Image’s Objects Detection

Chuong Nguyen

CST-451 Capstone Project Final Architecture & Design

Grand Canyon University

Instructor: Professor Mark Reha

Revision:

Date: 11/11/2018

**ABSTRACT**

The human can recognize the objects easily in the image. With the development of photography technology, there are massive number of images need to process. It is necessary to have algorithms to analyze data on the images. Object detection is an important part of images processing. It has been researched, developed and used in various industries. Object detection is applying widely in manufacture, traffic control, security ...

My application creates an application to detect objects in images and videos. From application, user input image and the object detection algorithm will return the objects in the image. All the images and objects will be saved into the database. Then users can search the images in the database. The application let the user input multiple images in the same time and find the images which contain specific input objects. The application can detect objects from stream video, and capture the stream video with its object.

Users can use this application in traffic monitoring. For example, the cameras at an intersection take a lot of pictures. An officer wants to find pictures of cars that passed through the red light. He can input all the images that were taking that day and search the images that contain cars and red light. After that, the officer can decide if he needs to send a ticket to the driver.

**Design Planning Summary**

The solution of the application is to create a web server using Python and Flask. JavaScript, HTML is used for the front-end. A TensorFlow Object Detection API which uses MobileNet Object Detection Model Architecture is called from Flask server to detect objects in the image and video. MySQL is used for the database.

|  |
| --- |
| History and Signoff Sheet |

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 10/15/2018 | Chuong Nguyen | Design Phase v1.0: Rough Draft Design |
| 12/15/2018 | Chuong Nguyen | Design Phase v2.0: Final Design |
| 02/10/2019 | Chuong Nguyen | Project Development and Implementation |
| 03/04/2019 | Chuong Nguyen | Testing Phase. |
| 04/10/2019 | Chuong Nguyen | Project Completion, |

**Change Record**

|  |
| --- |
| Project Development and Implementation:  The project does not use the MVC architecture due to the Flask framework’s architecture.  The project cannot implement on AWS due to TensorFlow environment and video stream detection. The project is deployed and demo on localhost.  Add function let user search from multiple images. Result are only the images which contain input objects from user.  Add function let user search from multiple images. Result are only the images which contain input objects from user. |
|  |

|  |
| --- |
| **Overall Instructor Feedback/Comments**  Create functions to search images contains specific objects from multiple images  Draw bounding box for the object in the image. And let the users choose which object they want to draw. |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

**Integrated Instructor Feedback into Project Documentation**

Yes  No

**TABLE OF CONTENTS**

Design Overview 6

Detailed High-Level Solution Design 6

Detailed Technical Design 12

Appendix A – Technical Issue and Risk Log 35

Appendix B – References 36

Appendix C – External Resources 38

# **Design Overview**

This is the technical design for the Object Detection Web application. The design provides the implementation of a web application to detect object from image. The design deliveries the technical approach, design decisions, proof of concept, logical design, system design, database design, class diagram, API design, UI wireframe, and other documents that will support the developer implement the application. The design supports these following requirements:

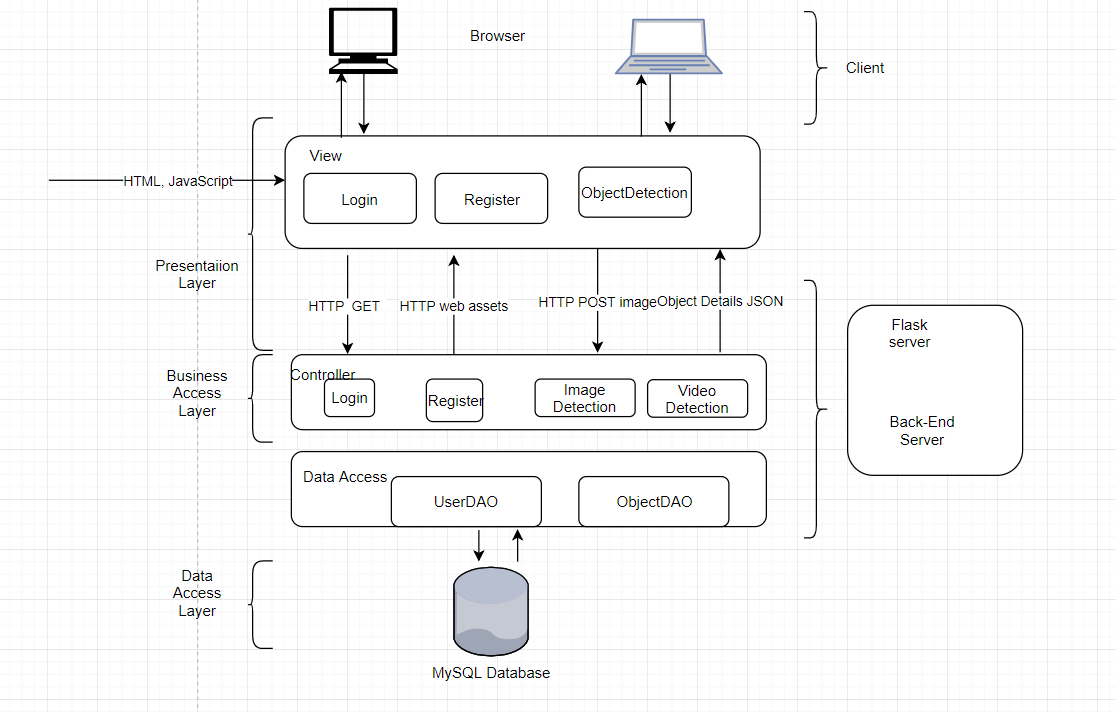
* The application will implement user registration module, login module, and object detection module.
* The application separate presentation layer, business layer and database layer.
* The application separates the view (jinja2 templates) and the controllers.
* The application uses TensorFlow Object Detection API to detect image’s objects.
* The application use relational database MySQL.
* The application is deployed on Flask server and localhost.

# **Detailed High-Level Solution Design**

## **Detailed overview.**

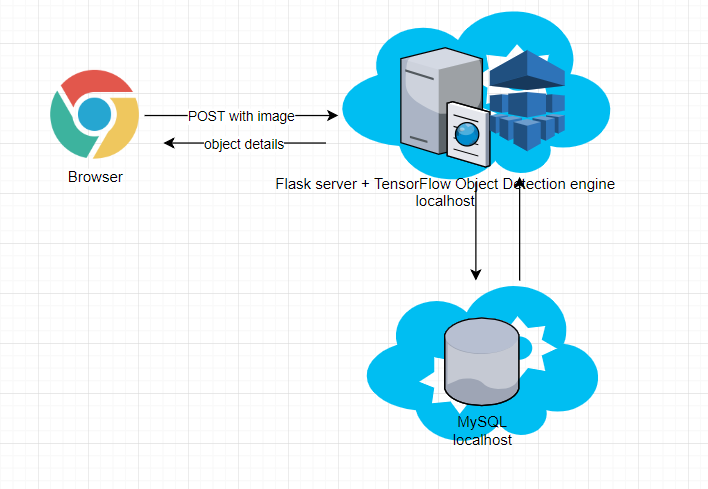
**Logical Solution Design:**

The application leverage the N- Layer architect design pattern. The application is divided into different layers based on the functionality: Data Access, Business Access, and Presentation layers. MySQL database is used to save data, and the database is accessed through the Data Access layer. The business layer contains all the logic of the application. And the presentation layer provides the application’s access for users. The solution divides the application to Controller (Python site) and View (Jinja2 templates). The server side includes a Flask server and TensorFlow Object Detection API.



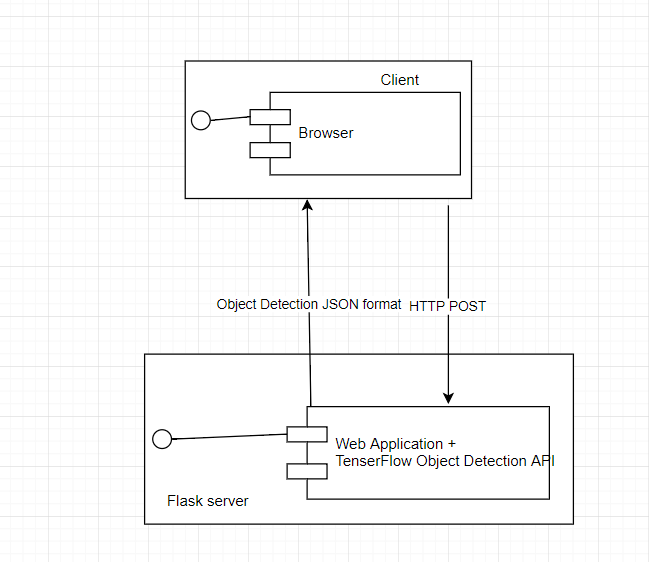
**Physical Solution Design:**

The Flask server, TensorFlow Object Detection API, and MySQL database are hosted and demo on localhost (Windows OS CPU 2.2GHZ x 8GB GPU x 16 GB RAM x 16 GB Storage).

