Senior Project

House Hawk

Home Security Application

DEVELOPED BY

Jake Caggese
Joshua DeNoble
Andrew Freitas
Joe LaGrossa
Ryan Lindsay
Matt Moore
Domenick Palmiotto

SCRUM MASTER

Joshua DeNoble

GITHUB LOCATION

https://github.com/RyanLindsay48/CSSeniorProject

SLACK WORKSPACE

seniorproject-4.slack.com

Project Summary

The House Hawk system is designed to provide basic home monitoring to an end user. It utilizes a Raspberry Pi outfitted with two modules to record any necessary home security information. This information is then forwarded via a RESTful service and stored in a backend database system. Finally, the user is able to interact with a mobile phone application to view any potential home security problems sent from the Raspberry Pi.

High Level Description

The Raspberry Pi (RPi) is outfitted with two attachments which enable it to determine if the home's security is potentially compromised and take pictures if any suspicious activity is occuring. The first module is a passive infrared (PIR) sensor, which uses infrared radiation to determine if a warm body is moving in a specified radius. If a body is found in the field of view then the PIR sensor returns a signal to the main script indicating that something is present. The second module is a camera based module, which takes pictures of the area every time step after motion is detected.

Images taken from the camera module are sent to a server instance located on Amazon Web Services (AWS) through a Flask-RESTPlus API. Image files themselves are sent to and stored on the server file system under a unique identifier for each user. During this time, the endpoint also stores the file path to each image into a database system. The database chosen for the House Hawk system is a MySQL database due to exceptional performance and scalability should the House Hawk system require expansion in the future.

Once images are stored on the server and links to the files are created in the database, the mobile application can retrieve the images and notify the user. Images are retrieved from the server and sent to the mobile app through another endpoint located on the RESTPlus API system. The mobile app then sends a push notification to the user indicating that their system has recorded images to view.

Once the user clicks on the notification, the mobile device then navigates them to a screen used to view the most recently captured images. On this screen the user can click the refresh button to pull the newest pictures, and if the user determines that no further action is required they can send a signal which notifies the camera system to stop taking pictures and reset.

Problem Solving Approaches

Throughout the design process many different approaches were considered for various parts of the development process. Considerations include use of different information capturing systems, RESTful services, database systems, and mobile application development programs. This section of the report will walk through the different options that were considered during the design phase, and provide insight on why the current systems were chosen.

Image Capturing System

Two image capturing systems were considered for use in the House Hawk system. Initially, a webcam based camera with a built in motion sensor was considered, which allowed the system to be simplified by combining both the detection and capture systems together. However, after careful consideration a RPi based solution utilizing a sensor and camera module was chosen.

While the webcam based camera system allowed the image capturing to be simplified, it was determined that setting up one of these systems was much more expensive than the project allowed. Additionally, the webcam system would need to be completely replaced if the image capturing system was required to be replaced, or updated in the future (better resolution, increased functionality). The RPi based system on the other hand was relatively inexpensive and allowed the image capturing system as a whole to be broken into smaller components. This allows the system to be more expandable in the future, as only a certain module can be replaced instead of the image capturing system in its entirety.

RESTful API Services

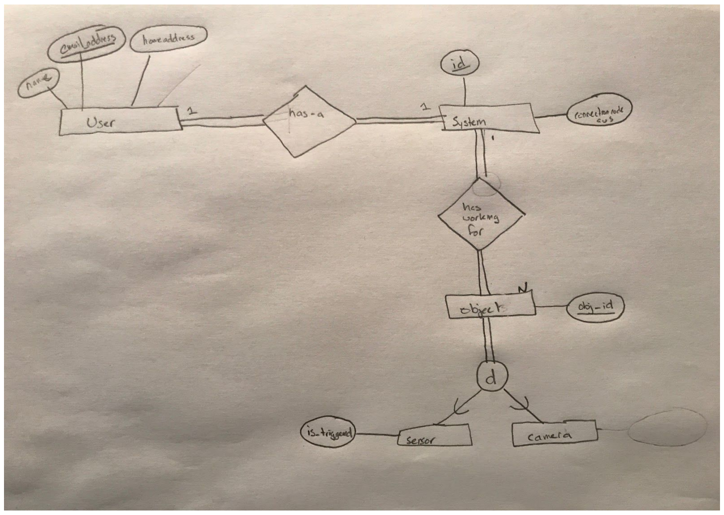
When looking at different frameworks to model the RESTful API services from two major contenders came to mind. Our first consideration was Django, the one of the most popular fully fletched out Python framework available. It consists of a full-featured Model-View-Controller framework, including a web-based IDE to develop in. Our second consideration was Flask, one of the most popular micro-frameworks available for Python. Because of its small size Flask does not include all the bells and whistles available in a full sized framework like Django. However, with the use of community extensions, Flask can be expanded to include Django-like capabilities.

