Economic Home Security System

Design Document

Group 42

Client - Goce Trajcevski

Advisor - Goce Trajcevski

Kamini Saldanha - Full Stack Engineer

Andrew Tran - Full Stack Engineer

Lucas Jedlicka - Engineering Lead

Sohum Sawant - Security and Testing Lead

Uma Abu - Full Stack Engineer

Merin Mundt - Project Manager

Email: sdmay20-42@iastate.edu

Team Website:

http://sdmay20-42.sd.ece.iastate.edu/

Revised: 10/06

Executive Summary

Development Standards & Practices Used

All of our software development will be done using Agile Development.

When we begin development, we will work on requirements in two week sprints. Given that this is an agile environment, we will conduct daily standup on Slack.

Standup is 5 minutes for each member to talk about what they worked on yesterday, what they will work on today and whether they're blocked in their current task.

Summary of Requirements

- Camera streaming
- Motion detection
- Push notification to user if threat was detected
- Continuous streaming until threat has left the frame
- Store clips when stream has ended.

Applicable Courses from Iowa State University Curriculum

Computer Science 309, Software Engineering 319, Computer Science 311, Computer Science 228, Computer Science 227

New Skills/Knowledge acquired that was not taught in courses

Designing web applications using a Python Django based tech stack.

Machine Learning libraries to detect threats.

Client and remote motion detection of video feeds.

Docker based development and deployment.

Authentication Tokens for security purposes.

We all have to learn Python to develop the backend.

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Definitions (This should be the similar to the project plan)

1 Introduction

1.1 Acknowledgement

Thank you to Goce Trajcevksi for serving as the client and advisor for this project. We acknowledge the time you are taking to work with us and resources you are providing us with to ensure the success of this project.

1.2 Problem and Project Statement

Given how common theft is in this day and age, we have set out the goal to build an economic home security system from recycled smartphones. Our security solution will give users the opportunity to disconnect from the cloud and avoid the monthly fees that most security systems charge with the use of RTSP (Real-time Streaming Protocol) which will allow for the extension and adoption of inplace security systems. We plan to target users who already have IP cameras or a potential excess of old smartphones. We are aware that a lot of times, the security system may pick up on movement that is not harmful to the house owner or user. For example, when the mailman is walking to the front door to drop off the mail in the mailbox, the security system will pick up on this movement. And the user could possibly not care about this movement that has been captured. So they could delete the clip from storage and move on with their day. A use case diagram on this possible scenario has been included below in section 1.5.

1.3 Operational Environment

The intended operational environment for the product is a home with areas that have reliable power and is safe from the elements (i.e. water or extreme temperatures). The supported phones are intended to be used 24/7 with the web application running. Existing IP cameras will have the ability to be used in conjunction to all cameras the user sets up and viewable from the web application. A network connection is required.

1.4 Requirements

Functional Requirements:

FR.1: Phones will locally detect motion. Upon detecting motion, the phone will begin streaming to the backend for processing.

Given the smartphone is an Android 4.4 or iOS 11 and above, it will support motion detection in the client. To detect motion, the client will look for a change in pixels in its environment which could mean potential movement.

Accomplishing motion detection in the client will enable us to have an economic use of bandwidth.

FR.2: IP cameras supporting RTSP will continuously stream to the backend for processing.

IP cameras do not have the capability to detect motion in the client. To combat this, all streams will be sent to the backend to be processed for motion detection.

FR. 3: Continuous streaming until the backend sends a kill signal.

Once motion has been detected in the client, the client will begin streaming to the backend. The backend will establish whether this movement is an object of interest. Once the object of interest has left the frame for a period of time, the backend will send a kill signal to the client to stop streaming.

FR. 4: Send a push notification if human was detected

As soon as we get 1 valuable frame of interest of object detection, the user will be notified.

FR. 5: Encourage user to delete clip.

A clip is a completed event stream that's stored as a video file. By asking the user if a clip can be deleted, we can save storage space.

FR 6: Store clip when unseen by user and stream ends.

We are storing all clips for the user. While we encourage the user to delete clips, it is valuable to keep them in case the user might want them later.

