

Smart Door Lock

MAJOR QUALIFYING PROJECT

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the
Degree in Bachelor of Science

Authors:

Aleksander IBRO
Augusto WONG
Mario ZYLA

Project Advisors:

Michael CIARALDI
Maqsood MUGHAL

April 28, 2019

This report represents work of WPI undergraduate students submitted to the faculty as evidence of a degree requirement. WPI routinely publishes these reports on its web site without editorial or peer review. For more information about the projects program at WPI, see <http://www.wpi.edu/Academics/Projects>.

Abstract

Smart Door Lock

Major Qualifying Project

The goals of this project were to build a modern, easy-to-use, smart door lock that allows for accessible unlocking and adds convenience, utility, and security to your home. It allows users to open their door remotely via the accompanying Smart Lock mobile app, or hands-free by using face recognition via a camera mounted on the door. The system was made up of three major components, including a cloud back-end, an on-board logical unit, and a mobile application.

Acknowledgments

We thank our MQP advisors professors Michael Ciaraldi and Maqsood Mughal for guiding us throughout the project, and for their great flexibility on agreeing to make this project an actual MQP.

Executive Summary

The goal of this project was to provide an easy and convenient method for unlocking a front door by removing the need for the old-fashioned key. We start by evaluating the need for such a system by sending a survey and analyzing the results. We follow the Software Development Life Cycle to set the project objectives and implement the design. The project has three main components: a Raspberry Pi, a cloud backend, and a mobile application. The Raspberry Pi is attached to the door and is responsible for controlling a servo motor, a camera, and an actuator. Users can open the door by either tapping a button on the mobile app, or just by approaching the door. When a user is within a certain radius, a PIR sensor detects them, and activates the camera. The Raspberry Pi then sends an image of the user at the door to the backend server.

The server is a Python Flask application hosted in Azure Cloud Services. It seamlessly runs a face recognition library, which compares each image sent by the Raspberry Pi with images already uploaded in the Cloud. If the face recognition algorithm identifies a face in that image, it automatically sends a positive response to the Raspberry Pi, which in turn activates the actuator and opens the door. The Azure backend is also responsible for serving requests coming from the mobile application, like storing images of users' faces, tapping a button to open the door, or saving each user's personal information. Azure App Services is a feature of Azure, which makes it easy to quickly deploy a cloud application.

The Smart Lock app is the forefront of the system, providing an intuitive User Interface. From this app, users can not only manage their accounts, but also add friends and family by simply providing their names and photos of their faces. Once a friend is added, the face recognition library will be able to identify them, and, if provided with access by the user, automatically unlock the door via the Raspberry Pi. To make sure that you don't have unexpected guests, you can pick the days and times that your friends can get in. You can also opt in for a push notification that will let you know whenever someone is at the door. From the Smart Lock app, users have access to a livestream of their front door. This feature allows users to let their guests in with a simple tap of a button, even if these guests haven't been added beforehand. Furthermore, users can keep track of everyone using the system with a dedicated view that shows every action performed on the door, including the time and day of the action, as well as the person performing it.

Our project integrates a board computer like the Raspberry Pi, a Cloud backend provided by Microsoft's Azure, and a mobile application built in Swift. These three components come together to create a seamless and convenient way for you and your family to control the doors of your home. Now you don't have to worry about losing your key, getting locked out, or having your hands full with groceries, because the Smart Lock system has you covered.

Table of Contents

<i>Abstract</i>	i
<i>Acknowledgments</i>	ii
Executive Summary	iii
Table of Contents	iv
List of Figures	vi
List of Tables	vii
1 Background	1
1.1 Internet of Things	1
1.2 Access Control Credentials and Readers	1
1.2.1 Keypads	2
1.2.2 Access Cards and Card Readers	2
1.2.3 Biometric readers	2
1.3 Smart Door Locks	3
1.3.1 August Door Locks	3
1.3.2 Kwikset Kevo Locks	4
2 Methodology	5
2.1 Introduction	5
2.2 Why We are Choosing this Strategy	5
2.3 Requirements Gathering	5
2.3.1 Survey	6
2.4 Analysis	6
2.5 Design	6
2.5.1 Platform Choice - Mobile Websites vs. Mobile Applications	7
2.5.2 Platform Choice - iOS vs. Android	7
2.5.3 Platform Choice - Raspberry Pi vs. Arduino	8
2.5.4 Platform Choice - Motion Sensor Ultrasonic vs PIR	10
2.5.4.1 Ultrasonic Sensor	10
2.5.4.2 PIR Sensor	10
2.5.5 Platform Choice - Actuator	11
2.5.5.1 Pneumatic Actuators	11
2.5.5.2 Hydraulic Actuators	12

2.5.5.3	Electric Actuators	12
2.5.6	System and Object Design	13
2.5.6.1	UML Class Diagrams	13
2.5.6.2	Components Diagrams	14
2.5.6.3	State Machine Diagrams	14
2.6	Implementation and Testing	15
2.6.1	First Component - Smart Lock App	16
2.6.2	Second Component - Raspberry Pi	17
2.6.3	Third Component - Cloud Backend	19
2.6.3.1	Azure Virtual Machine (VM) in the Cloud	19
2.6.3.2	Azure SQL Database	20
2.6.3.3	Azure App Services	21
2.6.3.4	Google's Firebase Authentication	21
3	Results and Conclusions	22
3.1	Future Work	22
Appendix A:	User Manual	23
Getting Started		23
Adjust your Settings: Account Tab		23
Who is at the door: Home Tab		24
Add Friends: Guests Tab		25
Add Admins: Guests Tab		26
Check who has been at your door: Activity Tab		27
Appendix B:	User Stories	28
Appendix C:	Survey Results	29
Bibliography		31

List of Figures

Figure 1.1 Keypad Access Reader	2
Figure 1.2 Card Reader	2
Figure 1.3 Biometric Reader	2
Figure 1.4 Activity View for August App	3
Figure 1.5 August Door Mount.....	3
Figure 1.6 Kwikset Kevo Lock.....	4
Figure 2.1 Software Development Life Cycle Phases	5
Figure 2.2 High-level class diagram of the Smart Lock main objects.....	13
Figure 2.3 Components Diagram.....	14
Figure 2.4 State Machine Diagram for the Smart Lock system mounted on the door	15
Figure 2.5 The three main components of the Door Lock System	16
Figure 2.6 Electrical Schematic of the Smart Door Lock device	18
Figure 2.7 The “upload_image” route that runs in the Flask server when an image is sent from RPi	20
Figure 2.8 Entity Relation Diagram for the app	20
Figure 2.9 Comparison of adding a new Friend to the DB using the Flask Server (left) and using the App Services (right). The Flask server code (bottom left) is completely gone after migrating to App services.....	21
Figure 3.1 Smart Door Lock system mounted on a mockup door	22

List of Tables

Table 2-1 Comparison between a website and a mobile app [11]	7
Table 2-2 A Comparison of React Native Pros and Cons [12]	8
Table 2-3 A comparison between Raspberry Pi and Arduino Uno [13].....	9
Table 2-4 A comparison between an ultrasonic and a PIR sensor	11
Table 2-5 A comparison among a pneumatic, hydraulic, and electric actuators	12
Table 2-6 Smart Lock App features submitted in proposal. Underlined features were not implemented	16

1 Background

In this chapter, we begin by introducing the Internet of Things, continuing with a discussion of the inclination people have shown toward IoT devices, and how different door access control systems have adhered to or missed the population's needs. We show examples of some products designed to offer a much needed convenience to door lock systems.

1.1 Internet of Things

In the last few years, the Internet of Things (IoT) has attracted attention from both academic and industrial worlds. IoT is a concept describing a vision in which everyday objects are connected to the Internet, are identified, and communicate with other devices [1]. These interconnected devices are referred to as smart devices. Each of these smart devices is a physical component with communication functionality, possessing a unique identifier, some basic computing capabilities and a way to react to different physical phenomena around them [2]. From a global IoT market perspective, the population of IoT devices will reach about 26 billion by 2020, up from only 900 million in 2009 [2].

The most common examples of smart devices are wearables like watches. Smart watches such as the Apple Watch and Fitbit have penetrated the market not very long ago. Such devices have also entered our homes with products such as Amazon Echo, a voice assistant, which users can talk with to play music, provide weather, and control other IoT devices. A less common example can be seen in Barcelona, where the city has implemented some IoT initiatives that have helped enhance smart parking [3].

An exponential growth is expected for the IoT market and an increasing number of smart devices show the inclination of the industry toward offering these interconnected solutions to increase convenience, but also the high adoption rate of the population [4].

1.2 Access Control Credentials and Readers

We have researched a few access control methods that are currently being used around the world. In this section, we will explain three of the most prevalent types, focusing on their advantages and disadvantages.

1.2.1 Keypads

The most basic types of ID readers are keypads, based on simple twelve-digit keypads, containing the numbers 0-9 and a * and # sign [5], shown in figure 1.1. These types of access control locks are very simple and cheap, and over time they have advanced to include some more secure functionality, like scrambling the LED number keys on the keyboard so that others who can detect the gesture and movement wouldn't be able to use it. The main disadvantages include people's need to share the code, like with package or food delivery or friends. If people share this code, you can't really know how many people have access or who accessed the keypad at what time.



Figure 1.1 Keypad Access Reader

1.2.2 Access Cards and Card Readers

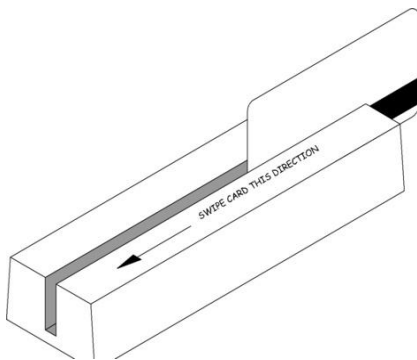


Figure 1.2 Card Reader

There are a number of different card reader types. Some common ones include magnetic stripes, barcode, etc. Magnetic stripe cards have a magnetic band laminated to the back of the card [5]. The card can contain a code, which serves for identification, the person's name, and other useful data, although only the ID code is commonly encoded. ID cards are effective, simple to build, and cheap, but they are very easy to duplicate. Other types of ID cards work in a very similar way to magnetic stripe cards.

1.2.3 Biometric readers

Any device that reads the identity of a person by comparing some attribute of their physiological being or behavioral traits against a sample database is called a Biometric Reader [5]. Some common biometric readers include facial recognition, fingerprint, and iris recognition. Advantages of biometric readers include increased security and convenience of not carrying a key or card. Disadvantages include the need to enroll users in a database by providing sufficient samples of user's face (for facial recognition) or fingerprint (for fingerprint scanners). Some modern day inconveniences include users who try to fool these systems. A common example is how facial recognition systems can be fooled by high-quality photos of user's face, even without the user being present [6].



