Software Requirements Specification

for

Object Detection and Classification API

Prepared by

<u>Group 10:</u>

Loveleen Amar (17100025)
Rajiv Nayan Choubey (17100041)
Sanchita Khare (17100047)

AUG, 2019 - NOV, 2019

Table of Contents

1. Introduction

1.1 Purpose

Object Detection can be used in many fields. It ranges from surveillance to medical purposes. But object detection requires high processing power and too many dependencies.The purpose of this project is to propose an object detection Application Programming Interface (API) which can be used by the clients to detect objects present in an image just by sending the image to a URL.

1.2 Document Conventions

YOLO	You Only Look Once
API	Application Programming Interface

1.3 Intended Audience and Reading Suggestions

This document is intended for software developer, web/app developer or any researcher who doesn't know about object detection but wants to get benefitted from it. Any internet user can also make use of this API using the HTML form. A mere user of this API should start from Section 3. The developers and any client should read the whole document for better understanding of the working of the API.

1.4 Product Scope

The purpose of the object detection API is to create a convenient and easy-to-use application for clients, trying to get benefit of object detection algorithms. This API can help millions of developers including the power of Object Detection into their projects/softwares.

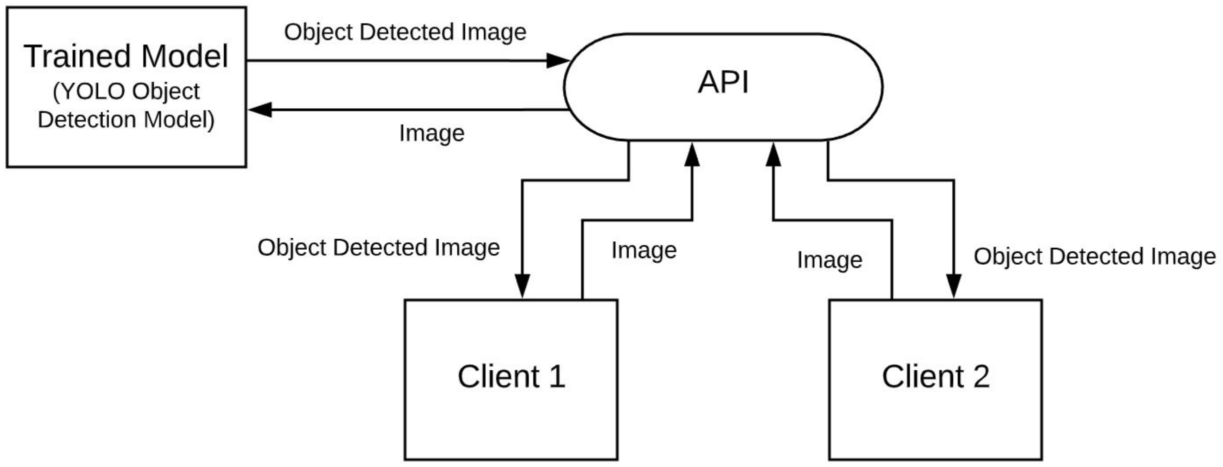
1.5 References

- *Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).*
- *https://docs.djangoproject.com/en/2.2/*
- *Viola, Paul, and Michael Jones. "Robust real-time object detection." International journal of computer vision 4.34-47 (2001):*

2. Overall Description

2.1 Product Perspective

Currently, most of the Object detection softwares do all the computations on the devices they're installed on which can be slow due to low hardware specifications. The product proposed in this document overcomes this problem by doing all the computation at the server side and in return provides the image with objects detected in only a few cycles of CPU.



2.2 Product Functions

- API accepts image of any extension and detects objects present in it.
- After detecting API, it returns the image with object detected in PNG format.
- All the computations are done at the server end.
- YOLO v3.0 algorithm is used for the Object Detection Task.

2.3 User Classes and Characteristics

LEVEL 1 user:
A first type of user of the API is a software developer who are building up big projects in which a part is of object detection , so the can integrate and object detection part using this. They should have the following technical skills:-

1. Generally has the skills and experience to design, construct, test, and maintain the software systems;
2. Understands processes and procedures to compile, link and load software for embedded systems;
3. Is proficient at writing software in the python language

LEVEL 2 user:
The second type of users for this API is a students who required not to do heavy backend work and can simply use this API by accessing it through URL.

LEVEL 3 user:
The third type of users for this API are typical internet users who only need to upload an image and wait for the results.

2.4 Operating Environment

The API could be accessed only via a URL, therefore the users can use the API through Mobile apps or web apps,by a simple Networking Program which could send data over the internet or even via internet browsers. The client should be connected to the internet and has rights to send data over the internet to APIs.

2.5 Design and Implementation Constraints

- At the server end, high-end GPUs and CPUs are required for scanning high quality images.
- At the client end, high speed internet is required to send high quality images

2.6 Assumptions and Dependencies

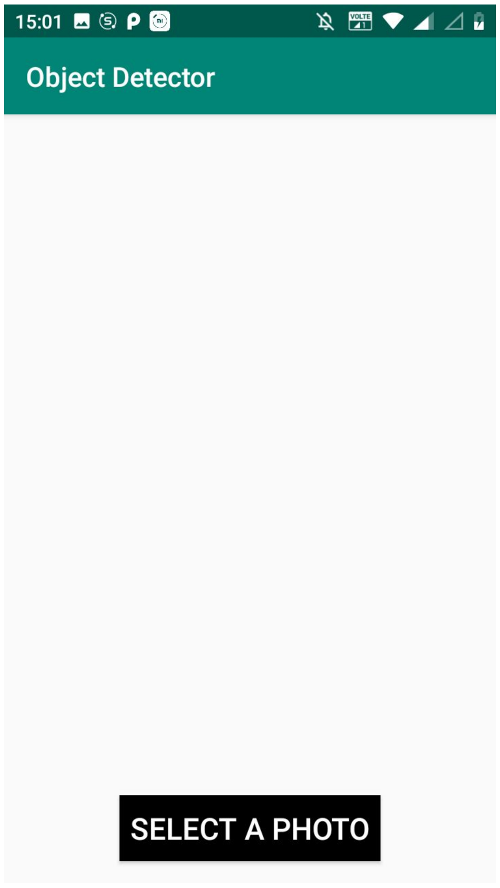
API response takes time as it involves lots of computations at server-end. Thus, it is assumed that the clients/developers are putting enough time limit for the response to receive. There is no dependency at the user end other than internet connection.

