

Report #3 Iteration #2: Home Security Automation

Group 2

14:332:452:01:00614 Software Engineering

Harmit Badyal

Nikunj Jhaveri

Abhishek Kondila

Kaavya Krishna-Kumar

Kaushal Parikh

Miraj Patel

Nirav Patel

Andrew Russomagno

Sagar Shah

Ashwin Suresh

April 14, 2019

Website: <https://sites.google.com/view/ruhome2019/home>

GitHub: <https://github.com/nikunjghaveri123/HomeSecurityAutomationApp>

Individual Contributions Breakdown

All members contributed equally towards this report.

Table of Contents

Individual Contributions Breakdown	2
Table of Contents	3
Summary of Changes	6
Customer Statement of Requirements	7
Glossary of Terms	10
System Requirements	12
Enumerated Functional Requirements	12
Enumerated Nonfunctional Requirements	16
User Interface Requirements	17
Functional Requirements Specification	19
Stakeholders	19
Actors and Goals	19
Use Cases	20
Casual Description	20
Use Case Diagram	23
Traceability Matrix	23
Fully-Dressed Description	25
System Sequence Diagrams	30
Effort Estimation With Use Case Points	35
Domain Analysis	38
UC-1: Take Picture	38
Domain Analysis UC-2: Facial Recognition	40
Domain Analysis UC-3: Light Control	43
Domain Analysis UC-4: Gyroscope is Triggered	46
Domain Analysis UC-7: System Control Features	48
Domain Analysis UC-10: User Profile Creation and Settings	50
Traceability Matrix	53
Interaction Diagrams	54
UC-1	54
UC-2	55
UC-3	56
UC-4	57

UC-7	58
UC-10	59
Class Diagram and Interface Specification	60
Class Diagram	60
Data Types and Operation Signatures	61
LoginActivity	61
FingerPrintHandler	63
FirstTimeUserSetup	63
MainControlActivity	64
AccessCamera	66
GridViewAdapter	67
Pictures	68
UserSettings	69
DeleteUser	69
NewUserSetup	70
AdminSettings	71
RegisteredFaces	73
RegisterNewFace	74
Photo	75
User	76
Uploads	77
Socket_AsyncTask	78
Traceability Matrix	79
Design Patterns	80
Object Constraint Language (OCL) Contracts	81
System Architecture and System Design	81
Architectural Styles	81
Identifying Subsystems	83
Mapping Subsystems to Hardware	84
Persistent Data Storage	84
Network Protocol	85
Global Control Flow	86
Hardware Requirements	87
Algorithms and Data Structures	87
Algorithms	87
Data Structures	88
User Interface Design and Implementation	89

Design of Tests	90
History of Work, Current Status, and Future Work	92
References	93

Summary of Changes

When we first started pursuing this Home Security Automation project we initially focused on the ability to give users access to control a multitude of devices such as cameras, lights, speakers, and motion sensors, in addition to other hardware components that can be added on at a later date. In conjunction with an Android App as well as enhanced facial recognition software, the system would be automated as to allow for the security system to require minimal human intervention and interaction. Once we started actually implementing our design and gave more thought to our project objectives, we realized that our project should place more emphasis on the impact of facial recognition and have that serve as the focal point of our Home Security Automation project. We decided to have the facial recognition be able to trigger our alarms and lights and also deactivate the security system as a whole, almost immediately upon taking a picture of an approaching intruder.

We also re-evaluated our need to give the base unit the ability to control the functions of these hardware components attached to the Arduino Base Unit. We realized that we are trying to modernize home security, so if we are relying on facial recognition to be the primary trigger point for the home security automation, there is no need to have control via the base unit as the mobile application should suffice as a very practical backup should the facial recognition falter. And after all, mobile technology is so ubiquitous and always in everyone's possession, that having all controls within the application should be more than plentiful for the average user, as this would significantly reduce costs for the typical user as well. Many of our use cases had to be adjusted because we decided to remove this hardware component to streamline our system as a whole.

In our desire to connect our Arduino to the Firebase, and give the arducam the ability to upload images directly to this database, we realized that Arduino itself in conjunction with the wifi shield would not suffice as the system would not be able to hold multiple high quality images and subsequently upload them to the Firebase. With this in mind, we decided to implement a Raspberry Pi which would be directly connected to the Arduino and also give the camera the ability to directly upload images to the Firebase, and also store images needed for the facial recognition software to execute and to have facial recognition software work directly within the Raspberry Pi, avoiding the need to have the software run on an external processing unit. Including the Raspberry Pi in our design affected several of our use cases as this would affect how the camera would communicate with the Firebase and then as a result affect how the user's mobile application got the image from the database.

Customer Statement of Requirements

As our world becomes more and more connected and open, the threat of intrusion and theft constantly burdens our thoughts. No matter what we do to protect ourselves and our property, there is always that fear in the back of our minds that we are not protected enough, and have left ourselves vulnerable to unwanted incidents. If cost was not a factor, and if accessibility was not an issue, everyone would without a doubt want to get their hands on an advanced home security system that would allow them to be more cognizant of household threats and be able to better manage devices in their own homes. There are a lot of products out there that that we could use to make our homes more secure, but it would be so much more convenient to have access to a singular application that could give us the ability to control multiple devices. Having a system consisting of a network of motion-sensors, accelerometers, cameras, lights, and speakers with the ability to work together, would without a doubt help scare off potential and actual threats. In addition to the security benefits that inherently come with the utilization of our product, we have taken it upon ourselves to go one step further and implement enhanced facial recognition technology to be able to better determine if unrecognized visitors to the house are low risk or are potential threats.

With our product, we aspire to make the consumer feel as safe as possible in their own home and to have them feel confident in their ability to protect their premises. Being able to have manual control of our homes from our wireless devices is a gamechanger, especially with how nearly everything today can be controlled by our smartphones. With our enhanced facial recognition feature, we want to make it easier for the user to be able to identify potential threats and also to be able to reduce the number of false alarms that the system may trigger. For example, we don't want systems going off multiple times in a day because of a person, whether it is the user itself or the user's friends or family, forgetting to disarm the system before entering the house. The facial recognition feature has the ability to filter people who come up to the front door, determine if they are a recognized person, and if not, upload the intruder/unknown individual's picture to the database, allowing for the user to extract the image from anywhere with internet connection, and if necessary, manually call emergency services. We believe that with our enhanced facial recognition feature, we will be able to reduce the number of home burglaries that occur as a direct result of the presence of our arducam in our home security system [3]. Even if our system doesn't deter threats, we will be able to capture images of the intruders, and with the cooperation of local police departments, we believe that you will feel much safer knowing that it will be easier to trace down perpetrators. With one singular application controlling various devices all on a singular network, and with the facial recognition

feature, we not only eliminate a majority of false alarms to the user, but also allow the user to become more aware of who is setting off the alarms upon entering the house.

Our system is comprised of a base unit, a mobile application, and a motion sensor/accelerometer that can trigger various devices, including lights, an alarm, and a camera. The camera will be able to take a picture whenever the motion sensor detects movement. Ideally, the camera would be installed within the door, facing the point of entry, in order to potentially capture the intruder's face [9]. The accelerometer would be attached to the door, and would be triggered upon movement of the door. The lights and alarm would be installed within the door and would be automatically triggered once the accelerometer identifies the opening of the door. With the use of the mobile application, the user can wirelessly connect to the base unit when connected to the same wifi network, and request for a picture from the camera [2]. This is especially useful when there are suspicions of an intruder trying to break in and would help scare off the threat and identify the criminal, all without even having to move from where you are. As a customer, this feature minimizes risk of having to deal with a violent criminal face-to-face, and gives them better information if they would like to call the authorities and report the incident. This feature would also come in handy if the user just wanted to check if the door was not properly closed. The lights and alarm work similarly, in that when the security system is engaged, if movement is detected by the motion sensor, the lights will turn on and the alarm will ring. We also plan to implement an "away mode" feature which would automatically turn all the lights off and turn on the motion sensor detection functionality when activated. This is especially useful for users who have left the house and do not recall if they turned off all the lights before leaving, as this feature would inherently turn off all the lights in the house. Once the system is engaged by the mobile application, the customer can be confident that their home is not only more secure, but also saving a lot of energy with reduced electricity consumption. We also plan on implementing a "home mode" feature, in which the motion sensor movement detection which we have to trigger the alarms and cameras will be deactivated, and turn the lights on. Below is a summarized list of features we plan to implement with our system:

- Devices
 - Camera
 - Picture
 - Can be taken manually through app
 - Substitution for live feed
 - Save a picture of unrecognized detected face
 - Low priority
 - Disable/enable motion sensor for triggering of camera
 - Lights
 - On/off manually

- On/off when motion sensor trips
 - Alarm
 - off manually
 - On when motion Sensor tripped
 - Disable/enable motion sensor
- Mobile App
 - Sign-In Functionality
 - Turn on/off light
 - Turn on/off alarm
 - Take picture
 - “Away Mode”
 - “Home Mode”
 - Profiles for each user
 - Call emergency services/primary contact
- Connection between base and phone
 - Arduino connected to Wifi Shield
 - Allow for HTTP request interaction between Arduino Base Unit and mobile app on the same wifi network

A key factor that will differentiate our system from the rest of the competition is the facial recognition functionality. The facial recognition functionality will allow for the home security system to automatically disarm the system once a recognized visitor approaches the door. This will help us to better identify visitors to the home, and also reduce false alarms by the user, making our system much more efficient. The mobile application will also be able to call emergency services or a saved emergency contact, if the user were ever to feel unsafe in their home so they can quickly get the assistance they need. This feature in combination with our facial recognition technology would put our system over the top.

Currently, the field for home security systems is disparate and small, with only a few companies that have developed solutions to home security. We have noticed that all of these solutions are proprietary and do not necessarily work with other systems. We also found that the prices of these bundles sets are unreasonably high, likely determining value based on the promise of security rather than the actual costs of research and development. Our product provides a different angle to home security, by allowing for the integration of various components such as cameras, motion sensors, lights, and alarms, and making it much more cost-effective. We also provide more advanced features than most of these systems currently on the market with our facial recognition feature, making our system one of the most innovative out there.

The market leader in home security is SimpliSafe's Home Security System (<https://simplisafe.com/build-my-system>), which runs close to \$300, including only a base unit that contains the alarm, a key fob, few entry sensors, and a motion sensor, with the ability to purchase a camera for an additional \$100. The functionality of this base unit is extremely limited; only alarming the user if the system is tripped by motion while engaged. The user would be able to disengage the system with the included key fob.

This system, the market leader right now, is neither modern nor convenient. Interaction with the system is strictly button based. Our solution would allow for disarming of the system via facial recognition, revolutionizing how we view modern day home security systems. The key fob is not convenient at all, as homeowners will likely not be carrying around the fob when they are in the house. With the development of mobile technology in the past decade, we believe it is crucial to have control of our home security system via mobile application. With our solution, the homeowner and any other member of the household can interact with the security system through the mobile application. With the omnipresence of mobile technology, and with our smartphones in today's era always being within reach, we believe this functionality will truly change the way consumers think about home security systems and make it much more user-friendly.

We will be using mass-produced, common hardware to solve the security issue. We will be able to keep costs down significantly, at approximately half the cost of SimpliSafe's camera bundle Home Security System. This makes our product appealing for all home owners and tenants, because of our ability to provide a feature as inaccessible as home security at a significantly reduced cost. Customers that previously decided to not invest in home security because of the initial costs will now be able to invest in our system and implement it within their homes, making their lives that much easier.

The absence of fear enables people to continue doing the great things that they love to do. We hope to make safety and security an afterthought for all, but still easily accessible when necessary. Our system accomplishes this mission, inspiring confidence and comfort to all.

Glossary of Terms

HCI: Human computer interaction (back-end or with user).

HUI: Handset user interface through the mobile app.

Arduino Base Unit (ABU): The Arduino Mega is a microcontroller electronic board. Its list of specs contains 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It possesses all the needed support for the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila [1]. This device is used to control the other hardware pieces of the security system such as the lights, alarm, camera, motion sensor and accelerometer. The arduino system consists of a Raspberry Pi unit which serves to connect the hardware components to the Firebase.

Security System: Refers to the entire unit as a whole which is composed of the Arduino, lights, alarm, camera, motion sensor, accelerometer, and the mobile app. Mobile interface/ User Interface(UI): A web-based interface converted for the user's Android smartphones, where they can view and execute operations through the application. These operations can include: device manipulation, system settings, personal information management, and system maintenance [4].

Properties of security: Refers to features such as confidentiality, authenticity, and/or integrity of transferring secure information.

Away Mode: Motion sensor is on, alarm is on. Lights are on/off depending on the customer's choice

Home Mode: Motion sensor is off, alarm is off. Lights are on/off depending on the customer's choice.

FireBase: Google's mobile platform that helps you quickly develop high-quality apps and store data.

Database: A collection of information that is organized so that it can easily be accessed, managed, and updated to suit the needs of the data. Said data can be classified based on the types of content: graphic, text, numeric, etc.

Facial Recognition: A method of identifying or verifying the identity of an individual using their face.

Unrecognized Visitor: An individual who does not have his/her face registered on the security system [10].

Recognized Visitor: An individual who has his/her face registered on the security system.

Accelerometer: A device used for measuring or maintaining orientation and angular velocity

Arducam: A camera shield available for Arduino devices that can be utilized to take pictures.

Motion Sensor: A device used to detect if there is motion in front of it.

User/Owner: An individual with access to the mobile app and is able to control the different security features provided by the security system.

User Profile: A personalized profile to interact with the security system for each person that has a registered profile on the security system.

User Privileges/Permission: An ability granted to a particular person who has access to the security system (i.e. Turning on lights and/or setting security modes)

Mobile App: An application that is run on a mobile device to interact with the security system.

Fingerprint Authentication: Using fingerprint to log into a user profile for the security system.

Saved Faces: Faces that have been registered and stored in the database.

System Requirements

Enumerated Functional Requirements

<u>Identifier:</u>	<u>Priority:</u>	<u>Requirement:</u>	<u>Acceptance Test Cases</u>
<u>REQ-1</u>	5	The system should have an “Away Mode” where alarm and motion sensor are enabled.	<ul style="list-style-type: none">• Test with user activating “Away-Mode” and alarm system and motion sensor subsequently being armed by default (pass).

<u>REQ-2</u>	5	The system should have a “Home-Mode” where alarm is disabled.	<ul style="list-style-type: none"> • Test with user activating “Home-Mode” and alarm system subsequently being disarmed by default (pass).
<u>REQ-3</u>	4	The user should be allowed to take a picture manually using the mobile app (phone must be in the same network as the ABU).	<ul style="list-style-type: none"> • Test with user taking a photo through the mobile app, capturing a real-time event, and subsequently being saved to the mobile application’s gallery (pass).
<u>REQ-4</u>	4	The Arduino base unit is connected to a motion sensor that is placed right before the doorstep to sense a motion. When motion is detected, it will signal the Arduino base unit to take a picture.	<ul style="list-style-type: none"> • Test with triggering motion sensor with human close to Arduino base unit to have picture taken (pass). • Test with triggering motion sensor with human far away (approximately 15 yards) from Arduino base unit to have picture taken (fail).
<u>REQ-5</u>	3	The facial recognition software will match a picture, taken with the Arduino Base Unit which occurs when the motion sensor is triggered, with pictures in the saved faces database on Firebase. If not matched, the picture will be uploaded to the Firebase and the mobile application’s photo gallery, with the appropriate time stamp.	<ul style="list-style-type: none"> • Test with triggering motion sensor by human who is in database and is a recognized tenant (pass) • Test with triggering motion sensor by human who is not in database and thus is not a recognized tenant (fail), resulting in their photo being uploaded to the Firebase and homeowner’s mobile application (pass)
<u>REQ-6</u>	1	The system shall allow the user to turn the lights on and off	<ul style="list-style-type: none"> • Test with user turning on the lights, and the Arduino base unit

		manually using the mobile app when on the same network as the ABU.	<p>powering the lights (pass)</p> <ul style="list-style-type: none"> • Test with user turning off the lights, and the Arduino base unit turning off the lights (pass)
<u>REQ-7</u>	3	The system should be able to turn on the lights after the accelerometer is triggered on the door, which identifies that the door has been opened.	<ul style="list-style-type: none"> • Test with triggering accelerometer at certain angle to react and notify Arduino Base Unit to turn on lights (pass)
<u>REQ-8</u>	5	When the system is in “away mode,” and the accelerometer is triggered, the alarm shall sound until the user manually turns off the alarm using the mobile app.	<ul style="list-style-type: none"> • Test with triggering accelerometer while Arduino Base Unit is in “away mode” to see if alarm turns on (pass)
<u>REQ-9</u>	5	When in away mode, the alarm will not sound when a registered face opens the door that triggers the accelerometer.	<ul style="list-style-type: none"> • Test with recognized person while Arduino Base Unit is in “away mode”, and trigger accelerometer to see if alarm stays silent (pass)
<u>REQ-10</u>	5	When in away mode, the alarm will sound if an unregistered face opens the door that triggers the accelerometer.	<ul style="list-style-type: none"> • Test with unrecognized person while Arduino Base Unit is in “away mode”, and trigger accelerometer to see if alarm sounds (pass)
<u>REQ-11</u>	2	The user should be able to turn off the alarm system through the mobile app.	<ul style="list-style-type: none"> • Test with alarm sounding off (begin with in ringing) when homeowner turns off alarm system in mobile app (pass).
<u>REQ-12</u>	2	The mobile app should be able to successfully authenticate users through fingerprint authentication	<ul style="list-style-type: none"> • Test with user logging in with correct username/password combination (pass).

		or passcode verification. If the wrong fingerprint or passcode is inputted, the app should not let the user in.	<ul style="list-style-type: none"> • Test with user logging in with incorrect username/password combination (fail). • Test with user logging in with recognized fingerprint iD (pass). • Test with user logging in with unrecognized fingerprint iD (fail).
<u>REQ-1</u> <u>3</u>	4	The mobile app shall give user control of the camera, lights, and alarm.	<ul style="list-style-type: none"> • Test with user turning lights, and alarm, off or on, as well as taking a photo (pass)
<u>REQ-1</u> <u>4</u>	4	Independent of “Home mode” or “Away mode”, the mobile app shall allow the user to separately enable or disable the camera and alarm functionality.	<ul style="list-style-type: none"> • Test with user disabling automatic photographing by camera and alarm system (pass)
<u>REQ-1</u> <u>5</u>	3	The mobile app should allow for multiple devices to control the system. (Addition/deletion of users).	<ul style="list-style-type: none"> • Test with users, other than the system admin, controlling and adjusting settings that they have permission for (pass) • Test if users, other than the system admin, can control and adjust settings that they don't have permission for (fail)
<u>REQ-1</u> <u>6</u>	2	The system should call 911 or an emergency contact in case of emergencies.	<ul style="list-style-type: none"> • Test with user pressing button and being connected on a call with an emergency contact (pass)
<u>REQ-1</u> <u>7</u>	3	The mobile app should allow for user to logout of their user account	<ul style="list-style-type: none"> • Test with user logging of their account on mobile application (pass)

Enumerated Nonfunctional Requirements

REQ 20, 21 22, 23, 27, 28 were removed. Numbers were kept the same for convenience.

