Group Members: Kyra Barnes, Anisha Parikh, Jaden Edgar

Git Repository: https://github.com/Security-System-Company/COSC-310-Project

FEATURES ADDED

- Connection (network) between android studio and processing
- Add database creation in processing when data is received over network
- Redesign UI for android studio: switching from access levels to rooms
- Add emergency event behavior for fires and intruder situations.
- Add tutorial to NFCScanner

OPEN-SOURCE LIBRARIES USED:

262588213843476, "TCP Echo Client," *Gist*. [Online]. Available: https://gist.github.com/HeptaDecane/dedc27e210ebd7e58c10eb38d6c26081#fileechoclient-java. [Accessed: 14-Nov-2022].

262588213843476, "TCP ECHO server java," *Gist*. [Online]. Available: https://gist.github.com/HeptaDecane/5b39bd01b03ff8fb9c44c7ebf65d9728#fileechoserver-java. [Accessed: 14-Nov-2022].

"Android.nfc : android developers," *Android Developers*. [Online]. Available: https://developer.android.com/reference/android/nfc/package-summary. [Accessed: 14Nov-2022].

"Color : Android developers," *Android Developers*. [Online]. Available: https://developer.android.com/reference/android/graphics/Color. [Accessed: 14-Nov2022].

CSVWriter (opencsv 5.7.0 API), 04-Sep-2022. [Online]. Available: https://opencsv.sourceforge.net/apidocs/com/opencsv/CSVWriter.html. [Accessed: 14Nov-2022].

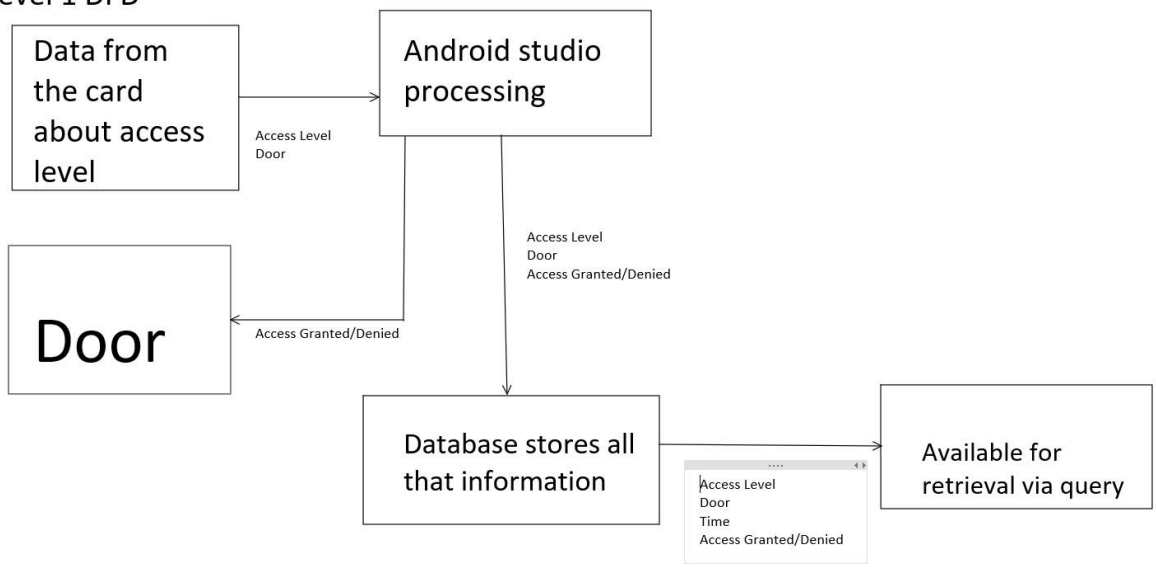
"FileWriter (Java Platform SE 7)," *FileWriter (Java Platform SE 7)*, 24-Jun-2020. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/io/FileWriter.html. [Accessed: 14-Nov-2022].

"Socket (Java Platform SE 7)," *Socket (java platform SE 7)*, 24-Jun-2020. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html. [Accessed: 14-Nov-2022].

