**CHAPTER 4**

**DESIGN AND IMPLEMENTATION OF THE SYSTEM**

This chapter presents the system design which aims to model an efficient method to face recognition system. Face recognition system is developed by using Electron Framework and Python Programming Language. Some interfaces and experimental results of the system are also described. This system is divided into three processes: Feature Extraction Process, Training Model Process and Recognition Process.

**4.1. Block Diagram of the System**

The block diagram of the system is shown in Figure 4.1. There are two types of the process: training phase and prediction phase.

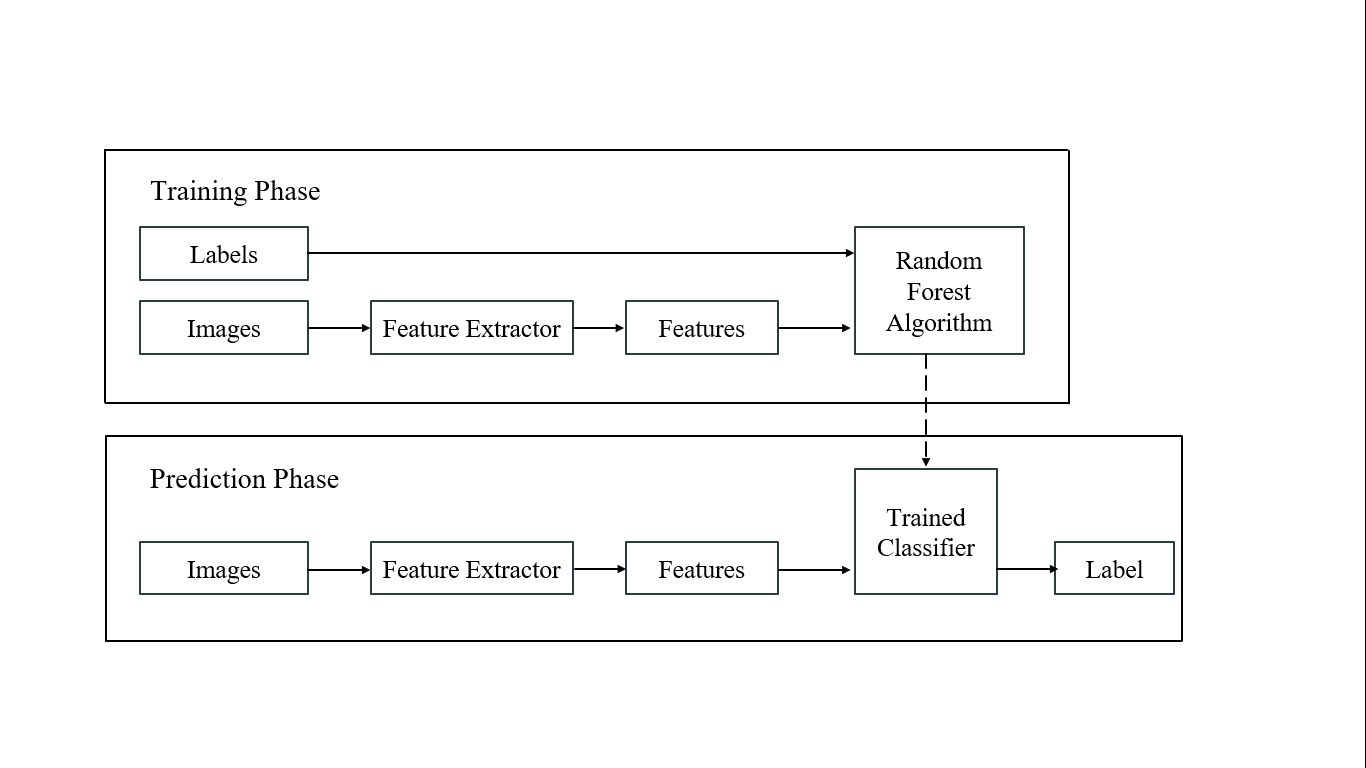


Figure 4.1. Block Diagram of the System

**4.2. Architecture of the Face Recognition System**

The architecture of face recognition system is an efficient method to recognize the person. The system provides to achieve the fast and effective ways to perform recognition processes for the images. These recognition processes are facial feature extraction process, training process and face recognition process. The face detection method is used to detect faces. Facial feature extraction process is to extract the embedding features from the face detection images. Face recognition process is to recognize person of the tested images. For face detection process, Convolutional Architecture for Fast Feature Embedding (Caffe) model is used. Torch model is used to extract 128-d embedding features of the images for the feature extraction method. Convolutional Architecture for Fast Feature Embedding (Caffe) model and Torch model are under the Deep Neural Network module of OpenCV. Random Forest Algorithm is to produce the classification model of the trained process and the tested images pass through the classification model. Random Forest Algorithm is the fastest running times among the best machine learning algorithms. The system aims to recognize the person of tested images and to use Electron Framework, face detection model, feature extraction and the machine learning algorithms.

**4.3. Face Detection Process**

Face detection process is firstly optimized. Convolutional Architecture for Fast Feature Embedding (Caffe) model is used to detect faces from the upload images and deploy.prototxt and res10\_300x300\_ssd\_iter\_140000.caffemodel files are loaded for face detection in the readNetFromCaffe function under Deep Neural Network module of OpenCV. Figure 4.2 and 4.3 show the Electron Framework user interface of the Face Recognition System using Random Forest.