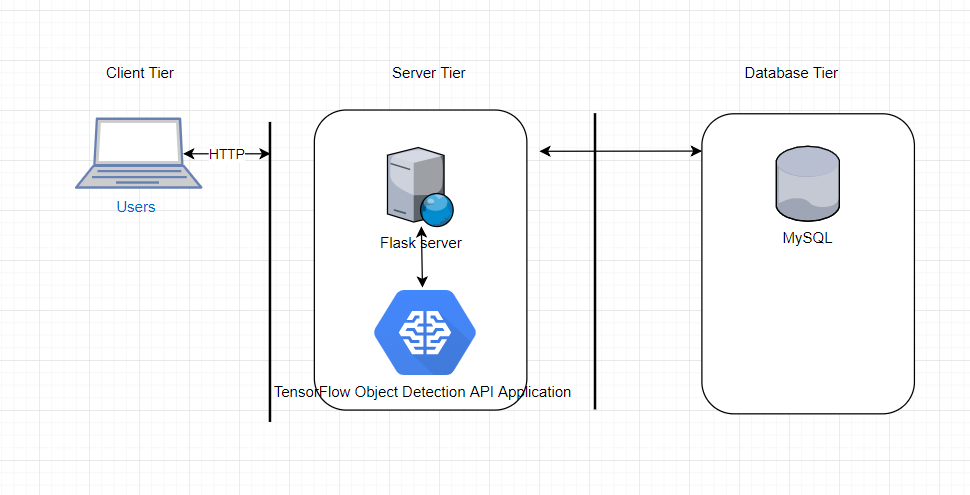


**System Design:**

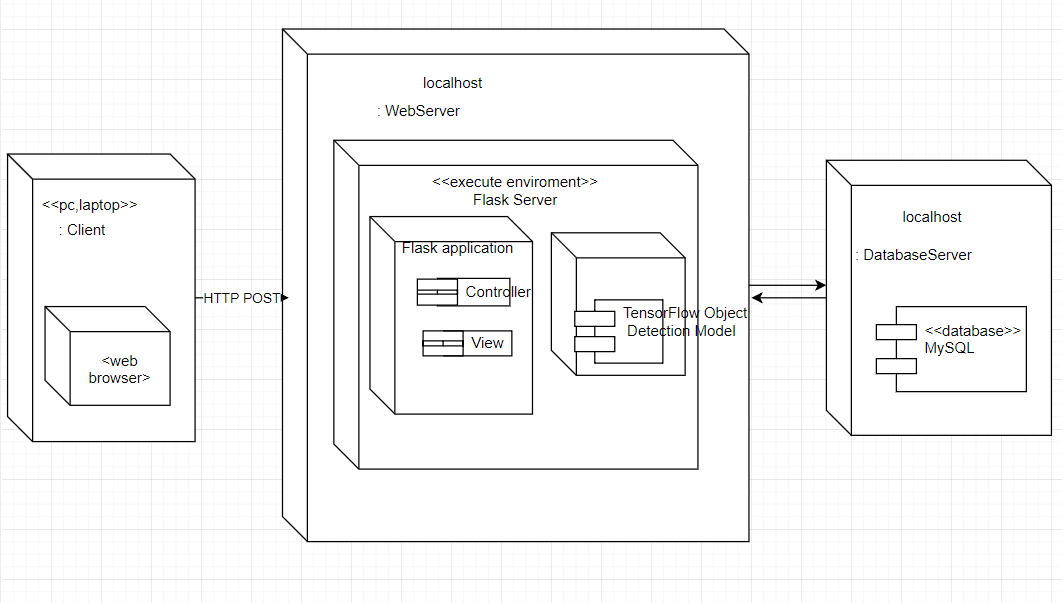
UML Component Diagram for Logical Design



UML Component Diagram for Physical Design



UML Deployment Diagram



## **Proof of concepts:**

Application’s Requirement: User input images, the application will return all instances of objects. Detect object directly on the stream video. Search images which contain specific objects.

Input: images or stream video

Output: instances of objects from the image or in the video

Ideas: Create a web application to let the user input image or stream a video and return all instances of objects such as people, cars, animal…from the image or video. All the images will be saved in the database so that users can search pictures with a specific object from them.

Problem: The main problem to make the application running is chosen the technology that can combine between the web application and machine learning for image object detection

Decision:

* Use Python and TensorFlow to build a TensorFlow Object Detection application to produce an object detection API and create a web application that uses the API
* A simple web application using HTML and JavaScript’s will let the user input and execute image detection, and search image from the web browser.
* Create Flask web server, and call TensorFlow Object Detection API to detect the objects from the image. TensorFlow is an open source machine learning library developed by Google. A python application running on Flask server will provide an Object Detection API service for object detection. Image after detection returns the instances of objects and their location in the image. The object’s detail is wrapped up in a JSON object and store into a MySQL database.
* Flask server is a micro web framework written with Python. Therefore, it is more convenient to call Object Detection API from there. This solution comes to a server that can execute object detection on itself. It is more effective than running the object detection from another machine.
* A MySQL database is used to store object detected results.
* Pre-trained object detection models: TensorFlow will provide models to create Object Detection API. In this project, a PASCAL VOC dataset and MobileNet SSD (Single Shot Detection) algorithm is used to create the Object Detection API (it is described detailly in 2.6).
* The application will be hosted locally.

|  |  |
| --- | --- |
| Proof of Concepts | |
| **Description** | **Rational** |
| 1. TensorFlow | To create object detection application |
| 2. PASCAL VOC 2012 | To provide training set of labelled images. There are twenty object classes that have been selected in the dataset are:   * *Person:* person * *Animal:* bird, cat, cow, dog, horse, sheep * *Vehicle:* aero plane, bicycle, boat, bus, car, motorbike, train * *Indoor:* bottle, chair, dining table, potted plant, sofa, tv/monitor |
| 3. MobileNet SSD (Single Shot Detection) | The algorithm to detect object from image |
| 4. Flask | to run the application. In this project, Flask server is deployed on Azure Virtual Machine |
| 5. Python application to detect objects from image and return object detail in JSON format | In this project scope, python application is deployed on local computer |
| 6. MySQL database | To store user information, Image and it’s object detection |
| 7. Host application and database | User localhost. |

,

|  |
| --- |
| Hardware and Software Technologies |
| 1 – MySQL 5.6: open-source relational database management system. It I GPL(version 2) license |
| 2 – Python 3.6: developed y Python Software Foundation |
| 3 – Flask 1.0.2: microframework for Python. It is BSD licensed. |
| 4 – TensorFlow 1.10: open source software library for high performance numerical computation. Originally developed by researchers and engineers from the Google Brain team within Google’s AI organization |
| 5 – PASCAL VOC 2012 dataset from PASCAL VOC project. |
| 6 – Windows 10 2.2 GHZ x 16GB RAM x 8GBGPU x 250 GB SSD |

# **Detailed Technical Design**

**General Technical Approach:**

Create a web application which user can use as an online tool to detect an object from images. Image after detection will return a list of objects. The application will be designed to leverage controller and view in a Flask framework. A Python application uses a TensorFlow Object Detection model to provides an API to the web application to detect images’ objects. A MySQL database is used store object detection results.

### Key Technical Design Decisions:

Embedding: The application is required TensorFlow Object Detection API. A Python application using TensorFlow will be running on the server to create a TensorFlow Object Detection API. This application needed a high-performance machine to perform training and testing Object Detection model. In this project scope, a computer with GPU is used to host the Flask server and run the TensorFlow Object Detection API.

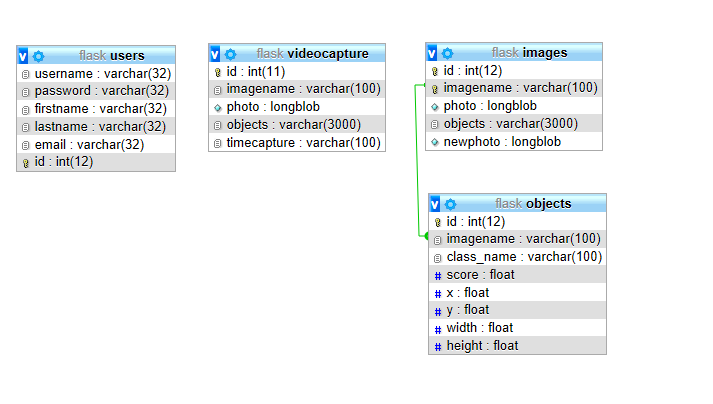
Back End Service: The TensorFlow Object Detection application is required to process and detect objects in the image; a back-end service using Flask will consume TensorFlow Object Detection API and get the detail of the objects to belong to the image to store in the database. The MySQL database is used for the project.

