Mid-Development Review
AI Smart Behavior and Weapon Detection

Final Year Project Proposal

BSCS

By

S#	Name	Registration #/Roll #/Section	Mobile #	E-Mail
1.	Saad Amir	FA21-BSCS-051/B	0328-5333392	saad.amir@live.com
2.	Nawab Iftikhar	FA21-BSCS-076/B	0333 1474584	nawabpanu@gmail.com

Supervised by:

Ma'am Anila Amjad

(Signature**)** _____

Department of Computer Science Lahore
Garrison University Lahore

1. Project Overview

AI Smart Behavior and Weapon detection is a desktop application designed to ease in surveillance and assist security personnel by analyzing and detecting weapons in the camera feed and also analyzing behavior. The model was trained on YOLO with a custom dataset. The model will then be integrated into desktop application with a UI either web based or desktop installable application, the key events such as a weapon detected will be stored in database i.e. postgresql/sqlite. The languages used were python, the model will then be integrated using API and deployed on edge devices or cloud.

1.2 Objective of the Report

This document shows the mid development review of AI Smart Behavior Analysis. It covers the progress made, revised implementation techniques, testing results, encountered challenges, and anticipated improvements. The report evaluates the project's status against its development milestones and justifies further investment in its completion.

1.3 Technologies Used

- **IDE:** Google Colab, Jupyter Notebook, Spyder
- **Frontend:** Python, tkinter
- **Backend:** Python, YOLO,
- **Database:** Postgresql/sqlite
- **APIs:** FastAPI, RestfulAPI
- **UI/UX:** Designed using tkinter

2 Development Progress

1. Data Scraping

1.1 **Data Collection and Scraping**

- Wrote scripts on spyder to scrape data from google images such as images of revolver, pistols, rifles etc…
- Stored the downloaded dataset then labeled the dataset using image labeler tool i.e. roboflow.

1.2 **Feature Engineering**

● Using Data Augmentation to generate new images for better model training. ● Using digital image processing for better image quality.

1.3 **Hyperparameter Tuning**

- Tuning the parameters of the model for training such as , epoch, batch size, learning rate etc…

2.1 **Model Training and Evaluation**

- Training the model on the dataset, and then evaluating model performance.

2.2 **UI/UX Implementation**

● Designed an intuitive and user friendly interface for better visuals. ● Integrated the UI with python using tkinter.

3.1 **Integration**

- Integration of model on the desktop application.
- Model integration on the web based application.

3.2 **Deployment**

● Model deployment on the web or desktop

---

3 Testing and Results

Model Evaluation

Test Case No.	Priority	Test Case	Expected Output	Actual Output	Status
1.	1st	Notification Alerts	Notify security when weapon is detected	Works as expected	Passed
2.	2nd	Data Logging	Accurately record logging	Works as expected	Passed
3.	3rd	Data Visualization	Displays detected weapon with timestamp	Works as expected	Passed

Weapon Detection

Test Case No.	Priority	Test Case	Expected Output	Actual Output	Status
1.	1st	Content Display	Display detected weapon	Works as expected	Passed
2.	2nd	Accessibility	Content is readable and accessible	Works as expected	Passed

---

Challenges Identified

- **Performance Optimization:**
  - App responsiveness on lower-end devices needs improvement.
  - Reducing load time for the dashboard is a priority.
- **Data Privacy and Security:**
  - Need for stricter data encryption measures.
  - Implementing role-based access control for sensitive user data.
- **UI/UX Refinements:**
  - Improving the layout for easier navigation.
  - Ensuring that the community forum is engaging yet safe.

Next Steps

- **Enhance performance:** Optimize the model performance to run on edge.
- **Improve security measures:** Implement advanced machine learning methods for better security.
- **UI/UX refinements:** Further testing and iterations to enhance usability.
- **Expand testing:** Conduct user testing with a diverse range of weapons and scenario.