***Non-functional Requirements*:**

NFR 1: Support a sub one second response time.

We need our solution to be fast in case of threatening events. As soon as motion is detected our users need to be notified.

NFR 2: Support more than three streams.

This scalability requirement will allow our users to monitor a larger area, further securing their residences.

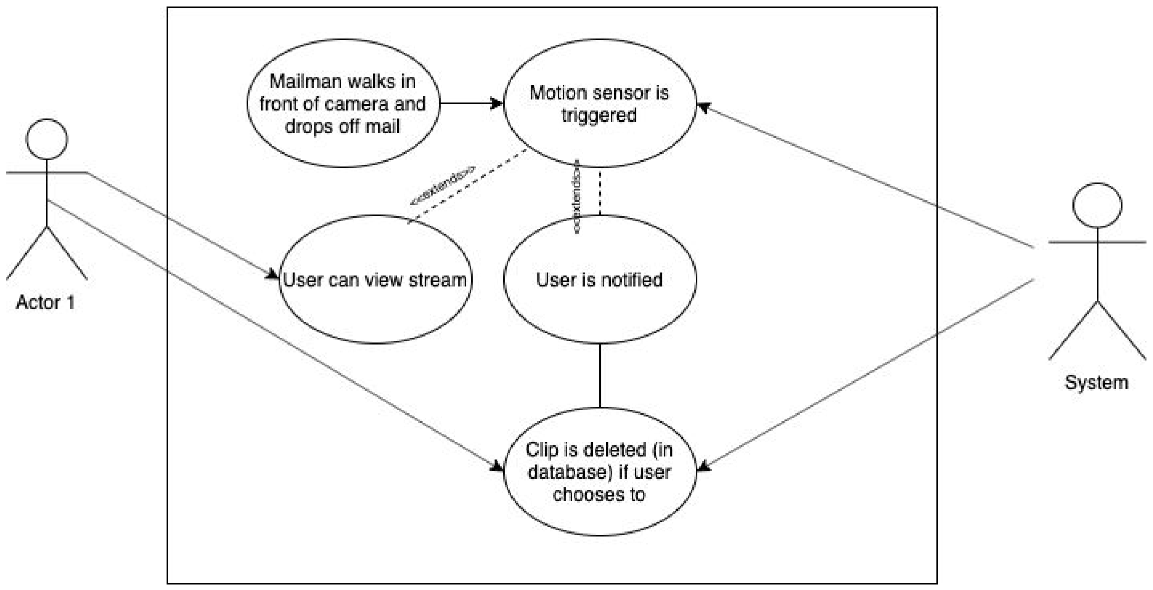
NFR 3: Support smartphone feeds from 11 and above and android 4.4 and above.

Get User Media API is not supported for operating systems below those versions. The Get User Media API is our way of being able to access video streams.

1.5 Intended Users and Uses

Actor 1. Homeowner in need of a security system with access to unused smartphones or IP cameras.

Actor 2. Homeowner with an existing security system that supports RTSP.


Mailman walks in front of camera and drops off mail
Motion sensor is triggered
<<extends>>
<<extends>>
User can view stream
User is notified
Clip is deleted (in database) if user chooses to
Actor 1
System

1.6 Assumptions and Limitations

- **Potential for low performance in old and cheap smartphones**
  - Given that users will be using recycled smartphones, there is no quality assurance for how well the smartphone will work with our security solution.
- **Camera limitations in low light and quality**
  - Smartphone cameras are not designed to work as well at night. This will mean low visibility in low light.
  - The quality of the camera will lower with the older the phone is.
- **Streaming at a bit rate which can be processed in a timely manner**
  - If the bitrate on streams is too high, the processing time will take too long. However, if the bitrate is too low we cannot ensure optimal motion detection. In order to achieve optimal performance, we need to find the correct medium.
- **Lack of time and learning curve**
  - Some members are not proficient in all the technologies selected. For example, the entire team is not proficient in Python.
  - Additionally, we have the additional time constraint of 1 year to finish this project.
- **Server**
  - The server may not be able to handle the amount of load coming in from constant streaming. The client will only start streaming if motion is detected in smartphones. However, IP cameras will constantly be streaming constantly because they do not support motion detection in the client.
- **Storage issues**
  - We preferably do not want to save all clips because this will take up a lot of space. To mediate this, we will encourage the user to delete clips.

