High Level Document

Andre Chia Wei Song
32179197

Diego Miguel Lozano
32179194

Jae Wook Lee
32153492

Jeong Young Hwan
32154231

Yong Jin Kim
32150881

8th October 2019

Mobile Open Source SW Utilization

Mobile Systems Engineering

TABLE OF CONTENTS

0. Document revision history

Date	Version	Description	Author
11/04/2019	0.0	Added Index	Lee jae wook, Andre Chia Wei Song, Diego Miguel Lozano, Jung Young Hwan, Kim Yong Jin
11/05/2019	0.1	Checking Index sub-contents	Lee jae wook, Diego Miguel Lozano Miguel Lozano, Jung Young Hwan, Kim Yong Jin
11/06/2019	0.2	worked on introduction and design considerations	Lee jae wook, Andre Chia Wei Song, Jung Young Hwan, Kim Yong Jin
11/07/2019	0.3	Worked on system architecture	Lee jae wook, Andre Chia Wei Song, Diego Miguel Lozano, Jung Young Hwan, Kim Yong Jin
11/08/2019	0.4	Worked on System Interface	Andre Chia Wei Song, Diego Miguel Lozano, Jung Young Hwan
11/08/2019	0.5	Worked on Detailed System Design (except diagram)	Lee jae wook, Andre Chia Wei Song, Jung Young Hwan, Kim Yong Jin
11/09/2019	0.6	Worked on Dynamic Model (except sequence diagram)	Andre Chia Wei Song, Diego Miguel Lozano, Kim Yong Jin
11/10/2019	0.7	Worked on diagrams (Class, Block, Sequence)	Lee jae wook, Andre Chia Wei Song, Jung Young Hwan, Kim Yong Jin
11/11/2019	1.0	First Draft for High Level Document finished	**Lee jae wook, Andre Chia Wei Song, Diego Miguel Lozano, Jung Young Hwan, Kim Yong Jin**
11/12/2019	1.1	Changed Diagram according to feedback of professor Jung	Diego Miguel Lozano, Jung Young Hwan, Kim Yong Jin
11/13/2019	1.2	Teammate Checked and got Feedback overall	Lee jae wook, Andre Chia Wei Song, Diego Miguel Lozano, Kim Yong Jin
11/14/2019	1.3	Changed Block Diagram	Lee jae wook, Andre Chia Wei Song, Diego Miguel Lozano, Jung Young Hwan

11/15/2019	1.4	Changed Sequence Diagram (Added specific functions)	Diego Miguel Lozano, Jung Young Hwan, Kim Yong Jin
11/16/2019	1.5	Team Feedbacks	Lee jae wook, Andre Chia Wei Song, Diego Miguel Lozano, Jung Young Hwan, Kim Yong Jin
11/17/2019	1.6	Finished first version.	**Lee jae wook, Andre Chia Wei Song, Diego Miguel Lozano, Jung Young Hwan, Kim Yong Jin**
11/18/2019	2.0	Added contents at Design Rationale	Andre Chia Wei Song, Jung Young Hwan, Kim Yong Jin
11/19/2019	2.1	Added contents at General Constraints	Lee jae wook, Andre Chia Wei Song, Diego Miguel Lozano, Kim Yong Jin
11/20/2019	2.2	added contents for 1.3 (definitions, acronyms and abbreviation)	Andre Chia Wei Song, Diego Miguel Lozano, Jung Young Hwan
11/21/2019	2.3	At dynamic Model for sub contents, divided Sequence diagram according to scenarios	Lee jae wook, Andre Chia Wei Song, Jung Young Hwan, Kim Yong Jin
11/22/2019	2.4	Changed Sequence diagram because of change of our code works	Lee jae wook, Andre Chia Wei Song, Diego Miguel Lozano
11/23/2019	2.5	Changed Block diagram.	Diego Miguel Lozano, Jung Young Hwan, Kim Yong Jin
11/24/2019	3.0	3rd version of High level document	**Lee jae wook, Andre Chia Wei Song, Diego Miguel Lozano, Jung Young Hwan, Kim Yong Jin**
11/25/2019	3.1	changed Class diagram. (added class)	Lee jae wook, Andre Chia Wei Song
11/26/2019	3.2	changed user interface (newly designed version)	Diego Miguel Lozano, Jung Young Hwan, Kim Yong Jin
11/27/2019	3.3	final draft finished	**Lee jae wook, Andre Chia Wei Song, Diego Miguel Lozano, Jung Young Hwan, Kim Yong Jin**

1. Introduction

1.1 Purpose

The purpose of this document is to provide a guideline for writing the software design document for our project.

1.2 Scope

Our project is aimed at not only just making a miniature IoT house but also developing a mobile application to control the devices and interconnect 4 different platforms (server, application, Arduino). By using our application, the different IoT devices present in the house can be controlled. The application can be easily navigated through an intuitive User Interface, in order to control the common household appliances that are connected with Arduino modules.

1.3 Definitions, acronyms and abbreviations

Arduino

Arduino is an open-source microcontroller designed to develop digital devices. It can be connected to various electronic components such as sensors, motors, displays, and wireless communication modules to help us build our project.

Android Studio

Android Studio is an Integrated Development Environment for creating Android and Android-only applications. The official Android

studio is using the Gradle build system. Android Studio uses Kotlin and Java as the main languages for developing android applications.

Encryption

As we will encrypt the password that is stored in the database, this section will be writing about the definition of database encryption. Database encryption can generally be defined as a process that uses an algorithm to transform data stored in a database into “ciphertext” that is incomprehensible without first being decrypted. Therefore, the purpose of database encryption is to protect the users’ data stored in the database from being read by unauthorized individuals with potentially “malicious” intentions. The technique we have chosen for our encryption is hashing the passwords in the database.

Hashing

Hashing is used in database systems and security systems worldwide as a method to protect sensitive data such as passwords. However, it is also used to improve the efficiency of database referencing. Inputted data is manipulated by a hashing algorithm. The algorithm converts the inputted data into a string of fixed length that can then be stored in a database. Hashing systems have two crucially important characteristics which are “unique and repeatable” and “hashing algorithms are not reversible”. This ensures that if an unauthorized attacker somehow gains access to the database, the attacker would not be able to read the data.

