San Diego State University

Software Development Plan For:
Kili Trekker System

Version: 1.0

Members: Aatusa Mehdiyan, Jenny Nguyen, Minh Nguyen, Jose Payan

GROUP 13

CS 250

Professor Bryan Donyanavard

October 15, 2021

System Description:

The Kili Trekker System is designed to help support Tanzanian tourist business in the Kilimanjaro area. It's primary mission is to provide the park an efficient system to maintain, control and display Kilimanjaro National Park's tourist information while ensuring data protection and durability. This document will cover our team's development planning process and the components of the software.

Killi Trekker Software System Overview:

All Users :
The users will need a connection to the internet for accessing the "killi Trecker System Web". All users are capable of "read-only" access to the "Killi Trekker Web" at all times.

Admin :
The admin will have to be classified with an ID and email address in order to log into the system and make changes. The classified Admins consist of Park Rangers , Park staff , and Authorized Tour Guide Companies. Admins will be able to submit changes to the "Killi Trekker API" to update the "Killi Trekker Web".

Trekker :
Trekkers are the General Public based users who have fulfilled the "signed up" process through the "Trekker Guide App" hosted by the Authorized Guide Companies.Onced authorized credentials have been assigned to the "General Public" user after the "Sign-Up" process, the Trekker now has the ability to sign into the "Tour Guide App" and choose any of the "Tour Guide Plans" to participate in.

The Killi Trekker API
Description:
The "Killi Trekker API", is a powerful API controlling the flow of data . It synchronizes data by pushing and pulling data to and from different API's including :
"The Weather API" , "Web Server" , "Database" and the "Tour Planning "Guide API".

The Killi Trekker System is functioning 24 hours a day , 365 days a year. The API is responsible for updating changes from the “Admins” and synching all changes to the “Killi Trekker Web” once submitted.The API also responds to requests made by the “Tour Planning API”.

The Weather API

The “Weather API” retrieves data on the weather. Its key function is to maintain and update weather information about the park and push data to the “Killi Trekker API”. The API pulls data from the internet by precipitation , day , week , and by city.

Web Server

Description:

The web server is responsible for keeping the “Killi Trekker Web” online 24hrs a day 365 days of the year. Its software uses efficient HTTP (Hypertext Transfer Protocol) software and hardware. It controls client requests submitted by the admins to by the “Killi Trekker API”. The server traffic load can exceed over 20,000 users at once

Databases

Descriptions:

The databases are efficient data storage structures, organized by collection of data types. The Killi Trekker API acts as the DBMS administrator pushing and pulling data from other APIs and organizing by field in the databases.

Database fields:

Video Storage, Backup Data , Accounts , storing “Updating & Changes “ as Versions, Cloud Save Changes, Tour Guide Database

Databases hold all the data and information hosted on the “Killi Trekker Web”.

Tour Guide App

Description:

The “Tour Guide App “ is hosted by Authorized Tour Guide Companies for the general public to sign up for their “Tour Guide Plans”. The easy to application allows users to easily sign up and become an official Trekker. The App is powered by the “Tour Planning Guide API” allowing the users to see all available trails by day, weekly, and monthly.The application lets users navigate the Tour Guide Plans as well as access the information associated with each plan.

Tour Planning Guide API

Description:

The “Tour Planning Guide API” creates plans according to the current conditions of the park. All requests are made in order to host safe tour guide plans. The API syncs safe tour guide plans available in the “Tour Guide App”.

Killi Trekker Web

Description:

The “Killi Trekker Web” is the display of all the information gathered by all of the APIs and databases. It has an interactive UI which allows users to easily navigate through all information, features, and emergency contact.

Search option on the web:

The web has a search bar that allows users to search any information displayed on the web. Search history is saved for every user for easy access.

Information on the web:

The information on the web includes Kilimanjaro trails and its current conditions, weather conditions, different facilities, wildlife information, local events , and contact information.

