Remote Dog Chip Reader

Project ID: 5821

Students: Mahmoud Sheikh Khalil
Jamal Tannous

Supervisor: Boaz Mizrachi

The Prof. Israel Cederbaum
High Speed Digital Systems Laboratory
HS DSL
המעבדה למערכות ספרתיות מהירות
ע"ש פרופ' ישראל צדרבאום

The Andrew & Erna Viterbi
Faculty of Electrical Engineering

Background

- **Most interfaces that deal with pet chip readers are hard to operate.**
- **They are relatively expensive.**
- **Available chip readers do not show sufficient and helpful information when scanning the pet.**

Background

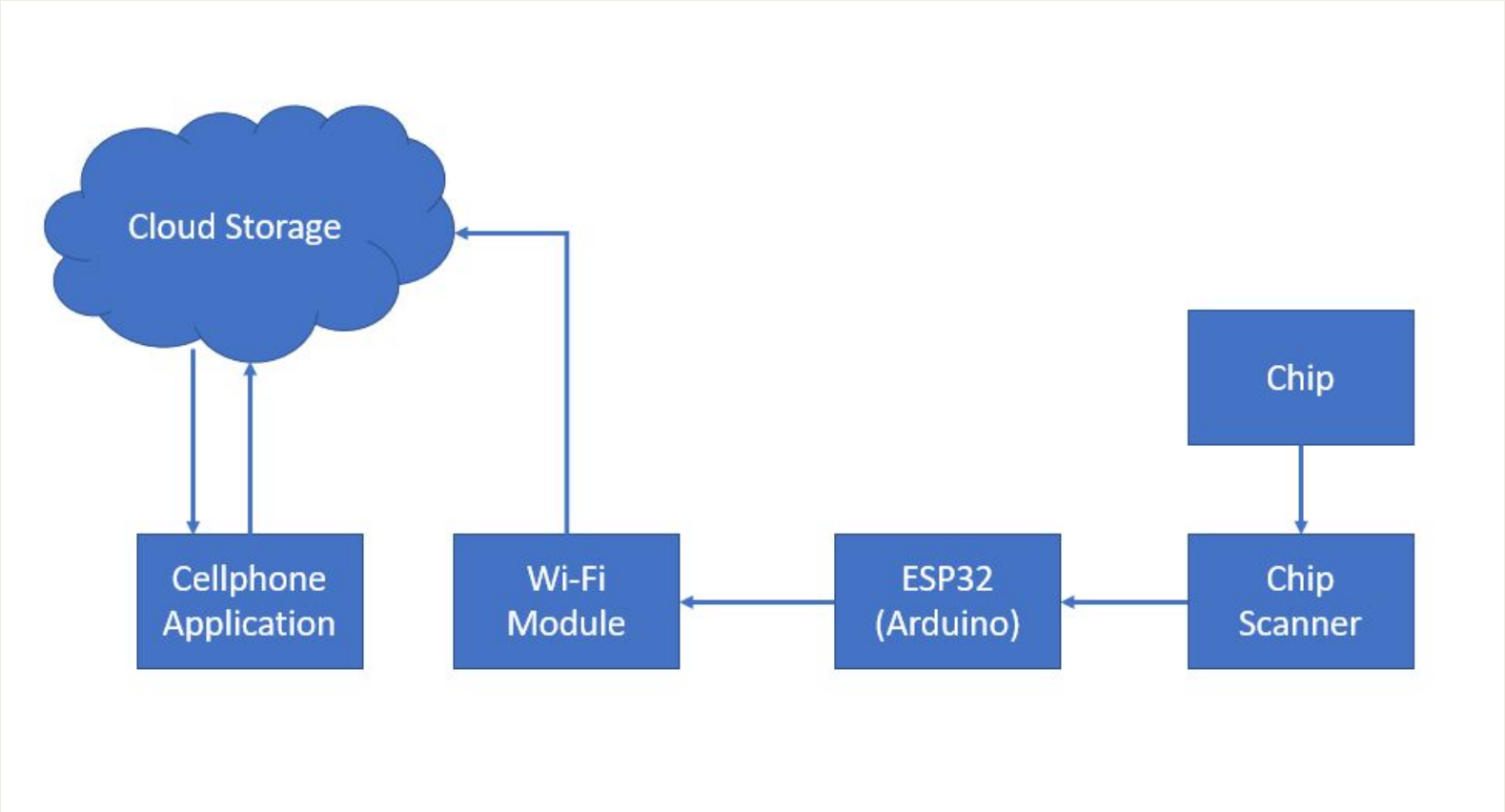
- **In order to get sufficient information about the pet, the user must search for this info manually.**