<u>Identifier:</u>	<u>Priority:</u>	<u>Requirement:</u>	<u>Acceptance Test Cases</u>
<u>REQ-18</u>	2	When the user turns on/off light by pressing a button in the mobile app, the mobile app will send a signal to the server on the arduino base to toggle the lights.	<ul style="list-style-type: none">• Test with user clicking the on button for lights on mobile application, and system responding by turning on light system (pass)• Test with user clicking the off button for lights on mobile application, and system responding by turning off light system (pass)
<u>REQ-19</u>	3	When the user enables or disables the motion sensor and accelerometer by selecting either “Home Mode” or “Away Mode” in the mobile app, the app should send a signal to the arduino base to enable or disable the motion sensor.	<ul style="list-style-type: none">• Test with user turning on “Away Mode” through mobile app, then seeing if system responds accordingly by simulating movement in front of Arduino and seeing if it acts as indicated (pass)• Test with user turning on “Home Mode” through mobile app, then seeing if system responds accordingly by simulating movement in front of Arduino and seeing if it acts as indicated (pass)

<u>REQ-24</u>	4	When connected to wifi, the user should be able to access all of the camera and alarm features.	<ul style="list-style-type: none"> • Test with user keeping mobile app open without WiFi connectivity and seeing if pictures are prevented from being taken and alarm system are prevented from being disabled (fail)
<u>REQ-25</u>	4	System should maintain and be aware of its state (i.e. “away mode” or “home mode”) even when the app is restarted on any device.	<ul style="list-style-type: none"> • Test with user recording current set settings in mobile app, then restarting device, and seeing if settings are maintained (pass)
<u>REQ-26</u>	4	Product should be up to date and usable on the current Android OS platform.	<ul style="list-style-type: none"> • Test with user with Android device on latest Operating System version if they are able to download mobile app (pass)
<u>REQ-29</u>	4	When a person walks up to the door and the motion sensor is triggered, the arduino base should take a snapshot of the person and save it to Firebase. The server should compare the picture to those stored in the saved faces album.	<ul style="list-style-type: none"> • Test with recognized user triggering motion sensor and system identifying correctly (pass). • Test with unrecognized user triggering motion sensor and system identifying as unknown user (pass).

User Interface Requirements

<u>Identifier:</u>	<u>Priority:</u>	<u>Requirement:</u>	<u>Acceptance Test Cases</u>
<u>REQ-</u>	5	The User Interface should be	<ul style="list-style-type: none"> • Test with user being able to

<u>30</u>		created such that every page of the UI is intuitive to the user. For example if they want to view photos taken by the arducam, there is a “photos button” on the UI. Or if they want to put the system in “home mode” or “away mode”, there should be buttons for those.	navigate to every mobile app window, as well as making the customizable settings convenient for the user (pass).
<u>REQ-31</u>	4	The User Interface should include a login for users to sign into the application, which can be both typed credentials or biometrics (i.e. fingerprint) [11].	<ul style="list-style-type: none"> • Test with user opening mobile application and being met with a login page, which offers fingerprint authentication or empty username/password fields (pass)
<u>REQ-32</u>	3	The User Interface should provide a button to turn lights on and off.	<ul style="list-style-type: none"> • Test with user being able to locate and switch light settings (pass)
<u>REQ-33</u>	3	The User Interface should provide a button to arm and disarm the alarm.	<ul style="list-style-type: none"> • Test with user being able to locate and switch alarm settings (pass)
<u>REQ-34</u>	3	The User Interface should provide a button to take a picture using the home system’s camera.	<ul style="list-style-type: none"> • Test with user being able to locate and take photo through camera window (pass)
<u>REQ-35</u>	4	The User Interface should provide a page/button which allows the user to view an image the arducam has taken.	<ul style="list-style-type: none"> • Test with user being able to locate and access photo library, which stores all photos the Arduino-Base Unit has taken (pass)
<u>REQ-36</u>	2	The User Interface should provide a button to alert the	<ul style="list-style-type: none"> • Test with user being able to connect to a call with

		authorities in case of an emergency.	emergency services (9-1-1) (pass).
<u>REQ-37</u>	3	The User Interface should provide a button to quickly toggle modes for the home system (“home mode”/”away mode”).	<ul style="list-style-type: none"> • Test with user being able to locate and switch between “home mode” and “away mode” and settings being adjusted accordingly (pass).
<u>REQ-38</u>	4	The User Interface should provide a settings menu to check the permissions of users connected to the same device and to add other users.	<ul style="list-style-type: none"> • Test with system admin user being able to create/delete users and change their permissions (pass).
<u>REQ-39</u>	4	The User Interface should be navigated through easily with proper placements of back and other buttons.	<ul style="list-style-type: none"> • Test with user being able to locate “RETURN/BACK” buttons and traverse to all other windows in mobile app from any starting window (pass)
<u>REQ-40</u>	4	The User Interface should provide a settings menu to allow the addition of a new face to the recognized list of faces.	<ul style="list-style-type: none"> • Test with system admin user being able to add new faces to list of recognized faces (pass).

Functional Requirements Specification

Stakeholders

Homeowners, Tenants, Landlords, Store Owners, Building/Property Managers, Home Insurance Companies, Home Security Companies

Actors and Goals

Owner - Set up arduino base unit, create and delete users, set privileges for each user

User - they can actively put the arduino base unit in away mode and home mode. They can take a picture from the mobile app of the front door if they suspect someone is there. They can control all of the system features from the app (as long as they have the privilege for it).⁹

Recognized Visitor - This is a person whose face has been registered in the system and will be recognized by the facial recognition.

Unrecognized Visitor - This is a person whose face has not been recognized in the system but they are not attempting to break into the home. This person is only passing by the door (ex. Mailman).

Intruder - This is an individual whose face is not recognized by the system and they are attempting to break into the home or open the door.

Arduino Base Unit - This consists of a Raspberry Pi and Arduino unit connected together. This actor controls, sends data, and receives data from all other hardware components of the entire system.

Mobile App - This is the application that runs on a smartphone. Users of the system will utilize this app to interact with the entire system.

Arducam - Camera connected directly to Arduino base unit, allowing us to take images of those outside of the door

Motion Sensor - senses when there is movement outside of the door, and activates the camera to take a picture

Database - stores recognized faces, and user login credentials

Lights - lights within the home, activated when user enters

Alarm - alarm within home, activated when unrecognized visitor enters the home

Accelerometer - sensor that measures the amount of movement: this measurement will be used to determine when the door is opened

Use Cases

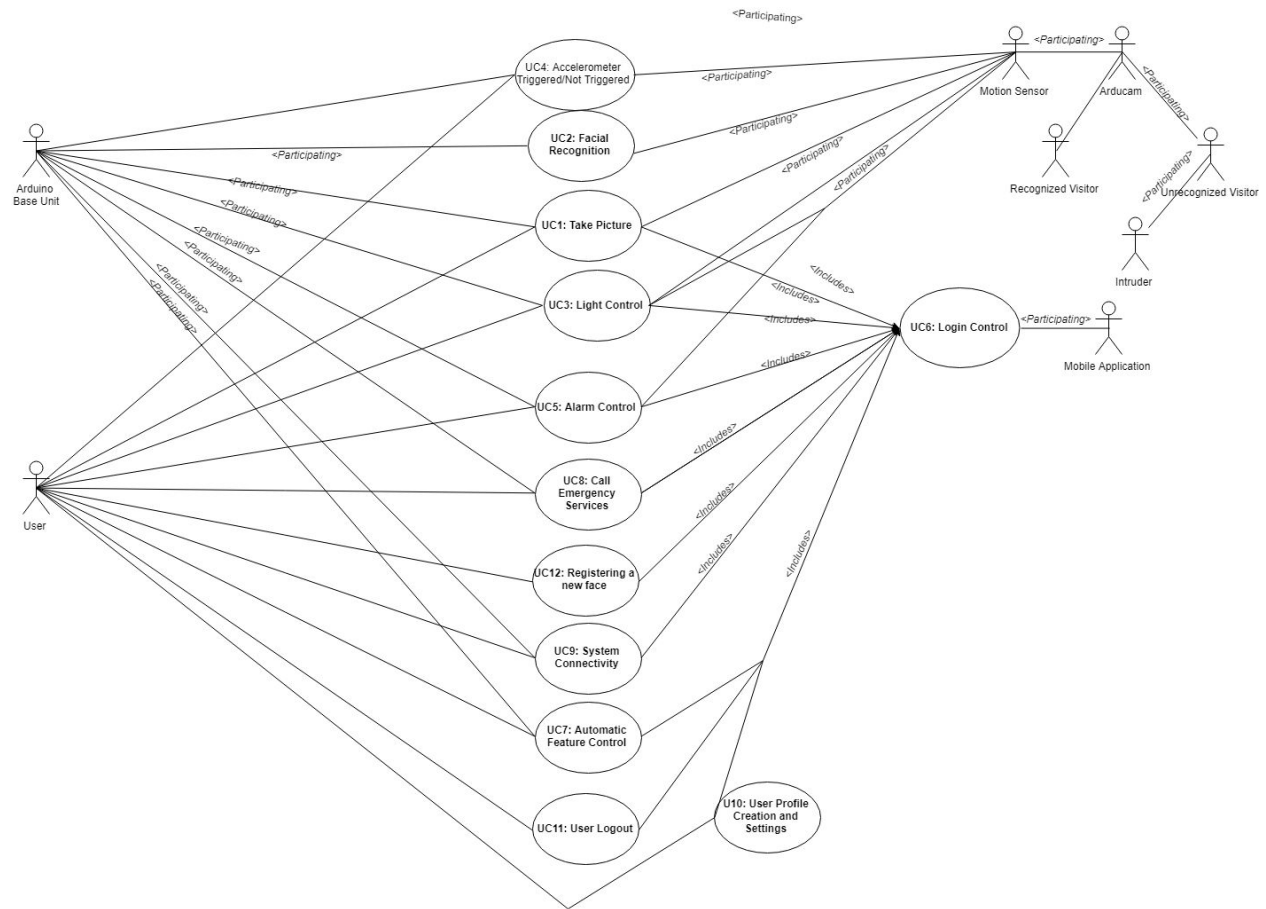
Casual Description

<u>Use Case Name</u>	<u>Actor</u>	<u>Actor's Goal</u>	<u>System Requirements</u>
UC-1: Take Picture	Owner, User, Arducam, Arduino Base Unit, Mobile App	To see what/who is outside the door	REQ-3, REQ-4, REQ-13, REQ-35, REQ-36
UC-2: Facial	Arduino Base	To determine if the	REQ-4, REQ-5,

Recognition	Unit, Arducam, Mobile App	visitor is an unknown person	REQ-8, REQ-30
UC-3: Light Control	User, Owner Lights, Arduino Base Unit, Mobile App	To turn lights on/off	REQ-6, REQ-7, REQ-13, REQ-19, REQ-31, REQ-33
UC-4: Accelerometer Triggered/Not Triggered	Arduino Base Unit, User, Owner, Alarm, Intruder, Arducam, Accelerometer, Mobile App	To turn on lights and turn on alarm once accelerometer sensor is triggered, if visitor is unknown/known	REQ-8, REQ-9, REQ-10, REQ-11
UC-5: Alarm Control	User, Owner, Alarm, Mobile App	To turn alarm on/off	REQ-11, REQ-13, REQ-35
UC-6: Login Control	User, Owner, Mobile App	To restrict access to user only (login authentication through fingerprint or email and password)	REQ-12, REQ-29, REQ-32
UC-7:Automatic Feature Control	User, Owner, Arduino Base Unit, Mobile App	To enable/disable automatic alarm, automatic camera, automatic lights functionality, as well as the sensors	REQ-1, REQ-2, REQ-14, REQ-15, REQ-38
UC-8: Call Emergency Services	User, Owner, Mobile App	To be able to call 911 if deemed necessary according to potential threats	REQ-17, REQ-37
UC-9: System Connectivity	Owner, User, Mobile App, Arduino Base Unit	To connect user's devices to the arduino base unit and connect the arduino base unit to the server via internet connection.	REQ-16, REQ-24
UC-10: User Profile	User, Owner,	To allow the owner to	REQ-29, REQ-39

Creation and Settings	Mobile App	create new user profiles and give different users different privileges such as which function the new user can control.	
UC-11: User Logout	User, Owner, Mobile App	To logout of current user account on mobile application	REQ-18
UC-12: Registering a New Face	Owner, User, Recognized Visitor, Unrecognized Visitor, Mobile App	To add another face to the list of recognized faces.	REQ-41

Use Case Diagram



Traceability Matrix

[illegible]

[illegible]

<u>37</u>	2								X				
<u>38</u>	3							X					
<u>39</u>	4										X		
<u>40</u>	4												
<u>41</u>	4												X
<u>Max PW</u>		4	5	5	5	4	4	5	2	5	5	3	4
<u>Total PW</u>		19	16	18	17	9	10	18	4	8	8	3	4

Fully-Dressed Description

Use Case UC-1: Take Picture		
<u>Related Req.:</u>	REQ-3, REQ-4, REQ-13, REQ-36, REQ-37	
<u>Init. Actors:</u>	User or Owner	
<u>Actor’s Goal:</u>	To see what/who is outside the door	
<u>Participating Actors:</u>	Arduino Base Unit (ABU), Mobile App, Arducam	
<u>Preconditions:</u>	None worth mentioning	
<u>Postconditions:</u>	The picture is sent from the arduino base unit to user/owner’s mobile application	
<u>Flow of Events for Main Success Scenario:</u>		
	→	1. The owner logs in to the mobile app and navigates to the camera section of the app.
	→	2. The owner taps the take picture button
	←	3. A signal is sent to the ABU for the Arducam to take a picture
	←	4. The ABU signals the Arducam and a picture is taken
	→	5. The ABU sends the picture back to the owners mobile app

Use Case UC-2: Facial Recognition		
<u>Related Req.:</u>	REQ-4, REQ-5, REQ-8, REQ-31	
<u>Init. Actors:</u>	Unrecognized Visitor	
<u>Actor’s Goal:</u>	To determine if the visitor is an unknown person	
<u>Participating Actors:</u>	Mobile app, unrecognized visitor, Arducam, recognized visitor, unrecognized visitor, Motion sensor, alarm	
<u>Preconditions:</u>	<ul style="list-style-type: none">• The arduino base unit (ABU) is in away mode, meaning the alarm is enabled	
<u>Postconditions:</u>	The owner is notified there is a recognized/unrecognized person outside the door	
<u>Flow of Events for Main Success Scenario:</u>		
	→ ← ← → →	<ol style="list-style-type: none">1. The motion sensor detects motion in the front door and sends a signal to the ABU2. The ABU sends a signal to the arducam to take a picture3. The arducam takes a picture and sends it to the ABU4. The ABU runs a facial recognition algorithm and checks whether there is a face in the picture that matches any saved faces in the Firebase5. If it is an unrecognized visitor, the ABU sends a signal to the user on their mobile application and rings alarm in the home
<u>Flow of Events for Alternate Success Scenario</u>		
	←	<ol style="list-style-type: none">1 - 4. Same as steps 1 to 4 of main success scenario5. The person at the door is a recognized visitor and the system does not do anything after.

Use Case UC-3: Light Control	
<u>Related Req.:</u>	REQ-6, REQ-7, REQ-13, REQ-19, REQ-32, REQ-34
<u>Init. Actors:</u>	User

<u>Actor's Goal:</u>	To turn lights on/off manually through the mobile app	
<u>Participating Actors:</u>	Arduino Base Unit (ABU), Light, Mobile App	
<u>Preconditions:</u>	<ul style="list-style-type: none">● The arduino base unit (ABU) is in away mode● The owner is in their bedroom as it is night time	
<u>Postconditions:</u>	The ABU was successfully able to turn on the lights	
<u>Flow of Events for Main Success Scenario:</u>		
	→	1. The owner is in their bedroom when they hear something at the door
	→	2. They log in to the mobile app however they see that no pictures have been updated to the app lately (No one is at the door)
	→	3. Yet still to for safety measures, the owner decides to turn on the lights to scare anyone away
	→	4. The user navigates through the mobile app by clicking the light tab
	→	5. They toggle the lights on with the light button
	←	6. A signal is sent through the google firebase to the ABU to turn the lights on.
	←	7. The lights are turned on.
	→	8. After a few minutes, as the owner gets assured they are safe, they toggle the lights to turn off from the mobile app.
	←	9. Another is signal sent to the ABU and the lights are turned off

Use Case UC-4: Accelerometer is triggered	
<u>Related Req.:</u>	REQ-8, REQ-10, REQ 11
<u>Init. Actors:</u>	Arduino Base Unit (ABU), User/Owner
<u>Actor's Goal:</u>	The ABU should make the alarm ring when needed and the user should be able to turn off the alarm if need be.
<u>Participating Actors:</u>	Arducam, Alarm, Motion Sensor, Recognized Visitor, Mobile App
<u>Preconditions:</u>	<ul style="list-style-type: none"> • The arduino base unit (ABU) is in away mode, meaning the alarm is

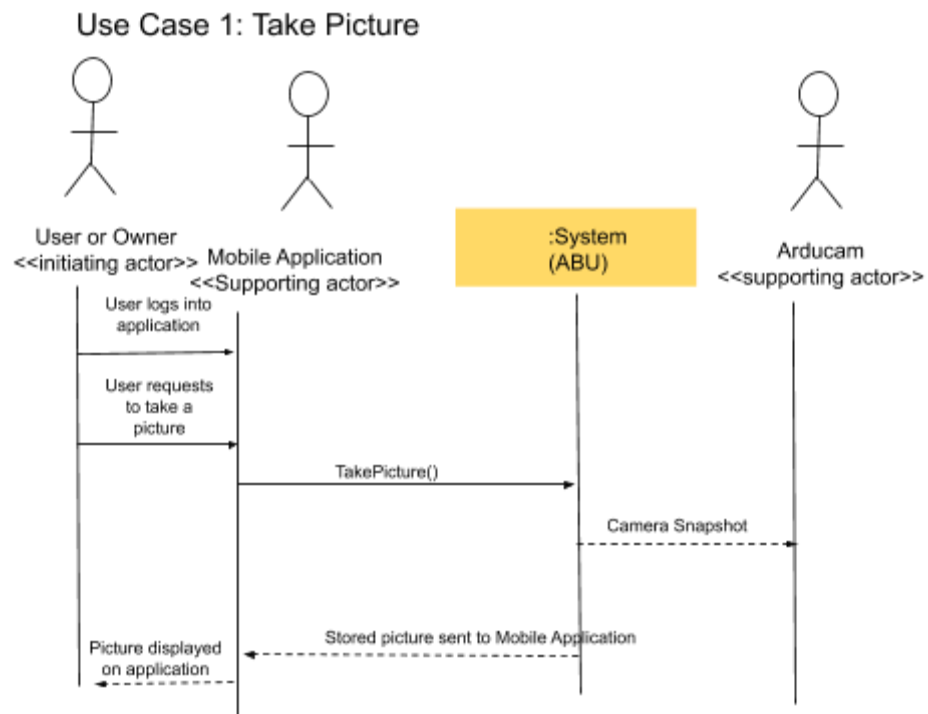
	enabled	
	<ul style="list-style-type: none">• No one is in the house• A recognized visitor is dropping something off for the owner and has the house key	
<u>Postconditions:</u>	The user was successfully able to turn off the alarm	
<u>Flow of Events for Main Success Scenario:</u>		
	→	1. A recognized visitor comes to drop something off at the owner's home
	→	2. They walk to the doorstep and the motion sensor detects the person, the arducam takes a picture and the ABU doesn't recognize the visitor as a known person
	→	3. The person opens the door with the key and the accelerometer is triggered
	←	4. A signal is sent to the alarm and it rings
	←	5. Meanwhile, the owner gets a notification from the ABU that there is someone at the door.
	→	6. The owner sees that it is their friend through the mobile app.
	→	7. The owner quickly navigates to the alarm tab of the mobile app and toggles the alarm system to off.