Features on the web:

The “Killi Trekker Web” allows users to navigate an interactive Map, Trail Map, as well as live trail cameras set up around the park.’It has a “Mountain Top View Option” where users can view live feed of the top of the mountain.

Emergency Contact on the web:

The “Killi Trekker Web” has an option to allow users to reach support in emergency situations. The support line will direct users to contact the emergency department in the park and appropriate help.

Development Plan and Timeline:

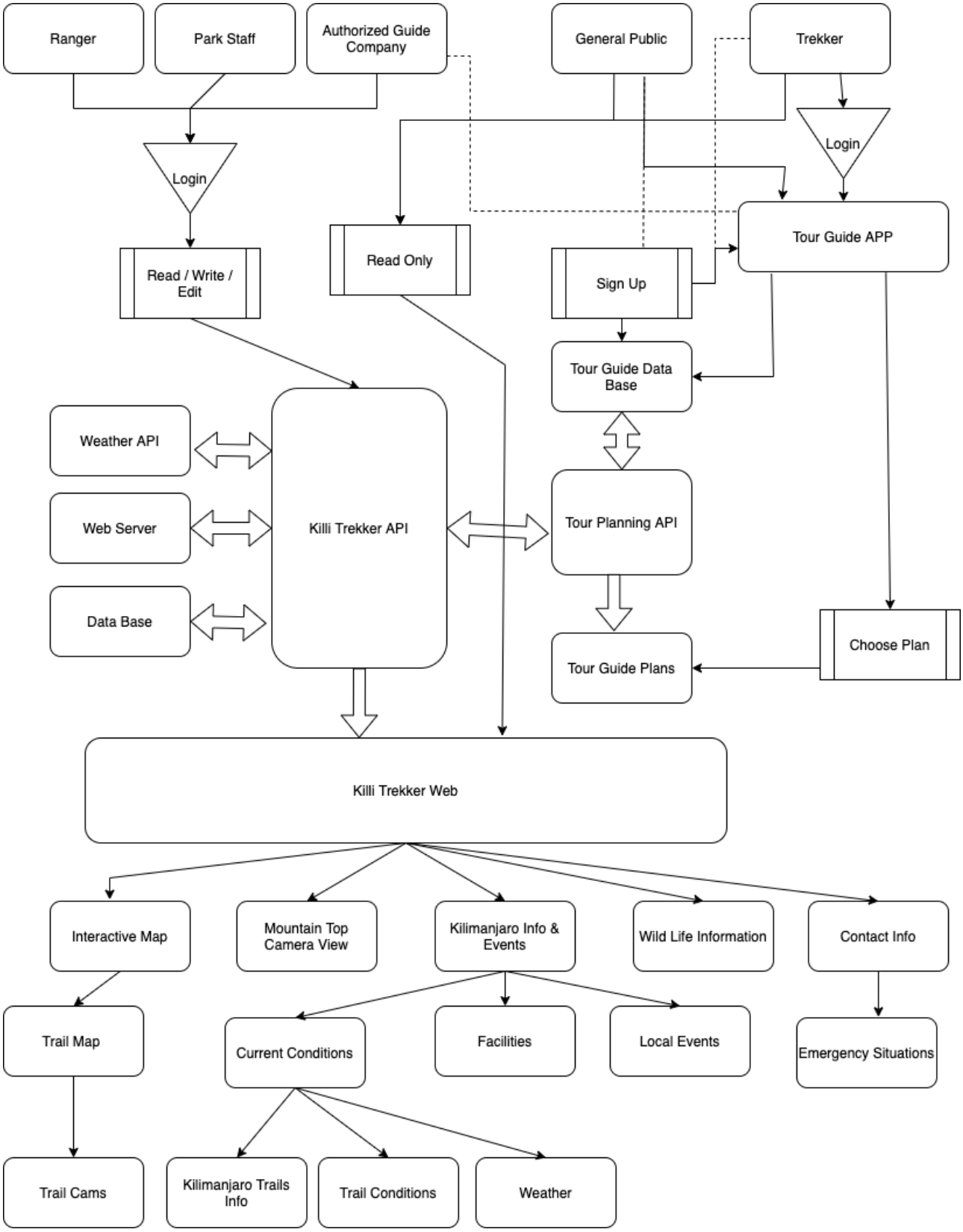
Members	**Roles**	**Timeline**
Aatusa Mehdiyan	Responsible for making Google Doc and development planning and timeline.	2 days
Jenny Nguyen	Responsible for making GitHub repository and UML diagram and operations descriptions	3 days
Minh Nguyen	Responsible for making class descriptions of attributes and operations.	2 days
Jose Payan	Responsible for making software overview and UML diagram.	2 days

