Benha University

Shoubra Faculty of Engineering

Computer Systems Engineering

INKuisitor

Offline Signature Verification Web Application

by

Ahmed Sayed Mohamed

Ahmed Mohamed Atef

Mohamed Ibrahim Abdelghaffar

Hassan Essam Hassan

Hossam Ali ElRefai

Mohamed Magdy AbdelMoneim

Supervised by

Prof. Dr. AbdulWahab ElSammak

Dr. Shady Yehia

July 2021

Acknowledgment

We are overwhelmed in all humbleness to acknowledge the sincere efforts and valuable time given by our instructors Prof. Dr. AbdulWahab ElSammak and Dr. Shady Yehia. Besides, we would like to thank all the professors and lectures who helped us by giving us advice and providing ideas through our webinars we are thankful to all of them. Also, would like to express love and thanks to our families who have always been there by our sides.

Table of Contents

List of Figures

List of Tables

List of Equations

Table of Abbreviations

Abbreviation	Description
ANN	Artificial Neural Network
CNN	Convolution Neural Network
ReLU	Rectified Linear Units
SPA	Single Page Web Application
URL	Uniform Source Locator
API	Application Programming Interface
REST	Representational State Transfer
UI	User Interface
UX	User Experience
HTML	Hyper Text Markup language
JSON	JavaScript Object Notation
XML	Extensible Markup Language
ROC	Receiver Operating Characteristics
AUC	Area Under the Curve

Abstract

Biometrics (or biometric authentication) refers to the method of identification of humans by their characteristics or traits. Biometrics is employed in computing as a sort of identification and access control which is one among the foremost secure methods to ensure human’s privacy.

Handwriting is a process done unconsciously and is learnt over time, and there are some pen gestures that are invariant and are not simply altered when forgeries are made , and that’s why a person’s signature is used as a method of identity verification by forensic document examiners ,in banking and many other fields.

Throughout this thesis, an image-preprocessing and machine learning based offline signature verification web application is proposed.

The application uses a Convolution Neural Network (CNN) for distinguishing between forged and genuine signatures through a Siamese Neural Network. With an input of three signature images to create a new signature profile , store in database for later verification requests and a single input signature image used for verification of person’s identity.

Chapter 1: Introduction

1.1) Introduction

Authentication and verification are important features that every company or basically every establishment that has a database of clients and employees must have to deny any sort of forgery, scam and fraud that may result in very bad effects on the clients and the company.

Authentication has many various types such as: accepting proof of identity like an ID card to complete a purchasing process, depositing or withdrawing money as in bank systems, examining an object for a creator's or an author's signature to verify its ownership and asking the person questions that only him or her knows which they previously had provided. With today technological improvements, many sorts of scamming and manipulating have risen which have made the need for authentication and security measurements highly demanded to protect the integrity of the establishment and their clients or staff's data, as data have become the most valuable thing in our worlds time. The main reason why we choose signature as a verification and authentication method in our project is that it is one of the most efficient ways to authenticate intent and identity. A question one might ask, why signatures? how come a very old method of authentication that has been used over centuries be efficient? well, one simple answer for is that it does not require that many tools or equipment all you need is INK and a paper or as in nowadays technology a smart touch screen and an electronic pen. Another reasons of how it is efficient is that there are infinite number of possibilities of how a person signature style could be, which makes it very easy to verify that this is the person signature and extremely hard almost impossible for a scammer to copy that person signature if they have not seen it before. The main goal is to develop a system that authenticate and verifies a person signature and intent by accepting images of his signature and the signature that is desired to verify and provide a satisfying result showing whether the signature is forged is genuine with a friendly-easy to use web application.

INKuisitor: The concept behind the application name is that INKuisitor consists of two parts , the first is INK which refers to the method input is given to the system using regular ink pen and the second part is a derivation from the Latin noun "Inquisitor" which literally means "detective" which refers to the purpose of the system; to detect the forged from genuine signature to verify person's identity.

1.2) Motivations

- **Hot topic.**

Nowadays, there is an undeniable need for a system that would authenticate a person's identity efficiently and quickly, and that is something the world craves for, because as has been mentioned before data and information is everything, so not everyone should have access to other person's data unless permitted so.

- **Widely implemented.**

Person's identity authentications methods are almost used everywhere, like in hospitals when a patient uses his signature to confirm his agreement on an operation or a surgery, in banks when a client wants to withdraw or deposit money by confirming this is his bank account, in schools when a student write his signature on his exam paper and confirm that this paper belongs to him/her.

- **Language exclusiveness of similar systems.**

Many identity verification systems that exist were developed and implemented around one or two languages and requires big amount of data making it inefficient outside that region.

- **Lack of a ready-to-use web/mobile applications.**

Many similar applications that exist are desktop applications which requires a decent computer requirements and specifications making it not easy to use.

- **Lack of datasets examples.**

The problems that sometimes face classification models in training phase are the lack of training examples.

1.3) Contributions

- **Adaptable to Several Languages**

INKusitior model has been trained on several datasets of different languages which does not make the model exclusive to a particular language like later systems.

- **Web Application System**

The INKuisitor model is a user-friendly web application that requires only an internet connection for a computer requirement making it easy to use.

- **Compatible to Small Datasets for later model Modifications**

INKusitor model implements One-Shot learning technique that allowed us to develop a model that does not require big data to achieve satisfying results, avoiding the problem of general classification and improving the quality of the system because it is not continent to ask a user to give us 50 images of his/her signature for the system to work.

1.4) Problem Statements

The development of this project consists of three phases:

- **First phase**: Machine learning system

  The backbone of the project which is to develop a machine learning system that:
  - Preprocess the signature images.
  - extracts the feature from the signature.
  - compares the extracted features with another corresponding signature's features by implementing Siamese Neural Network[9].
  - produces an efficient result showing whether the signature is genuine or forged giving some statistics with it.

- **Second phase**: Front-end

  Develop a user-friendly React.JS web application that is :
  - Interactive, reactive , single page , fast and easy to use.
  - Allow user to upload new signatures' profiles.
  - Allow user to verify a signature of a particular profile, showing results (Genuine or Forged and the percentage of similarity) in a clear manner.

- **Third phase:** Backend

  Connects the first phase with the second phase:
  - Receives the name of the user, the signatures of the user then, create a profile of this user for later verification requests.
  - Take a verification request "a signature need to be verified" from the user, send it to the machine-learning phase to be verified and sends the verification result back to the front-end to be represented.

Chapter 2: Background and Related Work

2.1) Former Knowledge

To fully understand and interpret the concept and methodology used in this paper, several terms must be clarified:

1. Convolution Neural Network (CNN): Type of neural network that deals with image recognition and preprocessing that is designed to deal with pixel data.
2. Transfer learning: Machine learning method where a model developed for a task is reused as the starting point for a model on a second task.
3. Data preprocessing: Techniques that transform raw data into an understandable format to be fed to a model.
4. Contours: a curve joining all the continuous points (along the boundary), having same color or intensity.
5. Erosion: Image processing technique used to erode the boundaries of a foreground object.
6. Dilation: The opposite of Erosion, increase the white region in the image or size of foreground object increases.
7. Rectified linear units (ReLU): a non-linear activation function that is used in multi-layer neural networks or deep neural networks, the output value is the maximum value between zero and the input value.
8. Omniglot dataset: Designed for developing more human-like learning algorithms. It contains 1623 different handwritten characters from 50 different alphabets.
9. Siamese network: Networks that are used to find the similarity of the inputs by comparing their feature vectors.
10. Standard deviation: Measurement of the amount of variation or dispersion of a set of values.
11. Gradient decent: First-order iterative optimization algorithm for finding a local minimum of a differentiable function.
12. AlexNet: Convolutional neural network that is 8 layers deep, designed by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton. The model achieves a top-5 error of 15.3%, more than 10.8 percentage points lower than that of the runner up on in the ImageNet Large Scale Visual Recognition Challenge on September 30, 2012, which is a dataset of over 14 million images belonging to 1000 classes.

13. VGG-16: Convolutional neural network model proposed by K. Simonyan and A. Zisserman. The model achieves 92.7% top-5 test accuracy in ImageNet. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another.
14. LeNet: convolutional neural network structure proposed by. LeNet is small and easy to understand yet large enough to provide interesting results and it is considered the first architecture of the CNN.
15. Residual Neural Network (ResNet): An artificial neural network (ANN) of a kind that builds on constructs. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers.
16. Single Page Web Application (SPA) : A single-page application (SPA) is a web application or website that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of a web browser loading entire new pages. The goal is faster transitions that make the website feel more like a native app.
17. ReactJS: A free and open-source front-end JavaScript library for building user interfaces maintained by Facebook and a community of individual developers and companies
18. React Router : a standard library for routing in React. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL
19. API : An API is a set of definitions and protocols for building and integrating application software. It is sometimes referred to as a contract between an information provider and an information user—establishing the content required from the consumer (the call) and the content required by the producer (the response).
20. REST API : (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services.
21. Axios : library that helps making http requests to external resources , thus consuming APIs provided by the backend

2.2) Related Work

For signature verification, many systems may agree on some general steps such as: data acquisition, data preprocessing and training the model. But the real magic happens within these titles; in data acquisition, various datasets are used in different systems and various numbers of it too. In data preprocessing[3] which considered to be a fundamental part in every system, some systems may require image scaling, image noise removal and limit the image colors to white and black and using a different network to train the machine learning model.

Some papers were reviewed and helped us understand where we stand and what we need exactly to deploy this project.

[i] A project was conducted by continuing the work of a previous research via transfer learning[2]. A dataset's been created consisting of 3000 signatures with 30 unique users which were exposed to resizing, filtering, cleaning and gray scaling. The machine learning network used was Inception-v3 which is a convolutional neural network architecture from the Inception family that makes several improvements including using Label Smoothing, factorized 7 x 7 convolutions, and the use of an auxiliary classifier to propagate label information lower down the network, illustrated in figure 1. The result of this project gave a 97% accuracy and cross entropy of 0.1. Despite the high accuracy of this system, it was limited to one data set and a network that needed larger dataset and data preprocessing was not efficient enough.

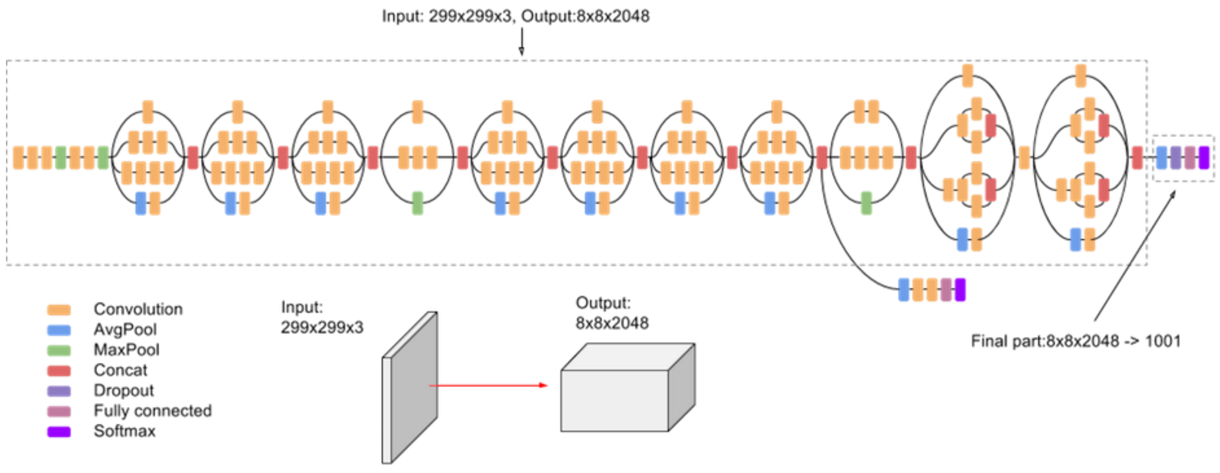


Figure 1. Inception-v3 CNN architecture

[ii]A research has been conducted in Canada by Toronto University designing an image recognition system using Siamese neural network. the standard model is a Siamese convolutional neural network with L layers each with N_l units, where $h_{1,l}$ represents the hidden vector in layer l for the first twin, and $h_{2,l}$ denotes the same for the second twin. They have exclusively used Rectified linear (ReLU)[7] units in the first L - 2 layers and sigmoidal units in the remaining layer, illustrated in figure 2.

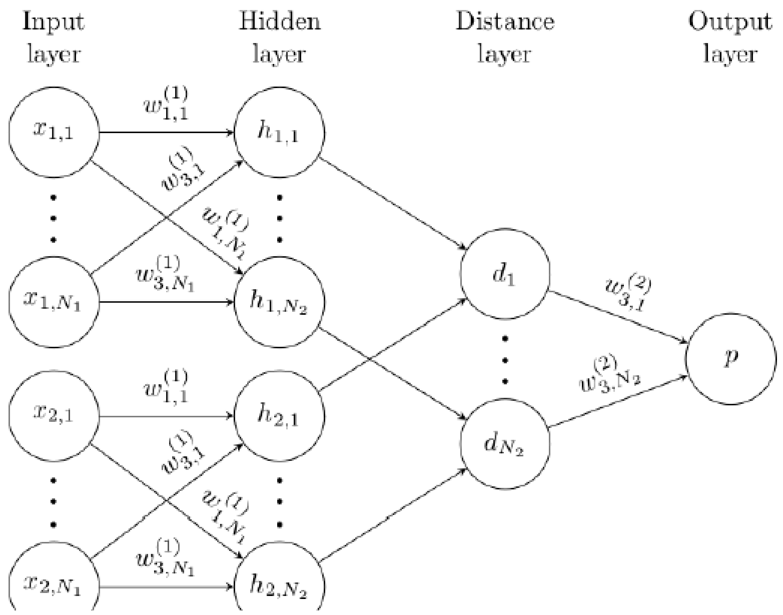


Figure 2. A simple 2 hidden layer Siamese network for binary classification with logistic prediction p.

the network which now conists of a series of convluation layers was fed with Omniglot dataset [8] in batches. They initialized the weights of the layers with zero-mean and standard deviation[10] of 10^{-2}, the learning rate could be decayed by 1 precent per epoch to reach the minima more easily. In the verification phase they have put together three different data set sizes with 30,000, 90,000, and 150,000 training examples by sampling random same and different pairs, this resulted in a test accuracy ranging from 90.61% to 93.42% depending on the data sizes, as the highest accuracy corresponds to the highest training example as shown in table 1.

Table 1. Result Statistics of applying the model on Omniglot dataset

Method	Test
30k training	
no distortions	90.61
affine distortions x8	91.90
90k training	
no distortions	91.54
affine distortions x8	93.15
150k training	
no distortions	91.63
affine distortions x8	**93.42**

[iii] Similar to research [ii] Siamese Neural Network is used for a writer independent offline signature verification system, consisting of a SigNet for the CNN, conducted by Autonomous University of Barcelona. For Data preprocessing, several techniques were applied; all images have been resized to a fixed size typically 155 * 220 using bilinear interpolation, Normalization has been applied by dividing each image’s pixel by its standard deviation. For the CNN architecture, the first convolutional layers filter the 155×220 input sig-nature image with 96 kernels of size 11×11 with a stride of1 pixels. The second convolutional layer takes as input the(response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size 5×5. The third and fourth convolutional layers are connected to one another without any intervention of pooling or normalization of layers as shown in figure 3.

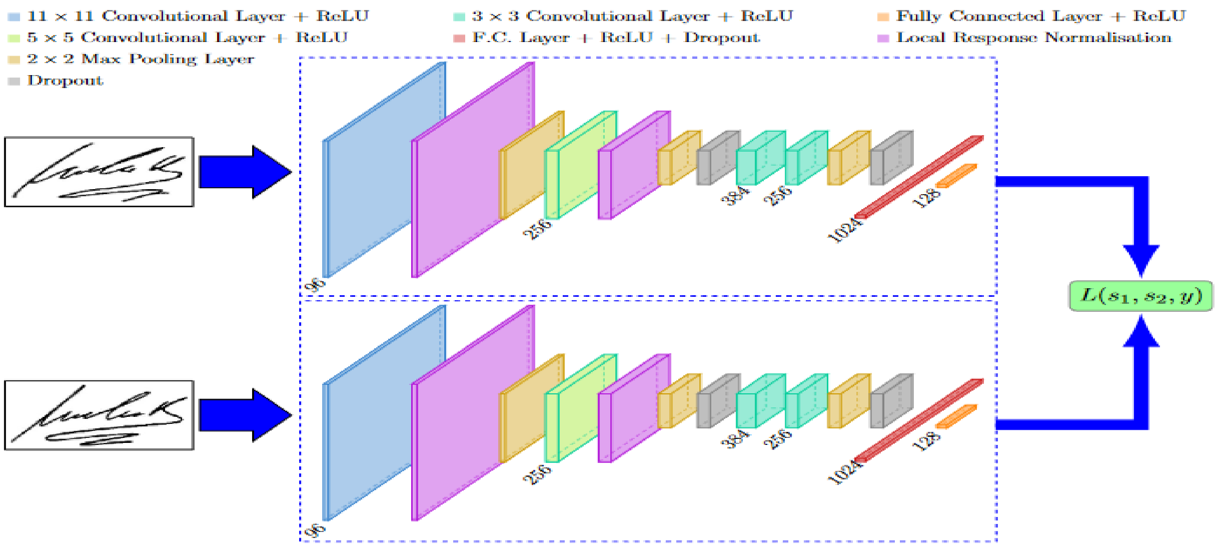
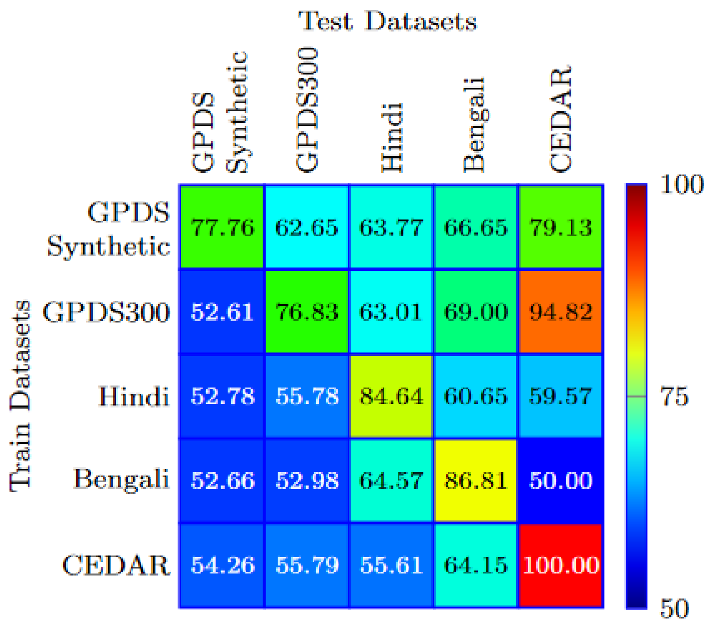

11 × 11 Convolutional Layer + ReLU
5 × 5 Convolutional Layer + ReLU
2 × 2 Max Pooling Layer
Dropout
3 × 3 Convolutional Layer + ReLU
F.C. Layer + ReLU + Dropout
Fully Connected Layer + ReLU
Local Response Normalisation
96
256
384
256
1024
128
$L(s_1, s_2, y)$

Figure 3. Architecture of SigNet CNN

The third layer has 384 kernels of size 3×3 connected to the (normalized, pooled, and dropout) output of the second convolutional layer. The fourth convolutional layer has 256 kernels of size 3×3. This leads to the neural network learning fewer lower-level features for smaller receptive fields and more features for higher level or more abstract features. The first fully connected layer has 1024 neurons, whereas the second fully connected layer has 128 neurons. This indicates that the highest learned feature vector from each side of SigNet has a dimension equal to 128. For the results, the accuracies obtained by SigNet on the cross dataset shown in table 2 , where the datasets used for training are indicated in rows and the datasets used for testing are shown through columns.



Train Datasets \ Test Datasets	GPDS Synthetic	GPDS300	Hindi	Bengali	CEDAR
GPDS Synthetic	77.76	62.65	63.77	66.65	79.13
GPDS300	52.61	76.83	63.01	69.00	94.82
Hindi	52.78	55.78	84.64	60.65	59.57
Bengali	52.66	52.98	64.57	86.81	50.00
CEDAR	54.26	55.79	55.61	64.15	100.00

Table 2. Accuracies obtained by SigNet with cross dataset settings

Chapter 3: Proposed System

3.1) Proposed System

As previously mentioned in Chapter 1 in problem statements, the development of this project consists of three phases: Machine learning phase, front-end phase and back-end phase. We will discuss each phase in details in this section starting with:

3.1.1) First phase: Machine learning Model

The challenges this phase faces can be specified in simple questions; How can we get different datasets different from each other that cover multiple languages? How can we preprocess this data to reduce any error in the training of the model? What methods do exist that can highlight and extract the signature from the image? Which Siamese architecture is suitable for this project and how can we implement it? To answer each question separately in details, we preview our methodology:

3.1.1.1) *Data Acquisition*

Several Datasets have been used:

a. ICDAR2011 (Dutch)

The ICDAR2011 dataset cover the Dutch language, has a total of 10 unique signatures, each unique signature branches to 24 forged signatures and 12 genuine signatures, a total of 360 signature, as shown in table 3.

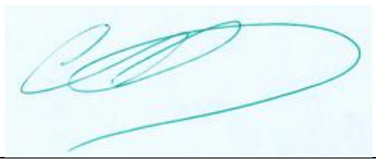
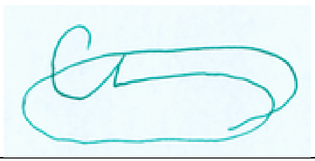
Table 3. ICDAR2011 dataset samples

User	Genuine Signature	Forged Signature
1	[illegible]	[illegible]
2	[illegible]	[illegible]

b. Kaggle, Cedar (English)

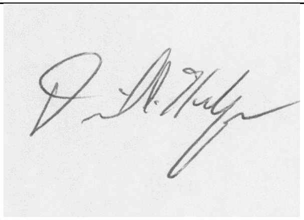
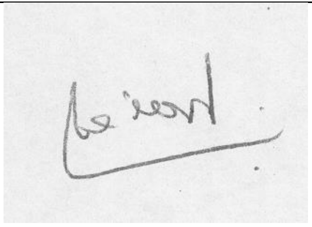
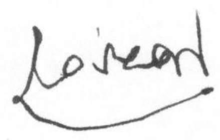
Two datasets cover the English language; **Kaggle** dataset has 30 unique signatures branching to 5 forged and 5 genuine signatures for each unique signature, a total of 300 signature, table 4 presents some samples from the dataset.

Table 4. Kaggle dataset samples

User	Genuine Signature	Forged Signature
1		
2		

Cedar dataset has 55 unique signatures branching to 24 forged signatures and 24 genuine signatures, a total of 2640 signature, table 5 presents some samples from the dataset.

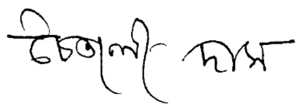
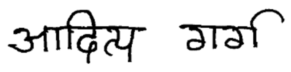
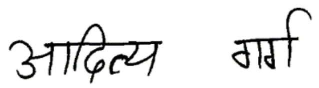
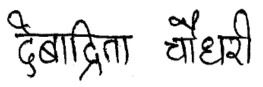
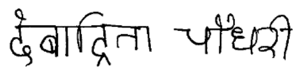
Table 5. Cedar dataset samples

User	Genuine Signature	Forged Signature
1		
2		

c. BHSig260 (Bengali, Hindi)

The BHSig260 dataset covers two languages: Bengali and Hindi, consisting of 6,240 genuine signatures and 7,800 forged signatures, with 100 unique signatures for the Bengali and 160 unique signatures for the Hindi (two Indian languages), as shown in table 6.

Table 6. BHSig260 dataset samples

User	Genuine Signature	Forged Signature
1		
2		
3		

3.1.1.2) *Data Preprocessing*

The acquired datasets should be prepared in such a way that reduces potential errors while using them. Several techniques have been used for this purpose including:

a. Noise Removal

The fine noises that are present in almost every image can be cleared using the gaussian blur technique. The goal of this technique is to eliminate any unwanted details in the signature image that might affect the result of the model. Gaussian blur technique affects pixels by applying a 9x9 filter to the pixel that want filtered, as the pixels near the center of this shape have more weight than the pixels further away as illustrated in figure 4.

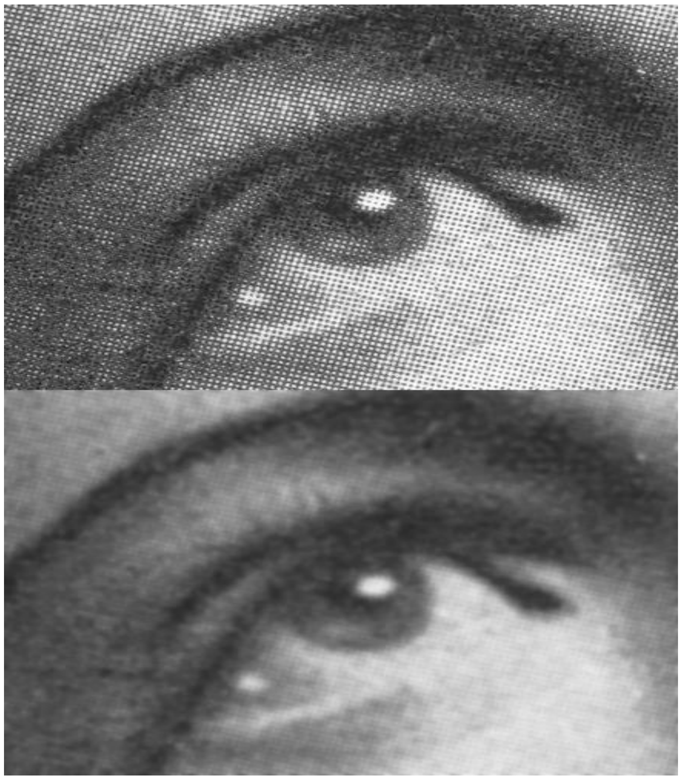


Figure 4. gaussian blur technique implementation before

b. Binarization

Most of the datasets we have used have three color channels, but since we are dealing with the format and the style of the signature, then we do not need to know its color as it will just make unnecessary complication in the implementation of our project. By turning the image into grayscale, the signature can be easily highlighted from the background for further preprocessing. This can be done by using Automatic OTSU algorithm, it automatically performs histogram shape-based image thresholding for the reduction of a gray level image to a binary image and considered to be one of the most successful global threshold methods if not the best, illustrated in figure 5.

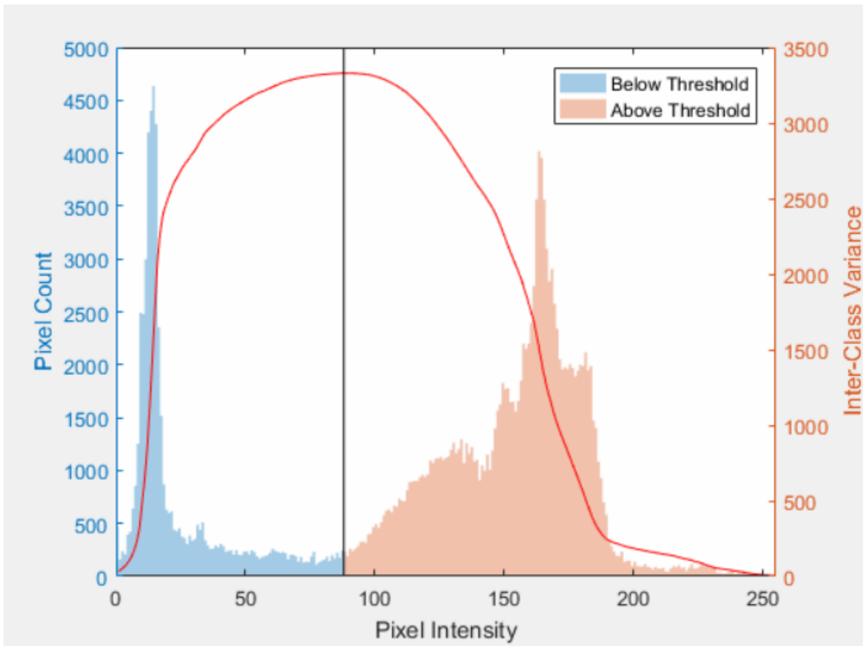


Figure 5. OTSU Algorithm Threshold

c. Signature Localization

The goal of this process is to locate the coordination of a rectangle that contains the signature. To do so , image biggest contour[4], which is typically the signature must be located, but since that in every language there are white spaces between the words, this can make finding the signature contour complicated. By applying closing technique, which is dilation[6] followed by erosion[5], we can eliminate the white spaces between the words and any noise of white dots inside the word by making the signature a complete block of white pixels, hence the contour can be located easily, figure 6 and figure 7.

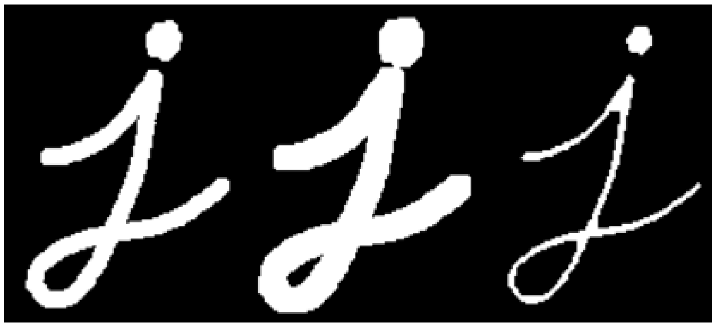


Figure 6.From left to right Original image, dilation and erosion

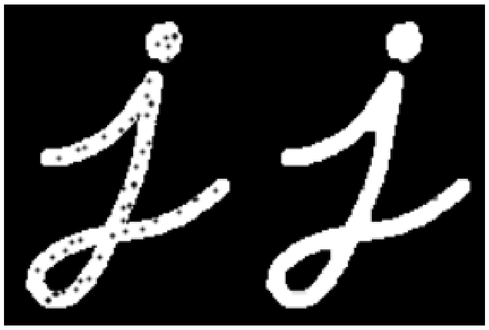


Figure 7. Closing technique

d. Signature Modification

After highlighting, locating the signature coordinates and details from the image and since we just need the signature and not the whole image, we crop the signature from its original image and copy it into a blank new image with dimension of 300x100 pixels, then we resize the extracted signature to the same new image dimensions, and for normalization purposes we divide the image pixels value by 255 to limit the computational process to be between zero and one.

