Kilimanjaro Trekker System

Prepared by: Ella Suchikul, Sawyer Blakenship, Isabelle Bernal

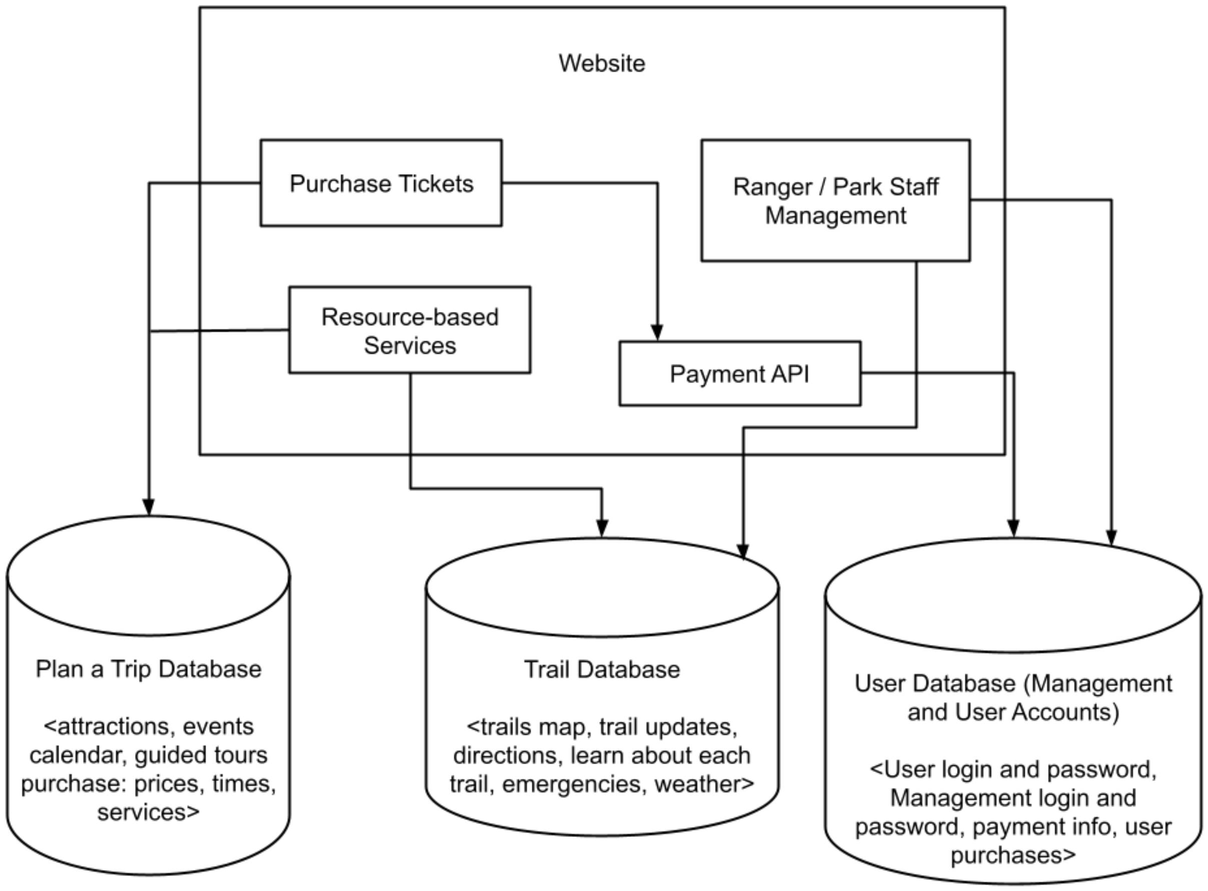
March 2, 2023

1. System Description

The Kilimanjaro Trekker Tracking System serves as an online support system for the Tanzanian tourist business so that the Kilimanjaro National Park is easier to navigate. The Kilimanjaro Trekker Tracking System is made available online through a website and through an interactive display at the park. The Kilimanjaro Trekker Tracking system maintains information about the twelve different trails, provides information on significant events in the area, helps people plan trips to the park, offers guided hikes through the trails, and gives updates in regard to evacuation and safety protocols. This website is essential because it allows not only the park rangers, but the tourists to have quick and easy access to anything they need to know about the park.

2. Software Architecture Overview

Software Architecture Diagram: Kilimanjaro Trekker System


Website
Purchase Tickets
Ranger / Park Staff Management
Resource-based Services
Payment API
Plan a Trip Database
<attractions, events calendar, guided tours purchase: prices, times, services>
Trail Database
<trails map, trail updates, directions, learn about each trail, emergencies, weather>
User Database (Management and User Accounts)
<User login and password, Management login and password, payment info, user purchases>

Software Architecture Diagram Description: The software architecture diagram covers all the major components of the Kilimanjaro Trekker Tracking System. The system is a website that can be accessed and read by any user, and edited by staff only. The major components of the software are resource-based services, purchasing tickets, and ranger/park staff management. The databases involved are planning a trip, trails, and users.