Hypothetical Plan & Timeline:

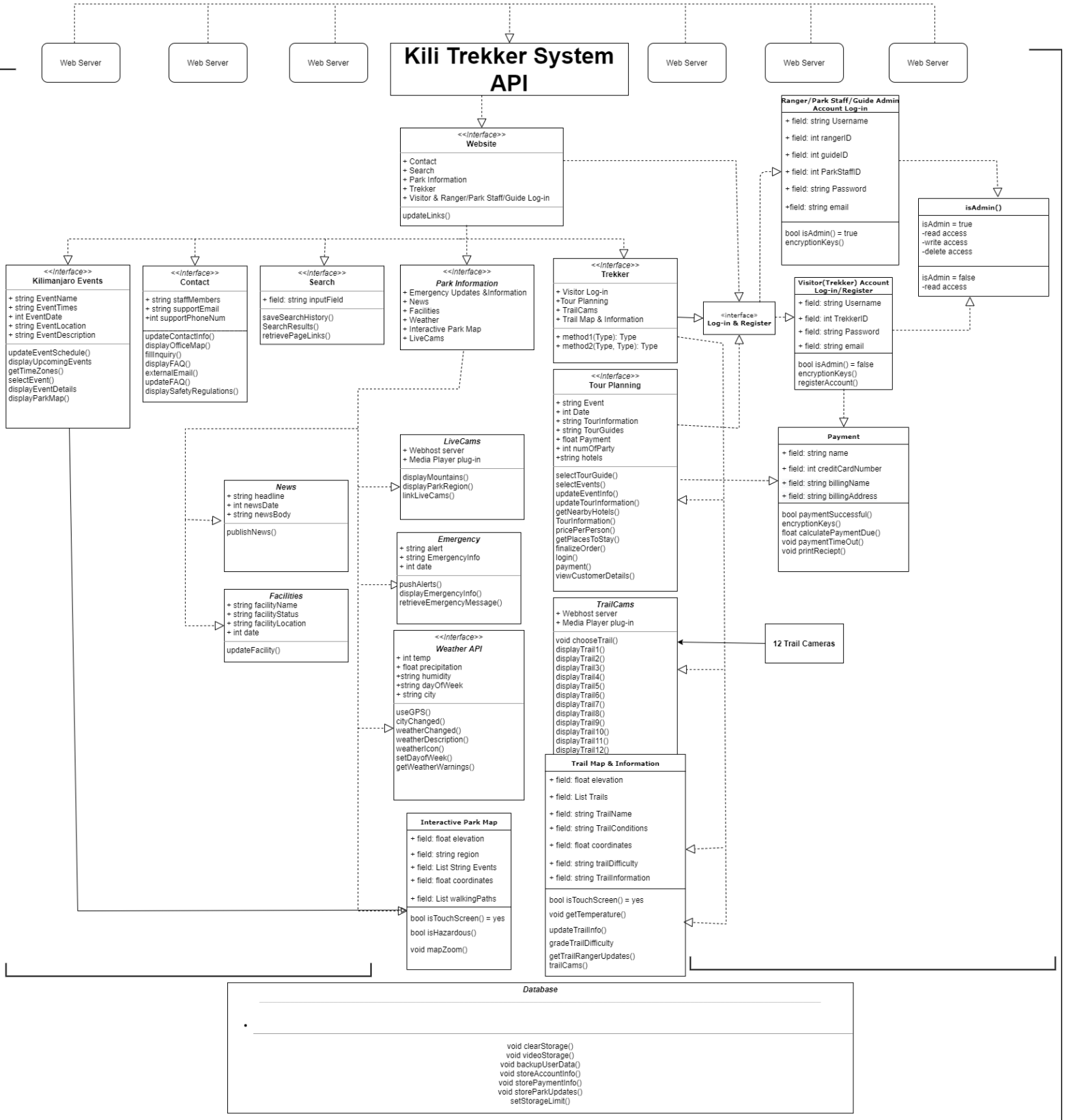
Members	**Roles**	**Timeline**
Aatusa Mehdiyan	Responsible for UML diagram and system design.	1 week
Jenny Nguyen	Class log-in, isAdmin() and payment. Integrating implementing the system.	2 weeks
Minh Nguyen	Website including Class Park Information, Emergency, News, Facilities,Trekker, Tour Planning, TrailCams, Trail Map & Information.	3 weeks
Jose Payan	Website including Class Kilimanjaro Events, Contact, Search.	3 weeks

