NETWORK SECURITY

CWE-262 & CWE 263: Underutilization of Password Aging and Expiration

(A07:2021 - Identification and Authentication Failures)

Project by : Shreeraksha R Aithal, 181CO149

Introduction

User Authentication is one of the widely required and used systems for anything that we do on a network. With growing demand and users over networks like the Internet, the threats against it have also risen. Attackers all over the world are on the tip of their toes to break into a system by impersonating someone else, despite the geographic location of the victim. So a more secure Authentication system is all that we need. But is the blame always on the service provider to provide safety and privacy to its users? Not always. Users must also realize the severity of the threat and the protection provided by the authentication system. Along with the end-users cooperation and stricter measures by the service provider, the vulnerability posed by the authentication system can be brought down to a much larger extent, even if not completely eradicated.

Problem Description

Authentication systems these days don't enforce stricter actions in favor of user convenience. Most authentication systems in the world won't have any concept of password expiry implemented while others just send reminders to change passwords. In this situation, it depends on the user to change the password as and when needed.

OWASP mentions "**Identification and Authentication Failures**" as the 7th most Web Application Security Risk of 2020. Under this, CWE-262 and CWE-263 revolve around password aging and expiry duration.

- ➔ CWE-262 - *Not Using Password Aging* :
    - ◆ If there are strict means of imposing password aging, users will have no incentive to update passwords in a timely manner.
    - ◆ Frequent password changes act as an overhead effort for the user to access the facility provided by the server. Even though password aging is encouraged for security purposes, frequent password changes might force users to reset to weaker or similar passwords. This again can be taken advantage of by the attacker to launch an attack. It is also useful to have a password aging mechanism that notifies users when passwords are considered old and requests that they replace them with new, strong passwords.
- ➔ CWE-263 - *Password Aging with Long Expiration*:
    - ◆ Unchecked password aging can pose as a point of vulnerability for attackers to exploit and break the authentication system. There is also a possibility of diminished password integrity.
    - ◆ Passwords must be given a maximum life span, after which a user is required to update with a new and different password.

Most authentication systems these days do not implement password aging. Tech giants (like Google, Microsoft, and Facebook) along the banking sector are a few of those fields that consider password expiry. Still stricter password expiry is far from implementation and only send reminders to change password.

Defining the duration of password expiry is a difficult task. A shorter duration forces the user to reset passwords frequently, where the strength of newer passwords tends to diminish; whereas a longer duration gives enough time for the attacker to guess the password and launch an attack.

Solution Design

Most Authentication systems out there either don’t implement Password Aging or a relaxed version of the same. Stricter mechanisms are needed to force the user to set a newer and stronger password when the previous password has expired. This can also be accompanied by sending reminders to users to reset their password when the expiration is nearing.

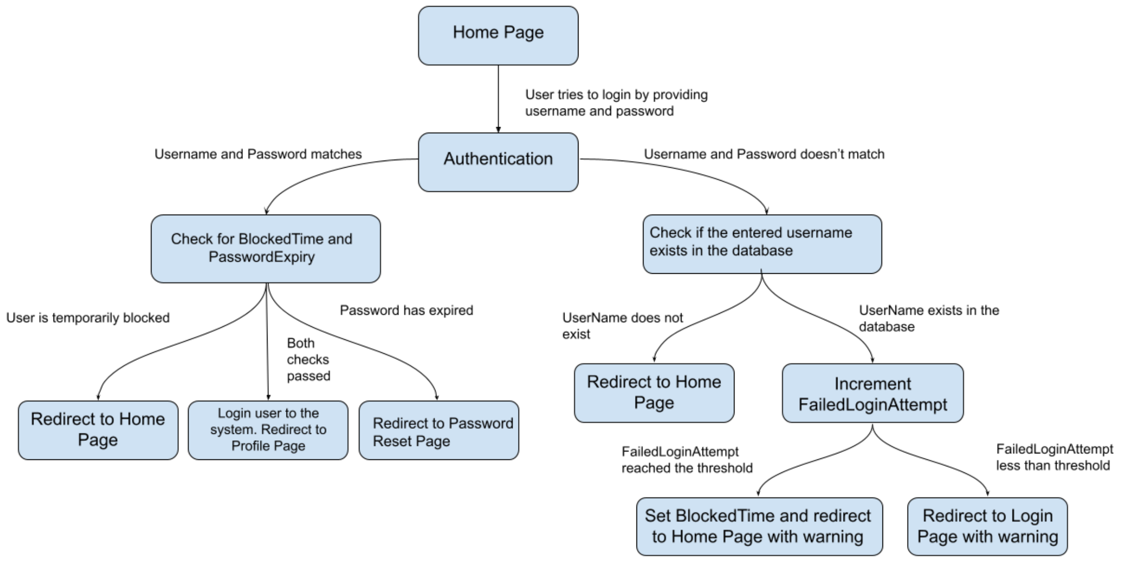
An unsuccessful attempt to break the authentication system by an attacker usually begins with multiple failed login attempts. If the authentication system doesn't provide any mechanism to regulate the number of failed attempts, it acts as a vulnerability. So blocking a user for a short period after a certain number of unsuccessful attempts can stall the attack to some extent.

Expiration duration is another matter of concern. A longer duration might lead to a threat of attack and a shorter duration might result in user inconvenience to use the service. So, a proper balance between the two is needed. If the service is one that requires higher security or whose services are used regularly (like an e-mail system) then a shorter duration works well; for services that people use only once in a while or something that doesn't require very high-level security, can have longer expiry span.

The design is divided into two phases as follows:

Phase 1: Implementation of Password Aging with Expiry

This phase includes the use of 'PasswordExpiry' time to keep track of the aging factor of the Password. To avoid multiple failed login attempts by the attack to take unduly advantage of the authentication system to launch a spoof attack, we also keep track of the number of failed login attempts in a certain window span and block the user temporary for a certain period when the number reaches/exceeds a certain threshold (threshold = 5 in our case).


Home Page
User tries to login by providing username and password
Authentication
Username and Password matches
Username and Password doesn't match
Check for BlockedTime and PasswordExpiry
Check if the entered username exists in the database
User is temporarily blocked
Both checks passed
Password has expired
UserName does not exist
UserName exists in the database
Redirect to Home Page
Login user to the system. Redirect to Profile Page
Redirect to Password Reset Page
Redirect to Home Page
Increment FailedLoginAttempt
FailedLoginAttempt reached the threshold
FailedLoginAttempt less than threshold
Set BlockedTime and redirect to Home Page with warning
Redirect to Login Page with warning

Phase 2: Additional authentication using Security Pin

Phase 2 is developed above Phase 1 and can be thought of as its extension. Following are some design ideas for the same:

- ➢ Send reminder mails to the users when their password is about to expire (like 1-2 days prior).
- ➢ Captcha can be used to counter multiple random login attempts by attackers.
- ➢ Implement a strong Password Strength checker to avoid weak passwords, which otherwise can easily be guessed by the attacker.
- ➢ Keep track of the 3 most recent passwords of a user, and restrict the user from using the same while setting a new password.

The other breakthrough feature in the design is ‘Secret Key’

- ❖ Secret key acts as an additional password for the user, used only for the purpose of administration aspect (of the user).
- ❖ Secret key is a 4-digit or 6-digit number entered by the user during the time of Sign-Up (account creation).
- ❖ Secret key can be exclusively used as first factor authentication when resetting password and other purposes. Other authentication factors like secret link or Captcha can be used over this.
- ❖ Secret key can be resetting only after logging in. So ‘Forgot Secret Key’ feature’ doesn’t exist.
- ❖ Incase the user has forgotten the Secret Key, they have to contact the administration to reset the same.
- ❖ ‘Secret Key’ may or may not have expiration time, unlike Password.

