Global Institute Of Management Sciences

Fingerprint Spoof Detector Generalization

SUBMITTED BY :
Sandeep Chandra Sagar R
20XHSB7051

Under the Guidance of :
Prof. Sumathi G K
(Dept. of Computer Science)

Contents

1. Introduction
2. Existing System
3. Proposed System
4. Methodology
5. System Architecture
6. DFD Diagrams
7. UML Diagrams
8. Module Description/ Implementation
9. Testing
10. Screenshots
11. System Requirements
12. Future Enhancements
13. Conclusion
14. References

Introduction

Biometrics aims to reliably identify individuals based on physical or behavioral traits like fingerprints, face, iris, voice, palm, or signature. It offers advantages over traditional security methods but can be vulnerable to fake biometric inputs.

Fingerprint systems, for example, can be easily spoofed using materials like gelatine or wood glue. To address this, two approaches are used: hardware-based methods involve adding devices to detect living traits, while software-based methods detect fake traits after acquiring samples with a standard sensor. Biometric recognition systems are widely used in various sectors due to their convenience and accuracy.

However, they are susceptible to direct attacks (e.g., using simple tools on the sensor) and indirect attacks (requiring deep knowledge of the system). To combat these attacks, researchers have been developing liveness detection systems for fingerprint biometrics.

Existing System

Traditional authentication such as passwords, personal identification numbers, smart cards were largely unable to meet convenience, reliability and security requirements in a wide variety of applications.

DISADVANTAGES

- Complexity
- Cost
- Security Risk

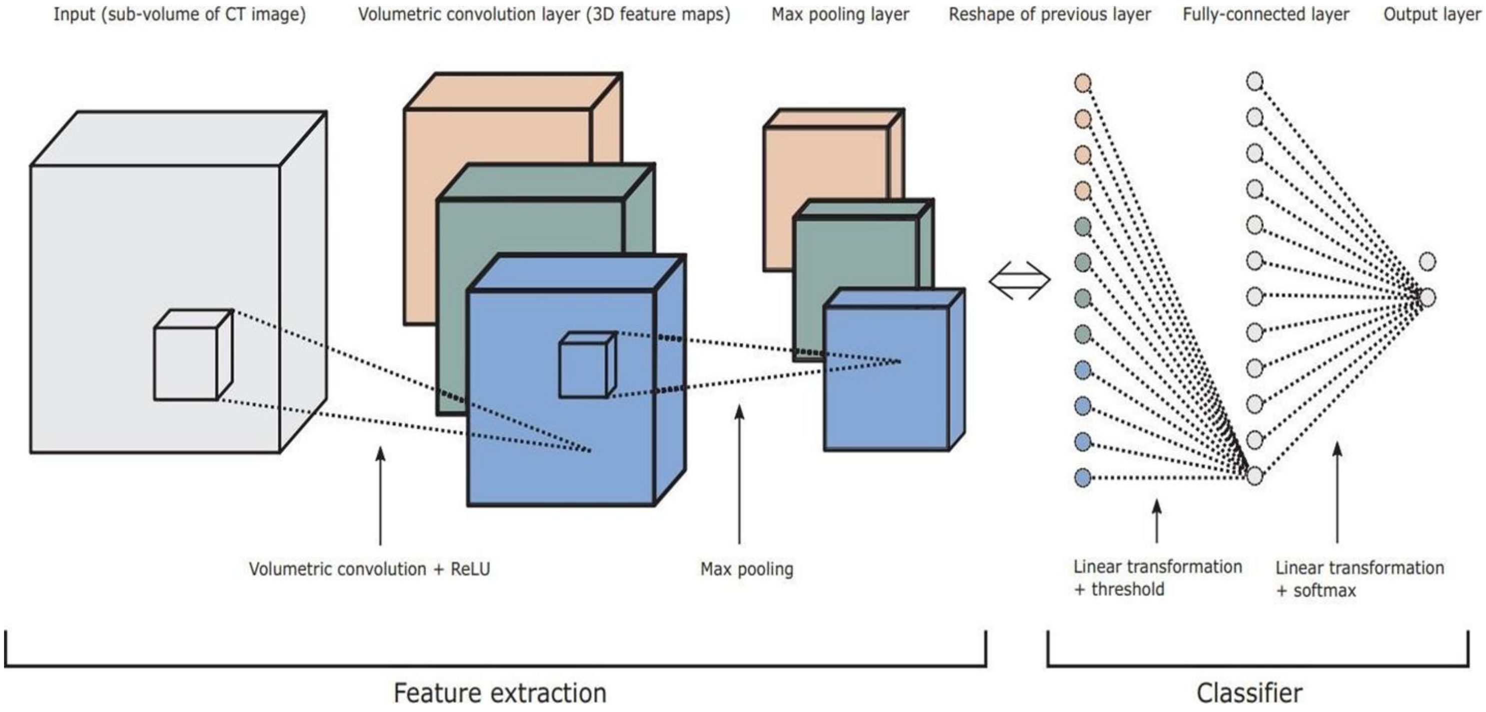
Proposed System

The proposed work on fingerprint spoofing detection and identification has utilized a traditional Convolutional Neural Network model. However, due to the limited fingerprint database available, the work has implemented a transfer learning model. CNNs consist of various layers, including convolutional layers, pooling layers, and fully connected layers.

