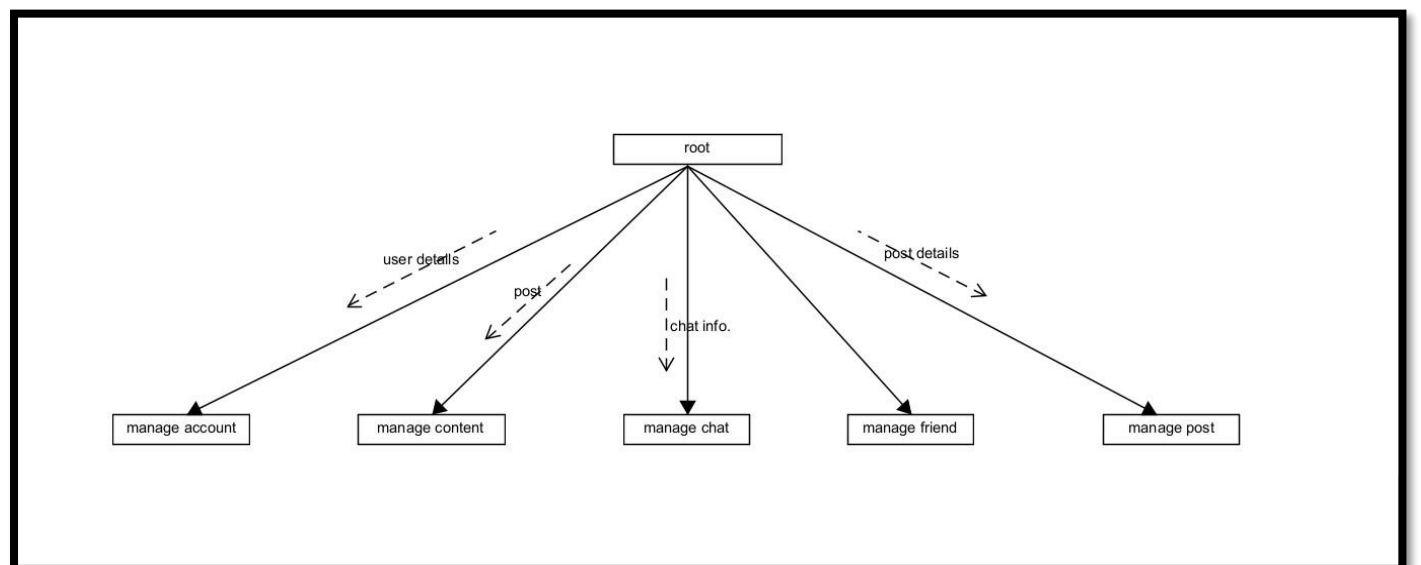


## ❖ Level – 2 Diagram for manage friend (0.6)

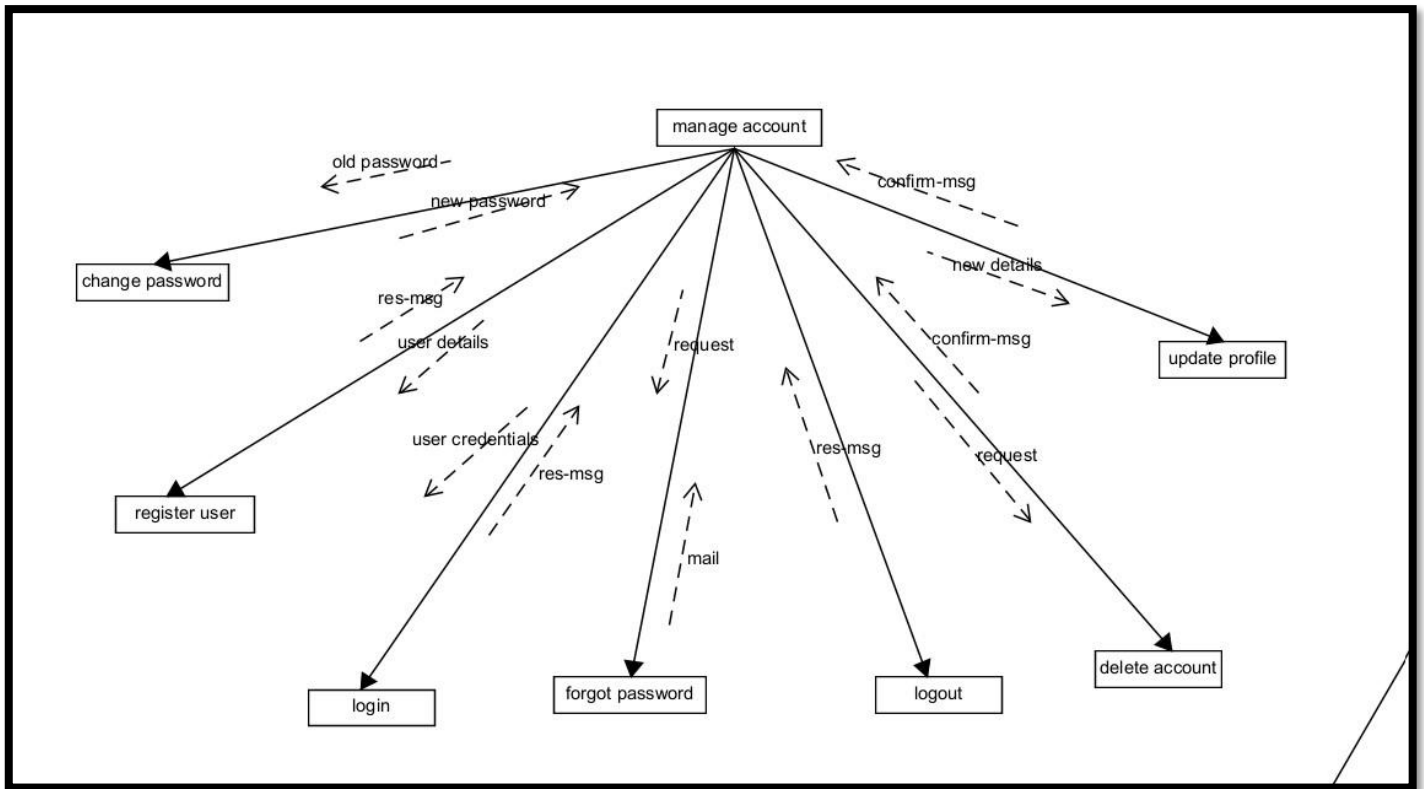


## 3.4) Structure Chart

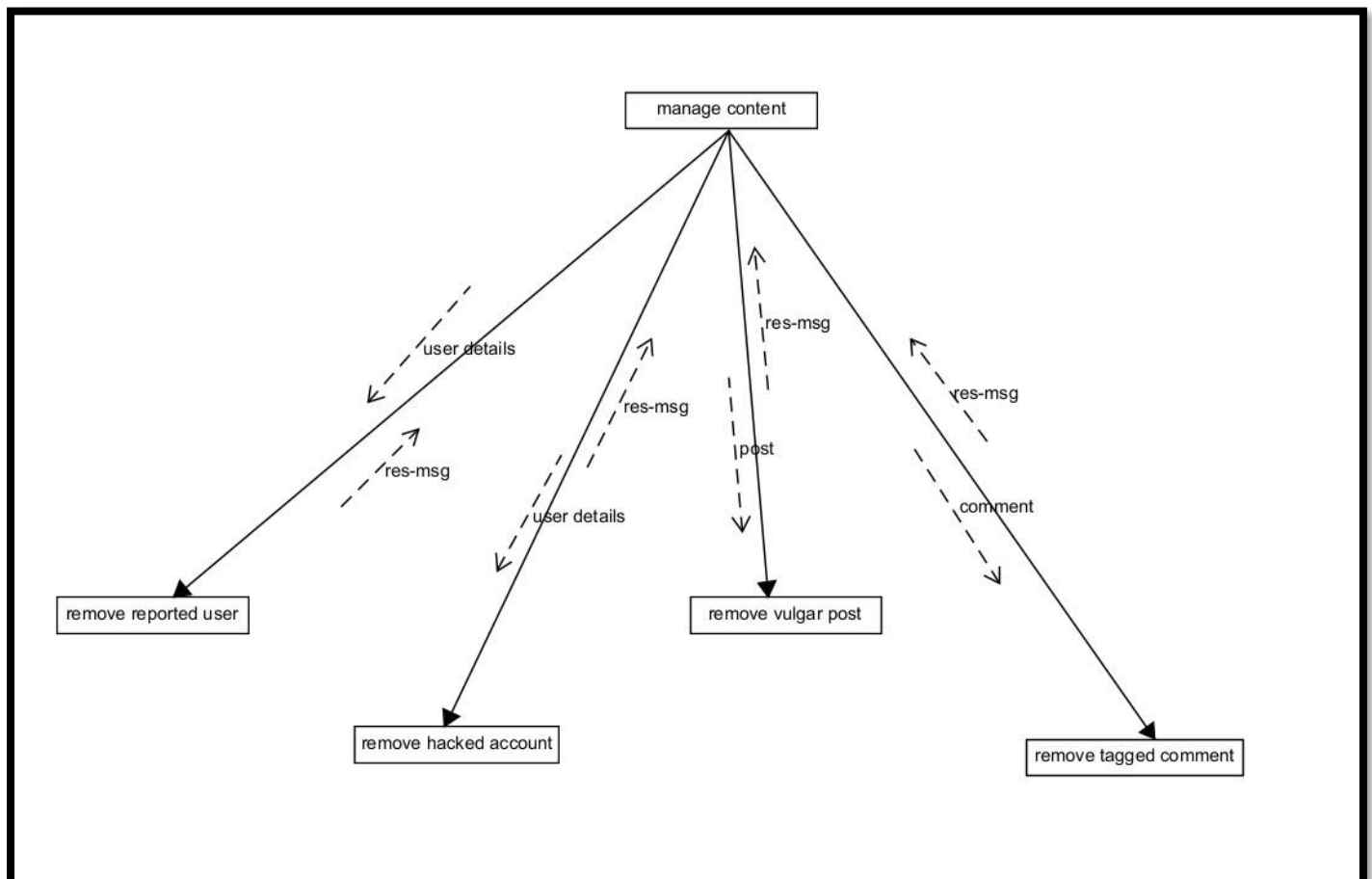
