Graphical Password Authentication System

Group members:

1. Aakanksha Sonawane 111903001
2. Aishwarya Yabaji 111903010
3. Kruti Walko 111903042

Problem Statement:

Graphical Password Authentication

Objectives:

1. To provides strong security against bot attacks or hackers.
2. To protects systems vulnerable to attacks.
3. System is user-friendly and has simple interface

Motivation:

As we know, password is a secret word or string of characters, numbers and symbols used for user authentication to prove his/her identity and going to access resources.

With increasing technical advancements the world is becoming digital at high pace. Everything is online there is risk of cybercrimes and privacy breaches is also increasing. Password plays a huge role in keeping our data safe online as well as offline.

Graphical Password Authentication is knowledge based where the user has to select from images, in a specific order, presented to them in a graphical user interface(GUI). It uses some combination of graphical images replacing the regular passwords. It is more easy to recognize images than alpha-numeric passwords. It is user friendly and provides higher security.

Summary of SRS:

1.Purpose : This project “Graphical passwords Authentication” may offer better security than text-based passwords because most of the people use regular, popular passwords everywhere and are prone to social engineering attacks. So graphical passwords can put stop to many attacks of this kind.

Product Perspective : The main aim of this project is to enhance security of user login using graphical passwords.

Product Functions: In this project, we are using functions such as:

1. Graphical password generation

2. Authentication

3.Reset password by sending link to user’s email id.

Design and Implementation Constraints :

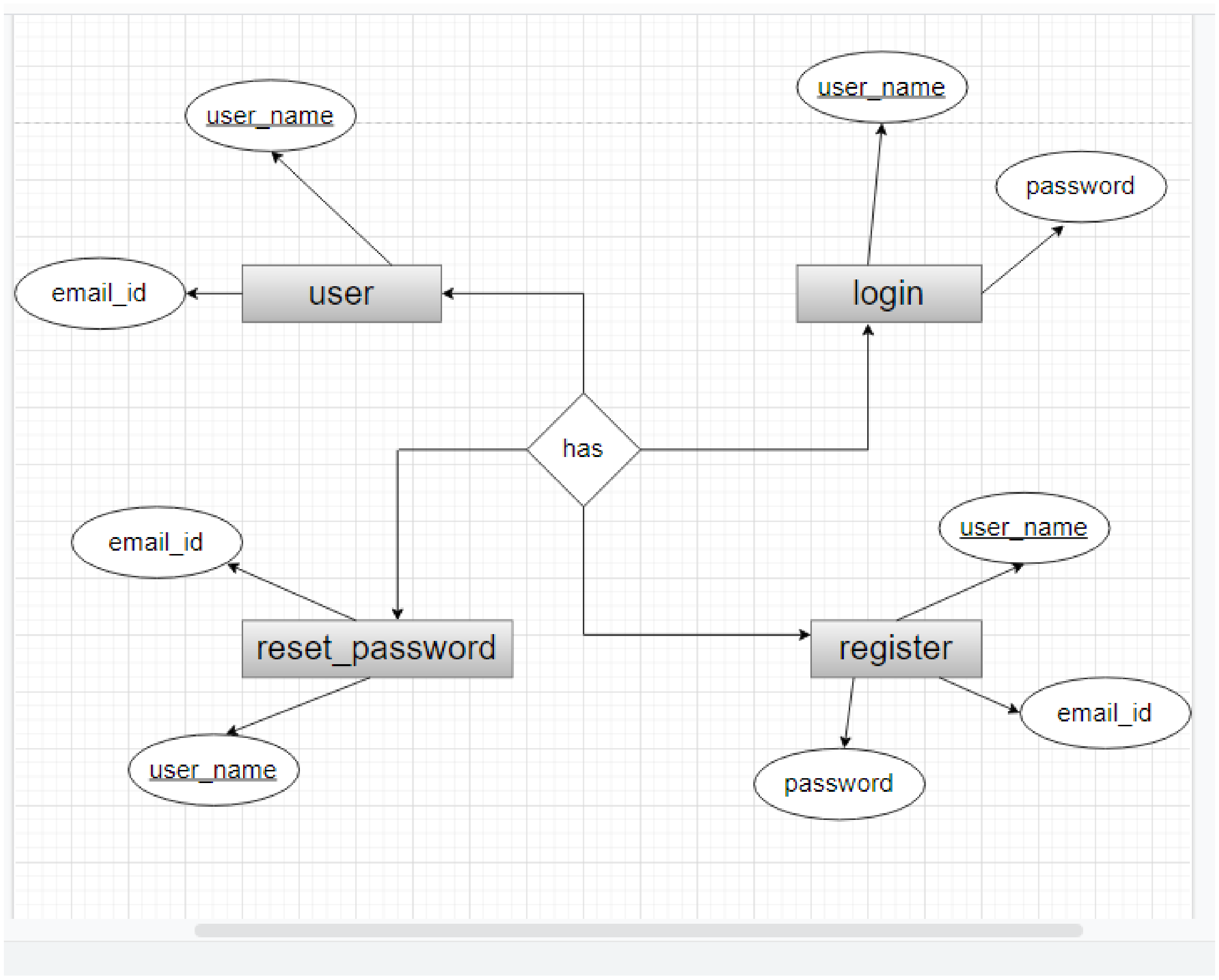
Input Design: This involve: Input to the front end of system is design to be the graphical password. Photos are used instead of typing text password while login process. Control Design: This involve: Before login, it is mandatory is create an account.

Software Quality Attributes:

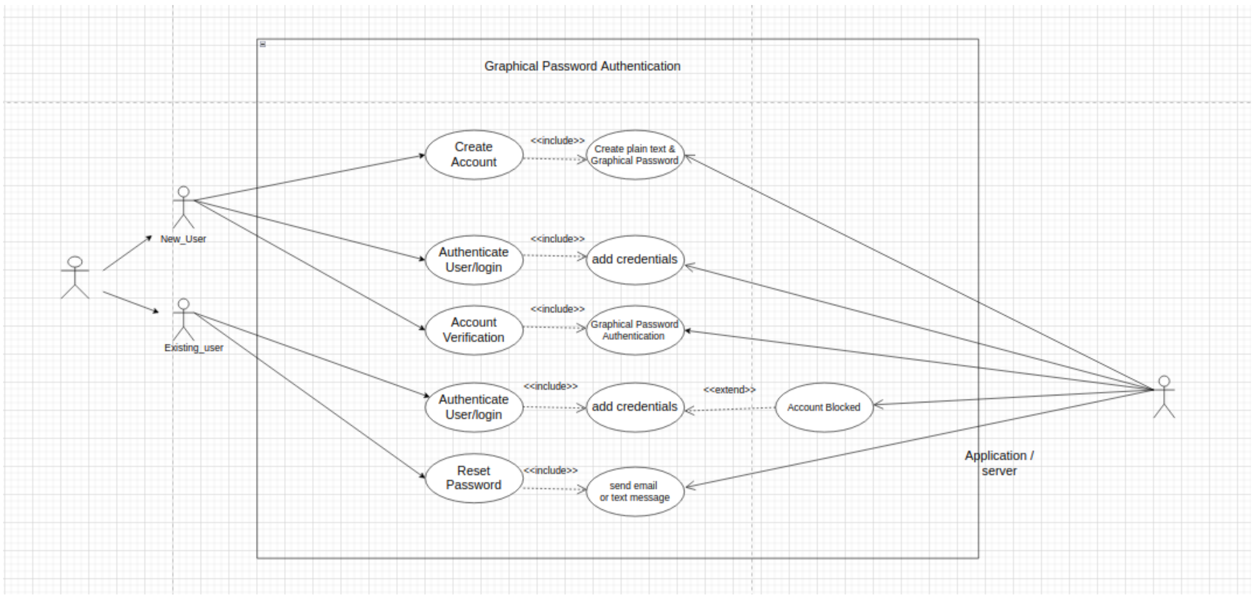
1. Planned approach towards working.
2. Maintainability.
3. Reliability.
4. No Redundancy.
5. Usability.
6. Easy to Operate.