Use Case UC-7: System Control Features	
<u>Related Req.:</u>	REQ-1, REQ-2, REQ-14, REQ-15, REQ-21, REQ-39
<u>Init. Actors:</u>	User
<u>Actor's Goal:</u>	To put the system in home mode
<u>Participating Actors:</u>	Mobile App, Arduino Base Unit (ABU), Motion Sensor, Alarm
<u>Preconditions:</u>	<ul style="list-style-type: none"> • The arduino base unit (ABU) is in home mode, meaning the alarm is disabled • The user just entered the house from work but forgot to put the system in home mode from the ABU • An unregistered person (friend) is to follow in the house
<u>Postconditions:</u>	The recognized visitor is able to walk in and the alarm does not ring
<u>Flow of Events for Main Success Scenario:</u>	

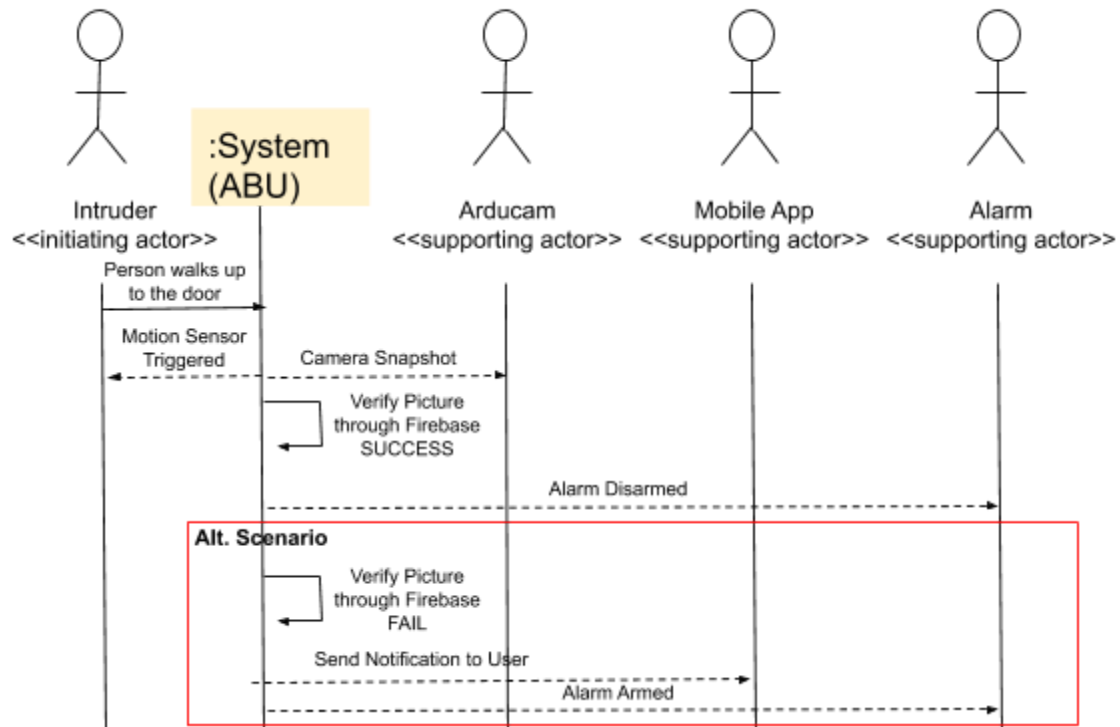
	→	1. The user gets into the house and closes the door
	→	2. He/She realizes that his friend, a recognized visitor, is still to follow and he hasn't put the system into the home mode yet
	→	3. He/She logs in to the mobile app and toggles the "home mode" switch
	←	4. A signal is sent to the ABU from thru the firebase for the system to turn into home mode
	←	5. The ABU disables the motion sensor and the alarm
	→	6. The friend enters the house and the alarm does not go off

Use Case UC-10: User Profile Creation and Settings		
<u>Related Req.:</u>	REQ-23, REQ-29, REQ-39	
<u>Init. Actors:</u>	Owner	
<u>Actor’s Goal:</u>	To add and give another family member control of system features.	
<u>Participating Actors:</u>	Mobile app, User, Owner, Database	
<u>Preconditions:</u>	● There are already a few devices connected to the system	
<u>Postconditions:</u>	A new family member was added to the group members who can control the system features on the mobile app	
<u>Flow of Events for Main Success Scenario:</u>		
	→	1. The owner wants to add a family member to the system
	→	2. He/She logs in to the mobile app and navigates to the settings page
	→	3. He/She taps on the create new user option
	→	4. He/She enters a new username and password for the family member
	→	5. The user gives all the permissions to the family member: can change lights, can change alarm, can take photo,, can change mode
	←	6. The mobile app sends this data to the Firebase and the new user is now officially added to the system.

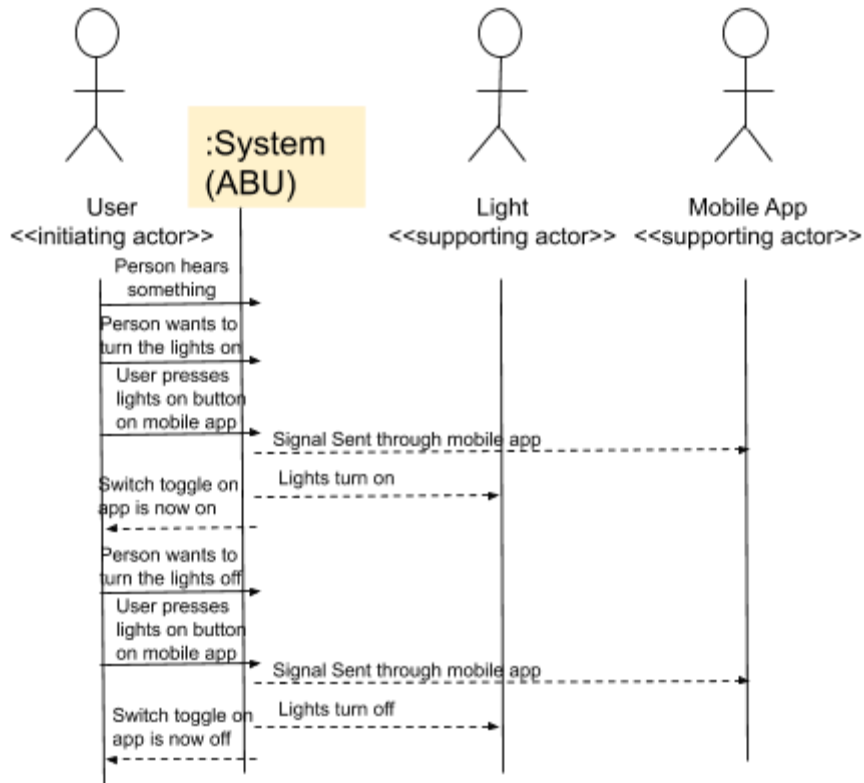
System Sequence Diagrams



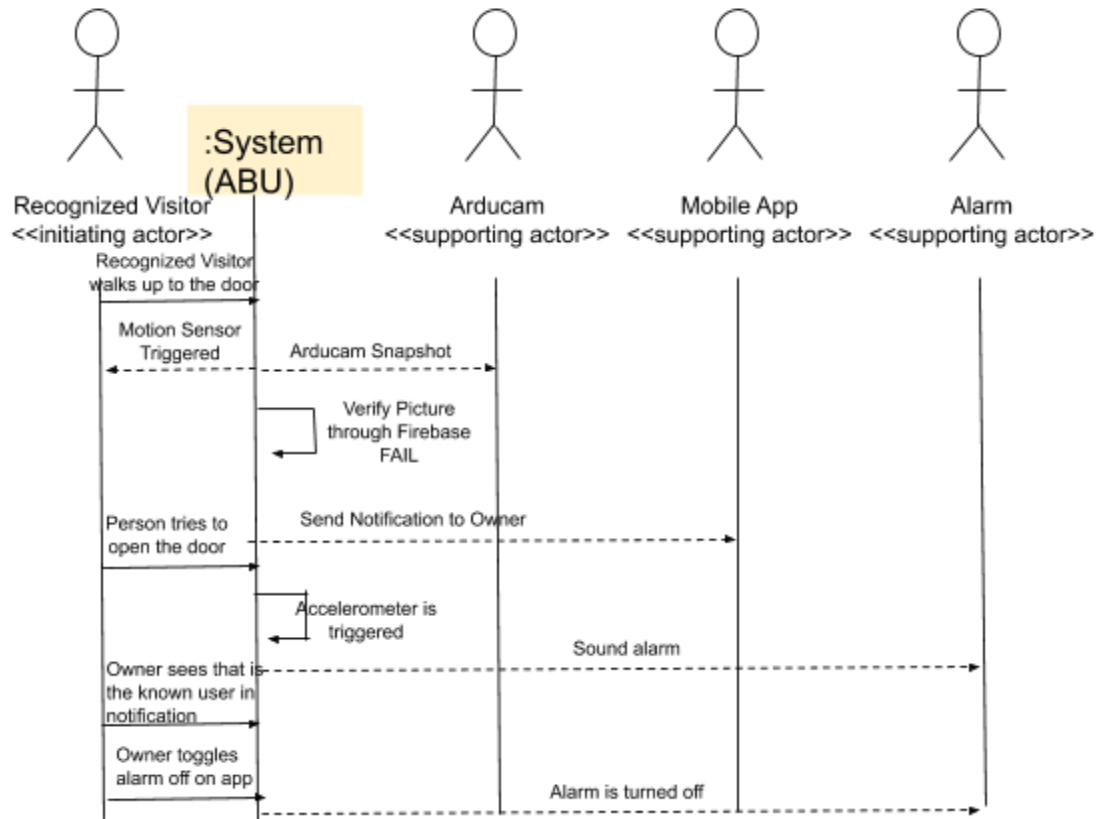
Use Case 2: Facial Recognition



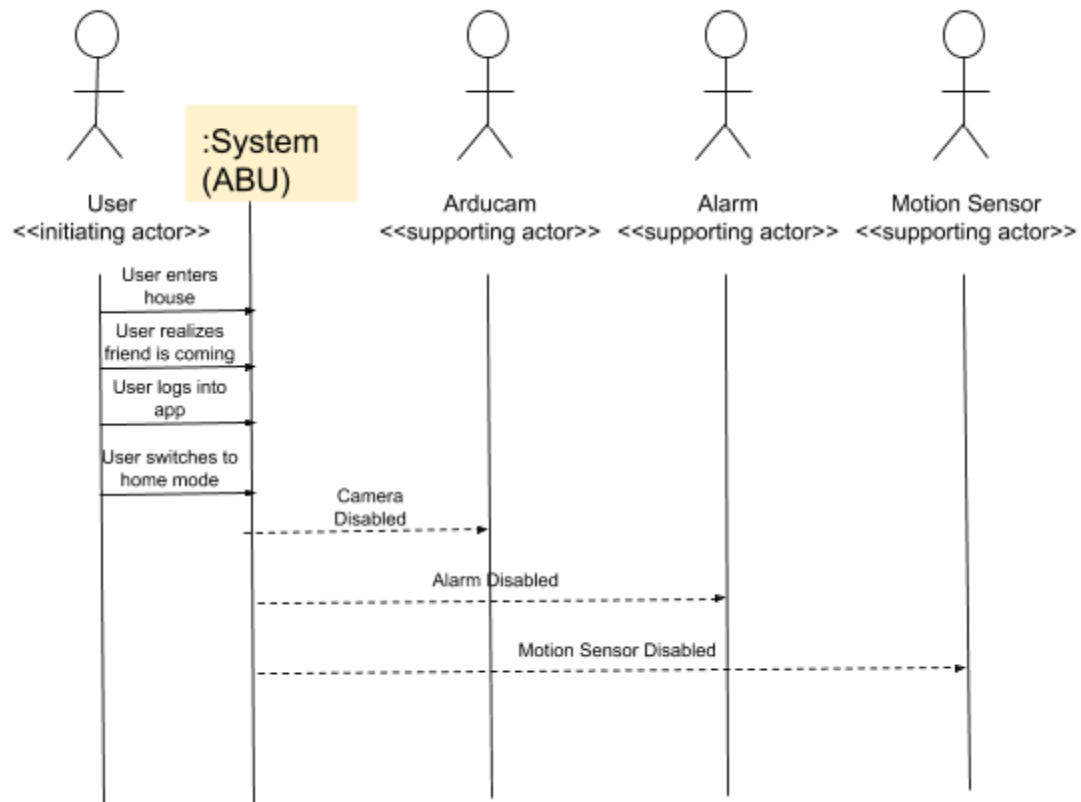
Use Case 3: Light Control



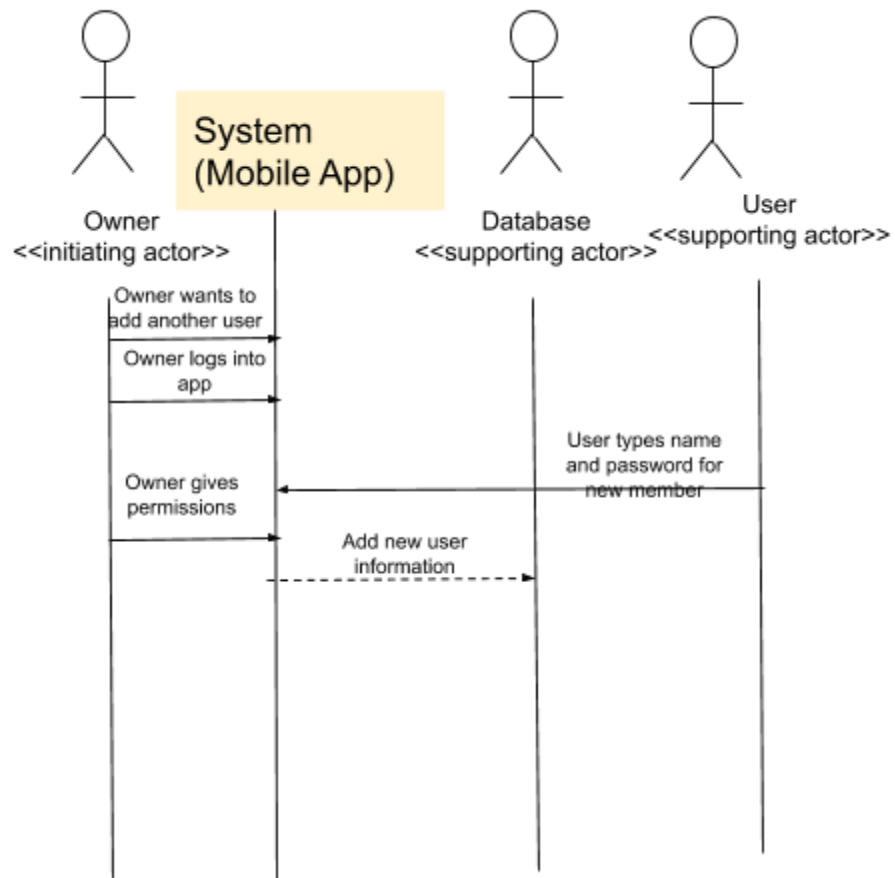
Use Case 4: Accelerometer is triggered



Use Case 7: System control features



Use Case 10: User Profile Creation and Settings



Effort Estimation With Use Case Points

	Category	weight
UC-1	Average	10
UC-2	Complex	15
UC-3	Average	10
UC-4	Complex	15

UC-5	Average	10
UC-6	Simple	5
UC-7	Average	10
UC-8	Simple	5
UC-9	Average	10
UC-10	Simple	5
UC-11	Simple	5
UC-12	Complex	15
	Total	115

UUCW - 115

	Description	weight	score	Factor
T1	Distributed system	2.0	3	6
T2	Response time/performance objectives	1.0	4	4
T3	End-user efficiency	1.0	4	4
T4	Internal processing complexity	1.0	2	2
T5	Code reusability	1.0	2	2
T6	Easy to install	0.5	4	2
T7	Easy to use	0.5	5	2.5
T8	Portability to other platforms	2.0	3	6

T9	System maintenance	1.0	2	2
T10	Concurrent/parallel processing	1.0	2	2
T11	Security features	1.0	5	5
T12	Access for third parties	1.0	2	2
T13	End user training	1.0	3	3
	Total			42.5

TCF - 1.025

Factor	Description	Weight	Impact	Factor
E1	Familiarity with development process used	1.5	2	3
E2	Application experience	0.5	3	1.5
E3	Object-oriented experience of team	1.0	4	4
E4	Lead analyst capability	0.5	2	1
E5	Motivation of the team	1.0	5	5
E6	Stability of requirements	2.0	5	2.5
E7	Part-time staff	-1.0	0	0
E8	Difficult programming language	-1.0	3	-3
	Total			14

ECF - .98

$$UCP = (UUCW + UAW) \times TCF \times ECF = (115+12)(1.025)(.98) = \mathbf{127.6}$$

Domain Analysis

UC-1: Take Picture

Concept Definitions

Concept Name	Type	Responsibility Description
Controller	D	Rs. 1: Coordinate actions of concepts associated with this use case and delegate the work to other concepts.
User Activated Picture	D	Rs. 2: Take in the requests of the user and relay the signal to the ArduCam to take a snapshot
Login Confirmation	D	Rs. 3: Takes login from user to allow them to access features in the app.
Signal to Server	K	Rs. 4: Take in request sent from mobile app interface to send a signal to the arduino base
Motion Activated Picture	D	Rs. 5: Takes picture when motion sensor is activated
Archiver	K	Rs. 6: Stores picture data that user requested, and stores the username and passwords of the system users (Firebase)

