Software Design Documentation

Supervisor

Date: 30th October,22

Written By: Jatin Kumawat

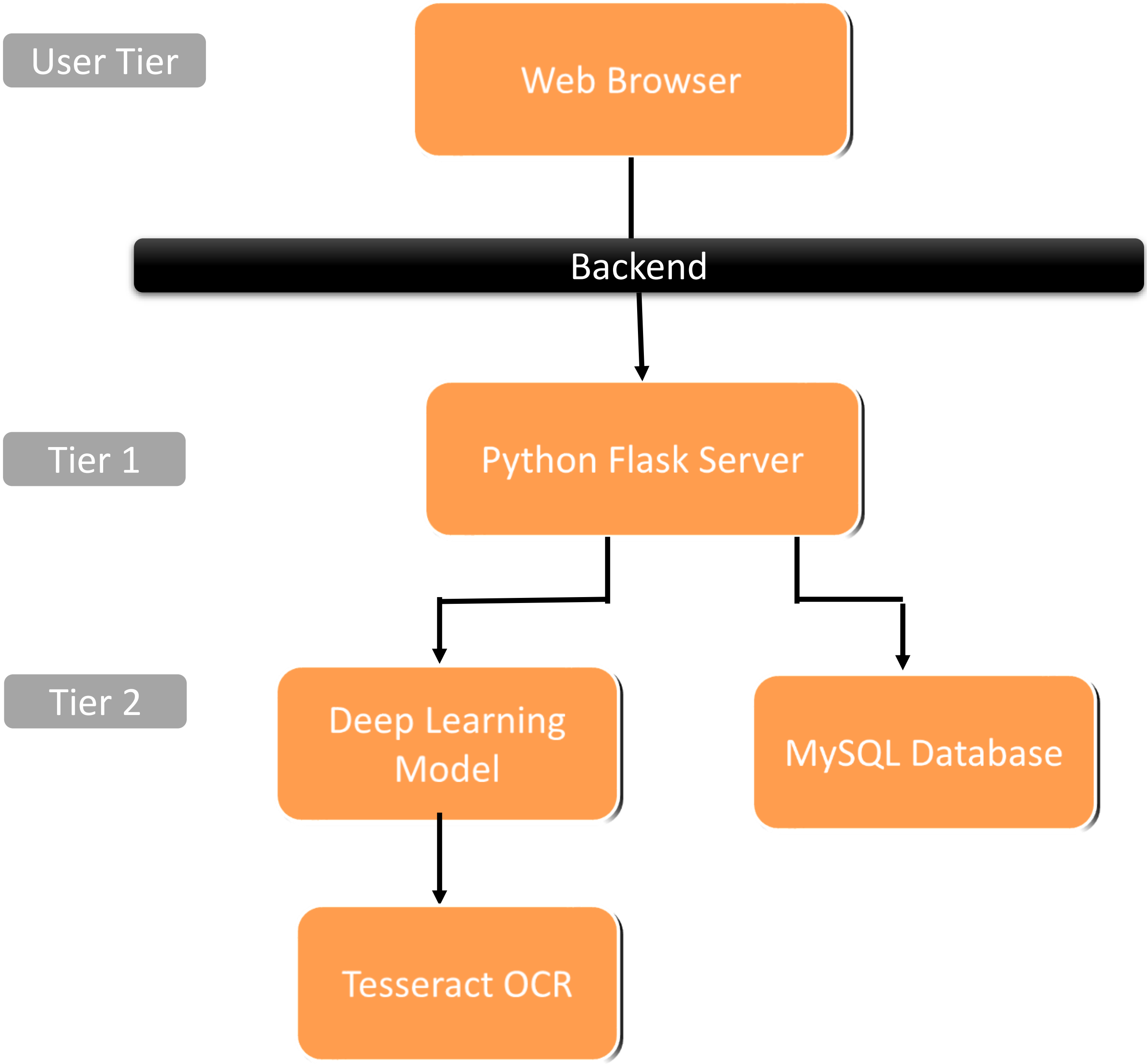
Introduction

In this Documentation, You will know about

System Overview

Supervisor, is a computer vision based AI number plate recognition project. The app will run in the web browser and local host.

Here user will choose an image and upload. As upload is clicked, deep learning model will process the input and returns output when it detects the number plate and then expected text.



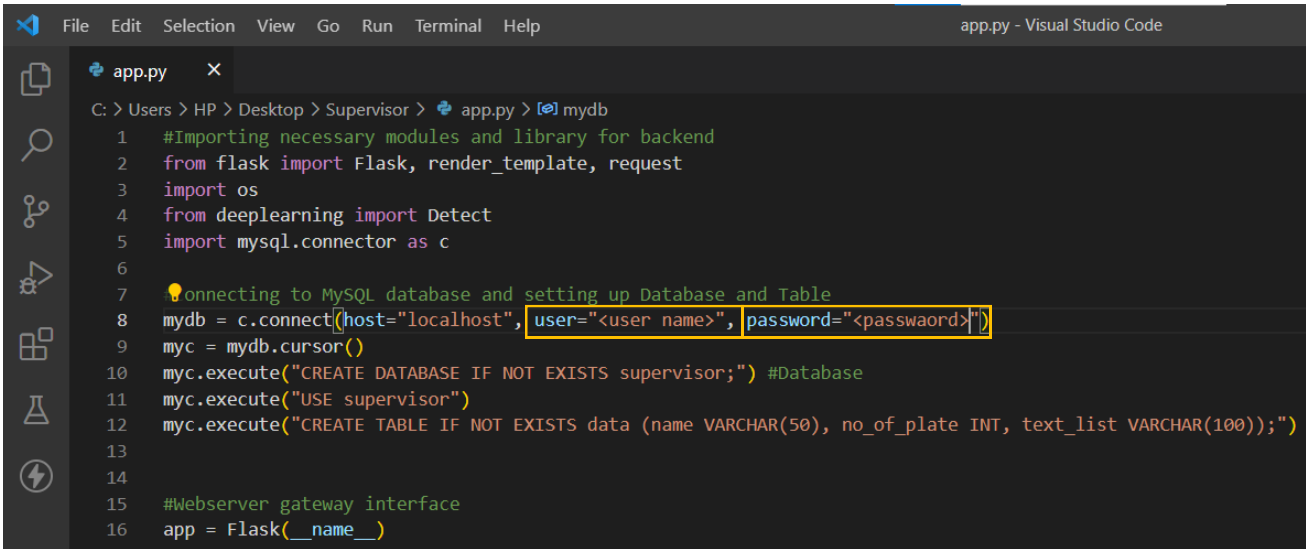
Design Considerations

Assumptions and Dependencies:

We need to install following application before running the application

1. MySQL is required for database management. Download from here and follow this installation procedure for
   Mac link
   Windows link

Then in go to app.py file and open it in any text editor like sublime or VS Code. Type your MySQL user name and password at place shown below


File
Edit
Selection
View
Go
Run
Terminal
Help
app.py - Visual Studio Code
app.py
C: > Users > HP > Desktop > Supervisor > app.py > mydb

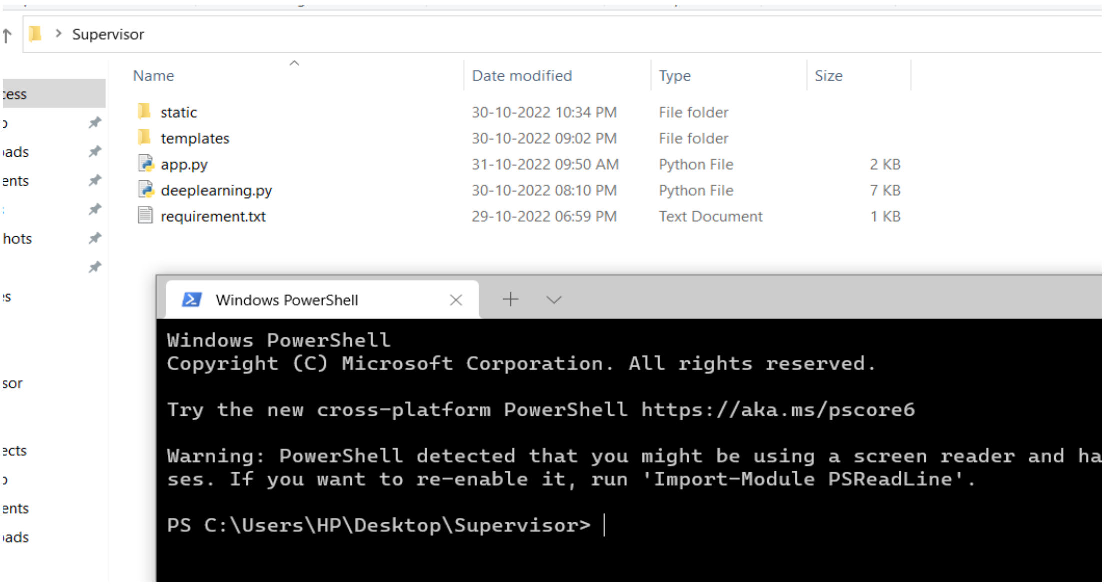
```python
#Importing necessary modules and library for backend
from flask import Flask, render_template, request
import os
from deeplearning import Detect
import mysql.connector as c

onnecting to MySQL database and setting up Database and Table
mydb = c.connect(host="localhost", user="<user name>", password="<passwaord>")
myc = mydb.cursor()
myc.execute("CREATE DATABASE IF NOT EXISTS supervisor;") #Database
myc.execute("USE supervisor")
myc.execute("CREATE TABLE IF NOT EXISTS data (name VARCHAR(50), no_of_plate INT, text_list VARCHAR(100));")


#Webserver gateway interface
app = Flask(__name__)
```

2. Tesseract-OCR for reading text from image.
Download it from here.
Follow the following installation part from link.

3. Python language to run the software.
Download python from here
And installation for Mac link and windows link

4. Python Modules for running our application.
Follow this steps:

- Create a folder name as ‘Supervisor’.
- Download all the code in this folder.
- Open your mac or windows terminal in this folder.



Name	Date modified	Type	Size
static	30-10-2022 10:34 PM	File folder	
templates	30-10-2022 09:02 PM	File folder	
app.py	31-10-2022 09:50 AM	Python File	2 KB
deeplearning.py	30-10-2022 08:10 PM	Python File	7 KB
requirement.txt	29-10-2022 06:59 PM	Text Document	1 KB

