

CONNECTING – JUET

AN FINAL PROJECT REPORT

Submitted by :-

Ayush Srivastava (171B041)

Kartikey Chandra (171B058)

Shashwat Shukla (171B117)

Under the guidance of :- Dr Ratnesh Litoriya



NOVEMBER 2019

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

Department of Computer Science & Engineering

**JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY, AB
ROAD, RAGHOGARH, DT. GUNA-473226 MP, INDIA**

DECLARATION

I hereby declare that the work reported in the B. Tech. 5th semester project entitled as “**CONNECTING JUET**”, in partial fulfillment for the award of degree of B.Tech submitted at Jaypee University of Engineering and Technology, Guna, as per best of my knowledge and belief there is no infringement of intellectual property right and copyright. In case of any violation I will solely be responsible.

Kartikey Chandra

Ayush Srivastava

Shashwat Shukla

Department of Computer Science and Engineering

Jaypee University of Engineering and Technology

Guna, M.P., India

Date: 25/11/2019



JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY

(Accredited with Grade-A by NAAC & Approved U/S 2(f) of the UGC Act, 1956)

A.B. ROAD, RAGHOGARH, DIST: GUNA (M.P.) INDIA Phone: 07544

267051, 267310-14, Fax: 07544 267011 Website: www.juet.ac.in

CERIFICATE

This is to certify that the work titled “**CONNECTING JUET**” submitted **Kartikey Chandra , Ayush Srivastava , Shashwat Shukla** in partial fulfillment for the award of degree of B.Tech of Jaypee University of Engineering & Technology, Guna has been carried out under my supervision. As per best of my knowledge and belief there is no infringement of intellectual property right and copyright.

Also ,this work has not been submitted partially to any other University or Institute for the award of this or any other degree or diploma. In case of any violation concern student will solely be responsible.

Signature of Supervisor

(Dr. Ratnesh Litoriya)

Date: 25/11/2019

ACKNOWLEDGEMENT

Any Endeavour cannot lead to success unless and until a proper platform is provided for the same. This is the reason we find ourselves very fortunate to have undergone our project work of 3rd year under the supervision of **Dr.Ratnesh Litoriya**. All the faculty of department have extended a warm and helping hand.

Our sincere gratitude to Dr Ratnesh Litoriya , our project guide for having faith in us and thus allowing us to carry out a project on a technology completely new to us . He helped immensely by guiding us throughout the course of the project.

SUMMARY

Connecting Juet is a student oriented social media platform where student are free to connect from there seniors,teachers and other students.Students can also expertise their skills by uploading post`s,feeds,comments and many more. For authentication student login id is his/her enrolment number. They can also add friends ,update their avatars,be aware of upcoming events,manage friend request ,cancel already sent friend request.In case, student forgot their password redirecting link will be generated to their respective emails where he/she can change their password.Admin has authority to delete anyone post, createpost, delete profiles, connect friends. It will help sustainable development in student career by making them aware of technologies and it will also help students to explore their knowledge.

LIST OF FIGURES

Figure 1.1 : Xtreme Programming

Figure 2.1: Use case diagram for Connecting Juet

Figure 3.1: Activity Diagram for User

Figure 4.1: Activity Diagram for Admin

CHAPTERS

1. INTRODUCTION
2. APPROACH
3. USE CASE DIAGRAM
4. ACTIVITY FLOW DIAGRAM
5. BACKGROUND MATERIAL
6. SCREENSHOTS

CHAPTER-1: INTRODUCTION

1.1: What is CONNECTING –JUET

- CONNECTING –JUET : An Integrated Website for educational purpose in college level which will help students to interact, connect between each other, where admin will have full control.
- It will help sustainable development in student career by making them aware of technologies and it will also help students to explore their knowledge.
- Students can connect to each other irrespective of their year of Btech.
- Students can also connect to Faculty members.
- When they need any help they can post in main wall or directly message to their connections.
- Generally students feels shy to ask their senior regarding their project or other guidance personally but by private messaging feature they can send message to their connection.

1.2: MOTIVATION FOR CONNECTING –JUET

Nowadays Social Media websites are very distracting for students, so we decided to design a platform on which students can't be distracted from their target.

Normally social media websites have no control on distracting posts , feeds, so our project is away from these scenarios.

This platform is only for college purposes as there is huge traffic in other social media websites.

CHAPTER2: APPROACH

1.1: AGILE MODEL (XTREME PROGRAMMING)

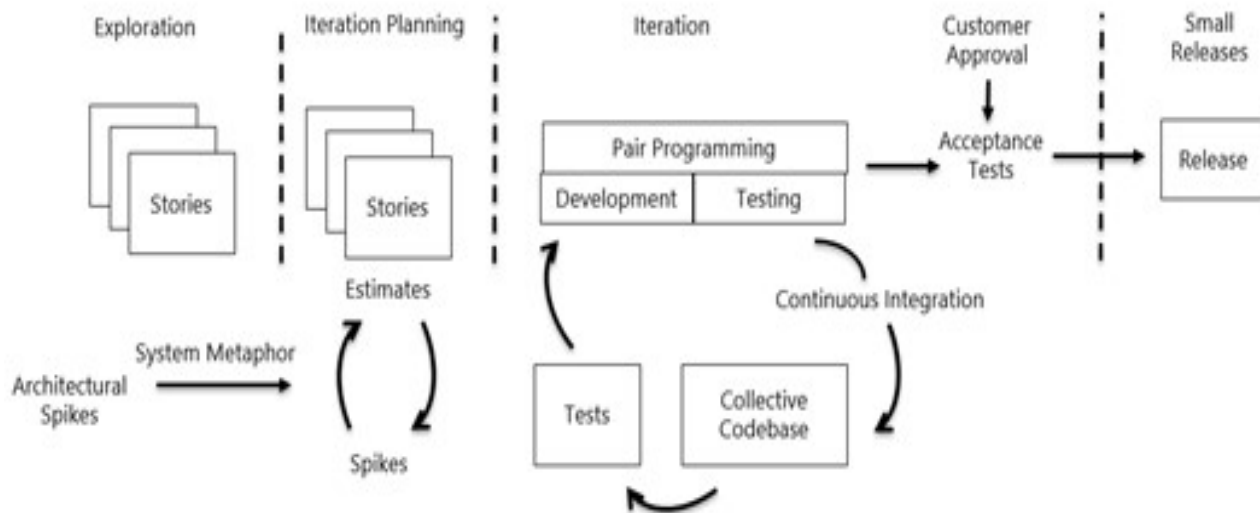


FIGURE 2.1

Xtreme Programming

We are using agile XP model because it uses pair programming and here we apply unit testing for smaller part of code and by using this model we can deploy project in less time.

In this model we begin with our requirements and needs that what we need in this project and what will the essentials steps in the completion of project.

Particular figure explains the whole process followed by the Extreme Programming XP and here at each step particular task is done from requirements gathering to creating small prototypes and testing of each prototype and creating whole of the project.

So if we analyse each step then .

- 1.) First is exploration. In this step we explore our needs of the project and create the architectural view of the project.

Key points used in this step.

a.) **Architectural spikes**: An architectural spike is a technical risk-reduction technique popularized by Extreme Programming(XP) where you write just enough code to explore the usage of a technology or technique that you are unfamiliar with.

b.) **System Metaphor**: An Extreme Programming system metaphor is a practice that used by the XP developers to replace the standard project architecture used in traditional software development methodologies .

2.) In 2nd step iteration planning is involved in which according to exploration of needs and requirements we define our projects estimates and redefine our architectural view of project.

3.) In 3rd step again iteration technique is involved and in this part coding part come and creating of prototype is done with the help of pair programming and testin of prototype is also done.

Key points used in this step.

a.) **Pair Programming**: In pair programming two developers team together and work on one computer to design project.

b.) **Collective Codebase**: In this everybody is encouraged to contribute new ideas to all segments of projects. Any developer can change any line of code to add functionality, fix bugs, improve design or refactor.

4.) In this step customer approval is taken for a particular prototype and error free running of particular prototype is shown to customers.

5.) In this step particular prototype is released after customer approval and testing.

