Economic Home Security System

Design Document

Group 42

Goce Trajcevski:	Client and Advisor
Uma Abu:	Full-stack Engineer
Lucas Jedlicka:	Engineering Lead
Merin Mundt:	Project Manager
Kamini Saldanha:	Full-stack Engineer
Sohum Sawant:	Security and Testing Lead
Andrew Tran:	Full-stack Engineer

Email: sdmay20-42@iastate.edu
Website: sdmay20-42.sd.ece.iastate.edu

Executive Summary

Development Standards & Practices Used

This semester, we have adopted a waterfall practice for the requirements gathering and research of this project. Next semester, we will adopt an agile test-driven development practice. Given that this is an agile environment, we will conduct daily standup on Slack. Standup is a 5 minute period where each member talks about what they worked on yesterday, what they will work on today, and whether they are blocked on their current task. Additionally, we will work in two week sprints where each team member will be assigned several tasks for that period.

Summary of Requirements

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

Applicable Courses from Iowa State University Curriculum

Computer Science 227 - Object-oriented Programming
Computer Science 228 - Introduction to Data Structures
Computer Science 309 - Software Development Practices
Computer Science 311 - Introduction to the Design and Analysis of Algorithms
Software Engineering 319 - Construction of User Interfaces
Software Engineering 329 - Software Project Management
Software Engineering 339 - Software Architecture and Design

New Skills/Knowledge acquired that was not taught in courses

Designing web applications using a Python/Django and React 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.
Learning Python to development the backend.

Table of Contents

List of Figures

List of Tables

List of Definitions

Event - Something entering the frame
RTSP - Real-time streaming protocol
getUserMediaAPI - Web API for accessing device cameras

1. Introduction

1.1 Acknowledgement

Thank you to Goce Trajcevski for serving as the client and advisor for this project. We acknowledge the time he is taking to work with us and resources he is 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 which will allow for the extension and adoption of inplace security systems. We plan to target users who already have IP cameras or an 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. The user might not care about this movement that has been captured. So, there is an option for the user to delete the clip from storage and move on with their day. A use case diagram for this possible scenario has been included in section 1.5 of this document.

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-hours a day with the web application running. Existing IP cameras will have the ability to be used in conjunction with all cameras the user sets up and viewable from the web application. A network connection is also required.

1.4 Requirements

1.4.1 Functional Requirements

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

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 one valuable frame of interest of object detection, the user will be notified via a push notification.

FR. 5: Encourage user to delete clip.

A clip is a completed event stream that is 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 will be 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.

FR 7: Perform object detection on frames.

Detected objects will be matched to an object of interest list such as humans, then the event will be deemed worthy of notification.

FR 8: Motion detection region filters.

Allows the blacklisting of common false positive areas, such as a tree in frame constantly moving.

1.4.2 Non-functional Requirements

NFR 1: Support a sub three 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: Our system will be accessible from any modern web browser e.g. Android, iOS, Raspberry Pi, laptop, etc.

We want users to be able to have a wide accessibility of our application.

