Kilimanjaro Trekker System

Team members:

Alfredo Flores, Yunis Tamang, Olivia Butters

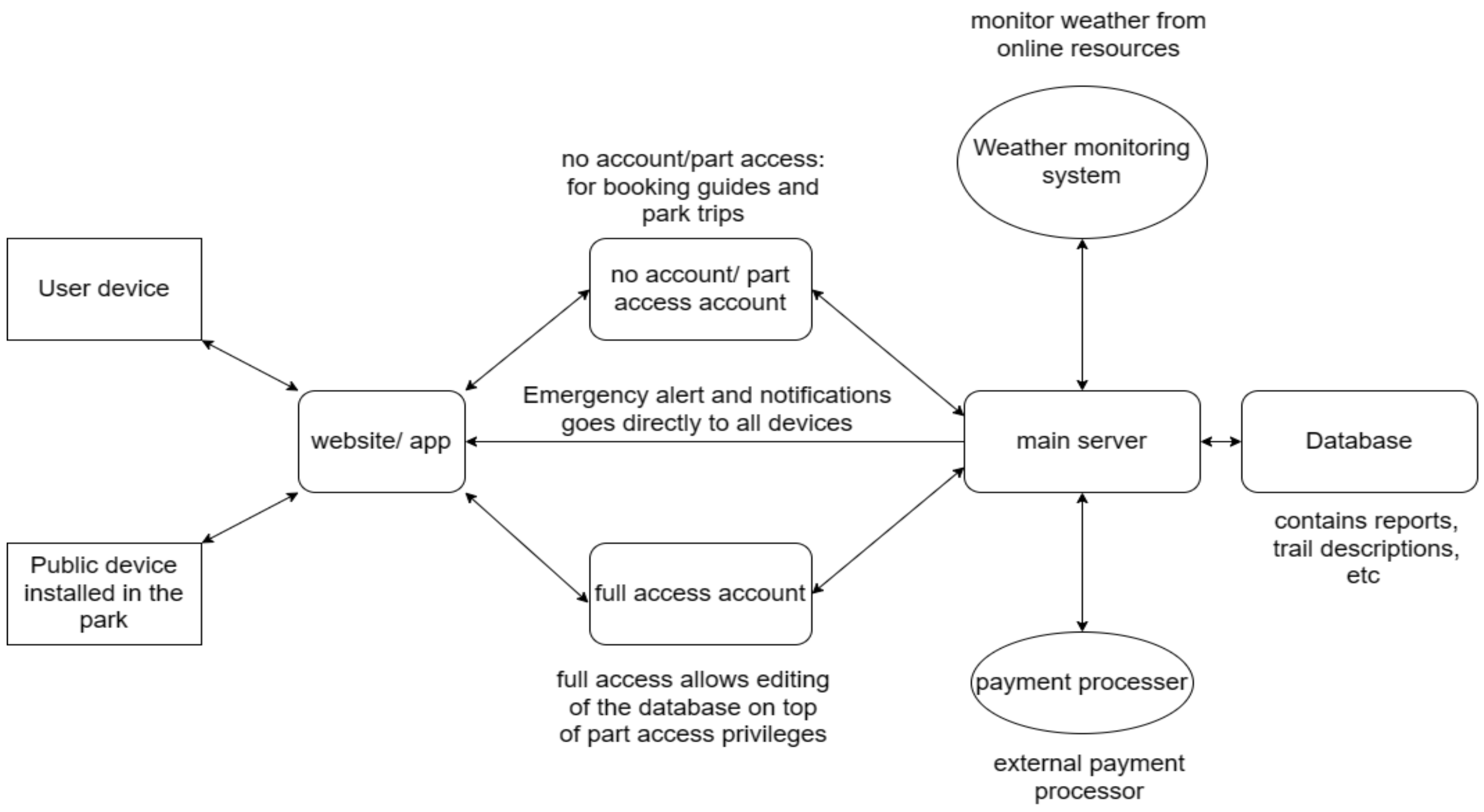
System Description

This software specification is about the Kilimanjaro Trekker. The is a trek guiding system for the Kilimanjaro National Park. The system's main purpose is to work as a guide for the trekkers through the many trails in the national parks. It also works as a notice board of sorts giving trekkers alerts on what is going on in the park. The national park covers a very large area, so the system is there to make the rangers' job easier and make the park safer for trekkers.