CHAPTER 3: USE CASE DIAGRAM

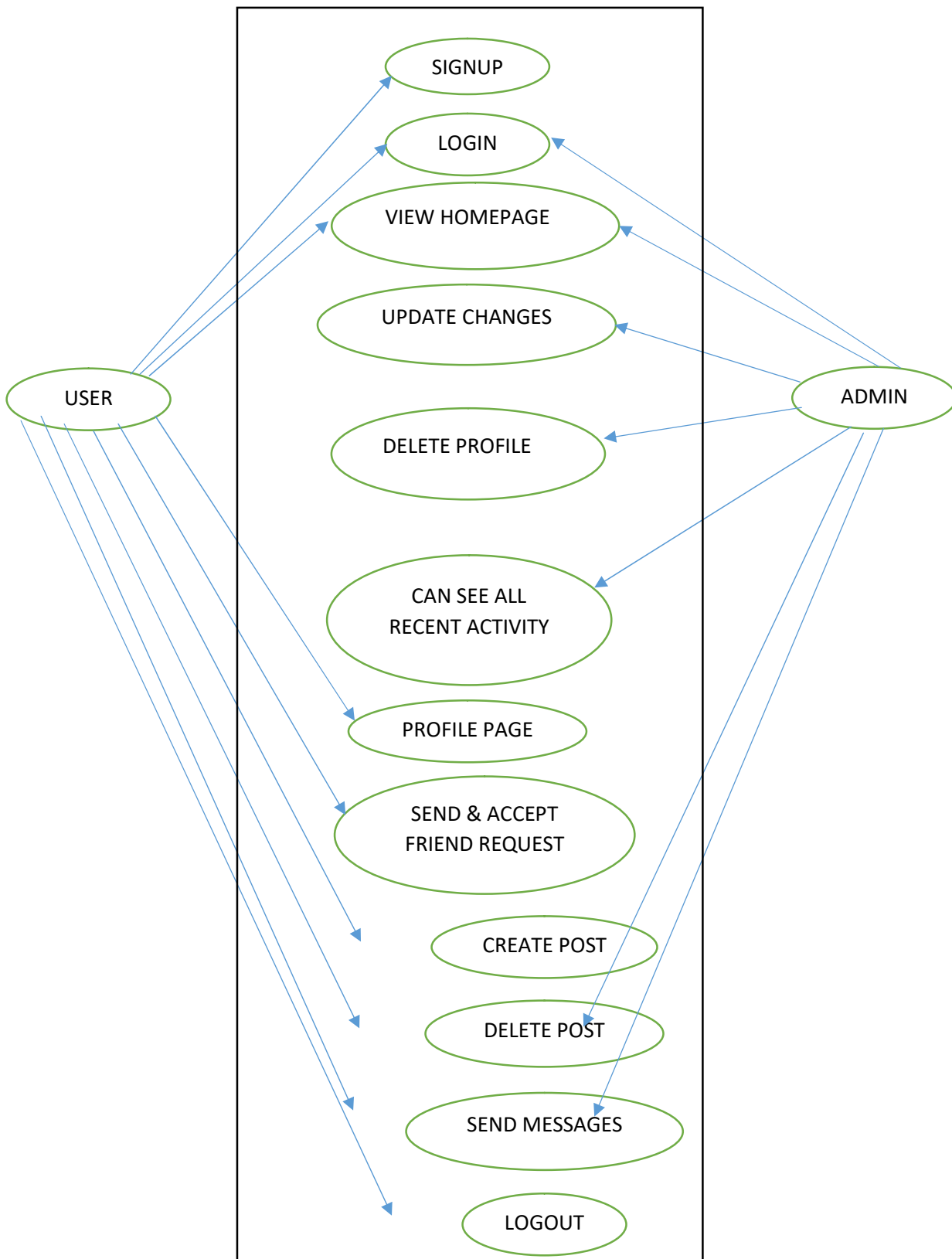


FIGURE –3.1 (USE CASE DIAGRAM FOR CONNECTING JUET)

In use case diagram we see how user interacts with the following system here are the following list of activities listed in the use case diagram such as signup, login, view homepage, update changes, delete

profile, block user, can see all recent activity, profile page, send and accept friend request, create post, delete post, send messages and logout.

Activities of user:

User can signup, login view homepage, see their profile page ,create post, delete their post, send messages and logout.

Activities of admin:

The interaction of admin side with this system is it can do following activities such as signup, login, view homepage, can see all recent activities, block user, delete profile, delete any post, send messages.

