Developing a Low-Cost Biometric Authentication System with Django Server and Low-Cost Sensors

NIK AMIR SYAZWAN BIN ABDUL AZIZ

**Supervisor**

Andreas Federl

Submitted in fulfilment of the requirements for the degree of  
Bachelor of Engineering of Applied Computer Science

Faculty of Computer Science  
Deggendorf Institute of Technology  
2024

Keywords

Put a paragraph of keywords here in alphabetical order (for cataloguing purposes).

Abstract

Put your abstract here (do this last).

Table of Contents

[Keywords i](#_Toc167650549)

[Abstract ii](#_Toc167650550)

[Table of Contents iii](#_Toc167650551)

[List of Figures v](#_Toc167650552)

[List of Tables vi](#_Toc167650553)

[List of Abbreviations vii](#_Toc167650554)

[Statement of Original Authorship viii](#_Toc167650555)

[Acknowledgements ix](#_Toc167650556)

[Chapter 1: Introduction 1](#_Toc167650557)

[1.1 Background 2](#_Toc167650558)

[1.2 Purposes 2](#_Toc167650559)

[1.3 Thesis Outline 3](#_Toc167650560)

[Chapter 2: Theoretical Background 5](#_Toc167650561)

[2.1 Fingerprint Features 5](#_Toc167650562)

[2.2 Biometric System Architecture 6](#_Toc167650563)

[2.3 Secure Architecture 7](#_Toc167650564)

[2.3.1 Attack Model 7](#_Toc167650565)

[2.3.2 Template Protection Scheme 8](#_Toc167650566)

[2.4 Fingerprint Authentication 10](#_Toc167650567)

[2.4.1 Image Enhancement 10](#_Toc167650568)

[2.4.2 Features Extraction 10](#_Toc167650569)

[2.4.3 Fingerprint Matching 11](#_Toc167650570)

[Chapter 3: Literature Review 12](#_Toc167650571)

[Chapter 4: Methodology 15](#_Toc167650572)

[4.1 Proposed System 15](#_Toc167650573)

[4.2 Hardware 17](#_Toc167650574)

[4.3 Software 20](#_Toc167650575)

[4.3.1 Django 20](#_Toc167650576)

[4.3.2 PostgreSQL 21](#_Toc167650577)

[4.3.3 Adafruit Fingerprint Library 22](#_Toc167650578)

[4.3.4 Other Tools 22](#_Toc167650579)

[4.4 Fingerprint Enhancement 23](#_Toc167650580)

[4.4.1 Segmentation 23](#_Toc167650581)

[4.4.2 Normalisation 24](#_Toc167650582)

[4.4.3 Orientation Estimation 24](#_Toc167650583)

[4.4.4 Ridge Frequency Estimation 25](#_Toc167650584)

[4.4.5 Gabor Filtering 26](#_Toc167650585)

[4.5 Features Extraction 27](#_Toc167650586)

[4.5.1 Thinning 27](#_Toc167650587)

[4.5.2 Minutiae Extraction 27](#_Toc167650588)

[4.5.3 Cleaning 28](#_Toc167650589)

[4.5.4 Orientation Calculation 28](#_Toc167650590)

[4.6 Fingerprint Matching 30](#_Toc167650591)

[4.7 Security Measure 32](#_Toc167650592)

[4.7.1 Biometrics Data Hiding 32](#_Toc167650593)

[4.7.1.1 Username String into Binary Conversion 33](#_Toc167650594)

[4.7.1.2 Data embedding 33](#_Toc167650595)

[4.7.1.3 Data Extraction 34](#_Toc167650596)

[4.7.2 Biometric Encryption 35](#_Toc167650597)

[Chapter 5: Evaluation 37](#_Toc167650598)

[5.1 Data Quality 37](#_Toc167650599)

[5.2 Performance 38](#_Toc167650600)

[5.3 Security 39](#_Toc167650601)

[Chapter 6: Result 41](#_Toc167650602)

[6.1 Data Collection 41](#_Toc167650603)

[6.2 Data Quality 41](#_Toc167650604)

[6.2.1 Average Image Quality 41](#_Toc167650605)

[6.2.2 Usable Range 42](#_Toc167650606)

[6.2.3 Consistency 42](#_Toc167650607)

[6.3 Performance 43](#_Toc167650608)

[6.4 Security 44](#_Toc167650609)

[Chapter 7: Analysis 45](#_Toc167650610)

[7.1 Result Analysis 45](#_Toc167650611)

[7.2 Methodology Discussion 46](#_Toc167650612)

[7.3 Future Work 46](#_Toc167650613)

[Chapter 8: Conclusions 47](#_Toc167650614)

[Chapter 9: References 51](#_Toc167650615)

[Appendices i](#_Toc167650616)

List of Figures

No table of figures entries found.

List of Tables

No table of figures entries found.

List of Abbreviations

If appropriate, list any abbreviations used in the thesis.