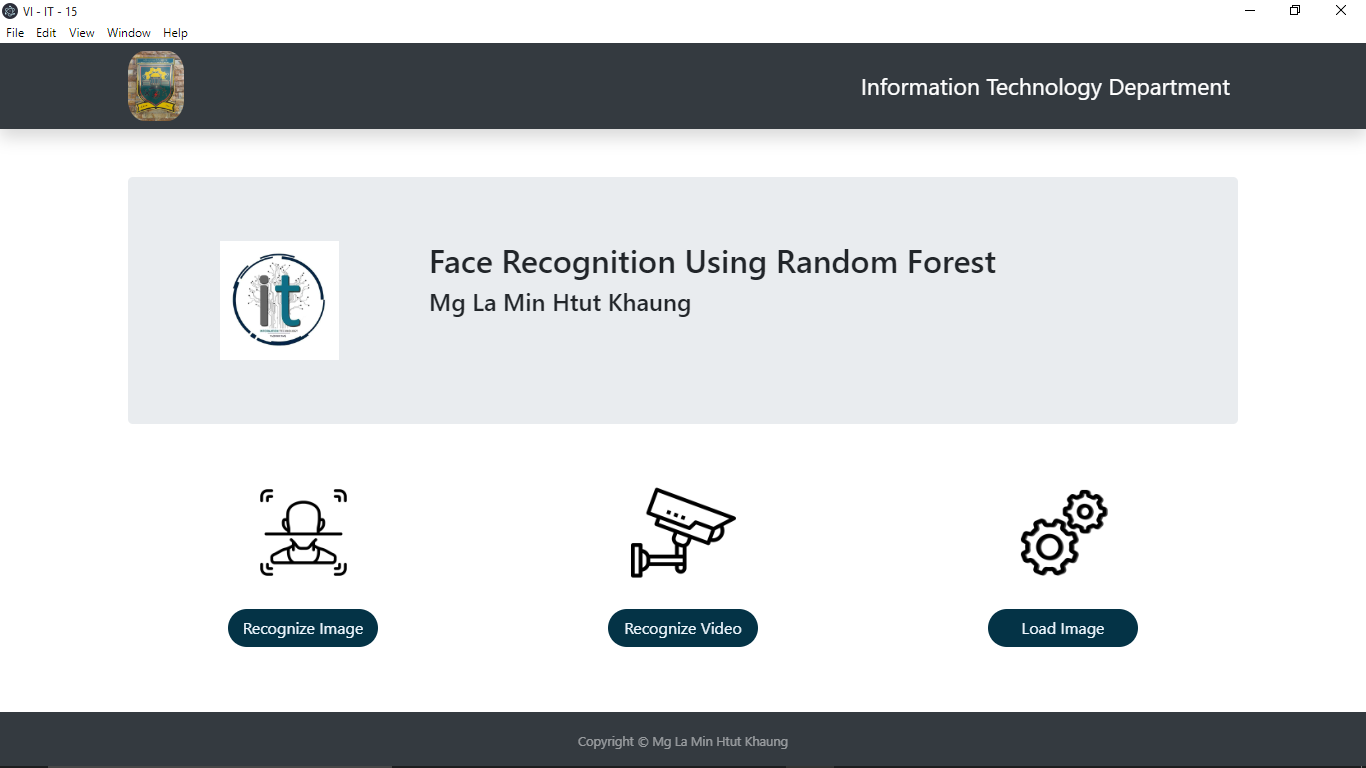


Figure 4.2. Main Interface

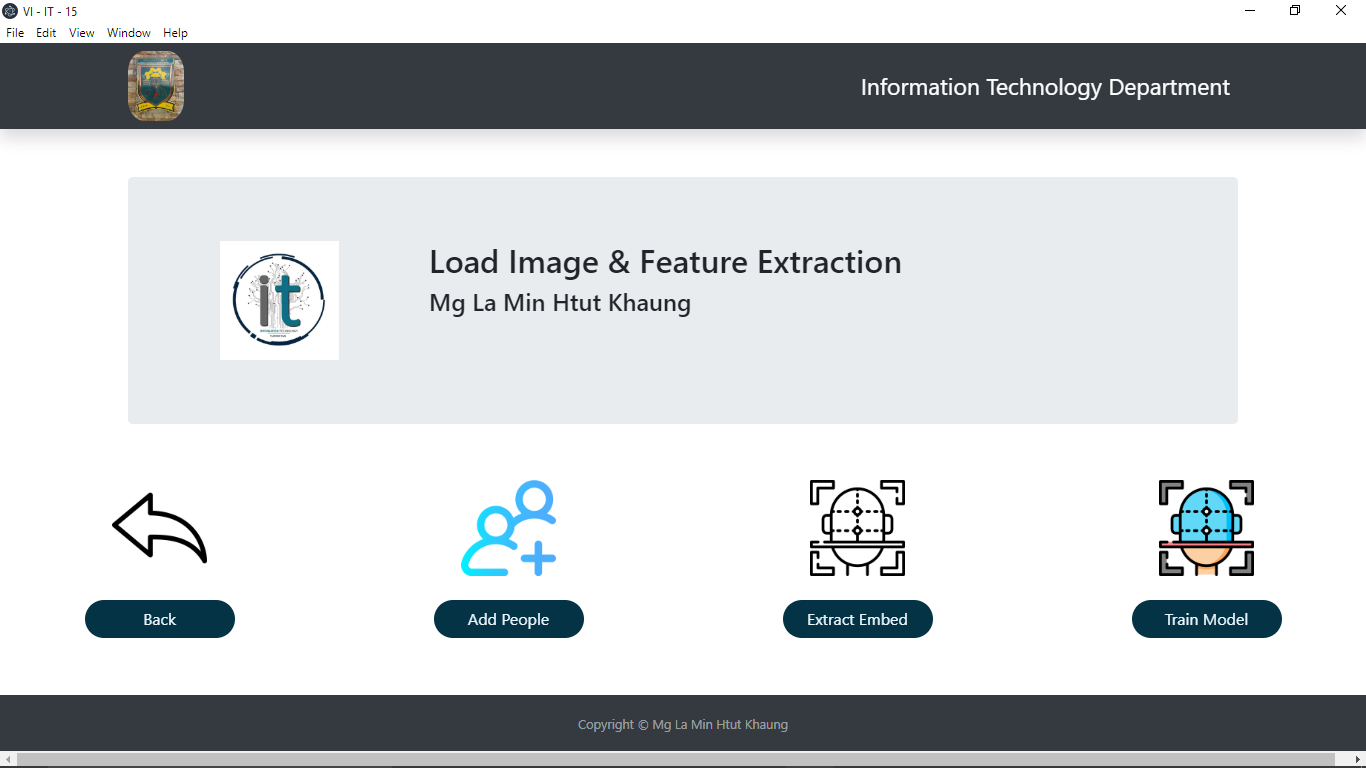


Figure 4.3. Load Image Interface

The face detection model (Caffe) detects the face from the upload images and Figure 4.4 shows the face detection process.

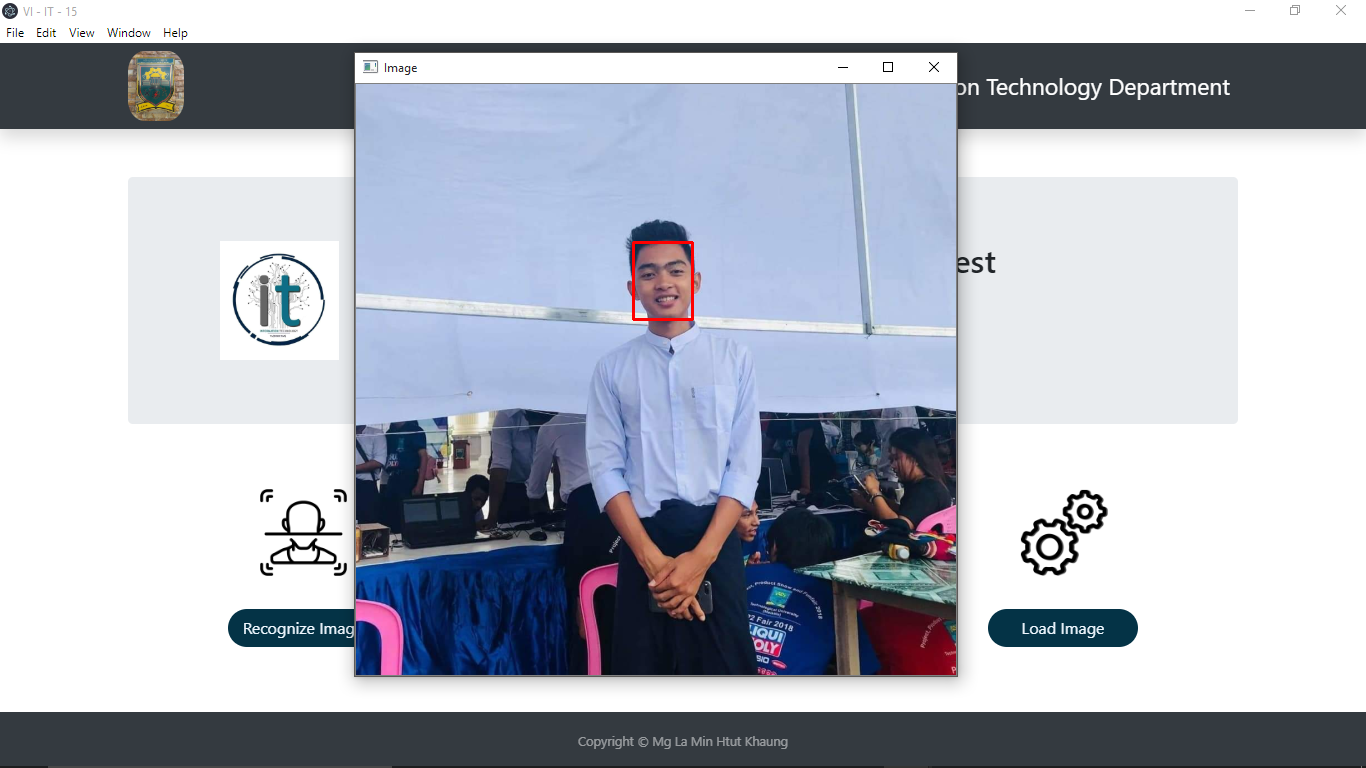


Figure 4.4. The Face Detection Process

**4.4. Feature Extraction Process**

After face detection process, openface\_nn4.small2.v1.t7 file is loaded in the readNetFromTorch function to load serialized face embedding model. Figure 4.5 shows the initial step of feature extraction process.

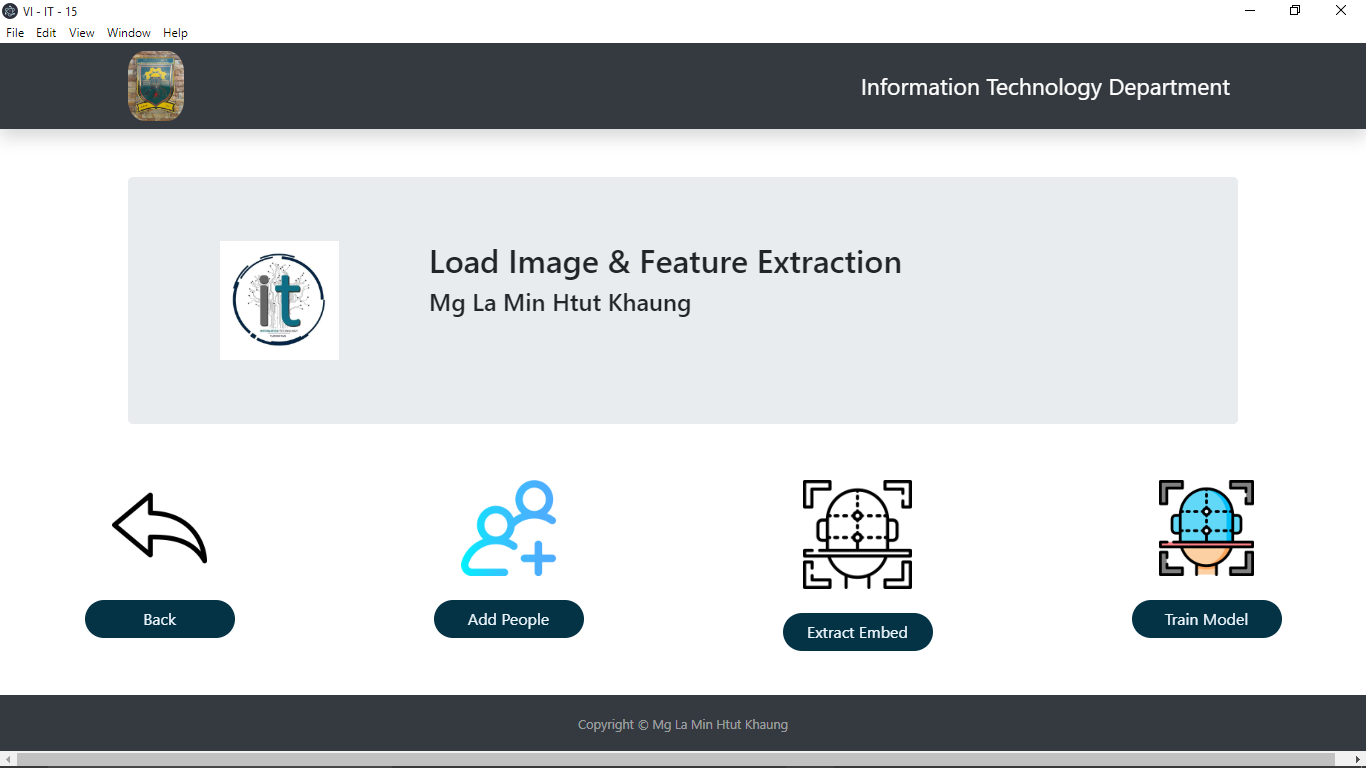


Figure 4.5. Initial Step of Feature Extraction Process

The system grabs the paths to the input images in the dataset and initialize the lists of extracted facial embedding and corresponding people names. The corresponding name from the image path are extracted. The image of faces using face detector are loaded and resized to have a width of 600 pixels. The extracted face images are stored in the embeddings.pickle file. Figure 4.6 shows how the extract embedding processes work.