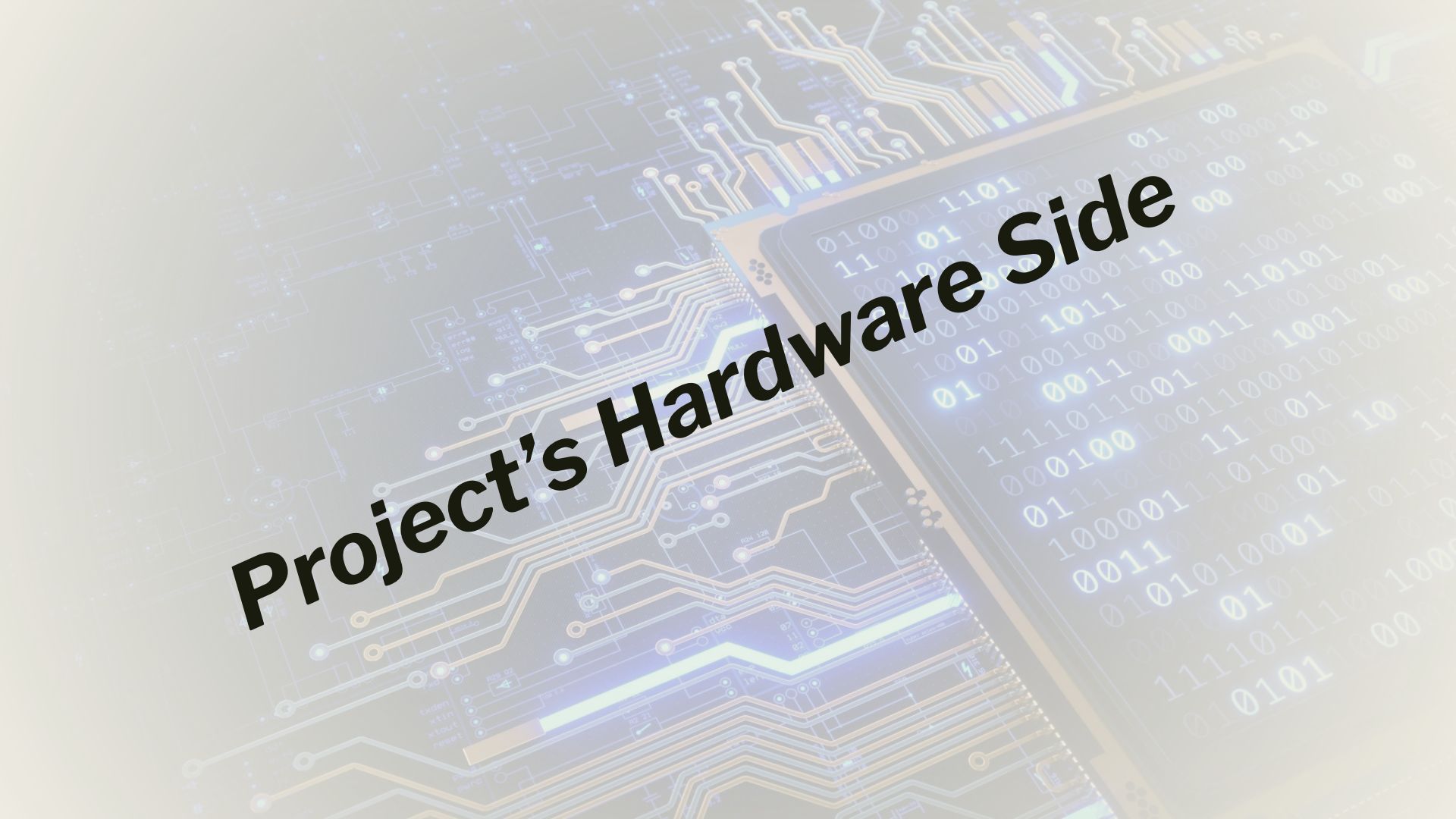
Project Goals

Our device should have the following characteristics:

- **Chip reader which is compact and battery powered.**
- **Scanning the pet should be straightforward and intuitive.**
- **Display the info to the user using an App or a Website.**

BLOCK DIAGRAM

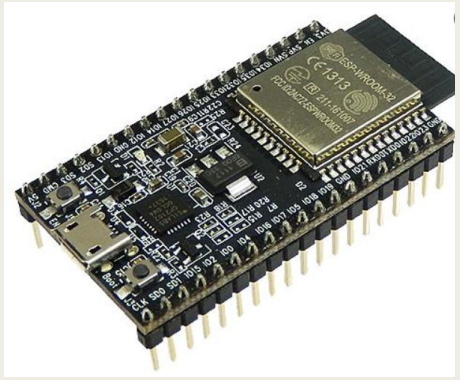



Project's Hardware Side

Hardware Components

ESP32

- **Relatively low cost.**
- **Does not consume a lot of power.**
- **Power Supply: External 3.7V**
- **Sends the desired data to the cloud storage through the integrated Wifi module.**



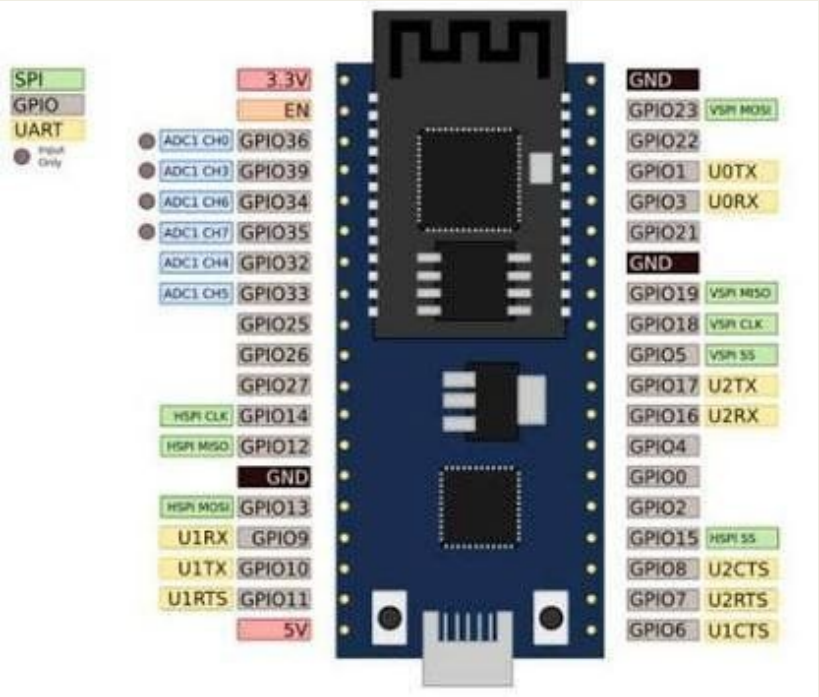
Hardware Components

RDM6300 (EM4100 RFID Reader for ESP32)

- **Reads the number on the microchip and sends it to the Arduino card.**
- **It is low cost.**
- **Does not consume a lot of power.**
- **Power Supply: External 5V**
- **Less than 100ms decoding time**

Some Hardware Schematics

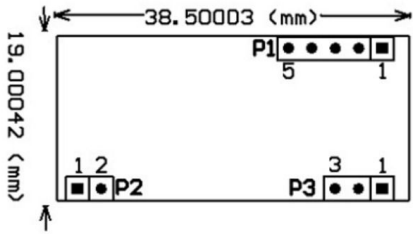
ESP32



Some Hardware Schematics

RDM6300 (EM4100 RFID Reader for ESP32)

Pinout:



2. Pin definition（TTL interface RS232 data format）:

P1:

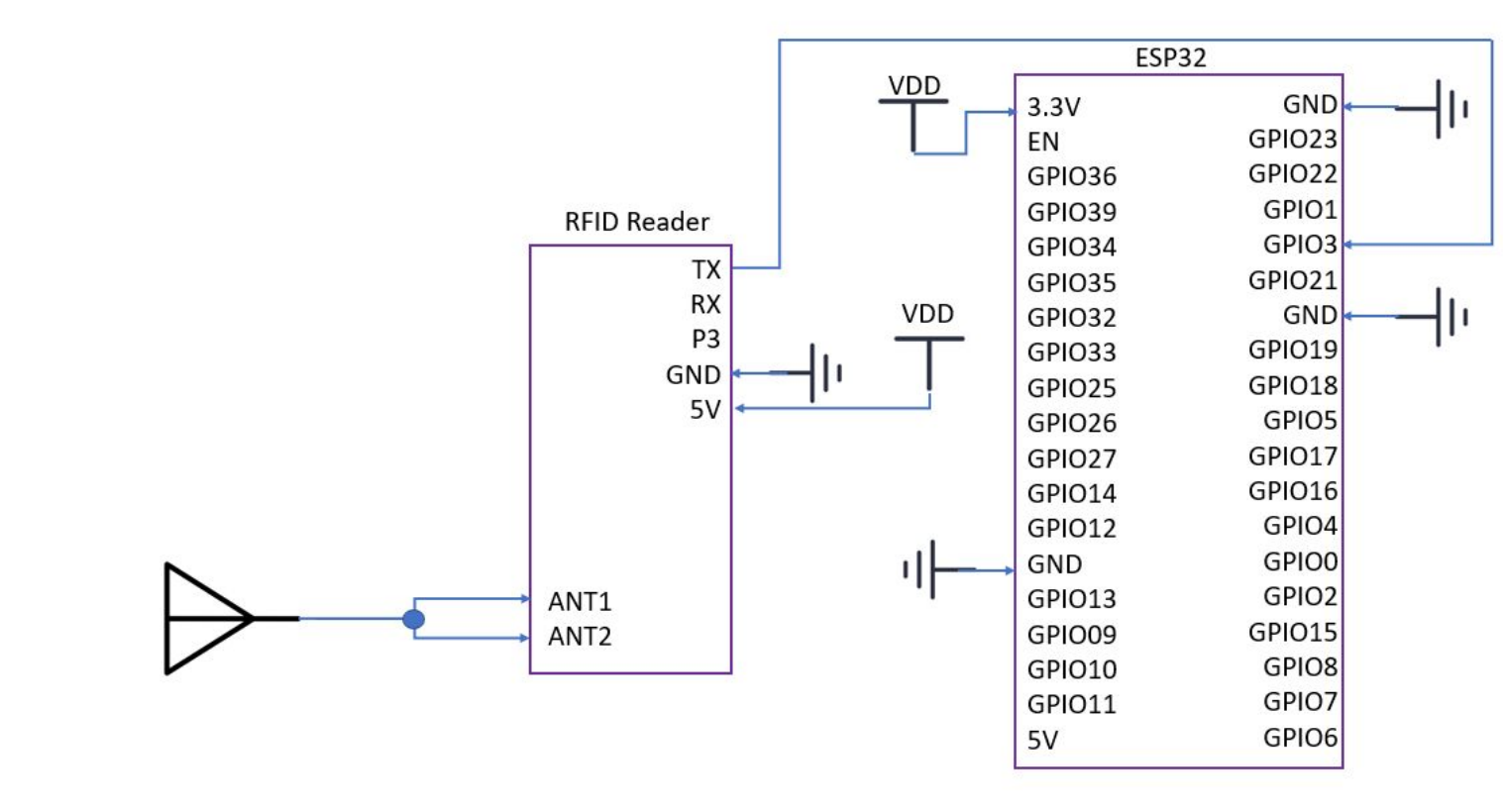
PIN1	TX
PIN2	RX
PIN3	
PIN4	GND
PIN5	+5V(DC)

P2:

PIN1	ANT1
PIN2	ANT2

Some Hardware Schematics

Connecting Hardware Components




Project's Software Side

IDEs and Tools

- Arduino IDE
- Android Studio
- Firebase (A cloud storage toolset)

IDEs and Tools

Arduino IDE

We used Arduino IDE to program the hardware part of the project.
Used some special libraries:

- “Wifi.h” - to connect the ESP32 to the Wifi.
- “FirebaseESP32.h”- to connect to Firebase’s realtime database.
- “rdm6300.h”- to connect to the chip reader module.

IDEs and Tools

Arduino Space: Rdm6300 class

The class called Rdm6300 implements the chip scanner program.

As we can see from the code:

The communication protocol is UART.

```
class Rdm6300
{
    public:
        void begin(Stream *stream);
        void begin(int rx_pin, uint8_t uart_nr=1);
        void end(void);
        bool update(void);
        uint32_t get_tag_id(void);
        bool is_tag_near(void);
#ifdef RDM6300_SOFTWARE_SERIAL
        void listen(void);
        bool is_listening(void);
#endif
    private:
#ifdef RDM6300_HARDWARE_SERIAL
        HardwareSerial *_hardware_serial = NULL;
#endif
#ifdef RDM6300_SOFTWARE_SERIAL
        SoftwareSerial *_software_serial = NULL;
#endif
        Stream *_stream = NULL;
        uint32_t _tag_id = 0;
        uint32_t _last_tag_id = 0;
        uint32_t _last_read_ms = 0;
};

#endif
```

IDEs and Tools

Arduino Space: update() method

The update method:

- Checks whether a data stream is available.
- Checks if the data format is correct.
- Checks using a checksum if the scanned data has errors.
- Checks if the available data is new.
- Returns the newly scanned data.

```
    /* if a new tag appears- return it */
    if (_last_tag_id != tag_id) {
        _last_tag_id = tag_id;
        _last_read_ms = 0;
    }
    /* if the old tag is still here set tag_id to zero */
    if (is_tag_near())
        tag_id = 0;
    _last_read_ms = millis();

    _tag_id = tag_id;
    return tag_id;
}
```

IDEs and Tools

Arduino Space: Main Loop

```
void loop() {
  if (Firebase.ready() && rdm6300.update()){
        currentId = rdm6300.get_tag_id();
        Serial.println(rdm6300.get_tag_id(), HEX);
        dtostrf(currentId,6,0,txString);
        Serial.printf("Set string... %s\n", Firebase.setString(fbdo, F("/scanner1"),txString ) ? "ok" : fbdo.errorReason().c_str());

    }

  delay(10);

}
```

In this loop the ESP32 checks for newly scanned data. If new data is available, then it sends it (along with other “metadata”) to the Realtime Database.

IDEs and Tools

Android Studio

We used Android Studio to program our application.

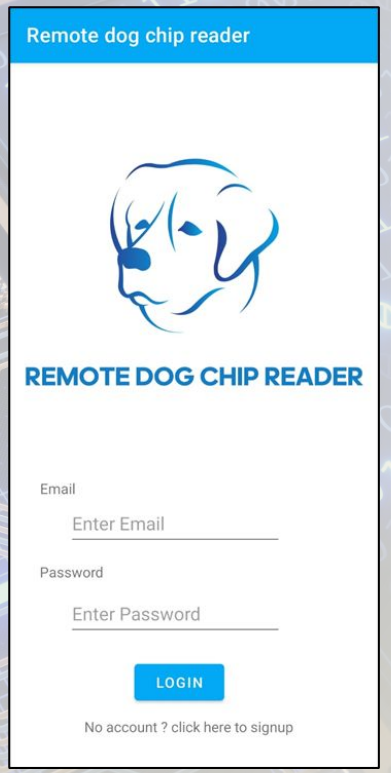
The application consists of 5 activities:

- Activity for user login.
- Activity for user signup.
- Activity for receiving scanned data.
- Activity for displaying the pet’s data.
- Activity for entering/editing scanned pet’s data.