Code for Data Scraping

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
# from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
import os
import time
import random
import requests


opt = Options()
opt.add_argument("--disable-notifications")
opt.add_argument("--disable-infobars")
opt.add_argument("--start-maximized")
# hide automation works
opt.add_argument('--disable-blink-features=AutomationControlled')
# opt.add_argument('--headless')  # runs in the background


cservice = Service(ChromeDriverManager().install())
driver = webdriver.Chrome(service=cservice, options=opt)
wait = WebDriverWait(driver, 10)


# function to download an image
def download_image(img_url, folder, img_name):
    try:
        img_data = requests.get(img_url, timeout=10).content
        with open(os.path.join(folder, img_name), 'wb') as img_file:
            img_file.write(img_data)
        print(f"Downloaded: {img_name}")
```

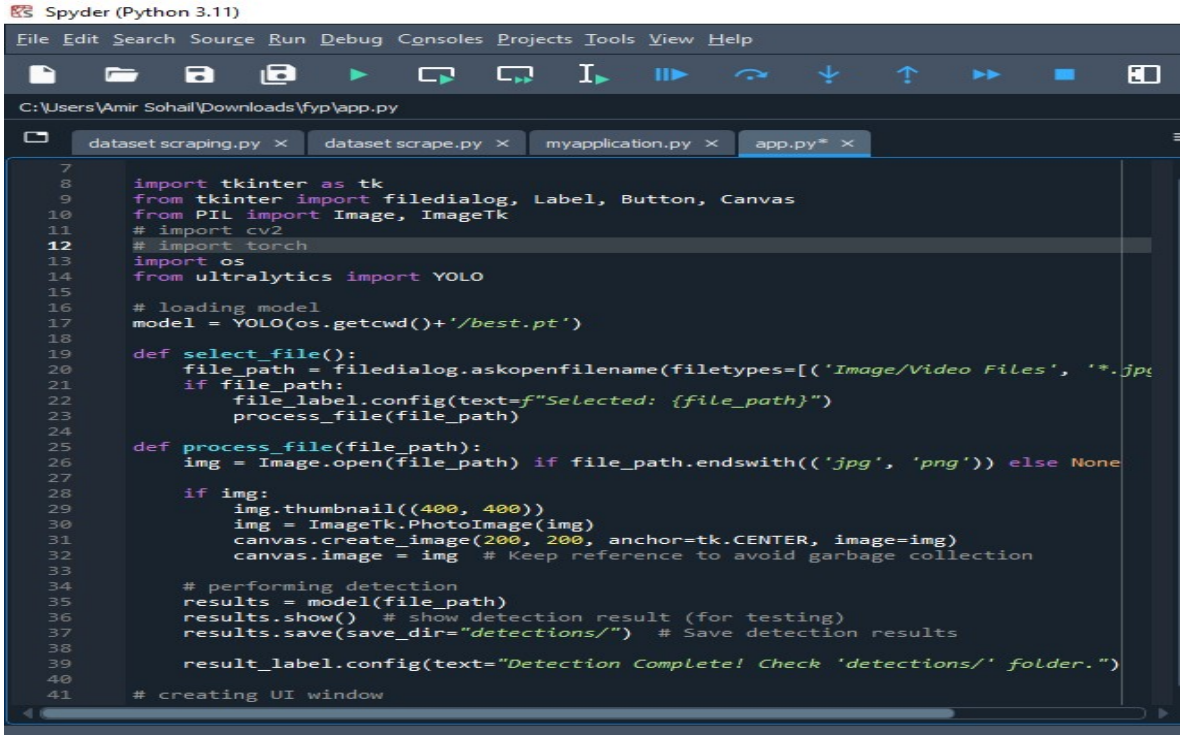
Code for Data Scraping

sers \Amir Sohail \Downloads \fyp \dataset scrape.py

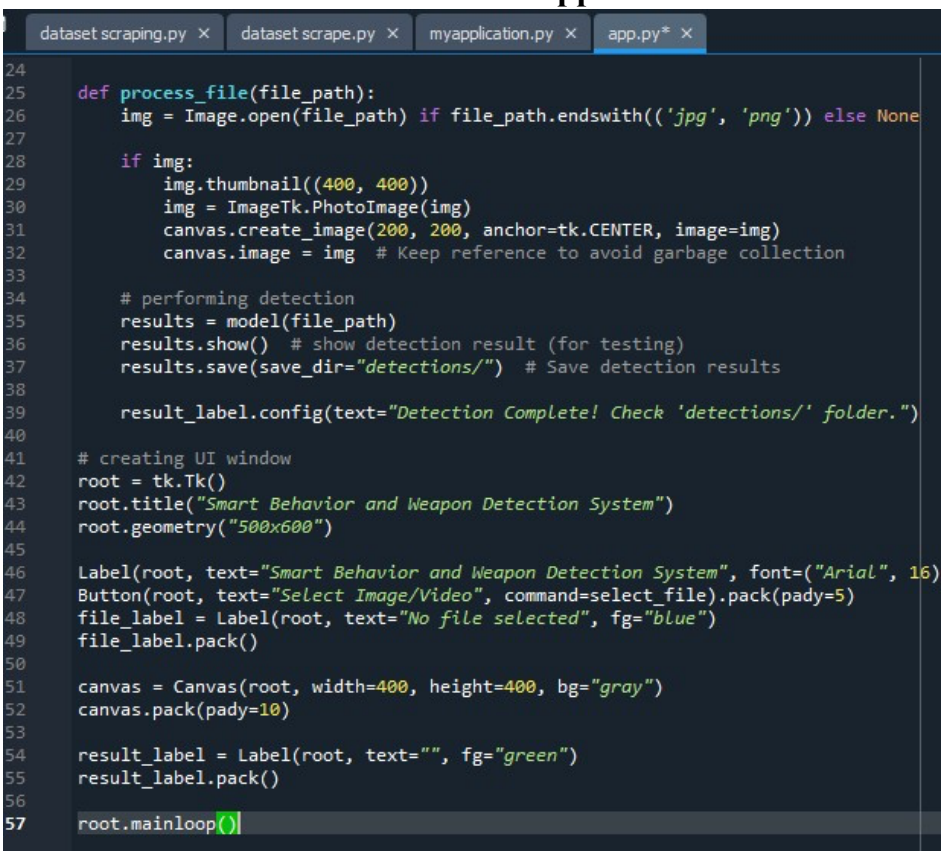
dataset scraping.py × | dataset scrape.py* ×

