Camera Software Report

Version 2.0
May 8, 2023

Low energy camera

Report Prepared by:

Karla Ferrel-Castillo, Benedict Frimpong, Youness Azzab, and Anwar Degu

Table of Contents

1. Introduction

1.1 Purpose

The project file provides a detailed examination of the technological capabilities and features of the ESP-32 camera and examines several topics, including wireless networking, sensor technology, and image processing. I've also included information about the camera's programming that provides the capability to identify faces and objects, which, in my opinion, will be a significant development in low-energy camera technology. Additionally, this report will discuss the camera's program design with an emphasis on the format to provide the best capabilities, which can be used to provide a visual representation of the camera's subsystems and how they interact with one another in a software component diagram.

1.2 User's Expectations

This camera will be a low-energy camera that can be easily installed into any product or for use on its own. This project aims to create a design that will maximize the camera's capability without losing the quality of the processed images or the user's interface integrity. This report aims to give an overview of the camera requirements, software components, and a comprehensive manual for users. The report will go through the essential functions of the camera, its construction, the face recognition technology, and possible uses for this item.

1.3 Glossary

Term	Definition

Author	A person submitting an article to be reviewed. In the case of multiple authors, this term refers to the *principal author* with whom all communication is made.
Computational Designer	Developer who takes raw data and assures implementation of all parameters and packages that allows implementation and automation of features on the camera.
Component	Part of software that is identified as a particular function
Database	Collection of all the information, modified images, and video monitored by this system.
Developer	An agile team member who is a software engineer-coder
Reader	Anyone visiting the site to read articles.
Review	A written recommendation about the appropriateness of an article for publication; may include suggestions for improvement.
Reviewer	A person that examines an article and has the ability to recommend approval of the article for publication or to request that changes be made to the report.
Software Requirements Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document.
Stakeholder	Any person with an interest in the project who is not a developer.
Tester	A developer that tests software components to assure quality and integrity.
User	Reviewer or owner

1.4 References

1. Ashshak, Sharifdeen. "ESP32-Cam Face Detection: Face Recognition." *Hackster.Io*, October 21, 2020, www.hackster.io/ashshaks/esp32-cam-face-detection-face-recognition-2468e0.

2. "Introduction to RTOS Part 1 - What Is a Real-Time Operating System (RTOS)? | Digi-Key Electronics." *YouTube*, 2021, www.youtube.com/watch?v=F321087yYy4&feature=youtu.be.

3.0 Javed, Amaan. "Python Opencv NODEMCU Mask Detection." *Hackster.Io*, March 8, 2022, www.hackster.io/javedbasira/python-opencv-nodemcu-mask-detection-9279bc.

4.0 Lemariva. "Lemariva/Micropython-Camera-Driver: Add Camera Support to MicroPython." *GitHub*, github.com/lemariva/micropython-camera-driver. Accessed May 9, 2023.

"Online Compiler and Debugger for C/C++." *GDB Online Debugger*, www.onlinegdb.com/MDypelvjv. Accessed May 9, 2023.

Shankhdhar, Priyansh. "ESP32 Cam Based Face & Eyes Recognition System." *How To Electronics*, August 20, 2022, how2electronics.com/esp32-cam-based-face-eyes-recognition-system/.

1.5. Overview of Document

The second chapter, the Requirements Specification section, of this document is written primarily for the developers and computation designers. It describes in technical terms the details of the functionality of the camera. Both sections of the document describe the same software product in its entirety but are intended for different audiences and thus use different languages and respective terminology.

2. Specifications and Requirements

The camera is a high-definition gadget that takes excellent photos and videos. A wide-angle lens on it allows it to take pictures of a large region. Two-way communication between the camera and the user is made possible via the camera's built-in microphone and speaker. Based on artificial intelligence algorithms, the camera's facial recognition technology recognizes faces. It operates by implementing a package that examines facial characteristics and contrasts them with a database of recognized faces. Even in low light, the camera can discriminate between similar-looking faces and recognize faces. The real-time facial detection capability of the technology enables rapid alerts and notifications.

The object recognition technology used in the camera is based on computer vision and machine learning algorithms. It works by analyzing the pixels in the camera's image and identifying patterns and features that match known objects. The camera can recognize a wide range of objects, including people, animals, vehicles, and other objects.

2.1 Design

The camera has a sleek and modern design, with a compact size that makes it easy to install in any location. It is equipped with a mounting bracket that allows for easy installation on walls or ceilings. The camera is also weather-resistant, making it suitable for both indoor and outdoor use.

2.2 Applications

Numerous uses for the camera equipped with face recognition technology are possible. It can be utilized for security applications in homes and companies, such as observing entrances, exits, and critical locations. To track consumer activity and enhance marketing initiatives, the camera can also be employed in retail settings. It can also be used in healthcare facilities to keep an eye on patient behavior and protect their security.

The camera with face recognition technology is an innovative product that has the potential to revolutionize the way we monitor and secure our homes and businesses. Its high-quality design, advanced technology, and wide range of potential applications make it an ideal choice for anyone looking for a reliable and effective security solution.

Camera's Resolution:

The purpose of this part of the report is to provide an overview of a camera with a resolution of 256x256 pixels (256p). The report will cover the definition of camera resolution, the features and limitations of a camera with 256p resolution, and the potential applications of this product.

2.3 Camera Resolution

The number of pixels a camera can record in an image or video is referred to as resolution. The resolution and level of detail in an image will increase as the number of pixels on a camera increases. Typically, camera resolution is expressed as several horizontal and vertical pixels, such as 256x256.

2.4 Features and Limitations of 256p Resolution Camera

By today's standards, a camera with a 256p resolution is low-resolution. Compared to higher-resolution cameras, it can record images that are very tiny and lack detail. While the 256p resolution is enough for simple surveillance and monitoring tasks, it might fall short for more intricate tasks like facial recognition or scanning license plates.

One advantage of a 256p camera is its small file size, which requires less storage space and bandwidth. This can be beneficial for applications where storage and bandwidth are limited, such as remote monitoring or mobile applications.