Front End Web Application: A front-end web application using HTML, JavaScript to implement a detect image’s object application. The application will leverage some common functions and display the object detection result, as well as display the images which contain specific objects when user look for it

TensorFlow Object Detection API: TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. TensorFlow provides several object detection models (pre-trained classifiers with specific neural network architectures) . In this project, a MobileNet SSD model is used to train the dataset and export the trained model that can use to detect the image’s object.

### Database ER Diagram:

### There are two tables in the database of this application. Table ‘user’ is used to store user information. Table ‘images’ is used to store image information. The table ‘object’ is used to store objects result .There are one to many relationships between two tables ‘images’ and ‘objects’. Table ‘videocapture’ is used to store videocapture data.



**Database DDL Scripts:**



**Flow Charts/Process Flows:**

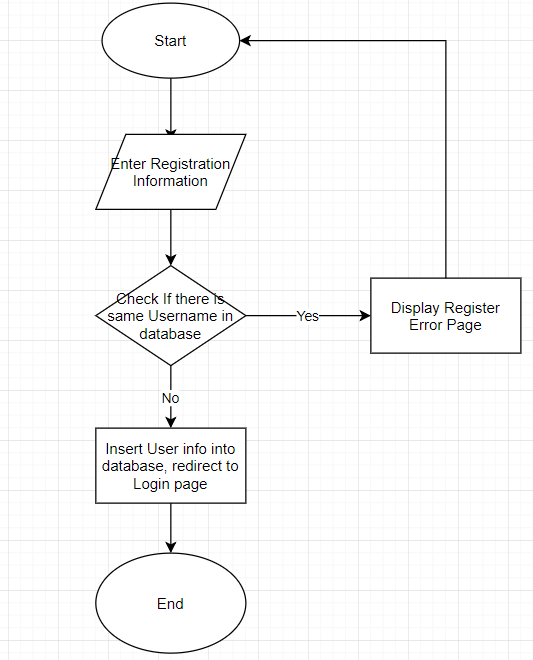


Figure 1 Flowchart of Registration functional

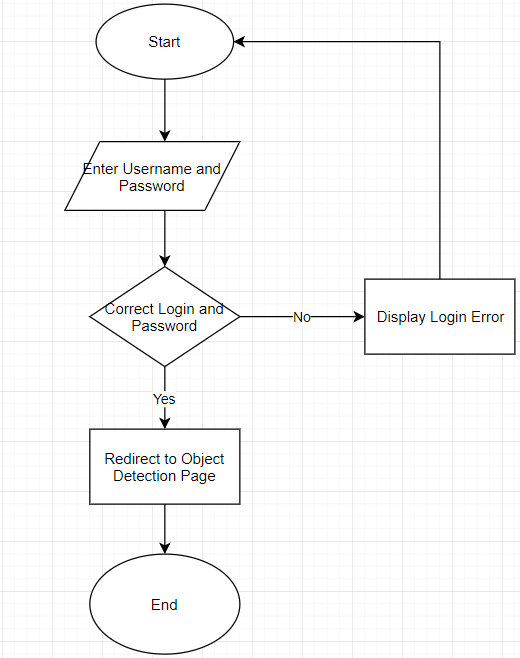


Figure 2 Flowchart of Login functional

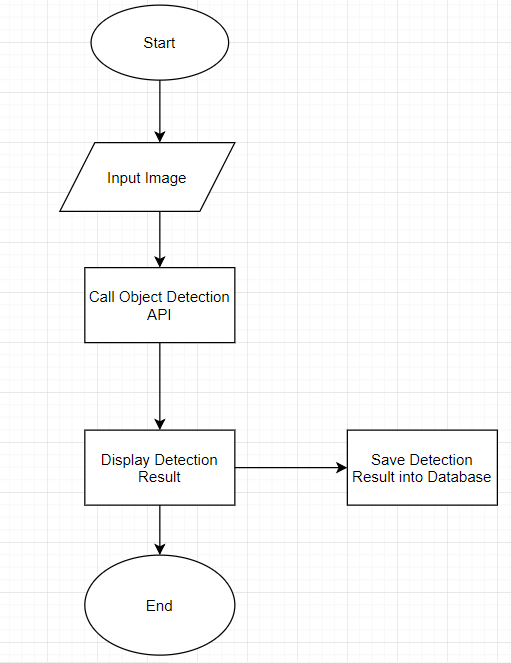


Figure 3 Flowchart of Object Detection functional

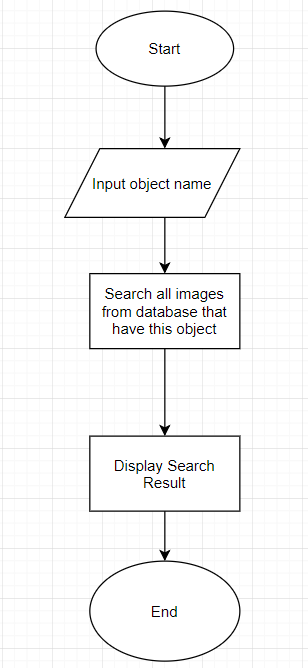


Figure 4 Flowchart of Search specific object from images

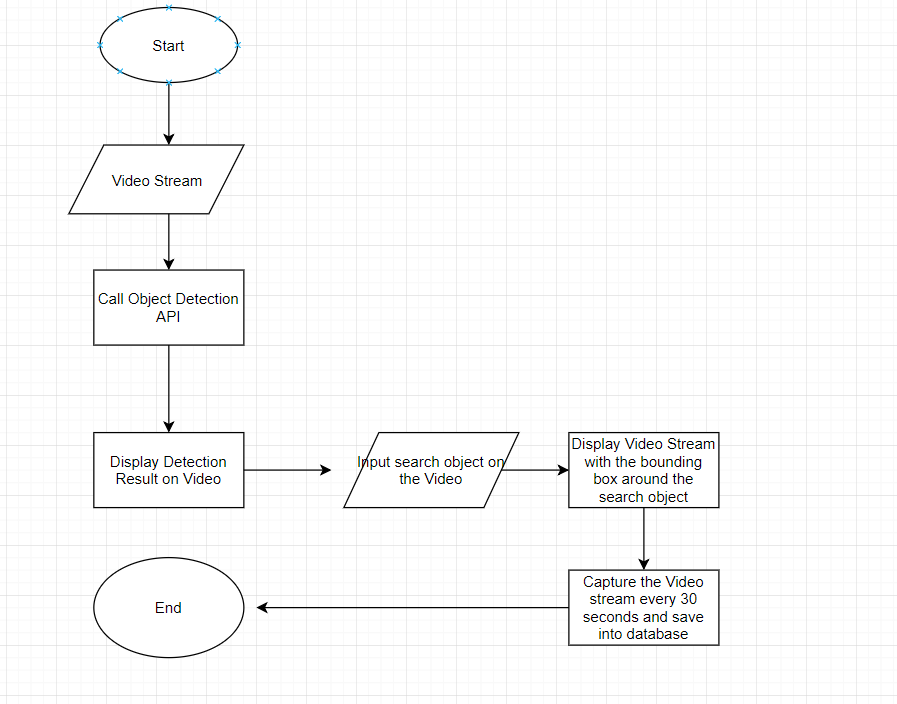
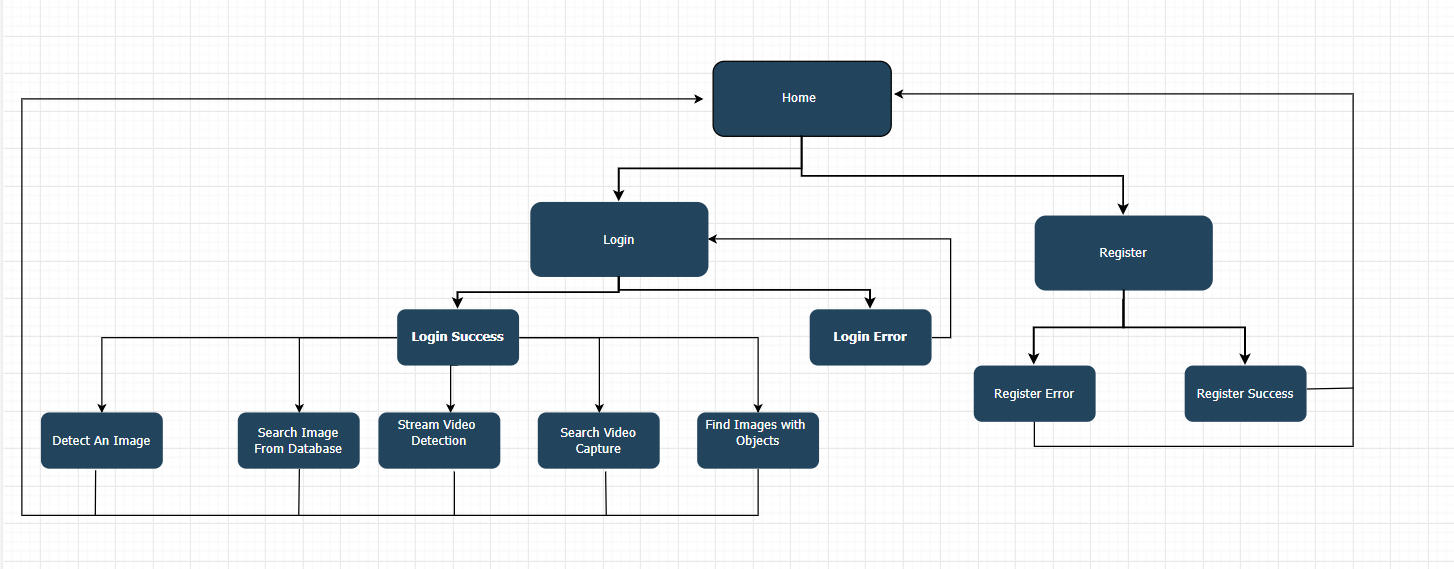


