CPEN 391
Module 1 Project Proposal

Team 10

Andrew Shieh
Aayush Bisht
Elbert Ng
Shade Wong
Vincent Sastra

TABLE OF CONTENTS

INTRODUCTION

Illegal wildlife trade has escalated dramatically over the last decade. Human population growth, increasing wealth and access to wildlife and improved global transport links have all played a part. In parts of Asia where the tradition of wildlife consumption is culturally embedded, demand for particularly high-value species has soared. At the other end of the supply chain, rural poverty in the countries that harbour these species is driving desperate people to plunder their own natural resources for scant reward.

Conservation interventions have historically focused on regulation by introducing new and stronger legislation and trade controls. However, regulation alone is not sufficient to combat illicit wildlife trade. Living in an era where technology is advancing at an unprecedented rate, we can take advantage of the current technology by utilizing it to protect species from poaching while also tackling the root causes of illegal trade.

Our project aims to improve the efficacy of wildlife conservation by providing a feasible solution to the following challenges: *(i)* Illegal poaching activities, *(ii)* Human-animal conflict, *(iii)* Aggressive animal behaviour tracking, and *(iv)* Environmental issues, such as wildfire. Our animal monitoring tag can be attached to an animal to collect various data to achieve the following functionalities: *(i)* Location tracking, *(ii)* Activity and heartbeat monitoring, *(iii)* Mood prediction, *(iv)* Human presence detection, *(v)* Geofencing alert, *(vi)* Poaching risk alert, and *(vii)* Wildfire detection alert. Most of the posed challenges can be handled or avoided if authorities are notified timely.

TARGET MARKET IDENTIFICATION

The target markets and list of constraints/needs imposed by each target market for our animal monitoring tag are:

Target Market 1: Wildlife conservation organizations & authorities

- Waterproof casing
- Wearable on any animal (ranging from small racoons to large elephants)
- Real-time data processing and alert system
- Web-based monitoring application that is well organized and easy to understand
- Robust service that works in an environment without WiFi
- Outdoor location tracking service
- Portable battery and long battery life

Target Market 2: Domestic pet owners that have to leave their pets unattended regularly (usually at the age of 20-50; employed adults)

- Intuitive user interface
- Mobile-based monitoring application
- Indoor location tracking service
- Portable battery and long battery life

PRODUCT REQUIREMENTS SPECIFICATION

1. Waterproof casing to ensure the product is resilient to different weather conditions
2. Portable and long-lasting battery supply to minimize the replacement of batteries, hence the interruption of animal activity
3. Real-time data processing and alert system to notify users about potential risks promptly
4. Real-time location monitoring service with geofencing capabilities that support both short and long-range tracking
5. Responsive application interface that supports different screen sizes, eg. desktop, laptop, tablet, mobile
6. Robust service that works in an environment without WiFi

HIGH-LEVEL DESIGNS

DESIGN A

I. Hardware Architecture - FPGA as Tag

1. **DE1-SoC Computer System with ARM processor**
   Perform hardware-accelerated matrix multiplication to train the animal activity prediction ML model using collected accelerometer data and potential poacher classification ML model with real-time video stream.

2. **Terasic RFS Daughter Board**
   - Bluetooth chip
     Detect nearby Bluetooth devices, which indicate human presence.
   - WiFi chip
     Connect and send processed data to a cloud-based database for long-term storage.
   - Temperature and humidity sensor
     Detect wildfire based on collected temperature data.
   - Accelerometer
     Predict animal activities such as resting, walking, running etc. based on collected sensor data.

3. **Arducam Camera Module**
   Detect human presence and identify unauthorised personnel based on the video captured.

4. **NEO-6M GPS Module**
   Track animal location and setup geofence.

5. **Pulse and Heart Rate Sensor**
   Predict animal mood based on collected heartbeat rate.