Level 0 DFD

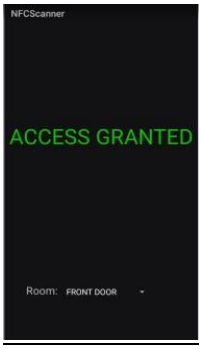
This diagram represents the level 0 data flow diagram. Very simply it shows that three variables are sent from the employee to the android studio app. Which then processes those variables and sends them as well as another two variables to be stored in a database.

Level 1 DFD

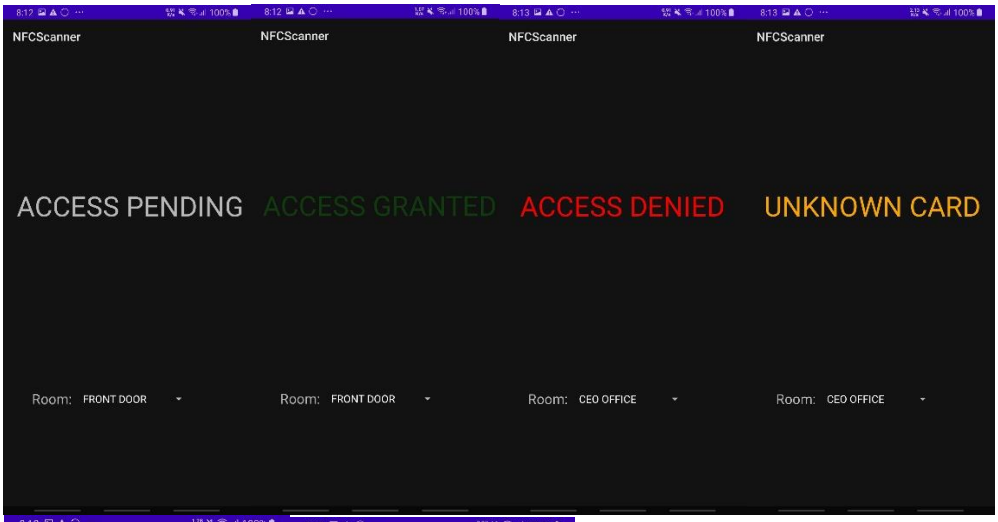


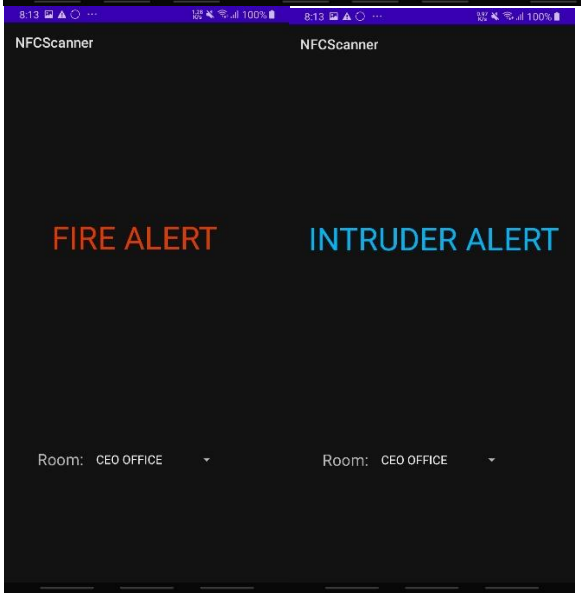
This diagram shows the data flow from the next level perspective. Here we see that the data starts off with the employee, goes to Android Studio where two more variables are added. This information can either be sent to the physical door for information on opening or remining closed. Or the information gets sent to the database. It is also an option to query through the database to search for information regarding door activity.

OUPUT



This image occurs when an NFC card with the right access level is tapped against the scanner, the room indicates what door is attempting to be opened.

