```
import sys
sys.path.append("..")
from unittest import TestCase
from app import app
from models import db, User, Story, Query
app.config['SQLALCHEMY_ECHO'] = False
app.config['TESTING'] = True
app.config['SQLALCHEMY_ECHO'] = False
app.config['DEBUG_TB_HOSTS'] = ['dont-show-debug-toolbar']
app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql:///newstracker-test'

db.drop_all()
db.create_all()

class User_Model(TestCase):

    def setUp(self):
        """Cleans up any existing users"""
        User.query.delete()
        user = User.register(
            "dummy", "dummypassword", "dumb@y.com", "dummy", "account")
        db.session.add(user)
        db.session.commit()
        self.user = user

    def tearDown(self):
        db.session.rollback()

    def test_register(self):
        user = User.register(
            "testuser", "test4444", "test@test.com", "test", "user")
```

```
        """Tests columns id and username as written in __repr__"""
        self.assertEqual(str(user), f"<ID: {user.id}, Username:{user.username}>")
        # self.assertEquals(str(user), f"<ID: 1, Username:testuser>")
        """Tests that columns are accessible as written"""
        # todo: make sure model is autonmatically assigning id, right now the following test fails
        # self.assertEquals(user.id, 1)
        self.assertEqual(user.username, "testuser")
        self.assertEqual(user.email, "test@test.com")
        self.assertEqual(user.first_name, "test")
        self.assertEqual(user.last_name, "user")
        """Tests that password is hashed and not accessible"""
        self.assertNotEqual(user.password, "test4444")

    def test_passes_authenticate(self):
        """Tests that authentication can be passed based off of pre-existing data"""
        user = User.authenticate(self.user.username, "dummypassword") #password is hashed on registration and needs to be passed as string to pass test
        self.assertNotEqual(user, False) # authenticate method returns false if fails, user object if passes

    def test_fails_authenticate(self):
        user = User.authenticate("incorrect", "dummypassword")
        self.assertEqual(user, False)

    def test_fails_authenticate_hashed(self):
        user = User.authenticate(self.user.username, self.user.password)
        self.assertEqual(user, False)
```