II. Software Architecture - Serverless Architecture

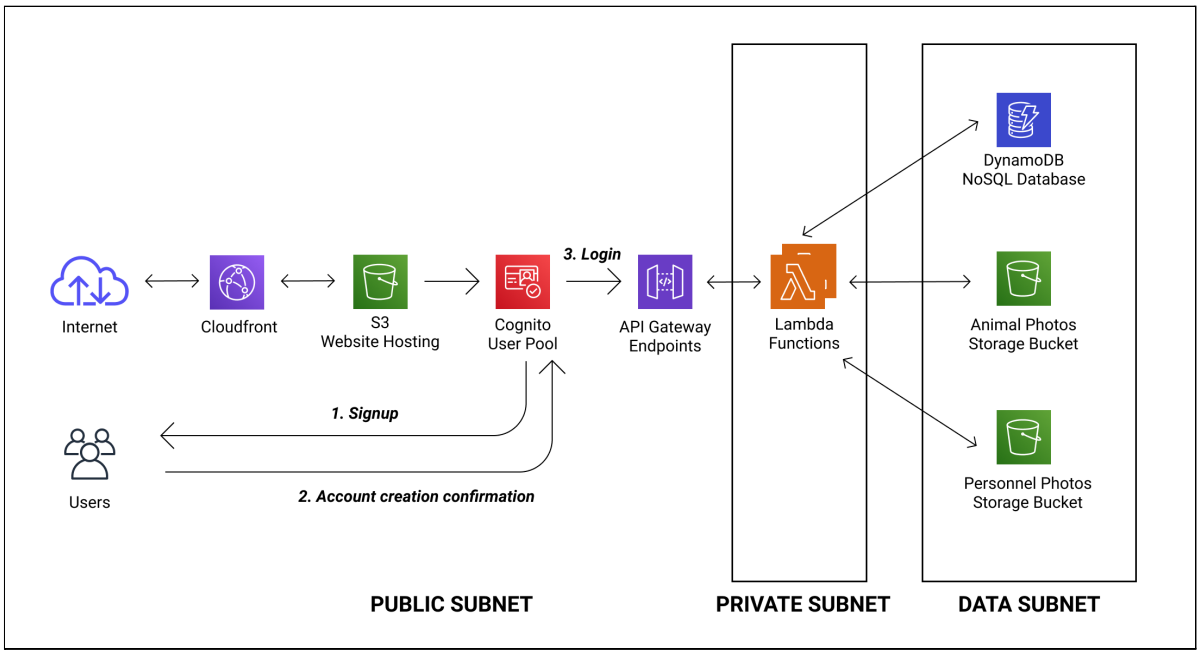


Diagram 1. AWS-based Serverless Architecture

The diagram above depicts a high-level overview of a possible implementation of serverless architecture for our web-based application. Each service composing the architecture and its functionality are detailed as follow:

1. **Website Hosting**
   *The front-end application is packaged in one set of static files downloaded by the user's browser at first URL access. Those static files are hosted on **S3** and exposed through **CloudFront**, a content delivery network (CDN) service that delivers data to users globally with low latency and high transfer speed.*

2. **Users and Authentication**
   ***Cognito User Pool** can be used in our application for authentication, user management, resource access control and external identity provider integration. It also provides flexibility to dispatch events and trigger Lambda functions, where we can use to customize the application authentication flow.*

3. **Resource Access Endpoints**
   *The front-end application needs to connect to the backend service in order to retrieve and/or push data. **API Gateway** is used to handle the HTTP connections and endpoint routes, synchronously triggering a **Lambda function** that fetches/pushes data from/to the database for each route.*

4. **Data Storage**

   DynamoDB *is a web-scale NoSQL database that provides low latency access to data and it is well suited to the serverless application as a primary datastore. It can be used in our application to store the data described in the "Major Data Structures" section.* ***S3 bucket*** *provides a flexible and scalable storage service where we can use to store large image assets uploaded by users to our application.*

III. Major Algorithmic Processing Required

1. Accelerometer and temperature data are collected from the accelerometer and temperature sensors in the RFS daughter board. Additionally, GPS coordinates are collected from the GPS module and the heartbeat rate is collected from the heartbeat sensor.
2. Real-time video is streamed from the Arducam Camera module into the FPGA and processed using face recognition libraries such as OpenCV, to identify potential poachers.
3. Collected sensors data is used to train our animal activity prediction model and mood prediction model. The training module is made using a hardware-accelerated machine learning program created with the Python Chainer Library.
4. All processed results in the FPGA are then uploaded to the cloud-based database through API Gateway endpoints via WiFi.
5. Processed data is then fetched by the web application from the cloud-based database again through API Gateway endpoints and is displayed in our web-based interface for the client use.

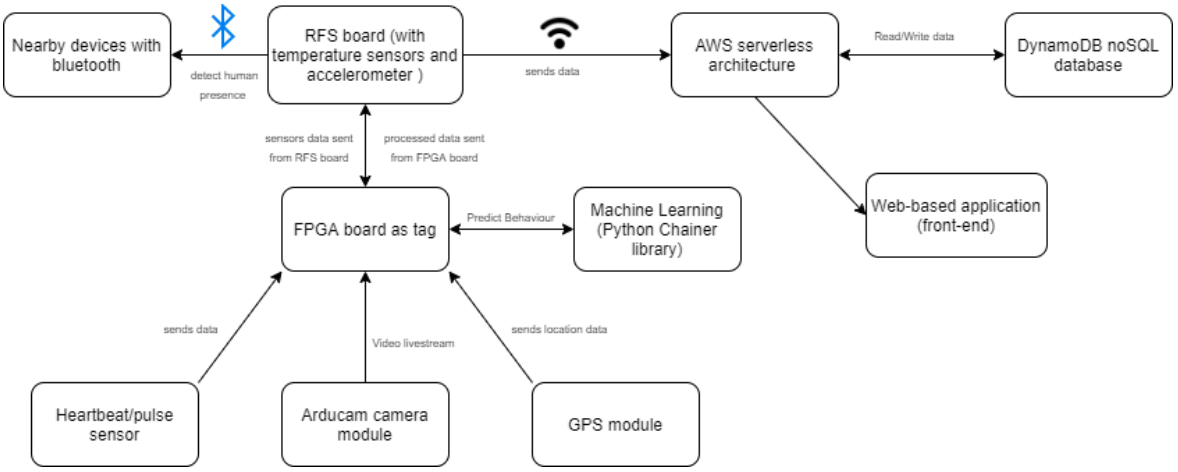


Diagram 2. Design A Flowchart

DESIGN B

I. Hardware Architecture - Arduino as Tag

1. **DE1-SoC Computer System with ARM processor**
   Act as a remote server that receives various Arduino sensor data at a constant interval via Bluetooth and performs hardware-accelerated matrix multiplication to train the animal activity prediction ML model using received data.