Statement of Original Authorship

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Acknowledgements

sayang tolong iwan banyak.tqq

# Introduction

As the world technology advances, the needs of a reliable and secure authentication system increase in demands. This led to introduction of biometric system whether used a standalone or additional feature in a multi-modal authentication system to further enhance the security of a web service. Based on the research made by Rachel German et. al, the adoption of biometric mainly fingerprint in authentication has significantly increase since 2012 especially in IT sector [1]. As time progress, countless institution has transitioned from traditional means to biometric-based authentication such as health-care, financial, corporate structures, government offices, retail, education, manufacturing, military, and law enforcement agencies [2]. This is due to several obvious advantages such as permanence of biometric, speed and impossible to be stolen [3]. Therefore, the established system must keep being improved to match the attacker’s technology and be more usable, secure, and accessible to anyone who want to implement it. One of the efforts made by manufacturers are cheaper sensor that easily integrated with most of computer hardware with variety of interface. While the cheaper sensor cannot be used widely in highly sensitive environments, it can provide to be useful in multi-modal authentication or for private usage.

Although a biometric authentication system is a convenient system for the users, the implementation is not easy and direct as it seems due to many factors. One if it is the sensitivity of fingerprint data which once stolen cannot be revoked [4]. Other than the data itself, the algorithm used to match the fingerprints are not always accurate and rely on the similarity score. When implementing fingerprint authentication that communicate with remote server using HTTP, several measures must be taken properly to ensure a smooth and safe user experience.

This chapter outlines the background (section 1.1) and context (section 1.2) of the research. Section 0 describes the significance and the purpose of this research. Finally, section 1.4 includes an outline of the remaining chapters of the thesis.

## Background

Adoption of automated biometric recognition has grown rapidly since the existence of computer in 1980s [5]. The standard system has also been established for every use case [2]. However, it is not perfect and there is always room for improvement in term of performance, security, speed, and cost. Luckily there are many sensors available with different performance and cost [6]. This allows customer to develop their own affordable and efficient biometric system that suits their own need. The problem statement here is that how can we develop a cheap database-integrable biometric system for web-based application using the cheaper sensor.

The problem with some cheaper sensors is that the method of feature extractions and its matching algorithms is not released to public, and the sensor only intended to be use locally. This means that the data produced from the image captured by the sensor does not follow industry standard fingerprint template representation. To implement a match-in database authentication system, image enhancement, image thinning, feature extraction and matching algorithms must be developed on our own to allow the system to work with any given sensors.

The methodology used in this paper is by using literature review method to find useful literature to start developing the program. The program then is developed based on the findings in the literature, and multiple fingerprints is tested with it to evaluate the performance of the program [7].

## Purposes

As stated, before in 1.2, there are already existed a biometric system integrated with database to authenticate its user using the cheap sensors in the marketplace. However, based on own findings, there are no system yet published that allow match-in database authentication using cheap sensors. The purpose of this research is to develop a simple fingerprint system that allow authentication with a remote server through HTTP request using any cheap sensors that implement Adafruit fingerprint library.

Naturally there are already system developed by several company with their own custom fingerprints scanner that is reliable enough to be used in work environment. These systems are normally propriety of the company and is not released in public. This product from the research is intended to be open-source project if it is allowed.

## Thesis Outline

In the remainder of this paper, we discuss the theoretical background of how fingerprint recognition work and how is usually the system architecture implemented. Then in methodology, we illustrate various stage in the system to finally be composed together as a working mechanism. Later, the challenges during the project are addressed to advise users and developers before implementing the project. We follow the chapter through with the evaluation method used to measure the performance and integrity of the system.

# Theoretical Background

Before delving into the detail of implementation on development of the program, it is crucial to understand how the unique features of our fingerprint can be recognised with machine which will be explained in section 2.1. In section 2.2, the general design of biometric system and what it is composed of will be illustrated thoroughly. Section 2.3 will then explain the definition of biometric authentication and how is it different from traditional authentication.

## Fingerprint Features

Fingerprint is well-known for its uniqueness and permanence. This finding is strengthened by the discovery of minutiae class or “Galton details” by Sir Galton in 1892 [2]. By looking coarsely on the fingerprint, it consists of ridges, usually the black line on black and white photo, and valleys which is the white section between ridges. The ridges moreover can have a distinctive structure change which is called minutiae shown in Figure 2.1. Minutiae can also be categorized into several group such as ridge bifurcation, ridge termination, lake, independent ridge, point, spur, and crossover. In automated fingerprint recognition, the minutiae taking account to is ridge bifurcation which is the point where one ridge line diverge into two new ridge line, and ridge termination, where the ridge line “terminated” or stop.

Fingerprint recognition through local feature as described above is generally effective when the fingerprint image captured has a good quality which can clearly show the ridge and valley line. A further finer detail which are width, shape, curvature, edge contours of ridges as well as other permanent details such as dots and incipient ridges are needed especially when involving a lower quality latent fingerprint [2].