Internet of Things (IoT)

IoT is a technology that connects devices or objects through the Internet, making use of embedded systems such as wireless sensors or

microcontrollers. Also, Artificial Intelligence technology is used to receive and provide information to users.

Server

A server is a computer system that provides information or services to a client through a network and refers to a computer program or a device.

Database

The database is a storage of data that consists of one system. The server will link the application and the database, it is because every user data that is displayed in the application is retrieved from the database.

Firebase

It provides users with cloud storage services and can function as a back-end. Firebase offers remote cloud servers which enable to store and retrieve data from the server. Firebase also offers authentication services for login and registration.

2. Design Considerations

2.1 Introduction

Our project involves a variety of platforms (Server, Android smartphone and Arduino). Therefore, we have developed different applications that work together as a whole.

These applications follow different design approaches, depending on the platform they run on. The different design approaches are:

- **Object-oriented design**: The Android application has been developed in Java, which is *per se* an object-oriented programming language.
- **Structured design**: Despite the Arduino code takes elements from both C (a structured language) and C++ (object-oriented), we could say that the approach taken is mainly structured.
- **Non-relational design**: Regarding our server, we are using Firebase as a development platform. Firebase implements a NoSQL (non-relational) database that stores the logs of using the Arduino device from different users.

Regarding the architecture of the system, the following paradigms have been used:

- **Client-server architecture**: It takes place between the server and the Android app. The server stores the logs from the users and acts as a provider for the Android application, which requests the data to the server in order to display it to the users.
- **Master-slave architecture**: It happens between Android and Arduino. Android takes the position of the master, sending signals to Arduino, which then processes them and acts accordingly.

Finally, the most relevant techniques and tools used for the development of our system are:

- **Paper**: as simple as it sounds. The first designs and outlines of our app were sketched on a piece of paper. The main advantage that paper offers is that changes can be easily made. This is especially handy in the first stages of the design process.
- **draw.io:** we have used this tool to generate the different diagrams that appear in this document.
- **Android Studio**: the Android app has been deployed making use of Google's IDE Android Studio, specifically designed for Android development.
- **Arduino IDE**: to write the Arduino program and load it into the actual Arduino.
- **Firebase**: development platform used for the back-end and server implementation.

2.2 Environment Overview

Our project is aimed at implementing a smart home, based on the concept of home automation. For that, we are making use of IoT technology. The project environment is a house equipped with different modules and sensors. These sensors include a motion sensor and a temperature sensor. The motion sensor can be used to activate the lights automatically. The temperature sensor controls the automatic activation of the fan. When the temperature is above 30ºC, the fan (LED light; blue) will be automatically turned on. If turned on automatically, the fan will turn off when the temperature reaches 20ºC or less.

2.3 Assumptions and Dependencies

Assumptions

We are assuming that the implementation of our system in the miniature house would be the same in a real house, but this might not be entirely true. It is because IoT actually means that the objects are linked all together by the Internet. But for our IoT miniature house, we have implemented it by connecting the components using Bluetooth technology. Also, there were some constraints in implementing a variety of IoT devices. Hence, we have replaced them with LED lights. (ex. a fan which works when the temperature is high). A motion sensor has also been implemented which turns on the lights automatically whenever the user enters the empty house. The lights are represented by a yellow LED light.

Dependencies

- **Server & Application**: when the user logs in to the application, the username and password are compared with the data stored in the server. Since the password is hashed in the server, we do not know the plaintext of the user's password. We are only comparing

hashes when the user logs in. If they match, the user will have access to the app features.

- **Application & Arduino**: the application and Arduino module are dependent on each other, being connected via Bluetooth. The application is the bridgehead between Arduino and the server. It has been configured to show logs in between to show users the overall reliability of the system.

2.4 General Constraints

Resource constraints

Limited budget. For example, we could build a custom PCB board to connect all our Arduino-related electronic devices. As this would be more expensive, the components will be soldered together with regular cables. Also, due to lack of resources, the actions are represented by LEDs instead of the real components. Our project actually implements a real-life IoT system. However, we believe that making it completely functional (e.g. installing all the components, such as the fan), is out of the scope of the subject and would make no difference in proving we are able to do it.

Linking Constraints

The project is aimed at implementing an IoT system. But as we lacked experience, it took time and cost a lot of money to find out the proper Arduino module. For example, we have found a Bluetooth module that works only with Bluetooth 2.0, but we actually needed a 4.0 module. Also, we needed to fully understand the electronic details of our circuit and components, so that the system worked properly.

Size constraints

Our project will include a simplified miniature model of a makeshift house. In a real house, some implementation aspects might have to be adapted. For instance, we are using a Bluetooth module, so we would have to take into account the Bluetooth range inside the house.

Communication constraints

In the beginning, we have tried to communicate by “KAKAO Talk” which is a mobile chatting application and “Github”. But, as each teammate had different classes, it was hard to communicate in real-time. Also as some of our teammates presented a lack of experience using Github, it was very difficult to collaborate our code together on Github. Because of the different programming techniques used by each team member, our code was incompatible. That made us realize the need for real-time communication. Since the actual collaboration is done offline, our collaboration was done differently.