1.7 Expected End Product and Deliverables

The expected delivery dates, end product, and deliverables are subject to change.

Deliverables:

V1 Design Document (October 6, 2019)

V2 Design Document (November 3, 2019)

V3 Design Document (December 13th, 2019)

Setup database (January 23rd, 2020)

Setup server (February 17th, 2020)

Backend - stream processing and motion detection (February 17th, 2020)

Test all components (February 24th, 2020)
API - for backend and frontend communication (March 11th, 2020)

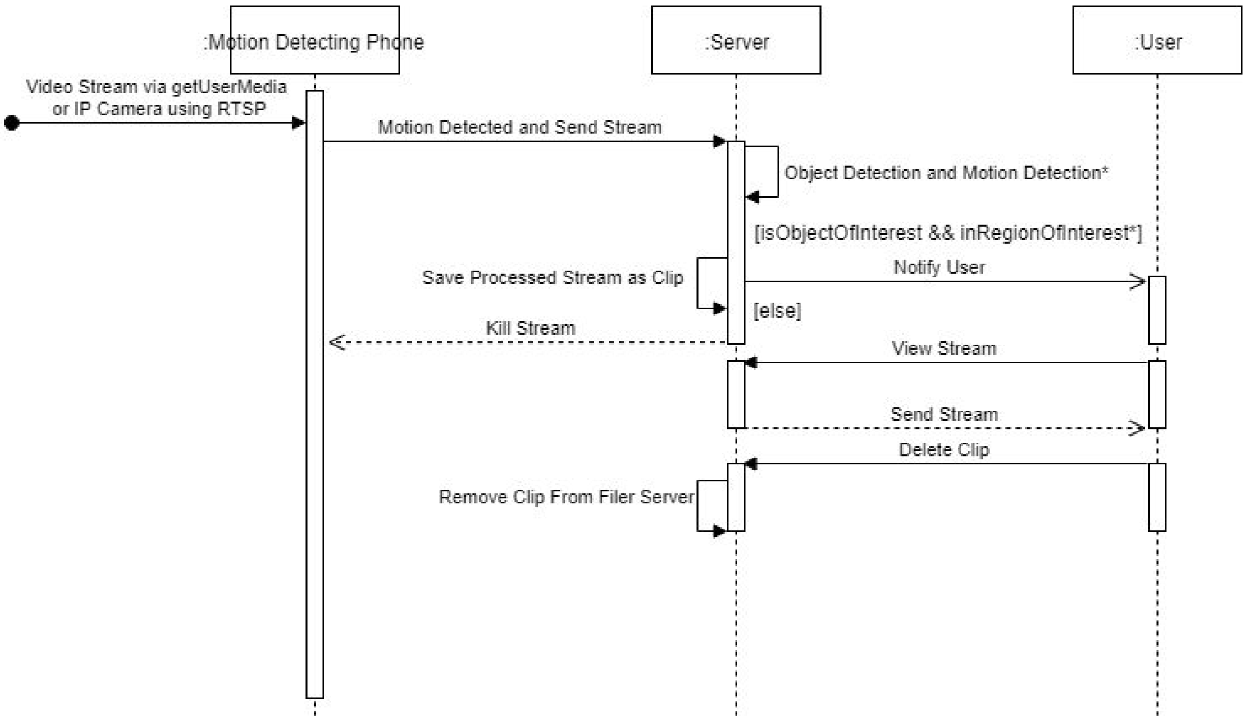
Test communication between backend and frontend (March 18th, 2020)

Scaling solution (April 3rd, 2020)

Final testing (April 17th, 2020)

2. Specifications and Analysis

2.1 Proposed Design


:Motion Detecting Phone
:Server
:User
Video Stream via getUserMedia or IP Camera using RTSP
Motion Detected and Send Stream
Object Detection and Motion Detection*
[isObjectOfInterest && inRegionOfInterest*]
Notify User
Save Processed Stream as Clip
[else]
Kill Stream
View Stream
Send Stream
Delete Clip
Remove Clip From Filer Server

2.1.1 Camera network

The camera network will be primarily populated via a web application. Users will access the web application via a web browser, then sign into their account. Once signed in, a page will be available to add the device-in-hand to their network of streaming cameras. Alternatively they can add an IP camera via a similar interface where a device IP, username, and password are entered. Once the phone has been added it will begin performing local motion detection, and communicating over a web socket. The phone will send its footage over a websocket. Communications will be more strictly defined in section 2.1.2.