ADVANTAGES

- Improved Cryptography
- Faster Calculations
- Better Simulations

System Architecture



Methodology

Data Set:

The dataset for training is obtained from Fingerprint spoofing Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI).

Image Segmentation:

The segmentation of photographs is the phase where the visual image is partitioned into several parts. This normally helps to identify artifacts and boundaries. The aim of segmentation is to simplify the transition in the interpretation of a picture into the concrete picture that can be clearly interpreted and quickly analyzed.

Methodology

Pre-Processing:

In preprocessing stage, the median filter is used to restore the image under test by minimizing the effects of the degradations during acquisition. Various preprocessing and segmentation techniques of lung nodules are discussed in. The median filter simply replaces each pixel value with the median value of its neighbors including itself. Hence, the pixel values which are very different from their neighbors will be eliminated.

Convolutional Neural Networks:

A CNN is type of a DNN consists of multiple hidden layers such as convolutional layer, RELU layer, Pooling layer and fully connected a normalized layer. CNN shares weights in the convolutional layer reducing the memory footprint and increases the performance of the network.

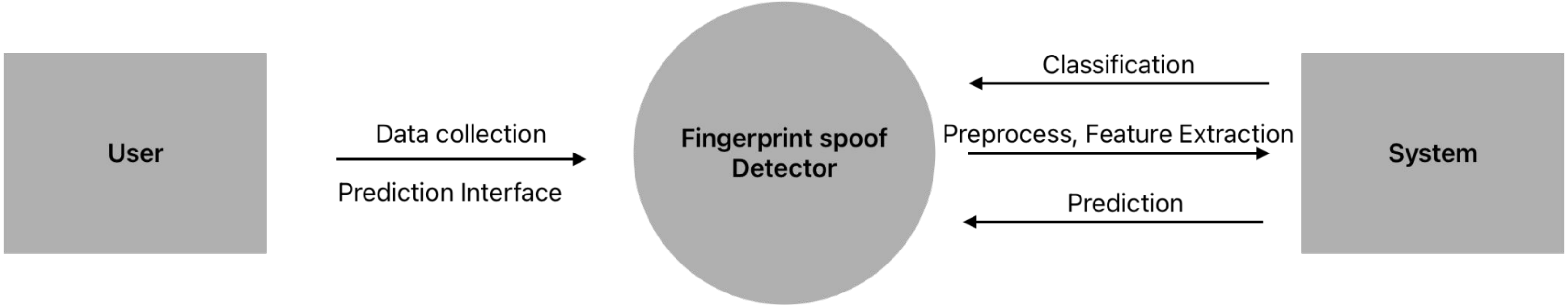
System Architecture Explanation

- Deep-CNN is type of a DNN consists of multiple hidden layers such as convolutional layer, RELU layer, Pooling layer and fully connected a normalized layer.

- The important features of CNN lie with the 3D volumes of neurons, local connectivity and shared weights.

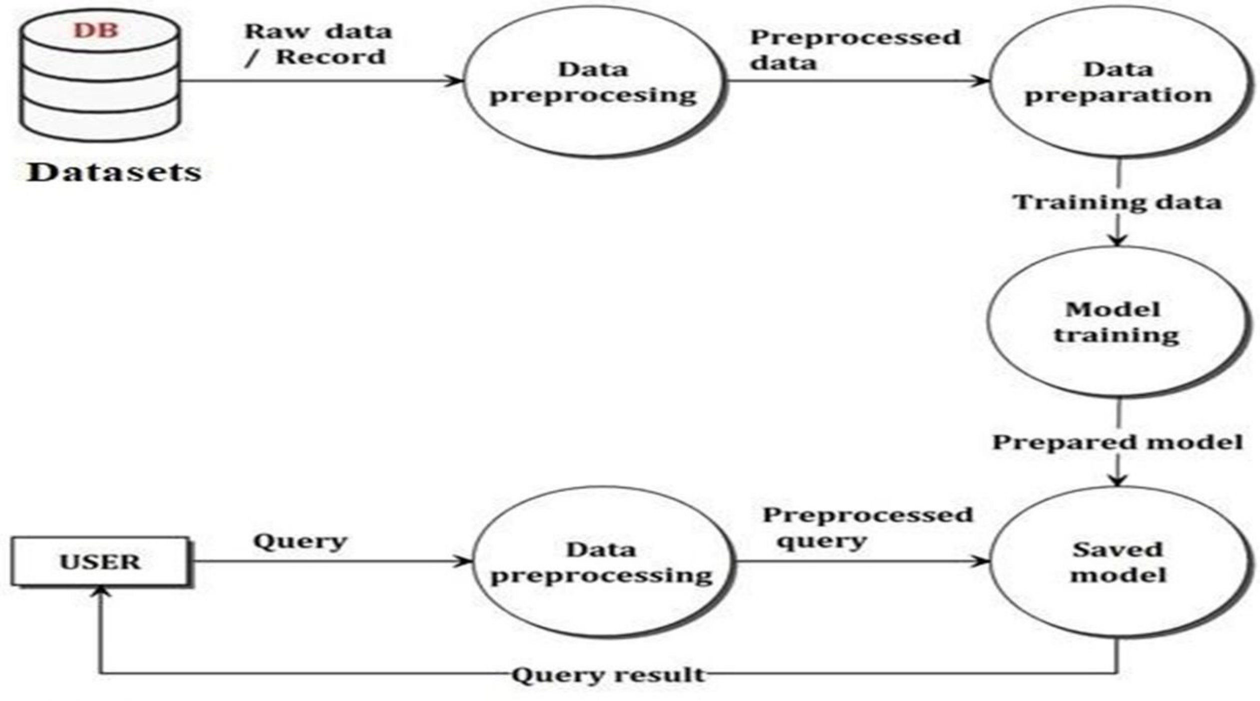
- This results a large reduction in the sample size. Sometimes, traditional Fully-Connected (FC) layer will be used in conjunction with the convolutional layers towards the output stage. In CNN architecture, usually convolution layer and pool layer are used in some combination.