Software Architecture Overview:

- **Architectural diagram of all major component**



- **UML class diagram**



- **Description of classes, attributes, and operations of Kili Trekker System**

isAdmin: This class is responsible for running, checking and managing secure logins from users who want to access the system. If the output is TRUE, they can have full access as read, write and delete to the system. If the output is FALSE, they only have read access.

➔ Attributes: bool isAdmin || *reads in accounts IDs. Determines whether to give read access, write access, delete access to user, returns true or false*

➔ Operations: bool isAdmin = false || *read access*
bool isAdmin = true || *gives user read, write, and delete access*

Ranger/Park Staff/Guide Admin Account Log-in: This class stores encrypted account information of the system administrators. Each admin will have a different username, ID, password and email used to log into the system. If it is TRUE, the class runs to isAdmin().

➔ Attributes: string Username, int rangerID, int guideID, int ParkStaffID, string Password, string email

➔ Operations: isAdmin() = true
-Void encryptionKeys() || *operation takes in user information details and returns an encrypted hash of data to the database to store*

Visitor(Trekker) Account Log-in: This class stores the encrypted account information of park visitors. This class also holds the function to allow first time users to register an account to the website.

➔ Attributes: string Username, int TrekkerID, string Password, string email

➔ Operations: bool isAdmin() = false
-Void registerAccount() || *registration request method if first time user*
-Void encryptionKeys() || *operation takes in user information details and returns an encrypted hash of data to the database to store*

Payment: This class is used by the TourPlanning class. Sends user to a safe payment processing page and takes in their payment information

➔ Attributes: int creditCardNumber, string name, string billingName, string billingAddress, float paymentDue

➔ Operations: bool paymentSuccessful() || *checks for valid card information and returns true or false*
-Float calculatePaymentDue() || *reads in user's tour planning order, calculates and returns paymentDue*
-Void paymentTimeout() || *goes to timeout page and redirects user to original order page when they do not submit payment within time frame or payment method fails*

-Void encryptionKeys() || *operation takes in user payment details and returns an encrypted hash of data to the database to store payment history*
-void printReceipt() || *prints a receipt page of order and sends customers a copy of their receipt to their email address.*

<<interface>>

Log-in: command class for access between user and system

Website: Upon successful login, it leads to the main information directory which is located entirely within the website to support users**.**

- ➔ Attributes: contact, search, park information, trekker, visitor and ranger/park staff/guide log-in.
- ➔ Operations: Links()

Kilimanjaro Events: It will contain the main information about the events of each trail including the name, time, date and location of the event taking place. It will be updated by Admins and presented on the system in the menu of Interactive Park Map.