2.5 Potential Applications

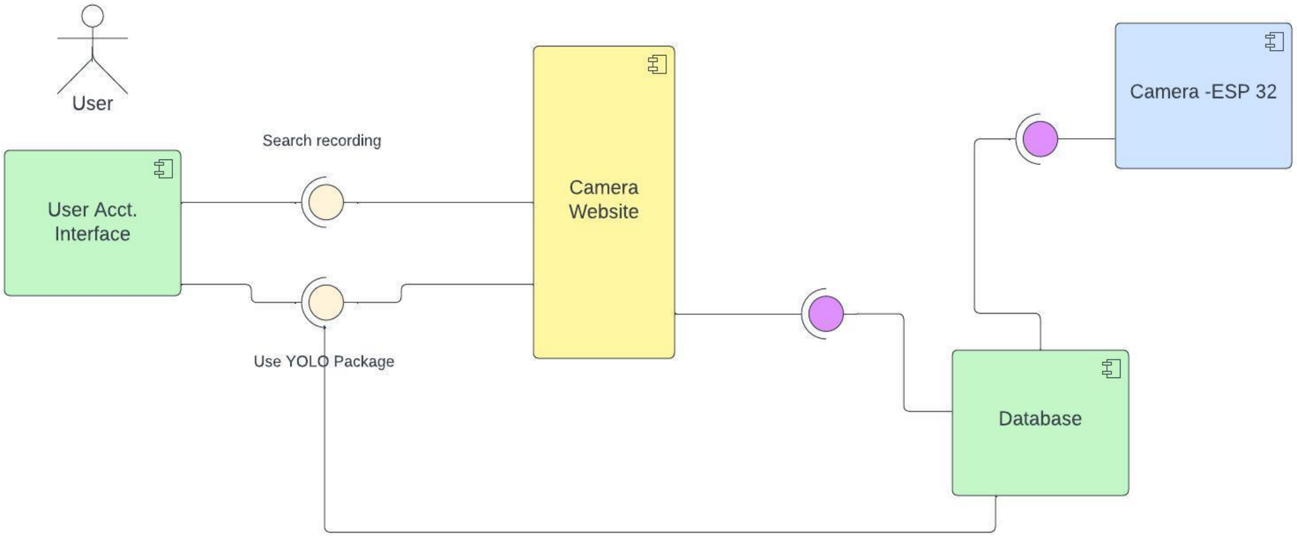
In some situations, such as the surveillance of small spaces like confined spaces or entranceways, a camera with 256p resolution may still be beneficial. Additionally, it can be used in low-light situations where a higher-resolution camera could find it difficult to get clear pictures. A 256p camera may also be used in mobile devices like drones or robots, which need a compact and lightweight camera. The low resolution might be adequate for fundamental visual navigation and obstacle avoidance activities.

As per the camera requirements outlined in the previous discussion, it is evident that certain features are highly expected in a modern camera. These features include recognizing sign language, detecting dangerous situations, recording footage for emergencies, and offering high-resolution capabilities for landscape and architectural photography.

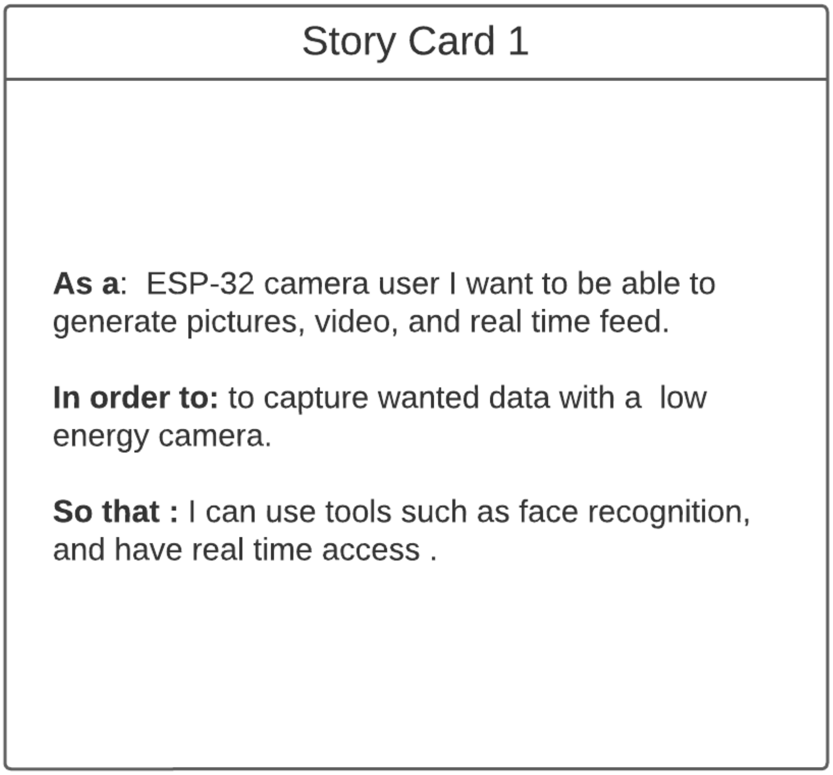
In addition, an OLED display, speech recognition, GPS tracking, night vision mode, focus stacking, movement sensors, and AI capabilities are anticipated for the camera. In order to connect to social networking apps and upload the video to the cloud, the camera is also anticipated to have wireless connectivity, such as Bluetooth, Wi-Fi, and cellular connectivity. The camera should also be made to be adaptable, simple to use, and furnished with a wireless, detachable touch OLED display that offers a user-friendly interface for adjusting the camera's settings and watching the footage. The camera should be able to operate solely on solar power or the battery charge it creates, as well as voice-activated shutter firing, an open source/standard wireless flash commander system, and solar power capabilities.

All things considered, these expectations show the necessity for a camera that is strong, adaptable, and created to cater to the many needs of contemporary photographers and videographers. It is conceivable to design a camera with these features that may provide excellent results, take lovely pictures, and give users peace of mind in emergency situations.

2.6 System Environment



2.7 Sprints of User's Story Cards for Agile Development

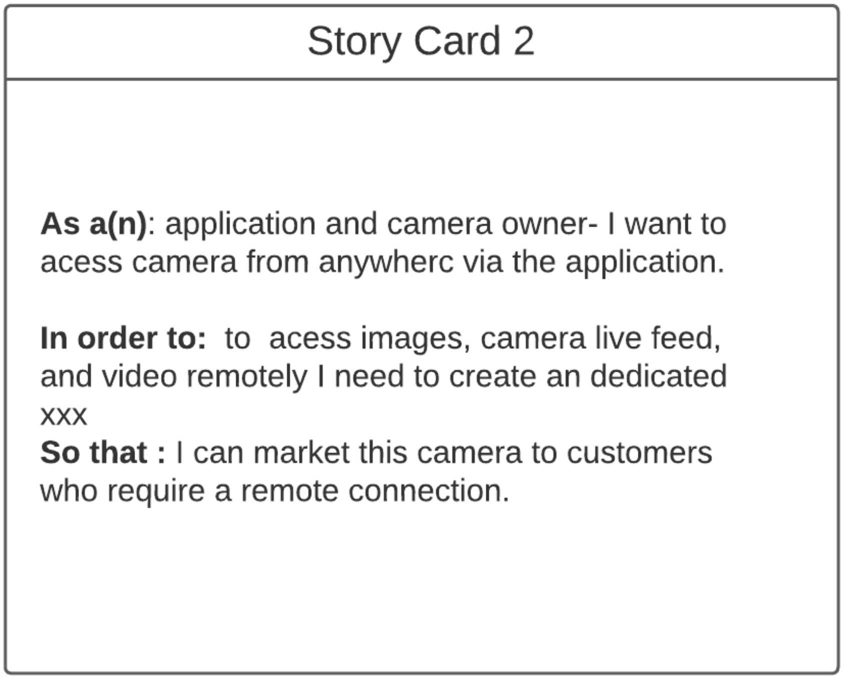


Story Card 1

As a: ESP-32 camera user I want to be able to generate pictures, video, and real time feed.