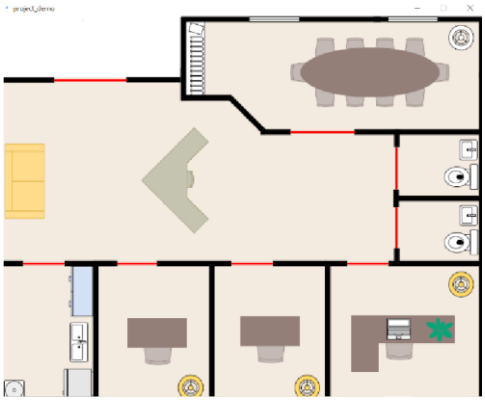




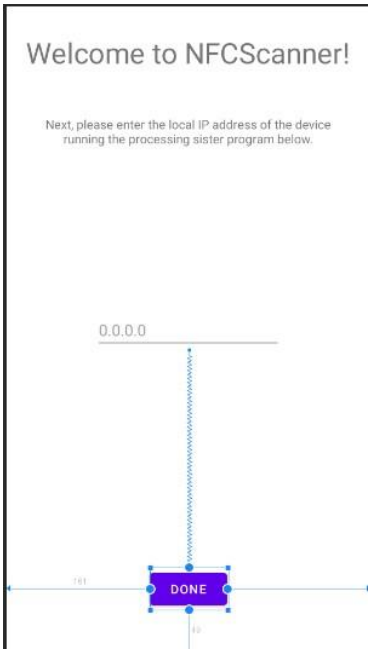
Here are the remainder of the possible outputs depending on the state of the NFCScanner app when an NFC card is scanned.

This image represents a visualization of the office when a fire happens, the emergency measures are implemented, and all doors are set to open.

The first issue occurs after an emergency is over, there is a small delay where all doors are set to red as shown below:



The other issue occurs when an IP address is put in on the screen below, any form of wrong input causes the app to crash



LIMITATIONS

- Resetting from emergency fire procedure simply requires scanning a regular NFC card, needs to be more secure
- Server attempts to connect with the port infinite times despite only needing to do it once
- Scanner will not wait for GUI to revert back to waiting state before taking in next scan attempt

5 FEATURES

- Sending information from Android Studio to processing using java.net library, specifically java network sockets. Here, using Java Sockets, we can send information in the form of strings to Processing. This allows us to store that information to a locally saved excel file using Processing.
    - Using Java network sockets allowed us to successfully connect both apps together, tying in the GUI of the NFCScanner app and the Processing office floor layout GUI to give a holistic GUI that can be used

- Writing information to an excel spreadsheet and editing the same excel spreadsheet to add on new data and not rewrite.
    - Being able to write to a excel file and save new access attempts without rewriting the file has given the program the ability to store an effective database of all access attempts, successful or not with precise times and which part of the office was accessed.

- included emergency features for fires or intruders that will have special behavior such that all doors either remain open or closed respectively, and an appropriate message is displayed to the user on the NFCScanner app.
    - having special behavior for emergencies and allowing the system to emulate what would happen should one of these emergencies occur is an important improvement to our system as it then gains the ability to handle exceptional circumstances and alter the systems behavior accordingly, outside of its default state of regular use.

- Reads data on NFC cards and decides what course of action to take based on the information received.
    - Altering NFCScanner to automatically read NFC data off the separate NFC cards created improves the system as it now acts closer to a real-life system, where testing functionality such as the "TEST" button and selecting roles instead of the room you wished to enter was used more as an early development prototype.

- virtual demonstration of how the employees would be provided access to rooms and how doors would open and close within processing
    - with both NFCScanner and Processing now multithreaded, and running the java socket and server respectively on separate threads from the main GUI, our system has improved by now being able to automatically emulate the opening of a door that a user has been granted access to as soon as they scan their NFC card on the NFCScanner app, instead of manually clicking the door to emulate the functionality that the java socket has now implemented.

UNIT TESTING

```
@Test
public void checkIpAddress() {
    InetAddress IP= null;
    try {
        IP = InetAddress.getLocalHost();
        assertEquals(IP, ipCheck(IP));
    } catch (UnknownHostException e) {
        e.printStackTrace();
    }
    System.out.println("IP of my system is := "+IP.getHostAddress());


}
```

This unit test method using JUnit in android studio checks to see whether the ip address inputted by the user is a valid ip address. This is necessary for the program to run correctly.

```
@Test
public void checkIntRoom(){

    if(intRoom == (int)intRoom) {
        System.out.println("Room number is valid");
    }
    else{
        System.out.println("Room number is not valid");
    }
```

```
    }
    public void checkNFCContent(){
        if(intNFCContent == (int) intNFCContent){
            System.out.println("Level is valid");
        }
        else{
            System.out.println("Level is not valid");
        }
    }
}
```