Image’s Objects Detection

Chuong Nguyen

CST-451 Capstone Project Final Architecture & Design

Grand Canyon University

Instructor: Professor Mark Reha

Revision:

Date: 11/05/2018

**ABSTRACT**

The human can recognize the objects easily in the image. With the development of photography technology, there is a massive number of images need to process, and the human cannot do them all. It is necessary to have algorithms to see and 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 can apply widely to manufacture, traffic control, security ...

My application is building a program to detect objects in an image. From web application, user input image from a local computer and the object detection algorithm will return the name of objects in the image. All of the images and their object detection result will be saved on the database. The application also uses to search a specific object from images in the database. The application can be applied in traffic monitoring. For example, the cameras at an intersection take a lot of pictures of the traffic. 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**

This project is a web application that users use to find objects from images. After a user logs in or register, they can input an image and display objects detail in the image. The image and its objects detail will be stored in the database. A user can retrieve the images and their objects detail from the database, and find the images which contain a specific object.

The solution of the application is to create a web server using Flask. A TensorFlow Object Detection API is called from Flask server to detect objects in the image which is input from a web application.

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

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 11/05 | Chuong Nguyen | Design Phase v 1.0 |
|  |  |  |
|  |  |  |

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

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

**Integrated Instructor Feedback into Project Documentation**

Yes  No

**TABLE OF CONTENTS**

Design Overview 5

Detailed High-Level Solution Design 5

Detailed Technical Design 10

Appendix A – Technical Issue and Risk Log 36

Appendix B – References 37

Appendix C – External Resources 38

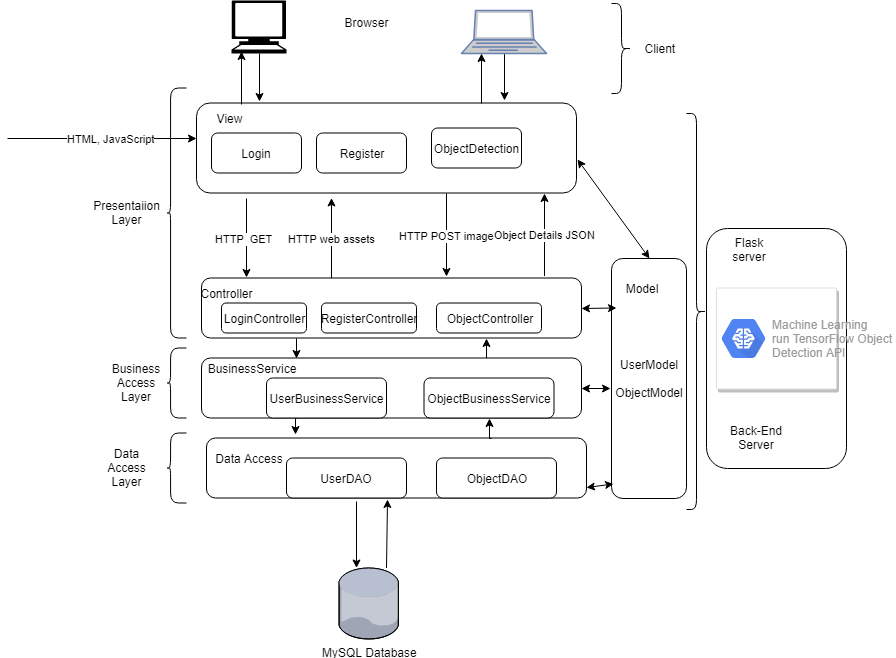
# **Design Overview**

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 provides 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.

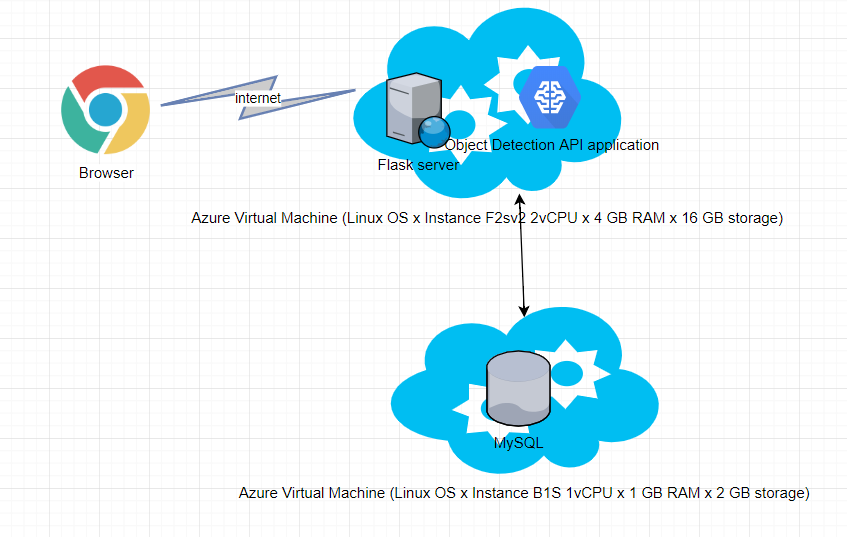
# **Detailed High-Level Solution Design**

## Detailed overview.

**Logical Solution Design:**



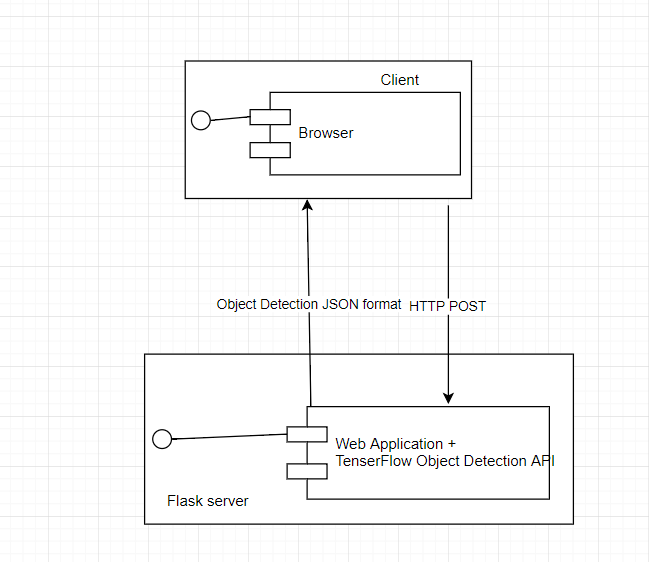
**Physical Solution Design:**



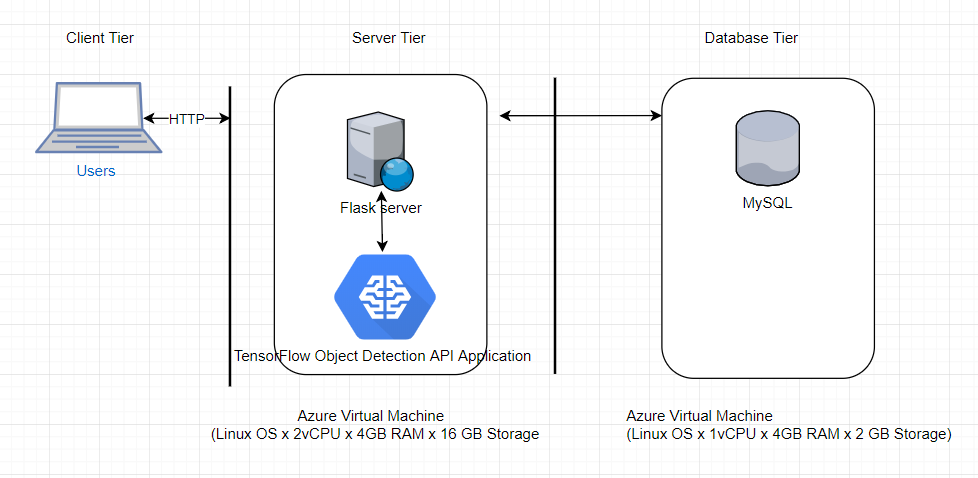
In this project, Flask server, Machine Learning engine and MySQL database server will be run on a Azure Virtual Host with the configuration as describe above.