In order to: to capture wanted data with a low energy camera.

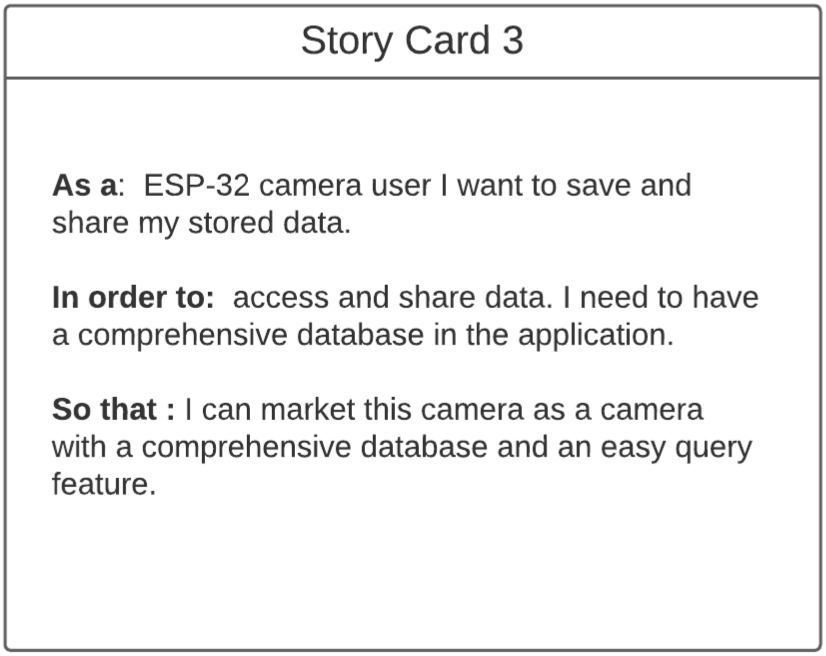
So that : I can use tools such as face recognition, and have real time access .



Story Card 2

As a(n): application and camera owner- I want to acess camera from anywherc via the application.

In order to: to acess images, camera live feed, and video remotely I need to create an dedicated xxx
So that : I can market this camera to customers who require a remote connection.



Story Card 3

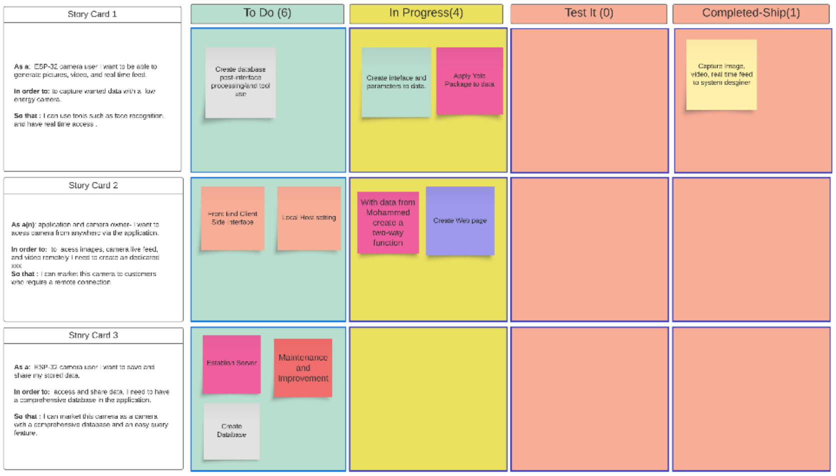
As a: ESP-32 camera user I want to save and share my stored data.

In order to: access and share data. I need to have a comprehensive database in the application.

So that : I can market this camera as a camera with a comprehensive database and an easy query feature.

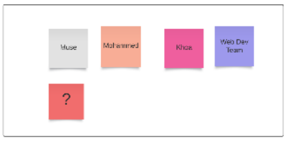
3. Sprint

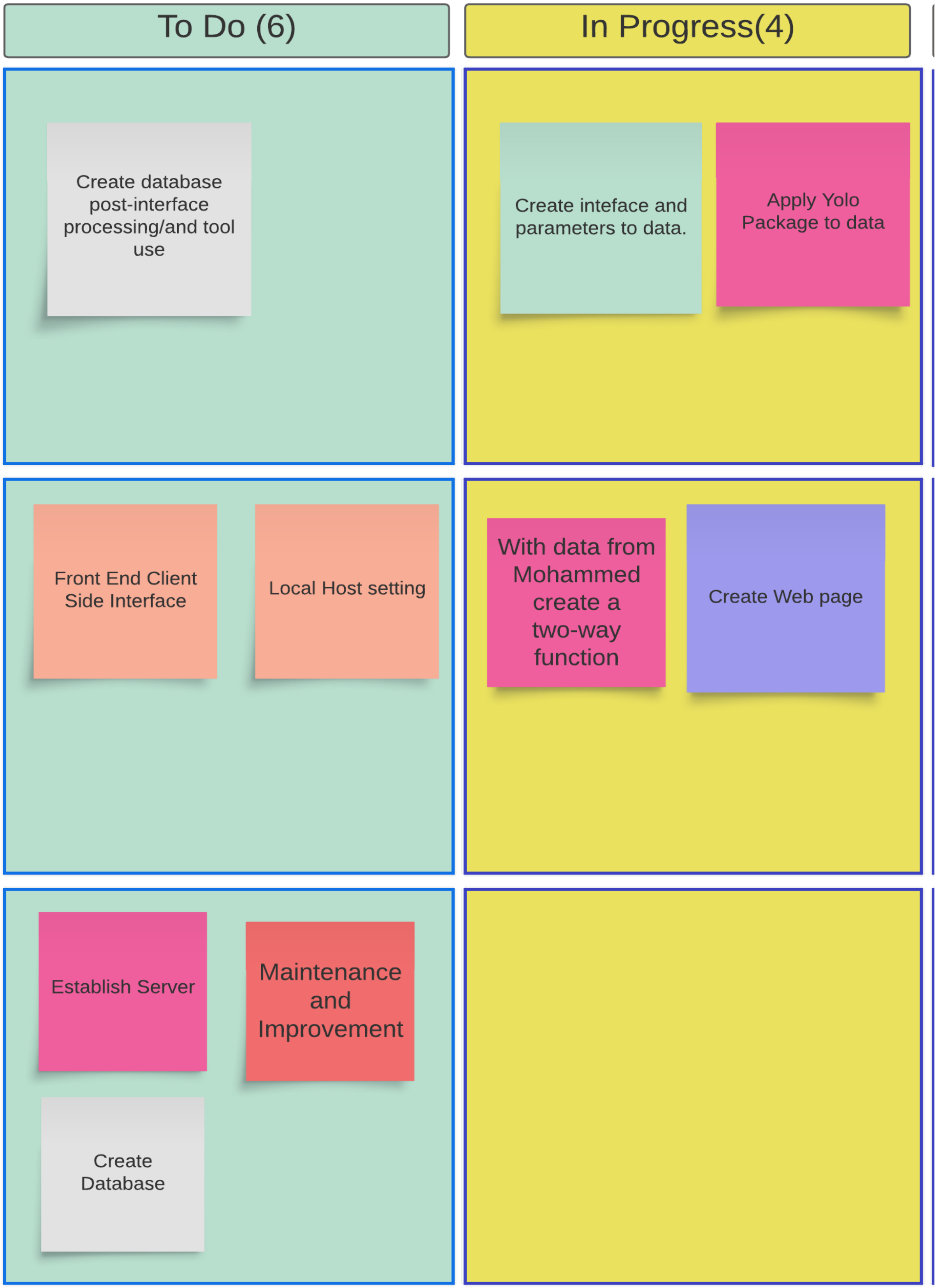
Below is the breakdown of our sprints to meet and modify our progress with the entire team and stakeholders of the product. We created the story cards to ensure that we remain faithful to the objectives and needs of the stakeholders. We also created a sprint board to ensure that we knew the status of each component of the software, system design, interface, and



