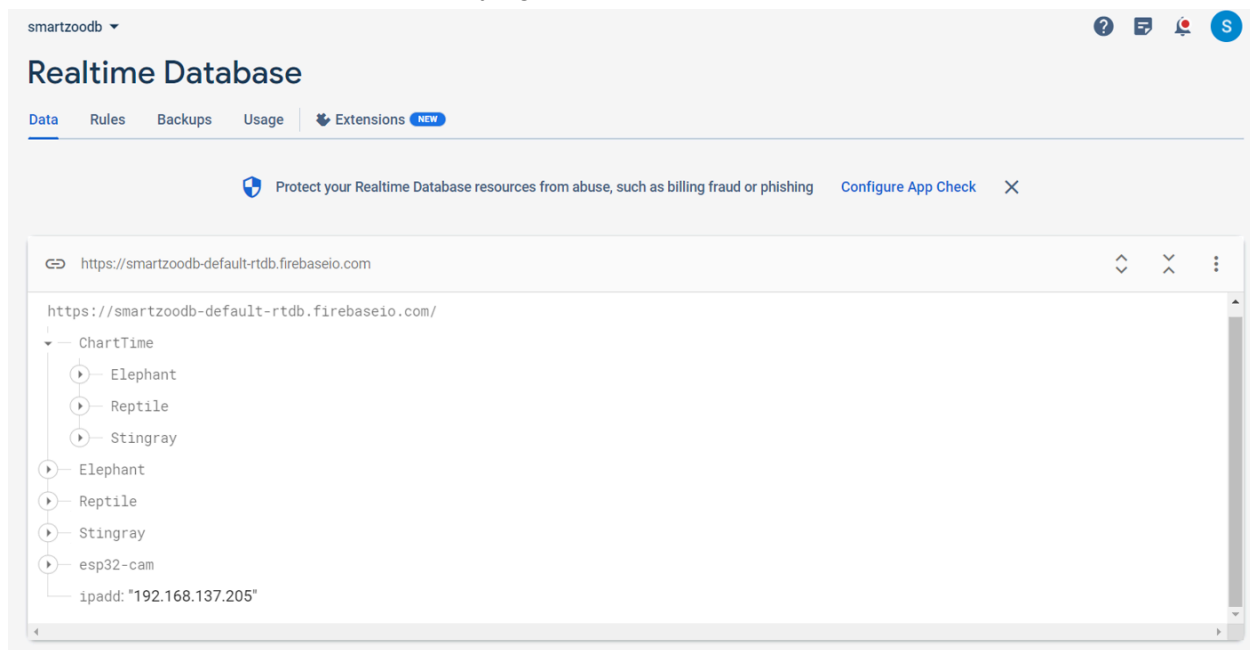


**Firestore Overview:** Our group has decided to change the AWS database cloud system to the Firestore cloud system instead. This decision was made to make cloud programming easier since AWS requires SQL database coding language to work with, whereas Firestore doesn't and is more object-oriented. We named our cloud "smartzoodb" and have been using it to analyze data collected from our microcontroller and sensor and store the data. Data collected will be stored in our cloud up until the last two days, however, after two days it will be deleted to utilize cloud storage. These are collecting data from our temperature, Humidity, Ph-14, and Oxidation sensor. We are also displaying the current time and date.



Our Firestore Real-Time Database has the structure shown in the figure above.

**Firestore Realtime Database:** In the above figure we can see the node called "ChartTime". This node stores sensor readings corresponding to the time they were taken. We have 30-minute intervals between collecting data so it is stored in 30-minute intervals at points such as: "1330", "1400", "1430"... and so on. This data was ordered like this in the ChartTime node so it can be easily pulled from the website, so we can show that data in a chart which gives users another way to interpret the data collected by our sensors.

In the above figure, we can see under the ChartTime node that we have the Elephant, Reptile, and Stingray nodes. These nodes also contain sensor readings for the respective animal and it contains the 48 previous values of the sensors. The main reason we need this and didn't include it in ChartTime node is the user can refresh and request data so we wanted to keep that data but also didn't want to disrupt our 30-minute interval system in the ChartTime node. All the data that is being collected has its path to the sensors in these 3 paths. For example, if the stingray module was updating the temperature it would insert the new data at the path "Stingray/Temperature/Value1" where value 1 holds the latest temperature. Then the cloud

functions take over and shift the data so “Value3” becomes “Value2”, “Value4” becomes “Value3” and so on till “Value48” becomes “Value47”.