Figure 1.3 Biometric Reader

1.3 Smart Door Locks

The idea behind the modern day smart locks is to unify the functionality provided by convenient and secure access control methods with the convenience of internet connectivity and remote control. There are a few products on the market that offer these systems and we will talk about two of them, once again focusing on the features they offer and potential disadvantages, as these products will serve as inspiration for our actual product implementation.

1.3.1 August Door Locks

PCMag has rated the August door locks as one of the best smart door lock systems [7]. They offer a mobile application, where you can check an history of the activity, like who locked or unlocked it at different times, shown in Figure 1.4. From the mobile app, you can also

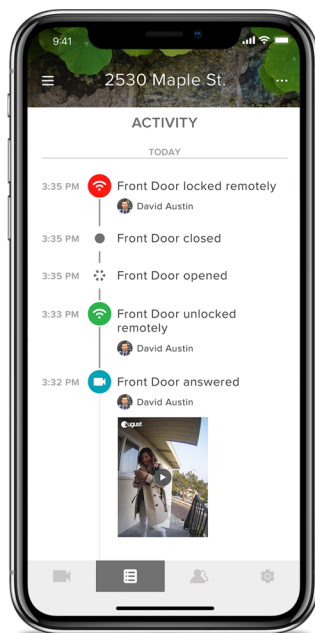


Figure 1.4 Activity View for August App

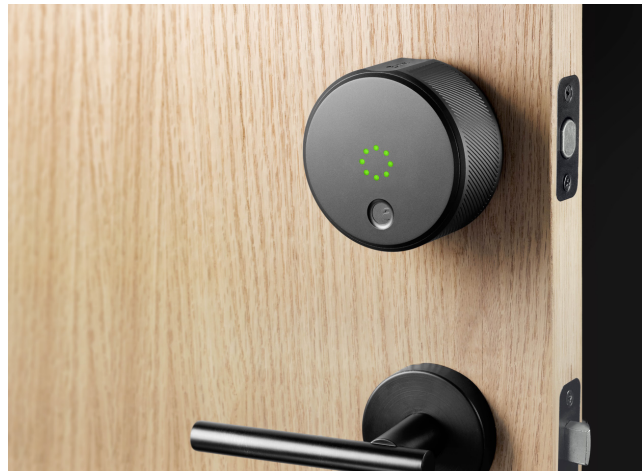


Figure 1.5 August Door Mount

grant unlimited digital keys to different friends. The hardware installed on the door, shown in figure 1.5 can also indicate whether the door is completely closed or not. Different versions with a range of prices are offered. Some of the higher-end versions include a camera for users to see who is at the front door and whether they want to unlock the door for them. Whomever you share the

code with needs to have the mobile application in order for the sensor in the lock to detect them when they are close [8].

1.3.2 Kwikset Kevo Locks

Kevo locks, shown in figure 1.6, are another brand of door locks highly rated by PCMag [7]. They offer similar features, but to unlock the door they use a digital keypad. Each user has its own access code, so the lock's accompanying mobile application can give a history of the lock's activity. For friends or temporary users needing to access the house, you can easily send a new access code and assign the time period for the temporary user to access the lock [9]. Since every user needs to use an access code to unlock the door, you can't unlock via proximity to your phone.



Figure 1.6 Kwikset Kevo Lock

2 Methodology

2.1 Introduction

The main goal of this project was to design and build a door lock system that allows users to unlock a door via face recognition, through a camera implanted on the door. In this methods chapter, we will discuss how we detailed the process of implementing this mechanism. We started our research by confirming the need of potential users for such a system and then followed a modified version of the Software Development Life Cycle (SDLC) approach to design and build a door lock system.

2.2 Why We are Choosing this Strategy

SDLC is a strategy that is used to ensure that products that are developed, are optimized for their users, based on a set of requirements. It is a very common strategy in software development projects. SDLC involves several steps shown in Figure 2.1: Requirements Gathering, Analysis, System Design, Object Design, Testing, and Implementation. Each step ensures that the developers are ready for the next one and the approach tries to minimize the development time by having predefined expectations for each step. By following this process, the developers ensure that their finished product actually meets and addresses the needs of their users.

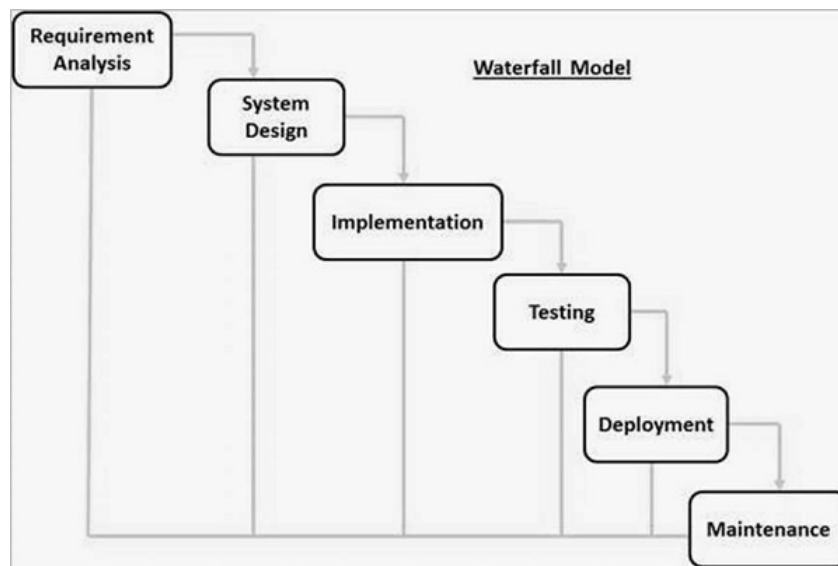


Figure 2.1 Software Development Life Cycle Phases

2.3 Requirements Gathering

The objective of this step is to obtain information about the potential need for such a door lock system as described in Section 2.1. Before we designed a solution to an acknowledged problem, we needed to know the specifics of user behaviors, which would especially help in designing the

features of our mobile application. Our background research indicated that smart locks are becoming a trend, but there is little research into:

1. What is the most common nuisance of all the traditional door lock systems
2. Whether people are ready to switch to a digital smart lock, completely controlled by their smartphones and with authentication based on facial recognition

2.3.1 Survey

Most of the motivation behind this project came from our own experience with traditional door locks and their inadaptability to the changing habits of modern time. Our first step involved discovering if others share the same experience with their door locks and inquiring about other potential issues that we had missed. The best way to get a quick idea into a potential problem was to survey people of our age about their habits. This provided some quick feedback which helped with the initial system and mobile application design. It also helped us in choosing a development platform: smartphone vs. computer vs. tablet; iOS vs. Android; etc.

The results from the 123 responses of our initial survey were very useful. The full scope of the answers can be found in Appendix C. Here were four of the most important observations:

1. Around 90% of people needed to physically go to the door whenever they had a visitor
2. A smartphone was people's preferred way to possibly unlock a door remotely
3. 50% of the people surveyed always or often find themselves with their hands full while trying to reach for the key to open a door
4. Around 65% of the people surveyed used an iOS device as their primary mobile device

2.4 Analysis

During analysis, developers aim to produce a model of the system that is correct, complete, consistent, and unambiguous [10]. From the initial survey we created user stories, use cases, and UI mockups that we then discussed with the team and project advisors. A complete set of user stories can be found in Appendix B. During the analysis phase, we ensured that there was no lack of understanding particular needs or possible features of the system we were building. Furthermore, by prioritizing requirements, we needed to create a scope of the project that should consider what we could build in the available time. Our analysis included:

1. a set of user stories created from requirement gatherings that will later turned into features of the application
2. Use cases, describing the system functionality from the user's point of view
3. UI Mockups, describing how each feature will look like on a screen

2.5 Design

This step of the Software Development Life Cycle process is concerned with the methodology behind creating an effective platform that is useful and appealing to our target population. This section starts with comparing a few software platforms and our reasoning on deciding to develop a mobile application for iOS, our reasoning for choosing Raspberry Pi as our on-board logical unit, and the methodology behind turning the system requirements into small building blocks.

2.5.1 Platform Choice - Mobile Websites vs. Mobile Applications

Websites and mobile applications are the two main ways of delivering a digital platform to a target audience. Before we started to design our features based on the needs of the surveyed community, we needed to decide on the most effective and useful platform for the targeted users. Table 2.1 gives a summarized comparison between these two basic methods.

Table 2-1 Comparison between a website and a mobile app [11]

Website	Mobile App
+ Easier to develop, websites are compatible across devices.	- More variability in development, because of different phone systems, types and operating system versions.
+ Easier to maintain.	- Difficult to maintain, because of manufacturer's constant system upgrades.
+ Instantly available, by providing the URL.	- Need download from respective application store first.
+ Broader reach because they are instantly available and compatible across devices.	+ Personalization - If target users have personal information and usage, then an app offers better experience, including push notifications
- No push notifications	+ Push notifications for real-time user response and information
- Less integration with specific device resources.	+ Easier access to device resources, like GPS or camera sensors.
- Relies on an active internet connection.	+ Offline access.

Table 2.1 offers a good summary of the most important advantages and disadvantages of each method, but we could only make a decision by factoring in our system interactions. For example, having interactive push notifications is necessary for our needs, so that the user can easily see who is at the door and open it with a single tap. Furthermore, easier access to device resources, like GPS and camera, is integral to our needs. The bolded points in the table above are the ones that helped us ultimately make the choice between a website and a mobile application.

2.5.2 Platform Choice - iOS vs. Android

A very important development decision was whether to create an app for an iOS or Android platform. There are software solutions that provide the ability to write and maintain a single code base for both iOS and Android. React Native is such a tool. We have compiled a list of pros and cons for using React.

Table 2-2 A Comparison of React Native Pros and Cons [12]

React Native Pros	React Native Cons
+ Mostly write a single code base	- Still need to maintain parts of Objective-C (iOS) and Java (Android) code base, as not everything translates properly
+ React uses JavaScript, which is fast and popular	- Lack of documentation, as it is a young platform
+ Uses Native modules whenever it can, which translates into fast and seamless experience	- Lack of third-party community-built components, which cannot be guaranteed to work in next OS releases
+ Great for quick UI design	- Still in development: Instability, compatibility issues, and errors

Although we wanted both an Android and iOS app, as that would increase our product's exposure and the speed of adoption, if it ever goes to market, React Native did not sound like a safe choice, considering especially our lack of experience with the tool and all the potential instabilities, particularly with third-party component integration. As we can't develop our app for both platforms, we made sure to include a question on the survey about what platform each of our respondents uses. The mobile OS with the most votes was iOS and that is why iOS is our platform of choice.

