Software Design Specification - Kili Trekker System

Prepared by Erica Lee, Jennifer Giovengo, and Bryce Garrod

March 11, 2023

System Description

The Kili Trekker Tracker System is a sophisticated software developed for the Tanzanian government to enhance tourism in the breathtaking Mount Kilimanjaro region on the Tanzanian-Kenyan border in Africa. Designed primarily for trekkers visiting the Kilimanjaro National Park, this system offers many features that allow hikers to access vital information and track their location while exploring the area. As well as providing real-time information about the twelve trails on Mount Kilimanjaro, the software also helps trekkers organize authorized trail guide companies to accompany groups to the summit with various paths, and guides those who signed up with guides to hike around the area. In addition to these features, the Kili Trekker Tracker System provides an extensive range of services to users, including tips, information, and disclaimers for those taking the solo route. Local events around the area, weather forecasts, and trail conditions can also be accessed. The weather tab even features a camera at the top of Mount Kilimanjaro to offer a real-time view of the summit and trail conditions. In case of emergencies, the software promptly displays notices about mandatory evacuations, warnings, or threats due to Kenyan rebels, ensuring that trekkers are kept informed at all times. Authorized personnel such as park officers, park rangers, guides, and IT technicians can have full access to the system via two-factor authentication to perform various functions, such as updating information, maps, trails, and available times for guides.