**System Design:**

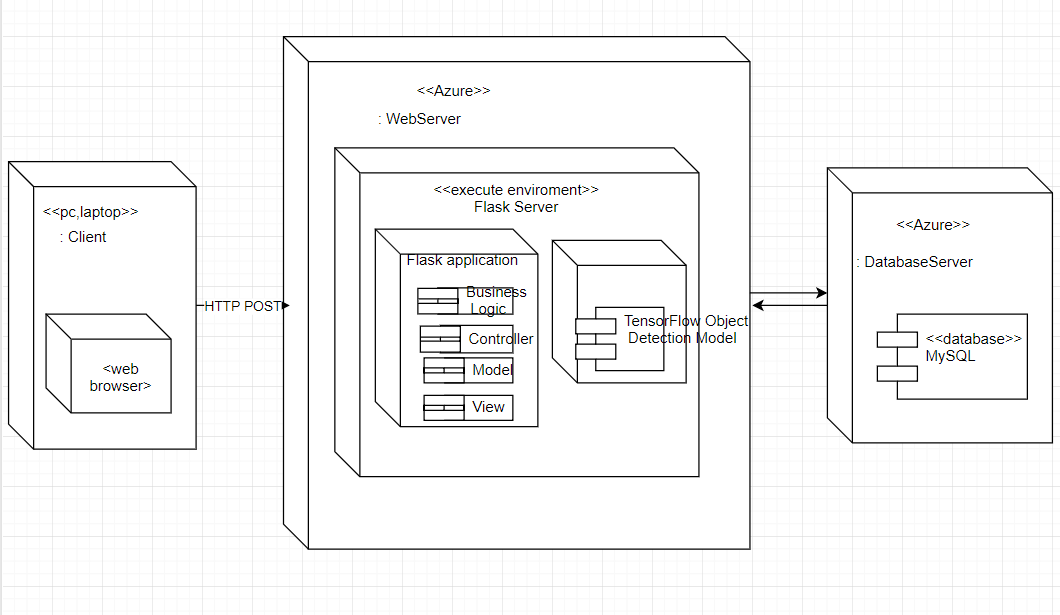
UML Component Diagram for Logical Design



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 such as people, cars, animal…

Input: images

Output: instances of objects from the image.

Ideas: Create a web application to let the user input image and return all instances of objects such as people, cars, animal…from the image after execute image object detection. 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.
* Flask is used for the web server. TensorFlow Object Detection API is used 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 YOLO (You Only Look One) algorithm is used to create the Object Detection API (it is described detailly in 2.6).
* The application will be hosted on Azure.

|  |  |
| --- | --- |
| 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. YOLO (You Only Live One) | 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 | Use Azure Virtual Machine to deploy Flask and TensorFlow Object Detection API on cloud, that user can access through internet. |

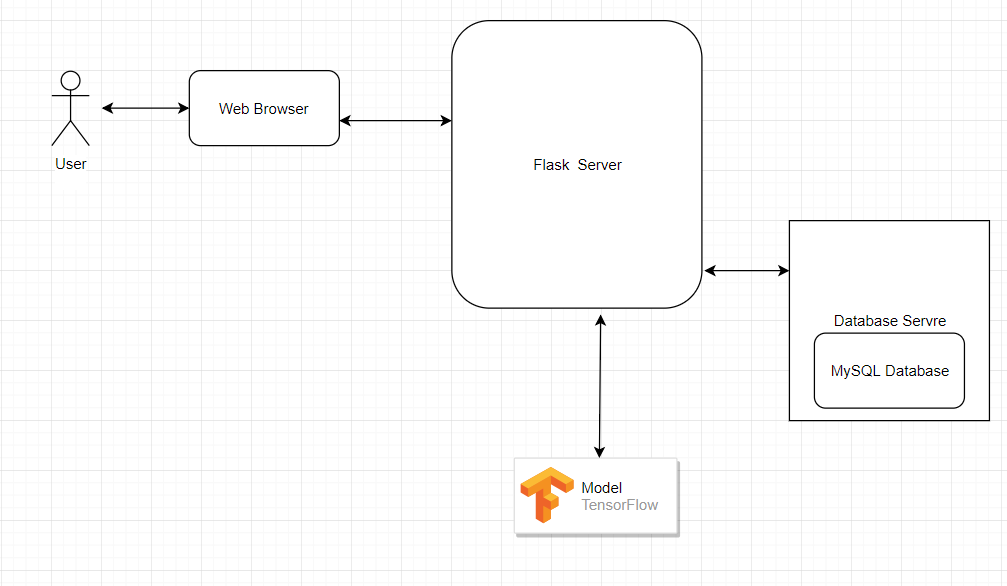
,

|  |
| --- |
| 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 – Azure Virtual Machine (Linux OS x Instance F2sv2 2vCPU x 4 GB RAM x 16 GB storage) for hosting Flask server and Object Detection application |
| 7 - Azure Virtual Machine (Linux OS x Instance B1S 1vCPU x 1 GB RAM x 2 GB storage) for host the MySQL database |

## 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 the MVC design pattern 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.



General Technical Diagram

### Key Technical Design Decisions:

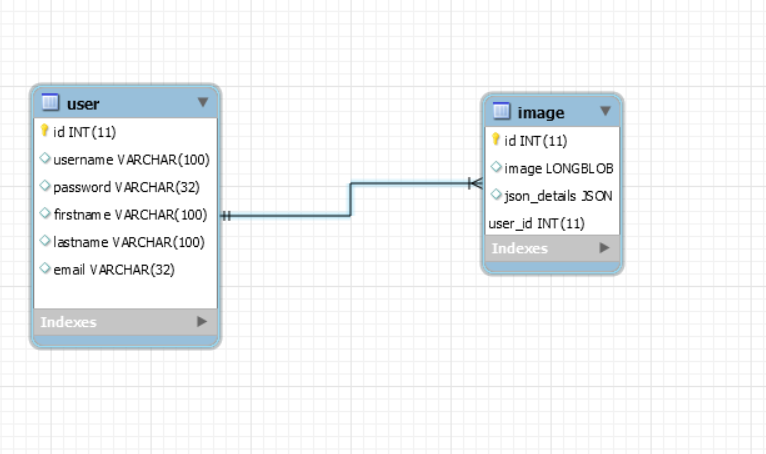
Embedding: The application is required TensorFlow Object Detection API. A Python application using TensorFlow will be running on the server to create a TensorFlow Object Detection API. This application needed a high-performance machine to perform training and testing Object Detection model. In this project 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 PASCAL VOC 2012 dataset and YOLO algorithm is used to train the dataset and export the trained model that can use to detect the image’s object.