3.1.1.3) *Neural Network Model*

As has been mentioned in previous chapters, Siamese neural network has been implemented. Siamese network is considered as an implementation of One-Shot learning technique, it consists of parallel convolution networks, could be two or more. The main reasons why we did not treat the problem showed in this paper as a normal classification, is because a normal classification problem requires large amount of training examples which are not present in the signature's datasets, if a new client or user wants to use our application, the model must be retrained from the biggening which requires huge computational power and takes a lot of time and it is clearly not practical. The Siamese network model solves this issue as it requires small samples to train the model for each class or each user, and if a new user signature is to be added to the model dataset, we do not need to train the model from the start but, we only require two or three images of his signature which would be treated as a reference and the signature image that wants to be tested, that's why it's called One-Shot learning. the model then evaluates the two input images by finding the similarity score between the two images between 0 and 1 clarifying whether the test image is genuine or forged based on the similarity score, showed in figure 8.

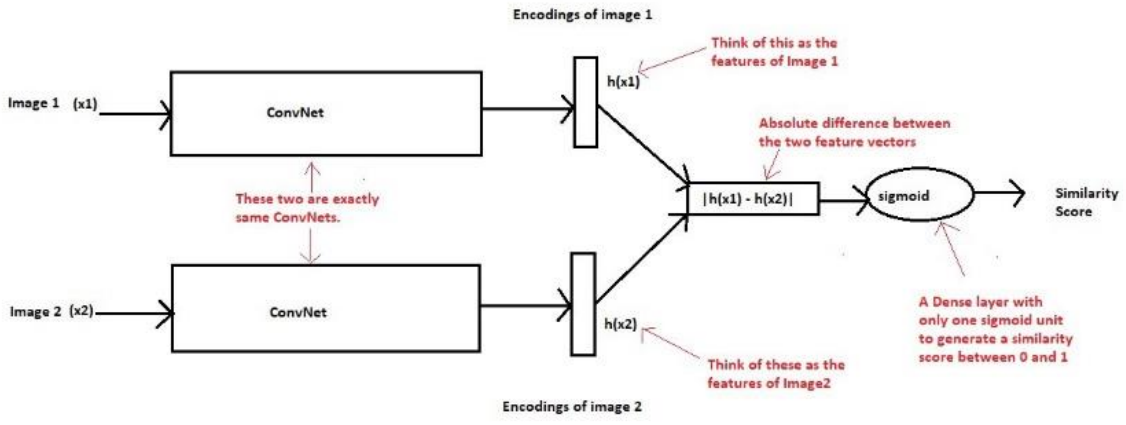

Encodings of image 1
Think of this as the features of Image 1
h(x1)
Image 1 (x1)
ConvNet
Absolute difference between the two feature vectors
These two are exactly same ConvNets.
|h(x1) - h(x2)|
sigmoid
Similarity Score
Image 2 (x2)
ConvNet
h(x2)
A Dense layer with only one sigmoid unit to generate a similarity score between 0 and 1
Think of these as the features of Image2
Encodings of image 2

Figure 8. One-Shot classification network.

1. **CNN**

The two parallel convolution neural networks are identical as they both share the same parameters. To talk about the architecture of the Siamese network, first we must talk in detail about its core which is the CNN. Dealing with images in straight forward way is not efficient or practical, because a very small image can have huge number of features for its small size. For example, a small image of 50 width, 50 height and 3 channels of pixels, has 7500 features which is an average number of features, but the image is too small to have such number of features. Another example, let us say we have an image of 1000x1000 pixels, giving 3,000,000 features (not mentioning the channels alone), if we to take these features as an input and make the first hidden layer has 1000 unit, then we have 3,000,000,000 element which is a huge number for a single image, if we process a dataset of similar images, we will want a very high-end computational power computers, hence comes the CNN. CNNs consist of a series of layers, each layer preforms two processes, convoluting and max-pooling; in convolution process, a certain number of a same size filters is applied to every pixel in the image resulting in reduced dimensions of the image with more depth (channels), in the max-pooling process, the depth of the image remains the same, but the dimensions of the image are remarkably reduced by applying a specific size max-filter to each pixel. In the following figure 9, we have a Siamese neural network consists of two parallel convolution networks, in the convolution process of the first layer, the image input dimensions are reduced from 105x105 to become 96x96 while increasing the depth from 1 to 64, in the max pool process, the dimensions are furtherly reduced to 48x48 while the depth kept the same 64. By applying the series of the layers in the CNN the features of the input image are reduced from 11,025 to 4096 features.

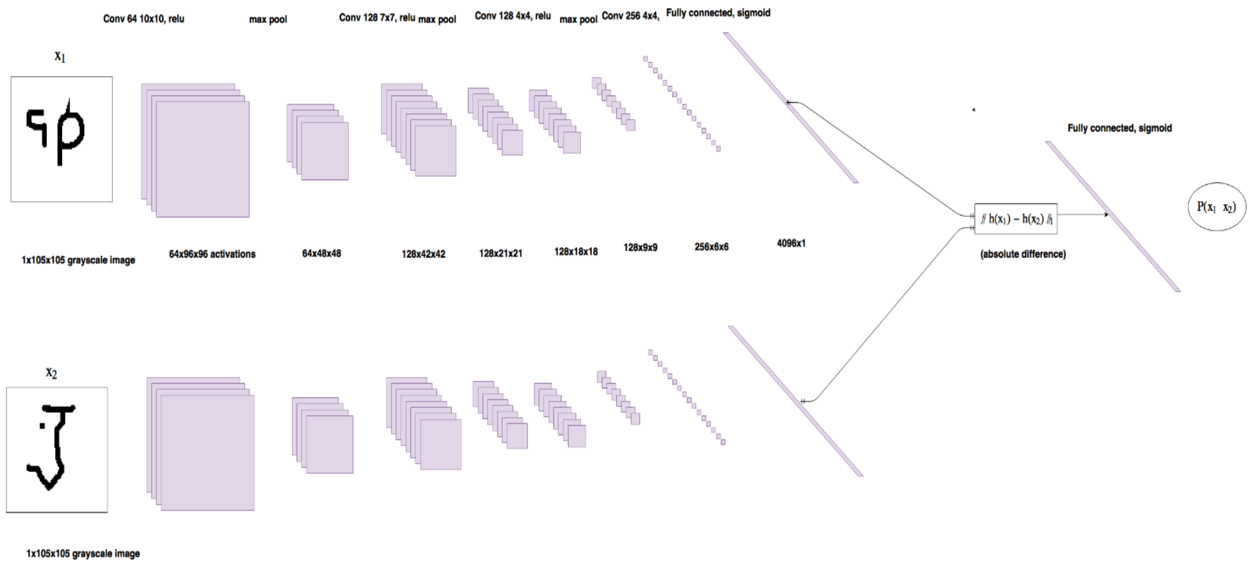


Figure 9. CNN structure in a Siamese network.

There are several types of CNN out there like AlexNet[12], VGG-16[13], Inception, LeNet[1414], and DenseNet. It is concluded through testing several types of CNNs varying in their internal structure filter sizes and layers, that a normal CNN took a lot of computational power and time and the VGG-16 did not give satisfying test results which is not suitable for our verification purposes, hence Dense 201 CNN type have been chosen for our Siamese model.

- **DenseNet**

In Dense net, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers making the network thinner and compact, this specific feature allows computational efficiency and memory efficiency as illustrated in the following figure 10, figure 11.

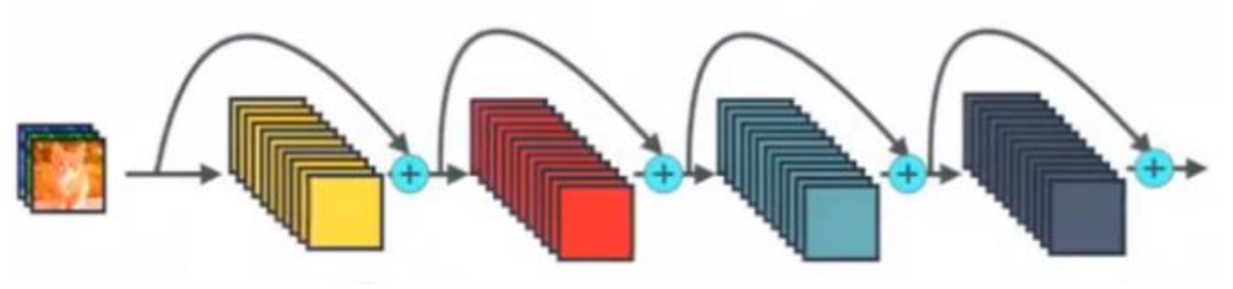


Figure 10.DenseNet element wise addition.

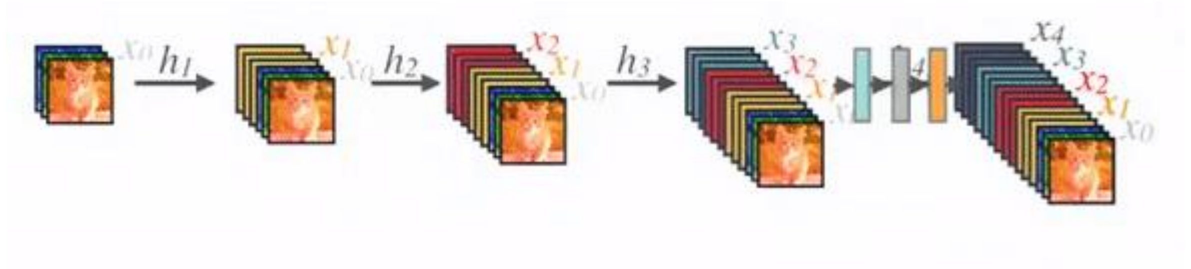


Figure 11. Layer concatenation in DenseNet.

For each composition layer, Pre-Activation Batch Norm (BN) and ReLU, then 3×3 Conv are done with output feature maps of k channels, say for example, to transform x0, x1, x3, x4 This is the idea from Pre-Activation ResNet, figure 12. But since every layer feeds the post layers through concatenating across the network, this could result in a very wide input fed to the deeper layers, this complexity and the huge size can be reduced by applying pre–Batch Norm, ReLU and 1x1 conv before Batch Norm (BN) and ReLU, then 3×3 Conv also shown in figure 12.

\

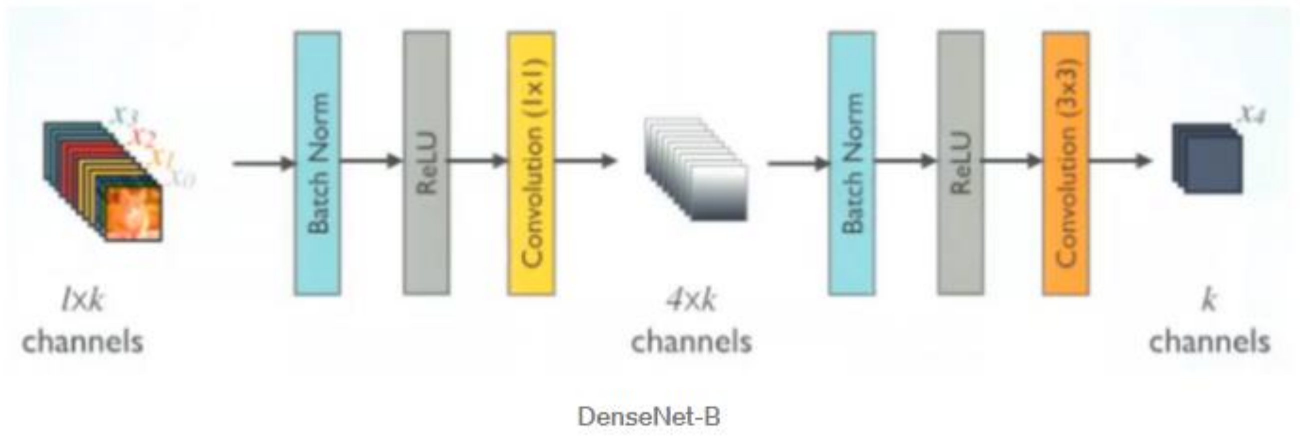


Figure 12. DenseNet Block architecture

2. **Similarity Function**

Now that we clarified the structure of the CNN in our Siamese structure, the output of these parallel CNNs is then fed into a similarity function. The similarity function calculates the difference between the output of the first CNN $h(x_1)$ and the output of the other parallel CNN $h(x_2)$ and then talking the absolute value to avoid any resulted negative values; $| h(x_1) - h(x_2) |$.

3. **Sigmoid Function**

As have been previously mentioned, Sigmoid function are often used in the output layer of the neural network. The Sigmoid function gives a probability output ranged between zero and 1 denoting the similarity between the two input images shown in figure 13.

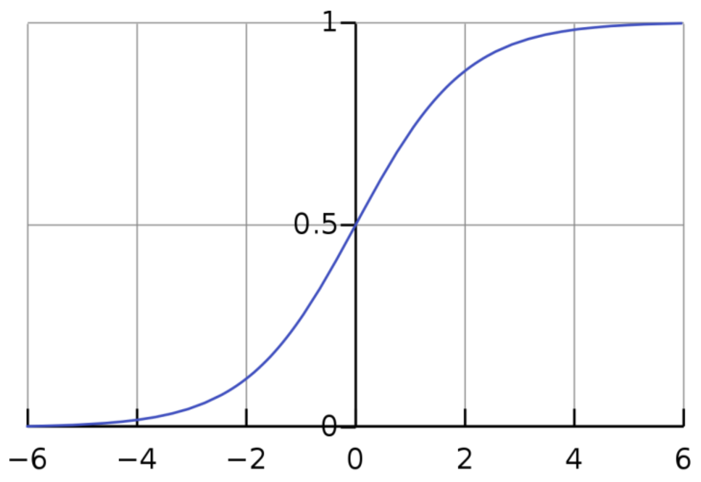


Figure 13. Sigmoid function properties

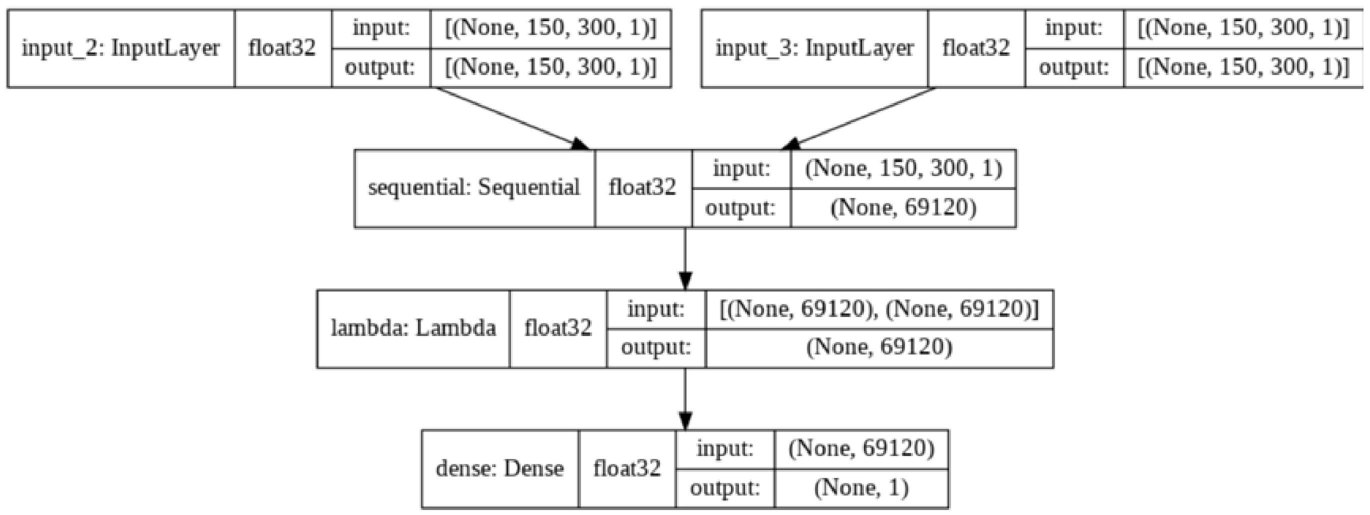


Figure 14. Siamese Network architecture

In summary, as figure 14 illustrates, the architecture of the system's Siamese network, two images are fed to the input layer with a size of 150x300x1 in pairs, each input runs through DenseNet CNN in which each layer the input goes through Batch Norm, ReLU, 3x3 convolution and average pooling, each input layer is concatenated with the previous layer output; allowing the error signal to be easily propagated to earlier layers directly and allows for more efficient computation and memory efficiency. Then in the similarity function, which is lambda, we calculate the difference between the features of the two images taking the absolute value of the output. The output is then fed to a sigmoid function to denote a result between 0 and 1 showing the similarity between the two images, and by setting a threshold value, we can verify whether the test image is forged or genuine.

3.1.1.4) First Phase: Machine learning model's Block Diagram

The above steps can be represented with the following block diagram in figure 15.

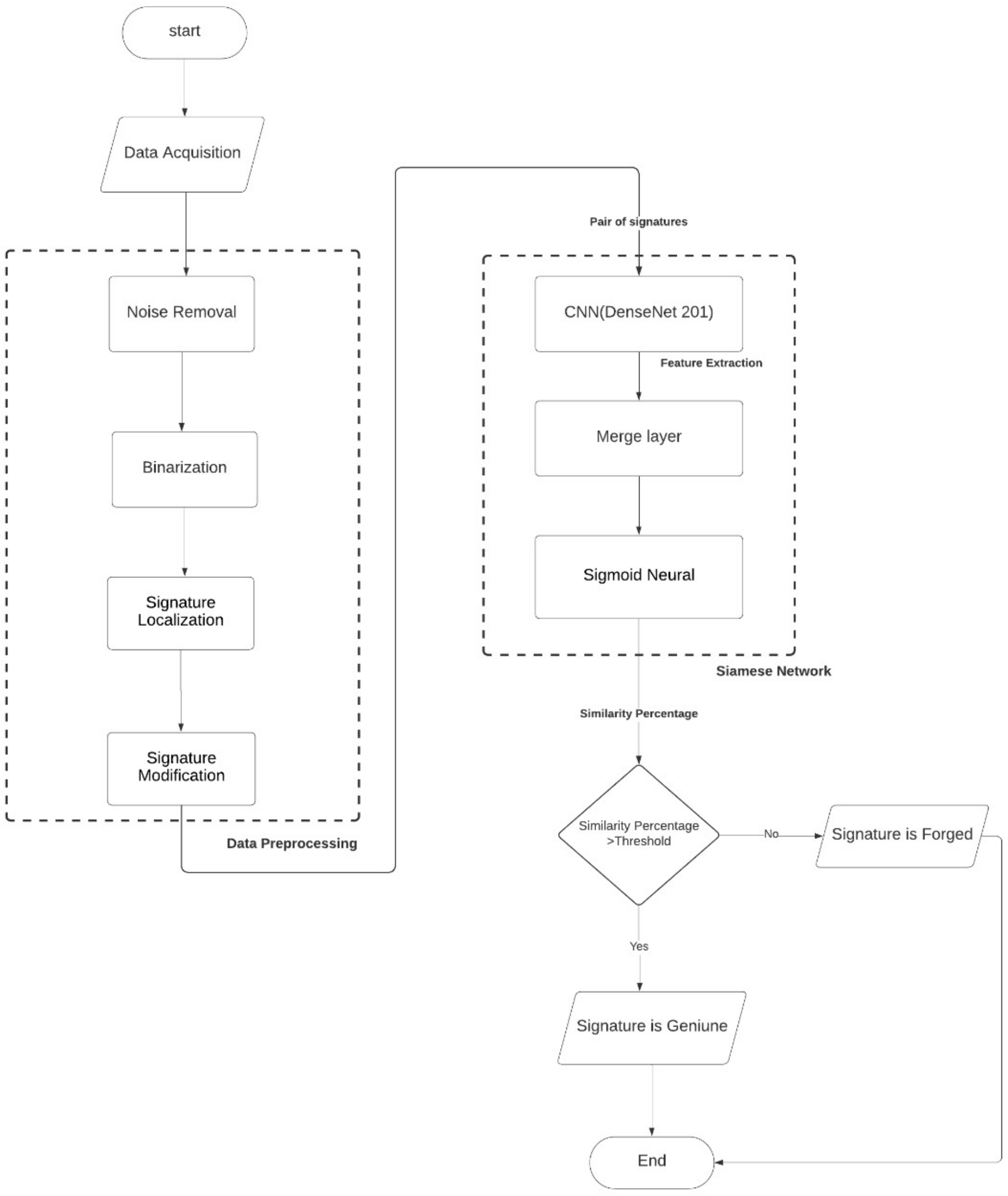


Figure 15. First Phase: Machine learning Model Block Diagram

3.1.2) Second phase: Front-end development

Goal of this phase is to develop a user-friendly web application that is interactive, reactive, single page, fast and easy to use, to provide a robust user experience.

3.2.1.1) *Front-End Framework*

ReactJS was chosen to develop the frontend part of the application. as it provides:

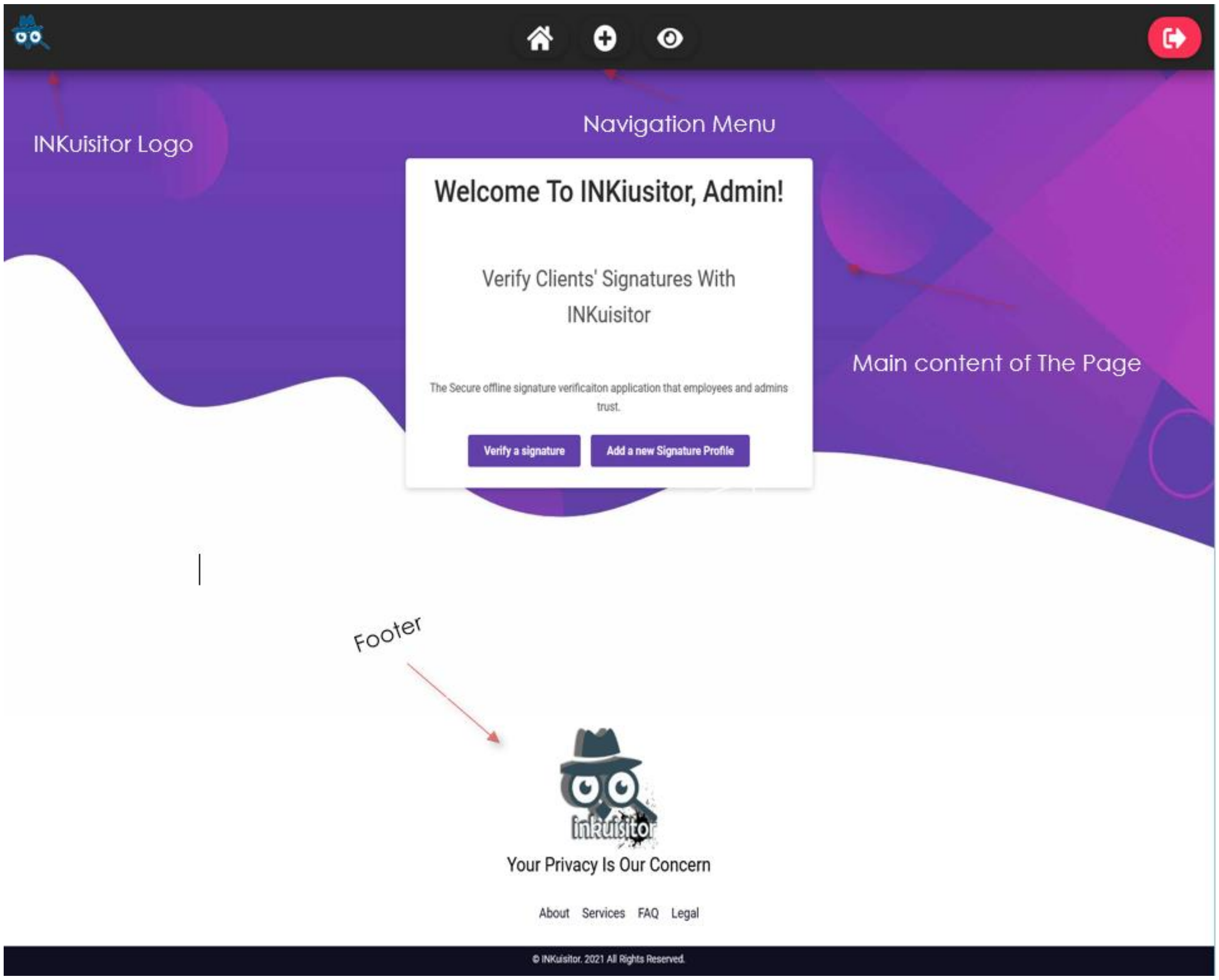
i. Easy Creation of dynamic web applications.
ii. Performance enhancements (faster and reactive web application).
iii. Reusable Components: Reusability + Separation of concerns.
iv. Reusability which helps avoid repetition.
v. Separation of concerns which gives the ability that many people can.
vi. Work on the same project and keeping our code manageable.
vii. Dedicated tools for easy debugging.
viii. Giving the ability for our project to be a mobile application as it is.
ix. developed in a native way (feature work).

3.2.1.2) *UI/UX Design*

To Provide a good user experience, UI was designed with a modern-compact look, making user interact with our web app in an ease. UI layout consists of 3 different parts:

1. A Navigation menu that has exactly 5 elements as illustrated in figure 16:
   a. INKuisitor Logo (on left of the nav menu), clickable and takes user to homepage.
   b. Three navigation options:
      i. Home button: onclick takes user to homepage.
      ii. Add button: onclick takes user to the add-new signature profile page.
      iii. Verify button: onclick takes user to the signature verification page.
   c. Logout button: logs user out account.
2. Main Functionality Part: content of a page is shown in a Card-based modern look, the main part which user interact with.

3. A Footer that contains the application logo, with some different navigation options



Welcome To INKiusitor, Admin!

Verify Clients' Signatures With INKuisitor

The Secure offline signature verificaiton application that employees and admins trust.

Verify a signature | Add a new Signature Profile

Your Privacy Is Our Concern

About Services FAQ Legal

© INKuisitor. 2021 All Rights Reserved.

Figure 16. Demonstration of INKuisitor web app UI

3.2.1.3) Application implementation

INKuisitor web app consists of three main pages, all have the same layout while the content of the card-based loadout changes depending on the page that is visited.

1. Homepage**:** simple welcoming screen that has two navigation options to the other parts of the web app.
2. Add New Signature Profile Page**:** content is a form, with two fields shown in figure 17.

   a. Client Name**:** inputting the new signature profile name, that will be stored in the Database.

   b. Signature Option**:** allowing user to choose between two different options in capturing the new signature profile.

1) Sign on-the-go **:** an HTML5 Canvas-based signature pad allowing user to sign three times inside the web app using a mouse (if opened on a PC or a laptop) or using a touch pen if opened on any touch screen device (mobile phone or a tablet with a touch pen) as shown in figure 17.
2) Upload signature**:** giving the user the option to upload 3 already-captured signature images from his storage.

   c. Add new Signature Profile (Submission Button)**:** a button that is clickable only when user has fulfilled the previous perquisites, when clicked sends the new signature profile data to the backend server as Form Data , to be stored in database for later verification requests, illustrated in figure 19.