Furthermore, the guide and information provided with this system is to be delivered through a Google Maps view of the park. The point of this is to allow the map to be interactive throughout the park and toward the visitors to make the trekker system an enjoyable and easy to navigate experience. The website can be accessed via the internet and anything connected to the internet. With that, the system will also display any information regarding updates, change, and emergency information in real time to the website. Each trail in the park will have an informative section in the website to allow for a more detailed description of the area. There will be Tanzanian cell phone towers throughout the park for real time updates. Further in this document it will contain the functional requirements, non-functional requirements and use cases.

Software Architecture Overview

I. Architectural diagram of all major components:


monitor weather from online resources
Weather monitoring system
no account/part access: for booking guides and park trips
no account/ part access account
User device
website/ app
Emergency alert and notifications goes directly to all devices
main server
Database
contains reports, trail descriptions, etc
Public device installed in the park
full access account
full access allows editing of the database on top of part access privileges
payment processer
external payment processor

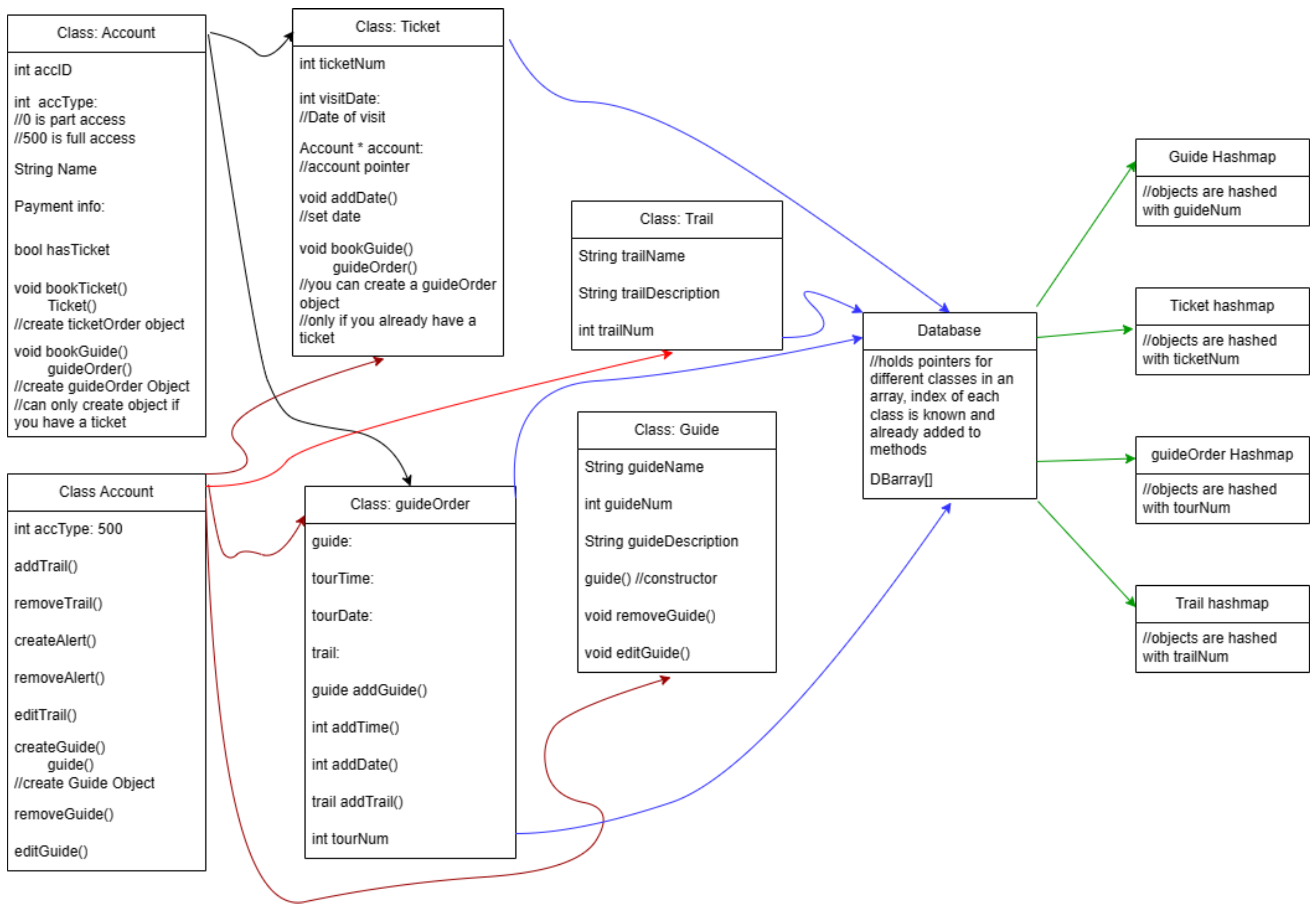
- Description of Architectural Diagram
    - The architectural diagram depicts a web-based application and outlines the different components and services that are needed to deliver the software system to all users. Starting from the left side of the diagram, users are able to get direct access to the application through either a website or an app that can be accessed via a user device such as a computer or smartphone, or alternatively, on-sight at the Kilimanjaro National Park through a physical computer station which has been installed. Once a user interacts with