1.5 Intended Users and Uses

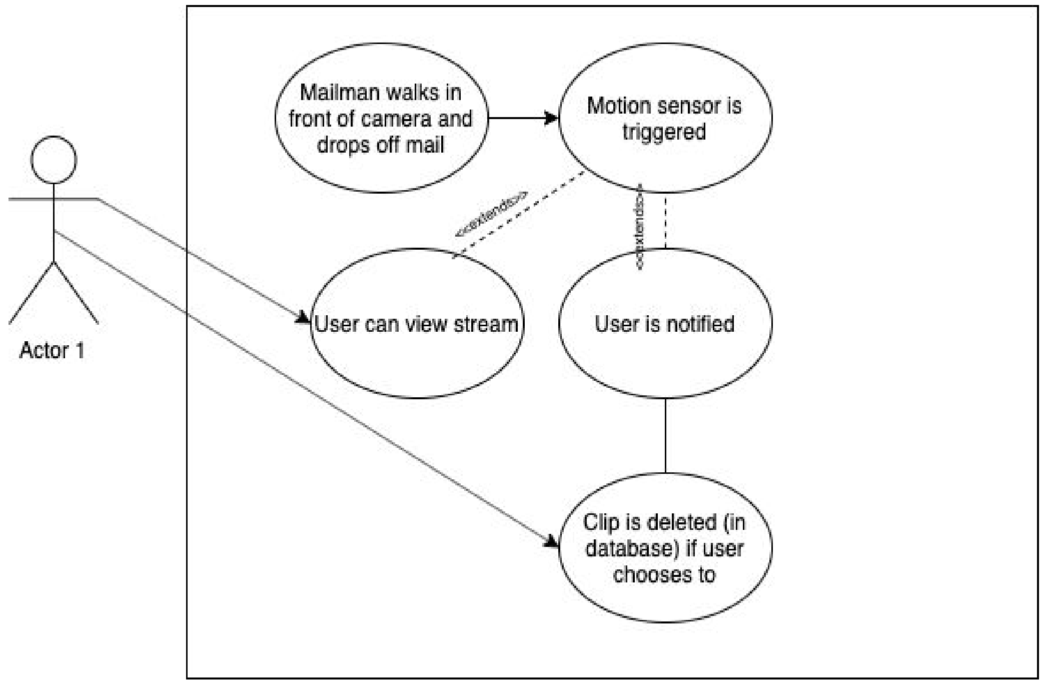

Mailman walks in front of camera and drops off mail
Motion sensor is triggered
<<extends>>
<<extends>>
User can view stream
User is notified
Actor 1
Clip is deleted (in database) if user chooses to

Figure 1. Use Case Diagram

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.

1.6 Assumptions and Limitations

1.6.1 Assumptions

- 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.
- 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.

1.6.2 Limitations

- 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.
- 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)

Components completed and tested (Feb 17)

Integration completed and tested (March 18)

System passing acceptance testing (April 17)

2. Specifications and Analysis

2.1 Proposed Design

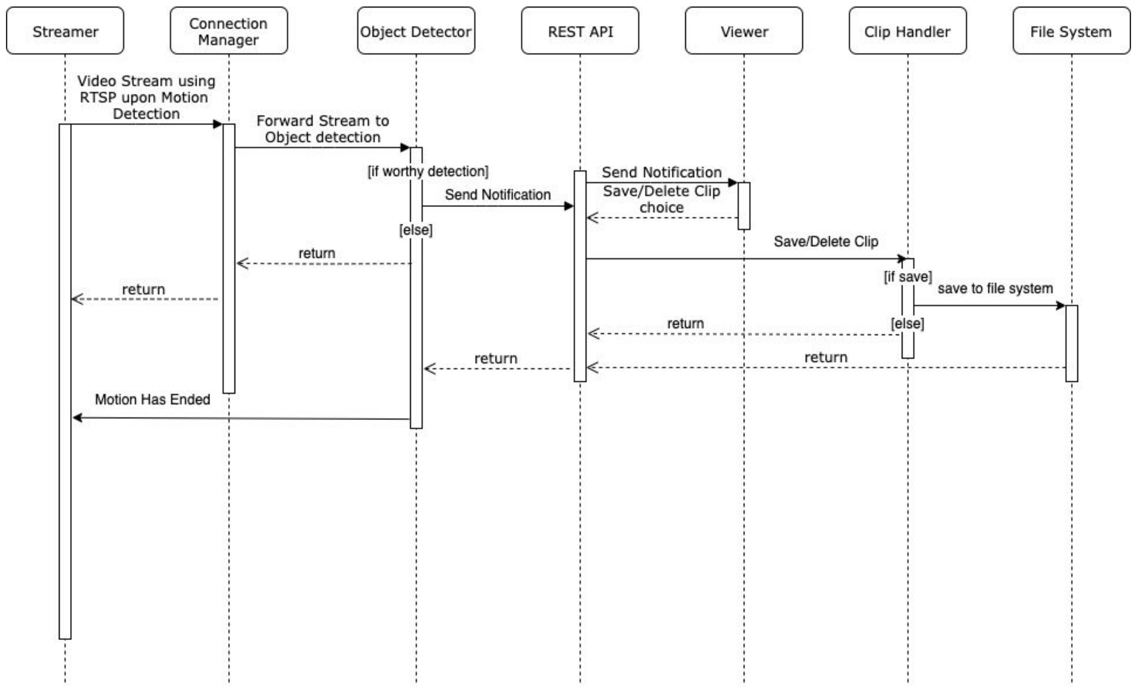


Figure 2. Sequence Diagram for Typical Scenario

2.1.1 Camera Network

The camera network will be primarily populated via a web application. Users will access the web application via a modern web browser and 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 device 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 smartphones. However smartphones 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. For example, objects of interest would include humans. There will be a list of objects of interest to limit the number of threats to look for this . 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. It serves as a client for the user to interact with and to communicate with the server.