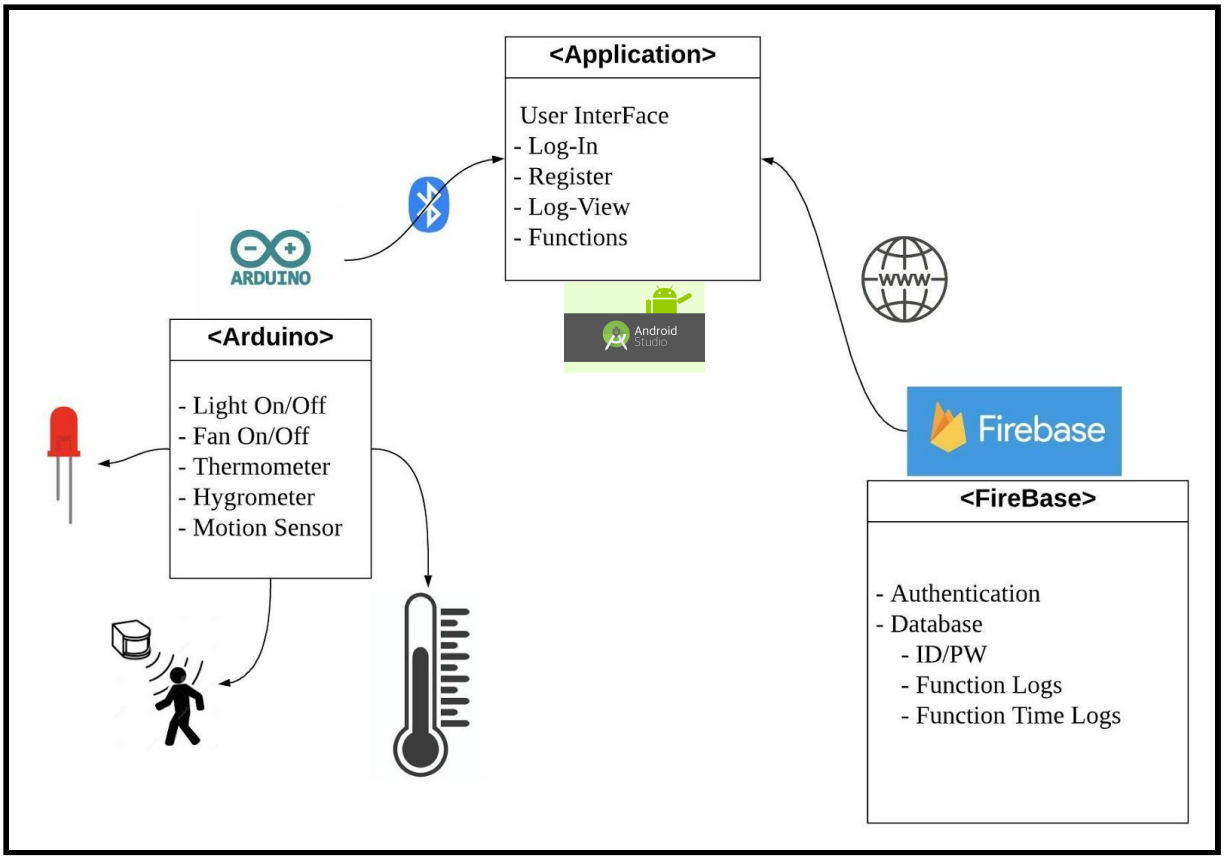
Team’s limited experience

Our team members’ experience is limited in certain aspects:

- Server deployment.
- Data encryption.
- Platform integration. Our project integrates three different platforms (server, application and Arduino). This poses some technical difficulties that needed to be tackled.
- Building the miniature house.

3. System Architecture

3.1 Overview of system architecture



3.2 Components of the system

Arduino

Arduino will be implementing the turning on and off functions of the light and fan, by using the motion sensor and thermometer it will display current status at the LCD panel. Also, the application and Arduino will be communicating using Bluetooth.

Application

The application has been built in mind of the user. By logging in, the user will be able to use the IoT service and view the logs. The data will be sent through the Internet which connects the application and the server.

Firebase (server)

In the database, the user's logs will be stored for them to access using the Server Log's page. Also, user's information such as ID/PW will be safely stored by encryption. This way we have handled the security problem.

4. System Interface

4.1 Software Interface

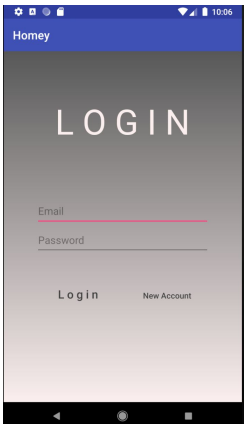
For the software, we have used Arduino for IoT devices, Android Studio for the application, and Firebase for the server. Therefore, the interface is only relevant in the case of the Android application.

Application

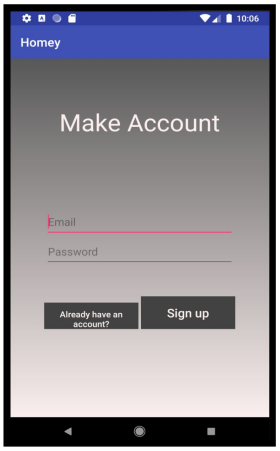
Android Studio is our main software interface. By using android studio we will construct the server of our own, and design our application.