2.1.2 Processing software

The processing software will refer to its database to find list of devices. Connections will be established with all IP cameras and phones. However phones won't stream video until motion is detected, or a user requests to view the stream. A connection will be maintained via a web socket for bi-directional communications without polling. The communications include a kill signal for streaming once the server determines the footage isn't of interest. The processing server will heavily utilize threads to enable concurrent low latency streams processing. For IP cameras, the motion detection will take place on the server. Object detection will be used to classify threats. Objects of interest would be cars and people, and everything else is not of interest. To reduce the possibility for false positives a region of interest will also be defined per device by the user. For example a tree in frame would constantly be causing motion in the wind. Motion detection results outside the region of interest will be ignored. Upon detecting an object of interest in a region of interest a notification will be generated, and the user will be able to view that stream by tapping the link.

2.1.3 web application

The web application is used to send streams, view streams, view clips, add devices, define regions of interest, and receive notifications.

2.2 Design Analysis

Python is the backend programming language of choice. Python has several machine learning libraries that would help with tasks such as human detection. Given that the motion detection would be implemented with Python, the API to communicate between the server and client will also be written in Python.

Given that we do not have much programming experience with Python, there will be a steep learning curve for the team. While this is not ideal, we strongly believe that it is the right choice given the functionality it can provide to this project.

In terms of the client, React.js is the primary language of choice. React.js will enable us to create an elegant, user-friendly frontend. Multiple members of the team are also experienced in this technology.

We have also been working on implementing user access tokens for security purposes. We are moving forward with using two tokens, a refresh token and an access token. The access token will grant access to the user to any internal services. This token can only be used once and access is taken away immediately after usage. The refresh token is used to request another access token. Request tokens are kept secure by storing them in a secure file system or database. They should only be used when requesting an access token from our server side. This means that we have to also implement security standards on our backend and also in our database.

2.3 Development Process

We will be following Agile software development. The following are reasons for choosing an Agile environment.

- Client engagement and collaboration.
- Quick deliverables.
- Flexibility.
- Focus on the user.

2.4 Design Plan

The security system utilizes a camera network consisting of IP cameras and compatible smartphones (Android 4.4+ and iOS 11+). Our smartphone selection is restricted due to browser implementations of the getUserMedia API used to access the cameras. The backend, which has a tech stack consisting of Python with Django framework, will be used to move the streams. FFMPEG is used for video transcoding and capturing frames. YoLo object detection is used for placing bounding boxes on humans or cars. The web application will serve as a view to support two primary functions, linking a smartphone or IP camera to the security network and viewing the live streams or saved clips. ReactJS is the library of choice for developing the front end web application.