2.2 Design Analysis

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

Given that we do not have much programming experience with Django, there will likely 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 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.

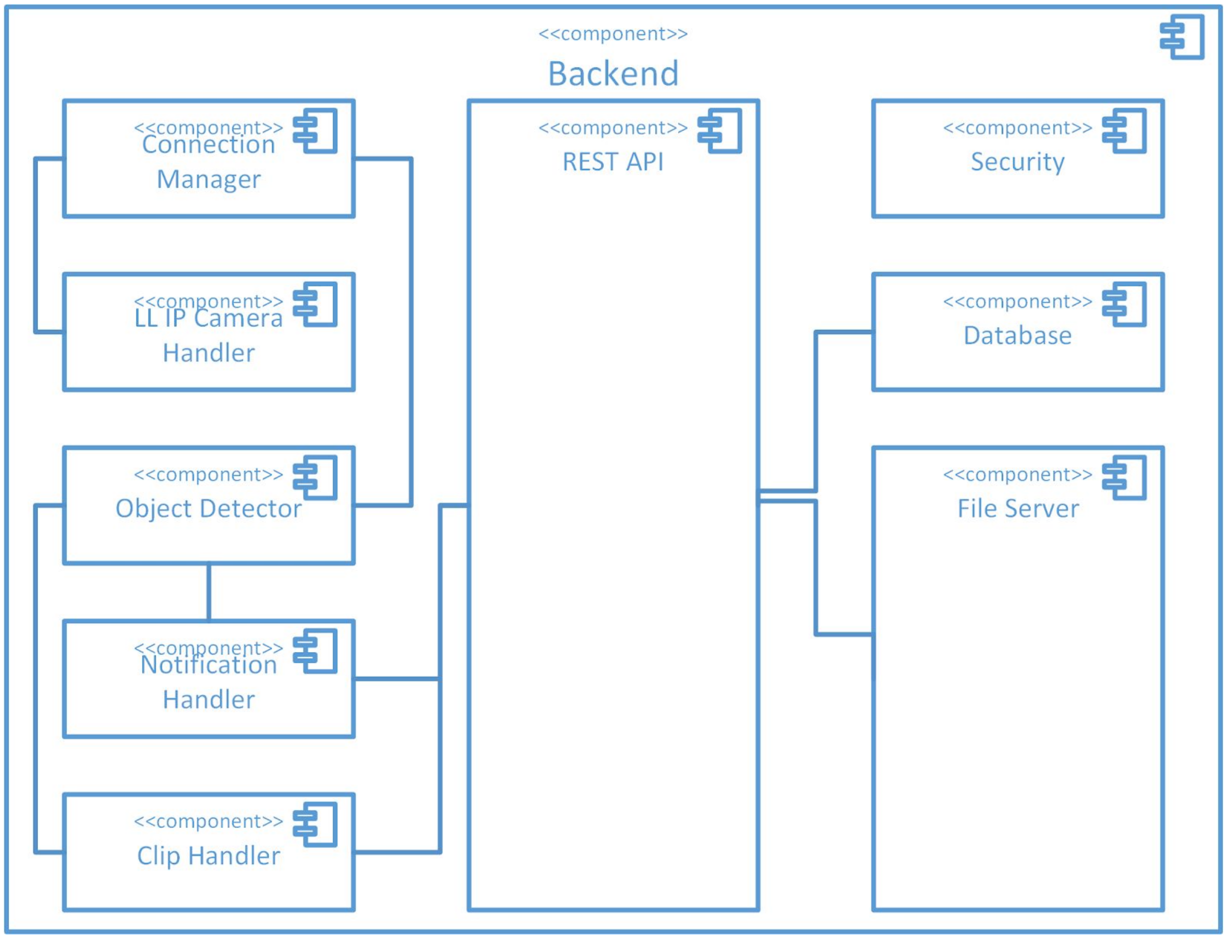