CHAPTER 4: ACTIVITY FLOW DIAGRAM

4.1: FOR STUDENTS:-

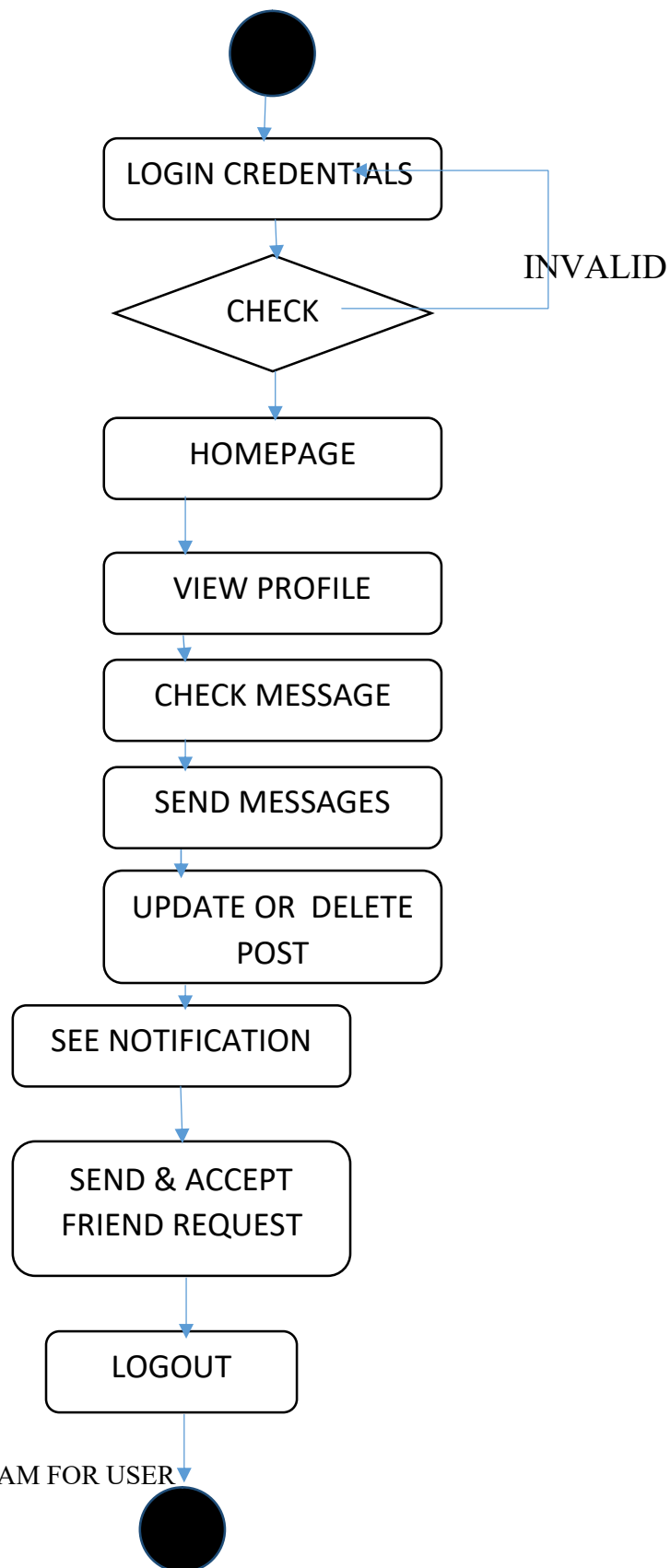


FIGURE-4.1 ACTIVITY DIAGRAM FOR USER

4.2: FOR ADMIN

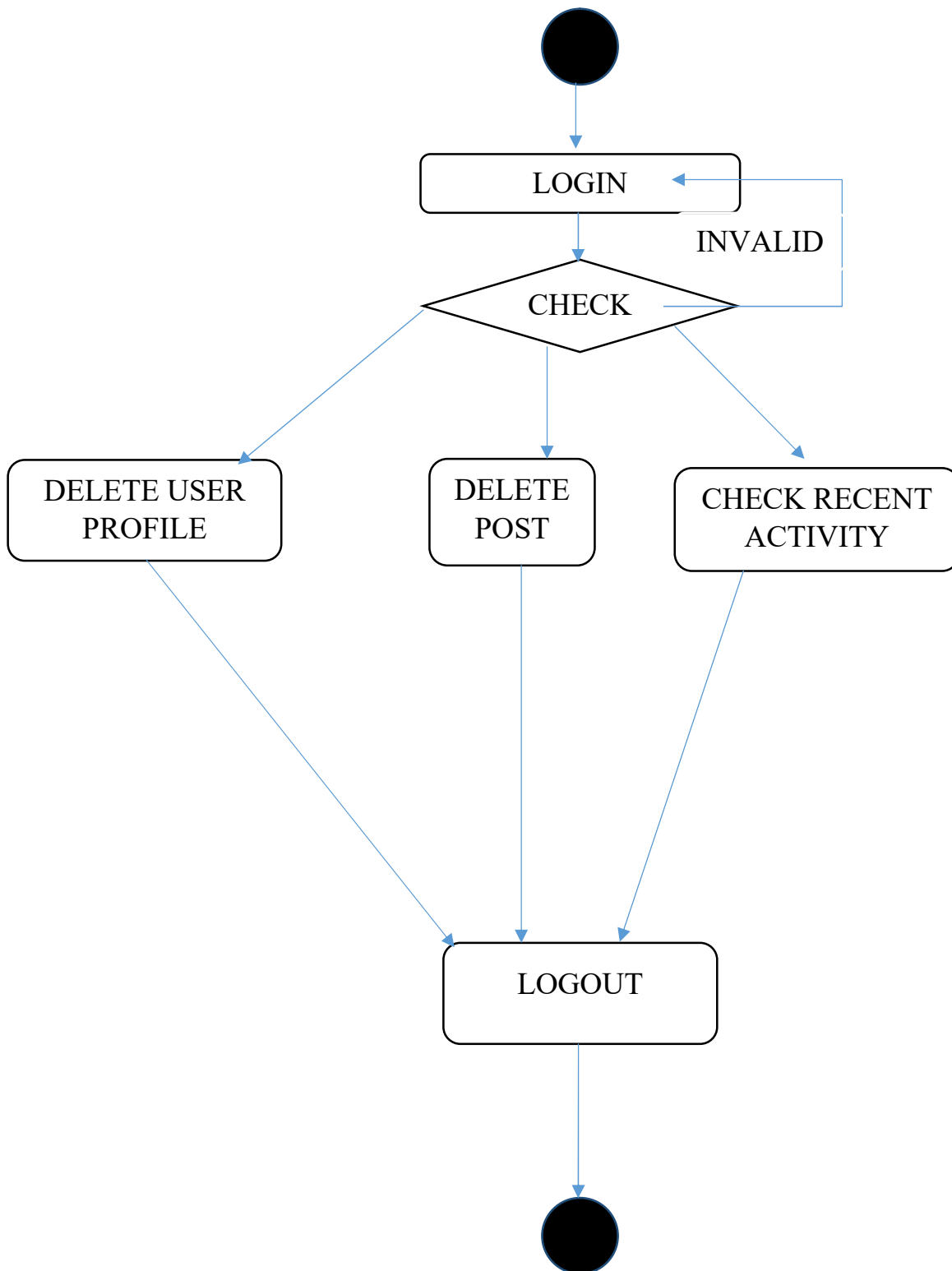


FIGURE 4.2 - ACTIVITY DIAGRAM FOR ADMIN SIDE

CHAPTER 5: CUSTOMER EXPECTATION

- Since human have a tendency to do easiest job, so User should not have difficulty to interact with their website.
- As our project revolve around students , it should not be deviated.
- Admin or Owner should control over a project on profiles ,posts,feeds.
- Students are our major target , only they could interact in our website.
- User interface is indirectly connected to User Experience therefore UI should not be too complicated , every buttons,fields should be clear enough to interact without any hassle.
- Some add-ons should be added for accessibility like Speech to text and many more.

CHAPTER 6: IMPLEMENTATION OF WEB APPLICATION

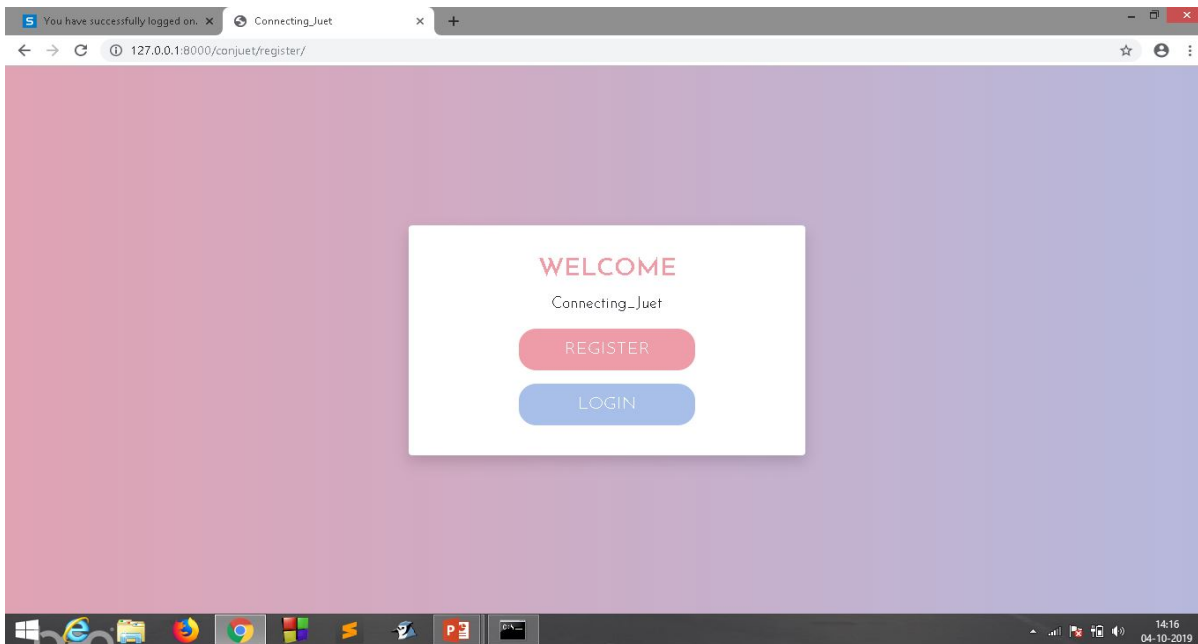


Figure 6.1

First page:- This is our main page where user decide whether he/she want to login or register, by clicking on any of these buttons dialog box will transform into new one, just to improvise visualization colour effect was changed from gradient to respective colour to their buttons while transformation. By clicking on Register it can go to Register page from where it can make a profile in this Web-site. he/she have to fill a form in order to form their profile.

```
path('register/', views.register , name='register'),
```

By giving this path , Program go to Views file to call register function

Register in which we take input from template textfield when we click on register button it send a POST request , eg-

Takes username from template by:-

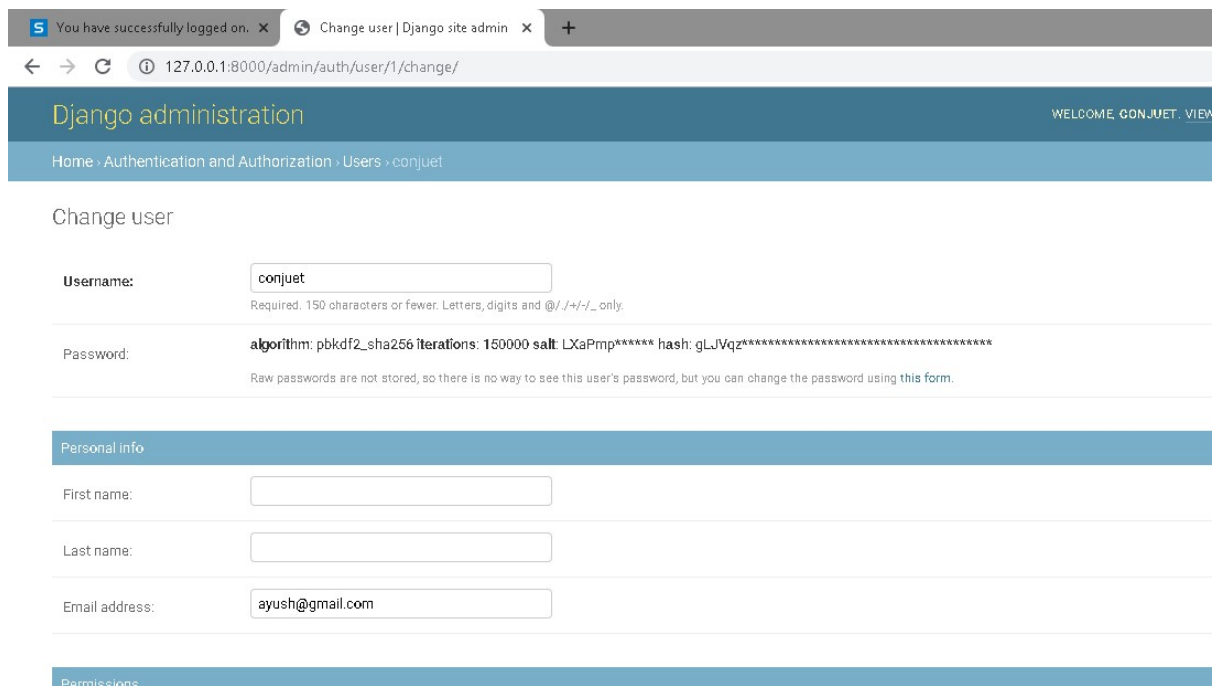
```
username=request.POST['enrollno']
```

Password , email by :-

```
user.set_password(params['password'])
```

```
user.email=params['email']
```


by using `set_password` it hashed the password so admin is also not able to see your password.