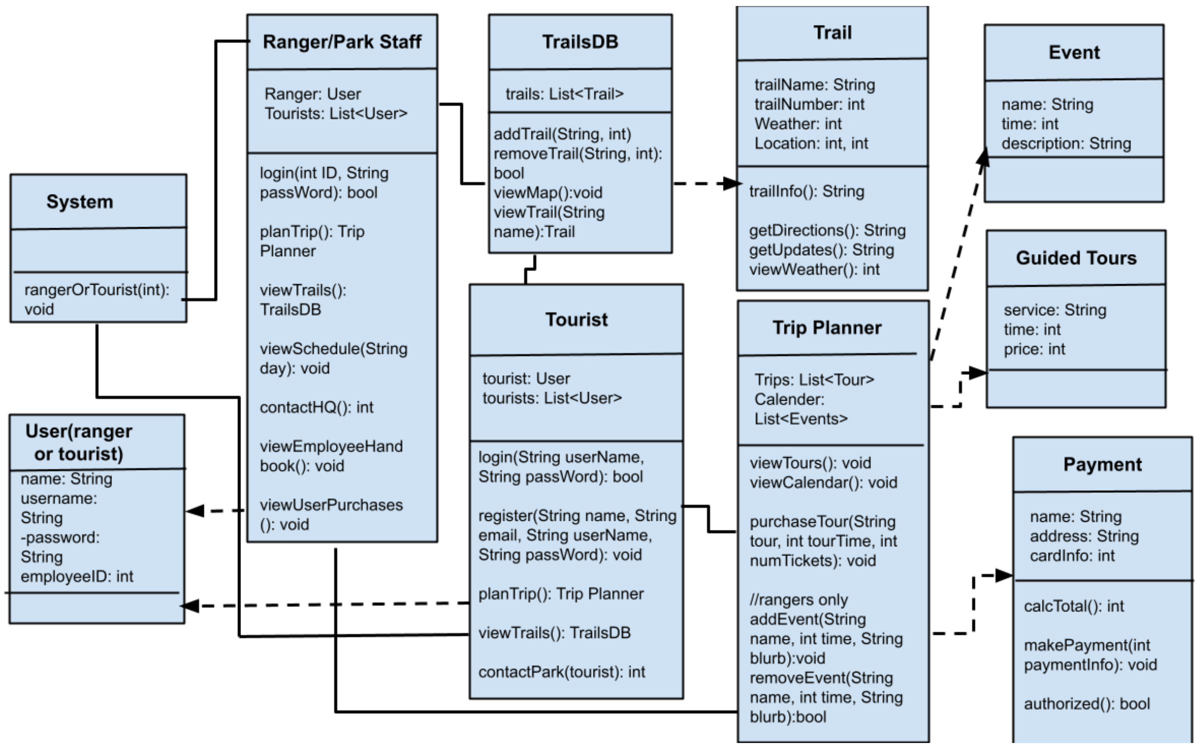
The resource-based services component entails a plan a trip database and a trail database. This is due to the resources and services offered by the software to support the users. The plan a trip database contains attractions, the events calendar, and information on guided tour purchases (prices, times, and services). The trail database contains the trails map, trail updates, directions, information on each trail (length, elevation, estimated timing, etc.), and trail, weather, and emergency updates.

The payment API component inherits the purchase tickets component and entails a user database. This component allows for a smooth payment process for the users and allows the Tanzanian tourist business to collect the money easily and efficiently. The purchase ticket component goes straight to the payment API component if the user wants to pay right away without looking at the plan a trip database. The user database then holds payment information, stores the purchase, and provides account management.

The purchase ticket component entails the plan a trip database and payment API component. This component supports the user with convenient trip information and a convenient payment process. The payment API component inherits the purchase ticket component and allows for all the information in regards to payment to be stored in the userdata base. The trip database contains all the information in regards to attraction, pricings, event calendar, and guided tours.

The ranger/park staff management component entails a user database and a trail database. The component allows for the management of the national park to gain full access to user and trail information. The user database holds management login information, payment information on clients, and client purchase history. The trail database gives management full access to updates and view trails, emergencies, weather, map, and directions.

UML Class Diagram: Kilimanjaro Trekker System



UML Class Diagram Description: The UML class diagram covers all the specific classes, attributes, and operations of the Kilimanjaro Tracking System software. The classes start off with the Kilimanjaro Trekker System login page then gives the option to go into the class of Ranger/Park Staff or Tourist/Trecker to get the best experience possible for the particular user. Within both of those classes there are some similar classes they call to, but with different operations. This is due to rangers/park staff having the ability to write into the system, while tourists and trekkers have only read access. Those classes consist of guided tours, trails, trip planner, payment, and trails which are all necessary to meet the requirements of this particular system. Within the diagram we have two different arrows which are an important part of how the system is able to function. The solid black line symbolizes aggregations/associations (i.e. X has a Y). The dotted arrow line symbolizes dependencies (i.e. X uses Y). It points to a specific class that can store data for functions that have more detailed information such as user's login information or trail specifications.

