INTELLIGENT BANK LOCKER

# By Team Members

# M .HEMA REDDY

# D.SUNIL

# T.RAVITEJA

# D.VENKATASAIRAM

# Table Of Contents

* Introduction
  + Overview
* Literature Survey
  + Motivation and Methodology of Proposed System
* Theoretical Analysis
  + Block diagram
  + Hardware/software design
* Flowchart
* Results
* Advantages & Disadvantages
* Conclusions
* Appendixes
  + Source code
  + UI output

# Introduction

## Overview

## Banks are considered as a soft target Of criminals. In this circumstance,

## ensuring security of bank lockers should Be taken into consideration. Therefore

## In this project we propose an IOT based intelligent bank locker and we also introducing

## face detention approach

## Purpose :

## As security has become a prime need this concern motivated us to propose such system

# Literature Survey:

Motivation and Methodology of Proposed System :

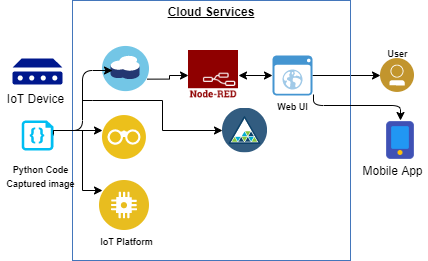
* This research is mainly based on IOT. Many researcher have developed, implement or proposed different system to ensure the security of bank locker as security has become a prime need this concern motivated us to proposed such system so in order to ensure security of individuals we can use this system. The whole system is designed to implement through internet server

## Proposed Solution

The designed system consists of a detector with the latest technology which detects the face and recognizes the user.

# Theoretical Analysis

## Block Diagram

****

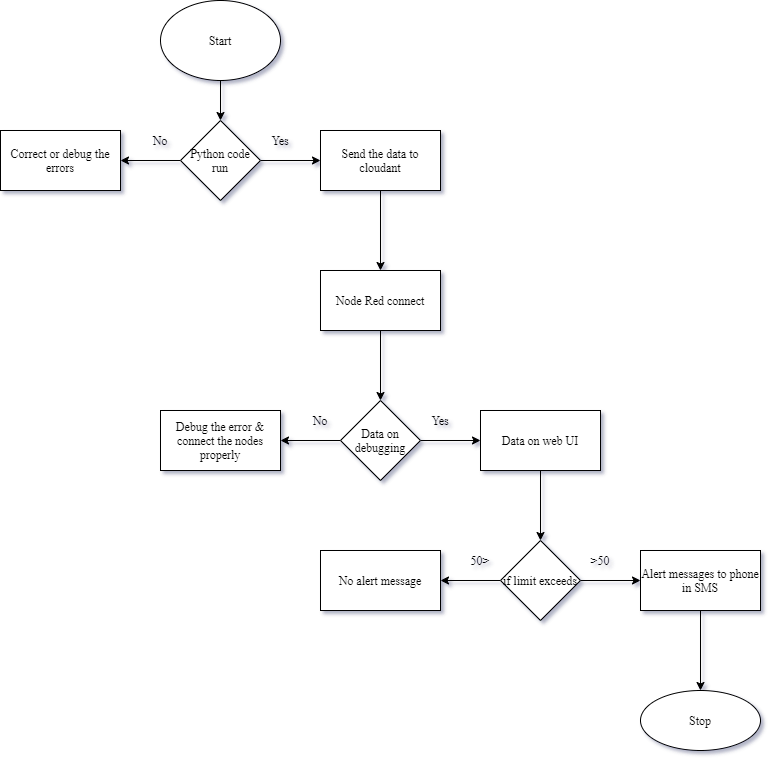
**Hardware/Software Design :**

We follow a step-by-step procedure to set up all the interfaces required for our project & developed the project in Python to detect the face of the user