	To Do (6)	In Progress(4)	Test It (0)	Completed-Ship(1)
Story Card 1 **As a:** ESP-32 camera user I want to be able to generate pictures, video, and real time feed. **In order to:** to capture wanted data with a low energy camera. **So that :** I can use tools such as face recognition, and have real time access .	Create database post-interface processing/and tool use	Create inteface and parameters to data. Apply Yolo Package to data		Capture image, video, real time feed to system desginer
Story Card 2 **As a(n):** application and camera owner- I want to acess camera from anywhere via the application. **In order to:** to acess images, camera live feed, and video remotely I need to create an dedicated xxx **So that :** I can market this camera to customers who require a remote connection.	Front End Client Side Interface Local Host setting	With data from Mohammed create a two-way function Create Web page		
Story Card 3 **As a:** ESP-32 camera user I want to save and share my stored data. **In order to:** access and share data. I need to have a comprehensive database in the application. **So that :** I can market this camera as a camera with a comprehensive database and an easy query feature.	Establish Server Maintenance and Improvement Create Database			

Sticky note color legend





To Do (6)

Create database post-interface processing/and tool use

Front End Client Side Interface

Local Host setting

Establish Server

Maintenance and Improvement

Create Database

In Progress(4)

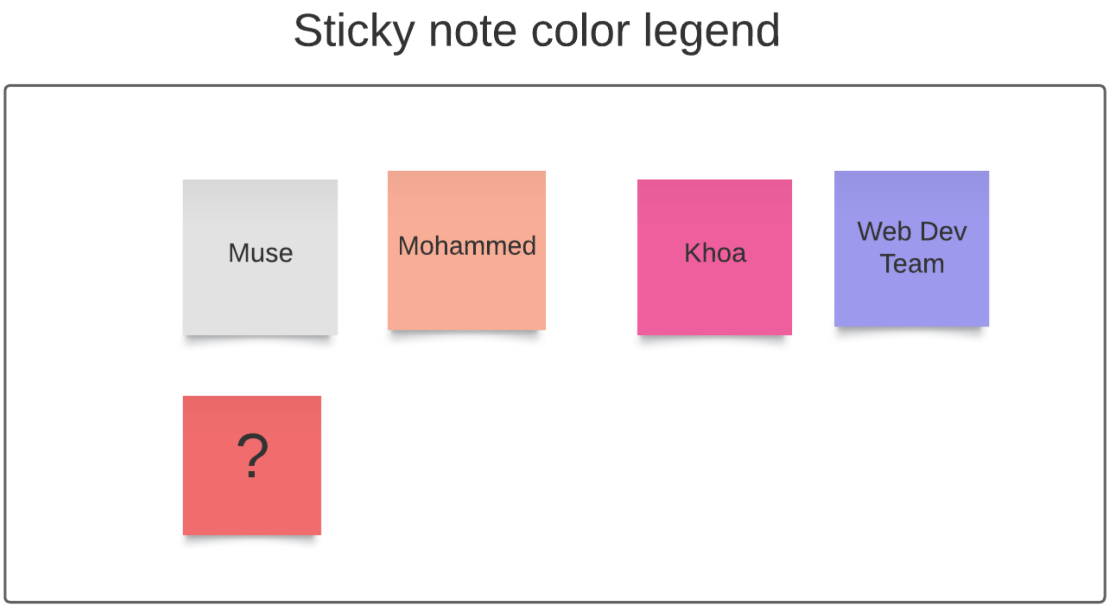
Create inteface and parameters to data.

Apply Yolo Package to data

With data from Mohammed create a two-way function

Create Web page

Test It (0)	Completed-Ship(1)
	Capture Image, video, real time feed to system desginer



3.1 Agile Team Organization, Assignments, and Management

We have formed an agile team. The team contains a project owner and a scrum master. We have split the job/task assignment among our team members.

Muse Tesfaye – Coder - will work on the interface and GUI using Qt5 Designer. He will also work on the functionalities and integrate the interface to work with the camera.

Mosunlade Ishola - coder - will create a website. Research PHP

Khoa Pham - Get code to run and test on camera ESP32. Help with coding and computation design.

Mohammned Bayo - Coder – will work with Muse to create computer vision algorithms for detecting faces and objects.

Anwar Degu - tester, however, since testing is not needed yet, will help the team leader to extract reports from all members and provide the first team report.

Thirno Sanousy Sow - coder - interface programming and code testing.

Benedict Frimpong – Scrum Master, will check in with all tasks and host the meeting on Wednesday.

Youness Azzab - Will work with Karla to provide the software specification document.

Joshua Proctor - will work with Mosunlade Ishola to create the website.

