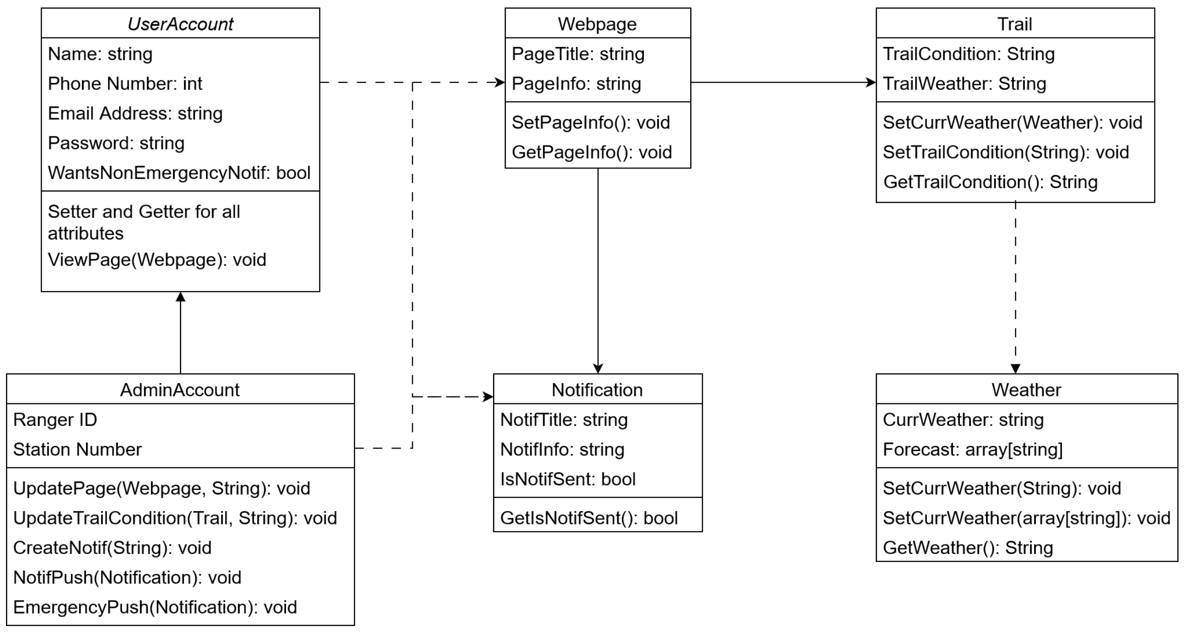
Kili Trekker System Test Plan

Prepared by: Matteo Gristina
March 20, 2023

Table Of Contents

1 Software Design Specification

1.1 UML Diagram



1.2 Diagram and Class Relationship Explanation

The Kili Trekker System is made up of a strong core of classes, the account classes, the webpage classes and the weather class. There are two account classes, one for normal users, which includes basic identification information, and the admin account. AdminAccount inherits basic information from the UserAccount class, but includes functions that are only accessible to admins, which help them update and maintain information in the system. The AdminAccount class has an “is a” relationship with the UserAccount class. Then, the webpage classes include the base webpage class, the notification class, and the trail class. The Webpage class is very barebones, but gets elaborated/more specialized with the Notification and Trail classes. The notification is a special type of webpage, shown to the user in special circumstances, and sent by an administrator. To synergize with this behavior, a bool is added to show the status of the notification. As for the trail class, two important characteristics are added on top of inherited attributes. The weather, like sunny, cloudy, raining, etc. and the condition of the trail, good, fine, bad, flooded, closed, etc. The Notification and Trail classes have an “is a” relationship with the Webpage class. The Trail class has a “uses” relationship with the Weather class in order to update the weather status, and display accurate information.

1.3 Attribute + Function Explanation

This section aims to clear up some non-obvious attributes and functions in the UML Diagram. First is the ViewPage(Webpage) function in the UserAccount class. The ViewPage(Webpage) function will show the user the webpage passed as an argument. This function will make a level of abstraction between the user and the system. In the AdminAccount class, there are a number of important functions, UpdatePage, UpdateTrailCondition, CrateNotif, NotifPush, and EmergencyPush. The admin account has more permissions in the system, which explains the expanded number of functions. UpdatePage(Webpage, String) updates the information section of the passed webpage. UpdateTrailCondition(Trail, String) allows a ranger/admin to update the condition of a trail in case its condition changes. CreateNotif passes a string which gets set as the notification’s NotifTItle. NotifPush and EmergencyPush both pass a Notification object, the difference between the two functions is one pushes the Notification as an emergency, reserved for notifications that need to be seen urgently by users and contain very important information. The Weather class has an overloaded SetCurrWeather function. One where you can pass a String, and another where you can pass an array at a certain index. These end up both passing strings, but it is overloaded to explain the intent behind the Forecast attribute. Lastly, in the Trail class, the SetCurrWeather function utilizes the Weather class as an object to manipulate the TrailWeather attribute in the Trail class.