Association Definitions

Association Name	Concept Pair	Association Description
Conveys request	Mobile App Security Interface → Database Connection	Mobile app security Interface checks password against what is stored in database
Conveys request	Mobile App Picture Interface	When user requests to take a picture, the app will send a signal to server to execute request

	→ Server Connection	
Conveys request	Server Connection → Controller	Request from server is sent to the Arduino base controller
Execute	Controller → Arducam	Controller instructs arducam to take picture
Execute Request	Arducam → Server	Arducam sends signal to server when picture has been taken
Save Data	Server connection ← → database connection	Server signals save picture to database Database notifies server to update mobile app picture interface
Send Information	Database connection → Mobile app picture interface	Mobile picture interface is updated with picture from database

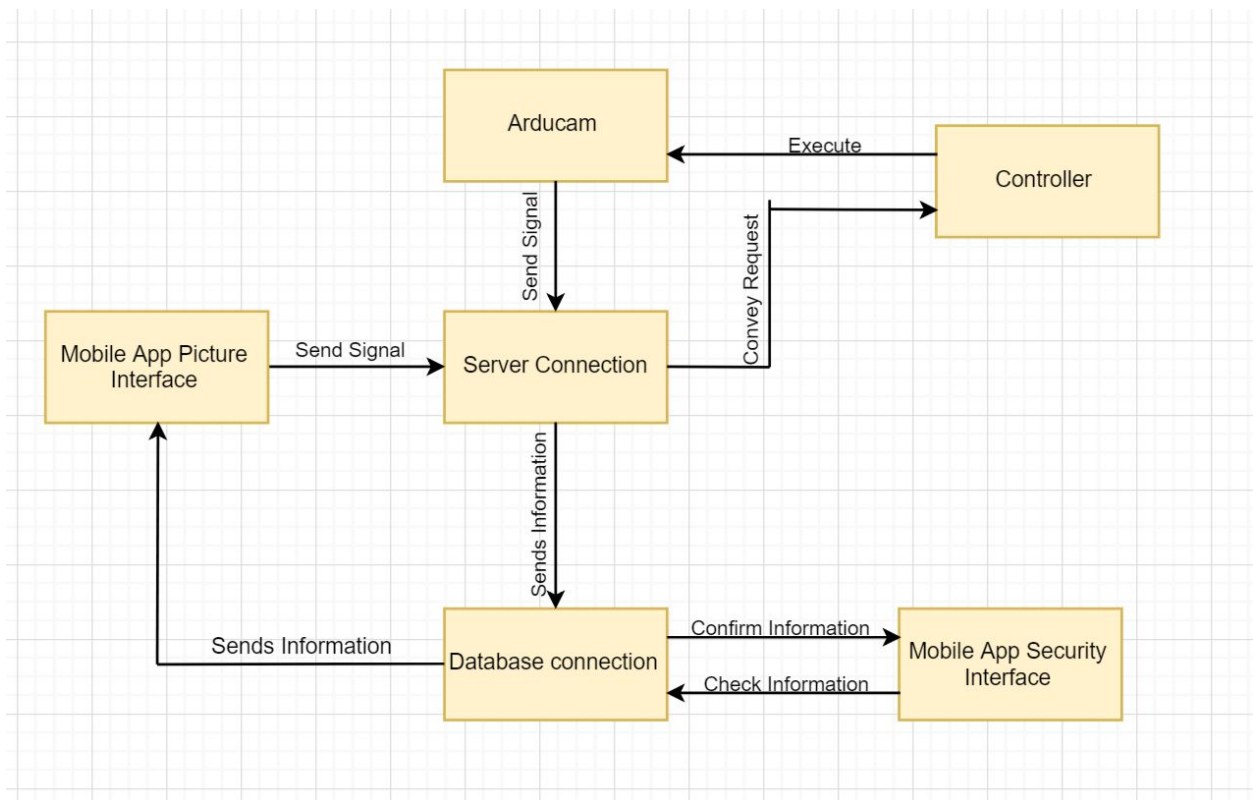
Attribute Definitions

Concept	Attributes	Attribute Description
Login Confirmation	User identity	Used to determine the user's credentials to authorize them to enter application
Signal to Server	Application Display	Execute request from the user and display necessary information on app (front-end display)
Archiver	Picture storage	Needs to store and update database for system update as a new picture is taken (read and write to Firebase)
Controller Request	Call for Action	Tells arduino basse to execute function (ie. take picture)
	Call for Storage	Calls to Database to store information
	Call for Search	Searches existing stored information to verify user request (ie. authorize if user is allowed to enter application)

System Operations Contract

Operation	Take Picture
Preconditions	<ul style="list-style-type: none">• One user is already registered into the system and has access to the mobile app.• The Arduino system has already been set up.• The Arducam is on and working
Postconditions	<ul style="list-style-type: none">• A new photo is added to the database• A new photo is displayed on user's mobile device

Domain Model Diagram



Domain Analysis UC-2: Facial Recognition

Concept Definitions

Concept Name	Type	Responsibility Description
Controller (Arduino Base)	D	Rs. 1 : Coordinate actions of concepts associated with this use case and delegate the work to other concepts.
Connect to Database	K	Rs. 2 : Reads and writes pictures that have been taken to Firebase
Motion Activated Picture	D	Rs. 3 : Takes a picture when triggered by the user or motion sensor
Trigger Motion Sensor	D	Rs. 4 : Determines when there is someone at the door so the arducam can take a picture
Mobile App Notifier	D	Rs. 5 : Sends a notification to a user's device to notify them about suspicious persons
Signal to Server	K	Rs. 6 : Take in requests when the ArduCam takes a picture of a suspicious person and sending a signal to the arduino base
User Display Picture	K	Rs. 7: Provides a way for the user to view a picture taken by the ArduCam

Association Definitions

Association Name	Concept Pair	Association Description
Conveys Request	Motion Sensor → Controller	Motion Sensor sends a signal to the controller when there is movement at the door.
Execute	Controller → ArduCam	Controller instructs ArduCam to take a picture
Save Data	Server Connection ↔ Database Connection	Server signals to save the picture to the database and perform facial recognition. The Database notifies the server to update the mobile app picture interface.
Send Information	Server Connection → Controller	Send and receive signals from the server to the controller and vice versa.
Send Information	Database Connection → Mobile App	Mobile app picture interface is updated with picture from database

	Picture Interface	
Send Information	Server Connection → Mobile App Notifier	The server updates the mobile app notifier to send a new notification to notify the user about the picture taken by the ArduCam.

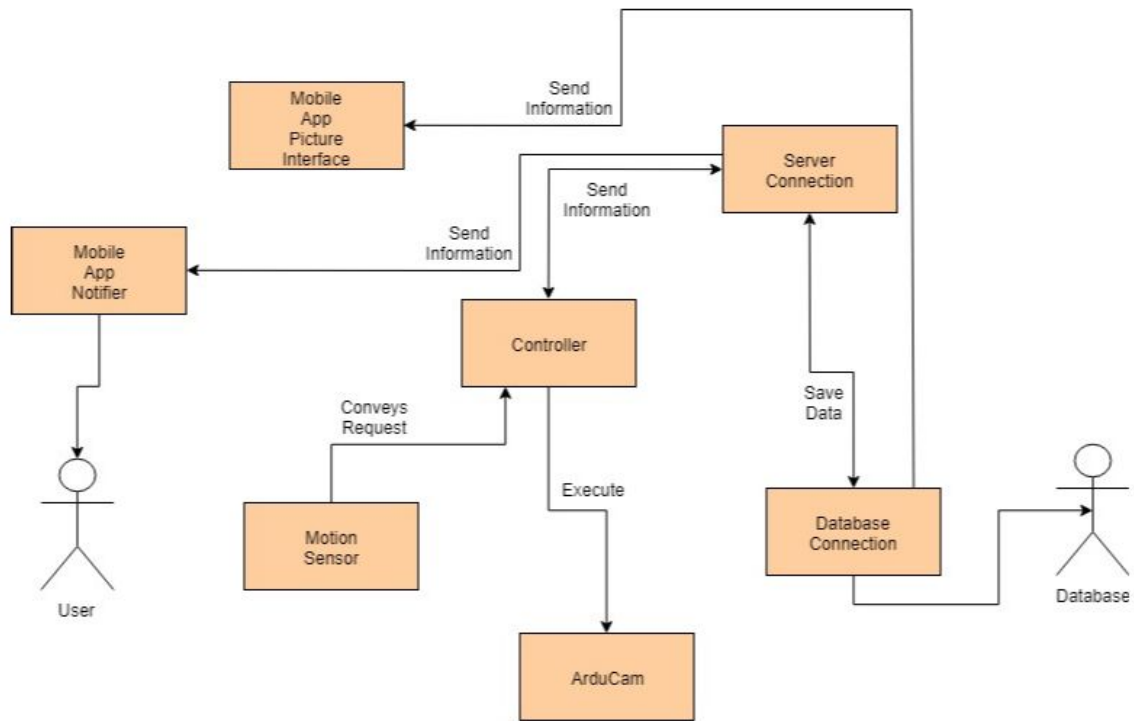
Attribute Definitions

Concept	Attributes	Attribute Description
Connect to Database	Access Database	Provides a means to access the database when required by the mobile app.
	Picture Transfer	Storing and retrieving pictures taken that are saved on Firebase
Controller Request	Call for Action	Tells arduino basse to execute function (ie. take picture)
	Call for Storage	Calls to Database to store information
	Call for server and database connect	Tells the controller to connect to the database and send/receive signals to the server

System Operation Contract:

Operation	Facial Recognition
Preconditions	<ul style="list-style-type: none"> One user is already registered into the system and has access to the mobile app. The Arduino system has already been set up. The Arduino system is armed and the ArduCam is active.
Postconditions	<ul style="list-style-type: none"> A new photo is added to the database A notification is sent to the user on their mobile app.

Domain Model Diagram



Domain Analysis UC-3: Light Control

Concept Definitions

Concept Name	Type	Responsibility Description
Controller (Arduino Base)	D	Rs1. Coordinate actions of concepts associated with this use case and delegate the work to other concepts.
DetectAngle	K	Rs2. Uses Gyroscope to calculate and store the angle measurement
LightsStatus	K	Rs3. Holds the current state of the lights in the house. 1 for being on, 0 for being off.
LightsChecker	D	Rs3. Checks that the LightsStatus is 0 when the controller is in away mode. Checks that the LightsStatus is equal to the UserLightsPref when controller is in home mode. Triggers the LightsStatus to 1 when receiving a signal from the Controller

TargetAngle	K	Rs4. Stores value that, when exceeded, says that the door is opened
UserLightsPref	K	Rs6. Stores the value of lights that the user's inputs.
LightsUI	D	Rs7. Shows the user a switch to turn the lights on or off and sends the current status of the switch to the UserLightsPref

Association Definitions

Association Name	Concept Pair	Association Description
Sends Angle Data	Controller + Gyroscope	Gyroscope passes angle data to the Controller.
Open Door	Controller + TargetAngle	Controller checks if Gyroscope angle is greater or equal to the TargetAngle
Control Lights	LightsChecker + LightsStatus	LightsChecker gets the LightsStatus and changes it according only
Trigger Lights	Controller + LightsChecker	Once the gyroscope angle exceeds the TargetAngle, the controller sends a signal to the LightsChecker to trigger the lights to 1.
User Preference	UserLightsPref + LightsUI	LightsUI sends the current status of what the user chose as their lights preference to the UserLightsPref. This value is stored.
Set Lights to User Pref	Controller + UserLightsPref	Controller sends a signal to the LightsChecker about the UserLightsPref so that it changes to what the user wants.

Attribute Definitions

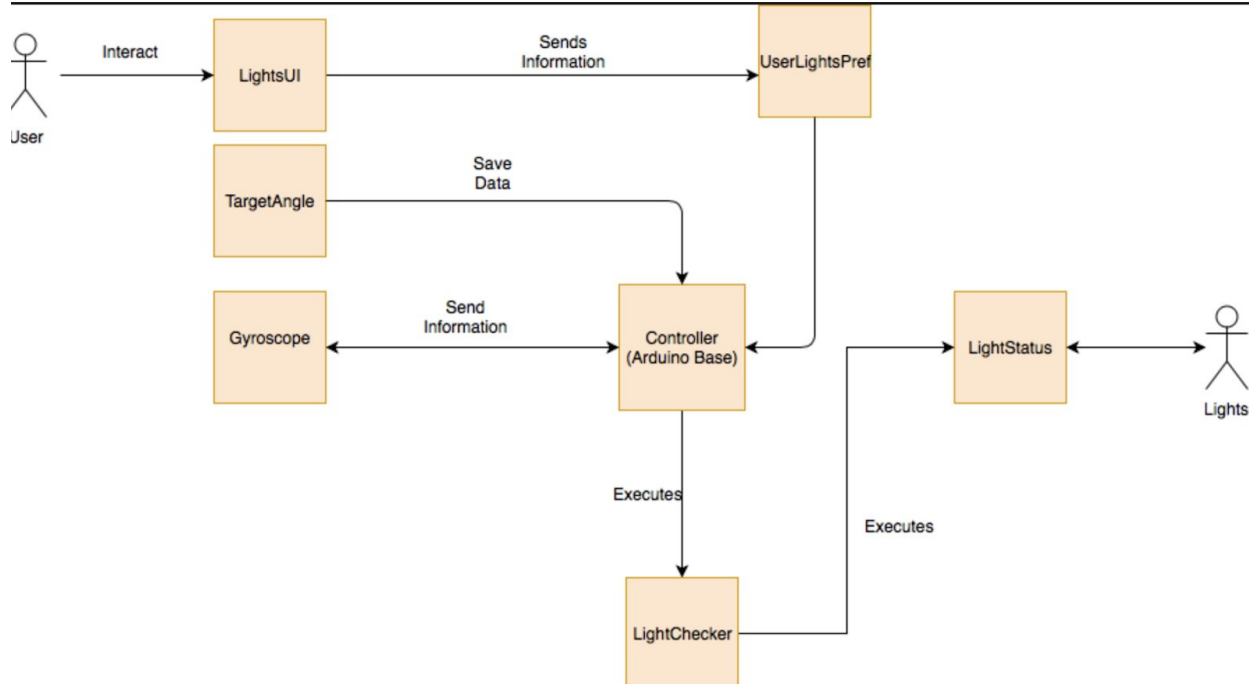
Concept	Attributes	Attribute Description
Target Angle	TargetAngle	Angle at which the door is considered open
	DoorAngle	Used to hold the current angle of the door.
Lights Status	Status	If Lights are on, the value is 1. If the Lights are off the value is 0.
Lights Checker	Status	Copied over from the Lights Status concept, holds the current value of the lights

	Trigger	Listens to Controller for a signal to change the value of the status is 1.
	UserLightsPref	Stores the value that the user puts for the lights.

System Operations Contract

Operation	Light Control
Preconditions	<ul style="list-style-type: none"> The Controller is set up. The Gyroscope is listening and transmitting data on the angle of the door. Lights work and respond to the Controller's commands.
Postconditions	<ul style="list-style-type: none"> Light is enabled/disabled manually by user via app. Arduino Base Unit enables light upon gyroscope being triggered.

Domain Model Diagram



Domain Analysis UC-4: Gyroscope is Triggered

Concept Definitions

Concept Name	Type	Responsibility Description
Controller	D	Rs1. Send signal to alarm and light to turn on
DetectAngle	D	Rs2. Read angle change when door is opened
TriggerAlarm	D	Rs3. Sound an alarm if unrecognized face is detected and door is gyroscope passes TargetAngle
TriggerLights	D	Rs4. Turn on light if unrecognized face is detected and door is gyroscope passes TargetAngle
TargetAngle	K	Rs5. Stored value that, when exceeded, says that the door is opened
FaceRecognized	K	Rs6. Store collection of recognized faces, and send signal to camera if unrecognized face is detected
MotionSensor	D	Rs7. Send signal to turn on camera if motionSensor detects motion
MotionPicture	D	Rs8. Take a picture if motion sensor is triggered.
MobileAppNotify	D	Rs9. User notified when unrecognized face opens door
DeactivateAlarm	D	Rs10. Send signal to controller to disable alarm

Association Definitions

Association Name	Concept Pair	Association Description
Sends Angle Data	Controller + Gyroscope	Gyroscope passes angle data to the Controller.
Alarm Activate	Controller + Alarm	Controller send signal to activate the Alarm noise.
Lights Activate	Controller + Lights	Controller send signal to activate the Lights on.

Gyroscope Trigger	Gyroscope + TargetAngle	TargetAngle checks if the Gyroscopes angle matches and activates the alarm system accordingly.
Face Recognized	Controller + FaceRecognized	Controller sends the picture the FaceRecognized for picture authentication.
User Notified	Controller + MobileApp	Controller sends a signal to the MobileApp to notify the user of alarm trigger.
Alarm Disabled	Controller + DeactivateAlarm	Controller sends DeactivateAlarmsignal to the alarm system to turn it off.
Alarm Disable Requested	MobileApp + DeactivateAlarm	MobileApp requests that DeactivateAlarm is activated.
Picture Capture Requested	MotionSensor + Camera	MotionSensor signals the Camera to take a picture.
Picture Authorization Requested	Controller + Camera	Camera sends picture data to Controller to be processed.

Attribute Definitions

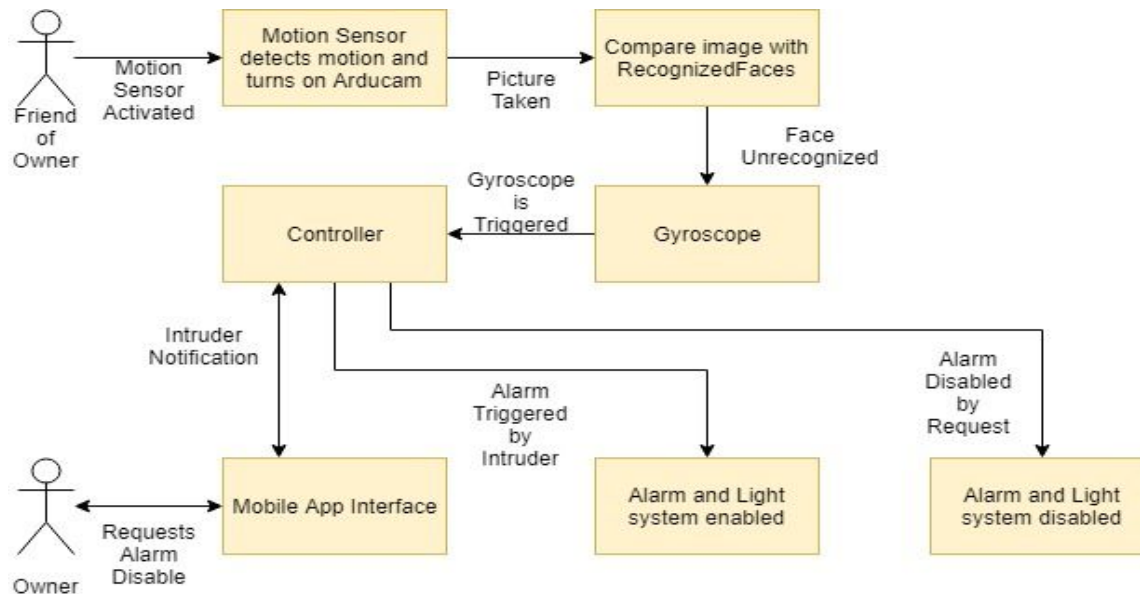
Concept	Attributes	Attribute Description
Controller Request	User's Identity	Used to determine if the user is a resident, which in turn allows a user to enter the house.
	Door Angle	Used to detect the possibility of unauthorized opening of the door.
MobileApp Notify	Mobile App Alert	Mobile app of the owner that gets notified of any alarm triggers.
Postprocessor	Recognized Picture	Recognized face for entry or denial of access.
	Trigger Angle	Marks when the door has been triggered as open.
	Disable	Ready to be disabled after alarm is activated.