### ❖ Level – 0 and Level – 1 Diagram



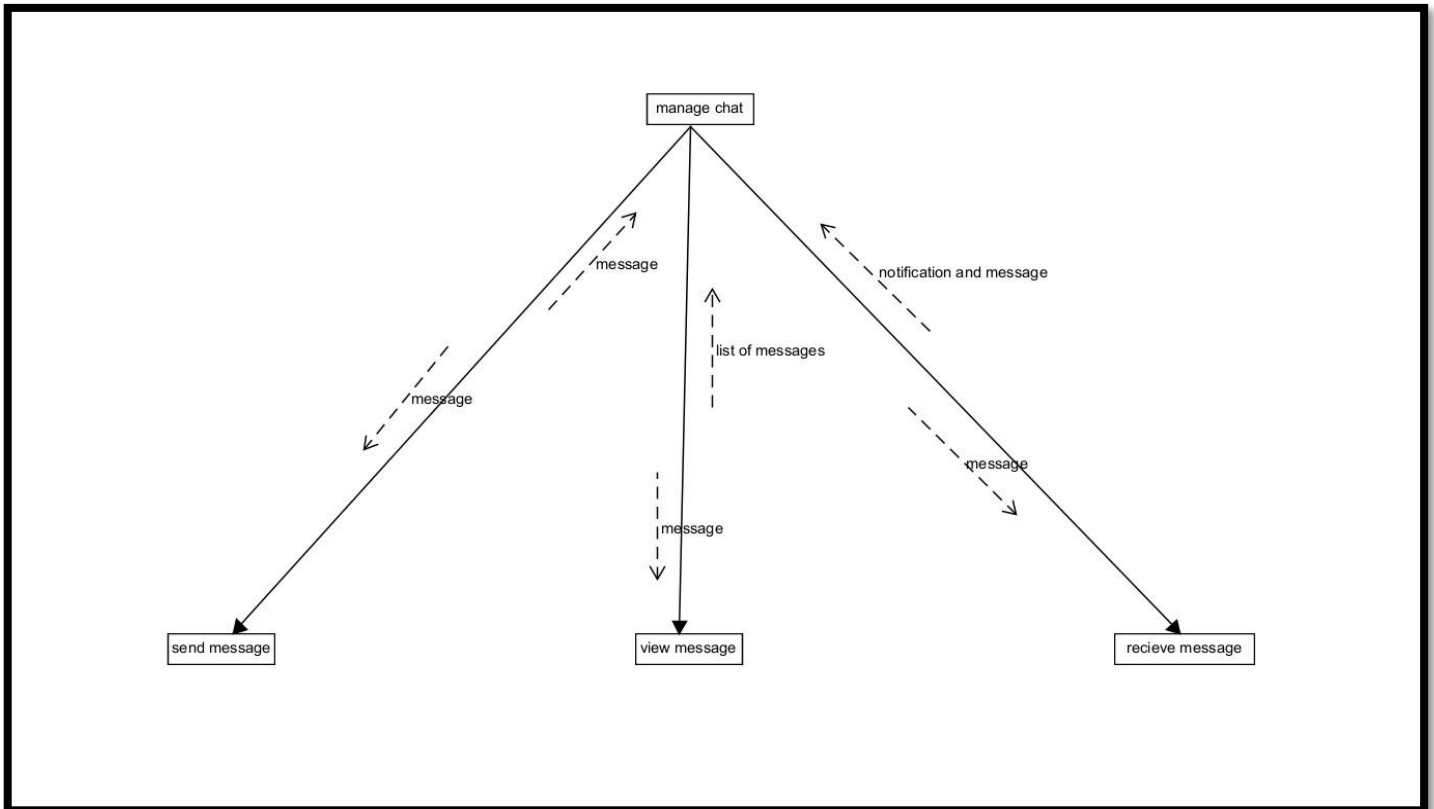
❖ Level – 2 Diagram for manage account



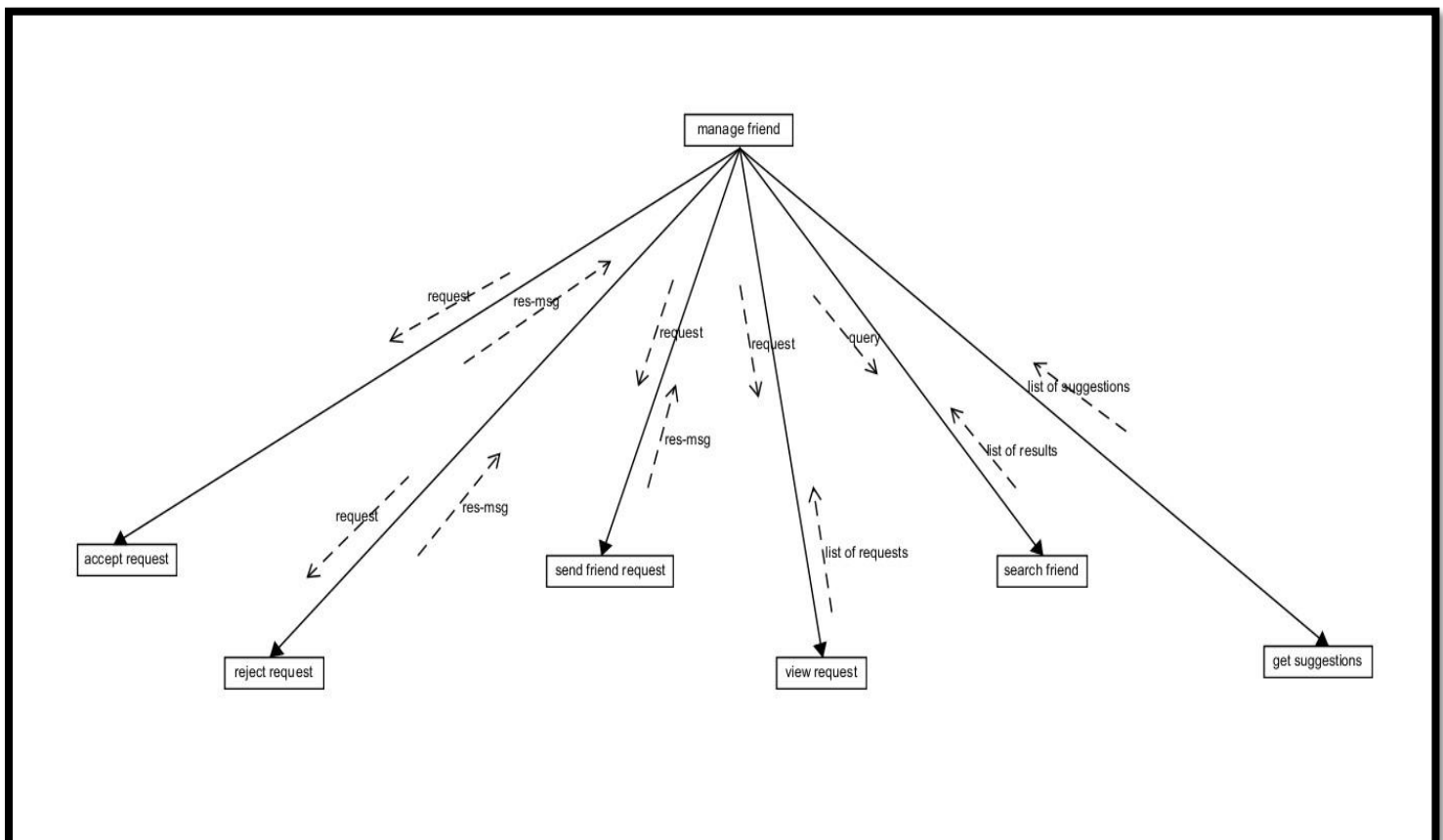