DFD Diagram - Level 0



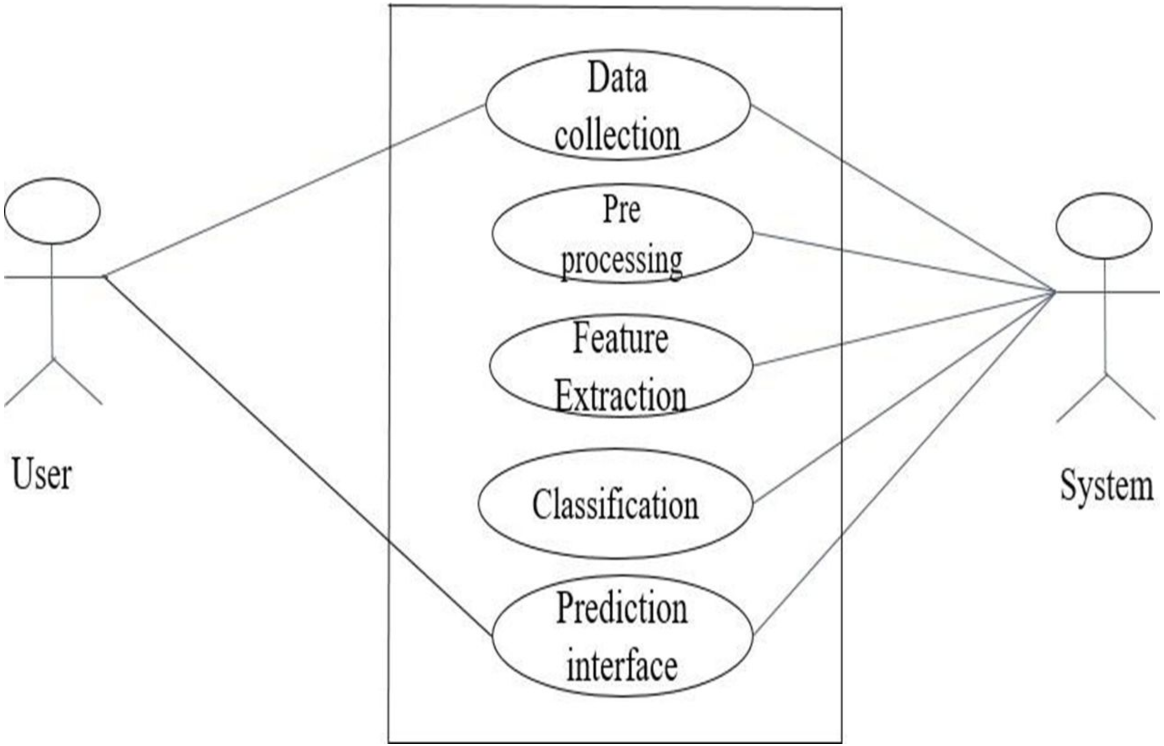
0 - LEVEL DFD

DFD Diagram - Level 1



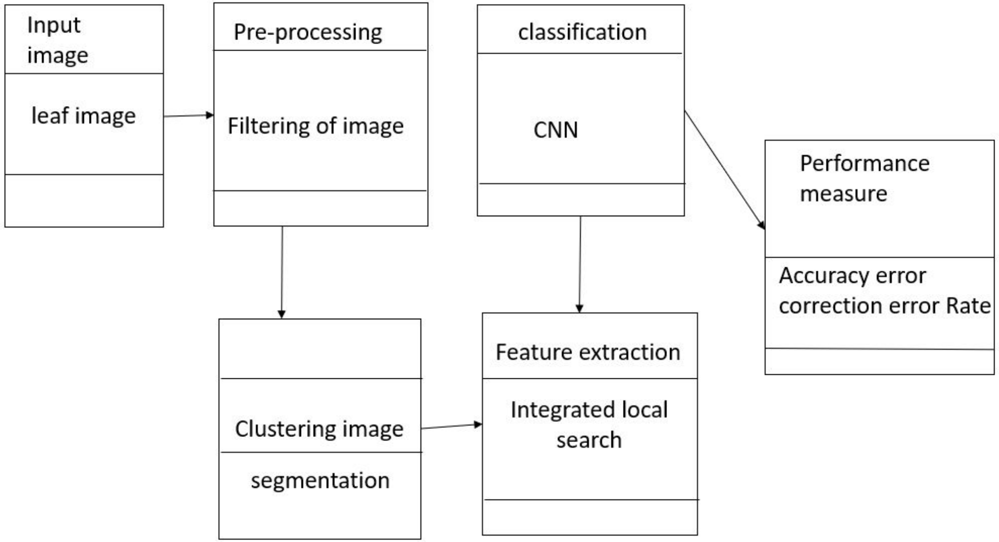
DFD-L1

Use-Case Diagram



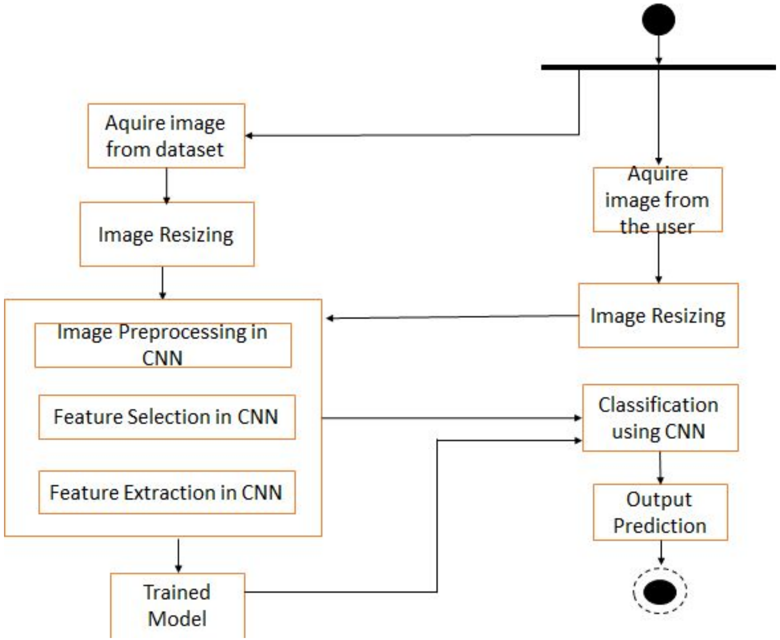
Class Diagram

Class diagrams are the main building block in object-oriented modeling. They are used to show the different objects in a system, their attributes, their operations and the relationships among them



Activity Diagram

