DHARMSINH DESAI UNIVERSITY, NADIAD

FACULTY OF TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

B. TECH. CE SEMESTER – IV

SUBJECT: SOFTWARE PROJECT

PROJECT TITLE : **CHATTING WEB APPLICATION**

BY:

1) SANJAYKACHCHHAVA ROLL NO: CE053
2) ARSHITKAKADIYA, ROLL NO: CE054

GUIDED BY: PROF. PINKAL CHAUHAN

PROF. BRIJESH BHATT

PROF. JIGAR PANDYA

Contents:

1. Abstract

Teleconferencing or Chatting is a method of using technology to bring people and ideas together despite of the geographical barriers.This technology has been developed recently in decade.Our project Is an example of a chat server.To start chatting client should get connected to server where they can do private and group chat with security.Security measures were taken seriously.It enables users to communicate in real time using simply accessible web interface.it is ind of web online chat distinguished by its simplicity and accessibility to users who don’t want to install and learn to use specialized chat software.

In Our app user can login and send friend request to other registeres user and can easily chat with them.User can chat in group by creating it and using friends as its members.It I siple chat app many other features also can be added to it.It is a good way of communication with others.

2. Introduction

2.1 Brief Introduction

This project is to create a app that helps people to connect with each other using web-appication.This connection will be set up using chat application.This chat application enables the users to chat with each others.It is a instant messaging facility.Project features should be very simple so that a non-technical person can be able to understand it.User can be able to enter and use the application creating account on the app.While creating account user must provide some basic information like email ID , username so that it can be useful aunthitication and it must be editable whenever user want. Application should provide facility to send requests to the user with whom he/she want to chat and person should be able to chat with the person only if the other person accept request.

2.2 Tools/Technologies Used

Technologies:

- Django
- Python
- MySQL
- JavaScript
- HTML
- jQuery
- CSS
- Redis
- Django Channels
- Websocket

Tools:

- Git
- IDE

Platform:

- Local development server

3. Software Requirement Specifications

1.Manage User:

R.1.1 : Login Account

- Description : User can login into his/her account
- Input : choose a option login (Email and Password)
- Output : Confirmation Message ("Successfully logged in")

R.1.2 : Create Account

- Description : User can create a new account
- Input : Information like Email-id, Username, Password
- Output : Confirmation Message("Account Created")

R.1.3 : Log out

- Description : User can logout from his/her account
- Input : choose a option log out
- Output : Confirmation Message("Logged out")

R.1.4 : Forgot Password?

- Description : In any case If user forgot password
- Input : choose option "forgot password"
- Output : Get a mail of your password

2.Manage Account :

R.2.1 : Edit Profile

- Description : User can edit the information about him/her
- Input : New Information
- Output : Confirmation Message(" Profile updated")

R.2.2 : Visibility of Account

- Description : User can hide his/her email from others
- Input : Choose option
- Output : Confirmation Message(" Account updated")

R.2.3 : Change Password

- Description : User can change the paasword
- Input : Current password,New password,Confirm new password
- Output : Confirmation Message(“ Password chnaged”)

3.Manage Friends :

R.3.1 : Add Friend

- Description : Send the request to the another user
- Input : Choose the option
- Output : Confirmation Message (“Request sent”)

R.3.2 : Request Deny/Accept

- Description : We can deny or accept the request from others if has been sent
- Input : Choose the option
- Output : Confirmation Message (“Request Accepted/Denied”)

R.3.3 : Remove from Friend list

- Description : User can remove the the user from his/her friend list
- Input : Choose the option
- Output : Confirmation Message (“Removed from Friend List”)

R.3.4 : Search Friends

- Description : User can search new Friends
- Input : The username or email of friend to be searched
- Output : User search results if Existed otherwise “No User Found”

4.Manage Chat :

R.4.1 : 1-1 Private Chat

- Description : User can chat one to one with the other user
- Input : Choose Option
- Output : “Start Conversation”

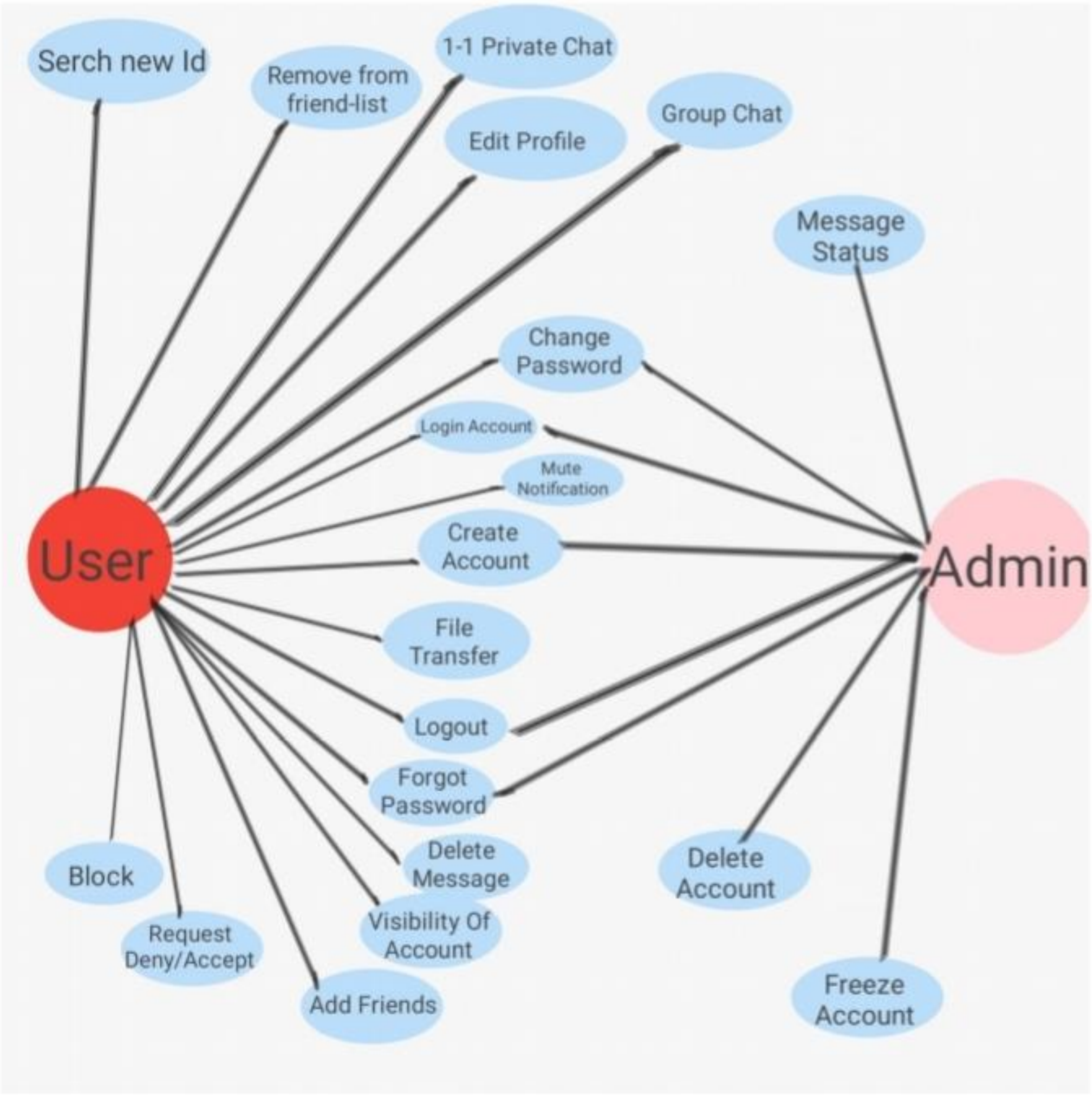
R.4.2 : Create Group

- Description : User can create group
- Input : Choose Option for creating group and chose the friends to be in group

- Output : Confirmation Message(“Group Created”)