❖ Level – 2 Diagram for manage content



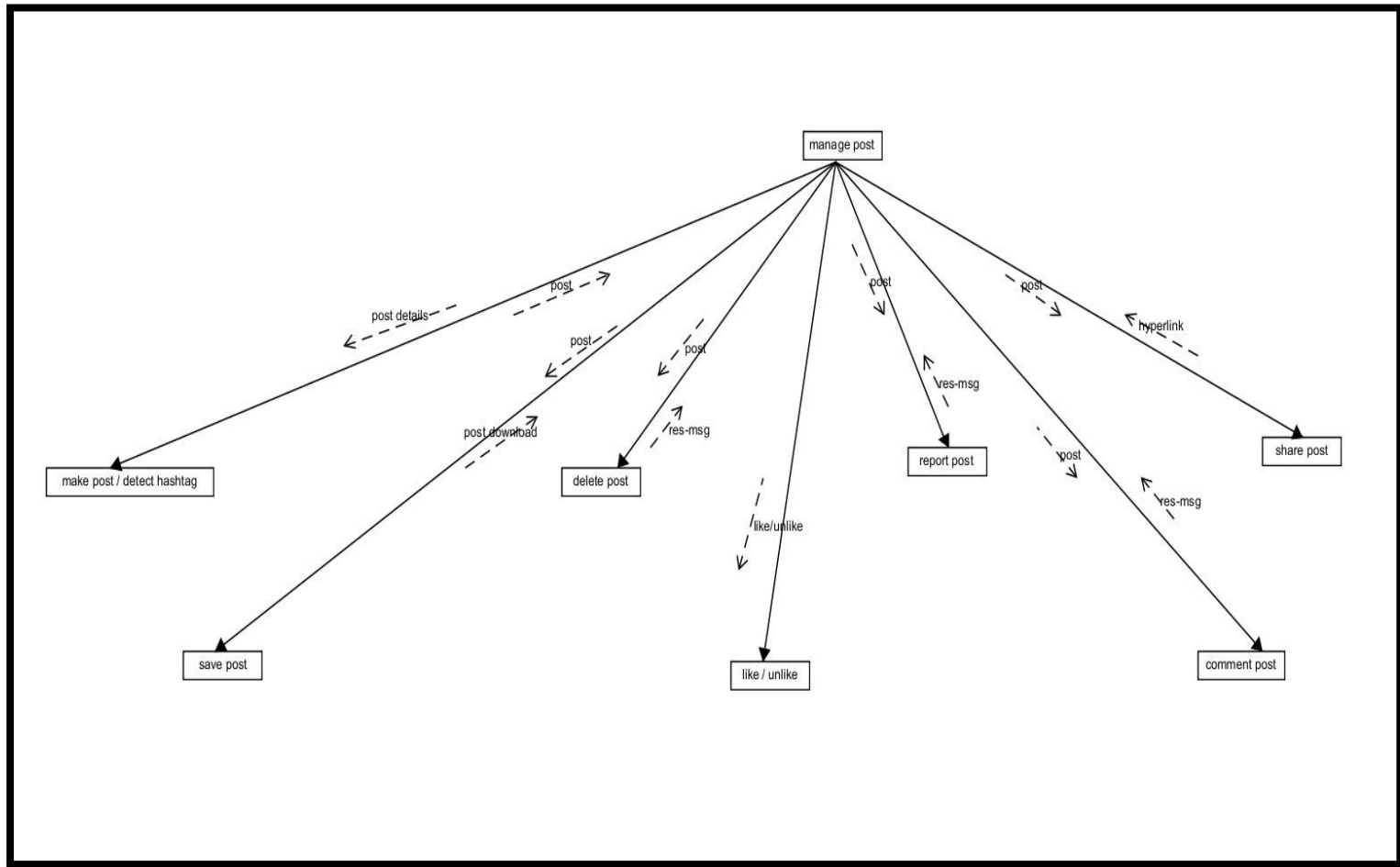
❖ Level – 2 Diagram for manage chat



❖ Level – 2 Diagram for manage friend

