A PROJECT REPORT ON

InstaBook

By

BHAVIK ARDESHNA
(19CEUOS032)

B.Tech CE Semester-IV
Subject: SEPP & SP

Guided by:
Prof. Brijesh Bhatt,
Prof. Jigar Pandya
Prof. Pinkal Chauhan

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

1. Abstract

InstaBook is a clone of social media site “Instagram” and Social networking site using *Django python full-stack web framework, Dharmsinh Desai University (DDU Nadiad)*.

InstaBook was smartly created using Django and other tools such that it supports all features which are provided by Instagram and I also maintained the design of InstaBook in such a way the user gets a complete feel of Instagram. InstaBook is easy to use and highly flexible with the user. New user needs to create account and login to respective user id. After successfully logged in user will be taken to the home page where the user can see all the posts posted by his/her followers and can like and comment on any post. InstaBook also provides an astonishing feature of Notification. So the user gets notified when any friend likes and comment in his/her post.

InstaBook users would also be able to put a new post with captions and tags using a user-friendly post page. The authenticated
User can also see their profile and set his/her profile picture with some bio descriptions. InstaBook also provides the list of all posts posted by the user in the Profile section.

InstaBook is created to change social media communication and many new users would get attracted and use InstaBook and be socialize.

2. <u>Tools/Technologies Used</u>

Technologies:

- Django
- Python3
- SQLite
- Bootstrap
- JavaScript
- TailwindCSS
- HTML

Tools

- Git
- Visual Studio Code

Platform

Local development server

Deployment

pythonanywhere

3. Software Requirement Specifications

1. **Product Scope**

This system is designed to enable the user to socialize with friends and able to share the memories in social media and also able to chat with them

2. **System Functional Requirements**

R.1: SignUp

Description: New user who visited instabook first time needs to create Account

R.1.1: Signup

INPUT: username, email id, password1, password2

PROCESSING: check syntax of email, match password1 and password2

OUTPUT: user-created

NEXT: validation is true, It will go to R.2

R.2: Login

Description: Once the user is successfully signed up, the user needs to login by entering username and password.

R.2.1:

STATE: the user is signed up

INPUT: enter username and password

OUTPUT: Takes to a user account page

PROCESSING: Validate entered details in the database

R.3: Create Post

Description: Once the user is logged in, he would be to make a new post.He/She
need to add a photo, tag, captions.

R.3.1: Log in

INPUT: username and password

OUTPUT: User able to create a new post and able to another user post

R.3.2: Create Post and Story

STATE: The user must be logged in.

INPUT: Photo, tag, caption

OUTPUT: All following will able to see the post and able to comment
and like the post.

R.3.3: Deleting Post

INPUT: press deleted icon is created post

OUTPUT: the post will be deleted

R.4: Create Profile and Following other users

Description: Once the user is logged in, he would be to make his profile by
adding a profile picture and some bio to it and also follow different people
on instabook and when he will create a post or story his friend will able to see
the post.

R.4.1: Search User

INPUT: Enter the username to search

OUTPUT: If the user will be available then it will be taken to his profile

R.4.2: Create Profile

STATE: The user must be logged in.

INPUT: Add a Profile picture, bio, status

OUTPUT: your profile is updated.

R.4.3: Change Profile

STATE: The user must be logged in and already created a profile.

INPUT: selecting edit profile option

OUTPUT: R.4.4

R.4.4: Edit Profile details

STATE: User must be logged in and already created a profile.

INPUT: update name,bio,status,profile picture

OUTPUT: profile is changed

R.5: Chatting with friends

Description: Once the user is logged in, he would be to chat with his friend like messenger.

R.5.1: Search Friend

INPUT: Enter the username of a friend to search

OUTPUT: User will able to send a message and also receive

R.5.2: Type message and Send

STATE: The user must be logged in.

INPUT: Add a message to send in the input field

OUTPUT: And on the other hand user will receive a message.

NEXT FUNCTION: R.6.1 if a message is sent and use is not in a chat window.

R.5.3: Deleting Message

STATE: The user must be logged in and the message is sent.

INPUT: a selected message which needs to be deleted

OUTPUT: selected message deleted

NEXT FUNCTION: R.5.2 type new message

R.6: Notifications

Description: Once the user is logged in, he would be to also see the notification of comments and messages arrived in a chat window.

R.6.1: Notify

Icons will show with small numbers that how many notifications are arrived in the user's account.

R.6.2: Send Reply and See all comments

STATE: The user must be logged in.

INPUT: Reply to the message by clicking reply message or comments

OUTPUT: And on the other hand user will receive a message.

NEXT FUNCTION: R.5.1 if a message is sent and use is not in a chat window.

R.6.3: Clear Notifications

STATE: The user must be logged in.

INPUT: pressing clear notification window

OUTPUT: it will remove all the notification and make notification window empty

R.7: Remove Account

Description: Once the user is logged in and has successfully created a profile and had put some post and wants to remove an account from instabook.

<u>R.7.1: Remove Account</u>

STATE: The user must be logged in.

INPUT: select remove account

OUTPUT: User will get notification of warning to confirm that he needs to remove the account.

NEXT: R.7.2

<u>R.7.2: User Removed from Instabook</u>

STATE: User must be logged in and getting the warning message in R.7.1

INPUT: confirming message the remove account message

OUTPUT: The user’s account will permanently be removed from the database.

3 Other Nonfunctional Requirements

1. Performance

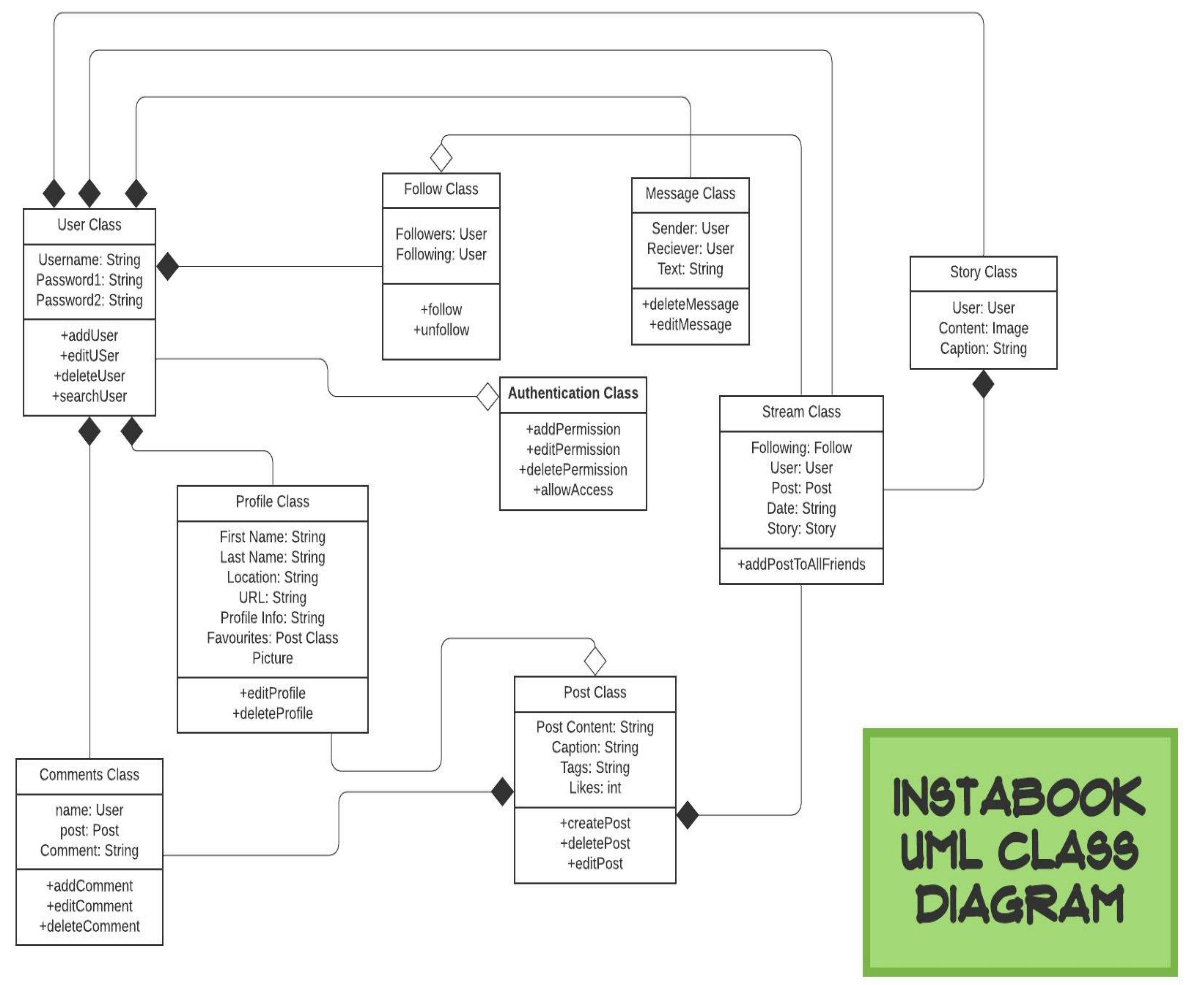
The system must be interactive and must not involve long delays. Though in case of opening the app components or loading the page the system shows the delays less than 2 seconds.

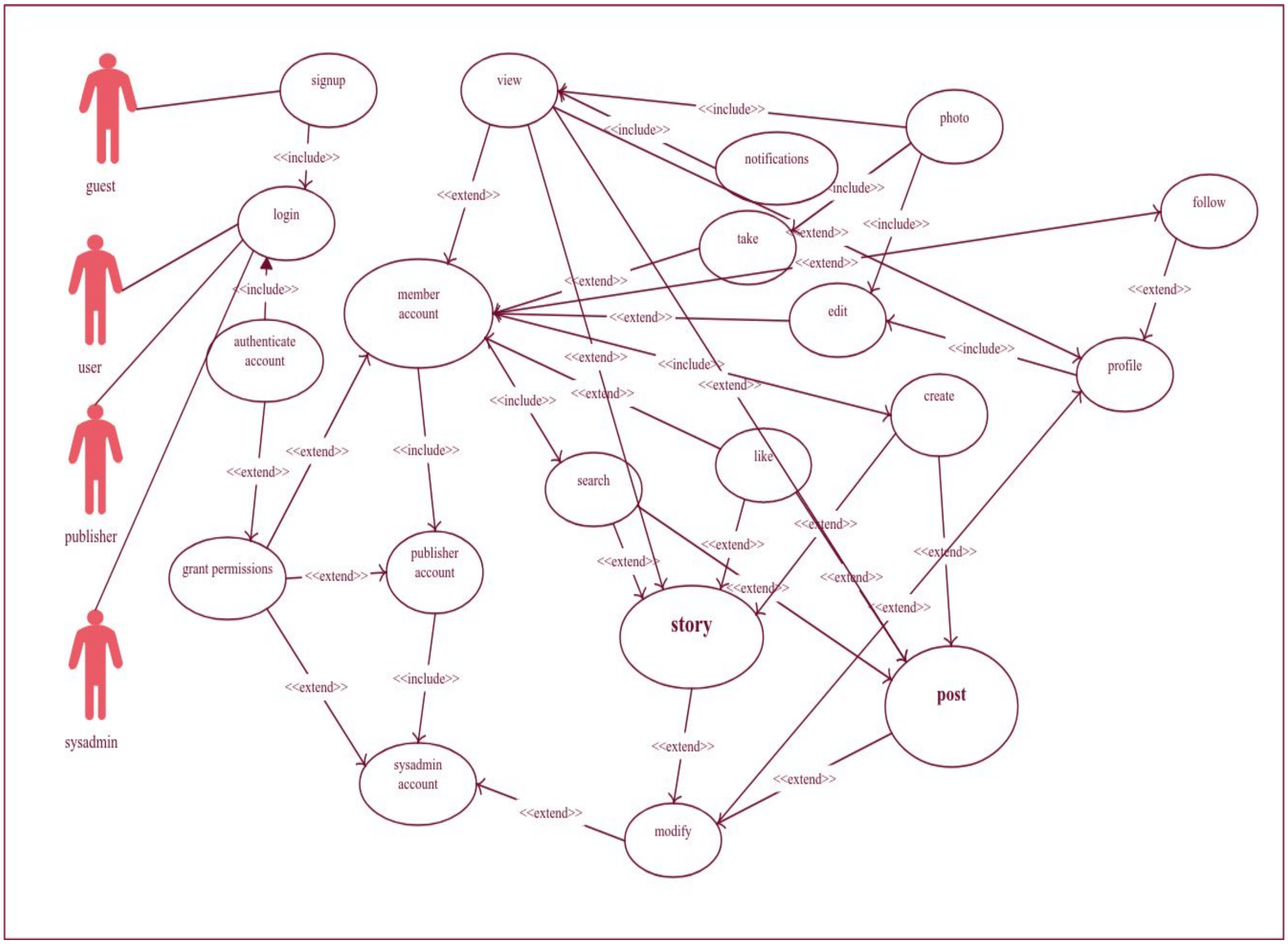
2. Safety

The users' data is highly personal. The system has authorization to avoid any un-authorized access to user's private data.