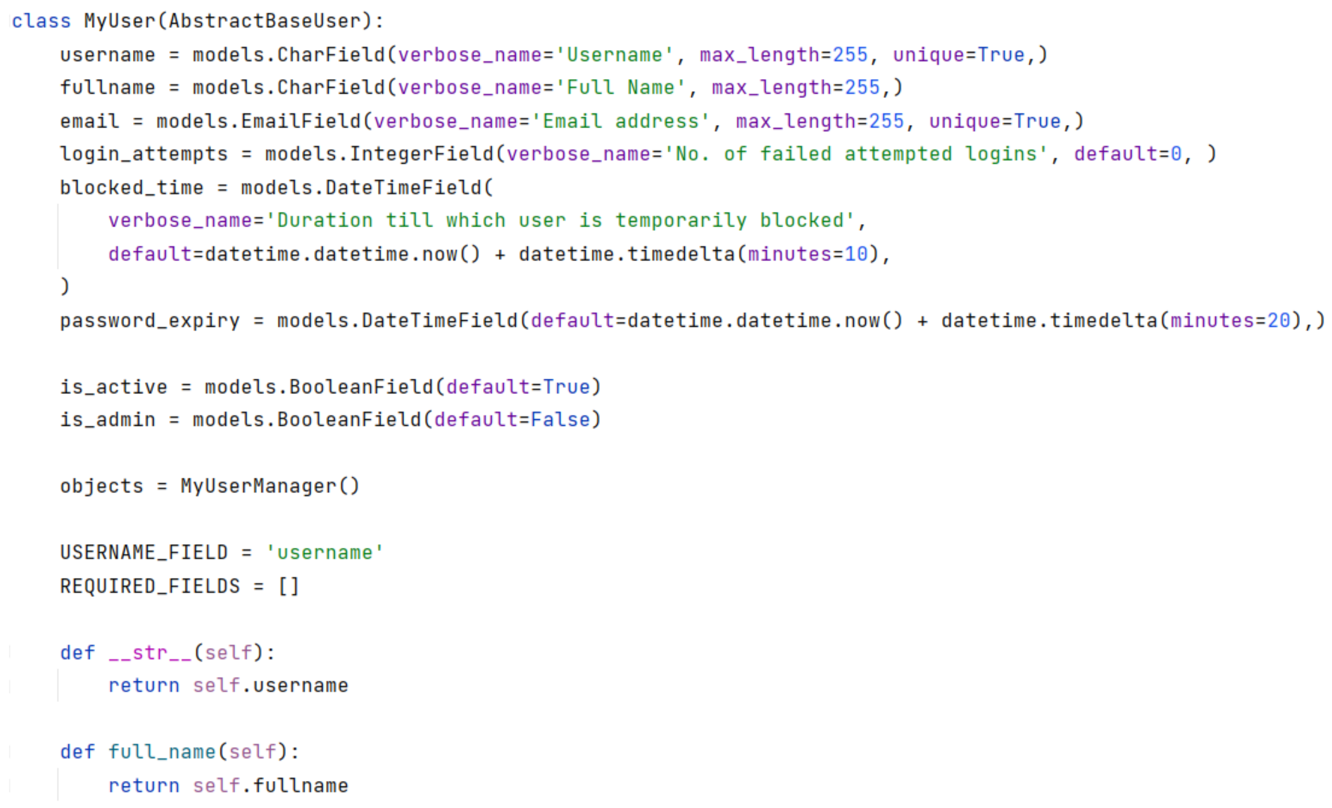
Implementation

The solution mentioned in the above section is implemented using the Django application. The implemented authentication system, named “SecureLogin”, is developed above the regular authentication of Django.

Phase 1

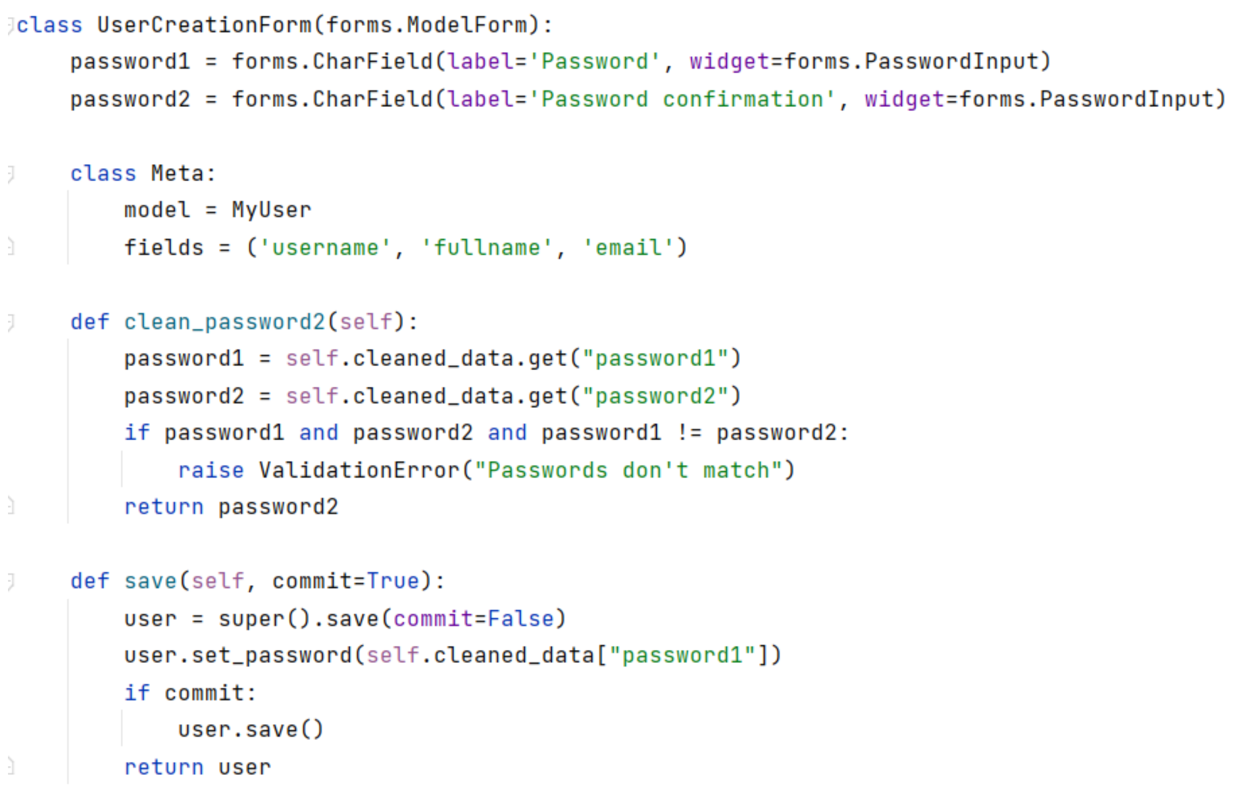
The functionalities implemented in Phase 1 include, and are not limited to the following.

★ Extending the existing User class to include additional parameters associated with them.



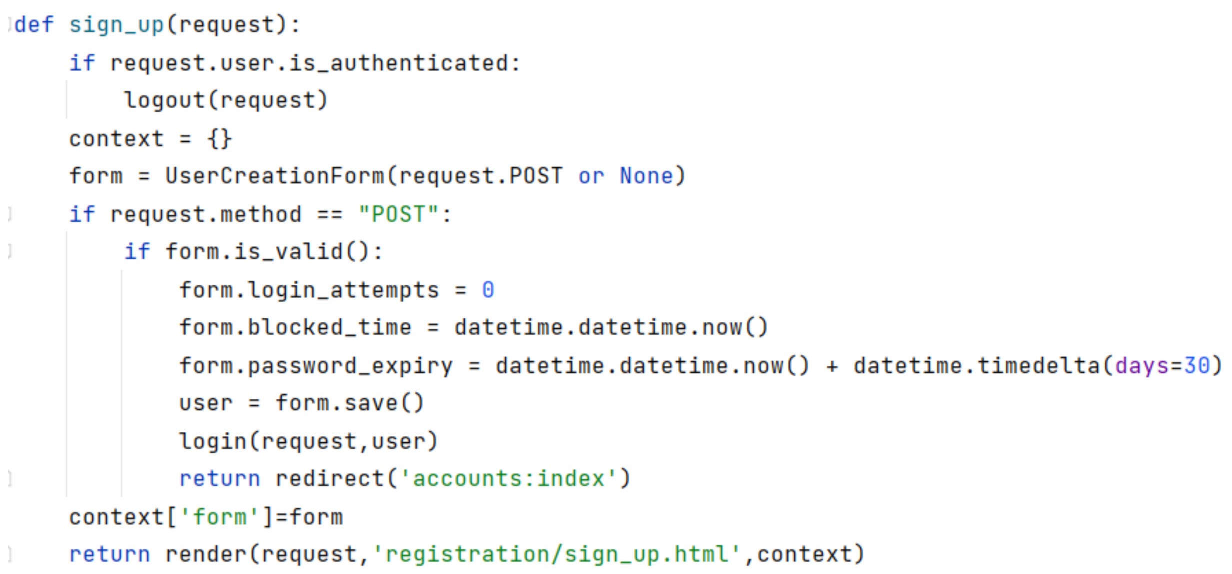
```python
class MyUser(AbstractBaseUser):
    username = models.CharField(verbose_name='Username', max_length=255, unique=True,)
    fullname = models.CharField(verbose_name='Full Name', max_length=255,)
    email = models.EmailField(verbose_name='Email address', max_length=255, unique=True,)
    login_attempts = models.IntegerField(verbose_name='No. of failed attempted logins', default=0, )
    blocked_time = models.DateTimeField(
        verbose_name='Duration till which user is temporarily blocked',
        default=datetime.datetime.now() + datetime.timedelta(minutes=10),
    )
    password_expiry = models.DateTimeField(default=datetime.datetime.now() + datetime.timedelta(minutes=20),)

    is_active = models.BooleanField(default=True)
    is_admin = models.BooleanField(default=False)

    objects = MyUserManager()

    USERNAME_FIELD = 'username'
    REQUIRED_FIELDS = []

    def __str__(self):
        return self.username

    def full_name(self):
        return self.fullname
```