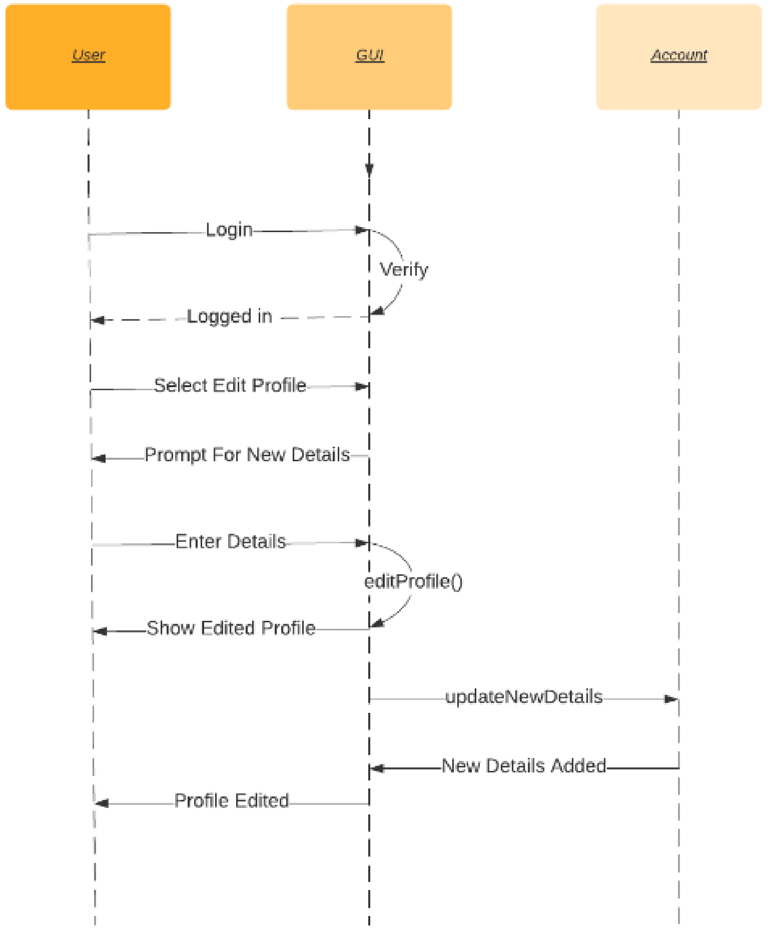
4. Design

4.1 Use Case Diagram

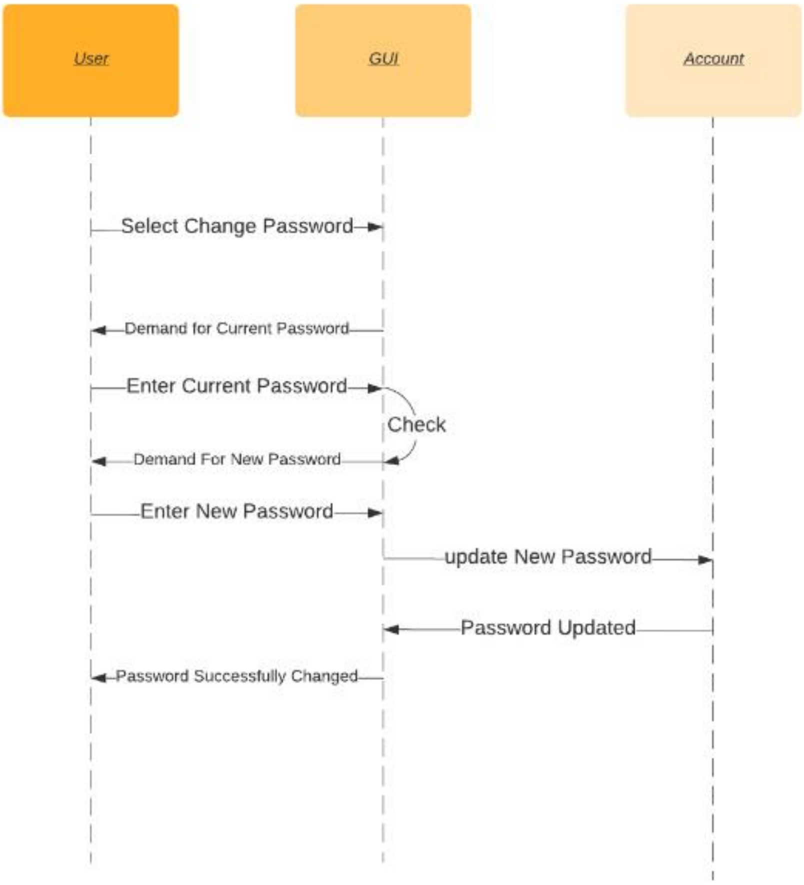


4.2 Sequence diagram

- **Use case:** **Make donation**

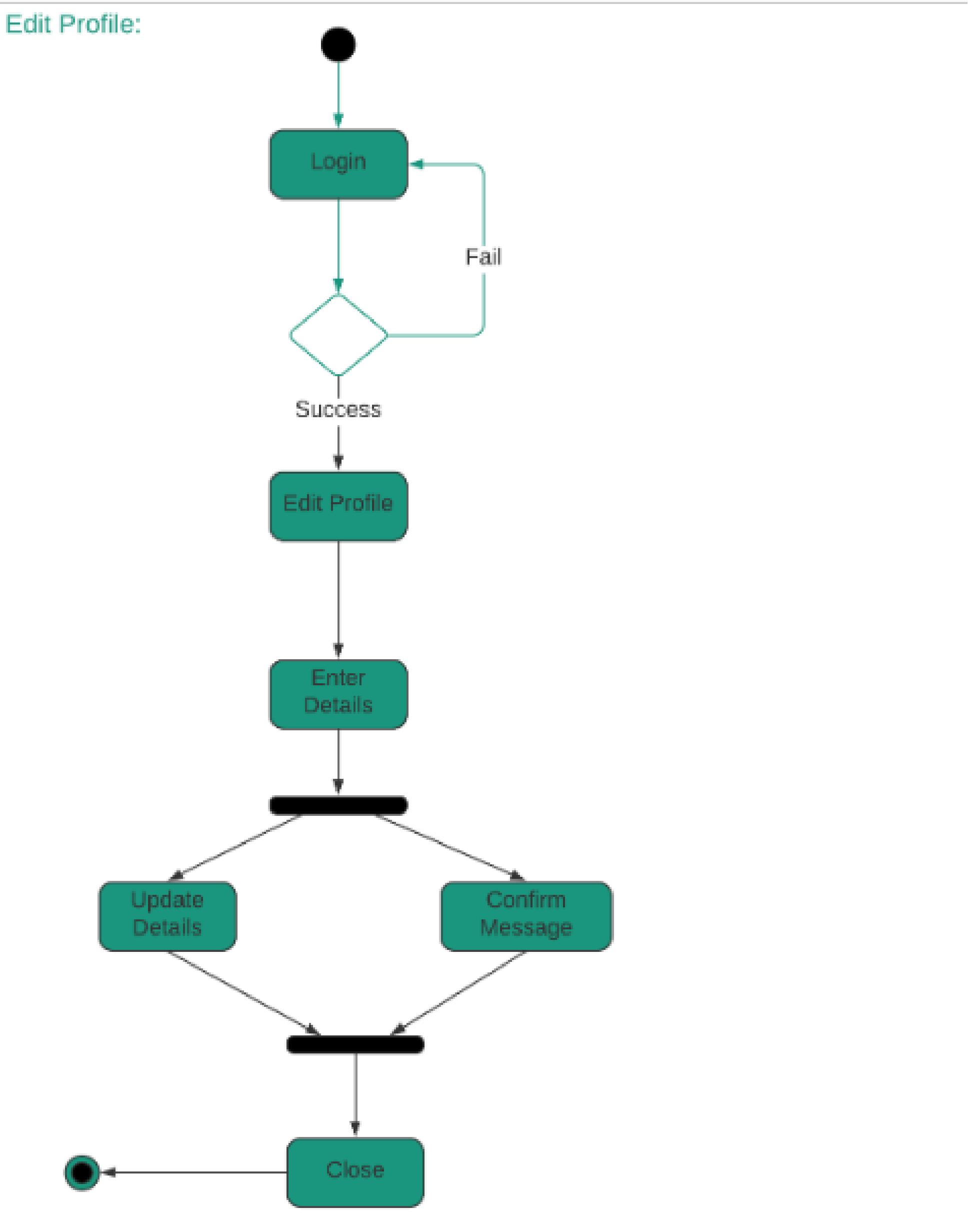


- **Use Case : Change password**

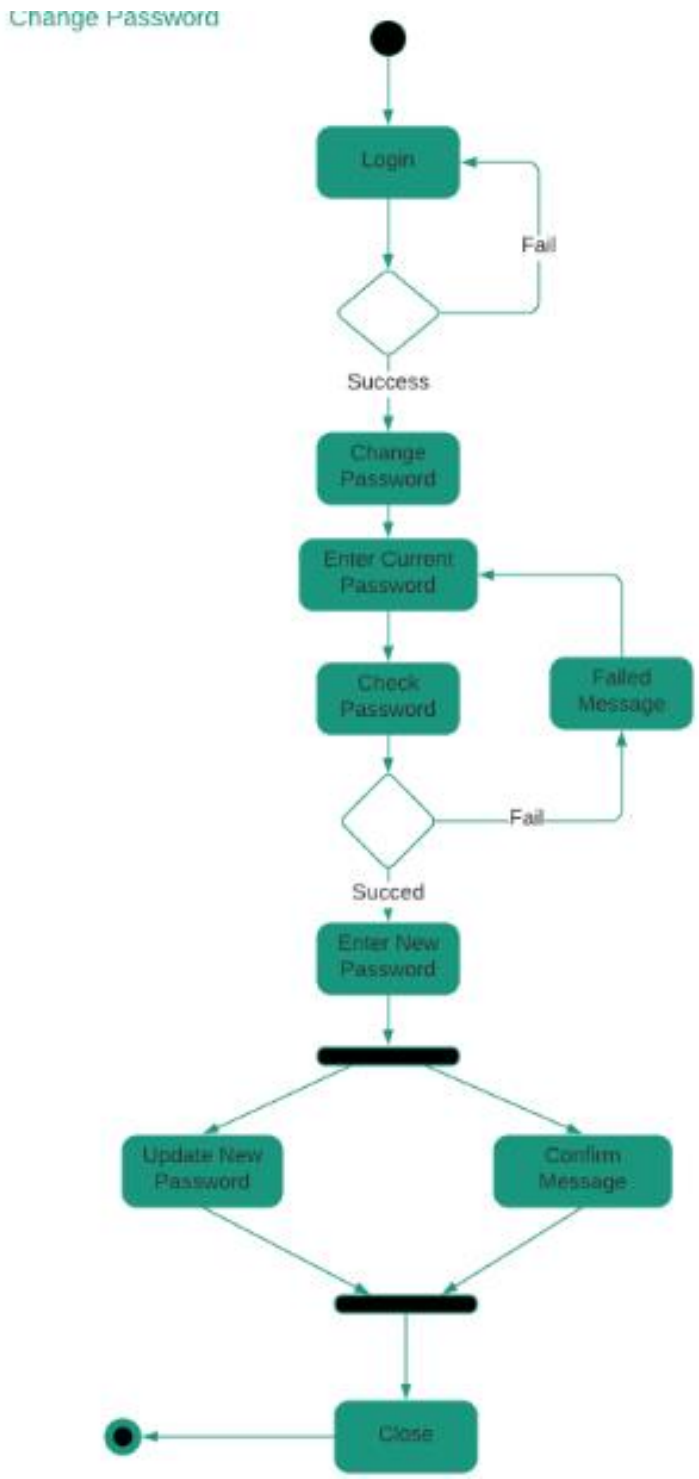


4.3 Activity diagram

- **Use case : Make donation**

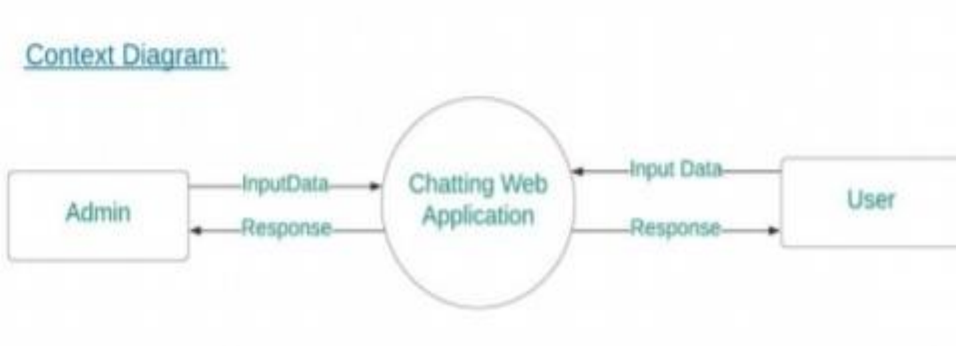


- **Use case : Change Password**

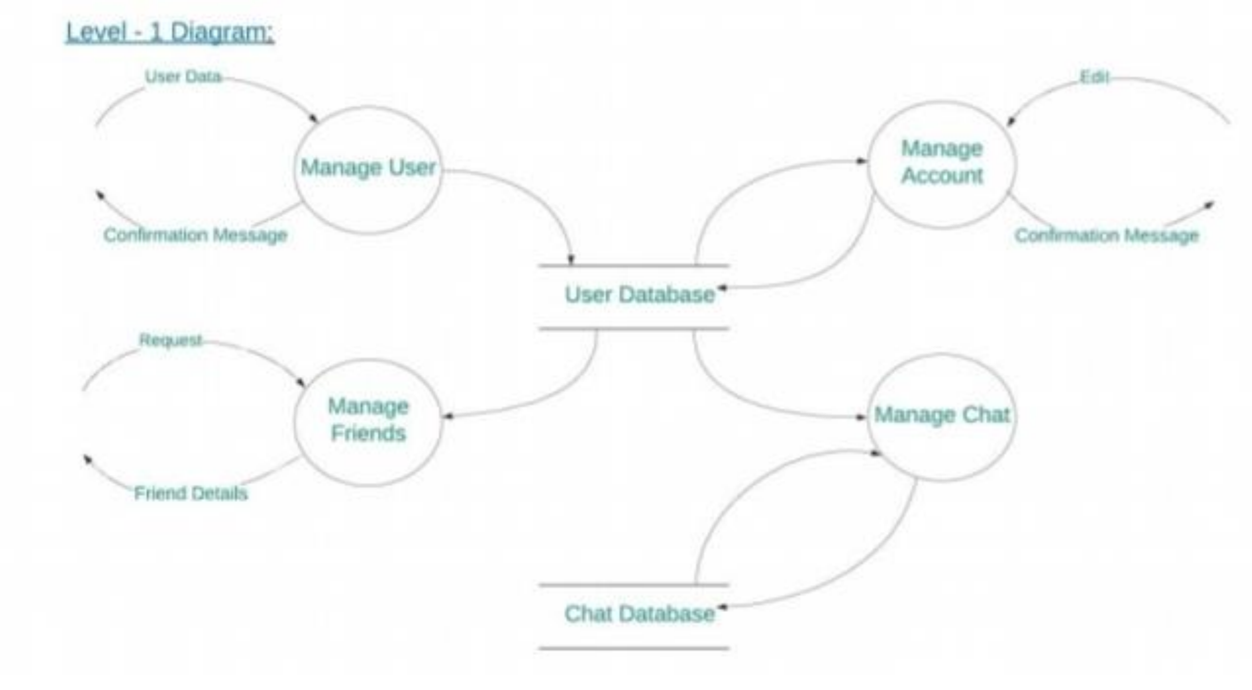


4.4 Data Flow diagram

- · **Context diagram:**

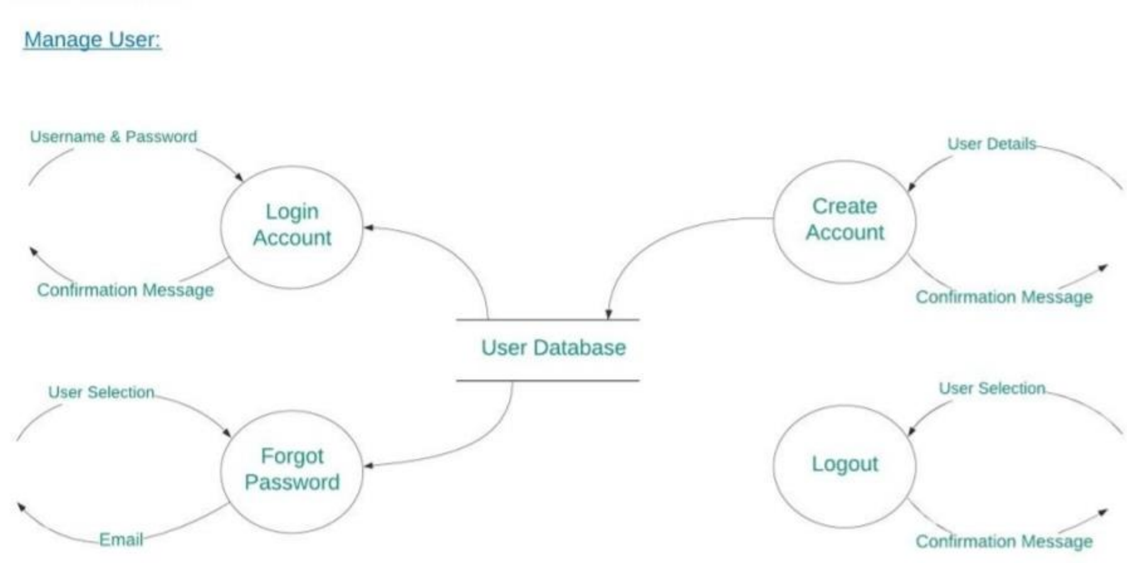


