McMaster University – Software Engineering Technology

SFWRTECH-3RQ3 Software Requirements and Specification

Requirements Documentation

Student Name: Yihuan Zhang

Student Number: 400350335

Date submitted: October 29, 2020

Submitted to: Sean Watson

1. Introduction

This system is dedicated to outdoor activities, it will allow users to organize their photos based on geographical info. The users not only will have records of their own travel, but also be able to share location with each others, and find places to explore using the integrated rating system.

2. Description

In this section we will describe the basic who and what of the system. What is the outline of the system, who will be the end users, who will be the owners and maintainers and what constraints do we have.

2.1. System Outline

The system will collect the users' location info, allows user to upload a photo(s) and select whether they prefer to share with others in the system. On an integrated map, users will be able to view their own pictures on a map, or explore other places based on third-parties or other users' pictures.

2.2. Users

The end-users of the system are mainly:

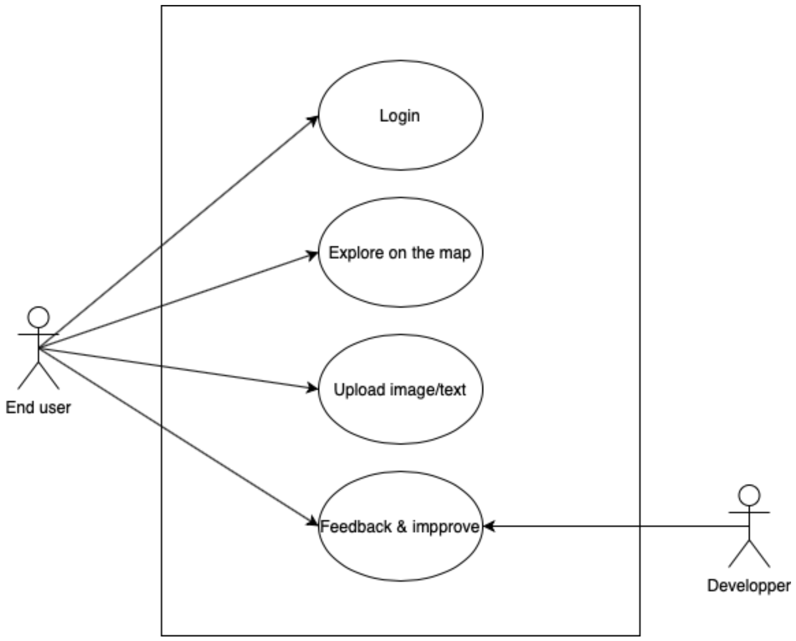
- outdoor enthusiasts or photographers who want to organize their photos;
- People who seek places to travel during weekend or vacation;
- Administrational organizations who want to publish information to the tourists.

2.3. Owners

The owners and maintainers of the system will include:

- The investor of the project (angel investor, governors, or other party who sponsors the system) will be the proprietor of the system, and will have access to the database.
- The administrational organizations or the landlords of the places will help maintain and support the system by updating their important information.
- The software team will be responsible to fix bugs and other issues according to the customers reports.

2.4. Use Case Diagrams



3. Constraints

The system must be accessible on all major browsers on mobile and desktop devices, must be hosted by Linux O/S, and must comply with local laws, codes & regulations.

4. System Features

The following functional requirements will detail what tasks the system must accomplish.

4.1. *LOGIN*

The system offers end-users an option to log in. In order to use some system features such as *privacy preference* and *personal album*, login is required. The login needs username as primary key, with passwords contains at least one letter and one number.

4.2. *PHOTO UPLOAD*

The system shall allow an end-user (whether login or not) to upload his photo to the database along with some other information.

4.2.1. *LOCATION*

The system shall retrieve a geographical information from the end-user's device before uploading the photo. If the geolocation is turned off by user, use IP address location.

4.2.2. *DESCRIPTION*

The system shall allow a short description (up to 140 characters) given from the end-user.

4.2.3. *PRIVACY PREFERENCE*

a) The system shall allow a login user to select the privacy preference before uploading the photo(s).

b) By choosing "private", only the user himself can only view the photo through his *Personal album* page; by choosing "public", the photo will be display on both *Personal Album* (if possible) and *Explore* pages.

c) The default selection is "public".

4.3. ALBUM & EXPLORE

The system shall have two sections under this feature to display end-users the photos. The system shall associate the updated photos with location in physical space and display along with a short description.

4.3.1. *PERSONAL ALBUM*

The system shall generate a map and display all the login user's photos on the integrated map. The login user will also have permission to delete and edit *photo/description/privacy preference* in his personal album.

4.3.2. *EXPLORE*

The system shall generate a map and display all end-users' public photos on the integrated map. End-users will only have permission to review.