★ Customized User object Creation module.



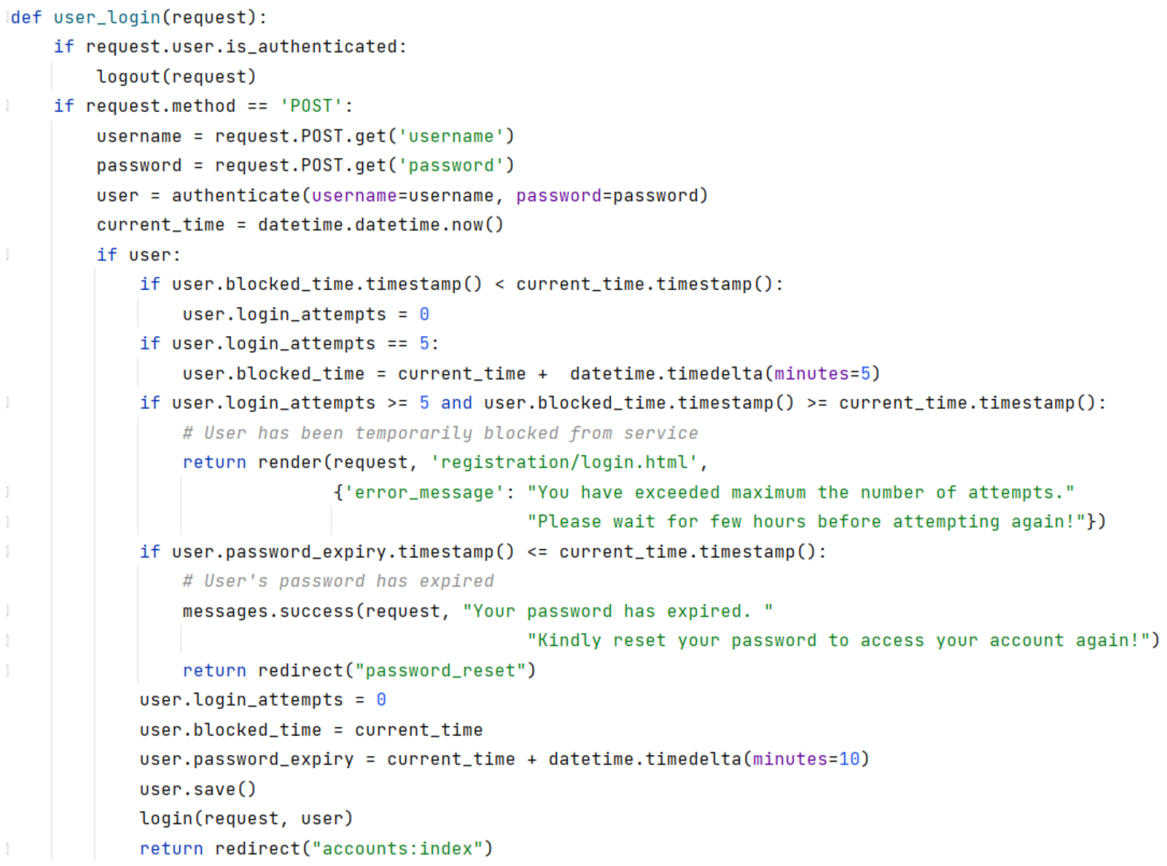
```python
class UserCreationForm(forms.ModelForm):
    password1 = forms.CharField(label='Password', widget=forms.PasswordInput)
    password2 = forms.CharField(label='Password confirmation', widget=forms.PasswordInput)

    class Meta:
        model = MyUser
        fields = ('username', 'fullname', 'email')

    def clean_password2(self):
        password1 = self.cleaned_data.get("password1")
        password2 = self.cleaned_data.get("password2")
        if password1 and password2 and password1 != password2:
            raise ValidationError("Passwords don't match")
        return password2

    def save(self, commit=True):
        user = super().save(commit=False)
        user.set_password(self.cleaned_data["password1"])
        if commit:
            user.save()
        return user
```

- ★ Verification during User Sign-Up

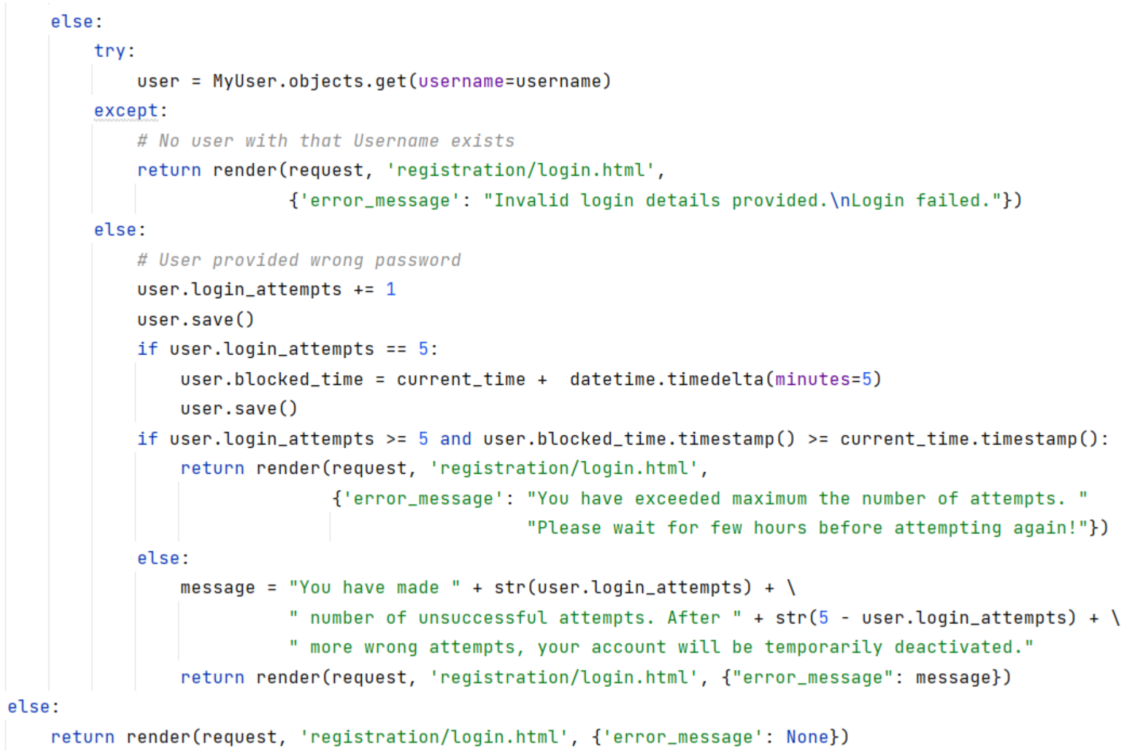


```python
def sign_up(request):
    if request.user.is_authenticated:
        logout(request)
    context = {}
    form = UserCreationForm(request.POST or None)
    if request.method == "POST":
        if form.is_valid():
            form.login_attempts = 0
            form.blocked_time = datetime.datetime.now()
            form.password_expiry = datetime.datetime.now() + datetime.timedelta(days=30)
            user = form.save()
            login(request,user)
            return redirect('accounts:index')
    context['form']=form
    return render(request,'registration/sign_up.html',context)
```