the application, user requests are then sent across multiple different instances of the web server being used for the system, allowing for each user request to have a better direct connection to the server, while also improving the overall reliability and performance of the application. When it comes to the software functions that users will be able to utilize, regular users who will be visiting the Kilimanjaro trails will be able to create an account with limited access to the software. The functions they will have access to include, being able to see and receive any emergency alerts and notifications of current conditions on the trails, as well as being able to book any guided trips or general admission tickets for the trails. For park rangers, they will be provided with full access accounts that will allow them to have all the functions present in the limited access accounts, but also be able to directly update and edit information from the software database as they see needed. To ensure that weather conditions are able to be monitored and updated in real-time, the weather monitoring system for the software is directly linked to the main server which allows for a quick and reliable upload of data into the database. In addition, the payment processor system for the software is also connected to the main server to allow for safe and quick payment transactions for all users booking any of the guides or general admission tickets. Lastly, behind the main web server, there is a main database collection that manages and stores all of the application data. This database will be designed to be highly available and scalable as we should expect a large amount of data to be accumulated over time.

II. UML Class Diagram

III. Description of UML Diagram

- The following UML Diagram demonstrates the classes, attributes and operations needed for the Kilimanjaro Trekker System to properly function. The main part of this diagram comes from the database which then connects to the main class, the account class which branches into other classes. The diagram contains the different classes which have different capabilities for client use. Each class is important for the make up of this software and allows for the system to function smoothly and efficiently.



IV. Description of Classes

- **Account Class:**

    - The account class is responsible for managing the personal information of the customer. This class acts as the main class, connecting all the other classes. This includes the customer creating a unique ID for their account along with adding their payment details which utilizes the Kili Trekker System. This account section is responsible for the customer ability to obtain a ticket. The customers are in control of the information inputted into the system however admin have access to that information.
- **Ticket Class:**
    - The ticket class is the class in control of the ticket that the customer receives after their purchase. This ticket is used to get into the park and access the information that the park provides. The ticket class is branched off from the account class with the account class inheriting some of the features from the account class.
- **guideOrder Class:**
    - The guideOrder class is derived from the account class. In order to access this class, a ticket purchase is required. This class provides and allows for customers to learn about guides/tours that are present at the park. It also enables users to pick a time and date.
- **Guide Class:**
    - The guide class is for the client to view the tour guides working that day. This class enables the customer to learn more about their tour guide as well as gain more information about the guide’s experience in the park to benefit their experience in the park.
- **Trail Class:**

    - The trail class is an informative class that provides information about the trails in the Kilimanjaro Trekker System. This class displays the trail number, a description of the trail and the name of it.
- **Database:**
    - `DBarray[]:` The database is used to store pointers for the different classes in an array with the index of each class already known and already added to the methods.
- **Guide Hashmap:** The objects are hashed with guideNum.
- **Ticket Hashmap:** The objects are hashed with TicketNum.
- **guideOrder Hashmap:** The objects are hashed with tourNum.
- **Trail Hashmap:** The objects are hashed with trailNum.

V. Description of attributes

- **Account Class:**
    - `Int accID:` This is the account number/ID. It is displayed as an integer.
    - `Int accType:` The account type is the label or type of account that is created. 0 is the part access and 500 is full access. This is labeled as an integer.
    - `String Name:` The name of the customer is placed in a string.
    - `Bool hasTicket:` Is a boolean that returns true if the account has a ticket and false if there is no ticket.
- **Ticket Class:**
    - `Int TicketNum:` Each ticket has a unique number therefore ticketNum assigns a ticket number. This is displayed as an integer.
    - `Int visitDate:` This integer is the date the customer chooses to visit.
    - `Account* account:` This is an account pointer pointing to the account.