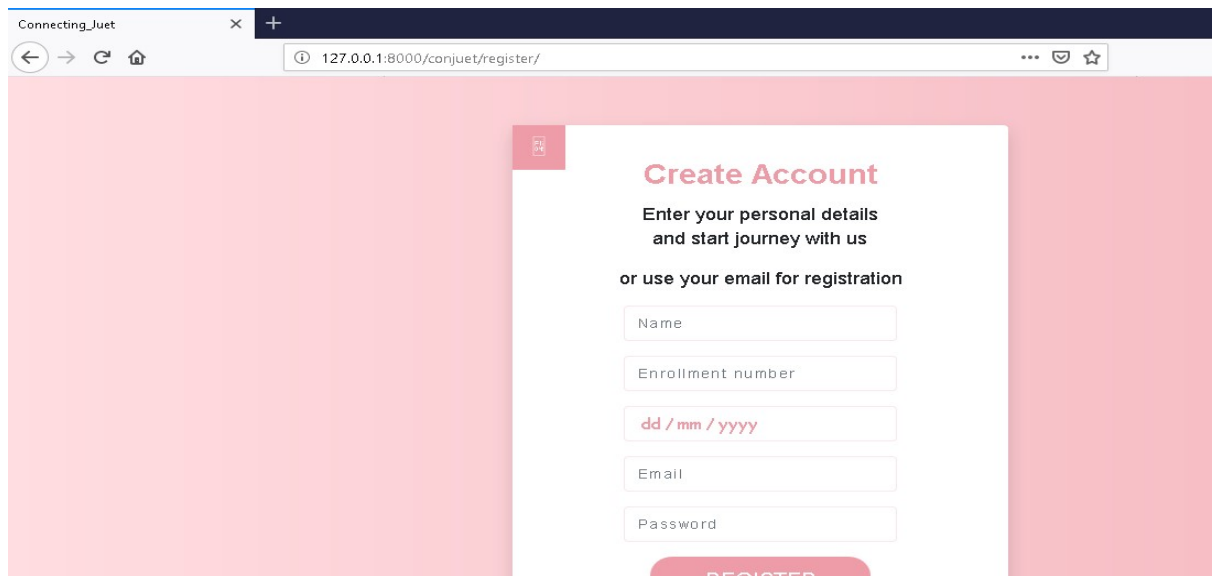


The screenshot shows the Django administration interface. The browser tab is 'Change user | Django site admin'. The URL is `127.0.0.1:8000/admin/auth/user/1/change/`. The page title is 'Django administration' with a 'WELCOME, GONJUET. VIEW' link. The breadcrumb is 'Home > Authentication and Authorization > Users > conjuet'. The main heading is 'Change user'. There are two sections: 'User details' and 'Personal info'. In 'User details', the 'Username' is 'conjuet' and the 'Password' is masked with asterisks. The password algorithm is 'pbkdf2_sha256 iterations: 150000 salt: LXaPmp***** hash: gLJVqz*****'. A note states: 'Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.' The 'Personal info' section has fields for 'First name', 'Last name', and 'Email address' (which is 'ayush@gmail.com'). A 'Permissions' section is partially visible at the bottom.

Figure 6.2

In this screen you can see that by using admin account we cannot able to see user's password

This screen is from Admin account where we can check all the entry from web page.



The screenshot shows a web browser with the URL `127.0.0.1:8000/conjuet/register/`. The page has a pink background. A white dialog box titled 'Create Account' is centered. It says 'Enter your personal details and start journey with us' and 'or use your email for registration'. The form fields are: 'Name', 'Enrollment number', 'dd / mm / yyyy' (with a red example), 'Email', and 'Password'. A red 'REGISTER' button is at the bottom.

Figure 6.3

Register :- This is our register dialog box with fields- Name, Enrollment number ,DOB,Email,Password.

By clicking Register button it call POST method:-

```
<button class="btn btn-lg" type="submit" value="Post">REGISTER</button>
```

Value = POST sends the post request to views function which save the data entry from register page to User data model.

```
def register(request):
```

```
    if request.method=="POST":
```

```
        params=request.POST
```

```
        notif=User.objects.get_or_create(username=params['enrollno'])
```

```
        if(notif):
```

```
            user.set_password(params['password'])
```

```
            user.email=params['email']
```

```
            user.save()
```

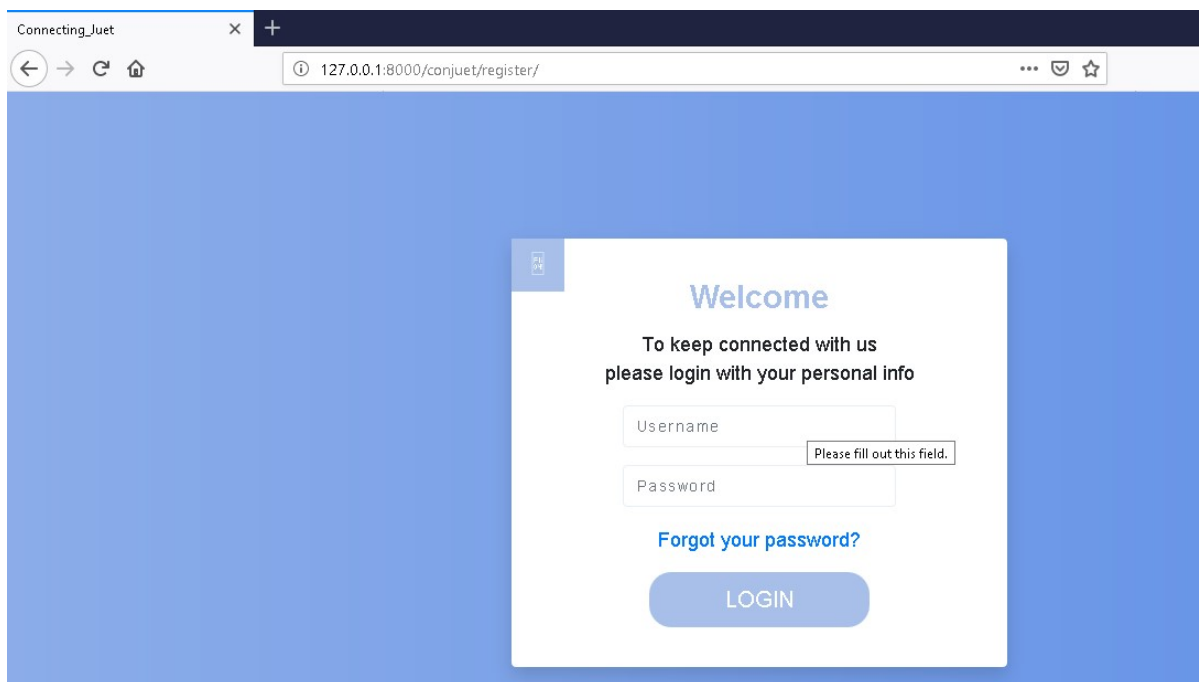


Figure 6.4

Login:-After registration user will face our login page where Enrollment number and password will be checked in database after authentication he/she would see our main Home page. Here we have two dialog box username and password. When we enter login button it call “POST” method :

```
<a class="btn btn-lg btn-login js-btn" data-target="login">LOGIN</a>
```

POST method sends request to views to call login function and see in if particular user exists in User database model.

```
def login(request):

    if(request.method=="POST"):

        params=request.POST

        user=authenticate(params['username'], params['password'])

        if(user is not None):

            login(request,User.objects.get(user__username=params['userne'])
```

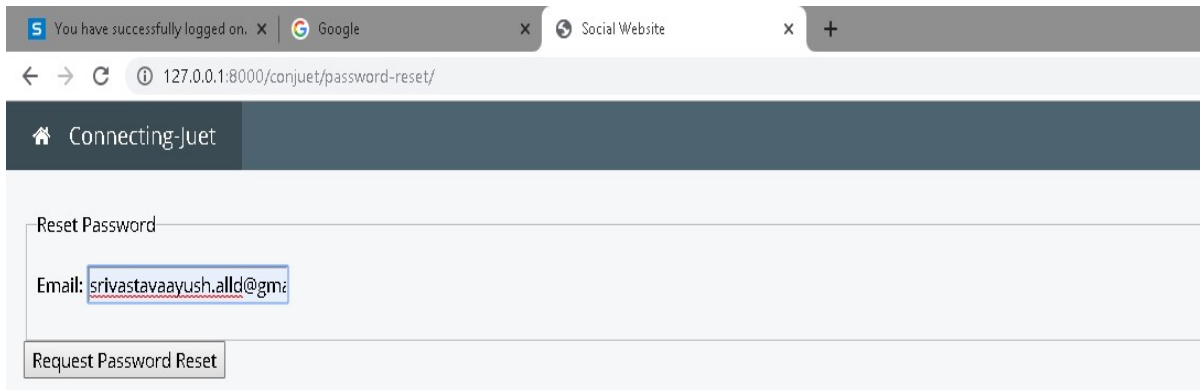


Figure 6.5

Password Reset:-In this page User can reset their password . He/she can enter their email id , which is connected to their profile .

Then we check if the entered email is in registered profile if the entered email is in registered profile then it sends a email to reset your password, it automatically detect your username as you have given your email account and it generate a link to reset your password ,

As you click on the given link in the email it redirect you to a new web page where you can update or create new password.

```
path('password-reset/', auth_views.PasswordResetView.as_view(template_name='account/password_reset.html'))
path('password-reset/done/', auth_views.PasswordResetDoneView.as_view(template_name='account/password_rese
path('password-reset-confirm/<uidb64>/<token>/', auth_views.PasswordResetConfirmView.as_view() ,name='pass
path('password-reset/complete/', auth_views.PasswordResetCompleteView.as_view() ,name='password_reset_comp
```

Figure 6.6

Here we can see there are four path to reset or change your password

By using 1st path you will redirect to page shown above which allows you to enter your email address.

