# Real Time Contact Tracing: An OpenCV approach

## Introduction

COVID-19 has forced lockdowns and strained the public healthcare system with enormous outbreaks. It is a highly contagious virus, and infected people do not always show symptoms, with some remaining asymptomatic. As a result, a non-negligible proportion of the population can be a hidden source of transmission at any one time.

This project proposes a real-time distributed facial recognition system to address contact tracing concerns.

## Contact

Anyone who cared for the suspect or confirmed case, including a health care worker (including those involved in cleaning, waste management, laboratory technicians, doctors) or family member, or anyone who had close physical contact; anyone who stayed in the same place (lived with or visited) while the index patient was symptomatic.

## Contact Tracing

The process of finding, diagnosing, and managing people who have been exposed to a disease to prevent transmission is known as contact tracing. People who may have been exposed to the virus need to be tracked for 14 days from the date of the probable last exposure/arrival from afflicted areas, according to the Ministry of Health and Population (MoHP).

## METHODOLOGY

### Real-time Face recognition system

OpenCV – a collection of open-source computer vision algorithms – is used to collect frames from CCTV and send them into an object detection model, which provides information about where the faces are on the frame. This information is utilized to locate, and extract faces from the frame, which are then fed into a deep learning network, which provides 128 face encodings specific to an individual. This information can be used to identify that person in the future.

### Technologies in the client side

### Face recognition

Face recognition is a popular Python library that can be used to recognize faces. Built on top of the Dlib, an open-source C++ library that implements machine learning methods, this library detects faces with classification, regression, clustering, data transformation, and structured prediction.

### Technologies in the server side

### SQLite3

SQLite is a relational database management system (RDBMS). It uses real-time processing to handle workloads whose state is constantly changing. This differs from traditional databases containing persistent data, mostly unaffected by time. For example, a stock market changes very rapidly and is dynamic.

## System Architecture

### Overview

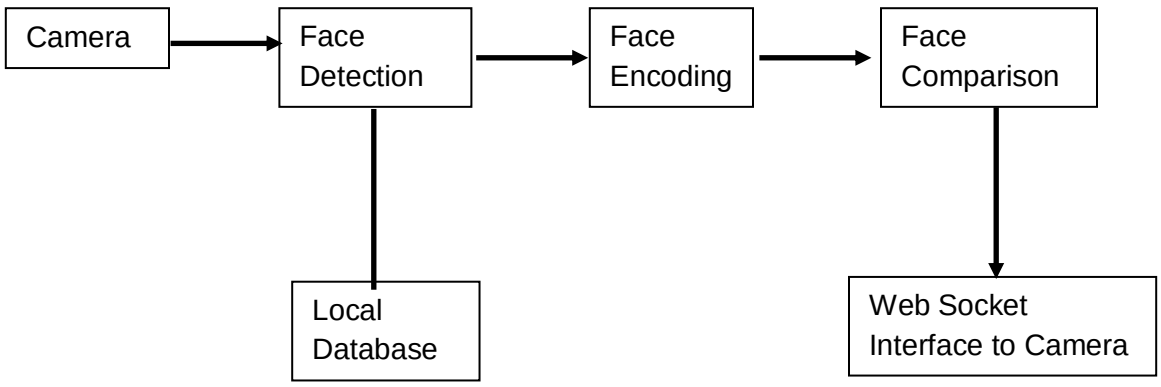
This project consists of two major parts:

- Client software
- Master server

Client software is installed on the main computer of the CCTV camera. It should perform two actions: Keep track of everyone appearing on the camera, search people whose information is uploaded on the server.

Client system consists of 6 components:

- Camera
- Local database
- Face detection
- Face encoding
- Compare faces
- Web socket interface



**Camera**

A camera is an optical device that captures images. Cameras are, at their most basic, sealed boxes with a small hole that allows light to pass through to capture a picture on a light-sensitive surface. Different systems in cameras govern how light falls onto the light-sensitive surface.