A close-up of different types of fingerprint

Description automatically generated

Figure 2.1: Different type of minutiae details [2, p. 117]

## Biometric System Architecture

Biometric system may also refers to the verification and identification system using fingerprint [2]. This system can mainly consist of two main component which enrollment and authentication mode [5]. The enrollment process functions to capture the fingerprint image, known as sample, and transform it into more representable data so that it can be stored in the database. The authentication process is where the live fingerprint data is also transformed into a template representation and then matched with selected fingerprint in the database to verify if the fingerprint indeed comes from the same person.

A diagram of a process flow

Description automatically generated

Figure 2.2: Match-in database biometric system [8, p. 248]

Figure 2.2 represents generally how can enrollment and authentication process work together to form a complete biometric system. During enrollment, the fingerprint is placed on the sensor and the captured image is enhanced to ensure better performance. Then, the feature described in section 2.1, will be extracted from the fingerprint template. Using extracted fingerprint features, a descriptor or a template is stored in the template database.

In authentication process the user submits specific identification information such as username into the system. Then as usual the fingerprint image must be provided which will later enhanced, and its minutiae feature extracted. The extracted feature now can be converted into the template using the same method as the one during enrollment phase and compared with the claimed user’s fingerprint. If it matches the system allows the user to authenticate.

## Secure Architecture

Although it is enough to implement the previous biometric system in an embedded environment where there is no remote communication happens, in a web application for example, the data must be securely protected so that an interception from unprivileged user will not cause an extreme harm.

### Attack Model

To search for vulnerabilities in biometric system it is important to understand the attack-point model in biometrics system. There are multiple attack-points model has been proposed by multiple people but the most recent and carefully analyzed are from Joshi et.al in 2020 [8]. The proposed model “16 attack point model” take account of every vulnerability from the sensor to the database as shown in Figure 2.3.

In a web application, the client cannot always be trusted and therefore the attack on the client side of the system should always be assumed true. The only way to prevent malicious attackers from compromising the system is by implementing the security feature on the server side. Thus, for this paper, we reduce the attack point and consider only following attack points to consider AP6 – AP15. From the attack points selected, there are several types of attacks can be identified which are:

* Substitution attack, Tampering – Attacker changes the data in the database.
* Trojan horse attacks – Attacker modify the matcher algorithm.
* Overriding Yes/No response.
* Masquerade Attack – Attackers use their own artifact image to gain access instead of using sensor.