By using 2nd path it checks that email is linked with any profile and send a email with a password regenerate link and shows done page which says:-

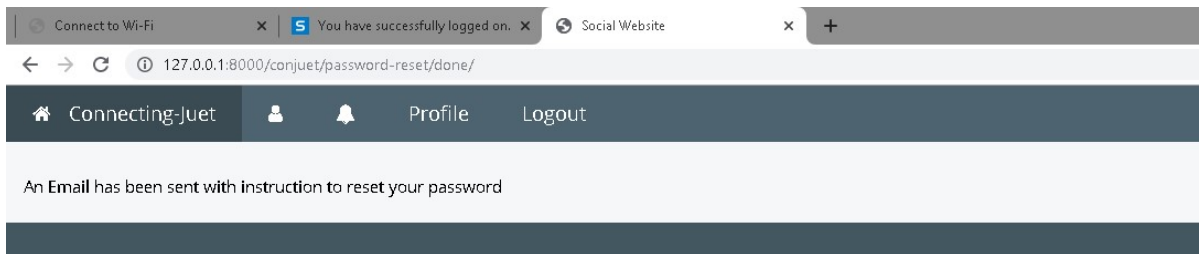


Figure 6.7

Here we see that this page confirms that a email has sent to your email account.

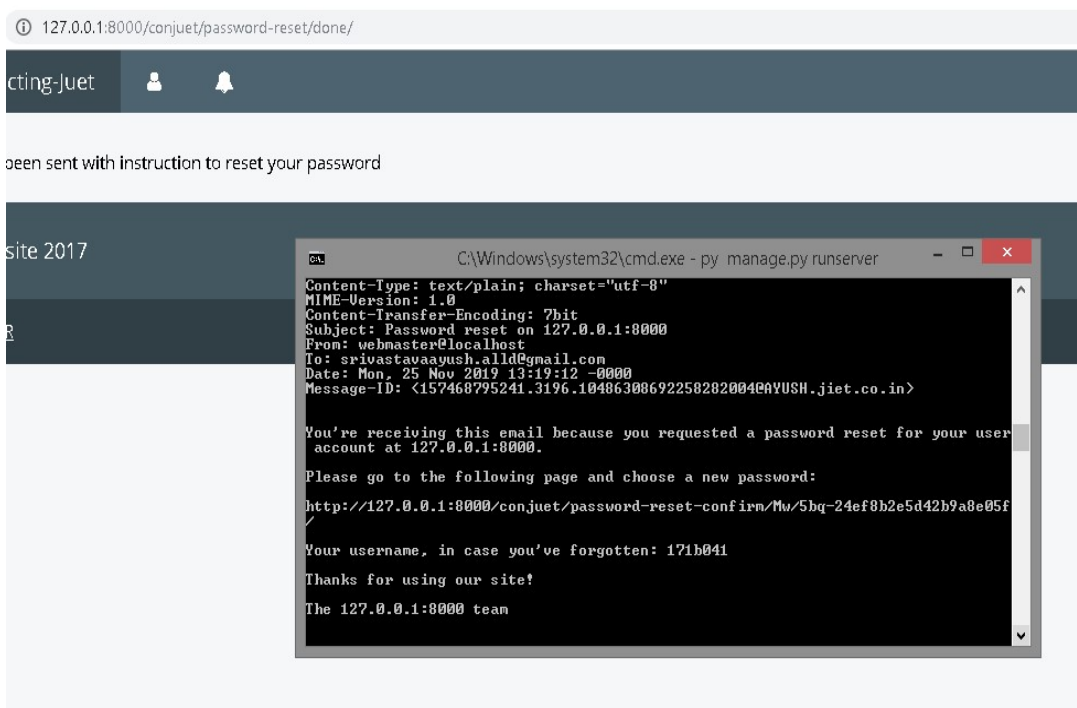


Figure 6.8

Password reset link:- Reset page show you a link that will help to open a page that will redirect you to a new page that will allow you to change your password .

Allows a user to reset their password by generating a one-time use link that can be used to reset the password, and sending that link to the user's registered email address.

If the email address provided does not exist in the system, this view won't send an email, but the user won't receive any error message either. This prevents information leaking to potential attackers.

If you want to provide an error message in this case, you can subclass **PasswordResetForm** and use the **form_class** attribute.

Users flagged with an unusable password (see `set_unusable_password()`) aren't allowed to request a password reset to prevent misuse when using an external authentication source like LDAP. Note that they won't receive any error message since this would expose their account's existence but no mail will be sent either.

```
You're receiving this email because you requested a password reset for your user
account at 127.0.0.1:8000.

Please go to the following page and choose a new password:
http://127.0.0.1:8000/conjuet/password-reset-confirm/Mw/5br-b0e5f6780f2e65126708
/

Your username, in case you've forgotten: 171b041

Thanks for using our site!

The 127.0.0.1:8000 team
```

Figure 6.9

Here you can see message that you will find in email to change your password.

Attributes:

- **template_name**: The full name of a template to use for displaying the password reset form. Defaults to `registration/password_reset_form.html` if not supplied.
- **form_class**: Form that will be used to get the email of the user to reset the password for. Defaults to `PasswordResetForm`.
- **email_template_name**: The full name of a template to use for generating the email with the reset password link. Defaults to `registration/password_reset_email.html` if not supplied.
- **subject_template_name**: The full name of a template to use for the subject of the email with the reset password link. Defaults to `registration/password_reset_subject.txt` if not supplied.
- **token_generator**: Instance of the class to check the one time link. This will default to `default_token_generator`, it's an instance of `django.contrib.auth.tokens.PasswordResetTokenGenerator`.
- **success_url**: The URL to redirect to after a successful password reset request.
- **from_email**: A valid email address. By default Django uses the `DEFAULT_FROM_EMAIL`.
- **extra_context**: A dictionary of context data that will be added to the default context data passed to the template.
- **html_email_template_name**: The full name of a template to use for generating a `text/html` multipart email with the password reset link. By default, HTML email is not sent.
- **extra_email_context**: A dictionary of context data that will be available in the email template. It can be used to override default template context values listed below e.g. `domain`.

Figure 6.10

These function are already generated by Django function. As in

`Auth_views`.

A screenshot of a web browser showing the Django administration interface. The browser's address bar displays the URL '127.0.0.1:8000/conjuet/password-reset-confirm/Mw/set-password/'. The page has a dark blue header with 'Django administration' in yellow. Below the header is a light blue navigation bar with 'Home > Password reset confirmation'. The main content area is white and titled 'Enter new password'. It contains a message: 'Please enter your new password twice so we can verify you typed it in correctly.' There are two input fields: 'New password:' and 'Confirm password:', both with masked characters (dots). A blue button labeled 'Change my password' is at the bottom.

Figure6.11

Change Password:-You will be redirected to new page where you have to enter new password twice, to enter and confirm .

After that you will be redirect to next page that says Your Password has changed now you can login with this password.

That will be your Password reset Confirm Page.

That will change your password attached to the given Email .

After that you can go to login page again to login with new Password with the help of newly generated password you will be able to login to your profile .

After that you always can use this password to enter .