2.5.3 Platform Choice - Raspberry Pi vs. Arduino

For the purposes of our systems, a computational device is required. This device needed to fulfill a series of requirements, which we divided in three separate categories, Development, Network connectivity, and physical ports. Each of the following conditions have been defined based on the system that makes the features of our platform possible with the most convenient development environment.

The three development requirements that we looked into were as below:

- Ability to run a scripting language
- Extensive community support
- Availability of documentation

The development requirements were essential to successfully achieve the project goals in time and to create a platform that can be easily maintained. A device that uses a scripting language such as Python will allow the integration of multiple subsystems that are required in our product. Having to work with cloud systems, a camera and different types of motors, makes a scripting language the most favorable choice. Because this device will be the controller for almost every component in our platform, good documentation and support will make the development significantly easier.

The three Network and Connectivity requirements that we looked into were as below:

- On-board WiFi connectivity
- On-board Bluetooth connectivity
- Ethernet connectivity

Because the system needed to connect to cloud services to process the authentication of a user and their device, as well as video streaming, network and wireless connectivity stand at the core of our product requirements.



The two I/O ports that we thought of as crucial were as below:

- A port for the camera connection
- Ports for motor connection

With the implementation of camera to provide a live feed of the door view and motors to operate the lock and the door itself, the computing device needed to be able to operate all the hardware that make these functionalities possible.

After determining the requirements for the computational device, a market and specifications research was conducted. A few products from popular providers such as Raspberry Pi and Arduino were taken into consideration. Table 2.3 offers a side-by-side comparison between the specifications of the most popular microcontrollers in the market.

Table 2-3 A comparison between Raspberry Pi and Arduino Uno [13]

Raspberry Pi 3	Arduino Uno
	
Stronger and quicker processor, multitasking available	Can run one code at a time, so can't multitask activities, slower speed
Built in Ethernet port, Wi-Fi and Bluetooth capability	No internet connectivity (requires shield or module)
OS can be switched easily, scripting language used (Python)	Bigger learning curve since it uses C/C++ language and need outside sources
Audio output, Camera port, USB ports, HDMI output included	Requires shields to implement most of the I/O devices

Requires installation of programs to get simple actions going	Easier to connect to analog sensors, motors and other electronic components
---	---

From the comparison of the features offered from the Raspberry Pi and the Arduino boards, we determined that the Raspberry Pi, unlike the Arduino, is more suitable for projects that involve several functionalities at the same time, need easy access to the internet, and need media accessibility. On one hand, the Arduino fits better in projects that require easy reads from sensors and have to do operations based on the data, while being able to communicate easily with a variety of motors or machine parts. However, this board does require extensions in the form of shields, to expand on features that are required to accomplish this project. Therefore, the Raspberry Pi was the most suitable board for our system.

2.5.4 Platform Choice - Motion Sensor Ultrasonic vs PIR

To automate the operation of the door lock system a motion sensor is required. The sensor is used to detect any activity that will be happening in front of the door, like a person approaching it. Therefore, the sensor required has to be able to detect movements within a reasonable controlled area. The detection range needs to be limited to prevent the system from being triggered by background activity.

2.5.4.1 Ultrasonic Sensor



An ultrasonic sensor is generally used to measure the distance of an object using sound waves. The sensor consists of a transmitter and receiver pair. The distance can be measured by taking into account the elapsed time between the sound wave being generated and the sound wave bouncing back to the receiver [14].

2.5.4.2 PIR Sensor

Passive Infrared Sensors are made of pyroelectric sensors, which can detect levels of infrared radiation. Everything radiates some low level of radiation. However, the hotter the object is, the more radiation is emitted. The sensor in a motion detector is split in two halves in order to detect change between the two. The two halves work with each other by cancelling out normal IR levels in the background. When one half detects more IR radiation than the other, the output will swing high or low, therefore detecting motion. The heat released of the human body is enough to trigger this sensor, making it one of the most commonly used in security and lighting appliances [15].

Table 2.4 provides a side by side comparison between the Ultrasonic and PIR sensors. By comparing the characteristics of each sensor type it was determined that the PIR sensor is more suitable for our product because of its higher accuracy in the types of environment it will be exposed to, as well as the wider detection angle.

Table 2-4 A comparison between an ultrasonic and a PIR sensor

Type	Ultrasonic	PIR
Image		
Range	2-400 cm	2.2 m
Angle	30 degrees	90 degrees for 1 m/s 44 degrees 0.5 m/s
Type of Info	Time that can be converted to distance	HIGH, LOW
Price (\$)	3.95	12.62

2.5.5 Platform Choice - Actuator

The implementation of such a door lock system involved some mechanical engineering in order to make the door adaptable for people with disabilities and to offer the feature of fully opening and closing the door via commands from the mobile application, or just by approaching, i.e. hands-free. In order to achieve this functionality, we used an actuator. There are many different types of actuators and during the design phase of the project we decided which one is the most appropriate for our needs. Table 2.5 offers an overview of the advantages and disadvantages of each type of actuator we considered: pneumatic, hydraulic, and electric actuators.

2.5.5.1 Pneumatic Actuators

Pneumatic Actuators consist of a piston inside a hollow cylinder. Pressure from an external compressor moves the piston inside. With increasing pressure, a linear force is created that makes the cylinder move along the axis of the piston. The piston is then returned to its original position by either a spring or fluid being supplied to the other side of the piston [16].

2.5.5.2 Hydraulic Actuators

Hydraulic actuators consist of a cylinder or fluid motor that uses hydraulic power to facilitate mechanical operation. These are similar in operation to pneumatic actuators, but an incompressible liquid from a pump moves the cylinder [16].

2.5.5.3 Electric Actuators

Electric actuators convert electrical energy into torque. Usually, their inside mechanism is constructed out of an electric motor that turns a lead screw. When the screw rotates, the nuts get driven along the threads. The nut moves depending on the direction the screw rotates [16].

Table 2-5 A comparison among a pneumatic, hydraulic, and electric actuators

Pneumatic	Hydraulic	Electric
+ Simple and precise: a small pressure change can enable considerable forces (citation)	+ High force, 25 times greater than pneumatic cylinders of equal size	+ Precise and quiet: they generate linear motion within +/- 0.000315 inches
+ From ½ to 8 inches, with forces from 30-7500 lb	+ Efficient, can hold force and torque constant without supplying more pressure	+ Development: can be networked and reprogrammed quickly. Immediate feedback for diagnostics
+ Cost effective (\$50-%150)		
- Low efficiency due to air pressure losses and air compression. Additional regulators and valves may be needed, raising the costs.	- Leakage: this type will leak fluid, leading to maintenance issues	- Difficulty holding a locked position
- Maintenance: compressed air usually gets contaminated	- Complex: they need many companion parts, including fluid reservoir, motors, pumps, release valves etc.	- Expensive, price range from \$150-\$2000

Table 2.5 offers an overview on the main advantages and disadvantages that we considered when picking our actuator. Based on our initial studies of the system model, we chose an electric actuator, as we believed it to be the right one for our use, particularly because of its very easy development setup and connection to our electrical components of the door lock system.

2.5.6 System and Object Design

During this step, the analysis of requirements, as explained in section 2.4, are turned into well-defined parts of the software. The goal of this step was to give the team all the resources needed to start implementing the system. The products of this step included a few different kinds of diagrams, which built upon the previously defined User Stories, Use Cases, and UI Mockups.

2.5.6.1 UML Class Diagrams

Class diagrams follow a clean object-oriented approach to software development [17], by clearly showing the classes of the system, their interrelationships, and the operations and attributes of the classes. We created a class diagram for the mobile application, which indicated the different objects of a Door Lock: Users, Administrators, Friends of each user, and a representation of the Door Lock itself. The high-level class diagram of our system is shown in Figure 2.2. It shows the most important objects in our mobile app: User, Friend, Smart Lock, and Activity. It also shows the relationships among these objects, e.g each User can add many Friends and each SmartLock can have many Activities associated with it.

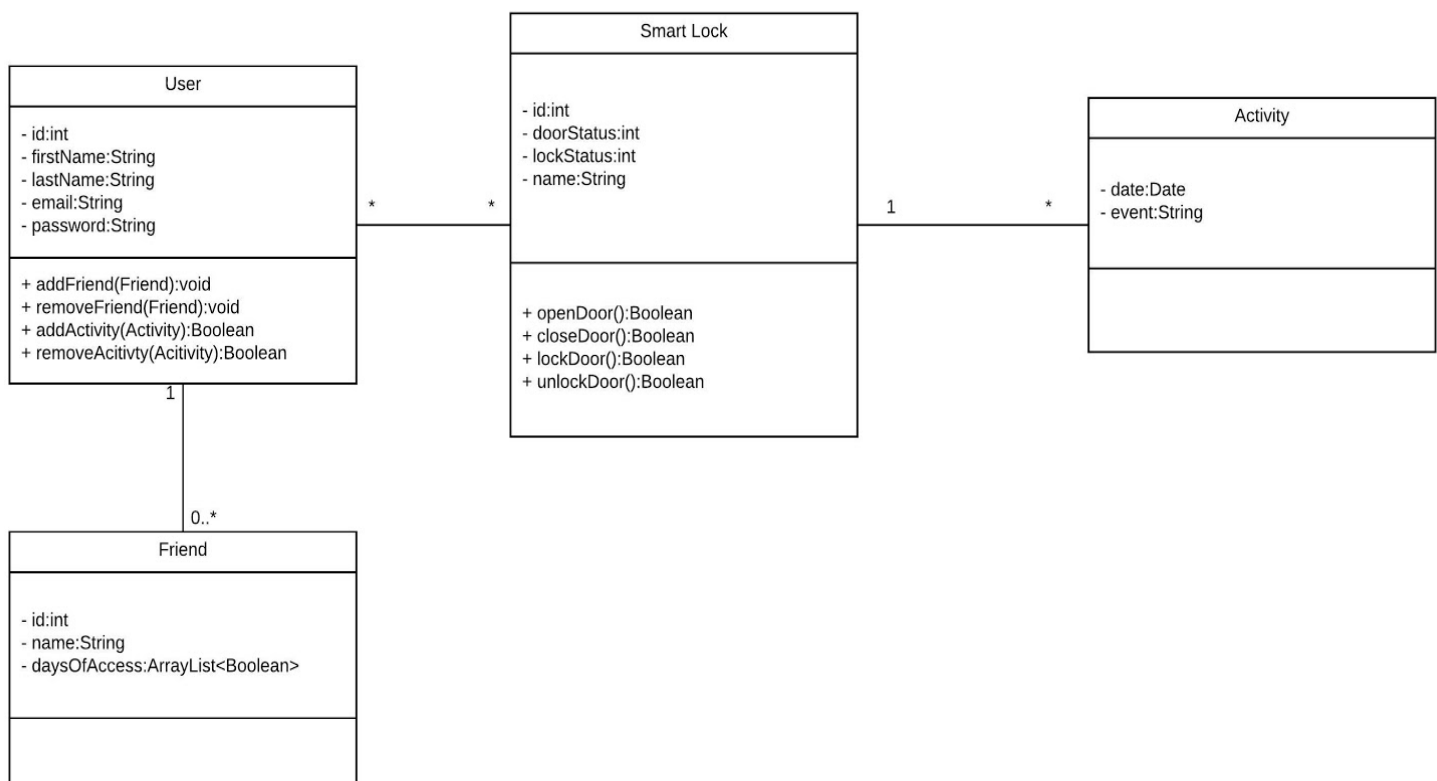


Figure 2.2 High-level class diagram of the Smart Lock main objects

2.5.6.2 Components Diagrams

With the Components Diagrams the team could see the interrelations among all the components of our system: Door Lock, the on-board logical unit mounted on the door; Mobile App, running on a mobile device connected to the Internet; the Cloud: an OS running our software, which is responsible for authenticating faces and sending commands from the Mobile App to the on-board unit. An initial ideation of such a diagram is shown in Figure 2.3.