Kao Shyang: Will build the website with Mosunlade Ishola and Joshua Proctor

---

3.2 Weekly Scrum Meeting- Sprint 1

Scrum Meeting Date: 03/22/2023
Sprint Planning

Participants [Benedict Frimpong; Karla Ferrel Castillo; Joshua Proctor; Khoa Pham; Muse Tesfaye; Mosunlade Ishola; Mohamned Bayo; Shyang Kao; Azzad Youness] (9)

Agenda:

- Review project requirements.
- Identify and prioritize tasks for the sprint process.
- Assign tasks to team members based on their expertise and availability.
- Set sprint goals and define success criteria and deadlines.

Summary:

We reviewed the project requirements and user stories in detail during the meeting. We identified the key functionalities of the ESP-32 camera, which included capturing pictures and sending them to a database. Based on the requirements, we created a prioritized list of tasks for the sprint process.

The team members were assigned tasks based on their skills and availability. We ensured that each task had a clear owner responsible for its completion. Additionally, we discussed the sprint goals and defined success criteria to measure the progress and achievement of our objectives.

Date: 03/29/2023

Participants [Benedict, Frimpong; Karla, Ferrel Castillo; Joshua Proctor; Khoa Pham; Muse Tesfaye; Mosunlade Ishola; Mohamned Bayo; Shyang Kao.] (8)

Summary of the meeting:

The coding team presented what they have so far. They have successfully written the code to communicate with the ESP-32 Camera in Python. As pointed out during the previous meeting, they now have a function to control the camera's flash feature to reduce the camera's overheating. They are working on establishing the communication protocol between the camera and our website before our next meeting. They have also shared the code with all of us on Discord.

The website team also presented what they have. They have two layouts right now and will be working on merging them before our next meeting. The scrum master suggested that they include the UDC colors in the styling of the website. They will also share the code.

3.2 Scrum Meeting Date: 04/05/2023

Benedict, Frimpong; Sow Thirno Sanousy; Shyang Kao; Khoa Pham; Anwar Degu; Karla Ferrel Castillo; Joshua Proctor; Muse Tesfaye; Mosunlade Ishola. (8)

Objective:
The purpose of this meeting was to provide a daily status update on the progress of the ESP-32 camera project. Each team member shared their accomplishments, challenges, and planned tasks for the day.

Agenda:

- Briefly discuss the progress made since the last meeting.
- Share any challenges or roadblocks encountered.
- Communicate the tasks planned for the day.
- Identify any dependencies or assistance required.

Summary:
In this weekly meeting, each team member provided a brief update on their progress since the previous meeting. Accomplishments were shared, including any completed tasks or milestones achieved. Challenges and roadblocks were discussed openly to ensure timely resolution. Team members also communicated their planned weekly tasks, aligning their efforts toward achieving the sprint goals. Any dependencies or assistance needed from other team members were identified, allowing for collaboration and support. The meeting concluded with a clear understanding of individual responsibilities for the day. Implementing face and object detection capabilities was added to the project requirement.

3.3 Weekly Scrum Meeting – Phase 2

Date: 04/26/2023

Sprint Review and Demo

Participants [Benedict, Frimpong; Joshua Proctor; Khoa Pham; Mosunlade Ishola; Muse Tesfaye; Shyang Kao; Sow Thirno Sanousy.] (7)

Objective:
The purpose of this meeting was to review the sprint's progress, demonstrate the implemented functionalities, and gather stakeholder feedback.

Agenda:

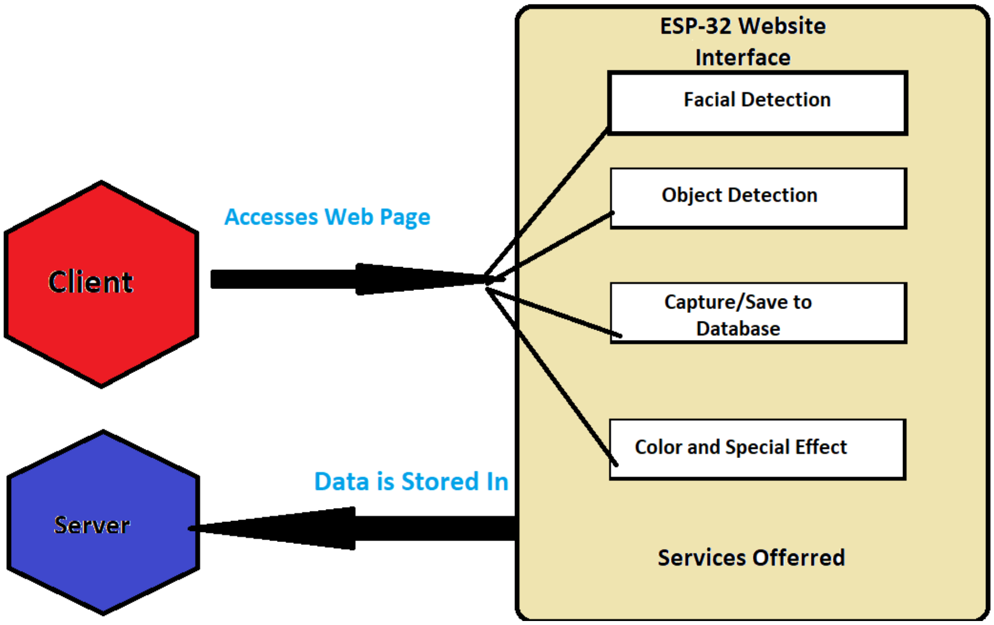
- Present the completed tasks and functionalities of the sprint.
- Perform a live demo of the ESP-32 camera, showcasing the picture capture, database integration, and face/object detection features.
- Collect feedback and suggestions from stakeholders. (In Our case, the professor)
- Discuss any changes or adjustments required for the next sprint.
-

Summary:

During this sprint review and demo meeting, the team presented the completed tasks and functionalities achieved during the sprint. A live demonstration of the ESP-32 camera was conducted, showcasing its ability to capture pictures, send them to a database, and perform face and object detection. Stakeholders provided valuable feedback and suggestions based on the demo, which was noted for future improvements. Discussions were held to address any concerns and clarify requirements, ensuring alignment between the project team and stakeholders.