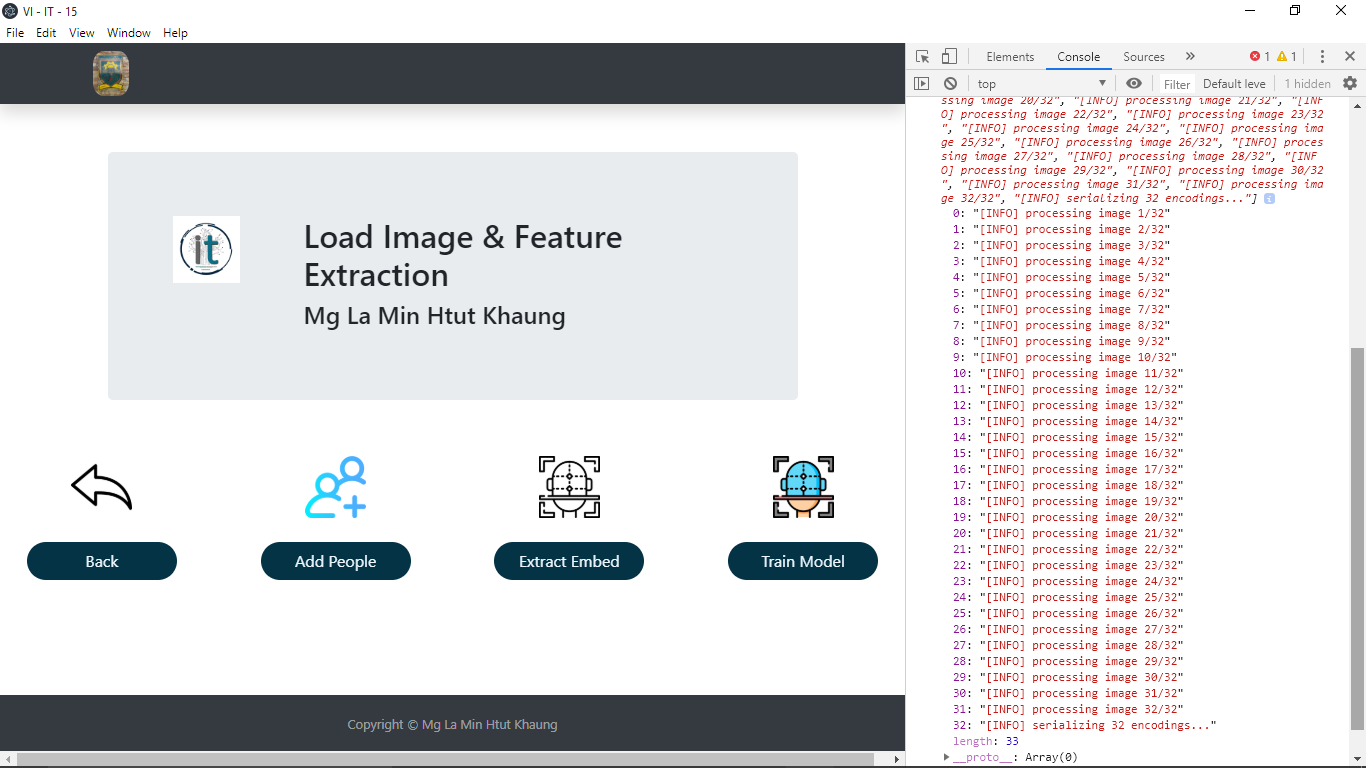


Figure 4.6. The Extracted Embedding Process

The process takes a few times while the process work. After extracted embedding processes have been done, the Electron framework user interface show the text which is Successfully Done Extracted Embedding Process in the model box. Figure 4.7 shows the Complete Extract Embedding Process.

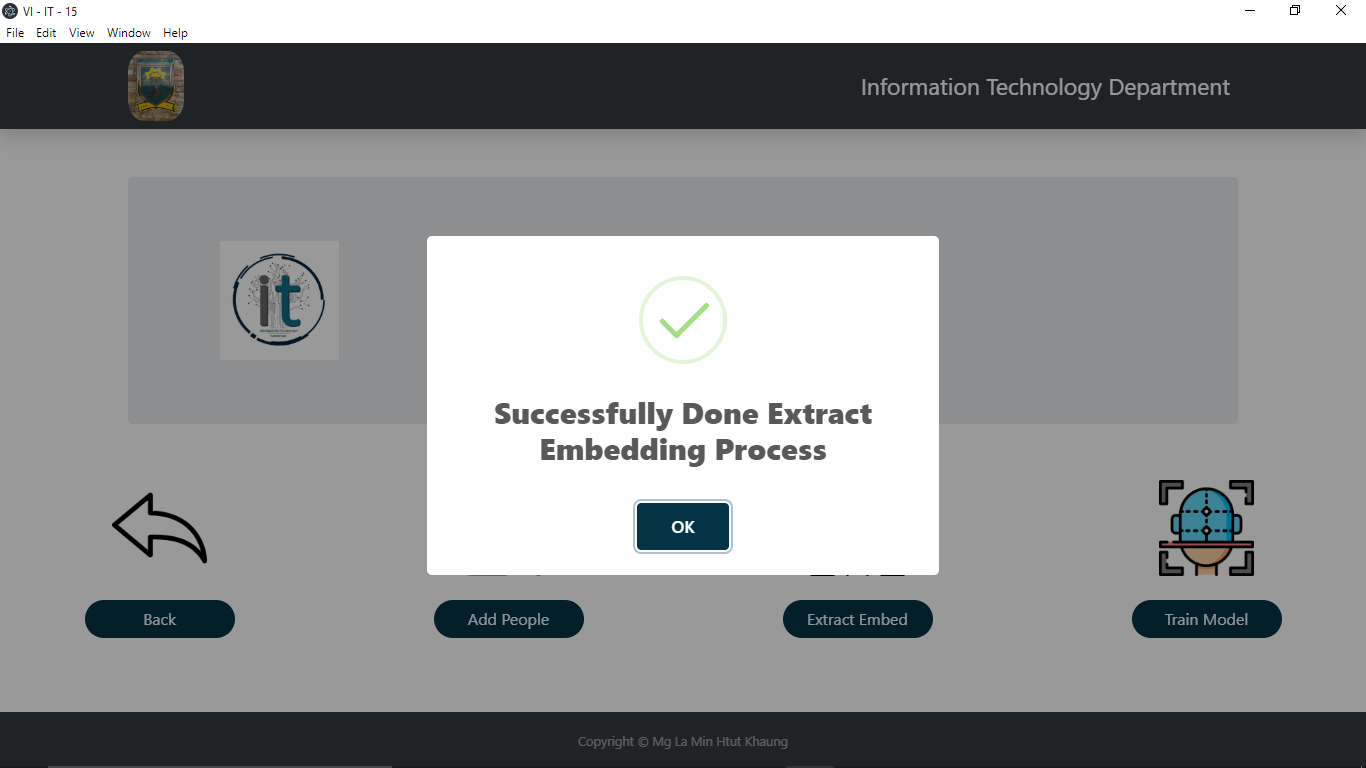


Figure 4.7. The Complete Extract Embedding Process

**4.5. Training Model Process**

Figure 4.8 is the initial step of the training model process.