An Activity diagram describes the dynamic aspects of the system. It is a flowchart that describes the flow from one activity to another.



Module Specification

Module Specification is the way to improve the structural design by breaking down the system into modules and solving it as an independent task. By doing so the complexity is reduced and the modules can be tested independently.

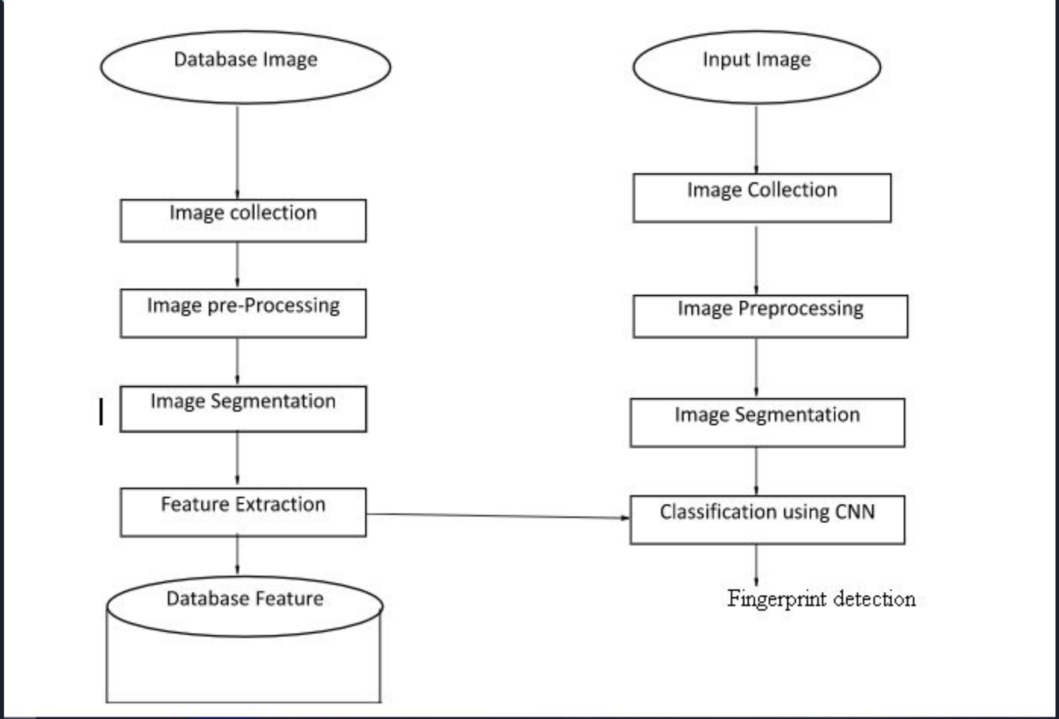
The number of modules for our model is three, namely pre- processing, identification, feature extraction and detection.