IDEs and Tools

Screenshots from the Application

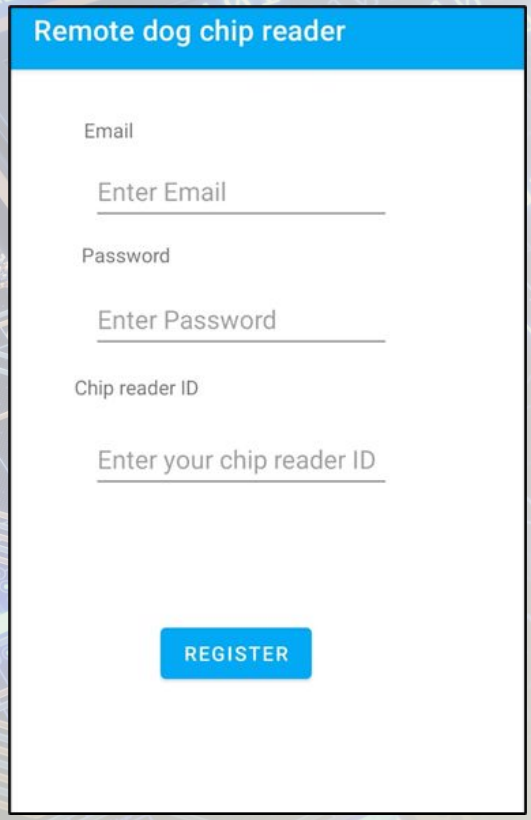
Activity for user login.



IDEs and Tools

Screenshots from the Application

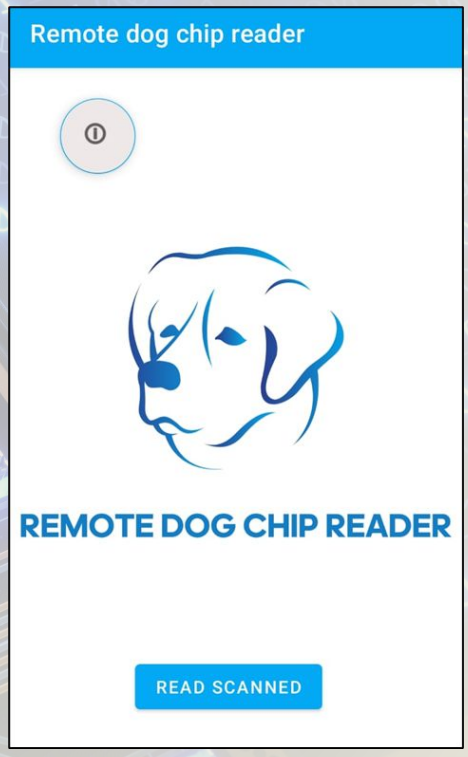
Activity for user signup.



IDEs and Tools

Screenshots from the Application

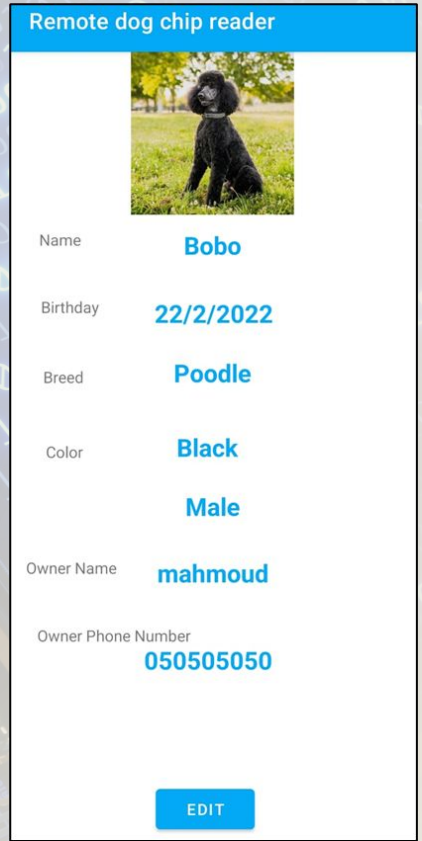
Activity for receiving scanned data.



IDEs and Tools

Screenshots from the Application

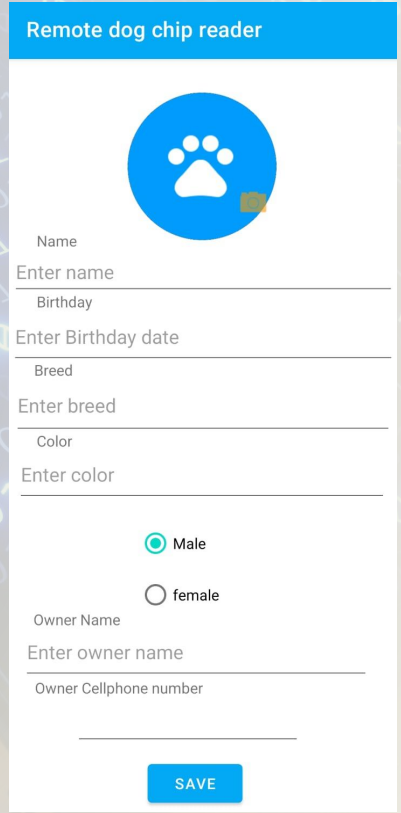
Activity for displaying the pet's data.



IDEs and Tools

Screenshots from the Application

Activity for entering newly scanned pet's data.



IDEs and Tools

Firebase

Firebase is a very popular toolset for managing cloud storage databases.

We used this toolset in order to store and modify the data of the users and the pets in our system.

IDEs and Tools

Firebase

From Firebase we used three types of databases:

- Realtime Database.
- 2 Instances of Cloud Firestore.
- Cloud Storage.

In the following slides we will shortly explain the use of those databases.

IDEs and Tools

Firebase

Firebase's Realtime Database: it is a cloud-hosted NoSQL database that lets you store and sync data between your users in real time.

Data is stored as Strings/JSON and synchronized in realtime to every connected client.