```python
def download_image(img_url, folder, img_name):
    try:
        img_data = requests.get(img_url, timeout=10).content
        with open(os.path.join(folder, img_name), 'wb') as img_file:
            img_file.write(img_data)
        print(f"Downloaded: {img_name}")
    except Exception as e:
        print(f"Error downloading {img_name}: {e}")

# search query
search_query = "revolver"
url = f"https://www.google.com/search?q={search_query}&tbm=isch"

# open Google Images
driver.get(url)

# scroll down to load more images
for _ in range(5):  # Scroll 5 times
    driver.find_element(By.TAG_NAME, "body").send_keys(Keys.END)
    time.sleep(random.uniform(2, 4))  # Random sleep to avoid detection

# extract image URLs
page = driver.page_source
soup = BeautifulSoup(page, 'html.parser')
all_images = soup.find_all("img")

# process image urls
image_links = []
for img in all_images:
    src = img.get("data-src") or img.get("src")  # Prefer data-src if available
    if src and "http" in src:
        image_links.append(src)

print(f"Found {len(image_links)} images.")

# create folder to save images
```

Code for Model/Application



Code for Model/Application



GUI

Conclusion

The development of AI Smart Behavior and Weapon Detection Security System has demonstrated significant progress in enhancing security measures through automated threat detections. By leveraging advanced image processing and machine learning techniques, the system can accurately identify potential weapons in real time, reducing human error and improving response times.

Although the project is still in its mid development phase, key functionalities such as object detection, and real time surveillance integration have been successfully implemented, and alert mechanism is underway. Further improvements, including optimizing detection accuracy, reducing false positives, and enhancing system efficiency, will be our focus in the final stages of development.

Once completed, this system has the potential to be deployed in high security areas such as airports, schools, and public places, providing fast approach to threat prevention.

With continued refinements, AI Smart Behavior and Weapon Detection Security System aims to contribute to a safer environment by ensuring quick and accurate threat detection.