LAMPIRAN

Lampiran 1

Hasil wawancara:

Q: Apa itu GDK dan peran satgas budaya dalam GDK?
A: GDK (GERAKAN DISIPLIN KAMPUS) adalah sebuah gerakan dibawah naungan Satgas Budaya kerja darmajaya yang didalamnya satgas memberikan materi materi tentang budaya THE BEST (Taqwa,heart,Empathy Briliant,Energetik,sinergy,trashworthy) kepada civitas akademik. Sementara GDK adalah penerapan nilai nilai materi budaya kebaikan tersebut.

Q: Apakah terdapat SK mengenai gerakan tersebut?
A: Ada

Q: Berlaku kepada siapa saja GDK tersebut?
A: Seluruh civitas akademik(dosen,karyawan,mahasiswa/i).

Q: Dimana saja tempat yang dilarang dan sering melakukan pelanggaran?
A: Seluruh kampus iib darmajaya kecuali kantin dan ruang yayasan.

Q: Jenis-jenis pelanggaran? pelanggaran terbanyak yang dilakukan?
A: Merokok sembarangan,pakaian tidak sopan,merusak fasilitas kampus.

Q: Mengapa tidak diperbolehkan?
A: Karena tidak sesuai dengan peraturan kampus yang melanggar nilai THE BEST

Q: Bagaimana dalam mengatasi pelanggaran yang dilakukan oleh masyarakat kampus?
A: Dilakukan pencegahan dengan teguran dan penilangan

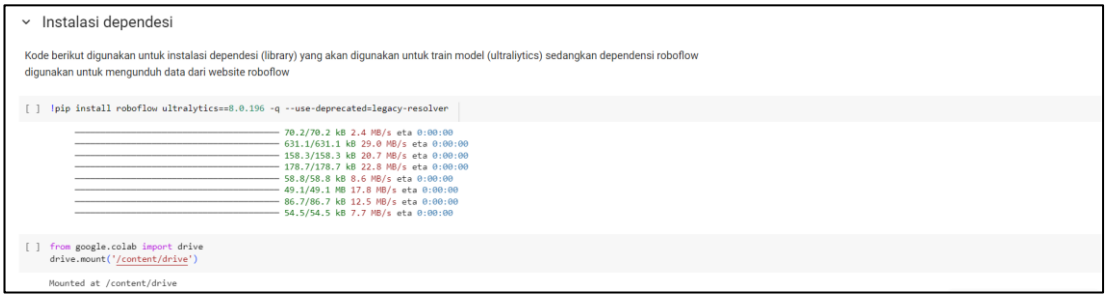
Q: Bagaimana alur penilangan?
A: Alur penilangan
- Terbukti melanggar
- Menghadap keruang satgas untuk menerima surat tilang
- Menahan ktp/sim/kartu mahasiswa/karyawan untuk jaminan pembayaran

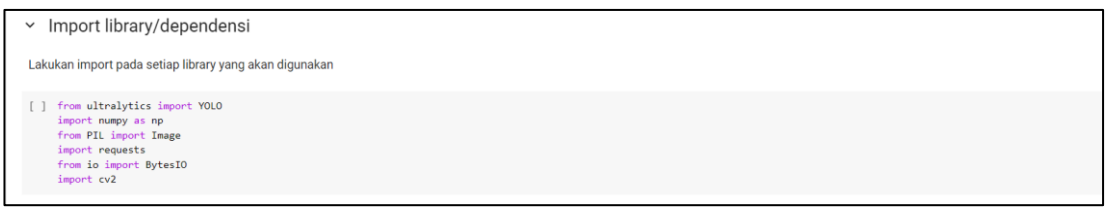
- Pelanggar melakukan pembayaran kebiro keuangan atau ke bank yang ditujukan kerekening Yayasan alfian husin.
- Konfirmasi kebagian keuangan jika sudah dilakukan pembayaran dengan bukti cap keuangan.
- Kembali keruangan satgas untuk mengambil ktp, sim dan npm setelah dilakukan pembinaan agar tidak mengulang Kembali.

Mahasiswi	Satgas Budaya
Yunita Tri Wulandari	Dr. H. Suratno, SPd.I,. M.H

Lampiran 2

Source code pembutan model





Download dataset dari roboflow

Untuk melatih model, dibutuhkan dataset. Dataset yang telah disiapkan dapat didownload langsung melalui kode berikut.