- ➔ Attributes: string EventName, int EventTimes, intEventDate, string EventLocation, string EventDescription
- ➔ Operations: updateEventSchedule() || *method for park staff to update and publish event schedule*

-displayUpcomingEvents() || *retrieves updates of events and displays upcoming events to website*
-void getTimeZones() || *gets user time, displays clock to page + countdown to event*
-void selectEvent() || *method for user to choose event and return a link to the event page*
-void displayEventDetails() || *function to print specified event details including event name, time, date and description*
-void displayParkMap() || *displays park's interactive map to event location*

Contact: It contains contact information of staff in the area including email and phone number so that everyone can easily connect and support when needed.

- ➔ Attributes: string staffMembers, string supportEmail, int supportPhoneNum
- ➔ Operations: updateContactInfo() || *The system shall add new contact info*
  -void displayOfficeMap() || *The system shall display the office map*
  -void fillInquiry() ||*method option for user to directly send an inquiry on website page*
  -void displayFAQ() || *displays updated FAQ strings to page*

-void externalEmail() || *directs user to an external email window after clicking on Kilimanjaro Park support email*
-displaySafetyRegulations() || *function for park staff to edit and publish park safety regulations to page*

Search: The function that helps the user to find all the necessary information of the park area that is displayed in the system.

➔ Attributes: string inputField, list searchResults
➔ Operations: void saveSearchHistory() || *takes in user inputs, saves and displays users search history to bottom of search bar*
-void SearchResults() || *takes in user's inputField string, searches database with keywords to search relevant pages*
-void RetrievePageLinks() || *prints list searchResults and gives links to pages relevant from searchResultsMethod()*.

Park Information: This page contains links for the user to view and interact with the website's major features

➔ Attributes: LiveCams, Interactive Park Map, Emergency, News, Facilities, WeatherAPI
➔ Operations: getWeather() || *connects page to weather API to display Kilimanjaro weather conditions*

Emergency: Any emergency situation taking place in the park will be updated and alarms by admins that are displayed in the Park Information directory.

➔ Attributes: string alert, string EmergencyInfo, int date
➔ Operations: pushAlerts || *sends alerts to all rangers, park staff, guides, and park visitors to mobile devices and desktop*
-displayParkRegion() || *prints screenshot of map where emergency is located*
-displayEmergencyInfo() || *prints emergency information to page*
-retrieveEmergencyMessage() || *collects rangers' emergency information and sends it to main server for staff to update to system*

News: This function allows read access for visitors to view latest and previous park news that are published by park staff who are granted write permissions to edit news information.

➔ Attributes: string headline, int newsDate, string newsBody
➔ Operations: PublishNews() || *method to edit, add, and publish city and park news article*

Facilities: This function allows users to know the construction of new facilities or the condition of the park's facilities.

➔ Attributes: string facilityName, string facilityStatus, string facilityLocation

➔ Operations: updateFacility() || *add, delete, edit and publish park facility information*

LiveCams: The live camera that observes and reports the weather data on the mountain top in the park area which is displayed in the Park Information section.

➔ Attributes: Web Host server, Media Player plug-ins
➔ Operations: displayMountains() ||
-linkLiveCams() || *establishes connection between main server to park cameras*
-displayMountains() || *switches cameras to mountain view*
-displayParkRegion() || *switches cameras to park region view*

Weather API: This widget gathers latest weather information from the internet and displays forecasts to the website. It is user interactable so that they are able to view weather conditions on a specified day.

