Password Change Test

from django.contrib.auth.forms import PasswordChangeForm

from django.contrib.auth.models import User

from django.contrib.auth import views as auth\_views

from django.urls import resolve, reverse

from django.test import TestCase

class PasswordChangeTests(TestCase):

def setUp(self):

username = 'john'

password = 'secret123'

# Create user

user = User.objects.create\_user(

username=username, email='john@doe.com', password=password)

# Login user

# Now logged in users can change the password

self.client.login(username=username, password=password)

self.response = self.client.get(reverse('password\_change'))

def test\_status\_code(self):

self.assertEqual(self.response.status\_code, 200)

def test\_url\_resolves\_correct\_view(self):

view = resolve('/settings/change-password/')

self.assertEqual(view.func.view\_class, auth\_views.PasswordChangeView)

def test\_csrf(self):

self.assertContains(self.response, 'csrfmiddlewaretoken')

def test\_contains\_form(self):

form = self.response.context.get('form')

self.assertIsInstance(form, PasswordChangeForm)

def test\_form\_inputs(self):

'''

The view must contain four inputs: csrf, old\_password, new\_password1,

new\_password2

'''

self.assertContains(self.response, '<input', 5)

self.assertContains(self.response, 'type="password"', 3)

self.assertContains(self.response, 'type="submit"', 1)

class LoginRequiredPasswordChangeTests(TestCase):

def test\_redirection(self):

url = reverse('password\_change')

login\_url = reverse('signin')

response = self.client.get(url)

self.assertRedirects(response, f'{login\_url}?next={url}')

class PasswordChangeTestCase(TestCase):

'''

Base test case for form processing

accepts a `data` dict to POST to the view.

'''

def setUp(self, data={}):

# Sign up then sign in

self.user = User.objects.create\_user(

username='john', email='john@doe.com', password='old\_password')

self.client.login(username='john', password='old\_password')

# Browsing Password Change Page and Posting data in it!

self.url = reverse('password\_change')

self.response = self.client.post(self.url, data)

class SuccessfulPasswordChangeTests(PasswordChangeTestCase):

def setUp(self):

super().setUp({

'old\_password': 'old\_password',

'new\_password1': 'new\_password',

'new\_password2': 'new\_password',

})

def test\_redirection(self):

'''

A valid form submission should redirect the user

'''

self.assertRedirects(self.response, reverse('password\_change\_done'))

def test\_password\_changed(self):

'''

refresh the user instance from database to get the new password

hash updated by the change password view.

'''

self.user.refresh\_from\_db()

# Make

self.assertTrue(self.user.check\_password('new\_password'))

def test\_user\_authentication(self):

'''

Create a new request to an arbitrary page.

The resulting response should now have an `user` to its context, after a

successful sign up.

'''

response = self.client.get(reverse('board:home'))

user = response.context.get('user')

self.assertTrue(user.is\_authenticated)

class InvalidPasswordChangeTests(PasswordChangeTestCase):

def test\_status\_code(self):

'''

An invalid form submission should return to the same page

'''

self.assertEqual(self.response.status\_code, 200)

def test\_form\_errors(self):

form = self.response.context.get('form')

self.assertTrue(form.errors)

def test\_didnt\_change\_password(self):

'''

refresh the user instance from the database to make

sure we have the latest data.

'''

self.user.refresh\_from\_db()

self.assertTrue(self.user.check\_password('old\_password'))