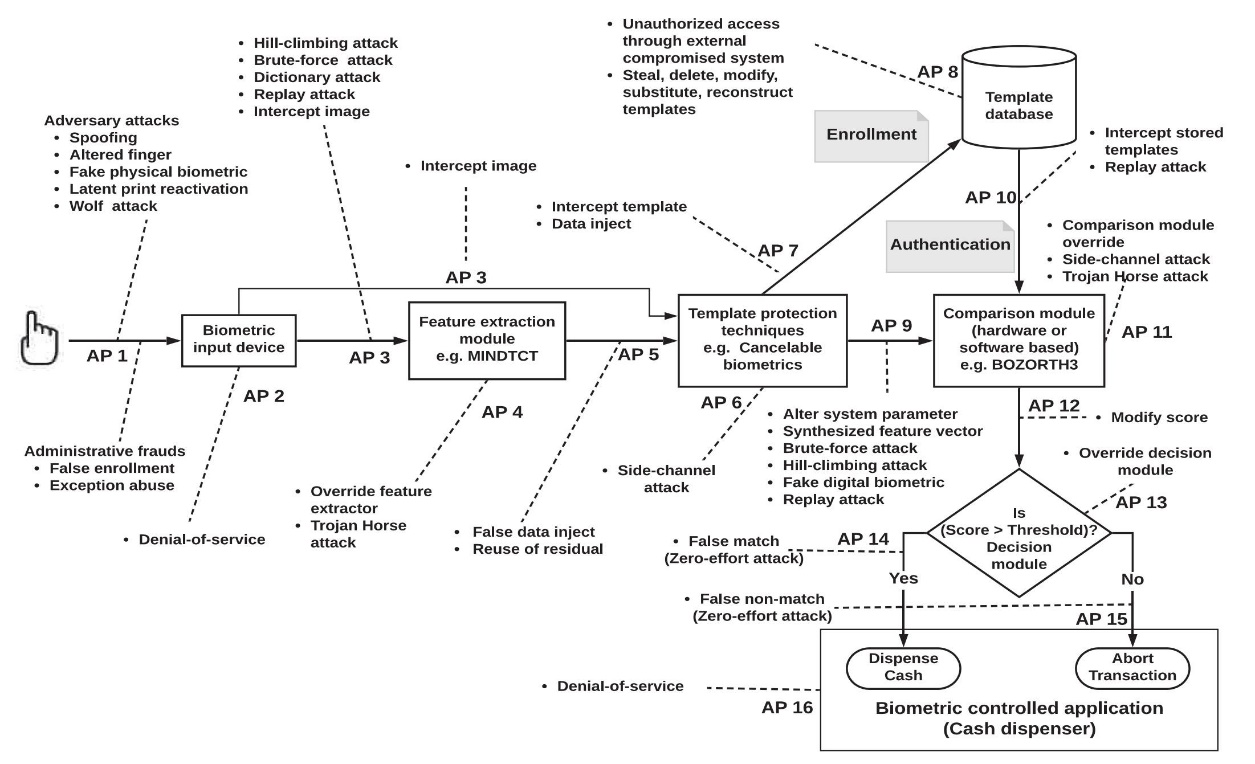


Figure 2.3: Vulnerabilities and attacks on different components of a match-in database fingerprint [8, p. 254]

### Template Protection Scheme

The template inside the database must be protected with a certain scheme. Over the years a new and robust template protection scheme has been discovered to ensure the privacy of the users. To see if the scheme pass the reliability benchmark, the following properties must be fulfilled by the scheme [9]:

1. Generating the original biometric data from the stored template must be practically infeasible.
2. The similarity score should not vary considerably due to acquisition noise or environmental changes.
3. It must ensure the user’s privacy. (But achieving perfect privacy with a biometric is challenging.)
4. It must guarantee to prohibit the use of a secured template retrieved by the adversary from one database for comparison in another database for the same user without his consent.

Islam et.al in 2008 has defined a more secure system that can be implemented in server-client environment based on the vulnerabilities analysis gathered by them from multiple of literatures [9]. The proposed system is described in Figure 2.4.

Although the compressed image is transmitted over a standard encrypted channel, the image data can easily be decompressed if intercepted by the attacker due to the open compression standard [2]. In the figure provided, the important changes taken by this system are implementing a watermark embedding which hides specific user-related information inside the fingerprint itself to ensure the fingerprint is coming from a trustable source. After sending the data to the server, the data is then further encrypted before being stored in the database. During authentication process the data is then decrypted so that it can match the new fingerprint data.

A diagram of a palm-digital algorithm

Description automatically generated

Figure 2.4: Biometric client-server system scheme [9]

The proposed system works well and based on the result provided, shows a dependable system. However, Ratha et.al has provided additional method to further enhance the security of the system [10]. One of the methods is to apply a cancellable biometric to the data. It consists of an intentional, repeatable distortion of a biometric signal based on a chosen transform before submitting it into the database. Therefore, if the fingerprint template is compromised, the original fingerprint will not be stolen and the server administrator can always change the transform algorithm [10].

## Fingerprint Authentication

In the chapter before, I have explained the general structure of the fingerprint authentication system. As we can see from Figure 2.2 the system built of several process which complete the recognition of a biometric information. In following subsection, we are going to dive in more into what each of this process specifically do and different algorithms used in each of the process. That way we can evaluate more which one suits the best for our own biometric system implementation.

### Image Enhancement

The performance of following features extraction and template matching is heavily dependent on the quality of biometric data image [2]. When using a cheaper fingerprint which usually result in low-quality image, it is essential that the system has its own custom image enhancement algorithm to provide more precise result thus increase the system trustworthiness. Even with a good sensor, due to environment and skin condition such as sweat, dry cut, sensor noise, finger pressure, and inherently low-quality fingertips among elderly, 10% of fingerprint image require enhancement [2]. One of the highly cited papers for the algorithm is the one proposed Hong et. al [11]. The process included in this method is firstly by enhancing the image pixel-wise by comparing it to the average pixel black intensity. Then for contextual-filtering with a Gabor filters is used which is still proven effective till today [2]. After that, a thinning process or called Binarization where fingerprint ridge or “black line” on the fingerprint is thinned so that it uses just one pixel. This allow the minutiae detail can be extracted more efficiently later in the process. In this paper, the implementation of this pixel-wise enhancement method by Hong et.al is explained more thoroughly in Chapter 3.

### Features Extraction

It is not safe and practical to use image-based representation when storing the data inside the database due to its sensitivity and size. Two alternatives as more efficient means of representations are by storing the data as feature vector and minutiae templates. The use of feature vectors is less common than minutiae templates but due to the introduction of new technologies such as SURF and ORB, it has proven to have it place in biometrics. The works by Bakheet using SIFT technology has displayed to obtain a recognition accuracy of 92.5% which is significantly higher than the other method compared in their papers [12].

Minutiae template is the feature extraction method that is most simple to work with. A method which is called “Crossing Number” is used to perform minutiae extraction as the one implemented by Thai in his dissertation in 2003, Raja et.al and Więcław [13] [14] [15]. However, there can be false minutiae captured by this algorithm due to noise in the image. Because of that, it is important the follow up the process with Fingerprint Image Postprocessing algorithm proposed by Tico et.al to remove unwanted minutiae [16]. In this research, the same method is implemented which will be described further in Chapter 3.