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

Warning: PowerShell detected that you might be using a screen reader and ha
ses. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\HP\Desktop\Supervisor>
```

- Type following command and press enter:

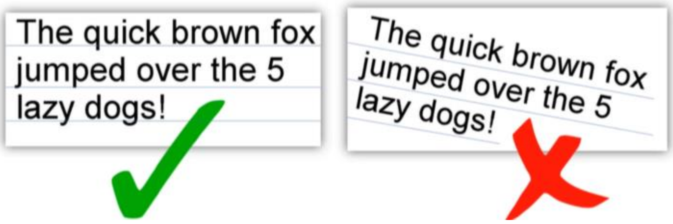
```
pip install -r .\requirement.txt
```

- Pip will automatically install all the necessary modules from requirement.txt.

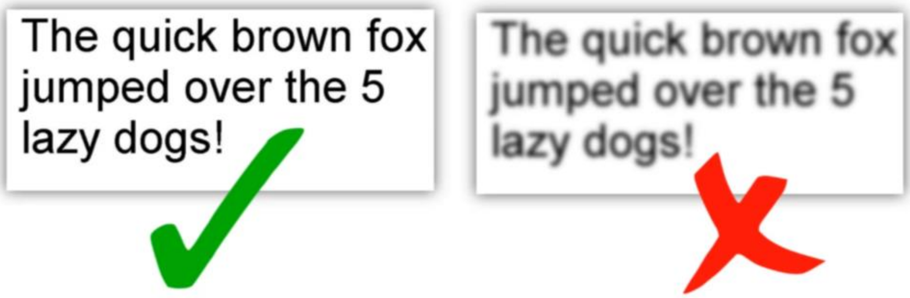
General Constraints:

Pytesseract OCR has following limitation for image of text to be detected:

1. Text aligned in order i.e. Text should not be rotated or skewed.



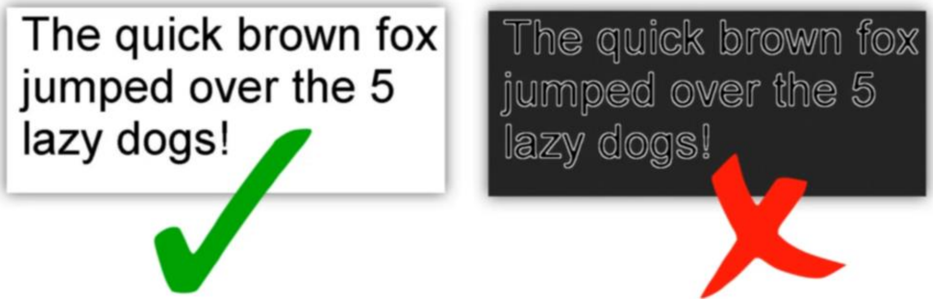
2. Text should not be blur



The quick brown fox jumped over the 5 lazy dogs!

The quick brown fox jumped over the 5 lazy dogs!

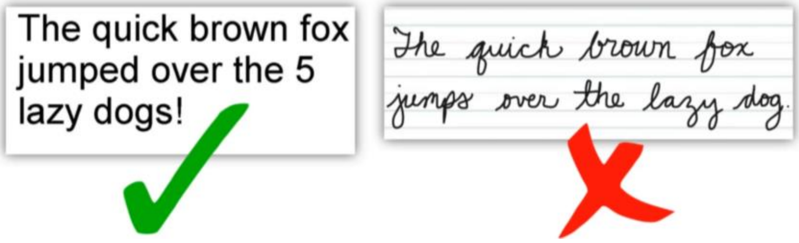
3. Simple text without any special effects.



The quick brown fox jumped over the 5 lazy dogs!

The quick brown fox jumped over the 5 lazy dogs!

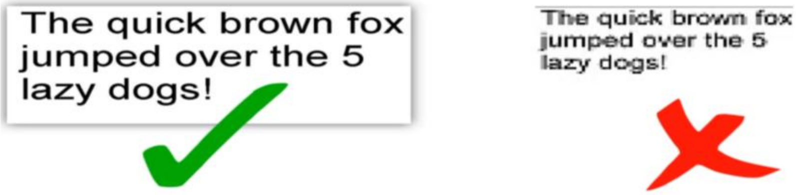
4. Does not work for cursive handwriting.



The quick brown fox jumped over the 5 lazy dogs!

The quick brown fox jumps over the lazy dog.

5. Resolution of the image should be at least 200dpi or width and height should be at least 300 pixels.



The quick brown fox jumped over the 5 lazy dogs!

The quick brown fox jumped over the 5 lazy dogs!

Goals and guidelines:

Transportation has increase significantly over past few years. It makes bustling cities livable, workable, manageable, safe, and sustainable.
And the management and supervising of vehicles by man himself has become difficult.
Every year the lives of approximately 1.3 million people are cut short as a result of a road traffic crash. Between 20 and 50 million more people suffer non-fatal injuries, with many incurring a disability as a result of their injury.