IDEs and Tools

Firebase

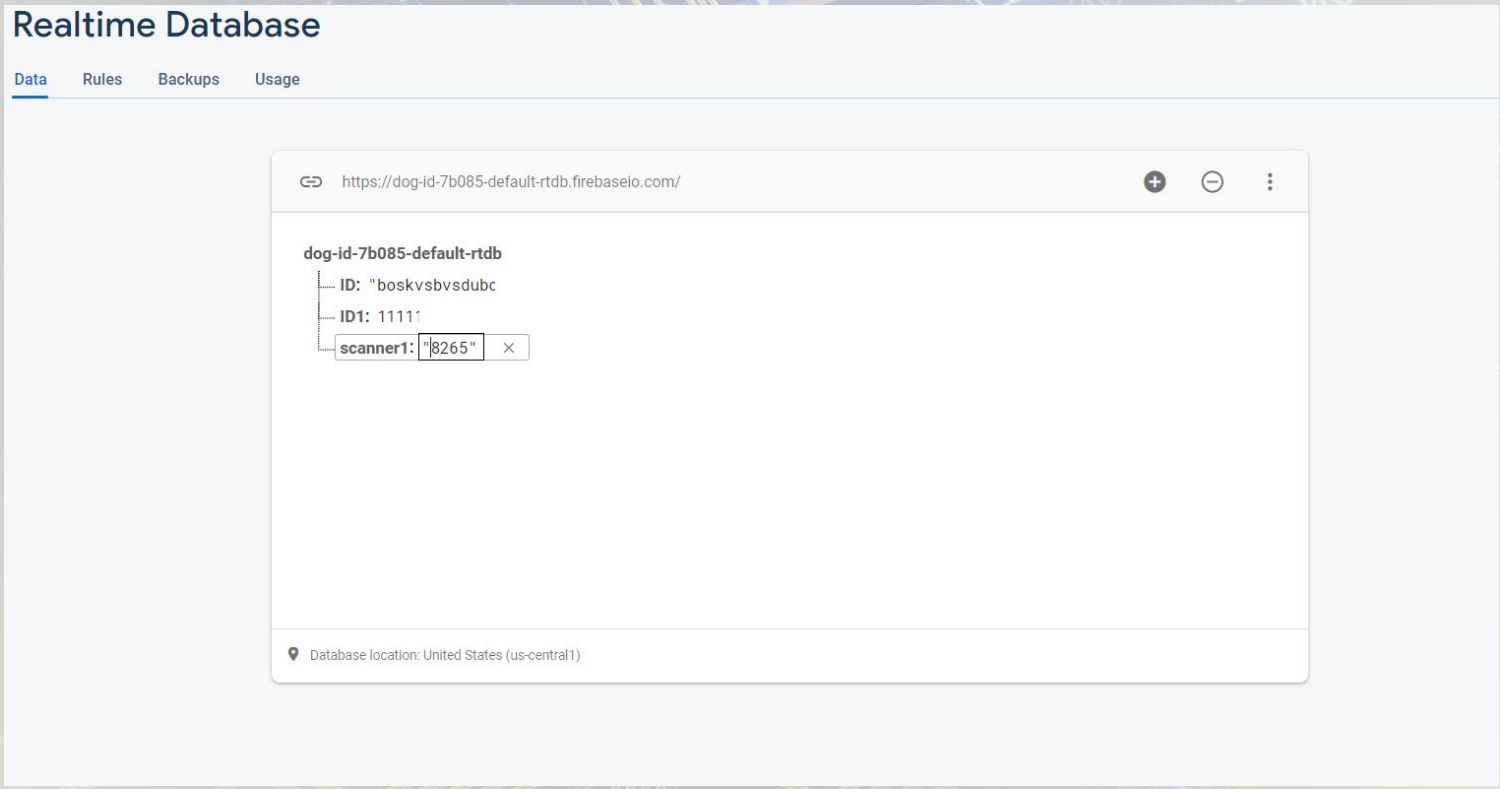
Firebase's Realtime Database: we used this database in the following cases:

- Connect to the ESP32 to receive the scanned data.
- Send information regarding the scanned data to the application.

This database gets updated within milliseconds after the scanning of the chip.

IDEs and Tools

Screenshot of our Realtime Database



IDEs and Tools

Firebase

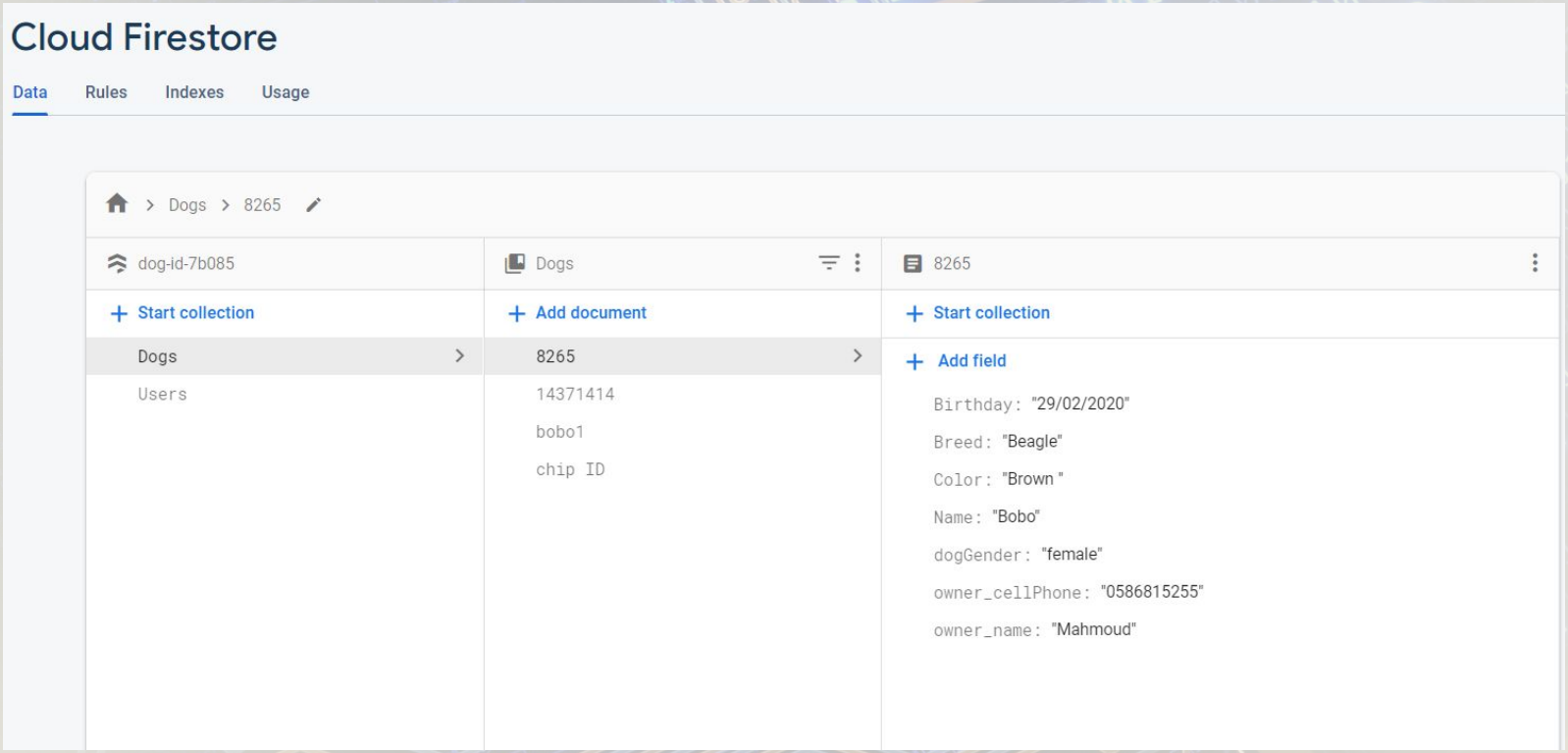
Firebase's Cloud Firestore: Cloud Firestore is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps at global scale.