### Database ER Diagram:

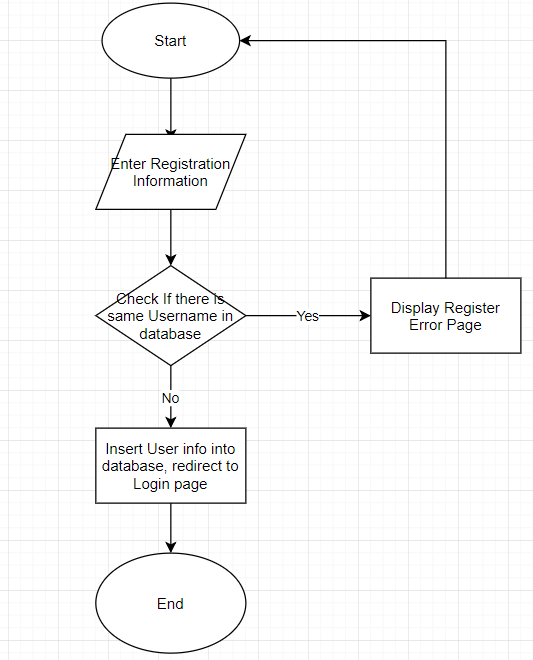


**Database DDL Scripts:**

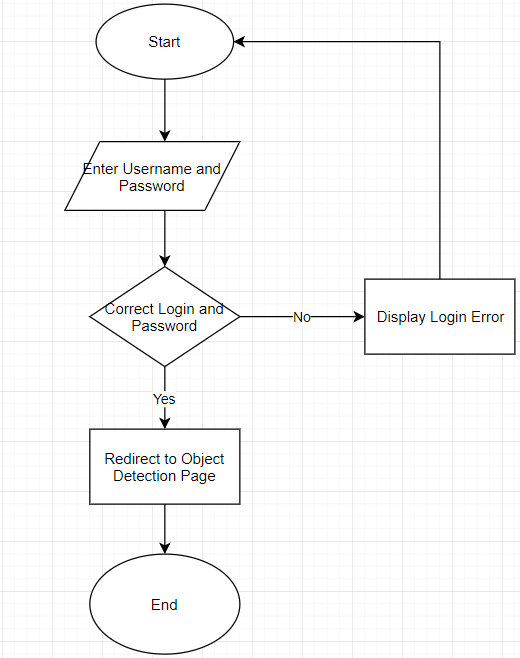


**Flow Charts/Process Flows:**

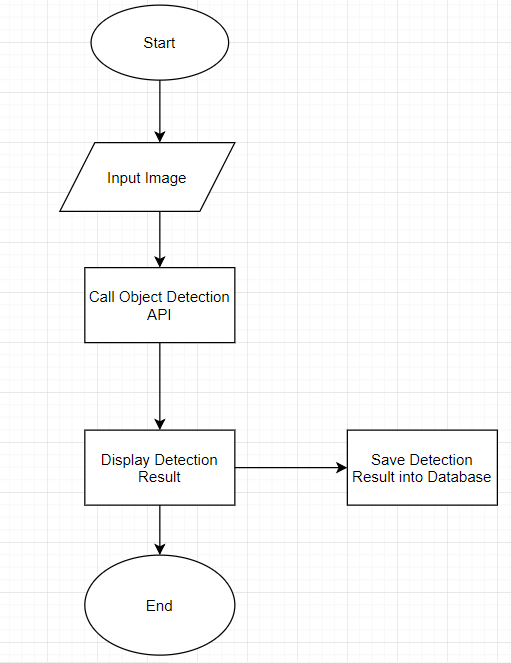
Flowchart of Registration functional



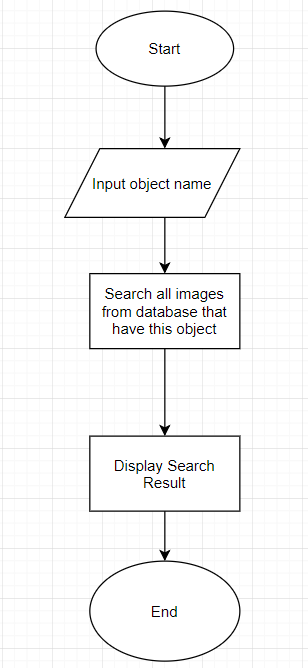
Flowchart of Login functional



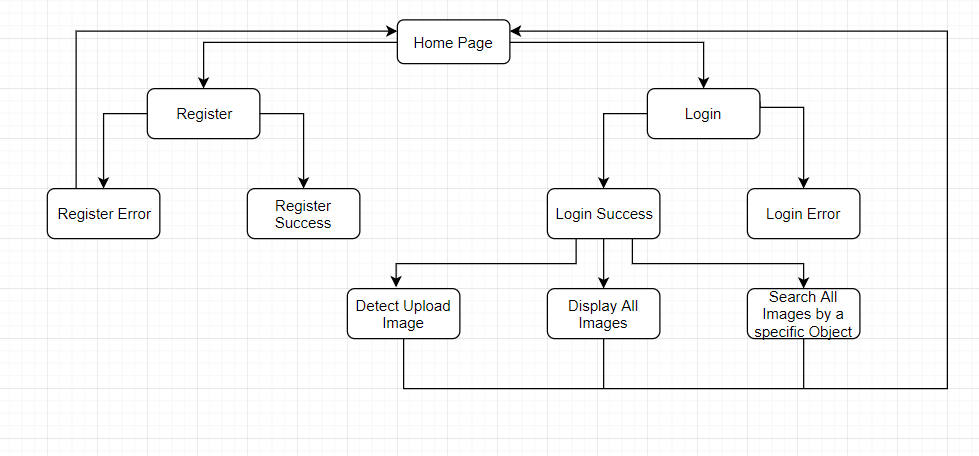
Flowchart of Object Detection functional