- ★ Verification during user Login

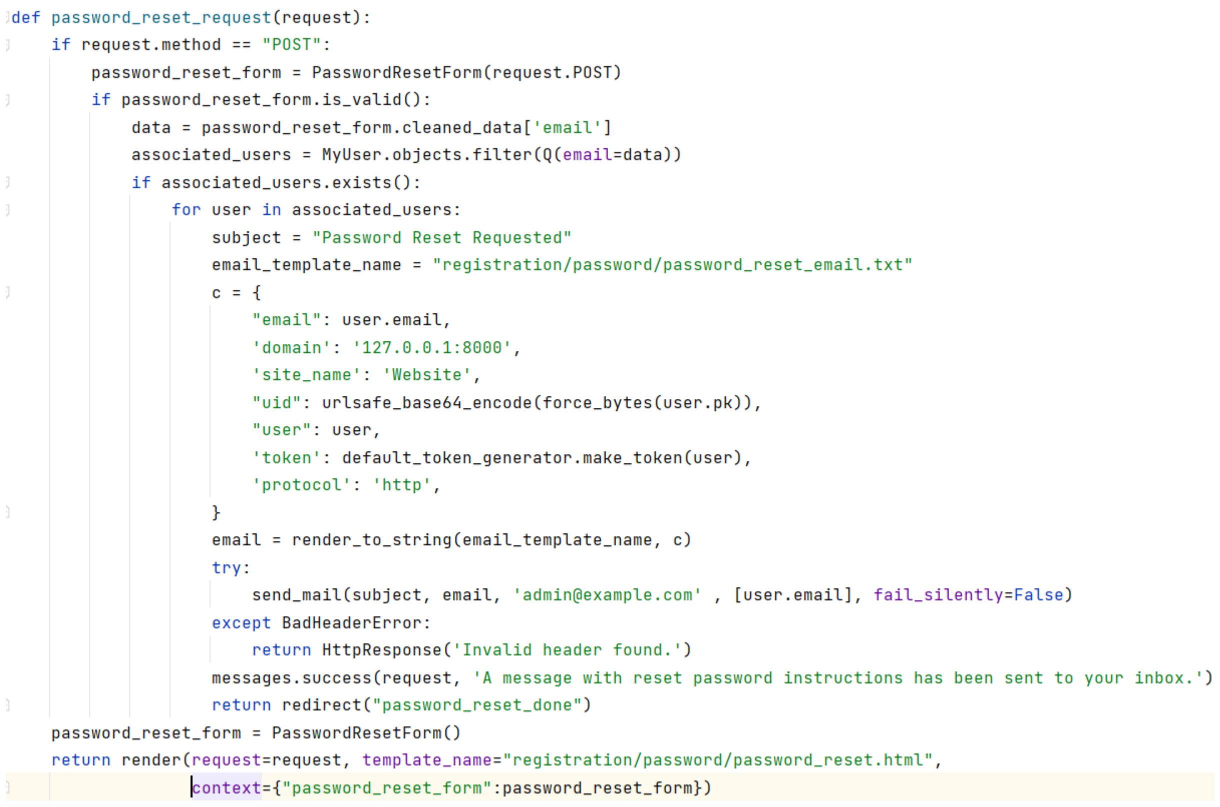


```python
def user_login(request):
    if request.user.is_authenticated:
        logout(request)
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(username=username, password=password)
        current_time = datetime.datetime.now()
        if user:
            if user.blocked_time.timestamp() < current_time.timestamp():
                user.login_attempts = 0
            if user.login_attempts == 5:
                user.blocked_time = current_time +  datetime.timedelta(minutes=5)
            if user.login_attempts >= 5 and user.blocked_time.timestamp() >= current_time.timestamp():
                # User has been temporarily blocked from service
                return render(request, 'registration/login.html',
                              {'error_message': "You have exceeded maximum the number of attempts."
                                                "Please wait for few hours before attempting again!"})
            if user.password_expiry.timestamp() <= current_time.timestamp():
                # User's password has expired
                messages.success(request, "Your password has expired. "
                                          "Kindly reset your password to access your account again!")
                return redirect("password_reset")
            user.login_attempts = 0
            user.blocked_time = current_time
            user.password_expiry = current_time + datetime.timedelta(minutes=10)
            user.save()
            login(request, user)
            return redirect("accounts:index")
```



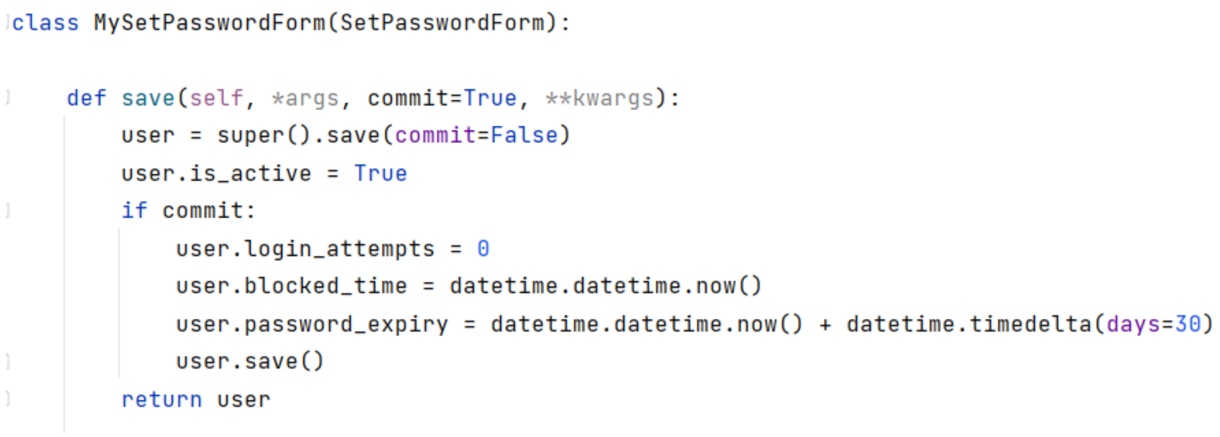
```python
        else:
            try:
                user = MyUser.objects.get(username=username)
            except:
                # No user with that Username exists
                return render(request, 'registration/login.html',
                              {'error_message': "Invalid login details provided.\nLogin failed."})
            else:
                # User provided wrong password
                user.login_attempts += 1
                user.save()
                if user.login_attempts == 5:
                    user.blocked_time = current_time +  datetime.timedelta(minutes=5)
                    user.save()
                if user.login_attempts >= 5 and user.blocked_time.timestamp() >= current_time.timestamp():
                    return render(request, 'registration/login.html',
                                  {'error_message': "You have exceeded maximum the number of attempts. "
                                                    "Please wait for few hours before attempting again!"})
                else:
                    message = "You have made " + str(user.login_attempts) + \
                              " number of unsuccessful attempts. After " + str(5 - user.login_attempts) + \
                              " more wrong attempts, your account will be temporarily deactivated."
                    return render(request, 'registration/login.html', {"error_message": message})
    else:
        return render(request, 'registration/login.html', {'error_message': None})
```

★ Verification during password reset request



```
def password_reset_request(request):
    if request.method == "POST":
        password_reset_form = PasswordResetForm(request.POST)
        if password_reset_form.is_valid():
            data = password_reset_form.cleaned_data['email']
            associated_users = MyUser.objects.filter(Q(email=data))
            if associated_users.exists():
                for user in associated_users:
                    subject = "Password Reset Requested"
                    email_template_name = "registration/password/password_reset_email.txt"
                    c = {
                        "email": user.email,
                        'domain': '127.0.0.1:8000',
                        'site_name': 'Website',
                        "uid": urlsafe_base64_encode(force_bytes(user.pk)),
                        "user": user,
                        'token': default_token_generator.make_token(user),
                        'protocol': 'http',
                    }
                    email = render_to_string(email_template_name, c)
                    try:
                        send_mail(subject, email, 'admin@example.com' , [user.email], fail_silently=False)
                    except BadHeaderError:
                        return HttpResponse('Invalid header found.')
                    messages.success(request, 'A message with reset password instructions has been sent to your inbox.')
                    return redirect("password_reset_done")
    password_reset_form = PasswordResetForm()
    return render(request=request, template_name="registration/password/password_reset.html",
                  context={"password_reset_form":password_reset_form})
```

★ Update the User parameters when a Password reset is confirmed. (“Forgot Password” and “Reset Password” are implemented as the same thing in this application)



```
class MySetPasswordForm(SetPasswordForm):

    def save(self, *args, commit=True, **kwargs):
        user = super().save(commit=False)
        user.is_active = True
        if commit:
            user.login_attempts = 0
            user.blocked_time = datetime.datetime.now()
            user.password_expiry = datetime.datetime.now() + datetime.timedelta(days=30)
            user.save()
        return user
```

Phase 2

Following are some of the functionalities/features that can be added to Phase 1 implementation to provide better security.