UML:

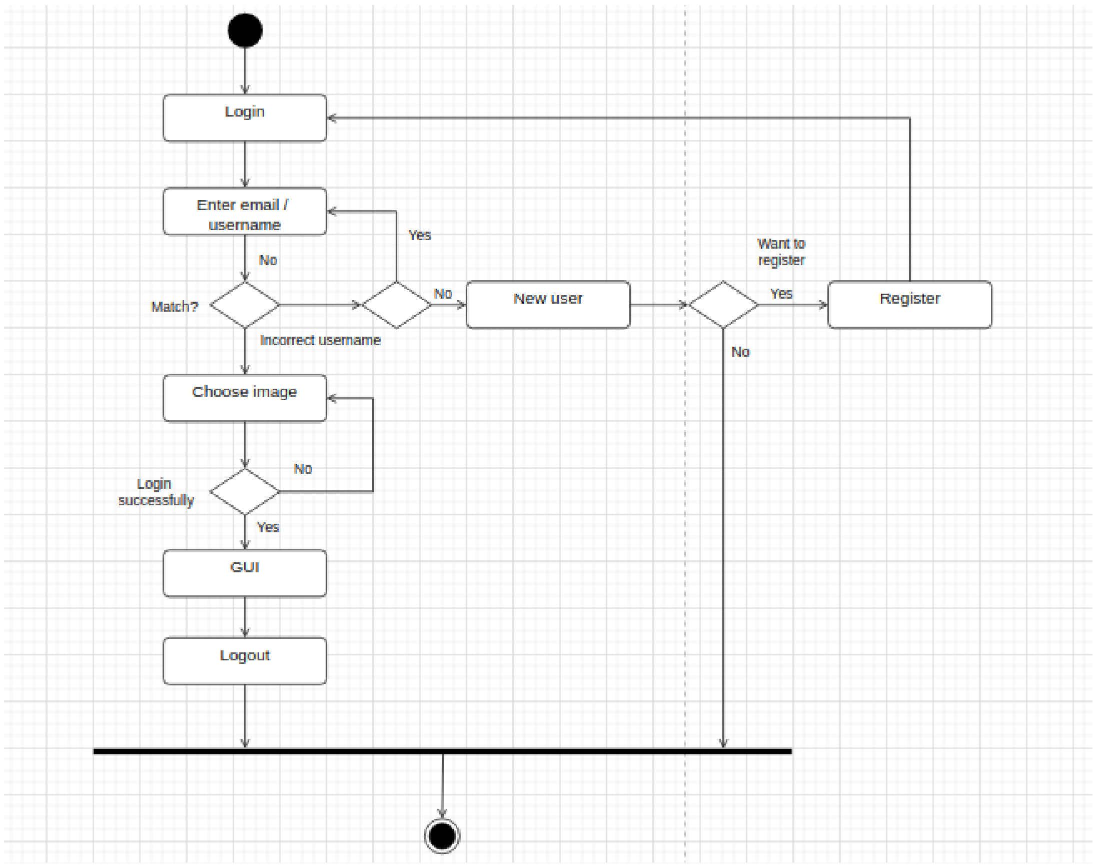
ER Diagram:



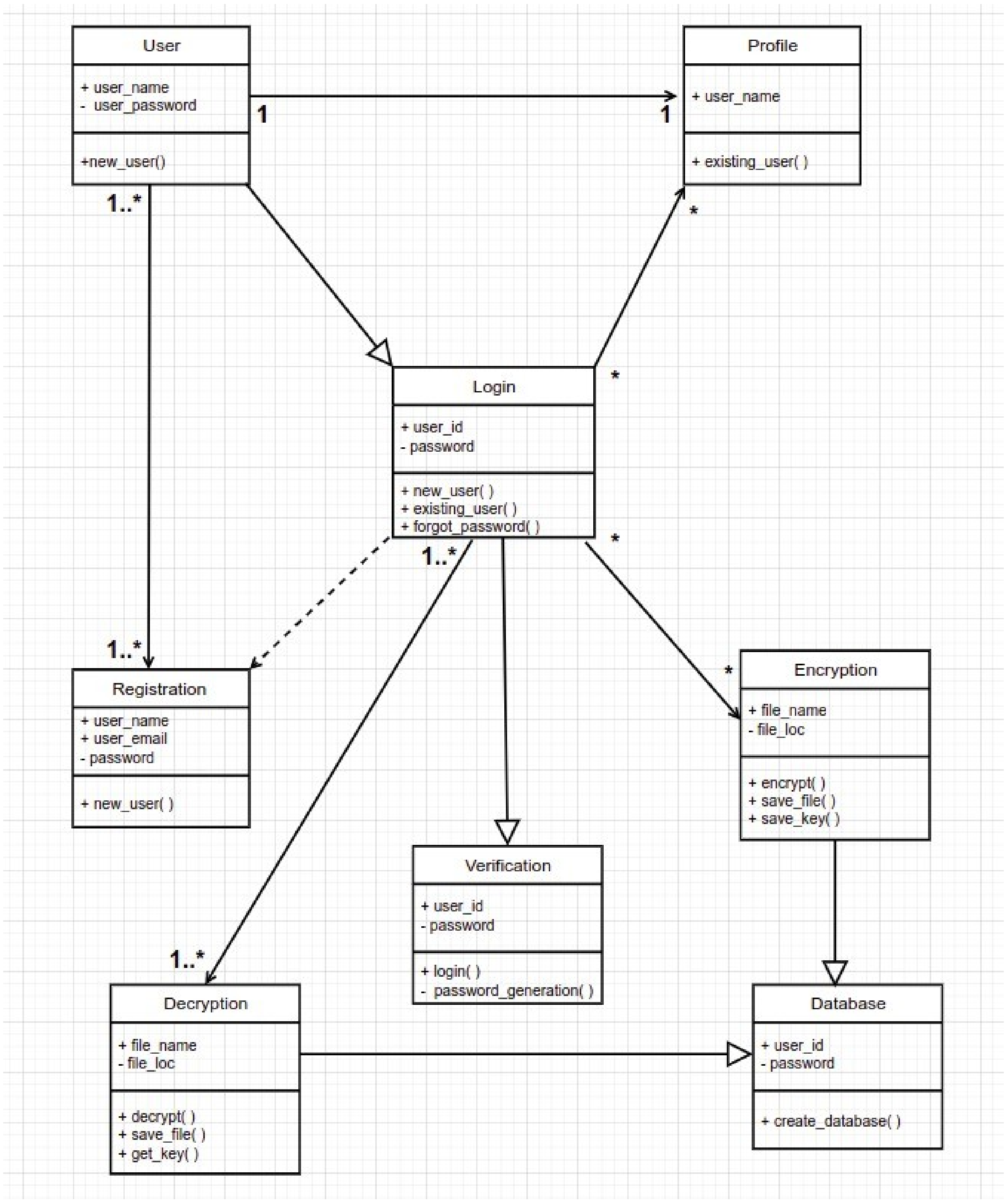
Use-case Diagram:



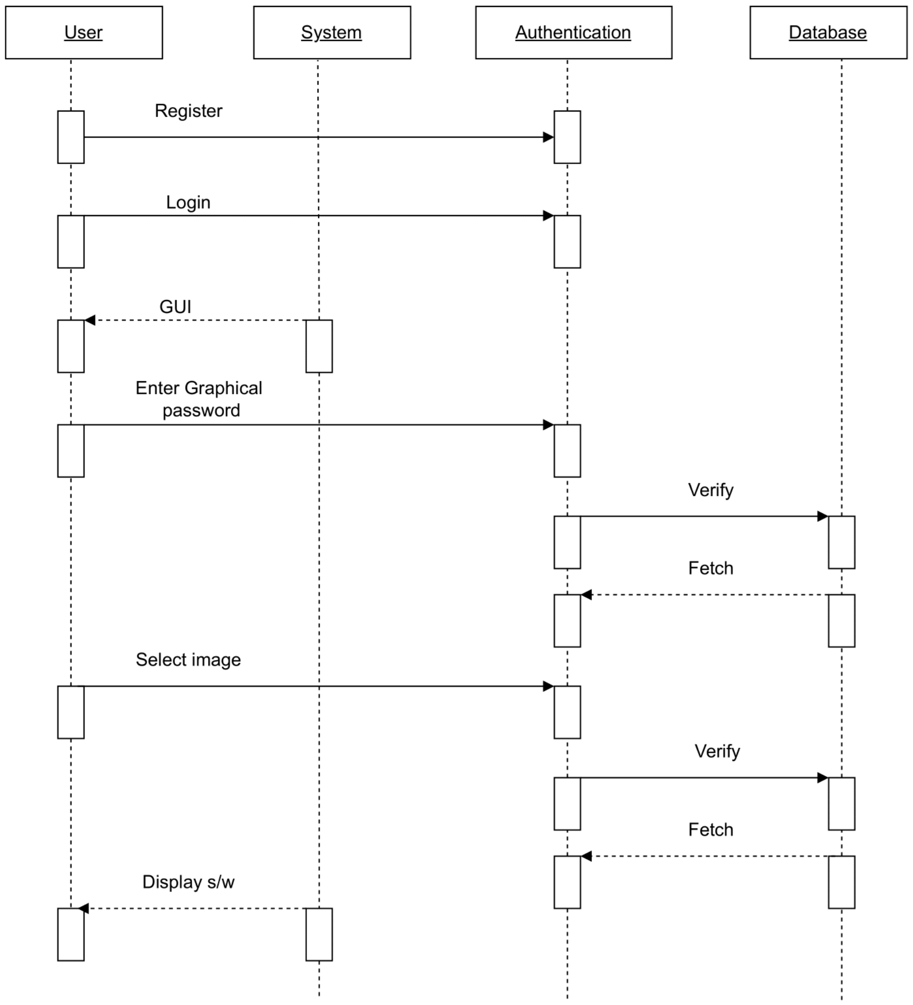
Activity Diagram:



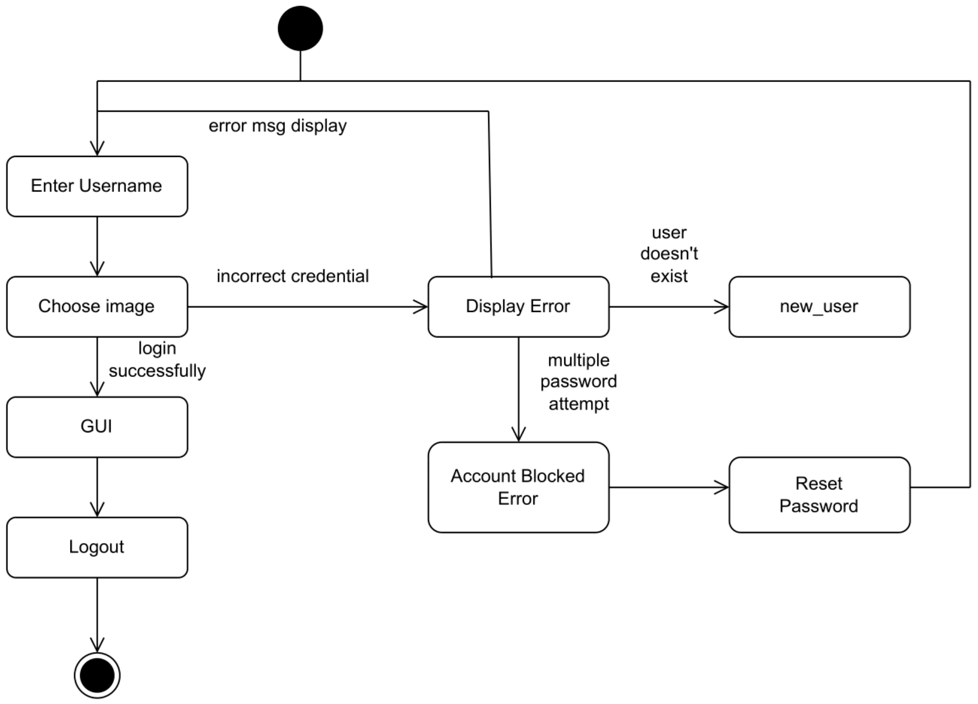
Class Diagram:



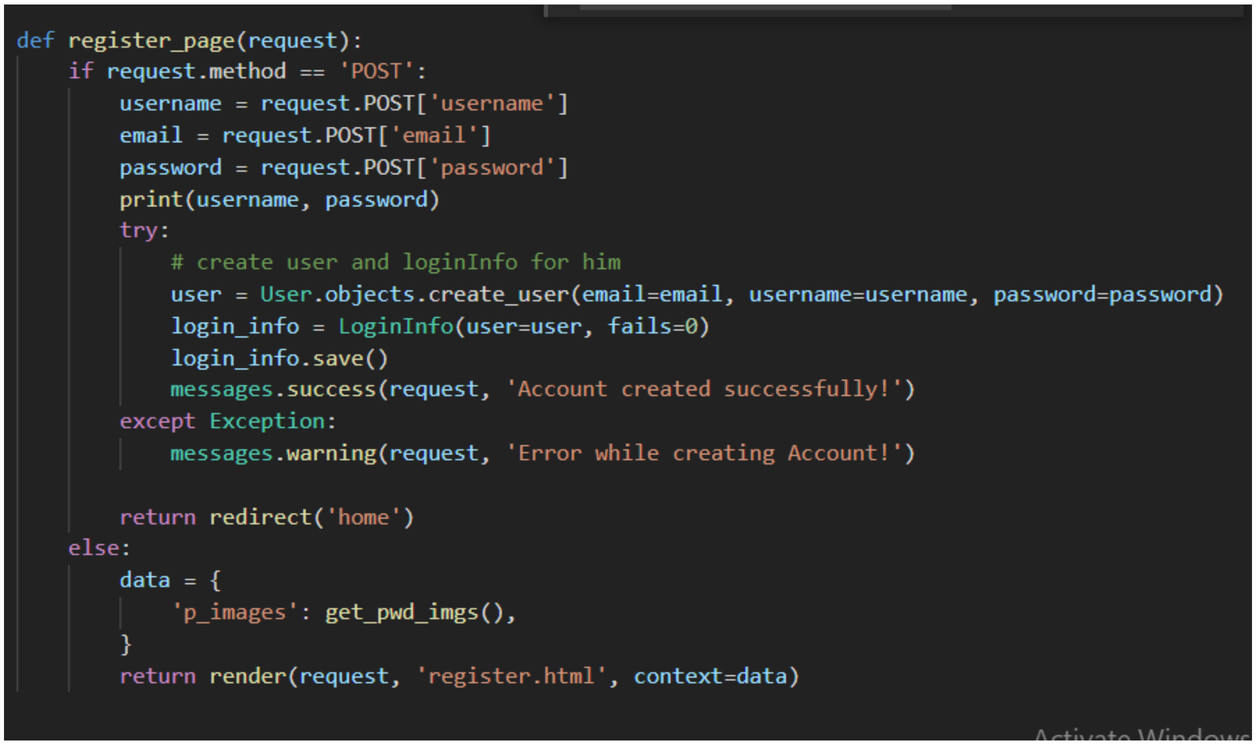
Sequence diagram:



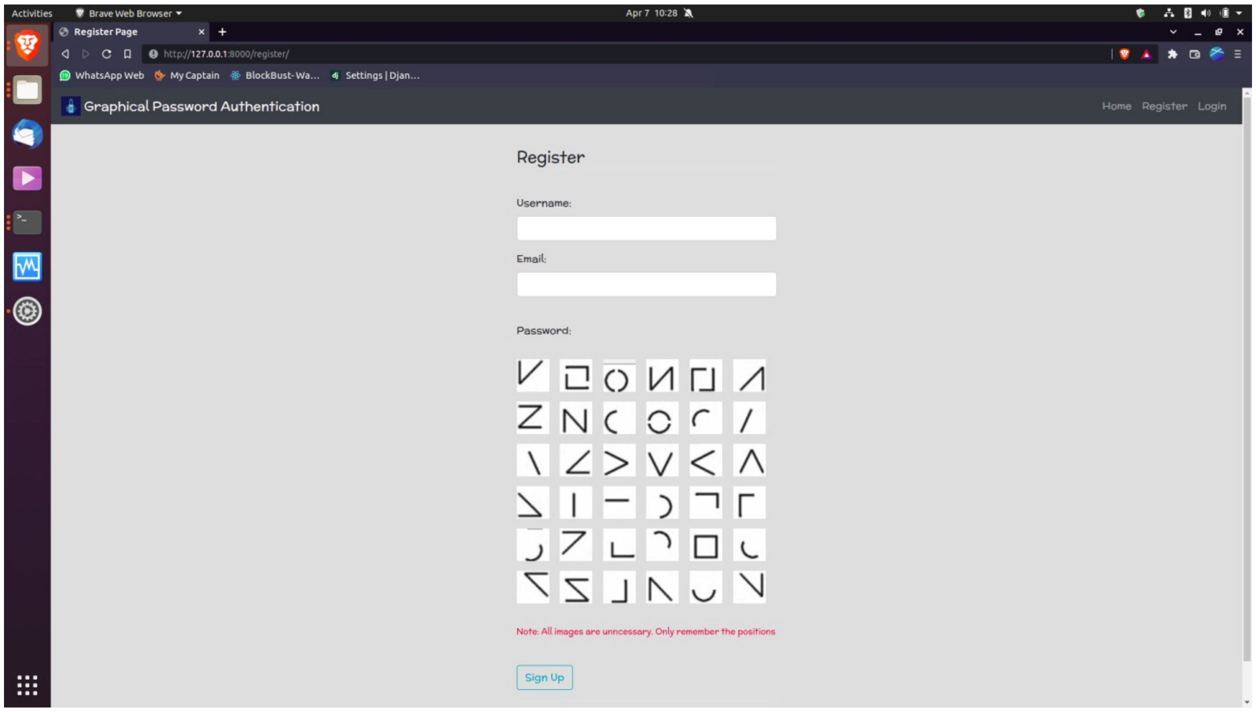
State Diagram:

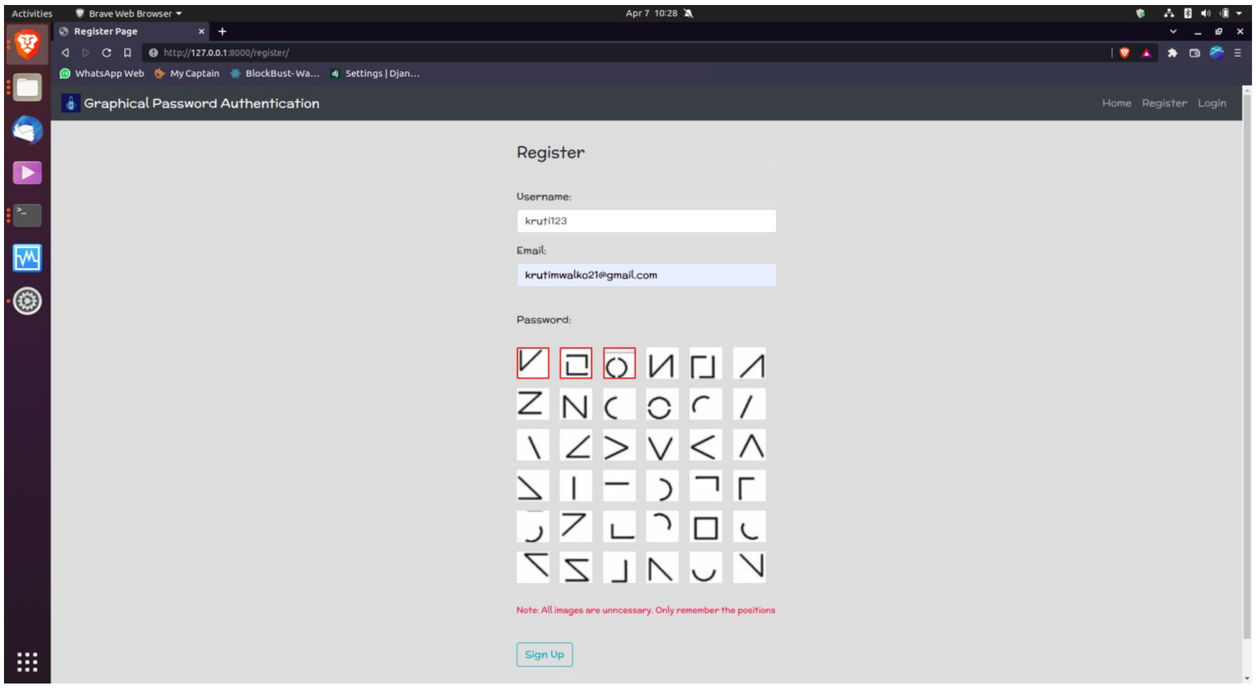

error msg display
Enter Username
incorrect credential
user doesn't exist
Choose image
Display Error
new_user
login successfully
multiple password attempt
GUI
Account Blocked Error
Reset Password
Logout

Coding Screenshots with Results :



```python
def register_page(request):
    if request.method == 'POST':
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']
        print(username, password)
        try:
            # create user and loginInfo for him
            user = User.objects.create_user(email=email, username=username, password=password)
            login_info = LoginInfo(user=user, fails=0)
            login_info.save()
            messages.success(request, 'Account created successfully!')
        except Exception:
            messages.warning(request, 'Error while creating Account!')

        return redirect('home')
    else:
        data = {
            'p_images': get_pwd_imgs(),
        }
        return render(request, 'register.html', context=data)
```