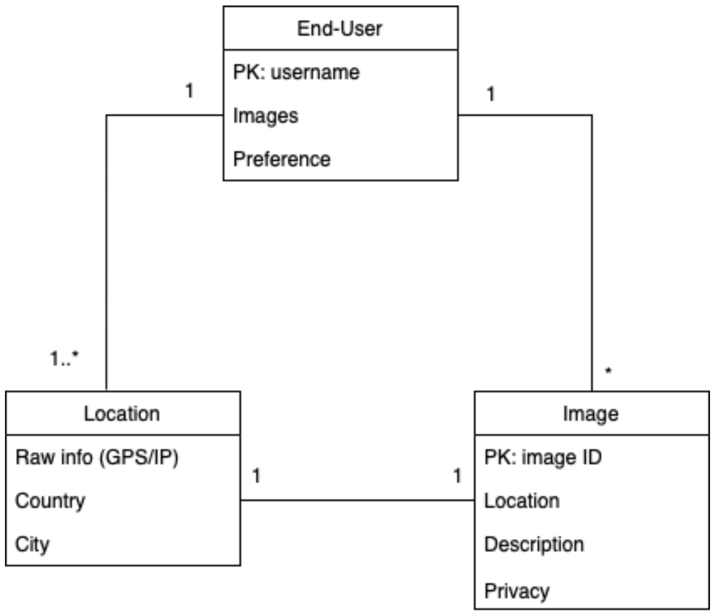
The system shall have a rating system that allows login users to rate the photos on the map.

5. Data Requirements

5.1. Overview

Identified Noun	Description	Notes
End-user	Uses the system to upload/explore photo and location	Included
Organization admin	Another name of end-user	Excluded
Developer/admin	Receives feedback from customer and moderates the photo/text in the system	Included
Image	Needs to contain the location and description	Included
Location	Needs to be one unique geographical data	Included
Description	Needs to be no more than 140 characters	Excluded

5.2. Class Diagram



6. Interface

The system shall interface with the GPS, photo album or camera on the device, it will also interface with an open source online map platform to generate the integrated map.

7. Quality attributes

7.1. USER INTERFACE

The system shall have an easy to use interface that complies with any application web standards.

7.2. FEED BACK

The system shall allow users to report bugs and issues to the development team.

7.3. LOGIN & REGISTRATION

The system shall require a password consisting of at least one letter and one number.

7.4. SECURITY

The system shall encrypt all passwords and usernames in the database.

7.5. PHOTO UPLOAD

The system shall use standard form for inputing description and selecting the privacy preference, these items include:

- Image

- Textarea
- Radio button
- Submit

7.6. ALBUM & EXPLORE

The system shall plot the map in no more than 5 seconds.

A. Background Research

In order to determine how to associate photos with geographical information, I looked into some other existed social media website and photography platform such as Facebook and Flickr. Flickr uses the Mapbox as mapping platform to plot their map. Other than Mapbox, Google Map SDK also provides API to embed static map image to the website.

In terms of the geographical information, the easiest way to locate the user's location is using HTML Geolocation API. two other possible ways to acquire location info are either to call the GPS module or through the IP address.

Moreover, I also looked up on Pinterest and Amazon to see how they save public and private lists for Wishlists/Pins, to help setup the *Private Album* feature.

B. Scenarios

Attached are descriptions of how system will behave in various situations, both in success and error scenarios as well as some descriptions of how the system will not behave.