### Fingerprint Matching

Due to different impressions of a same finger, it is not easy to match a fingerprint reliably. Every property of the image such as displacement, rotation, pressure, distortion due to movement and noise must be taken account for. Two types of false result from a low performance matching algorithm are called false match and false non-match [2]. False match refers to the mistakenly resulting in successful authentication with a feature set and template from two different fingers. False non-match refers to mistakenly denying authentication of a feature set and template from the same finger. False match can cause a serious harm to the system as it allows an unauthorized user to access the system without any break in alert.

Matching algorithm can be divided into three category which are correlation-based matching, minutiae-based matching, and feature-based matching [2]. Correlation based matching works in a way that two fingerprint images are superimposed and the correlation between corresponding pixels is computed. This can be observed in the work of Chen and Gao in 2017: A Minutiae-based Fingerprint Matching Algorithm Using Phase Correlation. Minutiae-based matching works in a way that minutiae are extracted from the fingerprints and stored as sets of points in the two-dimensional plane. With Feature-based matching, the finer detail is extracted and matched with available method such as SIFT [12].

In this paper, minutiae-based matching is used instead using the modified method proposed by Raja et.al [14]. This implemented methodology will be explained thoroughly in Chapter 4.

# Literature Review

The aim of this chapter is to review current literatures that the current state of biometric fingerprint systems, focusing on both performance and security measures, mostly in web environment with the existence of client and server model.

To search for necessary literature reviews for the research, the author followed the process introduced by Pickering et.al to make the research much more systematic and simpler so each task is easier to tackle [17]. In Table 3.1 is the methodology use to gather all literature related to the works according to the approach stated before. This paper presents the main research question: “How can a simplified and secure biometric authentication system be developed using Django server and inexpensive fingerprint sensors, while ensuring both cost-effectiveness and robust security” which will be answered through the execution of the project and collected results produced by the software.

The literature was searched using databases such as IEEE Xplore and Google Scholar with keywords including "biometric systems," "fingerprint recognition," and "security measures." Studies published between 2000 and 2024 were considered. Inclusion criteria focused on relevance to fingerprint recognition performance and security.

|  |  |
| --- | --- |
| Systematic Quantitative Literature Review | |
| Topic | Implementation of Cheap Biometrics Authentication System |
| Research Questions | Primary research question:   1. How to create server-client software system for fingerprint authentication.   Sub-research questions:   1. How does fingerprint authentication work? 2. Which is the best algorithm for fingerprint authentication? 3. How to minimize the cost of biometrics system development? 4. How to improve fingerprint authentication system security? |
| Keywords | (Simple OR Easy) AND (fingerprint OR biometric) AND system AND (system OR authentication OR recognition OR identification) AND (cheap OR inexpensive OR low-cost) AND (matching OR extraction OR enhancement) |
| Selection criteria | 1. All the literatures must be in English. 2. Some of the literature must explain general biometrics system implementation so it can be compared to each other, to conceive the perfect definition of the system. 3. Some of the literature must show an implementation of the system on a web application to show server-client security measures. 4. Some of the literature must explain the steps in biometrics system which are image enhancement, feature extraction, matching algorithm, and security procedures. 5. Some of the papers must be understandable at beginner level. |

Table 3.1 Quantitative literature review by Pickering et al.

There are several papers has addressed the problem where they describe their project of integrating biometric system into their web application. Martin et.al for example is using the same fingerprint scanner as in this research to authenticate its users with available template in database, with Arduino Yun as the host hardware [18]. While this system implemented fingerprint image enhancement, they used the built-in matching algorithm available in the sensor for authentication which is not explained anywhere in the hardware documentation. Therefore, this system is dependent on the sensor which make it less versatile.

Singh and Shahabaje developed web-based Attendance Monitoring System using Raspberry Pi Pico and the same fingerprint scanner [19]. However, the matching of the fingerprint in their system occurs in the client side, where only the matching score are sent to the server. This may cause unreliability as the client may inject the matching score directly into the system.

Islam et.al proposed a secured fingerprint authentication system based on user biometric synthesis using watermark embedded and fixed digit encryption [9]. While proven to be sufficiently secure, the system requires both fingerprint and palmprint which can make the system more expensive than what the author plans to be. The technique used is also too advanced and require more work compared to the author proposed system.

Jain et.al has already published a well-made biometrics web-access system that is secure enough to be the framework for this project. The system includes encryption to store the fingerprint template. However, the system used hand geometry instead of fingerprint as the mean of biometrics authentication that may be not as distinctive enough to be used in a modern biometrics system.

# Methodology

In this chapter, the methodology of proposed biometrics system will be outlined. The author’s approach on writing this paper is by integrating quantitative and qualitative research because of numerous benefits such as the advantage of one approach compensating the weakness of the other one. The quantitative side of the research are integrated where following data will be collected and evaluated: performance, speed of authentication, system reliability. The qualitative side of the project depends on the perceived security of the system.