Due to its small size and lower overhead, Flask was chosen for the House Hawk system. Because of its minimalistic approach, Flask has no built-in database services. However, with a community extension available the database system chosen could be easily incorporated into Flask, providing all the functionality required by the framework.

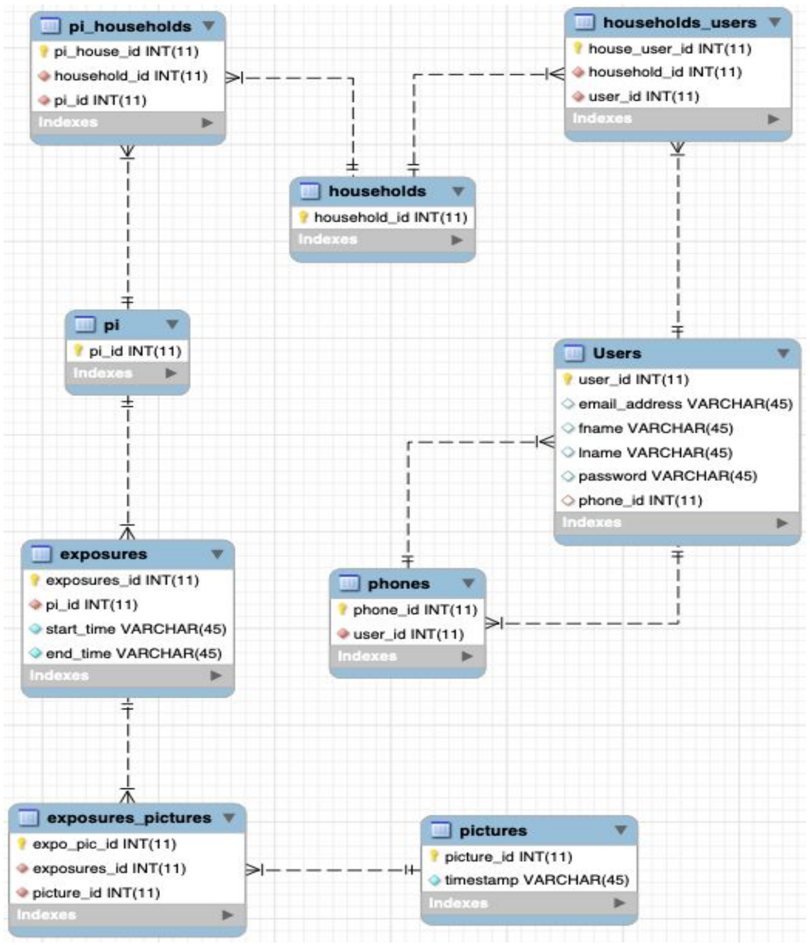
Database System

For our database system we chose to use MySQL. This choice was made based on the team's experience with database programming. MySQL is very well documented making it easy to learn and use compared to some of the less popular databases.

For our Database Model, we started simply. Our original model had entities for a User, the Raspberry Pi system, as well as the sensor and camera we'd be using.

