**IOT PROJECT**

**CSC 8223 Internet Of Things**

**Project: Digital Fountain**

**Semester: FALL 2023**

**Team: 11**

**Members: Huaiyuan Chu, Dong Yang, Omar Madjitov**

**Date: 12/03/2023**

INTRODUCTION

**Background and motivation:**

Fountains are often used in the Entertainment and Tourism industry(\*\*where there’s a lot of money).

There are several reasons why fountains are appealing to people:

**Aesthetic:** The synchronized play of water, light, and sound adds beauty and vibrancy to any environment.

**Community Engagement**: It becomes a focal point for community gatherings, enhancing social interaction.

**Therapeutic Effects**: The combination of soothing sounds and visual harmony can have a calming effect, contributing to mental well-being.

**Cultural Showcase**: The fountain can be programmed to reflect local culture and festivities through its music and light shows.

**Problem Definition:**

While this fountain offers numerous benefits, manual control of its features presents several challenges:

**Labor Intensive:** Constant human oversight for operation and adjustments.

**Inconsistency:** Human error can lead to irregularities in performance.

**Limited Responsiveness:** Difficulty in adapting to environmental changes and audience preferences in real time.

To address these challenges, we propose the development of a smart application, that will continuously collect the Image, Sound, and Light sensor data, and make decisions to control the Configurations of a fountain (Turn On/Off, based on whether there are people or not; Change Levels of Pressure, based on the number of people; Change the levels of Light, based on ambient brightness levels & Music, based on ambient sound levels).

**System Overview:**

Create a Fountain Control Model based on sensor data collected around a fountain.

Sensors: Image, Light, Sound

The function of each sensor is as follows

**Image sensor:** Collect the images of the surrounding crowd and feed into CNN for headcount, which will then be used to change the behavior of the fountain i.e. start a fountain show, when the area is densely crowded for entertainment, and stay in standard mode when there are not that many people in a casino.

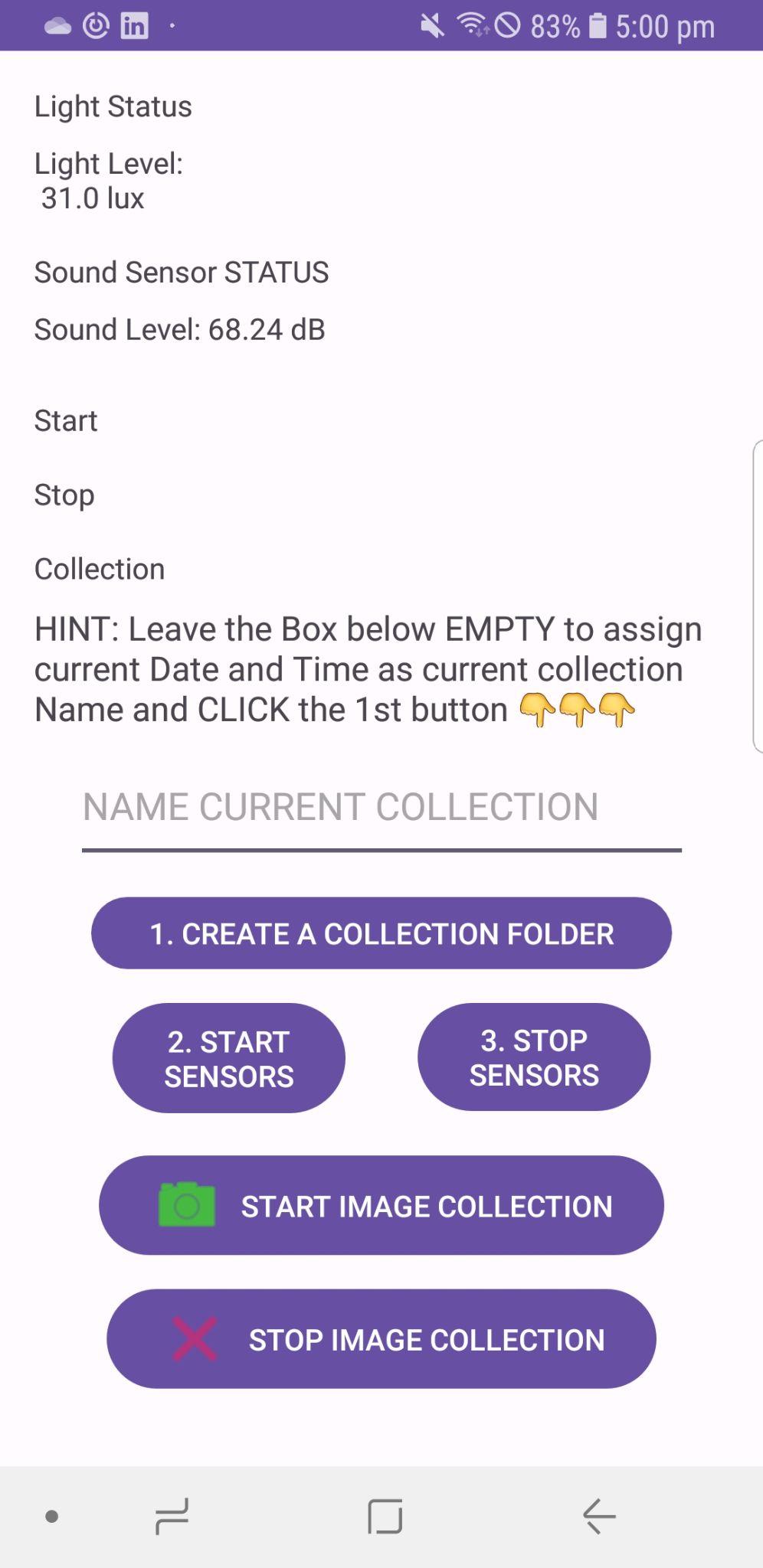
**Light:** It’s important to be efficient while maintaining the beauty of our fountain. Therefore we are going to collect the light sensor data. So we don’t waste electricity on Fountain Lights when the surroundings are already very bright, because the visitor won’t be able to see the fountain lights during the day anyway.