- **Level 1 diagram:**

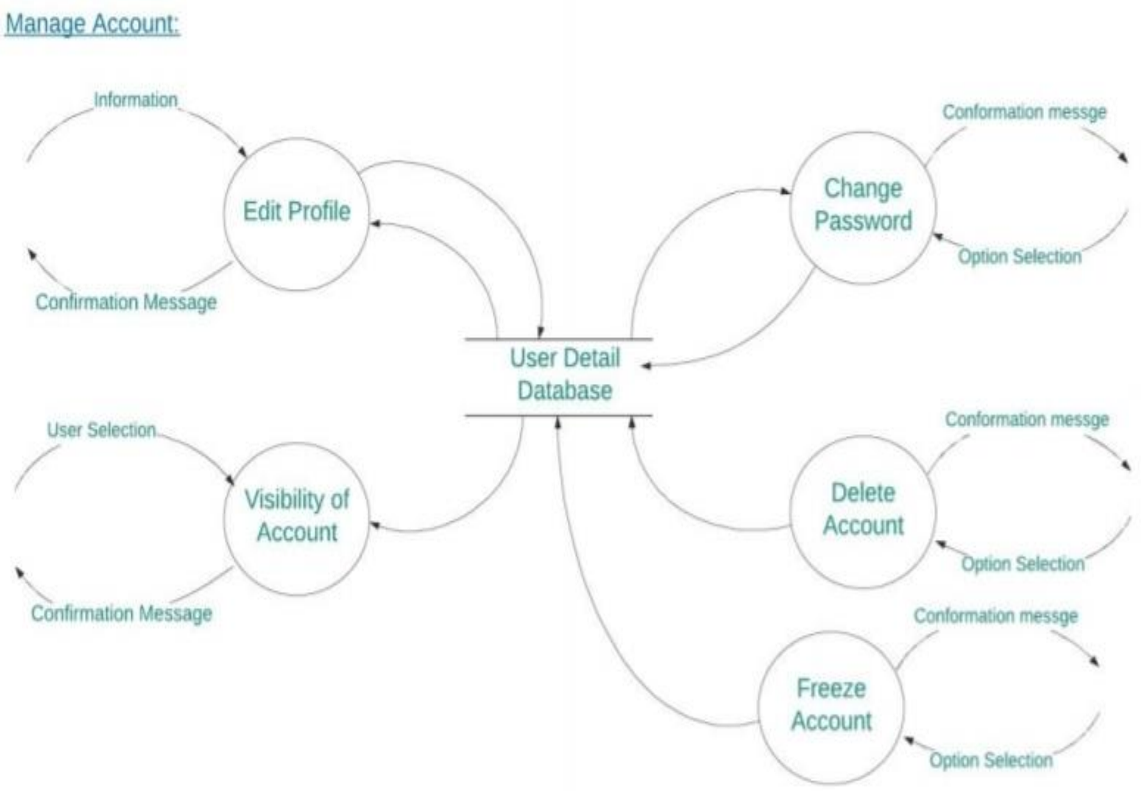


- · **Level 2 diagrams:**

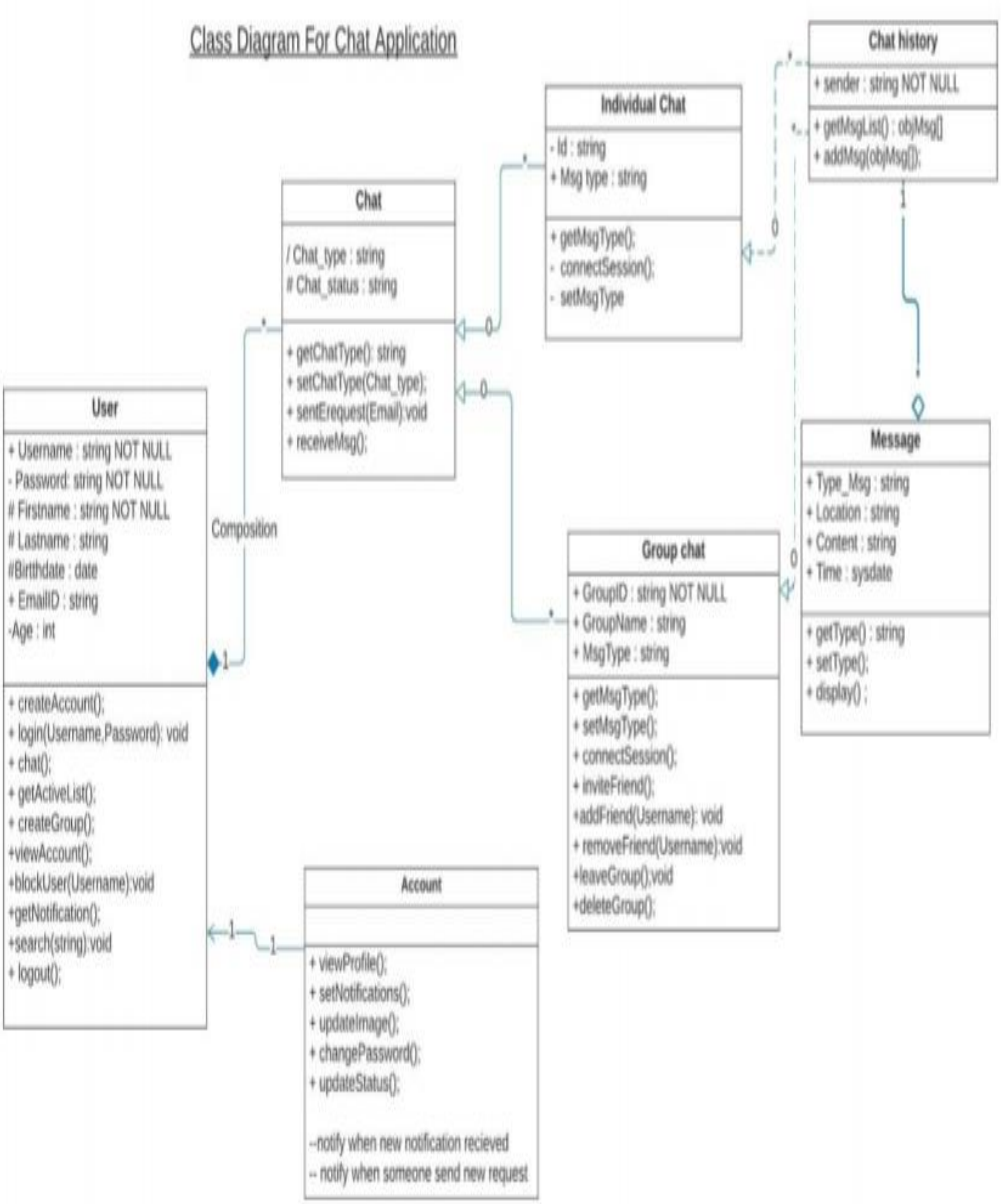
  - ➢ **Manage User:**



- **Manage Account:**



4.5 Class diagram



5. Implementation Details

5.1 . Modules :

- **Sign up/Register Module :**

  Guest users can register themselves to use this app by this function.User is asked to enter Username , Email Id , Password.

- **Login User Module :**

  Users or Admin will login into their account by Login function.

- **Friend Module :**