Starting the App

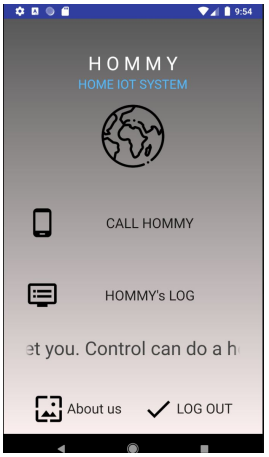
Log-In page



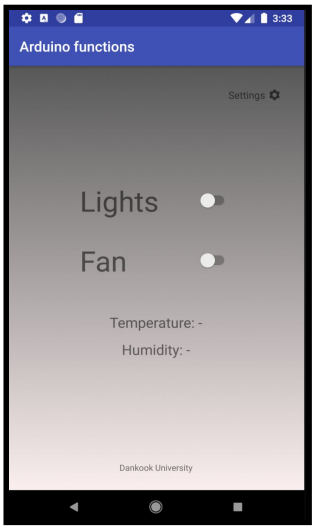
Register



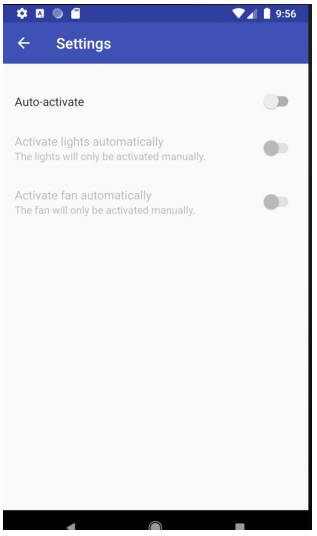
Main page



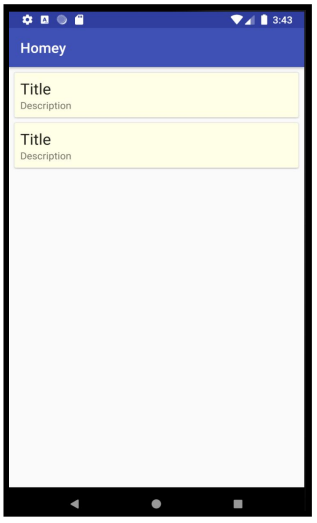
Function Page ("Call Hommy")



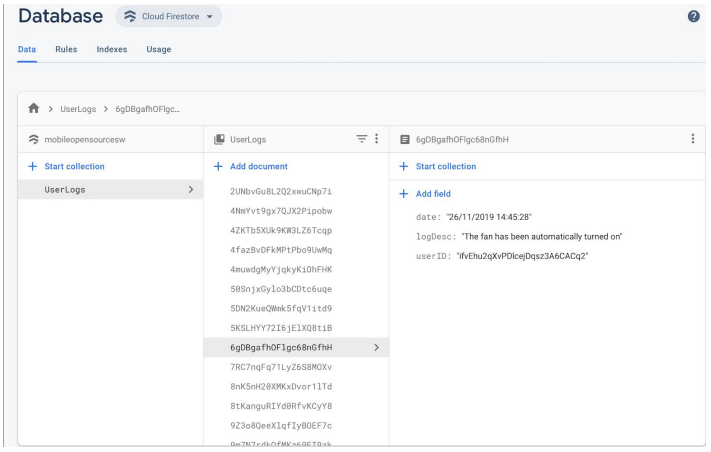
Settings for the functions



Log View



Firebase Logs



Arduino

Arduino is our sub-using software interface. Arduino will be used in controlling household appliances with our application.

Server

For the Server we have used Firebase. By using the Server, it will store the data about user's ID/PW and Logs for each individuals. The server will encrypt all of the information that user store, so the security problem have solved.

4.2 Hardware Interface

Our hardware interface includes the next components:

- 1 x Arduino UNO.
- 1 x Breadboard.
- 1 x JY-MCU Bluetooth Slave Module.
- 2 x LEDs.
- 2 x 200 Ohm Resistors (as many as LEDs).
- Breadboard cables.
- Miniature house.
- Battery Supplier.
- 1 x Temperature/ humidity sensor DHT-11.
- 1 x Infrared sensor HC-SR501.
- 1 x Arduino LCD IICI2C 16 x 2 size.

5. Detailed System Design

5.1 Design Explanation

The user is first greeted with the Log In page when they open the application. A user will be prompted to either Log In or choose to Sign Up for an account. When signing up for an account, the user will be prompted for an email and password to create an account. Firebase Authentication provides a way for us to check if the current email exists in the user database before allowing the user to create an account, if the email exists, the user is asked to input another email or else the account creation process will be successful. After logging in, the User can choose to navigate to either one of the three pages, our Team Introduction, Arduino Bluetooth Functions or the Logs created by the user from the Arduino Bluetooth Functions.

The Team Introduction page contains a description of each of us along with a photo.

The Arduino Bluetooth Functions page requires a Bluetooth connection to our Arduino Uno board that controls the many functions that we have.

The functions include issuing a request to Arduino to turn on the lights or the fan. This is done through the Android application. The requests are sent via Bluetooth to Arduino, which then processes the signals and acts accordingly. At the same time, a log is sent and stored in the server, keeping a record of the user's actions. At the same time, Arduino is continuously monitoring the temperature and motion conditions through a DHT sensor (temperature and humidity sensor) and a motion sensor, respectively. When the temperature rises above a predefined value, Arduino will automatically turn the fan on. The user is informed of this event through the app, and a log is stored in the server. Similarly, when the motion sensor detects movement, the lights are turned on, informing the user and logging the event in the server.

The UserLogs page will contain any Logs created by the user when using the Arduino Bluetooth functions. How the process works is, when the light or fan turns on and off, information will be sent to the application where a logging system has been configured to send any logs to the Cloud Firestore database. For example, when a user turns on the Fan at 2pm, Arduino tells the application

that the fan has been turned on at 2pm and by which user. The application then creates a User Log object which contains the date & time of when the fan was turned on, a description of what happened and the user ID of the user who turned on the fan. This information will then be stored in the database. When on the UserLogs page, the page will extract all of the documents in the database and display each log individually on the page. When on the UserLog page, an UserLogAdapter has been configured to listen for any new logs that may occur while on the page, hence the log data will always be updated.

5.2 Design Rationale

Arduino

1. The microcontroller, the libraries used, as well as the development environment are all open-source, being thus aligned with the purpose of the subject : “mobile open source software utilization”.
2. Price : the Arduino board and all the components used are inexpensive (Less than KRW 20K).
3. Easy to program and use: Arduino has a gentle learning curve. The online documentation is complete and easy to follow. Also, the available libraries make the programming of the different components straightforward and uncomplicated.
4. The Arduino is a very good tool to link software and hardware. The codes for the software are easy to program. Also, connecting hardware components like sensor and LED is very easy, which is the reason many people use Arduino.