3. External Interface Requirements

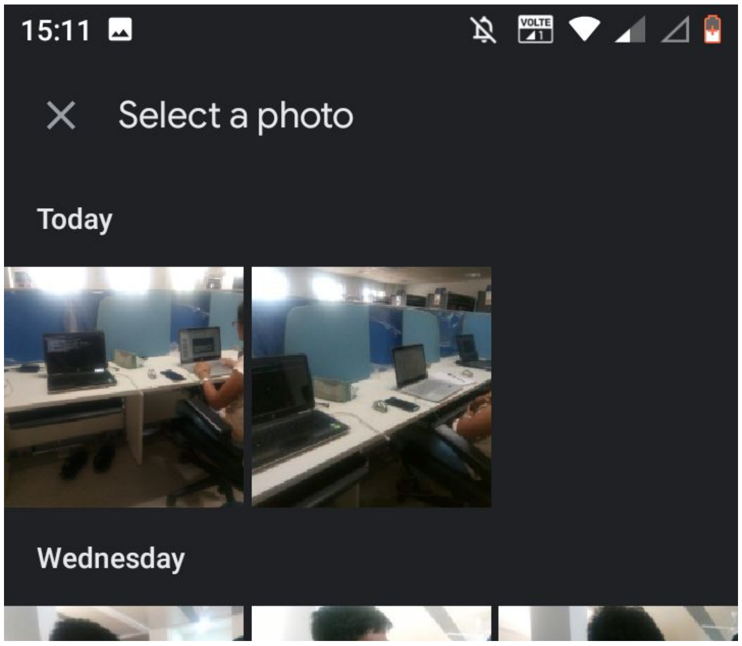
3.1 User Interfaces

API can be used for many applications. One of the applications of this API is an android app which sends an image from phone gallery to the server via API call and receives an image with all the objects detected.

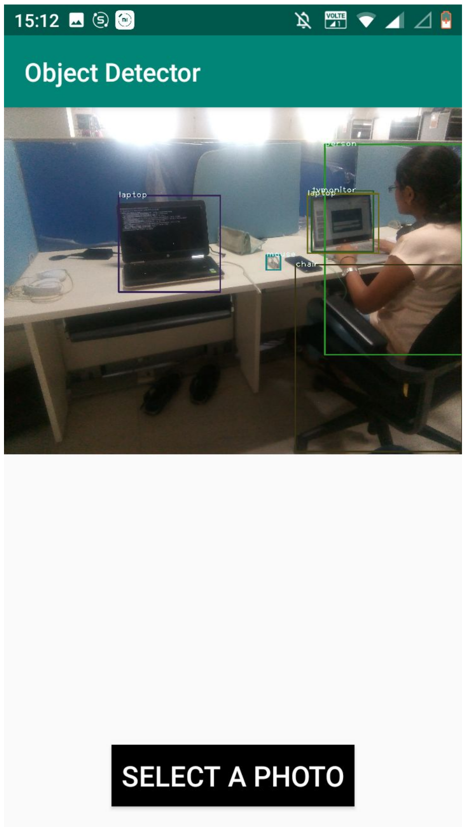
1. Install the android app and select image from gallery.



2. Select a photo from the gallery.



3. Send and wait for the response image.



3.2 Hardware Interfaces

- Any machine with Network Interface Card and Ethernet drivers to connect to the internet.
- Programmable devices to send the image to API dynamically.
- Devices able to process graphical matrix(image) eg. Raspberry Pi, Arduino etc.

3.3 Software Interfaces

This API requires internet drivers at client-side and any programming language which supports network programming. However, the application of this API can have vivid requirements. For example, the android app requires Android OS, access to camera and gallery & access to use the internet.

3.4 Communications Interfaces

Internet connection is required for frequent upload and download of image stream bytes to/from the server. Every request requires the client to be connected to the internet to make an API call.

4. System Features

This section gives a detailed description of the various features of the API.

4.1 Object Detection over images.

4.1.1 Description and Priority

The main task of the API is to detect the objects present in the image and plot the boundary box around them. When a client sends any image to the server via the API, the server calls YOLO algorithm on the image to detect objects. It plots the boundary boxes and then revert back the new image with detected objects to the client.

Priority: High

4.1.2 Stimulus/Response Sequences

1. Client send an image to the API.
2. API, after authenticating the validity of image, sends it to server.
3. Returns the new image to the client.

4.1.3 Functional Requirements

Sending data:

A valid image file should be sent to the server at a sufficient interval depending on the image size. Invalid images or any other file types will be discarded.

Invalid Image File:

In case of invalid image file, the server will respond with an error message stating "Invalid Image Type".

5. Other Nonfunctional Requirements

5.1 Security Requirements

There is no privacy issue as the images sent to the server are not stored anywhere in the database. Also, the API requires data to be sent over https connection thus it is secured from man-in-the-middle attack. However, other applications of this API could pose security threats which the API is not responsible for.