★ **Strong Password Checker**: A functionality that accepts new users (or new passwords through reset) only when the password that is being set is strong and hard enough to break. This can include compulsion use of some digits, lowercase letters, uppercase letters, and some special character in the password. This will overall increase the possible combination of passwords, thus making it difficult for attackers to guess the correct password.

★ Use of **Security Pin** as Master key for all user authenticating purposes inside the application (other than for Login).

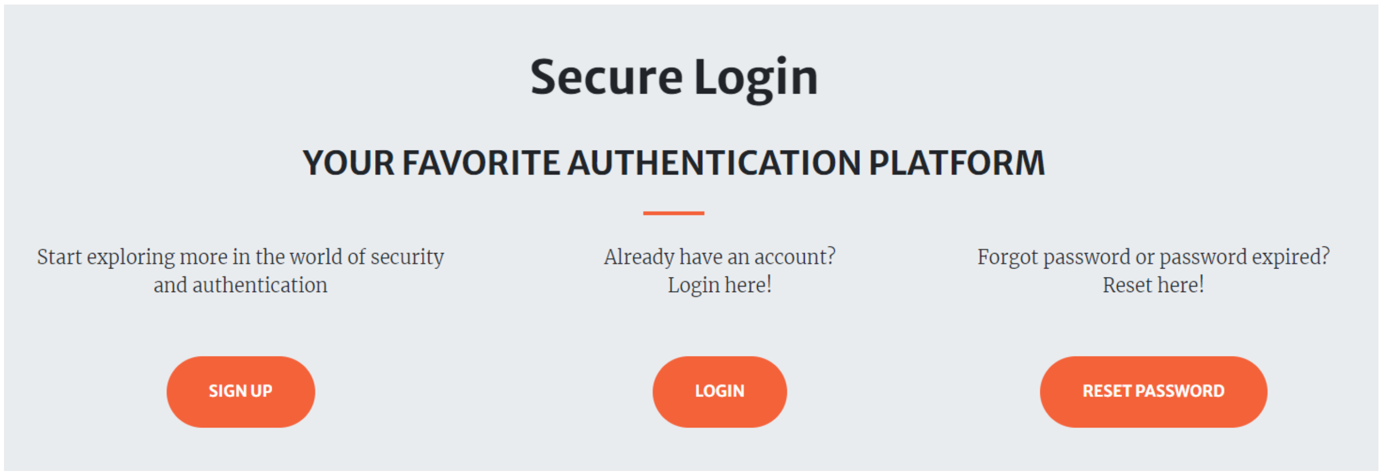
★ Use of **several factor authentication** like - OTP + Security Pin + Captcha to minimise Spoof attack.

★ **Password Expiry Reminder**

★ Storing **3 most recent password** to avoid password repetition

Results

Following are some of the screenshots from the Authentication platform designed



Secure Login

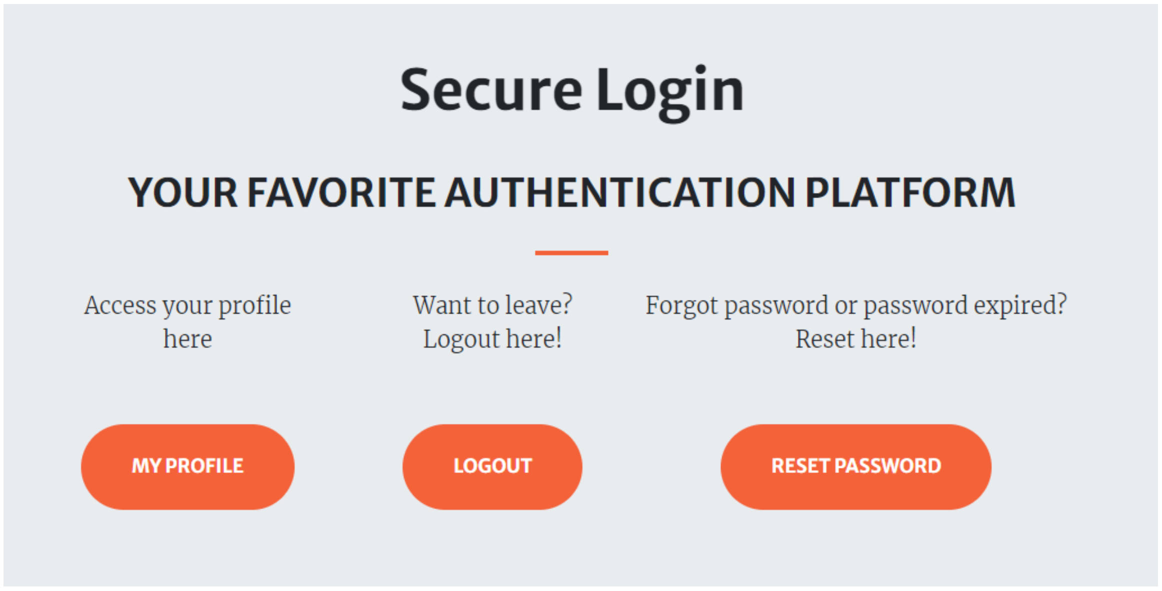
YOUR FAVORITE AUTHENTICATION PLATFORM

Start exploring more in the world of security and authentication

Already have an account? Login here!

Forgot password or password expired? Reset here!

(Home page view when the user has not been authenticated yet)



Secure Login

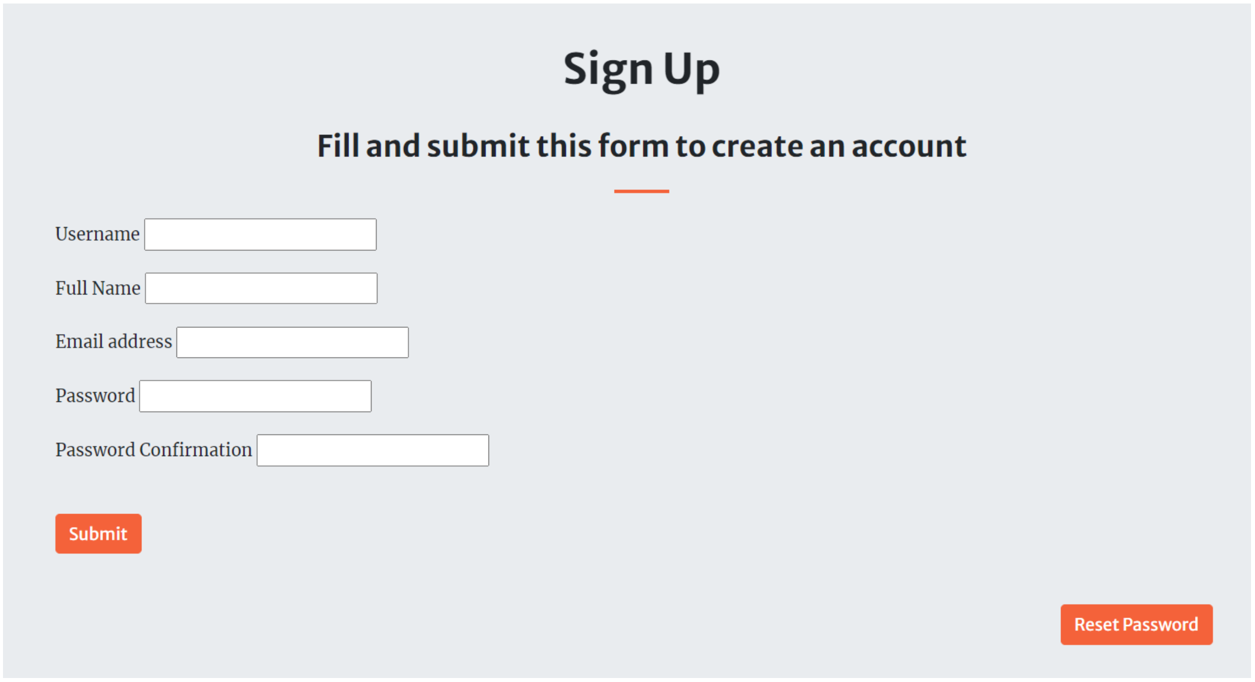
YOUR FAVORITE AUTHENTICATION PLATFORM

Access your profile here

Want to leave? Logout here!