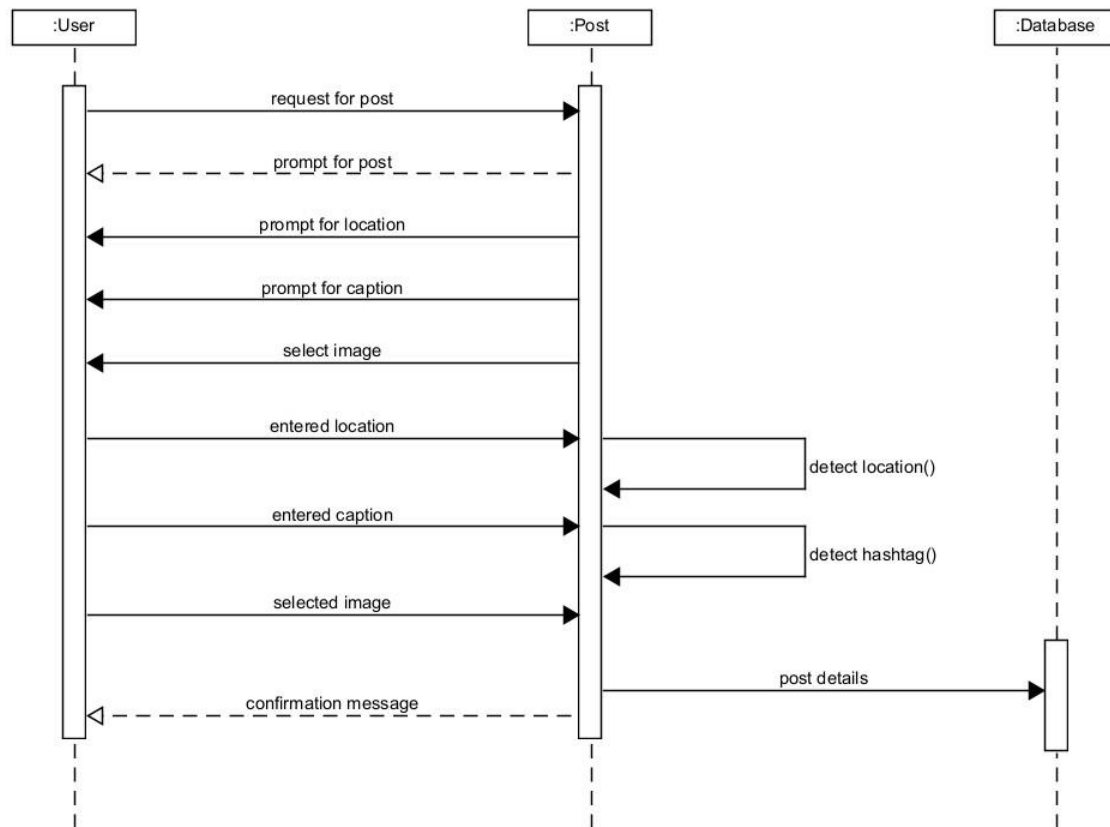


## ❖ Level – 2 Diagram for manage post

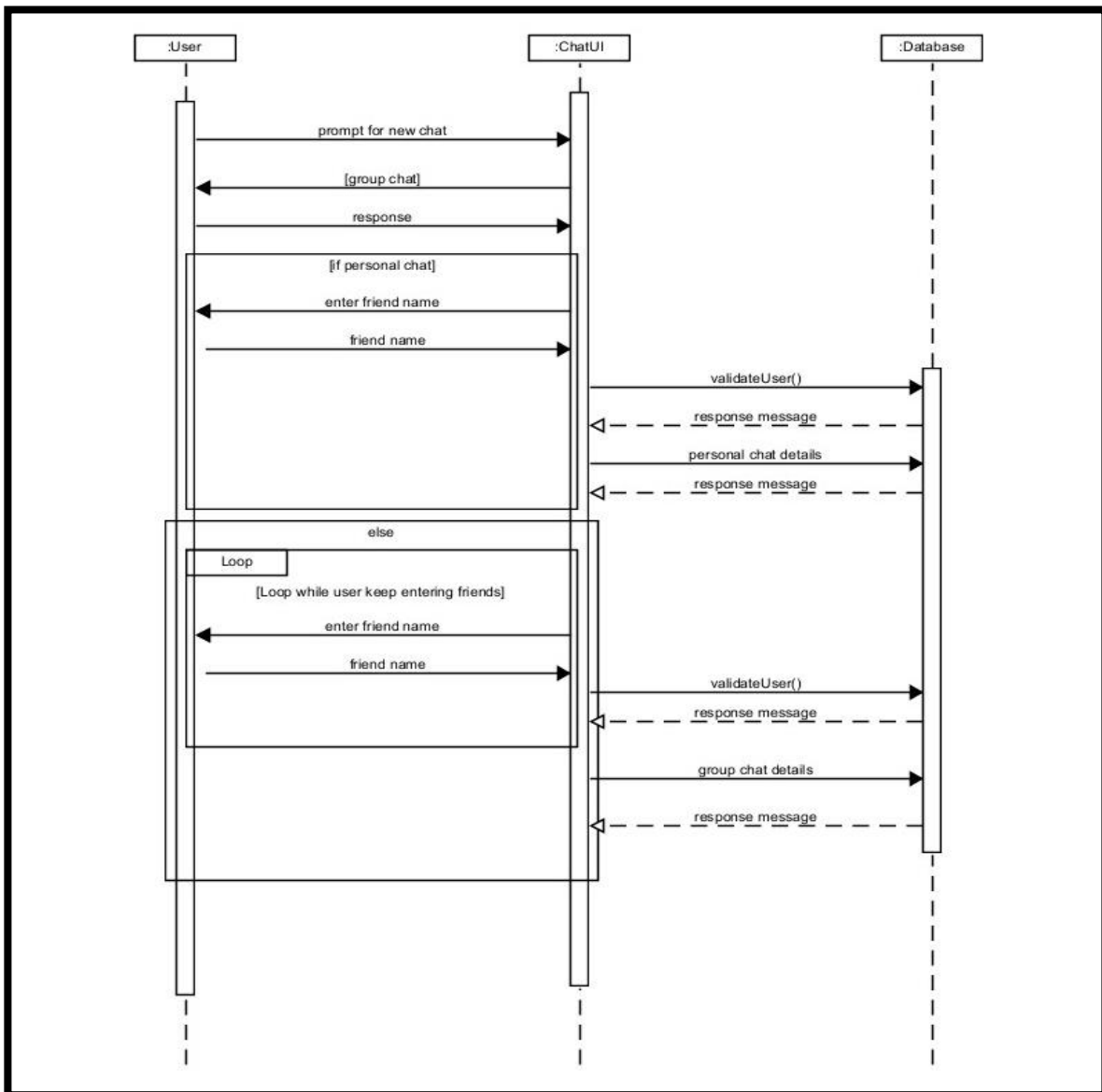


### 3.5) Sequence Diagram

- Use Case: Make Post

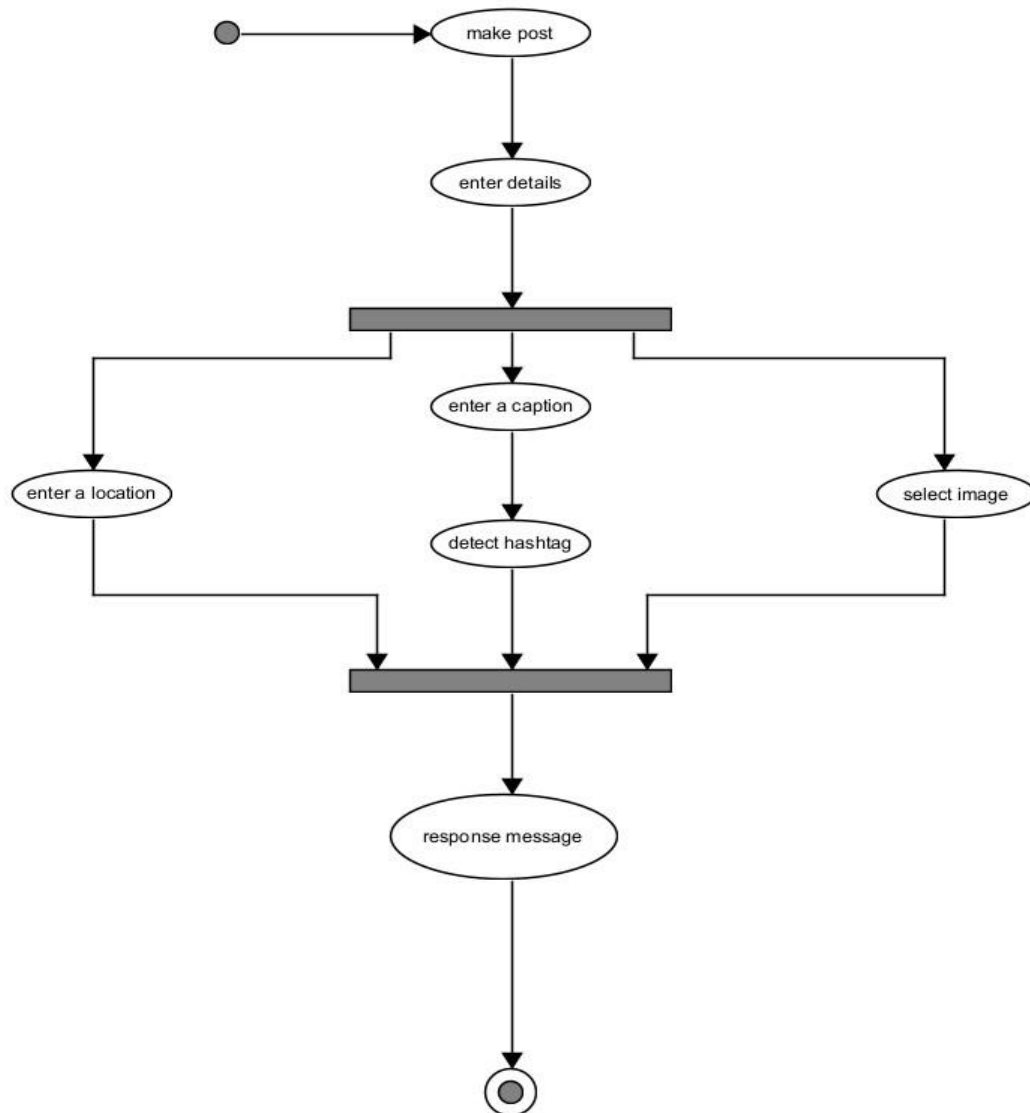