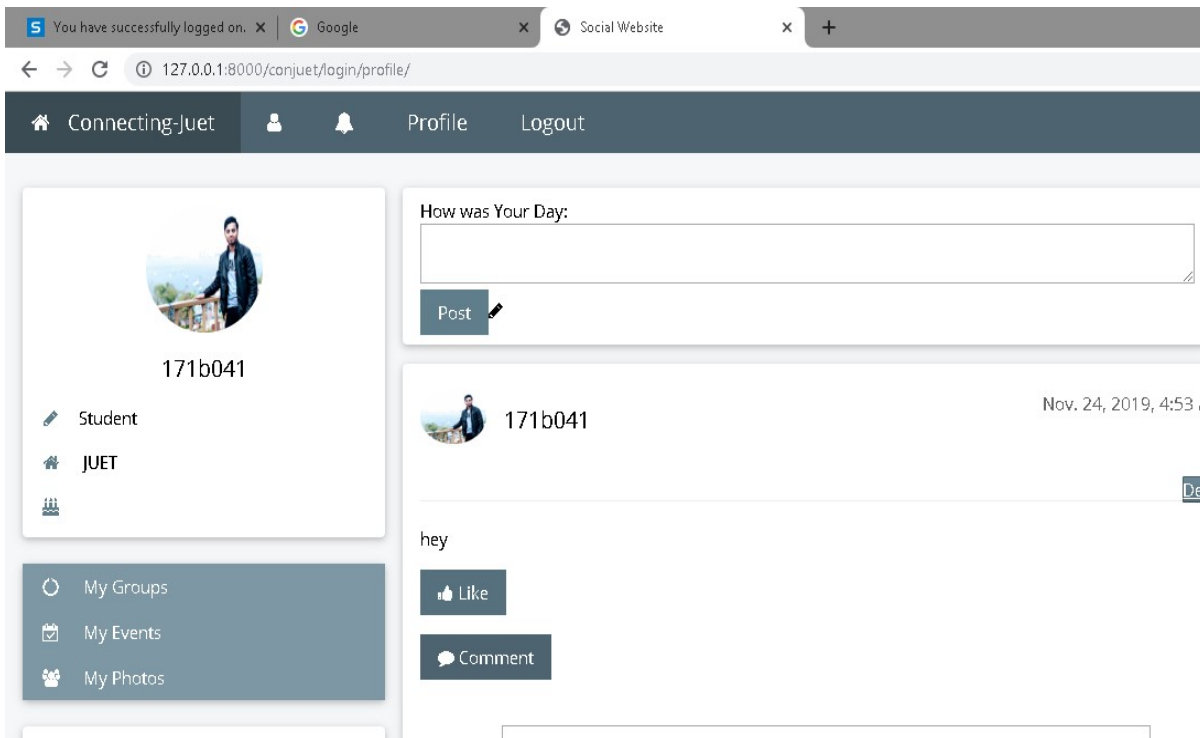


Figure 6.12

Homepage:-

This is our main Homepage where you can see posts of our friends and yourself only until you don't become friends with anybody you cannot be able to see his/her posts .

Here you can see your Profile pic on Left top and your username just below that which is by default your Enrollment number , because other people will just get the idea of your year of batch , there are many people of same name and by default we provide a similar profile pic to new user so if he didn't change it ,it will be hard to recognize person.

So we used Enrollment number as your username.

From this page you are allowed to enter your query by the post method provide by this page you can type any question or your problem to get solution.

When you press that post button after writing your question it will save your post in data model named as Createpost.


```
class Createpost(models.Model):
    content = models.CharField(max_length=100)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    date_posted = models.DateTimeField(default=timezone.now)
    comments=models.ManyToManyField(Comments, blank=True)

    def __str__(self):
        return f'{self.user}'
```

Figure 6.13

Here you have content of your post thing that you have written in the textfield in the homepage to post will be save in this field which is varchar2 type.

After that user who posted that post , we save his name under the name of user by that we will now who have posted that post ,who have this query.

Date and time of post this is also a important key , this stores the date and time of post directly from your system , it fills time and data into database thatwe can see in template when this post was made.

Comments this field stores the comments of user that commented on posts , every post has different comment this is link by this createpost,

This does not mix all the comment by this we can see which post has which comments.

```
class Comments(models.Model):
    user=models.ForeignKey(Profile, on_delete=models.CASCADE)
    commentData=models.TextField(blank=True, null=True)
    postedOn=models.DateTimeField(auto_now=True)
```

Figure 6.14

This is the database that stores comments of user on posts.

First it stores user info who have commented on the post we have to show to the user that this person have commented on your post so he will know who helped him with the solution.

Second it stores the comment data that means the commet that he/she have entered on comment field.

After clicking on comment button on homepage you will be redirect to new page that will show youcomments this page is comment.html

That render all the data from backend and then wedisplayit on front end.

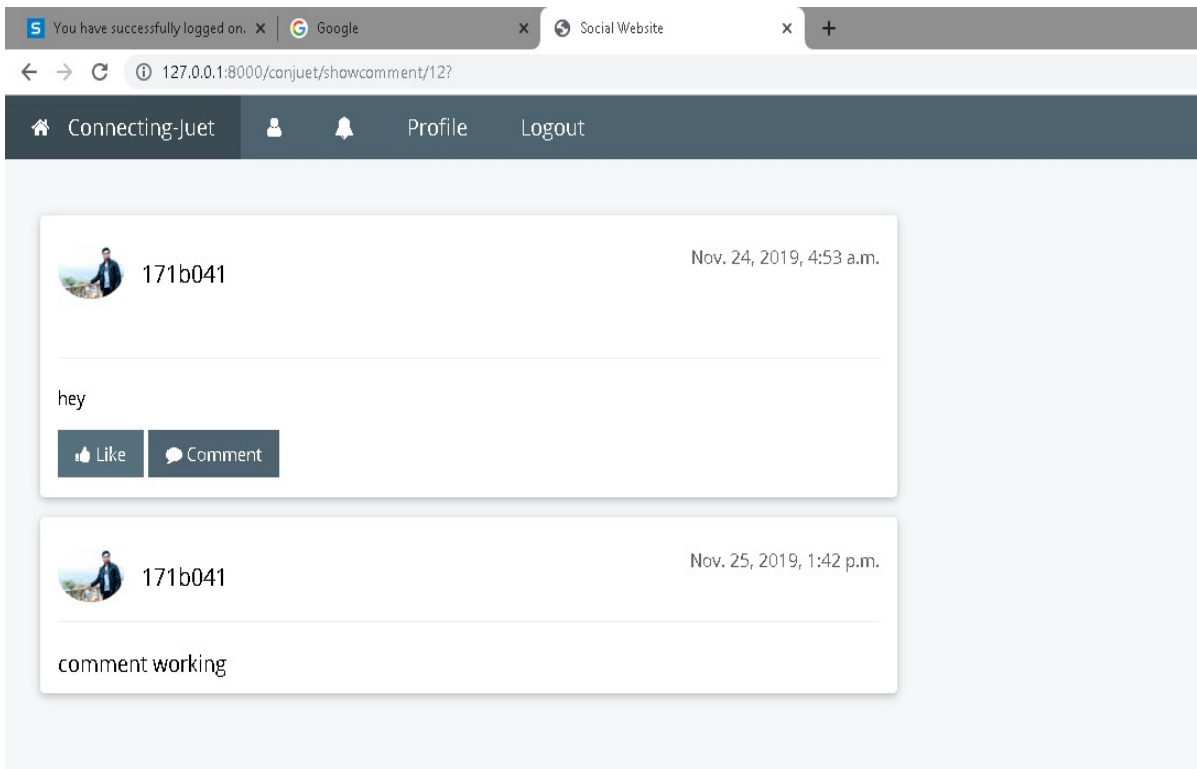


Figure 6.15

From this page you will get a way to go to many other pages as you can

use , as example you will now able to use Find friends page from where you can see other people that are in this platform already , you can send friend request to them , cancel if sent by mistake to someone else , Or you can receive friend request where you have option to accept that person's friend request or reject friend request ,the person will get notified if you accept his/her friend request .

He will be able to see your post if you accept his request now he/she can also comment on your post.

You can also go to notification page which is notification.html where you can check your notification if someone has accepted your friend request.

Or you can go to profile page from where you can change your profile picture.

When you click on notification it will lead you to the profile page of the person that you see notification of so you can see information about other person.

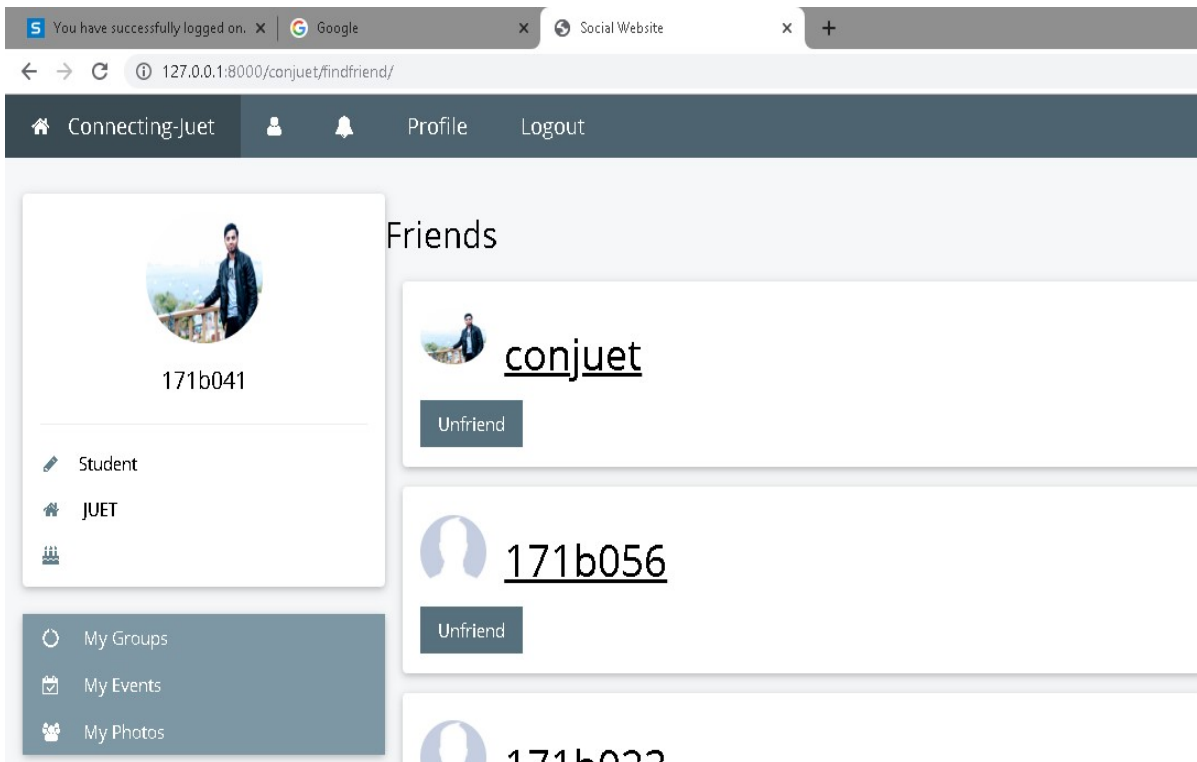


Figure 6.16

This is find friend page in which we can see our friends , or other user from which we can connect by sending friend request.