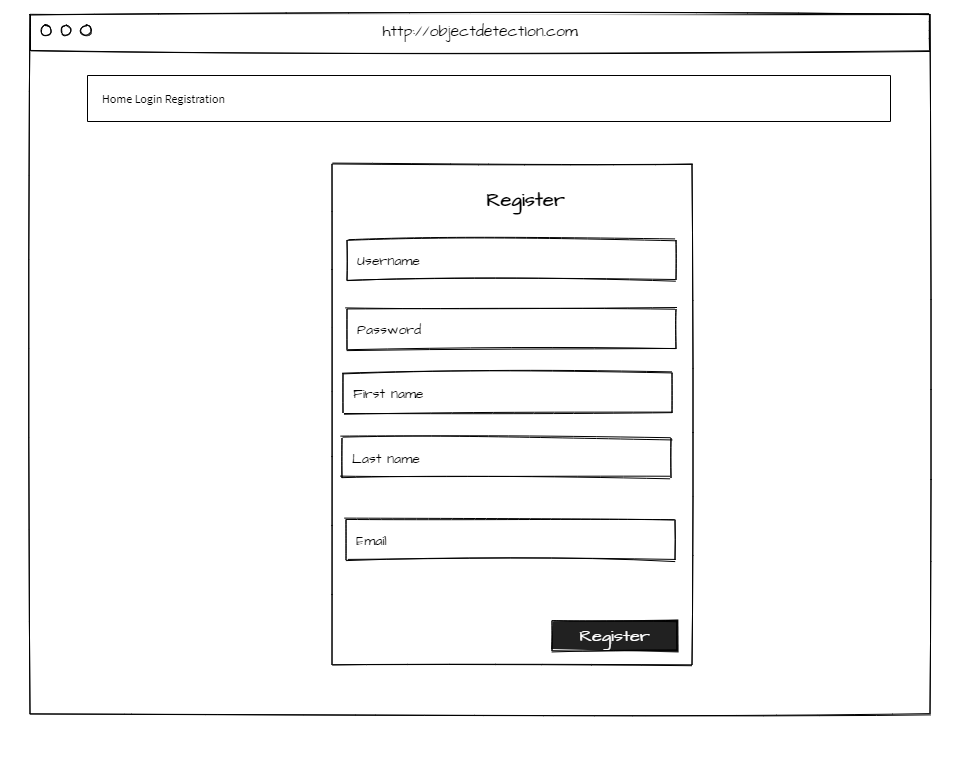
Figure 5 Flowchart of Stream Vide detection