  In this module user is able to see the friend list and user can search user on the basis of the name and also on the basis of email.User can send request and also accept it from other Users.

- **Chat Module :**

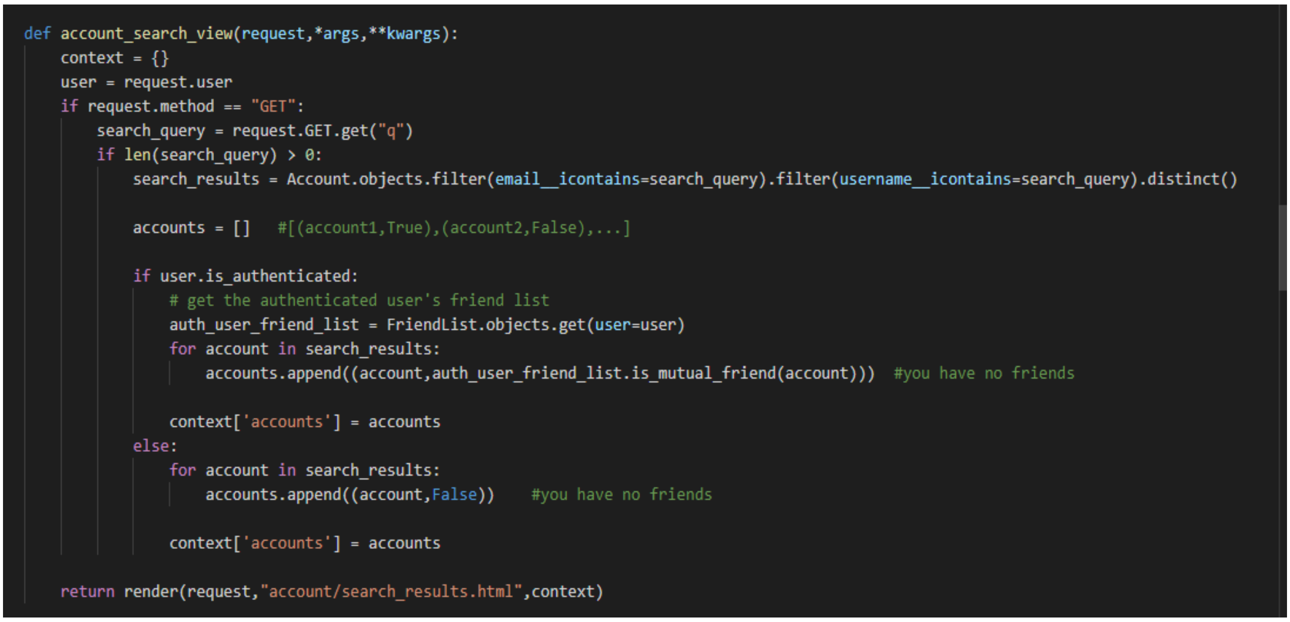
  In this module user can chat with other friends.User can be able to see all Users with whom user has interacted.User can create chat groups too.

- **Account Module :**

  Account module shows some important details of user.User's username and profile picture can be changed with this module.

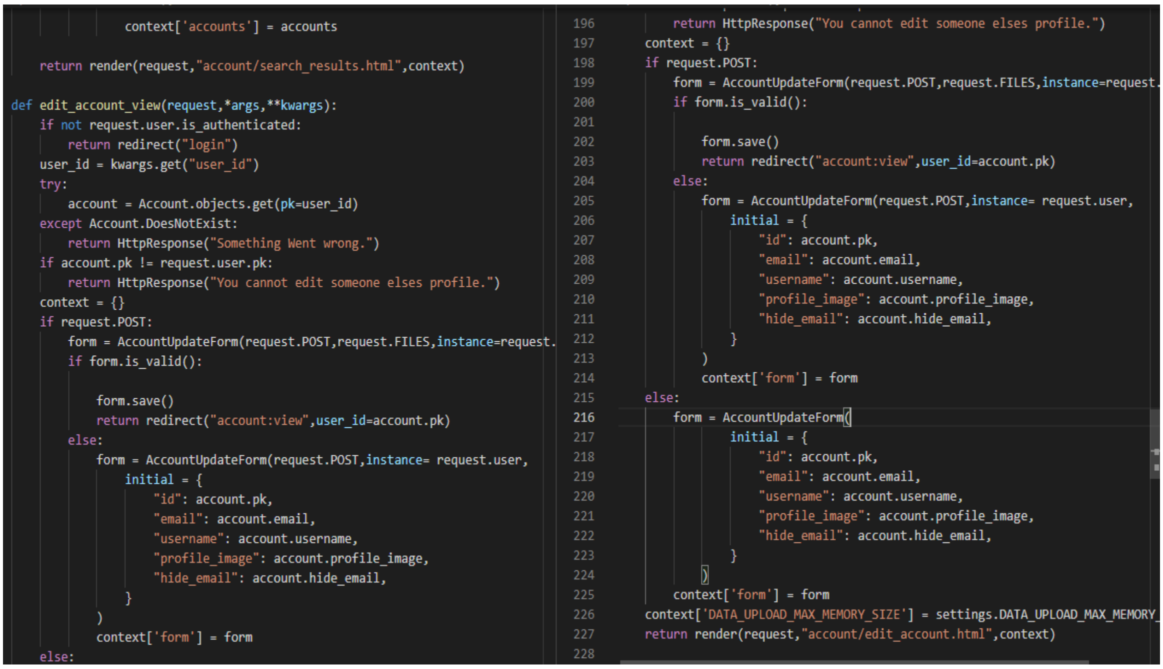
5.2 Functionalities :