We can see our friends profile picture their name by clicking on their name we will redirect to their profile page.

We can also see other users profile picture that are not our friend by which we can recognize them and send them request to connect.

From this friend request we have created two data models 1st:-

```
class FriendFormation(models.Model):
    fromUser=models.ForeignKey(User , related_name='friendOwner',on_delete=models.CASCADE)
    toUser=models.ForeignKey(User , related_name='friendReceiver',on_delete=models.CASCADE)
    isFriend=models.BooleanField(default=False)
    isFriendRequest=models.BooleanField(default=True)
    created=models.DateTimeField(auto_now=True)
```

Figure 6.17

fromUser stores the info which user has sent friend request. If you send someone friend request you will be saved as fromUser in this database.

toUser stores the info of the user whom friend request is sent .

isFriend is the Boolean field which shows you whether you are friends with other user or not . By default isfriend is false until you don't accept request you are not friends.

isFriendRequest is also a Boolean field which shows whether you sent request to connect or not. If you press button ADD friend this will sent post request and isFriendRequest data is formed.

When you accept friend request function works as below:-

```
def accept_request(self, current_user, new_friend):
    if(FriendFormation.objects.filter(fromUser=new_friend, toUser=current_user).count()>0):
        obj = FriendFormation.objects.get(fromUser=new_friend, toUser=current_user)
        obj.isFriendRequest=False
        obj.isFriend=True
        obj.save()
```

Figure 6.18

When you reject friend request function work :-

```
def reject_request(self, current_user, new_friend):
    if(FriendFormation.objects.filter(fromUser=current_user, toUser=new_friend).count()>0):
        FriendFormation.objects.get(fromUser=current_user, toUser=new_friend).delete()
    else:
        FriendFormation.objects.get(fromUser=new_friend, toUser=current_user).delete()
```

Figure 6.19

It will delete the enrty from database as we reject the request.

When we cancel send request it also delete enrty from data base.

```
def cancel_request(self, current_user, new_friend):
    FriendFormation.objects.get(fromUser=current_user, toUser=new_friend).delete()
```

Figure 6.20

When we unfriend a friend then the function we use-

```
def lose_friend(self, current_user, new_friend):
    if(FriendFormation.objects.filter(fromUser=current_user, toUser=new_friend).count()>0):
        FriendFormation.objects.get(fromUser=current_user, toUser=new_friend).delete()
        Notifications.objects.get(fromUser=current_user, toUser=new_friend).delete()
    elif(FriendFormation.objects.filter(fromUser=new_friend, toUser=current_user).count()>0):
        FriendFormation.objects.get(fromUser=new_friend, toUser=current_user).delete()
        Notifications.objects.get(fromUser=new_friend, toUser=current_user).delete()
```

Figure 6.21

In this we see that it also delete the notification of friend request because we are not friends anymore so that notification is not required.

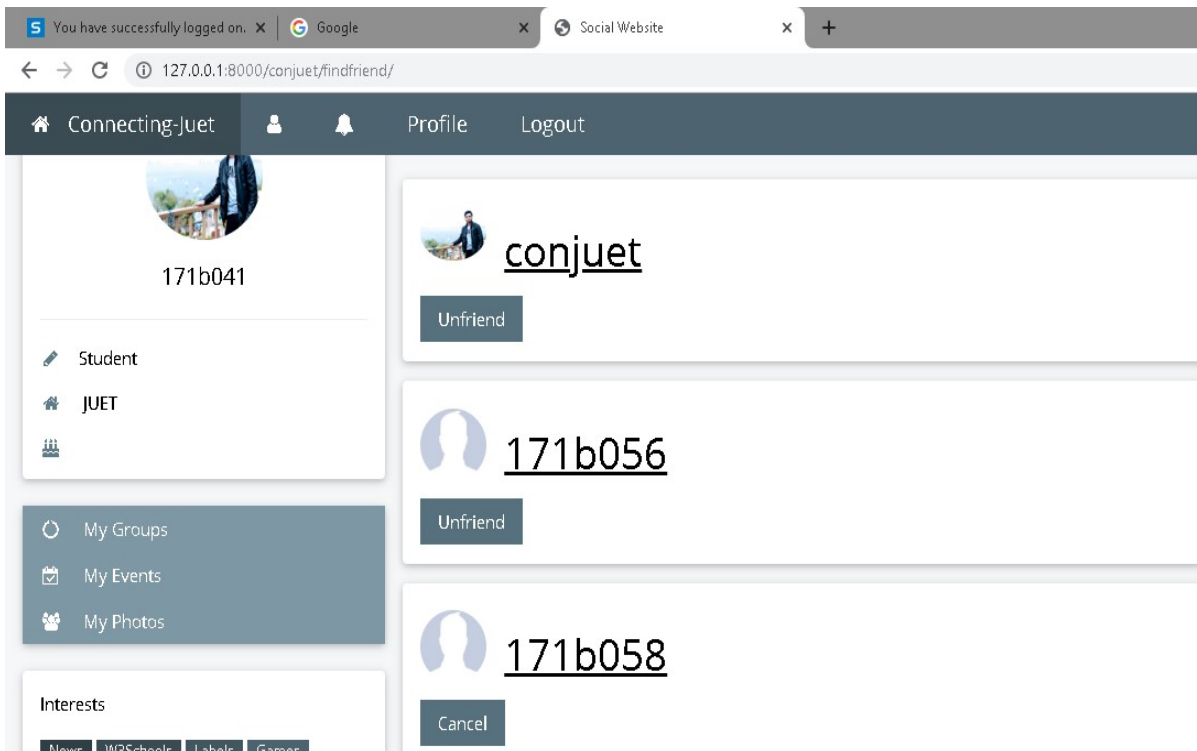


Figure 6.22

In this page we can see that we have friends and we have option to unfriend any of our friends for that function we have discussed above.

We have checked whether the person is in our friends list then we set button to unfriend , if the person is not in our friend list then we have set button to add friend.

As you can see their is option to cancel a request , this shows that we have sent a request to that person and we can cancel that request.

Function for cancel we have defined above.