**Noise Sensor / Microphone:** Since our fountain is also going to play music during the “water show” it is important for our fountain to be able to capture surrounding ambient noise levels to be able to adjust the volume of the music. For example, if the surrounding area is loud, the fountain will increase the volume of the music, so that visitors are able to comfortably enjoy the show.

Combining sensor data overall to change the water style, light, and Music volume of a fountain.

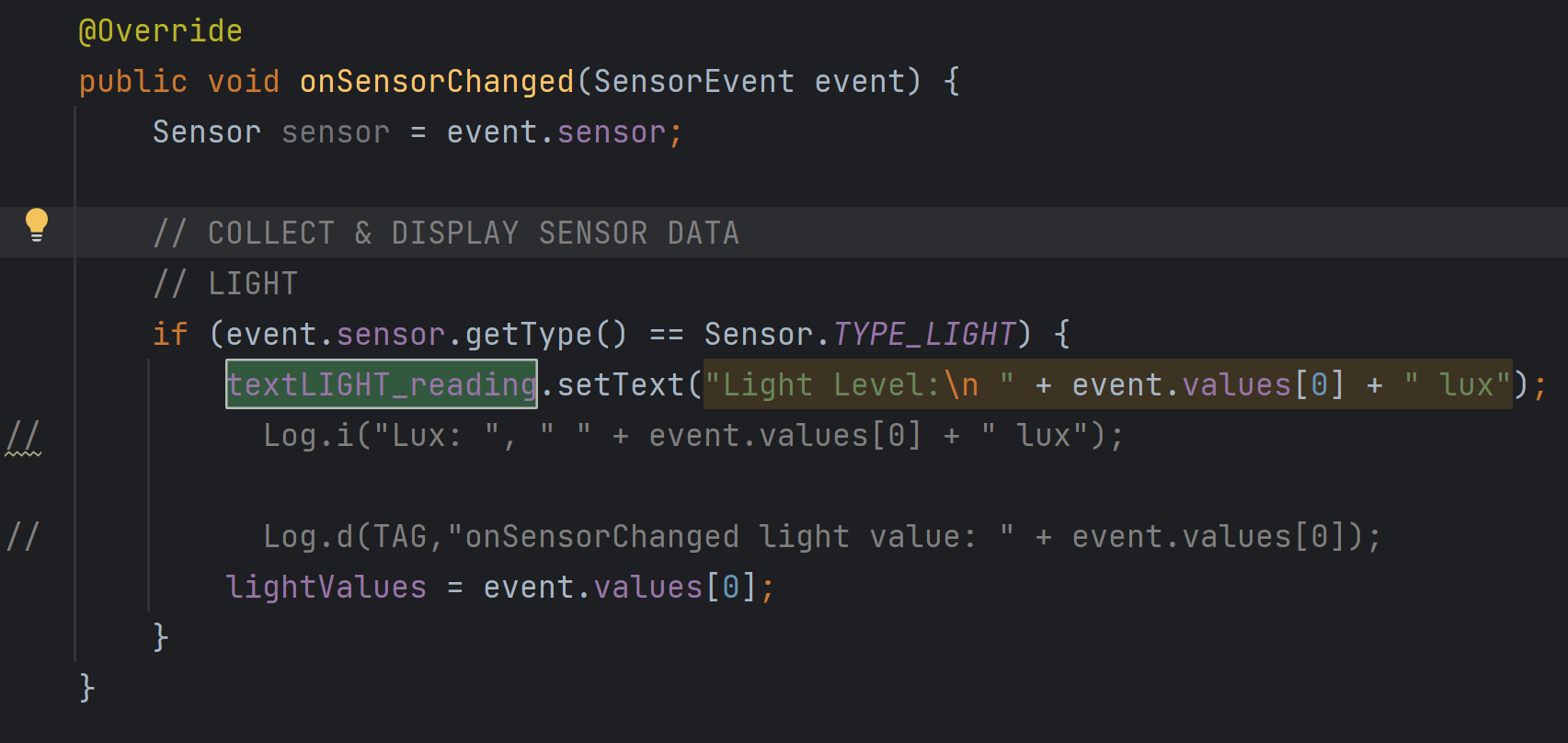
METHODOLOGY

## **Application** Implementation**:**

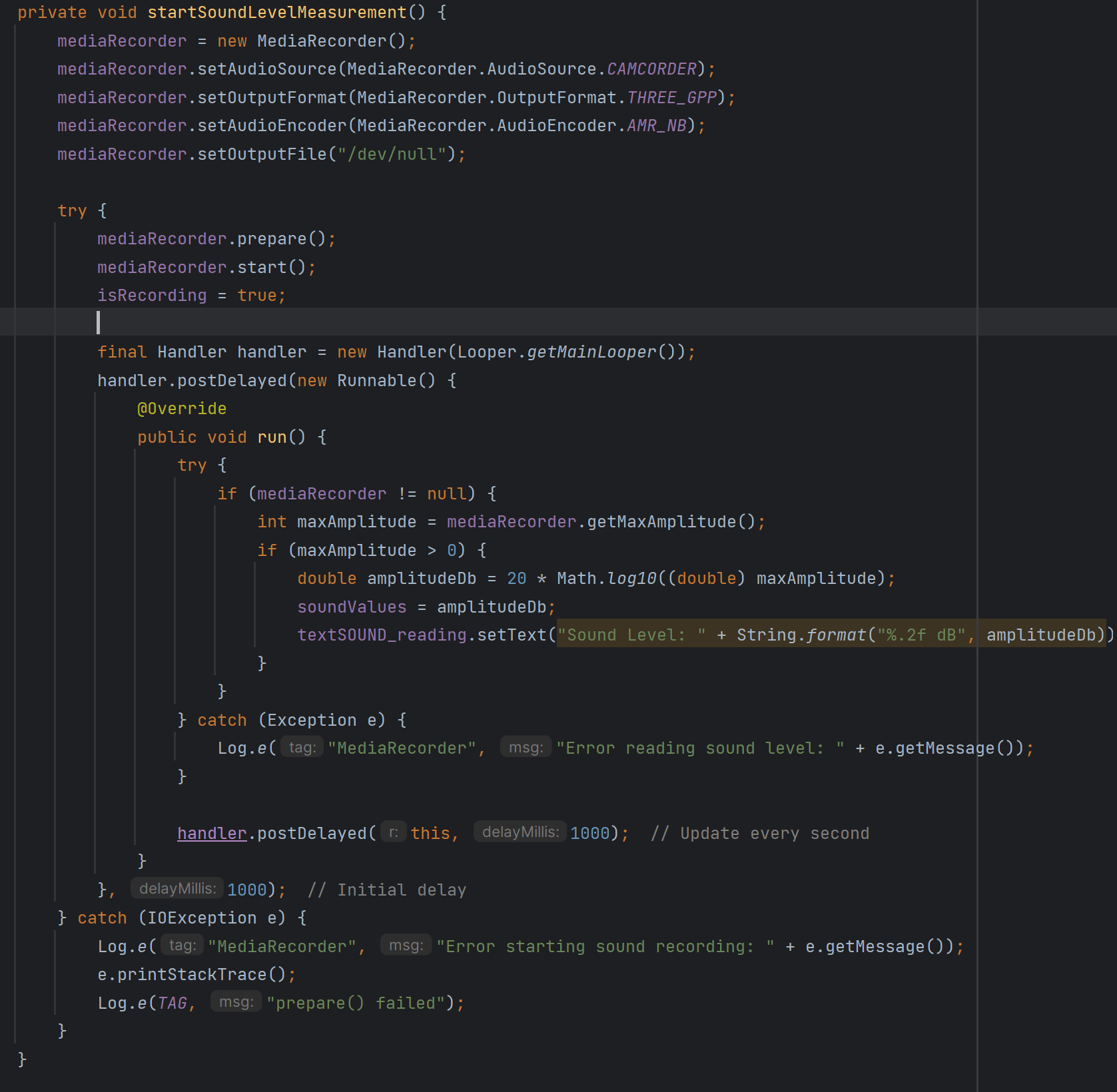
First things first we need to be able to collect and Upload the Data to a shared Cloud Space. This Android application, part of the IoT Team