**Local Database**

Each client system has its own local database for facial recognition. This database contains information about everyone who appeared on the CCTV cameras of the client system. These records are utilized to gather information about people. Records are stored and retrieved using a simple JSON file.

**Face Detection**

Face detection is the first and most important step in face recognition; it detects faces in photos. It is a type of object detection that can be applied to a variety of applications, including security, biometrics, law enforcement, entertainment, and personal safety.

It detects faces in real time for surveillance and tracking of people or things. It's extensively employed in cameras to recognize several appearances in the frame of DSLRs and Ex-Mobile cameras. Face detection techniques are also used by Facebook to find and recognize faces in photos.

**Face Encoding**

For facial recognition, the algorithm considers crucial characteristics on the face such as the color, size, and slant of the eyes, the distance between the brows, and so on. The face encoding information retrieved from an image and used to identify a given face is defined by all of these.

**Face Comparison**

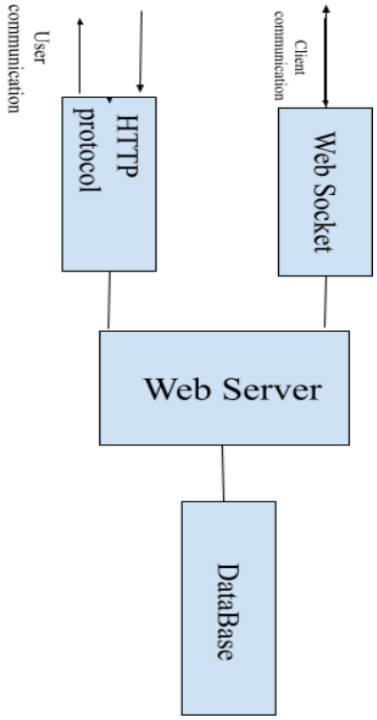
Face encodings from the local database are compared to the most recent frames' face encodings. If there is a match, the result is sent to the web socket interface block.

**Web Socket Interface**

The WebSocket API is a cutting-edge technology that allows a client and a server to establish a two-way interactive communication session. You can use this API to send messages to a server and receive event-driven responses without polling the server for a response. To communicate with the master server, a web socket interface is used. When comparing face block matches, the client will send a web socket event to the server.

**Master Server Architecture**

The master server manages all client systems and serves as an interface between the user and the client systems. Only healthcare workers and law enforcement officers have access to the master server. We can stop people from abusing the system by restricting access to it.




**Database Design**

On the server, data is stored in a relational database. A relational database is a database type. It employs a structure that enables us to identify and access data in connection to other data in the database. Data in a relational database is frequently grouped into tables. Four separate tables are used to keep and organize various records.

- Person Table
- Location Table
- Face Recognizer Index Table
- Online Systems Table

**Person Table**

The Person table is used to store information about persons for whom we want to search. This table contains a total of six columns.

- Id – primary key
- Name – Person's name
- File – Photo stored location on the server
- Disc – Short description of the person
- Date – Date and Time
- Live_track – To identify whether we want to track this individual in real-time

**Location Table**

The location table was used to contain all the client's locations and descriptions. This table has five columns.

- id – primary key
- uid – auth token
- camera number – on the client system
- place – address
- desc – description of the place

**Face Recognizer Index Table**

This table offers information about all the race results. This table has four columns for storing various records.

- Id – primary key
- Time – date and time
- Person_id – foreign key to person table
- Location_id – foreign key to location table

**Online Systems Table**

This table displays all the client systems that are available online. This table has four columns.

- Id – primary key
- Sid – socket.io id of the online system
- Time – date and time
- location_id – foreign key to location table

**Working**

**Client Software**

Client software is installed on the main computer of the CCTV camera. Client software should do two functions: keep track of everyone who appears on the CCTV camera, and search for people whose images have been uploaded to the server.