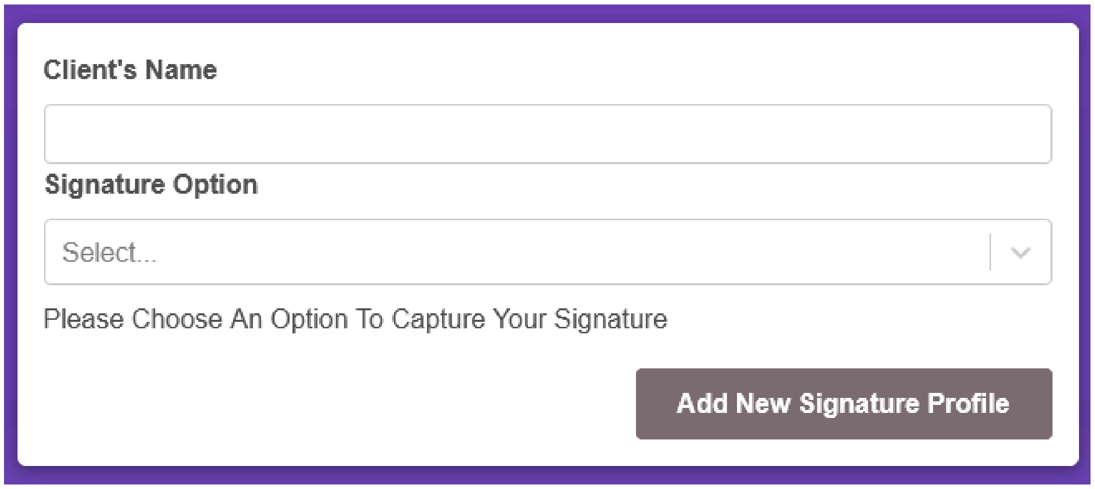
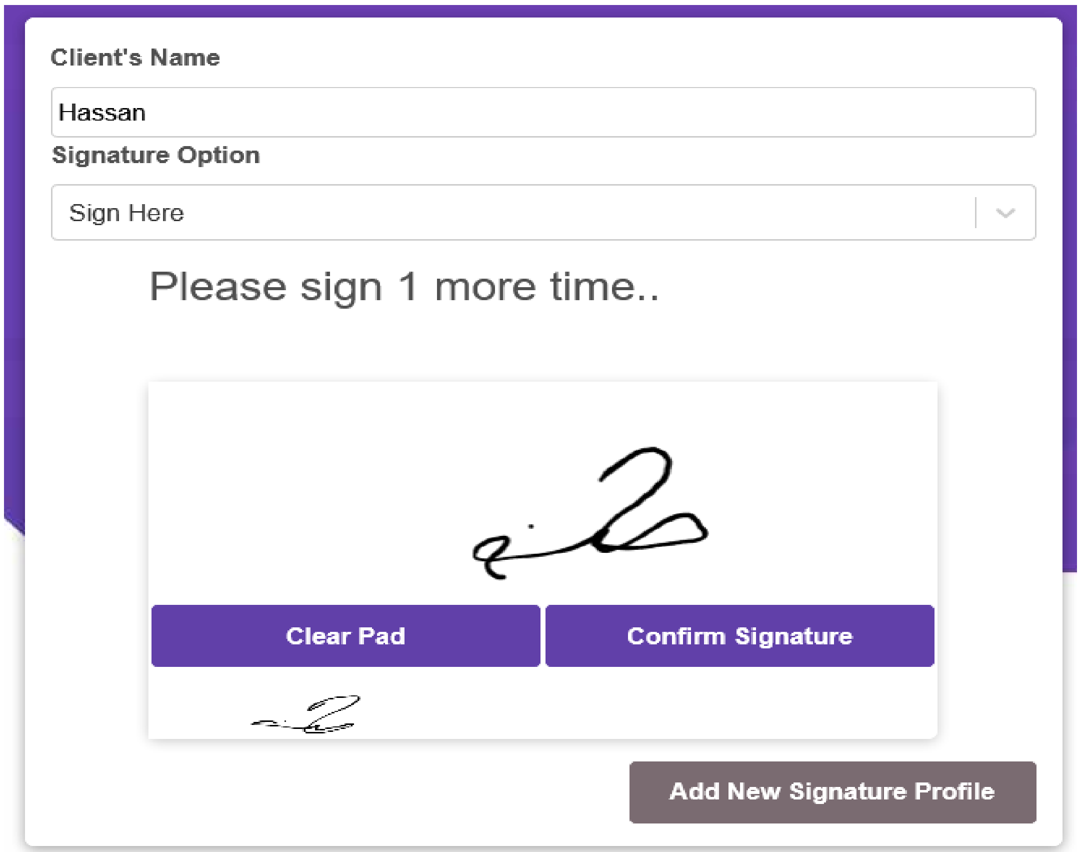

Client's Name
Signature Option
Select...
Please Choose An Option To Capture Your Signature
Add New Signature Profile

Figure 17. Demonstration of INKuisitor Add New Signature Page content


Client's Name

Hassan

Signature Option

Sign Here

Please sign 1 more time..

Figure 18.Demonstration of capturing user signature with signature pad.

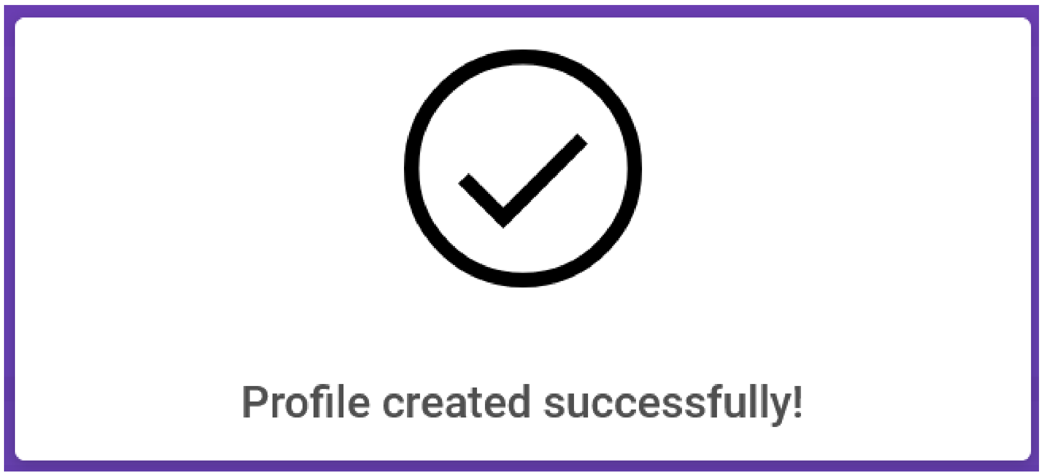

Profile created successfully!

Figure 19: Demonstration of successful creation of a new signature profile.

3. Verify Signature Page: content is a form, with two fields
   a. Choose a profile: a drop-down list, filled with data [clients' names] of profiles stored in the database

b. Signature Option: allowing user to choose between two different options in capturing the new signature profile;

1) Sign on-the-go : an HTML5 Canvas-based signature pad allowing user to sign only one time , inside the web app using a mouse (if opened on a PC or a laptop) or using a touch pen if opened on any touch screen device (mobile phone or a tablet with a touch pen).

2) Upload signature: giving the user the option to upload one, already-captured signature image from his storage, as shown in figure 21.

c. Verify (Submission Button): a button that is clickable only when user has fulfilled the previous perquisites shown in figure 20;

1) When clicked sends a verification request, pay loaded with the data to the backend server as Form Data to be verified using INKuisitor system.

2) When INKuisitor system successfully verifies the sent signature, the backend server sends a response that contains the result of the verification, the average percentage of similarity between the sent signature data and the three signatures.

3) This response which contains that percentage is then fed to a gauge that would be shown on the UI of the user, and if it exceeds an already set threshold a message indicating that the signature is genuine, otherwise forged as shown in figure 22 and figure 23.

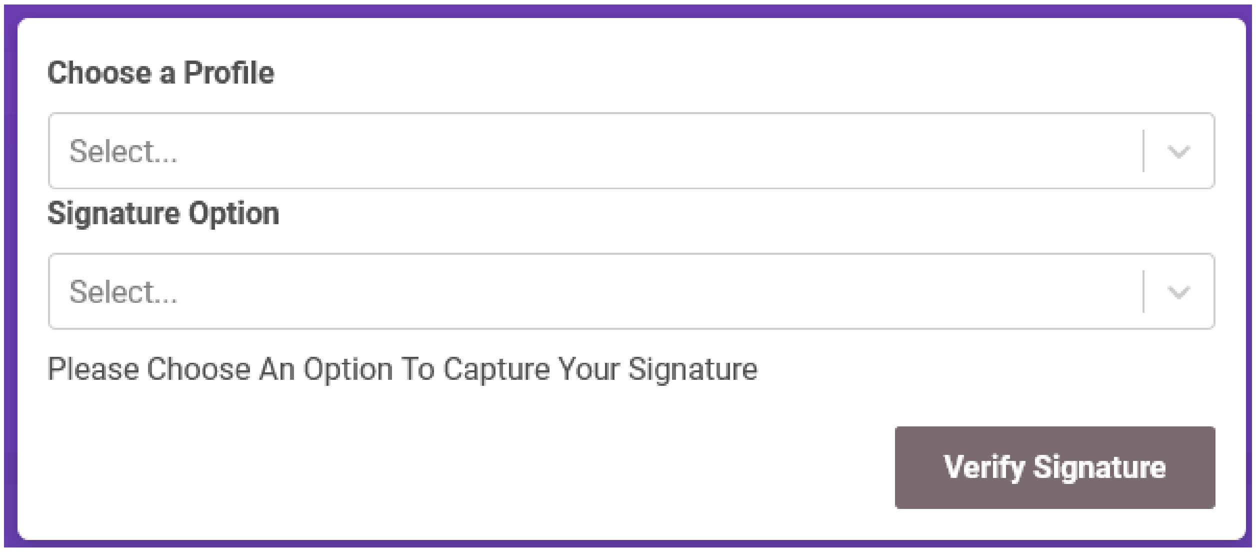
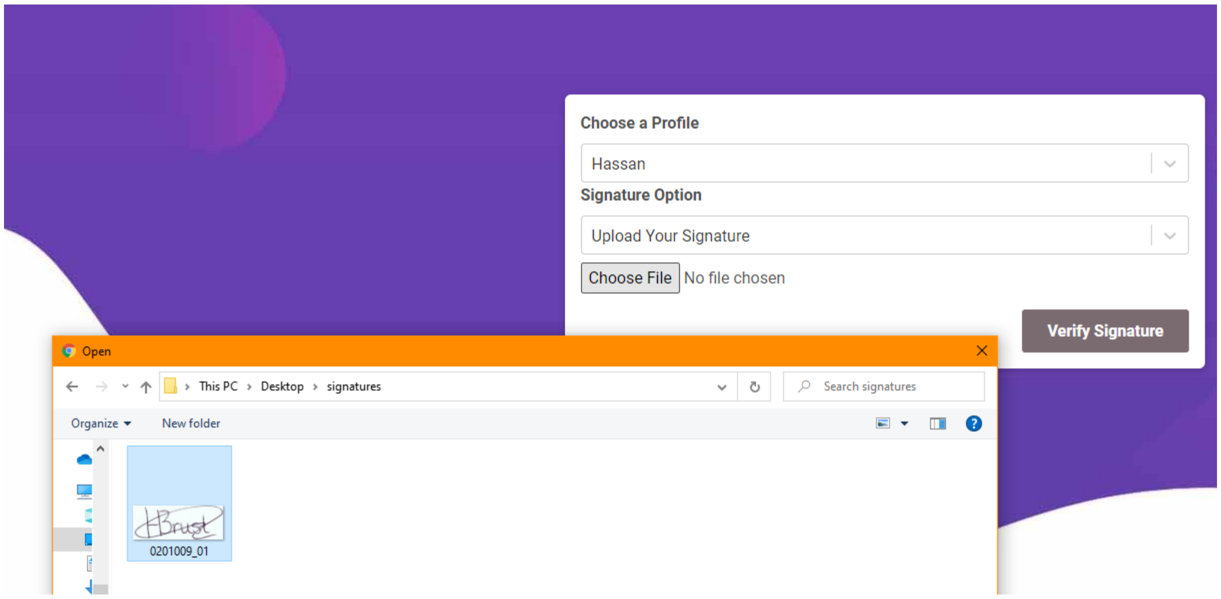

Please Choose An Option To Capture Your Signature

Figure 20. Demonstration of INKuisitor Verify Page content..



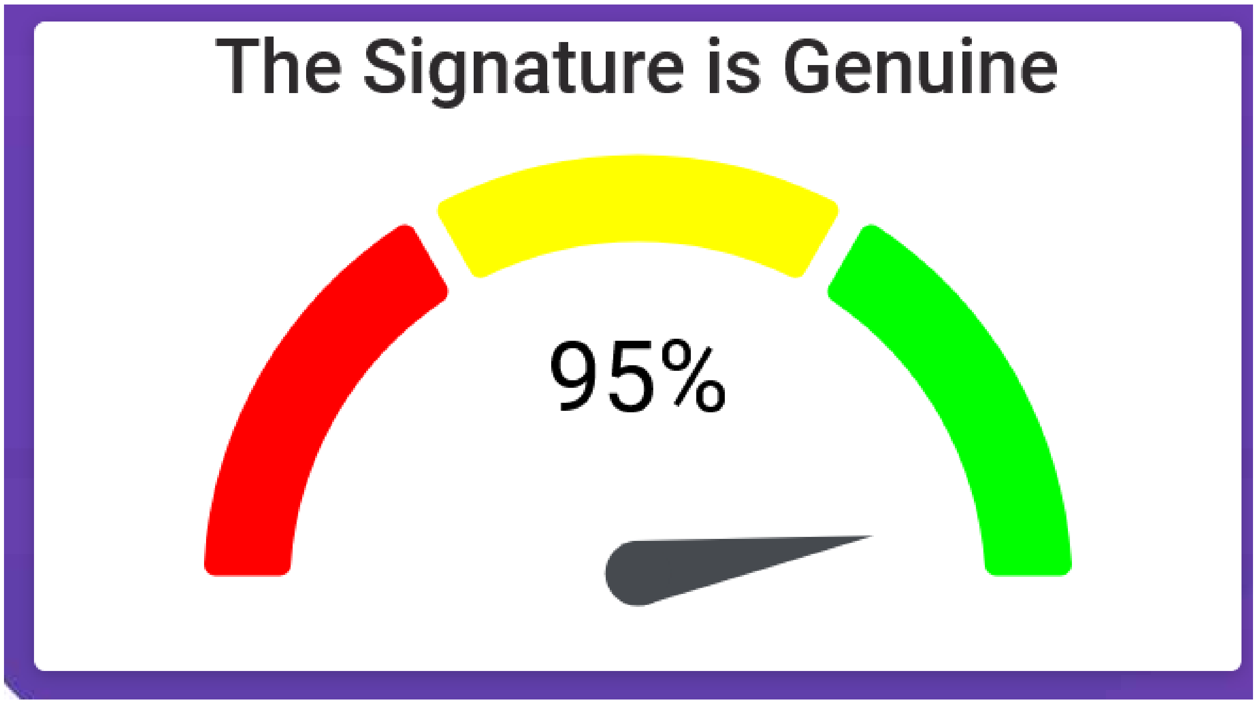
Choose a Profile

Hassan

Signature Option

Upload Your Signature

Figure 21. Demonstration of INKuisitor uploading signature from storage.



The Signature is Genuine

95%

Figure 22. Demonstration of INKuisitor Verify Result in a genuine signature case.

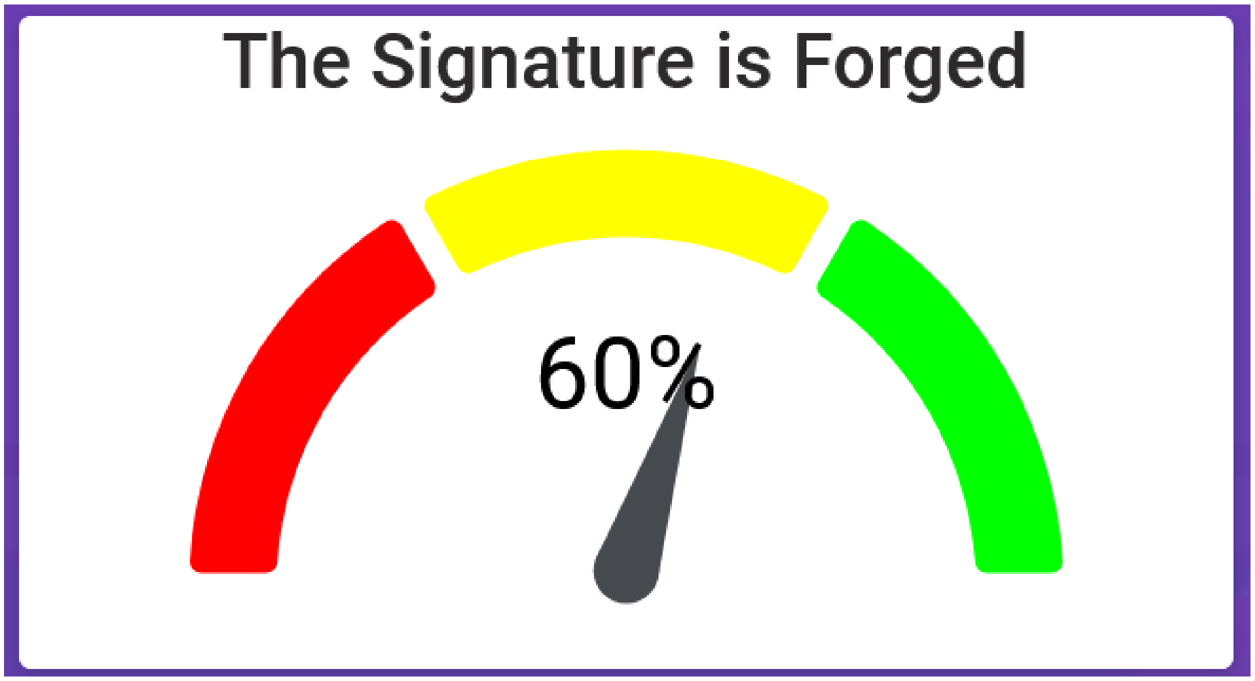


Figure 23. Demonstration of INKuisitor Verify Result in a forged signature case.

3.2.1.4) Application Routing

React Router was used to add the routing feature through INKuisitor application, which provides the feeling of the Single page application, when any of the navigation menu button is pressed.

3.2.1.5) Integration with backend

Connection to backend was established by consuming REST APIs provided by the backend using Axios.

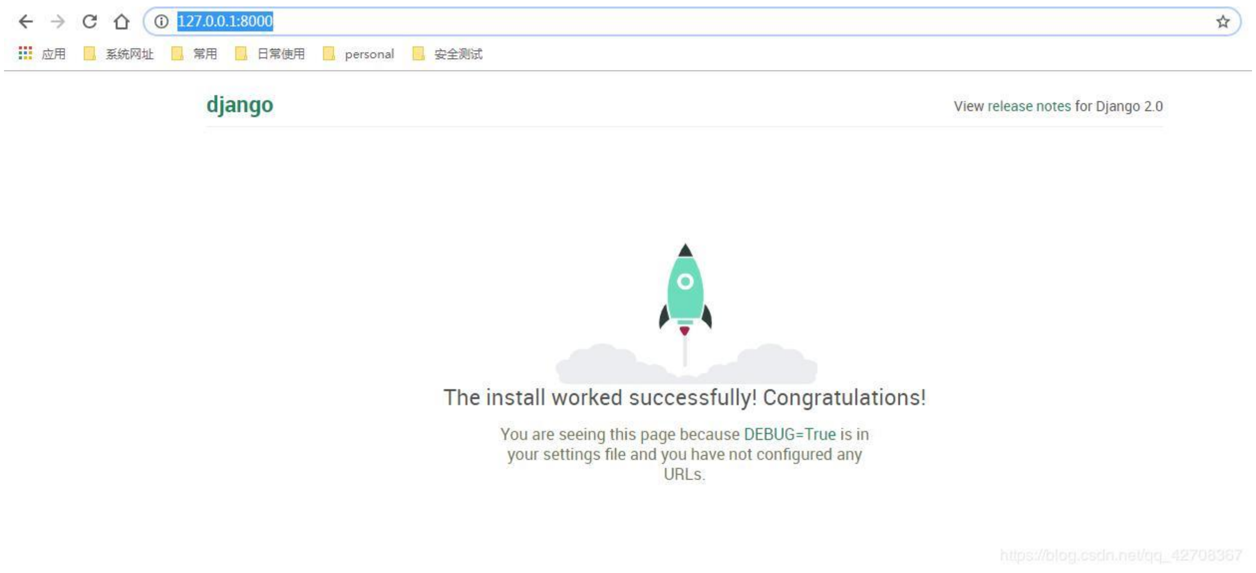
3.1.3) Third phase: Back-end phase

The challenges in this phase are How to build the backend server? How to build the database of the project? How to make rest APIs? And finally, how to integrate both the machine-learning model and the frontend with the backend server?

By doing the following:

3.1.3.1) Creating Django project

We create our project and name it INKusitor. After that, we start our app calling it "clerk" which contains the database model, admin site that register the database model to be showed in Django server and contain views and serializers that handles the integration between the machine-learning model and the frontend. Then we launched the Django server. The server is localhost "127.0.0.1:8000", as shown in figure 24.



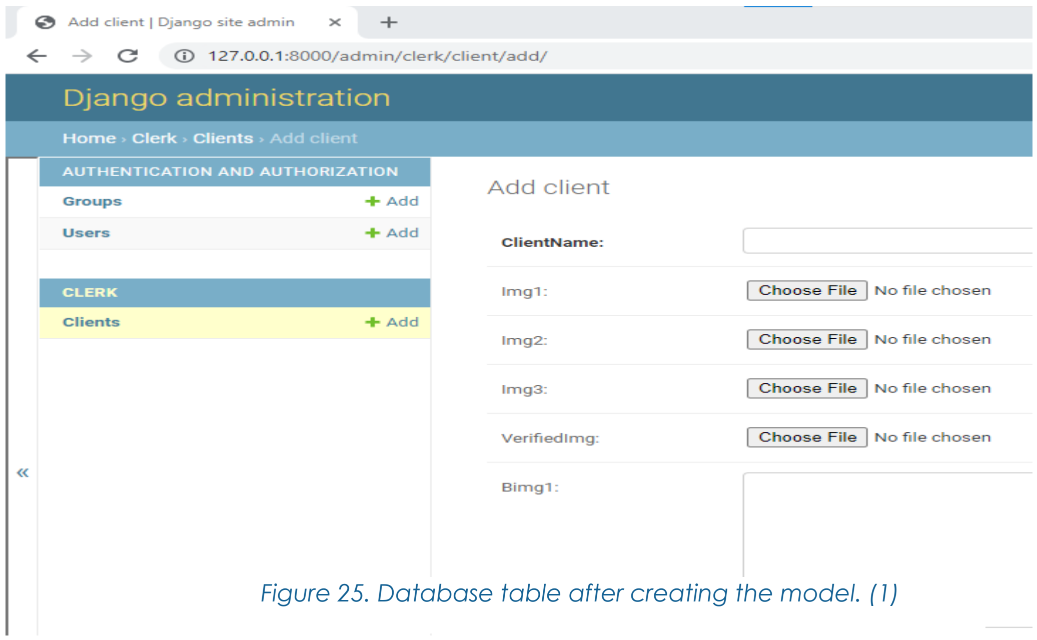
The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Figure 24. Demonstration of the server running after installation

3.1.3.2) Building Database

We build our model to create the database table in Django server as figure 25 shows:


Add client | Django site admin
127.0.0.1:8000/admin/clerk/client/add/
Django administration
Home › Clerk › Clients › Add client
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
CLERK
Clients + Add
Add client
«

ClientName:	
Img1:	Choose File No file chosen
Img2:	Choose File No file chosen
Img3:	Choose File No file chosen
VerifiedImg:	Choose File No file chosen
Bimg1:	

Figure 25. Database table after creating the model. (1)

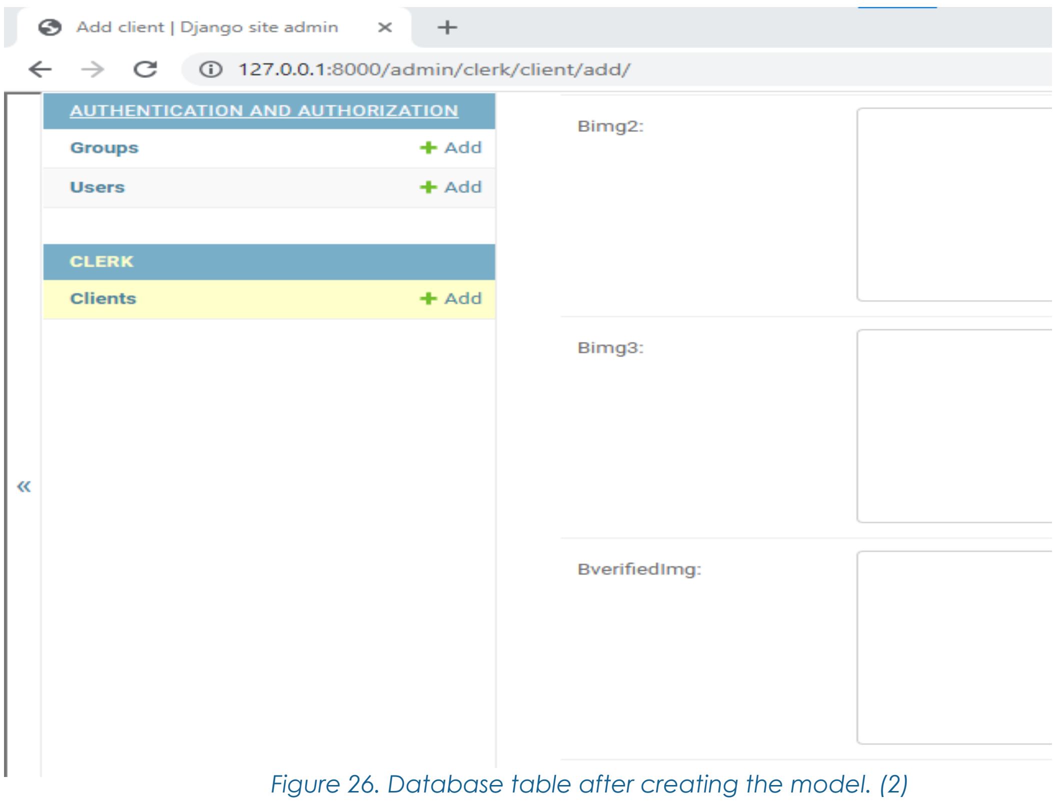

Add client | Django site admin
127.0.0.1:8000/admin/clerk/client/add/
AUTHENTICATION AND AUTHORIZATION
Groups
+ Add
Users
+ Add
CLERK
Clients
+ Add
Bimg2:
Bimg3:
BverifiedImg:

Figure 26. Database table after creating the model. (2)

- ClientName field: character field that stores the name of the client who need to make the verification process.
- Img1, img2, img3 fields: image fields that store the original signatures of the client.
- VerifiedImg field: image field that takes the uploaded image need to be verified to check whether the signature is forged or genuine.
- Bmg1, Bimg2, Bimg3 fields: like img1, img2, img3 fields but they store images of type "Base64" not PNG images.
- BverifiedImg field: like verifiedImg field but it checks the verification of "Base64" image not PNG image. As all shown in figure 26.

3.1.3.3) Creating Serializer

Serializers allow complex data such as querysets and model instances to be converted to native Python datatypes that can then be easily rendered into JSON, XML or other content types. Serializers also provide deserialization, allowing parsed data to be converted back into complex types, after first validating the incoming data. The serializers in REST framework work very similarly to Django's Form and ModelForm classes. We provide a Serializer class which gives you a powerful, generic way to control the output of your responses, as well as a ModelSerializer class which provides a useful shortcut for creating serializers that deal with model instances and querysets.

3.1.3.4) Views

Our views are the functions that deal with the requests coming from the frontend. These functions take those requests, process on them the required functions then send the response back to the frontend.

Our views are:

3.1.3.4.1) clientdetails_view

It takes the request from the frontend then return names of all clients stored in the database to be displayed.

3.1.3.4.2) verify_view

It takes the request from the frontend which is the signature need to be verified "whether the image is PNG or BASE64". Then it passes the image to the machine-learning model which process the image, make processes on it and produce the percentage of similarity of this signature with the original signatures of this client that is stored in the database. This response is returned to the frontend "the percentage of similarity" to decide within a pre-defined threshold whether this signature is genuine or forged.

3.1.3.4.3) createprofile_view

This view takes the request from the frontend which containing the name of the client and three images "PNG or BASE64".Then it creates new profile in the database with the name of the client and his three original signatures.

