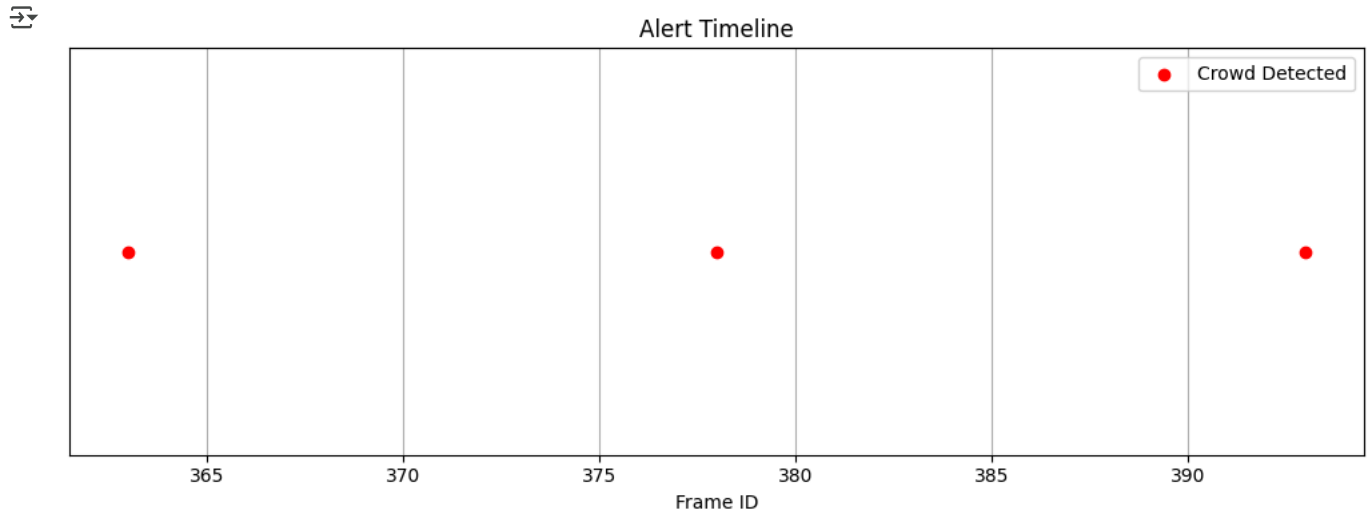
TASK 2

```
import cv2
from ultralytics import YOLO
import matplotlib.pyplot as plt
from collections import deque
import json
import os
from datetime import datetime
```

```
video_path='/content/people-detection.mp4'

model = YOLO("yolov5s.pt")

frame_skip = 3
alert_threshold = 3
alert_window = 5
alert_log_path = "alert_log.json"
timeline_plot_path = "alert_timeline.png"

cap = cv2.VideoCapture(video_path)
frame_id = 0
recent_people_counts = deque(maxlen=alert_window)
alert_times = []
alerts = []

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    if frame_id % frame_skip == 0:
        results = model(frame)[0]
        person_count = 0

        for box in results.boxes:
            cls_id = int(box.cls[0])
            label = model.names[cls_id]
            if label == "person":
                person_count += 1

        recent_people_counts.append(person_count)

        if len(recent_people_counts) == alert_window and all(p >= alert_threshold for p in recent_people_counts):
            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            alert_msg = {
                "frame": frame_id,
                "timestamp": timestamp,
                "message": "Crowd Detected"
            }
            alerts.append(alert_msg)
            alert_times.append(frame_id)
            print(f"[ALERT] Crowd detected at frame {frame_id} ({timestamp})")
            recent_people_counts.clear()

    frame_id += 1

cap.release()

with open(alert_log_path, 'w') as f:
    json.dump(alerts, f, indent=2)

print(f"\nTotal Alerts Triggered: {len(alerts)}")
if alerts:
    print("Sample Alert:", alerts[0])
```

```
0: 384x640 1 person, 296.2ms
Speed: 3.8ms preprocess, 296.2ms inference, 1.2ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 person, 302.8ms
Speed: 4.1ms preprocess, 302.8ms inference, 1.2ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 person, 390.2ms
Speed: 3.8ms preprocess, 390.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 person, 478.6ms
Speed: 4.5ms preprocess, 478.6ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 persons, 460.5ms
Speed: 3.6ms preprocess, 460.5ms inference, 1.4ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 person, 489.4ms
Speed: 3.5ms preprocess, 489.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 person, 482.2ms
Speed: 3.8ms preprocess, 482.2ms inference, 2.2ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 430.3ms
Speed: 4.9ms preprocess, 430.3ms inference, 0.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 303.8ms
Speed: 3.4ms preprocess, 303.8ms inference, 0.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 296.1ms
Speed: 3.9ms preprocess, 296.1ms inference, 0.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 292.3ms
Speed: 4.3ms preprocess, 292.3ms inference, 0.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 306.6ms
Speed: 3.8ms preprocess, 306.6ms inference, 0.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 292.2ms
Speed: 4.1ms preprocess, 292.2ms inference, 0.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 292.0ms
Speed: 3.0ms preprocess, 292.0ms inference, 0.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 313.2ms
Speed: 3.7ms preprocess, 313.2ms inference, 1.1ms postprocess per image at shape (1, 3, 384, 640)
```

```
plt.figure(figsize=(10, 4))
plt.scatter(alert_times, [1]*len(alert_times), color='red', label='Crowd Detected')
plt.title("Alert Timeline")
plt.xlabel("Frame ID")
plt.yticks([])
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig(timeline_plot_path)
plt.show()
```



How I did it – Step-by-step:

1. Loaded the video using OpenCV from the given local path.

2. Loaded the YOLOv5 model to detect objects, especially focusing on people.
3. Processed every 3rd frame from the video to simulate a real-time stream.
4. For each processed frame:
- Ran object detection using YOLOv5.
- Counted how many people (label == "person") were detected.
- Saved this count in a rolling window of the last 5 frames.
5. If 3 or more people appeared in 5 consecutive frames, triggered an alert:
- Captured the frame number and timestamp.
- Logged the alert in a list.
6. After processing the video:
- Saved all alert data in a .json file.
- Counted the total number of alerts.
7. Created a timeline scatter plot:
- Each red dot shows a frame where a crowd was detected.
8. Saved all results:
- Alert log in .json
- Timeline chart in .png