- Use Case: Start chat

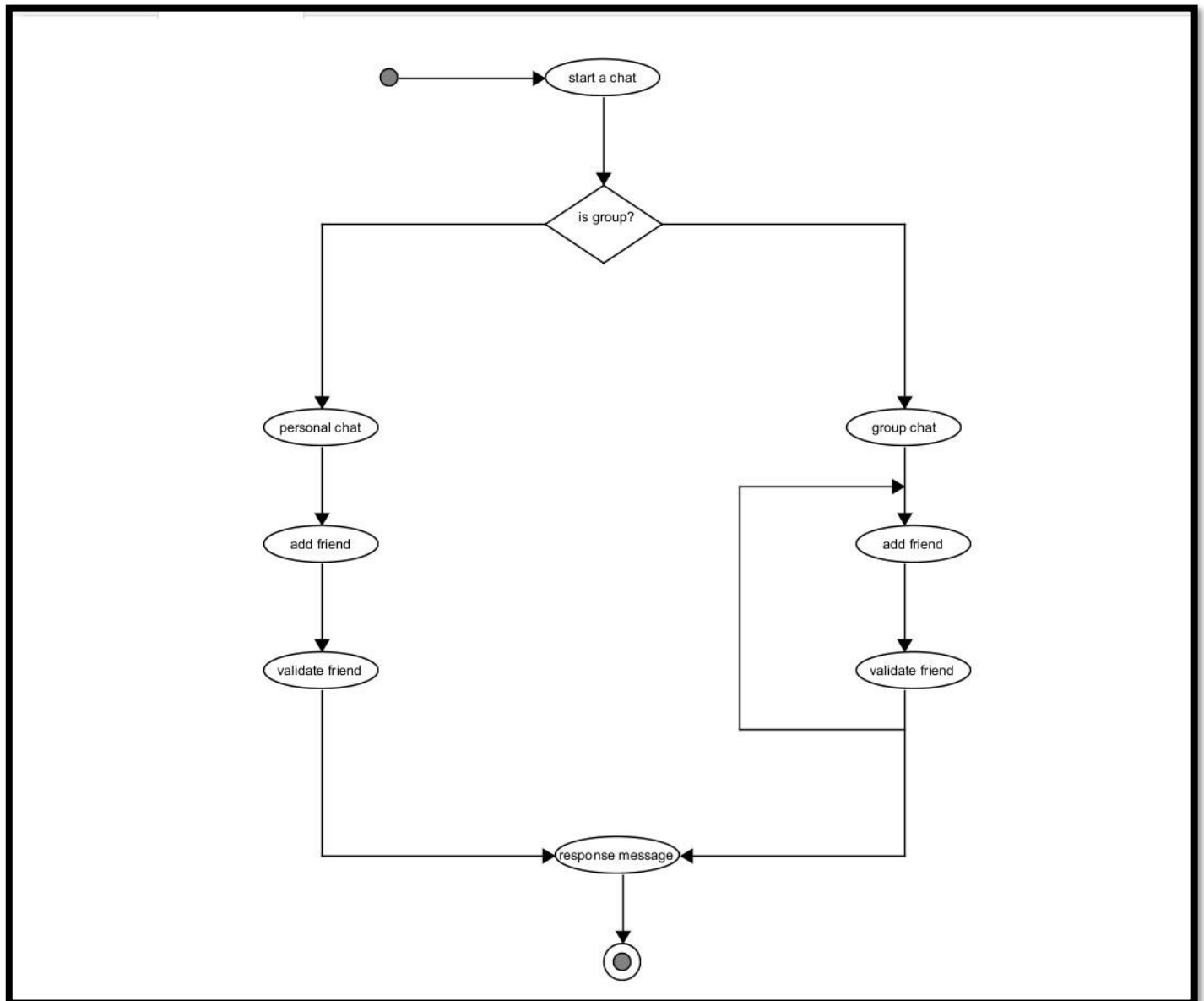


### 3.6) Activity Diagram

- Use Case: Make post



- Use Case: Start Chat




### 3.7) Data Dictionary

❖ Chatroom:



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	chat_room_id 🔑	int(11)			No	None		
2	is_group	tinyint(1)			No	None		
3	owner_id 🔑	int(11)			No	None		