Flowchart of Search specific object from images

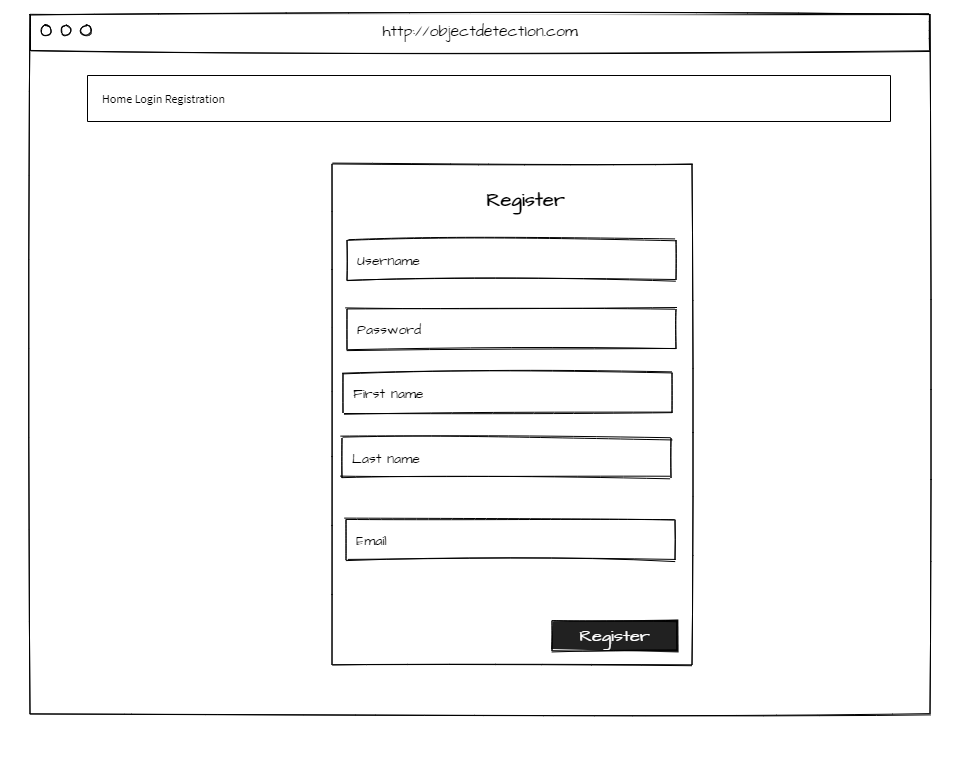


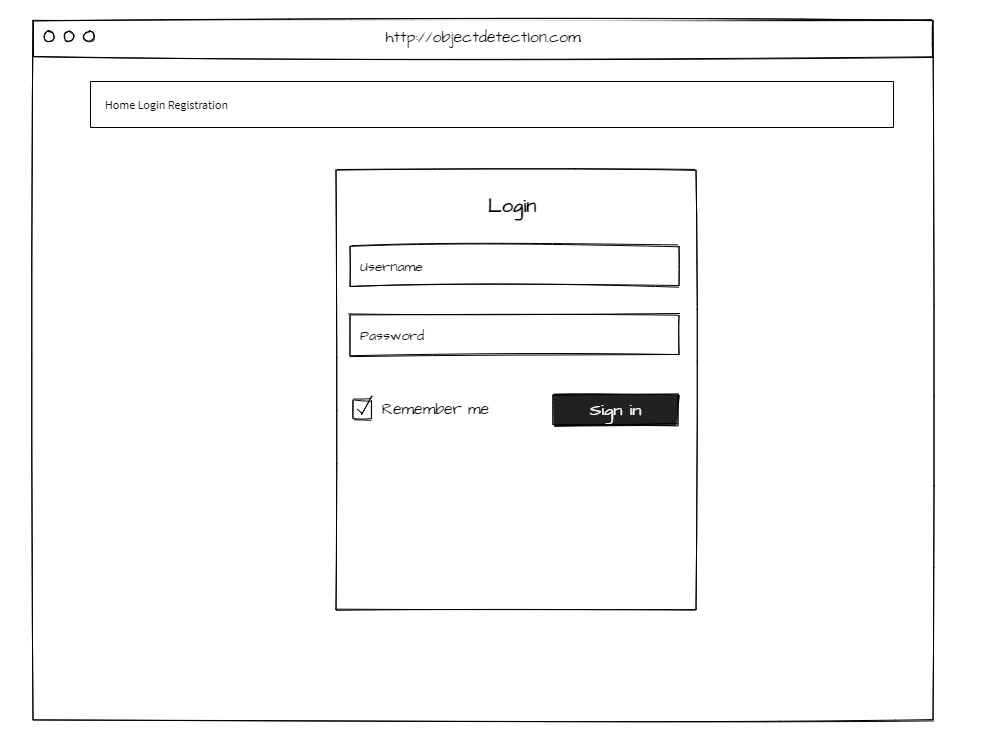
**Sitemap Diagram:**

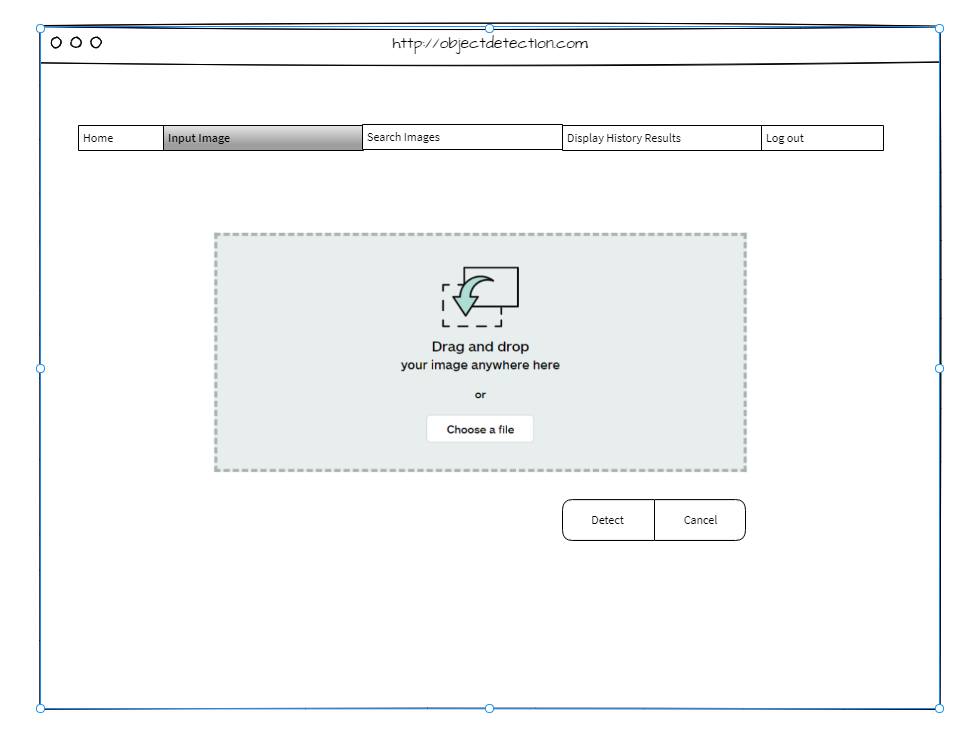


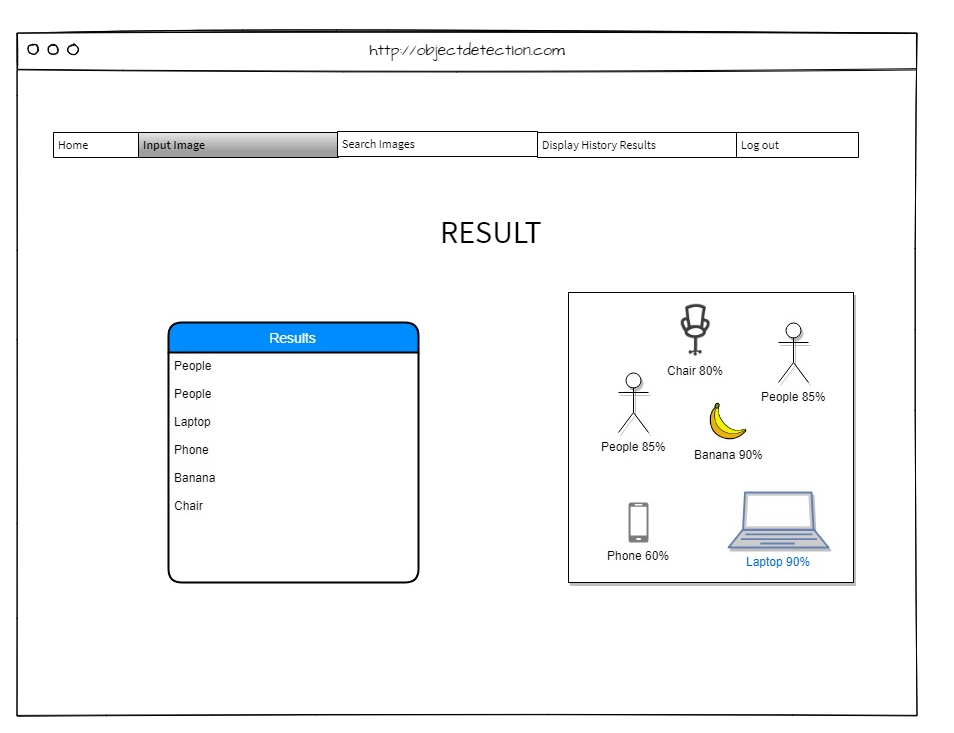