We used two instance of Cloud Firestore in our project:

- Database to save the users of the system (Email + Password + Scanner ID)
- Database to save the pet's information (Name, Birthday…)

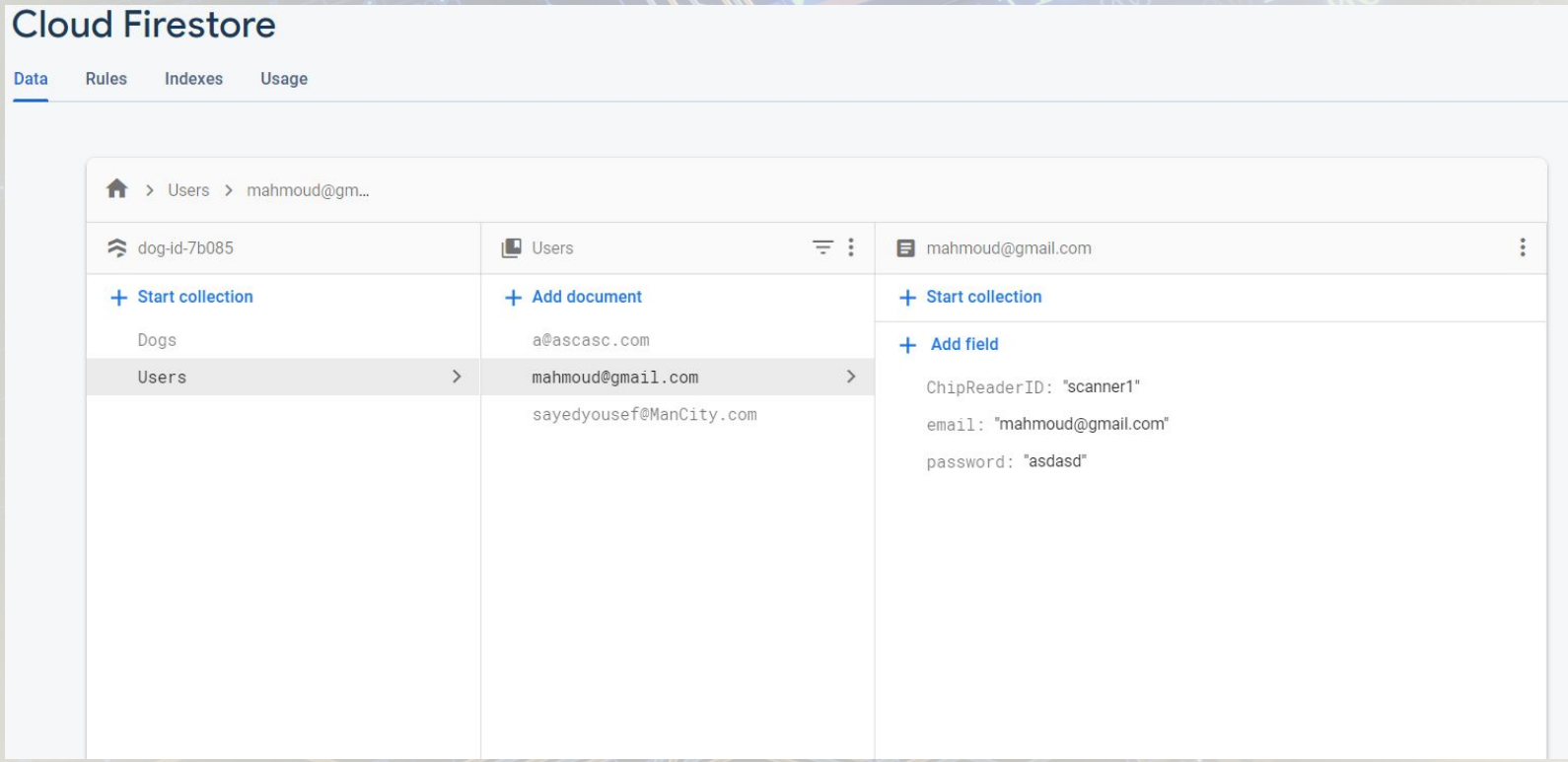
IDEs and Tools

Screenshot of our Cloud Firestore



IDEs and Tools

Screenshot of our Cloud Firestore



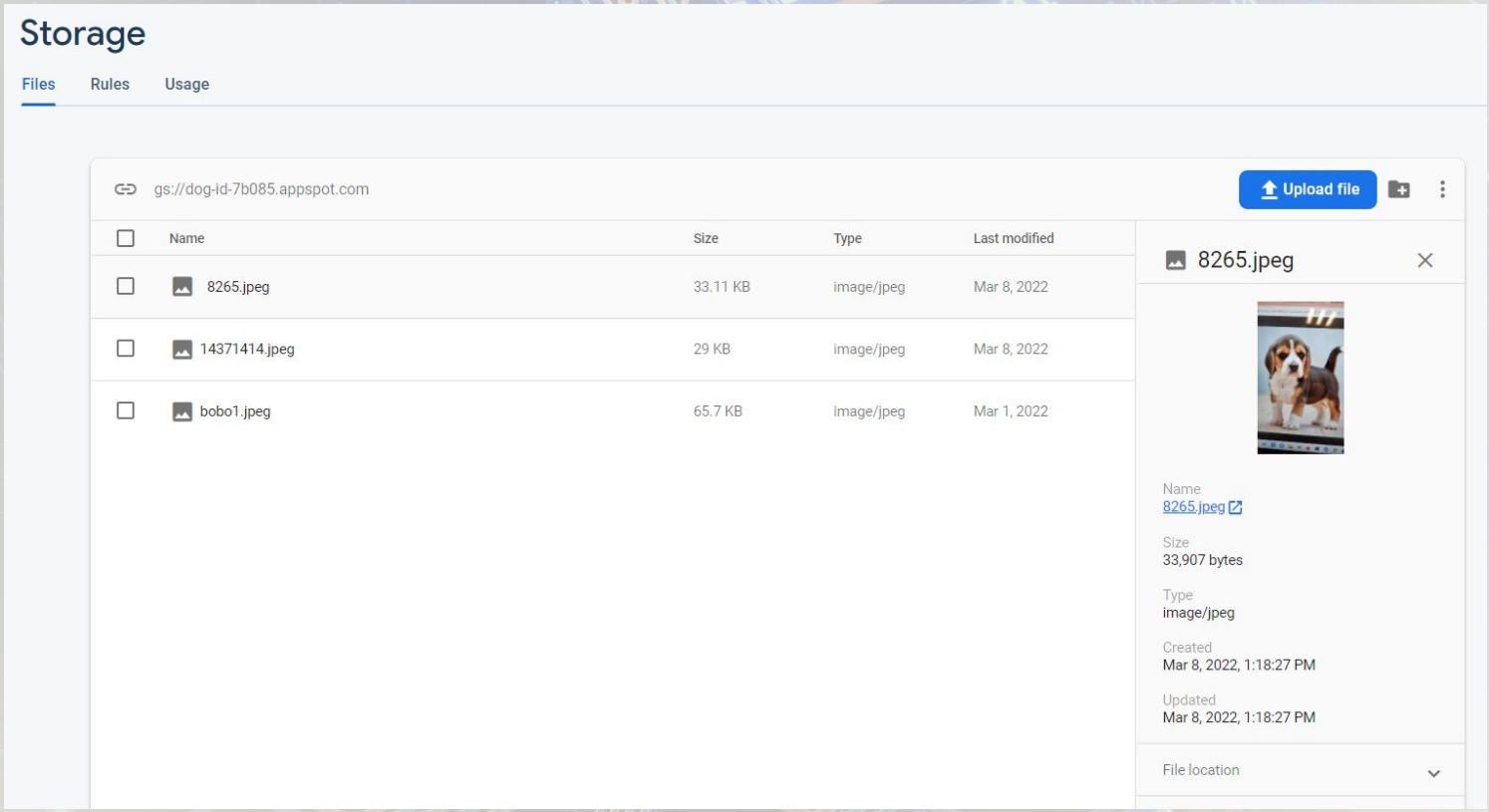
IDEs and Tools

Firebase

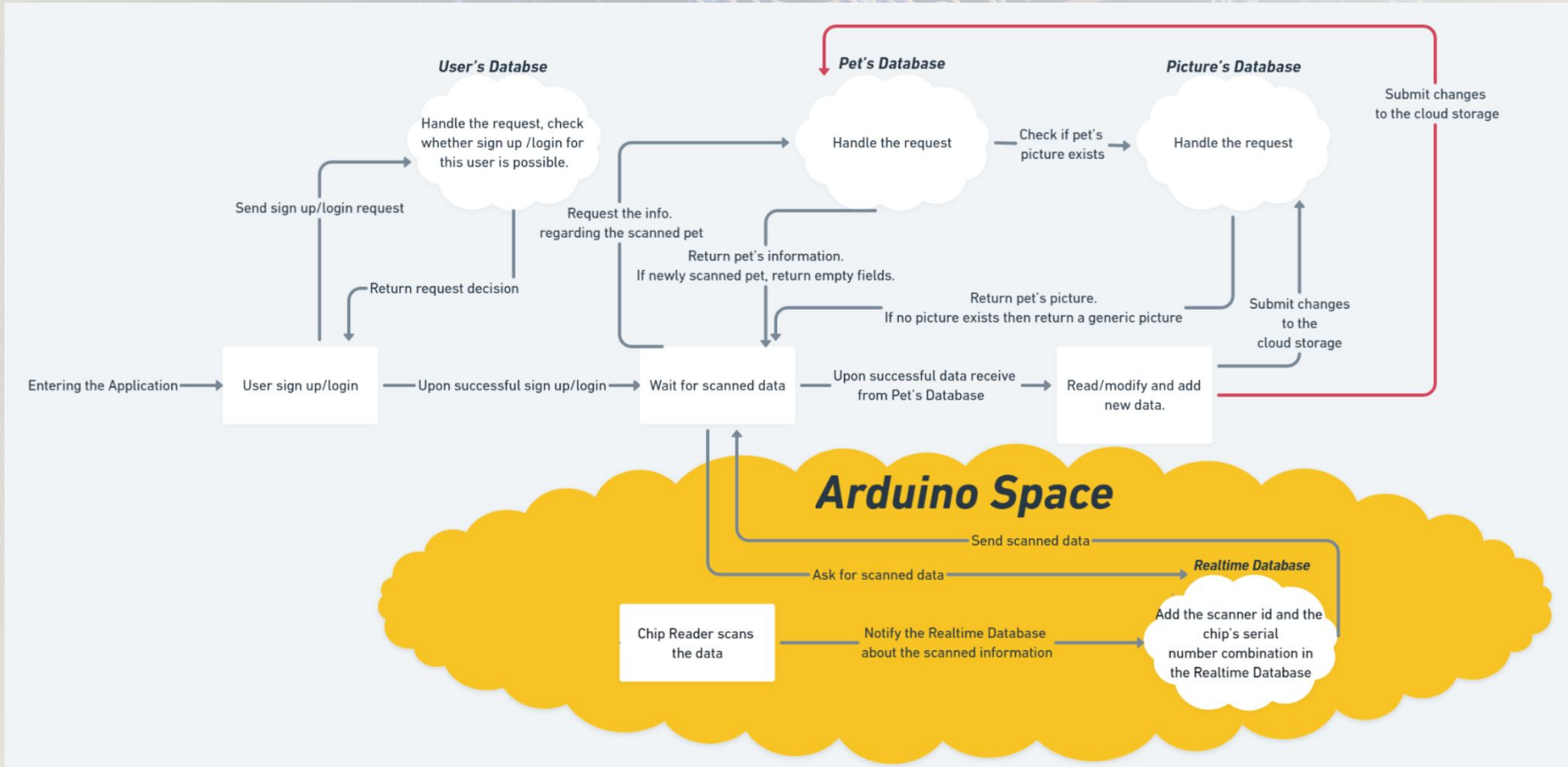
Firebase's Cloud Storage: Cloud Storage is designed to help you quickly and easily store and serve user-generated content, such as photos and videos.
The client can also easily retrieve that stored data using queries.