So each phase signify the functionalities provided by the proposed system. In the data pre-processing phase noise removal using median filtering is done.

Module specification

Train Phase

Test Phase



The System design mainly consists of

1. Image Collection
2. Image Preprocessing
3. Image Segmentation
4. Feature Extraction
5. Training
6. Classification

Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Types of testing

Software testing methods and traditionally divided into two: white-box and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

a) **White-box testing** (also known as clear box testing, glass box testing, transparent box testing and structural testing, by seeing the source code) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user.

Testing

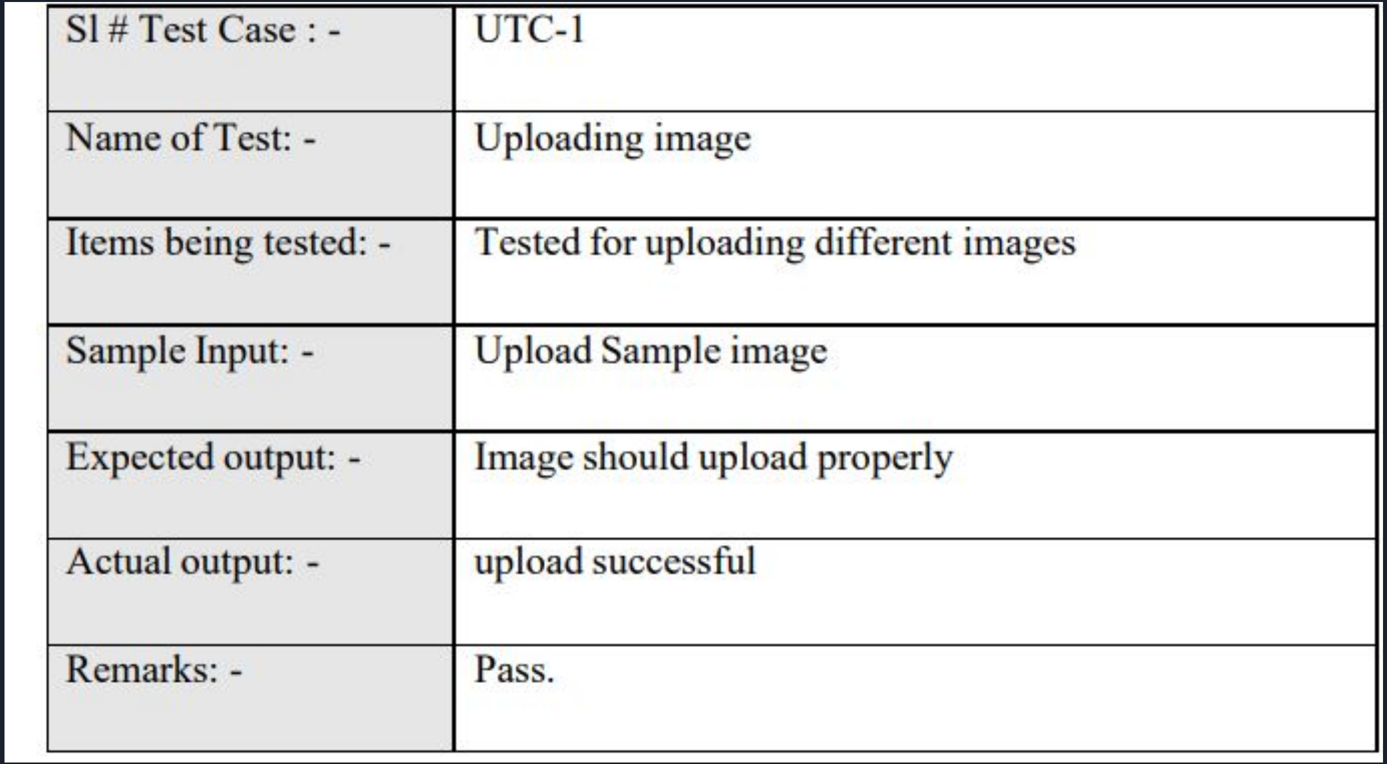
In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases.

a) **Black box testing**: The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code.

Unit Testing

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application.