Project, is designed to collect various types of sensor data from a user's device and upload them to a Firebase database for further analysis. The app focuses on collecting light and sound levels and capturing images.

## Features

**\*\*Light Level Measurement\*\***: Detects the ambient light level using the device's light sensor.



**\*\*Sound Level Measurement\*\***: Measures the sound level using the device's microphone.



**\*\*Image Capture\*\***: Allows the user to take pictures and upload them to Firebase Storage.



**\*\*Data Upload\*\***: Light and sound data are uploaded to Firebase Firestore in real-time.

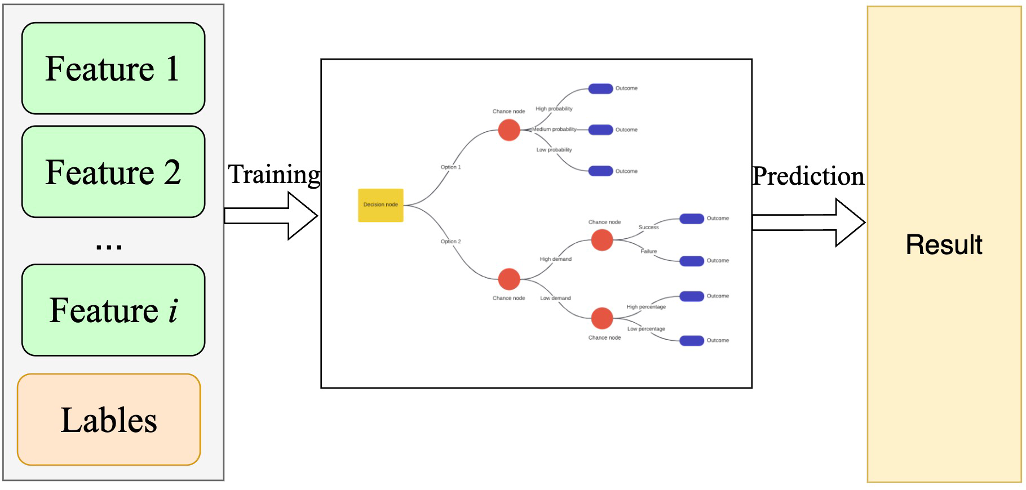
**\*\*Dynamic Data Collection\*\***: Users can start and stop data collection as needed