Android Studio

Android Studio is used to design application. It only support for Android OS for mobile. The Android uses XML for designing application and JAVA for other functions. The main reasons we chose Android Studio are:

1. The special feature of JAVA is that it inherits classes and enables dense development at the level of child classes, and makes information difficult to access from outside by hiding information. Android faithfully follows that advantage. It is designed to easily preserve the integrity of the data by importing and removing such user data or log records through the adapter
2. The big advantage of Android Studio is its compatibility with multiple platforms. It is common to go through the encoding and decoding process of Bluetooth in order to get the data value of Arduino sensor, but it enables easy development by utilizing the function as an adapter in Android itself.
3. It is very difficult to memorize the grammar of a language. However, Android borrows a grammar assist system that is superior to the general integrated development environment, so that data can be matched precisely.
4. Android Studio automatically handles the stack-like data structures that create and destroy activities allowing developers to focus on their development. It's a different problem from optimization, but it's not as efficient as those problems.
5. Android is free to commit or download to GitHub in the form of packaging. Besides, it is a universal tool that can be distributed in "apk" format, so it can be developed smoothly.

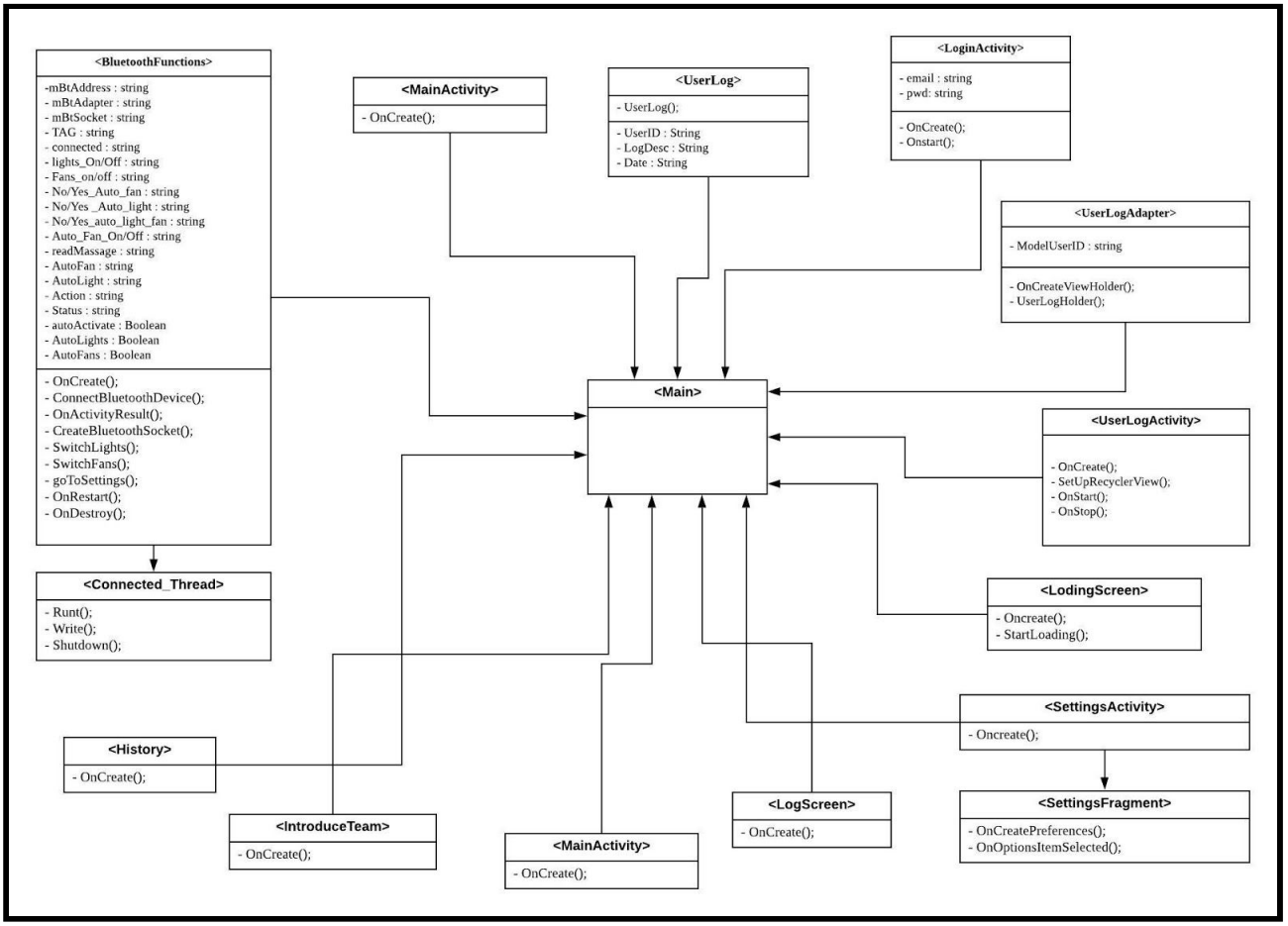
Firebase

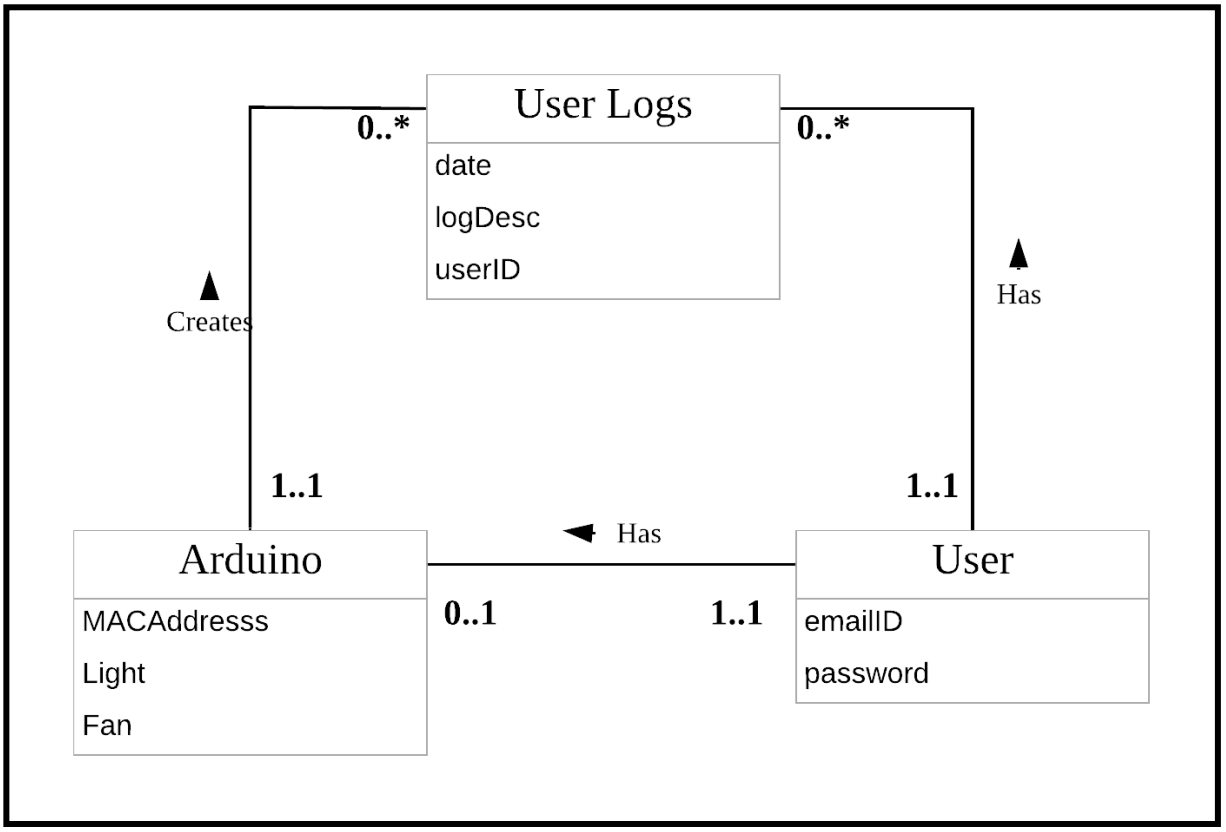
Firebase is one of the most supported backend frameworks that is used to develop Android applications. The main reason we chose Firebase are:

1. Firebase also works better because it is provided by Google which is the owner of the Android Operating System hence they are directly compatible with each other and provide better results compared to other backend options.
2. Provides Authentication when registering users and for logging them in. All of the user data is stored in a Firebase Authentication server which can be conveniently accessed online through the firebase console.