4. Design

4.1 Use Case Diagram




guest
user
publisher
sysadmin
signup
login
authenticate account
grant permissions
member account
publisher account
sysadmin account
view
photo
notifications
take
follow
edit
profile
create
like
search
story
post
modify
<<include>>
<<extend>>

Use Case Description :

➢ **Use case:** Social Media InstaBook

➢ **Summary**: User can create post and stories which can be seen by its followers.

➢ **Actors**: Social Media User

➢ **Preconditions**: User must be successfully signup before creating a profile and creating post.

➢ **Description**: The site started with a login and signup page, if a user is new can create an account by going to the signup page or else login with a username and password. After login user will able to see his/her profile. And also able to create posts and stories and even able to chat with his/her friends in private mode. Also, get notification of received unseen message and comment on his/her post if anyone commented. Users can also search for people on InstaBook who have created accounts in InstaBook and able to follow them. A user would also be able to edit his/her profile.

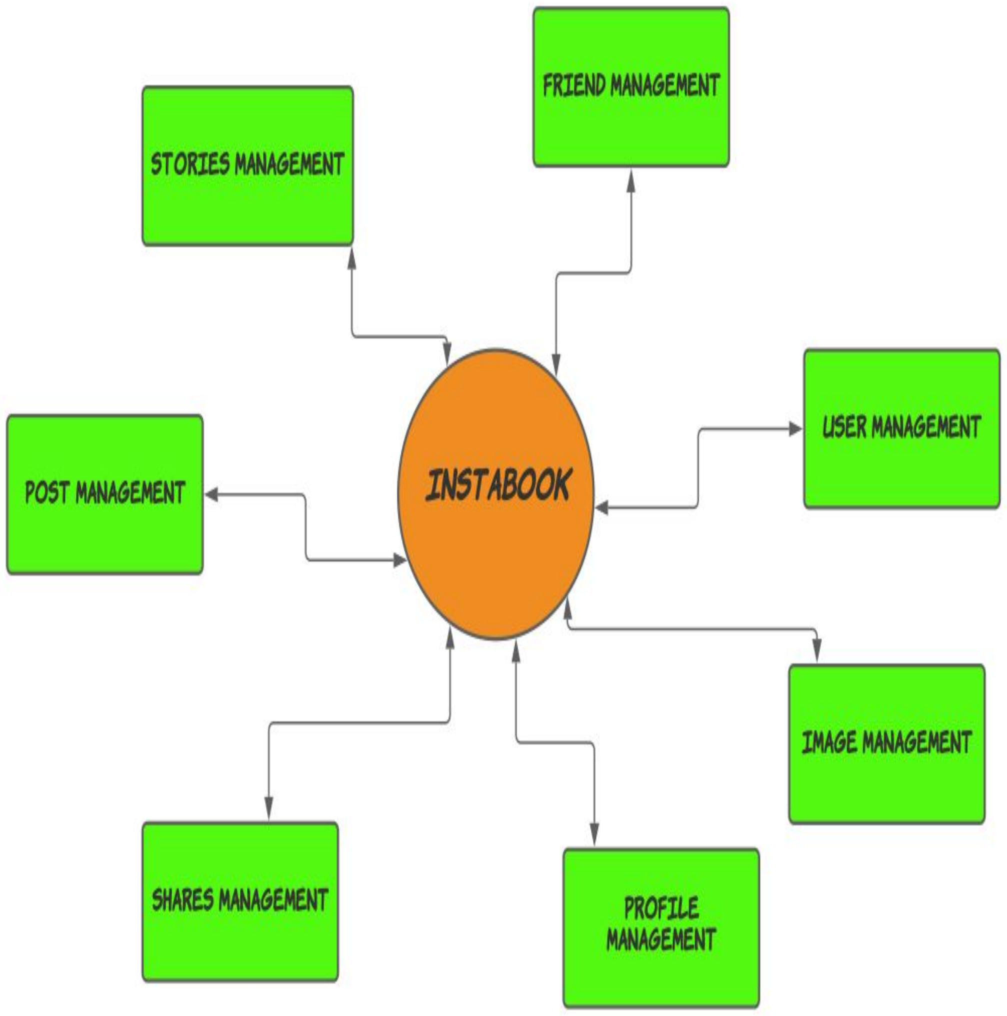
➢ **Exceptions** :

✗ Image not allowed: if user want to create a post and upload an image with the incorrect format of an image and if size exceeds size of default upload capacity provided by InstaBook database.

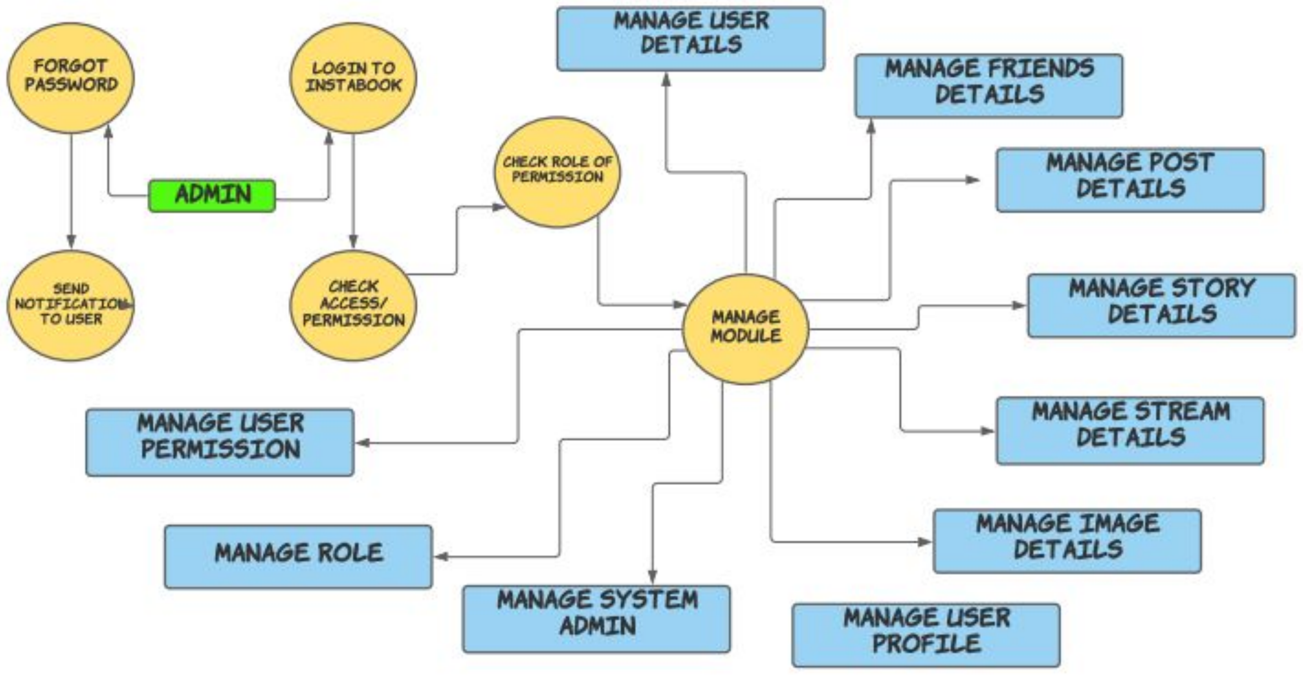
✗ Character exceed: if user try to enter a caption more than 100 words, the message will be displayed "Character exceed above 100, Error.".

✗ Deleted: if user deleted his/her post or story which they created before some time but now they don't want it so by selecting the delete button his/her post or story will be deleted with the message "Successfully Deleted".