Software Architecture Overview

Architectural diagram of all major components

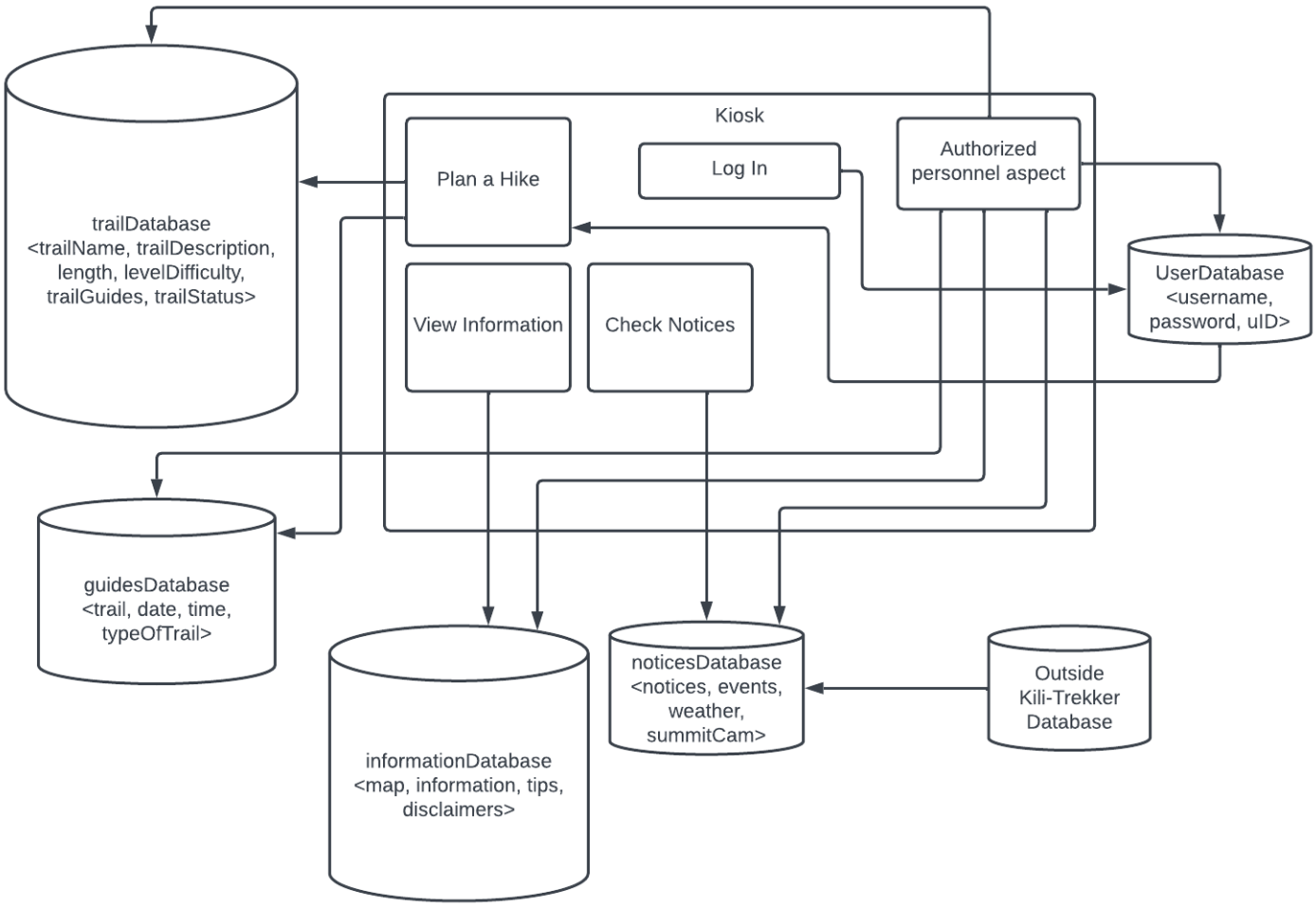

Figure 1: Software Architecture Diagram for the Kili Trekker Tracking System

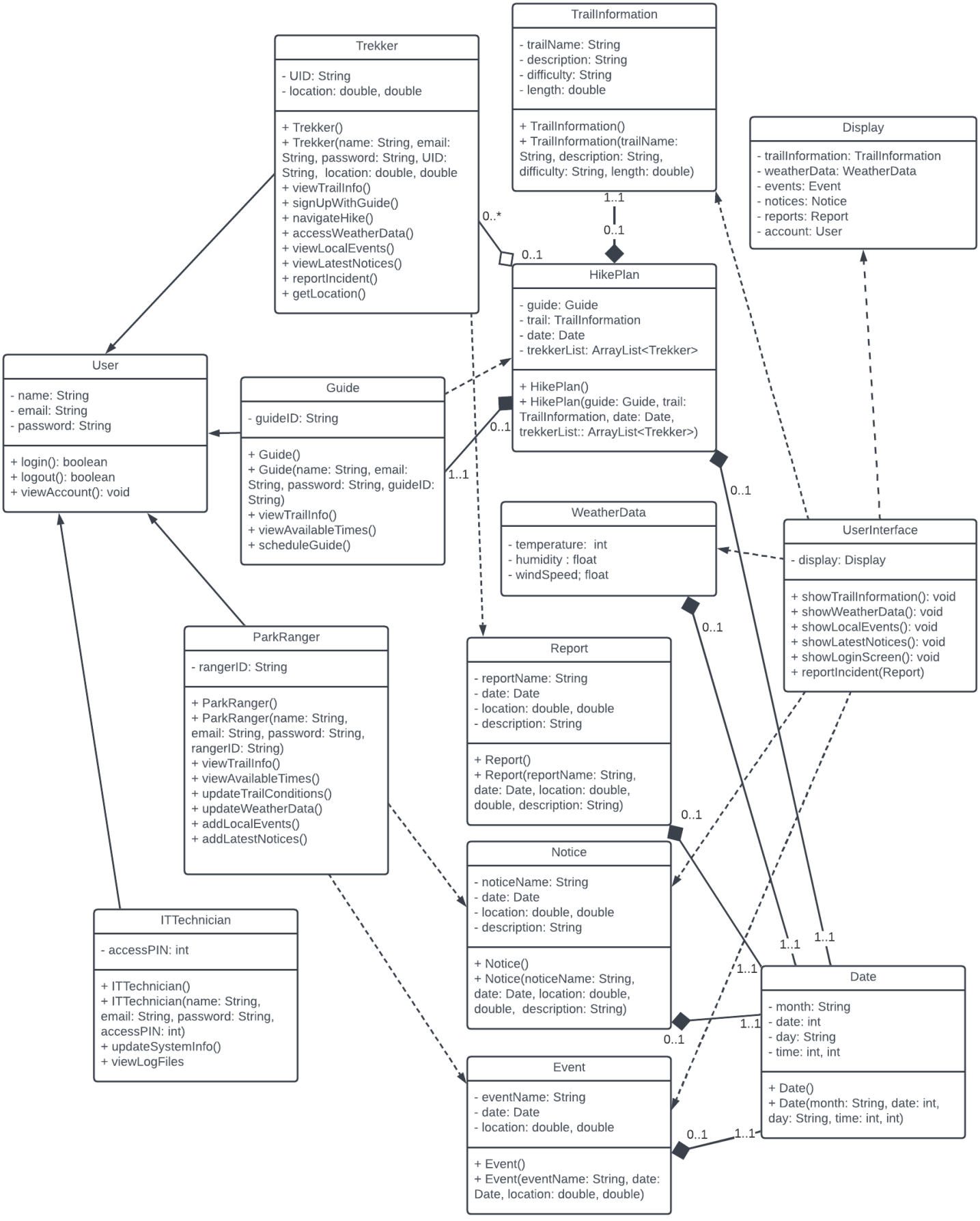

Figure 2: UML Diagram for the Kili Trekker Tracking System

UML Class Diagram

Software Architecture Diagram Description

(Figure 1.) Any authorized personnel has the ability to access the user database to manage and access user information. This could be for recovering user accounts that have been lost by the user or to fix something about a user's account that needs to be altered. Authorized personnel can also get access to and modify any information regarding trails, available guides, notices, and general information. Alteration of these databases could be necessary if a trail has been destroyed or is inaccessible, a guide quits working at the mountain or a new guide is hired, a new notice needs to be made for an event or a danger, or adding any piece of general information that is needed by trekkers or deleting obsolete general information. As a user, there are three main types of functions that a user would want to access: planning and making reservations for a hike, checking current park news, and viewing general information about the park. To access planning a hike, the user must log in and the information from the user database will be used to verify the login is valid. When planning a hike, the trail that will be taken and the guide who will be with the group are needed, so the plan a hike function is connected to both the trail database and the guide database. For checking notices, all notices, events, and weather news are in the same database, and the check notices function is connected to the notices database. The way the database receives information is either by an authorized person manually altering something like creating a new event or by sourcing it from an outside database, like updating the weather from a weather station. Finally, the view information function accesses the information database, which consists of general park information that is sometimes updated by authorized personnel.

Description of classes, attributes, operations

User class
The User class manages all users in the system including trekkers, guides, park rangers, and IT technicians. These listed users share a generalization relationship with the User class. Not all trekkers need to have an account, however, an account needs to be used for those who wish to sign up for a guided hike. Attributes of this class include name, email, and password which are then inherited by the four users. Using these attributes, users can perform actions such as logging in, logging out, and viewing their account details under viewAccount(). These operations act as getter functions.

Trekker class