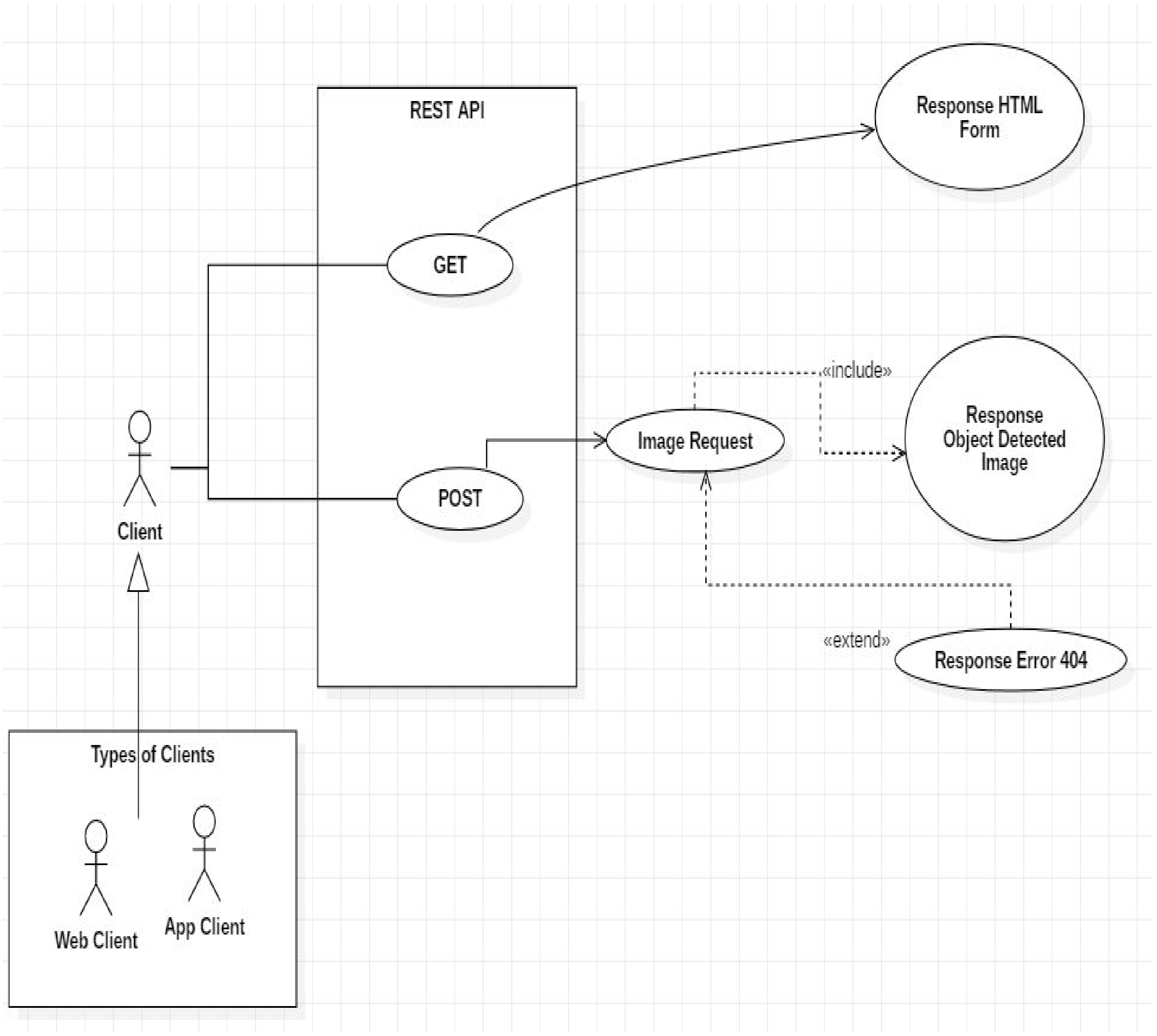
Appendix A: Glossary

YOLO v3.0: It is deep learning based algorithm which are used to detect objects in image. v3.0 represents the 3rd improvement in the algorithm.

HTTPS: *HTTPS stands for Hypertext Transfer Protocol Secure. It is the protocol where encrypted HTTP data is transferred over a secure connection.*

