How To Develop Frontend for PPE Violation Detection

Step 1 - Creating a Workspace Directory

First we create a directory using the following command where we keep all the files related to the user interface of our PPE Violation Detection application.

```
mkdir PPE-Violation-Detection
```

```
cd PPE-Violation-Detection/
```

Step 2 - Creating a Virtual Environment

Now we create a virtual environment using the following command in our workspace directory.

```
python3 -m venv venv/
```

```
source venv/bin/activate
```

Step 3 - Installing Requirements

Now, its time to install the packages that will be required for our frontend development.

```
pip3 install Flask
```

```
pip3 install opencv-python
```

```
pip3 install validators
```

We will also be installing some additional requiremnets for opencv-python required by Ubuntu platform.

```
sudo apt update
```

```
sudo apt install ffmpeg libsm6 libxext6 -y
```

> ***Note:*** *For now we will focus only on how to upload our test video and save it onto the server. Other functionalities of the frontend will be explained soon as we will move forward in our project.*

Step 4 - Creating HTML File

Let's head towards our HTML skeleton for the user interface of our PPE Violation Detection application.

Since it is a flask application, we will be needing a *template/* directory to render our HTML file. So first, we create *template/* directory.

```
mkdir templates
```

```
cd templates/
```

Now, create an `index.html` file using the following command:

```
nano index.html
```

Now copy paste the following script in your *nano* editor.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!--=============== REMIXICONS ===============-->
    <link href="https://cdn.jsdelivr.net/npm/remixicon@2.5.0/fonts/remixicon.css"
rel="stylesheet">

    <!--=============== CSS ===============-->
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename="css/styles.css") }}">

    <title>PPE Violation Detection</title>
</head>

<body>
    <!--=============== AJAX ===============-->
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```
    <script>
        $(document).ready(function () {
            $('#myform').submit(function (event) {
                event.preventDefault()
                //submit_form(event);
            });
        });

        $(document).ready(function () {
            $('#alert_email_checkbox').change(function (event) {
                event.preventDefault()
                data = {
                    'alert_email_checkbox':
$('#alert_email_checkbox').is(':checked'),
                    'alert_email_textbox': $('#alert_email_textbox').val(),
                }
                $.ajax({
                    type: 'POST',
                    url: '/submit',
                    data: data,
                    success: function (data) {
                        alert(data);
                    },
                    error: function (error) {
                        alert('Checkbox submission failed!');
                    }
                });
            });
        });


        function upload_file() {
            $.ajax({
                type: 'POST',
                url: '/submit',
                data: new FormData($('#myform')[0]),//formData,
                processData: false,
                contentType: false,
                cache: false,          //Required
                success: function (data) {
                    alert(data);
                },
                error: function (error) {
                    alert('Form submission failed!');
                }
            });
        }

        function download_file() {
            data = {
                'download_button': 'True',
            }
            $.ajax({
```

```
                type: 'POST',
                url: '/submit',
                data: data,
                xhrFields: {
                    responseType: 'blob' // to avoid binary data being mangled on
charset conversion
                },
                // to download file as an attachment
                //Reference - https://stackoverflow.com/questions/16086162/handle-
file-download-from-ajax-post
                success: function (blob, status, xhr) {
                    // check for a filename
                    var filename = "";
                    var disposition = xhr.getResponseHeader('Content-
Disposition');
                    if (disposition && disposition.indexOf('attachment') !== -1) {
                        var filenameRegex = /filename[^;=\n]*=((['"]).*?\2|
[^;\n]*)/;
                        var matches = filenameRegex.exec(disposition);
                        if (matches != null && matches[1]) filename =
matches[1].replace(/['"]/g, '');
                    }

                    if (typeof window.navigator.msSaveBlob !== 'undefined') {
                        // IE workaround for "HTML7007: One or more blob URLs were
revoked by closing the blob for which they were created. These URLs will no longer
resolve as the data backing the URL has been freed."
                        window.navigator.msSaveBlob(blob, filename);
                    } else {
                        var URL = window.URL || window.webkitURL;
                        var downloadUrl = URL.createObjectURL(blob);