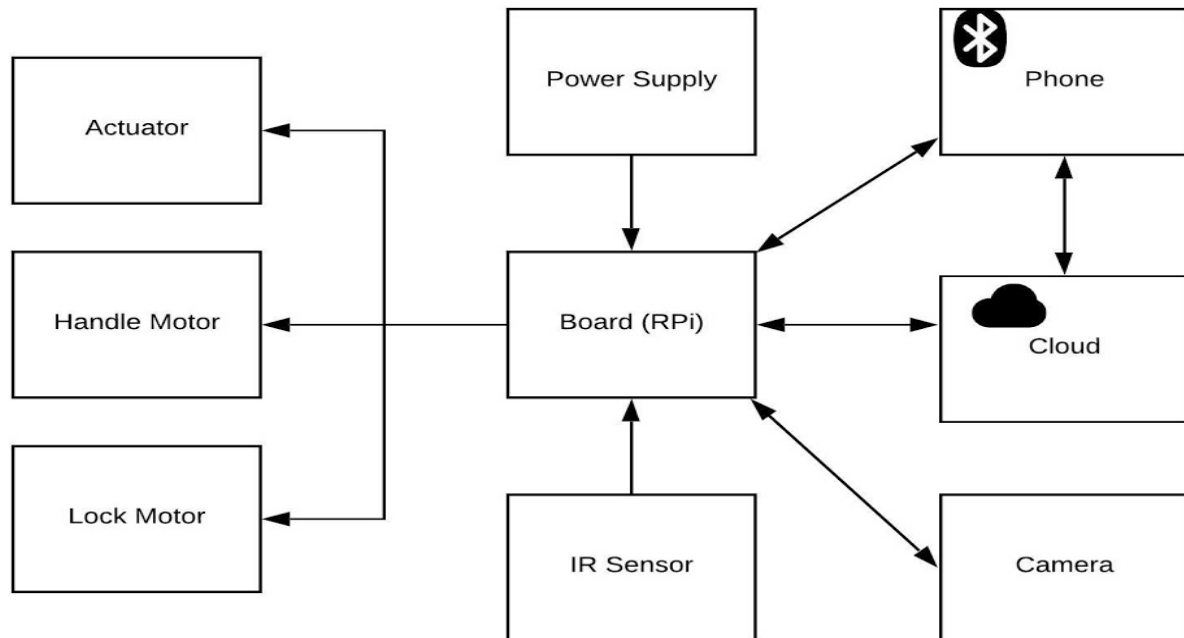


Figure 2.3 Components Diagram

2.5.6.3 State Machine Diagrams

State Diagrams are particularly useful for complex components who can't fully be represented in a Class Diagram. UML state machine diagrams depict the various states that an object may be in and the transitions between those states [18]. For our system, we are going to build a state machine diagram that indicates the different states of our on-board logical unit, which will run an algorithm that controls the state of the door based on commands from the mobile app or presence of a person next to the door. An initial ideation of a state machine diagram is shown in Figure 2.4.

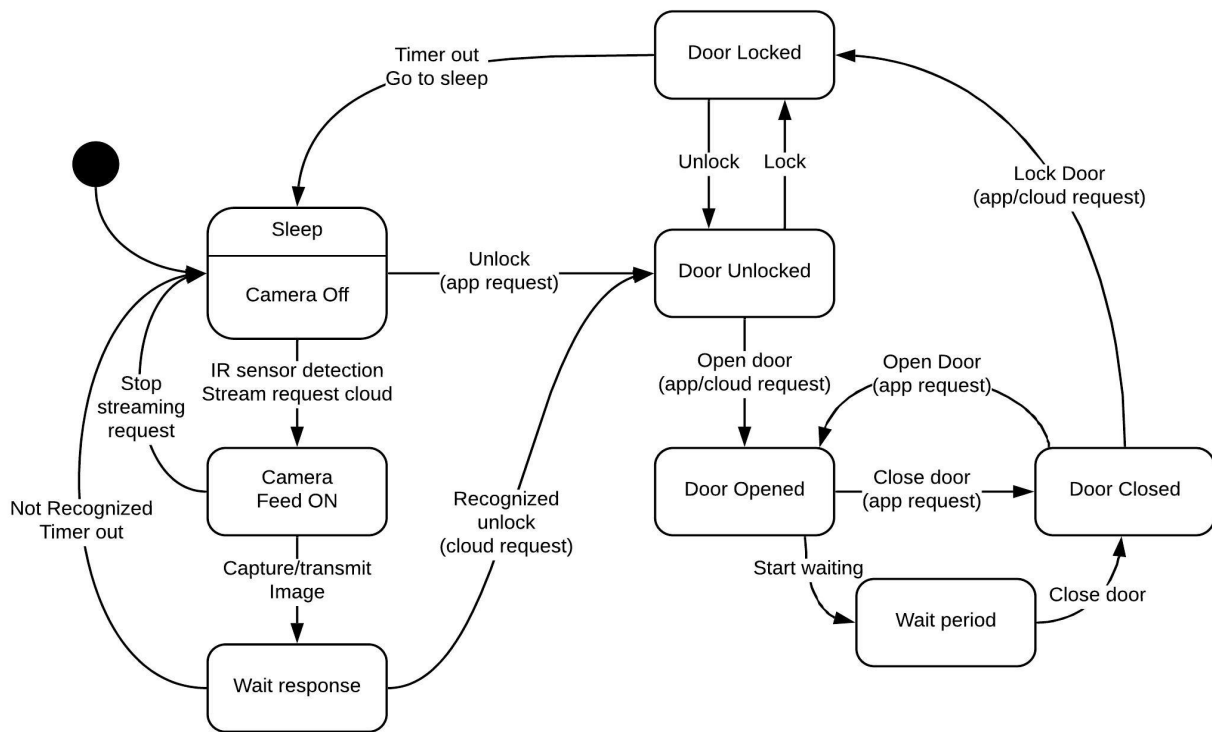


Figure 2.4 State Machine Diagram for the Smart Lock system mounted on the door

2.6 Implementation and Testing

After Requirements Gathering, Analysis, and Design, there is the Implementation phase. In this step, the deliverable product was implemented. For the software part of our system, during this step, we translated the solution domain into source code, which included implementing the attributes and methods of each object and integrating all the objects such that they function as a single system [10]. For the hardware part of the system, during this step we implemented the state diagrams on a Raspberry Pi and connected the unit to the Cloud and paired it with a mobile application, from which it will receive commands. For the mechanical engineering part of our system, we used an electric actuator, modified it according to our needs by using 3D printed parts and installed it on a simple made-up door. This step took into account all previous design decisions we had made in precedent steps: tools like programming languages and face recognition software, platforms like mobile operating systems, and hardware choices like the on-board logical unit. Most of these design choices should not be changing in this step, although minor improvements were needed. Whenever we had to change a considerable amount of the previously defined models, then we reiterated through all the steps of SDLC. In this section, we talk about the implementation of each of the three main components of our system, shown in Figure 2.5. We conclude with a subsection on how we connected all these components together.

An important part of this section is our discussion on what changed from the submitted proposal and the reason behind the change.

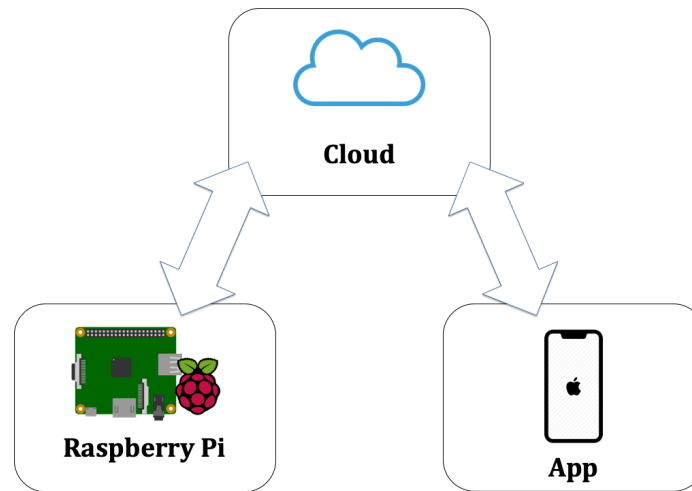


Figure 2.5 The three main components of the Door Lock System

2.6.1 First Component - Smart Lock App

Table 2.6 shows a set of features that we wanted the Smart Lock app to support. The “Must Have” column shows a set of features that we submitted on our project proposal. The column marked all the features that we deemed important at the beginning and that were very important for our application to have. The “Good to Have” features were also submitted in the proposal and the column marks all the features that we would implemented if time permitted. We underline all the features that we were not able to implement due to time constraints.

Table 2-6 Smart Lock App features submitted in proposal. Underlined features were not implemented

Must Have	Good to Have
Common UI/UX patterns, used in similar apps (familiar application design, for high adaptability)	
User personal account, including verification email and <u>alternative sign up with a Facebook or Google account</u>	Support for Admin level - admin users can add/remove other normal access users
Open/Close and Lock/Unlock door commands	
Livestream of the camera in the door lock	<u>Speaker and Microphone - for communication with different visitors while they are at the door</u>

Activity History - a list of events/commands accessible by the user	
Support for one Door Lock Device	<u>Multiple “Smart Door Lock” devices - ability to add multiple door locks to the same profile, for different doors inside the same residence or other residences</u>
Friends - list of people with moderate access, as defined by the user	
Push Notifications - informational notifications about people being at the door	<u>Ability to open/close and lock/unlock doors right from the notification, without opening the app</u>

Table 2.6 clearly indicates that we were able to implement all the “Must Have” features, while also providing support for Admin level, a “Good To Have features”. For a detailed description of what each features does and how to use it, see the Smart Lock User Manual in Appendix X. In the User Manual, we also provide screenshots of the app indicating each feature.