In India, many road accidents happen on highway. In major number of cases, the truck driver who is culprit is never caught and run away due to lack of a system of vehicle surveillance.

With this project, the sole purpose is to generate a automated system to distinguish vehicles by recognizing number plates.

With main social impact of creating a 24 by 7 surveillance system to monitor vehicles on highways which will help in easily identifying culprits in case of accidents. As Vehicle number is unique to every vehicle can be identified easily.

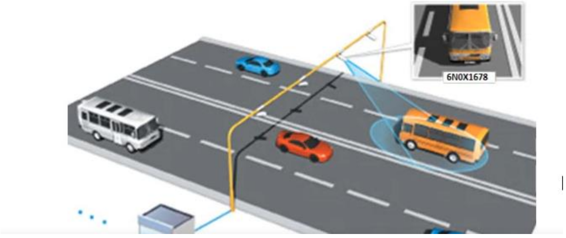
Other application of this project is in:

OOH 848Y
OOH 848Y

Vehicle Parking

LANE 4 ↓
MANUAL / SEMI AUTOMATIC TOLL LANE
LANE 3 ↓
MANUAL / SEMI AUTOMATIC TOLL LANE
LANE 2 ↓
MANUAL / SEMI AUTOMATIC TOLL LANE
LANE 1 ↓
MANUAL / SEMI AUTOMATIC TOLL LANE
STOP
STOP
STOP

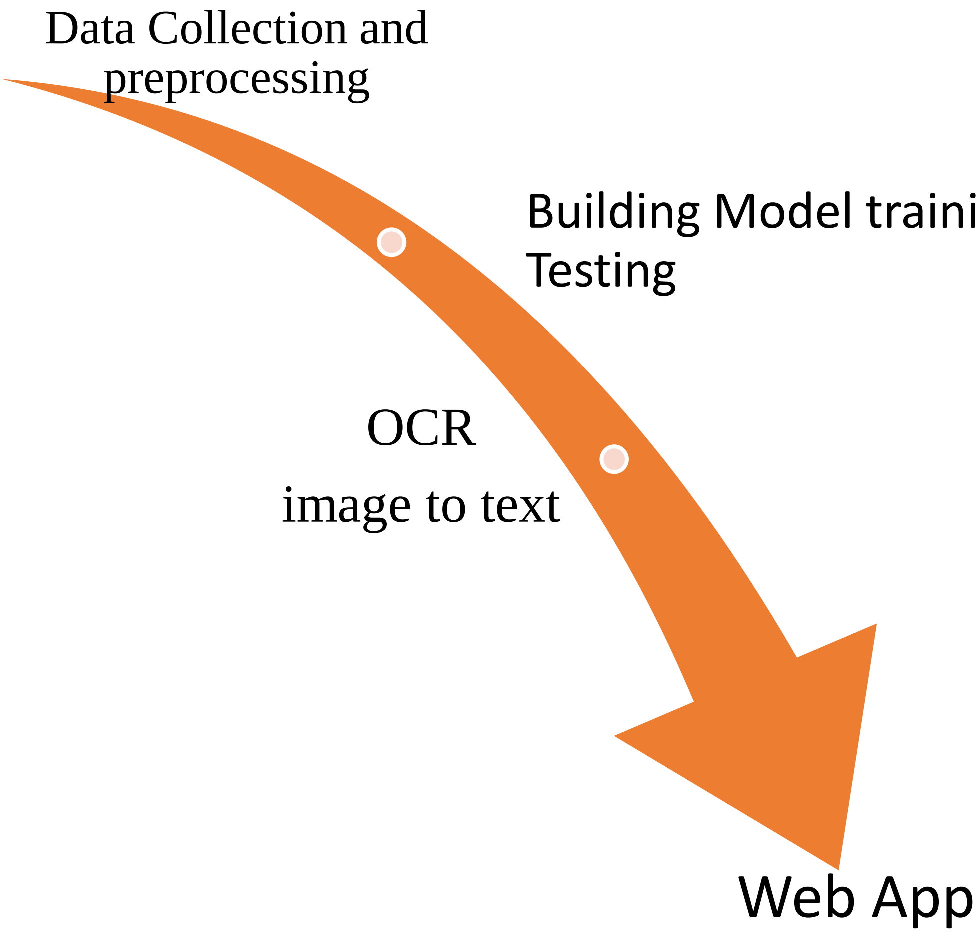
Toll Enforcement



Traffic control

Development Methods:

In making of our system, we will



Data Collection and preprocessing

Building Model training and Testing

OCR

image to text

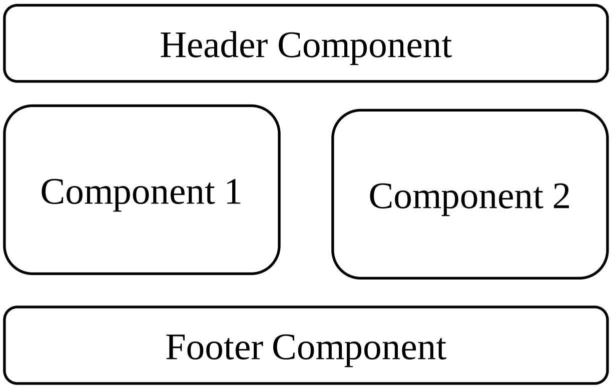
Web App

Architectural Strategies

Here, you will know about various strategies in architecture

- Component Wise approach for Frontend

This type of design helps to build consistent, solid and reusable design systems. In my Project, the main HTML is simply written in Django HTML.



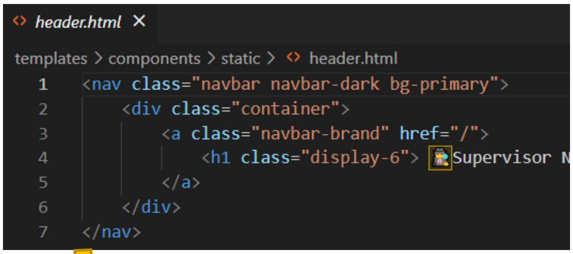
Header Component

Component 1

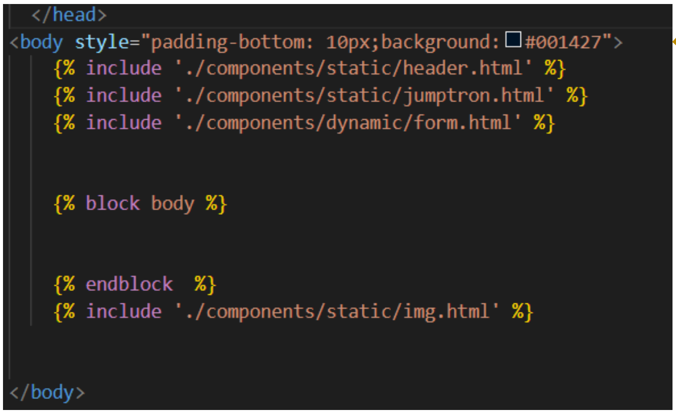
Component 2

Footer Component

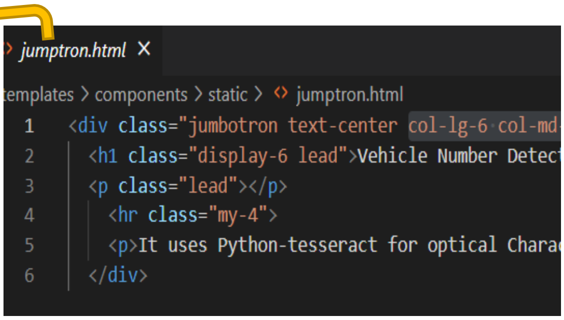
Our Home page is broken into components Navbar, image, form and etc. Each component is coded in different HTML files and merged into main file through include block of Django HTML.