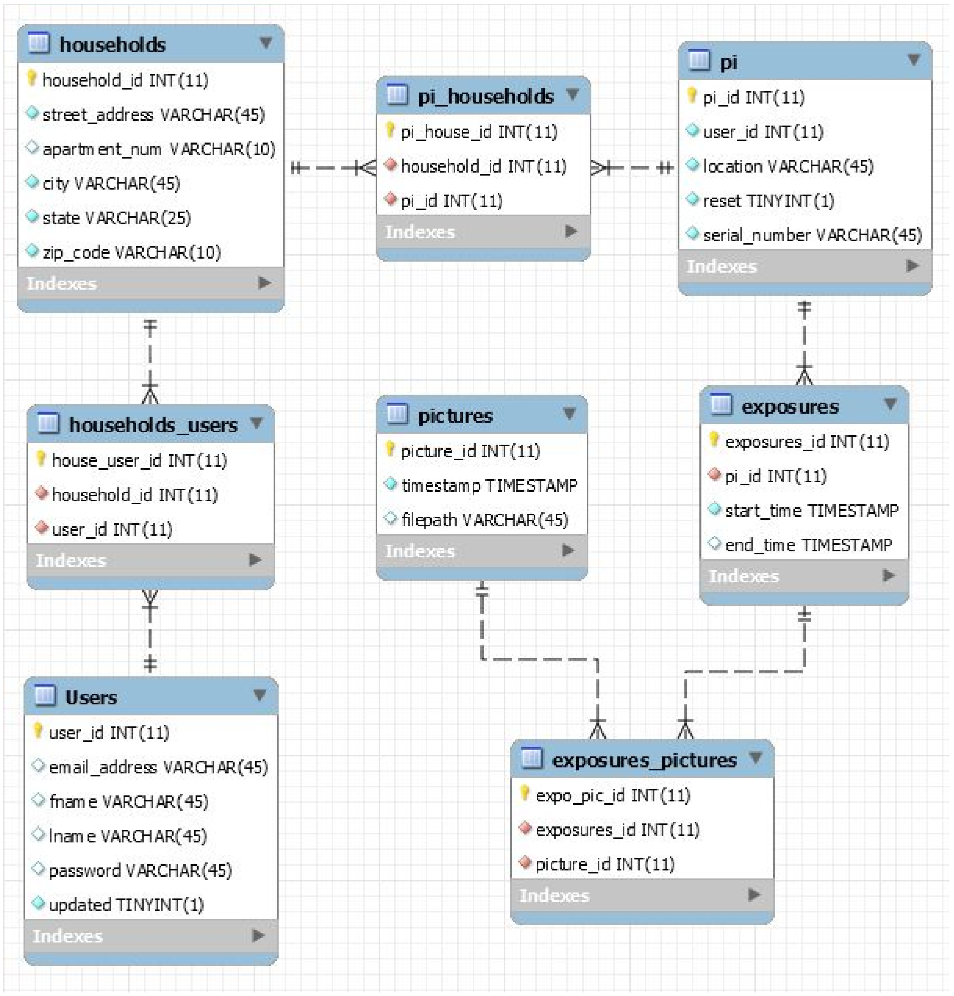


The next model we came up with was more detailed and built with growth in mind.



We kept the User and Pi tables, but added several columns on to the Users table that we thought might be necessary. We also got rid of the Sensor and Camera tables since we realized we don’t really need to store any information about them individually. Instead we added tables for Exposures and Pictures. An exposure can be described as a set (represented by a folder) of pictures taken between the time the sensor was activated and when either the user resets the system or enough pictures have been taken to meet the preset threshold. We also added the `households` table. Each household can have many pis and Users, though for our use case, pi’s can only belong to one Household at a time.

Our final model had a few alterations. We completely removed the `phones` table since our use case no longer required that we store any information about the user’s phone.



We also completed adding the necessary columns to households, pi, and pictures. For users we got rid of the phone_id column, since we no longer have that table, and added an “updated” column, which holds a single digit (in our case we’ll only change it to either 0 or 1) and that let’s the mobile application know whether or not a new row has been added to the exposure table, which then activates a notification being sent to the mobile app.

Mobile Application OS

The mobile phone market is mainly ruled by two primary contenders at this time: Google's Android, and Apple's iOS. Both of these OS's have different development platforms, and their own respective stores/markets. In order for House Hawk to have the largest marketing potential, it would be important to create mobile phone applications for both the Android and iOS systems. While this would typically mean developing an application on Android Studio (for Android) and creating a sister app on XCode (for iOS), a more recent development platform has been produced recently.

Flutter is an open-source mobile application development SDK developed by Google for use in development of mobile phone applications. This SDK allows for development in Android, iOS, as well as Google Fuchsia (an upcoming new OS). Because of its ability to reach all of the main mobile phone platforms, Flutter was chosen as our mobile OS development software of choice. This allows House Hawk mobile app development team to streamline their work, and develop the application once, instead of two times for two different operating systems.

App Screen Mockups

Although the House Hawk mobile application itself is not extremely complex, it still consists of different screens that allow the user to access their specific information. Since all information stored in the server is tied directly to the user's account, a login screen allows users to insure that the mobile application is only displaying their information (**Figure 1**). If the user does not have an account already setup, information can be collected and an account can be created (**Figure 2**).