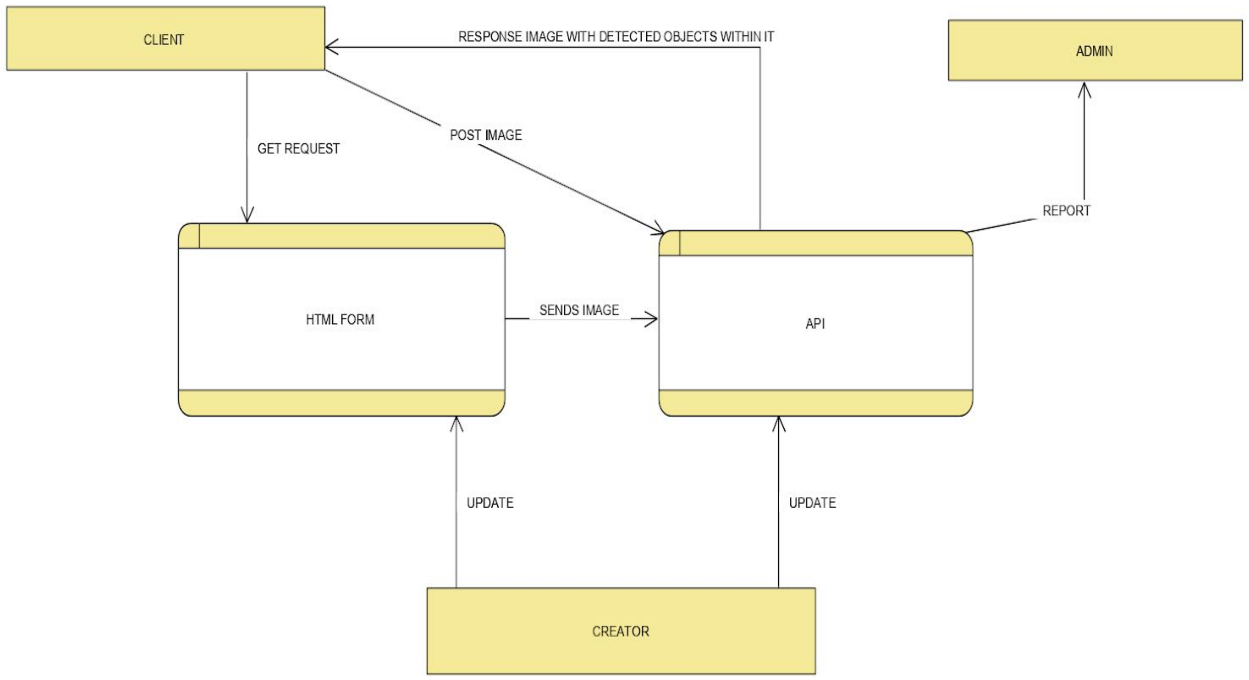
Appendix B: Analysis Models

1. **Use-Case Diagram**

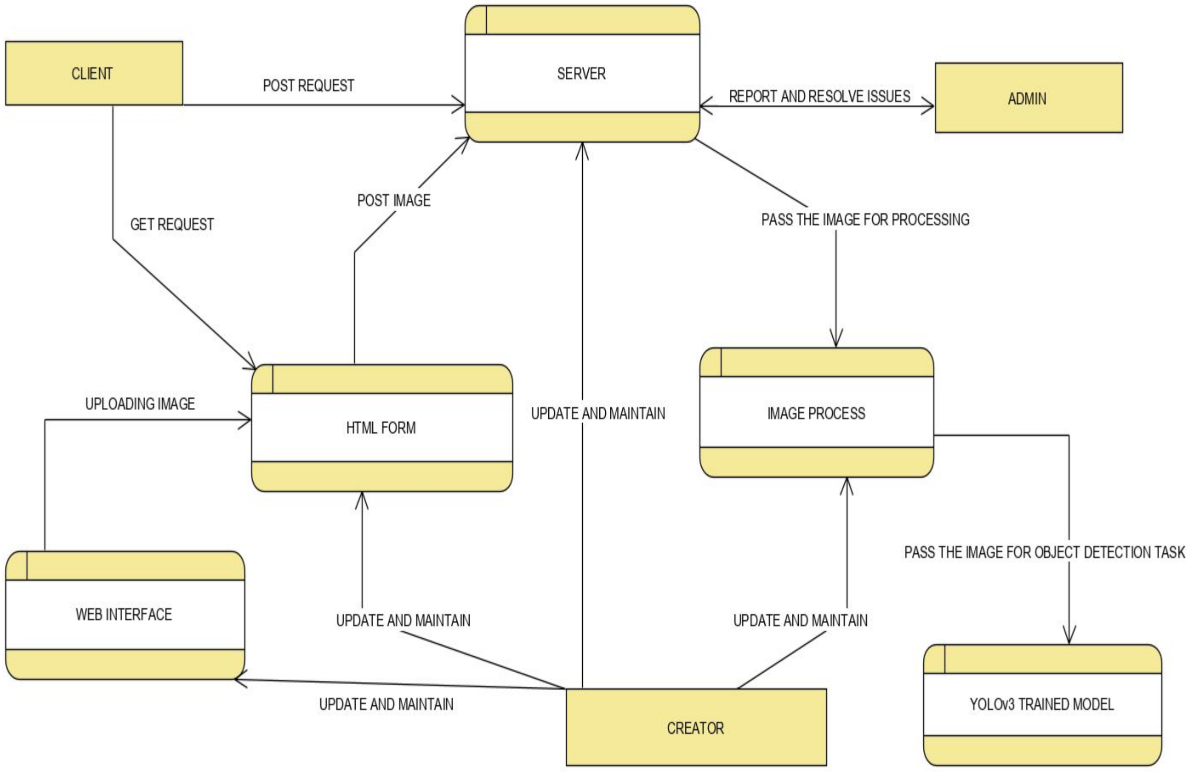


2. Data Flow Diagram (DFD)