                        if (filename) {
                            // use HTML5 a[download] attribute to specify filename
                            var a = document.createElement("a");
                            // safari doesn't support this yet
                            if (typeof a.download === 'undefined') {
                                window.location.href = downloadUrl;
                            } else {
                                a.href = downloadUrl;
                                a.download = filename;
                                document.body.appendChild(a);
                                a.click();
                            }
                        } else {
                            window.location.href = downloadUrl;
                        }
                        setTimeout(function () { URL.revokeObjectURL(downloadUrl);
}, 100); // cleanup
                    }
                },
                error: function (error) {
                    alert('Form submission failed!');
                }
```

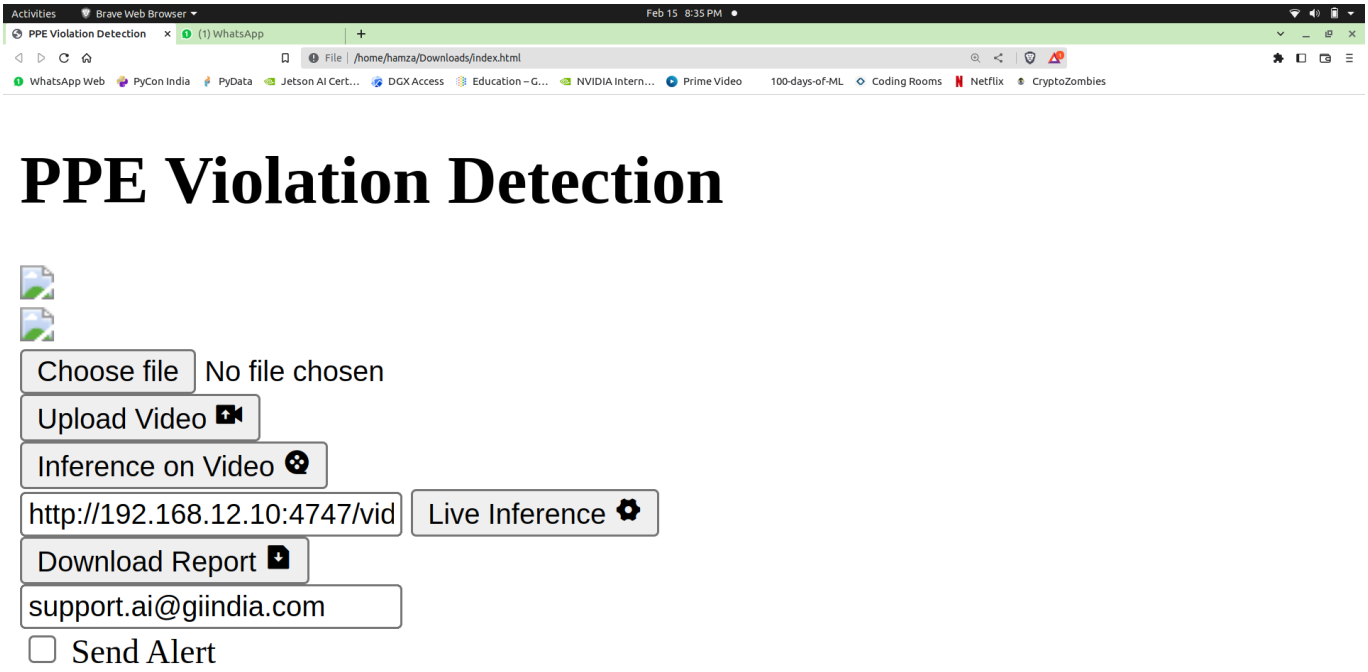
```
            });
        }

        function video_inference() {
            data = {
                'inference_video_button': 'true',
            }
            $.ajax({
                type: 'POST',
                url: '/submit',
                data: data,
                success: function (data) {
                    //alert(data);
                    //window.location.href = '/';
                },
                error: function (error) {
                    alert('Video inference failed!');
                }
            });
        }

        function live_inference() {
            data = {
                'live_inference_button': 'true',
                'live_inference_textbox': $('#ip_address_textbox').val(),
            }
            $.ajax({
                type: 'POST',
                url: '/submit',
                data: data,
                success: function (data) {
                    //alert(data);
                    //window.location.href = '/';
                },
                error: function (xhr, status, error) {
                    alert(xhr.responseText);
                }
            });
        }
    </script>

    <!-- ============== HEADER ============== -->
    <header class="header" id="header">
        <div class="title">
            <h1>PPE Violation Detection</h1>
        </div>
    </header>

    <!-- ============== VIDEOS ============== -->

    <div class="gallery_container">

        <div class="gallery">
            <img src="{{ url_for('video_raw') }}" />
```

```
        </div>

        <div class="gallery">
            <img src="{{ url_for('video_processed') }}" />
        </div>
    </div>