Sitemap of application

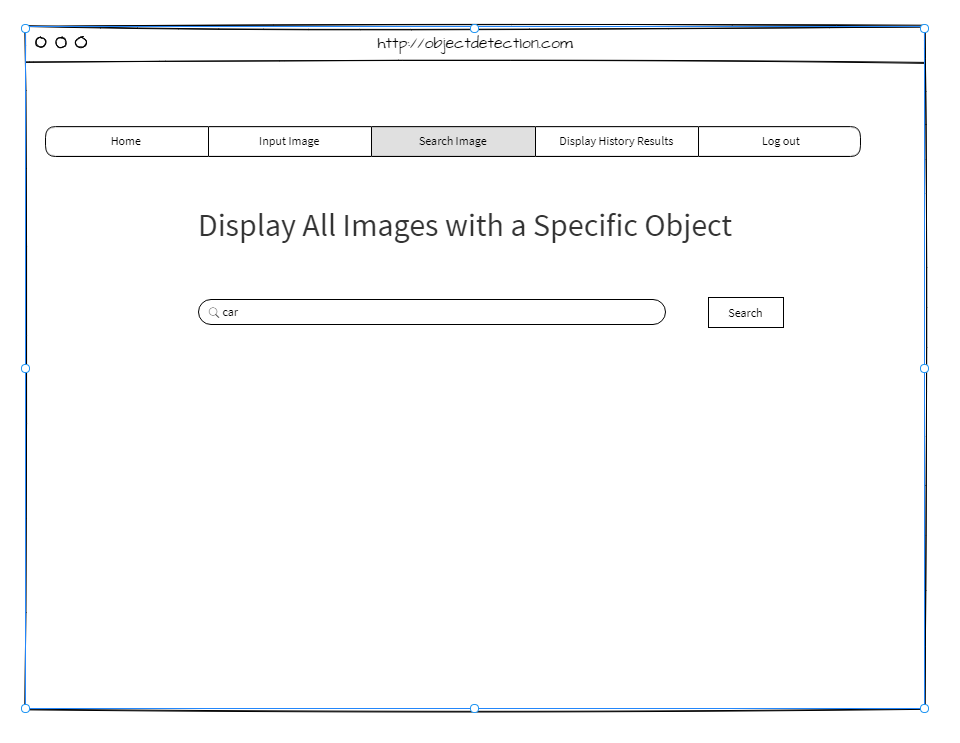
**User Interface Diagrams:**

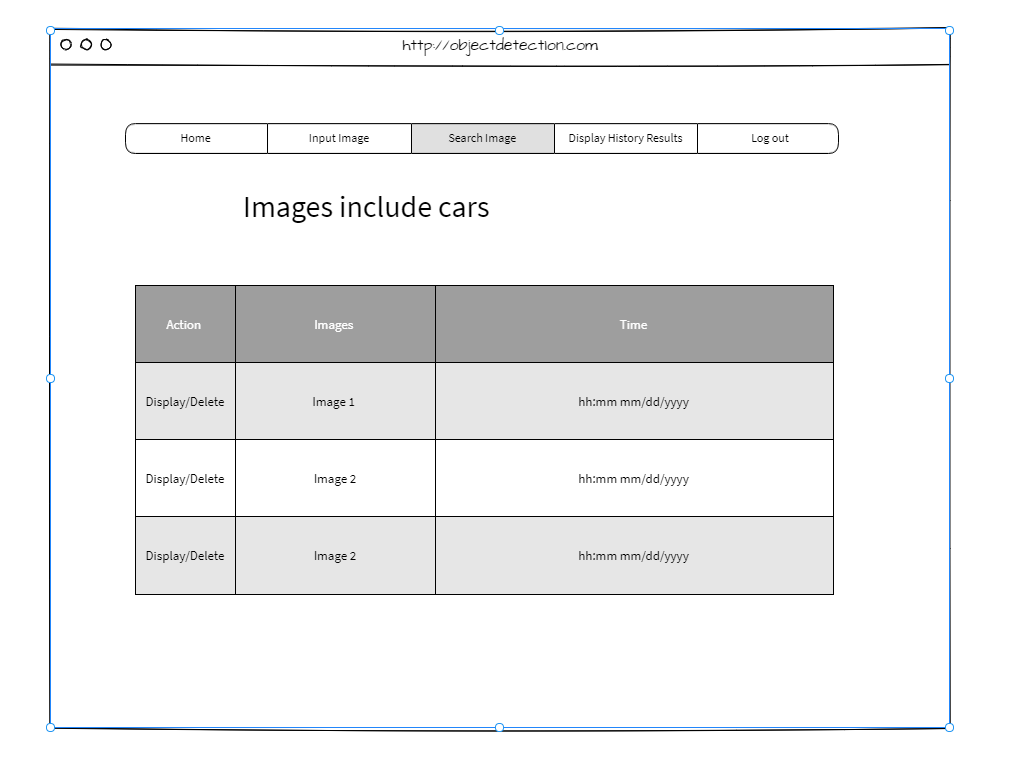


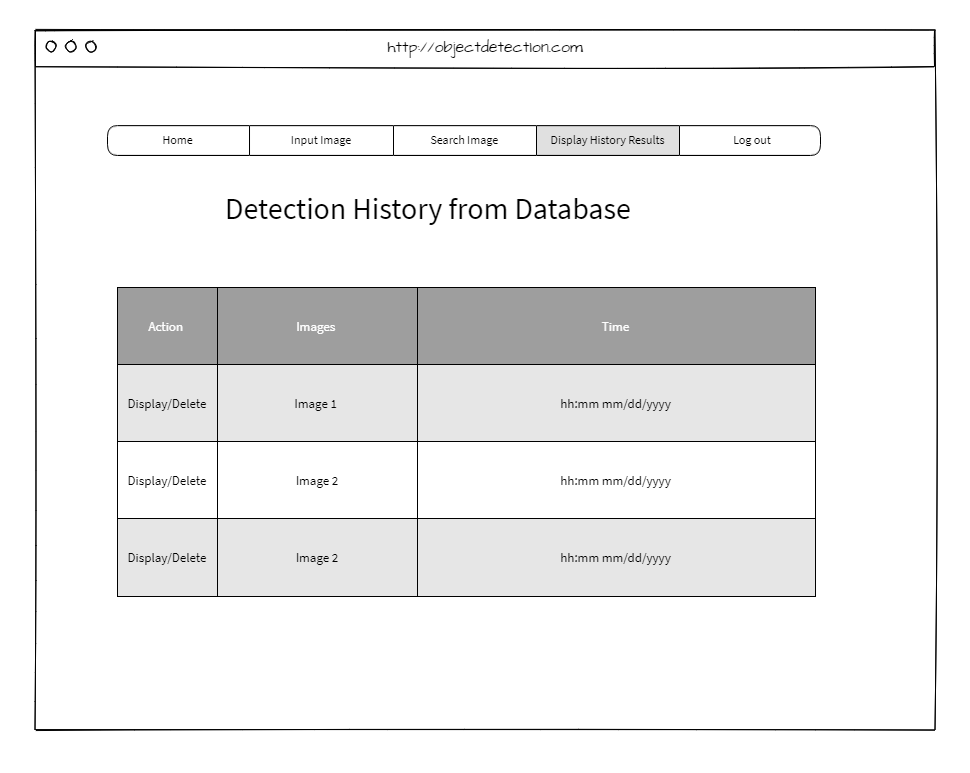




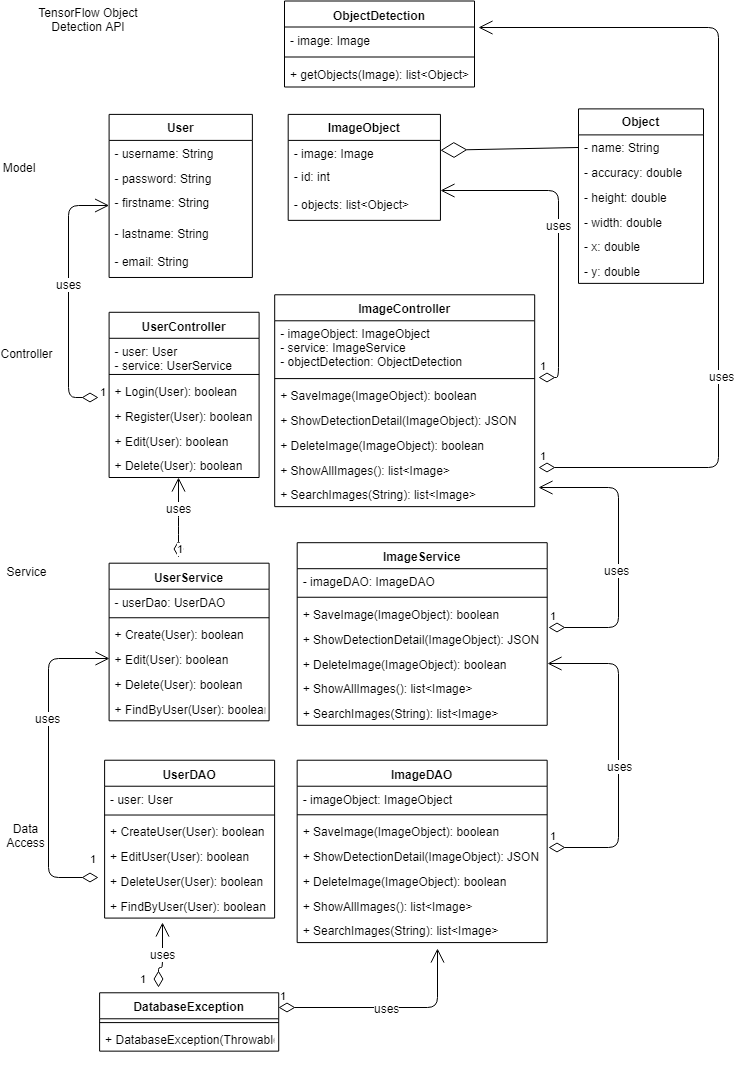






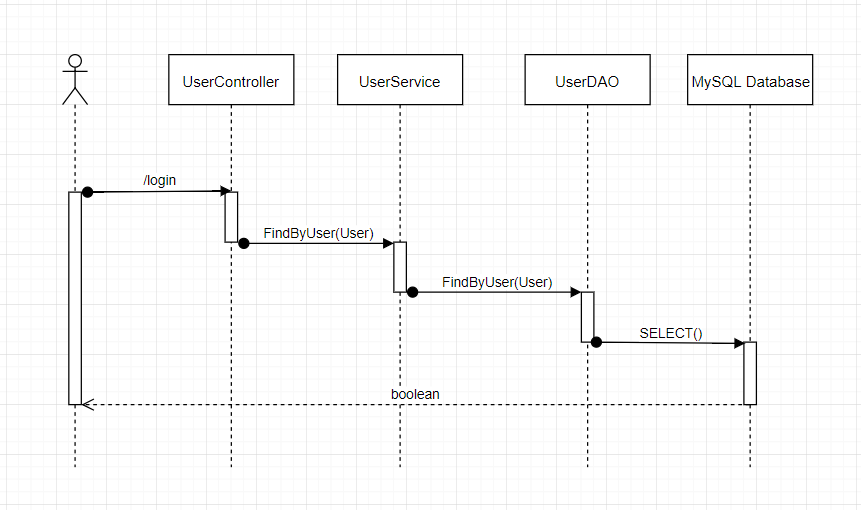