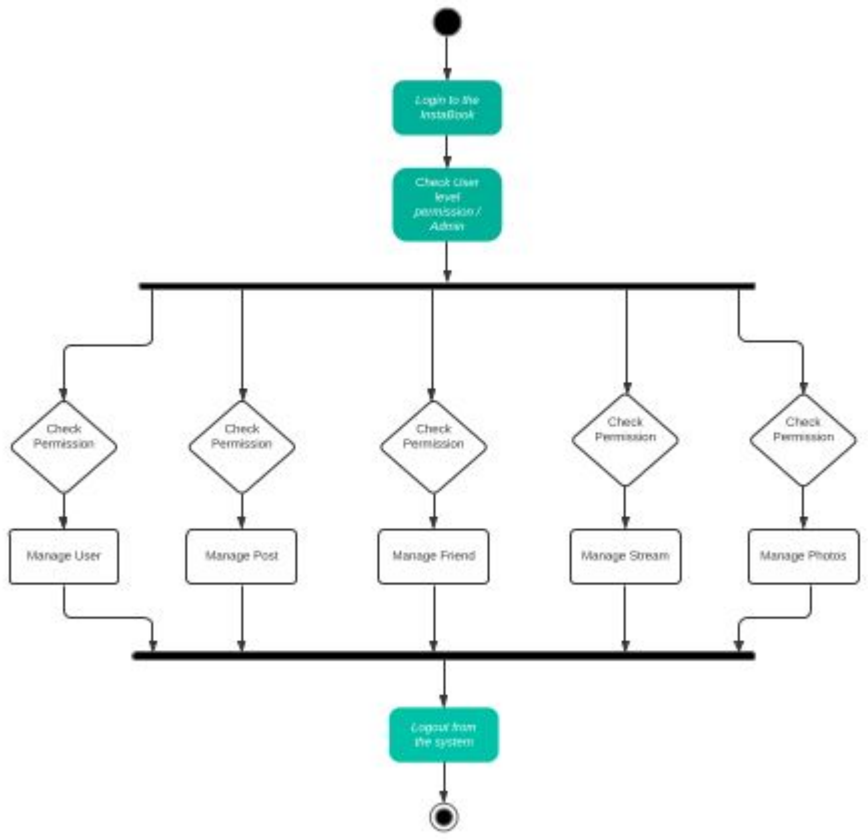
4.2 DFD Model and Structure Chart

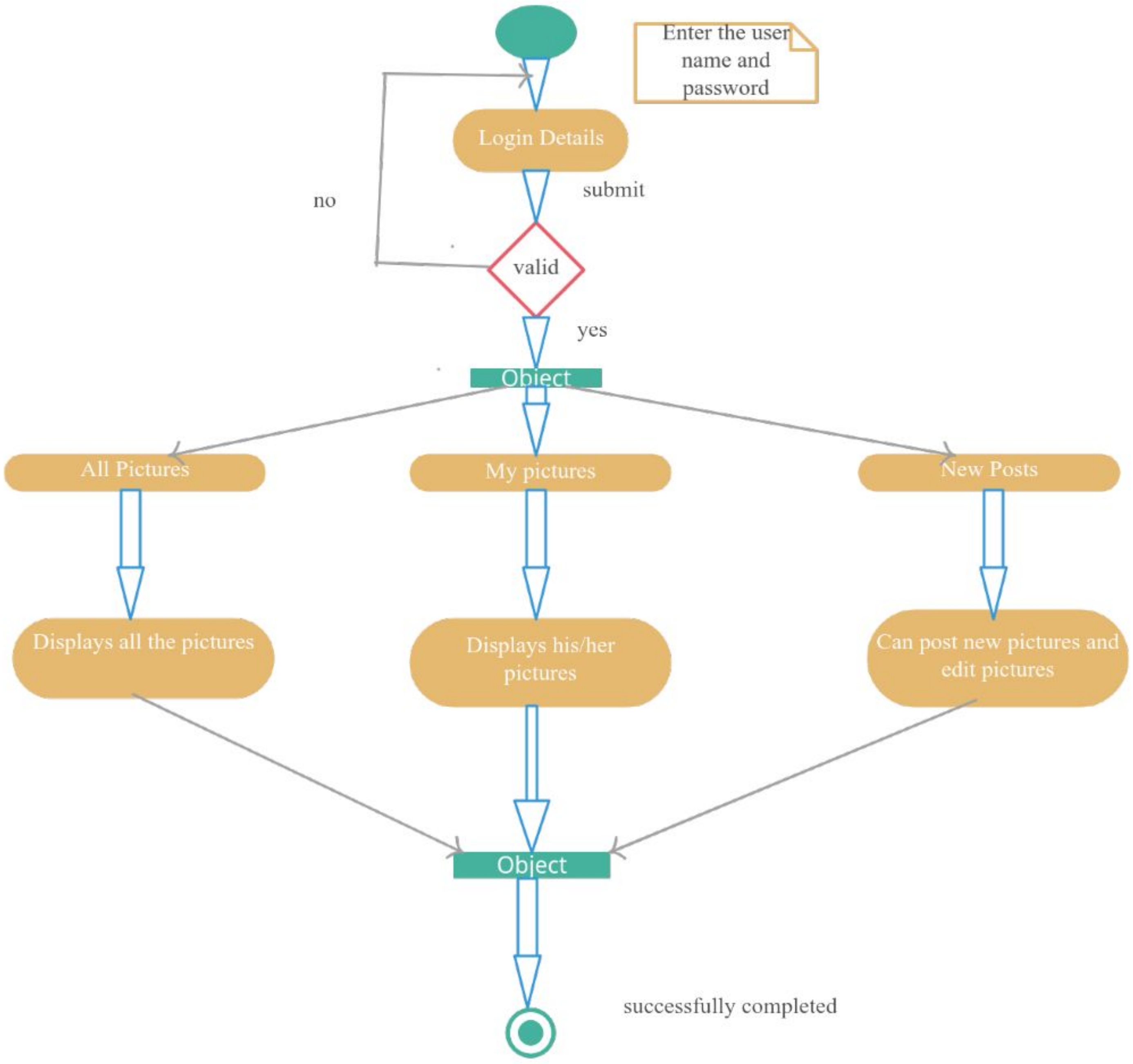


Sequence diagram of add version

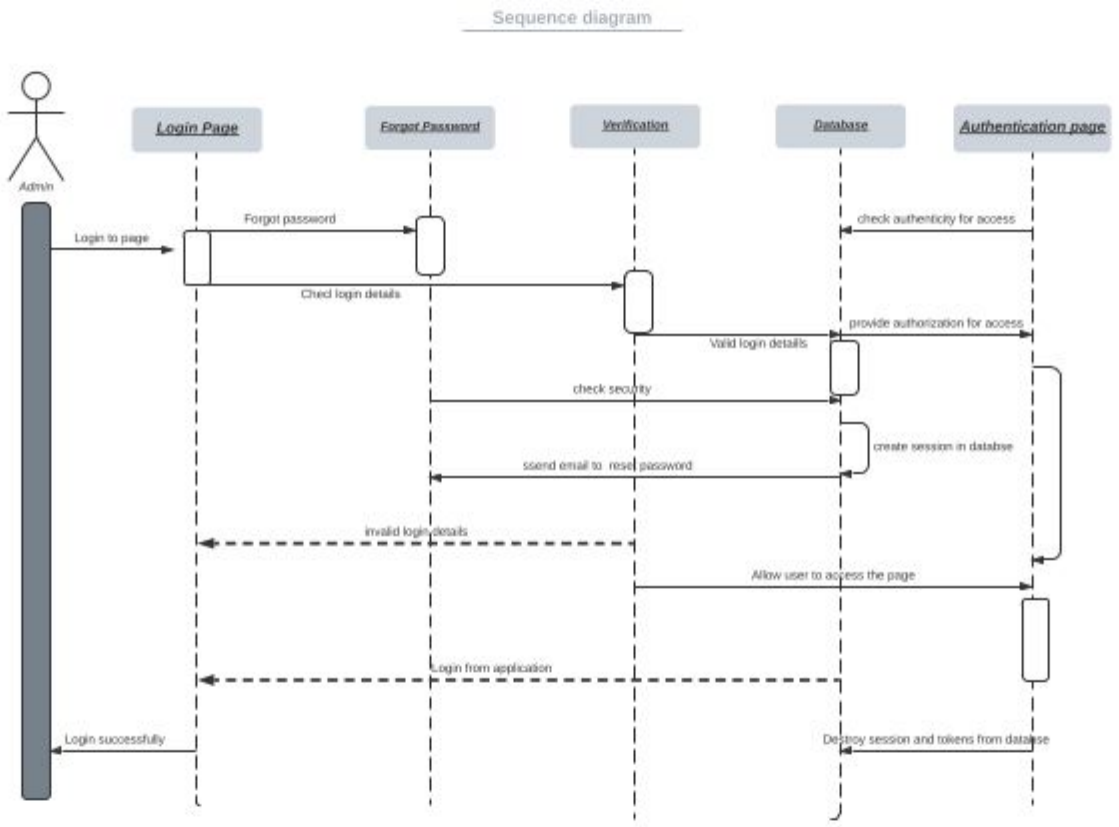

FORGOT PASSWORD
LOGIN TO INSTABOOK
ADMIN
SEND NOTIFICATION TO USER
CHECK ACCESS/ PERMISSION
CHECK ROLE OF PERMISSION
MANAGE MODULE
MANAGE USER DETAILS
MANAGE FRIENDS DETAILS
MANAGE POST DETAILS
MANAGE STORY DETAILS
MANAGE STREAM DETAILS
MANAGE IMAGE DETAILS
MANAGE USER PROFILE
MANAGE SYSTEM ADMIN
MANAGE ROLE
MANAGE USER PERMISSION

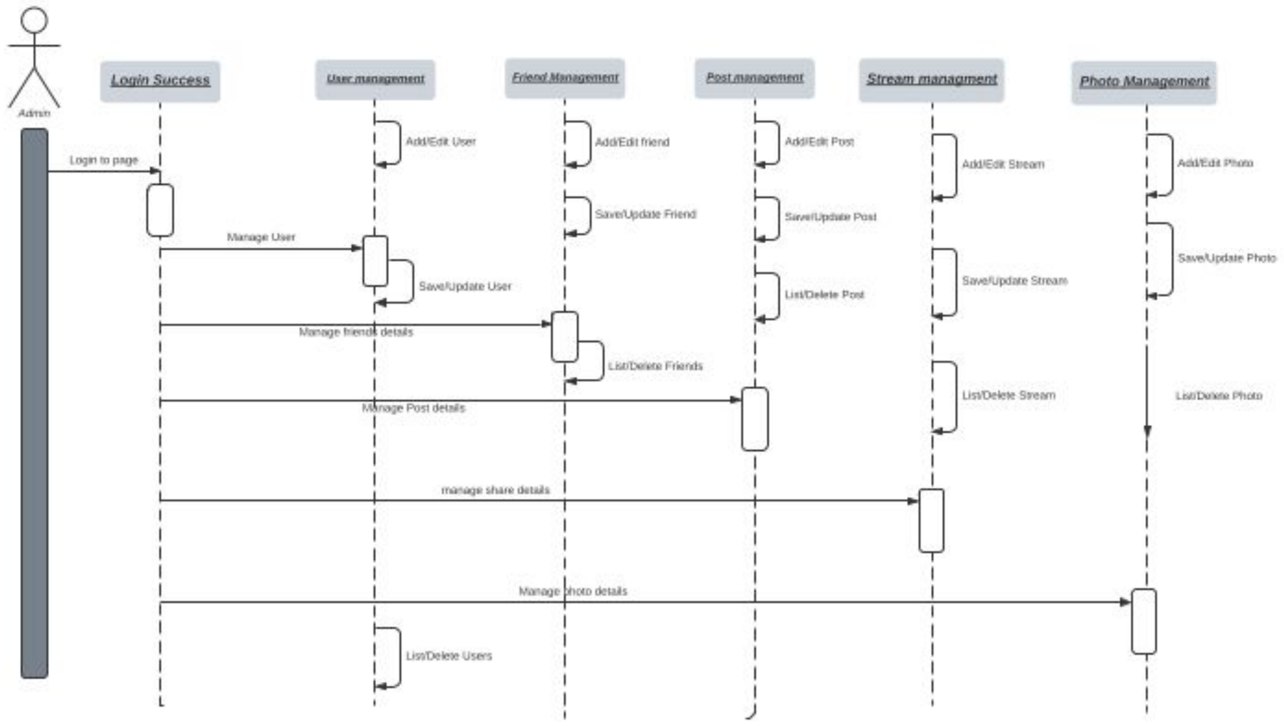
<u>4.3 ActivityDiagram</u>




Enter the user name and password
Login Details
submit
no
valid
yes
Object
All Pictures
My pictures
New Posts
Displays all the pictures
Displays his/her pictures
Can post new pictures and edit pictures
Object
successfully completed