- **Search Friend :** This functionality proides us feature to search any friend/User.



```python
def account_search_view(request,*args,**kwargs):
    context = {}
    user = request.user
    if request.method == "GET":
        search_query = request.GET.get("q")
        if len(search_query) > 0:
            search_results = Account.objects.filter(email__icontains=search_query).filter(username__icontains=search_query).distinct()

            accounts = []   #[(account1,True),(account2,False),...]

            if user.is_authenticated:
                # get the authenticated user's friend list
                auth_user_friend_list = FriendList.objects.get(user=user)
                for account in search_results:
                    accounts.append((account,auth_user_friend_list.is_mutual_friend(account)))  #you have no friends

                context['accounts'] = accounts
            else:
                for account in search_results:
                    accounts.append((account,False))    #you have no friends

                context['accounts'] = accounts

    return render(request,"account/search_results.html",context)
```

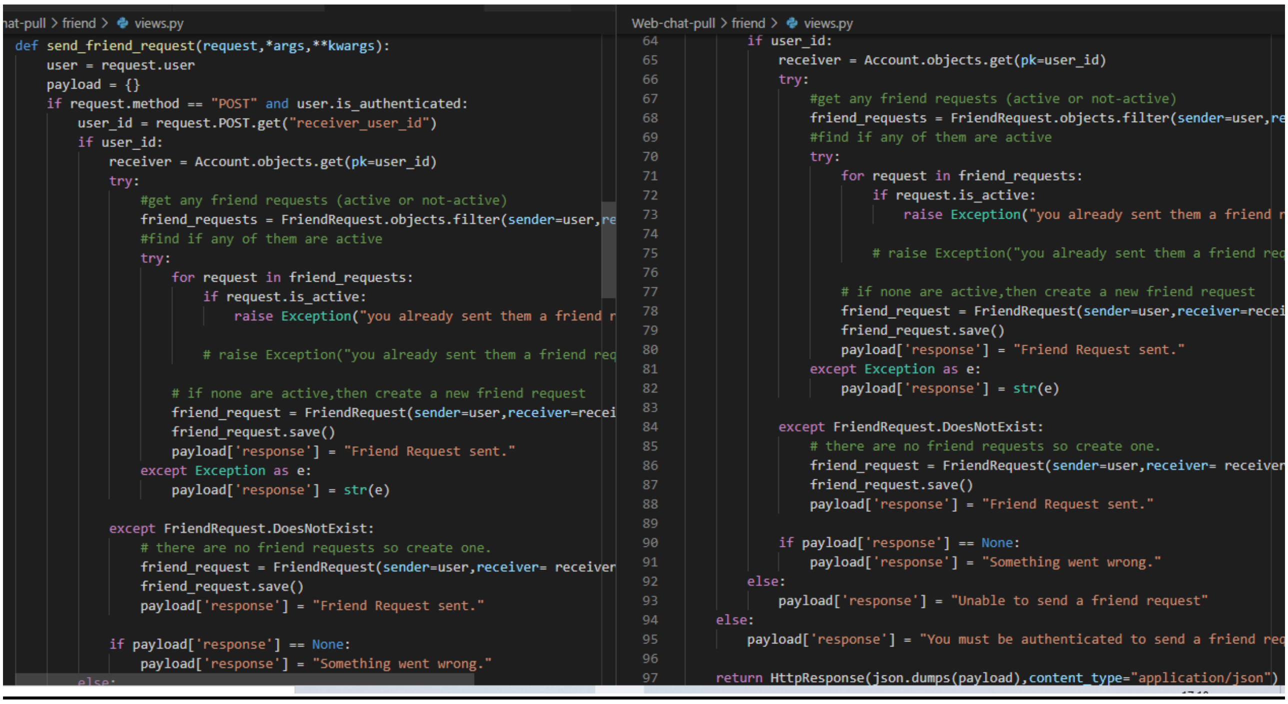
- **Edit Account :**



```python
                context['accounts'] = accounts

    return render(request,"account/search_results.html",context)

def edit_account_view(request,*args,**kwargs):
    if not request.user.is_authenticated:
        return redirect("login")
    user_id = kwargs.get("user_id")
    try:
        account = Account.objects.get(pk=user_id)
    except Account.DoesNotExist:
        return HttpResponse("Something Went wrong.")
    if account.pk != request.user.pk:
        return HttpResponse("You cannot edit someone elses profile.")
    context = {}
    if request.POST:
        form = AccountUpdateForm(request.POST,request.FILES,instance=request.
        if form.is_valid():

            form.save()
            return redirect("account:view",user_id=account.pk)
        else:
            form = AccountUpdateForm(request.POST,instance= request.user,
                initial = {
                    "id": account.pk,
                    "email": account.email,
                    "username": account.username,
                    "profile_image": account.profile_image,
                    "hide_email": account.hide_email,
                }
            )
            context['form'] = form
    else:
```

```python
        return HttpResponse("You cannot edit someone elses profile.")
    context = {}
    if request.POST:
        form = AccountUpdateForm(request.POST,request.FILES,instance=request.
        if form.is_valid():

            form.save()
            return redirect("account:view",user_id=account.pk)
        else:
            form = AccountUpdateForm(request.POST,instance= request.user,
                initial = {
                    "id": account.pk,
                    "email": account.email,
                    "username": account.username,
                    "profile_image": account.profile_image,
                    "hide_email": account.hide_email,
                }
            )
            context['form'] = form
    else:
        form = AccountUpdateForm(
                initial = {
                    "id": account.pk,
                    "email": account.email,
                    "username": account.username,
                    "profile_image": account.profile_image,
                    "hide_email": account.hide_email,
                }
            )
        context['form'] = form
    context['DATA_UPLOAD_MAX_MEMORY_SIZE'] = settings.DATA_UPLOAD_MAX_MEMORY_
    return render(request,"account/edit_account.html",context)

```

- ## Send Friend Request :



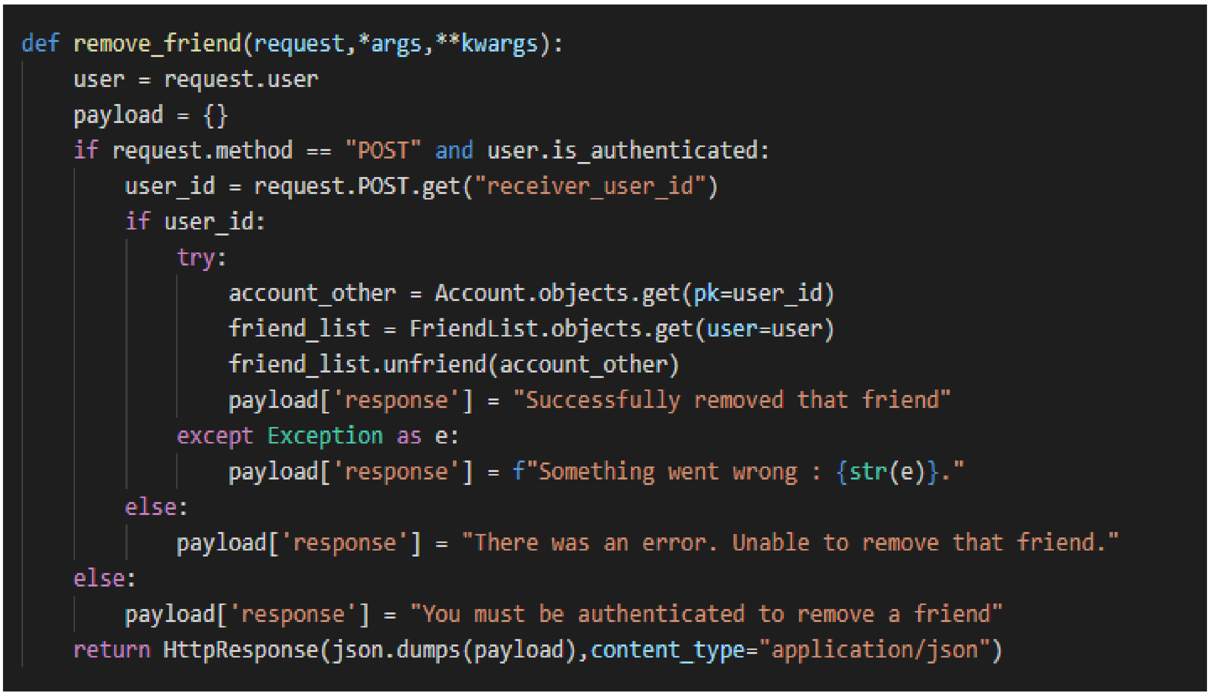
nat-pull > friend > views.py

```
def send_friend_request(request,*args,**kwargs):
    user = request.user
    payload = {}
    if request.method == "POST" and user.is_authenticated:
        user_id = request.POST.get("receiver_user_id")
        if user_id:
            receiver = Account.objects.get(pk=user_id)
            try:
                #get any friend requests (active or not-active)
                friend_requests = FriendRequest.objects.filter(sender=user,re
                #find if any of them are active
                try:
                    for request in friend_requests:
                        if request.is_active:
                            raise Exception("you already sent them a friend r

                        # raise Exception("you already sent them a friend req

                    # if none are active,then create a new friend request
                    friend_request = FriendRequest(sender=user,receiver=recei
                    friend_request.save()
                    payload['response'] = "Friend Request sent."
                except Exception as e:
                    payload['response'] = str(e)

            except FriendRequest.DoesNotExist:
                # there are no friend requests so create one.
                friend_request = FriendRequest(sender=user,receiver= receiver
                friend_request.save()
                payload['response'] = "Friend Request sent."

            if payload['response'] == None:
                payload['response'] = "Something went wrong."
        else:
```