```
<nav class="navbar navbar-dark bg-primary">
    <div class="container">
        <a class="navbar-brand" href="/">
            <h1 class="display-6">Supervisor N
        </a>
    </div>
</nav>
```



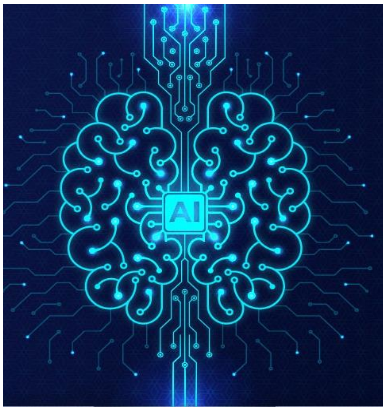
```
  </head>
<body style="padding-bottom: 10px;background:□#001427">
    {% include './components/static/header.html' %}
    {% include './components/static/jumptron.html' %}
    {% include './components/dynamic/form.html' %}


    {% block body %}


    {% endblock  %}
    {% include './components/static/img.html' %}

</body>
```



```
<div class="jumbotron text-center col-lg-6 col-md-
  <h1 class="display-6 lead">Vehicle Number Detect
  <p class="lead"></p>
    <hr class="my-4">
    <p>It uses Python-tesseract for optical Chara
  </div>
```

Further classified as static and dynamic components. This increase code reusability

- AI functionality on Backend



My Project, Our Model will detect image and process, it requires a proper setup to run. It also depends on other application like MySQL. Frontend involves only web browser which cannot execute this complex model.
Integrated AI in backend

- Rest API usage

Our backend uses Flask which will use POST and GET method of Rest API. This is simple, friendly to developer and uses web standards.

- SQL for Database

My application uses SQL for database purpose. Our application stores information about number of plates detected and text identified.

SQL databases are a better fit for heavy duty or complex transactions because it’s more stable and ensure data integrity.

System Architecture

The Project was divided into and assigned to 5 sub systems

- Labelling and Preprocessing

Collecting 200+ images from different sources. Labeled each and every image of vehicle manually using Labelimg. Its time consuming process.
Labelimg is a open source image annotation tool.
Also involve dividing data into train and test set.

- Saving the model

Involves building, training and validation of prediction of model. After the satisfactory results are obtained, the model is saved for further integration.

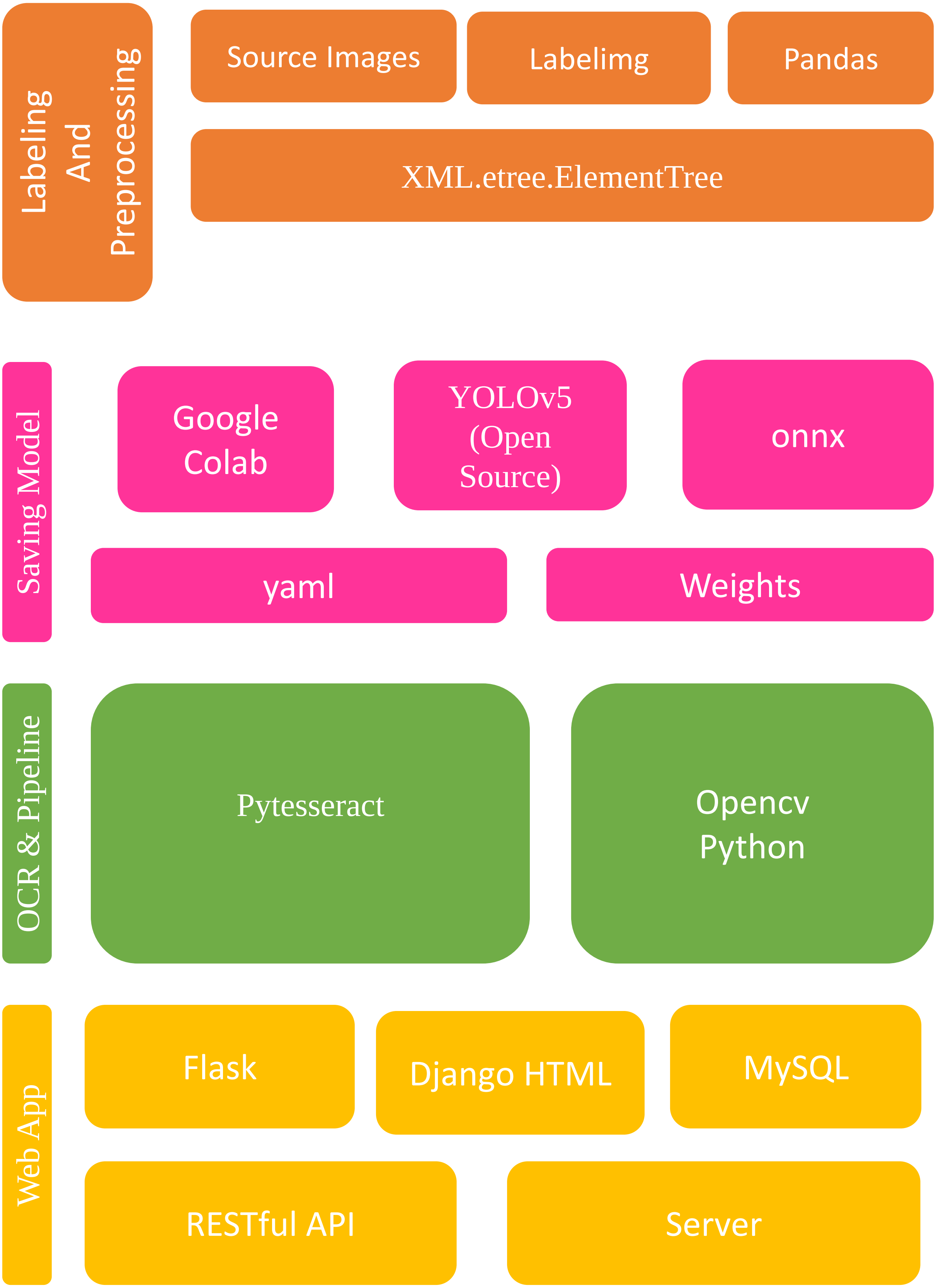
Our model is built from an open source Model YOLOv5

- OCR and Pipeline

This subsystem involves joining the pipeline created during saving the model part. With this, we include Pytesseract to get text from image of plate.

- RESTful API and SQL

Creating web application. On back side, server processes the user uploaded image and returns the vehicle number.
Do database management and stores information.



Labeling And Preprocessing

Source Images

Labelimg

Pandas

XML.etree.ElementTree

Saving Model

Google Colab

YOLOv5 (Open Source)

onnx

yaml

Weights

OCR & Pipeline

Pytesseract

Opencv Python

Web App

Flask

Django HTML

MySQL

RESTful API

Server

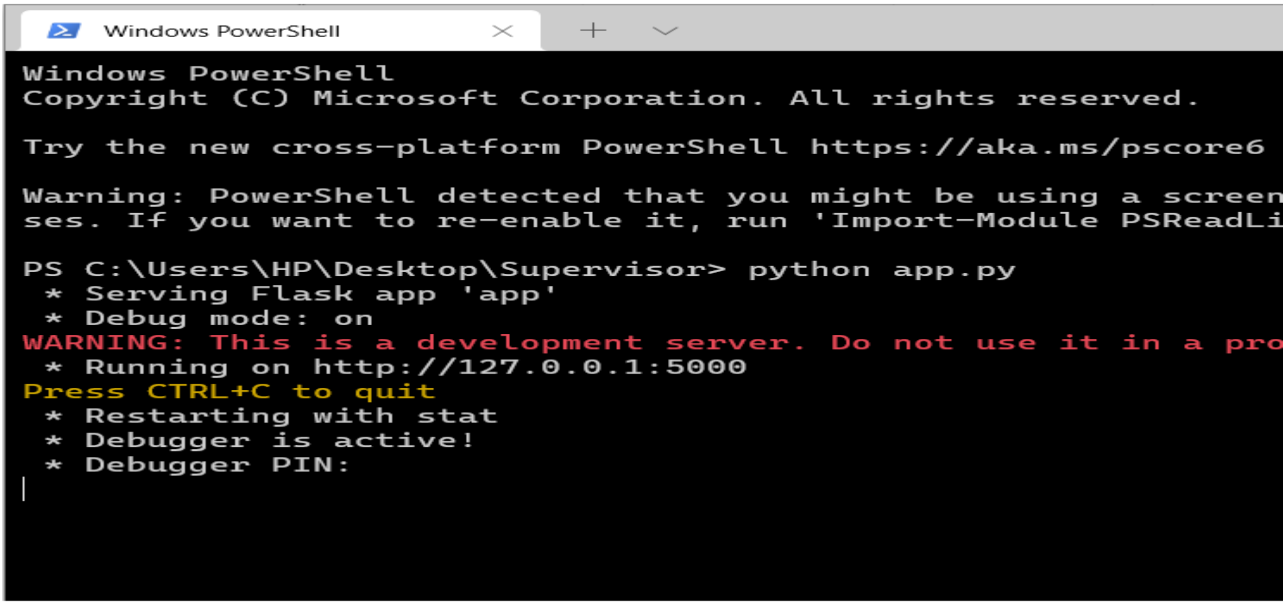
Detailed System Design

Running the Application:

- Install all the Dependencies stated above in document.
- Open the folder Supervisor in your mac or windows terminal
- Write following command and press enter

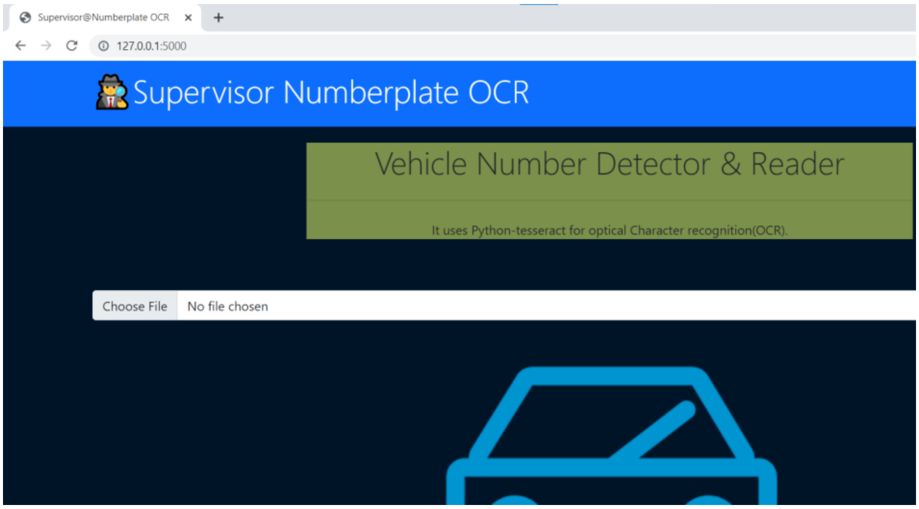
```
python app.py
```

- Server will start running.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

Warning: PowerShell detected that you might be using a screen
ses. If you want to re-enable it, run 'Import-Module PSReadLi

PS C:\Users\HP\Desktop\Supervisor> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a pro
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN:
```

