**CS673 Software Engineering (AIDAN)**

**Team 5 - Fafi**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Zengrui Luo | Security leader | *Zengrui Luo* | 9/26/2022 |
| Matthew Kluska | Team leader | [Matthew Kluska](mailto:kluska@bu.edu) | 9/26/2022 |
| Brendan Truong | QA leader | *Brendan Truong* | 9/26/2022 |
| Aidan Chang | Design and Implementation leader | *Aidan Chang* | 9/26/2022 |
| Derric Syme | Configuration leader | *Derric Syme* | 9/26/2022 |
| Patounezambo  Ouedraogo | Requirement leader | *Patounezambo* | 9/26/2022 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 1 | Aidan Chang | 9/26/2022 | Iteration 1 submission |
|  |  |  |  |

[Introduction](#_87t9hln2vjz0)

[Software Architecture](#_3ipvmjgn6clp)

[Class Diagram](#_ky60nv8suxxm)

[UI Design (if applicable)](#_7ucksmkf6rzx)

[Database Design (if applicable)](#_tcmuor4nl1kz)

[Security Design](#_x18fj36s1121)

[Business Logic and/or Key Algorithms](#_mtfbusfb0eq3)

[Design Patterns](#_9zvwkmc4luo5)

[Any Additional Topics you would like to include.](#_15tmymhipvdv)

[References](#_50ojo9i46ytq)

[Glossary](#_8n34lvocupub)

# Introduction

The FaFi application implements the YOLOv5 pre-trained model to detect any person objects in a live streaming video through the computer's camera. A Surveillance mode can be turned on and stores a snapshot of the video along with other information whenever people are detected inside the video, then an email notification will be sent to the email on file.

The design goal of FaFi are the following:

1. Efficiency- The ML algorithm used in this application must be efficient.
2. Reliability- The people detection and email notification must be reliable.
3. Security- The security of the end user’s account and the privacy of the history must be secured.
4. Robustness- The application must be able to run in all environments, handle errors, be fault tolerant and pass thorough testing.

# Software Architecture

Front end - Flask, Flask-Login, Flask-Mail

Back end server - SQlite3, Flask Framework

ML framework - Pytorch, Resnet 18 model

# Class Diagram

Entities:

* FaFI Web Application
* Images

Facial detection results

* Users

Controls:

* Login
* Upload File
* Label prediction for model training
* View History
* ML Facial detection

Boundaries:

* Main Menu interface
* LoginPage
* SignupPage
* HistoryPage
* ConfigPage

# UI Design

* + Login page - Users can create and login to their account
    - Fields: Username and Password
    - Buttons: Create Account and Sign In
      * Create Account redirects to Sign Up page
      * Sign In redirects to Home Page
  + Sign Up page - User creates a username and password for their account
    - Fields: Username and Password
    - Button: Submit
      * Submit redirects to Login Page. Accounts with passed username and password are created.
  + All Pages hereafter:
    - Navigation Bar: Links to Home, History, and Notifications tabs
  + Main page
    - Functionalities:
      * Displays the live stream video with people detection feature
      * Upload an image for facial detection

* + History page- Displays a history of events where people are detected.
    - Buttons: Click on an event to view the screenshot enlarged.
    - History Feature: Each event shows a screenshot on the left with time it took place, and number of people detected on the right.
  + Notification page- A page that an end user can turn on the “surveillance” feature and enter an email address for potential notifications from the application.
    - Fields: Email Address to receive notifications
    - Buttons:

1. Surveillance Feature: An end user can set up a time frame that if the application detects any people within that time period, toggle email notification will be sent to the user's email address, and a screen shot of a video will be saved and stored in the History page.

# Database Design

The Database Design will consist of a single table. This table will be created natively in python using the SQLite3 module. This table will house metadata about the uploaded image including attributes like person detected or not, unique identifying code, and timestamp. This SQL Database will then be used by the history tab, notification tab and future features. We had initial thoughts to store the uploaded photo directly inside the sql database as a base64 encoded text. We are now alternatively taking a safer approach of storing an image in a folder outside of the sql database and instead on the server, referencing the file directly instead of encoding/decoding its contents. This is to ensure images are not corrupted and retain their resolution.

Future enhancements could be to replace our SQLite3 module with a dedicated database system and encode/decode the image.

# Security Design

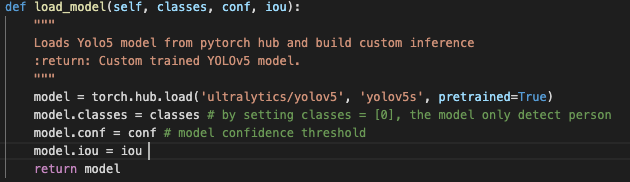
We plan to implement an authentication mechanism to prevent unauthenticated use of our application by creating a username and password. These values will be encrypted and stored within our database. We’ll make use of the flask-login plugin to provide user session management. This makes use of a login manager to handle tasks of logging the user in and out, along with managing a user’s session over a period of time. The web application will be enforced to only display and authorize history access to that corresponding to a given user.