As our survey indicated (Appendix C), 65% of our surveyed population indicated that they use an iOS device, which is why we chose iOS as our mobile platform, as explained in Section 2.5.2. The app was developed in Swift 4, using the XCode development environment. For the full mobile app code, see Appendix D.

2.6.2 Second Component - Raspberry Pi

The Raspberry Pi was our on-board logical unit of choice, as explained in Section 2.5.3. The RPi is responsible for the below functionalities:

1. Control the servo motor and actuator via commands coming from the mobile app
2. Check the PIR sensor for movement and using the camera to snap a picture of the person who is at the door, whenever one is detected
3. Send the picture to the Cloud and receive response: if a positive one, control the servo and actuator to unlock and open the door
4. Easily connect to an available Wi-Fi connection in a user-friendly way

Figure 2.6 shows how each electrical component is connected with the RPi.

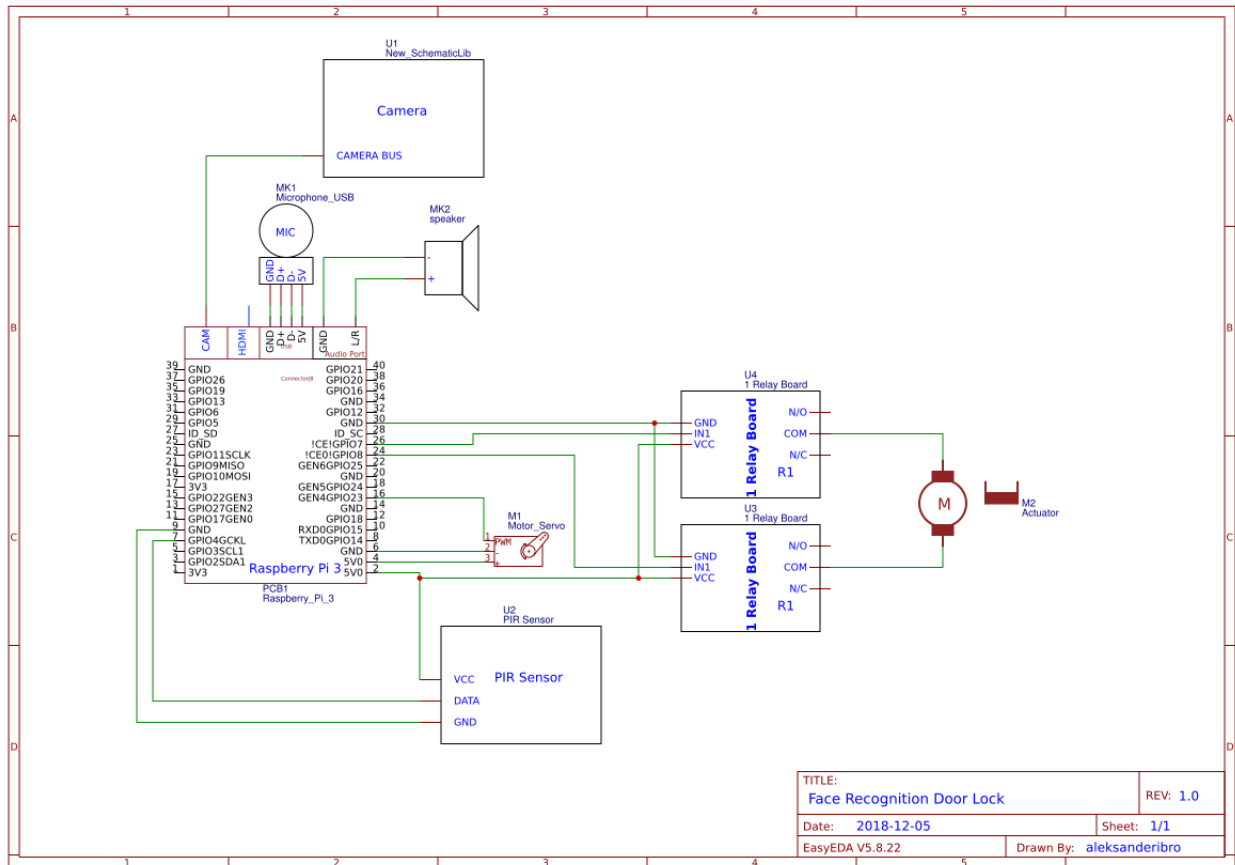


Figure 2.6 Electrical Schematic of the Smart Door Lock device

An important issue that we faced while working with the Raspberry Pi was finding a user-friendly way to get the Raspberry Pi connected to a certain user's home Wi-Fi connection. We solved this by designing the RPi in such a way that whenever it started for the first time, it would try to connect to a mobile device via Bluetooth. On the phone's side, the Berrylan app (cite Berrylan) would make sure to detect the open Bluetooth connection. Once a user connects to the RPi's Bluetooth using BerryLan, which is easily available in the App Store, the user can manually choose their SSID and type in their home Wi-Fi password, which in turn get sent to the Raspberry Pi. Once the RPi connects to the provided SSID, it shuts down the Bluetooth connection and starts communicating with the Azure Cloud. For a closer look at how BerryLan connects the RPi to Wi-Fi, see the Smart Lock User Manual in Appendix A.

In order to satisfy all of the objectives above, we wrote a Flask server in Python which starts just in the moment the Raspberry Pi turns on. The Flask Server activates all of the components. The way a Flask server is written in Python is by providing different "routes". To hide the dynamic IP changes, we used a service called No-IP (cite no-ip), which let us assign a hostname to the RPi. Throughout the project, instead of using the IP address for requests sent to the RPi, we used that assigned hostname. Each route in a Flask server corresponds to the appendix of each URL. For example, in a request sent to RPi with the URL "http://doorlockmqpboard.ddns.net/open_door" the "open_door" route, with the Python function that follows it, will run. For the full Flask server

code running on the RPi, see Appendix D. The routes are called from the Smart Lock app via the respective buttons to unlock and open the door.

The PIR sensor on the Raspberry Pi is always active. It is facing the outside of the door and it waits to detect a close-range movement. Once the movement is detected, the PIR sensor delays any other execution, until the person has stopped moving. At that point, the camera takes a snapshot and sends the image to the Cloud VM, another Flask server with a specific URL. The Cloud responds back with either a negative identification, i.e the person is not added to the system and the face wasn't recognized, or a positive identification, i.e. the person is known and the system should let them in.

If we detect a positive response, then the RPi activates the servo to unlock the door and right after that it activates the actuator to fully open the door. Once the door is opened, after an adjustable delay, the door automatically closes and locks when it has fully done so.

2.6.3 Third Component - Cloud Backend

For our cloud services provider we chose Microsoft's Azure. The Azure backend was made up of three different parts: the Virtual Machine in the cloud, the SQL database, and the App Services platform, and we discuss each part in detail in the following sections. For user login and account management, we used Google's Firebase Authentication.

2.6.3.1 Azure Virtual Machine (VM) in the Cloud

The first step into building our backend was to create a Linux VM hosted by Azure in the Cloud. We created the Linux VM using Azure's default parameters. This VM was designed to be an always running system, and once it starts, it hosts a Flask server. This Flask server is set up similarly to the one in the Raspberry Pi (see Section 2.6.2), but the routes look different and they provide different functions. The main responsibility of this Flask server is to run the face recognition software and identify each picture uploaded by the Raspberry Pi of the person who is at the door, by comparing it to all the other pictures uploaded in the system.

Whenever a user logs in, the app contacts the Flask server and grabs all the pictures of a user's added friends and loads them into the app. These pictures are stored locally in the VM.

Whenever the Flask server gets a request from the Raspberry Pi, it compares the pictures to all the ones added for the requesting device, and it sends back a positive message if it can identify the user in the picture, or a negative one if it cannot identify a user's face. The Flask server is also responsible for sending a push notification to the user's logged in and paired with the specific device. The push notifications can let users know that someone is at the door. Figure 2.7 shows the complete workflow of the "upload_image" route of this Flask server, which is the main route that serves the Raspberry Pi.

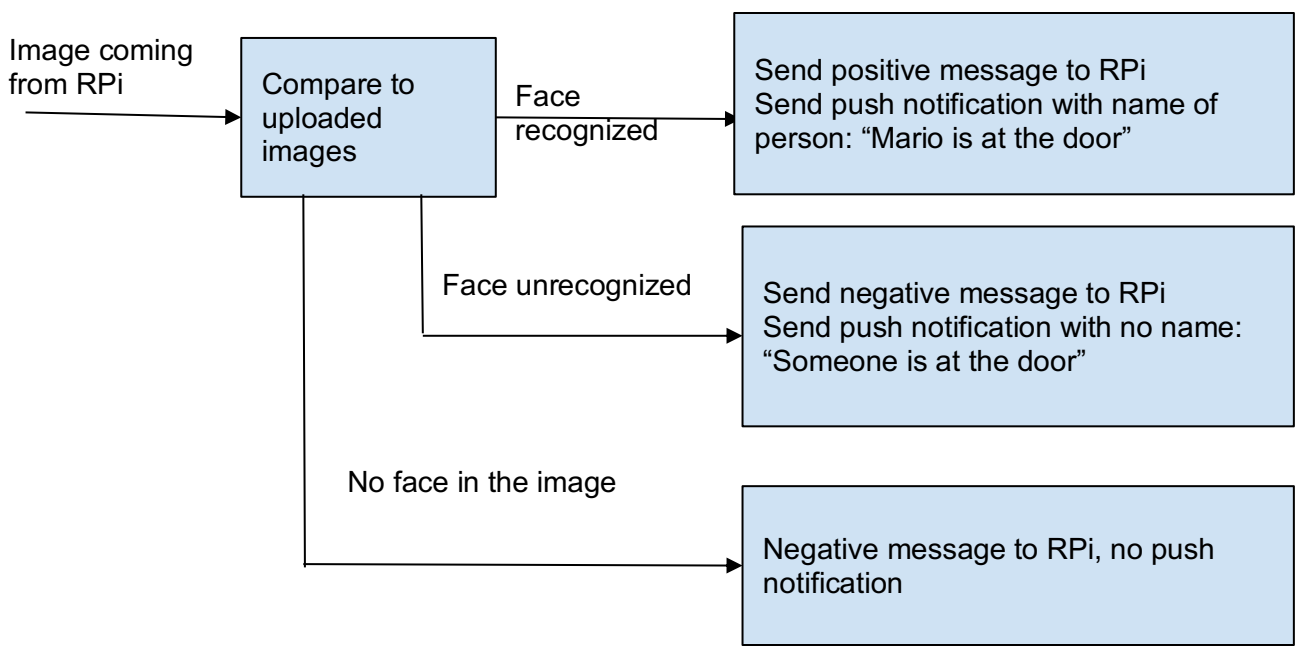


Figure 2.7 The "upload_image" route that runs in the Flask server when an image is sent from RPi