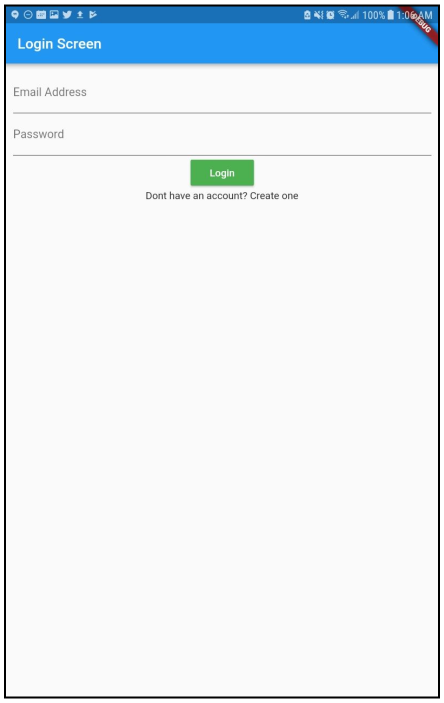


Figure 1: House Hawk Login Screen

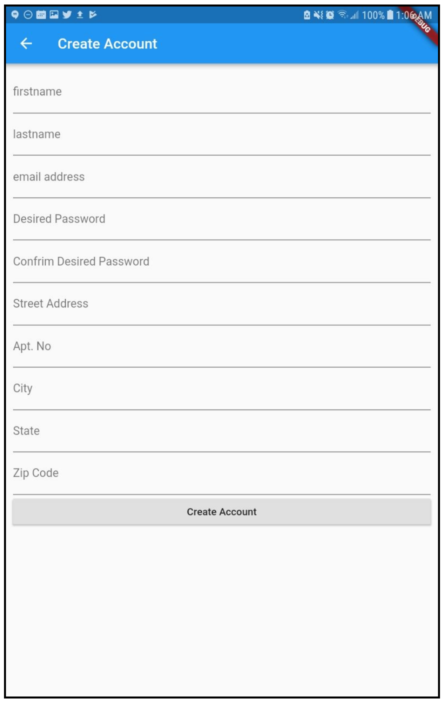


Figure 2: Create Account Screen

Once the user has logged into their account, a home screen is displayed (**Figure 3**). This screen allows the user to view the most recent exposure, their cameras, and account information. If the user clicks to view their account info, their information will be displayed (**Figure 4**).

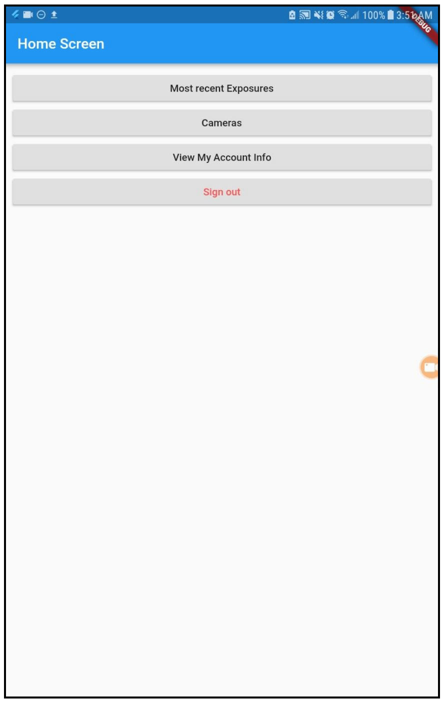
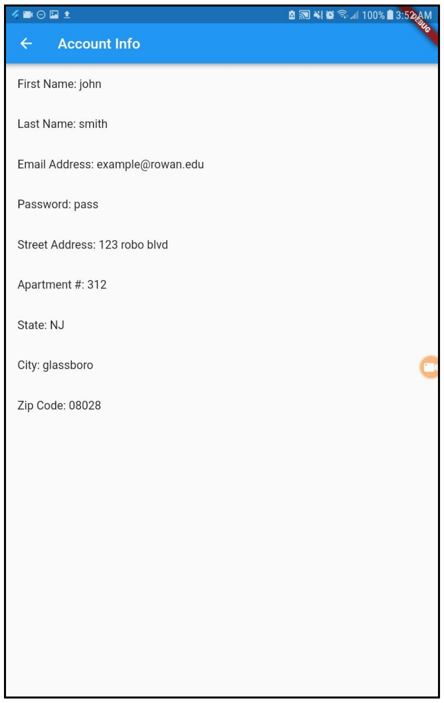


Figure 3: Home Screen



First Name: john

Last Name: smith

Email Address: example@rowan.edu

Password: pass

Street Address: 123 robo blvd

Apartment #: 312

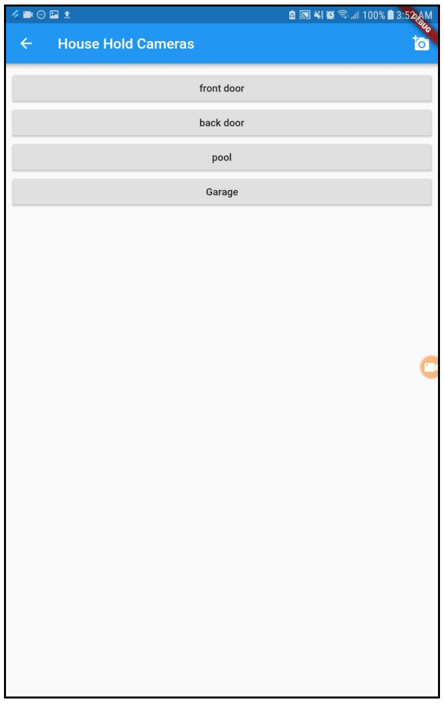
State: NJ

City: glassboro

Zip Code: 08028

Figure 4: User's Account Information

The user can also click the Cameras button to view all cameras associated with their account (**Figure 5**). The add camera button may be used to add a camera to the users account given user supplied information (**Figure 6**).