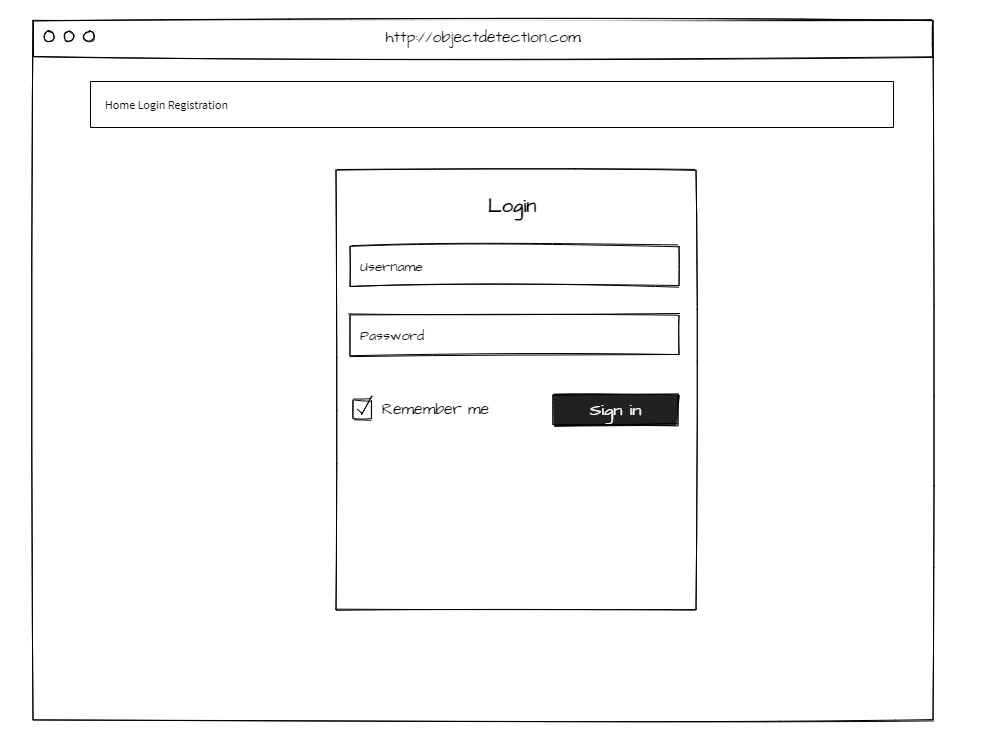
**Sitemap Diagram:**

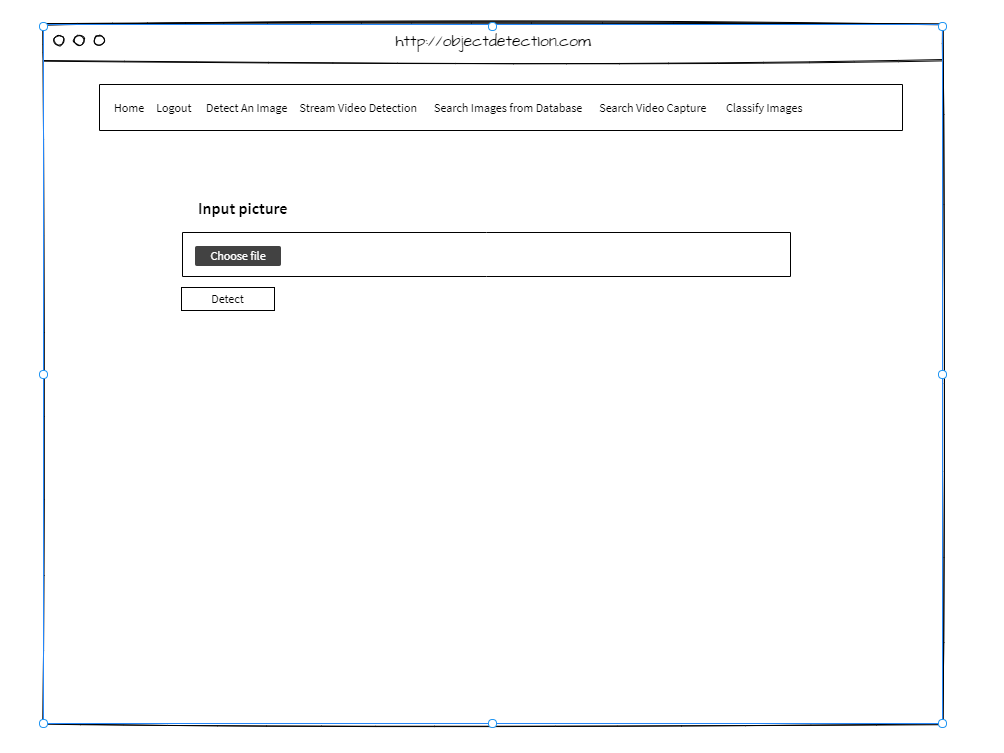


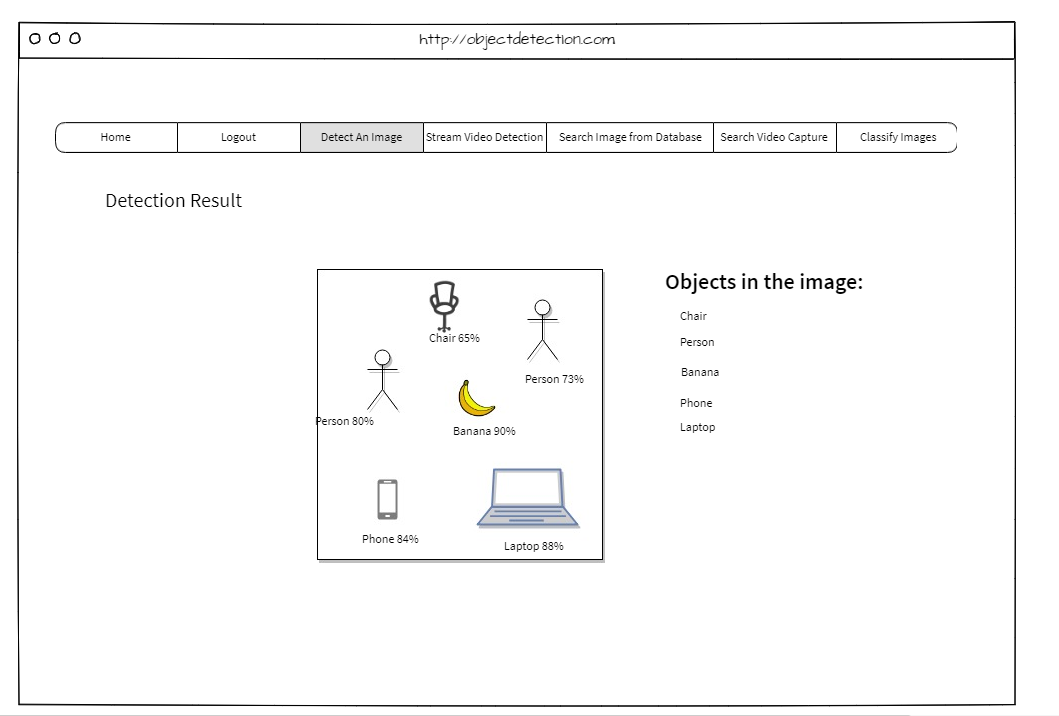
Sitemap of application

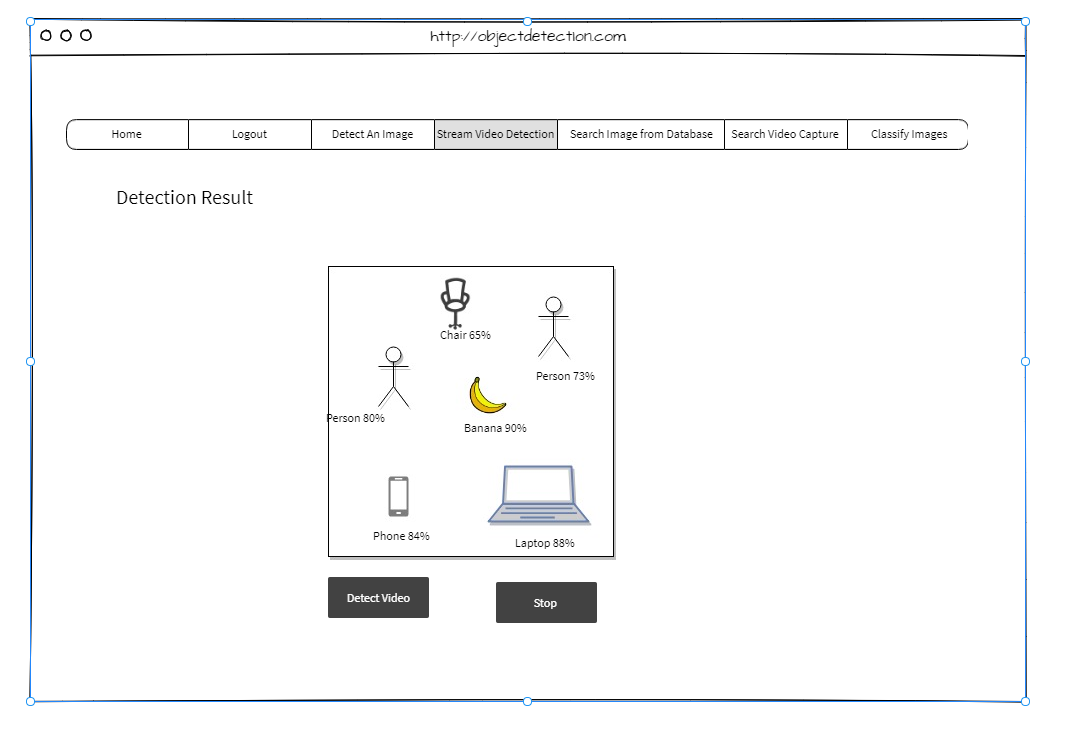
**User Interface Diagrams:**



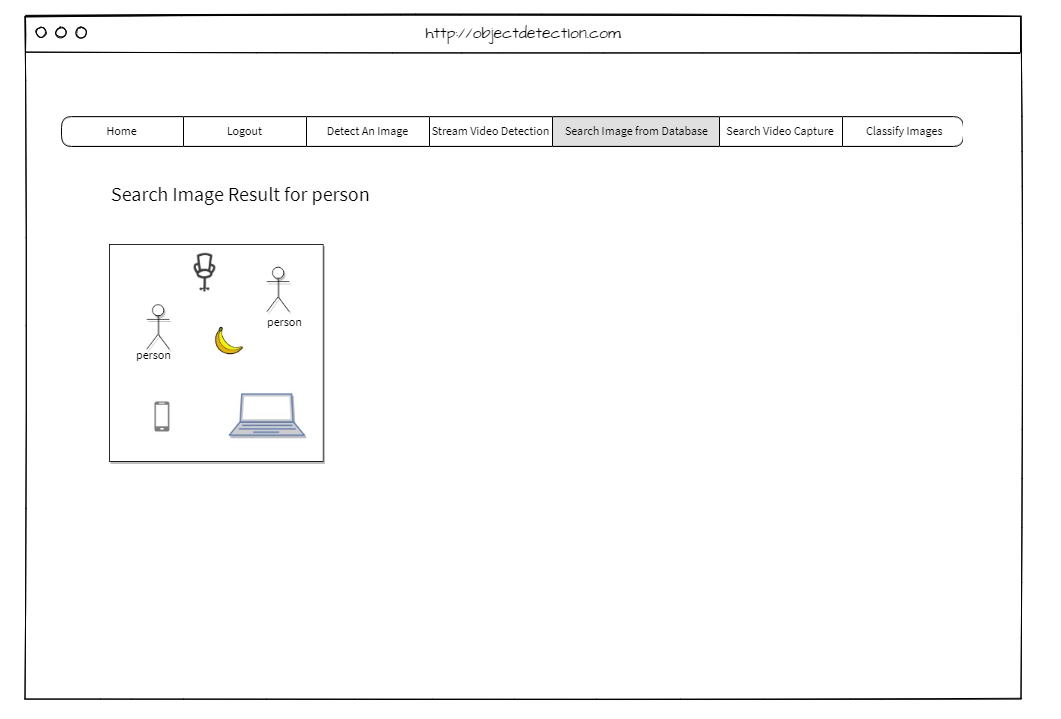


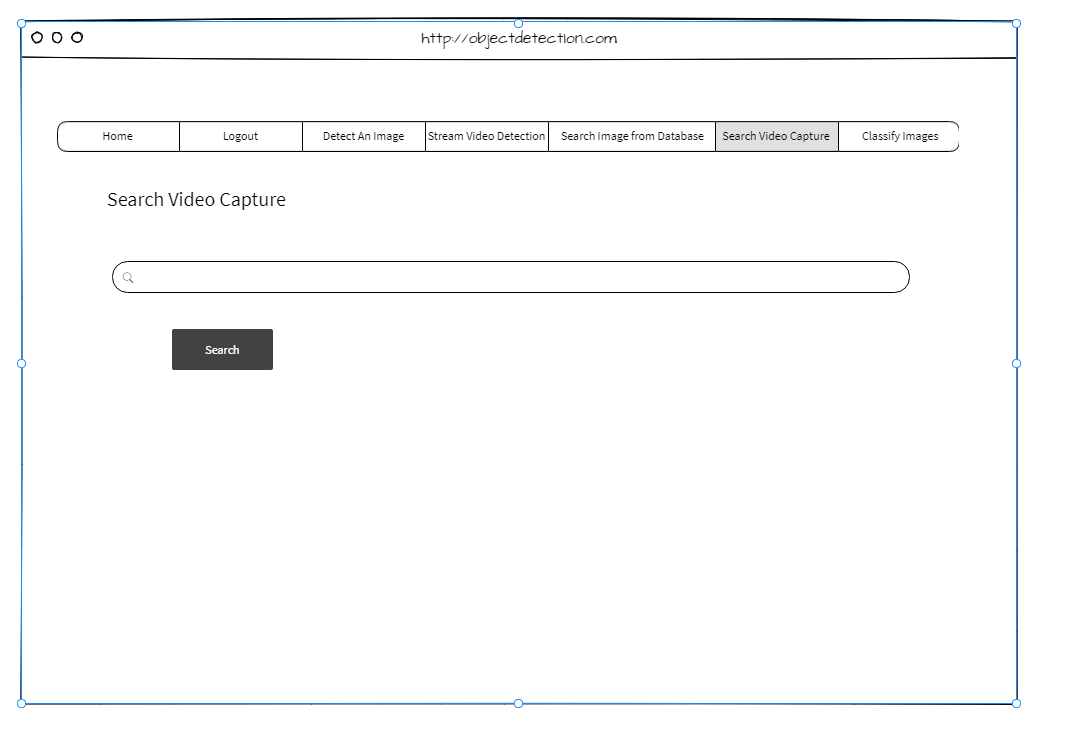


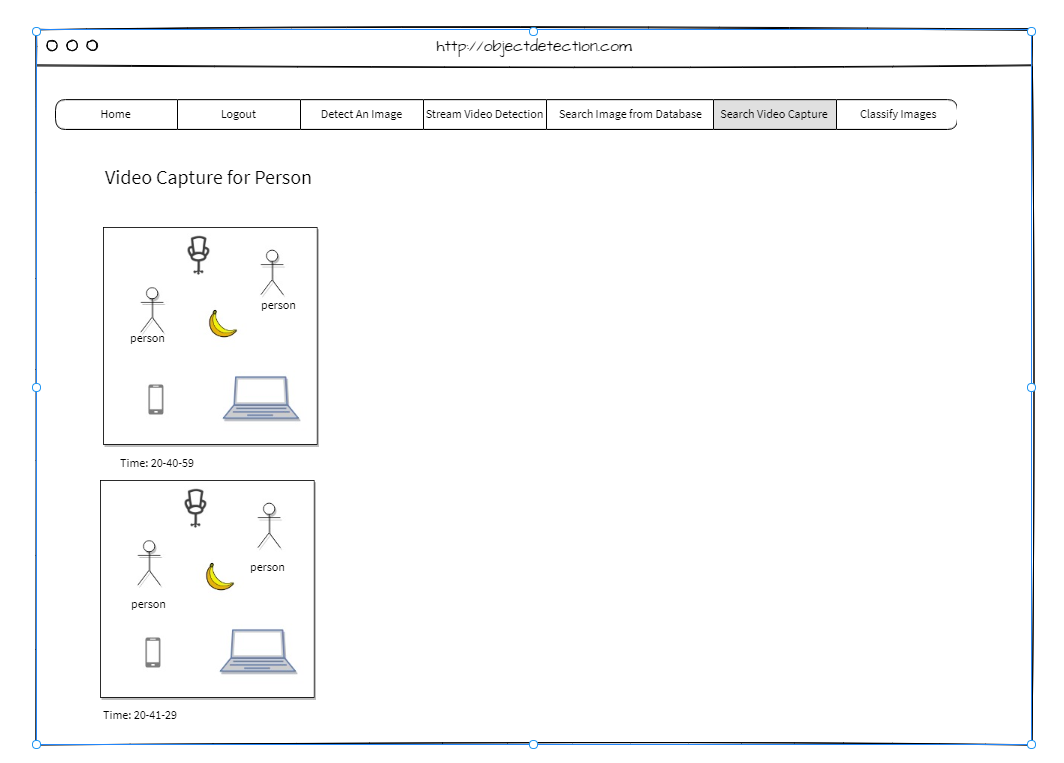


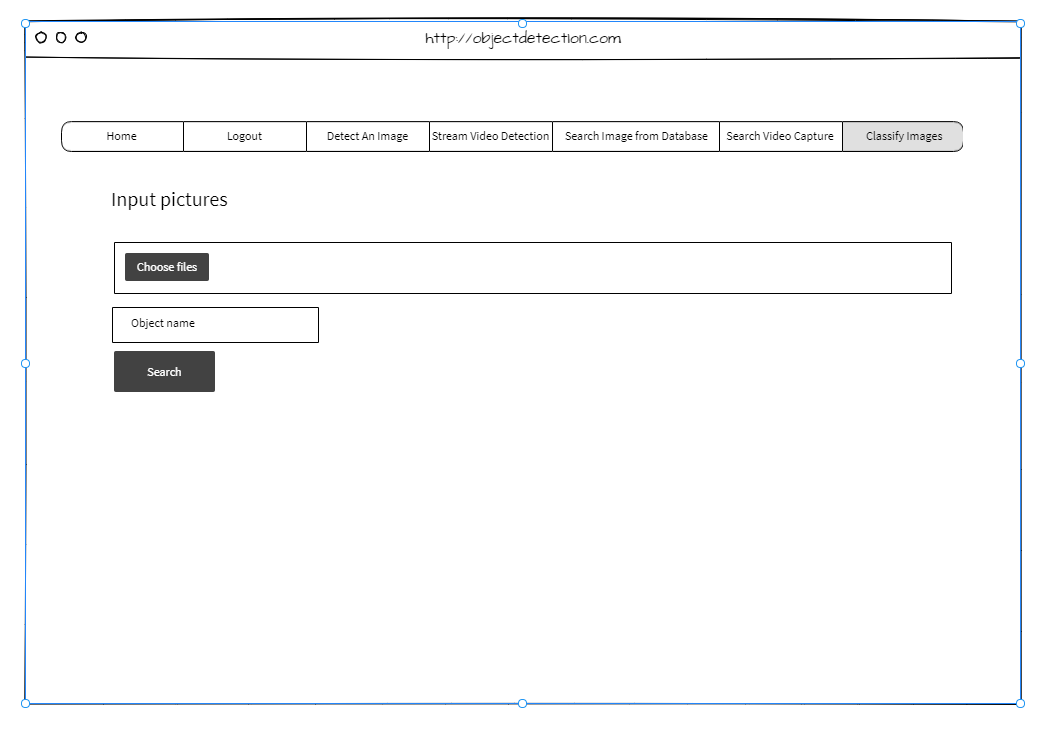




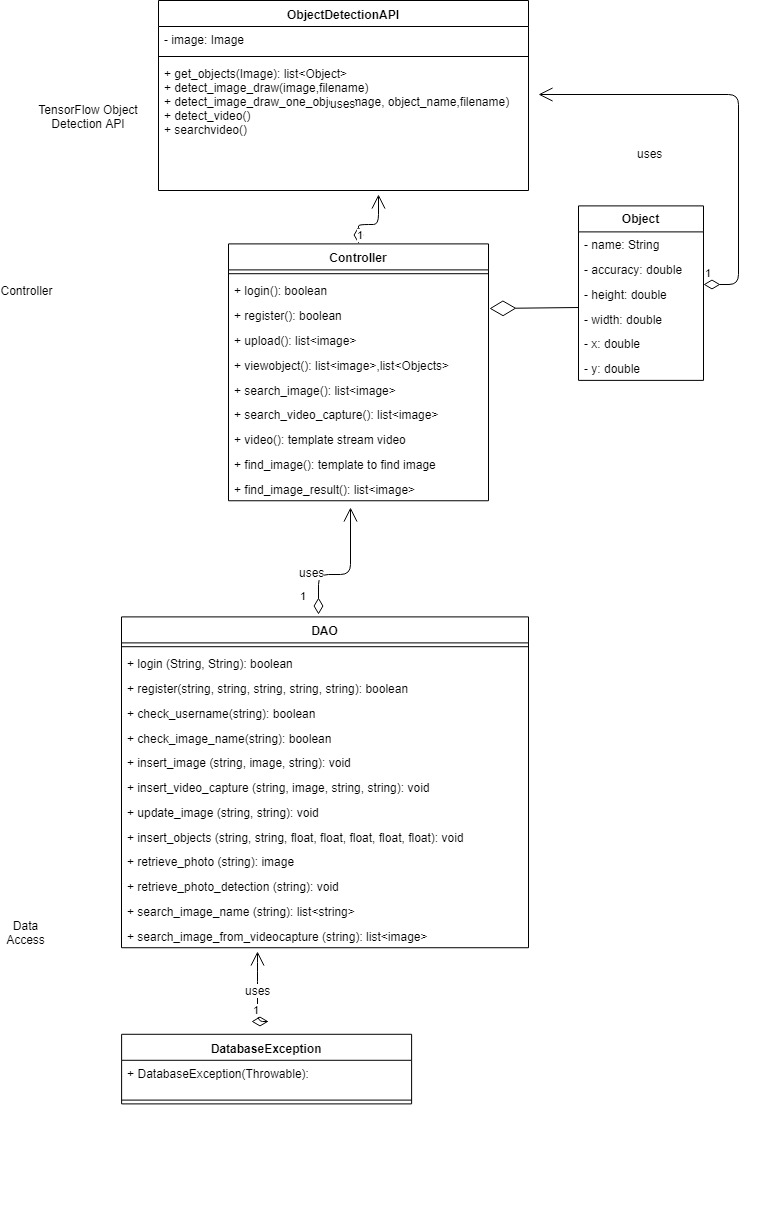






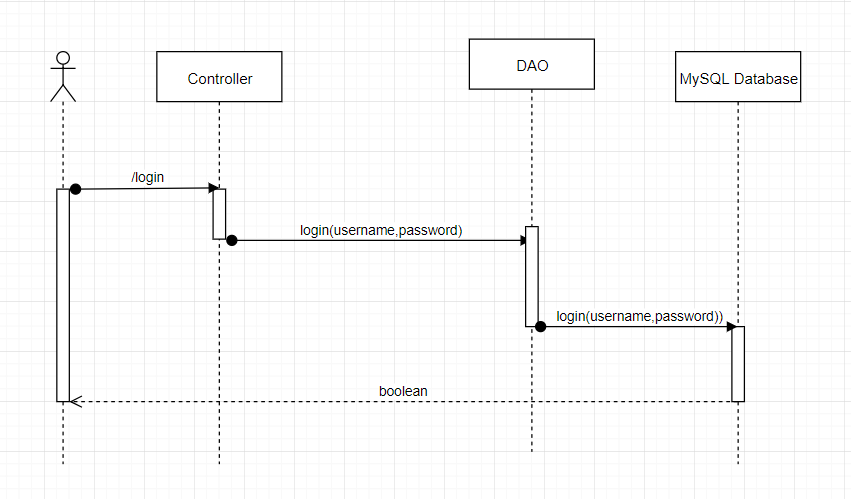


**UML Diagrams:**

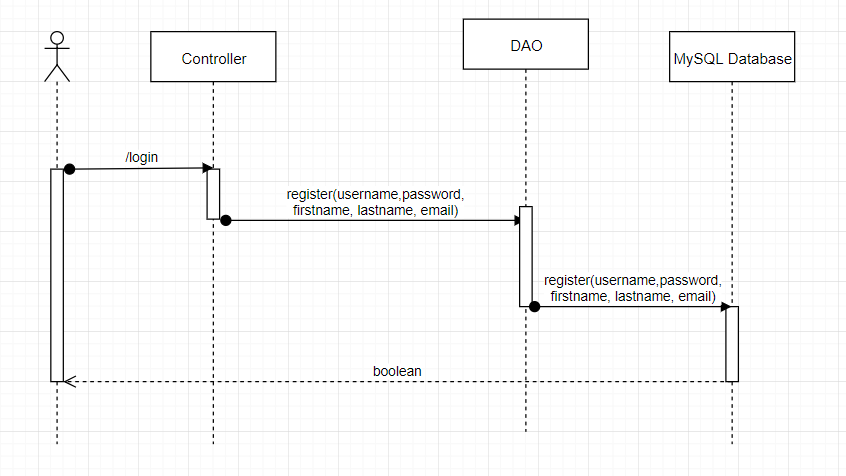


**Sequence Diagram**

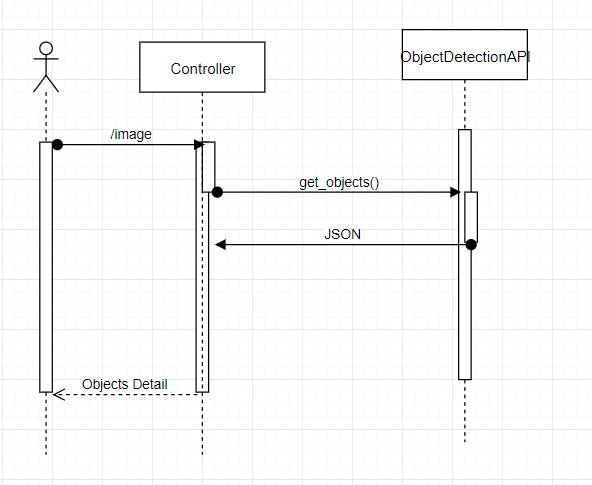
Sequence Diagram of Login:



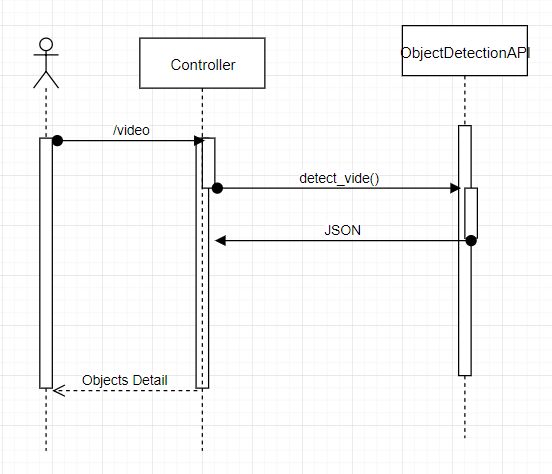
Sequence Diagram of Registration:



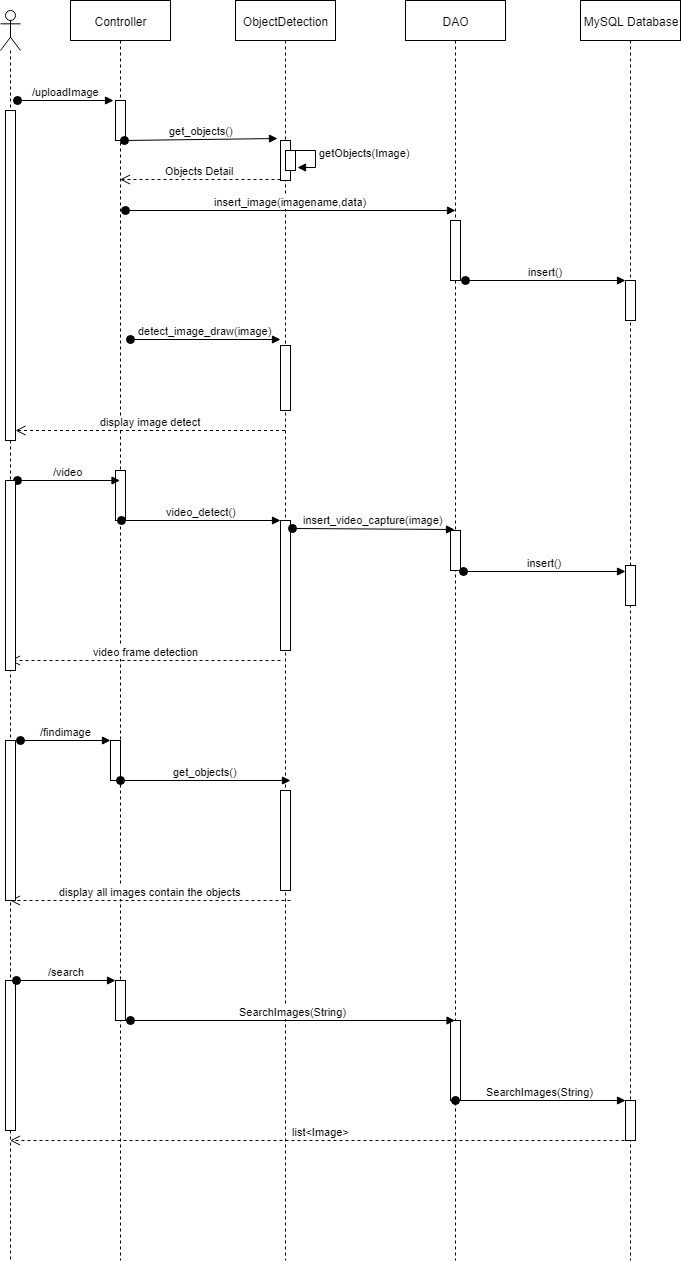
Sequence Diagram of call an Object Detection API to detect an image



Sequence Diagram of call an Object Detection API to detect a video



Sequence Diagram of detect an image and save the result into database, detect a video and save the video capture, find all images which contain input objects from multiple images and search images from database:



**Service API Design:**

TensorFlow Object Detection API