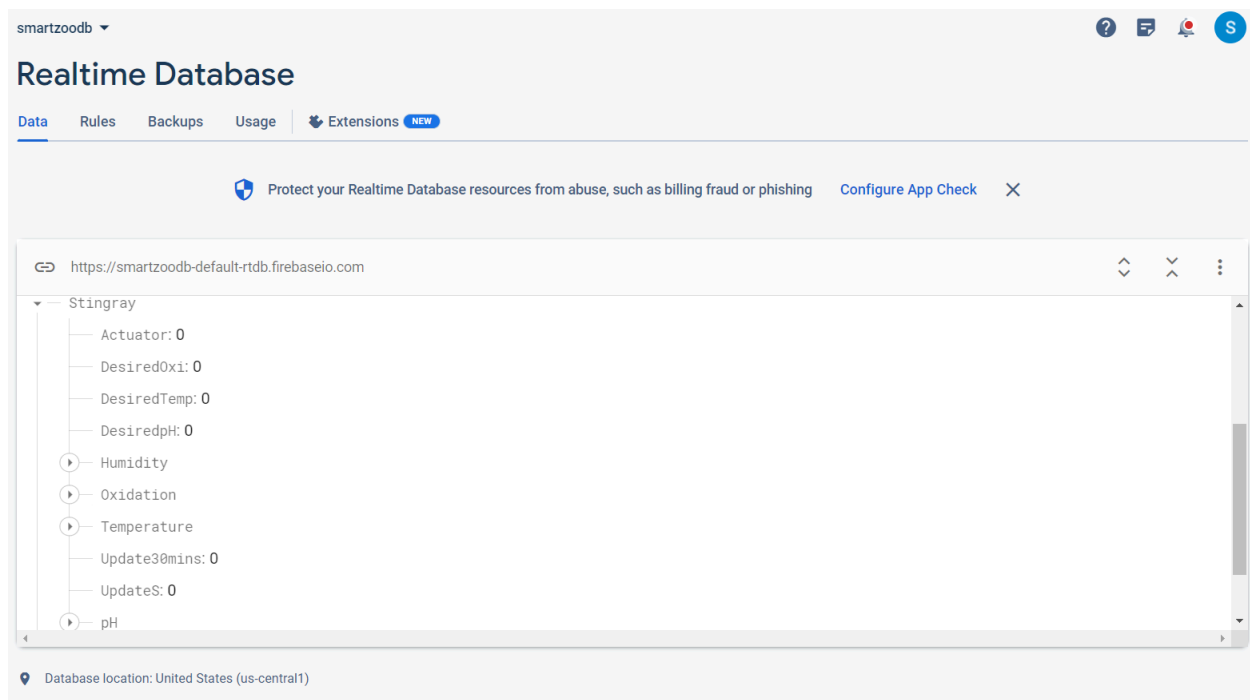


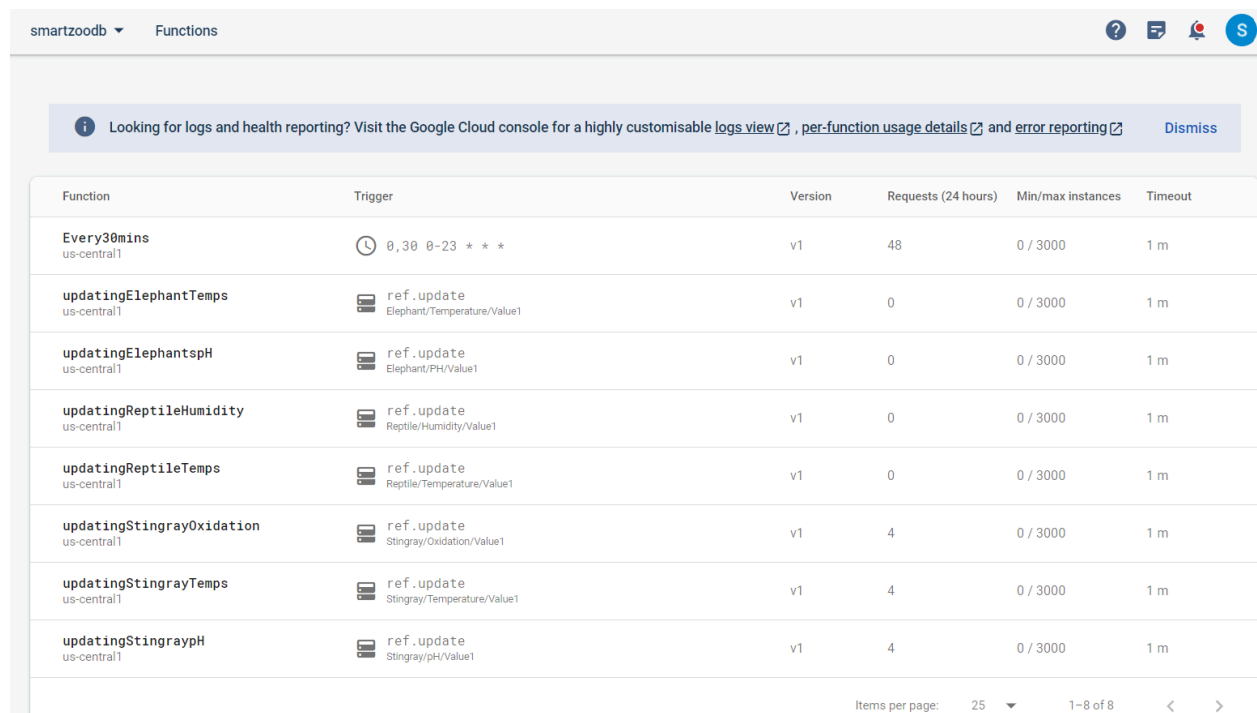
Figure shows whats inside of an animals's nodes