2 Verification Test Plan

2.1 Unit Tests

2.1.1 UserAccount: WantsNonEmergencyNotif

Test Set 1: A part of the Kili Trekker system is to send notifications to users. To help admins and rangers, there are two types of notifications, emergency and non-emergency. Some users may not prefer to have non-notifications active, so a feature of the system is to let users toggle their preference. Test Set 1 tests this feature, manipulating the WantsNonEmergencyNotif attribute within the UserAccount class.

```
#Test Case 1#
input:
new UserAccount user1
user1.SetWantsNonEmergencyNotif(true)

expected output:
user1.GetWantsNonEmergencyNotif()
with return value "true"

#Test Case 2#
input:
new UserAccount user1
user1.SetWantsNonEmergencyNotif(false)

expected output:
user1.GetWantsNonEmergencyNotif()
with return value "false"
```

These test cases cover the “non-emergency notification preference” feature fully because there are only two possible things that the bool WantsNonEmergencyNotif can be. No matter what the value of the bool is before SetWantsNonEmergencyNotif is run, the output is expected to be the bool passed by the set method. If the output gained from testing does not match the expected outputs in the test cases, then the feature has been implemented incorrectly, and we know that the error lies somewhere in the UserAccount class, because no other classes are involved in the test.

2.1.1 Webpage: PageInfo

Test Set 2: A main part of the Kili Trekker System is the Webpage structure, editing page info is important. You pass a string as the arg through SetPageInfo

at least 3 test cases:

1 null
2 a single character
3 fancy characters or something (non unicode or ascii characters)

this fully covers the targeted feature because we test edge cases for a string

*manipulated data is PageInfo in webpage class

2.2 Integration Tests

2.2.1 Trail: TrailCondition

Test Set 3: A part of the Kili Trekker system is to let admins write and publish information through the system for the user to view. To help admins and rangers, the AdminAccount class has functions directly intended for that purpose. The UpdateTrailCondition function passes a Trail object and a string to manipulate the TrailCondition attribute of the passed Trail object. This lets an admin or ranger manually update trail conditions without needing to directly change exact attributes in all of the trail objects.

```
#Test Case 1#
input:
new AdminAccount admin1
new Website testTrail
admin1.UpdateTrailCondition(testTrail, “closed”)

expected output:
ERROR with explanation: “Website object is not of type
Trail”

#Test Case 2#
input:
new AdminAccount admin1
new Notification testTrail
admin1.UpdateTrailCondition(testTrail, “closed”)

expected output:
ERROR with explanation: “Notification object is not of
type Trail”
```

```
#Test Case 3#
input:
new AdminAccount admin1
new Trail testTrail
admin1.UpdateTrailCondition(testTrail, "closed")

expected output:
testTrail.GetTrailCondition()
with return value "closed"

#Test Case 4#
input:
new AdminAccount admin1
new Trail testTrail
admin1.UpdateTrailCondition(testTrail, null)

expected output:
testTrail.GetTrailCondition()
with return value null

#Test Case 5#
input:
new AdminAccount admin1
new Trail testTrail
admin1.UpdateTrailCondition(testTrail, "c")

expected output:
testTrail.GetTrailCondition()
with return value "c" (string)
```

These test cases cover the “set trail condition” feature fully because there is only one valid Website class that has the attribute TrailCondition, the Trail class. Website is the parent class, and Notification is a child of Website, but both do not include the correct attribute. In addition, the string portion of the input is tested with values of null, c, and closed, which are cases very near to the partition boundaries of the String datatype. If the output gained from testing does not match the expected outputs in the test cases, then the feature has been implemented incorrectly, and we have located an error that has resulted in a failure.