B.1. LOGIN

B.1.1.POSITIVE NORMAL

(1) User clicks the login button and inputs valid username & password

(2) System checks the validity at the back-end

(3) System prompt "login successful" window

B.1.2.POSITIVE ABNORMAL

(1) User clicks the login button and inputs invalid username & password

(2) System checks the validity at the back-end

(3) System prompt "login fail" window

B.2. LOCATION ACQUIRING

B.2.1.POSITIVE NORMAL

(1) The end-user submits the photo

(2) The system requests the location info

(3) The end-user returns a location info

B.2.2.POSITIVE ABNORMAL

(1) The end-user submits the photo

(2) The system requests the location info

(3) The end-user returns a void location info

B.3.DESCRIPTION

B.3.1.POSITIVE NORMAL

(1) User types no more than 140 characters and submits

(2) User leaves the description part blank

B.3.2.POSITIVE ABNORMAL

(1) User exceeds 140 characters

(2) System prompt error notification

B.4. PRIVACY PREFERENCE

B.4.1.POSITIVE NORMAL

(1) User leaves default selection (public) without login

(2) User selects "public" after login

(3) User selects "private" after login

B.4.2.NEGATIVE

(1) User selects "private" without login

(2) User selects more than one option

B.5.IMAGE BROWSING

B.5.1.POSITIVE NORBAL

(1) User clicks the "EXPLORE" button

(2) User is redirected to a new page

(3) User brows all the image, clicks the interested one

(4) The selected image zooms in

B.5.5.POSITIVE ABNORMAL

(1) User clicks the "EXPLORE" button

(2) User is redirected to a new page

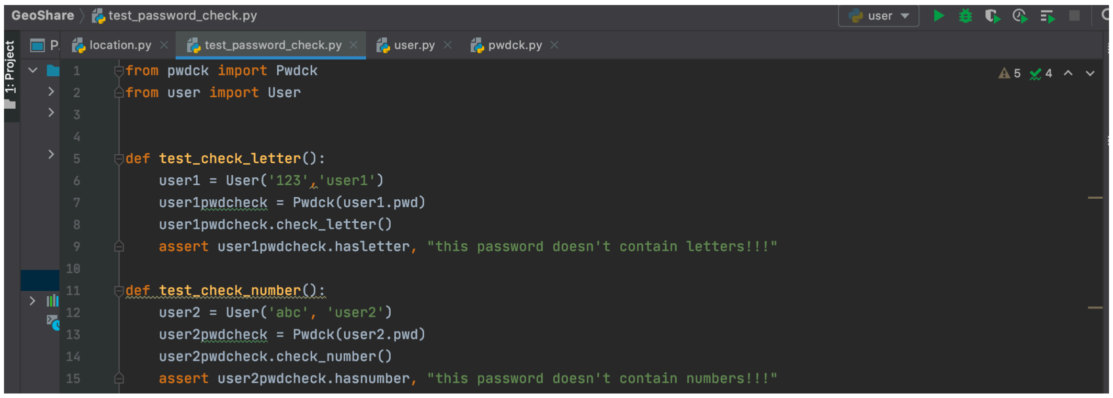
(3) User brows all the image, clicks the interested one

(4) The selected image cannot be loaded

(5) The systems prompts error manage

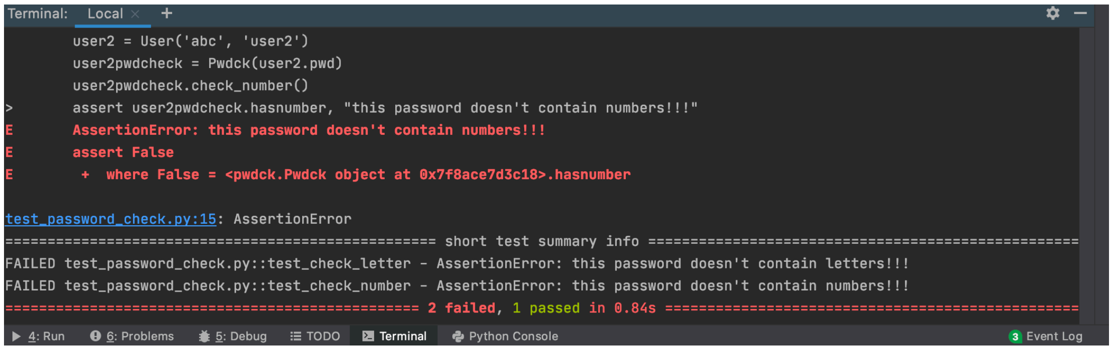
C. Tests

C.1. Functional requirement: the login needs username as primary key, with passwords contains at least one letter and one number.



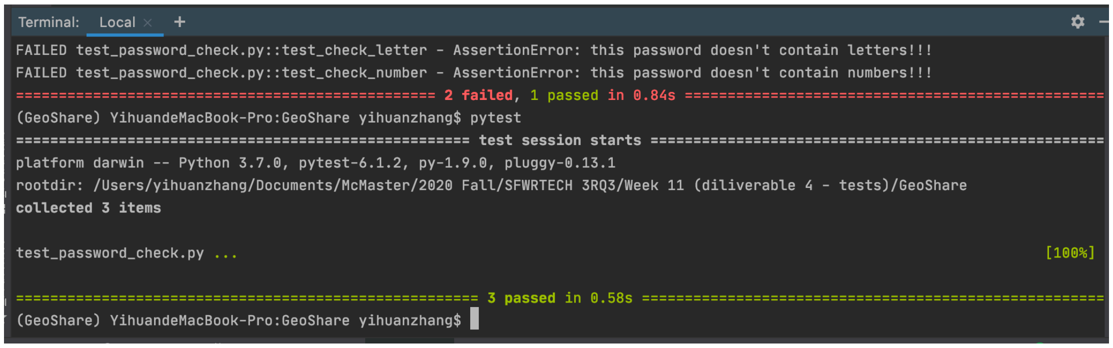
```
from pwdck import Pwdck
from user import User


def test_check_letter():
    user1 = User('123','user1')
    user1pwdcheck = Pwdck(user1.pwd)
    user1pwdcheck.check_letter()
    assert user1pwdcheck.hasletter, "this password doesn't contain letters!!!"

def test_check_number():
    user2 = User('abc', 'user2')
    user2pwdcheck = Pwdck(user2.pwd)
    user2pwdcheck.check_number()
    assert user2pwdcheck.hasnumber, "this password doesn't contain numbers!!!"
```

The above testing methods is used to check if user's password contains letter or number, user1 has the password of "123" (which does not contain letter) and user2 has the password of "abc" (which does not contain number), both of them are expected to be failed:



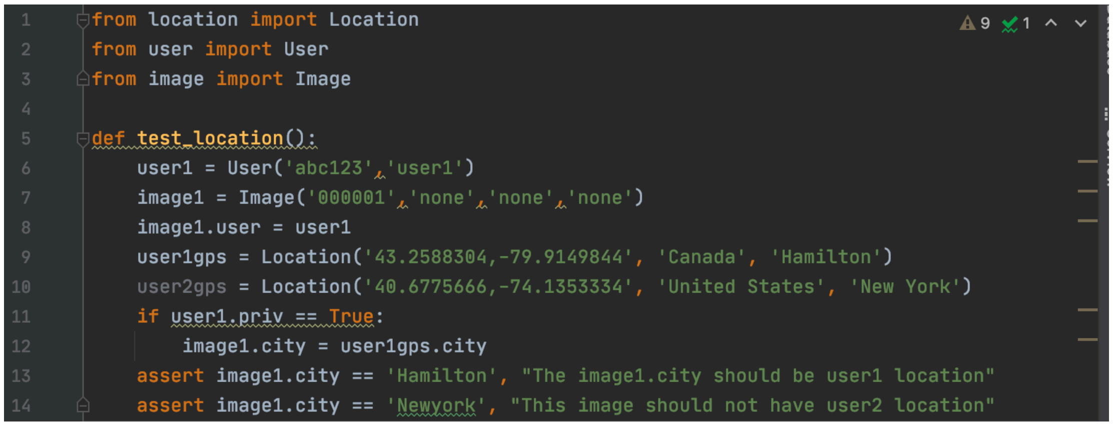
```
        user2 = User('abc', 'user2')
        user2pwdcheck = Pwdck(user2.pwd)
        user2pwdcheck.check_number()
>       assert user2pwdcheck.hasnumber, "this password doesn't contain numbers!!!"
E       AssertionError: this password doesn't contain numbers!!!
E       assert False
E        +  where False = <pwdck.Pwdck object at 0x7f8ace7d3c18>.hasnumber

test_password_check.py:15: AssertionError
=================================================== short test summary info ===================================================
FAILED test_password_check.py::test_check_letter - AssertionError: this password doesn't contain letters!!!
FAILED test_password_check.py::test_check_number - AssertionError: this password doesn't contain numbers!!!
================================================= 2 failed, 1 passed in 0.84s =================================================
```

However, when switching the password values of user1 and user2, both tests are passed. This verifies both "check_letter()" and "check_number()" methods.



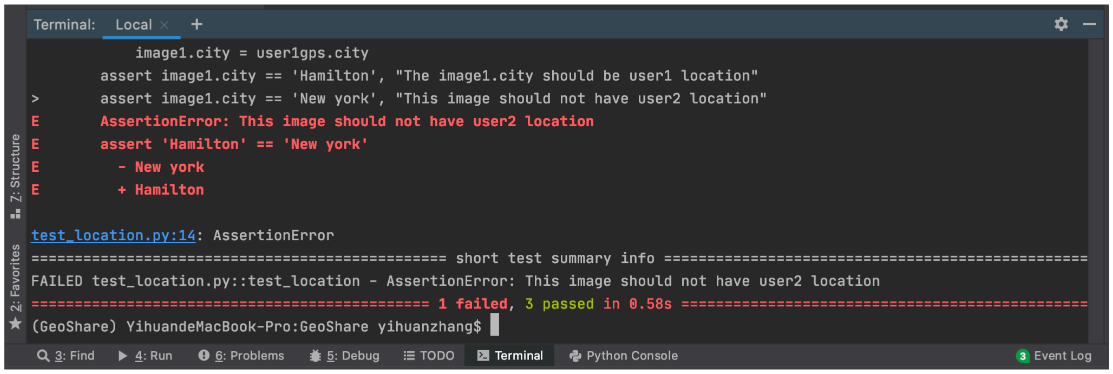
```
FAILED test_password_check.py::test_check_letter - AssertionError: this password doesn't contain letters!!!
FAILED test_password_check.py::test_check_number - AssertionError: this password doesn't contain numbers!!!
================================================= 2 failed, 1 passed in 0.84s =================================================
(GeoShare) YihuandeMacBook-Pro:GeoShare yihuanzhang$ pytest
===================================================== test session starts =====================================================
platform darwin -- Python 3.7.0, pytest-6.1.2, py-1.9.0, pluggy-0.13.1
rootdir: /Users/yihuanzhang/Documents/McMaster/2020 Fall/SFWRTECH 3RQ3/Week 11 (diliverable 4 - tests)/GeoShare
collected 3 items

test_password_check.py ...                                                                                            [100%]

====================================================== 3 passed in 0.58s ======================================================
(GeoShare) YihuandeMacBook-Pro:GeoShare yihuanzhang$
```