    <!-- =============== OPERATIONS =============== -->
    <div class="operations_wrapper">
        <form id="myform" enctype="multipart/form-data" method="post">
            <div class="btn">
                <div class="upload__button">
                    <input type="file" class="custom-file-input" name="video"
id="video" value="video">
                    <button type="submit" class="btn-primary"
name="video_upload_button" id="video_upload_button"
                        onclick="upload_file()">Upload Video
                        <i class="ri-video-upload-fill button__icon"></i>
                    </button>
                </div>
                <div class="live__button">
                    <button class="btn-primary" name="inference_video_button"
id="inference_video_button"
                        onclick="video_inference()">Inference on Video
                        <i class="ri-movie-2-fill button__icon"></i>
                    </button>
                </div>
                <div class="inference__button">
                    <input type="text" class="ip_address-input"
name="ip_address_textbox" id="ip_address_textbox"
                        placeholder="http://192.168.12.10:4747/video"
value="http://192.168.12.10:4747/video">
                    <button type="submit" class="btn-primary"
name="live_inference_button" id="live_inference_button"
                        onclick="live_inference()">Live Inference
                        <i class="ri-settings-4-fill button__icon"></i>
                    </button>
                </div>
                <div class="download__button">
                    <button type="submit" class="btn-primary"
name="download_button" id="download_button"
                        onclick="download_file()">Download Report
                        <i class="ri-file-download-fill button__icon"></i>
                    </button>
                </div>
                <div class="email__sending">
                    <div class="send__email">
                        <input type="email" placeholder="Enter Valid Mail"
value="support.ai@giindia.com"
                            name="alert_email_textbox" id="alert_email_textbox">
                    </div>
                    <div class="toggle__content">
                        <label class="toggle__label">
```

```
                            <input type="checkbox" class="toggle__check"
name="alert_email_checkbox"
                                id='alert_email_checkbox' />
                            <span class="email__label">Send Alert</span>
                            <div class="toggle__rail">
                                <span class="toggle__circle"></span>
                                <span class="toggle__border"></span>
                            </div>
                        </label>
                    </div>
                </div>
            </div>
        </form>
    </div>
</body>

</html>
```

Save the file by pressing `Ctrl+X` → `Y` → `Enter`

If you open `index.html` in a browser, it should be looking someting like the following screenshot.



Move back to parent directory

```
cd ..
```

Step 3 - Styling Our User Interface

Does the user interface looks satisfactory? No, lets add some styling to our user interface.

In flask application, we cannot directly import our CSS file in `index.html`. There something called static files in flask application where all the CSS, JavaScript and other types of scripts are kept. So, lets create our `styles.css` as described below:

1. First, create *static/* directory using the following command:

```
mkdir static
```

```
cd static/
```

2. Now, specify the type folders that we will be needing for storing different files for our flask application.

```
# To store our CSS file

mkdir css
```

```
# To store the uploaded video

mkdir video
```

3. Finally, creating and editing the `styles.css` file.

```
cd css/
```

```
nano styles.css
```

Now copy paste the following script into your *nano* editor.

```
/*=============== GOOGLE FONTS ===============*/
@import url('https://fonts.googleapis.com/css2?
family=Roboto+Mono:wght@500&display=swap');

/*=============== VARIABLES CSS ===============*/
:root {
  /*========== Colors ==========*/
  --light: #F6FAFD;
  --dark: #122272;
  --pri-blue: #193FAF;
  --sec-blue: #17A5F8;
```

```
  --pri-green: #23C99D;
  --alert: #FE7F0E;

  /*========== Font and typography ==========*/
  --body-font: 'Poppins', sans-serif;
  --h1-font-size: 1.5rem;
  --medium-font-size: 0.973rem;
  --small-font-size: 0.813rem;
  --smaller-font-size: 0.75rem;
}

/*Responsive typography*/
@media screen and (min-width: 1024px) {
  :root {
    --h1-font-size: 1.6875rem;
    --medium-font-size: 1.125rem;
    --small-font-size: .875rem;
    --smaller-font-size: .813rem;
  }
}

/*=============== BASE ===============*/

* {
  box-sizing: border-box;
  padding: 0;
  margin: 0;
}

body {
  font-size: 1em;
  font-weight: 500;
  font-family: var(--body-font);
  background-color: var(--light);
}

img,
video {
  max-width: 100%;
  height: auto;
}

/*form :where(i, p) {
  /* color: var(--pri-blue);
} */

form i {
  font-size: 1em;
}

form button {
  font-size: 16px;
  border: none;
  font: var(--body-font);
```

```
  background-color: var(--first-color);
  cursor: pointer;
  color: #F6FAFD;
}

a {
  text-decoration: none;
  color: var(--sec-blue);
}

.main {
  padding: 0.5rem;
}

/*=============== GALLERY ===============*/

.gallery_container {
  /*width: 100%;*/
  margin-left: 10%;
  /*position: absolute;*/
  display: flex;
  width: 80%;
}