2.2.2 Trail: CurrWeather

Test Set 4: A part of the Kili Trekker system is to have the system automatically source and update its weather information. The trail class has the SetCurrWeather which passes a weather object. The weather object has a CurrWeather attribute, which is passed into the method and set to the Trail's CurrWeather attribute. (need to change diagram so that weather class has "autosetweather", etc)

*manipulated data is TrailWeather in trail class

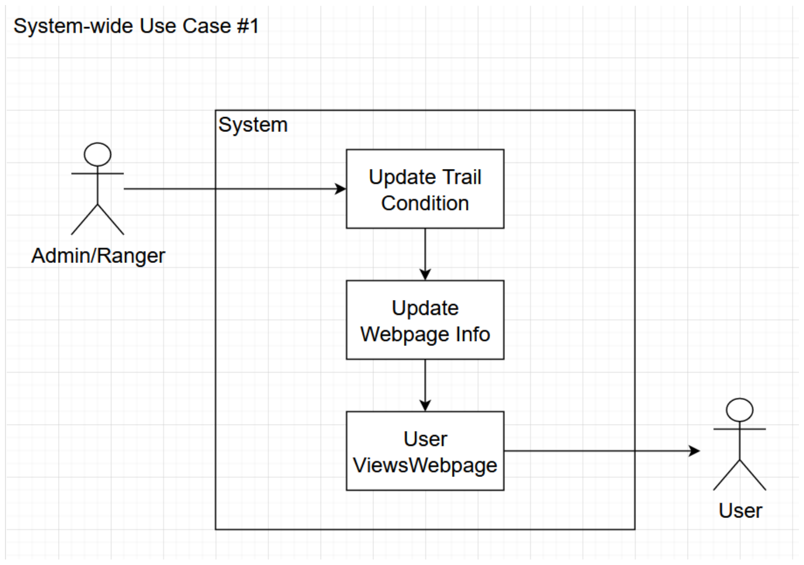
1 test case for NON weather object (wrong class, should fail)
1 test case for weather object (success)

this covers the targeted feature fully because it ensures that the class needs to be the appropriate child class of webpage, and that the string input is valid

2.3 System Tests

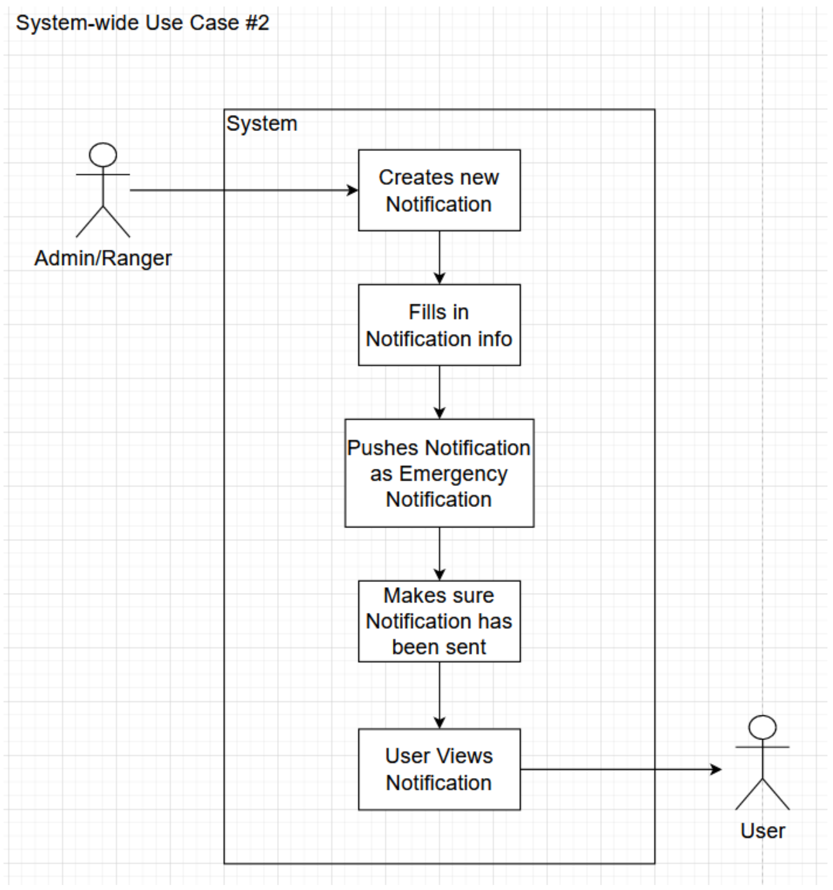
2.3.1 System Test 1: Trail Condition

System Use Case 1: Admin Updates Trail Condition -> Admin updates webpage info saying that the trail is now closed or something -> user uses viewwebpage to see new info



2.3.2 System Test 2: Emergency Notification

System Use Case 2: Admin Creates new Notification -> Admin populates notification with information (title, information) -> admin pushes notification as emergency -> makes sure emergency has been sent (isNotifSent) -> user views notif (viewwebpage, notif is child class of webpage)


System-wide Use Case #2
System
Admin/Ranger
Creates new Notification
Fills in Notification info
Pushes Notification as Emergency Notification
Makes sure Notification has been sent
User Views Notification
User