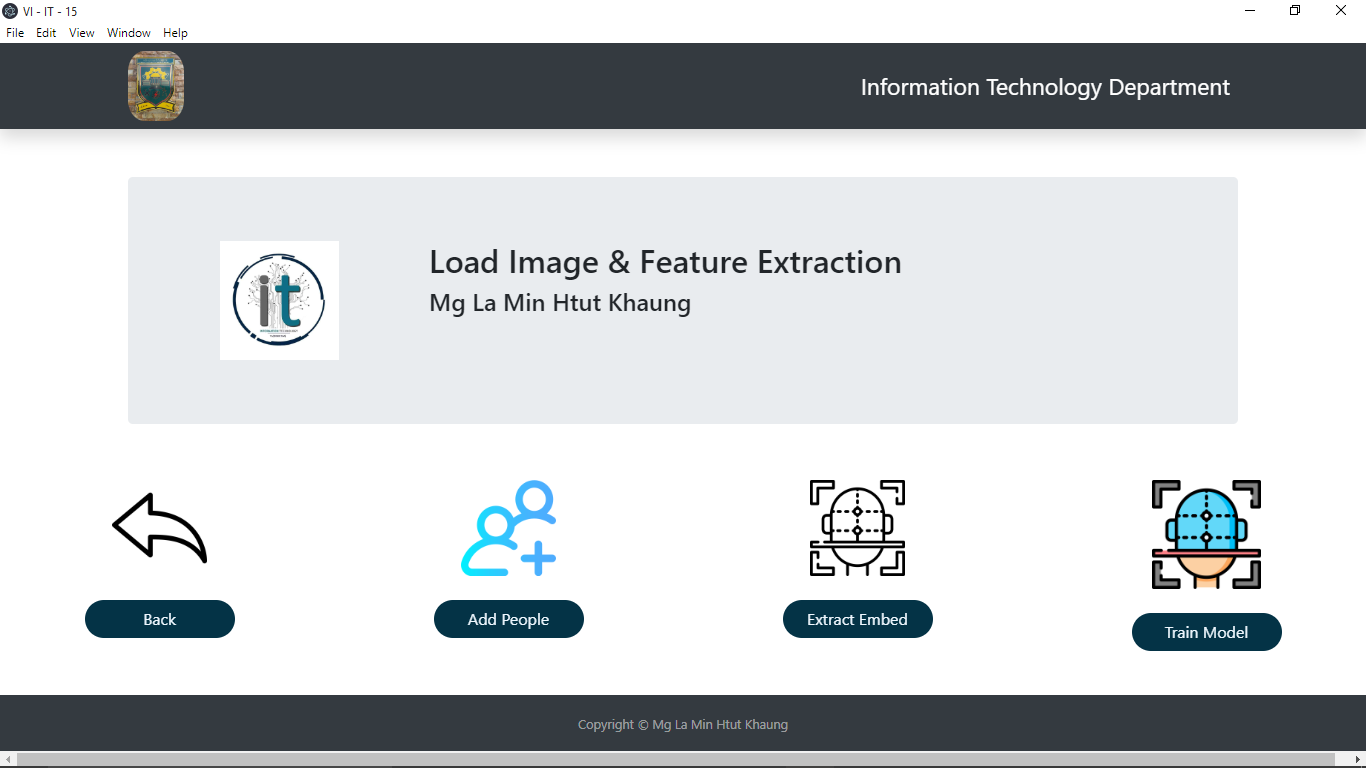


Figure 4.8. Initial Step of Training Model Process

Firstly, the system loads embeddings.pickle file which is stored the 128-d embedding features by the extract embedding process. The embeddings.pickle file also contain the label and the 128-d embedding features. In the training step, Random Forest Algorithm is used to train the process. Random Forest Algorithm has classification and regression methods. In the system, classification method is used. Random Forest Classifier under sklearn package is trained the face data with 100 tree times, 1 minimum samples leaf and 50 random state. The trained data using Random Forest Algorithm is stored in the recognize.pickle file as a classification model. The labels are stored in the le.pickle file.

The training model process reads the serialized face extracted features. After training model process, the system show the test which is Successfully Done Training Model Process in the model box. Figure 4.9 is the model box of training model process. The system flow is firstly to do extract embedding process which loads all the images in the dataset. The embedded process produces the embedded features as serialized database in the pickle file. The training model read the serialized pickle file and split feature and label. As a output, the training process produces the trained features using Random Forest Classifier and the labels. After training model process is completed, recognize process can be done.

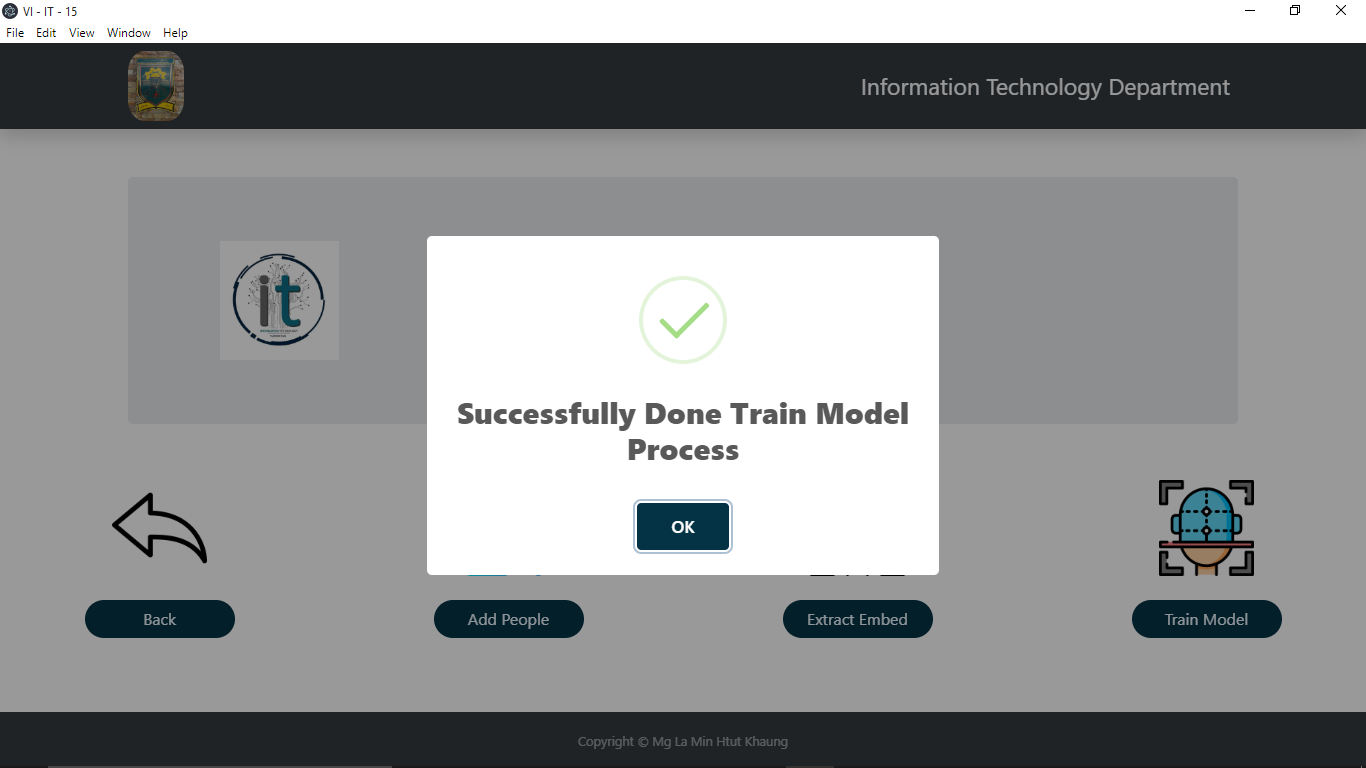


Figure 4.9. The Complete Training Model Process

**4.6. Recognized Process**

In the recognized process, the file chooser is used to choose the tested image. Face detection process detects the face of the tested image using Caffe model. Feature extraction process produces the tested image face features as 128-d embedding vector. The extracted features of face image are passed through into the trained model which are stored as recognizer.pickle file. The label encoder is loaded.