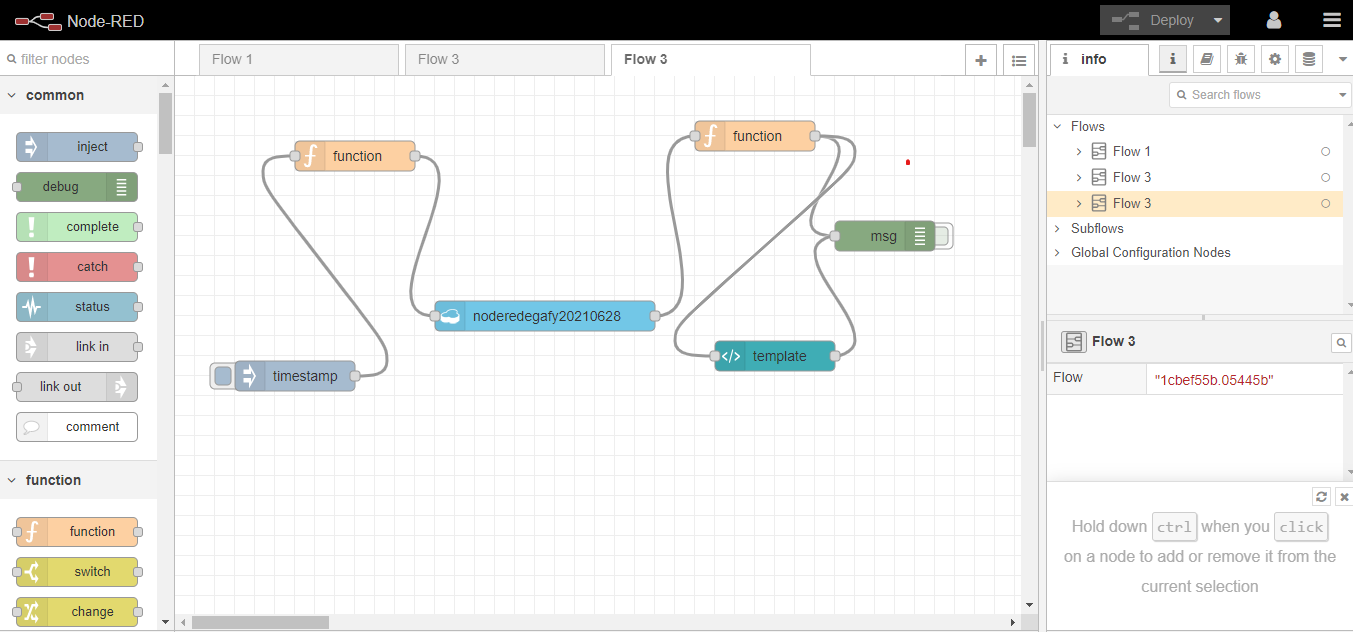
Software requirement

* + - Python Idle (with specified packages installed)
    - IBM Cloud
    - Node Red service

**FLOW CHART:**



# Results

****

**Advantages & Disadvantages**

## Advantages

* Low cost
* Low power consumption
* High accuracy
* The sensor has excellent sensitivity combined with a quick response time

## Disadvantages

* Decreases employment
* Smart phone is compulsory required
* Fails we users have same face identity

# Conclusions

* we have proposed an IOT based smart locker to ensure the security of valuables.
* This system has face detection approach that will count the number of user’s present in front of the locker at any particular time
* This smart locker is much better than traditional locker because it does not require any traditional key to unlock the locker
* It is highly reliable system to ensure the security of our valuables

# Appendixes

**Source Code**

import face\_recognition

import cv2

# This is a demo of running face recognition on live video from your webcam. It's a little more complicated than the

# other example, but it includes some basic performance tweaks to make things run a lot faster:

# 1. Process each video frame at 1/4 resolution (though still display it at full resolution)

# 2. Only detect faces in every other frame of video.

# PLEASE NOTE: This example requires OpenCV (the `cv2` library) to be installed only to read from your webcam.

# OpenCV is \*not\* required to use the face\_recognition library. It's only required if you want to run this

# specific demo. If you have trouble installing it, try any of the other demos that don't require it instead.

––

# Get a reference to webcam #0 (the default one)

video\_capture = cv2.VideoCapture(0)

# Load a sample picture and learn how to recognize it.

obama\_image = face\_recognition.load\_image\_file(r"hema.jpg")

obama\_face\_encoding = face\_recognition.face\_encodings(obama\_image)[0]

# Load a second sample picture and learn how to recognize it.

#biden\_image = face\_recognition.load\_image\_file(r"C:\Users\Kalkeseetharaman P K\Desktop\karan.jpg")

#biden\_face\_encoding = face\_recognition.face\_encodings(biden\_image)[0]

'''biden\_image1 = face\_recognition.load\_image\_file("nikhil.jpg")

biden\_face\_encoding1 = face\_recognition.face\_encodings(biden\_image1)[0]'''

# Create arrays of known face encodings and their names

known\_face\_encodings = [

obama\_face\_encoding

#biden\_face\_encoding,

#biden\_face\_encoding1

]

known\_face\_names = [

"hema"

#"Karan",

#"Nikhil"

]

# Initialize some variables

face\_locations = []

face\_encodings = []

face\_names = []

process\_this\_frame = True

while True:

# Grab a single frame of video

ret, frame = video\_capture.read()

# Resize frame of video to 1/4 size for faster face recognition processing

small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

# Convert the image from BGR color (which OpenCV uses) to RGB color (which face\_recognition uses)

rgb\_small\_frame = small\_frame[:, :, ::-1]

# Only process every other frame of video to save time

if process\_this\_frame:

# Find all the faces and face encodings in the current frame of video

face\_locations = face\_recognition.face\_locations(rgb\_small\_frame)

face\_encodings = face\_recognition.face\_encodings(rgb\_small\_frame, face\_locations)

face\_names = []

for face\_encoding in face\_encodings:

# See if the face is a match for the known face(s)

matches = face\_recognition.compare\_faces(known\_face\_encodings, face\_encoding)

name = "Unknown"