❖ Message:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>chat_room_id</b>	int(11)			No	None		
2	<b>message</b>	longtext	utf8mb4_general_ci		No	None		
3	<b>date_time</b>	datetime(6)			No	None		
4	<b>is_read</b>	tinyint(1)			No	None		
5	<b>sender_id</b> 	int(11)			Yes	NULL		


❖ Subscriber:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b> 	int(11)			No	None		AUTO_INCREMENT
2	<b>chat_room_id</b>	int(11)			No	None		
3	<b>user_fk_id</b> 	int(11)			Yes	NULL		

❖ Comment:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b> 	int(11)			No	None		AUTO_INCREMENT
2	<b>comment_text</b>	longtext	utf8mb4_general_ci		No	None		
3	<b>date_added</b>	date			No	None		
4	<b>commenter_user_id</b> 	int(11)			No	None		
5	<b>post_id_id</b> 	int(11)			No	None		

❖ FriendRequest:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b> 	int(11)			No	None		AUTO_INCREMENT
2	<b>sender_username</b>	varchar(100)	utf8mb4_general_ci		No	None		
3	<b>receiver_username</b>	varchar(100)	utf8mb4_general_ci		No	None		
4	<b>date_received</b>	date			No	None		
5	<b>request_status</b>	tinyint(1)			No	None		
6	<b>date</b>	date			No	None		

❖ Like:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>like_id</b> 🔑	int(11)			No	None		AUTO_INCREMENT
2	<b>date_liked</b>	date			No	None		
3	<b>liker_user_id</b> 🔑	int(11)			No	None		
4	<b>post_id_id</b> 🔑	int(11)			No	None		

❖ Notification:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>notif_id</b> 🔑	int(11)			No	None		AUTO_INCREMENT
2	<b>notif_title</b>	varchar(100)	utf8mb4_general_ci		No	None		
3	<b>notif_msg</b>	longtext	utf8mb4_general_ci		No	None		
4	<b>date_added</b>	date			No	None		
5	<b>notify_to_id</b> 🔑	int(11)			No	None		

❖ Post:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>post_id</b> 🔑	int(11)			No	None		AUTO_INCREMENT
2	<b>photo_url</b>	varchar(200)	utf8mb4_general_ci		No	None		
3	<b>is_reported</b>	tinyint(1)			No	None		
4	<b>location</b>	varchar(30)	utf8mb4_general_ci		No	None		
5	<b>date_posted</b>	date			No	None		
6	<b>caption</b>	varchar(40)	utf8mb4_general_ci		No	None		
7	<b>user_fk_id</b> 🔑	int(11)			No	None		

❖ UploadImage:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b> 🔑	int(11)			No	None		AUTO_INCREMENT
2	<b>myimage</b>	varchar(100)	utf8mb4_general_ci		No	None		
3	<b>uploaded_at</b>	datetime(6)			No	None		
4	<b>author_id</b> 🔑	int(11)			No	None		

❖ UserSecretKey:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b> 🔑	int(11)			No	None		AUTO_INCREMENT
2	<b>date_valid</b>	datetime(6)			No	None		
3	<b>secret_key</b>	int(11)			No	None		
4	<b>user_fk_id</b> 🔑	int(11)			No	None		

❖ CustomUser:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b> 🔑	int(11)			No	None		AUTO_INCREMENT
2	<b>city_of_residence</b>	varchar(40)	utf8mb4_general_ci		No	None		
3	<b>country_of_residence</b>	varchar(40)	utf8mb4_general_ci		No	None		
4	<b>Image</b>	varchar(100)	utf8mb4_general_ci		No	None		
5	<b>bio</b>	longtext	utf8mb4_general_ci		No	None		
6	<b>user_inher_id</b> 🔑	int(11)			No	None		
7	<b>email</b>	varchar(150)	utf8mb4_general_ci		No	None		

## **4) Implementation Details**

---

### **4.1) Modules**

#### **❖ Manage Account:**

In this module, User can login, signup for the website. User can update profile, change password, reset password if forgotten, report user, logout.

#### **❖ Manage Post:**

In this module, User can post a photo with location and caption. User can like and comments on post, view post, save post, delete post, report post, share post.

#### **❖ Mange Friend:**

In this module, User can send friend request, get notifications of follow requests, see followers and followings. User having 10000 followers is considered as public account.

#### **❖ Manage Content:**

Admin can remove vulgar posts, sends warnings to user, remove hacked accounts, remove reported users, remove tagged comments.

#### **❖ Manage Chat:**

User can send messages to friends, make groups, get notifications of messages.

## 4.2) Major Functionalities

### 1. Rendering home page:

By this this function, user will be able to see his/her friend's post based on most recent post being displayed first. User can see all notifications, can see pending follow requests.

```
@login_required
def index(request):
    user_name = request.user
    posts_to_show = []
    posts_query = getPostByFollowings(user_name)
    reqs = getRequests(request.user)
    for i in posts_query:
        owner = i.user_fk.username
        location = i.location
        caption = i.caption
        date_posted = i.date_posted
        likedby = getLikesByPost(i, request.user)
        image_url = i.photo_url
        current_user = i.user_fk
        current_user_profile = customuser.objects.filter(
            user_inher=current_user)
        user_image_url = current_user_profile[0].Image.url
        # just send 3-4 comments over here and then make another app for viewing whole full page posts
        comments = [{ 'name': 'jenil', 'comment': 'Wow when did you go here'}, {
            'name': 'Kenil', 'comment': 'Take us also'}]
        quer = likes.objects.filter(post_id=i.post_id, liker_user=user_name)
        final_bool = False
        if len(quer) == 0:
            final_bool = False
        else:
            final_bool = True
        posts_to_show.append({'owner': owner, 'location': location, 'caption': caption, 'date': date_posted, 'likedby': likedby,
            'image_url': image_url, 'post_id': i.post_id, 'poster_image_url': user_image_url, 'comments':
            getCommentsByPosts(i), 'isliked': final_bool})
    user_details_dict = {
        'name': str(user_name.username),
        'posts': posts_to_show,
        'notif': get_notifs(request.user),
        'pendings': reqs,
    }
    return render(request, 'home/home.html', user_details_dict)
```