Web-chat-pull > friend > views.py

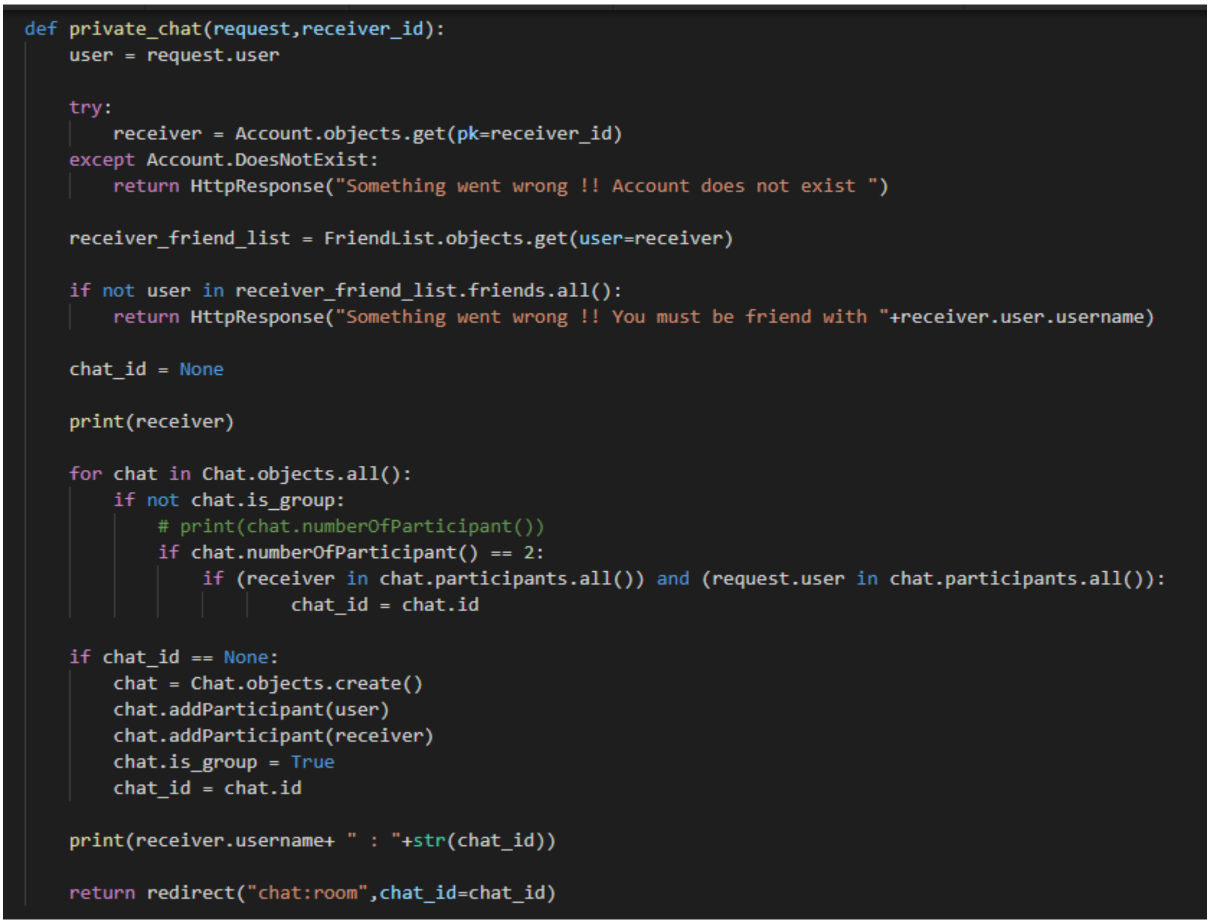
```
        if user_id:
            receiver = Account.objects.get(pk=user_id)
            try:
                #get any friend requests (active or not-active)
                friend_requests = FriendRequest.objects.filter(sender=user,re
                #find if any of them are active
                try:
                    for request in friend_requests:
                        if request.is_active:
                            raise Exception("you already sent them a friend r

                        # raise Exception("you already sent them a friend req

                    # if none are active,then create a new friend request
                    friend_request = FriendRequest(sender=user,receiver=recei
                    friend_request.save()
                    payload['response'] = "Friend Request sent."
                except Exception as e:
                    payload['response'] = str(e)

            except FriendRequest.DoesNotExist:
                # there are no friend requests so create one.
                friend_request = FriendRequest(sender=user,receiver= receiver
                friend_request.save()
                payload['response'] = "Friend Request sent."

            if payload['response'] == None:
                payload['response'] = "Something went wrong."
        else:
            payload['response'] = "Unable to send a friend request"
    else:
        payload['response'] = "You must be authenticated to send a friend req

    return HttpResponse(json.dumps(payload),content_type="application/json")
```

- ## Remove Friend :



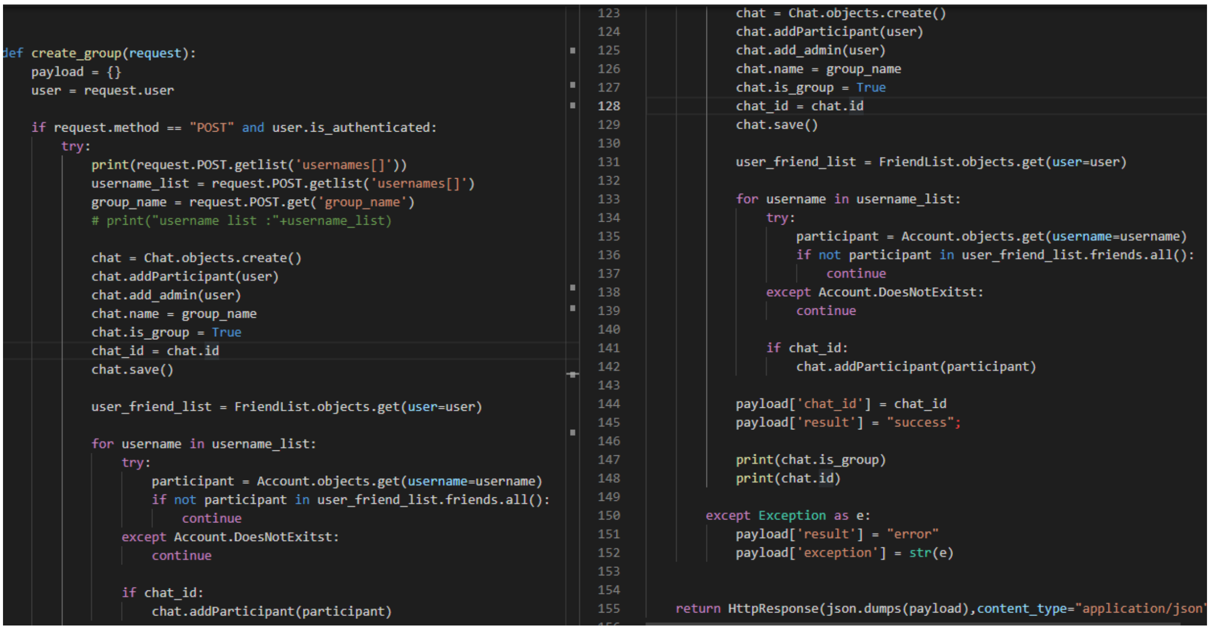
```
def remove_friend(request,*args,**kwargs):
    user = request.user
    payload = {}
    if request.method == "POST" and user.is_authenticated:
        user_id = request.POST.get("receiver_user_id")
        if user_id:
            try:
                account_other = Account.objects.get(pk=user_id)
                friend_list = FriendList.objects.get(user=user)
                friend_list.unfriend(account_other)
                payload['response'] = "Successfully removed that friend"
            except Exception as e:
                payload['response'] = f"Something went wrong : {str(e)}."
        else:
            payload['response'] = "There was an error. Unable to remove that friend."
    else:
        payload['response'] = "You must be authenticated to remove a friend"
    return HttpResponse(json.dumps(payload),content_type="application/json")
```

- **1-1 Chat :**



```
def private_chat(request,receiver_id):
    user = request.user

    try:
        receiver = Account.objects.get(pk=receiver_id)
    except Account.DoesNotExist:
        return HttpResponse("Something went wrong !! Account does not exist ")

    receiver_friend_list = FriendList.objects.get(user=receiver)

    if not user in receiver_friend_list.friends.all():
        return HttpResponse("Something went wrong !! You must be friend with "+receiver.user.username)

    chat_id = None

    print(receiver)

    for chat in Chat.objects.all():
        if not chat.is_group:
            # print(chat.numberOfParticipant())
            if chat.numberOfParticipant() == 2:
                if (receiver in chat.participants.all()) and (request.user in chat.participants.all()):
                    chat_id = chat.id

    if chat_id == None:
        chat = Chat.objects.create()
        chat.addParticipant(user)
        chat.addParticipant(receiver)
        chat.is_group = True
        chat_id = chat.id

    print(receiver.username+ " : "+str(chat_id))

    return redirect("chat:room",chat_id=chat_id)
```