**UML Diagrams:**

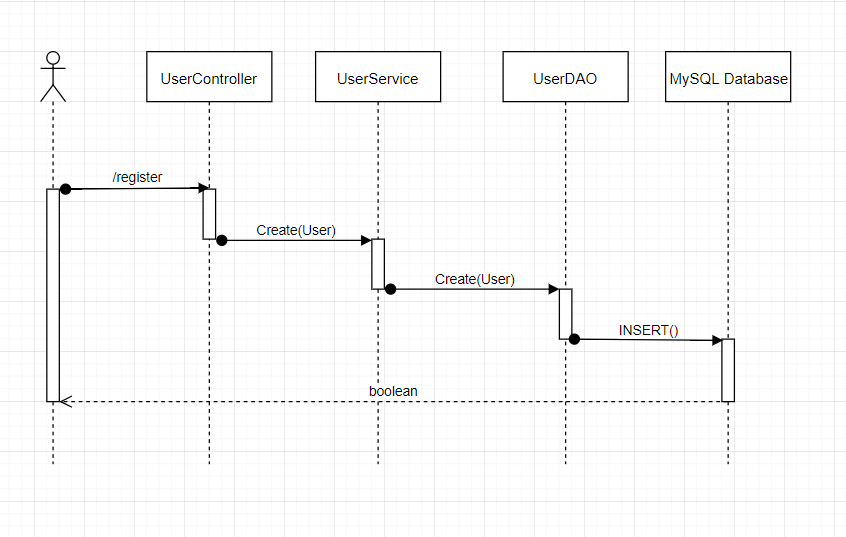


**Sequence Diagram**

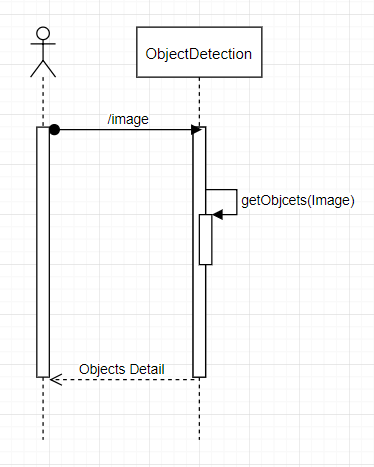
Sequence Diagram of Login:



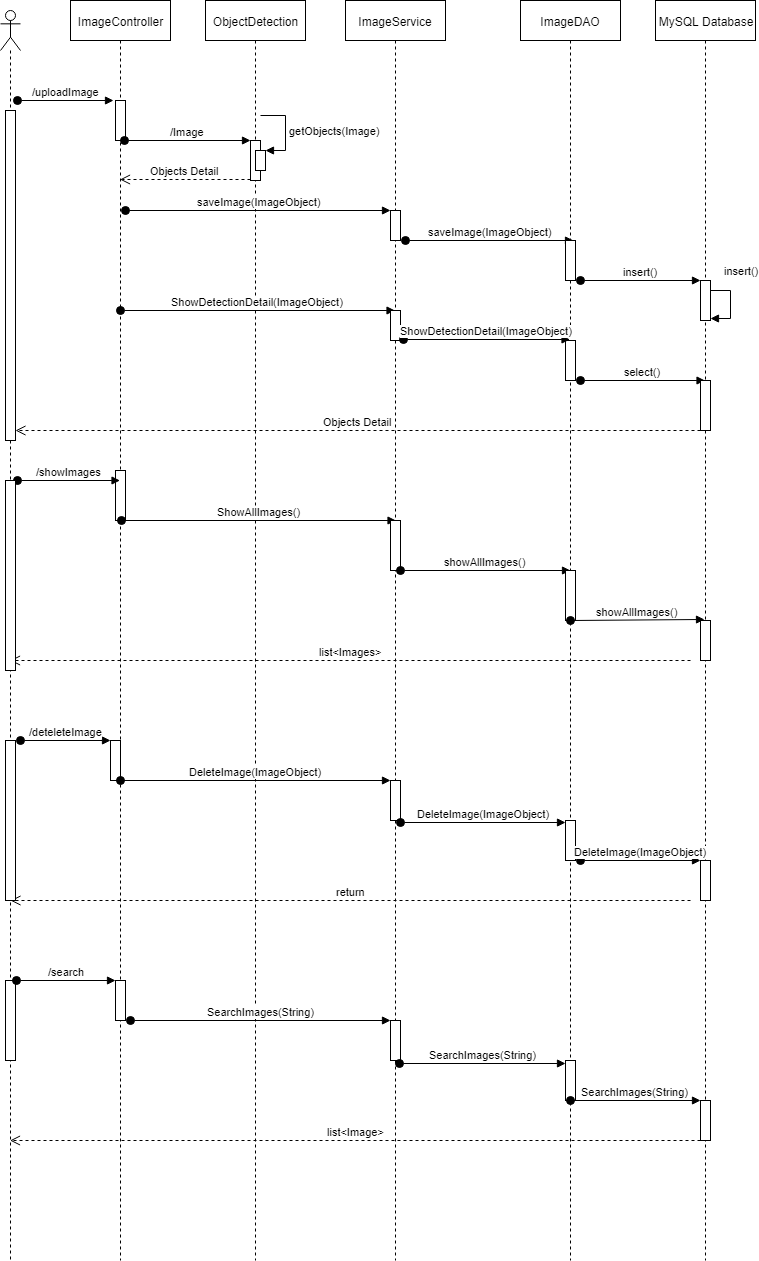
Sequence Diagram of Registration:



Sequence Diagram of call an Object Detection API



Sequence Diagram of Object Detection, Search objects from images and show database result:



**Service API Design:**

TensorFlow Object Detection API

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

Method: POST

URL: <http://azure-virtualhost/image>

Return objects detail of an image



{

"paths" : {

"/image" : {

"post" : {

"summary" : "post an image to Object Detection API",

"consumes" : [ "application/json" ],

"produces" : [ "application/json" ],

"parameters" : [ {

"in" : "body",

"name" : "ObjectDetection",

"description" : "Image send to AI engine and receive object detail in JSON type",

"required" : false,

"schema" : {

"$ref" : "#/definitions/ObjectDetection"

}

} ],

"responses" : {

"201" : {

"description" : "item created"

},

"400" : {

"description" : "invalid input, object invalid"

},

"409" : {

"description" : "an existing item already exists"

}

}

}

}

},

"definitions" : {

"ObjectDetection" : {

"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

}

}

}

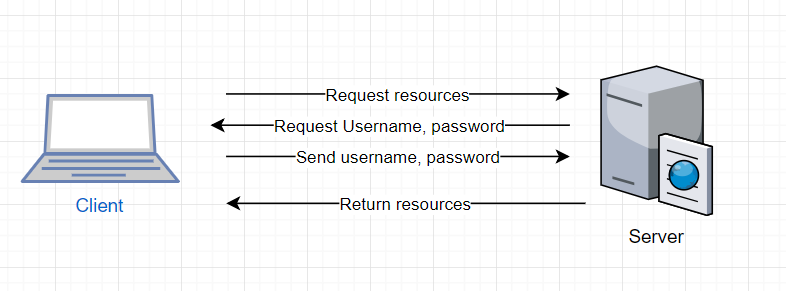
}

## 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 everywhere | I can use application wherever I want |
|  |  | User | I would like to perform all the tasks of application from everywhere | I can use application wherever I want |
|  |  | User | I would like to search a picture with a specific object from the database | I can use application wherever I want |
| 2 | Availability | User | I would like to access the web application anytime | I can use application whenever I want |
|  |  | User | I would like to perform all tasks of application anytime | I can use application whenever I want |
|  |  | User | I would like to search images with a specific object from database anytime | I can use application whenever I want |
| 3 | 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 |
|  |
|  |  | System | I would like to have multiple users be able to use the application at the same time. | Multiple users can perform object detections in same time |
| 4 | Reliability | User | I would like to have correct name of object detection results | To predict object accuracy |
| 5 | Recoverability | System | The system should host on cloud | To prevent lost data on the local host |
|  |  | Admin | I would like to have my local computer as a backup secondary host | I can recover the system in case have a disaster |
| 6 | 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 |
| 7 | Cost | User | I would like user to be able to access application with internet connection, no extra cost | No extra cost |
|  |  | Admin | I would like the system only to need hosting fee | No extra cost |
| 8 | Security | Admin | I would like to have security on websites | To secure application |
|  |  | Admin | I would like the virtual machine that hosts the application to be secure. | To secure application |
| 9 | Maintainability | Admin | I would like to test the application’s performance every week | To troubleshoot application’s performance |
|  |  | Admin | I would like to backup data every week | I don’t lose data |
|  |  | Admin | I would like to update application when necessary | I can improve application |
| 10 | 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 |
| 11 | Interoperability | User | I would like to access the web application on pc and mobile device | I can use it flexibility |
| 12 | 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 |

**Security**

In this project, username, password authentication for all request service from client.



**Deploy Flask on Azure**

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

Step 1: Install [mod\_wsgi](http://modwsgi.readthedocs.io/en/develop/index.html), which is a common interface between web servers and Flask applications.

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**>**

## Operational Support Design:

In Flask, there is integrated support for signaling. This support is provided by the [blinker](https://pypi.org/project/blinker/) library. Signals help to decouple applications by sending notifications when actions occur elsewhere in the core framework or another Flask extensions. In short, signals allow certain senders to notify subscribers that something happened.

User logging activities:

This project will use Blinker to create signal to keep track of any user’s actions such as login, register, detect image’s objects, search object.

Example :

from blinker import Namespace

my\_signals = Namespace()

def create\_user():

# add user code here

user\_created = my\_signals.signal('user-created')

Source: http://flask.pocoo.org/docs/1.0/signals/

Error logging:

For the error logging, python has logging library. Once the library has been imported, the RotatingFileHandler handler is used to to handler logging.

Example:

import logging

from logging.handlers import RotatingFileHandler

if \_\_name\_\_ == "\_\_main\_\_":

# initialize the log handler

logHandler = RotatingFileHandler('info.log', maxBytes=1000, backupCount=1)

# set the log handler level

logHandler.setLevel(logging.INFO)

# set the app logger level

app.logger.setLevel(logging.INFO)

app.logger.addHandler(logHandler)

app.run()

The logs are written into the info.log file

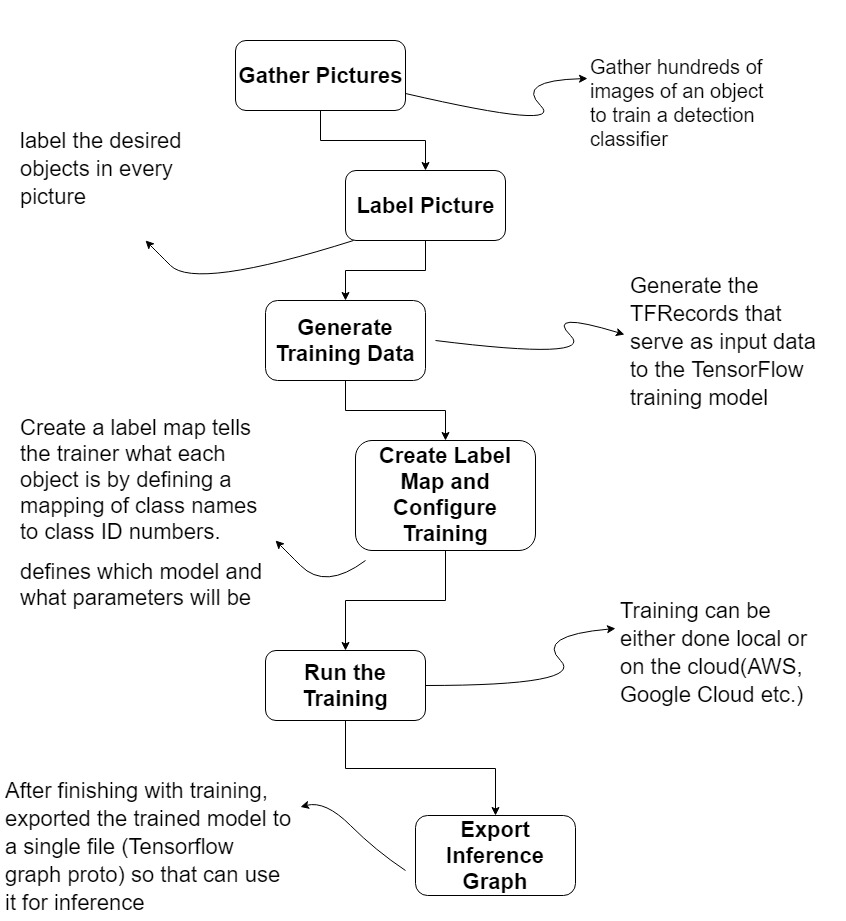
Source: https://codehandbook.org/writing-error-log-in-python-flask-web-application/

## Other Documentation:

**Tensor Flow Object Detection Algorithm.**

TensorFlow Object Detection Processing will include 4 steps:

1. Process images
2. Convert images to TFRecords
3. Train the model on Machine Learning engine using YOLO algorithm
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.