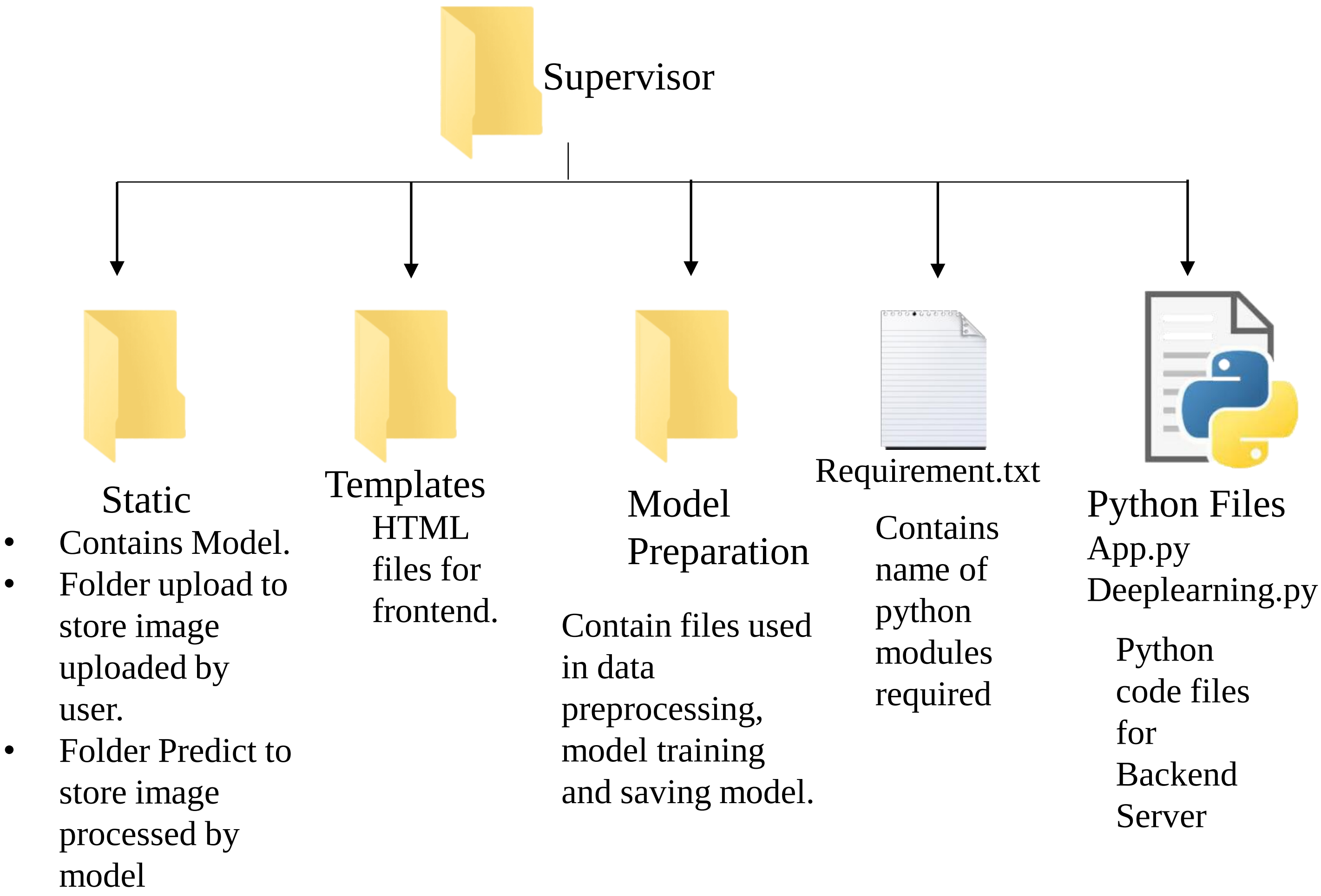
- In your Web Browser, go to http://127.0.0.1:5000/



Vehicle Number Detector & Reader

It uses Python-tesseract for optical Character recognition(OCR).

Folder Structure:



Supervisor

Static

- Contains Model.
- Folder upload to store image uploaded by user.
- Folder Predict to store image processed by model

Templates

HTML files for frontend.

Model Preparation

Contain files used in data preprocessing, model training and saving model.