House Hold Cameras

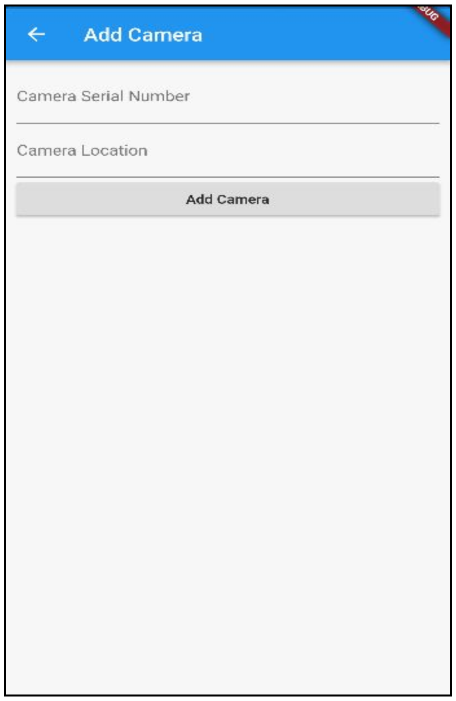
front door

back door

pool

Garage

Figure 5: User’s Camera Screen


Add Camera

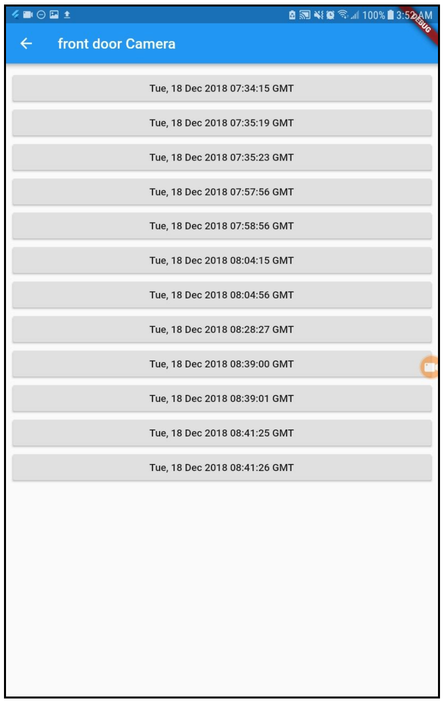
Camera Serial Number

Camera Location

Add Camera

Figure 6: Add Camera Screen

From the camera screen the user can select a camera to view all exposures for that given camera (**Figure 7**). A exposure can be selected on this screen to view all pictures listed in that exposure (**Figure 8**). Additionally, the most recent exposure screen looks similar to the pictures screen but only shows the most recent exposure.


front door Camera

Tue, 18 Dec 2018 07:34:15 GMT

Tue, 18 Dec 2018 07:35:19 GMT

Tue, 18 Dec 2018 07:35:23 GMT

Tue, 18 Dec 2018 07:57:56 GMT

Tue, 18 Dec 2018 07:58:56 GMT

Tue, 18 Dec 2018 08:04:15 GMT

Tue, 18 Dec 2018 08:04:56 GMT

Tue, 18 Dec 2018 08:28:27 GMT

Tue, 18 Dec 2018 08:39:00 GMT

Tue, 18 Dec 2018 08:39:01 GMT

Tue, 18 Dec 2018 08:41:25 GMT

Tue, 18 Dec 2018 08:41:26 GMT

Figure 7: Camera’s Exposures Screen

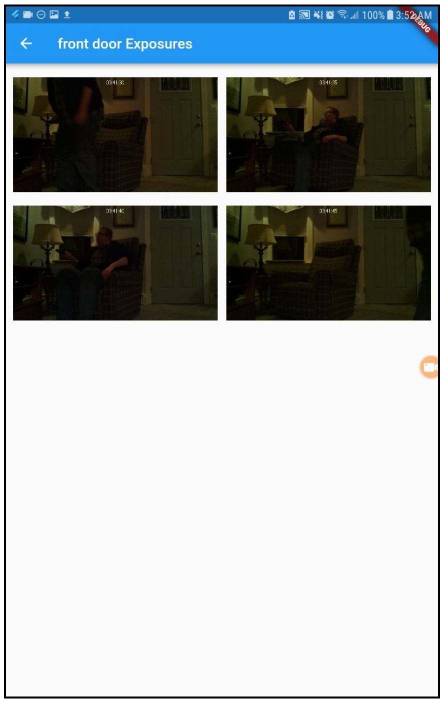
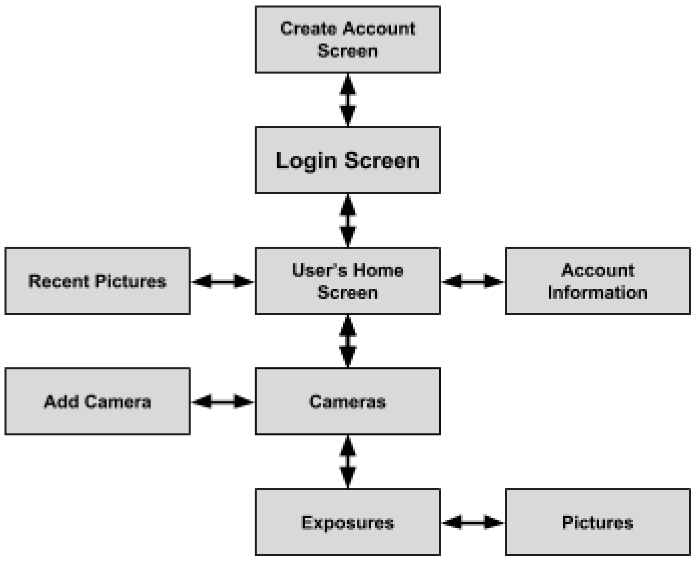