## 

## Proposed algorithms/methods:

To achieve accurate real-time prediction of the fountain's status, we initiated the process by preprocessing the data. We matched the environmental light data, ambient sound data, and timestamps of the number of people to derive the corresponding light, sound, and people data for each moment. Subsequently, we created a dataset, considering each moment as an entity with three features: light, sound, and people. Following that, we divided the dataset, allocating 70% as the training set for decision tree model training and 30% as the testing set for model validation. Finally, we applied the decision tree model to predict the fountain's overall status, fountain light status, and fountain music status. The framework diagram for the decision tree model's prediction of the fountain's status is illustrated in Figure 7, and the following details our proposed methodology.



### Figure 7. Schematic diagram of decision tree model to predict fountain status

**1. Data preprocessing**

The environmental light, ambient sound, and people data collected by sensors are stored in Firestore, and each type of data is timestamped. In order to accurately predict the status at each moment, it is necessary to match the light, sound, and people data based on the same timestamp.

**2. Creating the Dataset**

After implementing the functionality to match light, sound, and people data based on timestamps in the data preprocessing phase, we proceed to create the dataset. As illustrated in Figure 1, each timestamp serves as an entity in the dataset, with the light, sound, and people for each timestamp serving as the entity's features. For example, Feature 1 represents light, Feature 2 represents sound, Feature 3 represents the number of people, and Feature 4 represents the timestamp. Since the decision tree is a supervised model, labels need to be designed for each entity. Specifically, the labels for fountain light include levels 1, 2, 3, and off; for fountain music, the labels include levels 1, 2, 3, and off; and for fountain status, the labels include off, level 1, level 2, and level 3.

Regarding the labels for fountain light status, as shown in Table 1, if the number of people is 0 and it is during working hours (8 AM - 8 PM), we want the fountain lights to be off because it is daytime, and no one is around the fountain. If it is non-working hours (8 PM - 8 AM), even if there are no people, we do not want the fountain lights to be off. If the current environmental light is weak (e.g., detected light is less than 400 lux), we want the fountain lights to be at level 3 (the highest level) as it needs to provide sufficient illumination. If the current environmental light is strong (e.g., greater than 700 lux), we want the fountain lights to be at level 1 (the minimum illuminable level). In other cases, we prefer the fountain lights to be at level 2.

For fountain music status labels, as shown in Table 2, we want the fountain music to be at a higher level (level 3) when the environmental sound is loud (e.g., greater than 70 dB). This is because in a noisy environment, higher music volume is needed for a better music experience. The reverse is true for lower environmental sound levels.

Regarding fountain status labels, as shown in Table 3, we want the fountain to be off when the number of people is 0. For less than 2 people, less than 5 people, and greater than 5 people, the fountain status should be levels 1, 2, and 3, respectively. The threshold values for these labels are based on the optimal divisions in our simulation environment, considering a total of 4 participants in our experiment conducted in an office setting. If the simulation environment were an actual fountain, there might be more suitable division intervals. It is worth noting that different division intervals do not affect the overall proposed framework.

**Table 1 Fountain Light Status Labels**

| **People** | **Environmental Light (lux)** | **Daytime** | **Labels** |
| --- | --- | --- | --- |
| 0 | Any | Yes | 0 |
| 0 | Any | No | >=1 |
| >0 | [0,400] | Any | 3 |
| >0 | [400, 700] | Any | 2 |
| >0 | >=700 | Any | 1 |

**Table 2 Fountain Sound Status Labels**

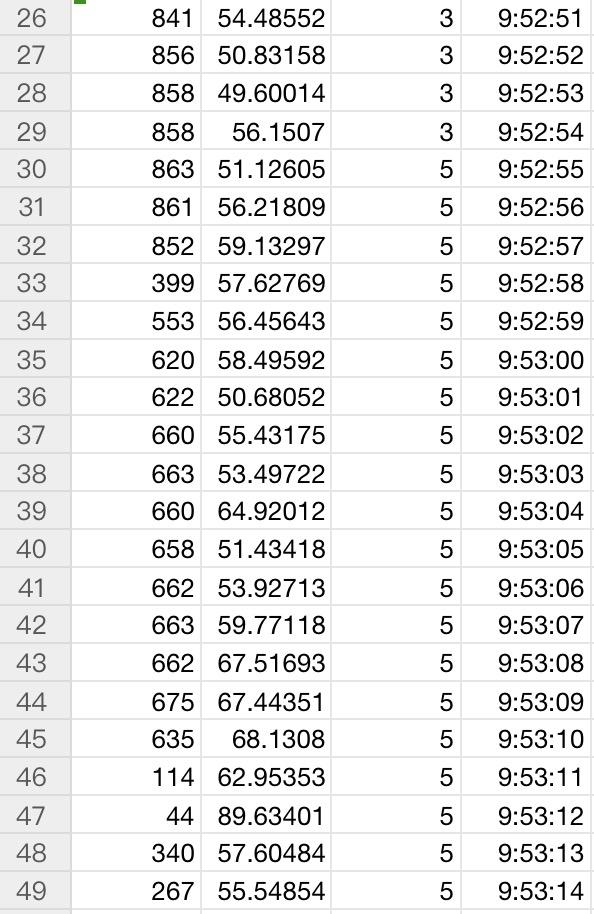
| **People** | **Environmental Sound (dB)** | **Labels** |
| --- | --- | --- |
| 0 | Any | 0 |
| >0 | [0,50] | 1 |
| >0 | [50, 70] | 2 |
| >0 | >=70 | 3 |