Forgot password or password expired? Reset here!

(Home page view for an authenticated/logged in user)



Sign Up

Fill and submit this form to create an account

Username

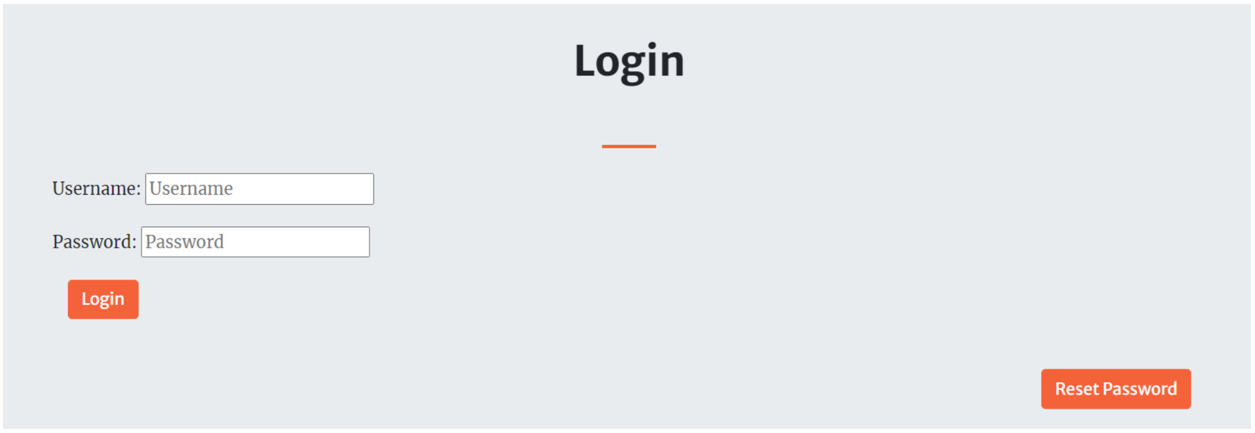
Full Name

Email address

Password

Password Confirmation

(Sign up page for an unauthenticated user. For already logged in users, they are first automatically logged out and this page is displayed)


Login

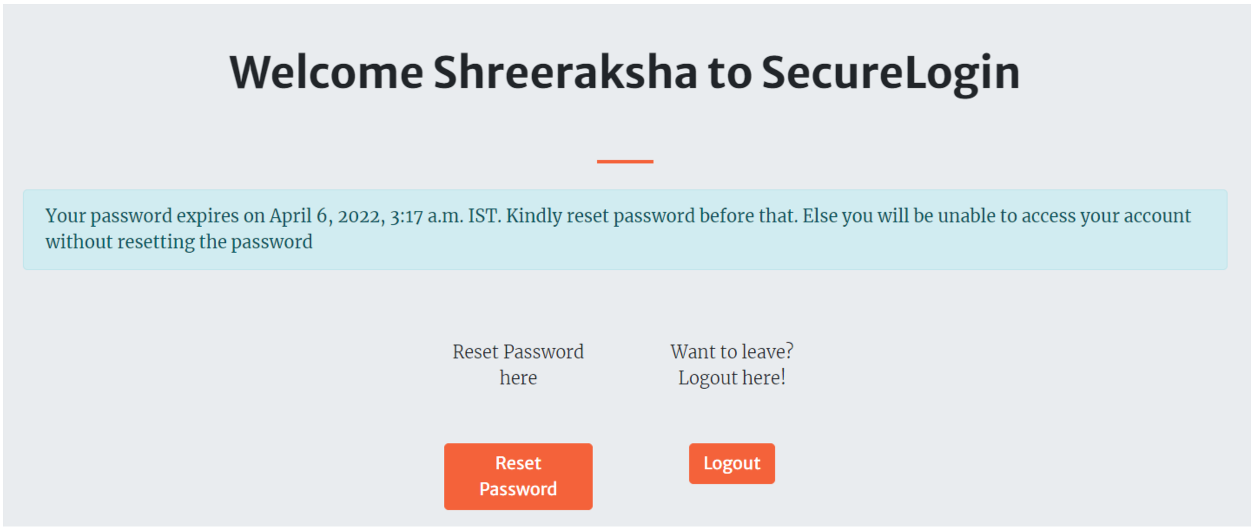
Username: Username

Password: Password

Login

Reset Password

(Login page for non-logged-in users. For already logged in users, they are first automatically logged out and this page is displayed)


Welcome Shreeraksha to SecureLogin

Your password expires on April 6, 2022, 3:17 a.m. IST. Kindly reset password before that. Else you will be unable to access your account without resetting the password

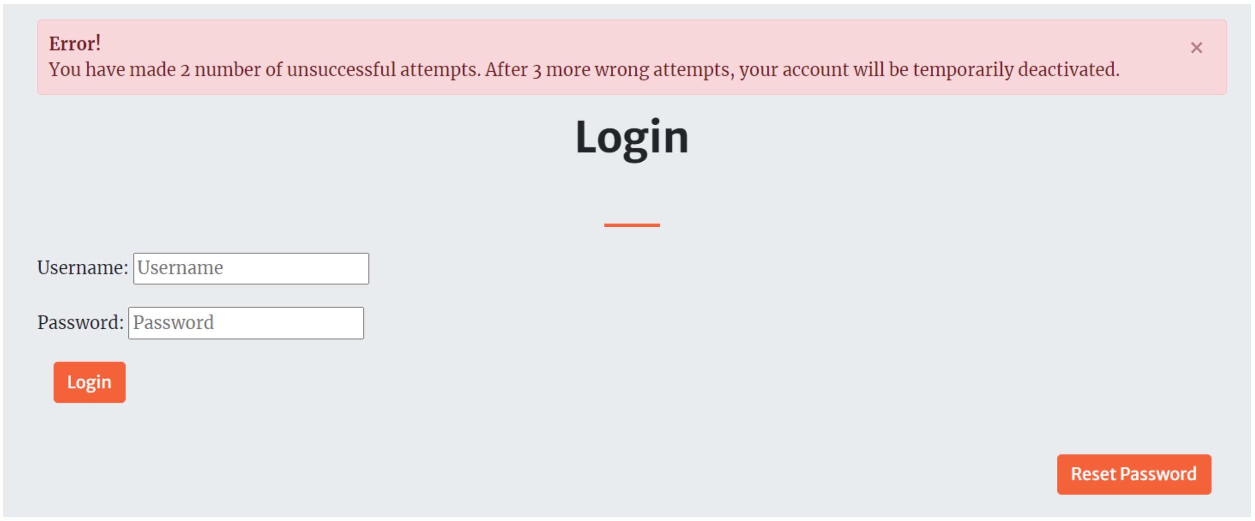
Reset Password here

Want to leave? Logout here!