The Trekker class encapsulates the actions and functions that a trekker can do. Even though the trekker may or may not have an account, the class allows them to view trail information, read the information on how to navigate their hike, and access weather data. Attributes of this class include user identification (UID) as a String value of a combination of letters and numbers as well as live tracking of their current location, stored as doubles for latitude and longitude, as long as they are logged on their mobile device in the Kilimanjaro vicinity. For all four users, we implemented their respective constructors, in this case, we have the Trekker() constructor which is an instance of the Trekker class that creates an object, Trekker. Trekker also inherits information from the User class, including the name, email, and password. The functionality of the Trekker class includes being able to view trail information, navigate hikes, access weather data, view local events, view the latest notices, and report incidents. However, to sign up with a guide, the trekker would need to have an account under the User class to book the time slot.

Guide class
The Guide class inherited from the user class This class, inheriting attributes of the User class, includes all the functions that a guide leader can do. As mentioned with the Trekker, we have the Guide() constructor, an instance of the Guide class that creates an object, Guide. Because the trail guide leaders refer to the trails made available to be scheduled with, the Guide class depends on the TrailInformation class. An additional attribute of the Guide class that wasn't inherited from the User class is the guide leader's Guide ID, a String value of a combination of letters and numbers, which they use to log into the database to access trail info, available times, and the guide schedule. Guide leaders can perform functions such as viewing trail information, and available times that they can schedule a guide with the tourists. They can also schedule appointment slots for the company-led guides themselves based on the 12 trails that Kilimanjaro's National Park has.

ParkRanger class
The ParkRanger class– like the Trekker and Guide class– takes in the inherited attributes of User: name, email, and password, along with its own attribute: rangerID used for authentication. An instance of ParkRanger is also used as an Object. The Park Ranger class consists of all the actions and functions a park ranger can do: viewing trail information, viewing available times, updating trail conditions, updating weather data, adding local events, and adding the latest notices. In terms of hierarchy, the park ranger performs most of the updating of information, while the Trekker and Guide classes have limited operations. It also is dependent on the latestNotices class and localEvents class.