> Sangat disarankan untuk mengumpulkan dataset(membuat dataset) menggunakan roboflow karena lebih mudah untuk diintegrasi ke YOLO

```
[ ] from roboflow import Roboflow
    rf = Roboflow(api_key="cjiSztLSD2t0eNdGaBTA") # merupakan api_key untuk tiap akun
    project = rf.workspace("yt-wulandari").project("gdk_pelanggaran") # mendefinisikan workspace dan nama project pada roboflow
    dataset = project.version(6).download("yolov8",location='/content/datasets') # mendownload dataset(project) sesuai dengan versi(misalnya 4) lalu diunduh ke folder bernama /content/dataset(folder ini hanya
```

```
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in /content/datasets to yolov8:: 100%|████████| 306777/306777 [00:18<00:00, 16526.92it/s]

Extracting Dataset Version Zip to /content/datasets in yolov8:: 100%|████████| 14126/14126 [00:05<00:00, 2648.90it/s]
```

Memuat pretrain model

```
[ ] model = YOLO("yolov8m.pt")   # memuat pretrained model untuk training
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8m.pt to 'yolov8m.pt'...
100%|████████| 49.7M/49.7M [00:00<00:00, 343MB/s]
```

Training

melatih model dengan dataset yang telah di download. Data dari dataset yang diperlukan berada pada file `data.yaml` Oleh karena itu sangat disarankan untuk menggunakan roboflow karena file ini sudah dibuatkan secara otomatis untuk di train pada model YOLO. Variable `EPOCH` di set menjadi 20 yang artinya model akan dilatih sebanyak 20 kali perulangan. Meningkatkan jumlah epoch memungkinkan model untuk menjadi lebih baik (karena semakin banyak data yang dilihat) tetapi proses training akan memakan waktu yang lebih lama. 1 epoch berjalan akan memakan waktu 6-10 menit pada free google colab.

```
EPOCHS = 20
results = model.train(data="/content/datasets/data.yaml", epochs=EPOCHS, pretrained=True, iou=0.5, visualize=True, patience=0)  # train model
```

```
                   all       1320       1850      0.631      0.591      0.598      0.322

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      13/20      7.02G       1.45      1.616      1.629         12        640: 100%|████████| 312/312 [02:38<00:00,  1.97it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|████████| 42/42 [00:21<00:00,  1.94it/s]
                   all       1320       1850      0.646      0.597      0.631       0.35

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      14/20      7.02G      1.413      1.527      1.581          9        640: 100%|████████| 312/312 [02:38<00:00,  1.97it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|████████| 42/42 [00:21<00:00,  1.94it/s]
                   all       1320       1850      0.661      0.618      0.641      0.359

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      15/20        7G      1.376       1.43      1.548          9        640: 100%|████████| 312/312 [02:38<00:00,  1.97it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|████████| 42/42 [00:21<00:00,  2.00it/s]
                   all       1320       1850      0.666       0.63      0.648      0.369
```

Evaluasi hasil train model

setelah model di train, kita dapat melakukan evalauasi dengan kode berikut. hasil evaluasi berupa persentase akurasi, dll.

```
results = model.val()  # evaluasi model dengan data pada validation set
```

```
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 218 layers, 25842076 parameters, 0 gradients, 78.7 GFLOPs
val: Scanning /content/datasets/valid/labels.cache... 1320 images, 13 backgrounds, 0 corrupt: 100%|████████| 1320/1320 [00:00<?, ?it/s]
val: WARNING ⚠️ /content/datasets/valid/images/knife_64_jpg.rf.50b09f4cf491bfe8b041b4c42bcac7f3.jpg: 2 duplicate labels removed
WARNING ⚠️ Box and segment counts should be equal, but got len(segments) = 294, len(boxes) = 1850. To resolve this only boxes will be used and all segments will be removed. To avoid t
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|████████| 83/83 [00:39<00:00,  2.12it/s]
                   all       1320       1850      0.754      0.684      0.731      0.443
                  Asap       1320        379      0.697      0.573      0.623      0.323
                 Rokok       1320        537      0.744      0.689      0.716      0.415
                Sandal       1320        602      0.796      0.637      0.734      0.411
          senjata_tajam       1320        332       0.78      0.837       0.85      0.624
Speed: 0.3ms preprocess, 21.2ms inference, 0.0ms loss, 1.9ms postprocess per image
Results saved to runs/detect/val
```

Source code sistem prangkat lunak:

yolo > admin.py

```python
from django.contrib import admin

# Register your models here.

```

yolo > apps.py > ...

```python
from django.apps import AppConfig


class YoloConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "yolo"