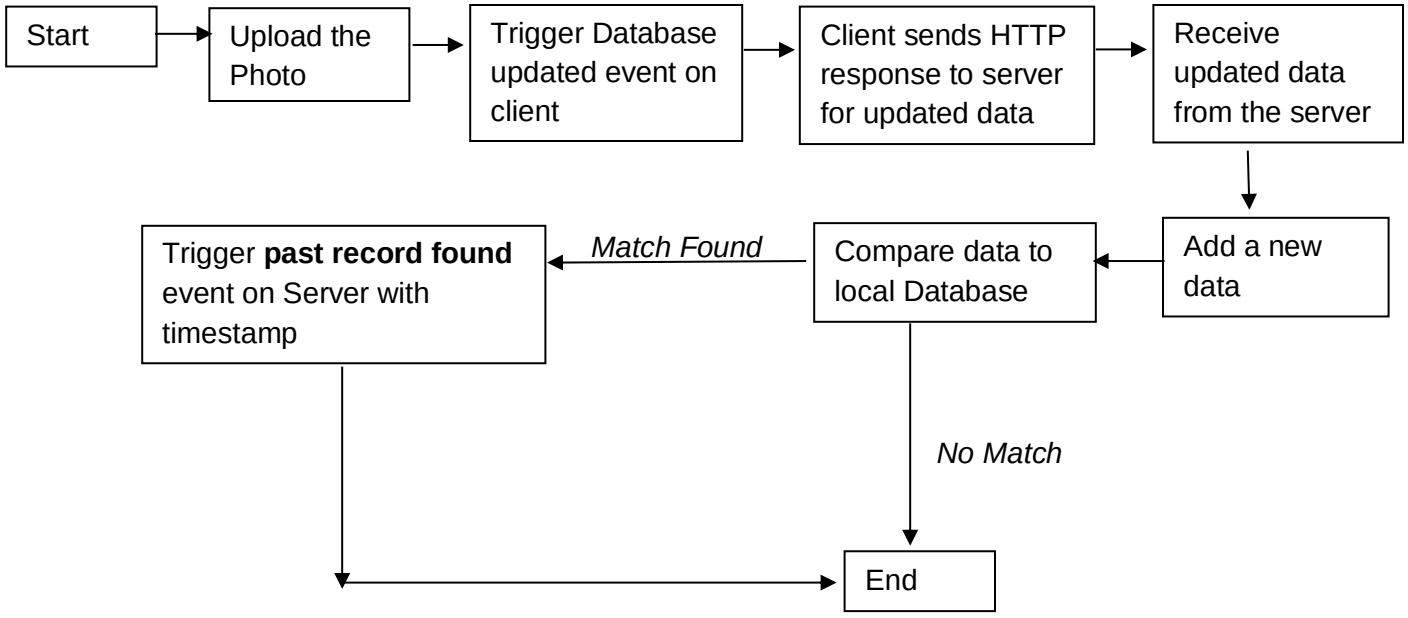
**Local Face Recognition Database**

Every client system has its own local facial recognition database. This database contains information about anyone who appeared on the CCTV cameras of the client system. These data are used to collect people's record. The algorithm below explains how it's done.

Step 1: Retrieve the frame from the CCTV camera.
Step 2: Use object detection methods to detect the faces.
Step 3: Determine the face encodings for all the faces on the frame.
Step 4: Check the face encodings against the local database.
Step 5: Add a timestamp to the person's record if they already exist.
Step 6: If not, make a new record for the person and add a timestamp to it.

**Search a person across all clients**

To search for a person across all client computers, we just upload a photo of the individual to the server. After uploading the photo, the client system will send a 'database updated' websocket event. The client will then issue an HTTP request, and the server will respond with the most recently updated database facts. The client's known faces list will be updated once the revised records are received. The client system then examines its local database for the person's previous records. If the individual has previously appeared on the client system, the client will send an 'update past record' websocket event to the server with the timestamp data and 'auth token'.


Start
Upload the Photo
Trigger Database updated event on client
Client sends HTTP response to server for updated data
Receive updated data from the server
Add a new data
Compare data to local Database
Match Found
Trigger **past record found** event on Server with timestamp
No Match
End