Unit Testing - Test Case 1

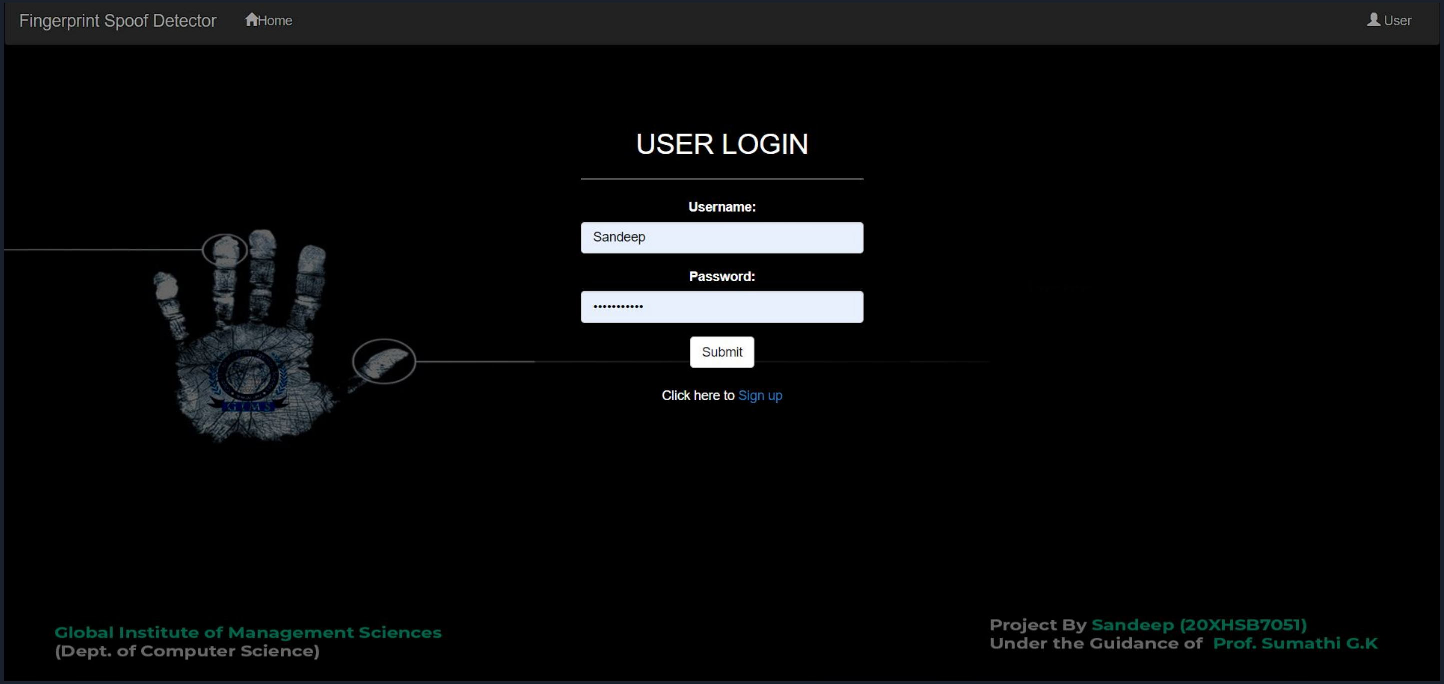


Sl # Test Case : -	UTC-1
Name of Test: -	Uploading image
Items being tested: -	Tested for uploading different images
Sample Input: -	Upload Sample image
Expected output: -	Image should upload properly
Actual output: -	upload successful
Remarks: -	Pass.

Unit Testing - Test Case 2

S1 # Test Case:	**UTC-2**
Name of Test	Detecting Thumb images
Items being tested	Test for different Thumb images
Sample Input	Tested for different images of Thumb
Expected output	Thumb should be displayed
Actual output	Should Display Live or Fake
Remarks	Predicted result