C.2. The system shall retrieve a GPS info from the user, then add to the image.



```python
from location import Location
from user import User
from image import Image

def test_location():
    user1 = User('abc123','user1')
    image1 = Image('000001','none','none','none')
    image1.user = user1
    user1gps = Location('43.2588304,-79.9149844', 'Canada', 'Hamilton')
    user2gps = Location('40.6775666,-74.1353334', 'United States', 'New York')
    if user1.priv == True:
        image1.city = user1gps.city
    assert image1.city == 'Hamilton', "The image1.city should be user1 location"
    assert image1.city == 'Newyork', "This image should not have user2 location"
```

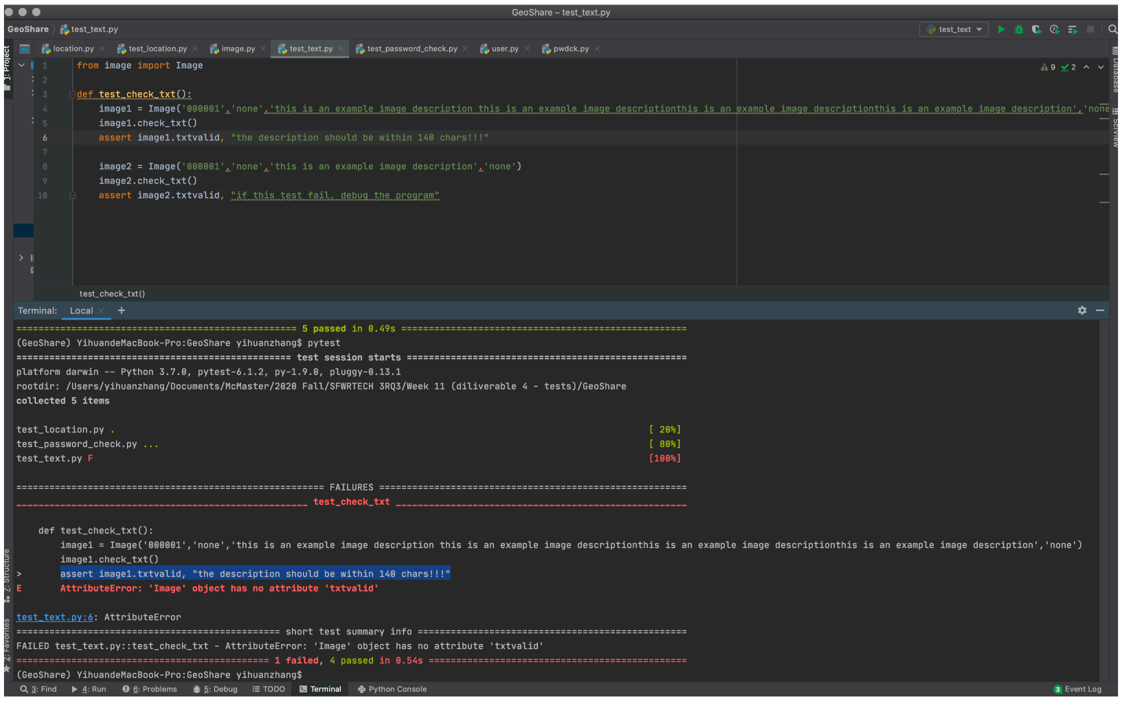
The testing for location attribute. Since the image1 belongs to user1, it will take user1's location into its attribute "image.city". Above testing includes 2 tests, one is to check if "image1.city" successfully inherits user1's location (Hamilton), the other test is to check if it has another user's location. The first test is expected to be passed, the second one is expected to be failed.



```
            image1.city = user1gps.city
        assert image1.city == 'Hamilton', "The image1.city should be user1 location"
>       assert image1.city == 'New york', "This image should not have user2 location"
E       AssertionError: This image should not have user2 location
E       assert 'Hamilton' == 'New york'
E         - New york
E         + Hamilton

test_location.py:14: AssertionError
============================================= short test summary info ==============================================
FAILED test_location.py::test_location - AssertionError: This image should not have user2 location
============================================= 1 failed, 3 passed in 0.58s ==============================================
(GeoShare) YihuandeMacBook-Pro:GeoShare yihuanzhang$
```

The second test failed as image1 only retrieves GPS info from user1.

C.3. The system shall allow an end-user to upload his photo to the database along with some description, the maximum character is 140.