.gallery {
  flex: 1;
  border-radius: 1em;
  outline: 3px dashed var(--pri-blue);
  margin: 2em;
}

.gallery:first-child {
  margin-right: 3em;
}

.gallery img {
  width: 100%;
  height: 100%;
  border-radius: 1em;
}

/*=============== OPERATIONS ===============*/
.operations_wrapper {
  width: 90%;
  margin: 5% 5% 0% 5%;
  background-color: #fff;
  border-radius: 1em;
}

/*=============== HEADING ===============*/
.title {
  text-align: center;
}
```

```css
.header {
  font-size: 2em;
  font-weight: 600;
  text-align: center;
  height: 5em;
  padding-top: 1em;
  color: #F6FAFD;
  margin-bottom: 5rem;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.1);
  border-bottom: 1rem;
  font-family: 'Roboto Mono', monospace;
  background: var(--pri-blue);
  background: linear-gradient(110deg, var(--dark) 38%, var(--pri-blue) 100%);
}

/*=============== BUTTONS ===============*/
.btn {

  padding: 2em;
  display: flex;
  justify-content: space-evenly;
  align-items: center;

}

.upload__button,
.download__button,
.live__button,
.inference__button {
  display: inline-flex;
  align-items: center;
  background-color: var(--pri-blue);
  color: #fff;
  border-radius: 0.5rem;
  padding: 0.5rem 1.5rem;
  cursor: pointer;
}

.upload__button:hover,
.download__button:hover,
.live__button:hover,
.inference__button:hover,
.email__sending:hover {
  background-color: var(--pri-blue);
  background: var(--pri-green);
  background: linear-gradient(144deg, var(--pri-green) 20%, var(--sec-blue) 100%);
  color: var(--light);
}

.upload__button:hover button,
.download__button:hover button,
.live__button:hover button,
.inference__button:hover button {
  color: var(--light);
```

```
}

.button__icon {
  margin-left: 0.25rem;
  transition: 0.3s;
  color: var(--light);
  font-size: var(--h1-font-size);
}

.download__button:hover .button__icon {
  transform: translateY(0.25rem);
}

.upload__button:hover .button__icon {
  transform: translateY(-0.25rem);
}

.inference__button:hover .button__icon {
  transform: rotate(1rad);
}

.ip_address-input,
.custom-file-input {
  margin-right: 1rem;
}

/*=============== TOGGLE SWITCH ===============*/
.email__sending {
  border-radius: 0.5rem;
  padding: 1rem 5rem 1rem 1rem;
  background-color: var(--pri-blue);
  display: inline-flex;
  align-items: center;
  /* margin: 2rem 5rem; */
}

.toggle__content {
  position: relative;
  margin-left: 2rem;
  bottom: 0.74rem;
}

.email__label {
  position: relative;
  left: 4rem;
  top: 0.85rem;
}

.toggle__label {
  cursor: pointer;
  padding-block: 0.5rem;
}

.toggle__check {
```

```
  display: none;
}

.toggle__rail {
  position: relative;
  width: 52px;
  height: 4px;
  background-color: var(--light);
  border-radius: 2rem;
}

.toggle__circle {
  display: block;
  width: 24px;
  height: 24px;
  background-color: var(--alert);
  /* box-shadow: inset 0 0 0 4px var(--dark); */
  border-radius: 50%;
  position: absolute;
  left: 0;
  top: 0;
  bottom: 0;
  margin: auto 0;
  transition: transform 0.4s, box-shadow 0.4s;
  z-index: 2;
}

.toggle__border {
  position: absolute;
  width: 32px;
  height: 32px;
  background-color: var(--light);
  border-radius: 50%;
  left: -4px;
  top: 0;
  bottom: 0;
  margin: auto 0;
  transition: transform 0.4s;
}

/*Toggle animation effects*/
.toggle__check:checked~.toggle__rail .toggle__circle {
  transform: translateX(28px);
  box-shadow: inset 0 0 0 12px var(--pri-green);
}

.toggle__check:checked~.toggle__rail .toggle__border {
  transform: translateX(28px);
}

/*=============== BREAKPOINTS ===============*/
/*For small devices*/

/*For large devices*/
```

Save the file by pressing `Ctrl+X` → `Y` → `Enter`

Move back to parent directory

```
cd ..
```

Again

```
cd ..
```

Step 4 - Creating the Flask application

Now we create our flask application in *PPE-Violation-Detection/* directory.

```
nano app.py
```

Copy and paste the following python script into your *nano* editor.