3. The user data is also automatically encrypted by Firebase for us using their provided hash algorithm which we can customize if necessary.
4. Firebase has a Cloud Firestore database which provides us a real-time database that allows us to send any data to the database quickly and receive any updates immediately.
5. The Cloud Firestore database is also a NoSQL database which makes it easier to store large amounts of data, provides better performance and is easy to scale.
6. Using the Firebase framework also gives us access to Google Analytics which is can provide us with lots of information regarding our application such as any app crashes or other events occurring.

5.3 Class Diagram





5.4 Class Descriptions

UserLog

Contains the UserLog Model Object consisting of variables such as the userID, date & time of the user log and a description of what was logged into the cloud firestore database.

UserLogActivity

UserLogActivity contains the code to display the UserLogs from the database in a RecyclerView. It initializes an instance of the UserLogAdapter class to help obtain the UserLogs.

UserLogAdapter

The UserLogAdapter is the code that takes the data from the documents stored in the cloud firestore database. The UserLogAdapter helps to store the data in cards that will be sent to the UserLogActivity to be displayed on the page.

BluetoothFunctions

Contains the methods to connect to the Arduino Uno and call the functions to off/on the fan and light. Also contains adding logs to the Firebase server.

IntroduceTeam

Contains Group C members' introductions.

LoadingScreen

The screen that will load while waiting for the application to launch into the MainActivity page.

LoginActivity

Checks if the user's email exists in the Firebase Authentication database, if the email exists compare the user inputted password to the password hash stored in the database. If the hashes are the same, allow the user to login successfully. Or else, an error prompt will appear.

LoginScreen

The sign up page for the user to signup for a new account using an email address and password. The email must not exist in the Firebase Authentication database else the signup will be unsuccessful.

MainActivity

The screen where the user is able to click the Login button and proceed to the Login page.