2. **Terasic RFS Daughter Board**
   - Bluetooth chip
     - *Establish connection with the Arduino board to receive sensor data and pass the data into FPGA for processing.*
     - *Detect nearby Bluetooth devices, which indicate human presence.*
   - WiFi chip
     Connect and send processed data to a cloud-based database for long-term storage.

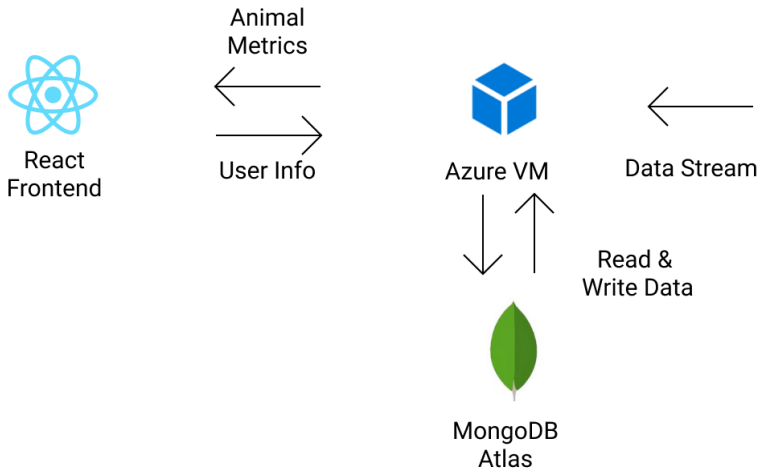
3. **Arduino Nano 33 BLE Sense**
   - 3-axis accelerometer
     Predict animal activities such as resting, walking, running etc. based on collected sensor data.
   - Temperature and humidity sensor
     Detect wildfire based on collected temperature data.
   - Microphone
     Detect poacher activity based on acoustic sound captured.

4. **NEO-6M GPS Module**
   Track animal location and setup geofence.

5. **Pulse and Heart Rate Sensor**
   Predict animal mood based on collected heartbeat rate.

II. Software Architecture - VM-based Architecture

1. VM Hosting



A cloud server is used to host our client's website. We will run 2 different programs on 2 different ports. The first program will establish a TLS connection with the FPGA to receive the animal metrics and write it down on the database. The second program will host the webpage and web app through the HTTP port.

2. SQL Database

A SQL database with four tables can be used to store the four major data structures described in the "Major Data Structures" section respectively. An Entity Relationship Diagram (ERD) describing the relationship between each of the table is included below:

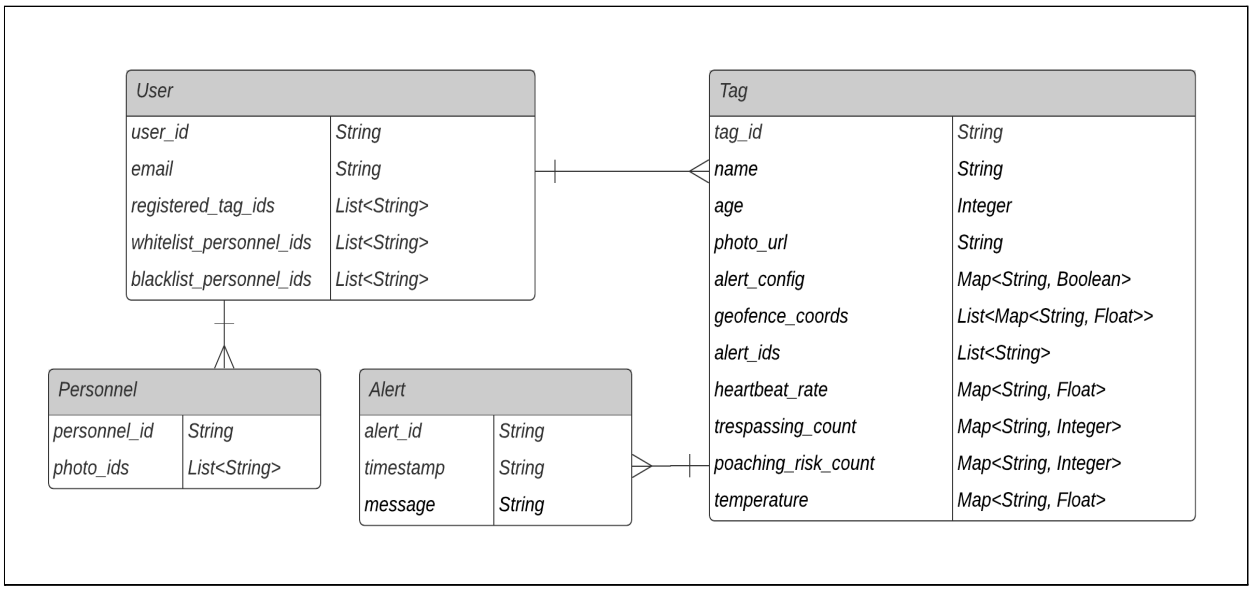


Diagram 3. Entity Relationship Diagram (ERD) for Major Data Structures

III. Major Algorithmic Processing Required

1. Accelerometer, temperature and acoustic sound data are collected from the accelerometer, temperature sensor and microphone on the Arduino board. Additionally, GPS coordinates are collected from the GPS module and the heartbeat rate is collected from the heartbeat sensor.
2. The Arduino then forwards the data to the FPGA, which acts as a remote server. The data is transferred by establishing a Bluetooth connection between the Arduino and the RFS Bluetooth chip.
3. Collected sensors data is used to train our animal activity prediction model and mood prediction model. The training module is made using a hardware-accelerated machine learning program created with the Python Chainer Library.
4. All processed results in the FPGA are then uploaded to the cloud-based database through VM-based backend service via WiFi.
5. Processed data is then fetched by the web application from the cloud-based database again through VM-based backend service and is displayed in our web-based interface for the client use.

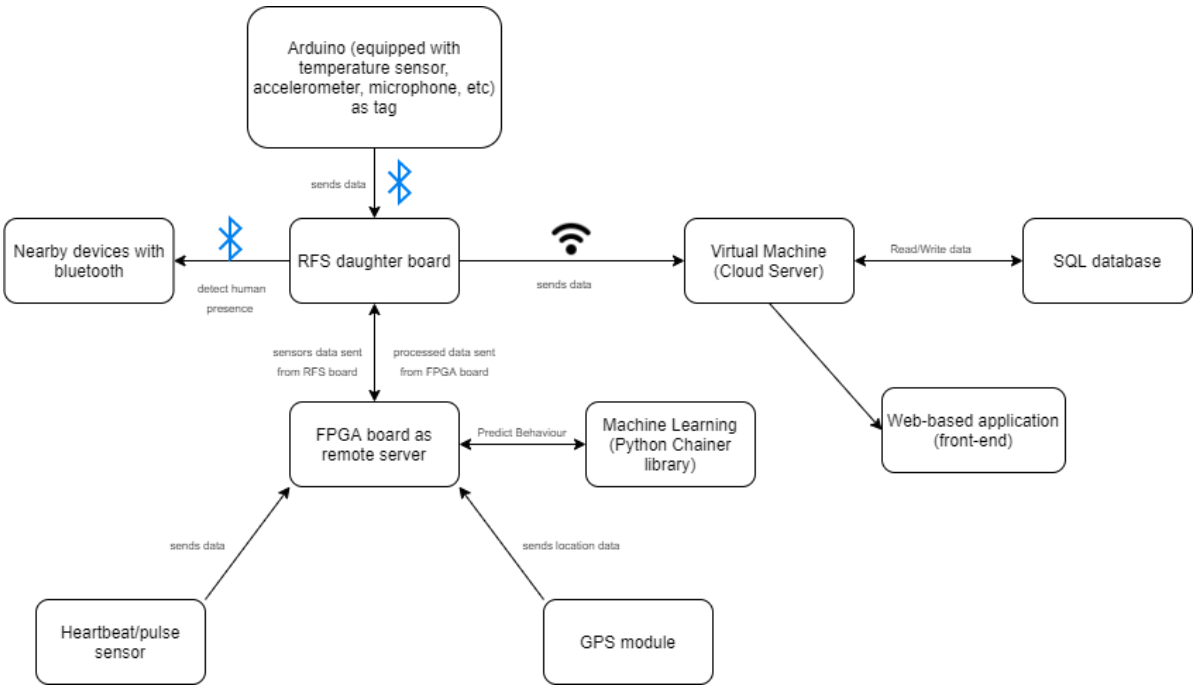


Diagram 4. Design B Flowchart

DESIGN A AND DESIGN B COMMON FUNCTIONALITIES

A. Software Functionalities

1. **Location tracking**
    - Display real-time GPS coordinates for every registered tag on a map

2. **Activity and heartbeat monitoring**
    - Display animal heartbeat rate over time using a line graph
    - Display animal daily activities in a pie chart, eg. 10% walking, 20% running, 65% resting, 5% aggressive behaviour
    - Send real-time notification to the user for any detected aggressive behaviour
    - Allow user to select desired start and end date for data display

3. **Geofencing alert**
    - Send real-time notification to the user when animal is moving out of defined area or moving into pre-defined risky zones
    - Allow user to define geofence and risky zone boundaries for every registered tag with multiple GPS coordinates that form an enclosed area when connected with adjacent coordinates

4. **Human presence detection**
    - Send real-time notification to user when unauthorised personnel is detected near the animal tag
    - Display a list of blacklisted people (*Design A*) or Bluetooth devices (*Design B*), which identified as unauthorised by the ML model in the past
    - Allow user to remove people (*Design A*) or Bluetooth devices (*Design B*) from the blacklist or move them from blacklist to whitelist