3.4 Progresses in System Design

Design: Architecture Design and System Design



Webpage Design with Clint and Server (Ishola/Kao/Josh)

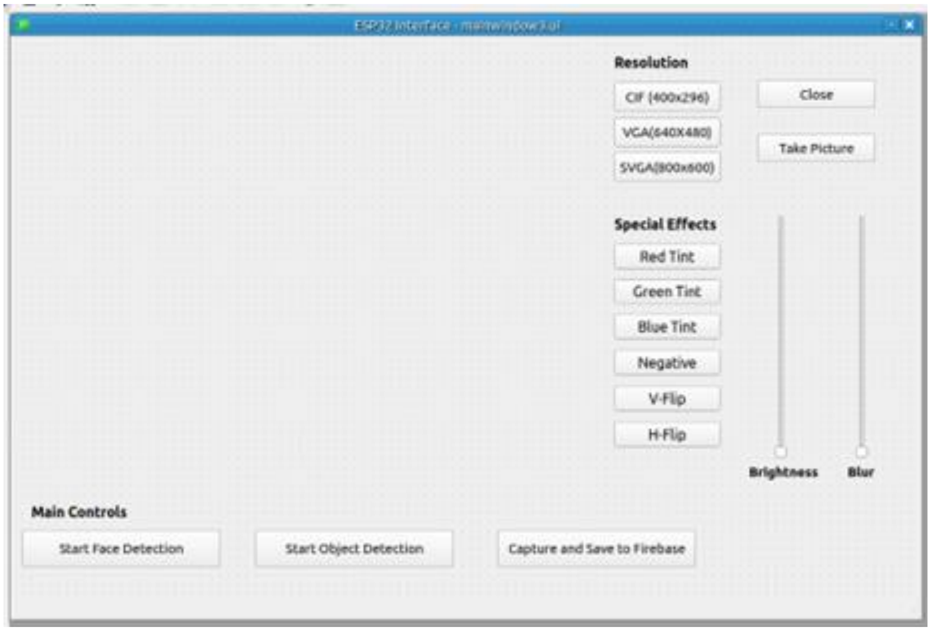
We are meeting tomorrow evening for a streaming video display.

3.5 Scrum Weekly Update 5.1.2023

Our lead software engineer, Muse Tesfaye, has been working on the interface, GUI. He first started designing the interface using Qt5 Designer. Muse had a couple of prototypes to go through to incorporate every requirement.

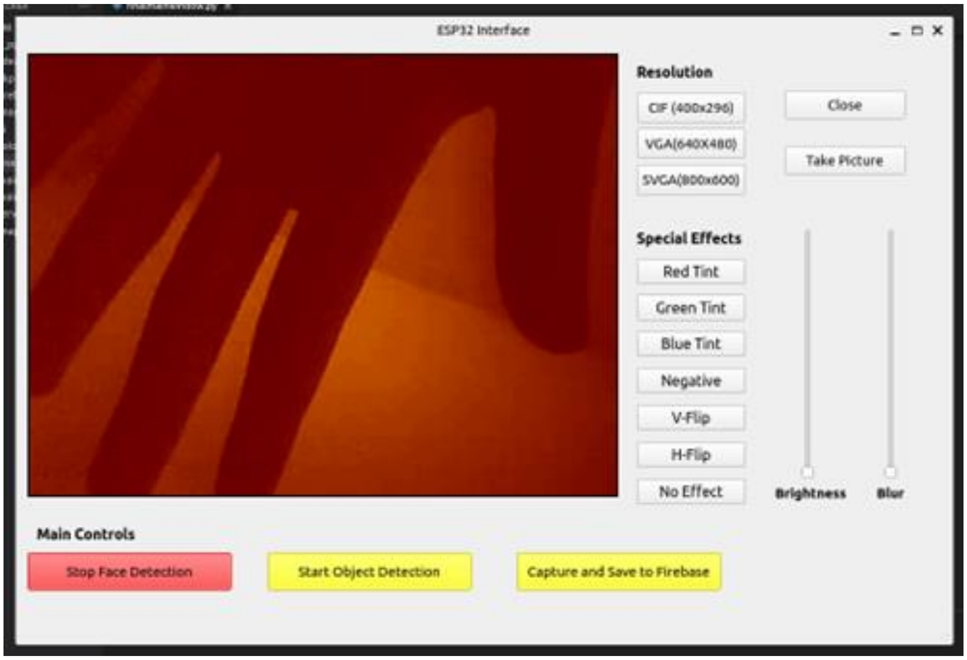
Below is Muse's Design for the interface GUI.

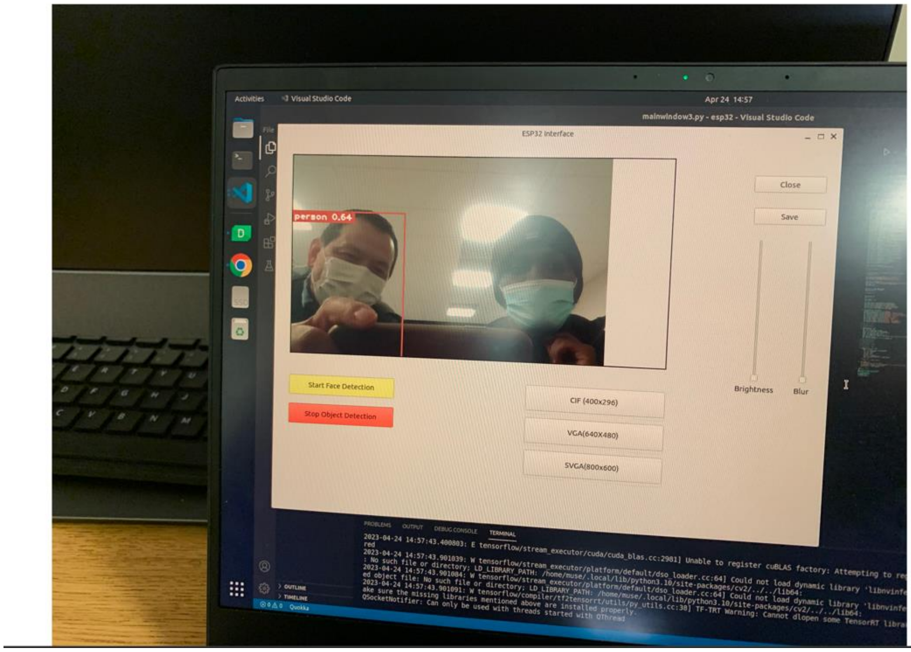
Image 1



Then, Muse began working on the functionalities and integrating the interface to work with the camera. He created an executable version of the interface so that he would be able to use it. Muse then hooked the buttons to their corresponding functions, and it all worked as intended.