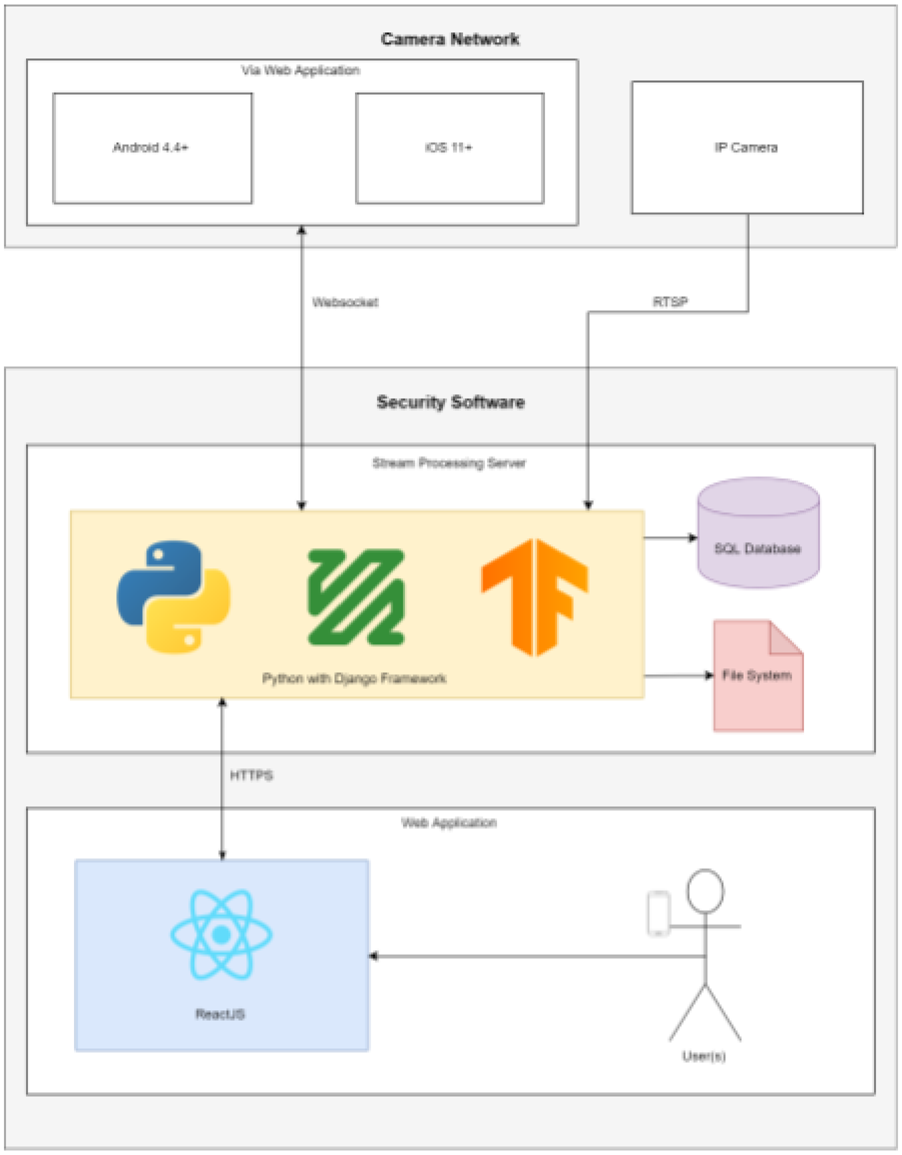


Figure 1. Design Diagram

3. Statement of Work

3.1 Previous Work And Literature

Dube, Ryan. "How to Build a Security Camera Network Out Of Old Smartphones." MakeUseOf, 12 July 2017, https://www.makeuseof.com/tag/how-to-build-a-security-camera-network-out-of-old-smartphones/.

This article describes how to build a security camera network out of old smartphones. This is the same problem statement as our senior design project. This author talks about information such as placing the smartphone, setting up the camera IP monitoring station, features of his security camera network and how to create a surveillance system.

Chen, et al. "An Implementation of Faster RCNN with Study for Region Sampling." ArXiv.org, 8 Feb. 2017, https://arxiv.org/abs/1702.02138.

This is a paper written by two Computer Science professors under the topic of Computer Vision and Pattern Recognition. In this paper, the authors talk about how they adapted the join-training scheme of Faster RCNN framework from Caffe to TensorFlow as a baseline implementation of object detection. We are also planning to also use TensorFlow as a programming language in object detection.

Similar Smartphone Security Application "AlfredCamera." AlfredCamera, https://alfred.camera/.

AlfredCamera is a home security smartphone security application that is already on the market that has the goal of making security free and easy for everyone. To use AlfredCamera, you need to download the application on two smartphones, sign into your gmail account on both smartphones, set one camera as the viewer and the other as the camera to enjoy the application.