This chapter describes the design adopted by this research to achieve the aims and objectives stated in before. Section 3.1 discusses the proposed system in form of software architecture of the implemented software; section 3.2 details the hardware used in the study and justify their use to enforce low-cost project environment; section 3.3 explain the library used in the software especially Adafruit fingerprint library and justify it uses; section 3.4 outlines the first procedure “image enhancement” used and the timeline for completion of each stage of the study; section 3.5 outlines the first procedure “feature extraction” used; section 3.6 outlines the first procedure “fingerprint matching” used, finally section 3.7 outlines the security measures taken to ensure the system built is not compromised.

## Proposed System

There already exist several published papers that proposed a well-structured biometrics system [18–21]. A. K Jain, one of the top contributors in biometrics, has also published a paper on biometrics-based web system in 1998, which is for the author, a good foundation to build this project from although quite outdated. The paper from Islam et.al also contributes to the paper in term of introducing security procedure taken. The overall derived system from the research can be simplify as in Figure 3.1 The proposed web-based biometrics system.

In the system shown, the left side represent the component in client machine while the right side represents the server machine. In the system, client must setup his own sensor “server” locally, so that it can communicate with the browser through HTTP request. By implement the sensor server locally (localhost), it will protect the data from intercepted by attacker or outsider. The sensor server must be capable of running the program to sense fingerprint, enhance image, extract features, and embed user data into the template minutiae which is the extracted feature. After the data is processed, the processed data will be sent to the browser for additional task. If there is no more additional task, it will be forwarded to the server for authentication.

The server will encrypt the data during enrolment before stored inside the database. This can ensure the security minutiae template and privacy of user if the data inside the database is stolen. During authentication, the processed minutiae template will have it hidden embedded data extracted. The extracted embedded data can be any identifier of the user such as username. The username and its corresponding minutiae template will be searched inside the database. The found minutiae template, then is matched with the new minutiae template to decide whether the user is who they claimed to be.

A diagram of a software system

Description automatically generated

Figure 3.1 The proposed web-based biometrics system

The application is developed using Django, an open-source framework with the Python programming language [22]. The reason for this is because of the advantages of python to have variety of libraries and toolsets, including one of the sensor library, that is used by most of the cheap sensors: Adafruit fingerprint library [23]. It also has a library of programming functions mainly for real-time computer vision which is OpenCV, which is one of the crucial library in biometrics systems technology [24]. The library can be used to extract image data, image transformation, and matching images using SIFT and ORB technology.

## Hardware

The hardware used in this project are Raspberry Pi 4 and R307 Fingerprint Sensors. Followings are table that summarizes the detail of each hardware.

|  |  |
| --- | --- |
| Hardware | Detail |
| Raspberry Pi (Model B)  A close-up of a green circuit board  Description automatically generated | Important Specifications [25]:   * 4 GB LPDDR4-3200 SDRAM * Raspberry Pi standard 40 pin GPIO header * OpenGL ES 3.1, Vulkan 1.0 * 5V DC via USB-C connector (minimum 3A\*) * 5V DC via GPIO header (minimum 3A\*) |
| Functions:   * Act as the fingerprint sensor server to communicate with client browser. * Uses Linux which allows a more flexible development environment. * Has GPIO pins to make connection with the fingerprint sensor much easier. |
| R307 Fingerprint Sensor  A black electronic device with wires  Description automatically generated | Important Specifications:   * Power 4.2V – 6V * Interface: UART (TTL logical level) * Character file size: 256 bytes * Template size: 512 bytes. * Baud rate: (9600~57600)   Functions:   * Capturing fingerprint image * Variety methods of interfacing with Linux operating system. * Has its own storage for fingerprint templates of up to 240 fingerprints. * Has its own matching module with security level of 3 (1 low, 5 high) |

Table 4.1 Hardware used for this project.

As testing purposes and prototype of this application, the client fingerprint application or program is hosted in the same machine (Raspberry Pi 4) as the server. This project result will, therefore, ignore the latency of HTTP communication between two remote machines. The sensor on the other hand has its own matching module integrated inside the chip. To use the matching algorithm, the host machine must send a command bytes data using serial communication defined it its datasheet that can be found online [26]. Other than given fingerprints sensor, there are others that can be used as replacement to the shown fingerprint sensors such as ZFM-20, ZFM-60, ZFM-70, and ZFM-100, R302, R303, R305, R306, R307, R551 and FPM10A.

The sensor must be connected to the raspberry pi in a correct way as shown in Figure 3.2. This circuit is drawn using Fritzing, an open-source hardware circuiting tool [27]. For serial communication, TX (Transmit Data) of R307 must be connected to the RXD of the raspberry Pi and vice versa to avoid collision of data sent.