Figure 8: Pictures/Recent Display Screen

App Screen Navigation



The House Hawk application will navigate between the different app screens based on user input. The base screen for the application is the login screen. If the user does not have a current account, they can click the 'Create' button to navigate to a screen which will allow them to create an account. Finally, if the user has all of their account information they can enter the information in the appropriate text boxes, and click the 'Login' button to enter the user's home screen for the House Hawk mobile application.

Once the user is on their home screen they can click the 'View my Account Info' button view their account information, click the 'Most Recent Exposure' button to view the most recent exposure, click the 'Cameras' button to view all their cameras, or click the 'Logout' button to return to the login screen.

Once the user has selected the camera screen they can click the 'Add Camera' button to add a new camera to their account, click on a camera button to navigate into that camera, or click the back button to go back to the home screen.

If the user has entered a specific camera system then all exposures for that camera will be displayed. The user can click on a exposure button to display all images located in that exposure, or click the back button to go back to the cameras screen.

A general overview of House Hawk navigation procedures can be seen in **Table 1**. This provides a listing of the current screen, and possible endscreens given a procedure.

Table 1: House Hawk Mobile Application Navigation Procedures

INITIAL SCREEN	**PROCEDURE**	**END SCREEN**
Login Screen	*Enter information & click login button*	*User's Home Screen*
	Click new user button	*Create Account Screen*
Create Account Screen	*Enter information & click create button*	*Login Screen*
	Click back button	*Login Screen*
User's Home Screen	*Click HouseHold Cameras Screen*	*HouseHold Cameras Screen*
	Click Most Recent Exposures	*Most Recent Exposures Screen*
	Click logout button	*Login Screen*
Account Info Screen	*Click back button*	*User's Home Screen*

HouseHold Camera Screen	*Click on a certain camera*	*Pi Exposures Screen*
	Click the add camera button in the top right	*Add Camera Screen*
	Click the back button	*Home Screen*
Add Camera Screen	*Enter information & click Add Camera button*	*HouseHold Cameras Screen*
	Click back button	*HouseHold Cameras Screen*
Pi Exposures Screen Screen	*Click on any exposure button*	*Show Exposure Screen*
	Click back button	*HouseHold Cameras Screen*
Show Exposures Screen	*Click back button*	*Pi Exposures Screen*
Most Recent Exposures Screen	*Click Refresh Button*	*Most Recent Exposures Screen (Calls screen again with updated picture list)*
	Click the back button	*Home Screen*

Backend Information

There are two main backend components used in the House Hawk program, a MySQL database responsible for storing information and a RESTful API connecting to it. The RESTful API is deployed through Flask, which utilizes SQLAlchemy integrated with Marshmallow. This Object Relational Mapper provides the application with a “Pythonic” way of interacting with the database. A schema was developed to show the relationships between database information stored on the tables. Fields and description of attributes for each table in the database can be seen in **Table 2**. In order to transfer information from the system to the database a RESTful API was developed. Multiple endpoints were created to handle the different types of information that would be passed between systems. A list of endpoints that have been developed can be located in **Table 3**.