2.6.3.2 Azure SQL Database

Another Azure service that we use is the provided SQL database. All of the information for the class objects, i.e. User, Admin, Friend, and Activity is stored in the SQL database, excluding

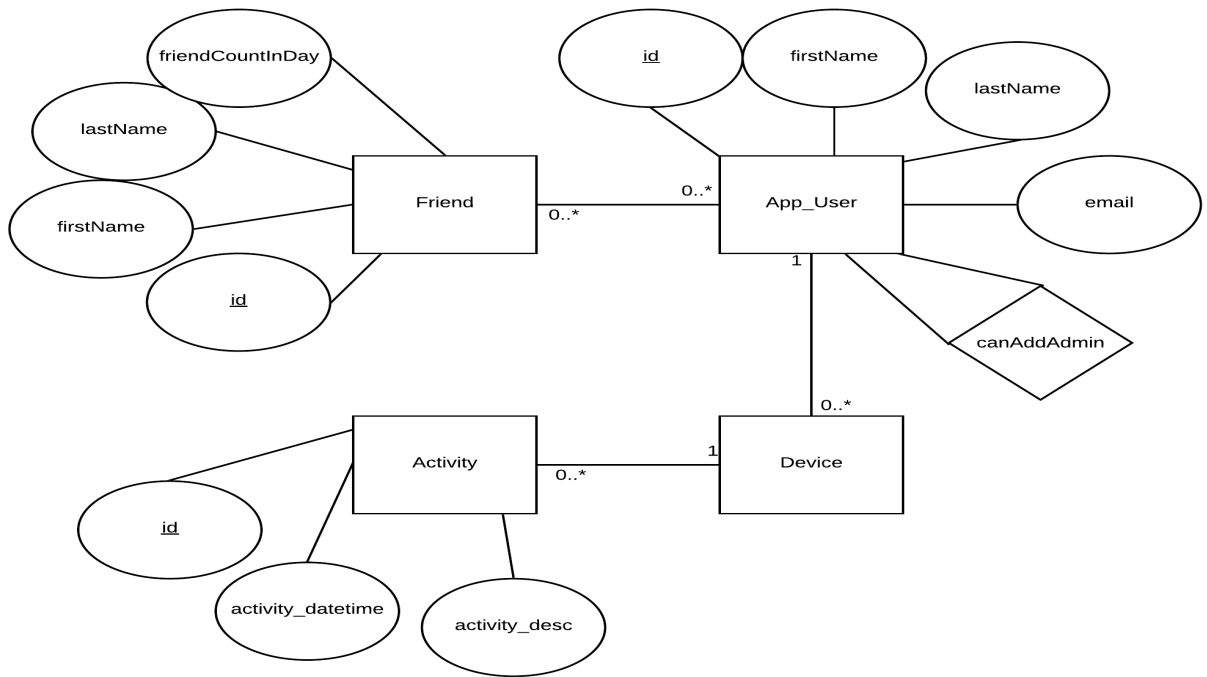


Figure 2.8 Entity Relation Diagram for the app

pictures, which are stored locally in the VM. Figure 2.8 shows an ERD of the database we implemented.

2.6.3.3 Azure App Services

App Services is an Azure tool that offers a backend for mobile apps, allowing quick development and deployment [19]. Initially, our cloud backend was provided by the Flask server, over HTTP requests to the SQL database. We realized this was very slow and not secure, as the it was over HTTP, which is why we migrated to Azure App Services. This allowed a quick connection via a token from the app to the SQL DB, thus avoiding request handling by the Flask server. Figure 2.9 shows the difference in the requests between requests being handled by the Flask server and requests being handled by App Services.

2.6.3.4 Google's Firebase Authentication

We wanted to provide the users of the Smart Lock app with an efficient and secure way to create and access their accounts. Google Firebase offers authentication services and provides a secure way, which avoids the need for us to write code for password management. It also offers users the ability to stay logged into the app and not have to log in every time they launch the app.

```
// adds the friend into the SQL database
func storeFriendInDB(_ friend: Friend){

    let parameters: Parameters = ["infoRequested": "postFriendToDB",
    "friendToAdd": [
        "id": friend.id,
        "firstName": friend.firstName,
        "lastName": friend.lastName,
        "comeInDays": friend.comeInDays,
        "doorNotification": friend.openDoorNotification,
        "imageName": friend.imageName,
        "belongsToUserId": currentUserID! ]

    let url = "http://doorlockvm.eastus.cloudapp.azure.com:5000/sqlQuery"

    Alamofire.request(url, method: .post, parameters: parameters, encoding:
    JSONEncoding.default).responseString { response in
        print("My response")
        print( response )
        if response.result.isSuccess {
            // create the user with the DB info
            print(response.result.value!)
        }else{
            // SHOW ERROR MESSAGE
        }
    }
}
```

```
# Store a friend into the DB
if infoRequested == "postFriendToDB":
    friend = iphoneRequest.get("friendToAdd")
    daysString = ""
    listOfBool = friend.get("comeInDays")
    for dayBool in listOfBool:
        if dayBool:
            daysString += "T"
        else:
            daysString += "F"
    # get friend info
    friendId = int(friend.get("id"))
    friendFirstName = friend.get("firstName")
    friendLastName = friend.get("lastName")
    friendComeInDays = daysString
    friendDoorNotification = 1 if friend.get("doorNotification") == True else 0
    friendImageName = friend.get("imageName")
    userFriendBelongsTo = int(friend.get("belongsToUserId"))

    # query for DB
    query = "INSERT INTO Friend VALUES (%d, '%s', '%s', '%s', %d, '%s', %d)"
    % (friendId, friendFirstName, friendLastName, friendComeInDays, friendDoorNotif
    ication, friendImageName, userFriendBelongsTo)

    print("Query to store friend", query)
    cursor.execute(query)
    cnxn.commit()
    return "Success"
```

```
// adds the friend into the Friend database
func storeFriendInDB(_ friend: Friend){

    let friendItem: [ String: Any] = ["friendFirstName": friend.firstName,
    "friendLastName": friend.lastName,
    "friendCountInDays": friend.getComeInDaysStr(),
    "friendDoorNotification": friend.openDoorNotification,
    "friendImage": friend.imageName,
    "userId": currentUserID! ]

    let azureClient = myAppDelegate.client
    let itemTable = azureClient.table(withName: "Friend")
    itemTable.insert(friendItem) {
        (insertedItem, error) in
            if (error != nil) {
                print("Error" + error.debugDescription);
            } else {
                print("Friend inserted")
            }
    }
}
```

Figure 2.9 Comparison of adding a new Friend to the DB using the Flask Server (left) and using the App Services (right). The Flask server code (bottom left) is completely gone after migrating to App services.

3 Results and Conclusions

Our MQP team successfully implemented a Smart Door Lock system with a seamless integration of a cloud backend, an on-board logical unit like the Raspberry Pi, and a user-friendly mobile application. All of the features initially proposed, except for a Facebook login, were implemented and tested, including some extra ones, as explained in Section 3.6. We made sure that the requests and users' information is secure by using Microsoft's Cloud and their App Services tool. We were able to build a mockup as a door system, using some wooden platforms and WPI's 3D printers. Figure 4.1 shows an image of the finished Smart Lock system mounted on the mentioned platform.



Figure 3.1 Smart Door Lock system mounted on a mockup door

3.1 Future Work

Limited by the available time and the complexity of integrating all three components of our system, we were not able to implement most of the proposed “Good to Have” features, as shown in Table 3.6. Out of all “Good to Have” features, we believe that the most important one to be implemented in the future is adding support for multiple Smart Lock systems. Although our backend is set up for this support, the mobile app and the Raspberry Pi Flask server do not support such a feature.

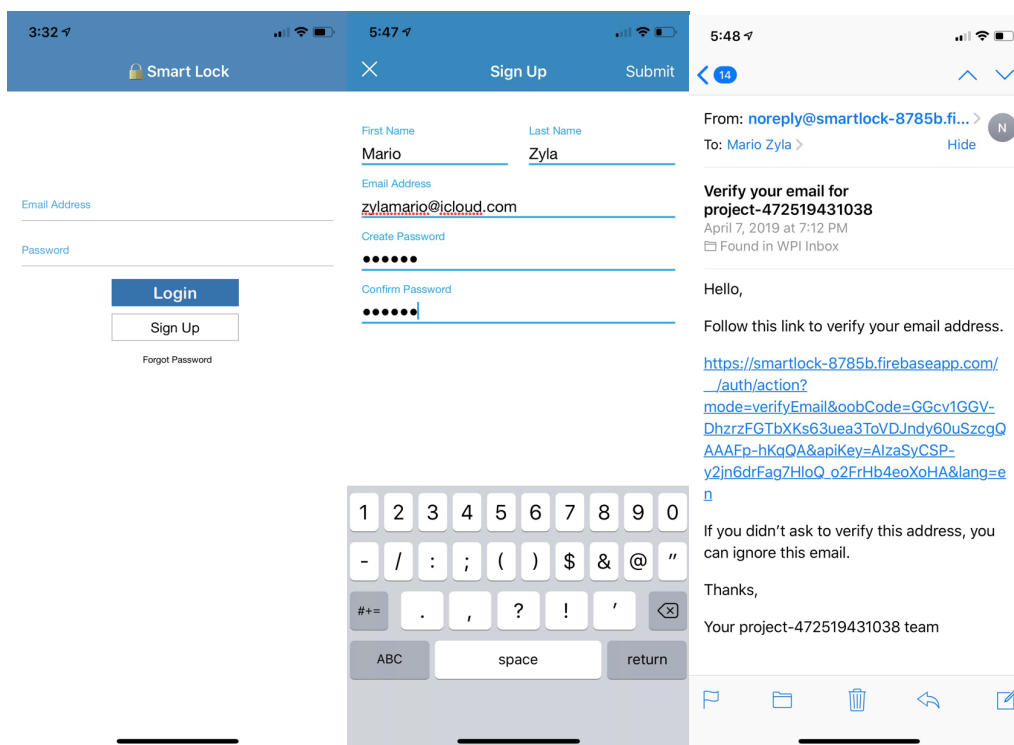
Another major feature would be the addition of a microphone and speaker on the outside of the door, right next to the camera. This feature would allow visitors to communicate with users, in cases when they don't recognize the person visiting.

