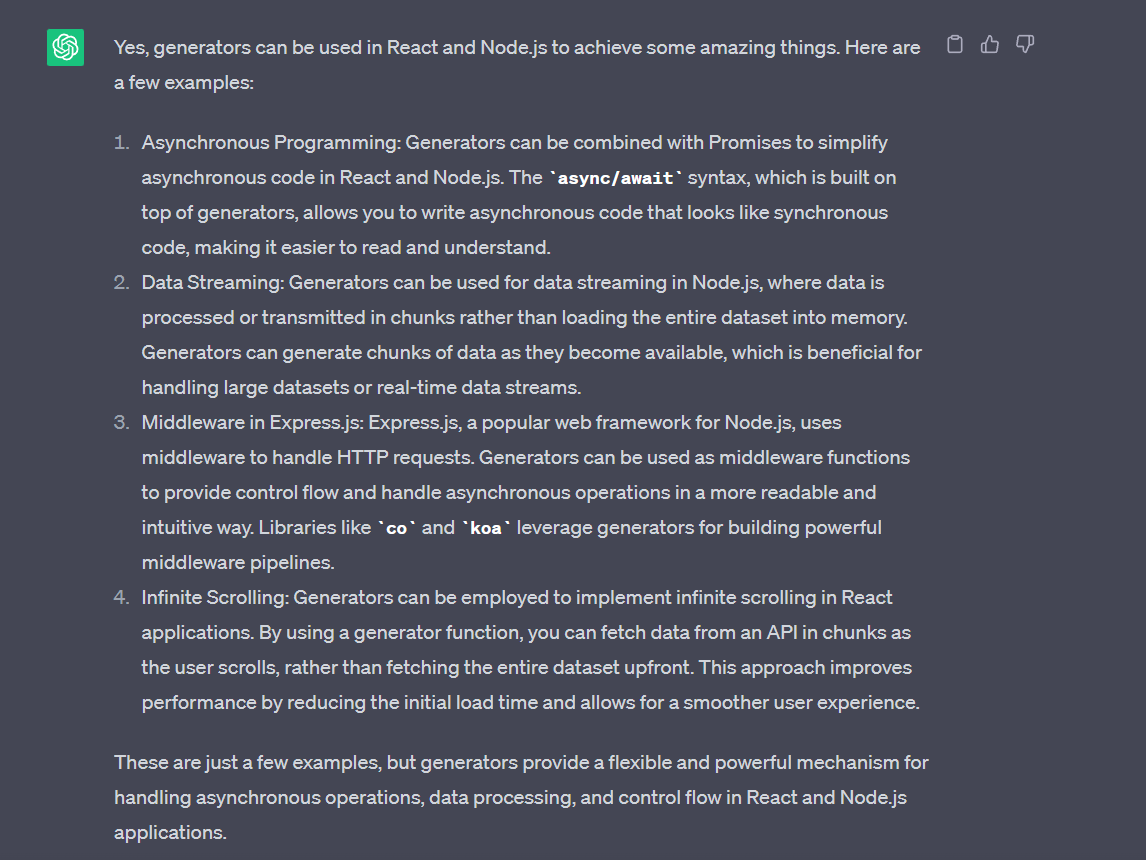
Generators are really amazing things

They can be used to do amazing things in react and node



**Data collection module**

We will be using OPENCV to initialize the camera and work with it

The images will be taken using the web camera and saved in local machine for this project duration

The saved images will be passed to MTCNN

MTCNN returns us three things

1. Bounding Box: The bounding box is a rectangle that encompasses the detected face in an image or a video frame. It specifies the coordinates (x, y) of the top-left corner and the width and height of the bounding box, allowing you to localize the face region accurately.
2. Facial Landmarks: Facial landmarks refer to specific points on the face, such as the corners of the eyes, the tip of the nose, and the corners of the mouth. MTCNN can also provide the coordinates of these facial landmarks, which are crucial for tasks like face alignment, expression analysis, and feature extraction.
3. Probability or Confidence Score: MTCNN also provides a probability or confidence score associated with each detected face. This score indicates the model's confidence in its face detection and can be used as a threshold for filtering out false positive detections. It is a confidence level whether there is a face or not

We will do all the necessary required operations in our image data like resizing the images as per the obtained bounding box from the MTCNN

**Generate face embeddings**

The information provided by MTCNN, such as the bounding box coordinates and facial landmarks, can be used as input for generating face embeddings using models like InsightFace.

Using Insightface we will be generating embeddings for each of the images

And, all the embedding vectors will be saved

An embedding vector is a numerical representation of a face obtained from a deep learning model, which encodes the unique facial features and characteristics in a high-dimensional space

**Building and Training the model**

Previously we did the data collection and generating face embeddings. Now, by using those embeddings we will be training our model

We will be using Keras to build our model because it provides a user-friendly and intuitive API for building and training deep learning models

It makes easier to focus on model development and experimentation without getting caught up in low-level implementation details.

Adtionally, Keras provides easy integration with Tensorflow

We will be using the sequential model because as per our research we found out that the architecture of the sequential model allows us to easily stack layers in a linear manner one by one

All the embedding vectors that we had previously generated will be passed to the model with all the other necessary required parameters to train the model

After training the model will be saved

**Prediction**

We will use open CV to initialize camera and initiate the video clips

While video is on we will be reading each of the frames one by one

Here also we will be using MTCNN for face detection

We will pass each of the frames to the MTCNN which will return us the bounding box and the facial landmarks

We will pass the frames, as well as the information about the bounding boxes and facial landmarks obtained from the MTCNN to the Insightface, and we will be generating embedding vector

The model that we had saved previously will be loaded in to the memory

We will also locate the previously saved embedding vectors

Cosine Similarity for Verification

Here we will use the concept of cosine similarity, we can define a function that can calculate the cosine similarity between two vectors

We will then perform the cosine similarity between the embedding vectors previously saved in the system and the ones that were captured during the prediction phase

We will sum up the calculated cosine similarity and calculate the average value

If the calculated average value lies within the certain threshold value then the face recognition is successful else the face recognition is unsuccessful