3.1.3.5) Second Phase and Third Phase Block Diagram

By implementing the steps illustrated in the second phase and third phase, the following block diagram in figure 27 illustrates the process of creating a new signature profile:

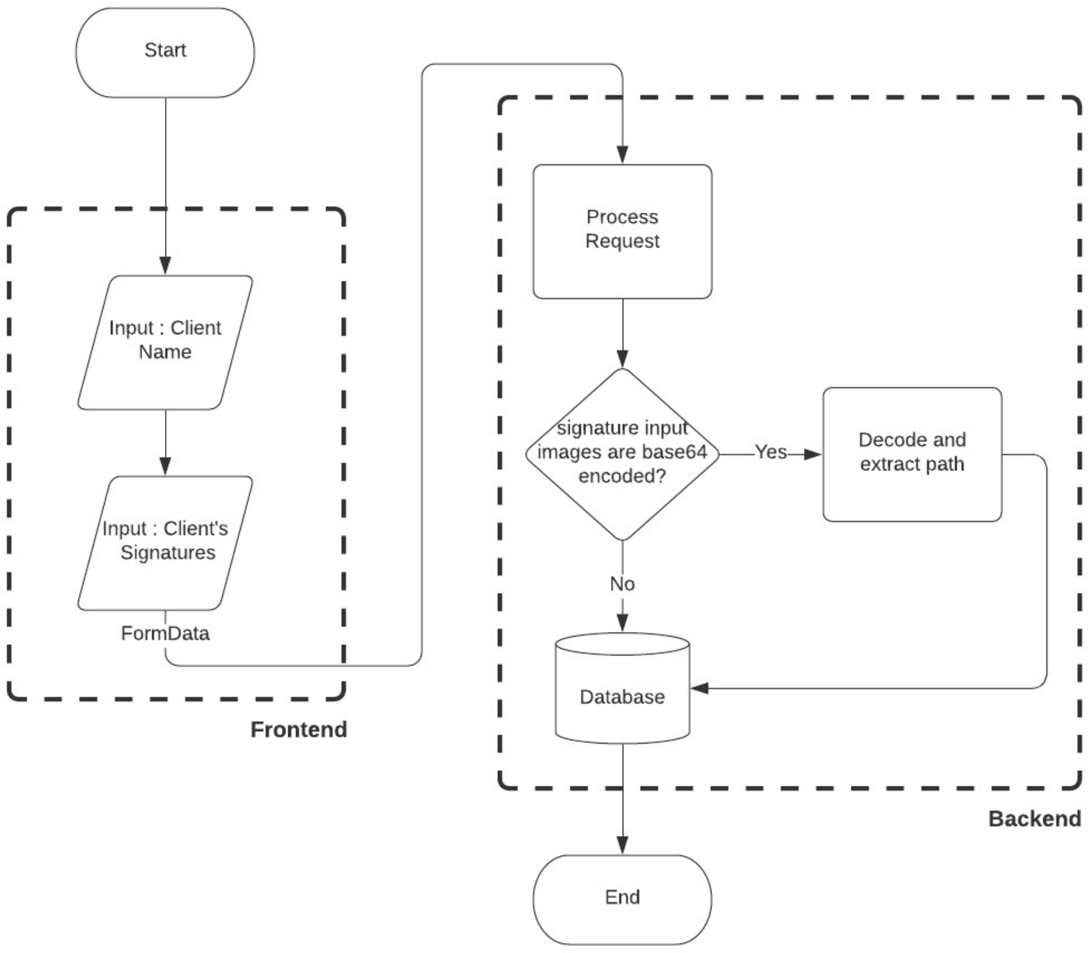


Figure 27. Creating a New Signature Profile Block Diagram

3.2) Proposed System block diagram

The previous phases includes the system's methodology of the INKuisitor model which can all be illustrated in the following block diagram in figure 28:

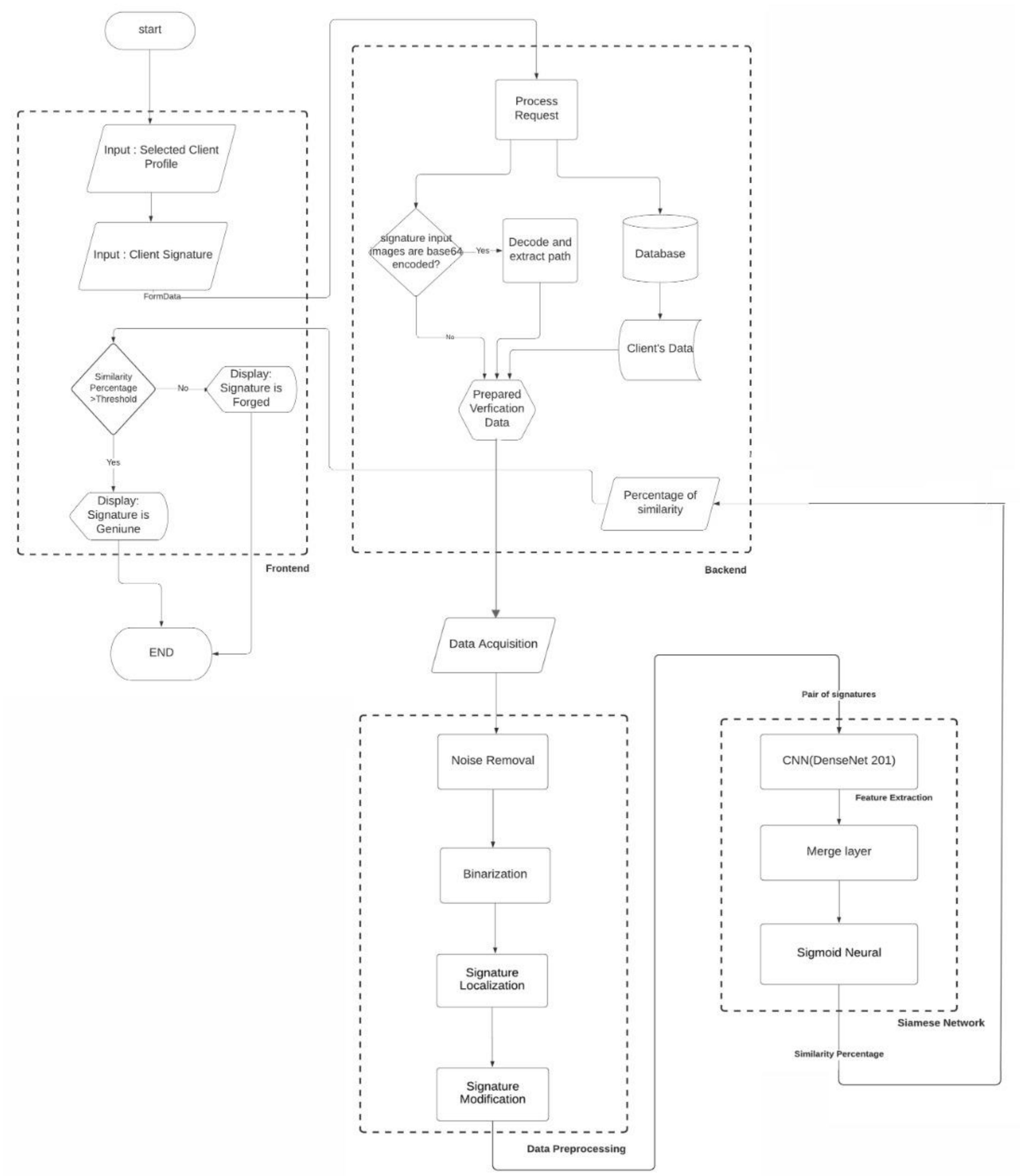


Figure 28. Signature Verification Block Diagram

Chapter 4: Experimental Results

4.1) Experiments

Before feeding the model with the previously mentioned datasets for training and testing purposes, they must be made into pairs of images to pass them to model for each signature (user), first Genuine-Genuine pairs are made without repetition and are given label one (1), second Genuine-Forged pairs are made by make each genuine signature image with all forged signature image to make pair and are given label zero (0), then fed to the model.

4.1.1) Image preprocessing

As have been mentioned, data preprocessing consists of several steps, in the following figure 29 showing the original image that is fed to the model for preprocessing before entering the neural network:

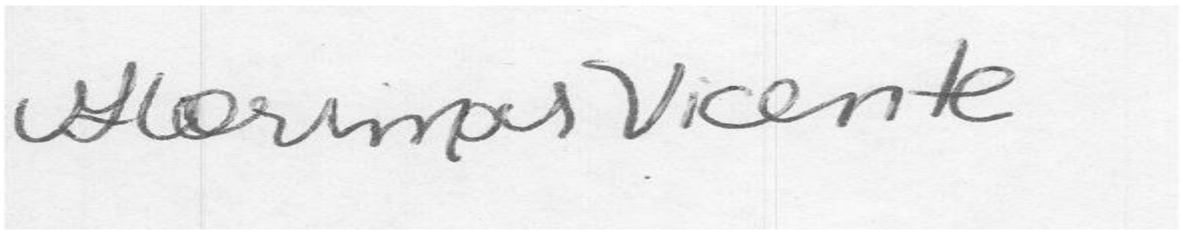


Figure 29. Original image before preprocessing

- **Step 1 : Noise removal**

By applying gaussian blurring technique to remove the random noises from the image as shown in figure 30.

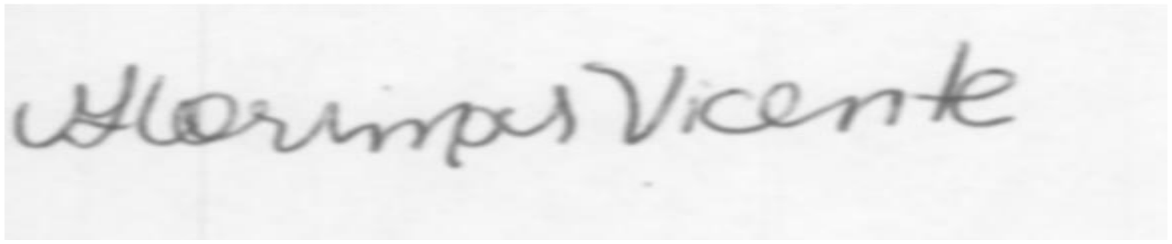


Figure 30. Original image after blurring

- **Step 2: Binarization**

The image is to be turned into black and white colors; black for the background and white for the signature , as shown in figure 31.

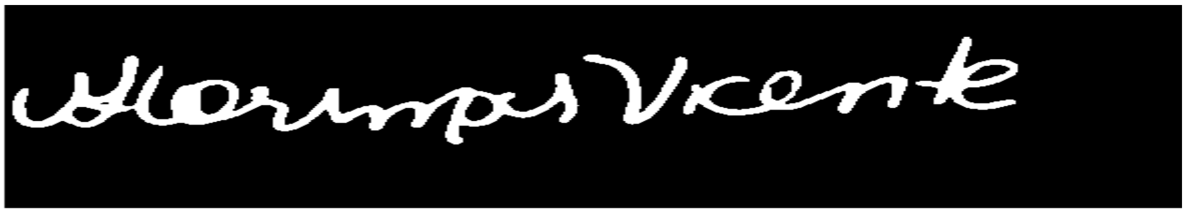


Figure 31. Original image after noise removal and binarization

- **Step 3: Segmentation**

Closing technique is used to close the gaps between the words in the signature for later processes, shown in figure 32.

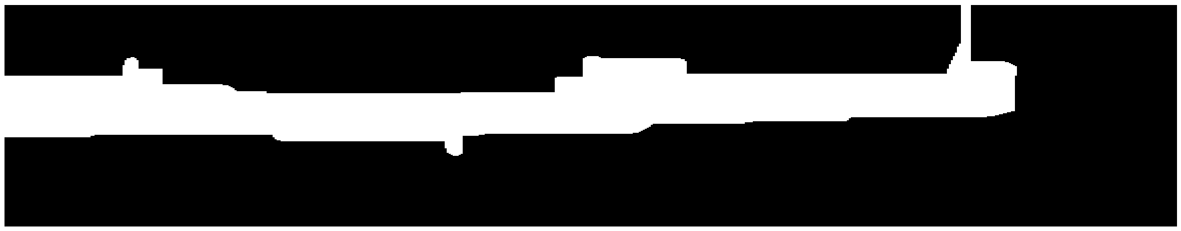


Figure 32. Applying closing technique to the modified image

- **Step 4: Finding the Contour**

To locate the signature, Contour method is used after using closing technique on the image, then the located contour is applied on the original image in figure 29, as shown in figure 33.

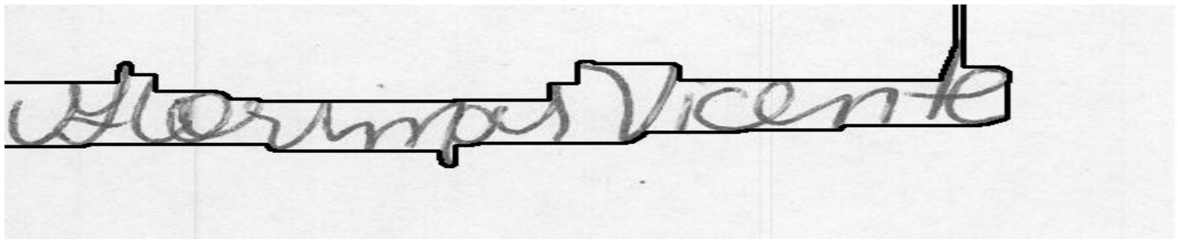


Figure 33. Applying signature contour to the original image

- **Step 5: Surrounding the Signature with a Rectangle**

For later modification on the image, a rectangle surrounds the signature is to be used, as illustrated in figure 34.

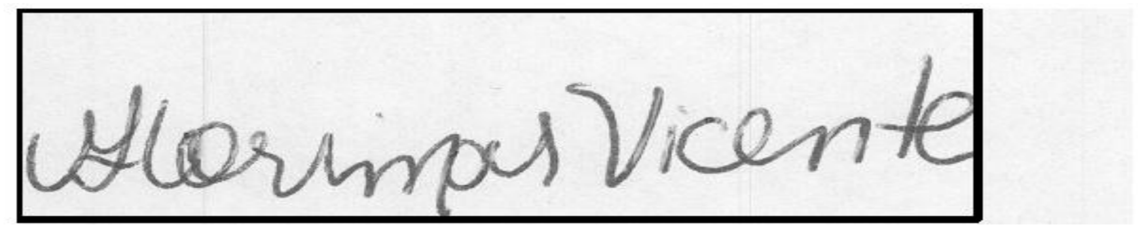


Figure 34. Applying a rectangle around the Signature on the original image

- **Step 6: Cropping the signature**

The signature is cropped from the original image after locating it, as shown in figure 35.

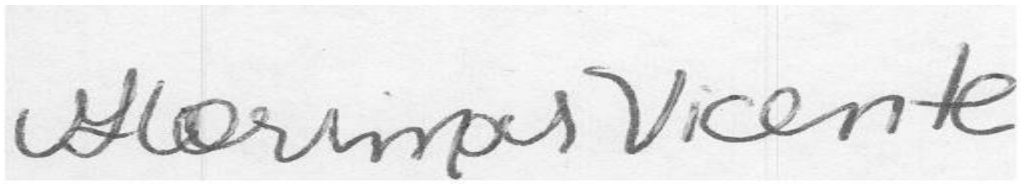


Figure 35. Cropping the signature from the original image

- **Step 7: Signature Resizing**

The cropped signature is to be resized to the dimensions of 150*300, shown in figure 36.

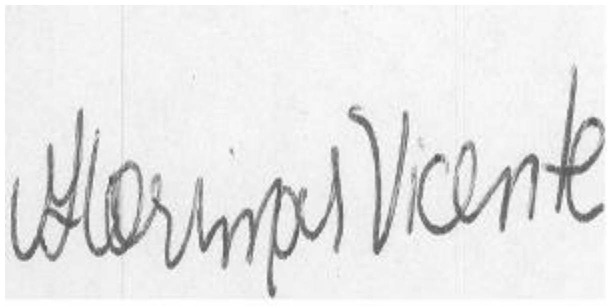


*Figure 36. Resized Signature of 150*300 dimensions*

- **Step 8: binarization**

Same as in step 2, the resized signature is binarized before being fed to the model, as shown in figure 37.

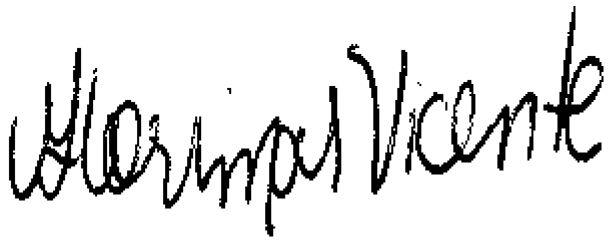


Figure 37. Binarized resized signature

4.1.2) Results

The model's results branches into two parts; statistical results and visual results:

4.1.2.1) Statistical Results

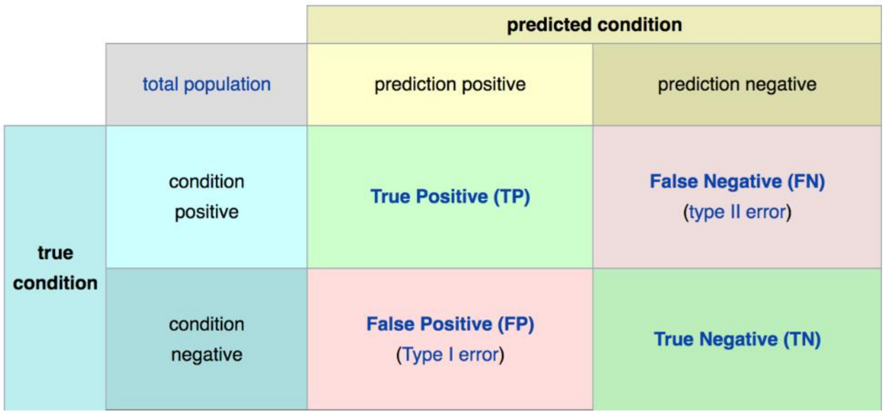
The model is ready to take input images for training and be tested for validation purposes, getting statistical results from each dataset like: Confusion matrix, accuracy score, classification report (precision, recall, f1-score, support) , zero one loss , Roc Curve and AUC:

- **Confusion matrix**

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class. The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e., commonly mislabeling one as another).

- **Terminology and derivations from a confusion matrix**
  - Condition positive (P): The number of real positive cases in the data.
  - Condition negative (N): The number of real negative cases in the data.
  - True positive (TP): Equal with hit.
  - True negative (TN): Equal with correct rejection.
  - False positive (FP): Equal with false alarm, type I error or underestimation.
  - False negative (FN): Equal with miss, type II error or overestimation.

As shown in the following figure 38 and figure 39.

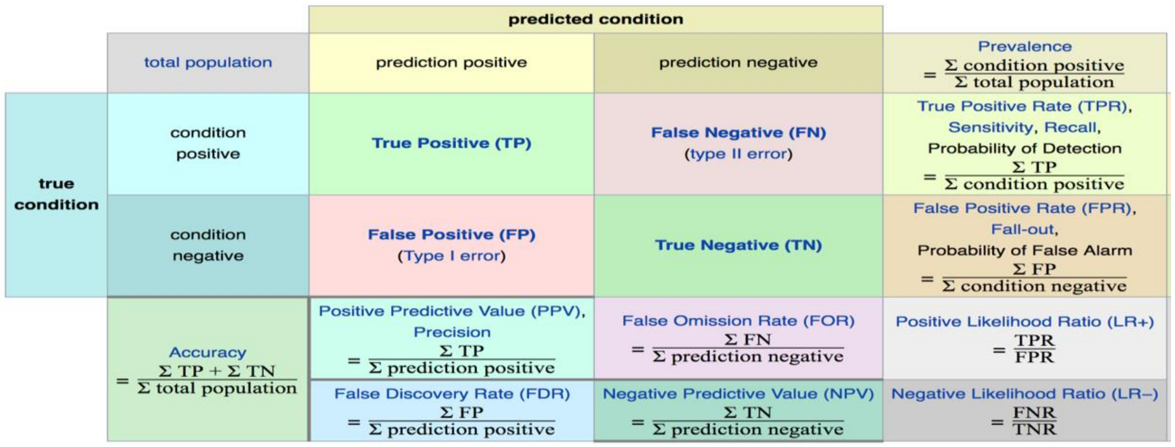


		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**

Figure 38. Confusion Matrix

A confusion matrix $\boldsymbol{C}$ is such that $\boldsymbol{C_{i,j}}$ is equal to the number of observations known to be in group $\boldsymbol{i}$ and predicted to be in group $\boldsymbol{j}$.

Thus, in binary classification, the count of true negatives (TN) is $\boldsymbol{C_{0,0}}$, false negatives (FN) is $\boldsymbol{C_{1,0}}$, true positives (TP) is $\boldsymbol{C_{1,1}}$ and false positives is (FP) $\boldsymbol{C_{0,1}}$.



		predicted condition		
	total population	prediction positive	prediction negative	Prevalence $= \frac{\Sigma \text{ condition positive}}{\Sigma \text{ total population}}$
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)	True Positive Rate (TPR), Sensitivity, Recall, Probability of Detection $= \frac{\Sigma \text{ TP}}{\Sigma \text{ condition positive}}$
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**	False Positive Rate (FPR), Fall-out, Probability of False Alarm $= \frac{\Sigma \text{ FP}}{\Sigma \text{ condition negative}}$
	Accuracy $= \frac{\Sigma \text{ TP} + \Sigma \text{ TN}}{\Sigma \text{ total population}}$	Positive Predictive Value (PPV), Precision $= \frac{\Sigma \text{ TP}}{\Sigma \text{ prediction positive}}$	False Omission Rate (FOR) $= \frac{\Sigma \text{ FN}}{\Sigma \text{ prediction negative}}$	Positive Likelihood Ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$
		False Discovery Rate (FDR) $= \frac{\Sigma \text{ FP}}{\Sigma \text{ prediction positive}}$	Negative Predictive Value (NPV) $= \frac{\Sigma \text{ TN}}{\Sigma \text{ prediction negative}}$	Negative Likelihood Ratio (LR–) $= \frac{\text{FNR}}{\text{TNR}}$

Figure 39. Complete confusion matrix

Confusion matrix is computed to evaluate the accuracy of a classification and we can calculate some of another statistical results like:

- **Accuracy score:**

In multi-label classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y-true. Classification Accuracy is what is usually meant when the term accuracy is used. It is the ratio of number of correct predictions to the total number of input samples as shown in equation 1 and 2.

Equation 2. Accuracy Formula. (1)

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

Equation 1. Accuracy Formula. (2)

Accuracy

$$= \frac{\Sigma\ \text{TP} + \Sigma\ \text{TN}}{\Sigma\ \text{total population}}$$

- **Classification Report (precision, recall, f1-score, support)**

It consist of some of statistical results:

➢ **Precision**: the number of correct positive results divided by the number of positive results predicted by the classifier, as shown in equations, 3 and 4.

Equation 4. Precision Formula .(1)

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Equation 3. Precision Formula. (2)

Positive Predictive Value (PPV), Precision

$$= \frac{\Sigma\ \text{TP}}{\Sigma\ \text{prediction positive}}$$

➢ **Recall**: the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive), as shown in following equation , 5 and 6.

Equation 6. TruePositiveRate Formula. (1)

$$TruePositiveRate = \frac{TruePositive}{FalseNegative + TruePositive}$$

Equation 5. TruePositiveRate Formula. (2)

True Positive Rate (TPR), Sensitivity, Recall, Probability of Detection

$$= \frac{\Sigma\ \text{TP}}{\Sigma\ \text{condition positive}}$$

- **F1-Score:** The Harmonic Mean between precision and recall. The range for F1-Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances). High precision but lower recall, gives an extreme accuracy, but it then misses many instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model. Mathematically, it can be expressed as the following Equation 7 and 8 shows:

Equation 8. F1-Score Formula. (1)

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

Equation 7. F1-Score Formula. (2)

$$F_1 \text{ score} = \frac{2 \cdot PPV \cdot TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$$

F1-Score tries to find the balance between precision and recall.

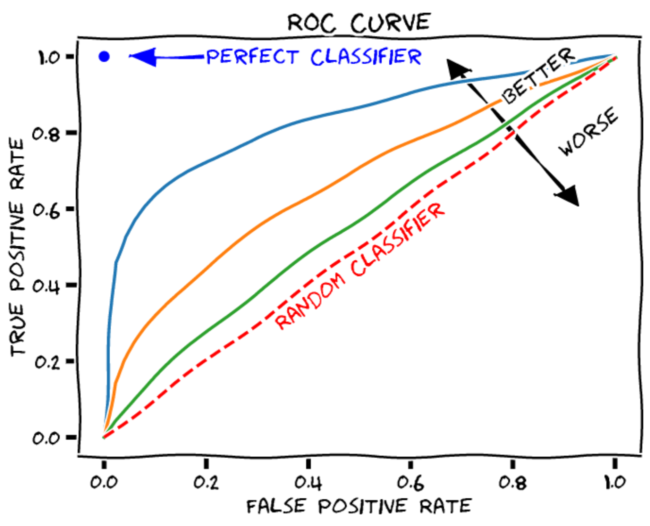
- **Support:** the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support does not change between models but instead diagnoses the evaluation process.

- **Zero One Loss**

The fraction of misclassifications or the number of misclassifications.

- **Roc Curve and AUC**

AUC - ROC curve, as shown in figure 40 is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of reparability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.



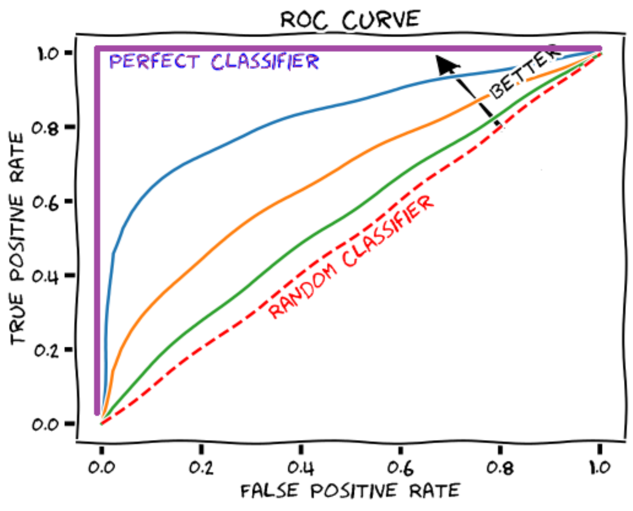


Figure 40. General ROC Curve.

ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds as illustrated in figure 41. This curve plots two parameters:

1. True Positive Rate (TPR)
2. False Positive Rate (FPR)

True Positive Rate (Sensitivity) (TPR): True Positive Rate is defined as TP/(FN+TP), as shown in equation 5 and 6. True Positive Rate corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points.

False Positive Rate (FPR): False Positive Rate is defined as FP / (FP+TN) as shown in Equation 9 and 10. False Positive Rate corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points.

Equation 9. FalsePositiveRate Formula. (1)

Equation 10. FalsePositiveRate Formula. (2)

$$FalsePositiveRate = \frac{FalsePositive}{TrueNegative + FalsePositive}$$

False Positive Rate (FPR),
Fall-out,
Probability of False Alarm

$$= \frac{\Sigma \text{ FP}}{\Sigma \text{ condition negative}}$$

AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus).

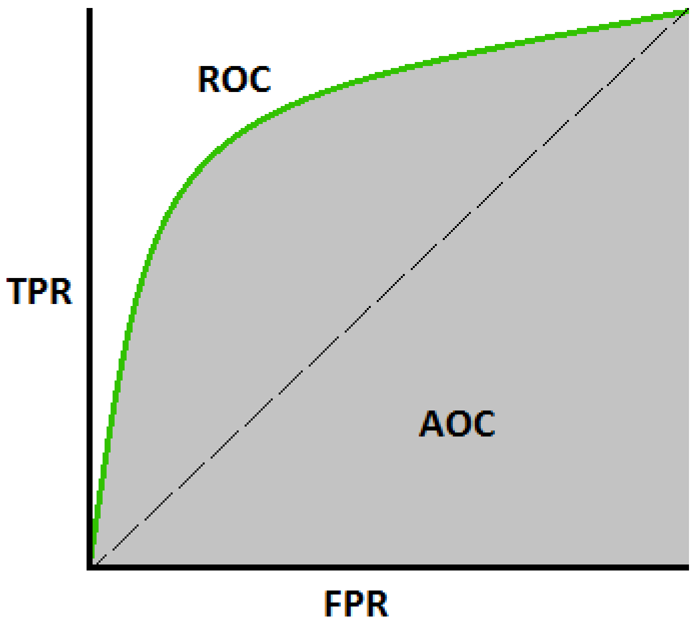


Figure 41. ROC and AUC Curve

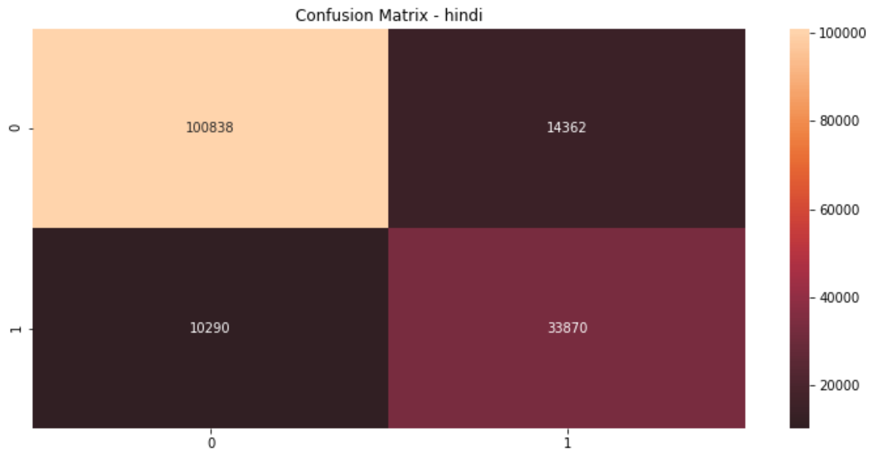
Now, for the dataset statistical results

- **BHSig260 (Hindi)**

It consists of 160 unique signatures each signature consists of 24 genuine signatures and 30 forged signatures for the Hindi signatures, and it make number of samples (pairs) 159360 pairs of images consist of 115200 Genuine-Forged pairs and 44160 Genuine-Genuine pairs.