ITTechnician class.
Like the other three inherited classes from the User class, the ITTechnician also acts the same where a constructor of the

ITTechnician class is created as an instance. The ITTechnician class inherits the User class's attributes: name, email, and password, and includes its own attribute: accessPIN which is used for updating the system information and for viewing log files to access the User database.

TrailInformation class
The TrailInformation class manages all the details that make up each of Kilimanjaro's 12 hiking trails. This includes the trail names, their respective descriptions, the level of difficulty of each trail, and a double data type describing the length of each trail (in kilometers). There is an instance of the TrailInformation class of type TrailInformation that acts as an object and sets values for any existing object properties like the name, description, difficulty, and length. The HikePlan is dependent on the TrailInformation because, without the trail attributes, there is no information on anything to hike on.

HikePlan class
This class encapsulates all the information needed for each planned hike. Each HikePlan has a Guide, TrailInformation, and a date associated with it. Attributes include the guide as an instance of Guide, the trail as an instance of TrailInformation, a date as an instance of Date, and the ArrayList of trekkers (the Trekker objects) signed up for the hike. Operations include HikePlan() as an empty parameter constructor, HikePlan(guide: Guide, trail: TrailInformation, date: Date, trekkerList: ArrayList<Trekker>) WeatherData class WeatherData class provides temperature information, humidity, and wind speed. UserInterface relies on the WeatherData class to display information. WeatherData is extracted from an outside database that provides the class with values for variables: temperature (int), humidity (float), and windSpeed (float).

Report class
The Report class consists of the elements that make up a report, used if a trekker wants to report an incident. The report name and the user-provided description of the report are two attributes using the String data type. The location is provided as two doubles to represent latitude and longitude. The Report class is associated with the Date class meaning it accesses the Date class to use the current month, date, day, and time for the reported incident.

Notice class
The Notice class has the attributes: noticeName, date, location, and description. The LatestNotices class is used by the user-Interface class meaning that LatestNotices is associated with. It is also associated with the ParkRanger class. LatestNotices provides a string noticeName, along with a dependency on the Date class. This class also provides a location.

Event class
The Event class includes the components that make up an event announcement, including the attributes, eventName (data type: String), date (attribute of type Date that holds month, date, day, and time), and location (data type double, double (representing the latitude and longitude). Event() is a constructor operation that creates an object of the Event class without any parameters. Event (eventName: String, date: Date, location: double, double) is instantiated as an object with parameters: eventName, date, and location.

Date class
The Date class includes the month (String), date (int), day of the week (String), and time (int, int). Other classes such as HikePlan, WeatherData, Report, Notice, and Event compose the Date class for their date attribute. Date also has its own Object instantiated which takes in the parameters of month, date, day, and time.

Display class
The Display class has the UserInterface class dependent on it. Attributes include trailInformation with the data type, TrailInformation; weatherData with the data type, WeatherData; events with the dataType, Event; notices with the data type, Notice; reports with the datatype, Report, and account with the data type, User. These all include the instantiated objects from all their respective classes, which allow them to access the parameters to display the actions under UserInterface.

UserInterface class
The UserInterface class uses the Display class to show information given the specific method. Attributes are showTrailInformation, showWeatherData, showLocalEvents, showLatestNotices, showLoginScreen, which are all getter functions, which return void. reportIncident is a setter function that takes in the attributes of the Report object.

Development Plan and Timeline

The Kili Trekker Tracking System will have three main phases of development: user types development, user-created objects development, and user interface development, partitioned among team members Jen, Erica, and Bryce, respectively The user types development stage will consist of making the Trekker, Guide, ParkRanger, and ITTechnician classes, the instances of

which will store the account information for a certain kind of user. During the development of these classes, making sure each class has the correct access to the features of the program is important to make sure that each type of account can be used correctly. To make sure that nobody gets access to an account type they should not have access to, the security of the program should be made robust with whatever encryption techniques the cybersecurity person or group sees fit. After the different types of users are created, the objects which they can create and access the information of will be developed. Through the user interface, any class should be able to access the weather data, which will be sourced from a weather API. Trekkers should be able to create reports, and those reports should be able to be viewed by users that are park rangers. Guides should be able to create hike plans and these should be available for all to access. Park rangers should be able to create notices and events, and those notices and events should be available for all to access. Finally, the last phase is to create the user interface and allow each type of user to create the objects they should be able to create and access the objects they should be able to access in an easy-to-understand format.