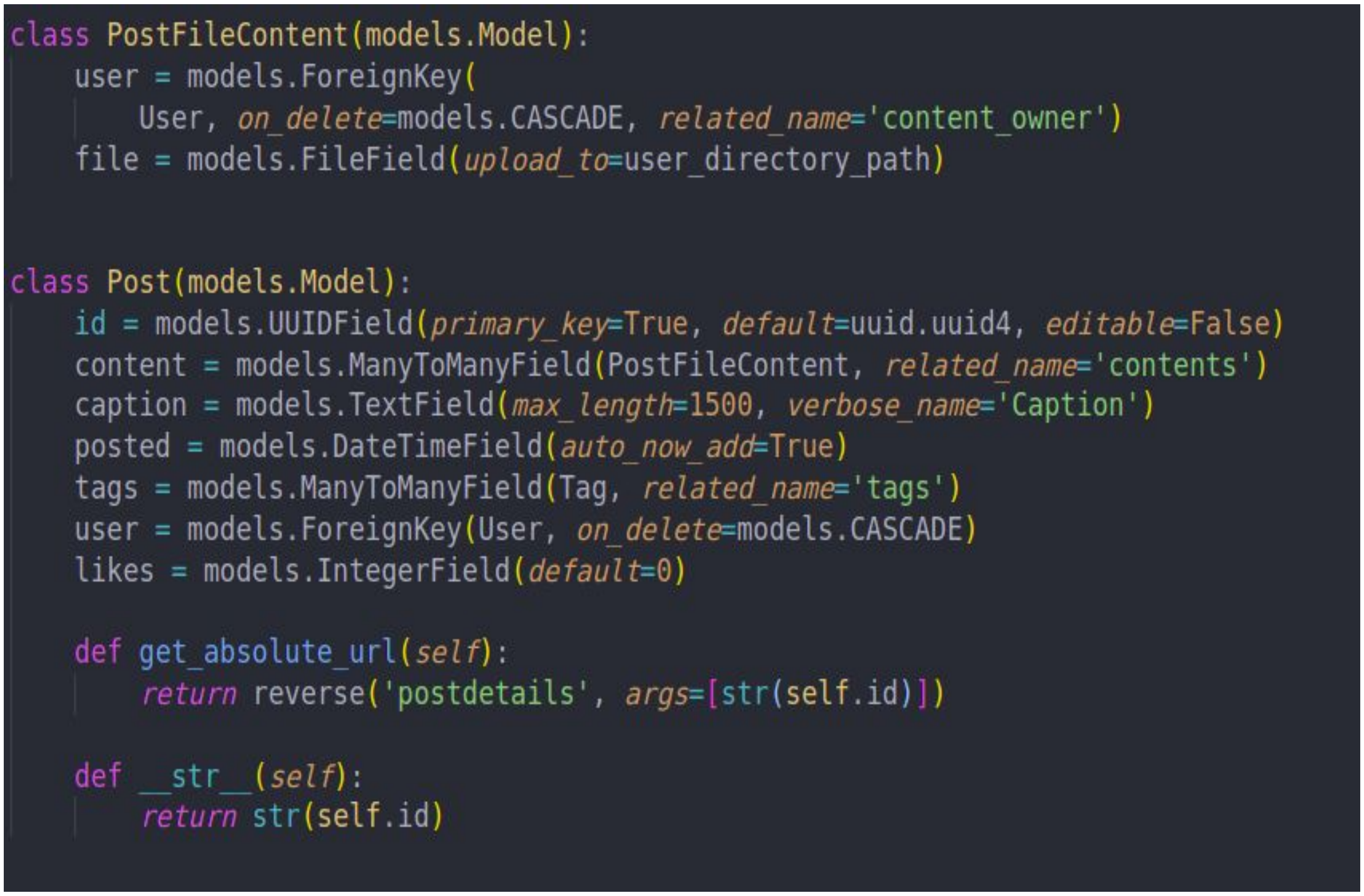
4.4 Sequence Diagram





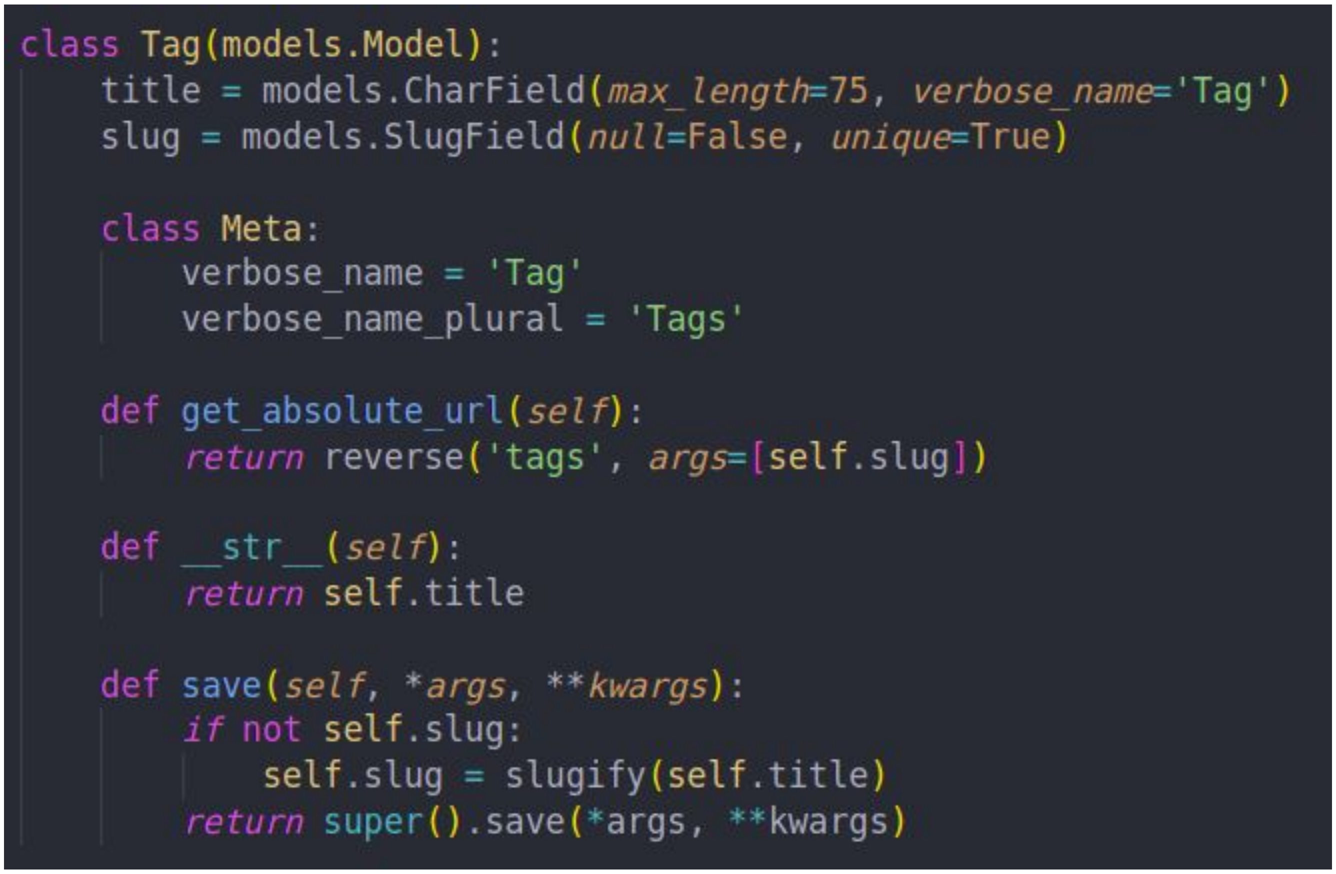
5. InstaBook Django Models

Post Models

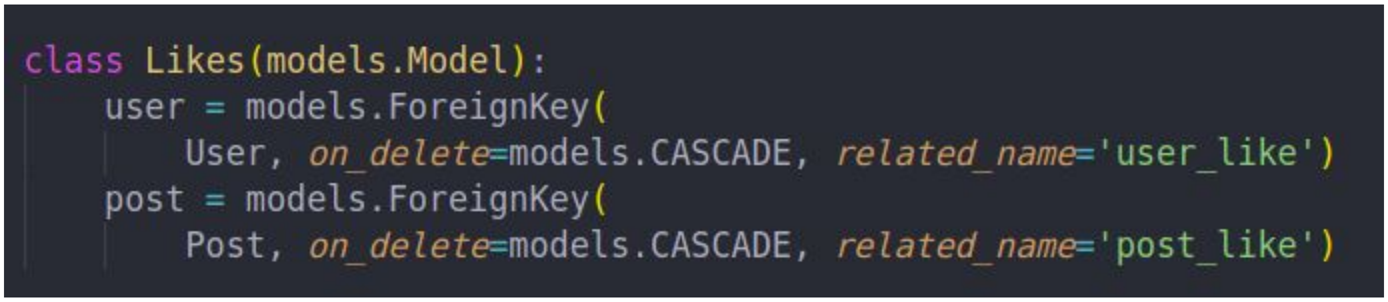


```
class PostFileContent(models.Model):
    user = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name='content_owner')
    file = models.FileField(upload_to=user_directory_path)


class Post(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    content = models.ManyToManyField(PostFileContent, related_name='contents')
    caption = models.TextField(max_length=1500, verbose_name='Caption')
    posted = models.DateTimeField(auto_now_add=True)
    tags = models.ManyToManyField(Tag, related_name='tags')
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    likes = models.IntegerField(default=0)

    def get_absolute_url(self):
        return reverse('postdetails', args=[str(self.id)])

    def __str__(self):
        return str(self.id)
```

Tag and Likes Models

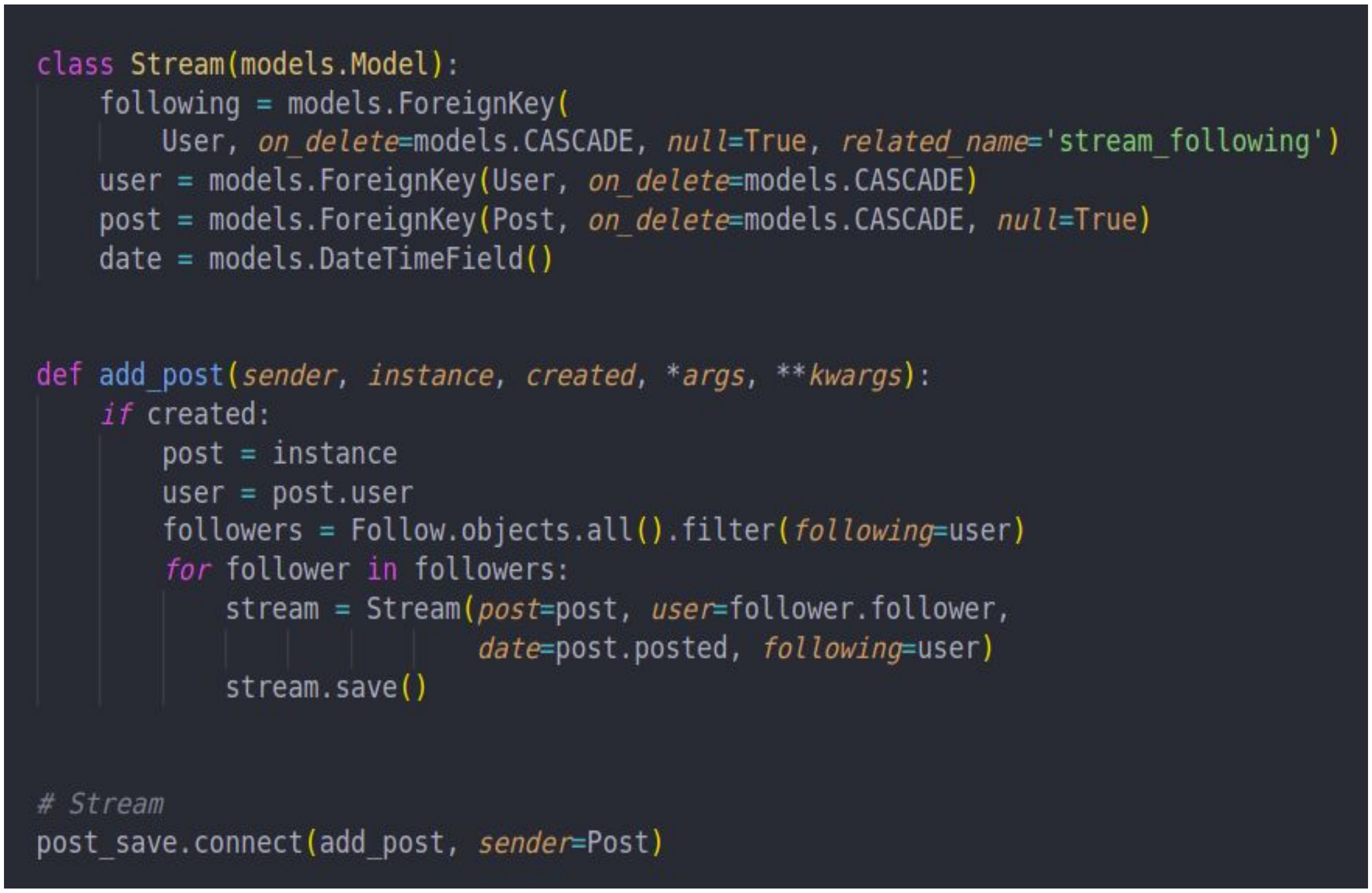


```python
class Tag(models.Model):
    title = models.CharField(max_length=75, verbose_name='Tag')
    slug = models.SlugField(null=False, unique=True)

    class Meta:
        verbose_name = 'Tag'
        verbose_name_plural = 'Tags'

    def get_absolute_url(self):
        return reverse('tags', args=[self.slug])

    def __str__(self):
        return self.title

    def save(self, *args, **kwargs):
        if not self.slug:
            self.slug = slugify(self.title)
        return super().save(*args, **kwargs)
```