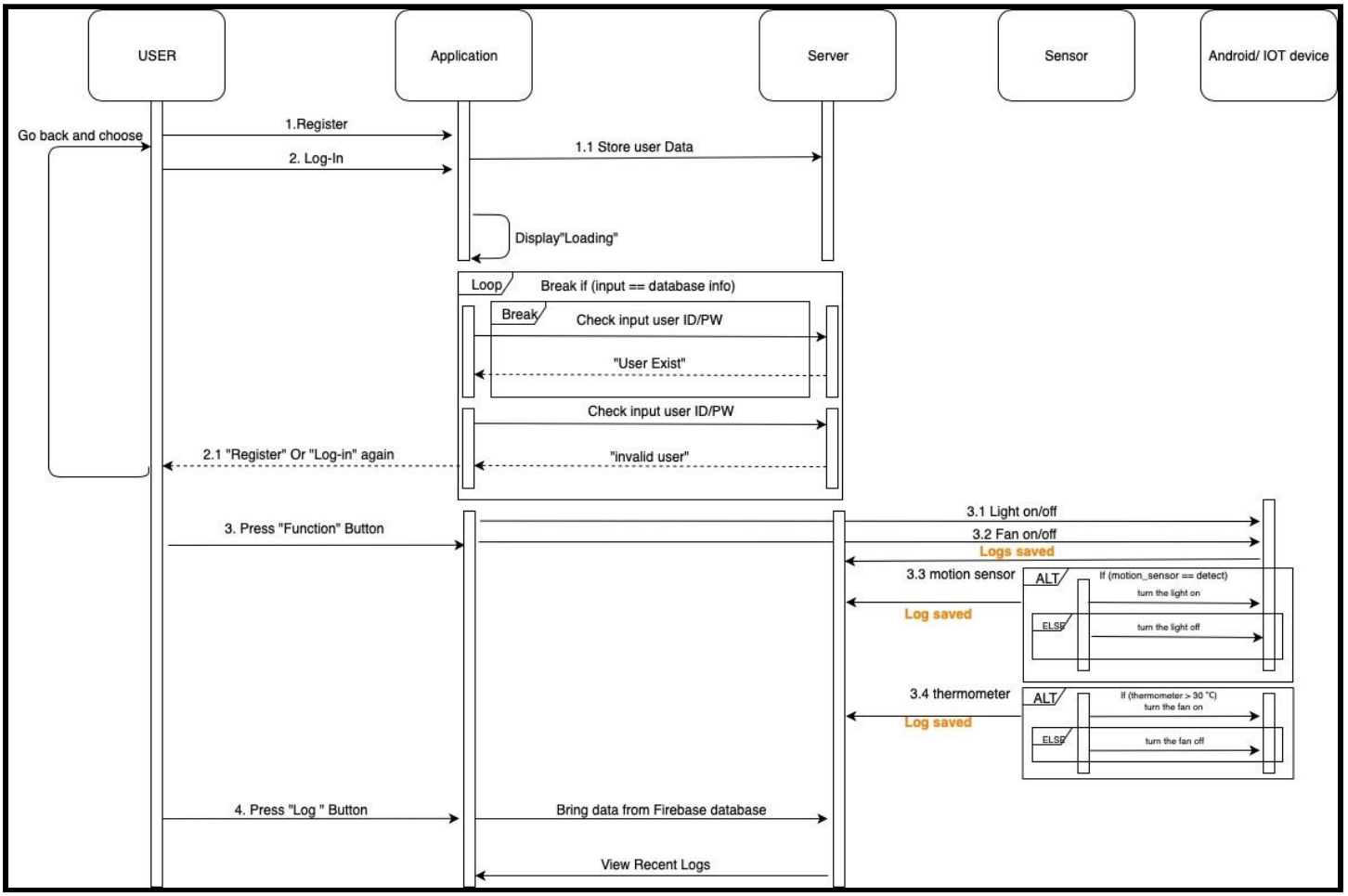
MainMenu

The user can navigate through the application and go to their own logs, introduction of the team and the Arduino functions.

SettingsActivity

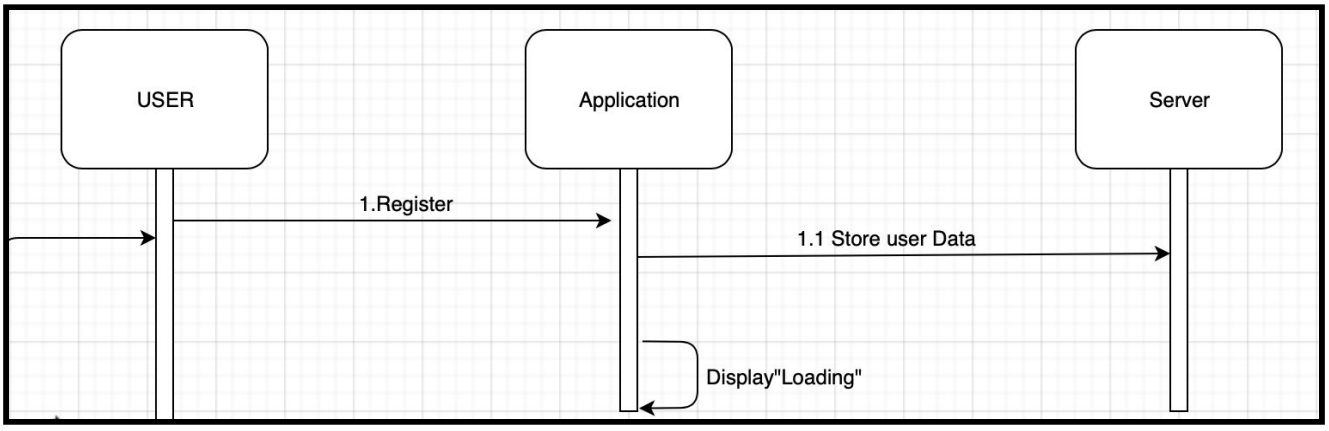
This class implements the connection between Arduino and the smartphone via Bluetooth. By using this class, the device will be able to use the Bluetooth to communicate with Arduino.