a. DFD Level 0



b. DFD Level 1



c. DFD Level 2

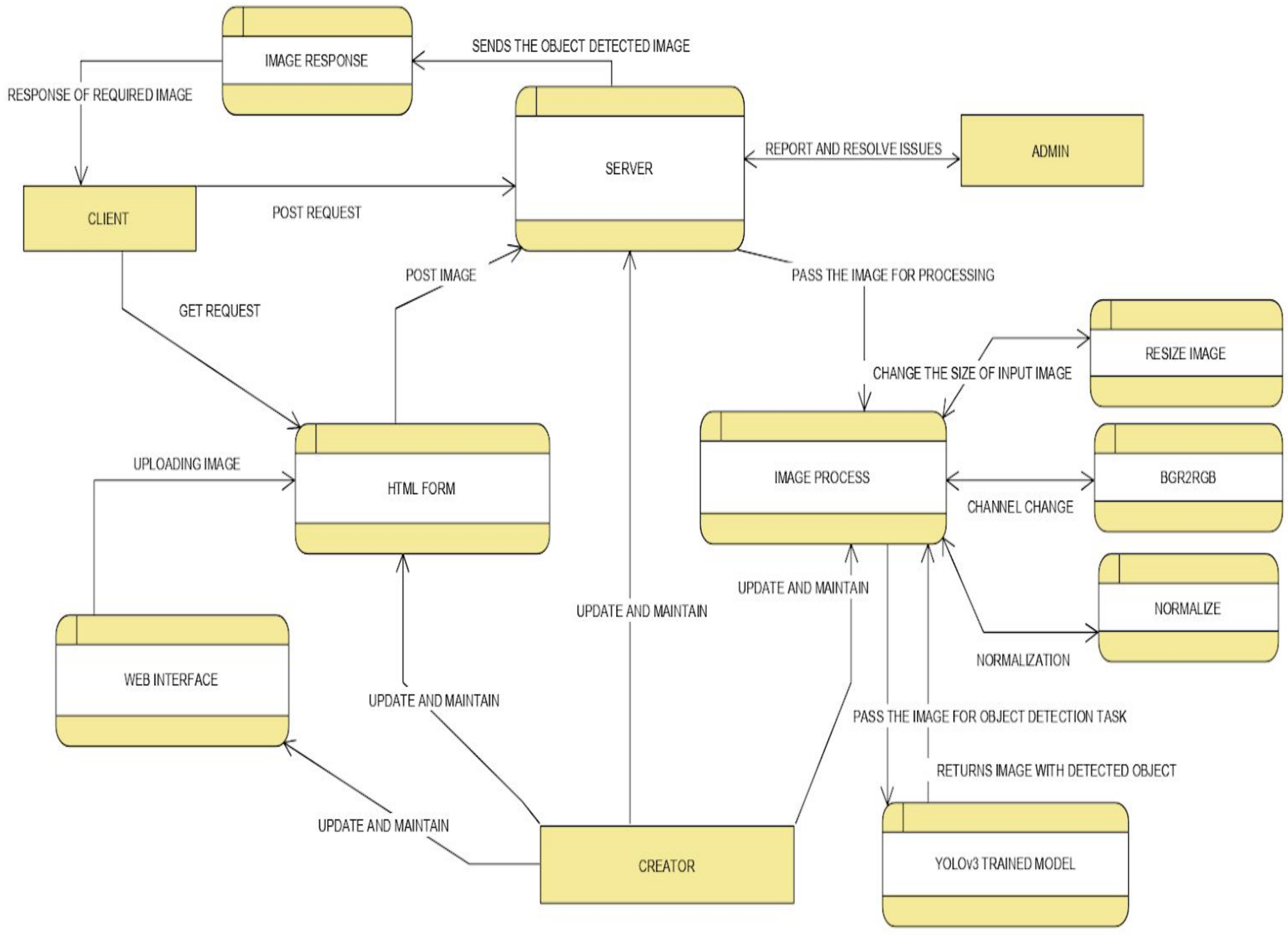
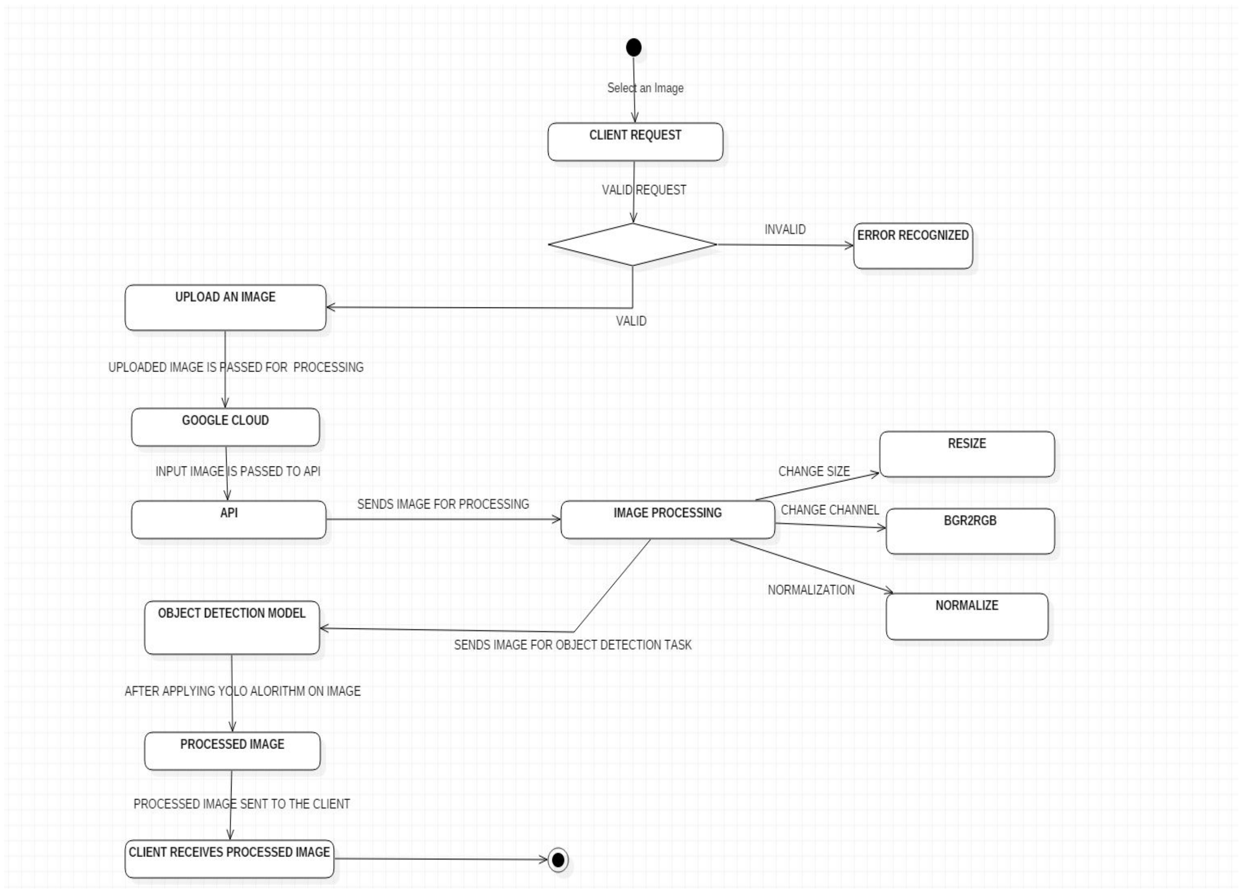
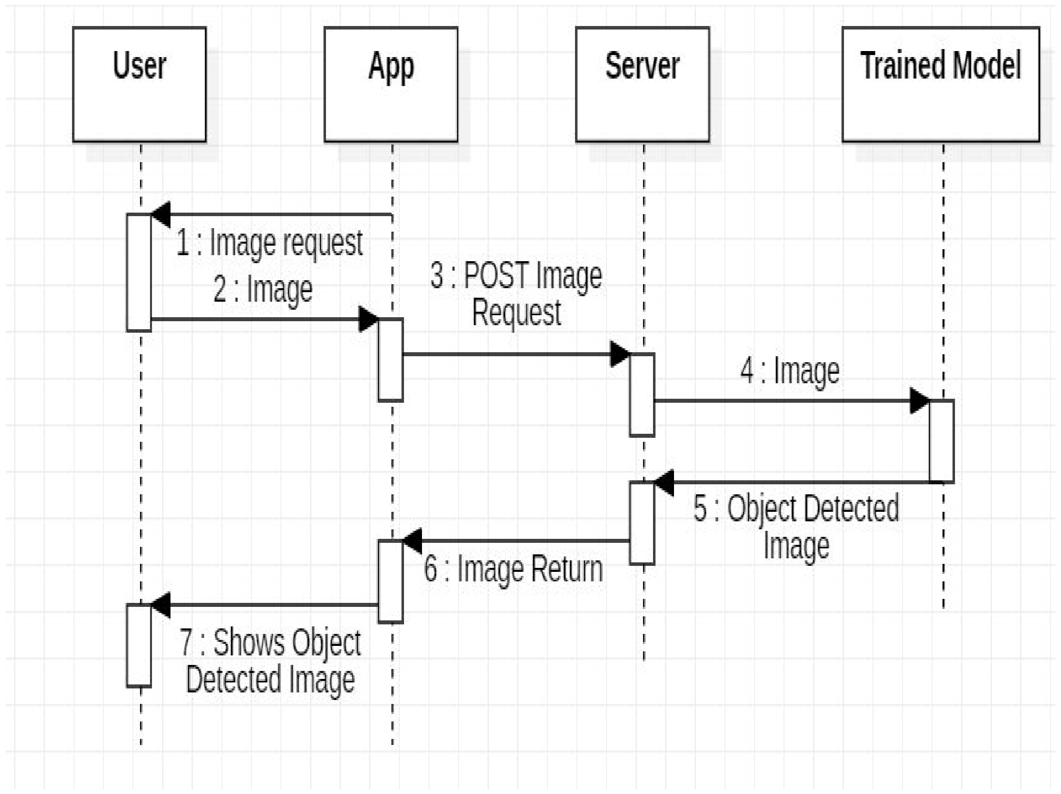