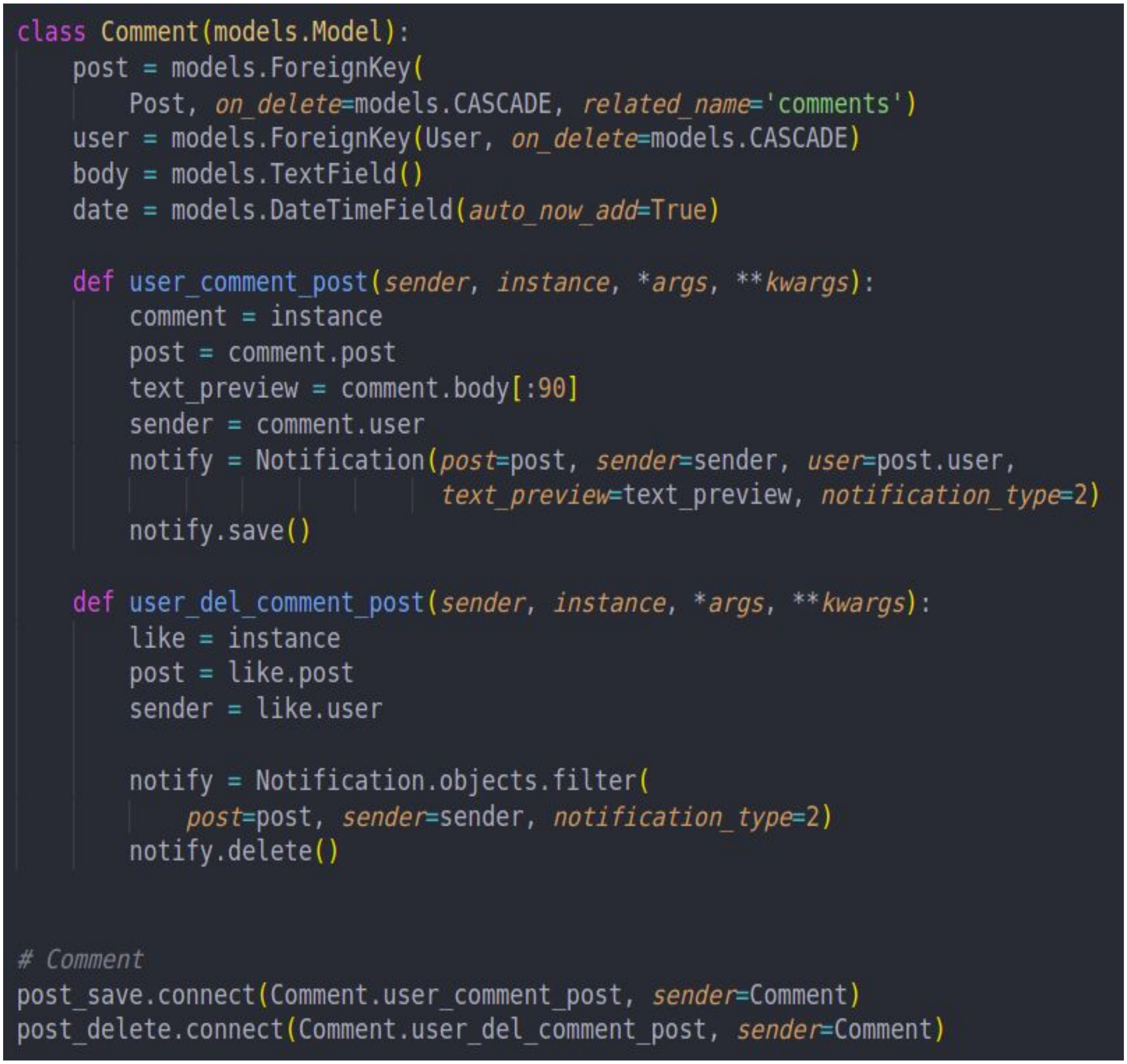
```python
class Likes(models.Model):
    user = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name='user_like')
    post = models.ForeignKey(
        Post, on_delete=models.CASCADE, related_name='post_like')
```

Stream Models



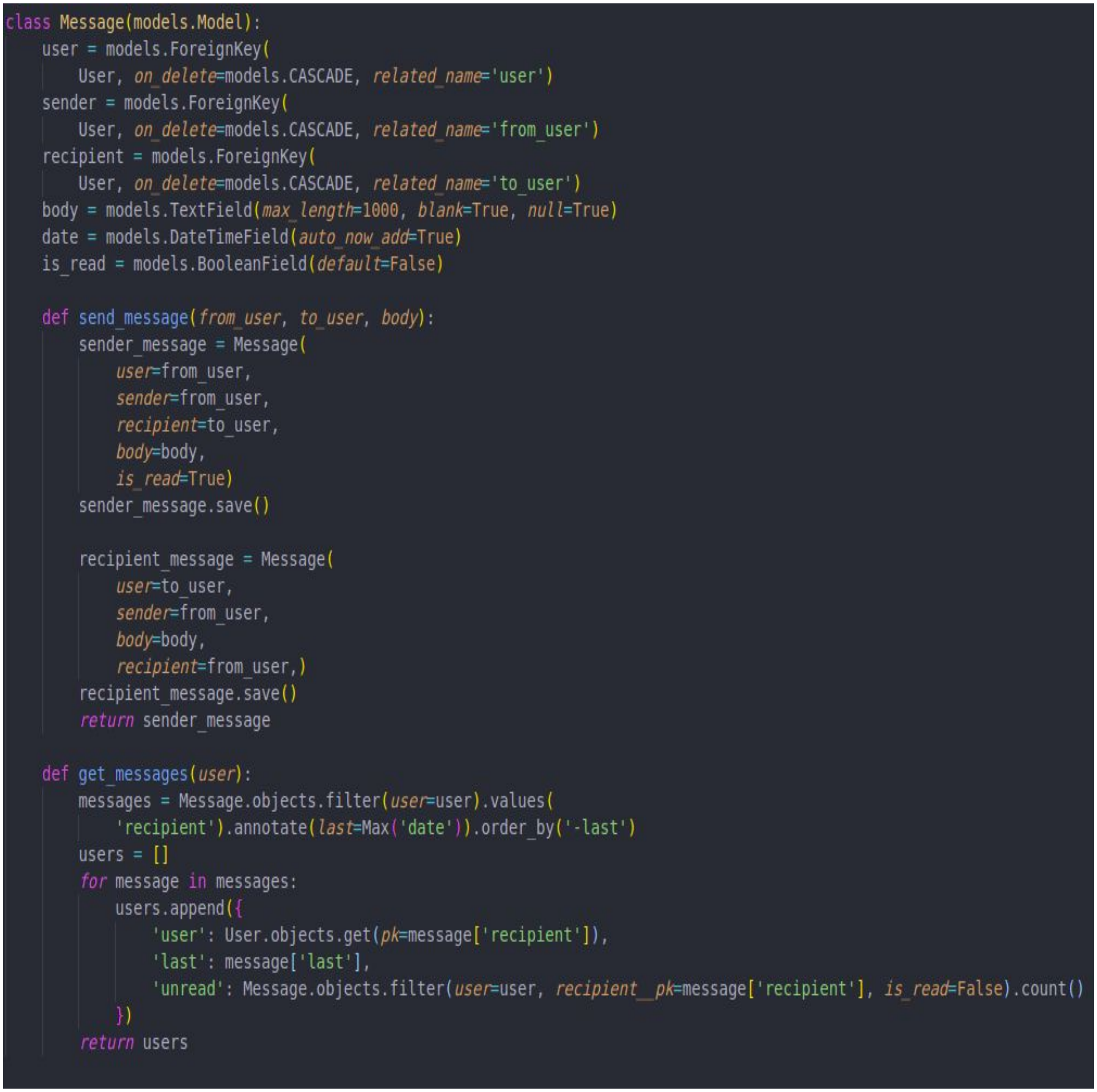
```python
class Stream(models.Model):
    following = models.ForeignKey(
        User, on_delete=models.CASCADE, null=True, related_name='stream_following')
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    post = models.ForeignKey(Post, on_delete=models.CASCADE, null=True)
    date = models.DateTimeField()


def add_post(sender, instance, created, *args, **kwargs):
    if created:
        post = instance
        user = post.user
        followers = Follow.objects.all().filter(following=user)
        for follower in followers:
            stream = Stream(post=post, user=follower.follower,
                            date=post.posted, following=user)
            stream.save()


# Stream
post_save.connect(add_post, sender=Post)
```