YOLO algorithm overview:

(from “Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3”, by Jonathan Hui, https://medium.com/@jonathan\_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088)

The image is divided into an S x S grid cell. Each grid cell predicts B bounding boxes and confidence score of those boxes. Each boundary box contains 5 elements: (x, y, w, h, p)

The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell.

The width w and height h are predicted relative to the whole image.

p : box confidence score, it reflects how likely the box contains an objects and how accurate is the boundary box. It represents the Intersection Over Union (IOU) between the predicted box and any ground truth box.

Box confidence score p = Pr(Object) \* IoU

Pr(Object): the probability the box contains an object

IoU: the measure of the overlap between two bounding boxes



IoU =

Each grid cell also predicts C conditional class probability (one per class for the likeliness of the object class). The **conditional class probability** is the probability that the detected object belongs to a particular class (one probability per category for each cell).

Conditional class probability = Pr(classi|object)

Pr(classi|object) is the probability the object belongs to classi given an object is presence

The class confidence score for each prediction box is computed:

Class confidence score = Pr(classi) \* IoU

= box confidence score \* conditional class probability

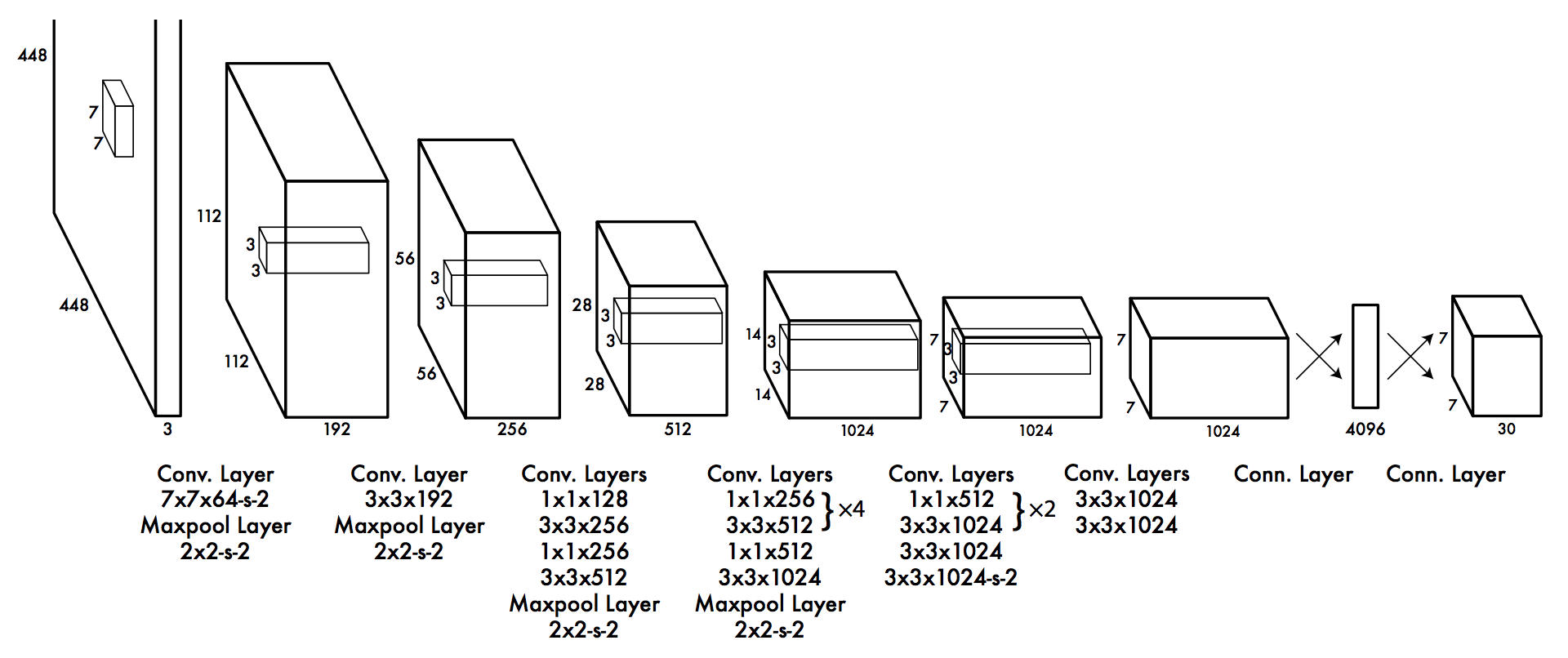
Pr(classi) is the probability the object belongs to classi

The system models detection as a regression problem. It divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an S × S × (B ∗ 5 + C) tensor

For evaluating YOLO on PASCAL VOC, S = 7, B = 2. PASCAL VOC has 20 labelled classes so C = 20. Final prediction is a 7 × 7 × 30 tensor.

The major concept of YOLO is to build a CNN network to predict a (7, 7, 30) tensor. It uses a CNN network to reduce the spatial dimension to 7×7 with 1024 output channels at each location. YOLO performs a linear regression using two fully connected layers to make 7×7×2 boundary box predictions (the middle picture below). To make a final prediction, the high box confidence scores (greater than 0.25) as final predictions.

YOLO has 24 convolutional layers followed by 2 fully connected layers (FC). Some convolution layers use 1 × 1 reduction layers alternatively to reduce the depth of the features maps. For the last convolution layer, it outputs a tensor with shape (7, 7, 1024). The tensor is then flattened. Using 2 fully connected layers as a form of linear regression, it outputs 7×7×30 parameters and then reshapes to (7, 7, 30), i.e. 2 boundary box predictions per location.



The Architecture. The detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers

SourceImage: <https://arxiv.org/pdf/1506.02640.pdf>

# **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? |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

# **Appendix B – References**

*List all Project Documentation References*

*List all references using APA style*

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>

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>

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>

Hard, C. (2017, Dec 3). *Computer Vision on the Web with WebRTC and TensorFlow.* Retrieved from <https://webrtchacks.com/webrtc-cv-tensorflow/>

TensorFlow project.[*https://github.com/tensorflow*](https://github.com/tensorflow)

The [PASCAL](http://pascallin2.ecs.soton.ac.uk/) Visual Object Classes Homepage. Retrieved from <http://host.robots.ox.ac.uk/pascal/VOC/>

[*https://www.tensorflow.org/*](https://www.tensorflow.org/)

[*https://www.tensorflow.org/tutorials/*](https://www.tensorflow.org/tutorials/)

[*https://opensource.google.com/projects/tensorflow*](https://opensource.google.com/projects/tensorflow)

*Deploy a Flask Application on an Azure Virtual Machine.* Retrieved from <http://leifengblog.net/blog/deploy-flask-applications-on-azure-vps/>

# **Appendix C – External Resources**

|  |  |
| --- | --- |
| **GIT URL:** | *The GIT URL (if applicable).* |
| **Hosting URL:** | *The Hosting URL (if applicable).* |