- **Group Chat :**



```
def create_group(request):
    payload = {}
    user = request.user

    if request.method == "POST" and user.is_authenticated:
        try:
            print(request.POST.getlist('usernames[]'))
            username_list = request.POST.getlist('usernames[]')
            group_name = request.POST.get('group_name')
            # print("username list :"+username_list)

            chat = Chat.objects.create()
            chat.addParticipant(user)
            chat.add_admin(user)
            chat.name = group_name
            chat.is_group = True
            chat_id = chat.id
            chat.save()

            user_friend_list = FriendList.objects.get(user=user)

            for username in username_list:
                try:
                    participant = Account.objects.get(username=username)
                    if not participant in user_friend_list.friends.all():
                        continue
                except Account.DoesNotExitst:
                    continue

                if chat_id:
                    chat.addParticipant(participant)
```

```
            chat = Chat.objects.create()
            chat.addParticipant(user)
            chat.add_admin(user)
            chat.name = group_name
            chat.is_group = True
            chat_id = chat.id
            chat.save()

            user_friend_list = FriendList.objects.get(user=user)

            for username in username_list:
                try:
                    participant = Account.objects.get(username=username)
                    if not participant in user_friend_list.friends.all():
                        continue
                except Account.DoesNotExitst:
                    continue

                if chat_id:
                    chat.addParticipant(participant)

            payload['chat_id'] = chat_id
            payload['result'] = "success";

            print(chat.is_group)
            print(chat.id)

        except Exception as e:
            payload['result'] = "error"
            payload['exception'] = str(e)


    return HttpResponse(json.dumps(payload),content_type="application/json"
```

6. Testing

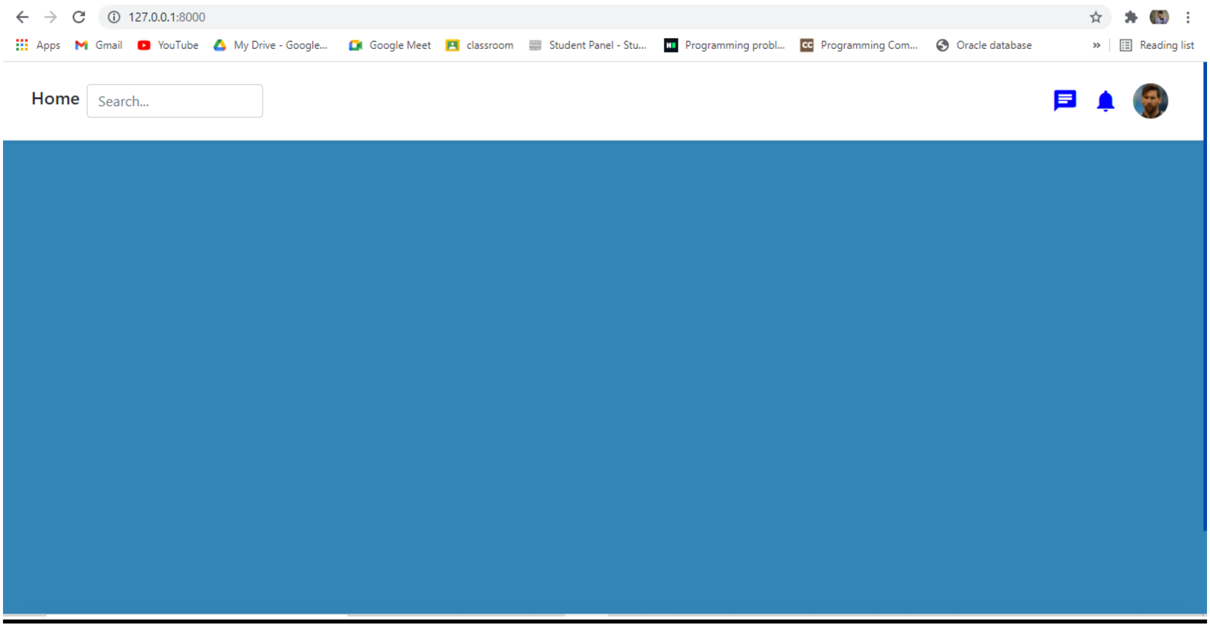
Manual testing was performed in order to find and fix the bugs in development process.

Testing Method: Manual Testing

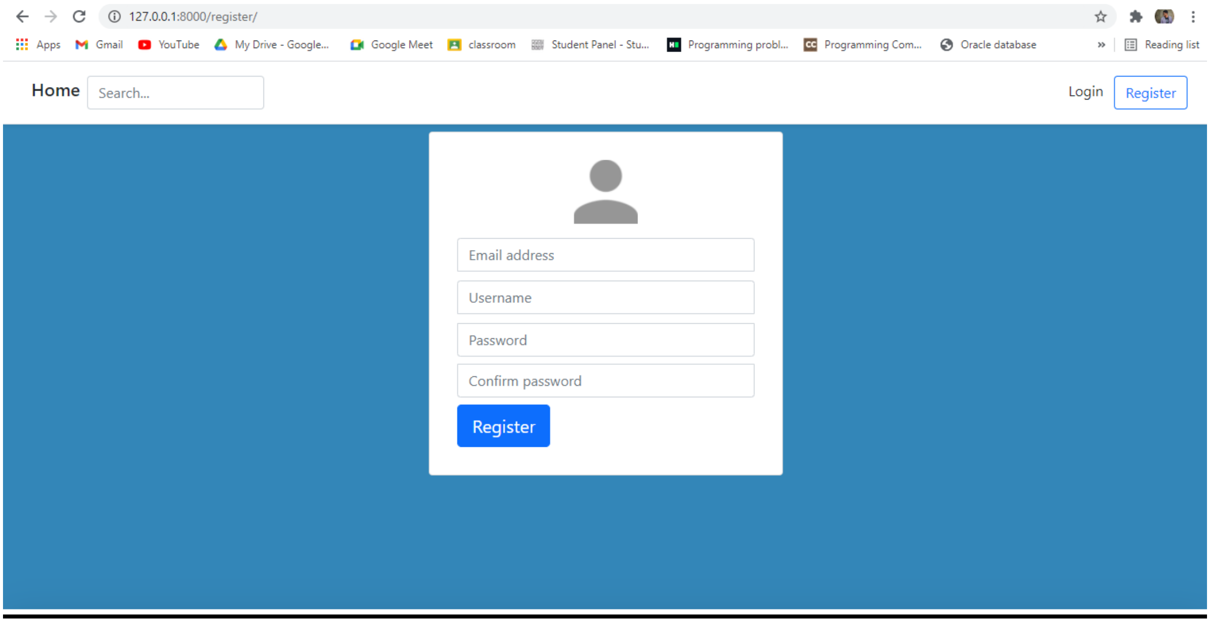
Sr No.	Test Scenario	Expected Result	Actual Result	Status
1.	Login with incorrect credentials	User should not able to log in.	User is given a message. And redirected to login page.	Success
2.	Login with correct credentials	User should be able to log in.	User is logged in and shown the dashboard.	Success
3.	Validations on registration	User should not be allowed to enter incorrect details	User is shown a message for any incorrect detail	Success
4.	Search Friends/Other User	User is able to search other users	When given a search query, matching Users are shown.	Success
5.	Log Out	User should be logged out and restricted from the system until next login.	User is successfully logged out and not able to access the system without signing again.	Success
6.	Add Friend/Send Friend Request	Friend request should be sent to other user	When Friend request is successfully sent option to cancel the request is shown.	Success
7.	Send Message	Message types on the box should be sent to the desired user.	Message sent.	Success
8.	Update Profile	Username, Email ,Profile Picture should be updated.	Username,Email and Profile Picture is Changed	Success

7. Work Flow / Layouts

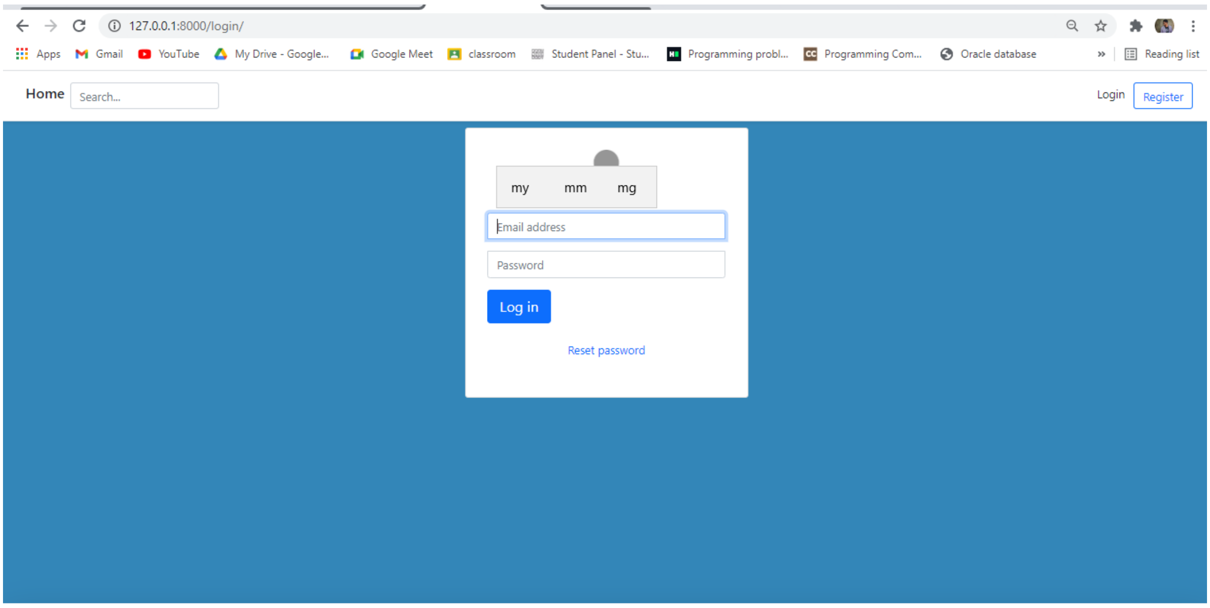
Home page:



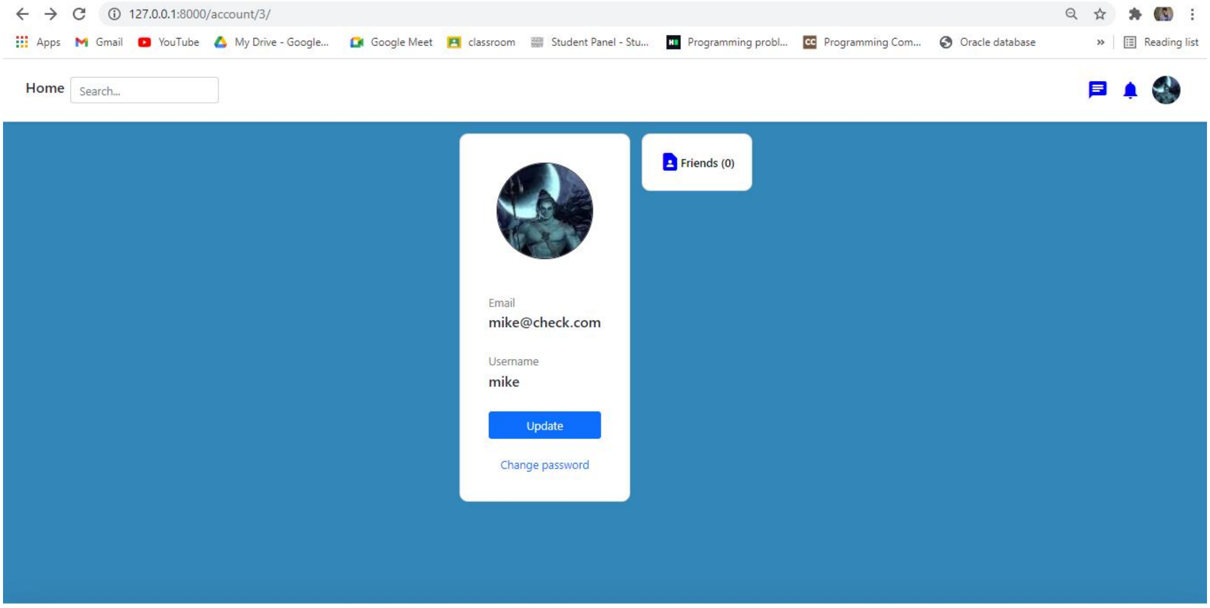
Sign Up:



User Login:



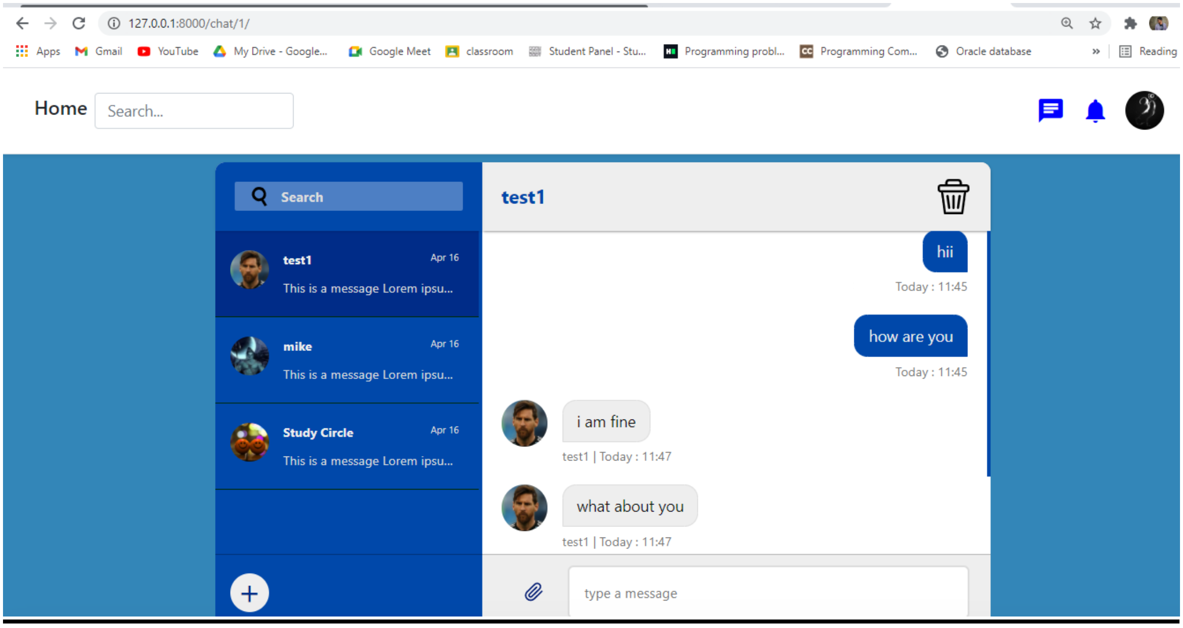
Account Page:



Email
mike@check.com

Username
mike

1-1 Chat:



test1 Apr 16
This is a message Lorem ipsu...

mike Apr 16
This is a message Lorem ipsu...

Study Circle Apr 16
This is a message Lorem ipsu...

test1

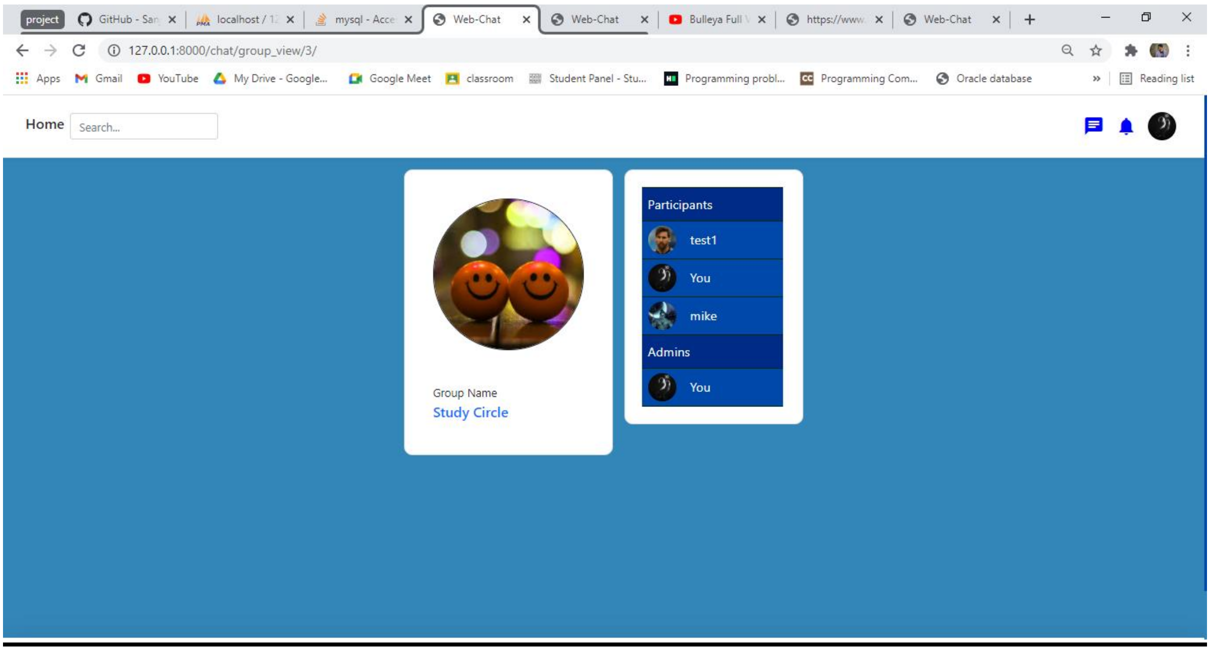
hii
Today : 11:45

how are you
Today : 11:45

i am fine
test1 | Today : 11:47

what about you
test1 | Today : 11:47

Group chat:



Group Name
Study Circle

Participants
test1
You
mike

Admins
You

8. Conclusion

The functionalities are implemented in system after understanding all the system modules according to the requirements. Functionalities that are successfully implemented in the system are:

- User registration
- Login
- Admin module
- 1-1 Chat
- Search User details
- Group chat
- Add Friend
- Edit Account

9. Limitations and Future Enhancements

- Here we can only do messaging by this app but we can add more features like voice calling, video calling.
- Here we can add features related to Stories , Activity related features like daily activity time on the app.
- We can add some gallery features like we can upload some pictures and some limited size videos.
- We can add features like we can share pdfs,documents etc with other friends.
- Some

10. Reference / Bibliography

Following links and websites were referred during the development of this project:

- stackoverflow.com
- https://github.com/fengyuanchen/cropperjs(for cropping image)
- github.com
- https://channels.readthedocs.io/en/stable/