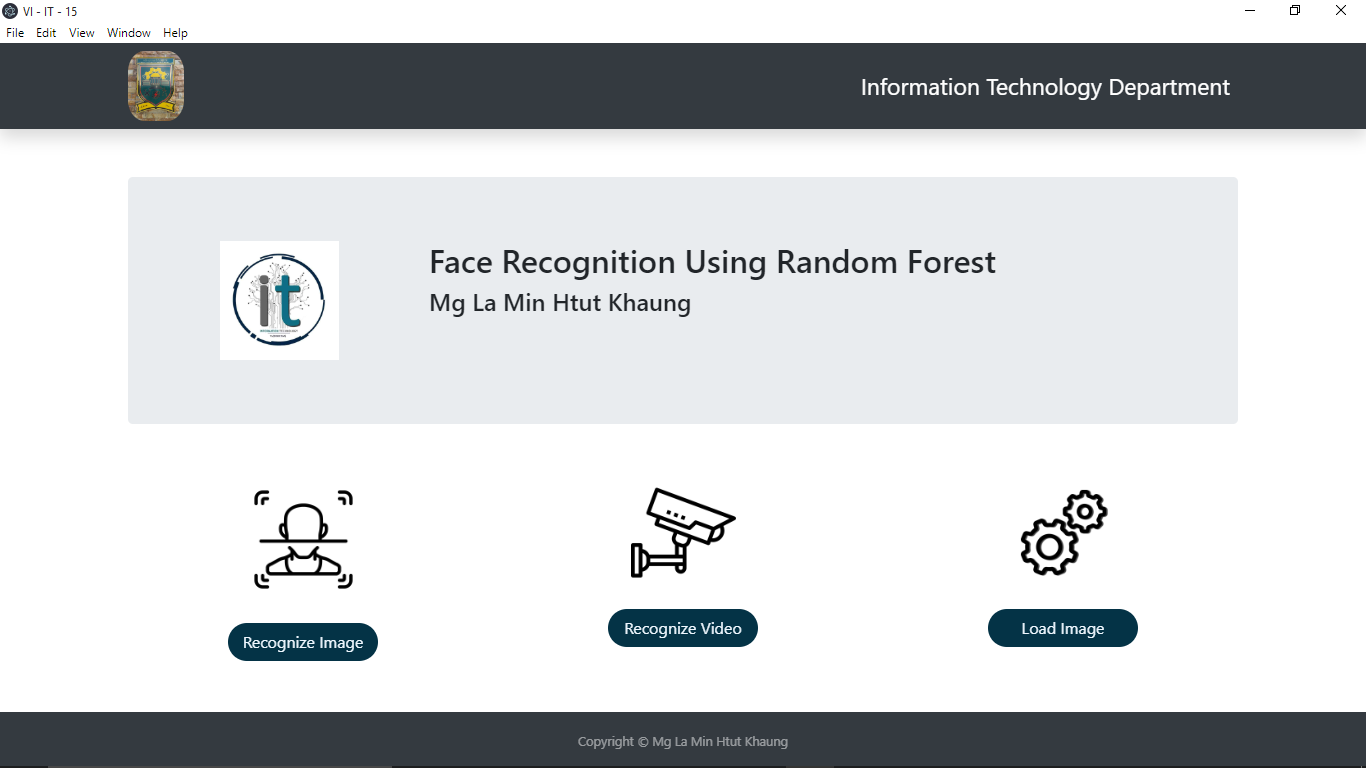


Figure 4.10. Initial Step of Recognized Image Process

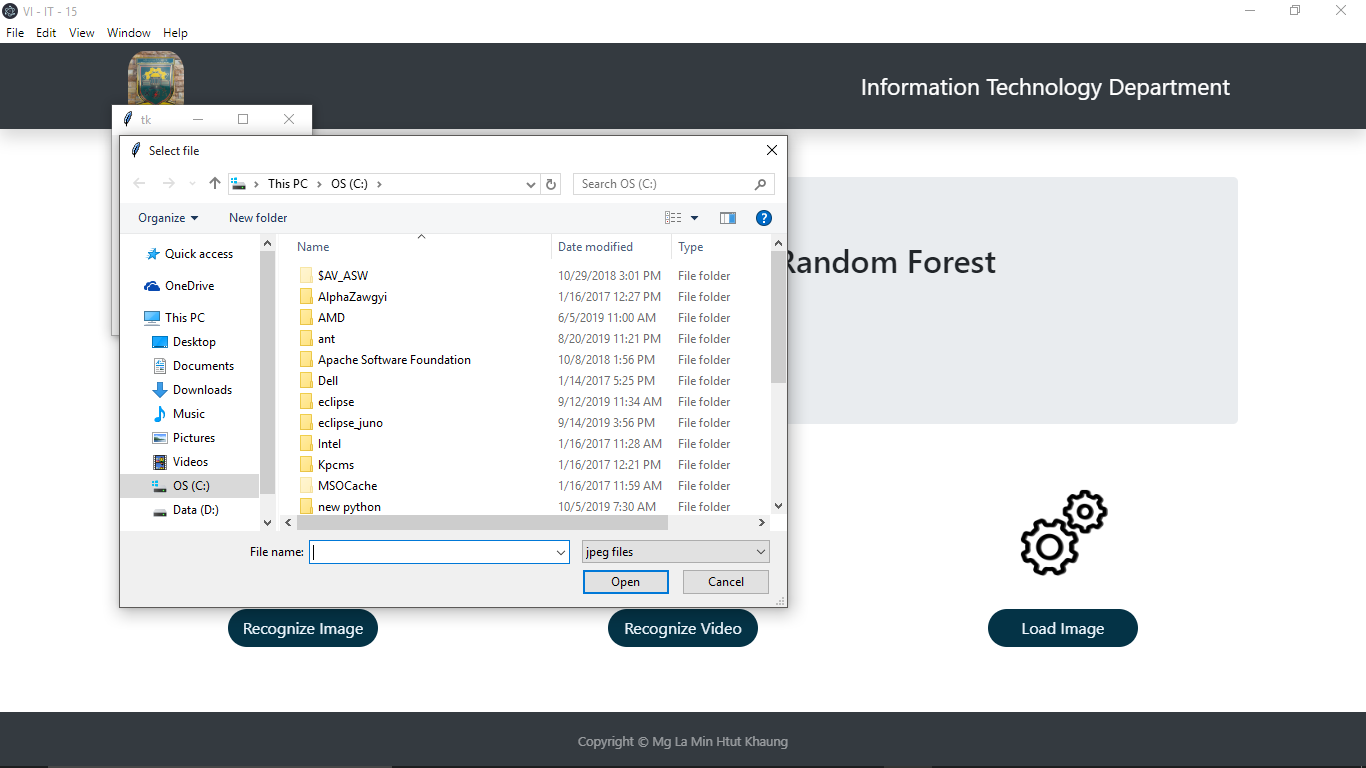


Figure 4.11. File Chooser to choose the tested Image

Figure 4.10 is the initial step of recognized image process. The recognized process can be divided into two process: recognize image and recognize video. After initial step, the file chooser box is shown to choose the tested image. If there is a tested image, the system starts recognize image process. Figure 4.11 is the file chooser to choose tested image. The file chooser is written by Tkinter package in Python Language. The image directory is loaded by the filedialog.askopenfilename function.

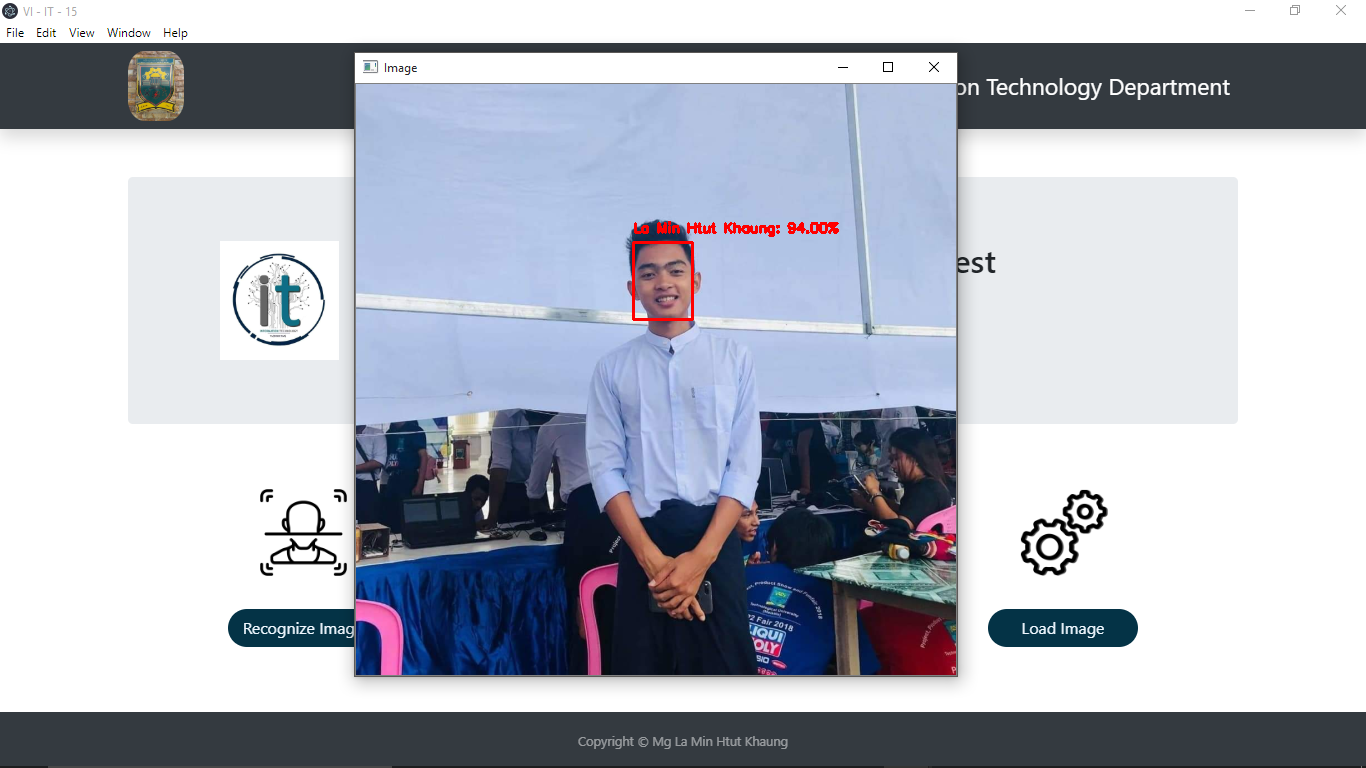


Figure 4.12. Result of Recognized Image Process

To read the image from the image directory, cv2.imread() function under cv2 package of OpenCV is used. After reading the input image, the image is resized as width 600 pixels and blobFromImage() function under Deep Neural Network module of OpenCV is used to construct a blob for the input image. The blob image is put into the face detector module (Caffe). Figure 4.12 is the result of recognize image process. As a result, the tested image is shown the face detected frame and recognize name of the corresponding person.

In recognized video process, the system is developed as a real time face recognition system. Figure 4.13 is the initial step of recognize video process. VideoStream and FPS model are imported under imutils package. VideoStram is used to open webcam and FPS is to count the elapsed time and display FPS information. The real time video is framed with 600 pixels frame. The resize frame is put into the blobFromImage() function under Deep Neural Network module. The blobFromImage() value is constructed into a variable. The system apply OpenCV’s deep learning-learning based face detector to localize face in the video. The setInput() function is used to put the variable into the detector module. The face detection process is optimized the variables. Face detection process detect the face in the video and extracted the embedding feature by using Torch model. Figure 4.14 is recognized video process. The accuracy depends on the face positions in the real time video from a web cam.