Implementation - Screenshots



USER LOGIN

Username: Sandeep

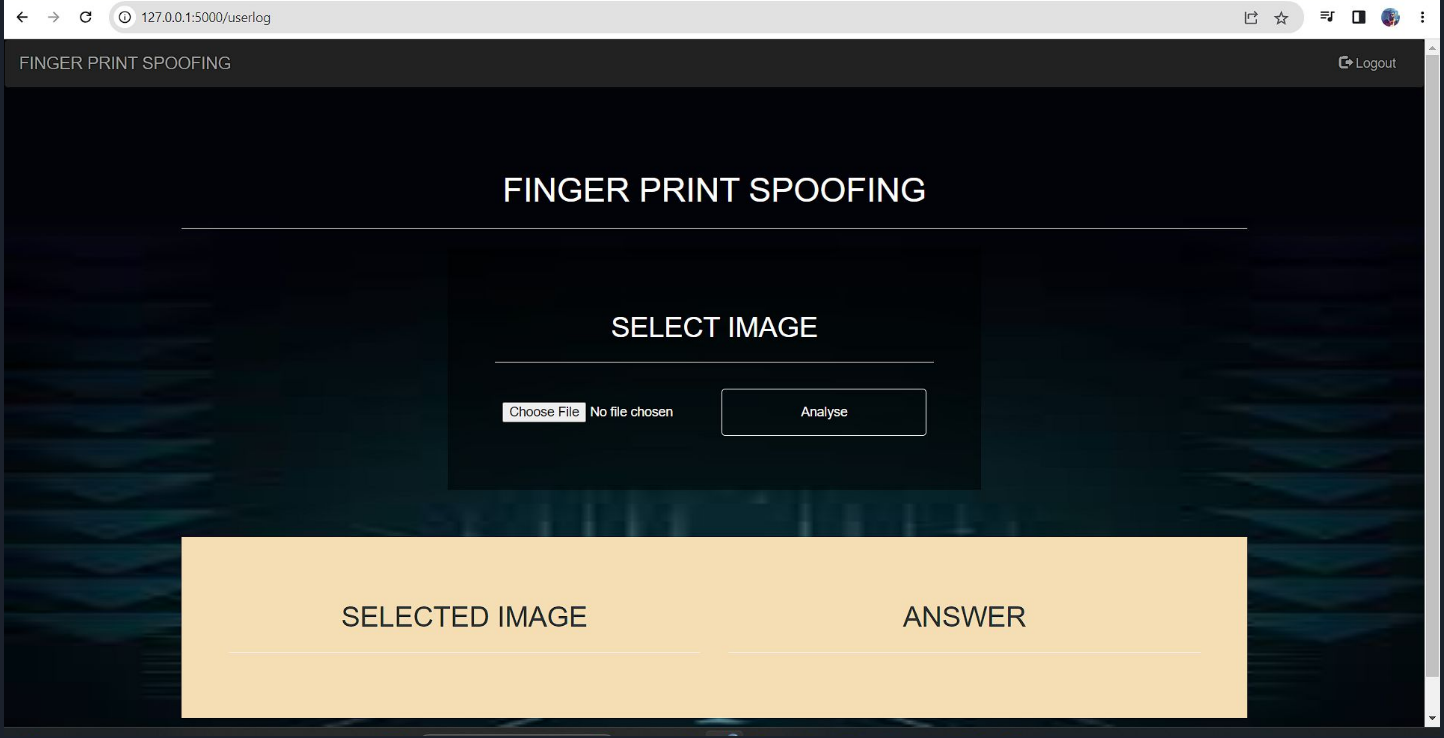
Password: •••••••••••

Click here to Sign up

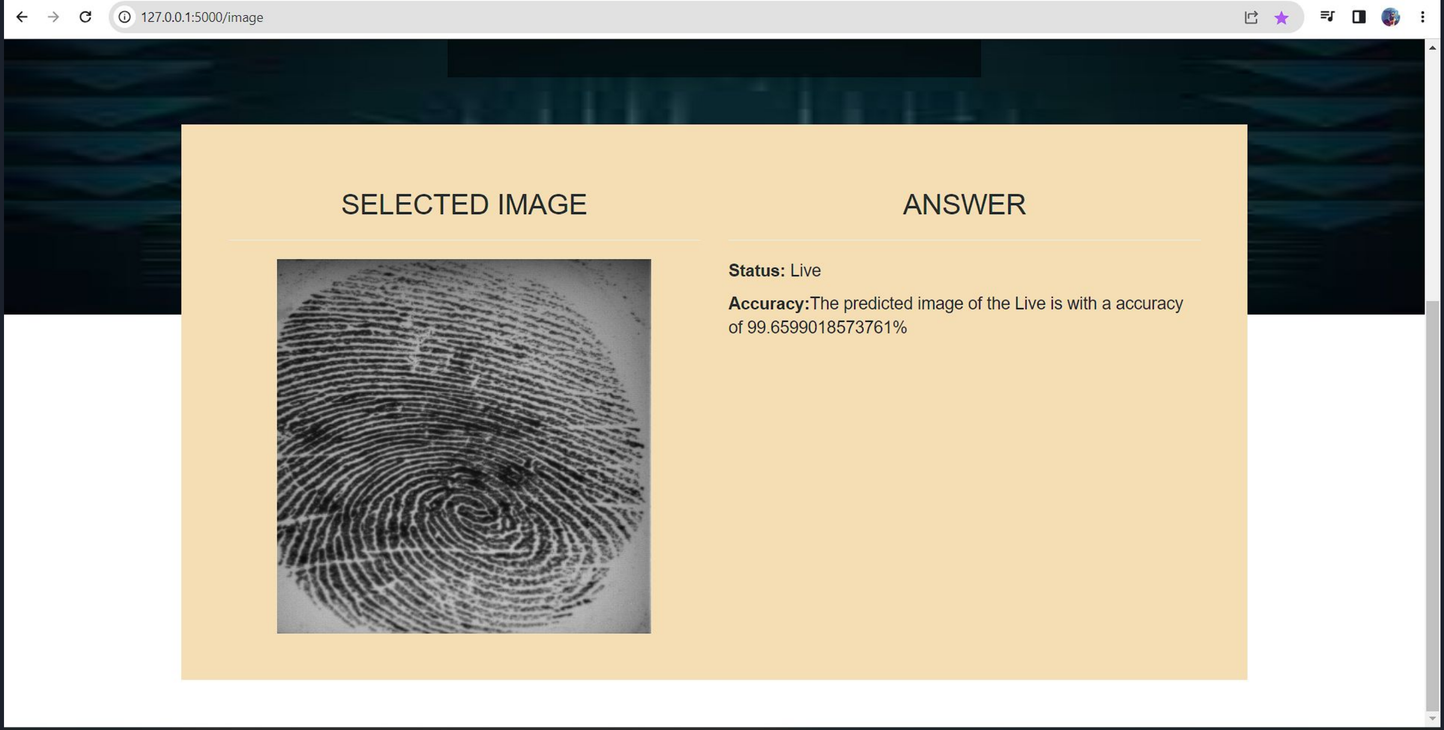
Global Institute of Management Sciences
(Dept. of Computer Science)