The Ranger/Park Staff class is needed as a particular login area just for the staff so they can get into the operations within the system that they either need to update or look at. We made the login operation a boolean so that the client knows whether the login was successful or not. We also included operations of updateTrail, viewSchedule, contactHQ, and view Employee handbook so that the requirements needed by the staff are met and they have easy access to each of these operations. Only updateTrail is a boolean return because the staff needs to know if the information that they uploaded was successfully updated or not. The rest of the operations return a string or an int because this is more of information that the client needs to search through. Specific attributes note mentioning are the name, employee id, and password since this is how the client will get into the system. The ranger class also has a list of all of the actual users so it can verify logins and purchases. The ranger and tourist class both use the user class as an object to store information. There is a dotted arrow line representing aggregation, because the ranger class uses the user class to store information. There is also a solid black line that points to the TrailsDB class and the Trip Planner class since these are the two pathways that the employees can choose to go to. We did not choose the dotted line symbol for this arrow because those classes do not store user information; it just provides more information about the specific class topic they chose.

The tourist class is used for the tourists to be able to access the other classes. It stores a list of all of the current users so it can verify logins by pointing the dotted line to the user class that is in charge of storing all the login information. Within the functions in the class it has two methods that will return Trip Planner & Trail DB so the tourists can access the trip planning and trail viewing methods. This is always why there is a black line pointing to Trip Planner and trails DB since this is the class that holds all the information for the tourists to view. The tourist class also has a register method where it creates an account for the user if they do not have one, which returns all the registered information. The last method is named contactPark which is crucial because it returns the number of the park in case there are any issues or questions.

The TrailsDB class is used to give information in regards to what specific trails are currently available and their status. There are methods of addTrail, removeTrail, and viewMap that allows just for the specific trail to be updated by the rangers and for the tourists to be able to view their location. The removeTrail returns a boolean to verify the success of the removal of the trail and viewMap is a void since it has all the information already stored in the class. Now the viewTrail method returns the Trail class which is very important because the Trail class has all the specifics about the trail. For instance, the Trail class has the specific directions, weather, and allows for the park ranger to update anything they need to in regards to the trail. This is also why the TrailDB class points to the Trail class with a dotted line because all the information in regards to the trail is stored into the Trail class.

The Trip planner class is used to provide specific information about the different events/tours that the park has to offer to its visitors/tourists. The methods of viewTours, viewCalendar, and purchaseTours specifically returns a void because this is only used for the tourists to look at and nothing more. Compared to the methods of removeEvent and addEvent where the addEvent does return a void but for the purpose specifically only for the rangers to use so they can update the website.

The Payment class is meant to process payments. Users will be able to pay for guided tours using it. The payment class needs to be able to authorize payment methods with the bank and process payments. It will be accessed by the Trip Planner class's purchaseTour method. The calcTotal method will determine the amount that the tour will cost and return it. The makePayment method will give payment info, and attempt to charge the user for their purchase. The authorized method will return whether or not the payment was successful.

3. Development Plan and Timeline

Development of the Kilimanjaro Trekker System will span over the course of 3 months.

The first task is to create the classes that store data for resource-based services: Trail, Event, Guided Tours, Payment, and User. Task 1 is expected to take three weeks because we are laying the groundwork for the software. These classes are the main requirements of the software we will be developing. Ella will be responsible for implementing the Trail and User class. Isabelle will be responsible for implementing the Event and Guided Tours class. Sawyer will be responsible for implementing the Payment class. We will be collaborating and helping each other as needed. We will also give daily updates and have weekly meetings to discuss our progress and make sure everyone is on track for the deadline.

The second task is to create the two main classes: Ranger Park Staff and Tourist. Task 2 is expected to be the hardest and has a timeline of four weeks. These classes will be pointing to other classes from task 1 that store the data. Hence, there will be a lot of room for error, and testing will take more time. Ella and Isabelle will be responsible for implementing the Ranger/Park Staff class, since park staff has more requirements for the system. Sawyer will be responsible for the Tourist class.

The third task is to create two classes that give important information about the park operations: TripPlanner and TrailsDB. Task 3 is expected to take three weeks because these classes hold a lengthy amount of methods that also call other classes. Therefore, testing time needs to be taken into consideration as well. Sawyer and Ella are in charge of the TripPlaner class since it has more methods to write compared to the TrailsDB class which Isabelle will be in charge of.

The fourth task is to create the System class which will be the very first class. Task 4 is expected to take two weeks because it only holds one method which makes it a very short class to write. Sawyer will be in charge of this class however, Isabelle and Ella need to be present in case of any testing mishaps and errors that may occur since it is the very first class in the

Team member responsibilities that need to be upheld are maintaining constant communication, being helpful to one another, attending meetings that will take place once a week, giving regular updates to each group member at the end of the day, and turning in work on time.