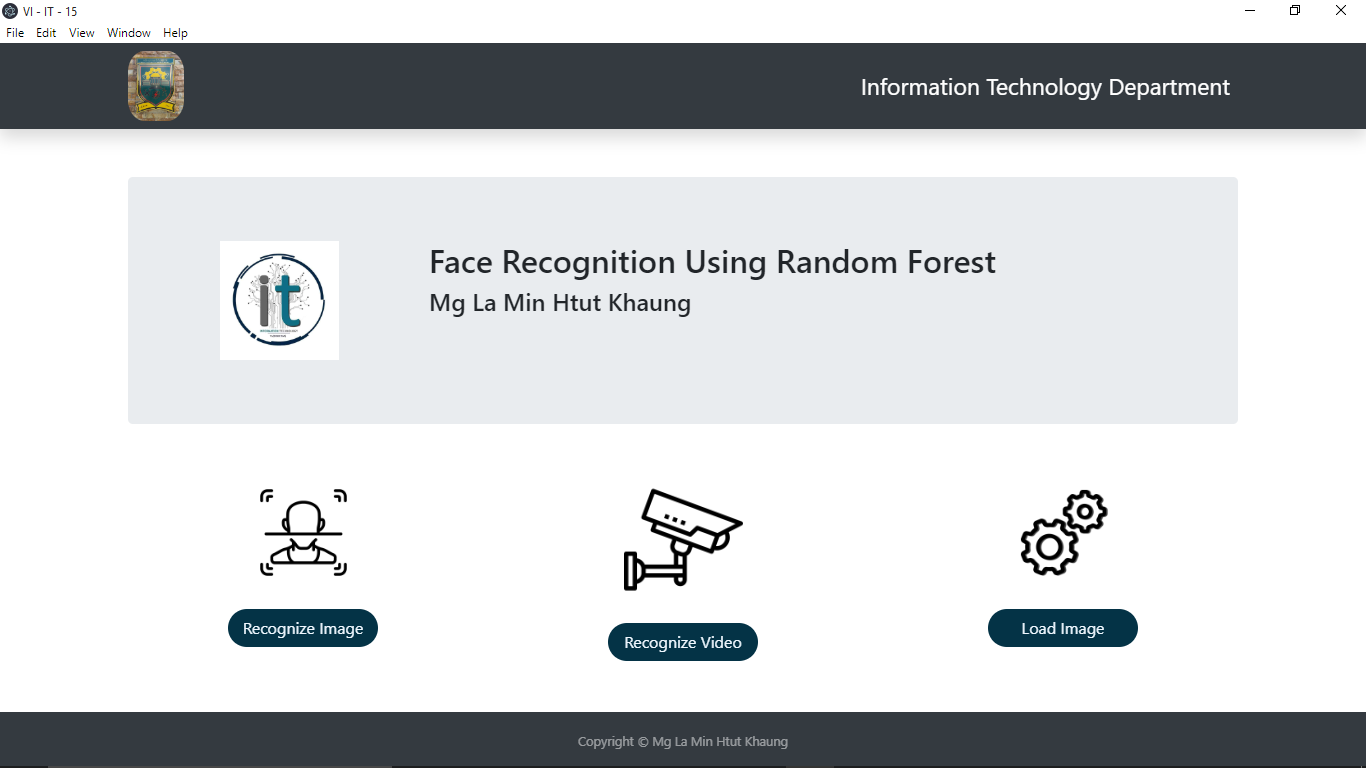


Figure 4.13. Initial Step of Recognized Video Process

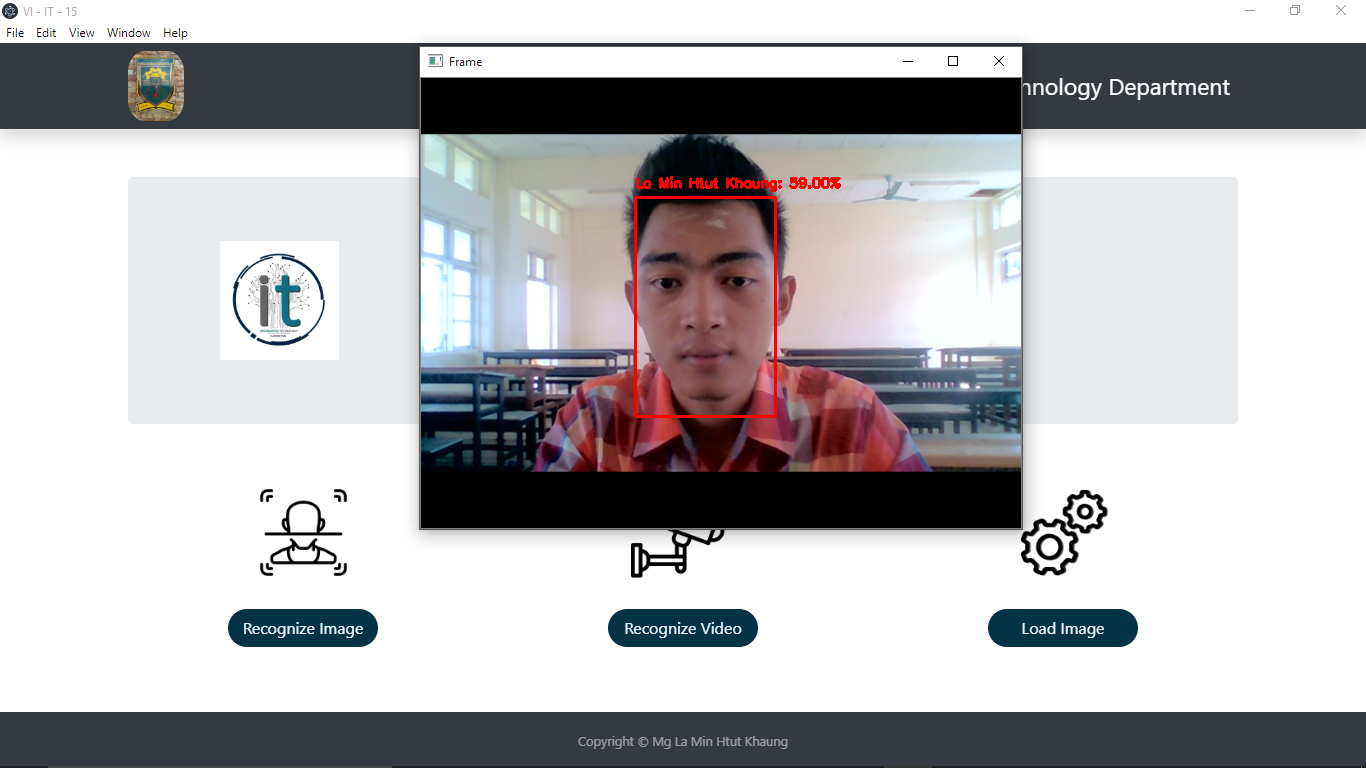


Figure 4.14. Recognized Video Process

**4.7. Load Image and Feature Extraction**

The system is developed to add easily another person from the user interface. In the user interface, there is a add people button. By clicking the add people button, the model box which contains Enter your name here label and Go button is shown. Name is added and clicked the Go button in the model box. After doing this, file chooser is shown to upload the images as the trained images. Figure 4.15 and Figure 4.16 are the initial step of image uploading process and the modal box to upload name and image.