3.2 Technology Considerations

- Local solution vs cloud based solution
  - Our project is intended to be affordable and easy to implement for users. If a cloud solution had been selected, the users would have to pay a monthly fee to use features such as cloud storage and servers.
  - Our goal is for this to be an economic solution to a problem that has been addressed many times. We wanted to create a solution that could be used by simply owning a computer, some old IP cameras and smartphones that aren't being used anymore.
  - To enable this local solution, we are using a teammates server that can handle the load required for this project. The server will be free and the electricity needed to run it will also be free for this use case.
- Python
  - We want to utilize all of the machine learning libraries in Python that would be useful for our project.
- Asking users to delete clips
  - We will be saving all clips. To cut down on space, we will encourage the user to delete the clip.

3.3 Task Decomposition

In order to solve the problem at hand, it helps to decompose it into multiple tasks and to understand interdependence among tasks.

3.4 Possible Risks And Risk Management

Technical Risks

TR1. Hardware is insufficient for video processing.
TR2. Limited technical experience or knowledge of the technologies
TR3. Technology used is no longer supported, interoperable, or scalable.
TR4. Unknown security fault leaves project vulnerable.

Cost Risks

CR1. Prices for tools, hardware, or technologies suddenly increase.
CR2. Server has issues or breaks down.
CR3. Inaccurate estimation of cost and budget for the project.

Scheduling Risks

SR1. Inaccurate estimation and scheduling of development time.
SR2. Requirements or features are added to the project.
SR3. Team member is unable to complete tasks on schedule.

Risk	Response Strategy
TR1.	Replacing hardware with something more capable or trying different processing or compression combinations.
TR2.	Allocate time and resources to seek for required knowledge.
TR3.	Stop work on anything related to the issue and seek for a replacement.
TR4.	Stop work on anything related to the issue and seek for options to patch the security flaw.
CR1.	Make adjustments to design or increase budget to meet requirements.
CR2.	Troubleshoot the server and revert to a working backup server. Ask ETG for replacement parts or machine if hardware related.
CR3.	Seek for free or cheaper alternatives and make changes to the design to match.
SR1.	Reduce testing or features to meet requirements and resource constraints.
SR2.	Define and prioritize requirements. Reduce testing or features to meet requirements and resource constraints. Schedule extra time in advance for this situation.
SR3.	Each team member should be aware and knowledgeable of each other’s work and be able to work together to pick up the task.

3.5 Project Proposed Milestones and Evaluation Criteria

M1. Conceptual Design
Completed and approved design document. Project architecture, plan, and requirements are well defined. Non-requirements and features not considered are addressed with reasoning.
M2. Simple End-to-End Scenario Prototype
The main components for the scenario are complete, reviewed, and pass all tests. Web application is able to send the device’s camera stream to the server, detect motion in live camera feed, notify user when camera detects something of interest in view, view user’s clips the server saved, delete user’s clips from server, and view the live feed from the camera. REST API is able to communicate with the client and server. Server is able to receive streams, do object detection on frames, transmit to client if frames contain something of

interest, send signal to client to stop or continue streaming to server, and contain a database storing user information and clips. Basic encryption and authentication should be implemented.

M3. Prototype With Multiple Motion Detecting Phones

The main components for this scenario are complete, reviewed, and pass all tests. Web application is able to send camera streams from multiple devices to the server, detect motion in live camera feed for each, sort user's clips, and view the live feed from any of the user's cameras.

M4. Prototype With Motion Detecting Phones and Non-motion Detecting Phones and IP Cameras

The main components for this scenario are complete, reviewed, and pass all tests.

M5. Testing

Reserved for future features and final design. Individual systems at this point is working without issue alone.

M6. Final Product

All documentation, reports, and code are complete, reviewed, and tested. The product meets all requirements and performs as expected with no bugs and performance issues. Security is considered at all points.

3.6 Project Tracking Procedures

Issues will be created from requirements and put on GitLab's backlog. Issues with highest priority are marked with the "To-do" tag. In-progress issues are marked with a "Doing" tag so others will know tasks being worked on and contribute if deemed necessary. Once complete, the issue is marked with a "Done" tag.