IMAGE RESPONSE
SENDS THE OBJECT DETECTED IMAGE
RESPONSE OF REQUIRED IMAGE
SERVER
REPORT AND RESOLVE ISSUES
ADMIN
CLIENT
POST REQUEST
POST IMAGE
PASS THE IMAGE FOR PROCESSING
GET REQUEST
RESIZE IMAGE
CHANGE THE SIZE OF INPUT IMAGE
UPLOADING IMAGE
HTML FORM
IMAGE PROCESS
BGR2RGB
CHANNEL CHANGE
NORMALIZE
UPDATE AND MAINTAIN
UPDATE AND MAINTAIN
WEB INTERFACE
NORMALIZATION
UPDATE AND MAINTAIN
PASS THE IMAGE FOR OBJECT DETECTION TASK
RETURNS IMAGE WITH DETECTED OBJECT
UPDATE AND MAINTAIN
CREATOR
YOLOv3 TRAINED MODEL

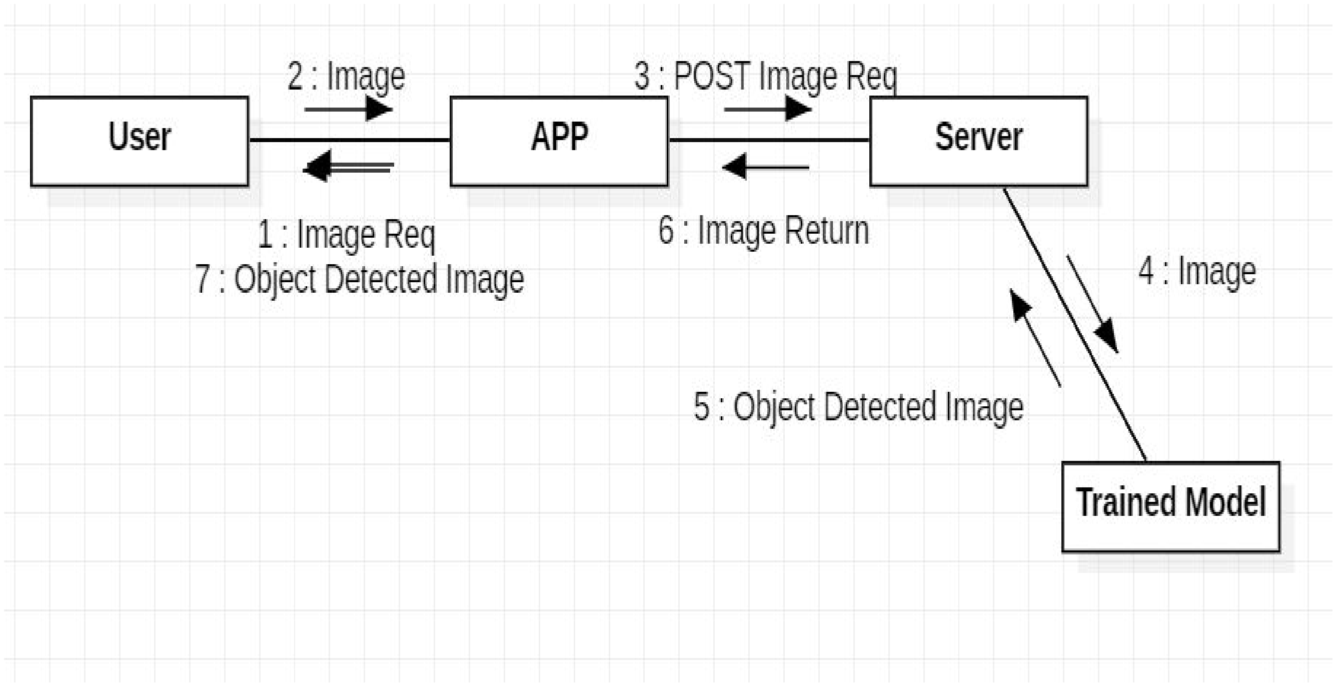
3. State Diagram


Select an Image
CLIENT REQUEST
VALID REQUEST
INVALID
ERROR RECOGNIZED
UPLOAD AN IMAGE
VALID
UPLOADED IMAGE IS PASSED FOR PROCESSING
GOOGLE CLOUD
INPUT IMAGE IS PASSED TO API
API
SENDS IMAGE FOR PROCESSING
IMAGE PROCESSING
CHANGE SIZE
RESIZE
CHANGE CHANNEL
BGR2RGB
NORMALIZATION
NORMALIZE
OBJECT DETECTION MODEL
SENDS IMAGE FOR OBJECT DETECTION TASK
AFTER APPLYING YOLO ALORITHM ON IMAGE
PROCESSED IMAGE
PROCESSED IMAGE SENT TO THE CLIENT
CLIENT RECEIVES PROCESSED IMAGE