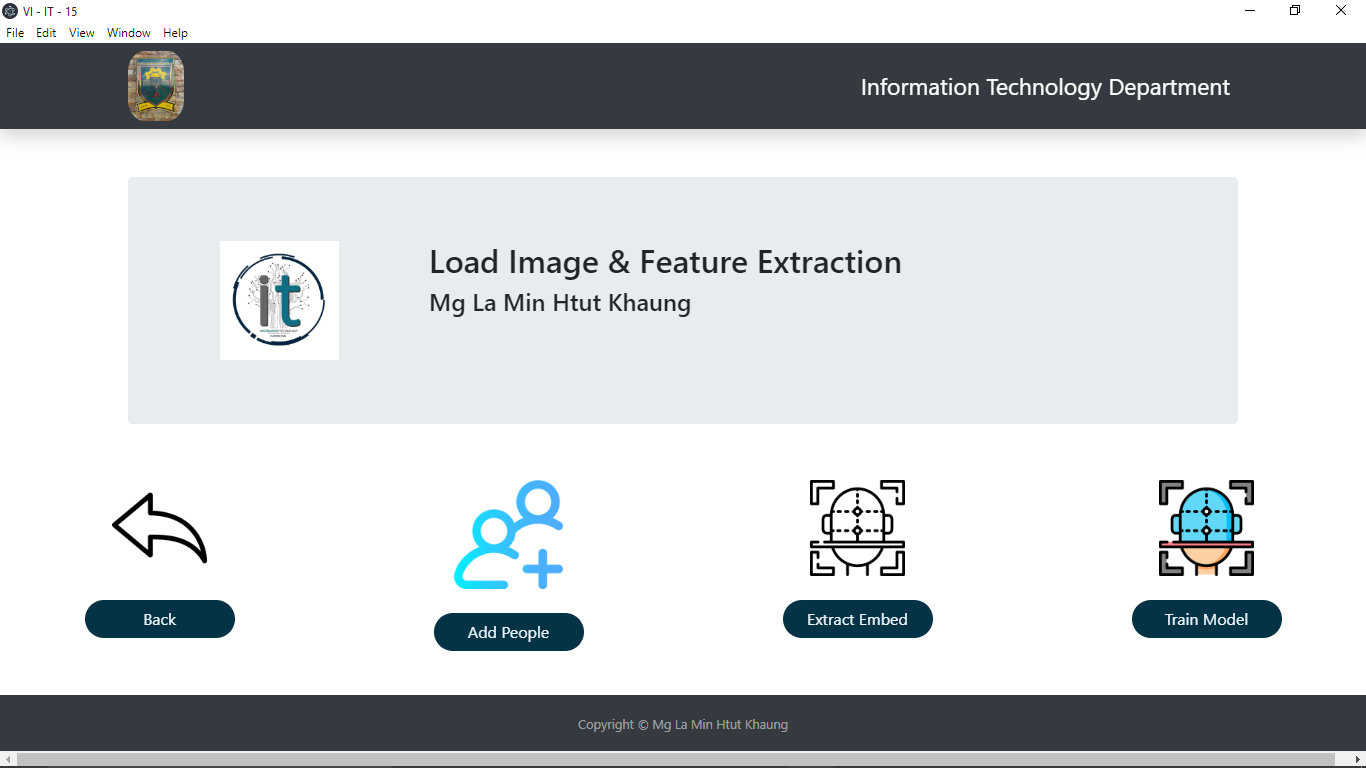


Figure 4.15. Initial Step of Image Uploading Process

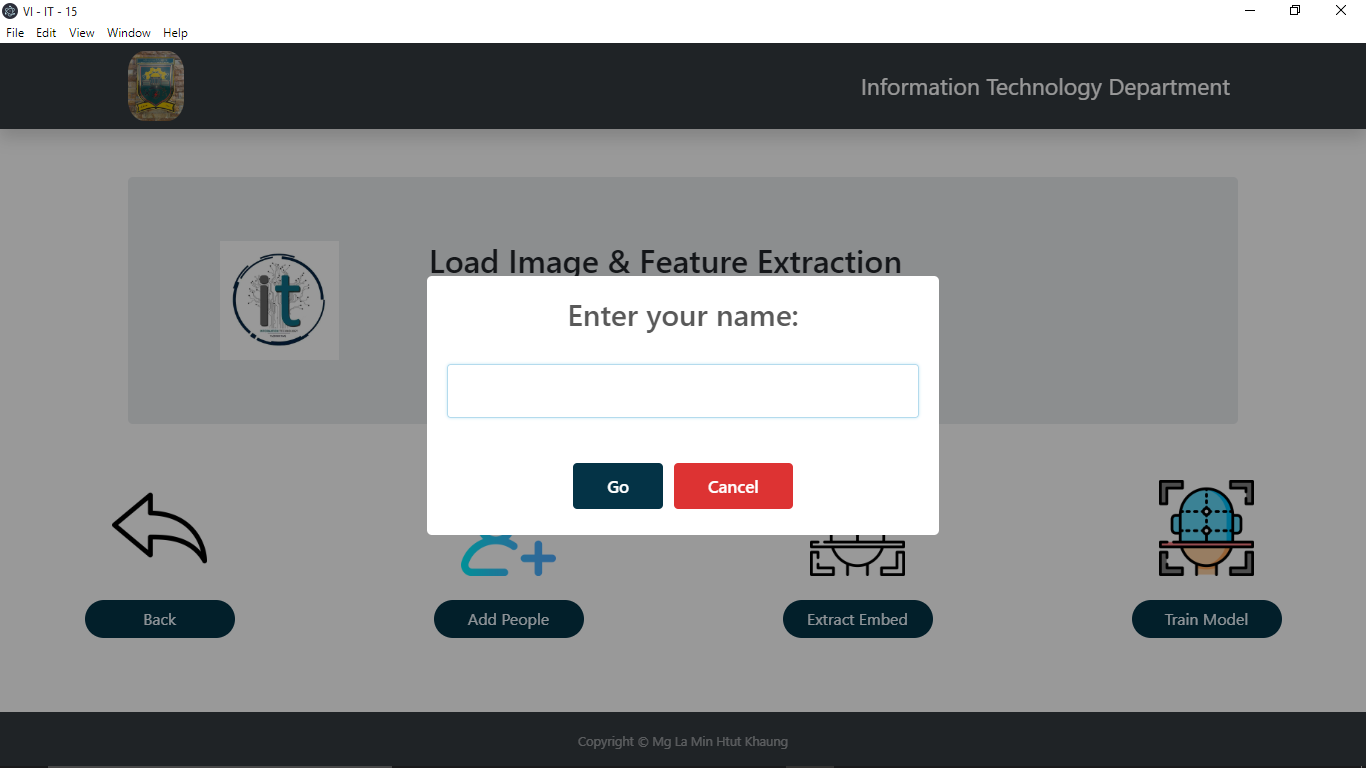


Figure 4.16. Modal Box to Upload Names and Images

After uploading the process, the system show the test which is Successfully Done Image Uploading Process in the model box. Figure 4.17 is the modal box of complete load image process.

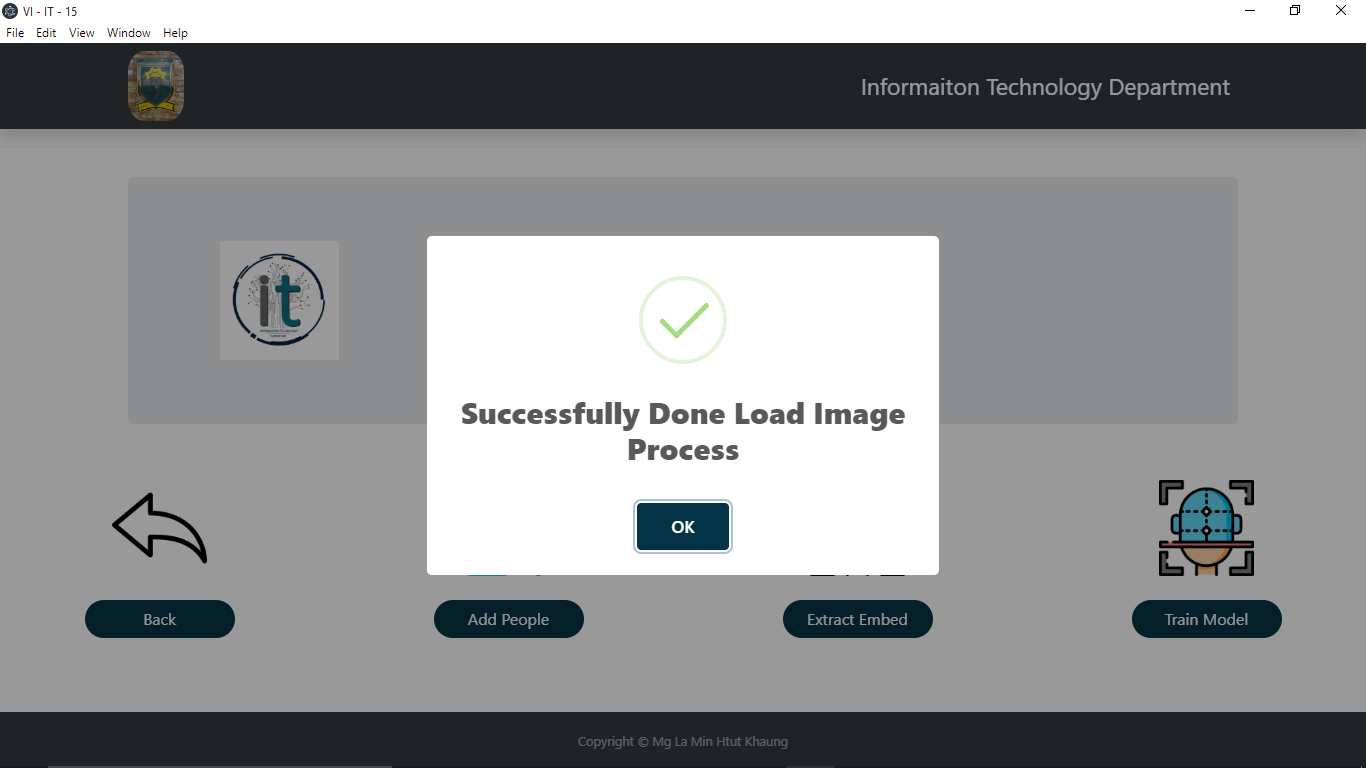


Figure 4.17. Modal Box of Complete Load Image Process

After the image uploading process complete, the system need to do extract embedding process and train model process again. Cause the new input images and label arrive in the project folder but not in the system file such as embeddings.pickle, le.pickle and recognize.pickle files. So, the extract embedding process and train model process must optimize again.

