Killi Trecker System

Design Specification

Yousif, Kelly, David

System Description

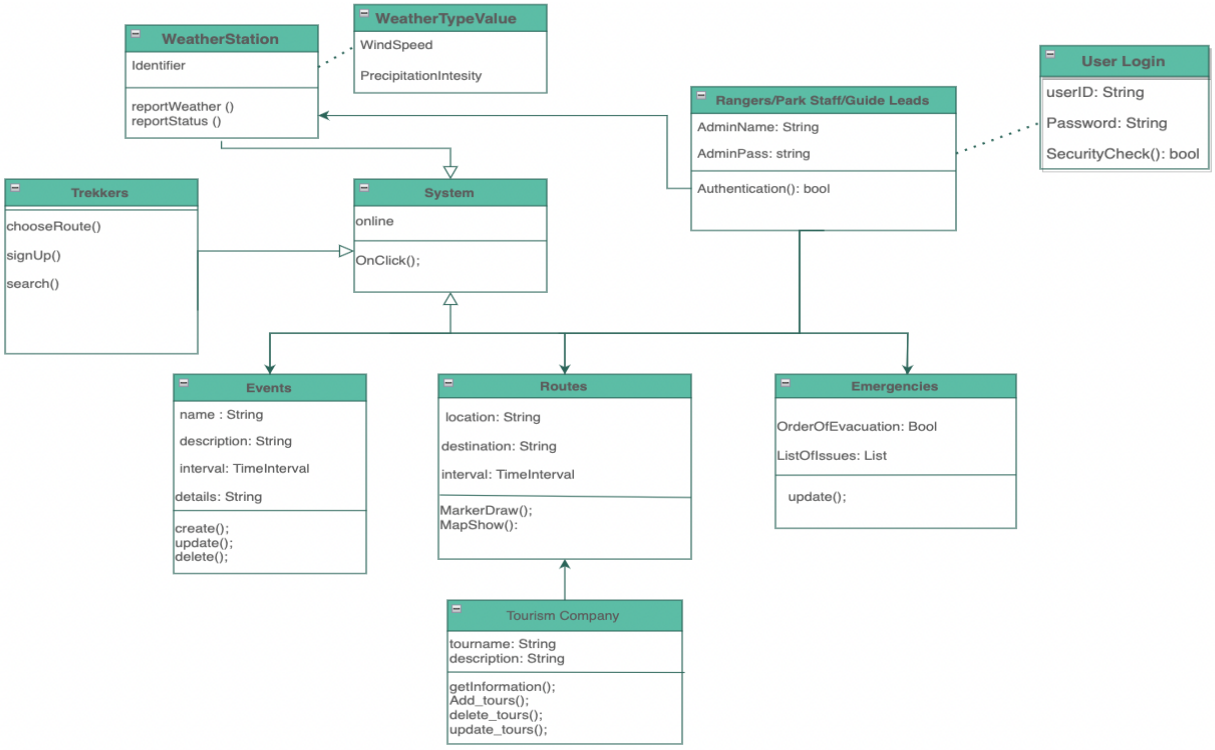
The Kilimanjaro national parks would like to make a system to allow tourists in the park to get info that they need. The software system must provide means of representing and accessing data transferred between servers. The system will be able to evaluate weather conditions with a camera that is mounted on top of the mountain. The park goer's will be able to get the information by going to the park's official website on their computers or phones. The most important design goal for the site is that it's easily usable by everyone who needs information in the park.

UML Class Diagram explanation:

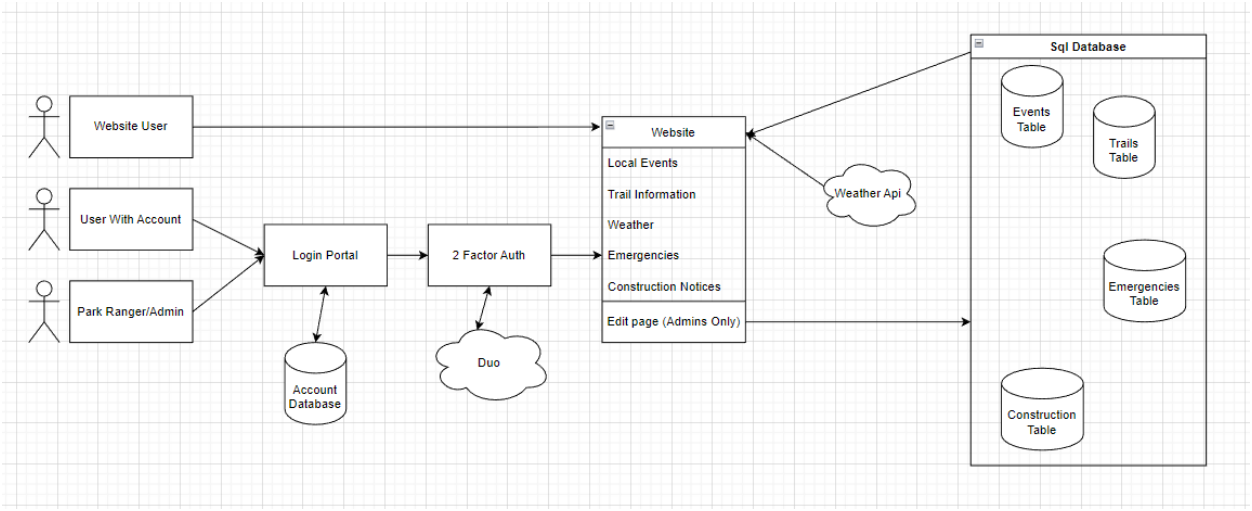
- **User Login**
    - This is where the string of the userID will be stored along with the password in a database that also contains a security check regarding the individual identification
- **Ranger/Park Staff/Guide Leads**
    - This will be admin(Ranger) who is able to access the system/hub which displays everything for tourists, staff, and guides. The system will require him to specifically sign in with his provided username and password with authentication to his identity as an admin.
- **[Emergencies, Routes, Events, Weather Conditions]**
    - **The Ranger** will have the ability to **update, modify, delete, and display the information on the system.**
    - **Emergencies:** The emergency will have a bool value of true or false which will be displayed along with the emergency name and will be updated in case there is an emergency like extreme weather conditions.
    - **Routes:** All the routes in the park will be displayed with their respective location,, which links to google maps,, name, and time interval which will represent how long the tour will run.
    - **Events:** In case of an event being created or updated by the **admin(Ranger)** it will immediately display on the system for tourists/staff/guides to see the event's name, description of the event, and the time the event will run for.
    - **Weather Conditions:** The weather will be monitored by the camera. The camera will watch the park trials condition and weather all the time. It will notify the system for any shift in the weather. **The Ranger** then will be able to change the weather condition name which can range anything from "Good weather" or "Bad weather" etc. It will also keep track of the weather's temperature. It will always be on the system just in case the tourist or staff wants to view it.