Graphical Password Authentication

Register

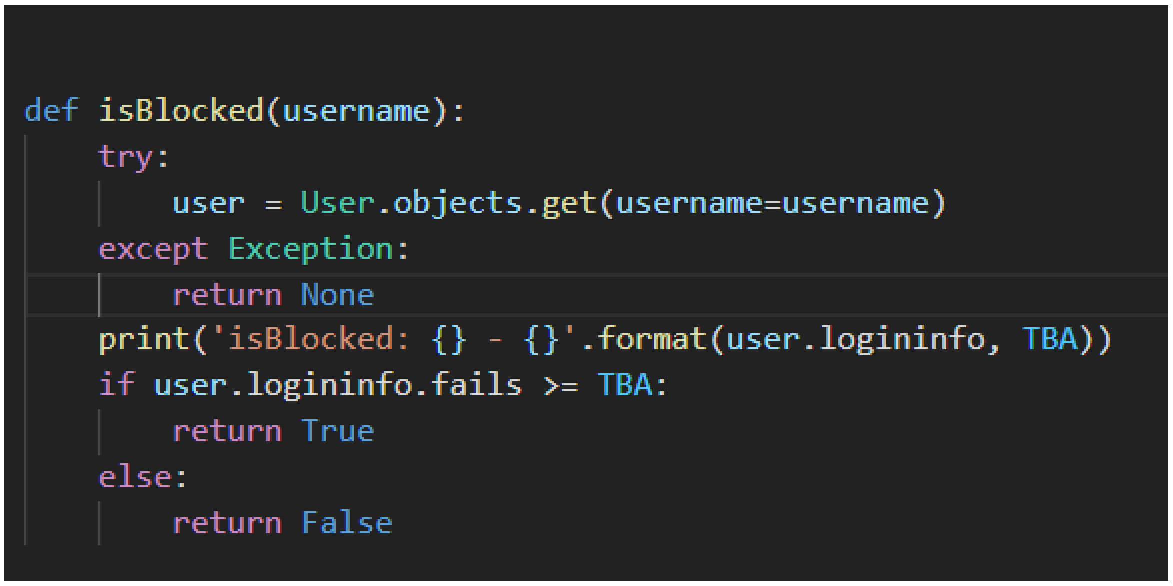
Username:
kruti123

Email:
krutimwalko21@gmail.com

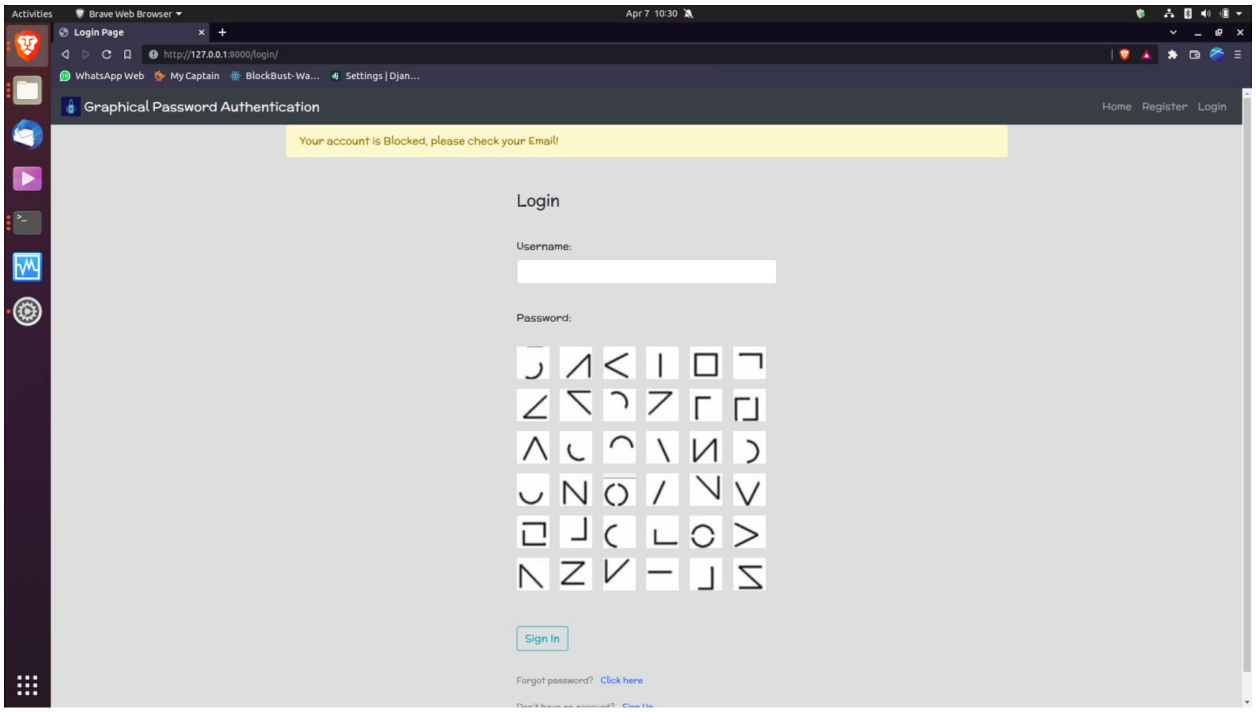
Password:

Note: All images are unncessary. Only remember the positions

Sign Up



```
def isBlocked(username):
    try:
        user = User.objects.get(username=username)
    except Exception:
        return None
    print('isBlocked: {} - {}'.format(user.logininfo, TBA))
    if user.logininfo.fails >= TBA:
        return True
    else:
        return False
```



Graphical Password Authentication

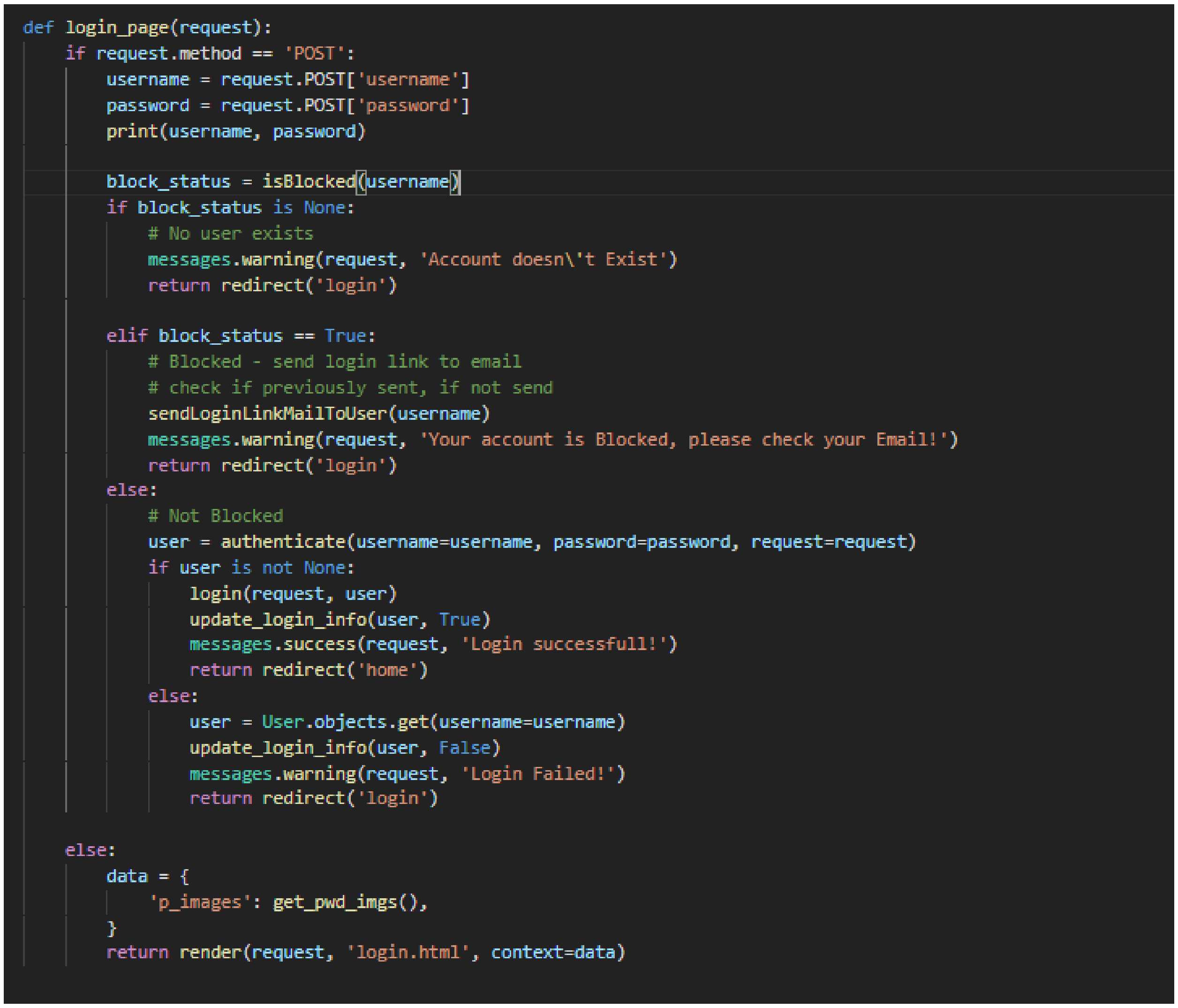
Your account is Blocked, please check your Email!