**Table 3 Fountain Status labels**

| **People** | **Labels** |
| --- | --- |
| 0 | 0 |
| [1,2] | 1 |
| [2,5] | 2 |
| >=5 | 3 |

**3. Dataset Splitting and Model Training**

To train the decision tree model, we collected a total of 1111 data points, each labeled with a corresponding label. The dataset is represented in Figure 8, where each row represents an entity, and each column corresponds to the collected light data, sound data, recognized number of people, and timestamp. After creating the dataset, we proceeded to split it, allocating 70% as the training set and 30% as the testing set.



### Figure 8. Dataset format

Since a single decision tree can only predict one target, such as predicting sound status or light status, we employed three independent decision tree models. These models were dedicated to predicting the status of light, music, and the fountain, respectively.

4. Validate the Model

Utilizing 30% of the dataset as the testing set, assess the effectiveness of the model on an independent dataset.

5. Deploy the Model

Identify the model with the best predictive performance on the testing set and save its model parameters locally. Subsequently, deploy it to the service layer as part of the digital twin space for real-time prediction of the physical space, encompassing the light status, music status, and the overall status of the fountain.

## Methodology for head counting:

Use Yolo and BCC (Bayesian Crowd Counting) for small and big numbers of people respectively. For every input image, do pre-processing and then forward these two models and choose the more trusted one as the input to the decision tree. Models are pre-trained using datasets published online with a larger volume, and further refined using images we collected.

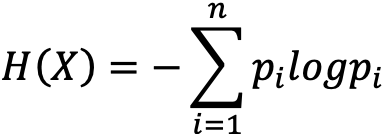
## **Adopted mathematical models:**

**Decision tree**

A decision tree is a machine learning model commonly used for classification tasks. It represents decision rules in a tree-like structure and is employed to make predictions on input data. The construction of a decision tree model is based on strategies for partitioning the training data, aimed at minimizing uncertainty or entropy. In this context, the decision tree model involves concepts such as information entropy, information gain, and the information gain ratio. The following provides a detailed description of these three concepts to enhance a better understanding of the decision outcomes in decision tree models.

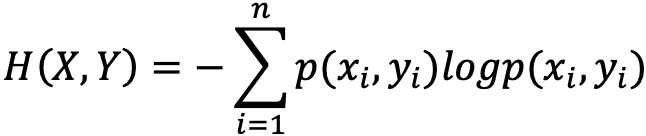
**Information entropy**

Entropy is a concept in information theory that measures the uncertainty of a system. The greater the uncertainty of a phenomenon, the higher its entropy. When the probabilities of events are equal, the randomness of their occurrences is maximal, resulting in higher entropy. The expression for the entropy of a random variable *X*, also known as empirical entropy, is as follows:

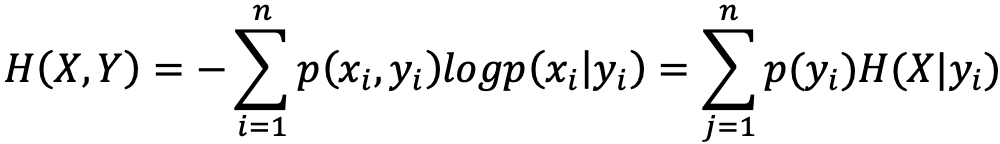


where n represents the number of different discrete values that X can take. The symbol pi represents the probability of X taking the value i.

The following are the joint entropy expressions for two variables *X* and *Y*:



The following is the expression for conditional entropy:



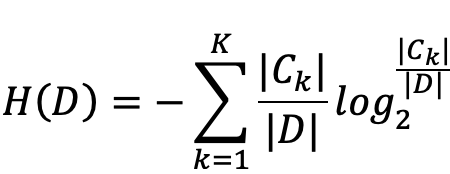
where *H*(*X*) measures the uncertainty of *X*, and the conditional entropy *H*(*X*| *Y*) measures the remaining uncertainty of *X* after knowing *Y*. The *H*(*X*)-*H*(*X*| *Y*) quantifies the reduction in uncertainty of *X* after knowing *Y*. This measure is known as mutual information in information theory, denoted as *I*(*X*, *Y*), and in decision tree algorithms, it is referred to as information gain. The decision tree (ID3 algorithm) uses information gain to determine which feature to use at the current node for constructing the decision tree. A higher information gain for a particular feature indicates that the feature is more suitable for reducing the uncertainty in classifying the dataset.

**The calculation of information gain**

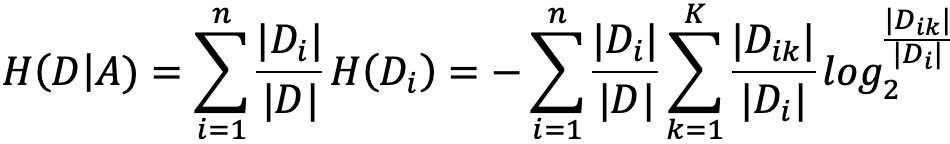
Input: Training set *D* and feature *A*

Output: Information gain *g*(*D*, *A*) of feature *A* for training set *D*

(1) Calculate the empirical entropy *H*(*D*) of *D*, where |*Ck*| represents the number of samples in the *k*-th category:



(2) Calculate the empirical conditional entropy *H*(*D*| *A*)of feature *A* for dataset *D*:



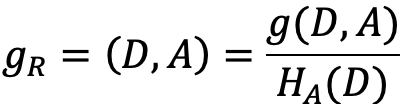
(3) Calculate information gain:



Information gain is not suitable for handling continuous features, such as the intensity of sound and light. Therefore, the next step is to introduce information gain ratio. Decision trees using information gain ratio can effectively handle datasets with continuous features.