- **guideOrder Class:**
  - `String Guide`: This string holds the name of the potential guides that the customer can choose from.
  - `Int tourTime`: This integer is the time that the tour will be held.
  - `Int tourDate`: This integer is the date in which the tour will be held.
  - `Int Trail`: This integer holds the trail number the tour will be taking place at.
  - `Int TourNum`: This integer will be the number of the tour that the customers will be participating in.
- **Guide Class:**
  - `String GuideName`: This string holds the name of the guide that the customer will be touring with.
  - `Int guideNum`: This integer holds the number of which the guide is labeled in regards to their `tourNum`.
  - `String guideDescription`: This string holds the description of the guide. This information includes the experience and unique capabilities of the guide.
- **Trail Class:**
  - `String trailName`: This string is the name of a specific trail that is selected.
  - `String TrailDescription`: This string is the description of the trail that has been selected.
  - `Int trailNum`: This integer is the number that identifies the trail. Ex: Trail 1 → easier trail route.
- **Guide Hashmap:** The objects are hashed with `guideNum`.

- **Ticket Hashmap:** The objects are hashed with `TicketNum.`
- **guideOrder Hashmap:** The objects are hashed with `tourNum.`
- **Trail Hashmap:** The objects are hashed with `trailNum.`

VI. Description of operations

- **Account Class:**
  - `Void bookTicket():` Creates ticketOrder object.
  - `Void Ticket():`
  - `Void bookGuide():`
  - `Void guideOrder():` Creates guideOrder() object. This object can only be created if a ticket was already purchased.
  - `addTrail():` Enables the user to add a trail to the account.
  - `removeTrail():` Enables the user to remove a trail from the account.
  - `editTrail():` Enables the user to edit the trails in their account.

    `createGuide():` Creates a guide object. `removeGuide():` Enables the user to remove the guide from profile.
  - `editGuide():` Enables the user to edit the guide from their profile.
- **Ticket Class**
  - `Void addDate():` This sets a date for the client. This passes nothing through its parameters and does not return anything.
  - `Void bookGuide():` Creates a guideOrder object. Does not return anything or does not pass anything through its parameters.

- `Void guideOrder():` This is the guideOrder object. Allows for a customer to order a guide. Does not pass anything through its parameters or return anything.

- **guideOrder Class:**
    - `Guide addGuide():` Sets the opportunity to add a guide to the account. There is no return type or parameters passing through.
    - `Int addTime():` Creates a function to add a time to their profile to look for a guide. There is no return type or parameters.
    - `Int addDate():` Creates a function to add a date to their profile when looking for a guide. There is no return type or parameters.
    - `Trail addTrail():` Creates a function to add a trail to profiles. There is no return type or parameters.
- **Guide Class:**
    - `guide():` A constructor for the guide class.
    - `Void removeGuide():` Creates a function to remove the guide from profile. There is no return type or parameters.
    - `Void editGuide():` Creates a function to edit the guide in the profile. There is no return type or parameters.

Development Plan and Timeline

I. Team Responsibilities

Alfredo Flores: Lead Project Coordinator

- In charge of project management, distribution of responsibilities, tracking progress and making sure timelines as well as consumer needs are being met. Will create a development plan that will be carried out by the development team and communicate directly with the Tanzanian park rangers about specific implementations they need for the system.

Olivia Butters: Research Team Leader

- Main responsibilities will be to lead a team to conduct research on the Kilimanjaro trails. Needed data will consist of information on the physical trails, Tanzanian cell phone systems, weather tracking systems, possible threats as well as needed safety protocols.

Yunis Tamang: Senior Project Developer

- Will be in charge of leading a development team who will be responsible for the development of the code behind the software. The team will develop UI, security systems, database, and compatibility with the Tanzanian cell phone systems. Will also keep progress of project documentation throughout the development process. Lastly, the development team will conduct debugging as well as testing of the software before release.

II. Timeline

The software release date will be 6 months from now.

- Month 1: Planning and research of project concept.
- Month 2: Early development of software, and early prototype.
- Month 3-4: Refining of prototype and debugging.
- Month 5-6: Completion and release of software.

III. Partitioning of Tasks

- The developers should design different charts/diagrams (flowcharts, Architecture) to ensure correct rhythm and layout of the software.
- Use the client's specifications and organize a system that follows those specifications.
- Collaborate with senior programmers to create unique and efficient code for the software system.

- Utilize and develop different tools within the IDE to make certain that the software is capable of running.
- Work with team members to debug, review code, and monitor the system as the development process continues.
- Manage the software development life cycle and timeline to make sure that timelines are being met.
- Problem solve, troubleshoot and brainstorm to come up with unique solutions to each individual problem that pops up in the coding process.