Figure 3. Backend Component Diagram

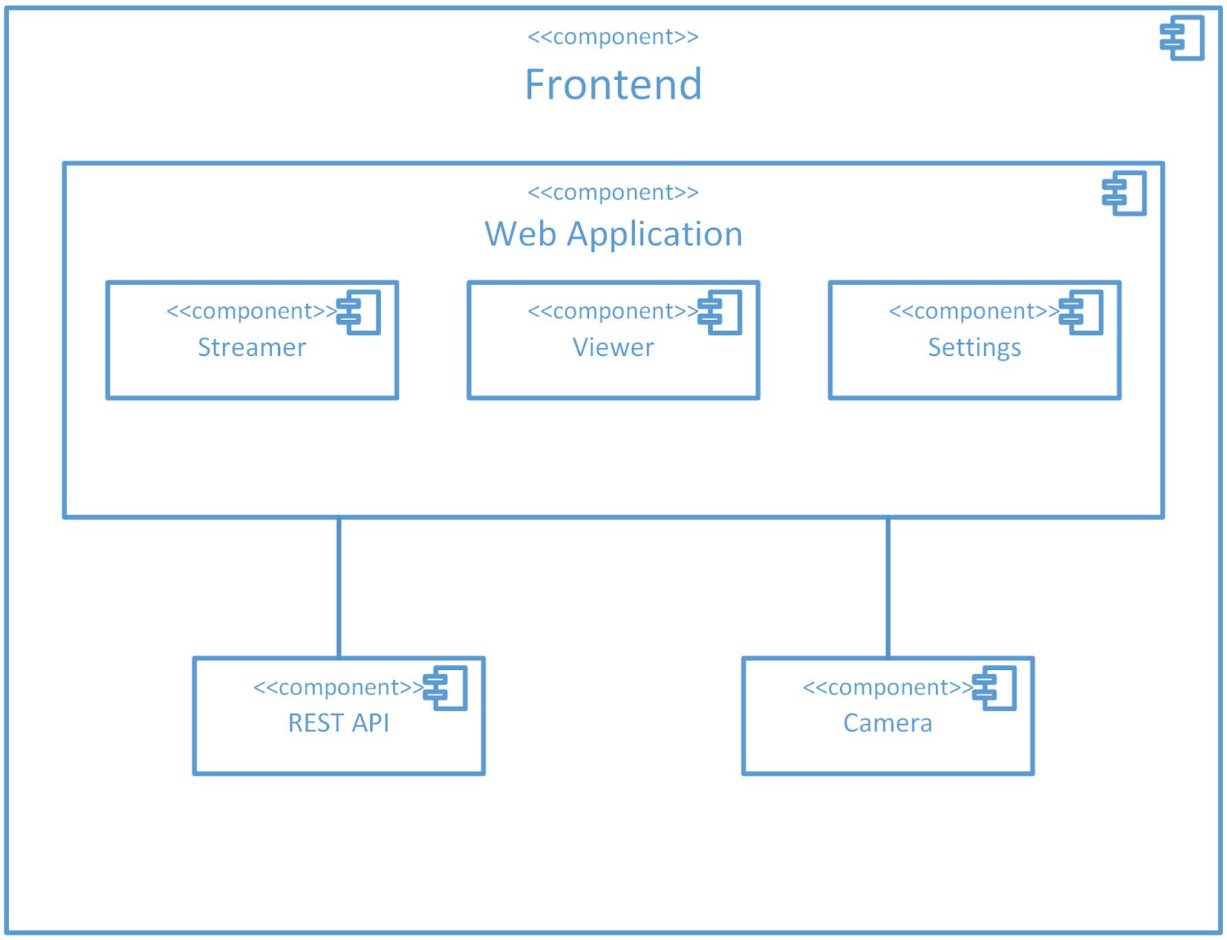


Figure 4. Backend Component Diagram

2.3 Development Process

In terms of the software development, as mentioned under the executive summary, we will be following an agile development practice. 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.