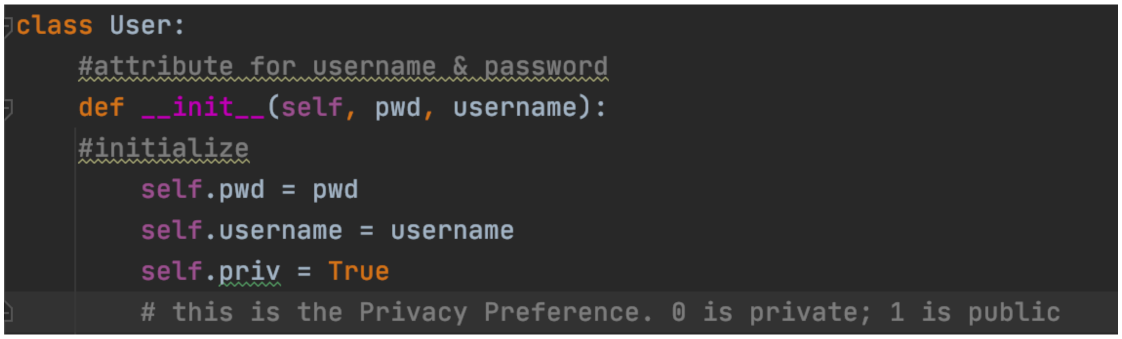


```
from image import Image

def test_check_txt():
    image1 = Image('000001','none','this is an example image description this is an example image descriptionthis is an example image descriptionthis is an example image description','none
    image1.check_txt()
    assert image1.txtvalid, "the description should be within 140 chars!!!"

    image2 = Image('000001','none','this is an example image description','none')
    image2.check_txt()
    assert image2.txtvalid, "if this test fail, debug the program"
```

```
==================================================== 5 passed in 0.49s ====================================================
(GeoShare) YihuandeMacBook-Pro:GeoShare yihuanzhang$ pytest
=================================================== test session starts ===================================================
platform darwin -- Python 3.7.0, pytest-6.1.2, py-1.9.0, pluggy-0.13.1
rootdir: /Users/yihuanzhang/Documents/McMaster/2020 Fall/SFWRTECH 3RQ3/Week 11 (diliverable 4 - tests)/GeoShare
collected 5 items

test_location.py .                                                                                                  [ 20%]
test_password_check.py ...                                                                                          [ 80%]
test_text.py F                                                                                                      [100%]

======================================================== FAILURES ========================================================
_____________________________________________________ test_check_txt _____________________________________________________

    def test_check_txt():
        image1 = Image('000001','none','this is an example image description this is an example image descriptionthis is an example image descriptionthis is an example image description','none')
        image1.check_txt()
>       assert image1.txtvalid, "the description should be within 140 chars!!!"
E       AttributeError: 'Image' object has no attribute 'txtvalid'

test_text.py:6: AttributeError
================================================= short test summary info =================================================
FAILED test_text.py::test_check_txt - AttributeError: 'Image' object has no attribute 'txtvalid'
=============================================== 1 failed, 4 passed in 0.54s ===============================================
(GeoShare) YihuandeMacBook-Pro:GeoShare yihuanzhang$
```

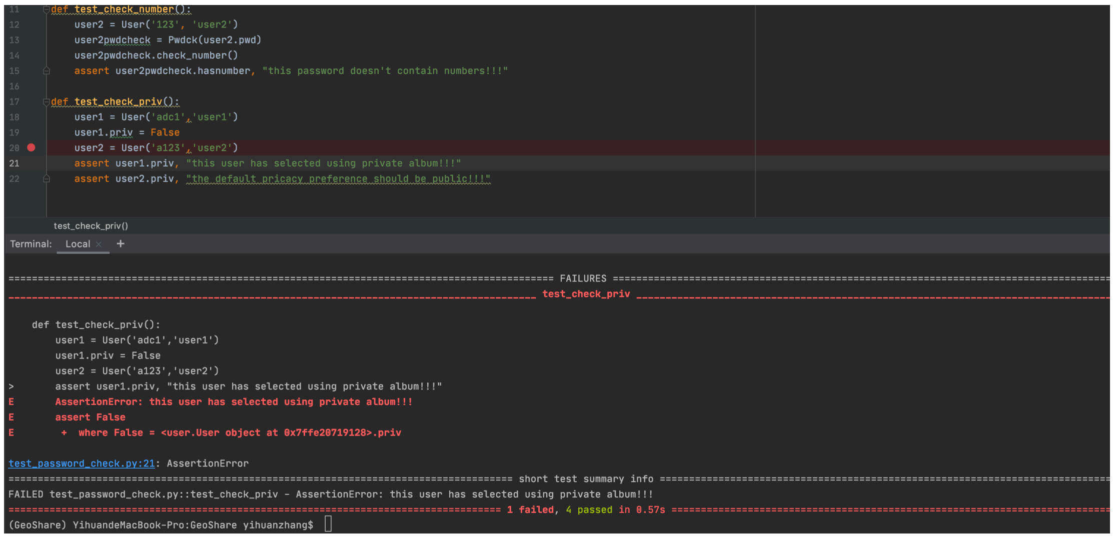
To check if the “check_txt()” method is functioning, two tests are implemented. The first test checks image1 contains 145 characters in its description and the second test checks image2 which contains 36 characters. The first test is intended to be failed and the second test is expected to be passed.

