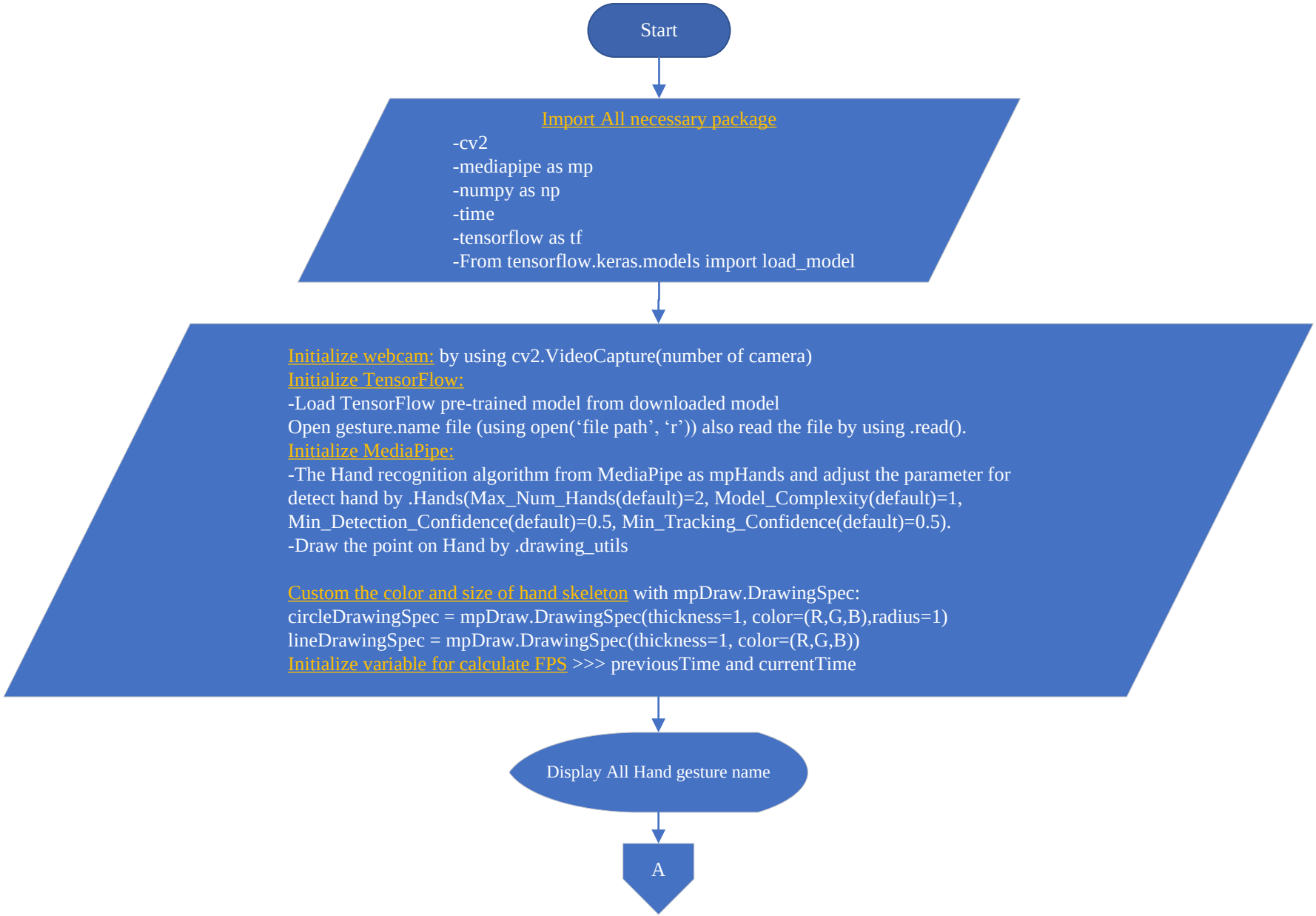
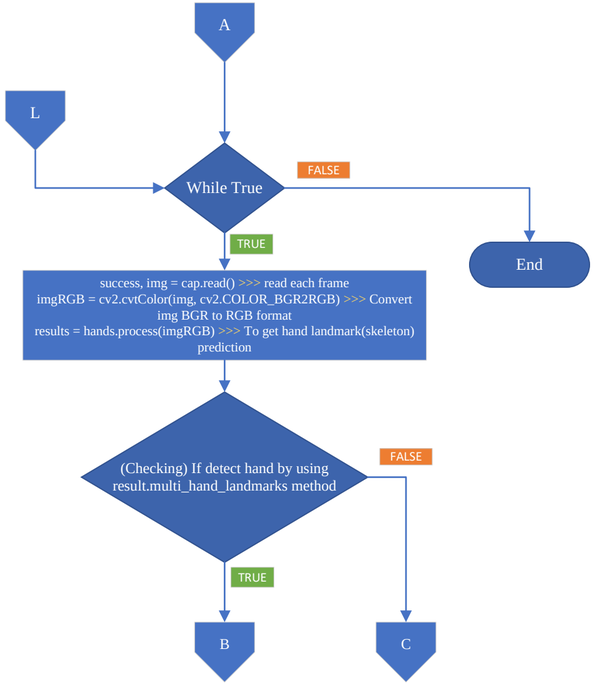
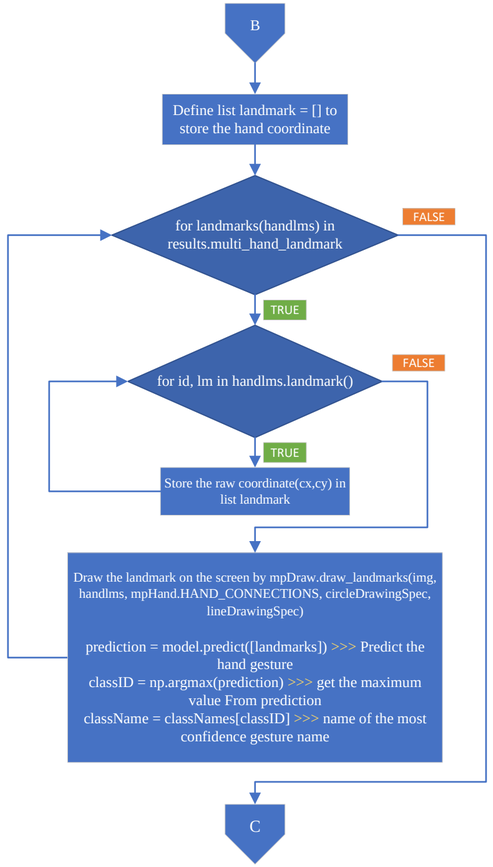
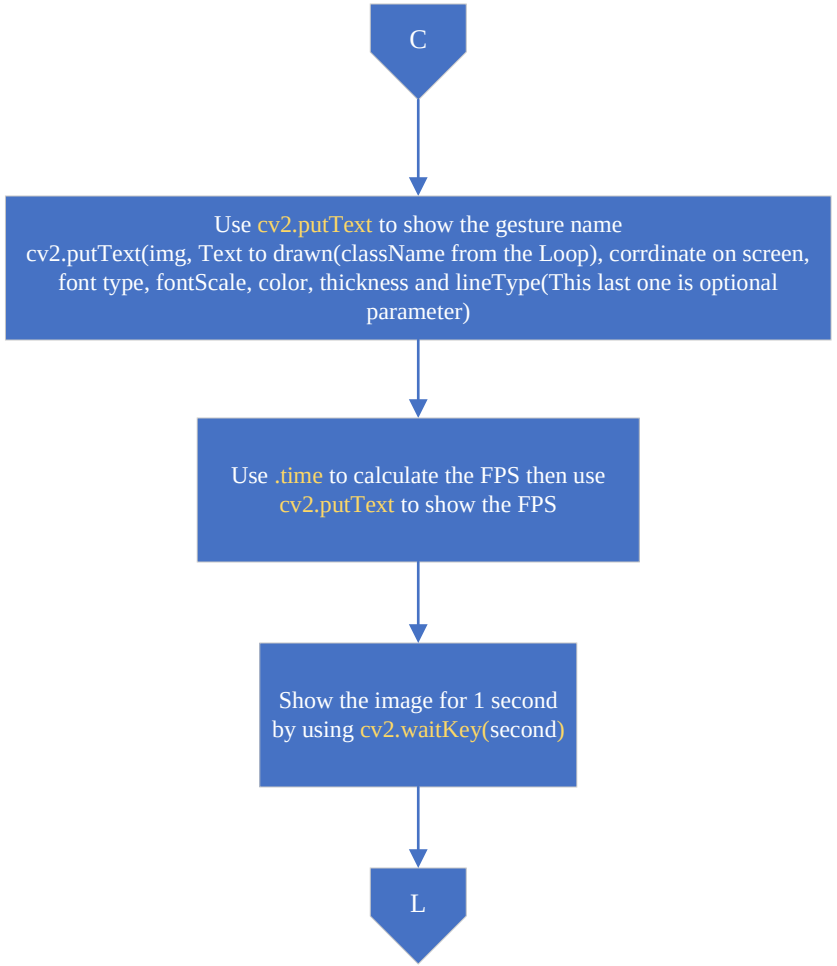

Start
Import All necessary package
-cv2
-mediapipe as mp
-numpy as np
-time
-tensorflow as tf
-From tensorflow.keras.models import load_model
Initialize webcam: by using cv2.VideoCapture(number of camera)
Initialize TensorFlow:
-Load TensorFlow pre-trained model from downloaded model
Open gesture.name file (using open('file path', 'r')) also read the file by using .read().
Initialize MediaPipe:
-The Hand recognition algorithm from MediaPipe as mpHands and adjust the parameter for detect hand by .Hands(Max_Num_Hands(default)=2, Model_Complexity(default)=1, Min_Detection_Confidence(default)=0.5, Min_Tracking_Confidence(default)=0.5).