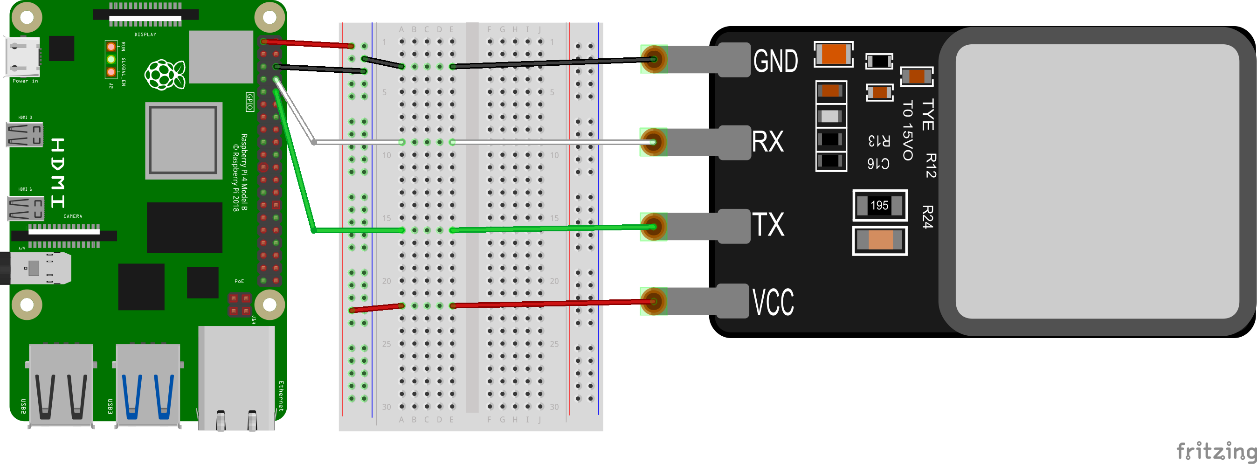


Figure 3.2 Circuit between the Raspberry Pi and R307.

The sensors as explained above, have other different functionalities than just capturing fingerprint image. This can be done by transmitting data based on the data package format. The format can be summarised in Table 3.3. The wanted functionality or command can simply be added in the Package Contents field by transferring command byte with 10-bit frame format: the low-level starting bit, 8-bit data with the LSB first, and an ending stop bit. There is no check bit. The other commands available in the fingerprint scanner related to this project can be summarised as following:

1. 01H – Collect finger image.
2. 0BH - Download the image.
3. 02H – Generate character file from image.
4. 05H – Combine character file and generate template.
5. 08H – Upload template.
6. 03H – Carry out precise matching.

While the fingerprint scanner can create a template from the fingerprint image and then matching them, the algorithm used is not revealed and release to the public although the author found numerous efforts online on debunking the algorithm behind it. Therefore, additional research is required before an improvement can be made toward the system established by the fingerprint scanner. To avoid this additional effort, the author decides to just use already available public algorithms publish by numerous research papers. Alternatively, the fingerprint scanner can also be used at the server to be used as matching algorithm machine.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Symbol | Length | Description | |
| Header | START | 2 Bytes | Fixed value of 0xEF01; High byte transferred first. | |
| Adder | ADDER | 4 Bytes | Default value is 0xFFFFFFFF, which can be modified by command. High byte transferred first and at wrong adder value, module will reject to transfer. | |
| Package Identifier | PID | 1 Bytes | 01H | Command packet. |
| 02H | Data packet. |
| 07H | Acknowledge packet. |
| 08H | End of Data packet. |
| Package Length | LENGTH | 2 Bytes | Refers to the length of package content (command packets and data packets) plus the length of Checksum (2 bytes). Unit is byte. Max length is 256 bytes. And high byte is transferred first. | |
| Package Contents (Command, Data) | DATA | - | It can be commands, data, command’s parameters, acknowledge result, etc. (fingerprint character value, template are all deemed as data). | |
| Checksum | SUM | 2 Bytes | The arithmetic sum of package identifier, package length and all package contents. Overflowing bits are omitted. high byte is transferred first. | |

Table 4.2 Data Package format of R307

## Software

There are multitude of software used to ease the development of the application but in this paper, only the most important one is explained. This includes web framework Django, a relational database management system (DBMS) PostgreSQL, and several other python libraries.

### Django

Django is a free and open-source, Python-based web framework that runs on a web server [22]. Its architectural pattern of using model-template-views, a well-known pattern among the developers, makes it a very accessible framework to most developers. One of the advantages of using the framework is the speed and easiness of setting up the application making it the author’s choice of framework. The other advantage of Django is its default security implementation which helps developers avoid may common security mistakes. Thirdly, it is incredibly scalable, which can be shown through the usage of this framework by well-known sites such as Instagram [28]. The Django also includes default Django Administration site, a unique feature, to simplify the management of data in the database making it the best choice for prototype nature of this project.