# If a match was found in known\_face\_encodings, just use the first one.

if True in matches:

first\_match\_index = matches.index(True)

name = known\_face\_names[first\_match\_index]

face\_names.append(name)

process\_this\_frame = not process\_this\_frame

# Display the results

for (top, right, bottom, left), name in zip(face\_locations, face\_names):

# Scale back up face locations since the frame we detected in was scaled to 1/4 size

top \*= 4

right \*= 4

bottom \*= 4

left \*= 4

# Draw a box around the face

cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

# Draw a label with a name below the face

cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)

font = cv2.FONT\_HERSHEY\_DUPLEX

cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

# Display the resulting image

cv2.imshow('Video', frame)

# Hit 'q' on the keyboard to quit!

if cv2.waitKey(1) & 0xFF == ord('q'):

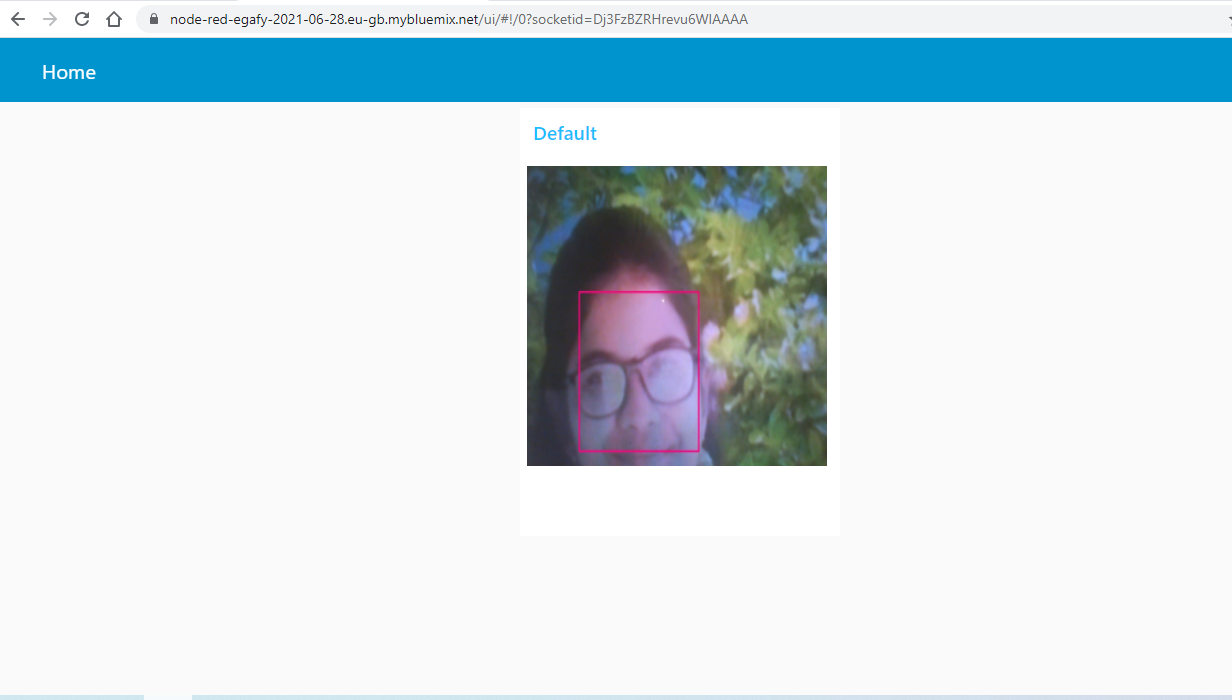
break

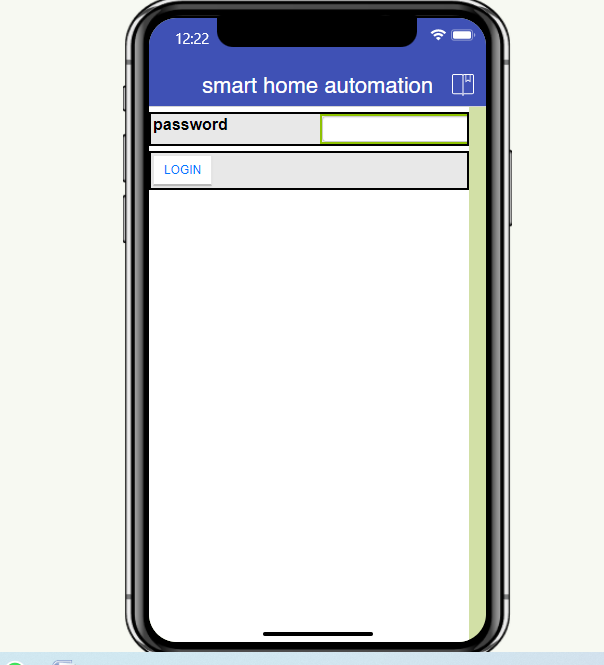
# Release handle to the webcam

video\_capture.release()

cv2.destroyAllWindows()

**UI output :**



**Mobile Output :** ****