- **Confusion Matrix**

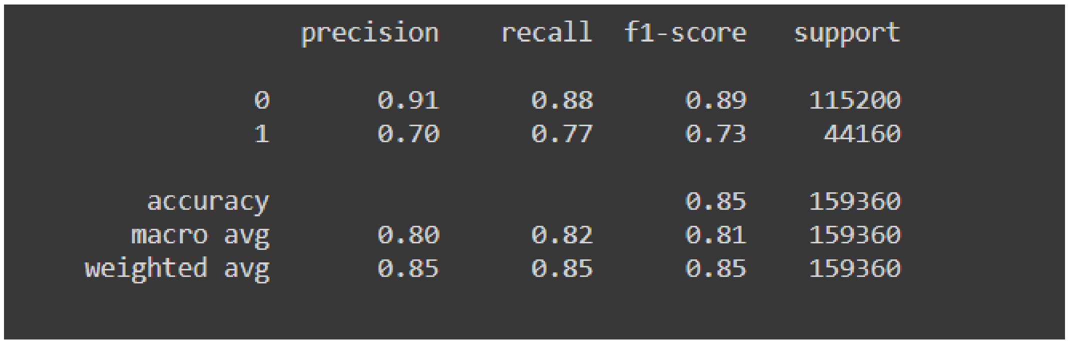
Table 7. BHSig260 (Hindi) Confusion Matrix



- **Accuracy Score**
    - The BHSig260 (Hindi) dataset gave accuracy of 84.53062248995984 %

- **Classification report (precision, recall, f1-score, support)**

Table 8. . BHSig260 dataset's precision, recall, f1-score and support results



```
              precision    recall  f1-score   support

           0       0.91      0.88      0.89    115200
           1       0.70      0.77      0.73     44160

    accuracy                           0.85    159360
   macro avg       0.80      0.82      0.81    159360
weighted avg       0.85      0.85      0.85    159360
```

- **Zero One Loss**
    - Wrong classification value: 24652
    - Wrong classification percentage: 15.47 %

- **ROC and AUC Curve**
  - False Positive Rate Value: [0.   0.12467014 1.   ]
  - True Positive Rate Value: [0.   0.7669837 1.   ]
  - Thresholds Value: [2 1 0]
  - AUC: 0.8211567783816425

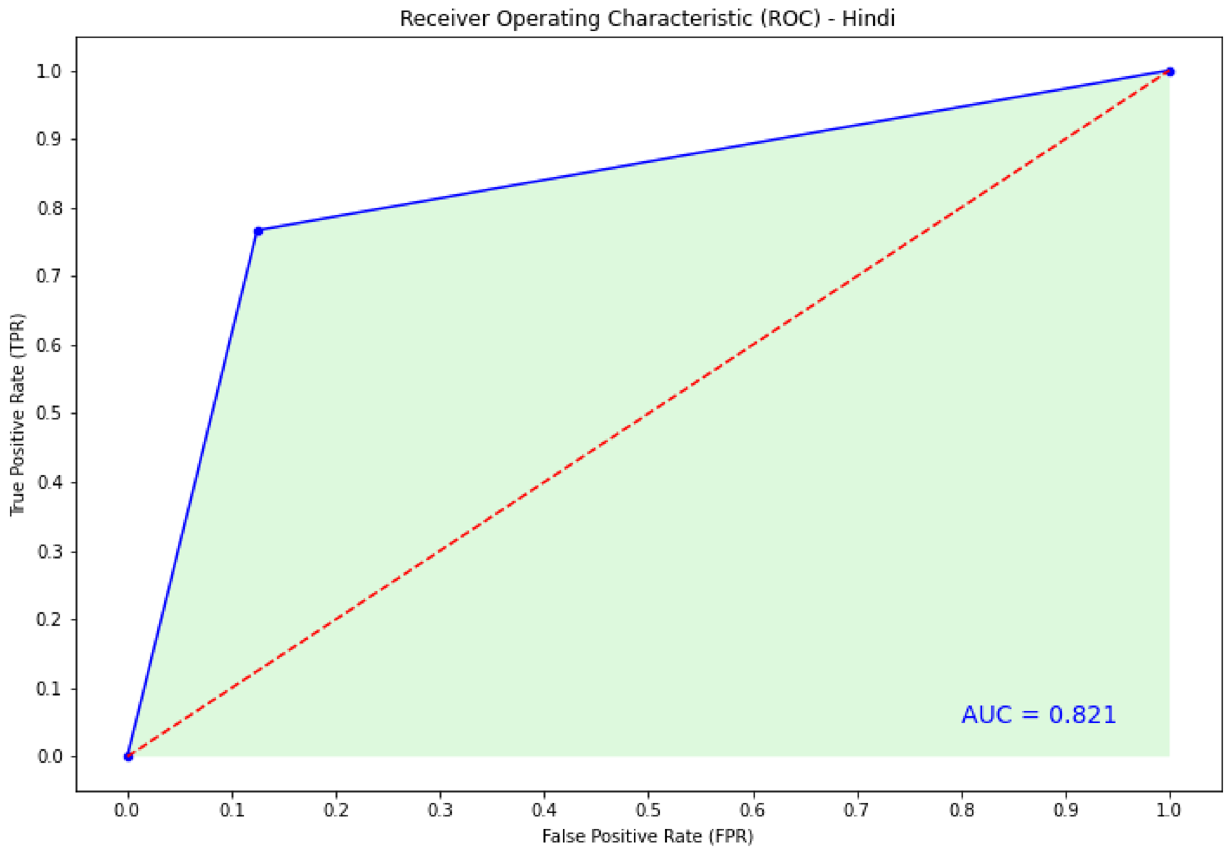


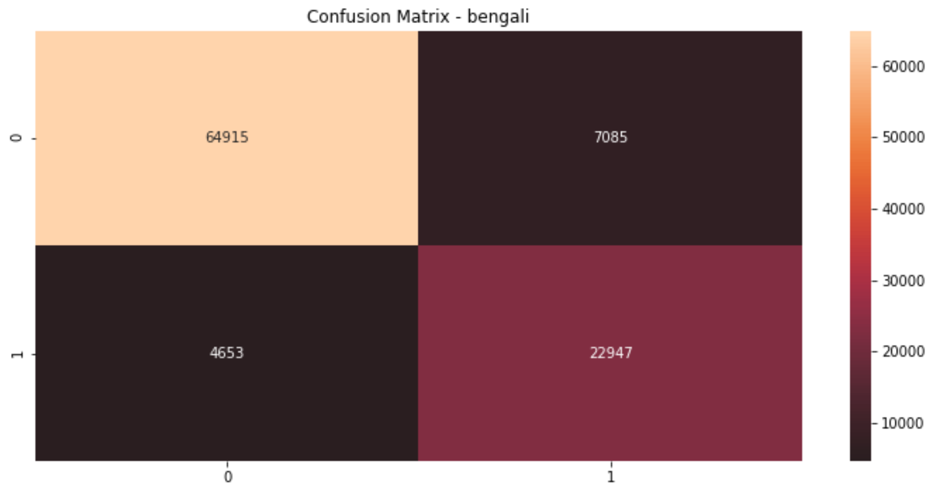
Figure 42. BHSig260 (HINDI) ROC AUC Curve

- **BHSig260 (Bengali)**

It consists of 100 unique signatures each signature consists of 24 genuine signatures and 30 forged signatures for the Bengali signatures, and it make number of samples (pairs) 99600 pairs of images consist of 72000 Genuine-Forged pairs and 27600 Genuine-Genuine pairs.

- **Confusion Matrix**

Table 9. BHSig260 (Bengali) confusion matrix

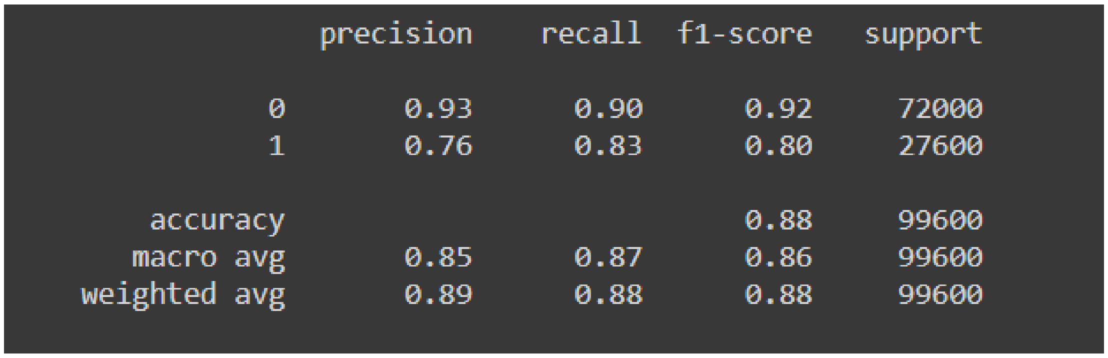


- **Accuracy Score**

The BHSig260 (Bengali) dataset gave accuracy of 88.214859437751 %

- **Classification report (precision, recall, f1-score, support)**

Table 10. BHSig260 (Bengali) Classification report



```
              precision    recall  f1-score   support

           0       0.93      0.90      0.92     72000
           1       0.76      0.83      0.80     27600

    accuracy                           0.88     99600
   macro avg       0.85      0.87      0.86     99600
weighted avg       0.89      0.88      0.88     99600
```

- **Zero One loss**
  - Wrong classification value: 11738
  - Wrong classification percentage: 11.79 %

- **ROC and AUC curve**
  - False Positive Rate Value: [0.   0.09840278 1.   ]
  - True Positive Rate Value: [0.   0.83141304 1.   ]
  - Thresholds Value: [2 1 0]
  - AUC: 0.8665051328502414

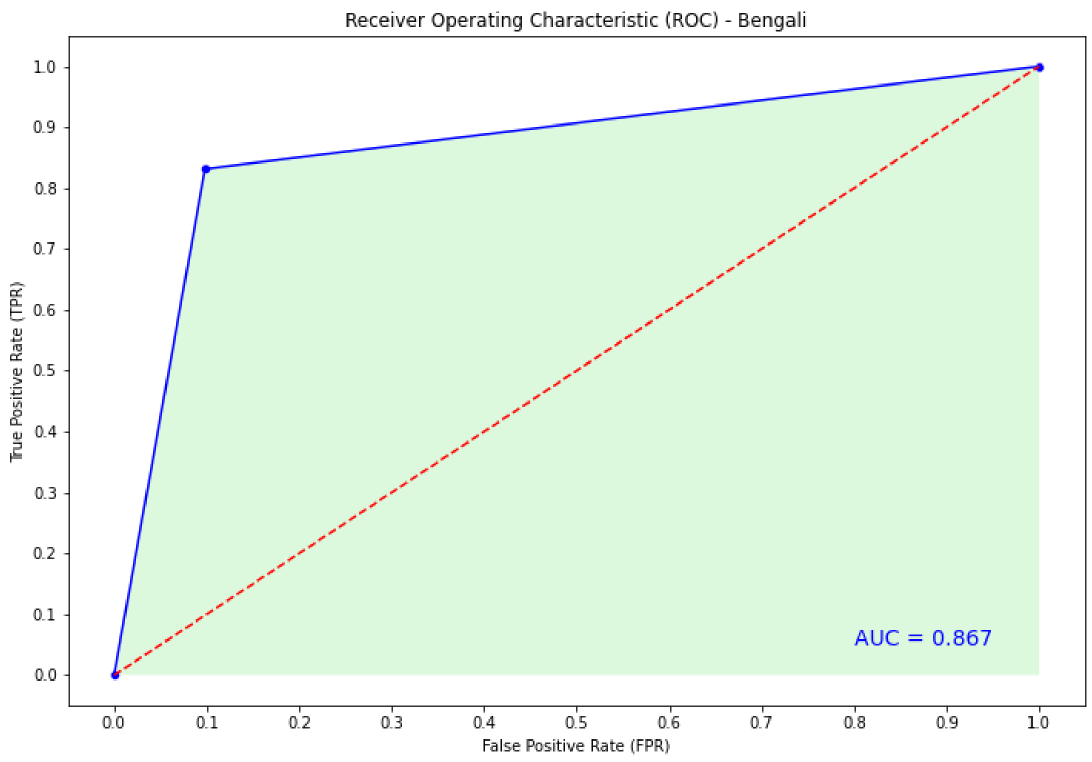


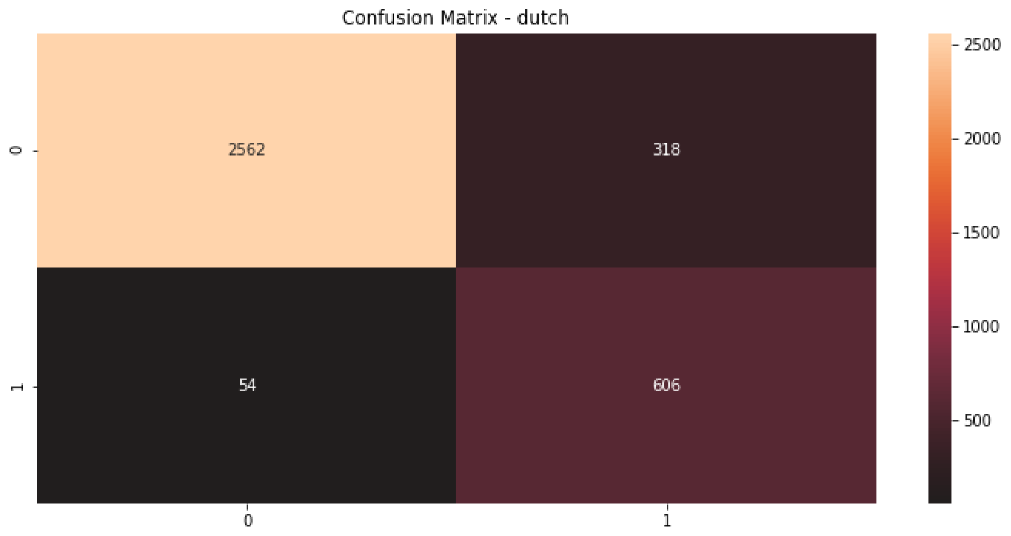
Figure 43. BHSig260 (Bengali) ROC AUC Curve

- **ICDAR2011 (Dutch)**

It consists of 10 unique signatures each signature consists of 12 genuine signatures and 24 forged signatures for the Dutch signatures, and it make number of samples (pairs) 3540 pairs of images consist of 2880 Genuine-Forged pairs and 660 Genuine-Genuine pairs.

- **Confusion matrix**

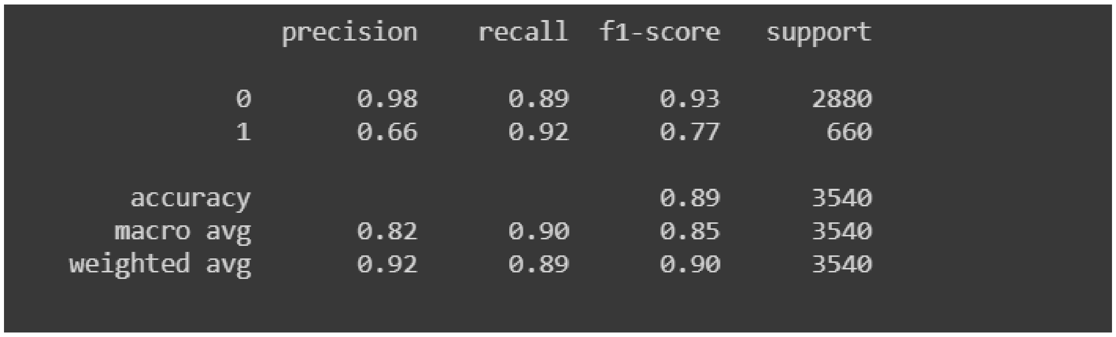
Table 11. ICDAR2011 (Dutch) confusion matrix



- **Accuracy**
  - ➢ The ICDAR2011 (Dutch) dataset gave accuracy of 89.49152542372882 %

- **Classification report (precision, recall, f1-score, support)**

Table 12. ICDAR2011 (Dutch) classification report



```
              precision    recall  f1-score   support

           0       0.98      0.89      0.93      2880
           1       0.66      0.92      0.77       660

    accuracy                           0.89      3540
   macro avg       0.82      0.90      0.85      3540
weighted avg       0.92      0.89      0.90      3540
```

- **Zero One Loss**
  - Wrong classification value: 372
  - Wrong classification percentage: 10.51 %

- **ROC and AUC curve**
  - False Positive Rate Value: [0.   0.11041667 1.   ]
  - True Positive Rate Value: [0.   0.91818182 1.   ]
  - Thresholds Value: [2 1 0]
  - AUC: 0.9038825757575758

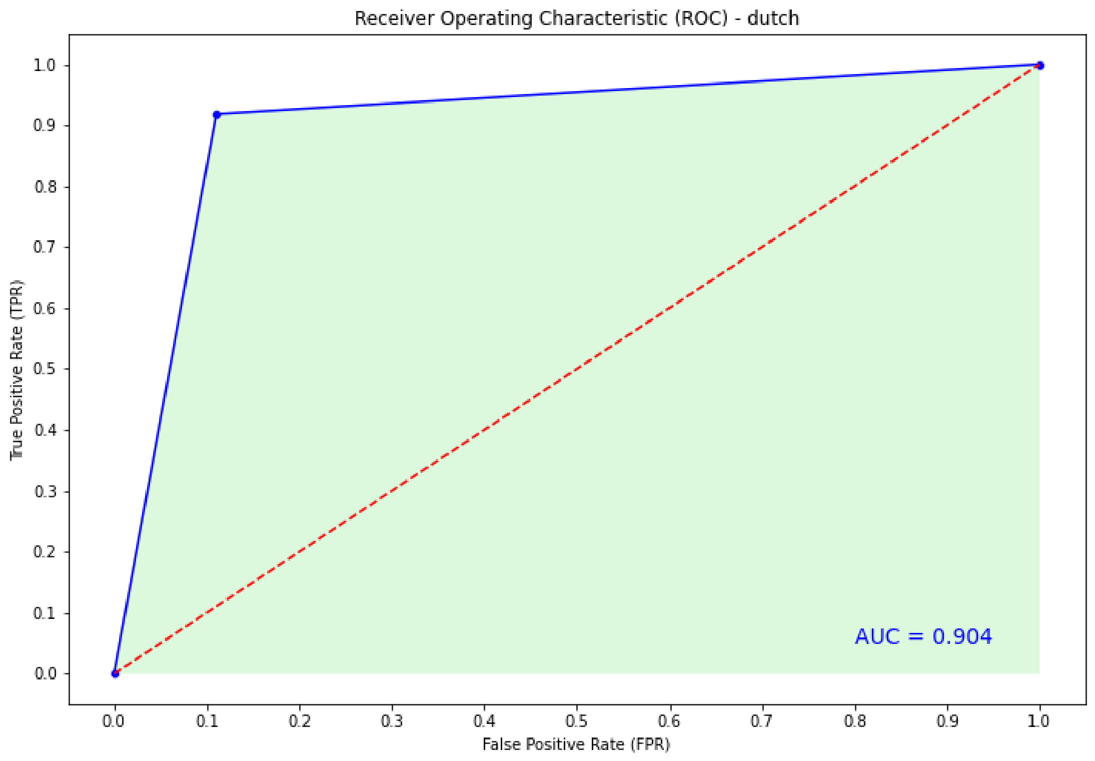


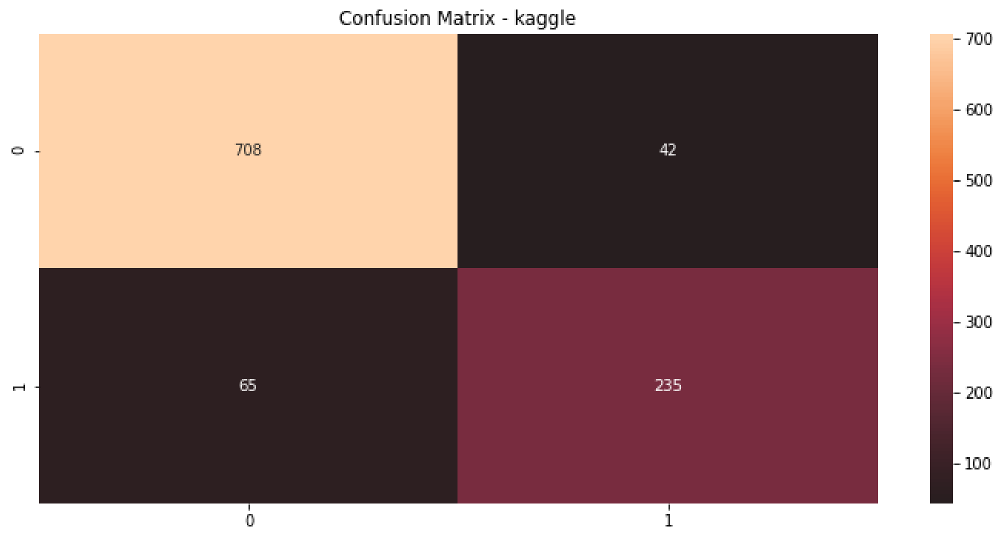
Figure 44. ICDAR2011 (DUTCH) ROC AUC Curve

- **Kaggle (English)**

It consists of 30 unique signatures each signature consists of 5 genuine signatures and 5 forged signatures for the Kaggle signatures, and it make number of samples (pairs) 1050 pairs of images consist of 750 Genuine-Forged pairs and 300 Genuine-Genuine pairs.

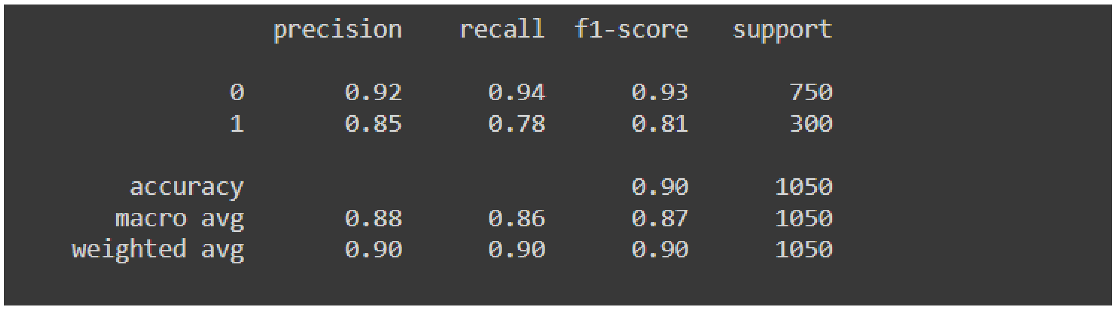
- **Confusion Matrix**

Table 13. Kaggle (English) Confusion Matrix



- **Accuracy Score**
  - The Kaggle (English) dataset gave accuracy of 89.80952380952381 %
- **Classification report (precision, recall, f1-score, support)**

Table 14. Kaggle (English) Classification report



```
              precision    recall  f1-score   support

           0       0.92      0.94      0.93       750
           1       0.85      0.78      0.81       300

    accuracy                           0.90      1050
   macro avg       0.88      0.86      0.87      1050
weighted avg       0.90      0.90      0.90      1050
```

- **Zero One Loss**
  - Wrong classification value: 107
  - Wrong classification percentage: 10.19 %

- **ROC and AUC Curve**
  - False Positive Rate Value: [0.   0.056 1.   ]
  - True Positive Rate Value: [0.   0.78333333 1.   ]
  - Thresholds Value: [2 1 0]
  - AUC: 0.8636666666666666

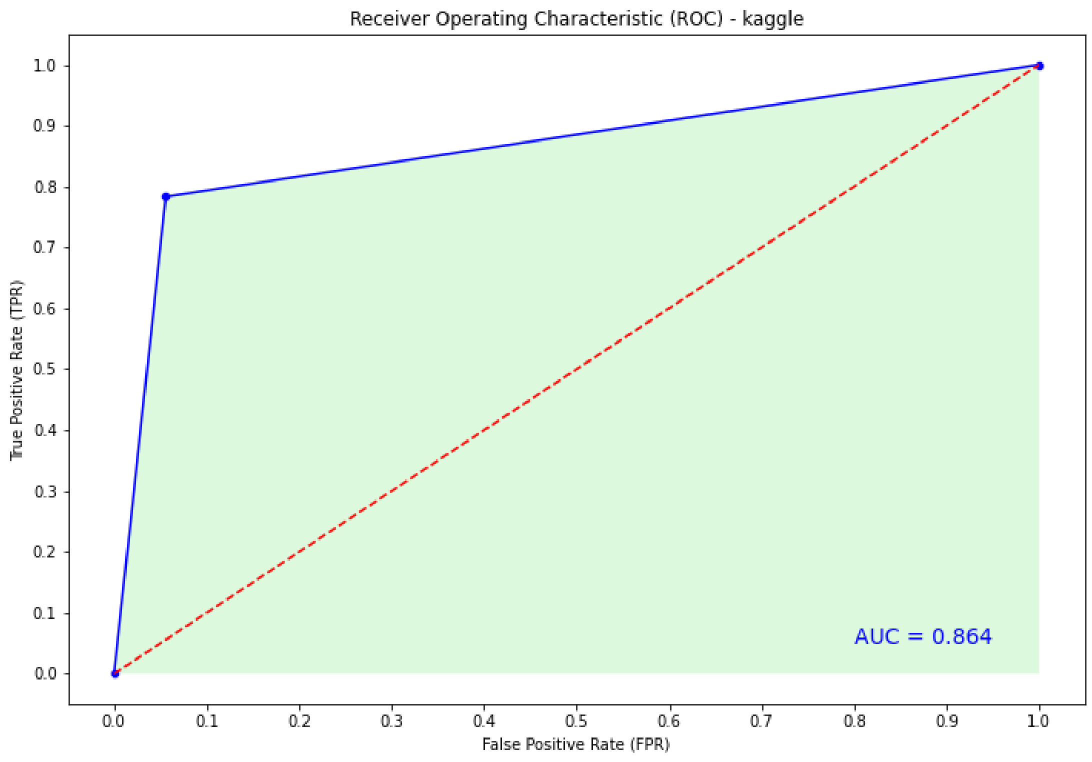


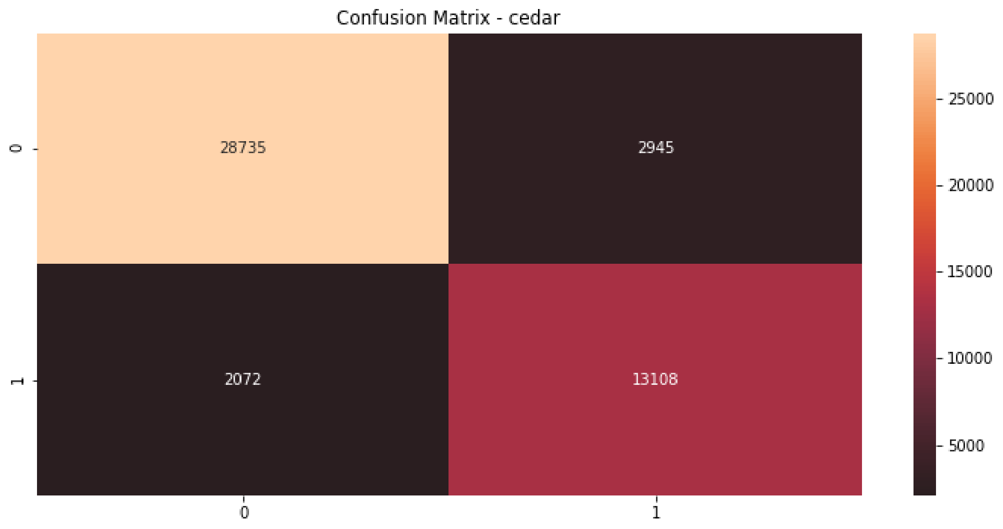
Figure 45. Kaggle (English) ROC AUC Curve

- **Cedar (English)**

It consists of 55 unique signatures each signature consists of 24 genuine signatures and 24 forged signatures for the Cedar signatures, and it make number of samples (pairs) 46860 pairs of images consist of 31680 Genuine-Forged pairs and 15180 Genuine-Genuine pairs.

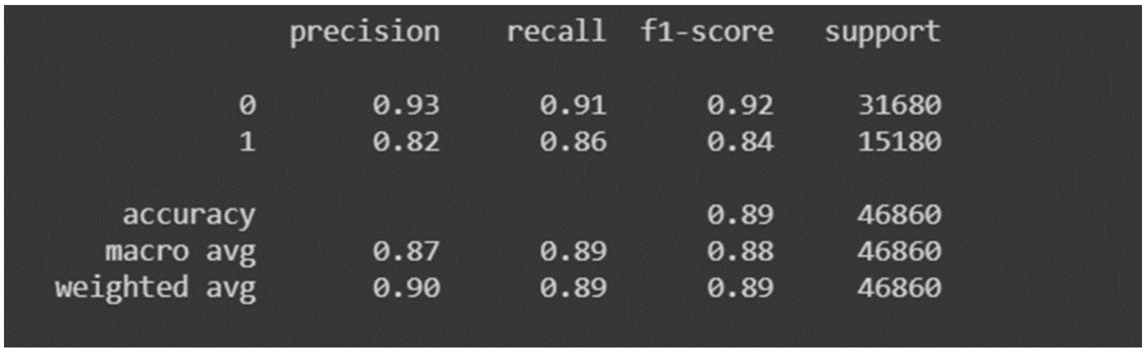
- **Confusion Matrix**

Table 15. Cedar (English) Confusion Matrix



- **Accuracy Score**
  - The Cedar (English) dataset gave accuracy of 89.29364063166881 %
- **Classification Report**

Table 16. Cedar (English) Classification Report



```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92     31680
           1       0.82      0.86      0.84     15180

    accuracy                           0.89     46860
   macro avg       0.87      0.89      0.88     46860
weighted avg       0.90      0.89      0.89     46860
```

- **Zero One Loss**

  - Wrong classification value: 5017
  - Wrong classification percentage: 10.71 %

- **ROC and AUC Curve**
  - False Positive Rate Value: [0.   0.09296086 1.   ]
  - True Positive Rate Value: [0.   0.86350461 1.   ]
  - Thresholds Value: [2 1 0]
  - AUC: 0.8852718763724199

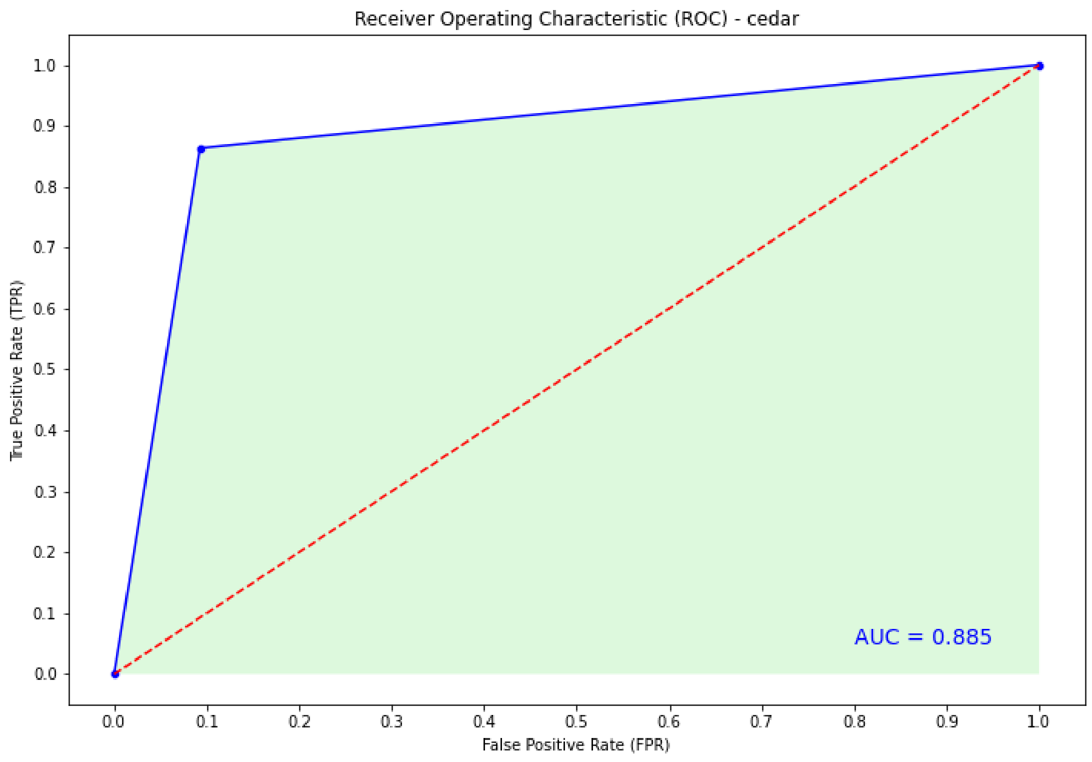


Figure 46. Cedar (English) ROC AUC Curve

4.2) Test Cases and Visual Results

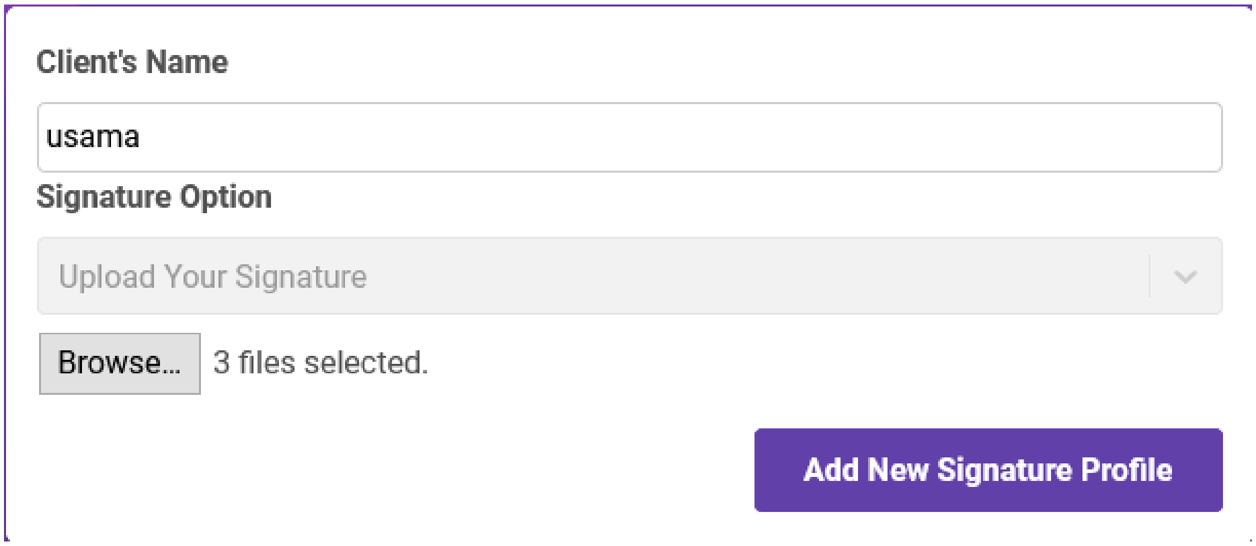
Now, for more testing and validation purposes, several test cases have been chosen for experimental purposes:

- **Case1: Testing General Case for a user "Usama"**

Table 17. signatures of user Usama used in test case 1

Signature Number	Signature Image	Signature Type
1	[illegible]	Original
2	[illegible]	Original
3	[illegible]	Original
4	[illegible]	Genuine
5	[illegible]	Forged

1) Creating a new signature profile for a user called Usama as shown in figure 47 and figure 48 :


Client's Name

usama

Signature Option

Upload Your Signature

Figure 47. Demonstration of creating new signature profile for new user Usama.