- **WeatherStation/WeatherTypeValue**
    - It is providing a collection of instruments and equipment to observe atmospheric conditions and provide information to make weather forecasts. It will notify the system of any shift in the weather. The admin will be able to change the status of the conditions according to the given reports. Will also be accesible for tourists and trekkers for future hiking arrangments.
    - Commonly measured environmental variables include light, temperature, relative humidity, rain, and wind. Specifically intensity of precipaion and wind speed.
- **Trekkers**
    - Will have access to weather conditions and the ability to access information about authorized guide companies and sign up for tours. Be able to search for information about significant events within the Kilimanjaro area to help plan trips in the park.
- **Tourism Company**
    - Guide companies can update, modify, delete, and provide information about themselves and the 12 Kilimanjaro trails. Companies will also have access to relevant events and weather conditions to plan and lead hikes.
- **System**
    - It is a basic interactive system capable of a single user on a computer, where rangers and guides have full capabilities at all ranger stations, providing the ability to directly update the information maintained by this system through web-enabled cell phones. The system is accessible at any of the park's ranger facilities and links to the camera at the top of the mountain showing the current weather and trail conditions. Designed to be accessible on the web to people planning trips to Kilimanjaro via Tanzania.

UML Class Diagram:



Architecture Diagram:



Data Management Strategy:
We decided to use two different databases to store the data for the website. The first database holds the login information for the users and admins. We decided to keep this as a separate database so that extra security precautions could be put in place. On the other hand we have the Information database which does not hold sensitive data. The information database is made

up of multiple tables which store information for different pages on the website. We thought it would be ok to store all of the information in one sql database because this didn't have as high security requirements. We of course could have also decided to merge the two databases which exist now, however we thought keeping the data separate would make it easier to keep the login information secure. A sql database with multiple tables allows for easy storage and retrieval of whatever information is needed for the page the user is looking at.

Tasks:

- Backend Development:
    - Server - Kelly
    - Testing - Yousif
- Frontend Development:
    - UI Design - David
        - Time: 2 months
    - CSS/HTML - David
        - Time: 3 months
    - Javascript - Kelly
        - Time: 3 months
    - Testing - Yousif
        - Time: 2 months

Testing:
Unit
<

Functional
System

Unit Test	Integration	Software System

```
/**
Events event;
void createShow() {
  name = "Great Tremor";
  if (name == event.name) {
     return Pass;
  } else
  {
     return Fail;
  }
}
*/
```

—----------

```
/**
Routes route;
void createMarker() {
  route.markerDraw((0, 0) ,
“testMarker”);
  For x in (route.mapShow().markers) {
       If (x == “testMarker) {
   return Pass;
  } else {
    return Fail;
  }
}
*/
```

```
/**
Ranger adminName
if (adminName.name ==
true) {
  You create an object of the
class Events;
  Events event = new
Event();
  event.name = "Great
Tremor";
  if(event.name == "Great
Tremor") {
     return Fail;
  }
  else
     return Fail;
}
return false;
*/
```

—----------

```
/**
Routes route;
void createTour() {

route.Add_tour(“TestName”)
;
  if
(route.getInformation().name
== “TestName”) {
     return Pass;
  } else
  {
     return Fail;
  }
}
*/
```

- The creation of an event will depend on the ranger(admin)
    - It will check whether the admin who is accessing the data is true or false if true proceed, else return false
- If Admin then you go and create the object of Events and name it.
    - Once this is done you will be inserting the event into the interface and then if its valid it will pass else returns nothing.
    - Then it will create the events and then the admin can update or modify the event to his liking.

—----------

If you are unable to complete any of these steps the test is a failure

1. Log onto the home page of the site
2. In the top right select register
3. Use the credentials TestUser and TestPass in the username and password spaces and select the register button
4. Now you should be back at the home page
5. Select log out at the top right
6. Select login on the top right
7. Type in the credentials TestUser and TestPass into the username and password spaces and press login
8. You should be on the home page and be logged into the same account.

Explanation:

1. **Unit Test 1 & 2:**
    a. The event class will create an object in which will hold the variable event. That event will connect to the function createShow(). If the event name will equal to the event that is being created then it will create the show.
    b. This unit test will display the map which is marked
2. **Integration 1 & 2:**
    a. We make a variable off the Ranger's class called adminName and then we check whether the admin name is authentic. If not authentic the condition below will not execute. If it's true we will move on to check the events name and if the eventName is equals to the event that is being added then it will create the event else it will fail to do so.
    b.