System Operations Contract

Operation	Gyroscope
------------------	------------------

<u>Preconditions:</u>	<ul style="list-style-type: none"> • The arduino base unit (ABU) is in away mode, meaning the alarm is enabled • No one is in the house • A friend of the owner is dropping something off for the owner and has the house key
<u>Postconditions:</u>	The user was successfully able to turn off the alarm

Domain Model Diagram



Domain Analysis UC-7: System Control Features

Concept Definitions

Concept Name	Type	Responsibility Description
Controller	D	Rs1. Send signal to alarm and light to turn off
TriggerAlarm	D	Rs2. Turn off alarm once Controller sends the off signal
FaceRecognized	K	Rs4. Store collection of recognized faces, and send signal to camera if unrecognized face is detected
DetectAngle	D	Rs5. Send signal to turn on camera if motion sensor detects motion

MotionPicture	D	Rs6. Take a picture if unrecognized signal is detected
MobileAppHome	D	Rs7. User clicks the”home mode” switch
DeactivateAlarm	D	Rs8. Send signal to controller to disable alarm
CheckcCredential	K	Rs9. Checks users login credentials.

Association Definitions

Association Name	Concept Pair	Association Description
Alarm Deactivate	Controller + Alarm	Controller send signal to disable the Alarm.
Face Recognized	Controller + FaceRecognized	Controller sends the picture the FaceRecognized for picture authentication.
User Notified	Controller + MobileApp	Controller sends a signal to the MobileApp to notify the user of alarm trigger.
Alarm Disabled	Controller + AlarmDisable	Controller sends AlarmDisable signal to the alarm system to turn it off.
Alarm Disable Requested	MobileApp + AlarmDisable	MobileApp requests that AlarmDisable is activated.
Picture Capture Requested	MotionSensor + Camera	MotionSensor signals the Camera to take a picture.
Picture Authorization Requested	Controller + Camera	Camera sends picture data to Controller to be processed.

Attribute Definitions

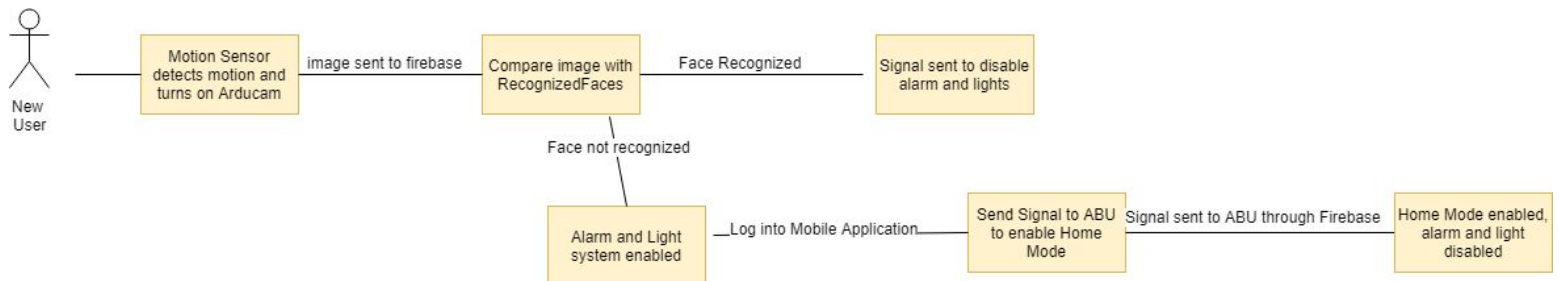
Concept	Attributes	Attribute Description
Search Request	User’s Identity	Used to determine if the user is a resident, which in turn allows a user to enter the house.
Notifier	Mobile App Alert	Mobile app of the owner that gets notified confirming the alarm disable.
Postproces	Disable	Ready to be disabled by the owner at any time.

SOR		
-----	--	--

System Operations Contract

Operation	Control Features
<u>Preconditions:</u>	<ul style="list-style-type: none"> The arduino base unit (ABU) is in home mode, meaning the alarm is disabled The user just entered the house from work but forgot to put the system in home mode from the ABU An unregistered person (friend) is to follow in the house
<u>Postconditions:</u>	The friend is able to walk in and the alarm does not ring

Domain Model Diagram



Domain Analysis UC-10: User Profile Creation and Settings

Concept Definitions

Concept Name	Type	Responsibility
Database	K	Rs 1. Stores the list of current users and each user's respective privileges.
UserAdder	D	Rs 2. Connect to and update database to add new Users and give new users privileges on the mobile app
Database Connection	D	Rs 3. Send request to database to retrieve all user information and current privileges.

Mobile App User List Interface	D	Rs 4. Provide an interface on the mobile app for a user to view all the other users with access to the system and view each user's privileges
FormProvider	D	Rs 5. Provide a form that asks for the new user's information such as name, picture etc.

Association Definitions

Association Name	Concept Pair	Association Description
Conveys requests	Database Connection + Mobile App User List Interface	Database Connection obtains the data from the database and pass the data to Mobile App User List Interface to displays the results to the user.
Provides new user form	FormProvider + Mobile App User List Interface	The Mobile App User List Interface provides the user the option to add a new user and the FormProvider gives the page to specify the user's credentials
Relays user data	FormProvider + User Adder	The inputted data from the user on the FormProvider is passed to the UserAdder which connects to the database and updates it with the new user.
Provides Information	Database Connection + Database	Database Connection access stored information the database and obtains the data to pass on to other concepts.
Sends user info	User Adder + Database Connection	UserAdder sends the new user's information to the Database Connection

Attribute Definitions

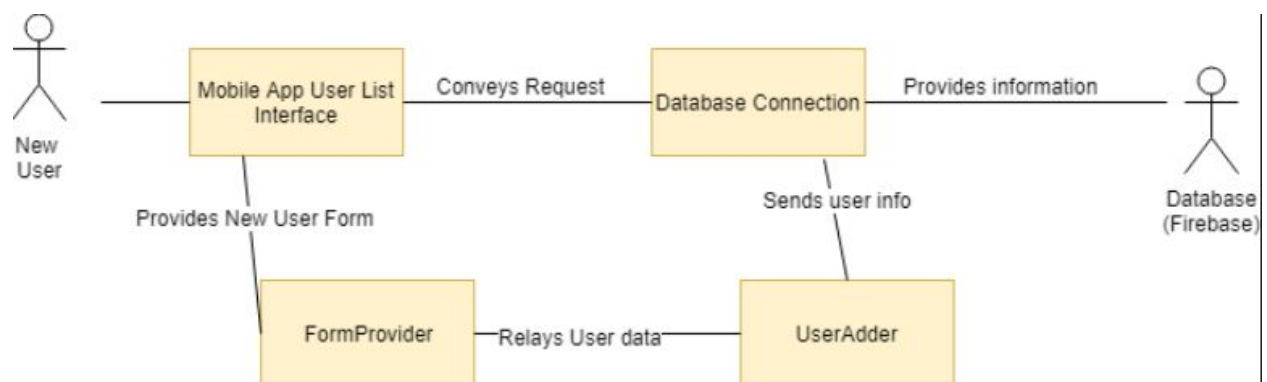
Concept	Attributes	Attribute Description
UserAdder	User's Information	Used to identify the person using the system
	User's privileges	Used to determine what features the user can access in the system.
Form	Textboxes	The user needs a place to write their credentials (name,

Provider		username, password, etc)
	Toggle buttons	Used to set the desired privileges for the new user (can control lights, alarm, camera, etc)
Database	User List	Contains the list of Users and each user's information
	Privileges	Contains data on each users different privileges
Mobile App User List Interface	User List	Shows all the recognized users of the system

System Operation Contract:

Operation	User Profile and Creation Settings
Preconditions	<ul style="list-style-type: none"> One user is already registered into the system and has access to the mobile app. The Arduino system has already been set up.
Postconditions	<ul style="list-style-type: none"> A new user was added to the system The database is updated and the new user's face is added to the list of recognized faces

Domain Model:



Traceability Matrix

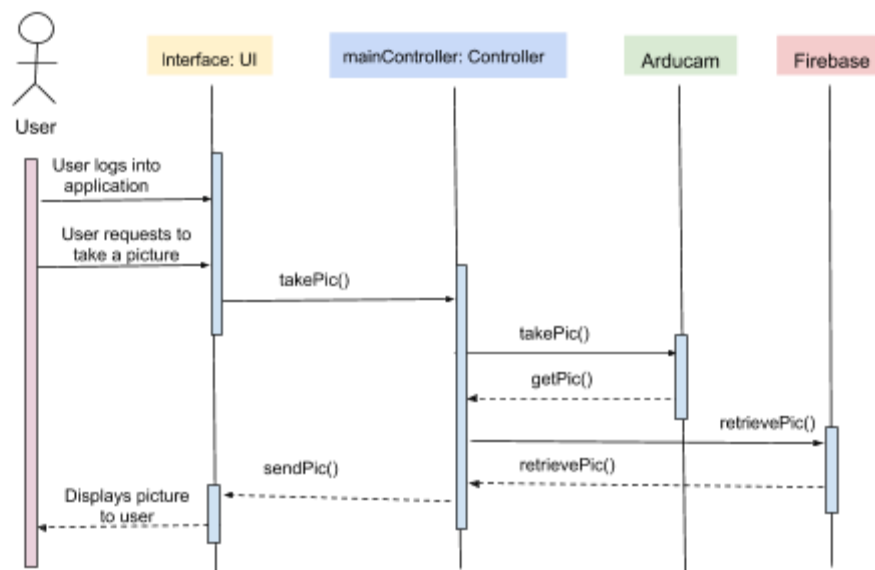
		<u>Use Cases</u>					
		UC-1	UC-2	UC-3	UC-4	UC-7	UC-10
	<u>PW</u>	19	16	18	17	21	11
<u>Domain Concepts</u>	Controller	x	x	x	x	x	
	Mobile App Pic Interface	x	x		x	x	
	Mobile App security Interface	x			x	x	
	Mobile App Notifier		x				
	Server connection	x	x	x	x	x	
	Arducam	x	x		x	x	
	Motion Sensor		x				
	Database Connection	x	x	x	x	x	x
	Database			x			x
	UserAdder						x
	Form Provider						x
	Mobile App User List Interface						x
	Gyroscope			x	x		
	Alarm				x	x	
	Lights			x	x	x	
	Target Angle			x			
	LightsChecker			x			
	LightStatus			x			
	User Lights Pref			x			
	Lights UI			x			

	RemoteAlarmDisable					X	
--	---------------------------	--	--	--	--	----------	--

Interaction Diagrams

UC-1

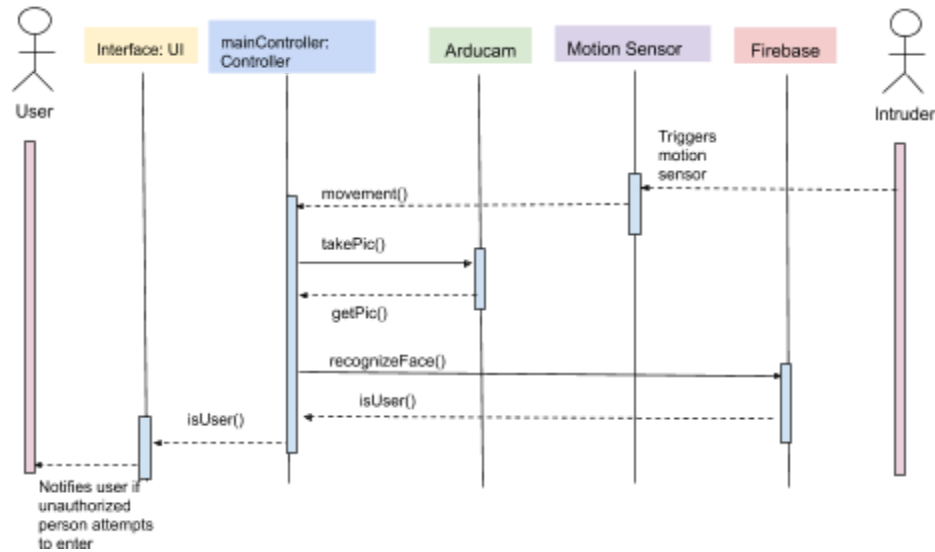
Use Case 1: Take Picture



This System Diagram will use the Single Responsibility Principle (SRP) as Firebase (our database connection) should only change if the user requests to take a picture. Further, we took into account the Liskov Substitution Principle (LSP) as the user interface for taking a picture (Arducam) should be able to execute taking and saving the picture. Lastly, we considered the expert doer principle as each part of our controller should know who it is communicating with in order to properly execute the request.

UC-2

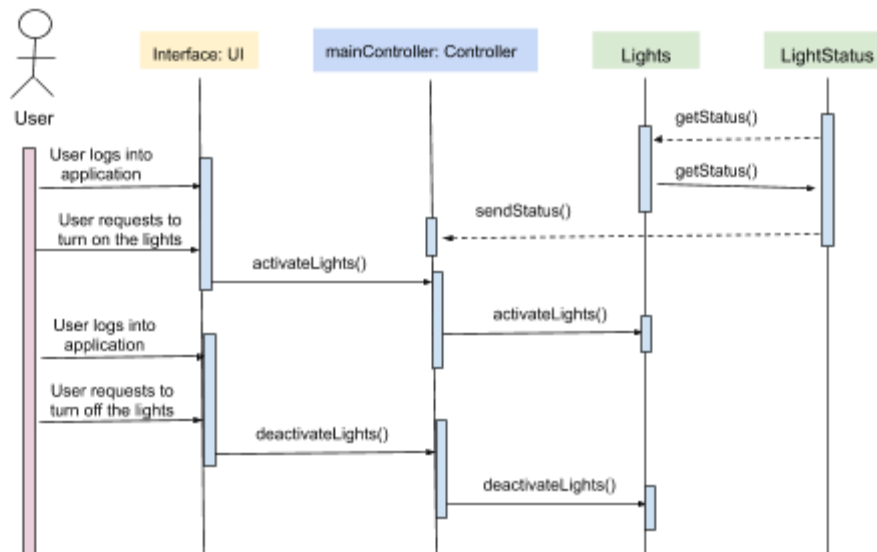
Use Case 2: Facial Recognition



For this system, we considered This System Diagram will use the Single Responsibility Principle (SRP) as Firebase (our database connection) should only change if the motion sensor is triggered to take a picture. We also considered the Low Coupling Principle as the controller should not take on too many responsibilities communicating as the sequence of actions between the Arducam, Motion Sensor and Database are highly dependant on the signals from the controller. Further, we took into account the Liskov Substitution Principle (LSP) as the user interface for taking a picture (Arducam) should be able to execute taking and saving the picture. Lastly, we considered the Expert Doer principle as each part of our system should know who it is communicating with in order to properly execute the desired tasks [17].

UC-3

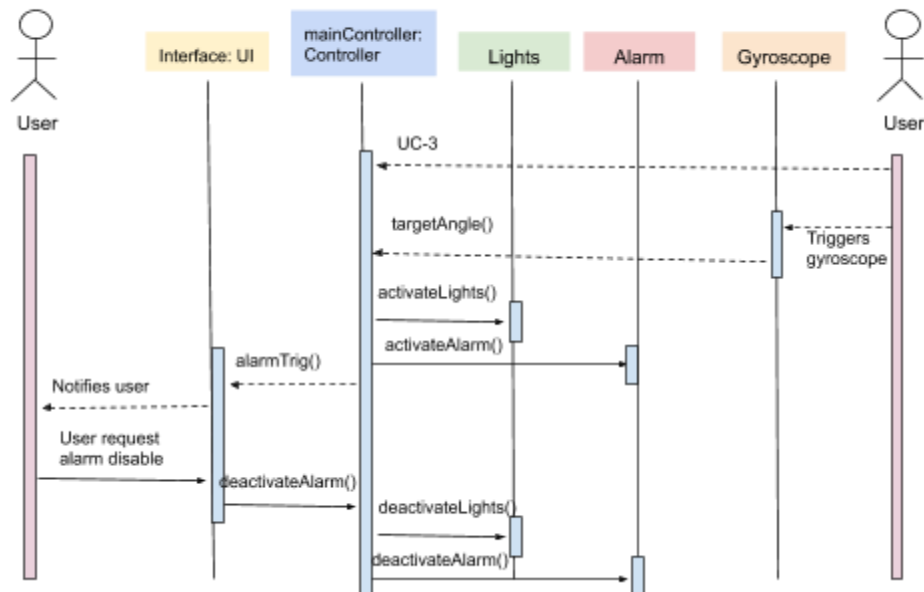
Use Case 3: Light Control



For this diagram, we considered the Low Coupling Principle as the controller should not take on too many responsibilities communicating since each of the following parts of the system is dependant on the controllers communication. Further we considered the LSP and Open Closed Principle (OCP) as each element of the system (ie. lights) be allowed to be enabled and disabled but the methods to activate should not be altered in the process. Lastly, we took into account the Interface Segregation Principle as the change of lights should only be dependant on the user request and controler interaction and nothing more.

UC-4

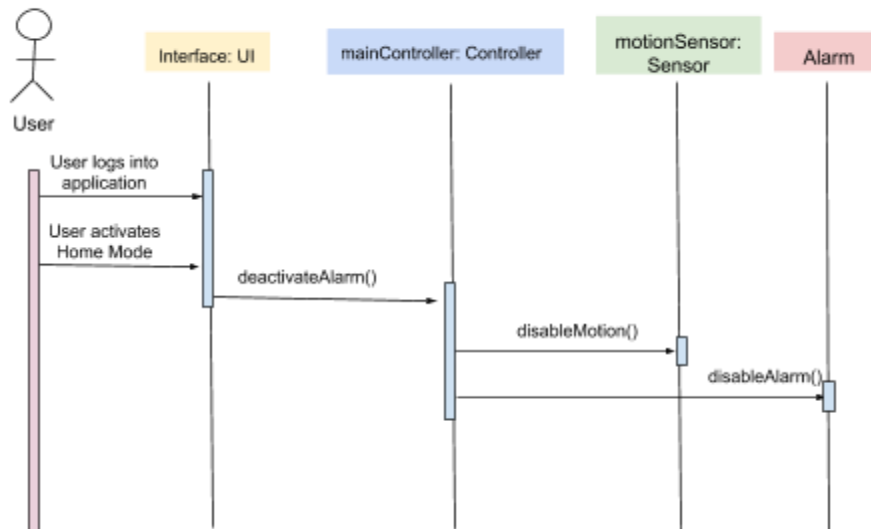
Use Case 4: Gyroscope is Triggered



For this diagram we took into account the Low Coupling Principle as the controller should not take on too many responsibilities communicating as the sequence of actions between the lights, alarm, and Motion Sensor are highly dependant on the signals from the controller. Further, we took into account the Interface Segregation Principle as the change of any part of the signal should only be dependant on the the activation from the gyroscope. Lastly, we took into account the Liskov Substitution Principle (LSP) as the user interface for disarming the alarm should be able to execute request properly.