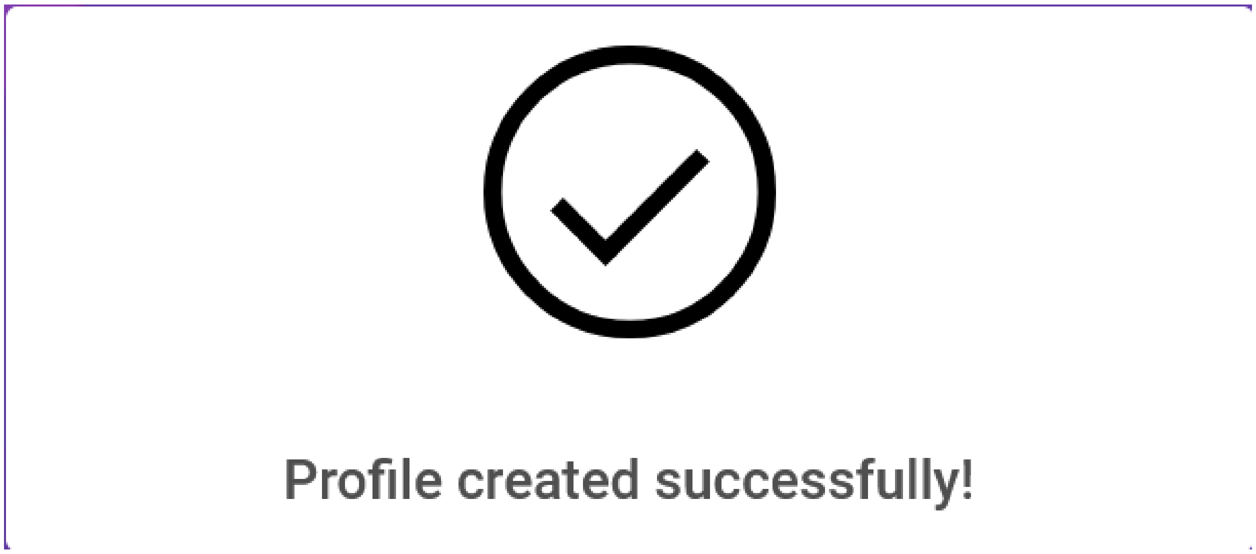
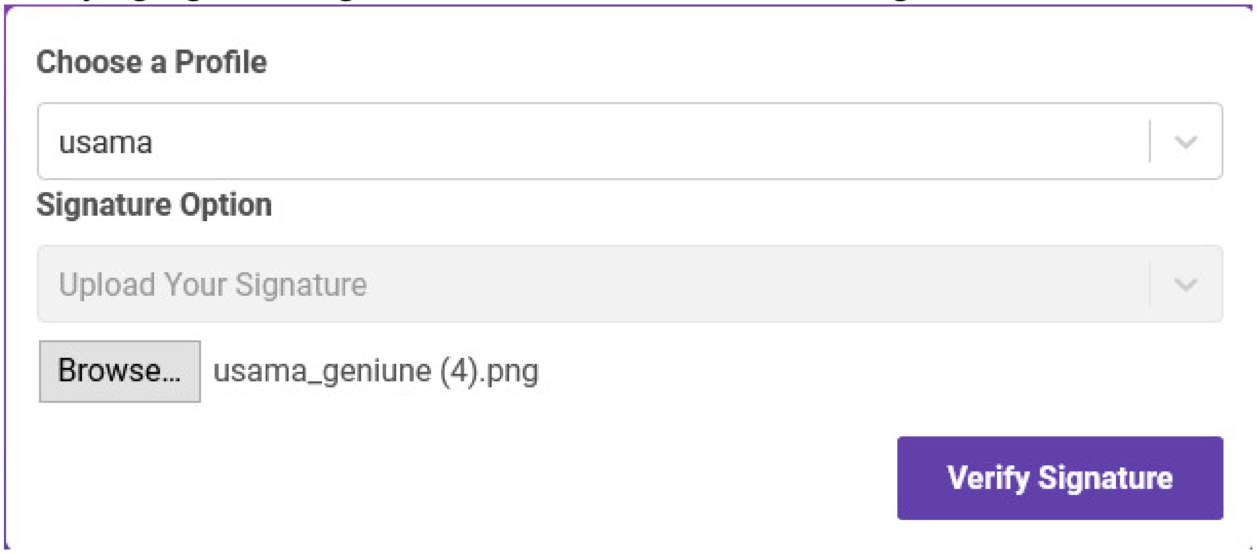

Profile created successfully!

Figure 48. Demonstration of successful creation message

2) Verifying a genuine signature of user Usama as shown in figure 49:


Choose a Profile

usama

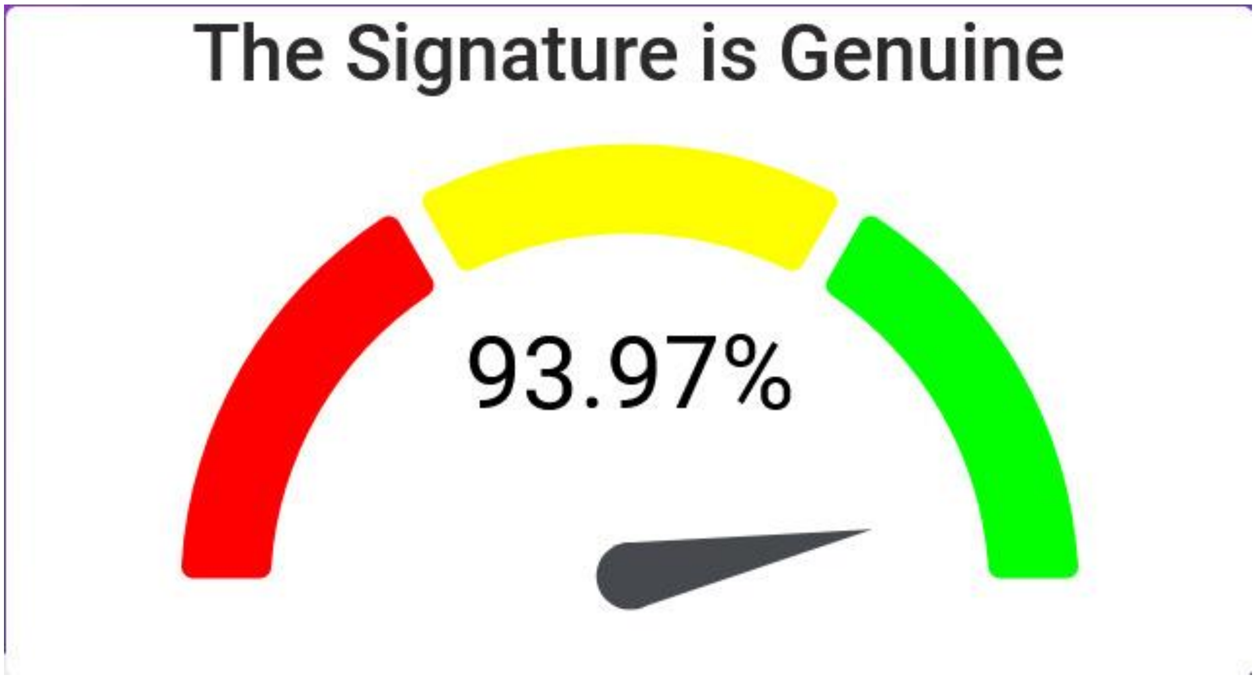
Signature Option

Upload Your Signature

Browse... usama_geniune (4).png

Figure 49. Demonstration of verifying a genuine signature of user Usama.

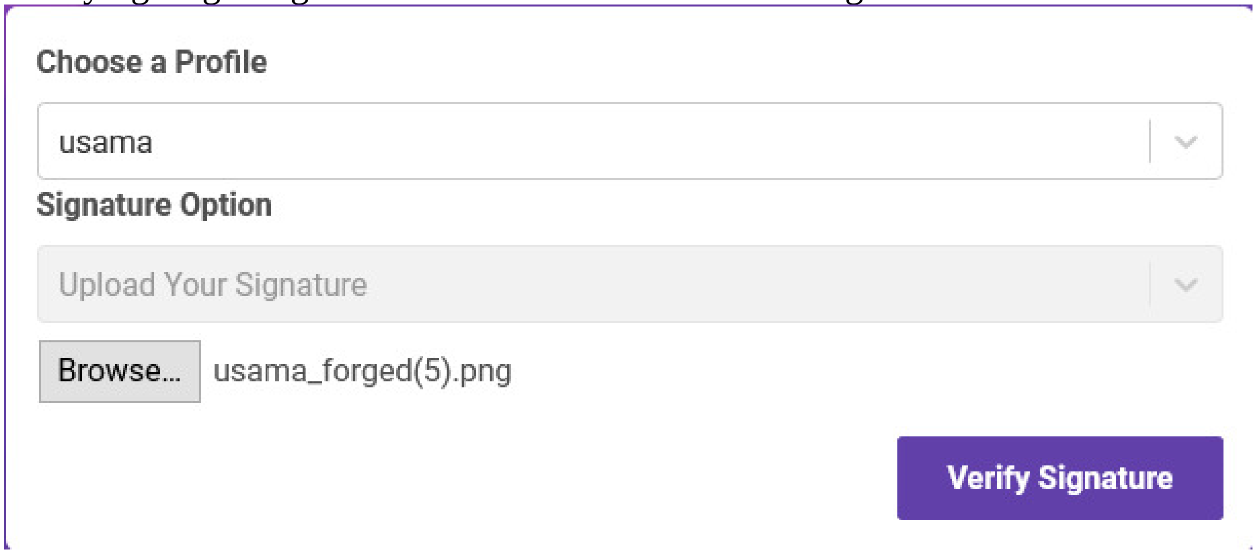
➢ Result of verifying with a genuine signature of user Usama, shown in figure 50:


The Signature is Genuine

93.97%

Figure 50. Demonstration of result of verifying a genuine signature of user Usama.

3) Verifying forged signature of user Usama as shown in figure 51:


Choose a Profile

usama

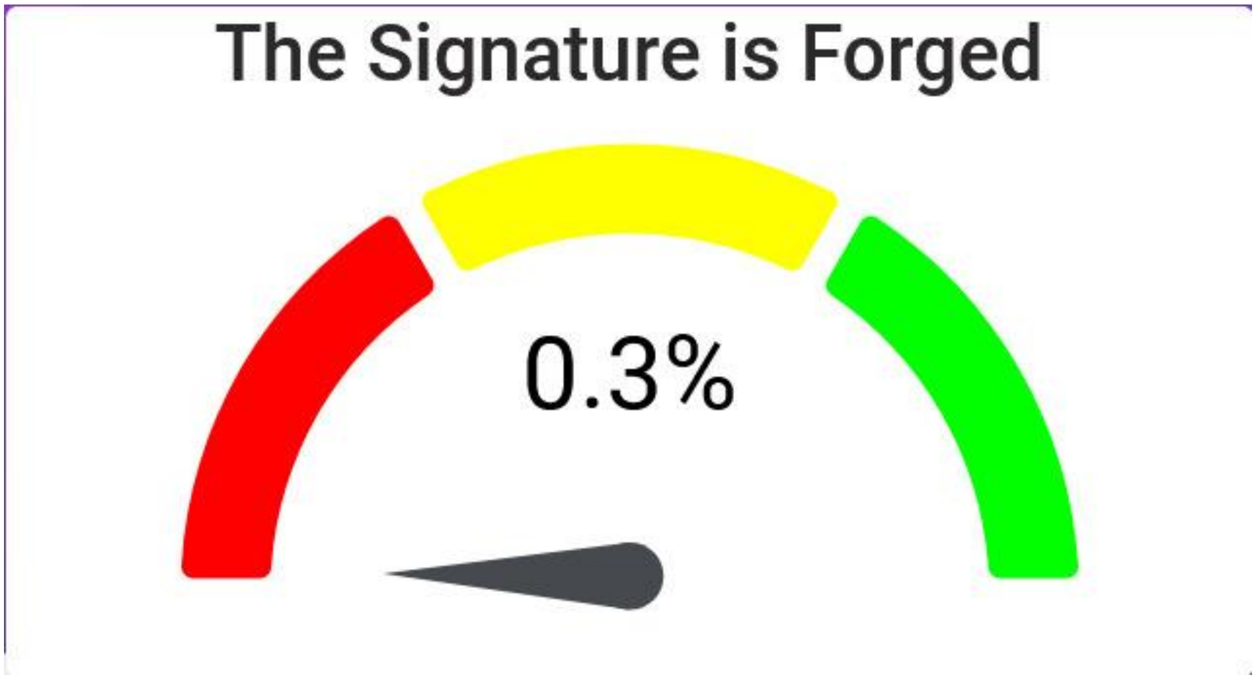
Signature Option

Upload Your Signature

Browse... usama_forged(5).png

Figure 51. Demonstration of verifying a forged signature of user Usama.

➢ Result of verifying with a forged signature of user Usama, shown in figure 52:

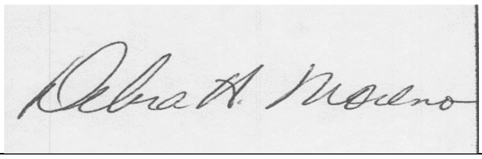
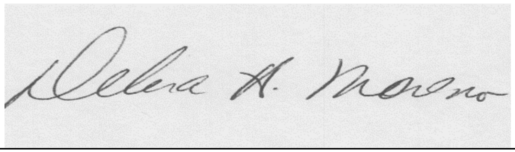
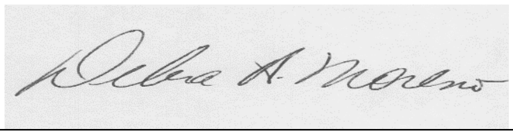
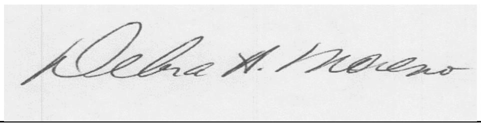
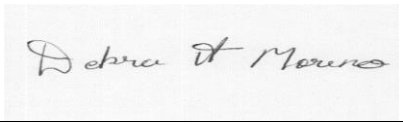
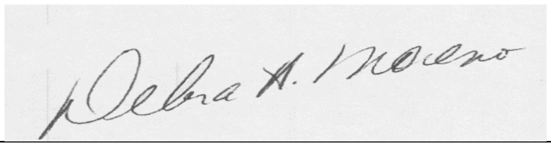
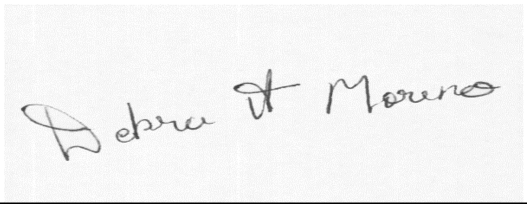

The Signature is Forged

0.3%

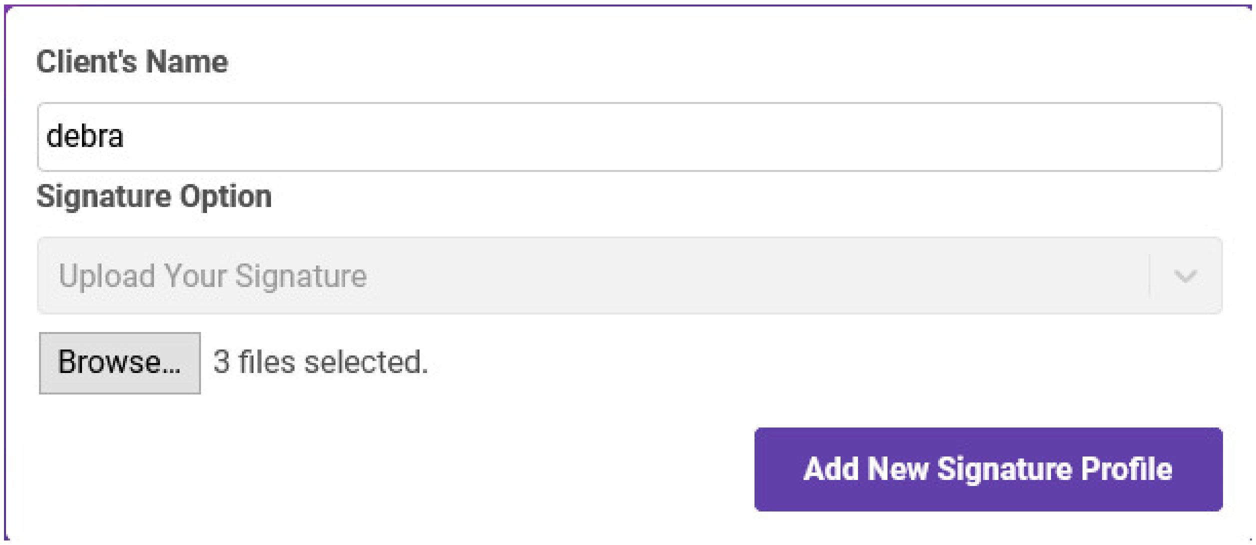
Figure 52. Demonstration of result of verifying a Forged signature of user Usama.

- Test Case 2: General Case with a genuine and forged signature rotated by 15^0

Table 18. signatures of user Debra used in test case 2

Signature Number	Signature Image	Signature Type
1		Original
2		Original
3		Original
4		Genuine
5		Forged
6		Genuine Rotated By 15^0
7		Forged Rotated By 15^0

1) Creating a new signature profile for a user called Debra as shown in figure 53 and figure 54:


Client's Name

debra

Signature Option

Upload Your Signature

Browse... 3 files selected.

Figure 53. Demonstration of creating new signature profile for new user Debra.

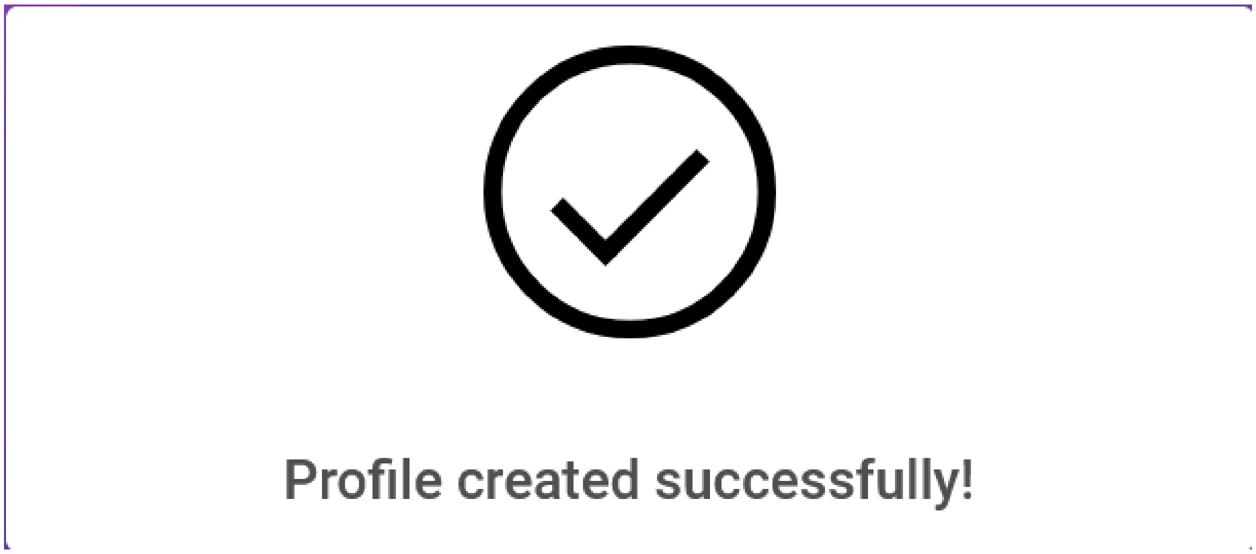
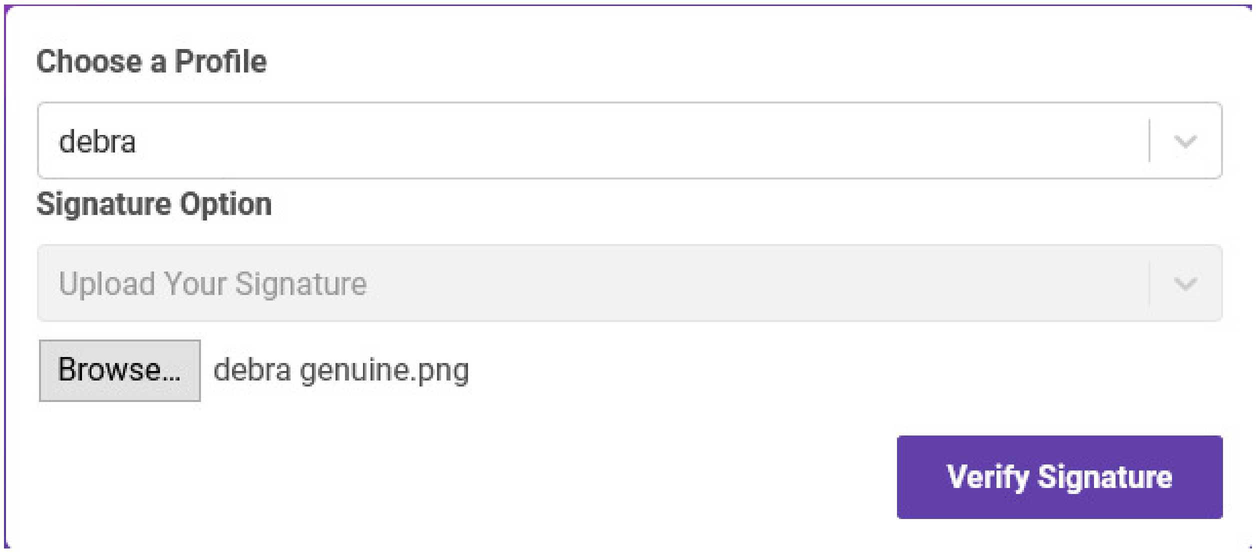

Profile created successfully!

Figure 54. Demonstration of successful creation message

2) Verifying a genuine signature of user Debra as shown in figure 55:


Choose a Profile

debra

Signature Option

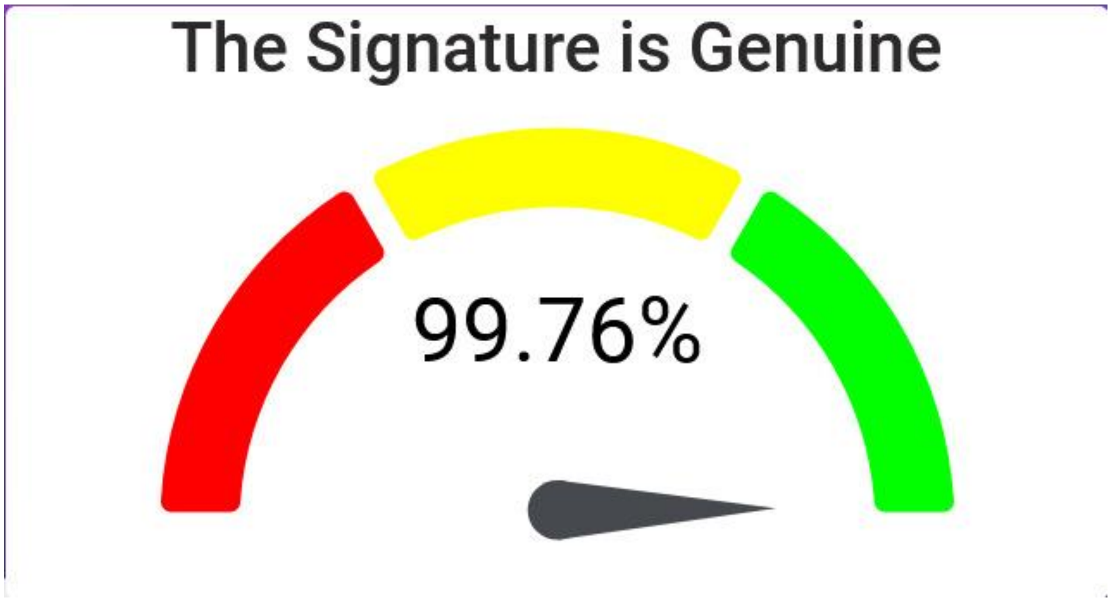
Upload Your Signature

debra genuine.png

Figure 55. Demonstration of verifying a genuine signature of user Debra

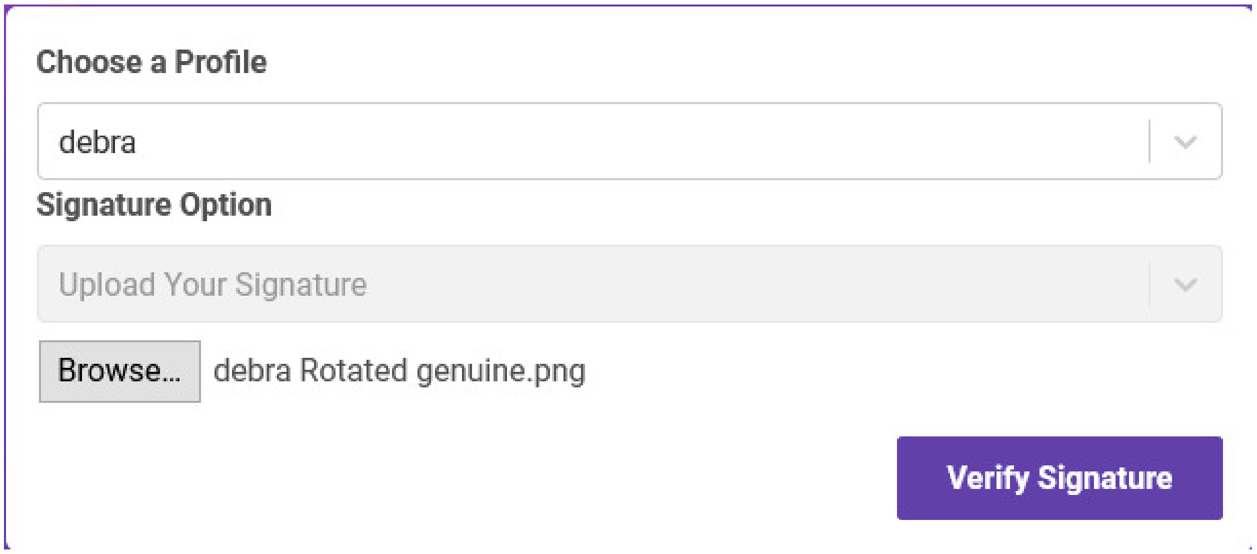
➢ Result of verifying with a genuine signature of user Debra, shown in figure 56:

Figure 56. Demonstration of result of verifying a genuine signature of user Debra


The Signature is Genuine

99.76%

3) Verifying a genuine signature rotated by 15^0 of user Debra as shown in figure 57:


Choose a Profile

debra

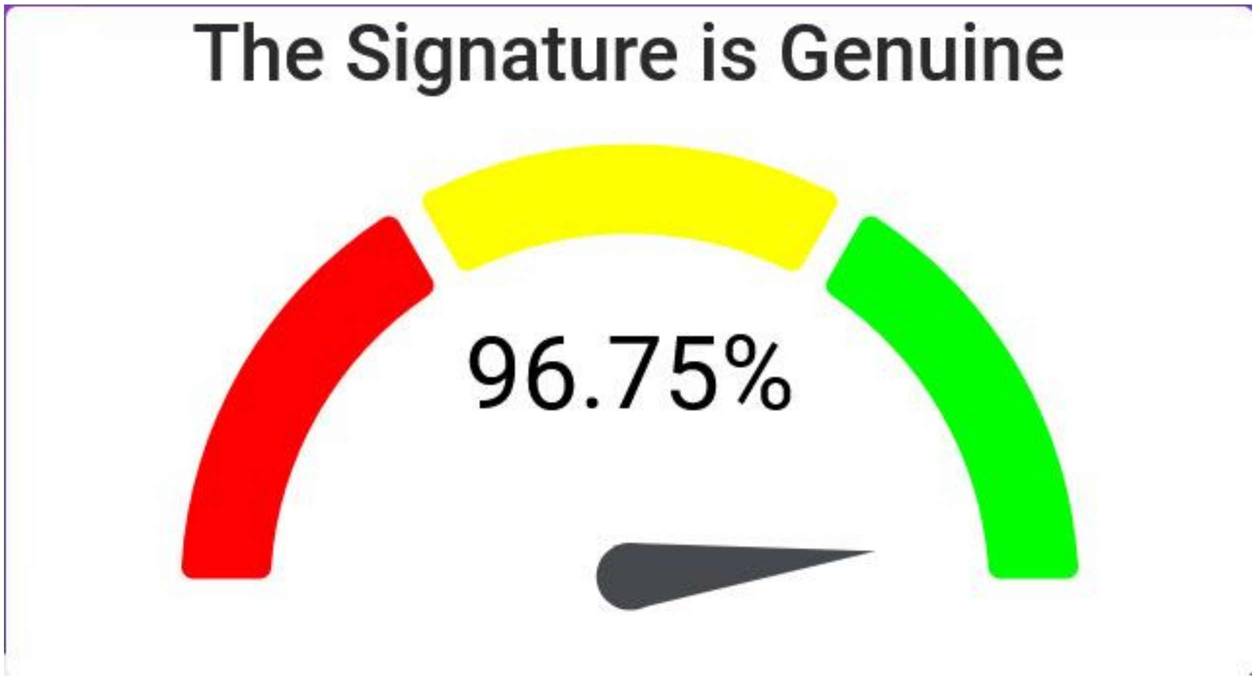
Signature Option

Upload Your Signature

debra Rotated genuine.png

Figure 57. Demonstration of verifying a genuine signature Rotated By 15^0 of user Debra

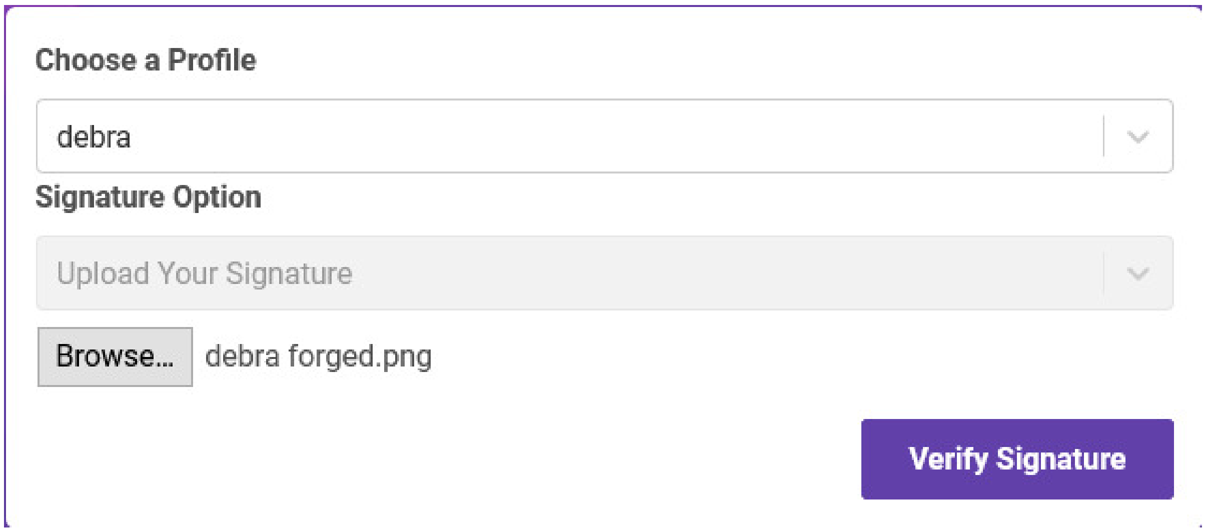
➢ Result of verifying with a genuine signature rotated by 15^0 of user Debra, shown in figure 58:


The Signature is Genuine

96.75%

Figure 58. Demonstration of result of verifying a genuine signature Rotated By 15^0 of user Debra

4) Verifying forged signature of user Debra as shown in figure 59:


Choose a Profile

debra

Signature Option

Upload Your Signature

Browse... debra forged.png

Figure 59. Demonstration of verifying a forged signature of user Debra.

➢ Result of Verifying with a forged signature of user Usama, shown in figure 60:

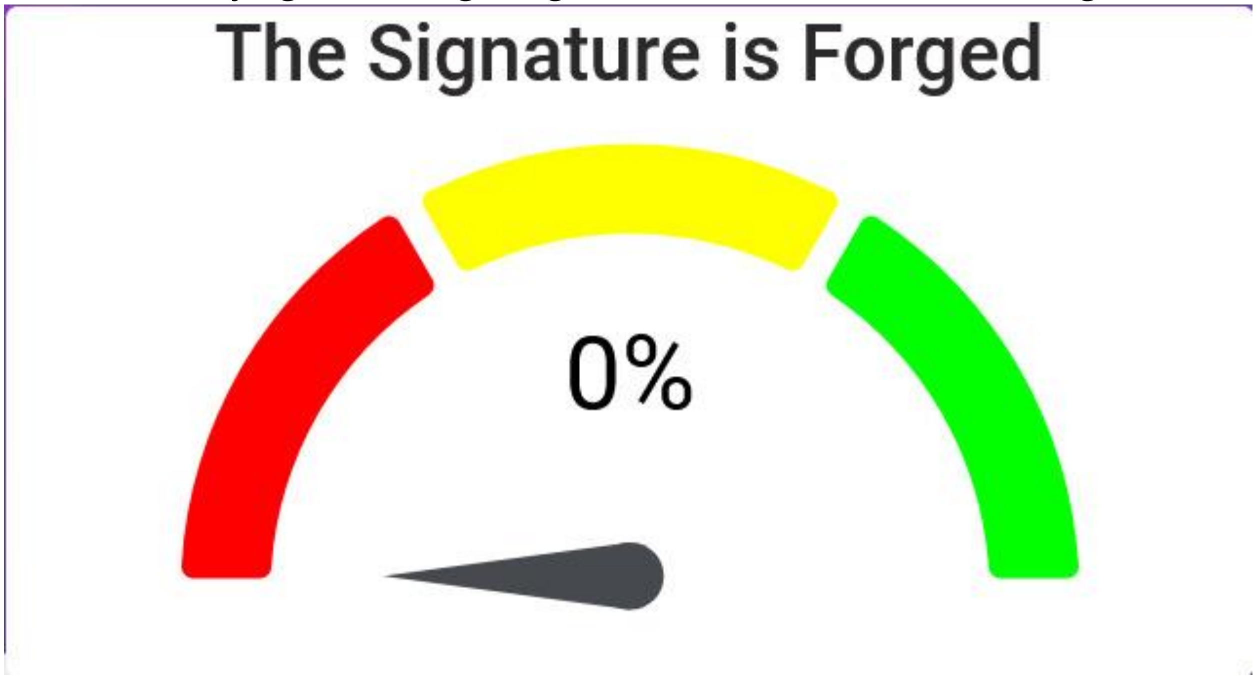

The Signature is Forged

Figure 60. Demonstration of result of verifying a Forged signature of user Debra.

5) Verifying a genuine signature rotated by 15^0 of user Debra as shown in figure 61:

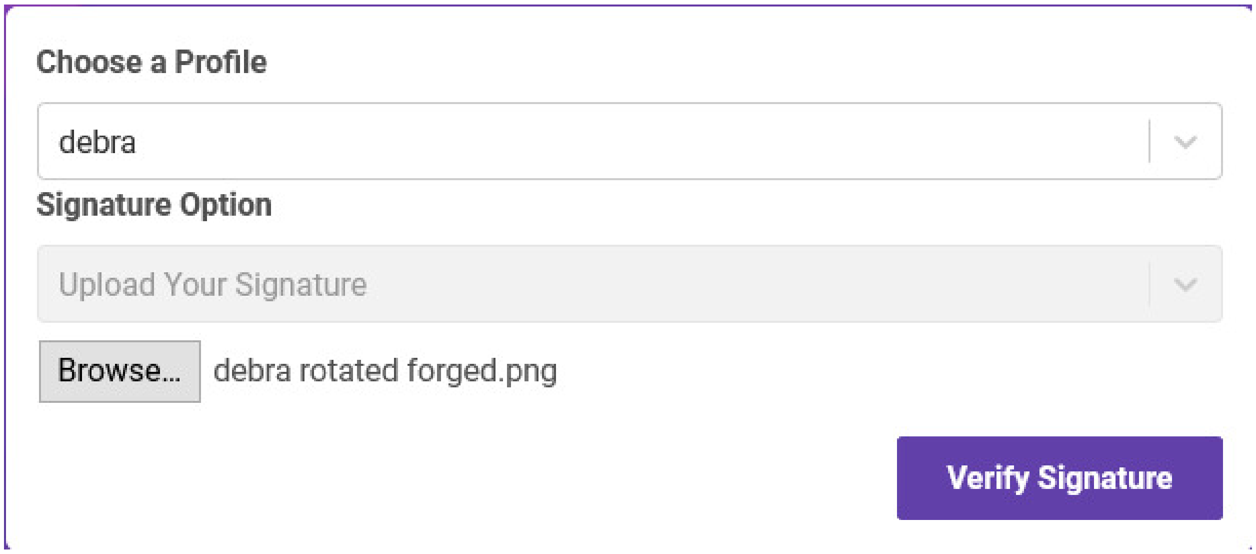


Figure 61. Demonstration of verifying a forged signature Rotated By 15^0*of user Debra.*

➢ Result of verifying with a forged signature rotated by 15^0 of user Debra, shown in figure 62:

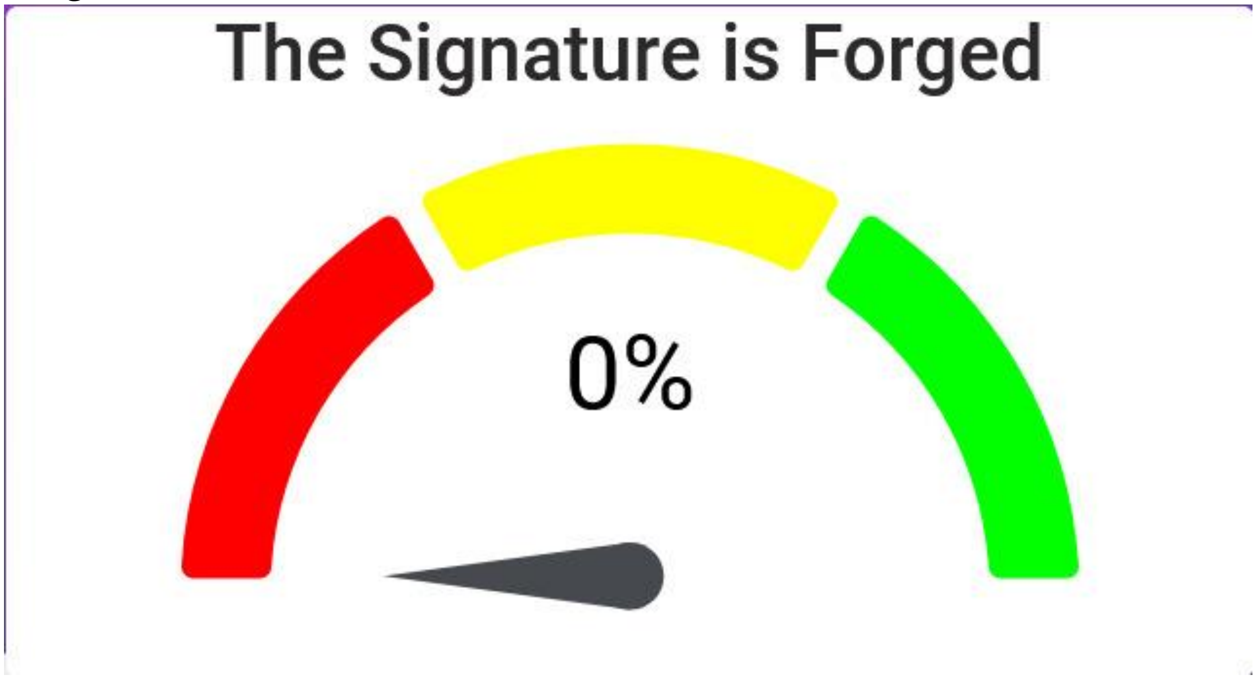
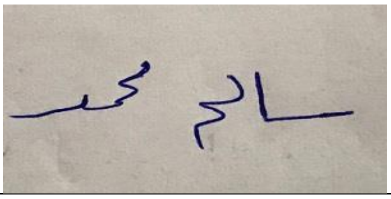
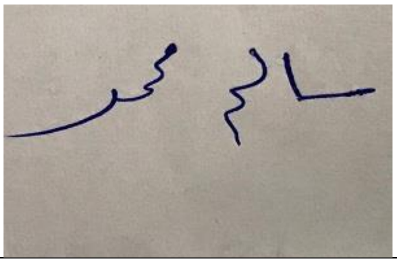
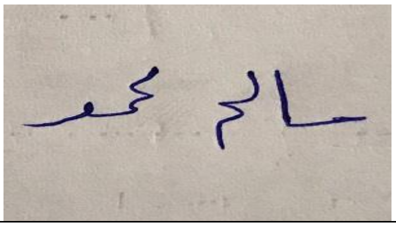
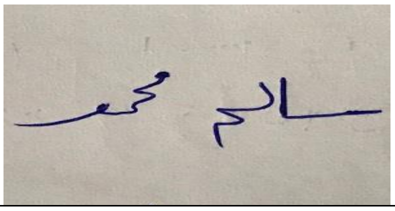
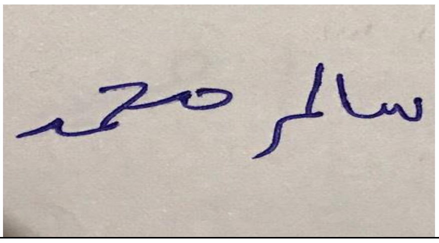

The Signature is Forged

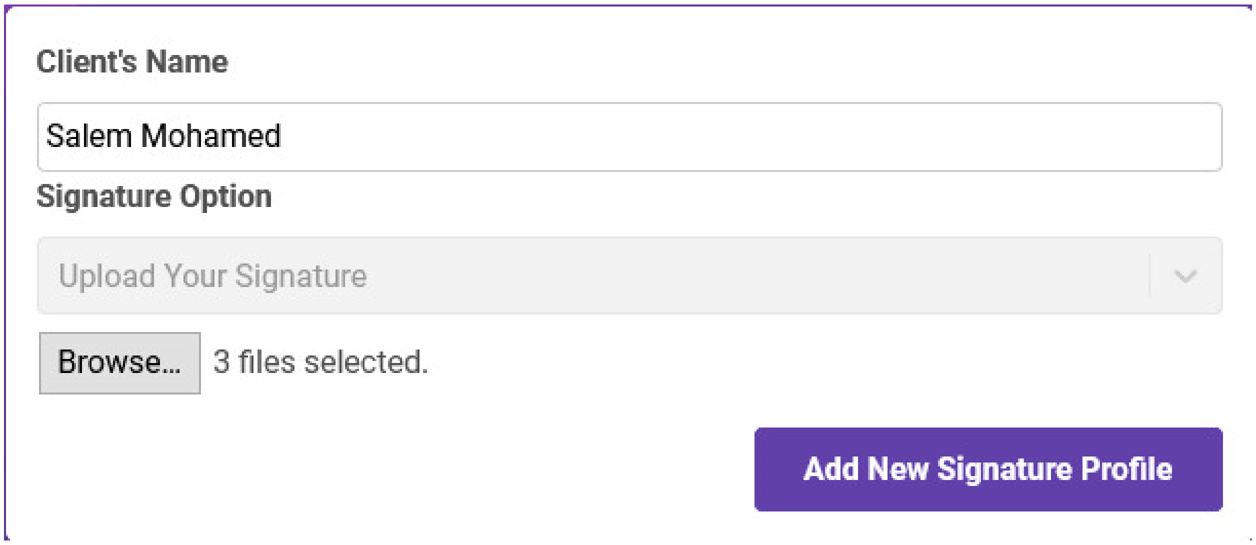
Figure 62. Demonstration of result of verifying a Forged signature Rotated By 15^0 *of user Debra.*

- **Test Case 3: Testing Arabic Signature for a user “Salem Mohamed”**

Table 19. signatures of user Salem Mohamed used in test case 3

Signature Number	Signature Image	Signature Type
1		Original
2		Original
3		Original
4		Genuine
5		Forged

1) Creating a new signature profile for a user called “Salem Mohamed” as shown in figure 63 and figure 64:



Client's Name

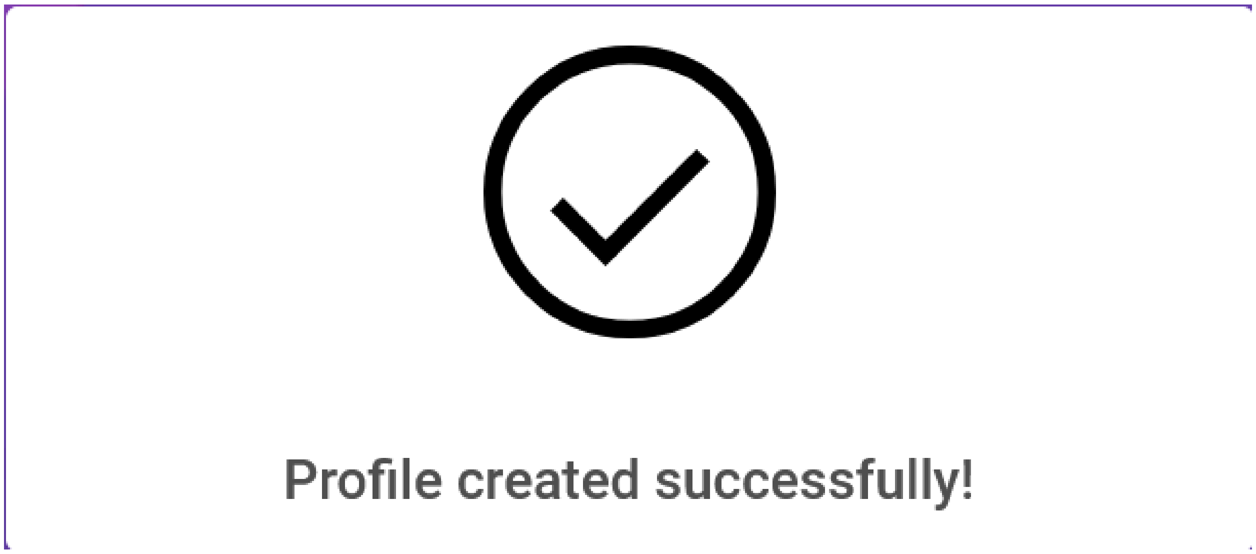
Salem Mohamed

Signature Option

Upload Your Signature

Browse... 3 files selected.

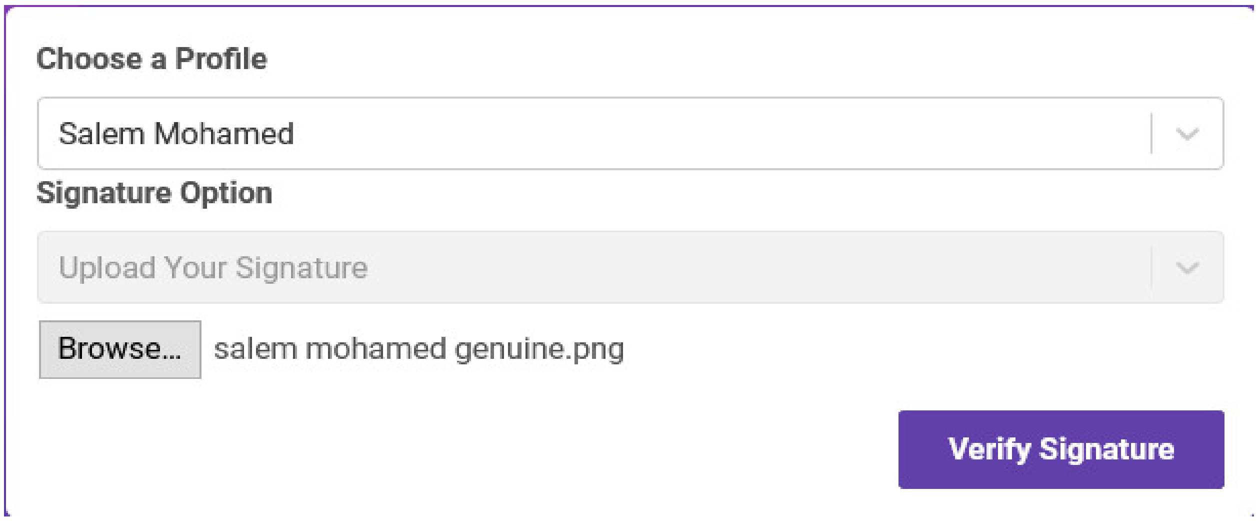
Figure 63. Demonstration of creating new signature profile for new user Salem Mohamed.



Profile created successfully!

Figure 64. Demonstration of successful creation message

2) Verifying a genuine signature of user Salem Mohamed as shown in figure 65:


Choose a Profile

Salem Mohamed

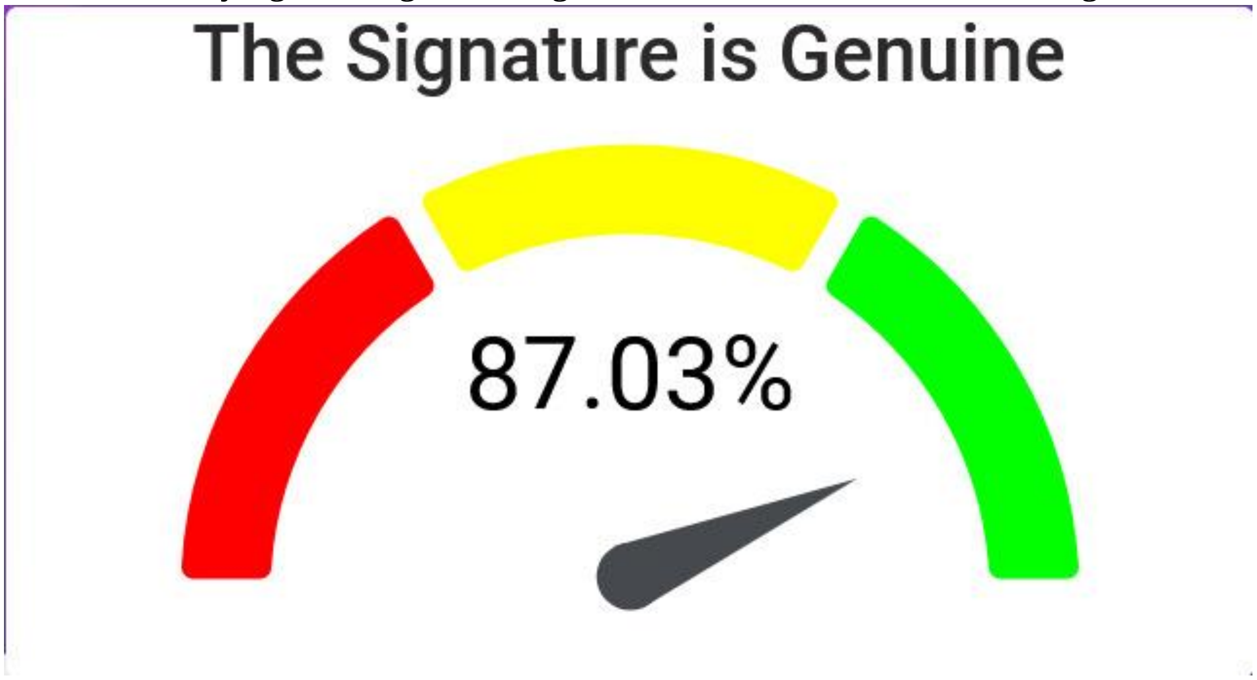
Signature Option

Upload Your Signature

Browse... salem mohamed genuine.png

Figure 65. Demonstration of verifying a genuine signature of user Salem Mohamed.

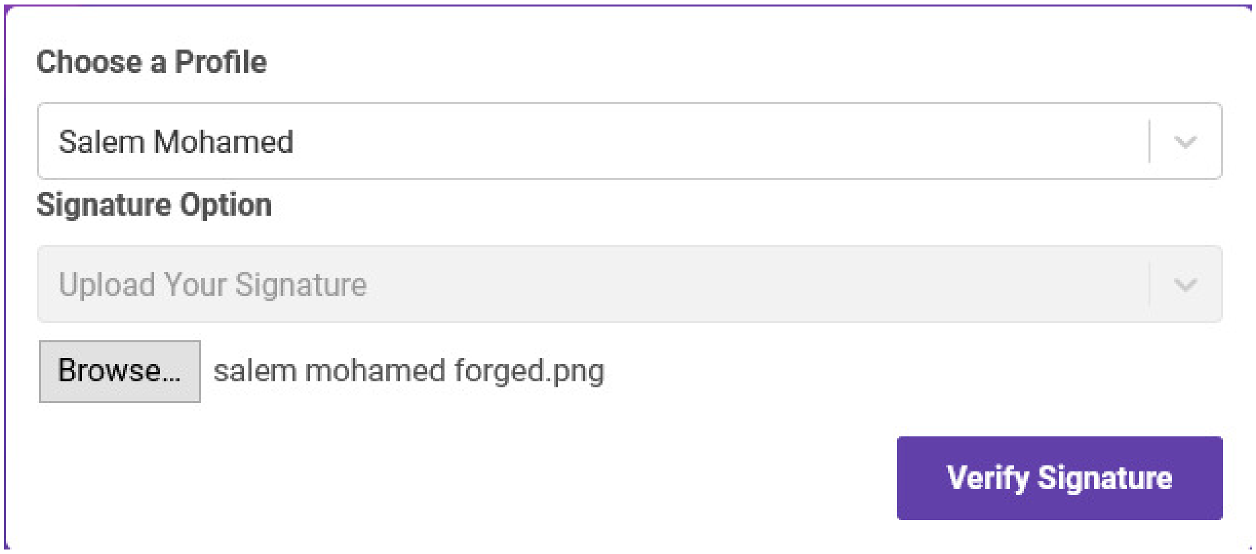
➢ Result of verifying with a genuine signature of user Salem Mohamed, figure 66:


The Signature is Genuine

87.03%

Figure 66. Demonstration of result of verifying a genuine signature of user Salem Mohamed.

3) Verifying forged signature of user Salem Mohamed as shown in figure 67:


Choose a Profile

Salem Mohamed

Signature Option

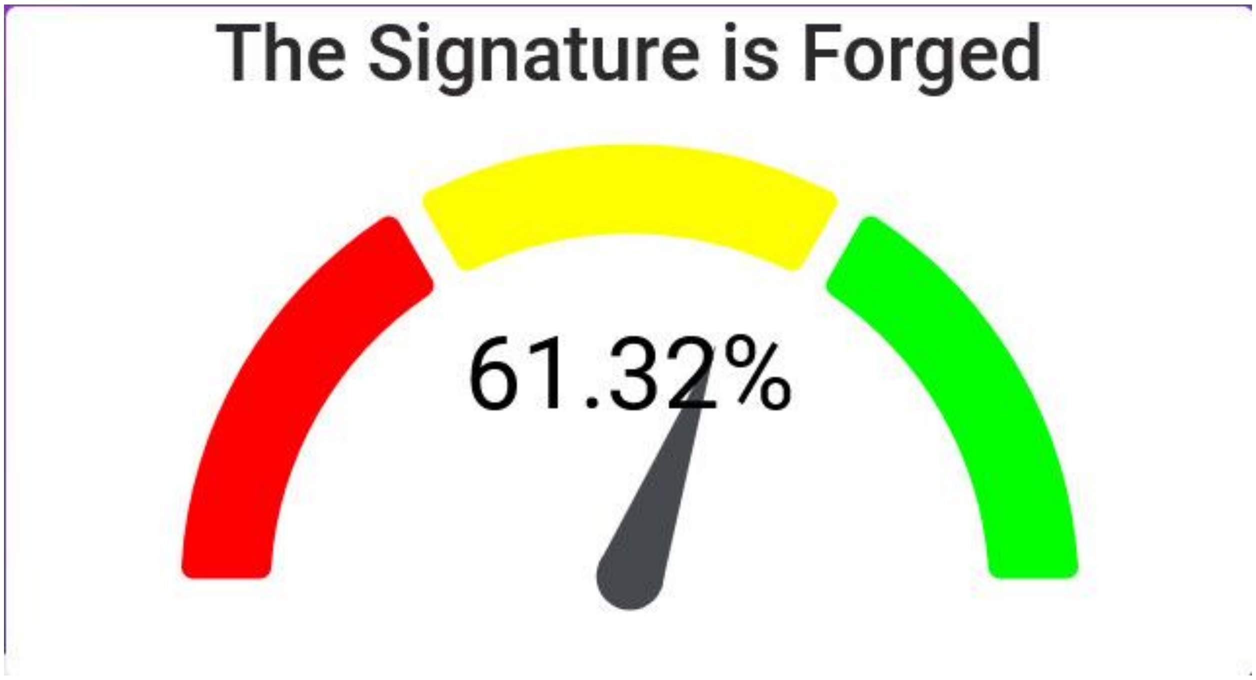
Upload Your Signature

Browse... salem mohamed forged.png

Verify Signature

Figure 67. Demonstration of verifying a forged signature of user Salem Mohamed.