```
import os.path
import cv2
import validators
from flask import Flask, render_template, request, Response

# Initialize the Flask application
app = Flask(__name__)
app.config["VIDEO_UPLOADS"] = "static/video"
app.config["ALLOWED_VIDEO_EXTENSIONS"] = ["MP4", "MOV", "AVI", "WMV", "WEBM"]

# Secret key for the session
app.config['SECRET_KEY'] = 'ppe_violation_detection'


def allowed_video(filename):
    """
    A function to check if the uploaded file is a video

    Args:
        filename (str): name of the uploaded file

    Returns:
        bool: True if the file is a video, False otherwise
    """
    if "." not in filename:
        return False
```

```
    extension = filename.rsplit(".", 1)[1]

    if extension.upper() in app.config["ALLOWED_VIDEO_EXTENSIONS"]:
        return True
    else:
        return False


def generate_raw_frames():
    """
    A function to yield unprocessed frames from stored video file or ip cam stream

    Yields:
        bytes: a frame from the video file or ip cam stream
    """
    pass


def generate_processed_frames(conf_=0.25):
    """
    A function to yield processed frames from stored video file or ip cam stream
after violation detection

    Args:
        conf_ (float, optional): confidence threshold for the detection. Defaults
to 0.25.

    Yields:
        bytes: a processed frame from the video file or ip cam stream
    """
    pass


@app.route('/video_raw')
def video_raw():
    """
    A function to handle the requests for the raw video stream

    Returns:
        Response: a response object containing the raw video stream
    """

    return Response(generate_raw_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')


@app.route('/video_processed')
def video_processed():
    """A function to handle the requests for the processed video stream after
violation detection

    Returns:
        Response: a response object containing the processed video stream
```

```
    """
    # default confidence threshold
    conf = 0.75
    return Response(generate_processed_frames(conf_=conf), mimetype='multipart/x-
mixed-replace; boundary=frame')


@app.route('/', methods=["GET", "POST"])
def index():
    """
    A function to handle the requests from the web page

    Returns:
        render_template: the index.html page (home page)
    """
    return render_template('index.html')


@app.route('/submit', methods=['POST'])
def submit_form():
    """
    A function to handle the requests from the HTML form on the web page

    Returns:
        str: a string containing the response message
    """
    # global variables
    # noinspection PyGlobalUndefined
    global vid_path, video_frames, frames_buffer

    # if the request is a POST request made by user interaction with the HTML form
    if request.method == "POST":
        # print(request.form)vid_ip_path.startswith('http://')

        # handle video upload request
        if request.files:
            video = request.files['video']

            # check if video file is uploaded or not
            if video.filename == '':
                # display a flash alert message on the web page
                return "That video must have a file name"

            # check if the uploaded file is a video
            elif not allowed_video(video.filename):
                # display a flash alert message on the web page
                return "Unsupported video. The video file must be in MP4, MOV,
AVI, WEBM or WMV format."
            else:
                # default video name
                filename = 'vid.mp4'
                # ensure video size is less than 200MB
                if video.content_length > 200 * 1024 * 1024:
                    return "Error! That video is too large"
```

```
                else:
                    # noinspection PyBroadException
                    try:
                        video.save(os.path.join(app.config["VIDEO_UPLOADS"],
filename))
                        return "That video is successfully uploaded"
                    except Exception as e:
                        print(e)
                        return "Error! The video could not be saved"

        # handle inference request for a video file
        elif 'inference_video_button' in request.form:
            vid_path = os.path.join(app.config["VIDEO_UPLOADS"], 'vid.mp4')
            video_frames = cv2.VideoCapture(vid_path)
            frames_buffer.clear()
            # check if the video is opened
            if not video_frames.isOpened():
                return 'Error in opening video', 500
            else:
                frames_buffer.clear()
                return 'success'

        # handle inference request for a live stream via IP camera
        elif 'live_inference_button' in request.form:
            # read ip cam url from the text box
            vid_ip_path = request.form['live_inference_textbox']
            # check if vid_ip_path is a valid url
            if validators.url(vid_ip_path):
                vid_path = vid_ip_path.strip()
                video_frames = cv2.VideoCapture(vid_path)
                # check connection to the ip cam stream
                if not video_frames.isOpened():
                    # display a flash alert message on the web page
                    return 'Error: Cannot connect to live stream', 500
                else:
                    frames_buffer.clear()
                    return 'success'
            else:
                # the url is not valid
                return 'Error: Entered URL is invalid', 500


if __name__ == "__main__":
    app.run(debug=True)
```

Save the file by pressing `Ctrl+X` → `Y` → `Enter`

Step 5 - Run Flask Application

Start the flask application by entering the following command in your terminal.

```
python -m flask --app app.py run
```

Your flask application should look something like this.