Weekly reports will also be done which include a work summary, accomplishments, pending issues, individual contributions, and future plans.

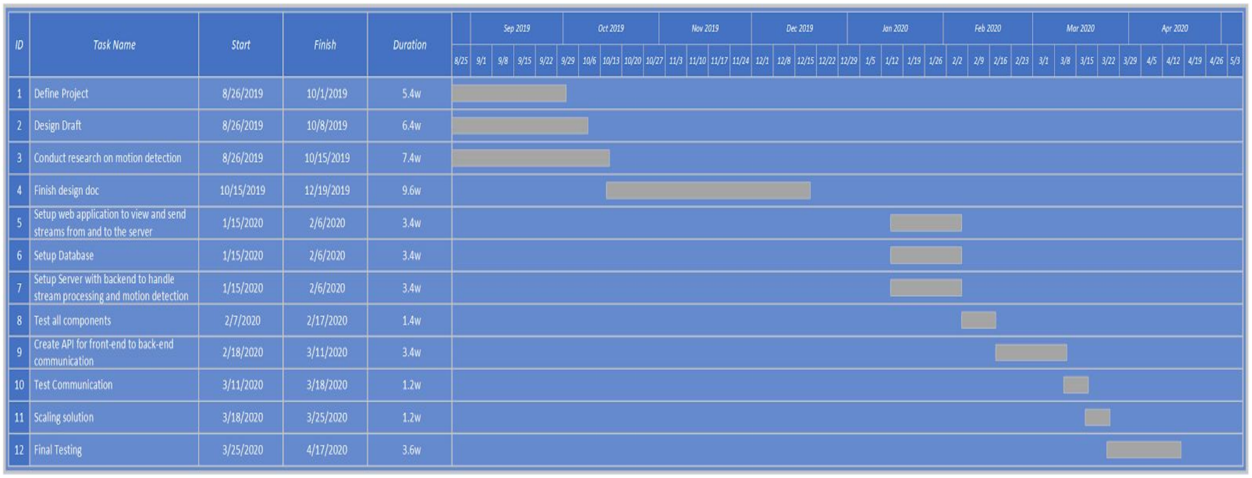
3.7 Expected Results and Validation

The desired outcome for this project is to:

- Create an economic home security solution that is easy to use.
- Create an economic home security solution that does not cost much to implement.
- All functional and non-functional requirements need to be met.
- The end result is expected to be bug free and solid, performing as expected as a security system.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline



ID	Task Name	Start	Finish	Duration
1	Define Project	8/26/2019	10/1/2019	5.4w
2	Design Draft	8/26/2019	10/8/2019	6.4w
3	Conduct research on motion detection	8/26/2019	10/15/2019	7.4w
4	Finish design doc	10/15/2019	12/19/2019	9.6w
5	Setup web application to view and send streams from and to the server	1/15/2020	2/6/2020	3.4w
6	Setup Database	1/15/2020	2/6/2020	3.4w
7	Setup Server with backend to handle stream processing and motion detection	1/15/2020	2/6/2020	3.4w
8	Test all components	2/7/2020	2/17/2020	1.4w
9	Create API for front-end to back-end communication	2/18/2020	3/11/2020	3.4w
10	Test Communication	3/11/2020	3/18/2020	1.2w
11	Scaling solution	3/18/2020	3/25/2020	1.2w
12	Final Testing	3/25/2020	4/17/2020	3.6w

We are proposing a fast timeline in order to leave time for any unforeseen circumstances or issues that may arise during the development of our project.

The first semester of our project will pertain of designing and going through use case scenarios. The second semester is as follows. We start with setting up the major components of our project. Those will be setting up our web application, database, and server. We have assigned subteams to handle these components. Server side backend will be handled by Andrew Tran, Kamini Saldhana, and Sohum Sawant. The Web Application will be developed by Uma Abu and Merin Mundt. The processing and database will be handled by Lucas Jedlicka. After this we will test all components to ensure that they are up to standard. Following the development and testing of our components we will begin working on communication between all of them. We will finish our project by scaling the solution to support more streams and then conducting final testing on the project.