-Draw the point on Hand by .drawing_utils
Custom the color and size of hand skeleton with mpDraw.DrawingSpec:
circleDrawingSpec = mpDraw.DrawingSpec(thickness=1, color=(R,G,B),radius=1)
lineDrawingSpec = mpDraw.DrawingSpec(thickness=1, color=(R,G,B))
Initialize variable for calculate FPS >>> previousTime and currentTime
Display All Hand gesture name
A


A
L
While True
FALSE
TRUE
End
success, img = cap.read() >>> read each frame
imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) >>> Convert img BGR to RGB format
results = hands.process(imgRGB) >>> To get hand landmark(skeleton) prediction
(Checking) If detect hand by using result.multi_hand_landmarks method
FALSE
TRUE
B
C


B
Define list landmark = [] to store the hand coordinate
for landmarks(handlms) in results.multi_hand_landmark
FALSE
TRUE
for id, lm in handlms.landmark()
FALSE
TRUE
Store the raw coordinate(cx,cy) in list landmark
Draw the landmark on the screen by mpDraw.draw_landmarks(img, handlms, mpHand.HAND_CONNECTIONS, circleDrawingSpec, lineDrawingSpec)
prediction = model.predict([landmarks]) >>> Predict the hand gesture
classID = np.argmax(prediction) >>> get the maximum value From prediction
className = classNames[classID] >>> name of the most confidence gesture name
C


C
Use cv2.putText to show the gesture name
cv2.putText(img, Text to drawn(className from the Loop), corrdinate on screen, font type, fontScale, color, thickness and lineType(This last one is optional parameter)
Use .time to calculate the FPS then use cv2.putText to show the FPS
Show the image for 1 second by using cv2.waitKey(second)
L