C.4. The system shall allow a login user to select the privacy preference, the default selection is “public”.



```
class User:
    #attribute for username & password
    def __init__(self, pwd, username):
    #initialize
        self.pwd = pwd
        self.username = username
        self.priv = True
        # this is the Privacy Preference. 0 is private; 1 is public
```

In User class, the “Privacy Preference” is determined by its attribute “priv”. And by default, this attribute is already set as “True” in the initialization.



```
def test_check_number():
    user2 = User('123', 'user2')
    user2pwdcheck = Pwdck(user2.pwd)
    user2pwdcheck.check_number()
    assert user2pwdcheck.hasnumber, "this password doesn't contain numbers!!!"

def test_check_priv():
    user1 = User('adc1','user1')
    user1.priv = False
    user2 = User('a123','user2')
    assert user1.priv, "this user has selected using private album!!!"
    assert user2.priv, "the default pricacy preference should be public!!!"
```

```
test_check_priv()
Terminal:  Local  +

========================================================================================== FAILURES ==========================================================================================
_________________________________________________________________________________________ test_check_priv __________________________________________________________________________________________

    def test_check_priv():
        user1 = User('adc1','user1')
        user1.priv = False
        user2 = User('a123','user2')
>       assert user1.priv, "this user has selected using private album!!!"
E       AssertionError: this user has selected using private album!!!
E       assert False
E        +  where False = <user.User object at 0x7ffe20719128>.priv

test_password_check.py:21: AssertionError
==================================================================================== short test summary info ====================================================================================
FAILED test_password_check.py::test_check_priv - AssertionError: this user has selected using private album!!!
=================================================================================== 1 failed, 4 passed in 0.57s ===================================================================================
(GeoShare) YihuandeMacBook-Pro:GeoShare yihuanzhang$
```

To check this function, two users are created to test them, user1 has changed the setting to "private", user2 leave the privacy preference unchanged. The results show that the first test has failed, the second test remains "True" (public) in the "priv" attribute.

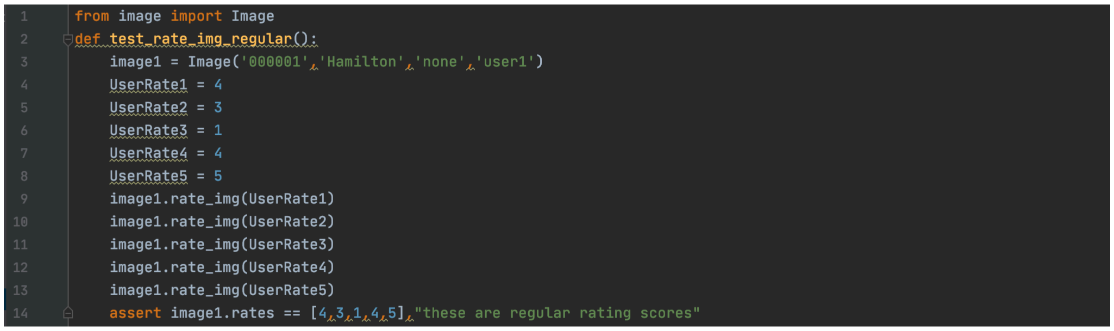
C.5. The system shall generate a map, and display all the login user's photos on the map. The login users will also have permission to delete and edit *photo/description/privacy preference* in his personal album.

This functional requirement is not able to be test, for it needs extra libraries and API to generate the map and involves some GUI testing which are beyond the scope of this class.

C.6. The system shall generate a map and display all end-users' public photos on the map. End-users will only have permission to review.

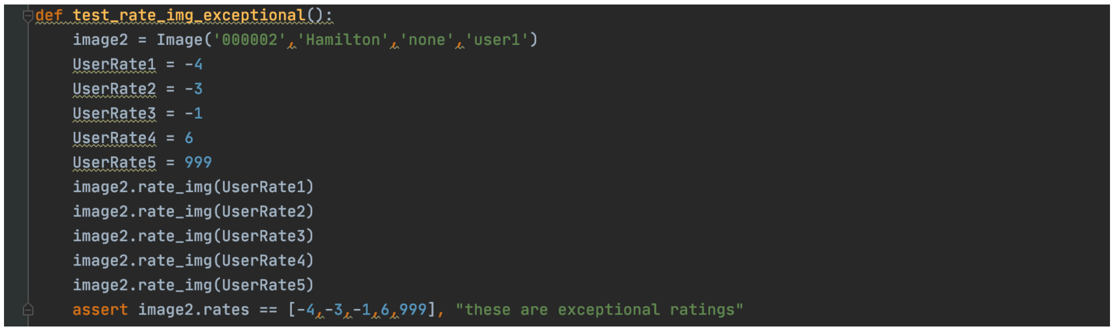
This functional requirement is not able to be test, for it needs extra libraries and API to generate the map and involves some GUI testing which are beyond the scope of this class.

C.7. The system shall have a rating system that allows login users to rate the photos on the map.