UC-7

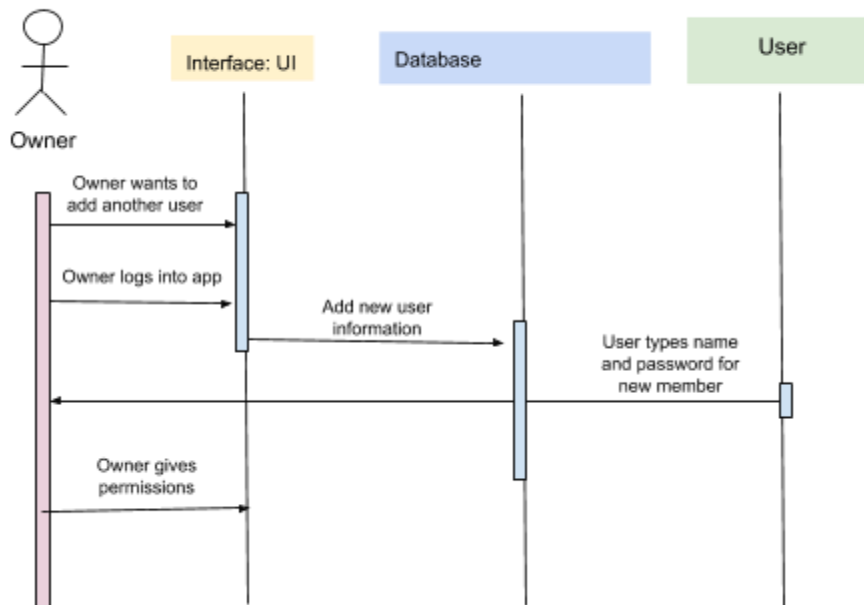
Use Case 7: System Control Features



For this diagram, we considered the Low Coupling Principle as the controller should not take on too many responsibilities communicating since each of the following parts of the system is dependant on the controllers communication. Further we considered the LSP and Open Closed Principle (OCP) as each element of the system (ie. motion sensor and alarm) should be allowed to be enabled and disabled but the methods to activate should not be altered in the process.

UC-10

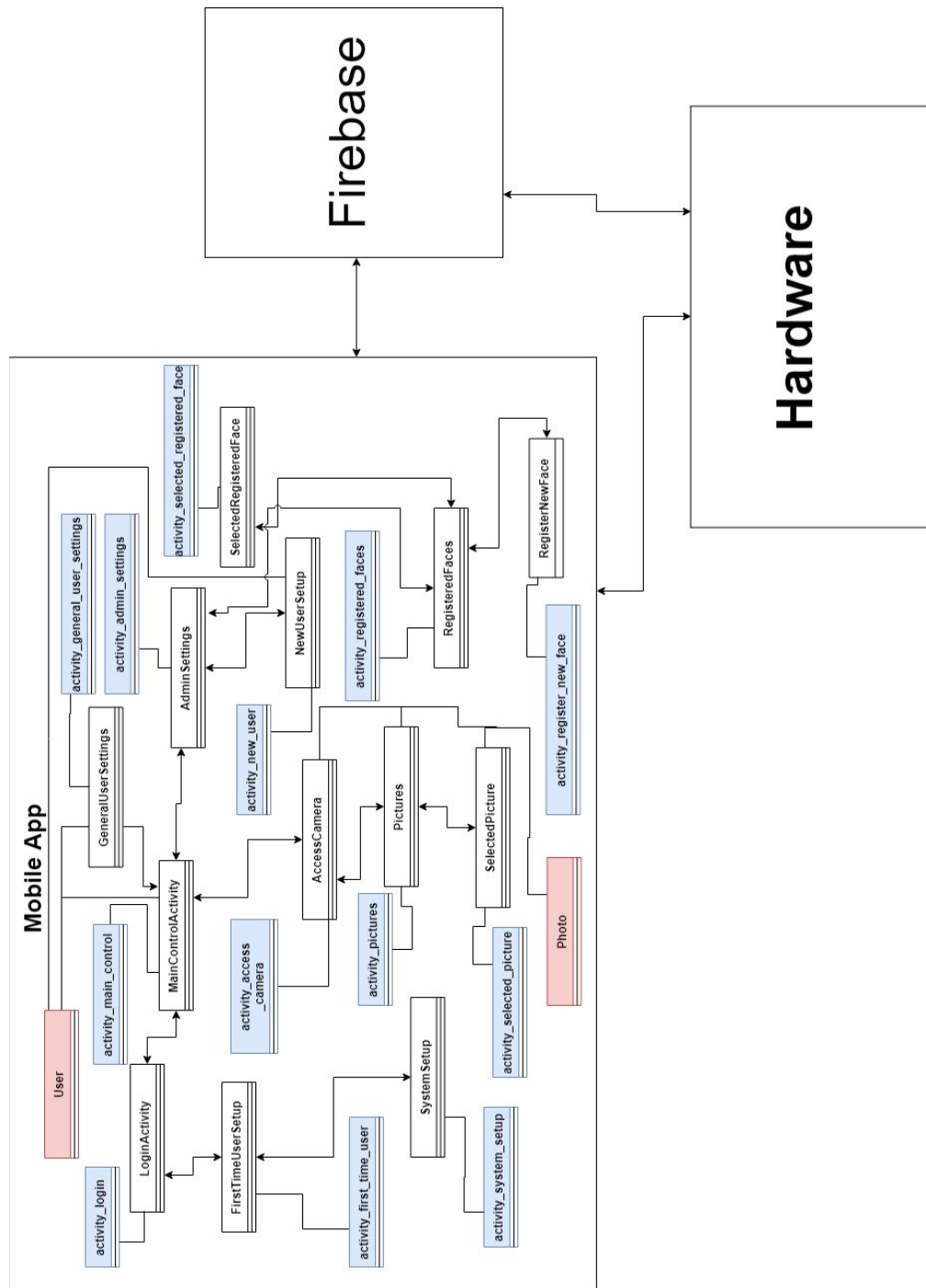
Use Case 10: User Profile Creating and Settings



For this system we took into account we considered the Low Coupling Principle as the controller should not take on too many responsibilities communicating since its main goal is to add the new user in the database. Moreover, we took into account the Interface Segregation Principle as the change to create the new user should not change any other element of the controller, base, or user app interface.

Class Diagram and Interface Specification

Class Diagram



Our class diagram has 3 main components, the mobile app, the hardware which incorporates the arduino, lights, camera, alarm and gyroscope, and Firebase. All three main components are connected to each other and are in constant communication. The main set of classes is contained in the mobile app. The mobile app is built using Android Studio and the code is written in Java. Our design requires that each new page in the app have its own class to control any events or actions on the page. Every double arrow line in the diagram describes a possible transition from one page to another. Each white background box represents a controller class in the UML diagram. These are the classes that each control one page in the app. The blue shaded boxes represent the UI (User Interface) code in the app. The UI code is done in xml. Each UI related file is shown connected to its controller class. The red shaded boxes represents objects we will create. The Photo object will allow us to easily utilize and display the photos taken from the Firebase and the User object will enable us to store the specific user's privileges to allow us to check whether or not a specific user action is allowed. These objects will also help us pass data back and forth between the classes without having to pull data from the firebase everytime. Our app will connect to the hardware through socket programming techniques. For each action we want to perform that involves the hardware the mobile app opens a socket to connect to the hardware, sends a signal, and then immediately closes the socket. This allows the hardware connection to remain open and listen for signals from different devices.

Data Types and Operation Signatures

LoginActivity

Login Activity
<ul style="list-style-type: none"> -LoginButton -SetupButton -EnableFPButton -editTextEmail -editTextPassword -TextViewSignin -firebaseAuth -fingerprintManager -keyguardManager -keyGenerator -KEY_NAME -cipher -cryptoObject +helper -userlist -newPass

-isPressed
-registerUser() -LoginUser() -onClick() #onCreate() #generateKey() +cipherInit() -FingerPrintScan() -LoginUser() +attemptLogin()

LoginButton:Button
 SetupButton:Button
 editTextEmail:EditText
 editTextPassword:EditText
 TextViewSignin: TextView
 firebaseAuth: FirebaseAuth
 fingerprintManager: FingerprintManager
 keyguardManager: KeyguardManager
 keyGenerator: keyGenerator
 KEY_NAME: String
 Cipher: Cipher
 cryptoObject: CryptoObject
 Helper: FingerprintHandler
 Userlist: List<User>
 newPass: String
 registerUser():void
 LoginUser():void
 onClick():void
 onCreate():void
 generateKey():void
 cipherInit():boolean
 FingerPrintScan():void
 LoginUser():void
 attemptLogin():void

In this page the User can login to the system with their email and password or for a first time user they can use the setup button to redirect them to a new page and begin the initial setup of the Home Security Automation system. The LoginButton and SetupButton are two buttons that allow the user to navigate to the main control activity page or the setup of pages to set up the initial system. The editTextEmail and editTextPassword objects are two input textboxes for the user to enter their login credentials into. The TextViewSignin is a text output message on the

screen. The firebaseAuth is part of the Firebase api that will allow the user to authenticate themselves to log in to the app. The registerUser() method will redirect the user to a new screen to register themselves and set up the system. The LoginUser() will check the user's credentials and log them into the app. The onClick() method will take in the user's button input and redirect to the proper method.

FingerPrintHandler

FingerPrintHandler
<ul style="list-style-type: none">- cancellationSignal- appContext- success
<ul style="list-style-type: none">+startAuth+onAuthenticationError+onAuthenticationHelp+onAuthenticationFailed+onAuthenticationSucceeded+isSuccess

cancellationSignal: CancellationSignal

appContext: Context

success: boolean

startAuth: void

onAuthenticationError: void

onAuthenticationHelp: void

onAuthenticationFailed: void

onAuthenticationSucceeded: void

isSuccess: boolean

This class is used to manage the Fingerprint authentication for the user. All of the onAuthentication methods are used to process the the fingerprint when scanned by the phone and output the appropriate response.

FirstTimeUserSetup

FirstTimeUserSetup

-createAccount -back -emailText -passwordText -firebaseAuth -database
+ registerUser() + onClick() # onCreate()

createButton: Button

back: Button

emailText: EditText

passwordText: EditText

firebaseAuth: FirebaseAuth

Database: DatabaseReference

registerUser(): void

onClick(): void

onCreate(): void

This page lets a user create the first account on their system by entering a email and a password which the firebase system will confirm and add to its authentication database. After successfully registering a user, the user will be redirected to a page to connect and setup up the arduino system. The createAccount is a button what will take the user's credentials and create a new user account on the firebase and for the app. The back button is used to take the user back to the previous screen. The emailText and passwordText objects are two input textboxes for the user to enter their login credentials into. The firebaseAuth is part of the Firebase api that will allow the user to authenticate themselves and register for the app. The registerUser() method is create the user's account on the firebase and then redirect them to the next page to setup the arduino system. The onClick() method will take in the user's button input and redirect to the proper method. The onCreate() method creates the buttons and onClicklisteners for page.

MainControlActivity

MainControlActivity

<ul style="list-style-type: none"> - lightON - alarmON - lightOFF - alarmOFF - alarmDIS - alarmARM - camera - call - homeButton - awayButton - offButton - settings + databaseReference + userP + action - LogoutButton

<ul style="list-style-type: none"> - onClick() - onCreate() - ExecuteAction() - dialContactPhone()
--

lightON: Button
 alarmON: Button
 lightOFF: Button
 alarmOFF: Button
 alarmDIS: Button
 alarmARM: Button
 Camera: Button
 Call: Button
 homeButton: Button
 awayButton: Button
 offButton: Button
 Settings: Button
 databaseReference: DatabaseReference
 userP: User
 Action: String
 LogoutButton: Button

onClick(): void
 onCreate(): void
 ExecuteAction(): void
 dialContactPhone(): void

This is the main class where the user can control most of the hardware features for the system. Each button corresponds to the specific action the user can do with the system. The user in this screen can control the lights, alarm, call for help, and navigate to other pages such as camera features and settings. The onClick() method takes care of dealing with which button has been pressed and performing the appropriate action.

AccessCamera

AccessCamera
<div><div>-takePic -call -back -mainPic -gallery +databaseReference +dataFace +faceRecStatus +faceListener +photoURI +FaceStatus</div><div>+onClick() +TakePicture() +LoadPic() +UpdateFaceRecStatus() - onCreate() +IntruderDetected()</div></div>

takePic: Button
call: Button
back: ImageButton
mainPic: ImageView
gallery: ImageButton
databaseReference: DatabaseReference
dataFace: DatabaseReference
faceRecStatus: TextView
faceListener: ValueEventListener
FaceStatus: String
photoURI: String

onClick(): void
 TakePicture(): void
 LoadPic(): void
 UpdateFaceRecStatus(): void
 onCreate(): void
 IntruderDetected():void

This class gives the user access to the camera and allows the user to take a picture of the current camera view and display the image on the app. The user can also navigate to a page to see older photos taken by the Arducam. The takePic attribute represents a button used to take a picture on the arduino came. The call is used by the user to quickly call 9-1-1. The back is a button object that will return the user to the previous screen. The firebaseDB and StorageRef attributes are a part of the firebase api to access the firebase storage and database systems. The image attribute is an ImageView object that is used to display the photo. The onClick() method will take in the user's button input and redirect to the proper method. The TakePicture() method responds to the TakePicture button and will take a picture on the arducam. The LoadPic() method captures the picture that was taken and loads it in the program. The UpdateFaceRec() method updates the database with the picture. The onCreate() method creates the buttons and onClicklisteners for page. After pressing the take picture button the screen will display the image and run the facial recognition algorithm and if the person is unrecognized the alarm will sound.

GridViewAdapter

GridViewAdpater
+ mContext +photoList
+ getCount + getItem + getItemId + getView

mContext: Context
 photoList: List<Photo>
 getCount: int

getItem: Object
getItemId: long
getView: View

This is an adapter class and is used to take an arraylist of Photo objects and load each picture into the View object so that they can be all displayed together in a gridview component. This allows the user to easily view multiple pictures at once and scroll through a selection of pictures. Classes such as the “Pictures” class and the “RegisteredFaces” class utilize this GridViewAdapter object to display photos on the app screen.

Pictures

Pictures
<ul style="list-style-type: none">- photos- back- databaseReference- gridview+adapter- mProgressCircle
<ul style="list-style-type: none">- onClick()# onCreate()

photos: List<Photo>
back: Button
databaseReference: DatabaseReference
gridview: GridView
adapter: GridViewAdapter

onClick(): void
onCreate(): void

This class displays the to user the set of pictures the arducam has taken over time. The pictures are loaded from the firebase storage reference and are stored into a List<photo> object. A user can select a specific image and have the image displayed in a full screen mode. This will take the user to another screen in the app. The photos object contains a list of photos that are obtained from the firebase. The back object is a button for the user to return to the previous screen. The

databaseReference object is used to obtain a firebase database instance to retrieve data and storage. The gridView object is used to display the images in a grid. The onClick() method will read in user input such as a button or photo press and perform the appropriate action.

UserSettings

UserSettings
<ul style="list-style-type: none">- table- userList- databaseReference- back
<ul style="list-style-type: none">+ onClick()# onCreate()+ initTable()+ xOrSpace()

table: TableLayout

back:Button

databaseReference: DatabaseReference

userList: List<User>

onClick(): void

onCreate(): void

initTable(): void

xOrSpace(): TextView

This page is used for the general user settings. The tableView object is used to display information about the user's privileges. The back button is used together to read user input and return to the previous screen. The onClick() method is used to listen for user input and respond with the appropriate method.

DeleteUser

DeleteUser

<ul style="list-style-type: none"> - back - delete + list + users - firebaseAuth - databaseReference - admin - index
<ul style="list-style-type: none"> - onClick() + deleteUser() # onCreate()

back: Button
 delete: Button
 list: ListView
 users: List<User>
 firebaseAuth: FirebaseAuthentication
 databaseReference: DatabaseReference
 Admin: User
 Index: int

onClick(): void
 deleteUser(): void
 onCreate(): void

This class is used by the admin to delete a user form the the home security system. The list object is ListView that displays the list of users to the admin. The admin can select a user from the list and press the delete button to remove a user. The onClick() method will take in any button input on the screen and perform the appropriate action. The deleteUser() method is called form the on click and proceeds to remove the user from the authentication system and the firebase database.

NewUserSetup

NewUserSetup
<ul style="list-style-type: none"> - emailText - passwordText - createAccount

- back - firebaseAuth - database - lightsToggle - alarmToggle - callToggle - modeToggle - pictureToggle
- onClick() + registerUser() # onCreate()

emailText: TextField
passwordText: TextField
createAccount: Button
back: Button
firebaseAuth: FirebaseAuth
database: FirebaseDatabase
lightsToggle: ToggleButton
alarmToggle: ToggleButton
callToggle: ToggleButton
modeToggle: ToggleButton
pictureToggle: ToggleButton

onClick(): void
registerUser(): void
onCreate(): void

This class serves as the backend for the first time user setup page. The emailText is the .xml component where the user enters their desired username. The passwordText is the .xml component where the user enters their desired password. The username and password variables will thus store the values passed, and this will be sent to the Firebase database. The firebaseDB and StorageRef attributes are a part of the firebase api to access the firebase storage and database systems. The onClick() method will take in the user's button input and redirect to the proper method. The Back() method will transition the user back to the previous page. The Toggle buttons allow the admin to predefine and set the new user's privileges.

AdminSettings

AdminSettings
-NewUserButton -FaceRecButton -back -deleteUser -EnableLight -EnableCamera -EnableAlarm -table + databaseReference +userList +adminSwitchList
+ onClick() #onCreate() + initTable() +xOrSpace()

NewUserButton: Button

FaceRecButton: Button

back: Button

deleteUser: Button

EnableLight: Switch

EnableCamera: Switch

EnableAlarm: Switch

table: TableLayout

databaseReference: DatabaseReference

userList: List<User>

adminSwitchList: List<Boolean>

onClick(): void

onCreate(): void

initTable(): void

xOrSpace(): TextView

This class is the backend for the Admin Settings page of the Mobile Application. This page will allow the Admin to navigate to other pages such as adding a new user, deleting a user, or facial recognition settings. The admin can also disable certain features of the app such as the lights, alarm or camera through the EnableLight, EnableAlarm, and EnableCamera switches. These switches would disable the features permanently for all users no matter their privileges until the admin chooses to enable them again. The initTable() method creates the table with the list of

users and their privileges for the admin to see. The onClick() method takes in any button or screen interaction and performs the appropriate action depending on which button or switch was pressed.

