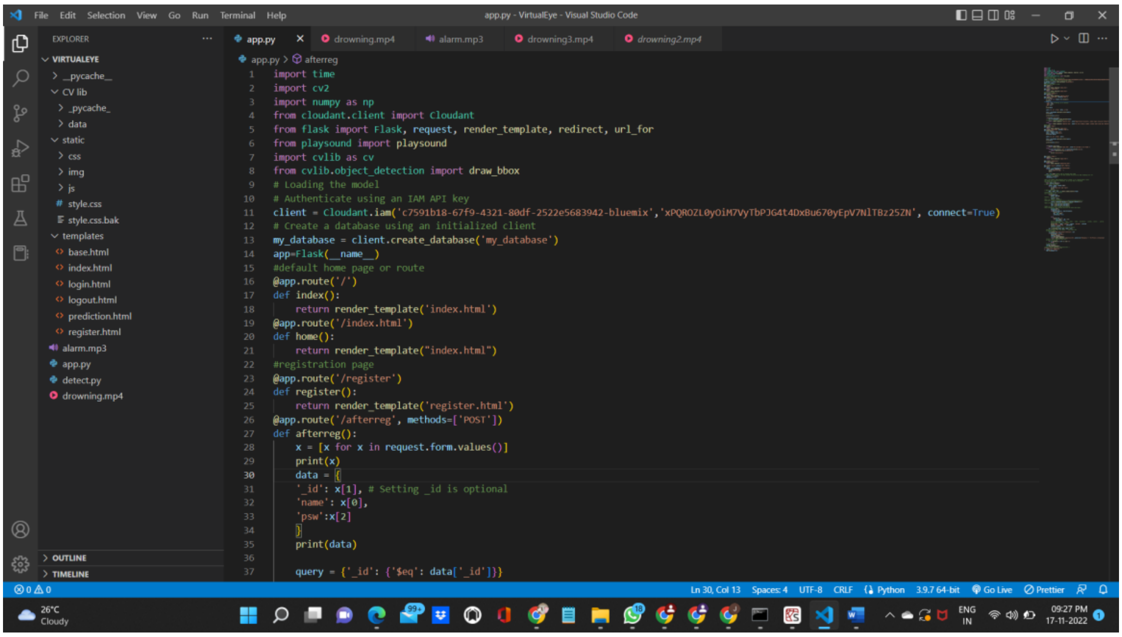
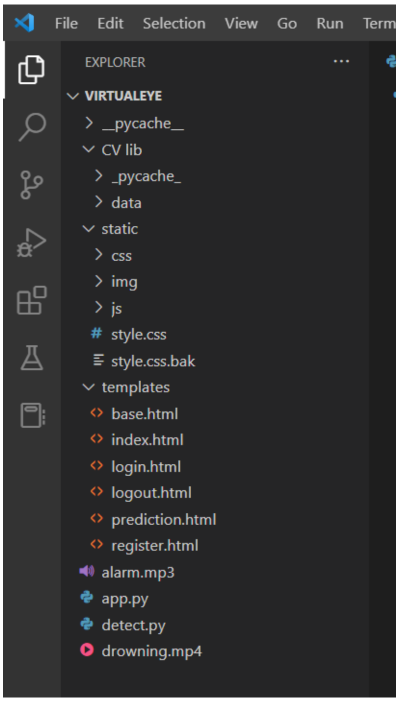
Project Structure

Date	17 Nov 2022
Team ID	PNT2022TMID34450
Project Name	Virtual Eye - Life Guard For Swimming Pools To Detect Active Drowning


app.py - VirtualEye - Visual Studio Code
File Edit Selection View Go Run Terminal Help
app.py
drowning.mp4
alarm.mp3
drowning3.mp4
drowning2.mp4
app.py > afterreg
Ln 30, Col 13 Spaces: 4 UTF-8 CRLF Python 3.9.7 64-bit Go Live Prettier

```python
import time
import cv2
import numpy as np
from cloudant.client import Cloudant
from flask import Flask, request, render_template, redirect, url_for
from playsound import playsound
import cvlib as cv
from cvlib.object_detection import draw_bbox
# Loading the model
# Authenticate using an IAM API key
client = Cloudant.iam('c7591b18-67f9-4321-80df-2522e5683942-bluemix','xPQROZL0yOiM7VyTbPJG4t4DxBu670yEpV7NlTBz25ZN', connect=True)
# Create a database using an initialized client
my_database = client.create_database('my_database')
app=Flask(__name__)
#default home page or route
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/index.html')
def home():
    return render_template("index.html")
#registration page
@app.route('/register')
def register():
    return render_template('register.html')
@app.route('/afterreg', methods=['POST'])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    data = {
    '_id': x[1], # Setting _id is optional
    'name': x[0],
    'psw':x[2]
    }
    print(data)

    query = {'_id': {'$eq': data['_id']}}
```



```
VIRTUALEYE
  > __pycache__
  v CV lib
    > _pycache_
    > data
  v static
    > css
    > img
    > js
    # style.css
    style.css.bak
  v templates
    base.html
    index.html
    login.html
    logout.html
    prediction.html
    register.html
  alarm.mp3
  app.py
  detect.py
  drowning.mp4
```

The object_detection.py python file would detect the different objects such as persons, bicycles, cars, chairs, etc.

- The data folder contains the caffemodel (Caffe (Convolutional Architecture for Fast Feature Embedding) is a deep learning framework that allows users to create image classification and image segmentation, models. Initially, users create and save their models as plain text PROTOTXT files. After a user trains and refines their model using Caffe, the program saves the user's trained model as a CAFFEMODEL file.)

- We are building a Flask Application that needs HTML pages stored in the templates folder and a python script app.py for server-side scripting

- The static folder has the CSS files which are necessary for styling the HTML page and for executing the actions.

- app.py contains the flask code which is used to detect the drowning person in a video input.

- Alarm sounds and demo videos are presented in project folders