Table 2: Description of Database Table Attributes

TABLE	COLUMNS	DESCRIPTION
exposures	*exposures_id* *pi_id* *start_time* *end_time*	*Provides a grouping of pictures from a given time period taken from a specified camera (Pi) system*
exposures_pictures*	*expo_pic_id* *exposures_id* *picture_id*	*Links pictures taken to a grouping of pictures given the picture and exposure IDs*
households	*household_id* *street_address* *apartment_num* *city* *state* *zip_code*	*Provides each household with a unique ID*
households_users*	*house_user_id* *household_id* *user_id*	*Links each user to a specified household given their user ID and associate household ID*
pi	*pi_id*	*Provides each Pi with a unique ID*

	user_id *location* *reset* *serial_number*	
pi_households*	*pi_house_id* *household_id* *pi_id*	*Links different camera (Pi) system to specified households given the Pi and household IDs*
pictures	*picture_id* *timestamp* *filepath*	*Provides pictures a unique ID and timestamp that the picture was taken*
Users	*user_id* *Email_address* *fname* *lname* *password* *updated*	*Provides contact information such as name, email address. Describes the user's account using their unique ID and password. Also determines which phone belongs to that user's account*

*Indicates a linking table

Table 3: List of RESTful API Endpoints used

ENDPOINT	**INPUT**	**OUTPUT**	**FUNCTIONALITY**
/api/user/login	*Username & password*	*Navigation to selected page*	*Login to user account*
/api/user/login	*Text for email*	*Json file of a user object from the database*	*Checks to see if the credentials entered by the user are valid.*
/api/user/household?user_id	*user_id*	*Json file of a house associated to a user's user_id*	*Used to save the user's household information in global variables so the user can view their information on the Account Info Screen*
/api/user/updated	*user_id*	*Returns the updated variable from the logged in user.*	*Used to see if the updated value in the logged in user has changed. If the updated variable in the database has changed that means that there are new exposures associated with a user.*
/api/user/updated/	*User_id, '0'*	*PUT*	*Used to set the value of the of updated for the user back to 0 in the database.*
/api/user/account	*Text for user information, username and password*	*Navigation to selected page*	*Takes data entered by user and stores it in the database.*
/api/users	*None*	*List of all uses on the server*	*Used to return a list of users from the database. We compare the users entered email address to those already in the database to see if it is valid.*
/api/user	*Users email address, password, firstname and lastname.*	*POST*	*Adds a user to the database*

/api/household	*Users user_id, street address, city, state, and zip code. Apartment number is optional for this call. .*	*POST*	*Allows a user's house to be added into the database.*
/api/exposure/recent	*Returns a list of all of the images from the user's most recent exposure.*	*user_id*	*Used to get all of the images from the user's most recent exposure.*
/api/user/pis	*Returns a list of all cameras associated with a user*	*user_id*	*Used to get all of the cameras associated to a user.*
/api/pi/exposures	*Returns all of the exposures from a camera for a user*	*A json list of all of the exposures from a specific camera*	*Used to return a list of all of the exposures from a specific camera.*
api/pi?user _id	*Takes a user_id, pi serial number and a location*	*POST*	*Allows the user to add another pi to their account.*
api/pi/reset	*Takes a pi_id and a reset value of 1*	*PUT*	*Allows the user to turn their camera off after they've determined that there is no intruder in their house.*

App Screen Functionality

The House Hawk mobile application has nine screens associated with it: Login Screen, Create Account Screen, Home Screen, Account Info Screen, Most Recent Exposures Screen, HouseHold Camera Screen, Add Camera Screen, Pi Exposures Screen and Show Exposures Screen.

The Login Screen is used so the user can enter their username and password to login and access the app.Once the user presses the Login button on this screen they will be redirected to the User's Home Screen. This page also has a link to allow the user to create an account. The Create Account button/link will redirect the user to the Create Account Screen.

The Create Account Screen allows the user to create an account for the House Hawk application. The user will be able to enter in their first name, last name, home address, email address, password, desired password, home address, apartment number(optional), city, state and zip code. If all of the information is entered in correctly and there isn't a duplicate username our database a popup will appear that tells the user the account was created successfully. Upon closing the popup the user will be redirected to the Login Screen. If there is a similar username already in the database the user will be prompted to enter all of their information into the form again. There is also a back button that will redirect the user back to the Login Screen.

The User's Home Screen page will display buttons for the user to view all of the HouseHold Cameras screen, the Most Recent Exposure Screen, their Account Information and a button to sign out.

The Account Info Screen shows all of the user's information that they entered when signed up for the app. There is also a button to go back to the HomeScreen.

The Most Recent Exposures Screen Shows all of the images from the most recent exposure associated with a user. There is a button to refresh the screen since the screen can be accessed from pressing on the push notification. When a user clicks the push notification they will only see the images in the most recent exposure at that point in time. They would need to click the refresh button to see the images that were taken after the screen initially loaded. There is also a Reset Camera button that resets the camera associated to the most recent exposure.

The HouseHold Camera screen allows the user to view all of the cameras associated to their account. The cameras are seen as buttons and when a user clicks on one of the buttons it redirects the user to the PiExposures Screen. There is a back button to go back to the Home Screen and a camera button that when clicked redirects the user to the Add Camera Screen.