Login

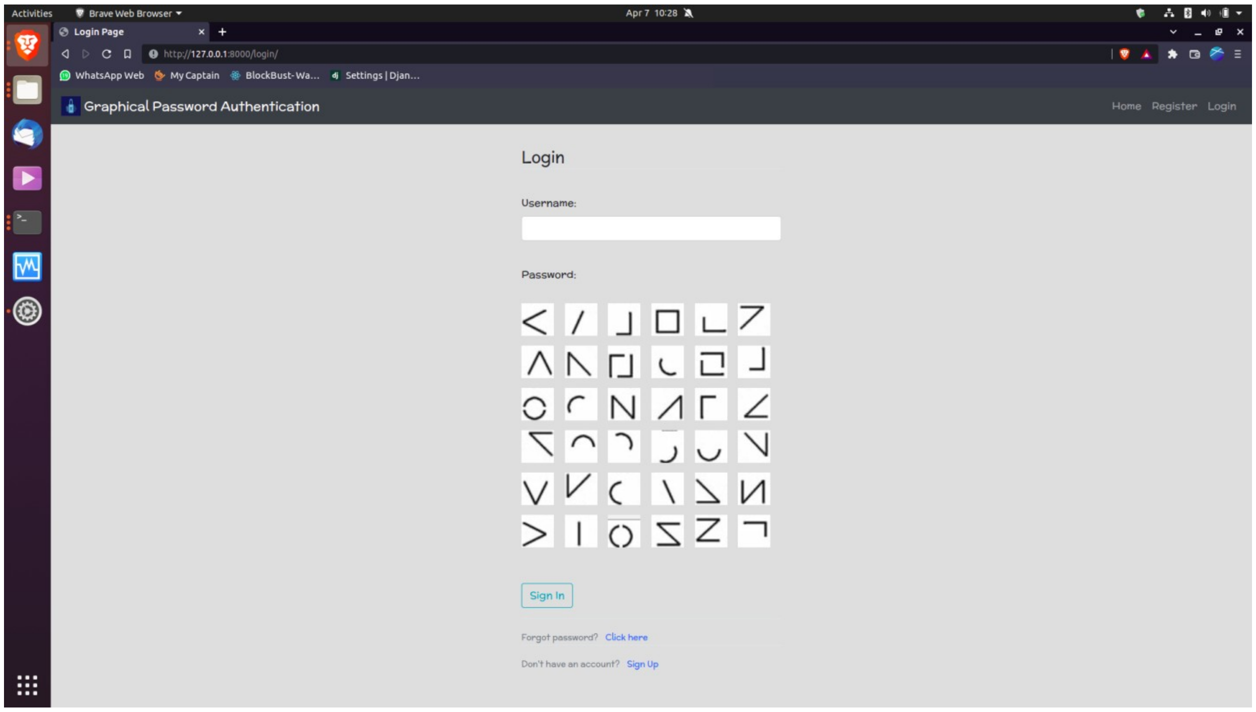
Username:

Password:

Forgot password? Click here



```python
def login_page(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        print(username, password)

        block_status = isBlocked(username)
        if block_status is None:
            # No user exists
            messages.warning(request, 'Account doesn\'t Exist')
            return redirect('login')

        elif block_status == True:
            # Blocked - send login link to email
            # check if previously sent, if not send
            sendLoginLinkMailToUser(username)
            messages.warning(request, 'Your account is Blocked, please check your Email!')
            return redirect('login')
        else:
            # Not Blocked
            user = authenticate(username=username, password=password, request=request)
            if user is not None:
                login(request, user)
                update_login_info(user, True)
                messages.success(request, 'Login successfull!')
                return redirect('home')
            else:
                user = User.objects.get(username=username)
                update_login_info(user, False)
                messages.warning(request, 'Login Failed!')
                return redirect('login')

    else:
        data = {
            'p_images': get_pwd_imgs(),
        }
        return render(request, 'login.html', context=data)
```



Login

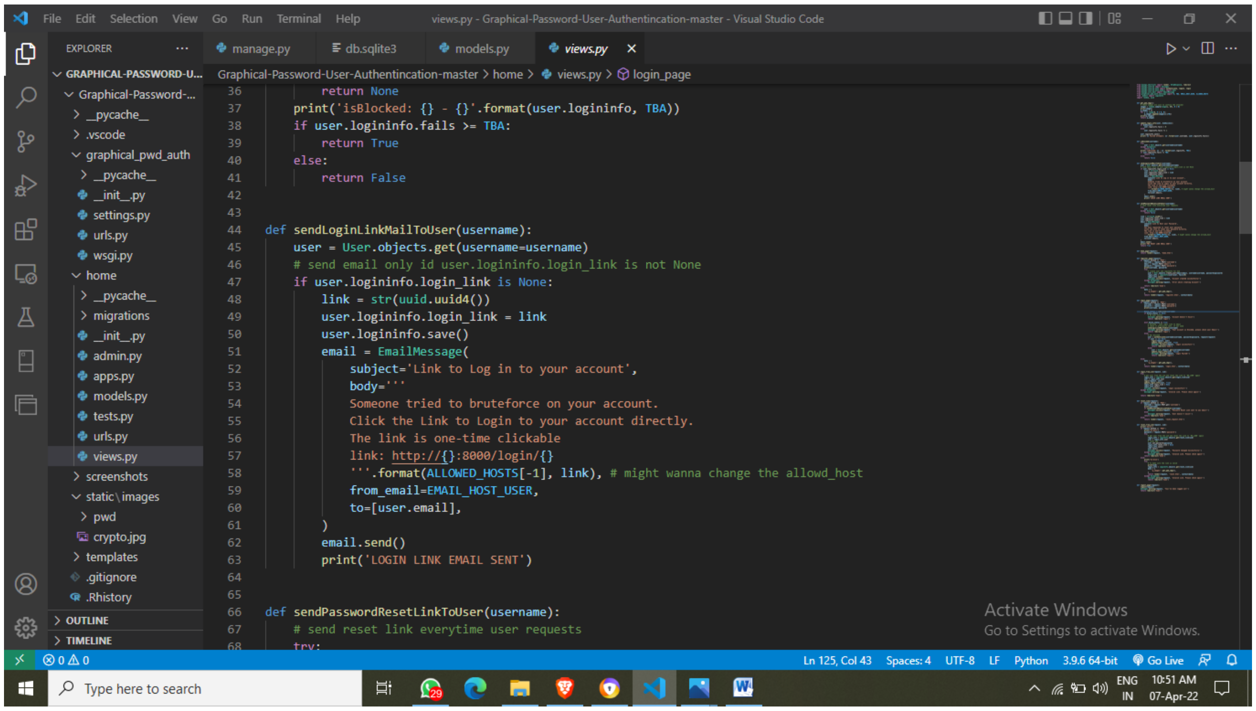
Username:

Password:

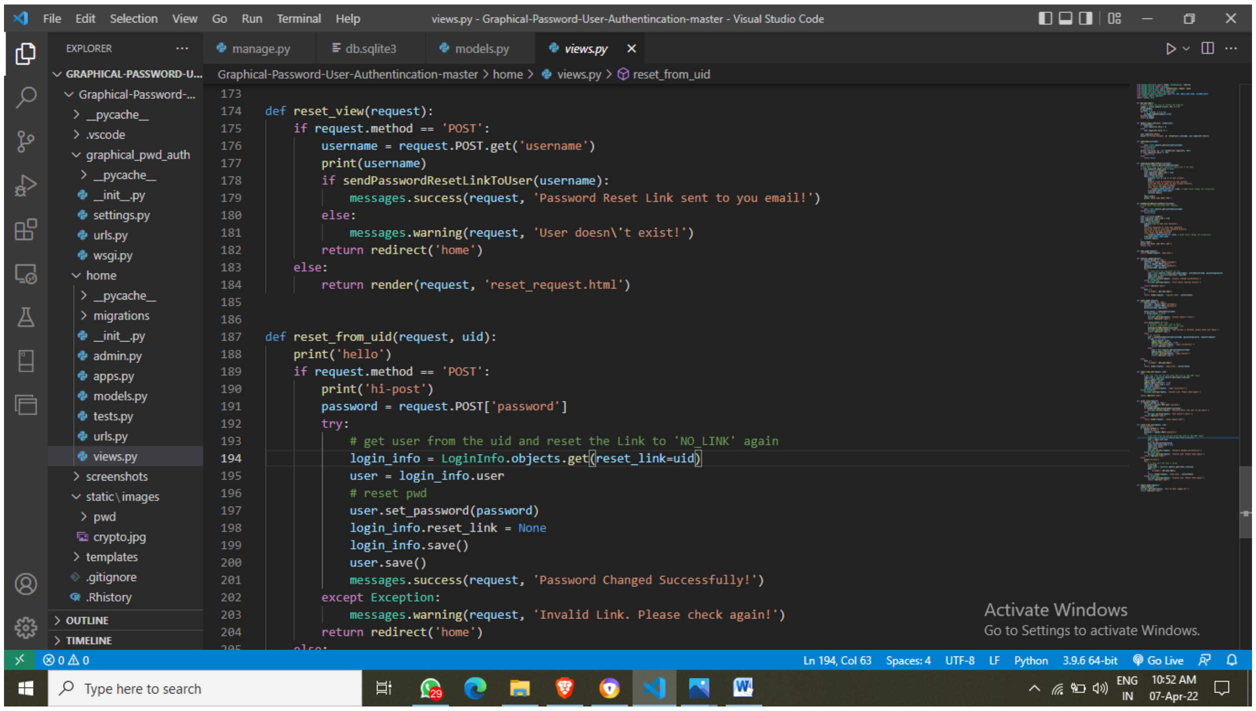
Sign In

Forgot password? Click here

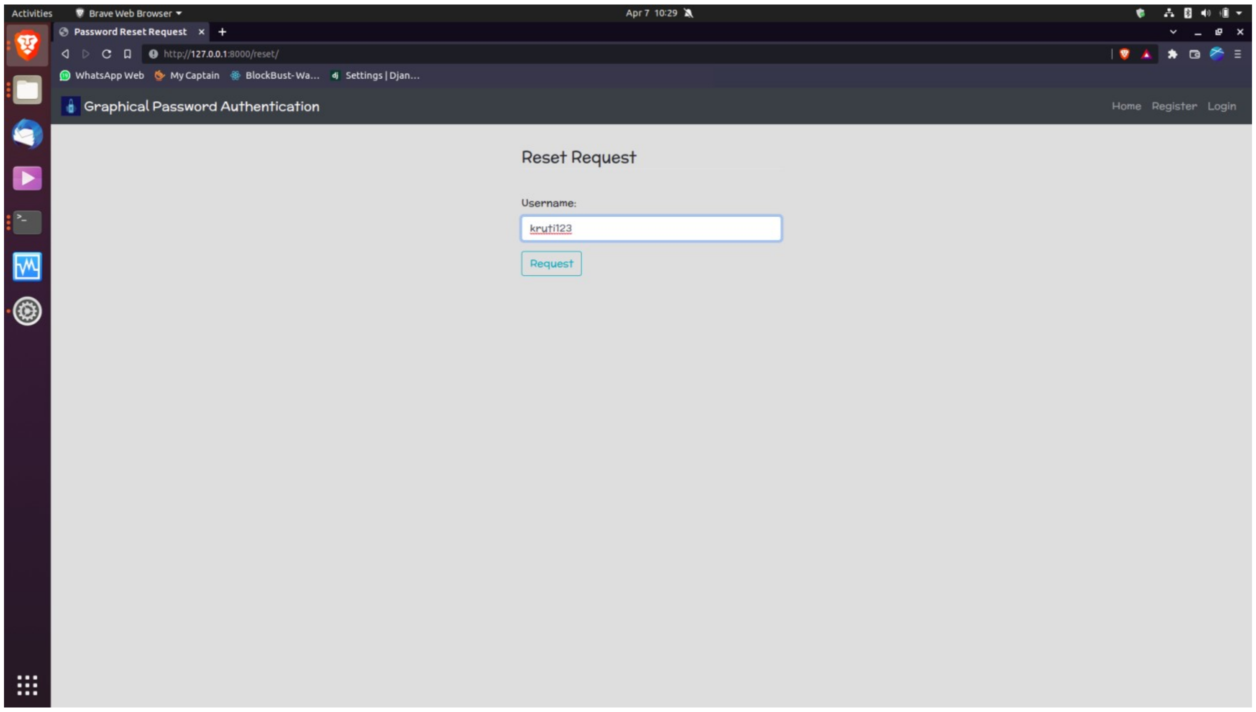
Don't have an account? Sign Up


views.py - Graphical-Password-User-Authentincation-master - Visual Studio Code
manage.py
db.sqlite3
models.py
views.py
Graphical-Password-User-Authentincation-master > home > views.py > login_page

```
            return None
        print('isBlocked: {} - {}'.format(user.logininfo, TBA))
        if user.logininfo.fails >= TBA:
            return True
        else:
            return False


    def sendLoginLinkMailToUser(username):
        user = User.objects.get(username=username)
        # send email only id user.logininfo.login_link is not None
        if user.logininfo.login_link is None:
            link = str(uuid.uuid4())
            user.logininfo.login_link = link
            user.logininfo.save()
            email = EmailMessage(
                subject='Link to Log in to your account',
                body='''
                Someone tried to bruteforce on your account.
                Click the Link to Login to your account directly.
                The link is one-time clickable
                link: http://{}:8000/login/{}
                '''.format(ALLOWED_HOSTS[-1], link), # might wanna change the allowd_host
                from_email=EMAIL_HOST_USER,
                to=[user.email],
            )
            email.send()
            print('LOGIN LINK EMAIL SENT')


    def sendPasswordResetLinkToUser(username):
        # send reset link everytime user requests
        try:
```


views.py - Graphical-Password-User-Authentincation-master - Visual Studio Code
Graphical-Password-User-Authentincation-master > home > views.py > reset_from_uid

```python
def reset_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        print(username)
        if sendPasswordResetLinkToUser(username):
            messages.success(request, 'Password Reset Link sent to you email!')
        else:
            messages.warning(request, 'User doesn\'t exist!')
        return redirect('home')
    else:
        return render(request, 'reset_request.html')


def reset_from_uid(request, uid):
    print('hello')
    if request.method == 'POST':
        print('hi-post')
        password = request.POST['password']
        try:
            # get user from the uid and reset the Link to 'NO_LINK' again
            login_info = LoginInfo.objects.get(reset_link=uid)
            user = login_info.user
            # reset pwd
            user.set_password(password)
            login_info.reset_link = None
            login_info.save()
            user.save()
            messages.success(request, 'Password Changed Successfully!')
        except Exception:
            messages.warning(request, 'Invalid Link. Please check again!')
        return redirect('home')
```