After receive image from Client through POST method, TensorFlow Object Detection API return objects JSON detail:



Example:

Method: POST

URL: <http://localhost/image>

Return objects’ detail of an image

{

"type" : "object",

"required" : [ "height", "id", "name", "score", "width", "x", "y" ],

"properties" : [

{

"score" : 0.8791,

"name" : "person",

"width" : 0.5467,

"x" : 0.0192,

"y" : 0.3956,

"id" : 1.0,

"height" : 0.9536

},

{

"score" : 0.7491,

"name" : "dog",

"width" : 0.3467,

"x" : 0.1292,

"y" : 0.5656,

"id" : 2.0,

"height" : 0.5636

}

]

}

## **NFR’s (Security Design, etc.)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | User stories | |
| ID | Features | As a(n) | I would like to <description> | So that outcome |
| 1 | Accessibility | User | I would like to access the web application from localhost | I can test the application |
|  |  | User | I would like to perform all the tasks of application from localhost | I can detect image or video as purpose |
|  |  | User | I would like to search a picture with a specific object from the database | Pictures with the input object |
| 2 | Capacity | User | I would like to detect an image that has 10 objects | I can test multiple objects detection |
|  |  | User | I would like to input an image that is 5MB size. | I have more options when choosing the input file |
| 3 | Reliability | User | I would like to have the correct name of object detection results | To predict object accuracy |
| 4 | Performance | User | I would like the application to be able to return the name of objects in the image in less than 5 seconds. | I can see the name of objects immediately to test it |
| 5 | Maintainability | Programmer | I would like to update application when necessary | I can improve application |
| 6 | Usability | User | The application should be easy to use without training | Everyone can use |
|  |  |  | The application should display objects detection in color text | I can see results clearly |
| 7 | Interoperability | User | I would like to access the web application on pc and mobile device | I can use it flexibility |
| 8 | Documentation | Developer | I would like the application to be commented | Other developer can read and develop it easier |
|  |  | Application’s owner | I would like my application well documented | I can refer to documents later |

**Deploy Flask on localhost**

This project uses localhost to host [Flask](http://flask.pocoo.org/) applications. There are 3 steps to install Flask.

