PUBLIC SURVEILLANCE SOFTWARE

1. Person and Gender Detection

- **AWS Rekognition:** Use the service to analyze video frames for person and gender detection.
- **AWS SDK (Boto3):** Call Rekognition APIs using Boto3 in your Python code to process the images.

2. Lone Women Detection

- **DBSCAN Algorithm**: Apply DBSCAN(Density based spatial clustering application with noise) to detect clusters of people. Identify lone women by finding those not within any significant cluster.
- **Create JSON Response:** Format the detected lone women data (location, time) into JSON.

-- " JSON.pdf "

3. Count Men/Women

- **Python (NumPy):** Use NumPy to process detection results and count the number of men and women in the frame.

4. Surrounding Men Detection

- **Geometric Logic (Bounding Box):** Define bounding boxes around detected people. Check if the bounding boxes around any lone women are surrounded by men.

5. Behavior Analysis

The above JSON response will be used in this analysis (STEP2)

- **AWS SageMaker:**
  - Train custom ML models for behavior analysis. { CNN(conventional neural networks) ,RNN(recurrent neural networks), SVM(support various machines) }.
  - Deploy the models to make real-time predictions on behavioral data.

➢ **Trigger Alerts**

- **AWS Lambda:**
  - Set up a Lambda function to be triggered when SageMaker detects abnormal behavior.
  - Lambda processes the prediction result and generates an alert if necessary (e.g., unusual activity detected).

Step 1: Threat Detection

- **AWS SageMaker**: Detect threat (e.g., lone woman, suspicious behavior) using custom ML models.
- **AWS Lambda**: Automatically trigger Lambda functions based on detection.

Step 2: Create Alert Message

- **Lambda Functions**: Format detection data into an alert message (location, threat type, timestamp).
- **JSON Response**: Prepare structured data for communication.

Step 3: Send Alert

- **Firebase Cloud Messaging (FCM)**: Send alerts to Android devices.
- **Apple Push Notifications (APN)**: Send alerts to iOS devices.
- **AWS SNS**: Optionally send SMS or email alerts.

Step 4: Geofencing and Proximity

- **Spatial Indexing (R-trees)**: Determine nearest patrol teams using location data.
- **Proximity Calculation**: Ensure alerts are sent to closest units.

Step 5: Patrol Team Response

- **Mobile App (Flutter)**: Patrol teams receive alerts on connected devices with location and threat details.
- **Real-Time Location Sharing**: Patrol teams share their location with the central system.

Step 6: Monitoring and Reporting

- **AWS CloudWatch**: Monitor alert delivery and system performance.
- **Central Dashboard**: Real-time tracking of alerts and patrol team responses.

Final Database Solution

1. **AWS S3 (Amazon Simple Storage Service)**
    - **Purpose**: Store large volumes of raw and semi-structured data.
    - **How to Store**:
        - **Raw Video and Images**: Upload surveillance footage and snapshots.
        - **Detection Results**: Store JSON files containing results from person and gender detection.
        - **GPS and Triangulation Data**: Store raw GPS positioning data and cell tower triangulation results.
    - **Implementation**: Use S3 buckets to organize data by type and date. Set up lifecycle policies for data archiving and deletion.
2. **AWS RDS (Relational Database Service)**
    - **Purpose**: Manage structured data and support complex queries.
    - **How to Store**:

        - **Structured Metadata**: Store information such as user details, historical counts of men/women, and aggregated results.
        - **Aggregated Results**: Store results from Kalman filters and historical hotspot data.
    - **Implementation**: Create relational tables in RDS for different types of structured data. Use SQL queries to manage and retrieve data efficiently.
3. **AWS DynamoDB (NoSQL Database)**
    - **Purpose**: Handle real-time data with high-speed access.
    - **How to Store**:
        - **Real-Time Geofencing Data**: Store dynamic data for geofencing and geospatial analysis.
        - **Alerts and Notifications**: Store real-time alerts, notifications, and behavioral data.
    - **Implementation**: Use DynamoDB tables for fast access to real-time data. Configure indexes for efficient querying and updating of geospatial and alert data.

How to Implement the Storage

1. **Set Up AWS S3 Buckets**
    - **Create Buckets**: Set up separate buckets for video footage, images, detection results, and geospatial data.
    - **Organize Data**: Use folders and prefixes to structure data by type and date.
    - **Configure Permissions**: Set up access controls to secure your data.
2. **Configure AWS RDS**
    - **Create a Database Instance**: Choose the appropriate relational database engine (e.g., PostgreSQL, MySQL).
    - **Design Schema**: Define tables for metadata, counts, and aggregated results.
    - **Set Up Security**: Configure VPC, security groups, and IAM roles for secure access.
3. **Set Up AWS DynamoDB**
    - **Create Tables**: Define tables for real-time geofencing data and alerts.
    - **Configure Indexes**: Set up primary and secondary indexes to optimize querying.
    - **Manage Throughput**: Adjust read/write capacity units based on expected load.
4. **Integrate and Manage**
    - **Data Ingestion**: Use AWS Lambda functions or data pipelines to ingest data into S3, RDS, and DynamoDB.

- **Monitor**: Use AWS CloudWatch to monitor database performance and storage usage.
- **Backup**: Implement backup strategies for RDS (automated backups) and S3 (versioning).

Summary

- **AWS S3**: Stores large and raw data (videos, images, JSON files).
- **AWS RDS**: Manages structured, relational data (metadata, aggregated results).
- **AWS DynamoDB**: Handles real-time, high-speed data (geofencing, alerts).

Combining these three services provides a robust and scalable solution for managing diverse data types in your public surveillance software.