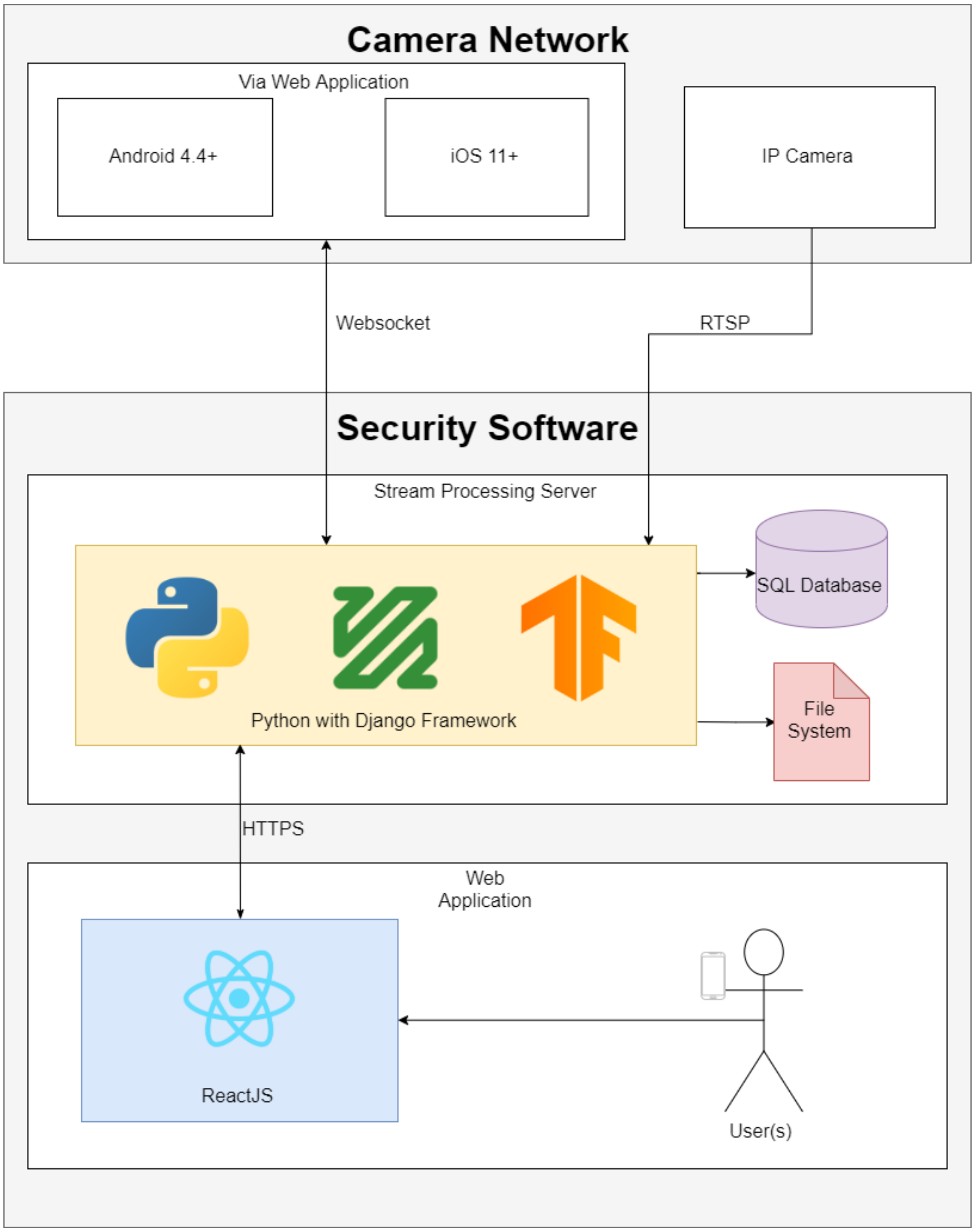


Figure 5. Design Diagram

3. Statement of Work

3.1 Previous Work and Literature

This article describes how to build a security camera network out of old smartphones [1]. This article describes the same problem statement as the one our senior design project does. 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.

This is a paper written by two Computer Science professors under the topic of Computer Vision and Pattern Recognition [2]. 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.

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 [3]. 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
    - 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.
    - Aiming for 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.
    - Empower users to own their own data given the number of recent security breaches.
    - Will be using a teammates 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 Django
    - Utilize all of the machine learning libraries in Django that would be useful for our project.