6. Dynamic Model



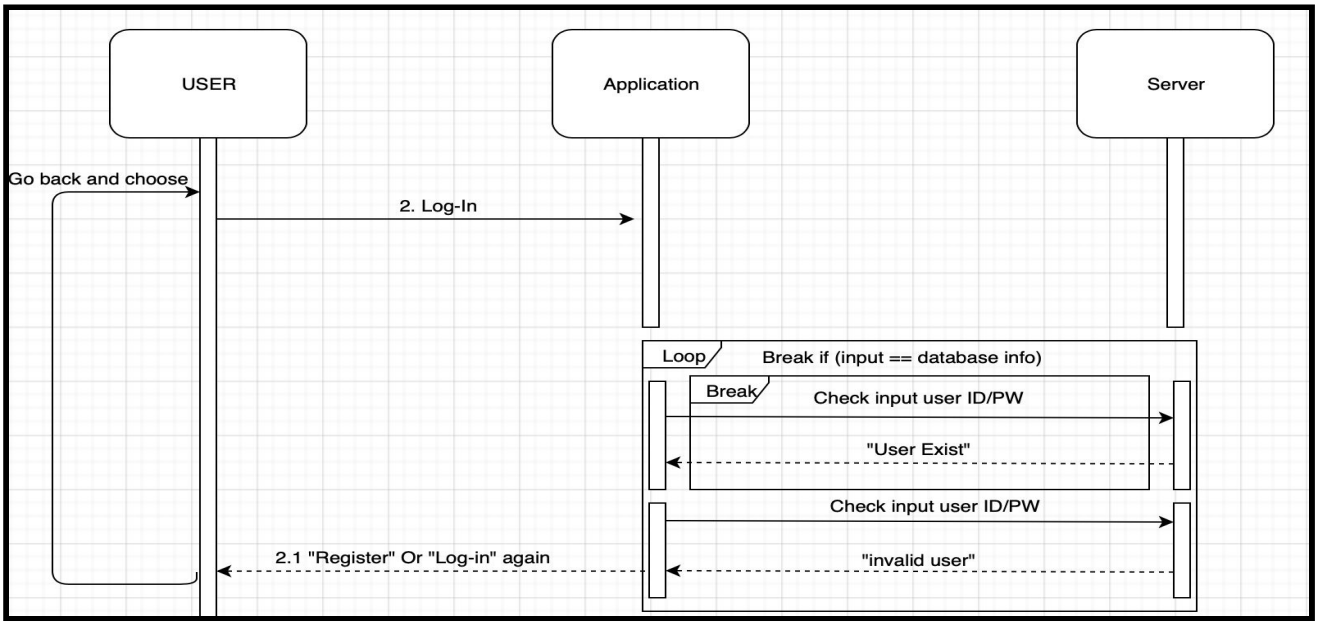
6.1 Explanation for each Scenario

Register



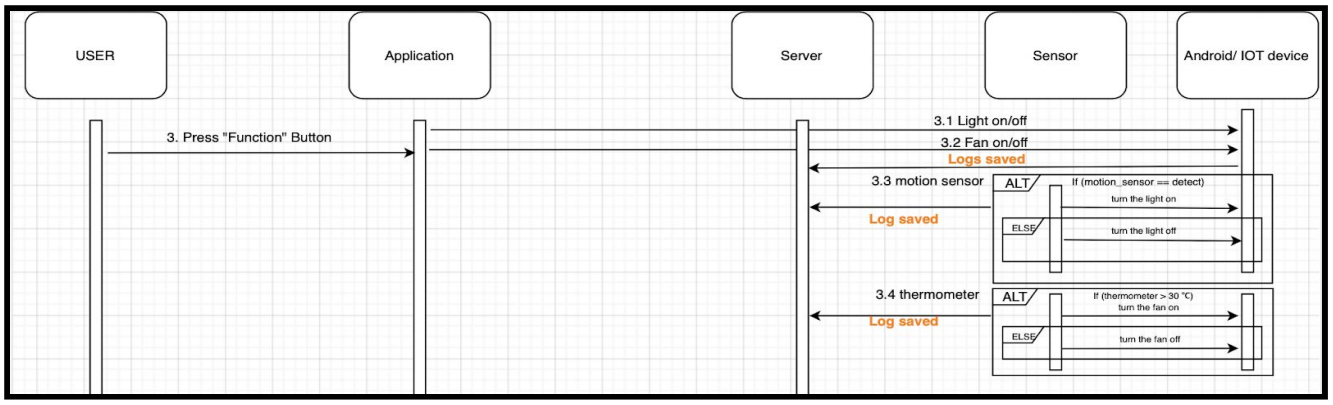
After starting the application, app will ask to Register or Log-in. As the user who uses at the first time do not have any ID/PW the sequence will begin by register. When user register, all the important data will be stored at the server database. ID/PW will be encrypted and then stored at the database.

Log-in



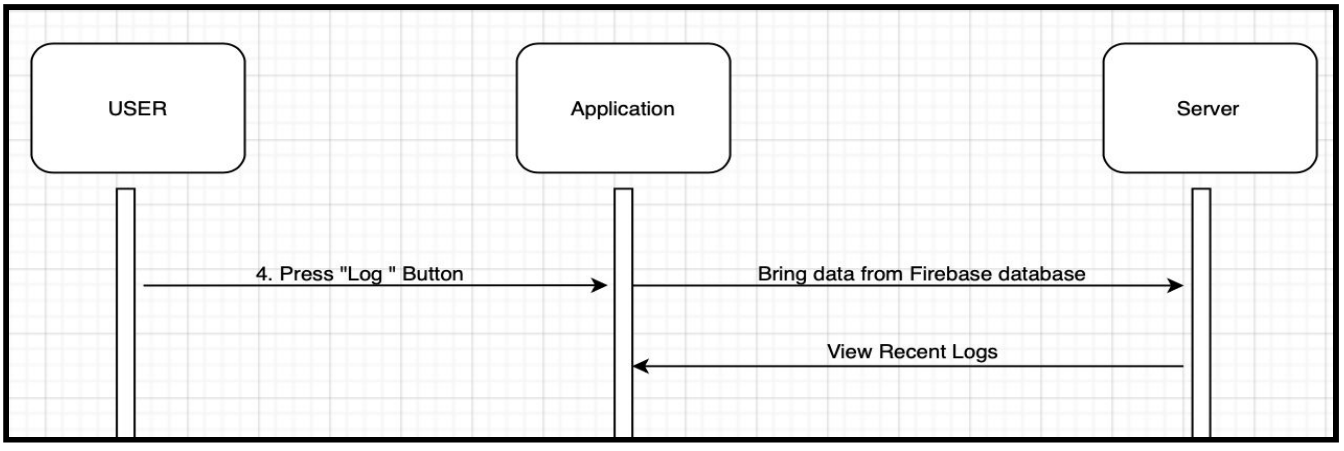
After Register, as the user's data is stored at the server, user will be able to log-in to the application. But if the user wrote wrong ID/PW, the application will check through the server and evaluate it. If the information is wrong, than the application will ask the user to rewrite the ID/PW. If the information is correct, user will start using the application.

Pressing Function Button



When the user logs in, application will show main page. There will be several buttons but the main button is “Function” and “Log View”. When the user press “Function” button, the user will be able to start control the IOT devices. The functions are separated into 4. Turning the ‘turning the light on/off’, ‘turning the fan on/off’, “turning the light on according to the condition of motion sensor” (if the motion sensor senses the movement, the light will turn on), “turning the light up according to the condition of thermometer” (If the thermometer senses the temperature above the 30 degree celsius, the light will turn on) .

Pressing Log Button



At the main application page, there is “Log View” button. When the user press the log viewing button, the application will ask the Firebase server for the log, and the server will bring out the Logs to the application. The logs will be containing which function the application operated and when that happened.