Comment Models



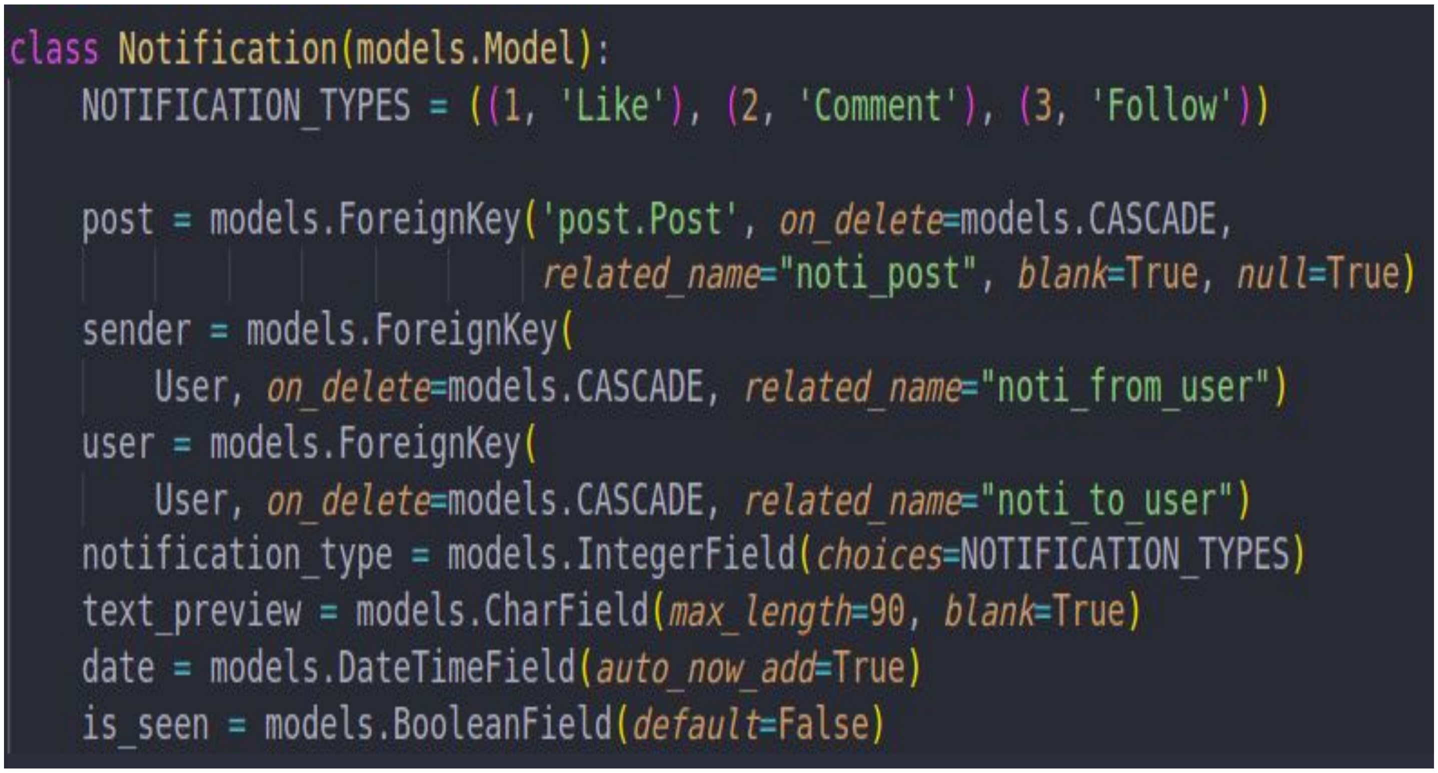
```
class Comment(models.Model):
    post = models.ForeignKey(
        Post, on_delete=models.CASCADE, related_name='comments')
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    body = models.TextField()
    date = models.DateTimeField(auto_now_add=True)

    def user_comment_post(sender, instance, *args, **kwargs):
        comment = instance
        post = comment.post
        text_preview = comment.body[:90]
        sender = comment.user
        notify = Notification(post=post, sender=sender, user=post.user,
                              text_preview=text_preview, notification_type=2)
        notify.save()

    def user_del_comment_post(sender, instance, *args, **kwargs):
        like = instance
        post = like.post
        sender = like.user

        notify = Notification.objects.filter(
            post=post, sender=sender, notification_type=2)
        notify.delete()


# Comment
post_save.connect(Comment.user_comment_post, sender=Comment)
post_delete.connect(Comment.user_del_comment_post, sender=Comment)
```

Direct Models



```
class Message(models.Model):
    user = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name='user')
    sender = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name='from_user')
    recipient = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name='to_user')
    body = models.TextField(max_length=1000, blank=True, null=True)
    date = models.DateTimeField(auto_now_add=True)
    is_read = models.BooleanField(default=False)

    def send_message(from_user, to_user, body):
        sender_message = Message(
            user=from_user,
            sender=from_user,
            recipient=to_user,
            body=body,
            is_read=True)
        sender_message.save()

        recipient_message = Message(
            user=to_user,
            sender=from_user,
            body=body,
            recipient=from_user,)
        recipient_message.save()
        return sender_message

    def get_messages(user):
        messages = Message.objects.filter(user=user).values(
            'recipient').annotate(last=Max('date')).order_by('-last')
        users = []
        for message in messages:
            users.append({
                'user': User.objects.get(pk=message['recipient']),
                'last': message['last'],
                'unread': Message.objects.filter(user=user, recipient__pk=message['recipient'], is_read=False).count()
            })
        return users
```

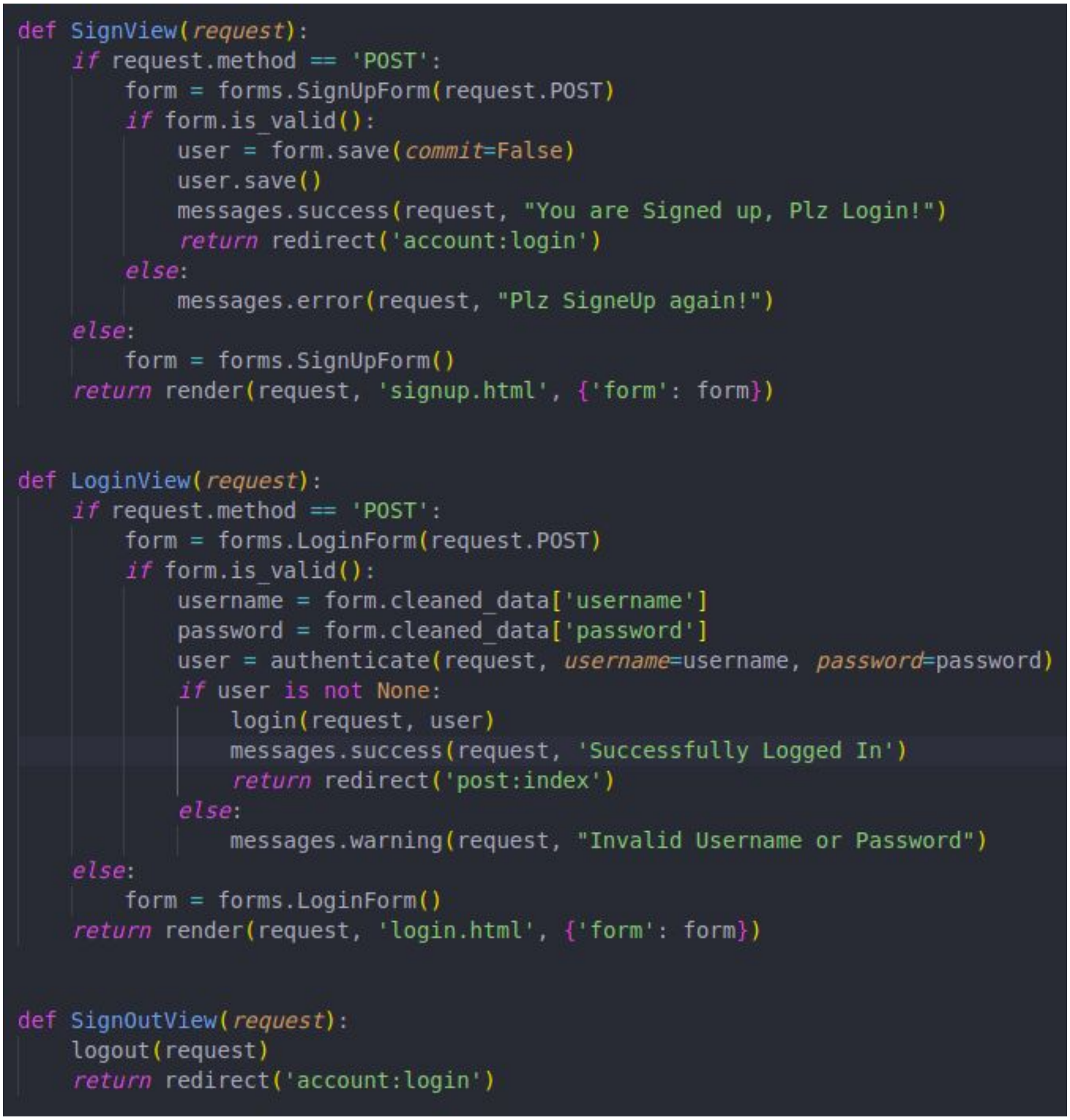
Notification Models



```python
class Notification(models.Model):
    NOTIFICATION_TYPES = ((1, 'Like'), (2, 'Comment'), (3, 'Follow'))

    post = models.ForeignKey('post.Post', on_delete=models.CASCADE,
                             related_name="noti_post", blank=True, null=True)
    sender = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name="noti_from_user")
    user = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name="noti_to_user")
    notification_type = models.IntegerField(choices=NOTIFICATION_TYPES)
    text_preview = models.CharField(max_length=90, blank=True)
    date = models.DateTimeField(auto_now_add=True)
    is_seen = models.BooleanField(default=False)
```