Step 1: Install [mod\_wsgi](http://modwsgi.readthedocs.io/en/develop/index.html), Web Server Gateway Interface (WSGI) is a simple calling convention for web servers to forward requests to web applications or frameworks written in the Pythonprogramming language

sudo apt-get update

sudo apt-get install libapache2-mod-wsgi-py3 #for python 3

Step 2: Install Python Environment and Flask

pip3 –version

sudo apt-get install python3-pip

pip3 install --user flask

Step 3: Deploy Flask Application

This is example of deploy a Hello world application. Built a directory of a hello-world Flask application using the structure looks something like:

|-----helloworld

|--------helloworld.py

|--------hello.wsgi

|--------hello.conf

|--------env

|-----------requirements.txt

The helloworld.py reads as:

from flask import Flask

app **=** Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Hello, World!'

The content of hello.wsgi is:

import sys

sys**.**path**.**insert(0, "/var/www/helloworld")

from helloworld import app as application

And the hello.config looks like:

**<**VirtualHost **\*>**

ServerName example**.**com

WSGIScriptAlias **/** **/**var**/**www**/**helloworld**/**hello**.**wsgi

WSGIDaemonProcess hello python**-**home**=/**var**/**www**/**helloworld**/**env

**<**Directory **/**var**/**www**/**helloworld**>**

WSGIProcessGroup hello

WSGIApplicationGroup **%**{GLOBAL}

Order deny,allow

Allow from all

**</**Directory**>**

**</**VirtualHost**>**

## **Other Documentation:**

a) Tensor Flow Object Detection Algorithm.

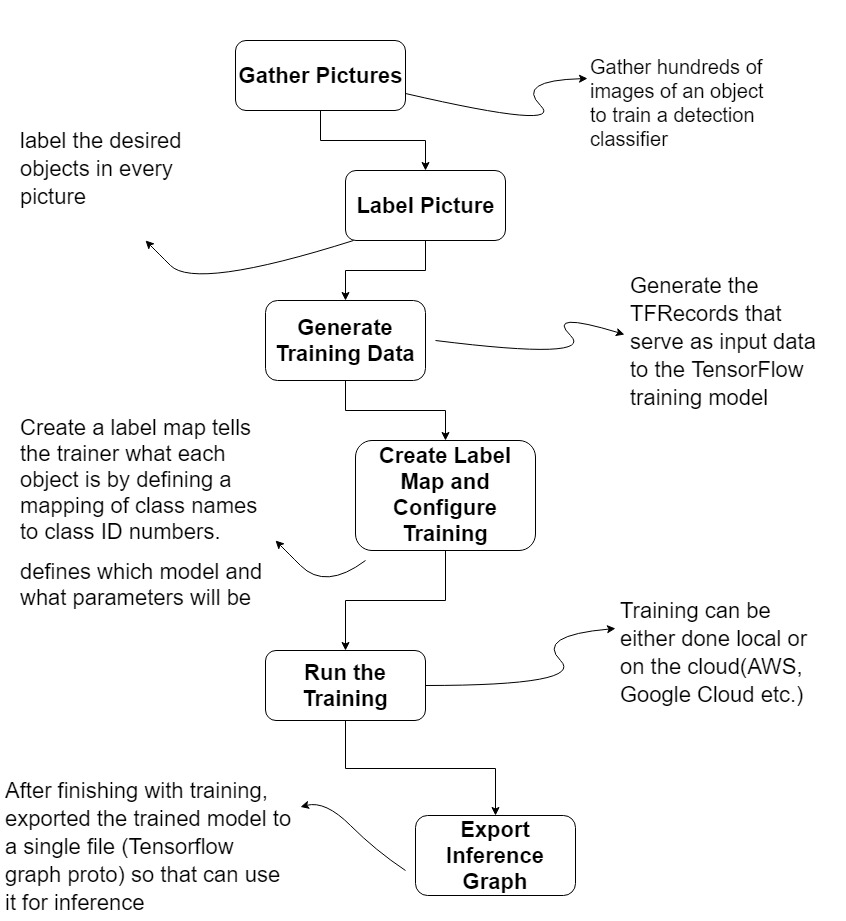
TensorFlow Object Detection Processing will include 4 steps:

Step 1: Process images

Step 2: Convert images to TFRecords

Step 3: Train the model on Machine Learning engine.

Step 4: Export the trained model and use that for inference



In this project, the PASCAL VOC 2012 dataset (it is dataset of images that comes with labeled and stored in an xml file) is used to convert to the TFRecord instead of gather the images and create it.

# **Appendix A – Technical Issue and Risk Log**

Use the template to identify and monitor project issues and risks.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Issues and Risk Log | | | | | | | | |
| **Issue or Risk** | **Description** | **Project Impact** | **Action Plan/Resolution** | **Owner** | **Importance** | **Date Entered** | **Date to Review** | **Date Resolved** |
| I/R | What is the issue or risk? | How will this impact scope, schedule & cost? | How do you intend to deal with this issue? | Who manages this issue? |  |  |  |  |
| I | The localhost’s performance is not enough to detect the large size images | Limit application’s performance. It will take more time to detect a user input image | Increase configuration, deploy on Cloud Machine Learning engine | Chuong Nguyen | important | 04/11 |  |  |
| I | The application isn’t deployed on cloud due to TensorFlow requirement environment | Limit the application’s performance. Users cannot test the application on cloud. The availability of application is limit | Develop in future | Chuong Nguyen | important | 04/11 |  |  |
| I | The PASCAL VOC dataset has 20 categories of objects, | Limit the performance of application. Application cannot detect unfamiliar objects which are not in PASCAL VOC dataset | Generate more dataset for the Object Detection API | Chuong Nguyen | important | 04/11 |  |  |

# **Appendix B – References**

1. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. *You Only Look One: Unified, Real-Time Object Detection*. University of Washington, Allen Institute of AI, Facebook AI Research. Retrieved November 2, 2018, from <https://arxiv.org/pdf/1506.02640.pdf>
2. Hui, J. (2018, Mar 17). *Real-time Object Detection with YOLO, YOLOv2, and now YOLOv3.* Retrieved from <https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088>
3. Tran, D. (2017, Jul 28). *How to train your own Object Detector with TensorFlow’s Object Detection API.* Retrieved from <https://towardsdatascience.com/how-to-train-your-own-object-detector-with-tensorflows-object-detector-api-bec72ecfe1d9>
4. Hard, C. (2017, Dec 3). *Computer Vision on the Web with WebRTC and TensorFlow.* Retrieved from <https://webrtchacks.com/webrtc-cv-tensorflow/>
5. *Deploy a Flask Application on an Azure Virtual Machine.* Retrieved from <http://leifengblog.net/blog/deploy-flask-applications-on-azure-vps/>
6. TensorFlow project.[*https://github.com/tensorflow*](https://github.com/tensorflow)
7. The [PASCAL](http://pascallin2.ecs.soton.ac.uk/) Visual Object Classes Homepage. Retrieved from <http://host.robots.ox.ac.uk/pascal/VOC/>
8. [*https://www.tensorflow.org/*](https://www.tensorflow.org/)
9. [*https://www.tensorflow.org/tutorials/*](https://www.tensorflow.org/tutorials/)
10. [*https://opensource.google.com/projects/tensorflow*](https://opensource.google.com/projects/tensorflow)

# **Appendix C – External Resources**

|  |  |
| --- | --- |
| **GIT URL:** | [*https://github.com/chuongngd/Image-Detection*](https://github.com/chuongngd/Image-Detection) |
| **Project Portfolio** | [*https://github.com/chuongngd/Images-Object-Detection*](https://github.com/chuongngd/Images-Object-Detection) |
| **Project Demonstration** | [*https://www.loom.com/share/b38373419963477faaa9f4d3b8eb0335*](https://www.loom.com/share/b38373419963477faaa9f4d3b8eb0335) |
| **Hosting URL:** | *The Hosting URL (if applicable).* |