```

yolo > models.py > ...

```python
from django.db import models

# History model
class History(models.Model):
    object_name = models.CharField(max_length=100)
    image = models.FileField(upload_to="media/")
    created_at = models.DateTimeField(auto_now_add=True)


# table name
class Meta:
    db_table = "history"
```

yolo > views.py > {} np

```python
from django.shortcuts import render
from django.contrib.auth import authenticate, login, logout
from django.shortcuts import redirect
from django.contrib.auth.decorators import login_required
from django.http.response import StreamingHttpResponse
from django.core.paginator import (
    Paginator,
    EmptyPage,
    PageNotAnInteger,
)

import cv2
import numpy as np
import time


from .models import History

CONF_THRESHOLD = 0.5 # Ganti dengan nilai yang lebih tinggi jika ingin lebih spesifik


# Create your views here.
# test view
def login_page(request):
    if request.user.is_authenticated:
        return redirect("index", permanent=True)
    if request.method == "POST":
        print(request.POST)
        username = request.POST.get("username")
        password = request.POST.get("password")

        user = authenticate(request, username=username, password=password)

        print(user)

        if user is not None:
            login(request, user)
            return redirect("index", permanent=True)
        else:
            return render(request, "yolo/login.html", {"message": "Invalid credentials."})

    return render(request, "yolo/login.html")

def logout_page(request):
    logout(request)
    return redirect("login_page", permanent=True)


@login_required
def index(request):
    return render(request, "yolo/dashboard.html", {"title": "Home"})

@login_required
def history(request):

    data = History.objects.order_by("-created_at").all()

    default_page = 1

    page = request.GET.get('page', default_page)

    paginator = Paginator(data, 5)
    try:
        data = paginator.get_page(page)
    except PageNotAnInteger:
        data = paginator.get_page(default_page)
    except EmptyPage:
        data = paginator.get_page(paginator.num_pages)
    return render(request, "yolo/history.html", {"title": "History", "data": data})


# DISPLAY CAMERA 1 ------------------
from .camera import *
from django.views.decorators import gzip

@gzip.gzip_page
def livefe(request):
    try:
        print(request)
        cam = VideoCamera()
        from ultralytics import YOLO
        model = YOLO('yolo/ml-model/best.pt')
        def gen(camera):
            while True:
                frame = camera.get_frame()
                frame = cv2.imdecode(np.frombuffer(frame, dtype=np.uint8), -1)
                results = model.predict(frame, project="media", name="predict")
                for r in results:
                    if r.boxes:
```

```
                    x = int(box.data[0][0].item())
                    y = int(box.data[0][1].item())

                    cv2.rectangle(frame, (x, y), (x + 100, y + 100), (0, 255, 0), 2)
                    cv2.putText(frame, object_name, (x, y), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)

                    # put conf level under the rectangle
                    conf  = round(box.conf[0].item(), 2) # only take the first element and round it to 2 decimal places
                    cv2.putText(frame, str(conf), (x, y + 20), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)
                    # save image to media if conf > 0.5
                    if conf >= CONF_THRESHOLD:
                        image_name = time.strftime("%Y%m%d-%H%M%S") + ".jpg"
                        cv2.imwrite("media/" + image_name, frame)
                        history = History()
                        history.object_name = object_name
                        history.image = image_name
                        history.save()

                _, jpeg = cv2.imencode('.jpg', frame)
                frame = jpeg.tobytes()
                yield (b'--frame\r\n'
                    b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

        return StreamingHttpResponse(gen(cam), content_type="multipart/x-mixed-replace;boundary=frame")
    except:  # This is bad!
        print("An error occurred!")
        pass


@login_required
def about(request):
    return render(request, "yolo/about.html", {"title": "About"})
@login_required
def information(request):
    return render(request, "yolo/info.html", {"title": "Information"})
@login_required
def change_password(request):

    if request.method == "POST": # jika method POST artinya user mengirimkan data
    # if empty
        print(request.POST)
        if request.POST.get("old_password") is None:
            return render(request, "yolo/change_password.html", {"message": "Password Lama tidak boleh kosong."})
        if request.POST.get("new_password") is None:
            return render(request, "yolo/change_password.html", {"message": "Password Baru tidak boleh kosong."})
        if request.POST.get("password_confirm") is None:
            return render(request, "yolo/change_password.html", {"message": "Konfirmasi Password tidak boleh kosong."})
        # get current user password
        if not request.user.check_password(request.POST.get("old_password")):
            print("Password lama salah.")
            return render(request, "yolo/change_password.html", {"message": "Password lama salah."})
        print(request.POST)
        password = request.POST.get("new_password")
        password_confirm = request.POST.get("password_confirm")
        if password == password_confirm:
            request.user.set_password(password)
            request.user.save()
            return redirect("index", permanent=True)
        else:
            print("Password tidak sama.")
            return render(request, "yolo/change_password.html", {"message": "Password tidak sama."})

    return render(request, "yolo/change_password.html")

```

yolo > urls.py > ...

```python
from django.urls import path
from django.contrib.staticfiles.urls import staticfiles_urlpatterns
from . import views

urlpatterns = [
    path("", views.index, name="index"),
    path('video_feed_1/', views.livefe, name="video-feed-1"),
    path("history/", views.history, name="history"),
    path("tentang/", views.about, name="tentang"),
    path("informasi/", views.information, name="informasi"),
    path("login/", views.login_page, name="login_page"),
    path("ganti-password/", views.change_password, name="ganti_password"),
    path("logout/", views.logout_page, name="logout_page"),
]
```