➔ Attributes: int temp, int precipitation, string humidity, string dayOfWeek, string weatherWarning
➔ Operations: useGPS() || *uses GPS mod to generate accurate precipitation predictions, returns precipitation*
-weatherChanged() || *detects significant change in weather, updates weather status immediately to API*
-DisplayWeatherDescription() || *collects weather data from selected dayOfWeek, returns temp and weather predictions*
-weatherIcon() || *displays rain, sunny, cloudy icons*
-setDayofWeek() || *prints display of weather of the week*
-getWeatherWarnings() || *calculator the temperature and humidity, grades them and returns weatherWarning status*

Interactive Park Map: Uses maps API to display park content and imagery for display on web pages and mobile devices. It features four basic map types (roadmap, satellite, hybrid, and terrain).

➔ Attributes: float elevation, string region, List String Events, float coordinates, List walkingPaths
➔ Operation: bool isTouchScreen() = yes || *allows user to interactive map with touch display devices*
-void mapZoom() || *takes touch and mouse input to return map zoom*
-void isHazardous() || *If there are hazardous conditions within the park reported by the staff, the system will update this information.*

Trekker: This class is geared towards the visitor allowing them to read, select, plan or print all the information about trails, tours, information and map of the park.

➔ Attributes: Visitor Log-in , Tour Planning, TrailCams, Trail Map & Information

➔ Operations:method1(Type) + method2(Type, Type)

Tour Planning: It contains reading information about the park tour when trekkers log into the system it helps them to plan a trip based on this information.

➔ Attributes: Event, Date, TourInformation, TourGuide, float paymentDue, numOfParty, hotels

➔ Operations:

-login() || *redirects user to login page before continuing through tour planning process*

-selectTourGuide() || *prints park tour guide option menu. User selects a tour guide and function returns tour guide name and saves it to tour planning order*

-selectEvents() || *prints list of events and menu options for user to view event information relevant to tour date*

-updateEventInfo() ||*edit, remove, and publish event information*

-getNearbyHotels() || *get Kilimanjaro nearby hotels and display booking choices for for visitor*

-TourInformation() || *edit, update, and publish tour information by admin*

-float pricePerPerson() || *takes in number of party members touring and calculate pricing before returning price to payment processing page*

-finalizeOrder() ||*takes user to payment processing page to finalize order*

-payment() || *system interacts with payment class*

-viewCustomerDetails() || *method for tour guide company to store and access customer information*

TrailCams: displays live cam of the trail conditions and allows visitors and guides to select a specific trail camera to view

➔ Attributes: Web Host server, Media Player plug-in

➔ Operations: chooseTrail() || *prints and takes input from trail menu with options to view specific camera*

-void displayTrail1()....displayTrail12() || *changes camera view to specified trail number*

Trail Map & Information: TrailMap class displays an interactive map to users and allows visitors and guides to gain access to trail information. Admin users have write permissions to edit trail information.

- ➔ Attributes: List string Trails, string TrailName, string TrailConditions, string coordinates, string trailDifficulty, string TrailInformation
- ➔ Operations: isTouchScreen() = yes || *Allows users to access and interact with trail map through mobile and other touch screen devices*
  -void gradeTrailConditions() || *interacts with weather API, takes in conditions of trail by ranger and grades the trail conditions whether it is safe or not to go to*
  -void gradeTrailDifficulty || *saves and stores a map of easy, medium, to hard levels of difficulty to list of trails*
  -void updateTrailInfo() || *edit, remove, and publish trail information by admin*
  -void getTrailRangerUpdates() || *retrieves status updates discovered by ranger, sends it to main server for park staff to update to system*
  trailsCams() || *interacts with TrailCams class to display camera views to page*

Database: holds video storage, backups user data and stores account information. Also contains edit website history and update history for a certain period of time.

- ➔ Operations: clearStorage() || *cleans up unnecessary files and functioned called when storage limit is hit*
  -backupUserData() || *makes a copy of user data*
  -storeAccountInfo() || *stores registered accounts to the website*
  -storePaymentInfo() || *saves an encrypted history of payment transactions*
  -storeParkUpdates() || *saves a yearly report with date, year, and time of updates to the park website's history*
  -setStorageLimit() || *storage holds data up to a certain amount of days*