5. **Poaching risk alert**
    - Send real-time notification to user when aggressive or feared acoustic sound from animal, a sudden increase in animal heartbeat rate or crowd voices/gunshots are detected

6. **Wildfire detection alert**
    - Display environmental temperature over time using a line graph, with optimal temperature overlays on the graph for a better comparison between the current temperature and the optimal temperature
    - Send real-time notification to user when the detected temperature goes beyond a certain threshold

B. GUI Prototype

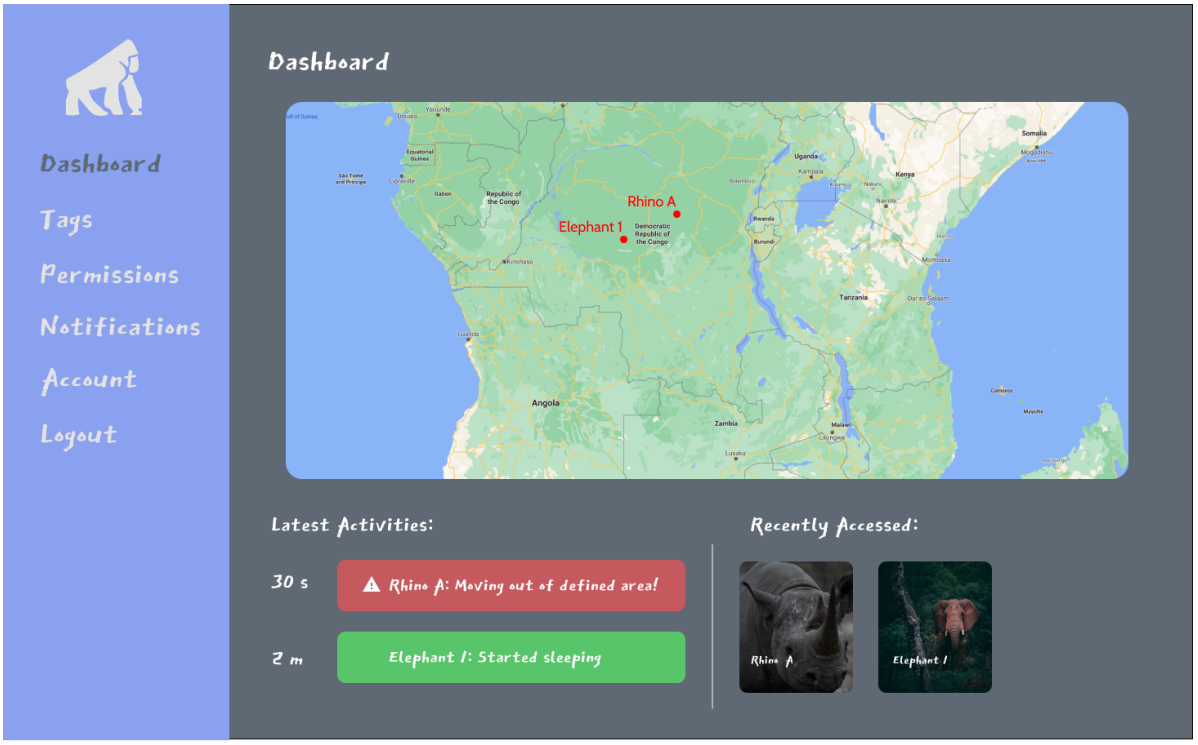


Figure 1. Main Dashboard

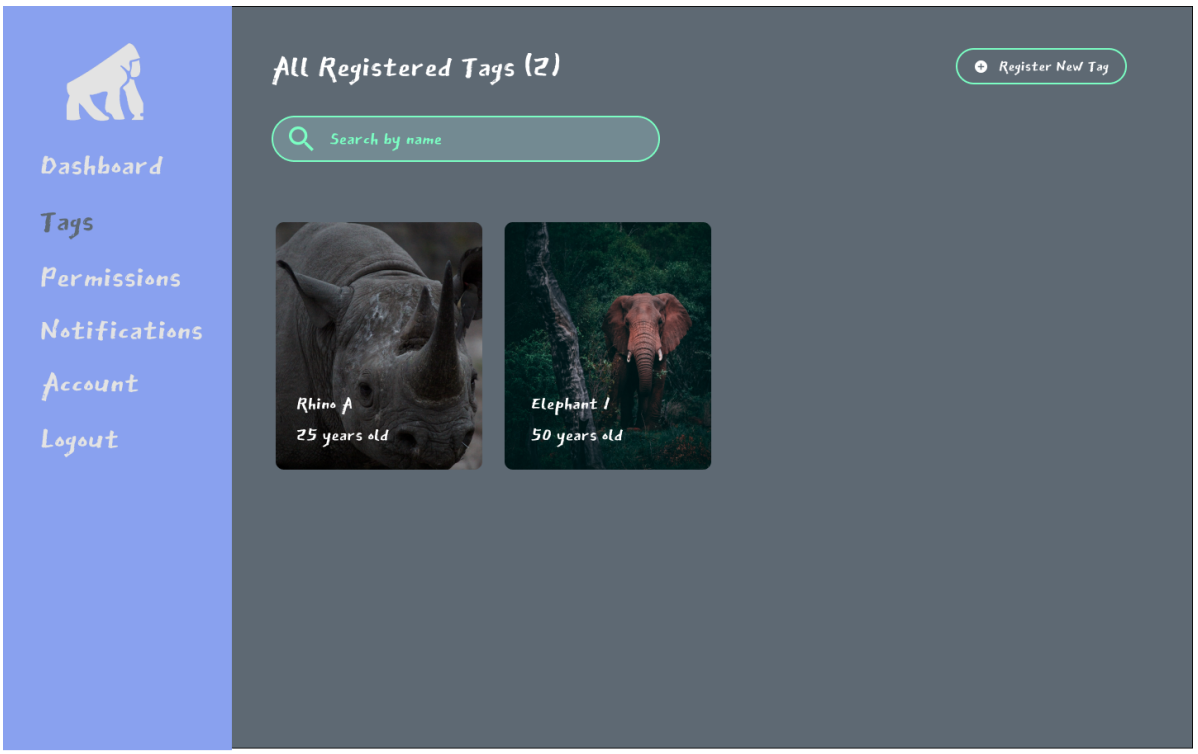


Figure 2. Registered Tags Page

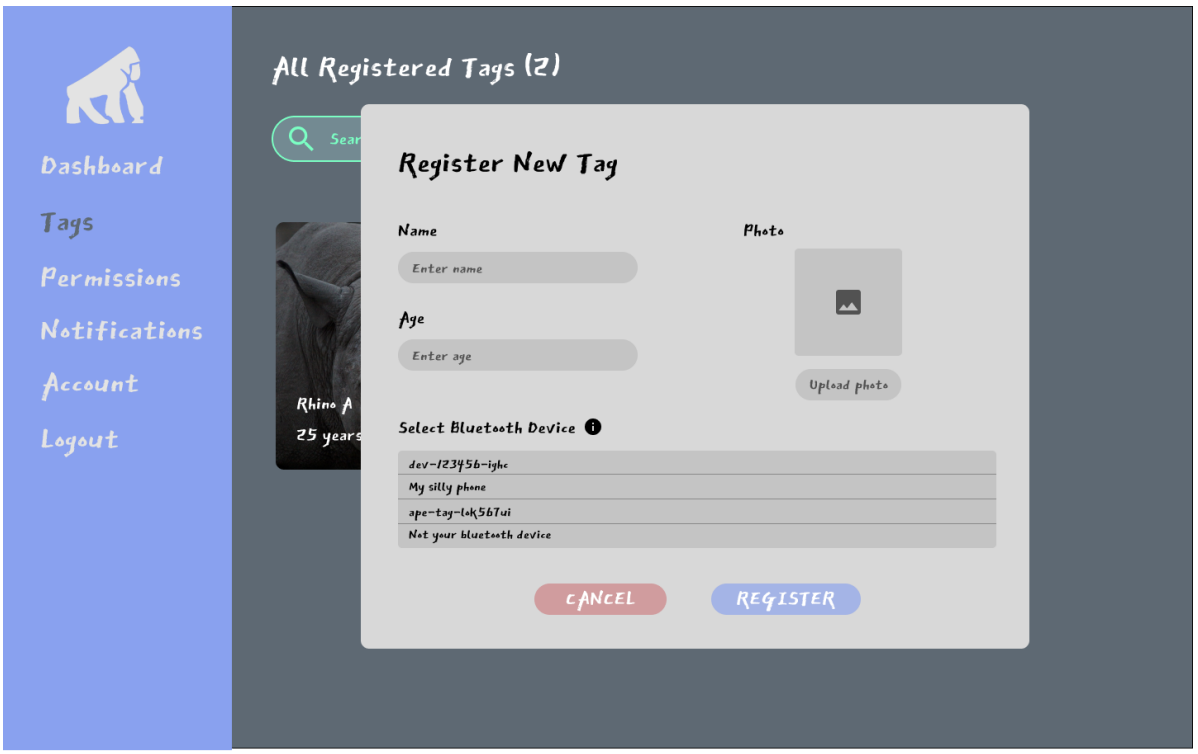


Figure 3. Register New Tag Dialog

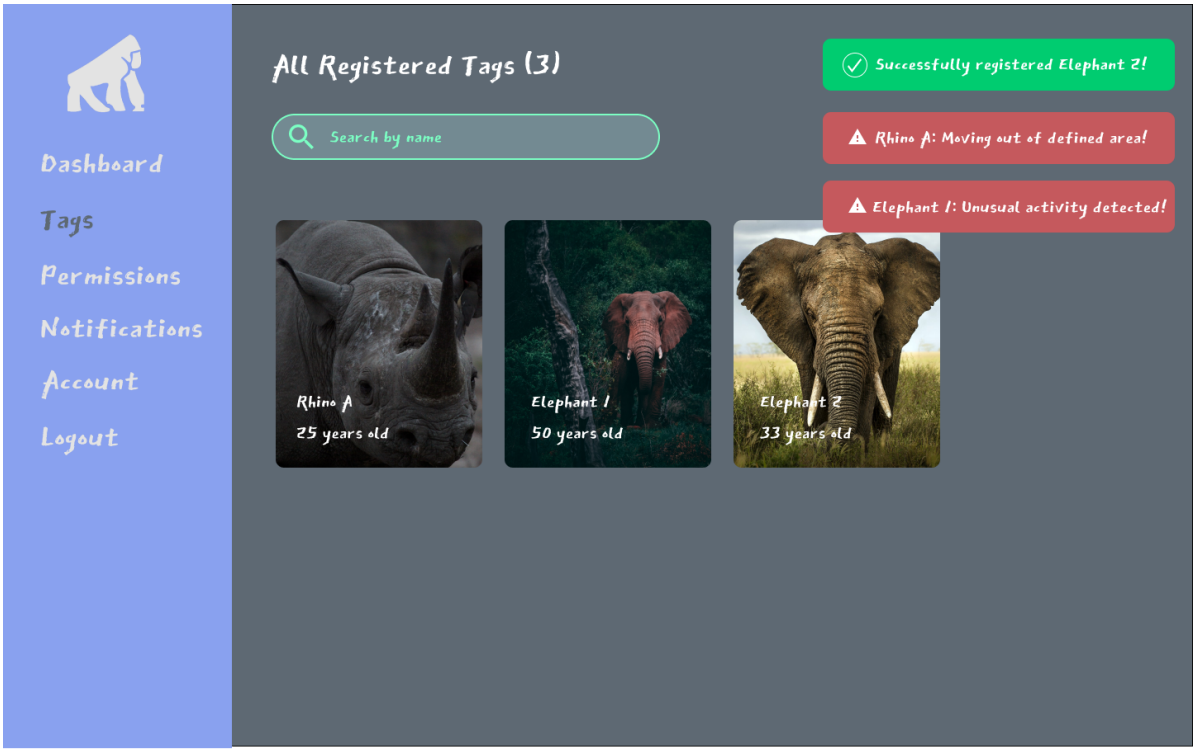


Figure 4. Notification Snackbar Display

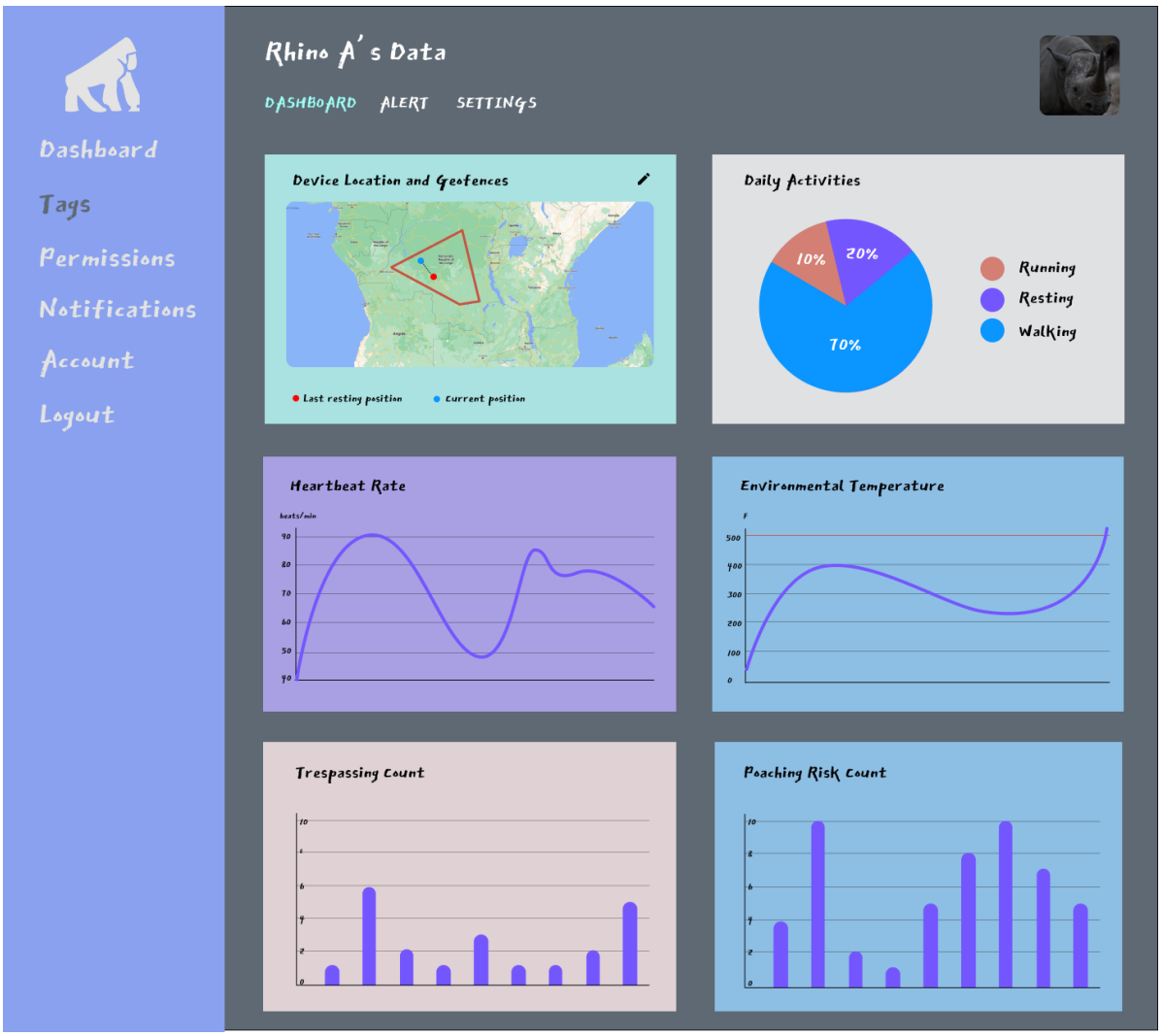


Figure 5. Individual Animal Dashboard

C. Major Data Structures

The major data structures for our application are included in the table below along with a list of important attributes for each of them. All of the data will be stored in a cloud-based database.

Data	Attributes
User	● user_id <String> ● email <String> ● registered_tag_ids <List<String>> ● whitelist_personnel_ids <List<String>> ● blacklist_personnel_ids <List<String>>