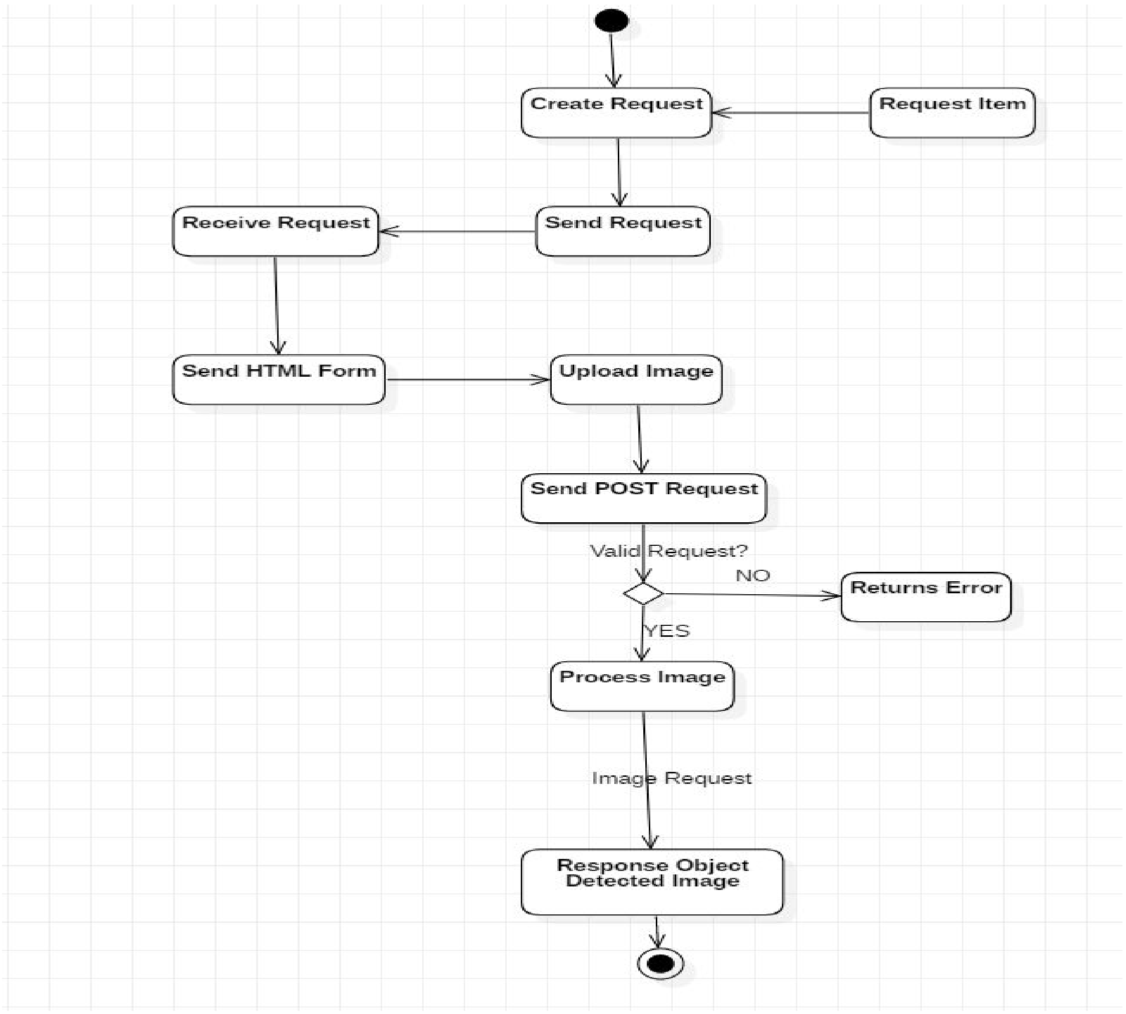
4. Sequence Diagram



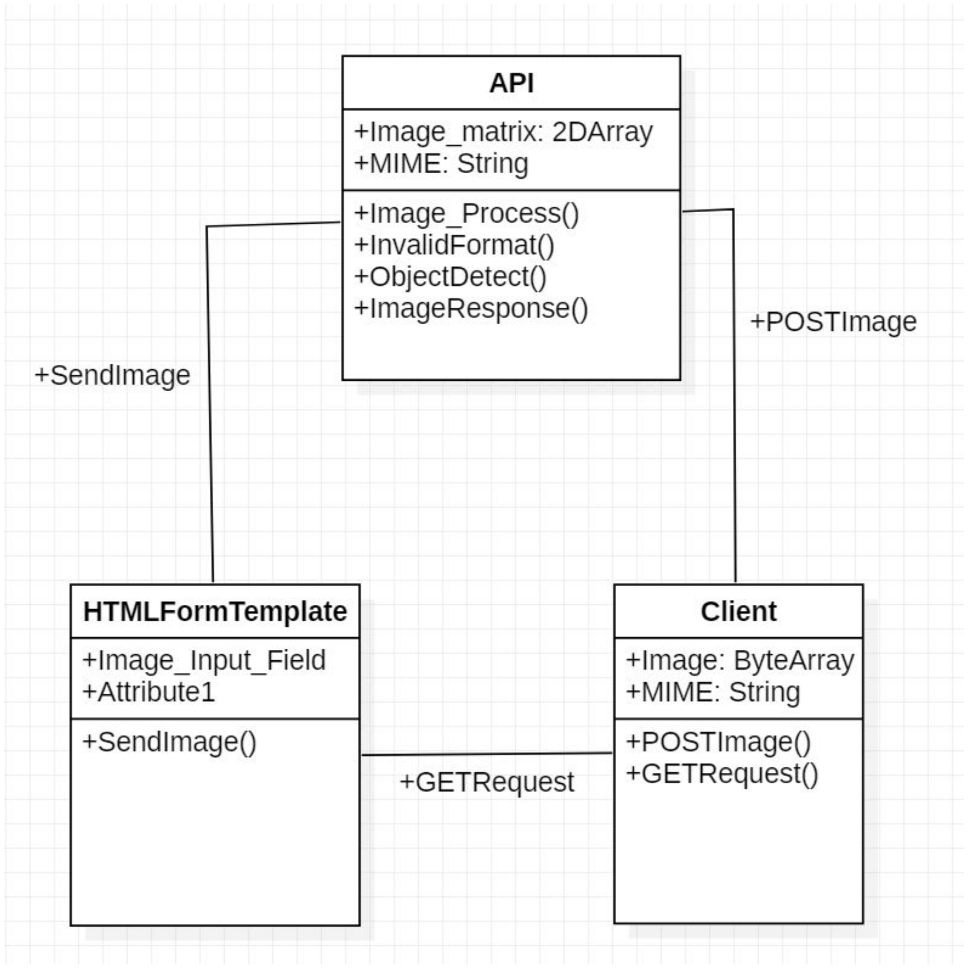
5. Collaboration Diagram



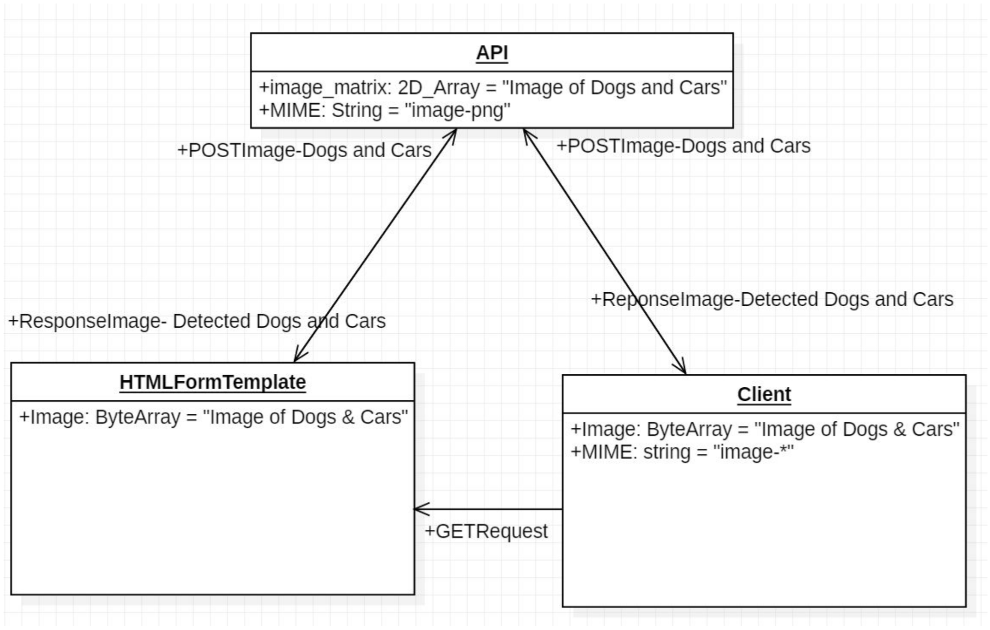
6. Activity Diagram



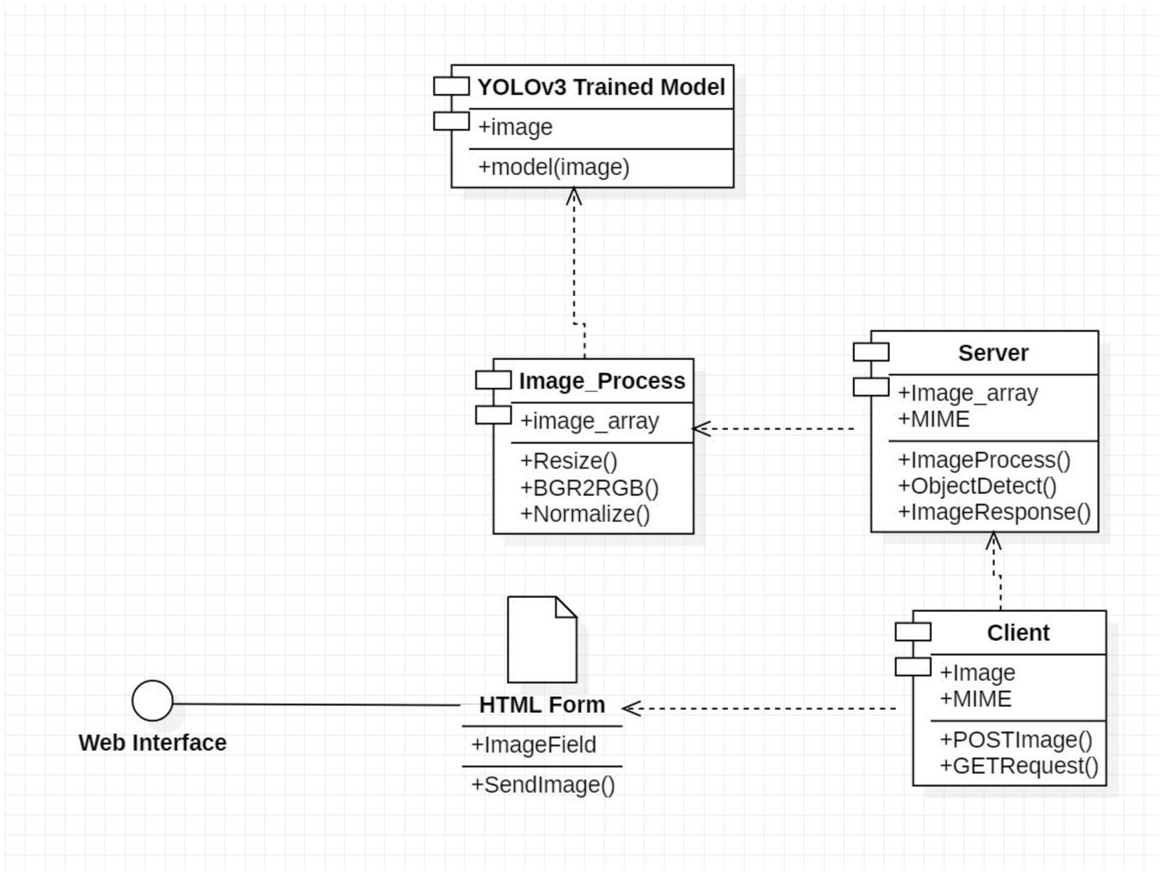
7. Class Diagram



8. Object Diagram


API
+image_matrix: 2D_Array = "Image of Dogs and Cars"
+MIME: String = "image-png"
+POSTImage-Dogs and Cars
+POSTImage-Dogs and Cars
+ResponseImage- Detected Dogs and Cars
+ReponseImage-Detected Dogs and Cars
HTMLFormTemplate
+Image: ByteArray = "Image of Dogs & Cars"
Client
+Image: ByteArray = "Image of Dogs & Cars"
+MIME: string = "image-*"
+GETRequest

9. Component Diagram


YOLOv3 Trained Model
+image
+model(image)
Image_Process
+image_array
+Resize()
+BGR2RGB()
+Normalize()
Server
+Image_array
+MIME
+ImageProcess()
+ObjectDetect()
+ImageResponse()
Client
+Image
+MIME
+POSTImage()
+GETRequest()
HTML Form
+ImageField
+SendImage()
Web Interface