Project By Sandeep (20XHSB7051)
Under the Guidance of Prof. Sumathi G.K

Screenshots



Screenshots



SELECTED IMAGE

ANSWER

Status: Live

Accuracy: The predicted image of the Live is with a accuracy of 99.6599018573761%

Database

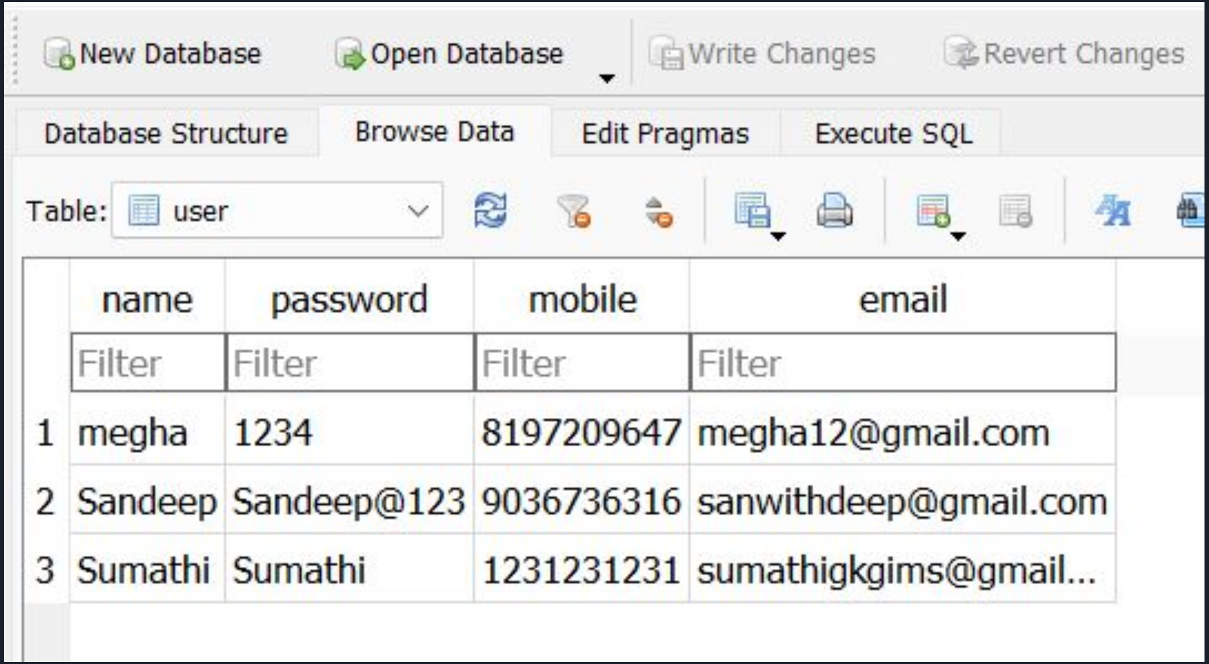


Table: user

	name	password	mobile	email
1	megha	1234	8197209647	megha12@gmail.com
2	Sandeep	Sandeep@123	9036736316	sanwithdeep@gmail.com
3	Sumathi	Sumathi	1231231231	sumathigkgims@gmail...

System Requirements

Hardware Requirements:

- System : Pentium IV 2.4 GHz/intel i3/i4.
- Hard Disk : 40GB.
- Monitor : 15 VGA Color.
- RAM : 512 Mb Minimum

Software Requirements:

- Operating system : Windows XP/ Windows 7 or More
- Software Packages : Tensorflow 1.14 , Open Cv
- Coding Language :Python.
- Toolbox : Anaconda.

Future Enhancements

1.**Robustness against advanced spoofing techniques:** Develop algorithms that can detect and differentiate between various types of spoofing materials, such as silicones, gels, or 3D printed fingerprints. This would help increase the system's resistance to more sophisticated spoofing attempts.

2.**Multimodal biometrics:** Combine fingerprint recognition with other biometric modalities, such as iris or facial recognition, to create a more robust and reliable authentication system. By leveraging multiple biometric features, the system becomes more resistant to spoofing attacks.

3.**Live presentation attack detection:** Implement real-time detection mechanisms to identify whether the presented fingerprint is from a live finger or a spoofing attempt. This can involve analyzing factors such as blood flow, temperature, or texture to differentiate between real and fake fingerprints.

Conclusion

In this paper, a system that can generate synthetic fingerprints and detect fake fingerprints is proposed. The experiments show that fingerprints generated by the proposed algorithm well capture the nature of real fingerprints. The presentation attack detection algorithm outperforms the existing algorithms in term of accuracy and processing time.

References

[1]“Universal Material Translator: Towards Spoof Fingerprint Generalization,” in IEEE International Conference on Biometrics (ICB), 2019.

[2]“Fingerprint Spoof Buster: Use of Minutiae-centered Patches,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 9, pp. 2190–2202, 2018.

[3]“Design and Fabrication of 3D Fingerprint Targets,” IEEE Transactions on Information Forensics and Security, vol. 11, no. 10, pp. 2284–2297, 2016.