RegisteredFaces

RegisteredFaces
<div><div>-back</div><div>-addNewFace</div><div>+databaseReference</div><div>-gridview</div><div>+photos</div><div>+adapter</div><div>-mProgressCircle</div></div>
<div><div>- onClick()</div><div># onCreate()</div></div>

back: Button

addNewFace: Button

databaseReference: DatabaseReference

Gridview: GridView

Photos: List<Photo>

Adapter: GridViewAdapter

mProgressCircle: ProgressBar

onClick(): void

onCreate(): void

This class displays to user the set of registered faces on the system. The pictures are loaded from the firebase storage and are stored into a List<Photo> object using the databaseReference to determine where the pictures are located in the storage. The photos object contains a list of photos that are obtained from the firebase. The back object is a button for the user to return to the previous screen. The gridview object is used to display the images in a grid. The onClick() method will read in user input such as a button or photo press and call the appropriate method. The addNewFace button and redirect the user to a new screen to register a new face for the security system.

RegisterNewFace

RegisterNewFace
<ul style="list-style-type: none">-takePictureButton-back-addFace-rotate-retake-TAG+Storage_Path+Database_Path+storageReference+databaseReference-textureView#cameraDevice#cameraCaptureSessions#cameraRequestBuilder
<ul style="list-style-type: none">- onClick()#startBackgroundThread()#stopBackgroundThread()# takePicture()# createCameraPreview()#updatePreview()- openCamera()-openCameraBack()+getFileExtension()- UploadImageToFirebaseStorage()# onCreate()

takePictureButton: Button

back: Button

addFace: Button

Rotate: Button

Retake: Button

TAG: String

Storage_Path: String

Database_Path: String

storageReference: StorageReference

databaseReference: DatabaseReference

textureView: TextureView

cameraDevice: CameraDevice
cameraCaptureSessions: CameraCaptureSessions
cameraRequestBuilder: CameraRequestBuilder

onClick(): void
startBackgroundThread(): void
stopBackgroundThread(): void
takePicture(): void
createCameraPreview(): void
updatePreview(): void
openCamera(): void
openCameraBack(): void
getFileExtension(): String
UploadImageToFirebaseStorage(): void
onCreate(): void

This class is used for the user to register a new face on to the system It gives user access to the phone's camera on the device and directs the user to take multiple pictures of themselves to store onto the firebase and add to the facial recognition database. Many on the variables and methods all work together to give the user access to the camera and allow them to take pictures. The UploadImageToFirebaseStorage() method takes all the pictures once the user is complete and uploads them to the firebase. The onCreate() method sets up and initializes the contents of this page for the user and the onClick() method takes in button input from the user and performs the appropriate action.

Photo

Photo
- url - date
+ getURL() + getDate()

url: String
date: String

getURL(): String

getDate: String

Up the Photo object. The url attribute is a string data type that contain information to display the object and the date attribute holds the date of the photo. The two methods are getter methods that are used to retrieve the attribute data for a Photo object.

User

User
<ul style="list-style-type: none">- username- password- lightsPriv- alarmPriv- callPriv- cameraPriv- modePriv
<ul style="list-style-type: none">+ getUsername()+ getPassword()+ getLightsPriv()+ setLightsPriv()+ getAlarmPriv()+ setAlarmPriv()+ getCallPriv()+ setCallPriv()+ getCameraPriv()+ setCameraPriv()+ getModePriv()+ setModePriv()

username: String

password: String

lightsPriv: Boolean

alarmPriv: Boolean

callPriv: Boolean

cameraPriv: Boolean

modePriv: Boolean

getUsername(): String

getPassword(): String

```

getLightsPriv(): Boolean
setLightsPriv(): void
getAlarmPriv(): Boolean
setAlarmPriv(): void
getCallPriv(): Boolean
setCallPriv(): void
getCameraPriv(): Boolean
setCameraPriv(): void
getModePriv(): Boolean
setModePriv(): void

```

This is the User object class. It stores the user's username and password along with their privileges. There are several attributes, lightsPriv, alarmPriv, callPriv, cameraPriv, modePriv, each control a different user privilege. Each attribute has a getter and setter method associated with it to get the value of the attribute and to set the value of the attribute.

Uploads

Uploads
<ul style="list-style-type: none"> - back - Storage_Path - Database_Path - ChooseButton - UploadButton - ImageName - SelectImage - FilePathUri - storageReference - databaseReference - Image_Request_Code
<pre> # onCreate() + onClick() # onActivityResult() + GetFileExtension() + UploadImageToFireba seStorage() </pre>

back: Button
Storage_Path: String
Database_Path: String
ChooseButton: Button
UploadButton: Button
ImageName: EditText
SelectImage: ImageView
FilePathUri: Uri
storageReference: StorageReference
databaseReference: DatabaseReference
Image_Request_Code: int

onCreate(): void
onClick(): void
onActivityResult(): void
GetFileExtension(): String
UploadImageToFirebaseStorage(): void

This is the Uploads class. This class is used to upload pictures to the firebase database. The chooseButton is used to select an image which then is uploaded via Uri to the firebase.

Socket_AsyncTask

Socket_AsyncTask
+socket +txtAddress +wifiModuleIp +wifiModulePort +CMD
+getIPandPort() +setMessage() #doInBackground()

socket: Socket
txtAddress: String
wifiModuleIp: static String
wifiModulePort: static int
CMD: static String

getIPandPort(): void
 setMessage(): void
 doInBackground(): void

This class is responsible for communicating with the hardware for the entire home security system. It utilizes socket programming to connect to the IP address and port of the raspberry Pi and sends string messages based on the actions the user wants to perform. The socket object is the socket that will open and connect to the raspberry pi everytime the user wants to send a message to the hardware. The txtAddress, wifiModuleIp, and wifiModulePort variables are used when attempting to connect to the hardware. The CMD string will contain the message that is sent to the hardware.

Traceability Matrix

Domain Concepts (Columns) vs. Classes (Rows)

		Domain Concepts							
		Controll er (ABU)	Mobile App Pic Interface	Mobile App Security Interface	Server Connect ion	Ardu cam	Databa se Conne ction	Alarm	Lights
Classes	LoginActivity			X	X		X		
	FirstTimeUserS etup	X	X	X	X		X		
	FingerPrintHan dler			X	X				
	MainControlAct ivity	X			X	X		X	X
	AccessCamera	X	X		X	X			
	GridViewAdapt er		X						
	Pictures		X		X		X		
	DeleteUser				X		X		
	UserSettings	X			X		X	X	X

	AdminSettings	X			X	X	X	X	X
	NewUserSetup		X	X	X		X		
	RegisteredFaces		X		X		X		
	RegisterNewFaces		X	X					
	User			X	X		X		
	Photo		X				X		
	Uploads		X		X	X	X		
	Socket_AsyncTask	X			X	X		X	X

We took the liberty of combining certain domain concepts:

Lights = [Lights + LightsChecker + LightStatus + UserLightsPref + Lights UI]

Controller (ABU) = [Controller (ABU) + Gyroscope + TargetAngle]

Alarm = [Alarm + RemoteAlarmDisable]

Design Patterns

For our project we implemented several design patterns that mix together to control our entire system. For the app, each UI screen has its own backend class to control the features on that UI screen. We chose to build the app in this form so that each page is responsible for its own functions and features and if one page does not work then the other pages will not be affected. Each page has sole responsibility over a majority of its own functions and features. This also includes connecting to the database to update and retrieve data as each class that requires the functionality does not need to go through another system to access the firebase. Our overall system also implements a client server interaction between our mobile app and our hardware. Through socket programming we were able to implement our ABU as a server and each mobile app as a client. Whenever the mobile app (client) needs to send a command to the arduino such as take picture, turn on lights, etc, the app connects to the arduino + raspberry pi (server) and sends a string containing the action to perform. For our hardware systems our design pattern involves having the ABU sends signals/commands to each of the other hardware pieces to perform a specific action. All hardware related actions first pass through the ABU which then sends information to a specific piece of hardware such as camera, lights alarm, etc. to perform a specific action.

Object Constraint Language (OCL) Contracts

Our app code has several main invariants, preconditions and postconditions that are required for the successful operation of the system. For our initial LoginActivity class, when trying to press the login button to sign into the app the preconditions are that the username and password fields are filled in. The post conditions are if the login credentials are accurate the user will be able to login, otherwise the app will not allow the user login. In all of our classes after the login page, one invariant that much hold true is that the user is always signed in the the User objects in any classes that require them are never null. If a User object is null the app should not allow any access to the features and should log out the user. Another invariant that involves the Socket_AsyncTask class is that the attributes relating to the IP address and the port should be properly set to match the ABU. Otherwise any hardware related functionality will not be able to operate. Due to our addition of user privileges, many preconditions for all features include checking to see if the user has been granted the privilege of a certain functionality such as taking a picture or turning on the lights. If the privileges for the user are true then they will be able to interact with the app to perform a specific function. Another precondition that is required for a user to perform a specific hardware related function is that if the Admin of the system has enabled or disabled the feature. If disables, it will overwrite any user privileges and disable the function until re enabled.

System Architecture and System Design

Architectural Styles

The architecture styles of this project are the typical styles you would find for a project revolving around the Internet of Things (IoT) [18]. The project can be encompassed by four conceptual models: database, server-client, event-driven architecture, and layered architecture.

Database Architecture: Since our whole product revolves around the use of facial recognition software to automatically disarm the system if a recognized user opens the door and to remain armed otherwise, it makes most sense to utilize a database to store pictures of users and pictures of visitors so that the software would be able to compare and match pictures. For our product, we are using the Google Firebase as the database in which we can store these images [25]. Furthermore, the database will not only store pictures taken, but it will also contain information about the users of the app such as their privileges. If the user ever requests pictures taken by the ArduCam, the user will be able

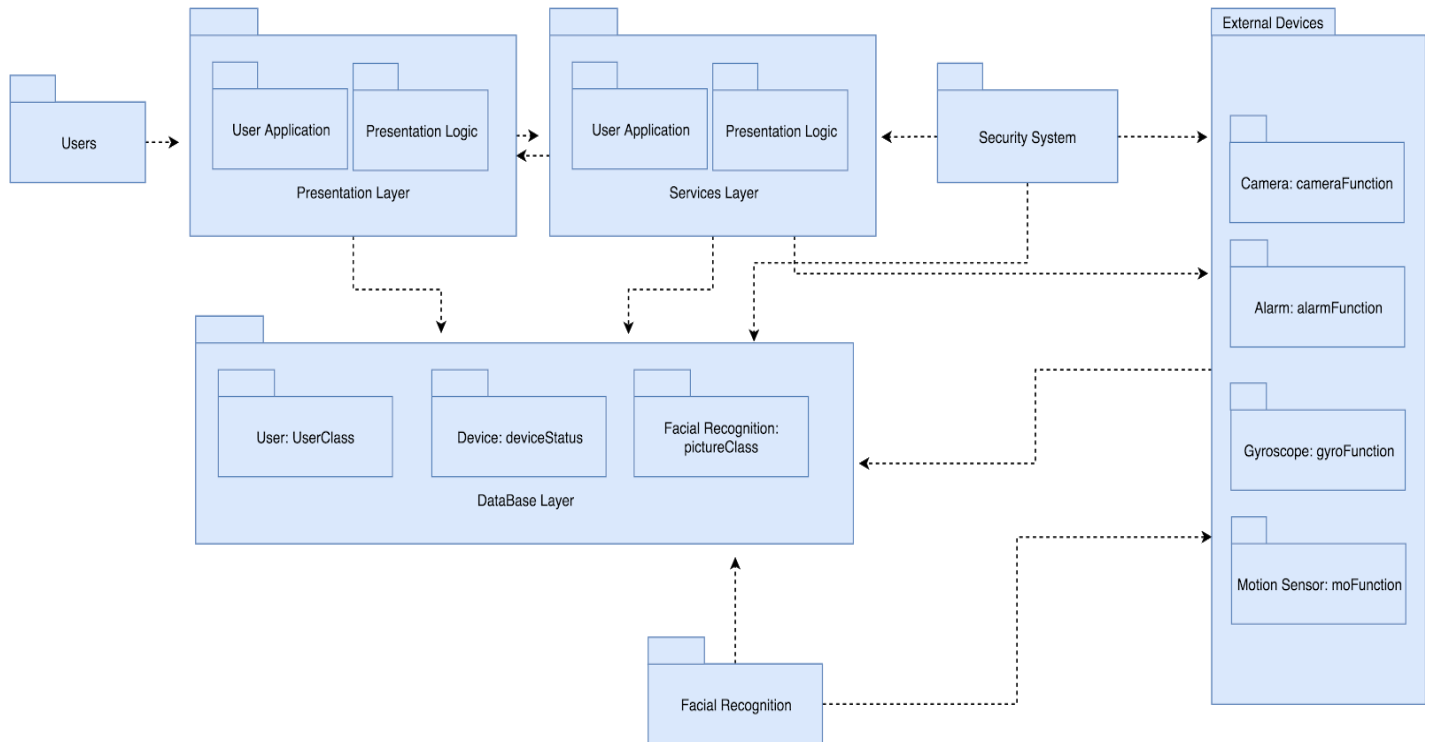
to get it from the Firebase [24]. Also any updates to the system, e.g. updating user privileges, needs to be stored somewhere to provide consistent behavior.

Server-Client Architecture: If we want multiple users to be accessing the product through different devices, some sort of controlled communication is important. In the modern world, the server-client architecture is the most widely used approach for such a problem. If multiple devices (clients) need to access to the Home Security Automation System, we need a central system (server) that can use multithreading to provide each client the resources/services they require [5]. We are implementing a Raspberry Pi unit as a server to communicate between the Arduino hardware components and the Firebase as well as with mobile devices that are running the mobile app [6]. The mobile app will have the ability to communicate with the Raspberry Pi Unit using HTTP request in order to manipulate settings locally for the Home Security Automation System.

Event-driven Architecture: The purpose of the Home Security Automation System is to provide the user with the ability to modify the internal devices (i.e lights, alarm, camera). In terms of software architecture, events are defined as “state changes” and so in order to drive the system on events we require communication between the user and the system. For example, if the user uses the mobile app to toggle the lights or alarms or to take a picture, this needs to be relayed to the actual hardware components for the event to occur.

Layered Architecture: We will be using a layered architecture because we need to make abstractions on how our product interacts with the user. In a layered architecture, each layer is separated in a way to create a hierarchy. Each layer provides the necessary services to the layer above it. For example, the most basic layer would be the technical layer which shows how the product is able to do work. This part includes all the mechanisms of the ABU (motion sensors, accelerometer, etc), the actual storing of data and the client and server. Next, we would have the domain layer in which the server would retrieve information from the technical layer for the user. This layer would also include general functions such as user authentication which allows a user to get to this layer. Last, we have the user-interface layer. In this layer, the app itself can be found. This is the layer in which the user can mostly directly interact with the system. You can see how the bottom layer (technical) does not need to know anything about any of the above layers, and this pattern follows on as you keep moving upward through the layers until you reach the top layer and the user itself. By dividing up our understanding of the system into layers, we have the ability to subdivide the entire system.

Identifying Subsystems



We were able to divide our system into several different subsystems to better represent the relationships of the layers in our project. There are two main layers that we divided our system into, the presentation layer and the services layer. The presentation layer consists of the view that the user will see. This includes the user application and basic presentation logic as this is the only part of the system that our user will have control over. The services layer is defined as the part of the user application that will be manipulated by external devices. The execution of requests on the external devices will influence the logic on the application and vice versa.

All of these layers need access to the database package of the system. The presentation layer and the service layer use the database package as a way to update their application and the external devices use the update the database package with every new request completed. We have a facial recognition package which is standalone, as the logic for the facial recognition is different than the logic used for the external devices. We also have the security package which authenticates the user before being able to enter the user application.

Mapping Subsystems to Hardware

Our system requires multiple connected devices in order to function normally. The system has a client and a server subsystem. We can identify our clients to be mobile devices running our mobile application which will represent their own independent machines. Each client has to connect to the server, which is represented by the Raspberry Pi unit we have working in conjunction with our Arduino Base Unit (ABU). The Arduino Wifi Shield will give the system wifi capabilities allowing for connection to the Firebase wirelessly as well as connection by the mobile devices via HTTP requests. The system will be able to send and receive data from the Firebase, particularly photos taken by the Arducam allowing for the facial recognition software to run efficiently. This server will also allow mobile devices (clients) to send command signals from one end, and the Raspberry Pi (server) to receive these signals and pass them to the Arduino Base Unit on the other end.

Persistent Data Storage

With our system, we need to be able to save data that can be accessed by both the user and the Arduino. We plan to use Firebase to satisfy our database storage needs [19]. In order to utilize our facial recognition functionality we need to store all the faces that should be approved by the facial recognition software to automatically disable the alarms when opening the door. The Arduino Base Unit should be able to access this and cross reference these pictures with pictures it takes to determine if the images match. Also, the ABU should be able to store images of perceived intruders in a separate collection in the Firebase, so that the user will be able to access these images remotely. In addition to pictures for the saved faces, and intruder images, we have to be able to store info about users that have access to the Android app and adjust privileges if necessary, and be able to authenticate them with a password [7]. Because we are implementing Firebase, we don't necessarily use relational tables, as it is an unstructured database that follows more closely to a json format. Json stands for JavaScript object notation, which is a form of storing and displaying information in a collection of key:value pairs.

Firebase

- store saved faces for facial recognition

- store pics of intruders/unrecognized visitors
- user info for login authentication
 - username, password, privileges,
 - sheets (collections) vs. tables
 - There are many commands that we can use when we connect firebase with our android app, such as
 - `FirebaseDataReference = FirebaseDatabase.getInstance().getReference();`
 - This establishes a connection to your firebase with the proper authentication. This lets you read and write through this reference.
 - `DatabaseReference messageRef = FirebaseDataReference.child();`
 - The child of the current reference is a way to navigate when a key value has multiple values.
 - `StorageReference storageReference = FirebaseStorage.getInstance().getReferenceFromUrl(imageUrl);`
 - Similar to the database reference, this is the reference to the storage area that will hold images.
 - `FirebaseDatabaseReference.child(MESSAGES_CHILD).push().setValue(friendlyMessage);`
 - Instead of reading from the database, this is how to write into the database
 - `StorageReference storageReference = FirebaseStorage.getInstance().getReference(mFirebaseUser.getUid()).child(key).child(uri.getLastPathSegment());`
 - `putImageInStorage(storageReference, uri, key);`
 - This is a way to put images into the storage of the database.