In the above figure, we have opened the stingray node and we can see all of its child nodes. The “Actuator”, “DesiredOxi”, “DesiredTemp”, and “DesiredpH” are related to the actuator part of the project. The “Humidity”, “Oxidation”, “Temperature”, and “pH” nodes contain the past 48 measurements of the respective sensor. The “Update30mins” is set to 1 every 30 minutes by the “Every30mins” cloud function. The arduino reads this and sends the latest measurements and then the arduino sets this back to 0 again. The “UpdateS” node is connected to the refresh button on the website and pressing that button will set it to 1. The arduino will read this and send the current data into the database and it will set the “UpdateS” to 0. The rest of the animal nodes are similar to this as well.

Finally we have the “esp32-cam” and “ipadd” nodes which hold information about our camera module. Our camera’s IP address is always changing so we have the node “ipadd” to hold the camera’s latest IP address. The “ipadd” node is used by the website which reads the IP address of the camera and uses it to display the camera’s live feed on the website.

**Firebase Cloud Functions:** The cloud functions were a important part of this project because it took burden off the arduino microcontrollers which provides us with more battery life since it has to compute less things. At first, we had the microcontroller insert and shift data around in the real time database. We measured this and used to take about 3 minutes for the arduino to insert and shift the data correctly in the database. This is mainly due to the fact that it had to send 48

read requests and 48 write requests to the database for each sensor. We then employed cloud functions into our project and saw that the same task now takes 10-15 seconds.



The screenshot shows the Google Cloud Functions console for the project 'smartzoodb'. At the top, there's a navigation bar with 'smartzoodb' and 'Functions'. Below this is a blue banner with a message: 'Looking for logs and health reporting? Visit the Google Cloud console for a highly customisable [logs view](#), [per-function usage details](#) and [error reporting](#)'. Below the banner is a table listing the functions. The table has columns: Function, Trigger, Version, Requests (24 hours), Min/max instances, and Timeout. The functions listed are: 'Every30mins' (cron trigger, 48 requests), 'updatingElephantTemps' (ref.update trigger, 0 requests), 'updatingElephantpH' (ref.update trigger, 0 requests), 'updatingReptileHumidity' (ref.update trigger, 0 requests), 'updatingReptileTemps' (ref.update trigger, 0 requests), 'updatingStingrayOxidation' (ref.update trigger, 4 requests), 'updatingStingrayTemps' (ref.update trigger, 4 requests), and 'updatingStingraypH' (ref.update trigger, 4 requests). At the bottom right, there's a pagination control showing 'Items per page: 25' and '1-8 of 8'.

| Function                                 | Trigger                                   | Version | Requests (24 hours) | Min/max instances | Timeout |
|--|---|---------|---------------------|-------------------|---------|
| Every30mins<br>us-central1               | 0,30 0-23 * * *                           | v1      | 48                  | 0 / 3000          | 1 m     |
| updatingElephantTemps<br>us-central1     | ref.update<br>Elephant/Temperature/Value1 | v1      | 0                   | 0 / 3000          | 1 m     |
| updatingElephantpH<br>us-central1        | ref.update<br>Elephant/pH/Value1          | v1      | 0                   | 0 / 3000          | 1 m     |
| updatingReptileHumidity<br>us-central1   | ref.update<br>Reptile/Humidity/Value1     | v1      | 0                   | 0 / 3000          | 1 m     |
| updatingReptileTemps<br>us-central1      | ref.update<br>Reptile/Temperature/Value1  | v1      | 0                   | 0 / 3000          | 1 m     |
| updatingStingrayOxidation<br>us-central1 | ref.update<br>Stingray/Oxidation/Value1   | v1      | 4                   | 0 / 3000          | 1 m     |
| updatingStingrayTemps<br>us-central1     | ref.update<br>Stingray/Temperature/Value1 | v1      | 4                   | 0 / 3000          | 1 m     |
| updatingStingraypH<br>us-central1        | ref.update<br>Stingray/pH/Value1          | v1      | 4                   | 0 / 3000          | 1 m     |

Figure shows all the cloud functions

The cloud functions are written in javascript, and most of the cloud functions are update functions and each sensor for each animal has its own cloud function. These update functions execute when data is inserted into the respective sensors path in the database. Once data is inserted, the cloud function shifts data. The shifting can be thought of as a First in First out (FIFO) datastructure where inserting a element will push an element into the front of a queue or list. At the same time, the element at the end of the list or queue is popped and removed which is essentially what this cloud function does. Also this update function also inserts this data into the correct time slot in the "ChartTime" node in the database. This update function only occurs every 30 minutes which is controlled by another cloud function called "Every30mins".