➢ Result of verifying with a forged signature of user Salem Mohamed, shown in figure 68 :

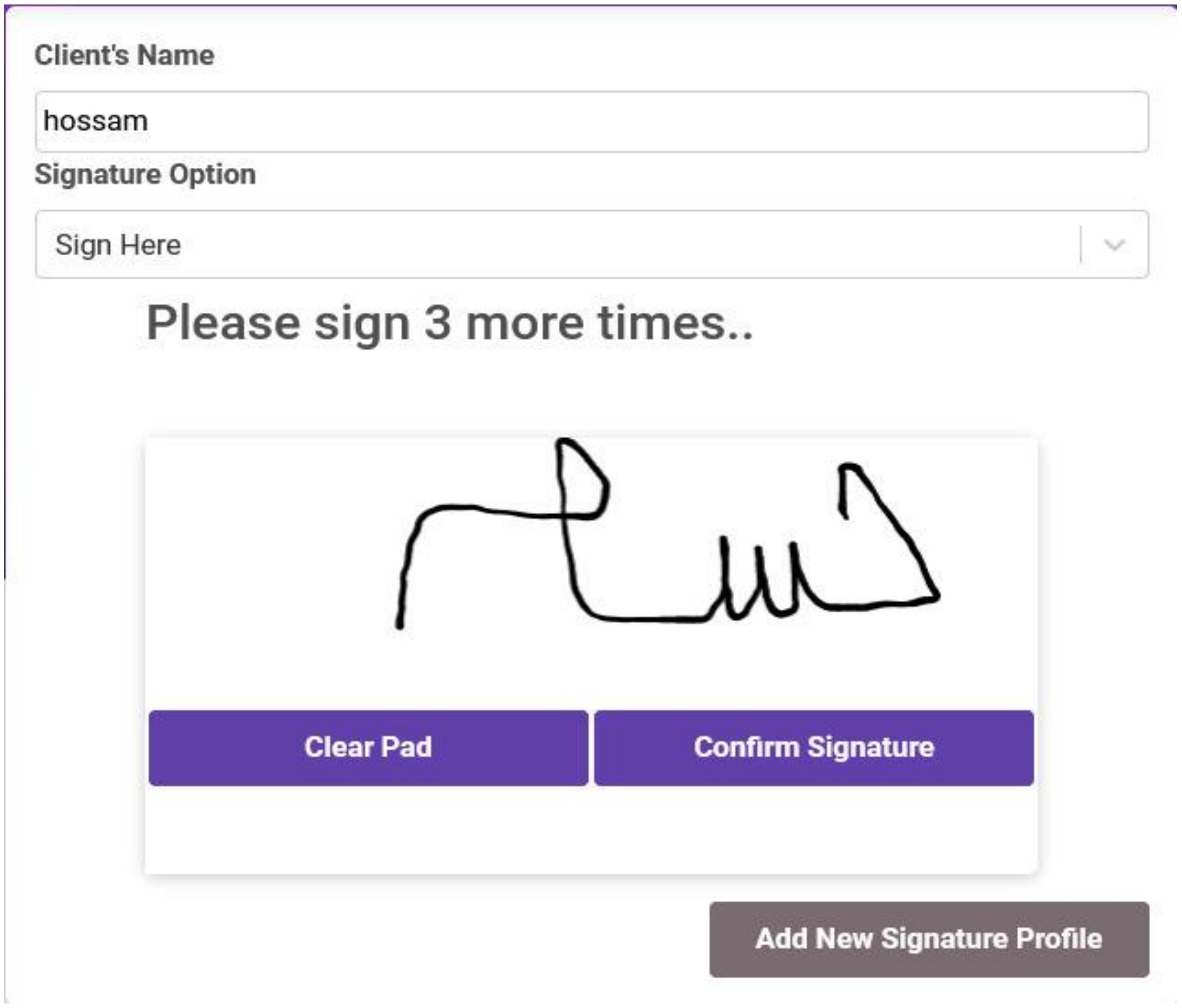

The Signature is Forged

61.32%

Figure 68. Demonstration of result of verifying a Forged signature of user Salem Mohamed.

- **Test Case 4: Testing Using the Signature Pad feature**

1) Creating a new signature profile with signature pad, using a mouse to capture the signature of a new user called “Hossam” as shown in figure 69, figure 70, figure 71, figure 72 and figure 73:



Client's Name

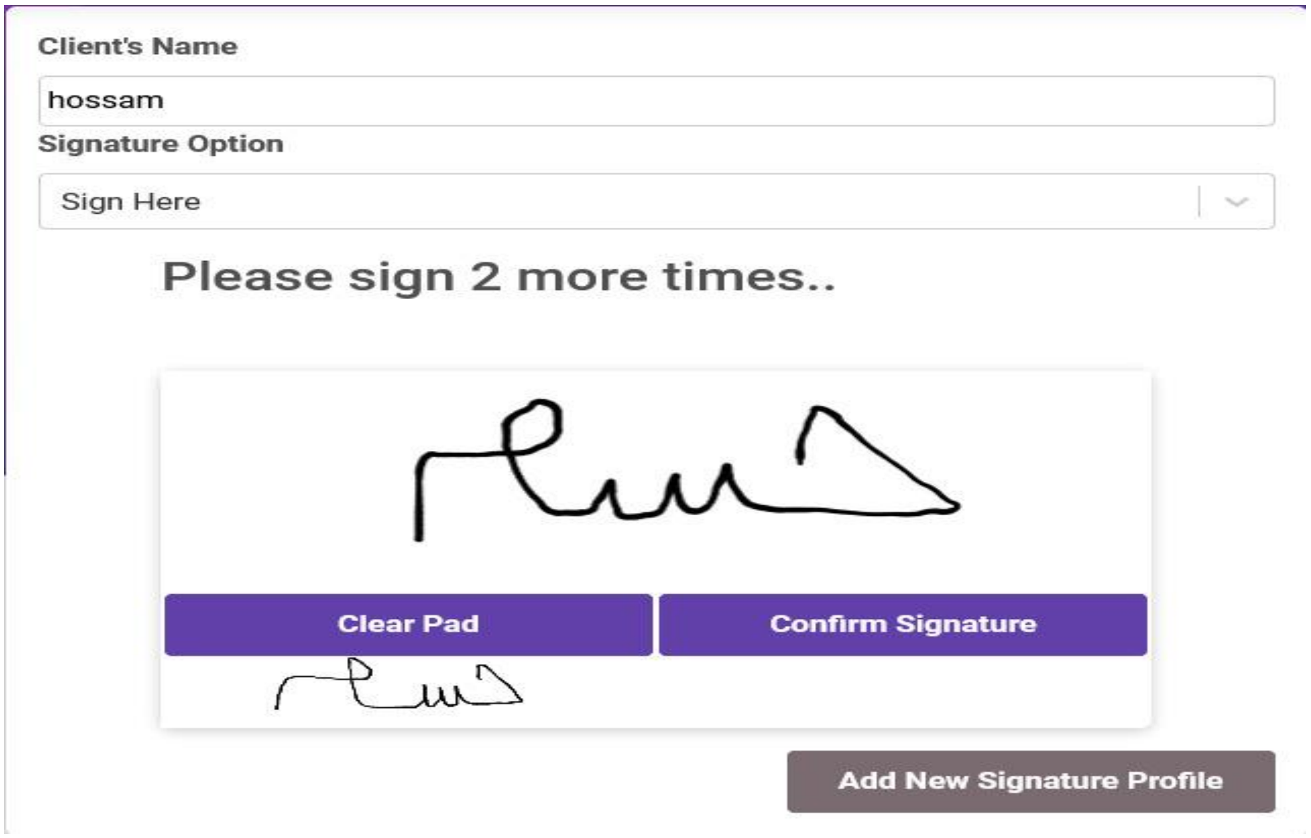
hossam

Signature Option

Sign Here

Please sign 3 more times..

Figure 69. Demonstration of capturing signature#1 of new user Hossam.



Client's Name

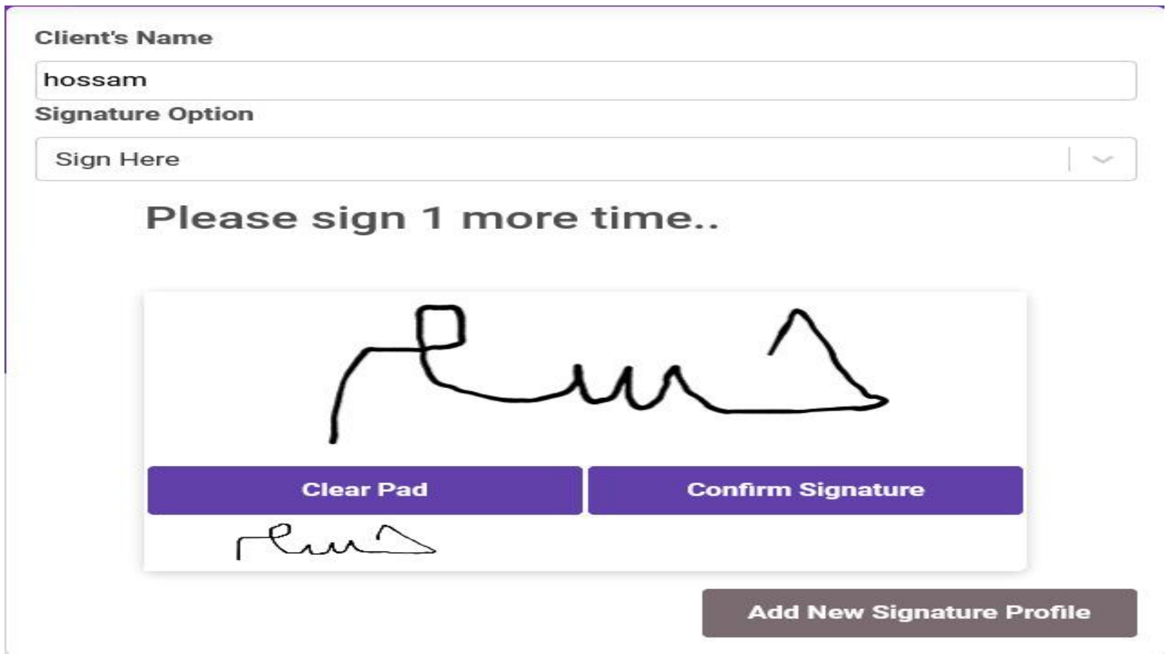
hossam

Signature Option

Sign Here

Please sign 2 more times..

Figure 70. Demonstration of capturing signature#2 of new user Hossam.



Client's Name

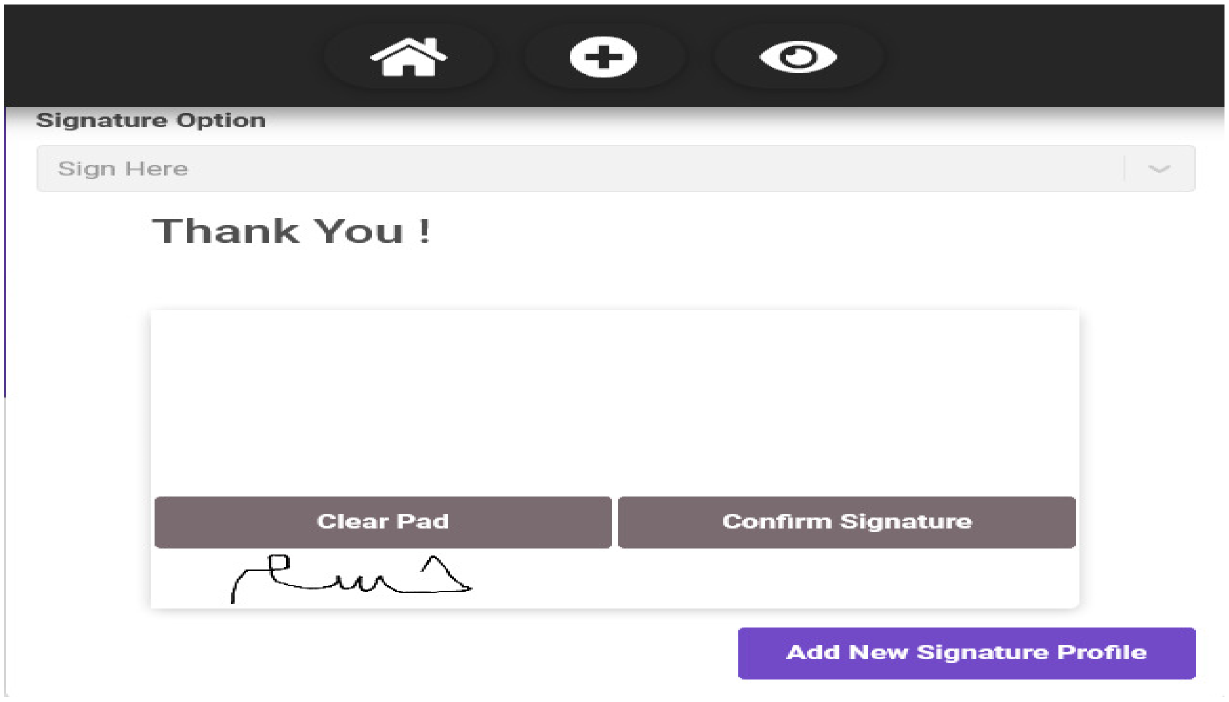
hossam

Signature Option

Sign Here

Please sign 1 more time..

Figure 71. Demonstration of capturing signature#3 of new user Hossam

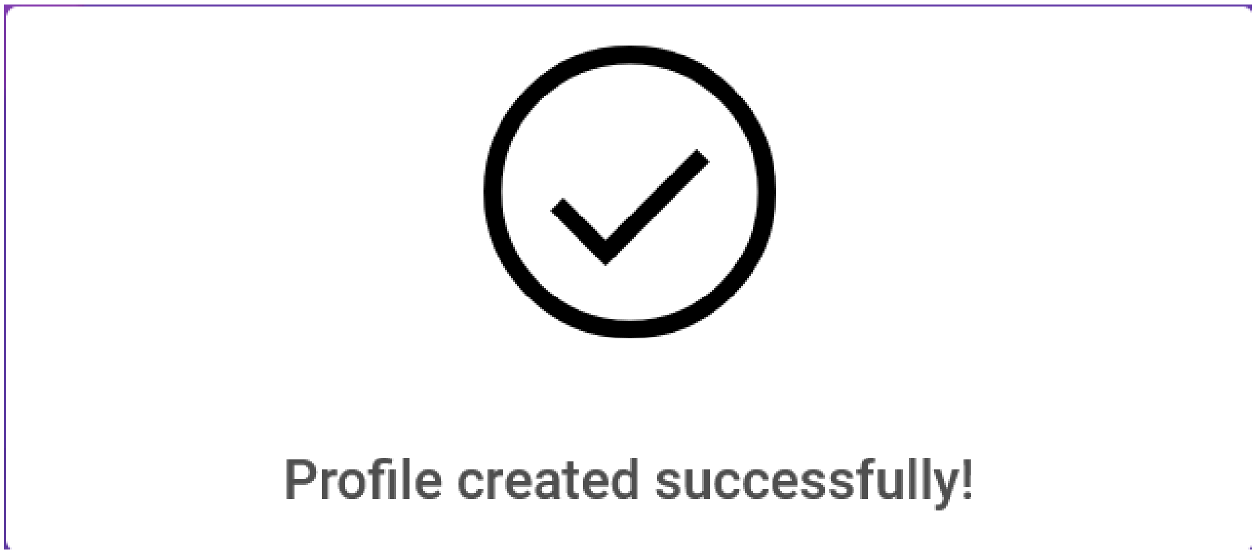


Signature Option

Sign Here

Thank You !

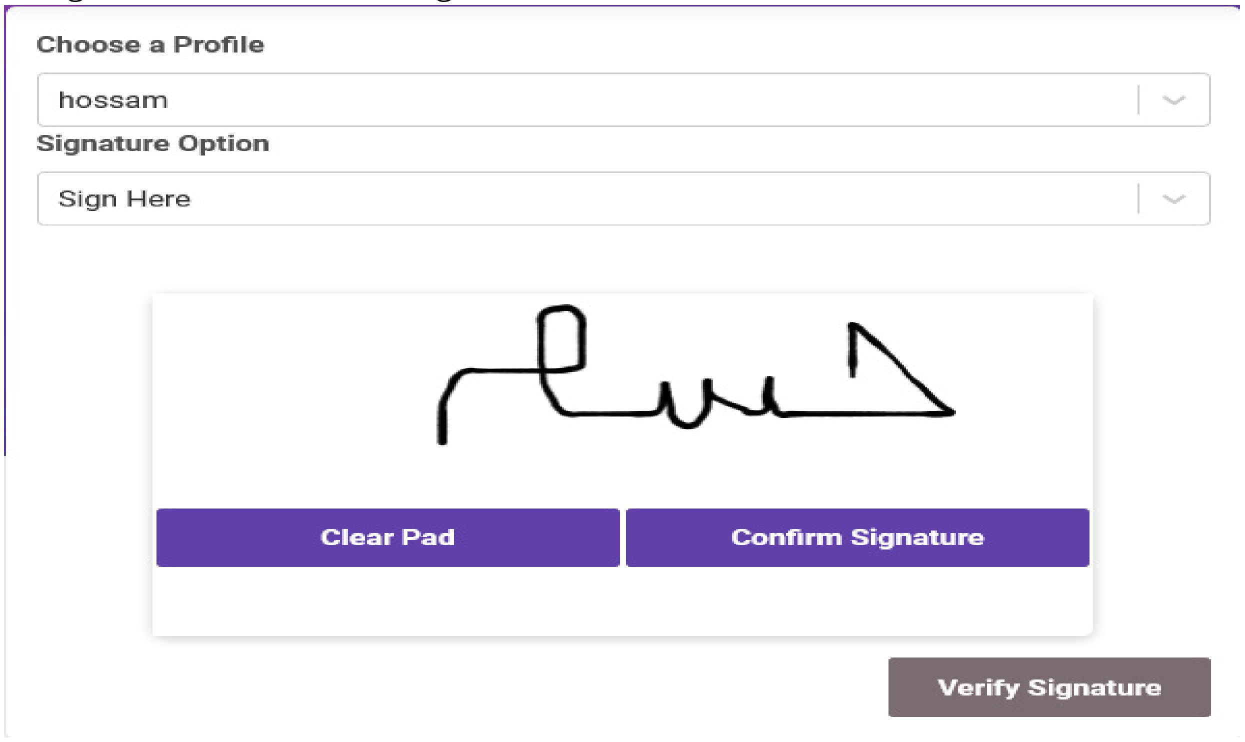
Figure 72. Demonstration of successfully capturing 3 signatures of user Hossam.



Profile created successfully!

Figure 73. Demonstration of successful creation message

2) Verifying a genuine signature of user Hossam captured with signature pad using a mouse as shown in figure 74:


Choose a Profile

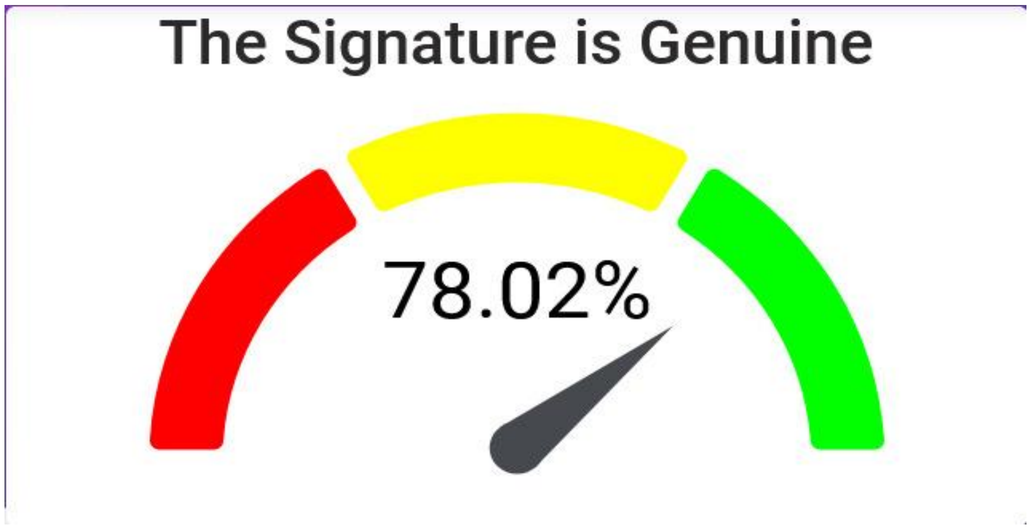
hossam

Signature Option

Sign Here

Figure 74. Demonstration of verifying a genuine signature of user Hossam captured with signature pad

➢ Result of verifying with a genuine signature of user Hossam captured with signature pad using a mouse, as show in figure 75:


The Signature is Genuine

78.02%

Figure 75. Demonstration of result of verifying a genuine signature of user Hossam captured with signature pad

3) Verifying forged signature of user Hossam captured with signature pad using a mouse as shown in figure 76:

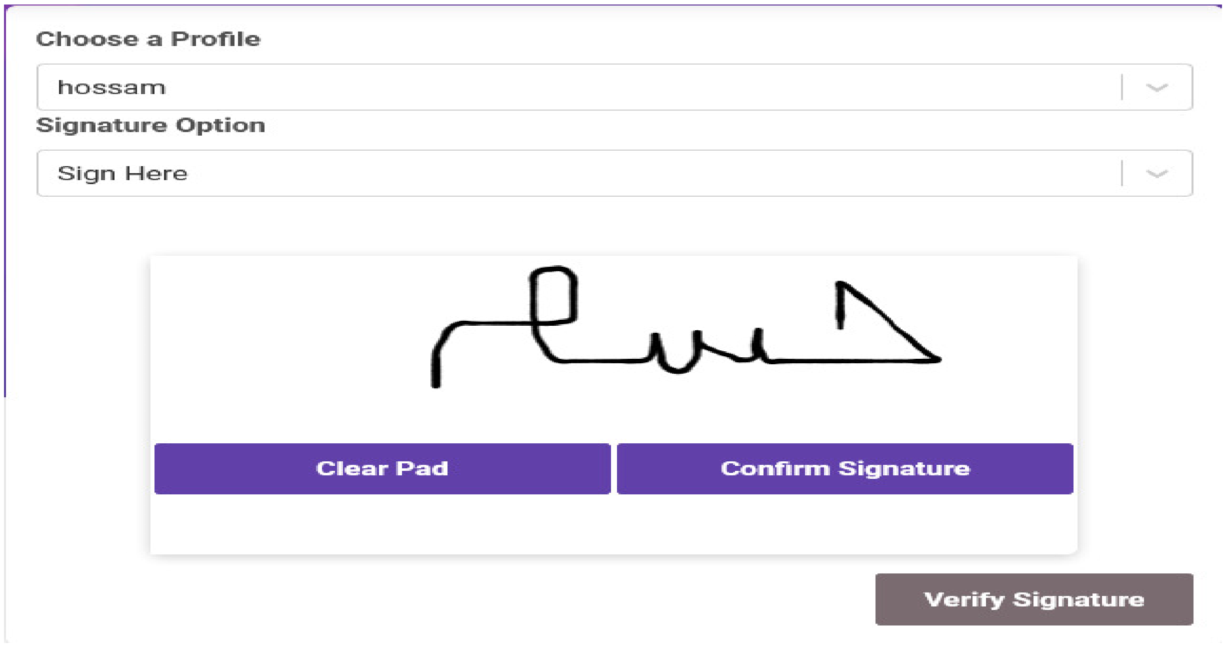


Figure 76. Demonstration of verifying a forged signature of user Hossam captured with signature pad

2. Result of verifying with a forged signature of user captured with signature pad using a mouse, shown in figure 77:

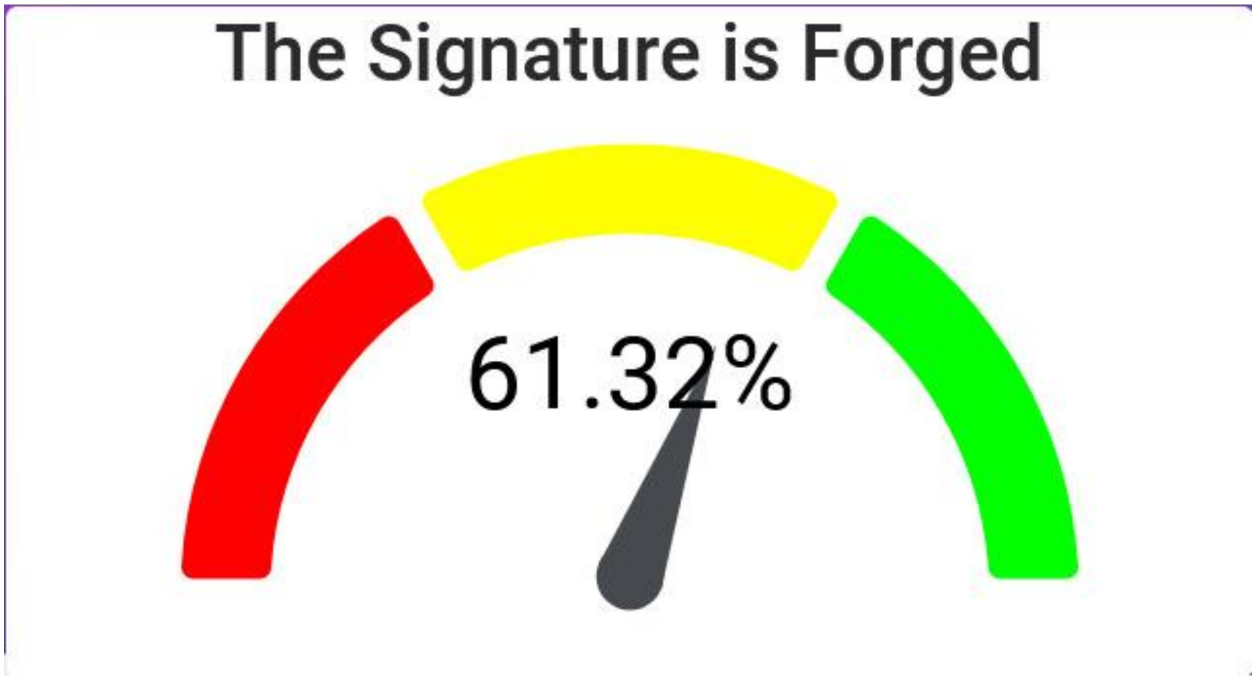


Figure 77. Demonstration of result of verifying a forged signature of user Hossam captured with signature pad

Chapter 5: Conclusion and Future Work

5.1) Conclusion

Signature authentication machine is implemented to provide a simple, safe, fast biometric behavioral security system. An off-line handwritten signature verification model have been purposed by using single known sample and based on a deep CNN network. The reliability of the experimental results have been ensured through a series of methods, including preprocessing (removing background noise), designing controlled groups for different sample sizes and network architectures, and applying visualization techniques (to provide interpretability of the model). The experimental results indicate that it is possible to perform automatic signature verification by single sample. The model can learn some useful features to discriminate among different signatures and authors. Even under the unfavorable conditions of small sample size, there is still a relatively high accuracy rate between 89.5% and 99.96%. The Interface of this application is very simple which makes it user-friendly and easy to use.

5.2) Future Work

- Deploy INKuisitor website application and publish it on the internet.
- Add a feature to take the signature using desktop or laptop camera.
- Train the model on more datasets and increase the accuracy.
- Make the model more adaptable to highly rotated signatures.
- Increase the Database security.
- Develop application into a native mobile application.

References

i. Offline Signature Verification Using Convolutional Neural Network (Undergraduate Project), by Nitu Shrestha, Pradeep Sharma Ghimire, Samita Neupane and Suresh Pokharel KATHFORD INTERNATIONAL COLLEGE OF ENGINEERING AND MANAGEMENT , Balkumari, Lalitpur.

ii. Siamese Neural Networks for One-shot Image Recognition, by Gregory Koch , Richard Zemel and Ruslan Salakhutdinov, Department of Computer Science, University of Toronto. Toronto, Ontario, Canada.

iii. SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification, by Sounak Dey, Anjan Dutta, J. Ignacio Toledo, Suman K.Ghosh, Josep Llad´os and Umapada Palb, Computer Vision Center, Computer Science Dept., Universitat Aut`onoma de Barcelona, Edifici O, Campus UAB, 08193 Bellaterra, Spain, Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, 203, B. T. Road, Kolkata-700108, India.

iv. Signature Recognition and Verification with ANN, by Cemil OZ Sakarya University Computer Eng. Department Sakarya Turkey, Fikret Ercal UMR Computer Science Department, Rolla, MO 65401 and Zafer Demir Sakaraya University electric electronic eng. Department sakarya , Turkey.

v. Signature Matching for Bank System using Neural Network, by 1 Ashwini Sharma and 2 P. R. Thorat , 1 M.E. Student, Department of Electronics, Savitribai Phule Womens Engineering College, Aurangabad, India, 2 Assistant Professor, Dept. of Electronics, Savitribai Phule Womens Engineering College, Aurangabad, India, International Journal of Research in Engineering, Science and Management, Volume-2, Issue-5, May-2019.

vi. https://towardsdatascience.com/one-shot-learning-with-siamese-networks-using-keras-17f34e75bb3d

vii. https://towardsdatascience.com/how-to-train-your-siamese-neural-network-4c6da3259463

viii. https://amaarora.github.io/2020/08/02/densenets.html

ix. https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

x. https://docs.opencv.org/4.5.2/d4/d73/tutorial_py_contours_begin.html

xi. https://docs.opencv.org/4.5.2/d9/d61/tutorial_py_morphological_ops.html

xii. https://en.wikipedia.org/wiki/Single-page_application

xiii. https://reactjs.org/

xiv. https://reactrouter.com/

xv. https://restfulapi.net/