    - Encourages rapid development and clean, pragmatic design
    - Some of the busiest sites on the Web leverage Django's ability to quickly and flexibly scale.
- 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

Frontend

- Web Application
    - Ask user to delete clip (Merin)
    - Motion detection (Uma)
    - Streaming (Merin)
    - Push Notifications (Uma)
    - Mobile and desktop capable UI (Merin)

Backend

- Connection Manager (Kamini and Sohum)
    - Websocket For Smartphone
    - Manage Connection (kill signal, stream forwarding, etc.)
- Low Level IP Camera (Kamini and Andrew)
    - Motion Detection
    - Region Filtering
- Object Detector (Lucas)
    - Find Object of Interest
    - Filter Objects
- Notifications Handler (Sohum and Andrew)
    - Push or Email Notifications
- Clip Handler (Kamini and Andrew)
    - Save, Retrieve, and Delete Clip From Storage
- REST API (Kamini, Andrew, and Sohum)
    - Establish Connection Between Components
    - Create Endpoints
- Security (Sohum)
    - Implement Tokens and Encryption
    - Determine and Solve Security Issues
- Database (Lucas)
    - Set Up
- File Server (Lucas)
    - Set Up

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.

Table 1. Risk Response Strategies

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.

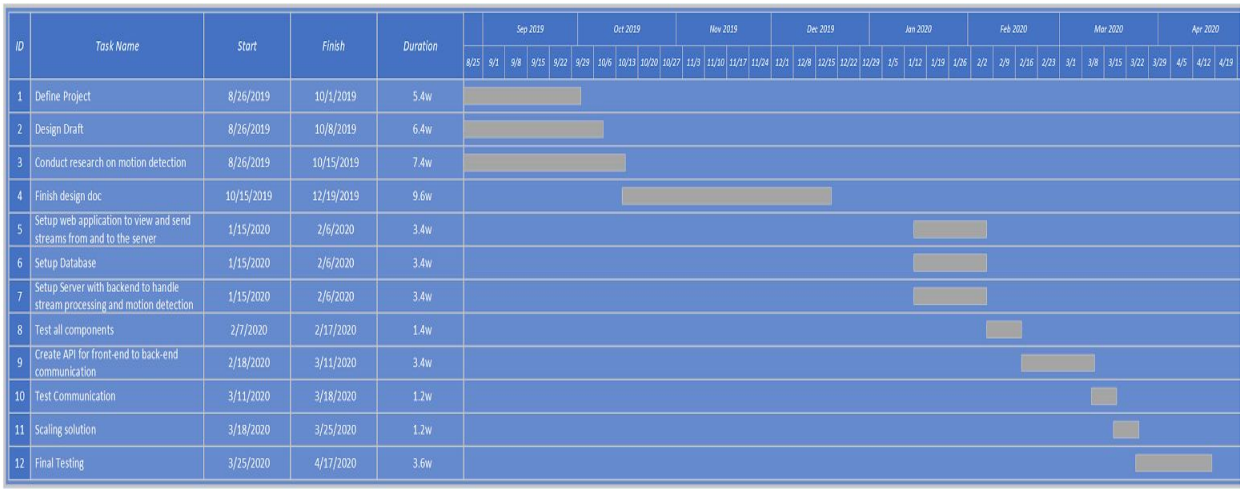
3.7 Expected Results and Validation

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

Figure 6. Timeline

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 three weeks 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 three 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 three 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.

Table 2. Task Time Estimations

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

4.4 Other Resource Requirements

We will require a server, mobile devices, and IP Cameras to test our home security system. 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

5.1 Interface Specifications

We will test the communication between the backend and frontend by testing our REST API with the mockito framework.

We decided to choose Mockito as it allows us to mock object or service functionality in unit testing. This will prove useful as it allows us to isolate particular components and services without having to make real time calls to their respective functions.

5.2 Hardware and Software

We will utilize the mockito testing framework to test the components that we develop in either the frontend or backend. We will have to test whether each function returns what we expect it to. How the mockito framework will be utilized is described in the previous section.

5.3 Functional Testing

We will conduct unit testing on each individual component. This will allow us to isolate potential bugs on the component level and prevent those bugs from affection other parts of the system.