Here is a sample:

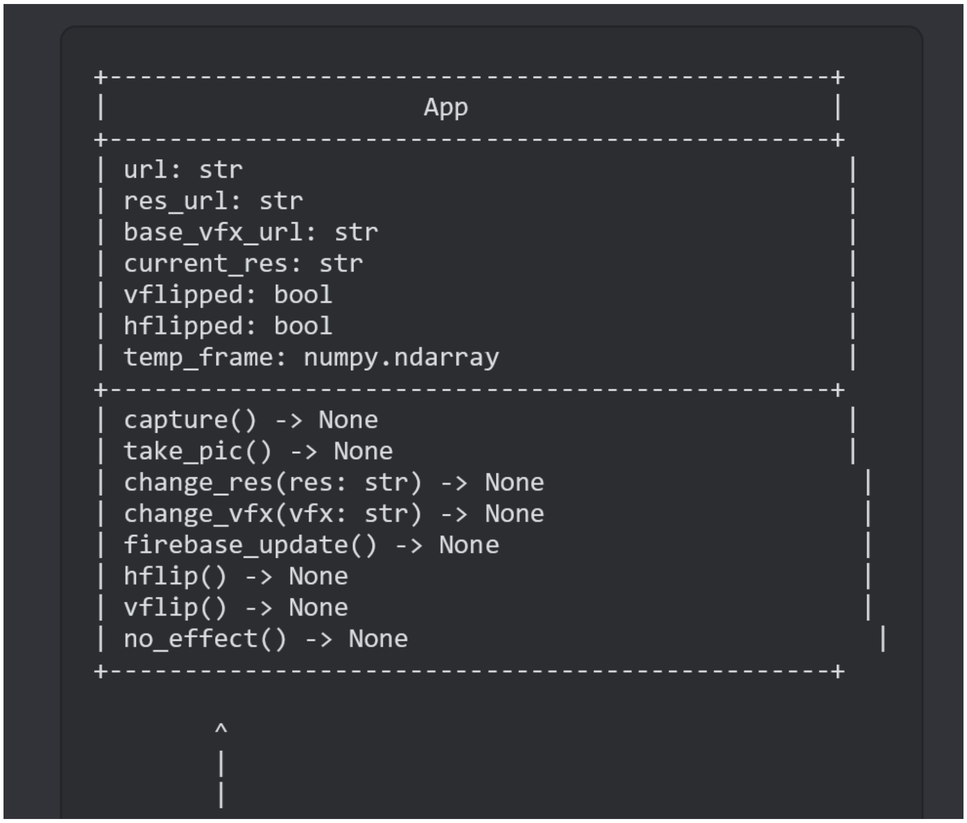




```
2023-04-24 14:57:43.400803: E tensorflow/stream_executor/cuda/cuda_blas.cc:2981] Unable to register cuBLAS factory: Attempting to re
red
2023-04-24 14:57:43.901039: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfe
: No such file or directory; LD_LIBRARY_PATH: /home/muse/.local/lib/python3.10/site-packages/cv2/../../lib64:
2023-04-24 14:57:43.901084: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfe
ed object file: No such file or directory; LD_LIBRARY_PATH: /home/muse/.local/lib/python3.10/site-packages/cv2/../../lib64:
2023-04-24 14:57:43.901091: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot dlopen some TensorRT libra
ake sure the missing libraries mentioned above are installed properly.
QSocketNotifier: Can only be used with threads started with QThread
```

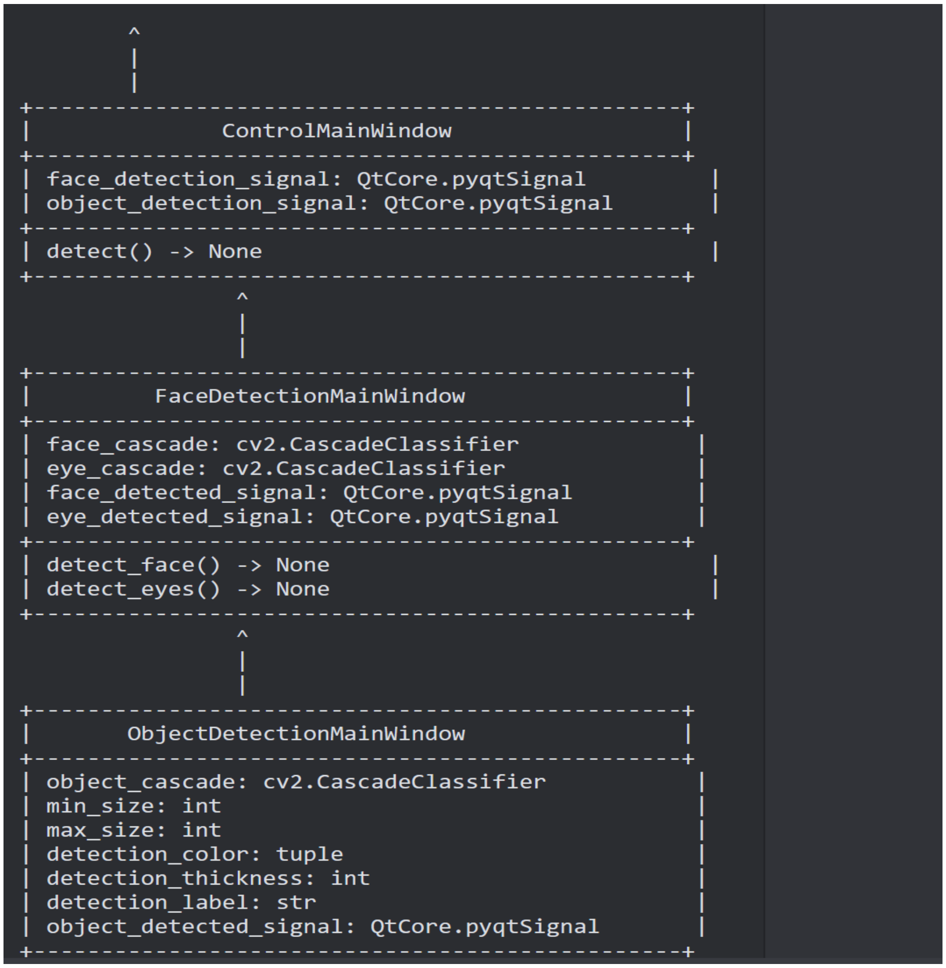
3.6 Parameters and Post-Image Design on Interface

Post Image generation, we will need to implement parameters and interface design for the images that will be generated and displayed through the application and stored on the database. Below is a class diagram for the parameters implemented to all captures, including images, video, and output to the user interface.