Reset Password

Logout

(Profile page as viewed by an authenticated user)

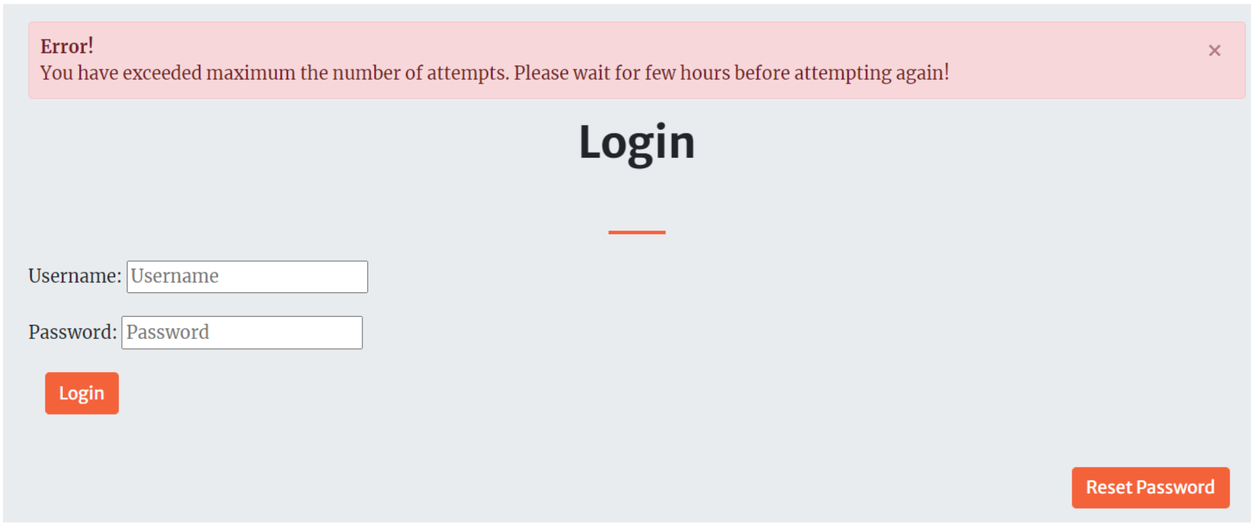

Error!
You have made 2 number of unsuccessful attempts. After 3 more wrong attempts, your account will be temporarily deactivated.

Login

Username: Username

Password: Password

(Failed login attempts)

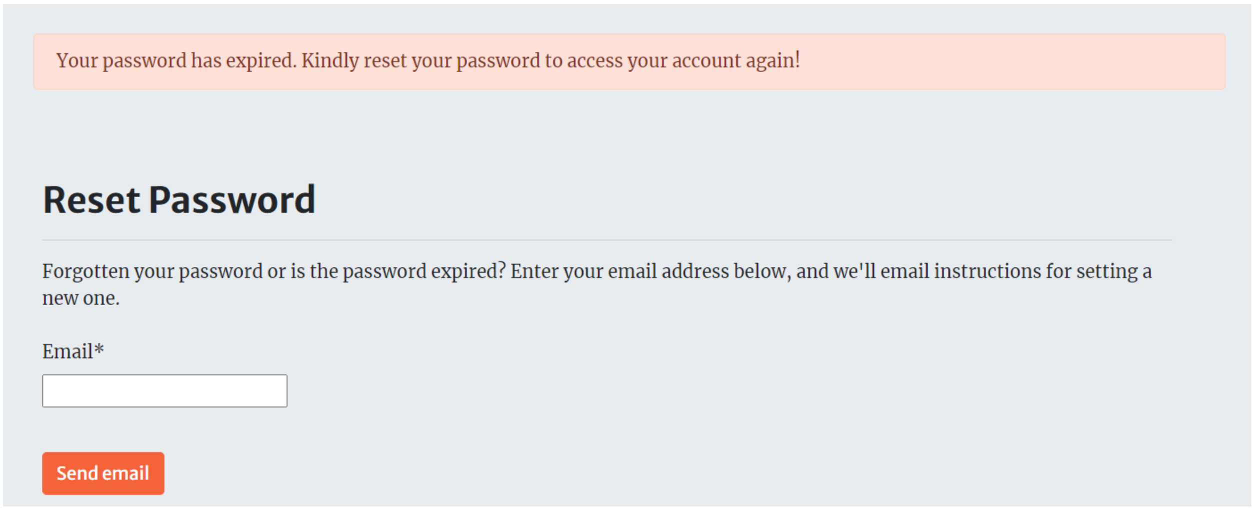

Error!
You have exceeded maximum the number of attempts. Please wait for few hours before attempting again!

Login

Username: Username

Password: Password

(User temporarily blocked for exceeding the number of failed login attempts within the given timespan)

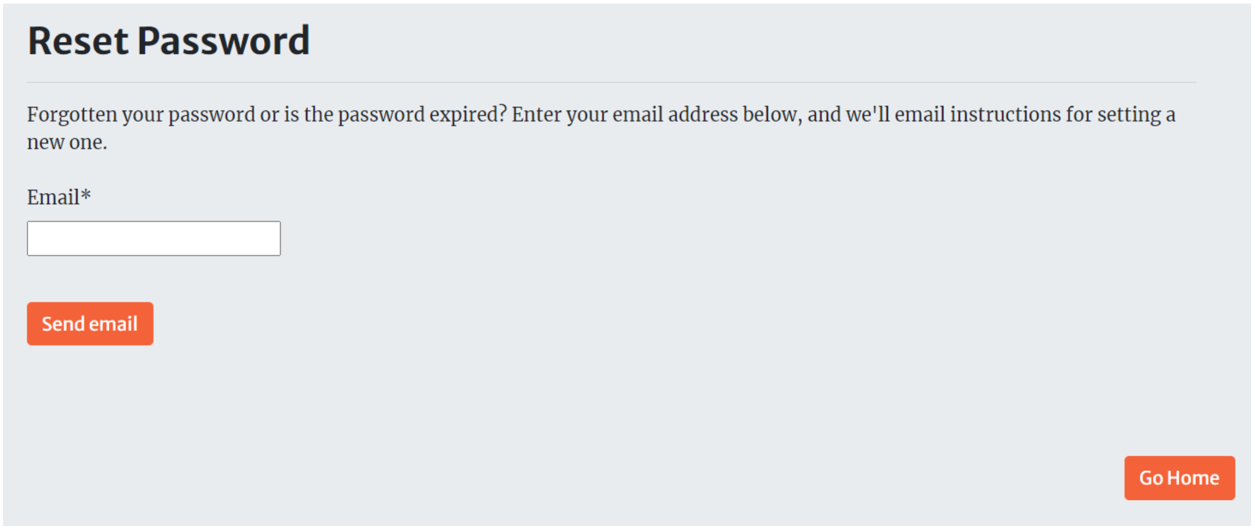

Your password has expired. Kindly reset your password to access your account again!

Reset Password

Forgotten your password or is the password expired? Enter your email address below, and we'll email instructions for setting a new one.

Email*

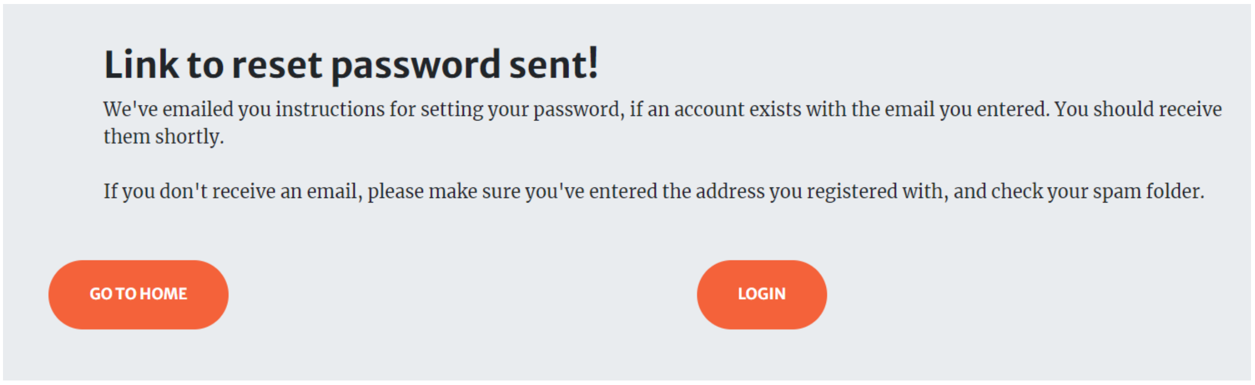
(User password has expired and the user is prompted to reset their password before further interaction)


Reset Password

Forgotten your password or is the password expired? Enter your email address below, and we'll email instructions for setting a new one.