Following the unit testing on components, we will conduct integration testing when trying to get the components to work in conjunction. This will allow us to testing individual component segments prior to full system or acceptance testing.

This will lead to us conducting the final system and acceptance testing. This is required in order to ensure that we deliver a polished final product which is error free.

5.4 Non-Functional Testing

One of our primary objectives will be to test the security of the overall system. We need to ensure that our project does not expose our users to things such as, others viewing the stream feeds, foreign parties disabling camera feeds, etc.

We will also conduct testing on the performance of the system. We will need to ensure our system provides prompt detection of motion as even a delay of a few seconds may determine whether or not a crime gets recorded. Our users are relying on this system as a means of security and we need to ensure that our product can perform well.

5.5 Process

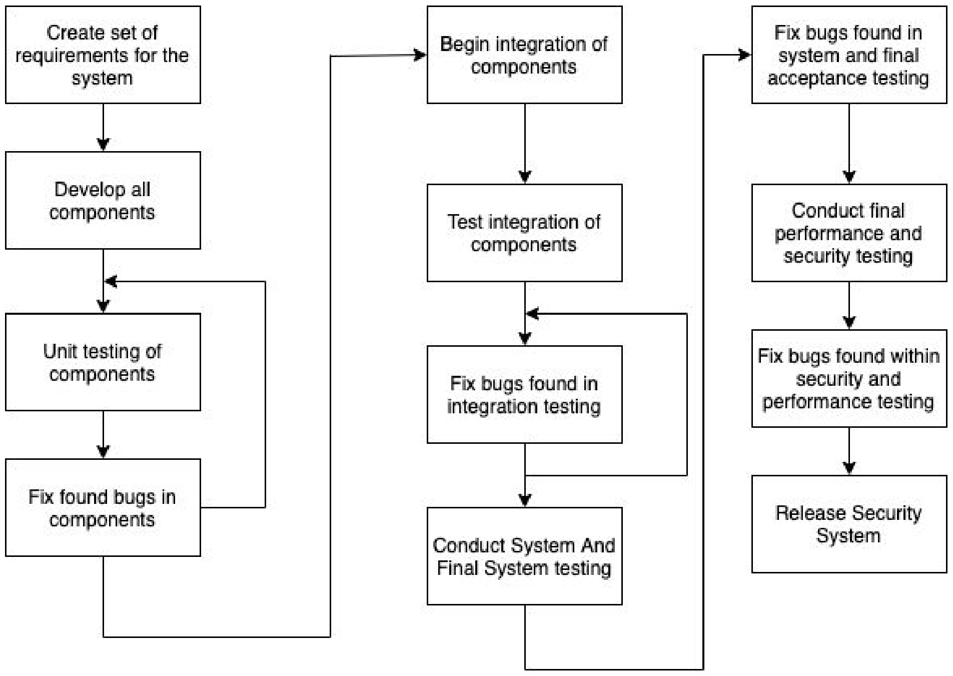

Create set of requirements for the system
Develop all components
Unit testing of components
Fix found bugs in components
Begin integration of components
Test integration of components
Fix bugs found in integration testing
Conduct System And Final System testing
Fix bugs found in system and final acceptance testing
Conduct final performance and security testing
Fix bugs found within security and performance testing
Release Security System

Figure 7. Testing Process

5.6 Results

We have not yet begun the testing of the system. We cannot report results of testing currently but plan to reform this section in the next semester.

Our largest hurdle in implementation will be having to learn the technologies that we need to use to create our system. While our team does have general knowledge on all technologies being used, there is still much to learn. We have been learning over this semester but we may run into issues with implementation due to the lack of familiarity with the technologies.

We have listed strategies above to mitigate this issue.

6. Closing Material

6.1 Conclusion

So far we have come up with a full system architecture along with the requirements for all individual components. We have also decided which technologies we will implement our solution in and the testing schedule and frameworks for the project.

Our goal is to provide a security solution that reduces the financial strain of existing security systems. We want to develop a system that does not require equipment purchases or monthly fees.

The main debate in our choice of a solution was in choosing a local server rather than a cloud solution. We chose to go with a local solution as cloud services are more expensive then having the user use a machine that they already own to process the streams. In the future we plan to scale to cloud support if we can get enough users to justify the cost.

6.2 References

[1] 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/.

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

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