Tag	● tag_id <String> ● name <String> ● age <Integer> ● photo_url <String> ● alert_config <Map<String, Boolean>> ● geofence_coords <List<Map<String, Float>>> ● alert_ids <List<String>> ● heartbeat_rate <Map<String, Float>> ● trespassing_count <Map<String, Integer>> ● poaching _risk_count <Map<String, Integer>> ● temperature <Map<String, Float>>
Personnel	● personnel_id <String> ● photo_ids <List<String>>
Alert	● alert_id <String> ● timestamp <String> ● message <String>

DESIGNS SELECTION

High-Level Designs Comparisons

Advantages of Design A

1. **Smaller Size**

   Size of the prototype for Design A will be significantly smaller than Design B, as most of the hardware components are connected directly to the FPGA board.

2. **Lower Latency**

   Design A boasts a lower latency design as the FPGA is inside the tag itself and does not need to fetch sensor data from another device as opposed to Design B. This means that Design A will perform faster than Design B.

3. **Lower Cost**

   Design A is sleeker and hence uses less hardware components therefore bringing down the cost of the project. AWS serverless cloud computing also costs significantly less than hosting a VM.

4. **Less Complexity**

   NoSQL database used in Design A is easier to work with but is less organised given the relationship between each data structures used in the project.

Advantages of Design B

1. **Shorter Setup Time**