**Information gain ratio**

The information gain ratio of feature *A* for the training dataset *D*, denoted as *gR*(*D*, *A*), is defined as the ratio of its information gain *g*(*D*, *A*) to the entropy *HA*(*D*) of the values of feature *A* in the training dataset *D*. This is expressed as follows:



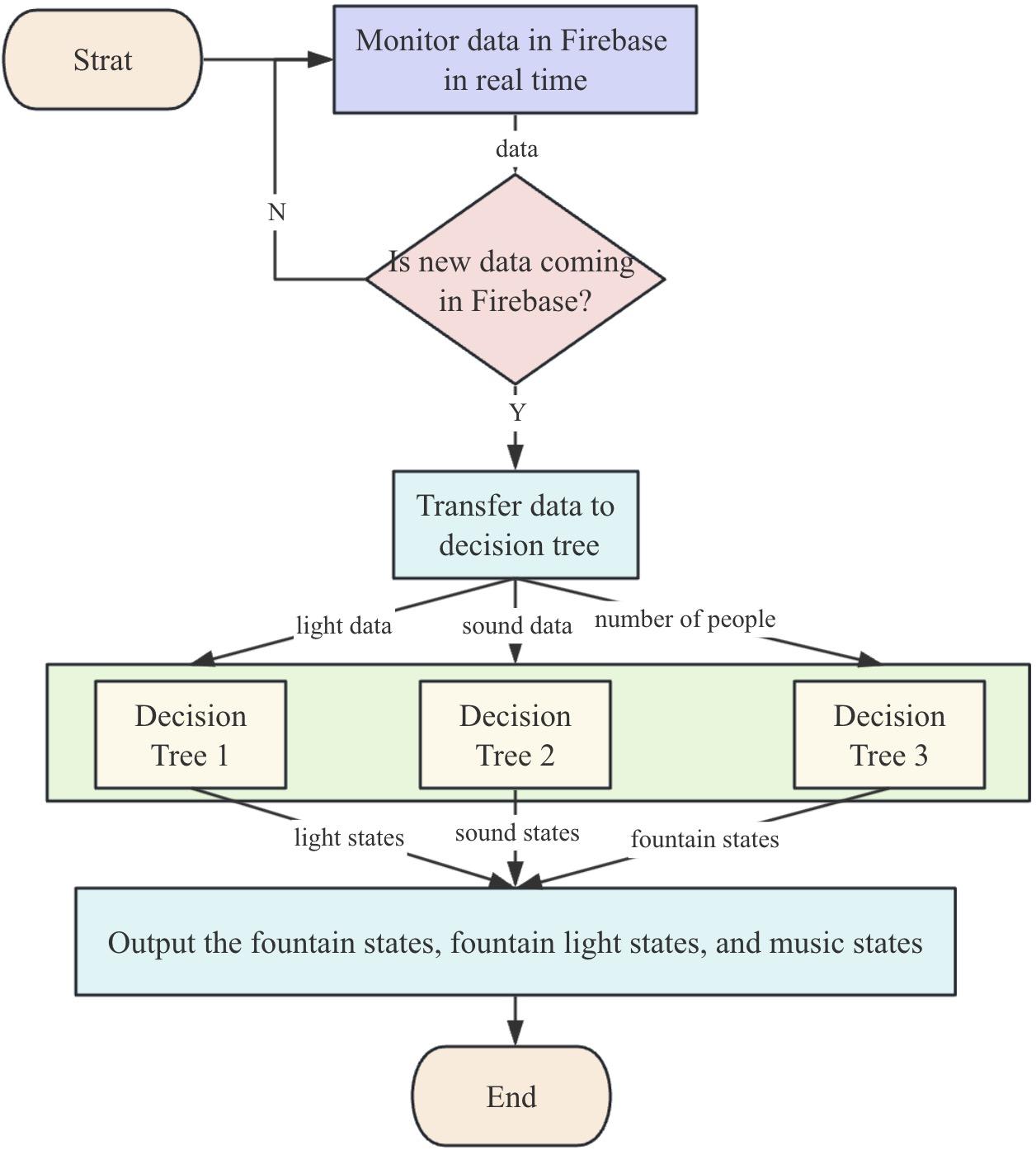
where *n* is the number of categories for feature *A*, |*Di*| represents the number of samples in the *i*-th category, and |*D*| is the total number of samples.