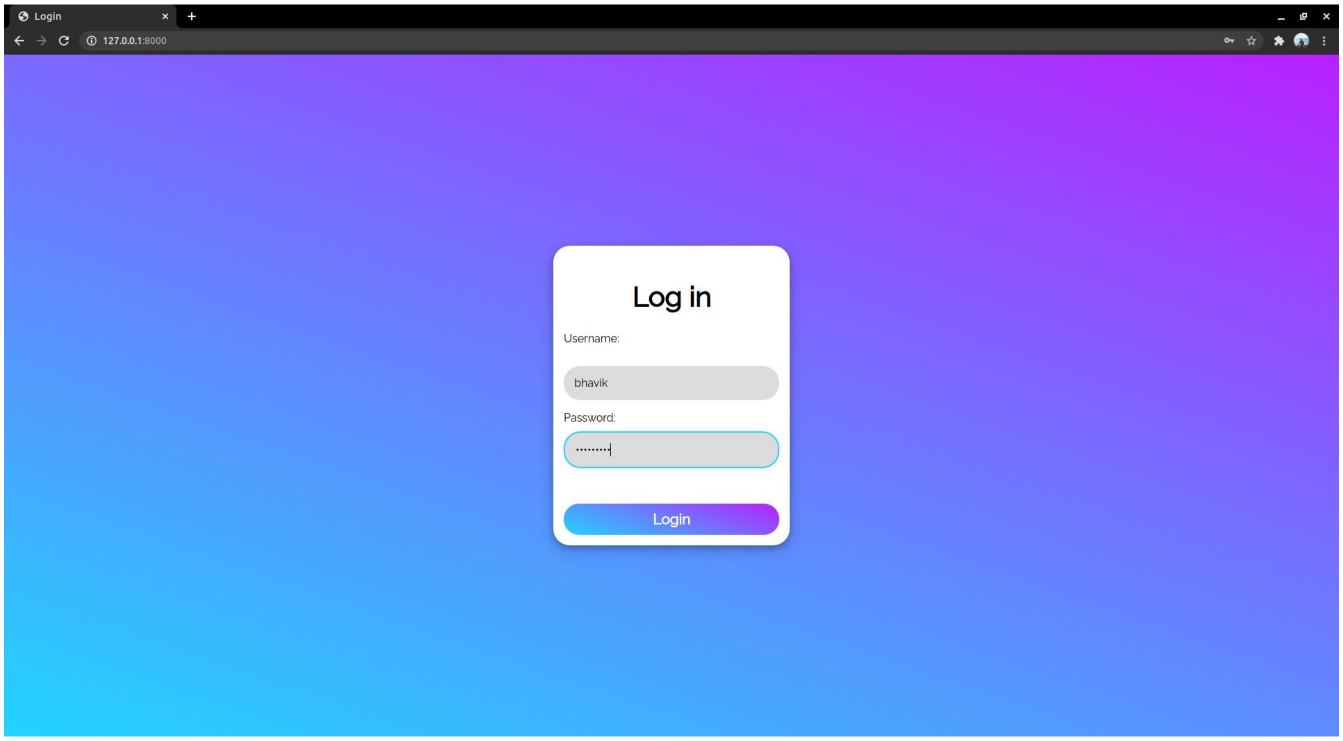
6. InstaBook Django Views

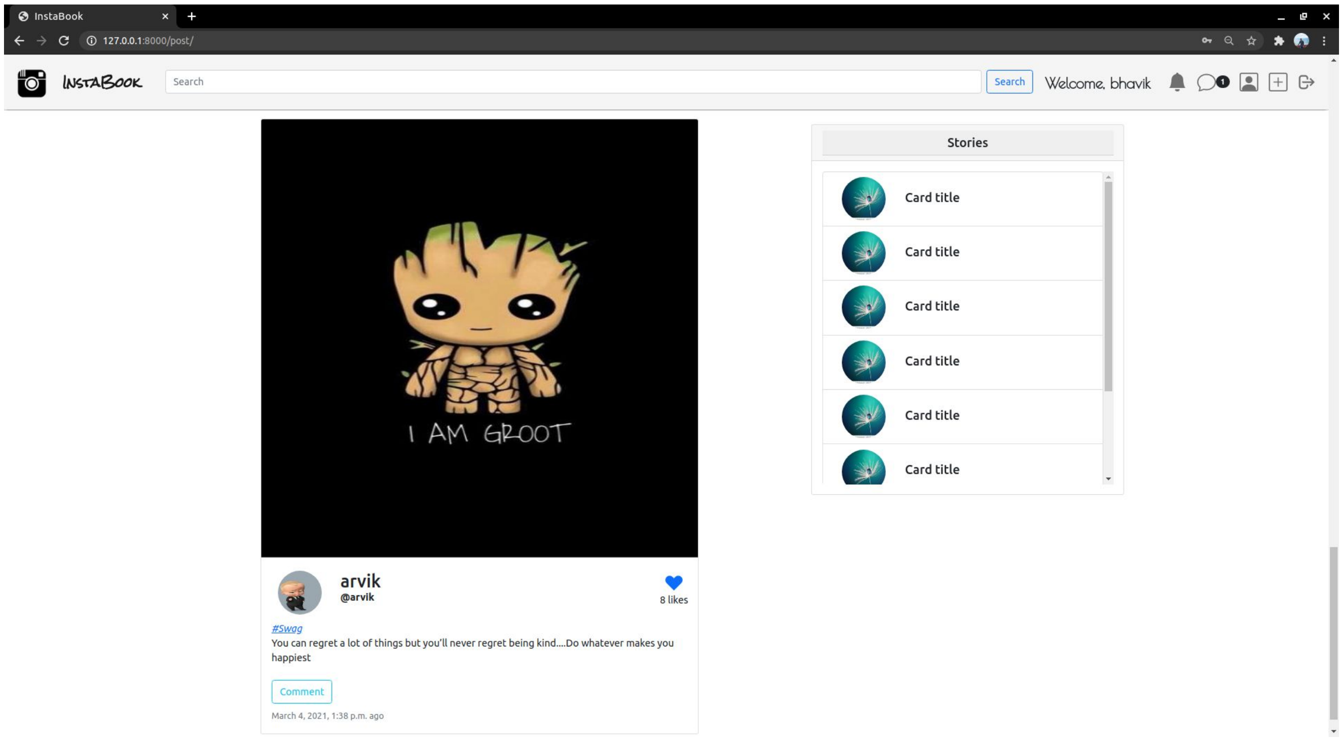
Login and SignUp View



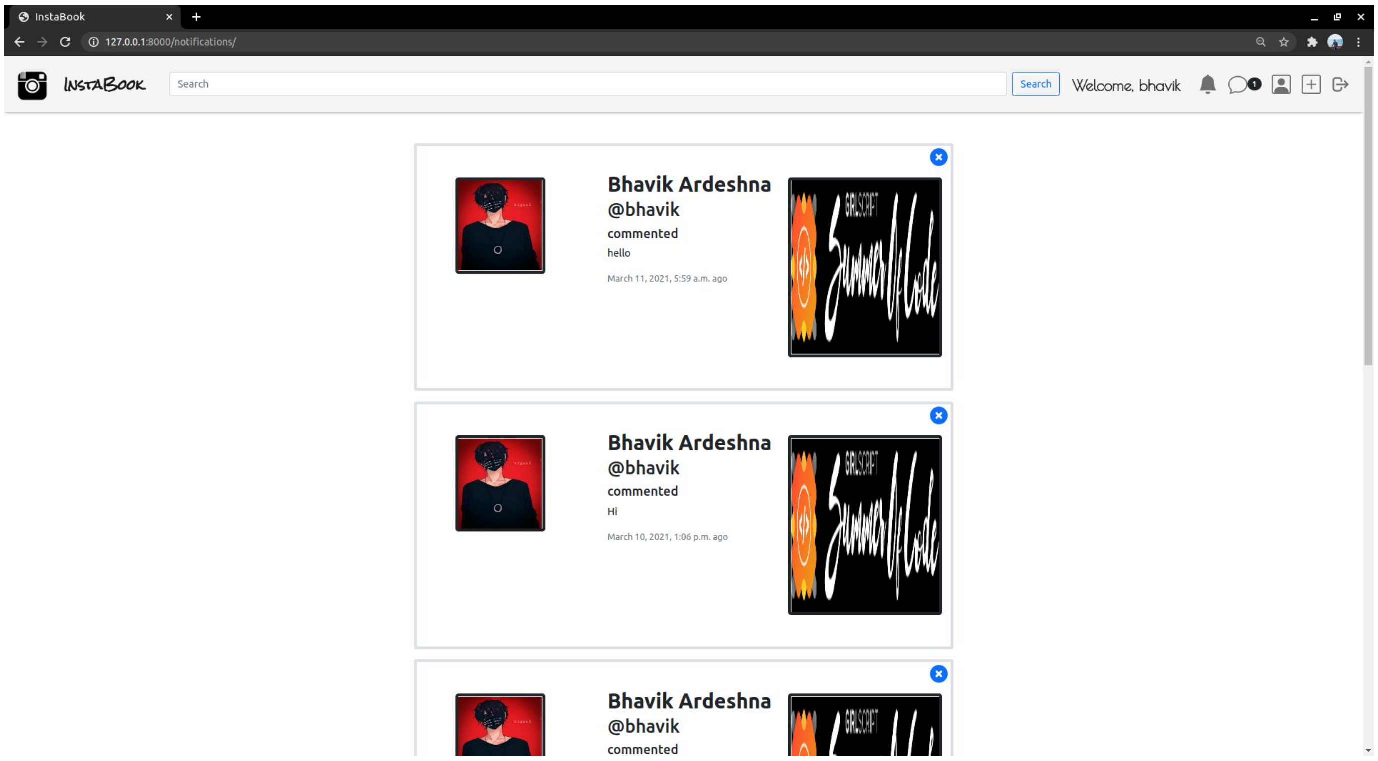
```python
def SignView(request):
    if request.method == 'POST':
        form = forms.SignUpForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.save()
            messages.success(request, "You are Signed up, Plz Login!")
            return redirect('account:login')
        else:
            messages.error(request, "Plz SigneUp again!")
    else:
        form = forms.SignUpForm()
    return render(request, 'signup.html', {'form': form})


def LoginView(request):
    if request.method == 'POST':
        form = forms.LoginForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            user = authenticate(request, username=username, password=password)
            if user is not None:
                login(request, user)
                messages.success(request, 'Successfully Logged In')
                return redirect('post:index')
            else:
                messages.warning(request, "Invalid Username or Password")
    else:
        form = forms.LoginForm()
    return render(request, 'login.html', {'form': form})


def SignOutView(request):
    logout(request)
    return redirect('account:login')
```

7. Screenshots





You can regret a lot of things but you'll never regret being kind....Do whatever makes you happiest