We used Cloud Firestore in our project in order to save the pictures of the pets whose informations are in the system.

IDEs and Tools

Screenshot of our Cloud Storage



All Encompassing Flowchart



Bill of Material

Quantity	Device	Description	Price
1	RDM6300 (EM4100 RFID Reader for ESP32)	Scan the RFID Microchip	7.3 ₪
1	ESP32 Arduino Card	Receive the scanned data and send it to the Realtime Database	12.75 ₪

Bill of Material

Realtime Database is free of charge, but with certain limitations

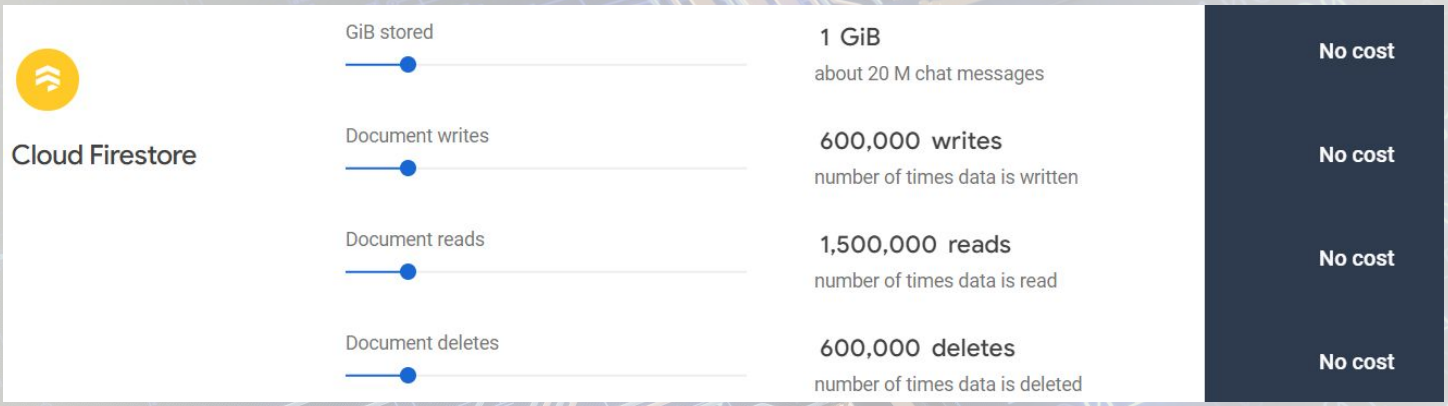
Realtime Database	GB stored	1 GB about 20 M chat messages	No cost
	GB transferred	10 GB about 200 M chat messages	No cost