## 

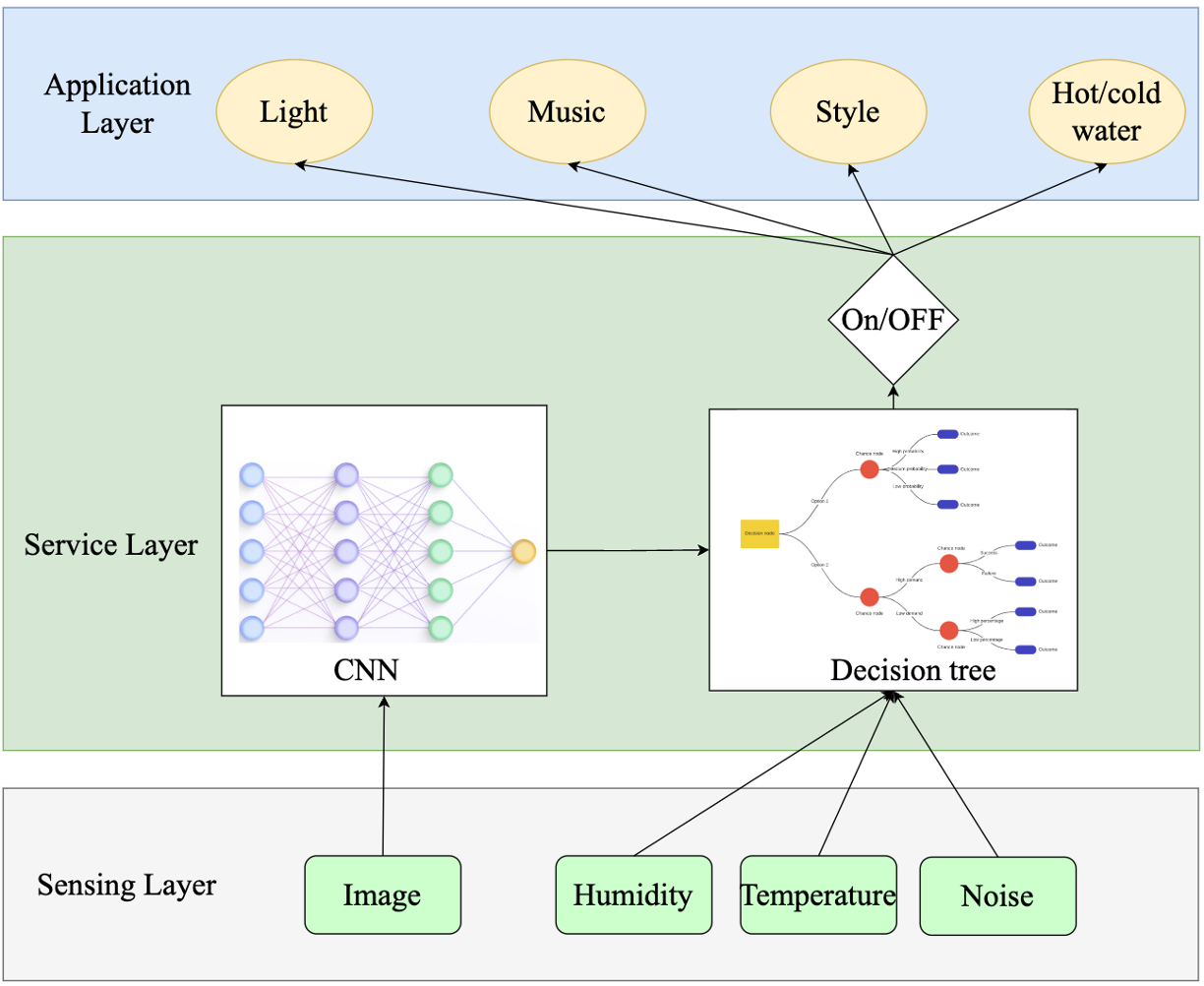
## **Flow chart, system architecture/components:**

**Decision-making process**

Figure 9 illustrates the decision process within the digital twin space. Specifically, the system first monitors real-time data in Firebase. If the sensors send new data to Firebase, the system captures and transfers this data to the trained decision tree models. Decision Tree 1 receives real-time light and people data, predicting the status of the fountain lights. Decision Tree 2 receives real-time sound and people data, predicting the status of the fountain music. Decision Tree 3 receives real-time people data, predicting the overall status of the fountain itself. The entire process is real-time, enabling the simulation and prediction of three states of the fountain: lights, music, and the fountain's general status.



**Figure 9. Decision tree prediction process**



**Figure 10. Overall framework**

## **Justifications and analysis, etc.:**

**A decision tree model is used**

In this project, it is necessary to predict the status of the fountain, such as whether it needs to be turned off, based on the environmental light data, ambient sound data, and the number of people collected by sensors. This is a classification task, and theoretically, it could be implemented simply using an 'If-else' statement in any programming language. However, I did not opt for this approach as it would be too straightforward for me, and such a method cannot capture the non-linear relationships within the data. Instead, I employed an advanced machine learning technique commonly used in the field of data science — the decision tree model. During the training process, the decision tree model selects features with the maximum predictive power for the target variable by evaluating metrics such as information gain ratio. This makes the decision tree highly effective when dealing with the dataset in this project.

EVALUATION

## Settings for experiments and demo:

Setting project\_id = "IOT-2" indicates the connection to a Firebase project named 'IOT-2'. Setting relative\_credentials\_path = "./iot-3-70517-firebase-adminsdk-5dieu-09196e6f14.json" is used to provide the credentials for connecting to Firebase.

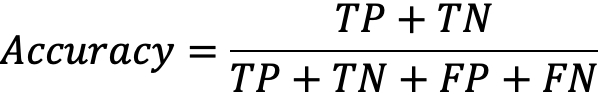
## Settings for Demo:

The Setting for the demo was a controlled environment, which was one of the rooms in the Computer Science Department. We used the room light control to simulate the change of the ambient light in the room. We used our own voices to change the sound levels of the environment, and we used other fellow students to walk into the frame to simulate the crowd surrounding the Digital Fountain.