The Add Camera Screen allows a user to add a camera to their account. The user will be prompted to enter in the serial_id for their camera and the location of their camera. There is an Add Camera button that will add the camera to their account. There is also a back button that will redirect the user back to the HouseHold Camera Screen.

The Pi Exposures Screen shows all the exposures associated to a certain camera. Each exposure is marked via their TimeStamp. When a user clicks on an exposure they will be redirected to the Show Exposures Screen. There is also a back button that will redirect the user back to the HouseHold Camera Screen.

The Show Exposures Screen shows all of the pictures associated to a specific exposure. There is also a back button that will redirect the user back to the Pi Exposures Screen.

TABLE 4: List of Endpoints Used by Mobile Application

APP SCREEN	ENDPOINT	INPUT	OUTPUT	Functionality
	/users	---	*Json of all user information*	*Displays all users*
	/user	*User information*	*Json of user matching information provided*	*Displays a single user's information*
	/households	---	*Json of all household info*	*Displays all households*
	/household	*Household information*	*Json of household matching information provided*	*Displays a single household's information*
	/pis	---	*Json of all pi information*	*Displays all pis*
	/pi	*Pi information*	*Json of pi matching information provided*	*Displays a single pi's information*

	/exposures	*---*	*Json of all exposures info*	*Displays all exposures*
	/exposure	*Exposures information*	*Json of exposure matching information provided*	*Displays a single exposure's information*
	/exposure/pictures	*Exposure ID*	*Json of exposure matching information provided*	*Display all pictures associated with exposure*
	/exposure/picture get	*Image name, Pi ID, Exposure ID*	*Image file matching information provided*	*Grab a specific image from the server*
	/exposure/picture post	*Timestamp, filename, Pi ID, Exposure ID, image file*	*Json of new picture created using the information provided*	*Add an image to the server based on information provided*
	/user/login	*Email address*	*Json of user matching information provided*	*Display a matching user's information*
	/user/pis	*User ID*	*Json of Pis belonging to user*	*Display all pis associated with a specific user*
	/user/updated	*User ID, value*	*Json of user matching user ID provided*	*Determine if user has been updated*
	/user/household	*User ID*	*Json of household with matching user information*	*Display household associated with a user*
	/pi/exposures	*Pi ID*	*Json of all exposures belonging to a Pi*	*Display all exposures in a given Pi*
	/pi/reset	*Pi ID, value*	*Json of pi matching Pi ID provided*	*Determine if Pi should be reset*
	/pi/sn	*Pi Serial Number*	*Json of Pi matching SN*	*Determine the ID of the Pi matching the SN*
	/exposure/recent	*User ID*	*Json of most recent exposure*	*Display most recent exposure*

User Authentication and Data Security Issues

In order to help ensure that users are only authorized to view their personal data, users are required to log into their House Hawk account prior to using the application. Although the user is required to login, due to time constraints, no further user authentication was implemented. Additionally, all information (user account, images, etc.) located in the House Hawk system is currently available to the public.

If the House Hawk project was to be continued, it is recommended that hashing algorithms and other cyber security measures be placed. At a minimum a user authentication phase should be implemented as well as hashing of the password. Images files should also be encrypted based on either a SHA or AES type encryption algorithm using a unique generated password.

Tech Stack

The House Hawk mobile application was designed using a Flutter plugin for the Android Studio mobile development application, which allows it to be run on Android, iOS and Fuchsia. No external/third-party APIs calls are made by the program. However, the RESTful API developed for the application is based on Python Flask microframework utilizing Nginx and Gunicorn, regulated by Supervisor. The RESTful service is located on an Ubuntu EC2 instance hosted by Amazon Web Service. This server is also running a MySQL database to handle storage of most app related data. Images are stored directly on the server instance under a unique identifier for each user. Python scripts were developed for use in the Raspberry Pi system. The mobile application was programmed using Dart, which compiles into JavaScript.

Mid-Assessment

For the mid-assessment we would have liked to have the majority of the project nearing completion, including having all endpoints setup on the RESTful service, having both the Raspberry Pi and mobile application able to connect to and send information over the endpoints, and finishing the mobile application. Unfortunately due to lack of past experience with a lot of the programming languages used, and issues with specifically Flutter/Dart development continued up until the project demo. A list of tasks that each member worked on can be found below (**Table 5**).

Table 5: List of Tasks Assigned to Each Member

	TASK	**TEAM MEMBER**
1	Setting up web server	Matt Moore
2	Finish creating endpoints	Matt Moore, Ryan Lindsay
3	Connect RPi to appropriate endpoints	Ryan Lindsay
4	Finish Rpi development	Ryan Lindsay, Joshua DeNoble
4	Connect app to appropriate endpoints	Matt Moore, Ryan Lindsay, Andrew Freitas, Domenick Palmiotto
5	Finish mobile app development	Andrew Freitas, Domenick Palmiotto