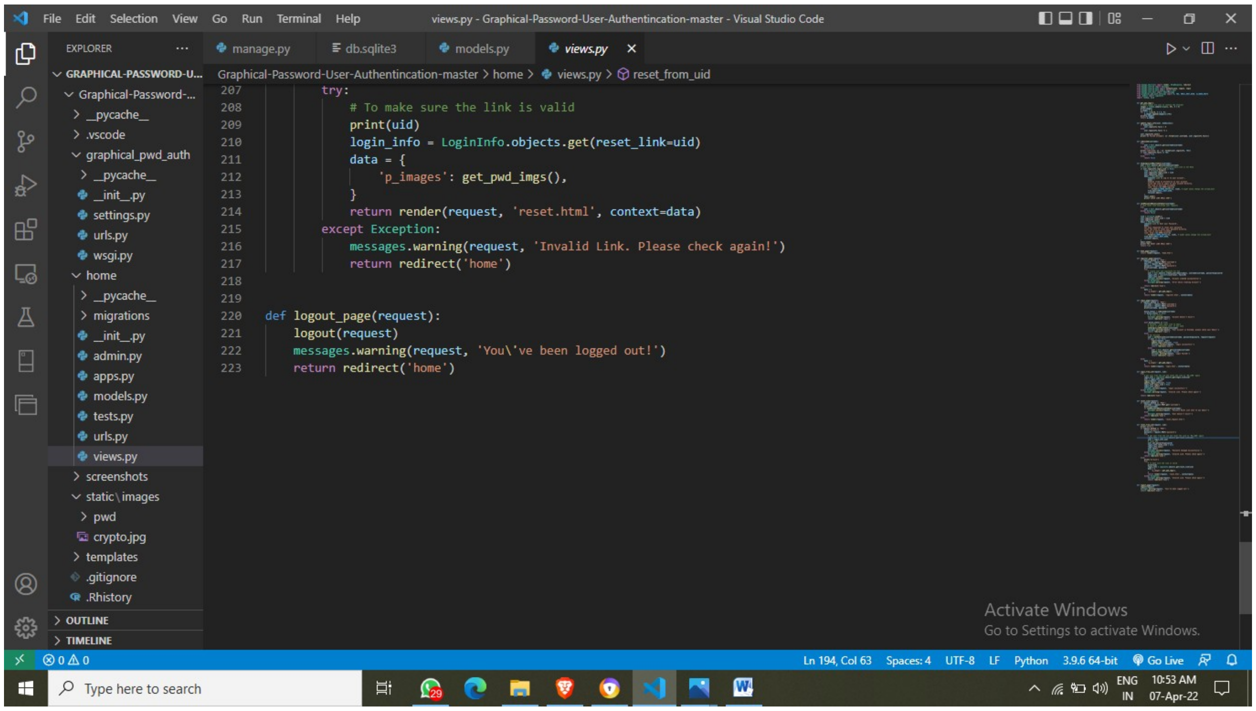

Password Reset Request
http://127.0.0.1:8000/reset/
Graphical Password Authentication
Home
Register
Login

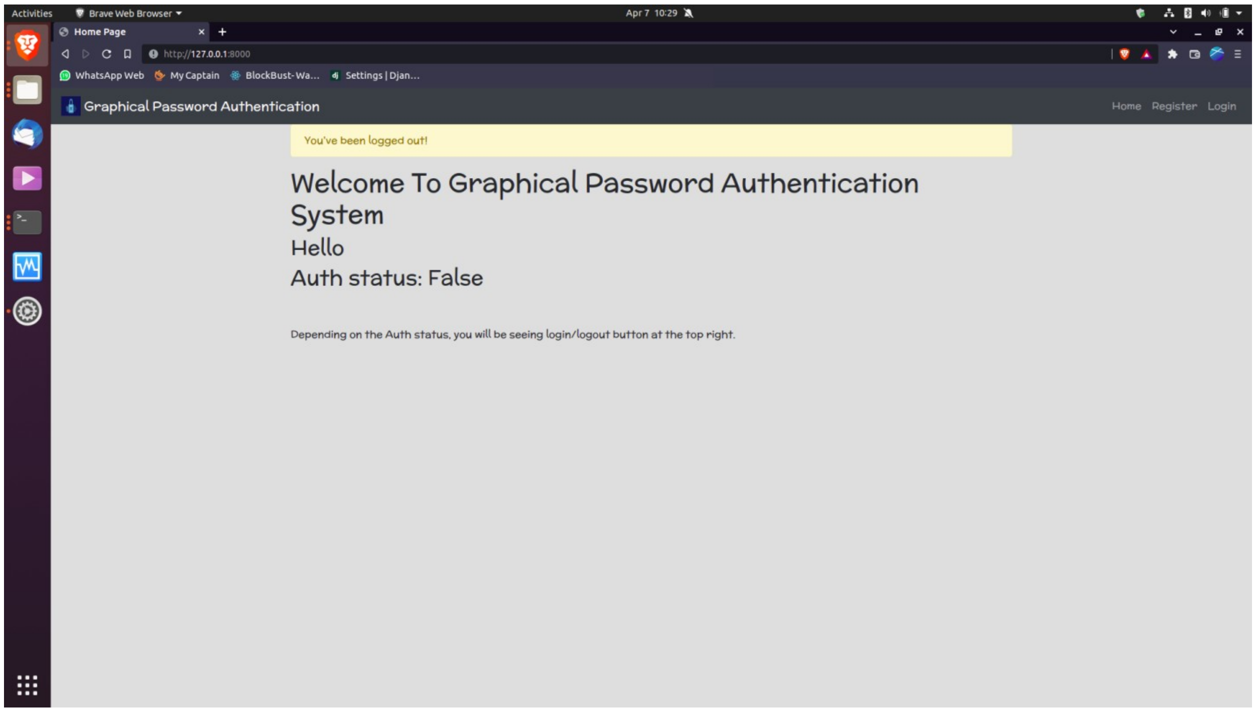
Reset Request

Username:

kruti123

Request


```
        try:
            # To make sure the link is valid
            print(uid)
            login_info = LoginInfo.objects.get(reset_link=uid)
            data = {
                'p_images': get_pwd_imgs(),
            }
            return render(request, 'reset.html', context=data)
        except Exception:
            messages.warning(request, 'Invalid Link. Please check again!')
            return redirect('home')


def logout_page(request):
    logout(request)
    messages.warning(request, 'You\'ve been logged out!')
    return redirect('home')
```


Graphical Password Authentication

You've been logged out!

Welcome To Graphical Password Authentication System

Hello

Auth status: False

Depending on the Auth status, you will be seeing login/logout button at the top right.

Testing:

Bugs:

Reporter: kruti and krutimwalko21@gmail.com

Product: We found bug in sending link to user’s mail id.

Component:

Platform : On every platform

Operating system: Windows, Linux

Priority : Minor loss of function.

Severity: We are unable to redirect the link to user’s email id because as we know out gmail account is secured by two step authentication and the link we are sending(SMTP) is not secure. It’s blocking the mail id which is sending it.

Status: Not fixed yet

Future Scope:

It can be used everywhere instead of text-based password .We can increase the security of this system. Presently there are many authentication system but they have their own advantages and disadvantages. Text password can be hacked easily with various methods where as biometric authentication can cause more cost. This system is more secure and cheap than old methodologies. As well as this system allows more reliable and easily recognizable system to the users. As how we have written over this system can be best alternative to the text password.

Conclusion:

The system promise as a usable and memorable authentication mechanism. By taking advantage of users’ ability to recognize images and the memory trigger associated with seeing a new image, it has advantages in terms of usability. In future development we can also add challenge response interaction. In challenge response interactions, server will present a challenge to the client and the client need to give response according to the condition given. If the response is correct then access is granted.

Also we can limit the number a user can enter the wrong password.

Github Projct URL:

https://github.com/
krutiwalko21Graphical_Password_Authentication.git