Requirement.txt

Contains name of python modules required

Python Files

App.py

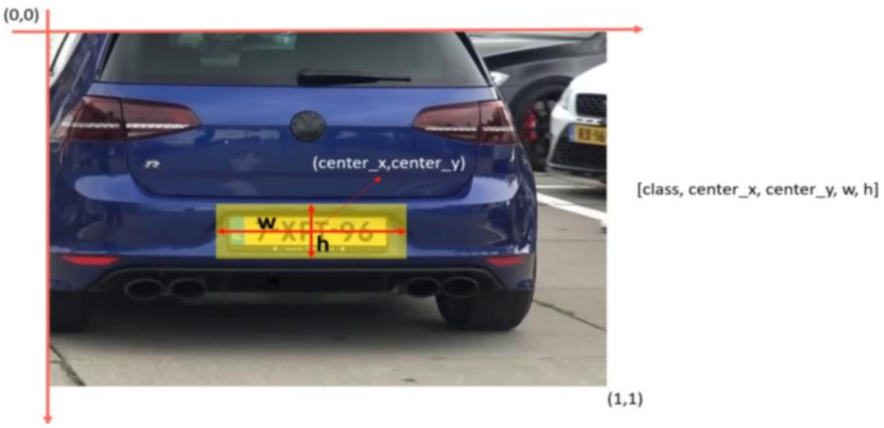
Deeplearning.py

Python code files for Backend Server

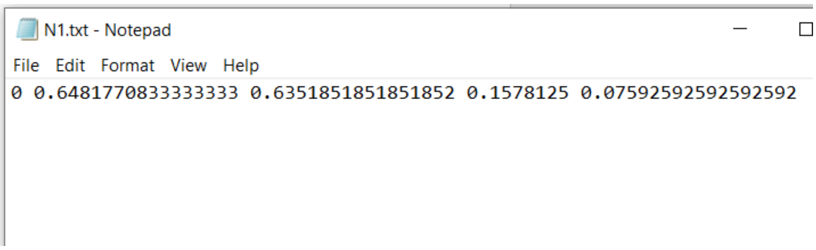
Labeling:

Collected 200+ image showing vehicle and number plate from internet.
Using labelimg, marked down the section of number plate manually and saved it in form of data in xml file.
Saved it in /Supervisor/Model_Preparation/yolo5/images
In the Google colab file
/Supervisor/Model_Preparation/yolov5/Data_Preparation.ipynb

For each image, calculated center x, center y of plate, width and height of region of interest.



Saving this information for each image as text file.
Since we are having 225 files, saving 200 files as train data in /Supervisor/Model_Preparation/yolov5/data_images/train and 25 for test in /Supervisor/Model_Preparation/yolov5/data_images/test

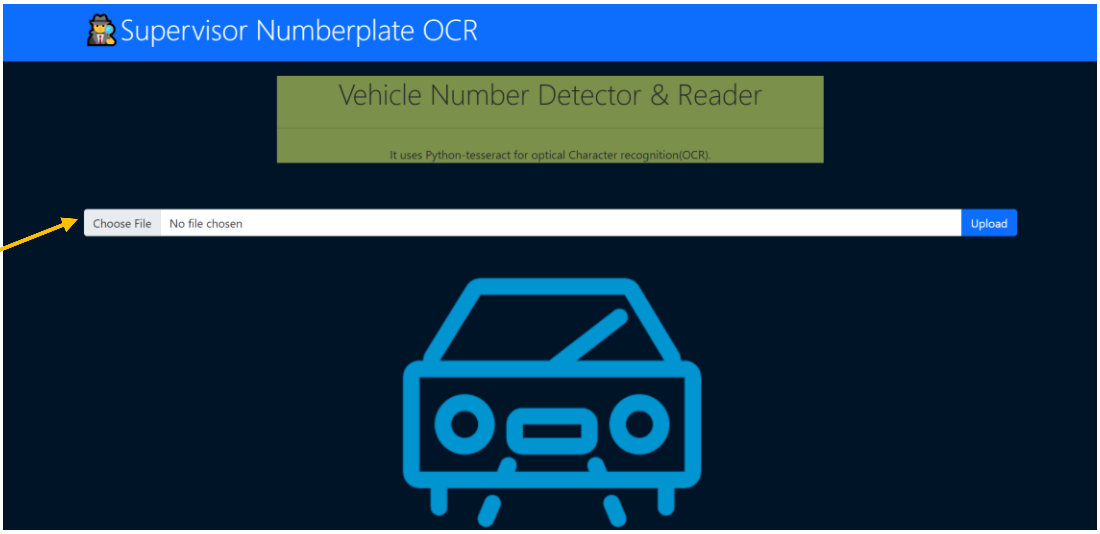


```
0 0.6481770833333333 0.6351851851851852 0.1578125 0.07592592592592592
```

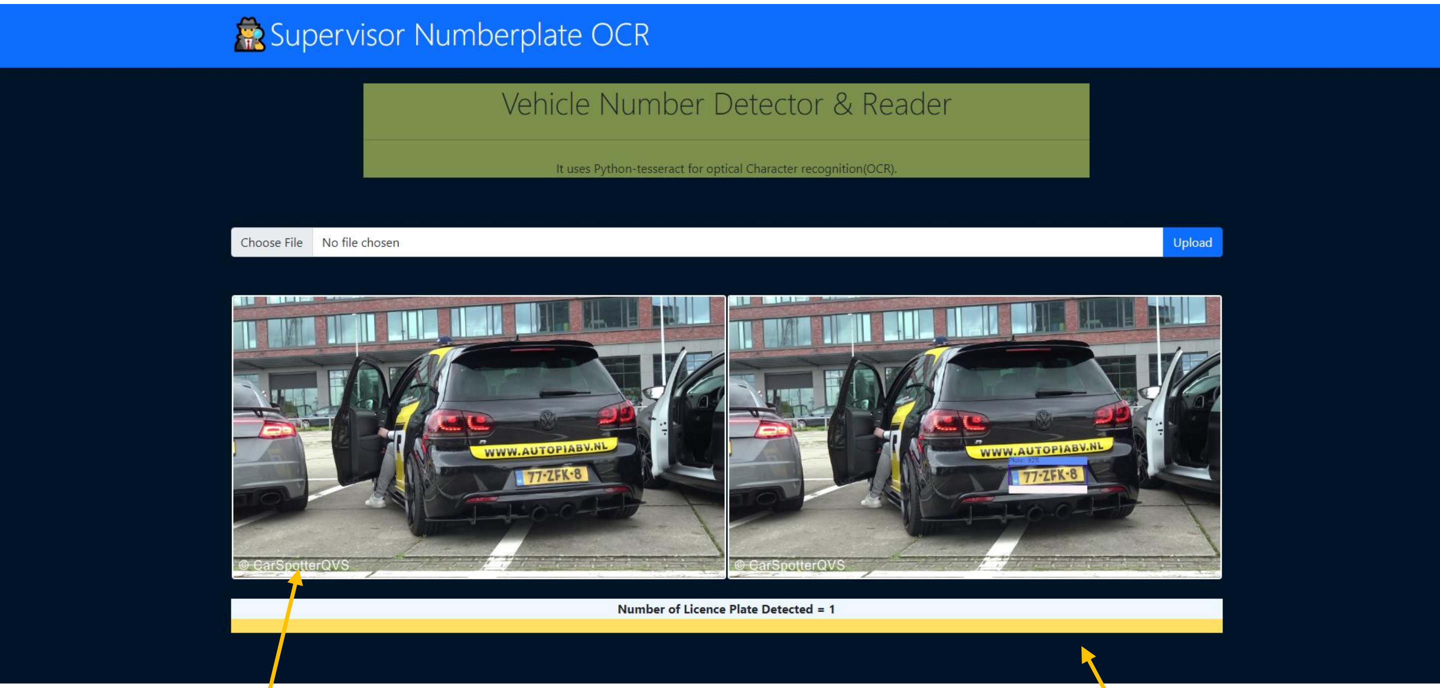
Saving the Model:

In the file /Supervisor/Model_Preparation/Training_Model.ipynb
Model is trained and prepared. And then made exportable.
Also data preprocessing, training of model and saving the model is done in google colab. Then the code is downloaded and putted in supervisor folder.
All the files are generated during above processes are present in folder yolo5.

Frontend working:


Supervisor Numberplate OCR
Vehicle Number Detector & Reader
It uses Python-tesseract for optical Character recognition(OCR).
Choose File
No file chosen
Upload

User upload image here.



Supervisor Numberplate OCR

Vehicle Number Detector & Reader

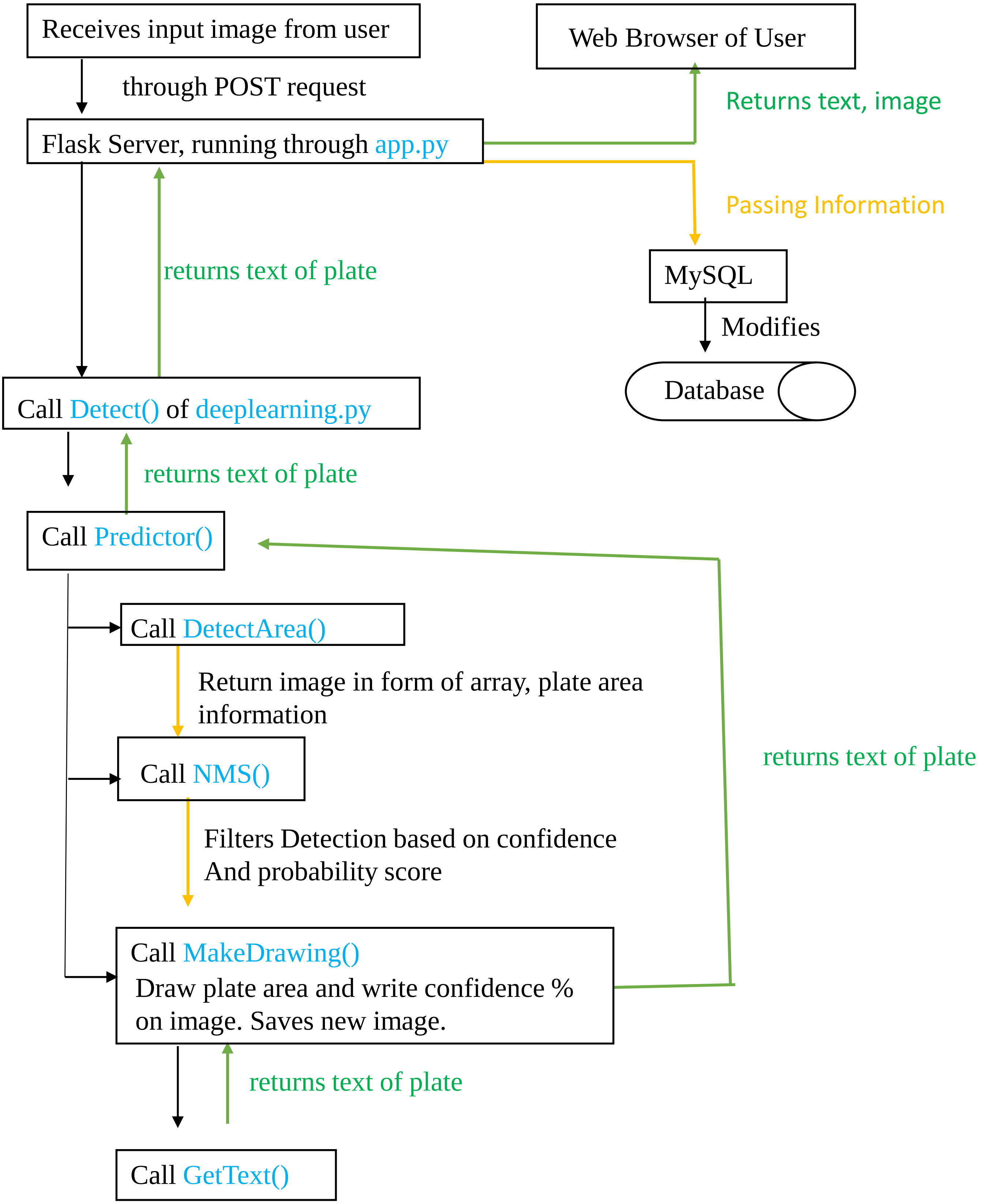
It uses Python-tesseract for optical Character recognition(OCR).

Number of Licence Plate Detected = 1

Also displays the number of Number plate and text written on each plate

Server processes the input, after that image with and without vehicle plate highlighted is displayed

Backend working:



Receives input image from user

through POST request

Flask Server, running through app.py

Web Browser of User

Returns text, image

Passing Information

MySQL

Modifies

Database

returns text of plate

Call Detect() of deeplearning.py

returns text of plate

Call Predictor()

Call DetectArea()

Return image in form of array, plate area information

Call NMS()

Filters Detection based on confidence
And probability score

Call MakeDrawing()
Draw plate area and write confidence %
on image. Saves new image.

returns text of plate

returns text of plate

Call GetText()