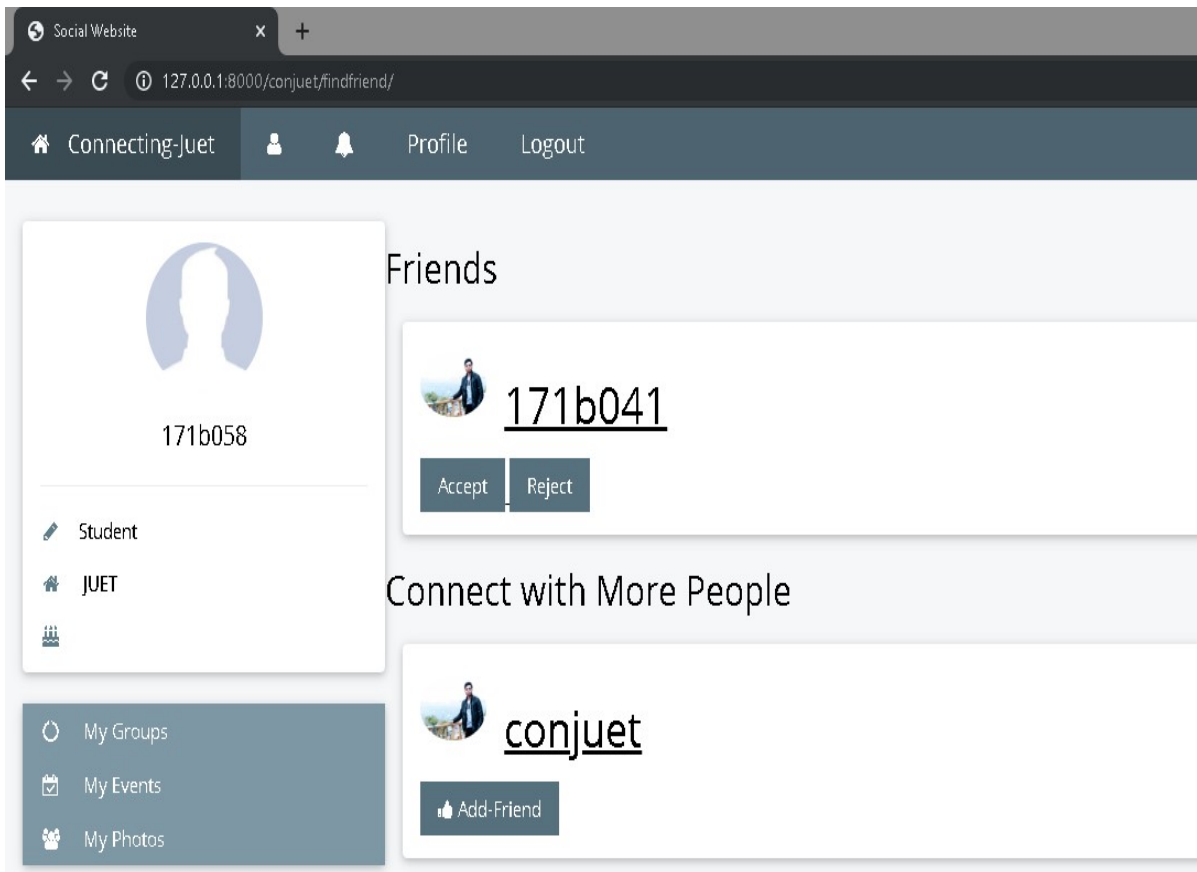


Figure 6.23

In this page we can see that we have received friend a friend request and we have option to accept or reject friend request ,its up to the user whether he want to be friends with that person or not if he accepts he will allow other user to see their posts or comment on their post.

When you accept friend request we will get a notification that you have accepted his friend request , and if he reject he will now able to send you friend request again.

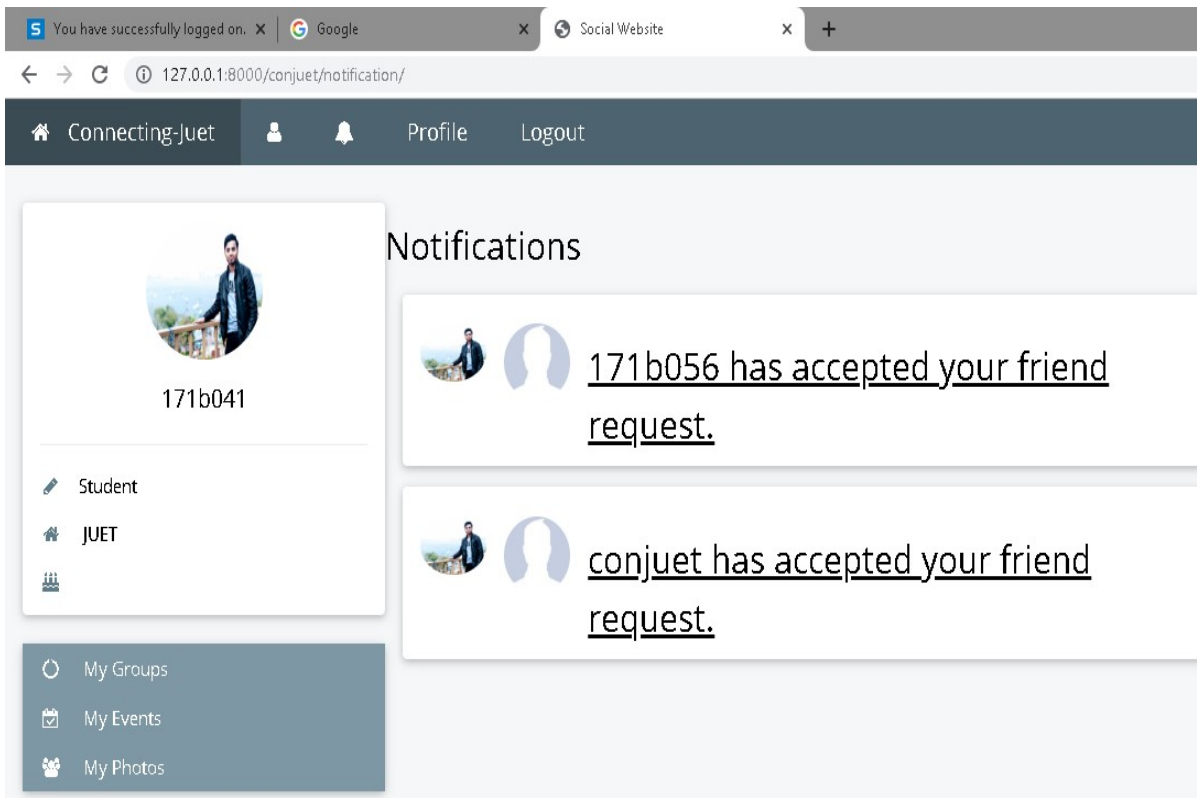


Figure 6.24

If the user accepted your friend request you will get a notification with message that the person has accepted your friend request.

By clicking on link in the notification you will be redirect to profile page of your friend's page.

As multiple person accepts your request you can see all of their notification by order of “-date posted” this means first you will see new notification first then older one.

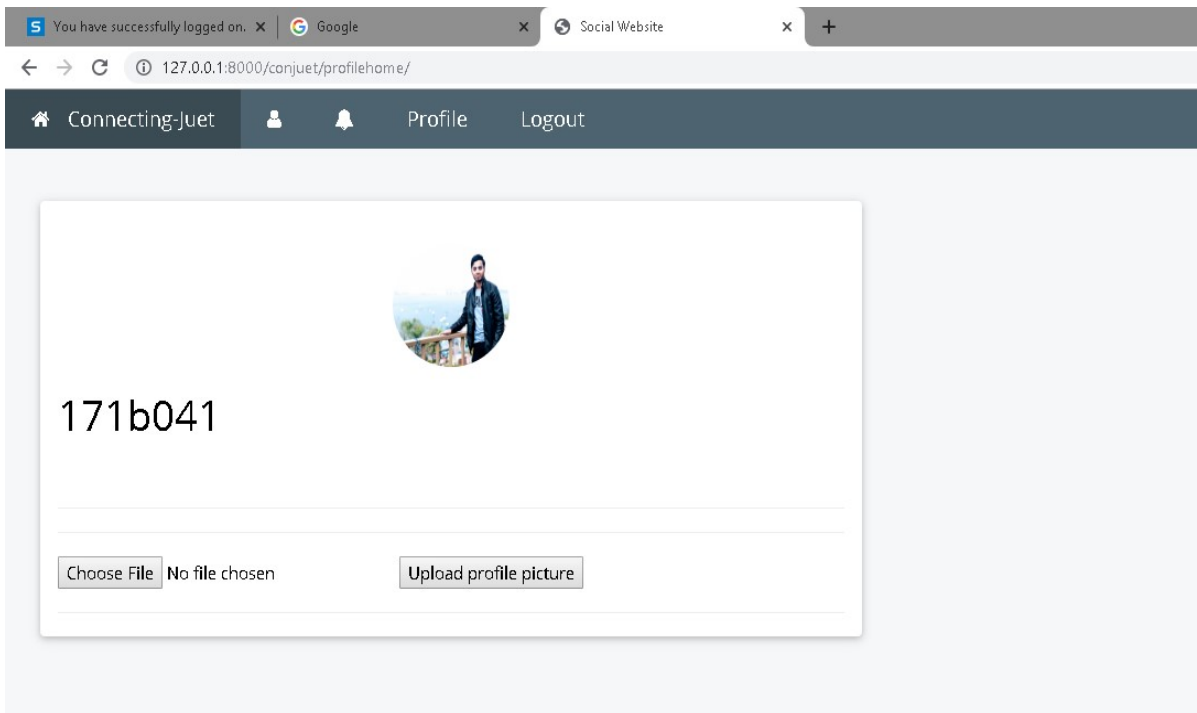


Figure 6.25

This is your profile page in which you can change your profile picture.

Using Multipart form data you are getting profile picture uploaded from your drive and then we save this photo in the profile data, it saves with your username in data base to render to your profile picture as photo is linked with your username each username is attached with its profile picture.

```
class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    image = models.ImageField(default='default.jpg', upload_to='profile_pics')

    def __str__(self):
        return f'{self.user}'
```

Figure 6.26

This is the data model of profile picture:-

It save username from user that is login at that time who is uploading photo and photo as a string type varchar to return picture back we use request.FILES to get image back.

CHAPTER 7: BACKGROUND MATERIAL

Front-end:- HTML5,CSS,Javascript,Bootstrap



Back-end: -Django

django

Database: -SQLite



NAME:- SHASHWAT SHUKLA

EN. NO:- 171B117



BATCH:- B4

CGPA:- 7.9

EMAIL:- shashwatshukla077@gmail.com

MOB NO:- 7985460335

NAME:- AYUSH SRIVASTAVA

EN. NO:- 171B041



BATCH:- B2

CGPA:- 8.1

EMAIL:- srivastavaayush.alld@gmail.com

MOB NO:- 7388455394

NAME:- KARTIKEY CHANDRA

EN. NO:- 171B058



BATCH:- B2

CGPA:- 7.0

EMAIL:- kartikeychandraa@gmail.com

MOB NO:- 7525947775