Appendix A: User Manual

Getting Started

In order to use the application you will need to first create an Account.

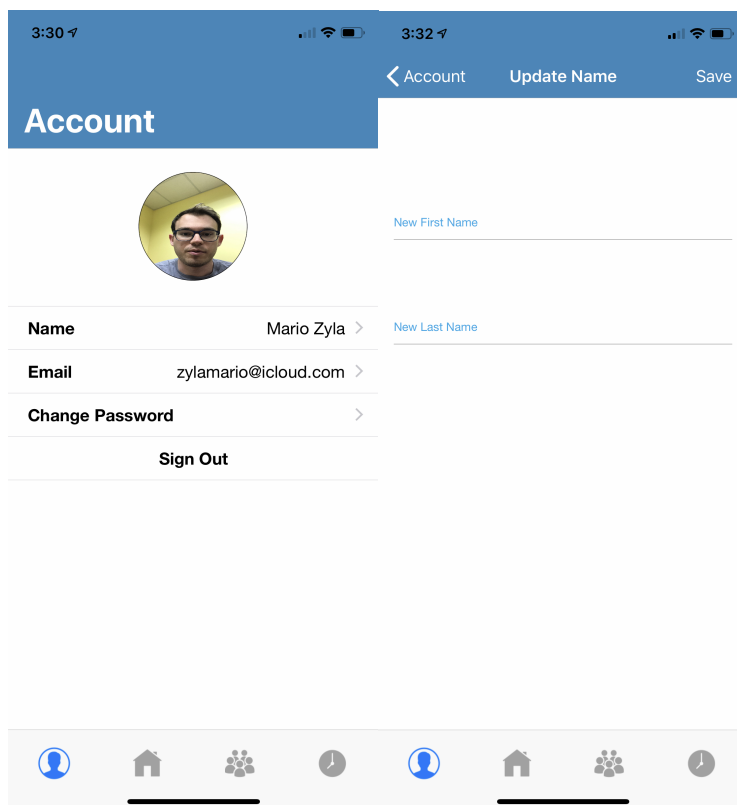
- 1) Click the “SIGN UP” button
- 2) Fill all the required information, and click the “SUBMIT” button.
- 3) Check your email inbox for a verification email in order verify your email address



Adjust your Settings: Account Tab

Once you have created an Account and you have successfully login, you should update your Account picture in order for the door lock system to recognize you when you are at the door. This can be easily done in the Account tabs, which is the leftmost tab, by clicking on the temporary image and then taking a picture of yourself. At the Account tab, you can also update your name, email address and password. Lastly, you will be able to sign out from the application, by clicking the “SIGN OUT” button.

1. Click on the leftmost tab in order to access your Account information
2. Click on the temporary Image in order to update your profile image
3. You can click on Name, Email, or Change Password to update your information as needed
4. Click on the “SIGN OUT” button to sign out



Who is at the door: Home Tab

At the Home Tab, the second left most tab, you will be able to see useful information with regards of your door system. At the top left corner, you will be able to see the current status of the door. Furthermore, If someone is at the door, you can click on the “Live” button, at the top right corner, in order to get a livestream from the door and thus see who is at the door. Once you are ready you can click on the “UNLOCK DOOR” button, and then click on the “OPEN DOOR” button to let someone inside. Once your friend is inside the house, you can click on the “LOCK DOOR” button and lastly the “CLOSE DOOR” to properly close the door system.

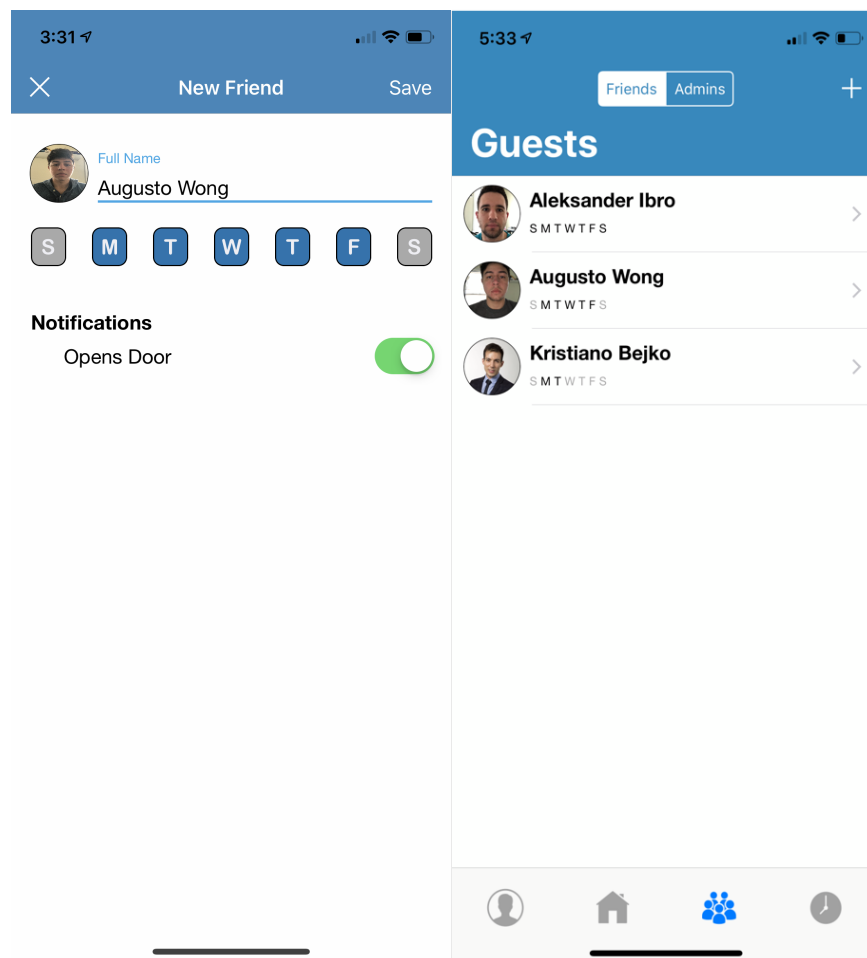
1. Click on the “LIVE” button to get a livestream from the door.
2. Click on the “OPEN DOOR” button to activate the actuator and open the door
 - a. Then you will be able to click on the “CLOSE DOOR” button to activate the actuator and close the door
3. Click on the “UNLOCK DOOR” button to turn the servo clockwise and unlock the door
 - a. Then you will be able to click on the “LOCK DOOR” button to turn the servo counter-clockwise and lock the door



Add Friends: Guests Tab

At the Guests tab, the second right most tab, you will be able to see and add new friends. These friends will have access to your door system on the selected days, from Saturday to Sunday, and you can decide if you would like to receive push notification from each of them. When you create a friend, make sure to take a good picture of their faces in order to guarantee that the door system will be able to recognize your friend when needed. As of now, there are no limits with regards on how many friends a user can have, so feel free to add as many as you want! You will be able to update your friend's information by simply clicking a friend. Lastly, you can delete friends by simply swiping left on the desired friend to be deleted.

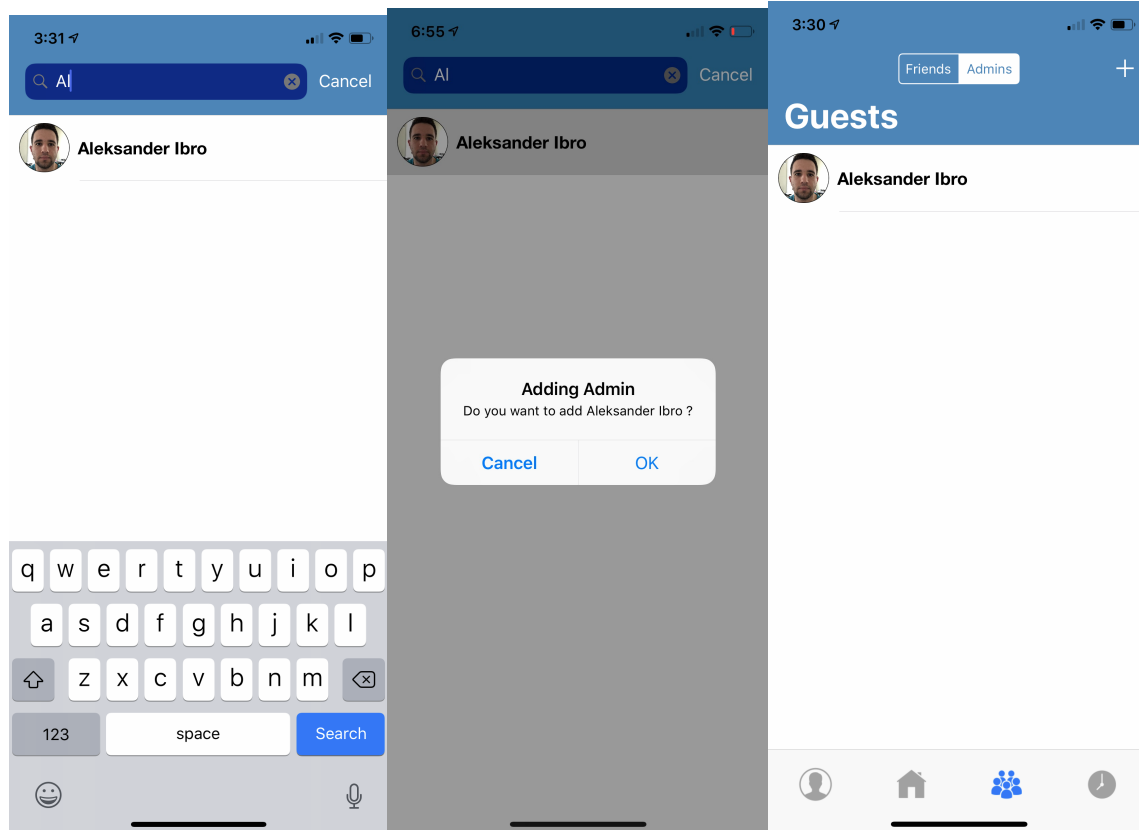
1. Click on the "+" Button
2. Click on the "FRIEND" option
3. Fill the "New Friend" required information: Full Name, Access days, and profile picture
4. Click on "SAVE" button to add your new friend.
 - a. You can always click the "X" button to not add the friend.