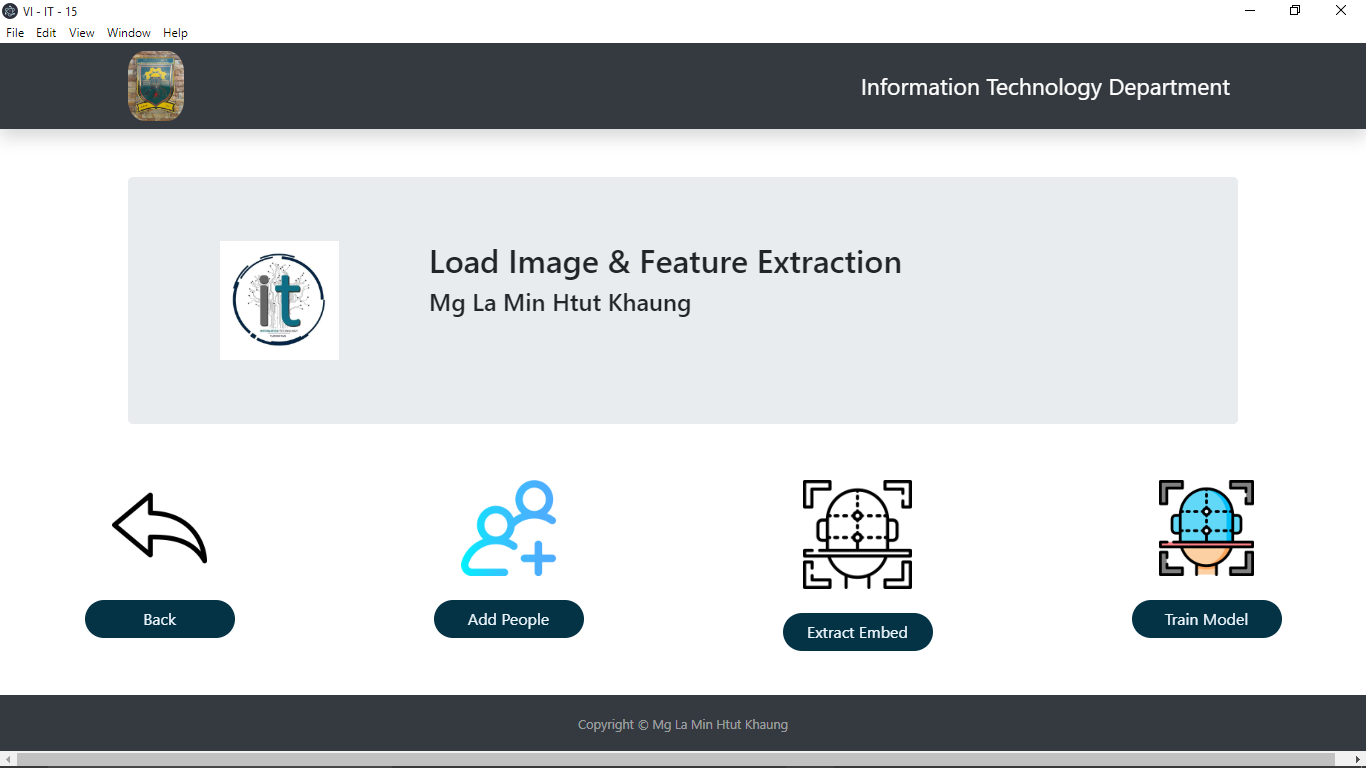


Figure 4.18. Extract Embedding Process for New Person

Figure 4.18 and 4.19 are the extract embedding process for new person and train model process for new person.

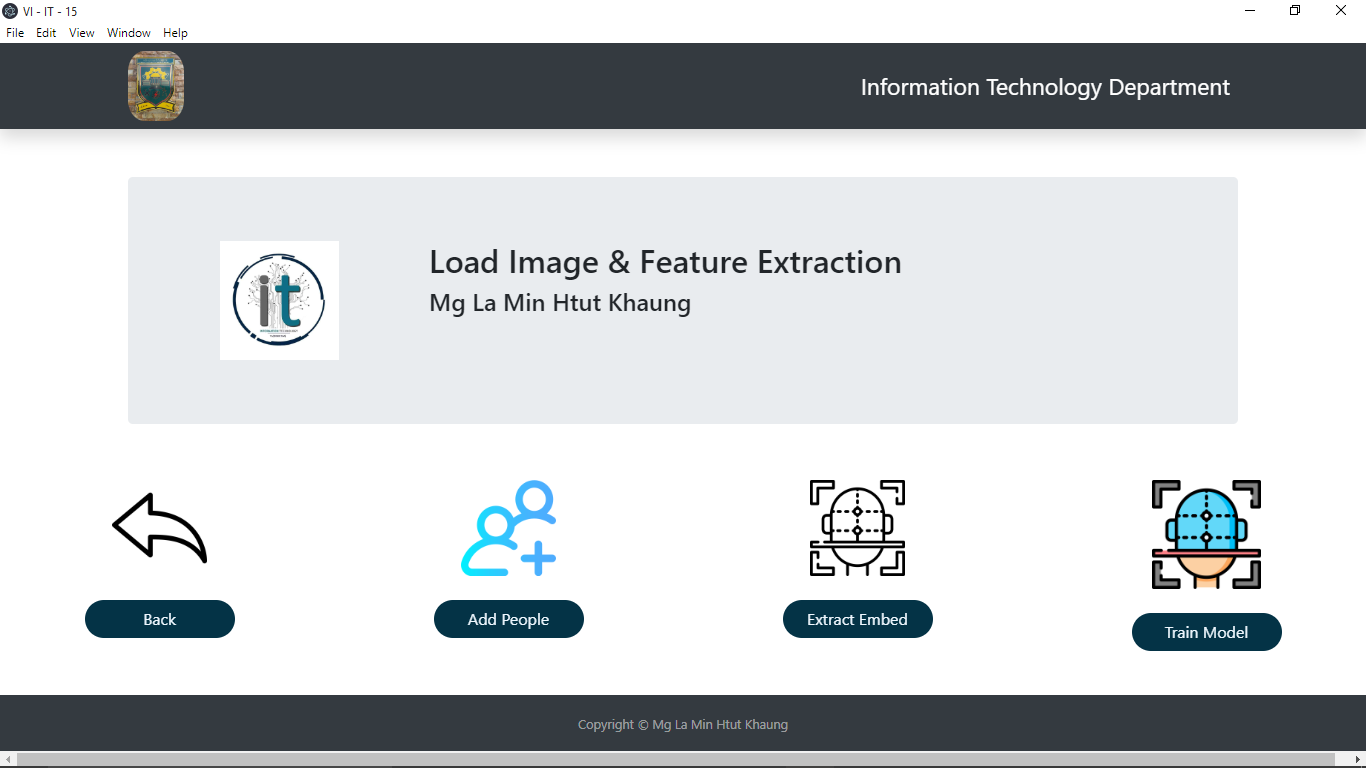


Figure 4.19. Train Model Process for New Person

After extract embedding process and train model process, the recognize image process can be done. Figure 4.20 show the recognize image process of new person.

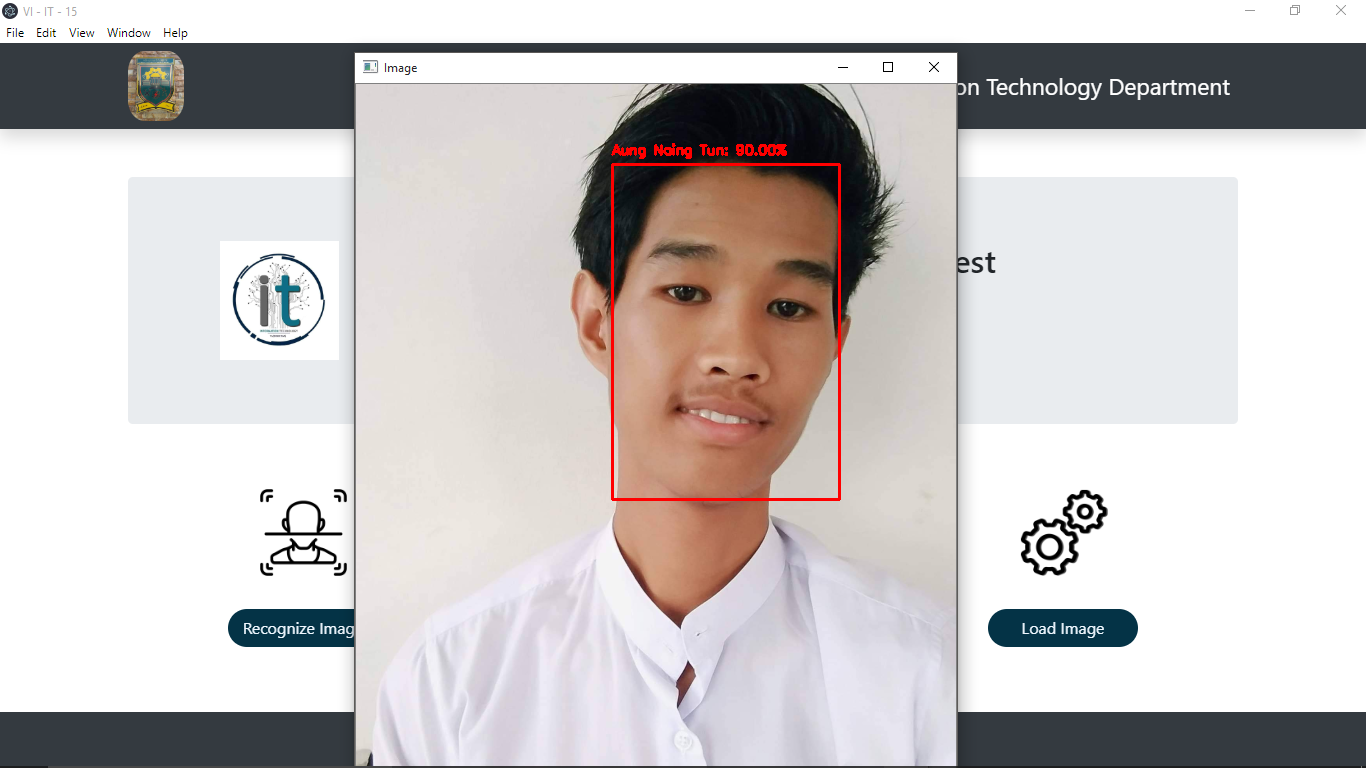


Figure 4.20. Recognize Image Process for New Person

**4.8. Summary**

The design and implementation of the system is presented in this chapter. The system is developed in real time. Load image process is developed to add another person easily. The system is combined with Electron (Node.js) Framework and Python Programming Language in order to use user interface easily.