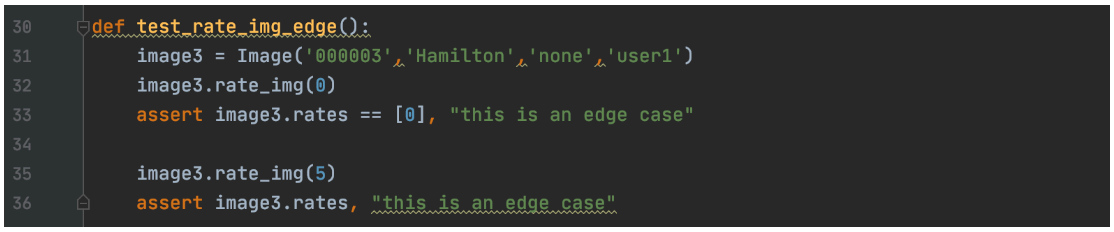
```
from image import Image
def test_rate_img_regular():
    image1 = Image('000001','Hamilton','none','user1')
    UserRate1 = 4
    UserRate2 = 3
    UserRate3 = 1
    UserRate4 = 4
    UserRate5 = 5
    image1.rate_img(UserRate1)
    image1.rate_img(UserRate2)
    image1.rate_img(UserRate3)
    image1.rate_img(UserRate4)
    image1.rate_img(UserRate5)
    assert image1.rates == [4,3,1,4,5],"these are regular rating scores"
```

To test the performance in rating function, we have the first test of regular cases, which lists all the possible rates. These numbers are within the range of 0~5, the test is expected to be passed.



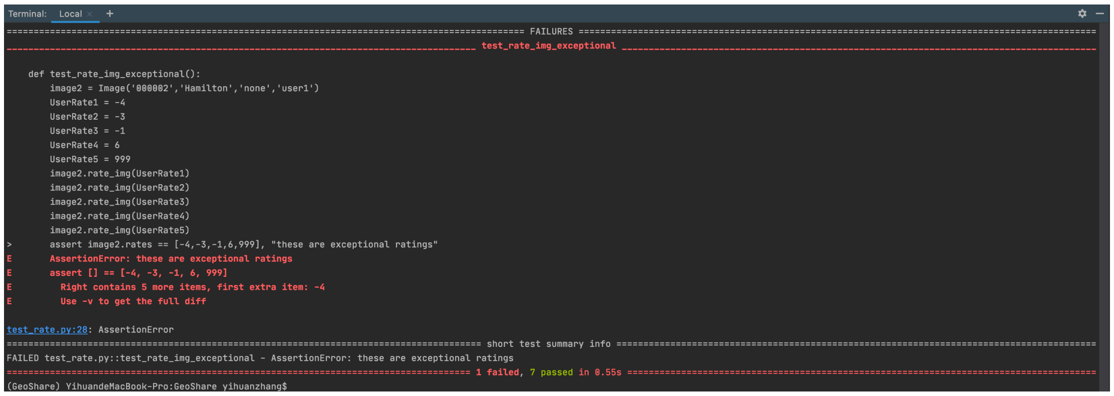
```
def test_rate_img_exceptional():
    image2 = Image('000002','Hamilton','none','user1')
    UserRate1 = -4
    UserRate2 = -3
    UserRate3 = -1
    UserRate4 = 6
    UserRate5 = 999
    image2.rate_img(UserRate1)
    image2.rate_img(UserRate2)
    image2.rate_img(UserRate3)
    image2.rate_img(UserRate4)
    image2.rate_img(UserRate5)
    assert image2.rates == [-4,-3,-1,6,999], "these are exceptional ratings"
```

The second test contains all illegal parameters (negative numbers, numbers greater than 5) are exceptional cases, which is expected to be failed.



```
def test_rate_img_edge():
    image3 = Image('000003','Hamilton','none','user1')
    image3.rate_img(0)
    assert image3.rates == [0], "this is an edge case"

    image3.rate_img(5)
    assert image3.rates, "this is an edge case"
```

The last test for this function is edge cases, which is 0 and 5. These two numbers are expected to be passed.



```
================================================================================ FAILURES ================================================================================
_______________________________________________________________________ test_rate_img_exceptional _______________________________________________________________________

    def test_rate_img_exceptional():
        image2 = Image('000002','Hamilton','none','user1')
        UserRate1 = -4
        UserRate2 = -3
        UserRate3 = -1
        UserRate4 = 6
        UserRate5 = 999
        image2.rate_img(UserRate1)
        image2.rate_img(UserRate2)
        image2.rate_img(UserRate3)
        image2.rate_img(UserRate4)
        image2.rate_img(UserRate5)
>       assert image2.rates == [-4,-3,-1,6,999], "these are exceptional ratings"
E       AssertionError: these are exceptional ratings
E       assert [] == [-4, -3, -1, 6, 999]
E         Right contains 5 more items, first extra item: -4
E         Use -v to get the full diff

test_rate.py:28: AssertionError
========================================================================== short test summary info ==========================================================================
FAILED test_rate.py::test_rate_img_exceptional - AssertionError: these are exceptional ratings
========================================================================= 1 failed, 7 passed in 0.55s =========================================================================
(GeoShare) YihuandeMacBook-Pro:GeoShare yihuanzhang$
```

The results of the “test_rate”, only the second test is failed.