Email*

(Password reset request page)

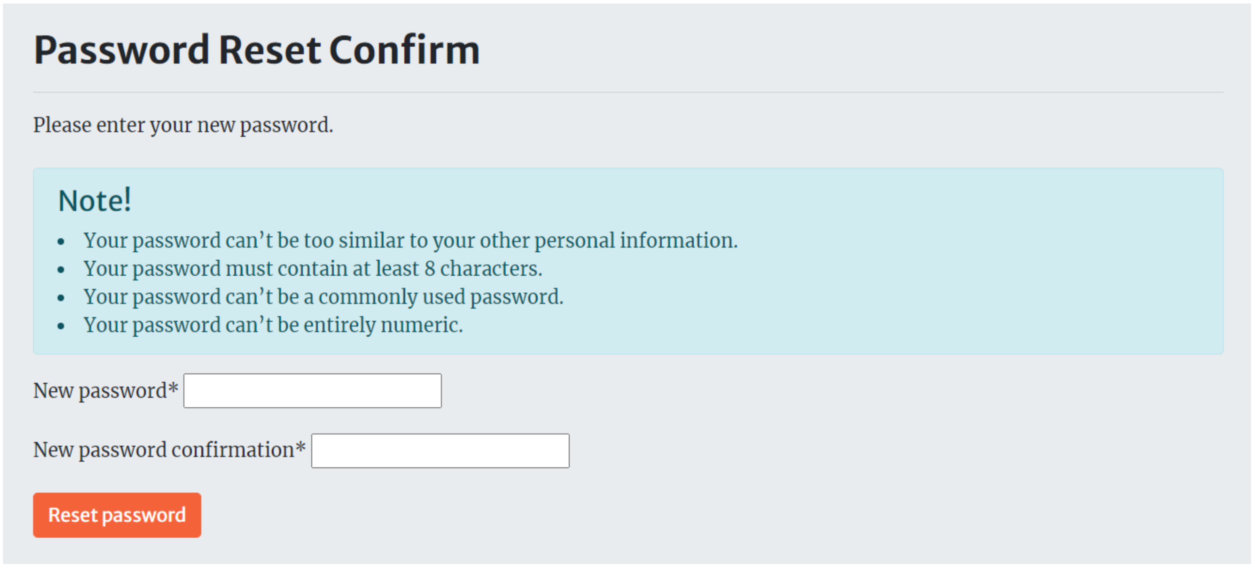


Link to reset password sent!

We've emailed you instructions for setting your password, if an account exists with the email you entered. You should receive them shortly.

If you don't receive an email, please make sure you've entered the address you registered with, and check your spam folder.

(Page after link for password resetting is mailed)



Password Reset Confirm

Please enter your new password.

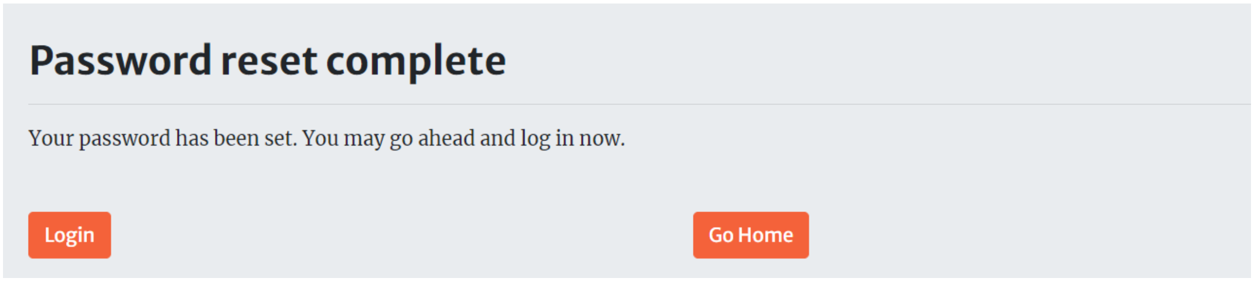
Note!

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password*

New password confirmation*

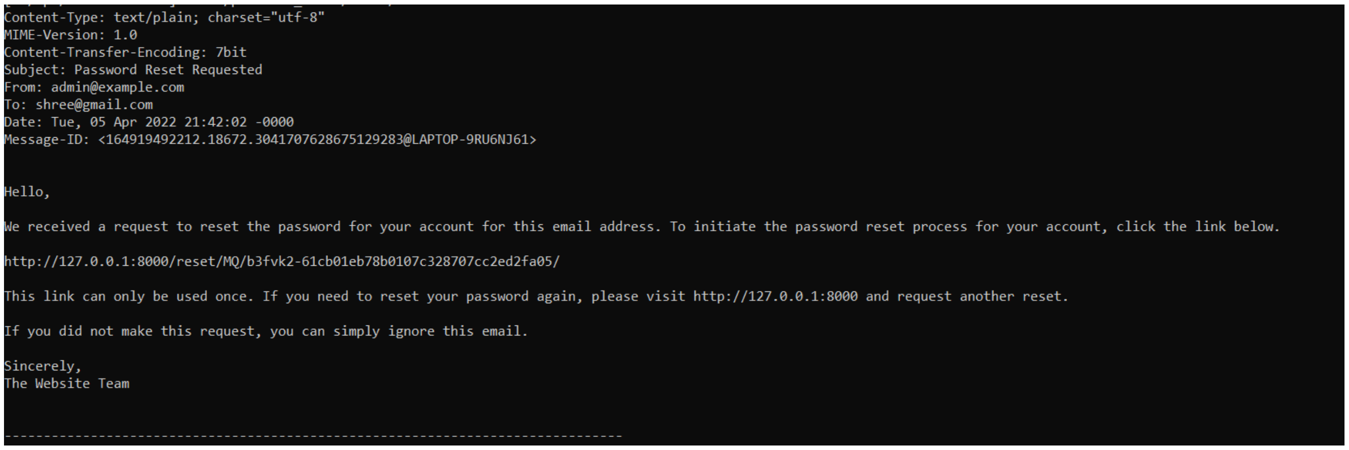
(Page containing form to set a new password)



Password reset complete

Your password has been set. You may go ahead and log in now.

(Page confirming the password update)



```
Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: Password Reset Requested
From: admin@example.com
To: shree@gmail.com
Date: Tue, 05 Apr 2022 21:42:02 -0000
Message-ID: <164919492212.18672.3041707628675129283@LAPTOP-9RU6NJ61>


Hello,

We received a request to reset the password for your account for this email address. To initiate the password reset process for your account, click the link below.

http://127.0.0.1:8000/reset/MQ/b3fvk2-61cb01eb78b0107c328707cc2ed2fa05/

This link can only be used once. If you need to reset your password again, please visit http://127.0.0.1:8000 and request another reset.

If you did not make this request, you can simply ignore this email.

Sincerely,
The Website Team


-------------------------------------------------------------------------------
```

(Sample mail containing a link to reset password)

Conclusions

An authentication system with 100% assured security is nearly impossible in the current world. Most service providers are still struggling to build a much secure system possible to thwart the attackers from gaining any sort of illegal or unauthorized access to their system/server. Though this is not entirely possible, it has attracted a lot of attention from all over the world. Most such systems now resort to using multiple levels of authentication like OTP generation, secret link generation, and many more. Even though these provide better security to some extent, it occurs at the cost of the user’s convenience to use that system.

Adding an additional layer of password expiry to the existing authentication system will definitely prove to enhance the system. This can considerably reduce the number of times an attacker gets to correctly guess the password, and hence prevent the attack to some extent. But there can also be a DNS type attack where the attacker simply attempts all the password wrong and temporarily disable the service thus disrupting users' convenience for that product.

References

[1] Secure Software, Inc. "The CLASP Application Security Process", 2005

The CLASP Application Security Process

[2] Michael Howard, David LeBlanc, and John Viega. "24 Deadly Sins of Software Security". "Sin 19: Use of Weak Password-Based Systems." Page 279, McGraw-Hill, 2010

[3] Rethinking Password Policies

https://www.usenix.org/sites/default/files/rethinking_password_policies_unabridged.pdf

[4] Password security — No change in 35 years?,   Viktor Taneski; Marjan Heričko; Boštjan Brumen, IEEE https://ieeexplore.ieee.org/abstract/document/6859779