### 2. Add Post:

In this function, user would be able to add the post with photo, caption, location.

```

# add post
def add_post(request):
    if request.method == 'POST':
        form = ImageFrom(request.POST, request.FILES)
        if form.is_valid():
            img = form.cleaned_data['Image']
            caption = form.cleaned_data['Caption']
            location = form.cleaned_data['Location']
            cur_user = request.user
            print(cur_user)
            timestamp = str(datetime.timestamp(datetime.now()))
            path = default_storage.save(
                'home/posts_images/'+timestamp+'.jpg', ContentFile(img.read()))
            print(type(path))
            user_post = post(user_fk=cur_user, photo_url=path, is_reported=False,
                             location=location, date_posted=datetime.date(datetime.now()),
                             caption=caption)
            user_post.save()
            tmp_file = os.path.join(settings.MEDIA_ROOT, path)
            return redirect("http://localhost:8000/home")
        else:
            print("invalid")
            return HttpResponse("YAY")

    else:
        form = ImageFrom()
    return render(request, 'home/add_post_1.html', {'form': form})

```

### 3. See trending's:

In this function, User would be able to see trending's post based on number of likes of that post on that day. User can search for other users.



```

def index(request):
    year=datetime.datetime.now().year
    day=datetime.datetime.now().day
    month=datetime.datetime.now().month
    today=str(year)+"-"+str(check_datetime(month))+"-"+str(check_datetime(day))
    x = likes.objects.raw(
        "Select like_id,post_id_id, count(post_id_id) as cpid From home_likes where date_liked='"+today
        +"'" Group by post_id_id Order by count(post_id_id) DESC;")
    print(len(x))
    post_details={}
    post_details['posts']=[]
    for i in x:
        post_tmp={}
        posts=post.objects.filter(post_id=i.post_id.post_id)[0]
        post_tmp['url']="http://localhost:8000/media/"+posts.photo_url
        post_tmp['likes']=len(likes.objects.filter(post_id=posts.post_id))
        post_tmp['comments']=getCommentsCount(i.post_id.post_id)
        # post_tmp["post_id"]=posts.post_id
        post_details['posts'].append(post_tmp)
    return render(request, "search/search.html",post_details)

```

#### 4. Forgot Password:

In this function, if user forgets his/her password, then reset link would be sent on the registered email-id of the user from local SMTP port.

```

def forgot_password(request):
    if request.method=="POST":
        username=request.POST["username"]
        print(username)
        subject, from_email, to = 'Reset Password for SocialHub', 'computerdummy960@gmail.com',
        'confusedjogger01@gmail.com'
        text_content = 'This is an important message.'
        html_content = 'Click <a href="http://localhost:8000">here</a> to reset your password'
        msg = EmailMultiAlternatives(subject, text_content, from_email, [to])
        msg.attach_alternative(html_content, "text/html")
        msg.send()
        return HttpResponse("An email has sent to the registered email ID")
    return render(request, 'login/forgot_password.html')

```

#### 5. Get Private chatroom:

This function returns the chatroom id between two particular users. If chatroom already exists then it returns the chatroom id, otherwise it creates a new chatroom and returns id of the same.

```
def getPrivateChatRoom(user1, user2):
    subs_of_1 = Subscriber.objects.filter(user_fk=user1)
    subs_of_2 = Subscriber.objects.filter(user_fk=user2)
    final_room = []
    for i in subs_of_1:
        for j in subs_of_2:
            if i.chat_room_id == j.chat_room_id:
                final_room.append(i.chat_room_id)
    # This logic needs to be changed for generic groups
    print(final_room)
    if len(final_room)==0:
        print("in if")
        chat_id=len(ChatRoom.objects.all())
        new_chat_room=ChatRoom(owner=user1,chat_room_id=chat_id+1)
        new_chat_room.save()
        subs_obj1=Subscriber(user_fk=user1,chat_room_id=chat_id+1)
        subs_obj2=Subscriber(user_fk=user2,chat_room_id=chat_id+1)
        subs_obj1.save()
        subs_obj2.save()
        print(subs_obj1,subs_obj2)

        final_room.append(chat_id+1)
        print(final_room)
    return final_room[0]
```



## **5) Conclusion**

---

- ❖ In this project, we have successfully implemented all basic and intermediate functionality required for social media website.
- ❖ We had also made successful implementation of web socket using Django channels. Also, we made our website from working on WSGI to ASGI (Asynchronous Gateway Interface) required for using web socket.
- ❖ We gave users the notifications about each and every event happening in his account like comments, likes, friend requests, etc.
- ❖ We also had functionality about finding trending posts based on post's likes. We also gave an insight to the user about the number of likes on his post, comments on his post, monthly gained followers.
- ❖ All this would not have been possible without the immense guidance from faculties.

## **6) Limitations and Future Extensions**

---

### **❖ Limitations:**

- All modules are hosted on single server, so website is not scalable and requires further optimization.
- Unread messages are currently not supported.
- The chat section is not encrypted and message flows in its normal form.
- The chat section also does not support sharing of multimedia files; however, it is done through html formatting.

### **❖ Future Extensions:**

- There is no classifier that classify vulgar images but it could be integrated in future extensions.
- The chat feature would be enabled with multimedia.
- Functionality about post like report post, edit post would be added in future extension.
- In business feature, the target advertising would be implemented using machine learning algorithms.
- In business feature, live post like tracking would be done using web socket.
- This project could also have developers' side where one can contact other senior developers.
- We have applied manual testing of project for now, but in future extension selenium testing would be applied.

## **7) Bibliography**

---

### ❖ References/resources used for developing project

- <https://github.com/>
- <https://docs.djangoproject.com/en/3.1/>
- <https://channels.readthedocs.io/en/stable/>
- <https://stackoverflow.com/>
- <https://www.w3schools.com/>
- <https://codepen.io/>
- <https://fontawesome.com/start>
- <https://getbootstrap.com/docs/5.0/getting-started/introduction/>
- <https://jquery.com/>
- <https://www.chartjs.org/>
- <https://selenium-python.readthedocs.io/>
- <https://www.jenkins.io/doc/tutorials/>