Add Admins: Guests Tab

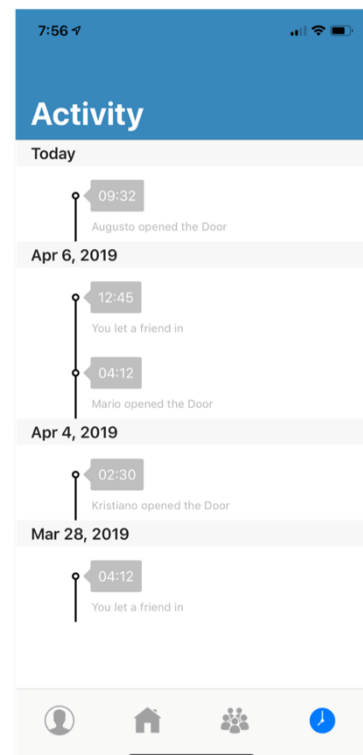
At the Guests tab, you will also be able to see all the Admins for your door system by clicking on the “ADMINS” Button at the top of the screen. Admins are users who have an account with the Smart Lock APP. These admins, once added, can add new friends and request to delete other admin’s friends. These added friends, will have access to your door system as explained before. In order to add a new admin you will simply need to know their names in order to look them up in the database. Lastly, you can delete admins by simply swiping left on the desired admin to be deleted.

1. Click on the “+” Button
2. Click on the “ADMIN” option
3. Click on the Search Bar and start typing the admin’s name
4. Once you have found your desired admin click on the name and then click “OK”



Check who has been at your door: Activity Tab

At the Activity Tab, the right most tab, you will be able to see a timeline with information about who was granted access to your door system, if you let someone inside, and if someone was outside the door. All of them with their respective time and date of the event.



Appendix B: User Stories

During the Requirements Analysis phase, we created user stories which in the Implementation phase turned into application features, as explained in sections 3.4 and 3.6. Below we list the main user stories that we created:

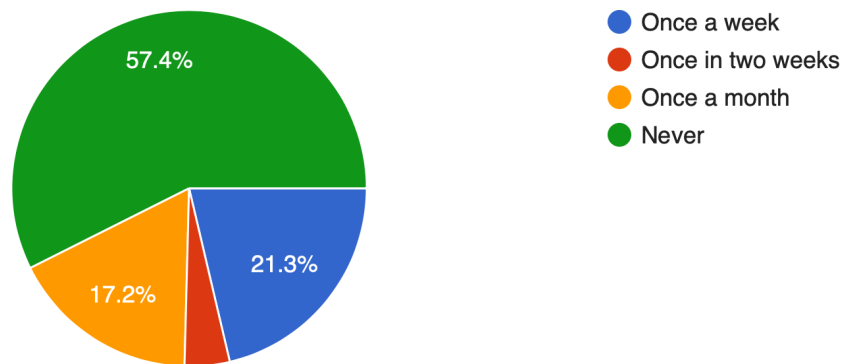
- As a User, I want to see a livestream from the door system so that I know who is outside
- As a User, I want to see a list of my Friends, so that I know who has access
- As a User, I want to specify which day my Friend has access.
- As a User, I want to be able to change my Friend's name
- As a User, I want to add new people on my list of friends so that they can get in
- As a user, I want the door to unlock/open for people I know so that I don't have to move and they can get in
- As a User, I want to unlock/open the door remotely with my phone so that my invitees can get in
- As a User, I want to lock/close the door remotely so that I don't have to move to lock/close the door
- As a User, I want to edit the different types of notifications I receive, so that I am always informed
- As a User, I want to edit notifications for each friend
- As a User, I want to delete a friend
- As a User, I want to see the status of my door, so that I know if it is unlocked/locked or open/closed
- As a User, I want to be able to log in with my credentials
- As a User, I want to sign up, so that I can have a new account for my Smart Lock App
- As a User, I want to edit the information of my account.
- As a User, I want to see the activity for my Smart Lock

Appendix C: Survey Results

At the beginning of the project, we sent a survey to our friends, WPI students and faculty, to determine the need for the system we were about to create. We received 123 responses and we have shown those results in this Appendix.

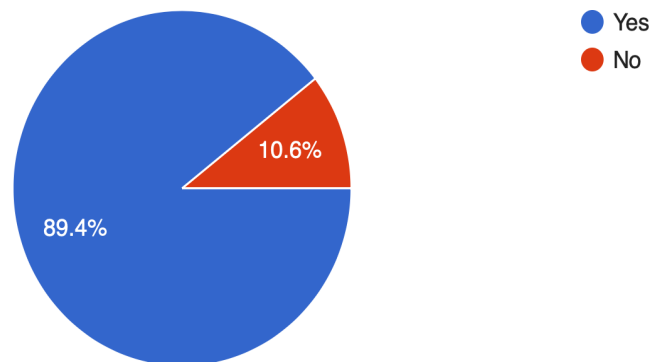
How often do you get locked out of your apartment?

122 responses



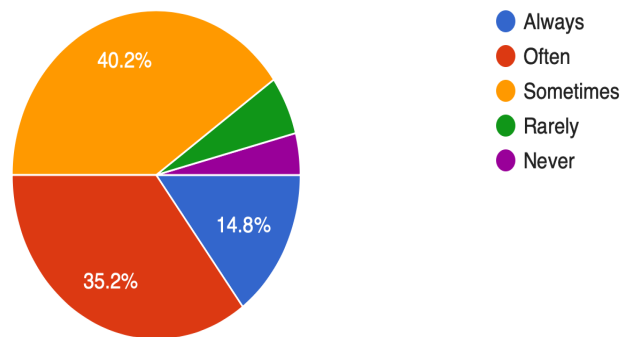
Do you have to go open the door each time someone visits you?

123 responses



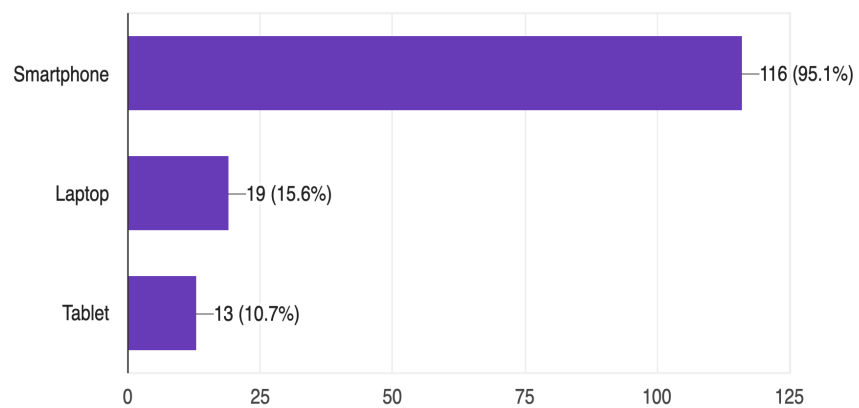
How often do you find yourself with your hands full while needing to open the door?

122 responses



What would you like to use to open your apartment's door remotely?

122 responses



Bibliography

- [1] G. Fortino, A. Guerrieri and W. Russo, *Middlewares for Smart Objects and Smart Environments: Overview and Comparison*, Springer, Cham, 2014.
- [2] C.-L. Hsu and J. C.-C. Lin, "An empirical examination of consumer adoption of Internet of Things services: Network externalities and concern for information privacy perspectives," *Computers in Human Behavior*, vol. 62, 2016.
- [3] L. Adler, "Data-Smart City Solutions," Harvard Kennedy School, 18 February 2016. [Online]. Available: <https://datasmart.ash.harvard.edu/news/article/how-smart-city-barcelona-brought-the-internet-of-things-to-life-789>. [Accessed 03 April 2019].
- [4] L. Columbus, "Forbes," *Forbes*, 6 June 2018. [Online]. Available: <https://www.forbes.com/sites/louiscolombus/2018/06/06/10-charts-that-will-challenge-your-perspective-of-iots-growth/#7b0e28d33ecc>. [Accessed 3 April 2019].
- [5] T. P. Norman, *Electronic access control*, Amsterdam: Butterworth-Heinemann, 2012.
- [6] J. Stern, "iPhone X Review: How We Tested (and Tricked) FaceID," *Wall Street Journal*, 31 October 2017. [Online]. Available: <https://www.wsj.com/articles/iphone-x-how-we-tested-and-tricked-faceid-1509465766>. [Accessed 4 April 2019].
- [7] J. R. Delaney, "The Best Smart Locks for 2019," *PCMag*, 15 April 2019. [Online]. Available: <https://www.pcmag.com/article/344336/the-best-smart-locks>. [Accessed 18 April 2019].
- [8] "August," *August*, [Online]. Available: <https://august.com>. [Accessed 3 April 2019].
- [9] "Kwikset," *Kwikset*, [Online]. Available: <https://www.kwikset.com>. [Accessed 3 April 2019].
- [10] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Pearson, 2009.
- [11] M. Zyla, C. Dickson-Burke, D. Kim and A. Karet, "Developing a Mobile Application to Reduce Risks for Runaway Youth in Bangkok, Thailand," March 2018. [Online]. Available: <https://digitalcommons.wpi.edu/iqp-all/2232/>. [Accessed 3 April 2019].
- [12] "The Good and the Bad of ReactJS and React Native," *altexsoft*, 10 September 2018. [Online]. Available: <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native/#pros%20of%20react%20native>. [Accessed 3 April 2019].
- [13] "Differences Between Raspberry Pi 3 vs Arduino," *EDUCBA*, [Online]. Available: <https://www.educba.com/raspberry-pi-3-vs-arduino/>. [Accessed 3 April 2019].
- [14] "What is an Ultrasonic Sensor?," *Keyence*, [Online]. Available: <https://www.keyence.com/ss/products/sensor/sensorbasics/ultrasonic/info/>. [Accessed 3 April 2019].
- [15] "Passive infrared sensor," *Wikipedia*, [Online]. Available: https://en.wikipedia.org/wiki/Passive_infrared_sensor. [Accessed 20 March 2019].
- [16] "What's the Difference Between Pneumatic, Hydraulic, and Electrical Actuators?," *MachineDesign*, 16 April 2015. [Online]. Available: <https://www.machinedesign.com/linear->

motion/what-s-difference-between-pneumatic-hydraulic-and-electrical-actuators.
[Accessed 15 April 2019].

- [17] "UML 2 Class Diagrams: An Agile Introduction," AgileModeling, [Online]. Available: <http://agilemodeling.com/artifacts/classDiagram.htm>. [Accessed 20 March 2019].
- [18] "UML 2 State Machine Diagrams: An Agile Introduction," Agile Modeling, [Online]. Available: <http://agilemodeling.com/artifacts/stateMachineDiagram.htm>. [Accessed 20 March 2019].
- [19] "Azure App Service," Microsoft, [Online]. Available: <https://azure.microsoft.com/en-us/services/app-service/>. [Accessed 20 March 2019].