4.2 Feasibility Assessment

Realistically, scaling our solution to support multiple camera streams or cloud technology will be a challenge. The hardware that we have can only support three streams. Without acquiring better hardware we cannot test to see if adding more streams will cause any problems. The same applies to possible cloud solutions. We do not have the capability to test if our solution will work for more and more streams.

4.3 Personnel Effort Requirements

The first tasks for the implementation of our project will be for everyone on the team to get their development environment set up. This will involve install/cloning the git repo, install the libraries and frameworks needed for development and also familiarizing ourselves with the libraries and frameworks. We expect that these tasks will not take too long and so we will allocate a few hours for them. The next tasks will be to set up the various components needed and possible host them. The components include the database to store our streams, the server processing the streams and the web application that will be sending the streams to the server. This will be a decent amount of work and so we will budget for at least 3 week for this task including testing. The tasks will be broken down into smaller parts and split between the members of the team. The next major task will be to create communication between the 3 major components. This will involve setting up API's for the web application to communicate with the server and for the server to send data back to the web application in the form of notification after it has done its processing. This will also include sending data from server to the database and having the web application retrieve data from the database. This will be a decent amount of work and so we are budgeting for about 3 weeks on the tasks. Like before, they will be split into smaller components and assigned to team members. Finally we plan to scale our application by introducing more smartphones for one user. This will include scaling the server up to meet demands and handing motion detection on the smartphones before sending the data to the server. This will take another 2 weeks. Lastly, we will do major testing of the whole application focusing on functionality and security. This will take a week.

Task 1 : Environment set up	3 hrs
Task 2: Set up components	3 weeks
Task 3: Communication between components	2 weeks
Task 4: Scaling	2 weeks
Task 5: Testing and Security	1 week

4.4 Other Resource Requirements

We will require a server as well as mobile devices to test our home security system. Both of these are provided by the team at no cost.

4.5 Financial Requirements

We have a total budget of $200 to spend on any project costs for the duration of 1 year given to us by our clients.

5. Testing and Implementation

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or a software library

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests (e.g. unit testing for modules, integrity testing for interfaces, user-study for functional and non-functional requirements)
2. Define the individual items to be tested
3. Define, design, and develop the actual test cases
4. Determine the anticipated test results for each test case

5. Perform the actual tests
6. Evaluate the actual test results
7. Make the necessary changes to the product being tested 8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you've determined.

5.1 Interface Specifications

– Discuss any hardware/software interfacing that you are working on for testing your project

We will support any mobile device as a client that features the following softwares and above - Android 4.4 and iOS 11. To ensure a successful product, we have to test on the following softwares and any versions released after.

5.2 Hardware and software

– Indicate any hardware and/or software used in the testing phase

– Provide brief, simple introductions for each to explain the usefulness of each

We will use a mockito testing framework to test our API. This will be useful as mockito allows you to make service calls and create instances of different services. This we can ensure that data is transmitting properly.

5.3 Functional Testing

Test to see that Users can be added to the database.

Test to see that Motion Detection notifications work.

Test to see User authentication works.

Unit testing on all modular functions we develop on frontend and backend.

Test our server to see if it can process multiple streams.

Test to see our bitrate on streams matches requirements.

5.4 Non-Functional Testing

Test to see if notifications to user are fast.

Test to see if Users can access streams in a quick time(5 seconds)

Test front end facing applications responsiveness.

5.5 Process

– Explain how each method indicated in Section 2 was tested

– Flow diagram of the process if applicable (should be for most projects)

5.6 Results

– List and explain any and all results obtained so far during the testing phase

- – Include failures and successes
- – Explain what you learned and how you are planning to change it as you progress with your project
- – If you are including figures, please include captions and cite it in the text

• This part will likely need to be refined in your 492 semester where the majority of the implementation and testing work will take place

-**Modeling and Simulation**: This could be logic analyzation, waveform outputs, block testing. 3D model renders, modeling graphs.

-List the **implementation Issues and Challenges**.

6. Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

6.2 References

This will likely be different than in project plan, since these will be technical references versus related work / market survey references. Do professional citation style(ex. IEEE).

6.3 Appendices

Not yet applicable