# Business Logic and/or Key Algorithms

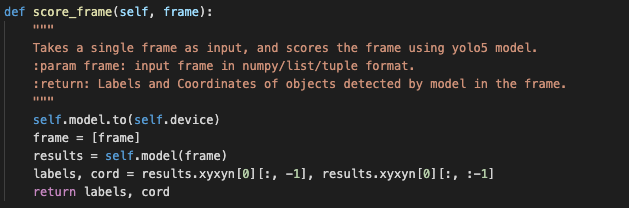
In order to detect people in a live stream video, we first use cv2.VideoCapture() method from OpenCV to capture the video, then use the .read() method to break the video into frames (images).



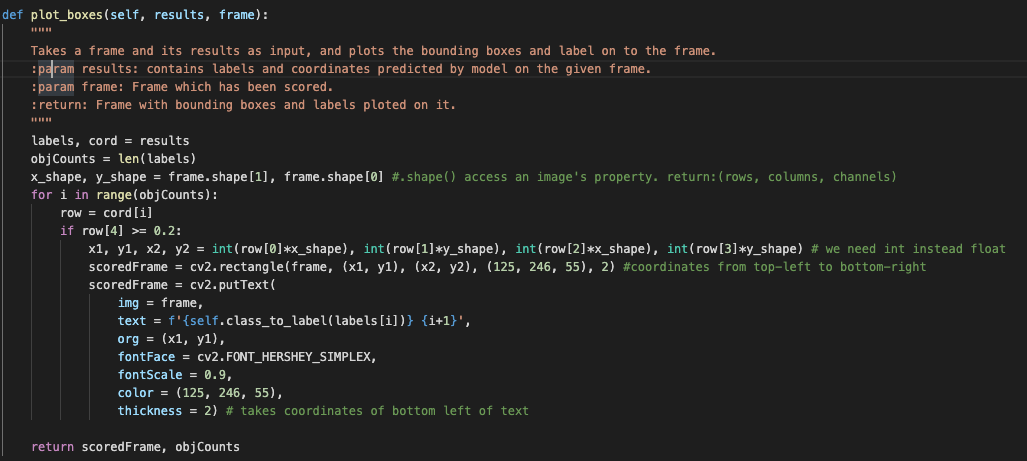
Then, we apply a pre-trained YOLOv5 model in the load\_model() method with custom inference, where our application requires higher confidence rate and person detection only.



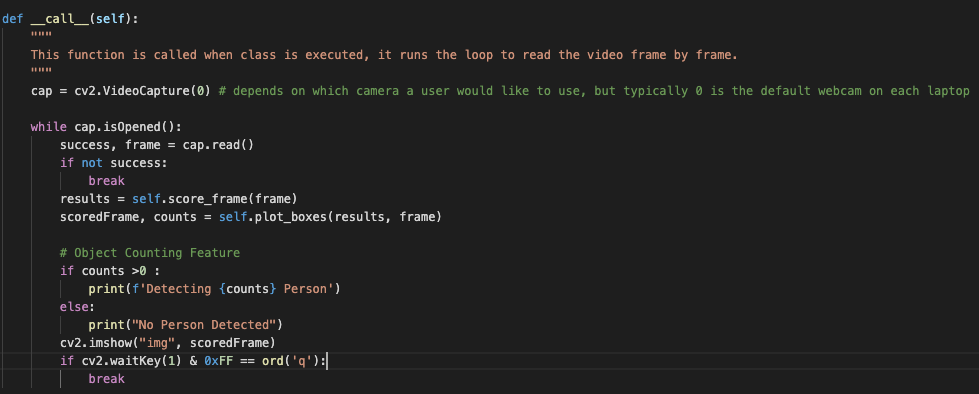
After we obtain the model and the frames (images), the model is ready to score the frames (images). The implementation is very easy thanks to Pytorch, first step is to transform the frames into numpy/list/tuple format, then declare whether our Pytorch is CPU or GPU version. Lastly, pass in the frames into the model as shown below. We can further breakdown the results into Classification and Coordinates of each object.



The Coordinates represent the top-left and bottom-right corners of a detected object. By using cv2.rectangle() method, we are able to plot the bounding boxes of each object for each model it detected.



Lastly, by using cv2.imshow() method, we can display scoredFrames. Also, a while loop is added to continuous display scoredFrames as a video output.



# Design Patterns

In the current iteration we are implementing a microservices architecture where all ‘services’ run independently and connect to one another via hooks.

# References

* SQLite 3: <https://docs.python.org/3/library/sqlite3.html>
* Flask Login: <https://flask-login.readthedocs.io/en/latest/>

# Glossary

* ML: Machine Learning
* FaFI: Facial Finder