   Working with a VM is easier for cloud computing while working with serverless requires more time to set up and is generally harder than working with a rented VM.

2. **Modular Design**

   Design B has an Arduino Nano board attached inside the tag which provides easier interfacing with sensors and a more modular design. New hardware components are easier to add or remove.

3. **Organised Database**

   Using SQL for Design B provides a much cleaner and much more organised database compared to a non-relational database. It eliminates issues such as data duplicity, provides ease of access and high security.

4. **Lower Skill Level Required**

   Design B uses a shared VM from AWS and an Arduino Nano to interface with sensors. Since these are robust programs developed by experienced professionals, Design B promises a robust architecture and requires less skill to execute.

High-Level Designs Evaluation

Design A and *Design B* were analyzed quantitatively using a Weighted Decision Matrix (WDM). In the WDM, the cost, latency, complexity of project, modularity, data organization, system testability, size and weight of product, were taken into consideration.

Weighing Factors Selection

Weighing factors and ratings were determined by prioritizing the costs and performance of the product. As cost and latency are our priorities, these two criteria were assigned a high weighting of 25%. We believe that having a fast performance product built with minimal cost are relatively important compared to the other evaluation criteria.

Size and weight of the prototype hold some importance in the design but are not weighted high. Since the actual product will be using custom chips, which are much sleeker than FPGA or Arduino Nano, this ultimately brings size and weight difference to be negligible.

Complexity, modularity, data organization and testability were given a weighting of 10% each, since these are fairly important criteria to be taken into consideration when evaluating the high-level designs. These criteria help to evaluate the feasibility in building the product, as well as the scalability of the product.

Weighted Scores Calculation

Each criterion is assigned a rating between 1 to 5 (inclusive) with higher rating means more desirable. The individual rating is then multiplied by its corresponding weight, producing a weighted score for each criterion. The weighted scores for each criterion are then summed up to obtain the total weighted score for each design.

Criteria	Weight	High-Level Designs	
		Design A	**Design B**
Cost	25%	4.5	3
Latency	25%	4	2
Complexity	10%	2	4.5
Modularity	10%	2	4
Organized DB	10%	3	5
Testability	10%	3	4
Size (Prototype)	5%	5	4
Weight	5%	3.5	2
Weighted Score		**3.55**	3.30
Ranking		**1**	2

Table 1. Weighted Decision Matrix

Conclusion

By comparing the two designs using WDM, *Design A* with a higher weighted score of 3.55 is selected.