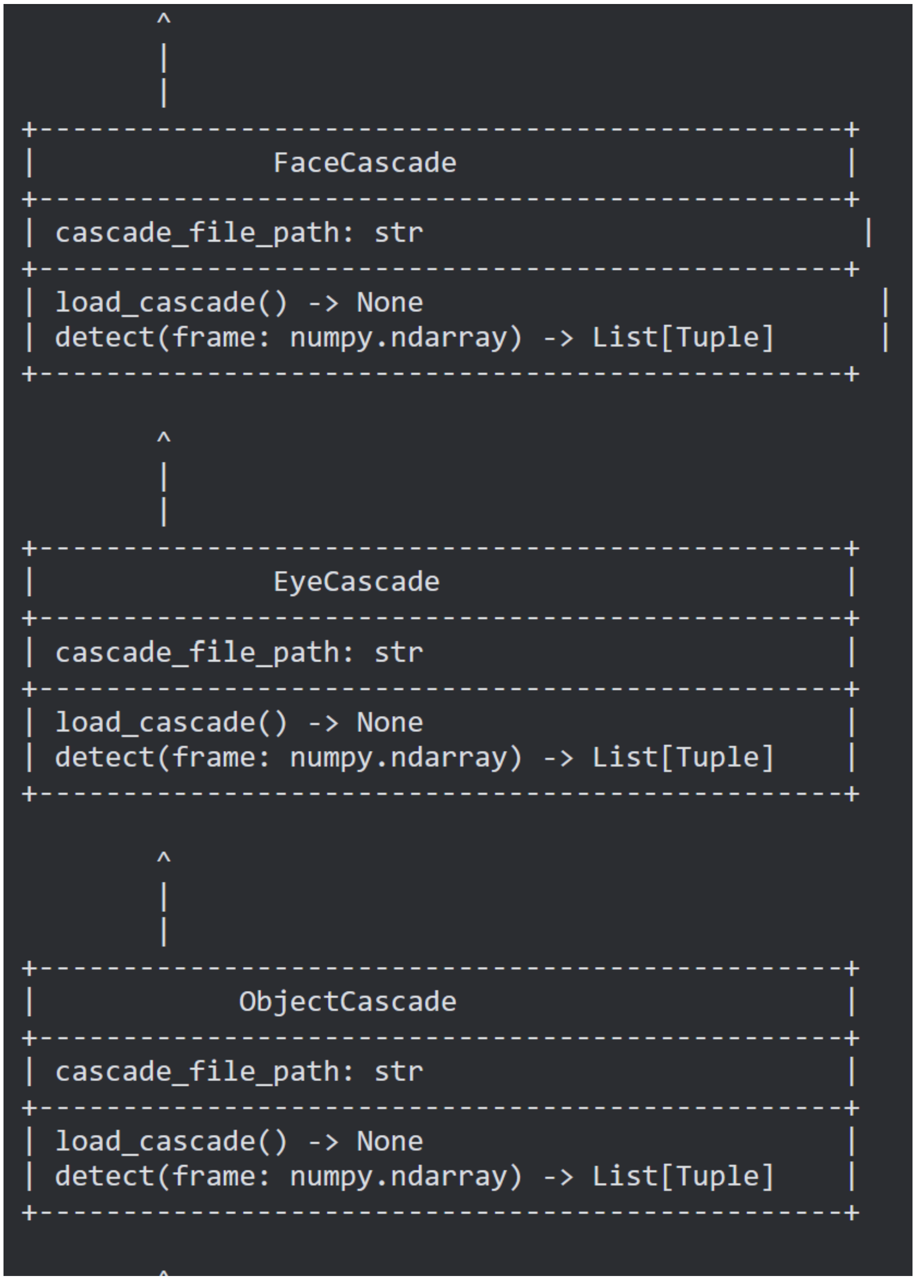


```
+------------------------------------------------+
|                     App                        |
+------------------------------------------------+
| url: str                                        |
| res_url: str                                    |
| base_vfx_url: str                               |
| current_res: str                                |
| vflipped: bool                                  |
| hflipped: bool                                  |
| temp_frame: numpy.ndarray                       |
+------------------------------------------------+
| capture() -> None                               |
| take_pic() -> None                              |
| change_res(res: str) -> None                     |
| change_vfx(vfx: str) -> None                     |
| firebase_update() -> None                        |
| hflip() -> None                                  |
| vflip() -> None                                  |
| no_effect() -> None                               |
+------------------------------------------------+

        ^
        |
        |
```



```
        ^
        |
        |
+-------------------------------------------------+
|              ControlMainWindow                  |
+-------------------------------------------------+
| face_detection_signal: QtCore.pyqtSignal          |
| object_detection_signal: QtCore.pyqtSignal        |
+-------------------------------------------------+
| detect() -> None                                  |
+-------------------------------------------------+
                ^
                |
                |
+-------------------------------------------------+
|         FaceDetectionMainWindow                 |
+-------------------------------------------------+
| face_cascade: cv2.CascadeClassifier              |
| eye_cascade: cv2.CascadeClassifier               |
| face_detected_signal: QtCore.pyqtSignal          |
| eye_detected_signal: QtCore.pyqtSignal           |
+-------------------------------------------------+
| detect_face() -> None                             |
| detect_eyes() -> None                             |
+-------------------------------------------------+
                ^
                |
                |
+-------------------------------------------------+
|       ObjectDetectionMainWindow                 |
+-------------------------------------------------+
| object_cascade: cv2.CascadeClassifier            |
| min_size: int                                    |
| max_size: int                                    |
| detection_color: tuple                           |
| detection_thickness: int                         |
| detection_label: str                             |
| object_detected_signal: QtCore.pyqtSignal        |
+-------------------------------------------------+
```



```
        ^
        |
        |
+------------------------------------------------+
|              FaceCascade                       |
+------------------------------------------------+
| cascade_file_path: str                          |
+------------------------------------------------+
| load_cascade() -> None                           |
| detect(frame: numpy.ndarray) -> List[Tuple]      |
+------------------------------------------------+

        ^
        |
        |
+------------------------------------------------+
|              EyeCascade                        |
+------------------------------------------------+
| cascade_file_path: str                         |
+------------------------------------------------+
| load_cascade() -> None                         |
| detect(frame: numpy.ndarray) -> List[Tuple]    |
+------------------------------------------------+

        ^
        |
        |
+------------------------------------------------+
|            ObjectCascade                       |
+------------------------------------------------+
| cascade_file_path: str                         |
+------------------------------------------------+
| load_cascade() -> None                         |
| detect(frame: numpy.ndarray) -> List[Tuple]    |
+------------------------------------------------+

        ^
```

4. Server installation

Due to time constraints, we were unable to generate a server for the camera. This was discussed as a need if we were going to create a database for the camera for post-processed images and storing access. This was included and is in the original system design. If we build a second generation or version of this camera, a server is highly recommended to provide all the features we initially set as goals for this product.

5. web design

While web design is the front end and a very important part of this product, we couldn't complete and launch the website. However, the coding and design of the website were completed. If this product can be further developed, attached is the GitHub link to the web development team's coding portfolio for this product. https://github.com/kaotantuong/Smart-Camera.github.io.

5.1 Implementation and Coding

Implementation and coding information is posted and will be made available via GitHub. The developer for this is Thirno Saousy. The developer provided two separate reports with the code for this specification.

6. Testing and Maintenance

As the product was updated in each sprint phase, testing was done to assure the quality and integrity of the code that was handed to the next phase of each sprint phase.