Example of a long term plan (billed monthly)

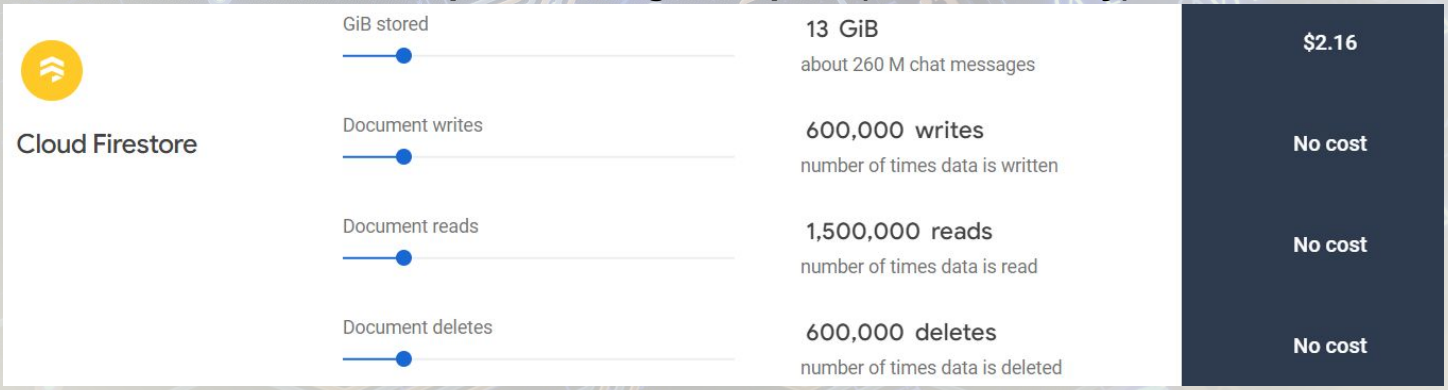
Realtime Database	GB stored	2 GB about 40 M chat messages	$5
	GB transferred	15 GB about 300 M chat messages	$5

Bill of Material

Cloud Firestore is free of charge, but with certain limitations

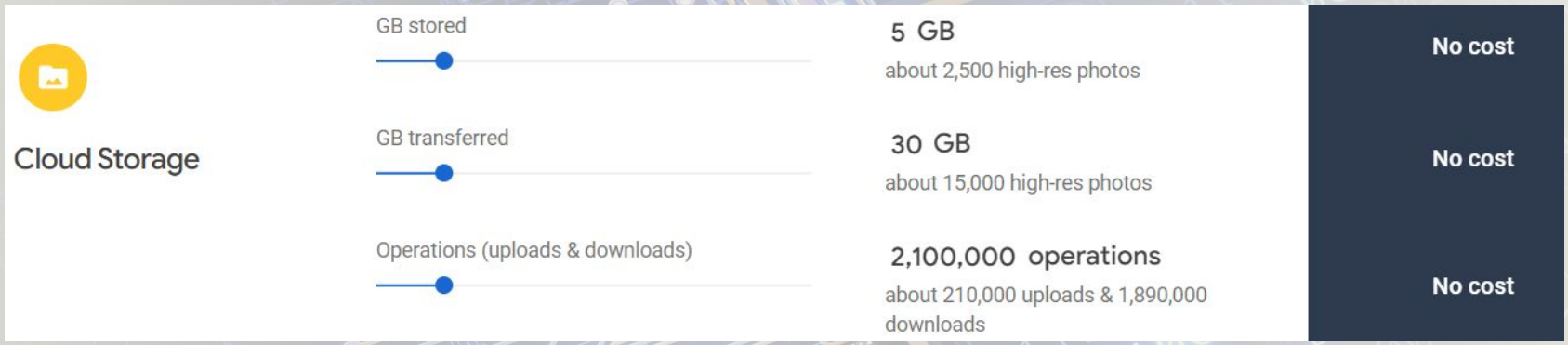


Example of a long term plan (billed monthly)

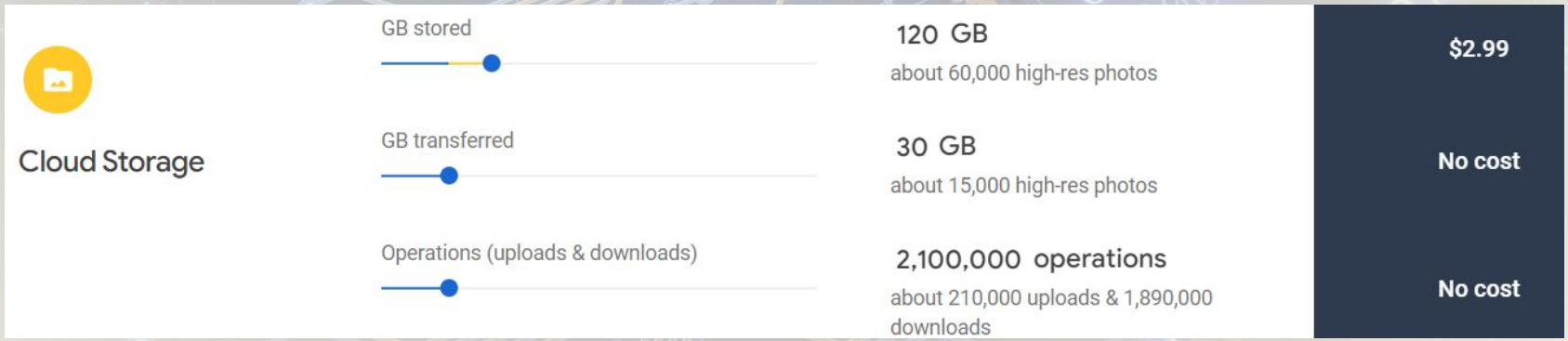


Bill of Material

Cloud Storage is free of charge, but with certain limitations



Example of a long term plan (billed monthly)



Conclusions

Project's Goals

- *We have designed a chip scanner that is compact, low cost and battery powered.*

- *Using the scanner is easy and intuitive.*

Project's Goals

- *Our dedicated application provides a seamless and a user friendly experience.*

- *The databases that we use to store the information are relatively cheap, secure and reliable.*

Future Projects

Ideas for Future Projects

- Adding capabilities that enable the pet's owner to automatically pay the vet, using the pet's RFID chip.
- Adding support for GPS trackers in order to track lost pets using the application.

Ideas for Future Projects

- Monitoring the pet's vaccines and health, alerting the vet whenever a pet's vaccination is close to becoming invalid.

- Adding privilege levels to enable certain authorized users to view sensitive.


THANK YOU
ANY QUESTIONS?