Network Protocol

In order to connect the entire system so that multiple devices can interact with it, a server client network system will be implemented. The server client network system will utilize simple java sockets. When a user logs into the mobile application, they, as a client, ping

the server (Raspberry Pi) and attempt to make a connection to it. By utilizing multithreading, multiple users can connect to the aforementioned server at the same time using their mobile application and all send requests which will be handled on a queue basis. Once a device is connected, the user can send HTTP requests to the Raspberry Pi, and tell the Arduino Base Unit to toggle its hardware components on or off. When the user toggles the lights, for example, on the mobile application, the request is sent from client to the server. The Raspberry Pi server will then send data to edit on/off values to the Arduino Base Unit.

Global Control Flow

Our program is event-driven, since certain events will occur depending on the user triggering it. For example, the alarm will only go off if the door is opened and away mode is still enabled. Furthermore, the order that the events occur are variable, since they are all dependent on what action the user initiates. While we do have loops and waits, they can generate different actions based on the order that they are executed.

As per time dependency, there are a few timers in our system. Mainly, once the alarm is triggered, the user will have 30 seconds to disable it through the mobile application. Despite that, our system is mostly an event response system. Some examples of this include:

- The motion sensor immediately triggers the Arducam to take a picture
- The picture will be automatically cross-checked with the list of saved faces stored in firebase
- Anyone with the mobile app should also immediately be able to view the pictures that the arducam captured
- If desired, the user can also take an immediate picture of what is outside the door with the Arducam
- Once the accelerometer detects motion of the door opening, the alarm and lights will be enabled immediately

For concurrency, we plan on using multiple threads when we initiate different users as they connect to the base unit. This will prevent multiple users from simultaneously changing the system and causing issues. We can prevent these issues by using mutex locks so variables controlling certain devices cannot get changed at the same time. We can also create threads when we push images up into the firebase server, as there might

be some wifi latency that we can shorten by having multiple images uploading at the same time with threads.

Hardware Requirements

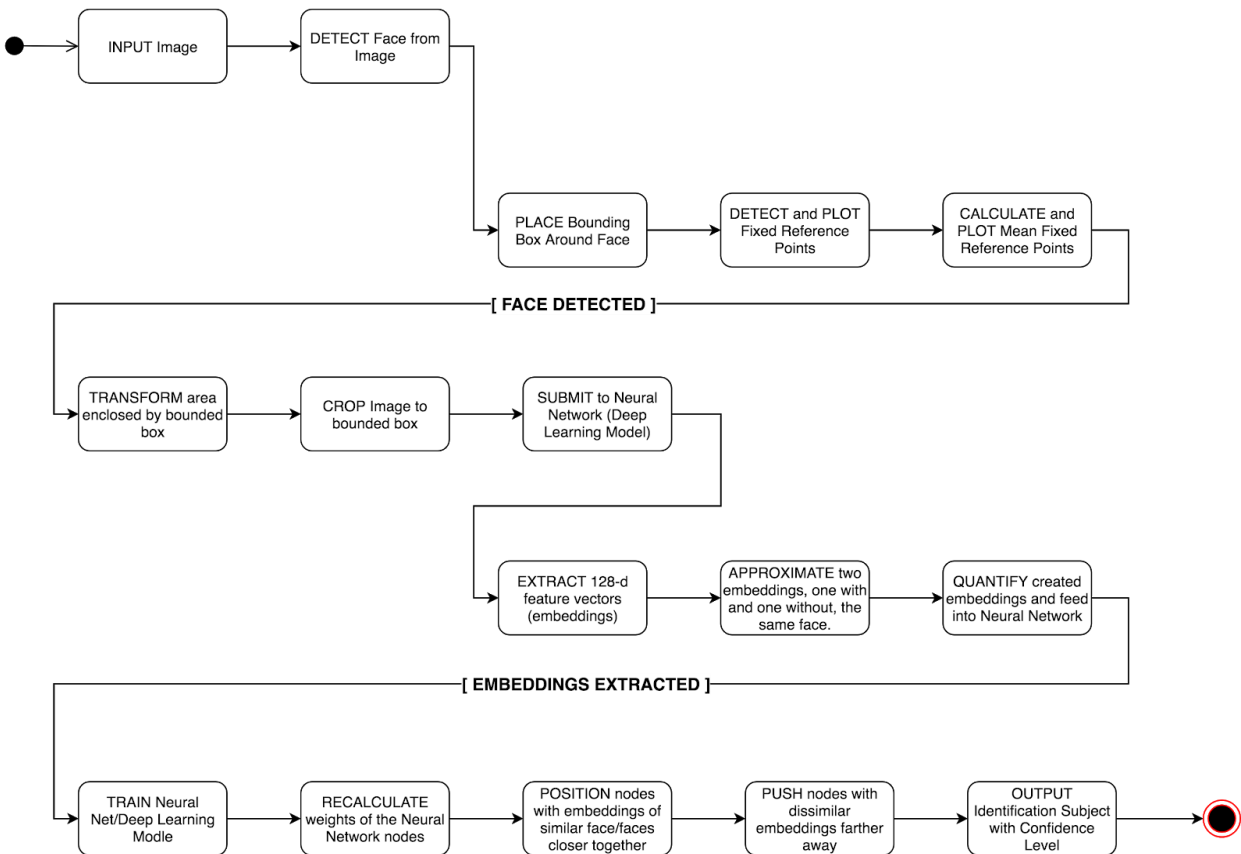
Below is the proposed structure for or hardware will be set up, and connected. In the base unit, the arduino will control the actions of the external devices. To communicate with the mobile application, the wifi shield is needed to let the arduino effectively become an Internet of Things device, allowing us to control and trigger events through the mobile application over the internet, and to connect the arduino to Firebase to pull data and photos from that online database. In addition we are utilizing a Raspberry Pi, which would allow us to store more pictures and give the Arduino system much more functionality, as it is a mini computer. We can connect the alarm directly to the Arduino in the base unit. However, to communicate with the light, since that will often be an external light somewhere else in the house, we will be using a transmitter and a receiver to communicate signals wirelessly. While we only have one LED, this system easily offers expandability towards adding multiple LEDs. Finally, the camera, motion sensor, and accelerometer will be in an enclosed box attached to the door, with the camera poking through the existing peephole of the door. They will be connected through a wire to the base unit.

Algorithms and Data Structures

Algorithms

The facial recognition API sourced in the OpenCV library utilizes face detection and deep learning in order to accurately recognize people in images and video streams. The overall process begins by receiving an input image. The presence and location of faces are detected and found from the input using pre-trained models from dlib and OpenCV, both of which are libraries with facial detection functions [21]. The detection algorithm places a bounding box around the profile of the face, and calculates average fixed reference points based on all the fixed reference points that can be detected [22]. Using the bounding box and mean fiducial points, the face is then transformed for the neural network, which is a construct in machine learning that utilizes different algorithms

to process complex data inputs [23]. Using the real-time pose estimation function from the dlib library and the affine transformation function from the OpenCV library, the input image is transformed to make the eyes and bottom lip appear in the same location on each image [22]. The deep neural network then quantifies the face (cropped by the bounding box) by embedding the face on a 128-dimensional unit hypersphere [22]. The embedding generalizes the inputted face, and is able to compare with other embeddings based on quantitative measures, making clustering, similarity detection, and classification more efficient and successful [21]. Creating the embedding involves training the neural net by positioning nodes (which have equal or near equal embeddings) closer to each other, and pushing the embeddings with different values (which don't have near similar faces) farther away [21]. The below activity diagram summarizes the process pipeline.



Data Structures

Our application makes minimal use of data structures as most of our data will be stored into the Firebase. Our app will utilize arraylists that contain all the images the user can view. The arraylist will contain photo objects that the app can easily manipulate and select from different pictures to display to the user. Storing the pictures in an arraylist also reduces the amount of times the app must interface with the Firebase to retrieve data in one session. Every time the app is loaded and the user wants to view the pictures stored on the Firebase the app will retrieve the data from the firebase and load it into an arraylist for the current session. We utilize arraylists because they can easily change size and allow for random memory access. Arraylists are also better for displaying data in different types of Android UI components as many UI functions are set to already interface with arraylists. Furthermore, we will also be using arraylists for the list of users that will appear on the mobile app. The users will of course be user objects which will contain the user privileges.

User Interface Design and Implementation

Utilizing Android Studio, we have created very similar User Interface Design for the project. While the process is still in progress, it seems as though there have not been any significant changes to our designs and we will be following the mock up designs. The mock up designs were structured in a way to keep the design intuitive and possible to replicate on an android development platform such as Android Studio. A potential edit would be to implement pop up boxes upon waiting for fingerprint authentication or the status of an action (i.e. succeeded or failed). There may also be the need to implement a photo inspection page so the user can choose whether they want to use the picture they just took to add to the facial recognition database. The feature of facial recognition will also require more than one picture so the model can be trained properly, instead it will require five pictures of the person being registered into the facial recognition database, so this user interface will indicate the number of pictures taken, the ability to view them and the ability to take other pictures.

The user interface that was utilized in this project is, to the best of our knowledge, the most optimal for our software. The User Interface models and schematics that were created prior to actually making this project were used and followed closely for the majority of designing the UI portion of this application. One of the changes that was made was part of the main menu page with all of the buttons that interacted with the hardware components of the Arduino Base Unit. This change was a functional change by adding the arm and disarm button to provide different settings for the alarm. Changes were implemented to handle better user experience and make it more intuitive while also allowing the user to be able to do everything they need to. The

aesthetics were unchanged as everything was already designed with color and text to make the application easy to navigate.

Design of Tests

Testing Manual Operation of Arduino Base Unit

- User manually takes picture on the arducam
 - We want to be able to have a user make the system take a picture of whatever the camera currently sees to show that the camera does have functionality beyond the automatic photo taken when the system is activated and can be controlled manually.
 - In our demo, we connected to the arduino and sent the signal for it to activate the camera to take a picture and retrieve the photo taken.
 - We connected the mobile app to the arduino system to have a more directly connected relationship to each other so the function in the application talks to the arduino to take the photo without needed so much of a middle man laptop.
- User manually turns on/off motion sensor
 - We want to be able to turn on/off the motion sensor to show how when the system is disengaged, the system's motion sensor will not be constantly seeking out movement, which would then consequently trigger the arducam to take a picture of any visitor that approaches the door.
 - In our first demo, we showed our progress by directly connecting a laptop to the Arduino Base Unit and using commands from the laptop to simulate how the user would be able to manually take a picture of the exterior of the door.
 - For our final product and second demo, we had the Android app have the capability to directly connect to the ABU and directly toggle this setting, without the need for a direct connection established between the laptop and the ABU.
- User can tell if door is open (accelerometer)
 - We want to be able to know if the door is opened or in the process of being opened by having the accelerometer report data of motion and angle and send notification of the doors status.
 - In our first demo, we made a miniaturized model door with a hinge and begin to open it until the accelerometer identifies the motion and angle and notifies on the device used of the door's status changing from open to close.
 - For our second demo, we had the mobile app connect more directly to the arduino so the notification is presented to the user in the application without needing any device in between.

Testing App Operation

- App should be able to log user in and out

- We want the app to be accessible by multiple users and we want to make sure that the mobile app itself is not accessible by just anyone. That is why we wanted to implement a login page to restrict access to users with privileges and also restrict users with restricted access.
- For the first demo, we demonstrated how our mobile app has the ability to restrict access to only those users that have access with their usernames and passwords. We demonstrated this by pulling up a version of the Android app on an Android device, and show the user authentication system for the app. We also showed how the main page allows for the creation of new accounts in addition to logging in.
- In the final product and for the second demo, we showed how we can create additional accounts with restricted access, and show how user settings/privileges can be added or modified. We still plan on implementing fingerprint authentication if the Android device allows for it.
- App should allow management of user settings
 - We want the app to not only be able to have more than one user but also possess the ability to show said user's settings so different features or priorities can be adjusted depending on the user.
 - In our first demo, we showed that after a authorized user is able to login then they can go to a page that displays their user settings and to show that they can't be changed and toggled.
 - In the second demo, we had the direct connectivity between the app and the whole system be more viable and have different users login and adjust their settings to then demonstrate what works and what does not based on what that user's settings allow.
- App should connect to firebase and read and write data
 - We want our app to be have a connection to our firebase in order for it to read and write information necessary for the system whether it is in the form of login data and face data for facial recognition for example.
 - In our demo, we plan to have some communication between the app and the firebase in order to show other functions such as logging in as that is a function that needs that connection and also show off the firebase and layout.
 - In the second demo, we made this connectivity to the next level so it will all be connected as a system allowing each function to work properly without a intermediary device acting as a trigger or anything and show interconnected actions that lead to the manipulation of the alarm system.

Facial Recognition

-Person Detection

- We want to be able to have the facial recognition software determine if the user is one of the registered users so that the system can confidently disarm the system knowing that a registered user is outside.
- For the first demo we demonstrated our progress made in the facial recognition software by showing how a picture will demonstrate the closeness of matches to an individual.
- For the final product and the final demo, we had this facial recognition software run automatically upon image capture, and determined if it wants to disarm the system. The results and matching were much more accurate.

History of Work, Current Status, and Future Work

As per hardware for our system, we reached an impasse when we realized we could no longer just use the wifi shield to communicate between the arduino and the app. The large amounts of data that needed to be sent was inefficient and slow. Therefore, we had to switch to using a Raspberry Pi since some processing power can be executed on it first, and then it can be sent to the app. We are able to get the Pi and mobile app to communicate directly, and the Pi is directly connected to the arduino as well.

For future work, we need to get the arducam to be able to send pictures through the Pi. Currently, we can only get the pictures saved on the laptop we are using to test. Also, while we can get readings from the motion sensor, we need to get those readings to activate the Arducam. Additionally, we are learning how to set up the receiver-transmitter module so that we can activate the light remotely.

For the mobile application, good progress has been made to be able to download pictures from firebase and display it on the app. This will be helpful for when users want to be able to see who is at the door. For future work, we need to find a way to upload pictures to firebase from the Pi. Also, we need to create user-triggered methods to activate and deactivate the lights and alarms through the app.

Specifically for firebase, we have been able to store images and URLs of the pictures. However, a lot of work needs to be done to connect the firebase to Pi. One possibility is to run a bash script to detect when a new picture is taken and stored in a folder. Then, the script will move that image into a destination server, run a python script on that folder that will upload the images to firebase, and take the URL of the images to retrieve the pictures in firebase. The app will go through the URL to get the images, and then run the facial recognition python script in bash in a loop once the pictures are uploaded. This is theoretically how we will approach solving this issue.

Finally, for facial recognition, we made great progress in understanding the best way to train the dataset necessary for accurate facial recognition. Things like neutral smiles and good lighting are good image types to use. Right now, we are working on getting the facial recognition

connected to the Pi. We are having some complications with installing OpenCV on the Pi, and firebase is frustrating to use since you cannot download entire folders - instead you must download only pictures you know the exact image names of. We are working on getting through these issues to connect all parts of the system.

References

[1]ARDUINO/BLEETOOTH APP CONNECTION.

<https://www.youtube.com/watch?v=evVRCL9-TWs>

[2]ARDUINO WIFI CONNECTION.

<https://www.instructables.com/id/How-to-connect-your-Arduino-WiFi-shield-to-a-custo/>

[3]ARDUINO CAMERA. <https://www.instructables.com/id/ArduCAM-Mini-ESP8266-Web-Camera/>

[4]ARDUINO/ANDROID APP REAL-TIME COMMUNICATION.

<https://medium.com/coinmonks/arduino-to-android-real-time-communication-for-iot-with-firebase-60df579f962>

[5]ARDUINO WEB SERVER.

<https://startingelectronics.org/tutorials/arduino/ethernet-shield-web-server-tutorial/basic-web-server/>

[6]RASPBERRY PI/ARDUINO COMMUNICATION.

<https://maker.pro/raspberry-pi/tutorial/how-to-connect-and-interface-raspberry-pi-with-arduino>

[7]ARDUINO/ANDROID APP CONNECTIVITY.

<https://www.makeuseof.com/tag/6-easy-ways-connect-arduino-android/>

[8]GANTT CHARTS. <https://www.officetimeline.com/make-gantt-chart/google-docs>

[9]ARDUINO FACE RECOGNITION.

<https://www.hackster.io/team-enzi/alexa-controlled-face-recognizing-arduino-door-bell-465a58>

[10]FIREBASE USER REGISTRATION. <https://www.youtube.com/watch?v=0NFwF7L-YA8>

[11]FIREBASE USER LOGIN AND USER SESSION.

<https://www.youtube.com/watch?v=KFULmVXpO-A>

[12]USER STORIES. https://en.wikipedia.org/wiki/User_story

[13]FURPS. <https://en.wikipedia.org/wiki/FURPS>

[14]PROJECT ROADMAP. <https://blog.asana.com/2018/08/product-roadmap-tips-templates/>

[15]ACCEPTANCE TESTING. https://en.wikipedia.org/wiki/Acceptance_testing

[16]INTERACTIVE DIAGRAMS.

https://en.wikipedia.org/wiki/Unified_Modeling_Language#Interaction_diagrams

[17]SEQUENCE DIAGRAMS. https://en.wikipedia.org/wiki/Sequence_diagram

[18]SOFTWARE ARCHITECTURE.

https://en.wikipedia.org/wiki/Software_architecture#Examples_of_Architectural_Styles_2F_Patterns

[19]ARDUINO-FIREBASE LIBRARY

<https://github.com/FirebaseExtended/firebase-arduino>

[20]LAYERED-ARCHITECTURE

<https://herbertograca.com/2017/08/03/layered-architecture/>

[21]OPENCV FACIAL RECOGNITION

<https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/>

[22]FACIAL RECOGNITION ALGORITHM

<https://cmusatyalab.github.io/openface/#overview>

[23]UDACITY TENSORFLOW COURSE

<https://classroom.udacity.com/courses/ud187>

[24]Accessing images from the Firebase database

<https://androidjson.com/retrieve-stored-images-firebase-storage/>

[25]Saving images into the Firebase database

<https://androidjson.com/upload-image-to-firebase-storage/>

[26]Retrieving images from Firebase storage

<https://codinginflow.com/tutorials/android/firebase-storage-upload-and-retrieve-images/part-6-retrieve-images>

[27]Controlling a servo remotely with a raspberry pi

<https://www.youtube.com/watch?v=t8THp3mhbdA>

Project Management

What a week! We were all very stressed for the last demo, especially since just a week before our scheduled time, even though we had a lot of the individual parts of the project working, we did not have the parts all connected. However, we spent basically every night together once we were all free to continue working on the project together as a unit, connecting each part of the project. On Monday night, the day before our demo, we stayed up until 2am finalizing the demo, the slide deck, the door model, and exactly which features we wanted to show off during the demo. We had a great time together, making plenty of jokes that alleviated the stress of the impending deadline. Overall, the team felt very close by the end of it. After the demo, we were very proud of how it went, and we took a cute picture together with our project. This was a rewarding experience that made us get significantly closer.