InstaBook

127.0.0.1:8000/notifications/

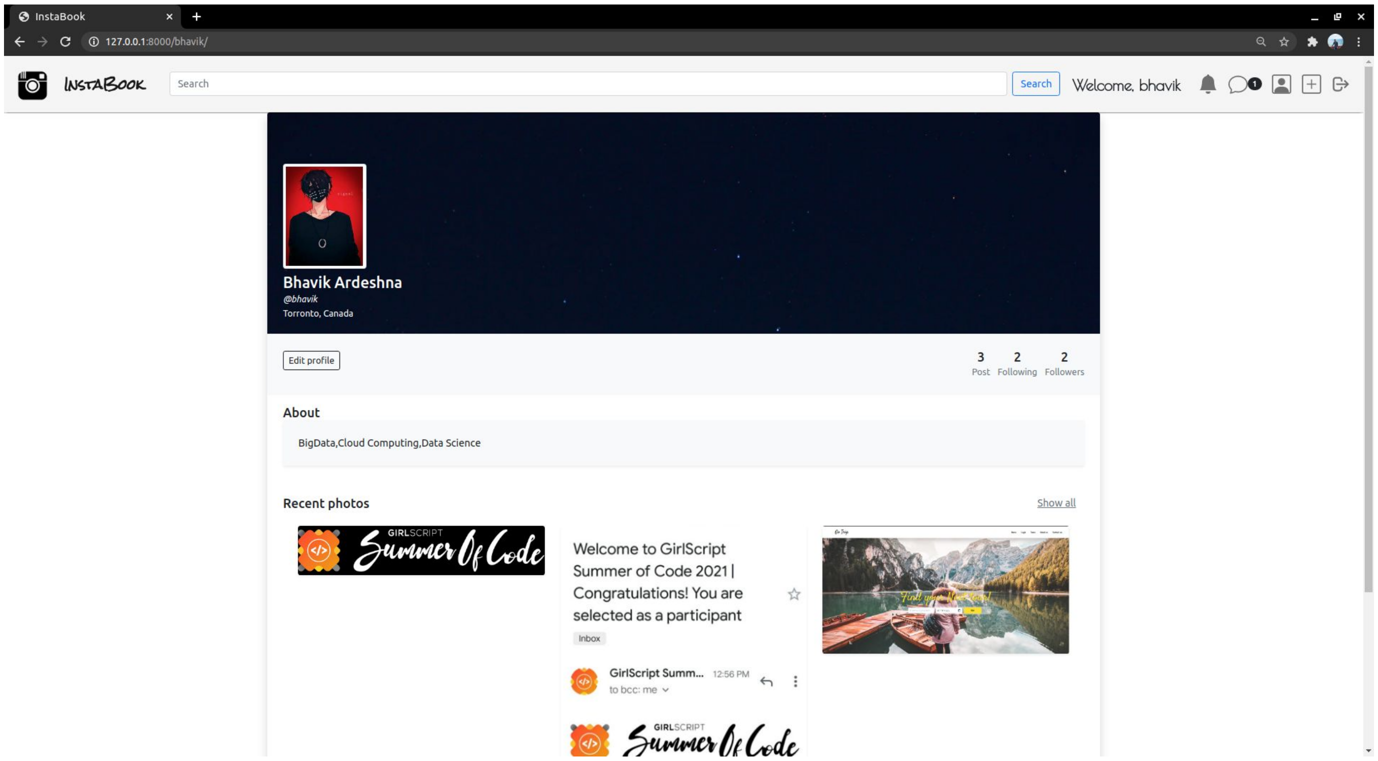
Search

Welcome, bhavik

Bhavik Ardeshna
@bhavik
commented
hello
March 11, 2021, 5:59 a.m. ago

Bhavik Ardeshna
@bhavik
commented
Hi
March 10, 2021, 1:06 p.m. ago

Bhavik Ardeshna
@bhavik
commented



InstaBook

127.0.0.1:8000/bhavik/

Search

Welcome, bhavik

Bhavik Ardeshna
@bhavik
Torronto, Canada

Edit profile

3	2	2
Post	Following	Followers

About

BigData,Cloud Computing,Data Science

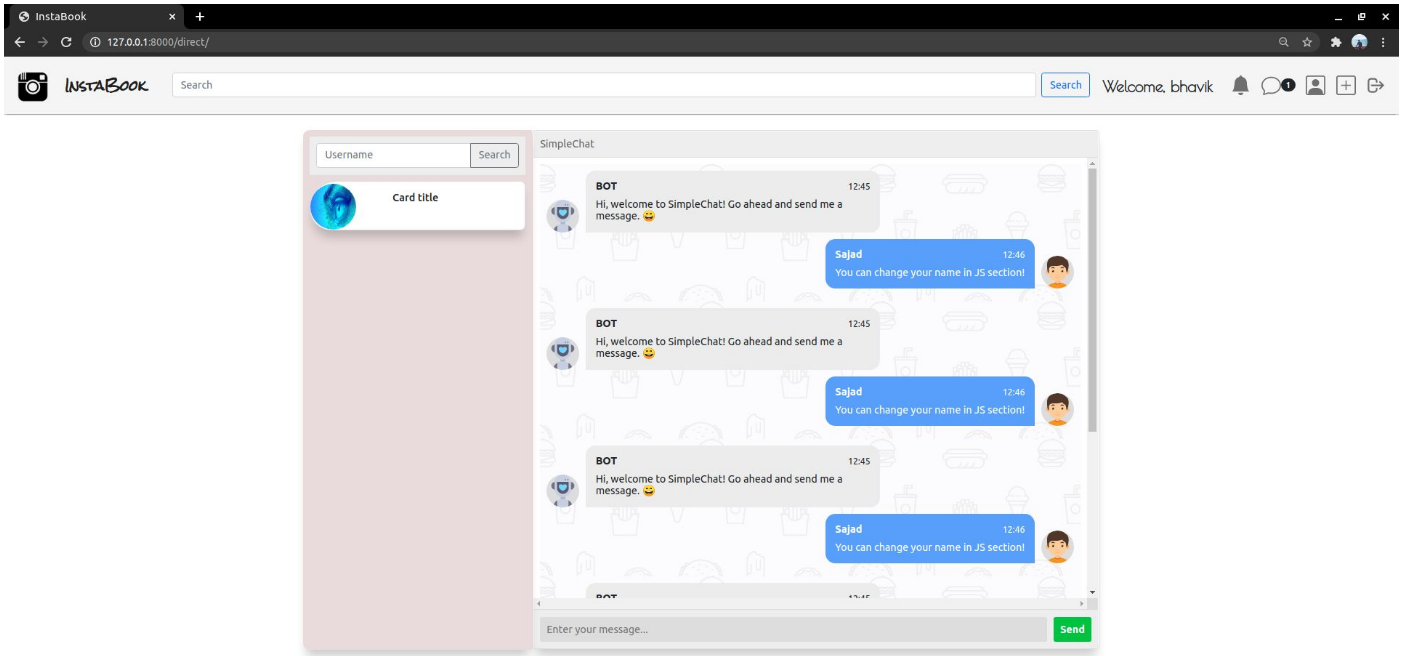
Recent photos

Show all

Welcome to GirlScript Summer of Code 2021 | Congratulations! You are selected as a participant

Inbox

GirlScript Summ... 12:56 PM
to bcc: me



BOT 12:45
Hi, welcome to SimpleChat! Go ahead and send me a message. 😄

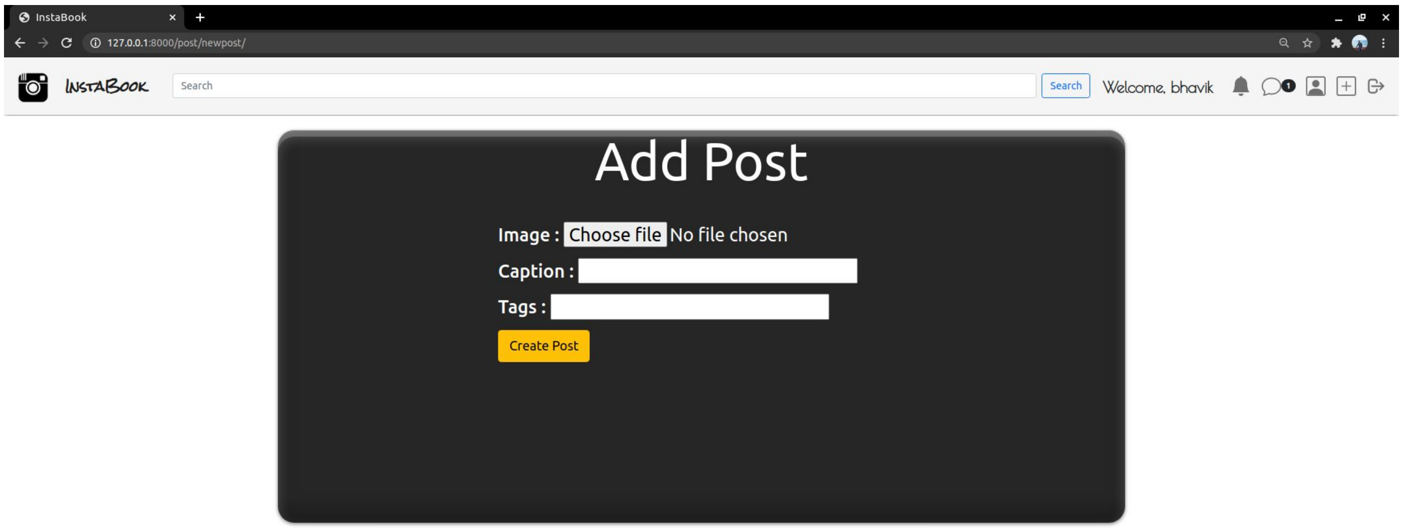
Sajad 12:46
You can change your name in JS section!

BOT 12:45
Hi, welcome to SimpleChat! Go ahead and send me a message. 😄

Sajad 12:46
You can change your name in JS section!

BOT 12:45
Hi, welcome to SimpleChat! Go ahead and send me a message. 😄

Sajad 12:46
You can change your name in JS section!

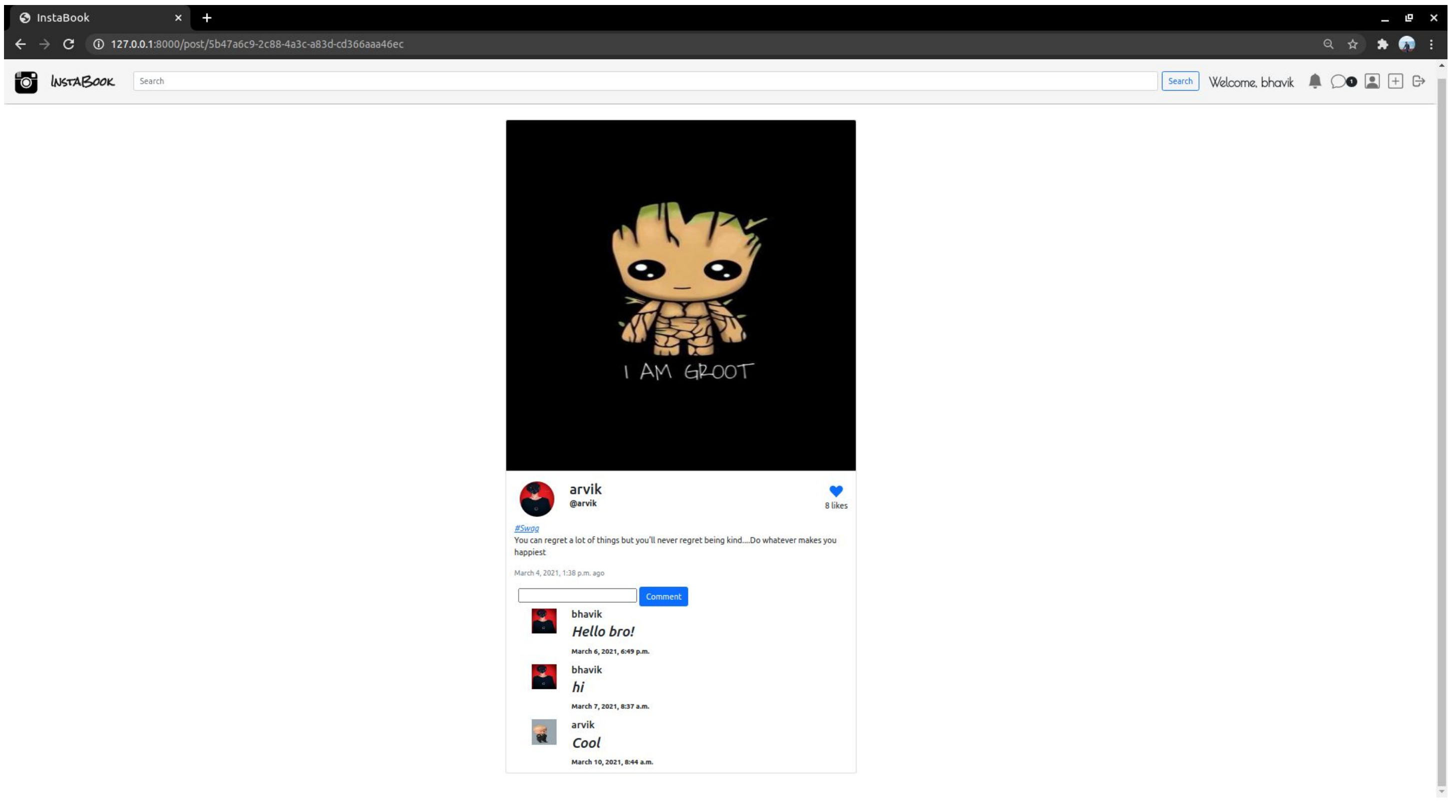


Add Post

Image : Choose file No file chosen

Caption :

Tags :



arvik
@arvik
8 likes

#Swag
You can regret a lot of things but you'll never regret being kind....Do whatever makes you happiest

March 4, 2021, 1:38 p.m. ago

bhavik
Hello bro!
March 6, 2021, 6:49 p.m.

bhavik
hi
March 7, 2021, 8:37 a.m.

arvik
Cool
March 10, 2021, 8:44 a.m.

Deployed Project Link : [InstaBook.com]

GitHub Repository : [InstaBook Github]

7. 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:

- Login
- User authentication
- Logout
- Add Post with caption and tags
- Edit Profile and change Profile picture
- Send Review Requests
- Able to like on post
- Also comment on the post
- Notification features

After the implementation and coding of system, comprehensive testing was performed on the system to determine the errors and possible flaws in the system.

8. Limitations and Future Enhancements

We are able to implement the functionality model of the "InstaBook".

Currently chatting feature is not working because I want to implement chatting feature without using external library.

Model and UI for chatting is ready but I need work differently on view section.

In future publication it will be added.

9. Reference / Bibliography

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

stackoverflow.com

medium.com

github.com

docs.djangoproject.com