## Experiment results:

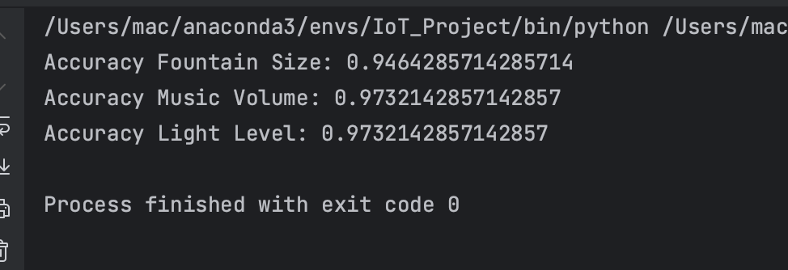
**Validate Decision Tree Models**

We utilized 70% of the dataset as the training set and the remaining 30% as the testing set to assess the accuracy of the models. The model accuracy is calculated as follows:



where TP stands for True Positives, TN stands for True Negatives, FP stands for False Positives, and FN stands for False Negatives.

Figure 11 illustrates the accuracy of the three decision tree models, achieving accuracies of 0.946, 0.973, and 0.973, respectively.



**Figure 11. the accuracy of the three decision tree models**

## Observations and inferences:

LEARNING OUTCOMES

## Tasks accomplished by each team member:

**Dong Yang:**

I am responsible for designing the decision tree model in the team. Firstly, I implemented the real-time retrieval of data collected by Omar and stored it in Firestore. Subsequently, I downloaded historical data from Firestore and the people detection data designed by Huaiyuan Chu to the local environment. I created labels for each data point, which served as the dataset for the decision tree model. Then, I trained and validated the model using the dataset, saving the best model parameters. Finally, I implemented real-time monitoring of new data in Firestore, combined with real-time people count data, and used the trained decision tree model for live classification predictions.

**Huaiyuan Chu:**

Pre-process the data in Firebase. For light and sound sensor data, pass them to a low-pass filter to avoid abrupt noises (e.g. sudden high signal in sound data). For images, crop them to a fixed size to feed into the neural network.

For head counting, first try to use YOLO (you only look once), which is a well-known real-time detection algorithm with pre-trained model parameters. Another BCC (Bayesian Crowd Counting) model is also used when the people in the input image form a crowd and YOLO can not detect properly. The latter one is doing estimation rather than detection, but can still reveal the crowd trend. For the same input, two models above will give their results and I will pick the more trusted one between them as one input to the decision tree.

**Omar Madjitov:**

As was mainly responsible for Mobile app development and Data Collection. I have managed to develop an Android application that is capable of using the sensors and collecting data in real-time. At the same time, I have also learned and managed to configure a cloud-based database database, namely Firestore and Storage. Additionally, I have learned to collect and push Image, Light, and Sound Data to a Cloud Database in real-time

## 

## Reasons for unaccomplished tasks or any failures:

We should use our collected images for training the head counting neural network, but time is so limited and the image resolution collected from Firebase is not high enough for training. So we just used pre-trained models published online and refined it used our collected images.

Another task that we weren’t able to accomplish was to collect Humidity and Temperature Data. The reason for this issue is the fact that the only means to collect the sensor data, that our team was provided with were Smartphones. Although Smartphones have a wide variety of sensors, they do not have sensors to collect Ambient Humidity and Ambient temperature. Given the time constraints that all of the teams had for this project, it wasn’t possible for our team to reconfigure the project, so we decided to move forward with whichever Sensor Data we were able to collect.

## Lessons learned from the project and teamwork experience:

This project employs machine learning (or deep learning) methods for real-time simulation prediction, deepening my understanding of the relationship between machine learning, digital twins, and the Internet of Things (IoT). Through the design and development of this project, I learned the process of collecting data from sensors to Firestore and then designing corresponding machine learning models for simulation prediction based on specific tasks. The project has provided me with a profound understanding of the four layers of the IoT model. The Sensing layer involves real-time collection of physical space data using sensors, which represents the data in the IoT. Deploying machine learning models to cloud servers constitutes the Network layer, where the cloud server facilitates the provision of data obtained from the lower layer to the upper layer. The machine learning model serves as the Service layer for the entire system, responsible for providing users with real-time intelligent decisions. Finally, the machine learning model outputs prediction results to users, serving as the Application layer in the system.

Effective communication within a team can greatly expedite the progress. In the early stages of the project, due to a lack of timely discussions, there were inconsistencies in understanding certain details among team members. For instance, there was a disagreement on when the decision tree should predict that the fountain lights should be set to the maximum. Eventually, we collectively agreed that if there are no people around the fountain and it is daytime, the decision tree should predict that the fountain lights be turned off. If it is nighttime, the fountain lights should not be turned off, or at least set to level 1. Implementing this change required adjustments to the dataset, and the decision tree had to relearn this feature, resulting in some unnecessary time consumption.

DEMO VIDEO LINK:

**REAL-TIME DECISION MAKING:**

## <https://drive.google.com/file/d/1DP8gGsZmwsuPr01A3SGgs-C1DYw7mZfE/view?usp=drive_link>

**LIGHT CONTROL:**

**https://drive.google.com/file/d/1gVhZmIv-HmKp4Xiw11\_cOpQf8kqPW2u9/view?usp=drive\_link**