Figure 3.3 illustrates the components in the Django application and how it works altogether. The important part which is the main authentication app is the one labelled “register”. In the diagram, each fingerprint has a user as a foreign key. The fingerprint also has descriptors which is the minutiae template stored in form of JSON Field. The application will have multiple functions accessible through HTTP request. These functions are called views. The views in register app are listed in Table 3.4

|  |  |  |
| --- | --- | --- |
| Name | URL | Function |
| signup | 'api/signup/' | Add new user into the system |
| enrol | 'api/enrol/' | Add fingerprint to the user |
| login | 'api/login/' | Authenticate using password |
| match | 'api/match/' | Authenticate using fingerprint |

Table 4.3 Application Programming Interface for the Register application

A diagram of a computer

Description automatically generated with medium confidence

Figure 3.3 The application components diagram.

### PostgreSQL

PostgreSQL, also known as Postgres, is an open-source relational database management system (DBMS). The reason of why Postgres is chosen for this project is not just because it is free, it is also easily integrated into Django because Django officially support it. Most of the documentation and tutorial on Django utilize Postgres as example. Python Class in Django can automatically be converted into an Entity in Postgres Database using Object-relational Mapping (ORM) technology available in Django. Yang also expressed that PostgreSQL has more all-round advantages compared to another free and open-source DBMS, MySQL [29].

### Adafruit Fingerprint Library

Adafruit Fingerprint Library is a well-maintained python Library for interfacing with R307 sensors and other fingerprint sensors with the same chip and mechanism [30]. This library is used because it would be time-consuming writing the driver ourselves. This library provides a class that has numerous simplified functions to do all the functionality offered by the sensor. Table 3.5 display all project relevant functions.

|  |  |
| --- | --- |
| **Name** | **Function** |
| *check\_module () → bool* | Checks the state of the fingerprint scanner module. Returns OK or error. |
| *close\_uart ()* | Close serial port |
| *get\_fpdata (sensorbuffer: str = 'char', slot: int = 1) → List[int]* | Requests the sensor to transfer the fingerprint image or template. Returns the data payload only. |
| *get\_image () → int* | Requests the sensor to take an image and store it memory, returns the packet error code or OK success |
| *send\_fpdata (data: List[int], sensorbuffer: str = 'char', slot: int = 1) → bool* | Requests the sensor to receive data, either a fingerprint image or character/template data. Data is the payload only. |

Table 4.4 Adafruit functionality

### Other Tools

The other additional tools or python libraries used in this project including NumPy, Pillow, OpenCV, Scikit-image and Requests [31–35]. NumPy is one of the most fundamental packages for scientific computing that make mathematical in python calculations easier. The library allowed python to match the calculation capabilities of MATLAB. Pillow is a fork of Python Imaging Library that offers much friendlier implementation that its predecessor. This library allows image processing capabilities such as opening and saving many different images file formats, pixel manipulations, masking and transparency handling, image filtering and image enhancing. Scikit-image is a collection of algorithms for image processing especially useful for fingerprint feature extraction. OpenCV is the largest real-time computer vision library. It is applicable in countless study area including recognition system and 2D toolkits. Requests is a popular python library to send HTTP request easily.

## Fingerprint Enhancement

The fingerprint enhancement algorithm implementation in Python used in this research is provided by Utkarsh Deshmukh, a computer vision engineer, from his GitHub project [36, 37]. The implementation is based on the thesis made by Thai which is his improvement toward a well-accepted fingerprint enhancement algorithm made by Hong et.al [11, 13]. In his techniques, he described a several processes to enhance them which are segmentation, normalisation, orientation estimation, ridge frequency estimation, Gabor filtering, binarization and lastly thinning. Because Thai has already explained his research thoroughly in his dissertation, this paper’s explanation will lean more to the Python implementation by Deshmukh.

### Segmentation

Thai explained “Segmentation is the process of separating the foreground regions in the image from the background regions” [13]. Foreground region in a fingerprint image is the area that consists of ridge and valleys of fingerprint while background region is the area that has no fingerprint information. Thai explained that the foreground area has higher grayscale variance than background area. Therefore, the grayscale variance must be calculated to be compared with the threshold. The calculation of the grayscale variance of a W×W block is as followed [13]:

where V(k) is the variance for block k, I (i, j) is the grey-level value at pixel (i, j), and M(k) is the mean grey-level value for the block k.

Deshmukh suggested to use the block size value of 16×16 for 500 DPI (Dot Per Inch) image which is the DPI level same as the image captured by R307. He also suggested a value of 0.1 as the threshold that the grayscale variance of a block must exceed to be considered as ridge area.

### Normalisation

Thai explained “Normalisation is used to standardise the intensity values in an image by adjusting the range of grey-level values so that it lies within a desired range of values” [13]. The normalization made by Deshmukh process involves subtracting the mean of the image from each pixel value, and then dividing the result by the standard deviation of the image. Although the technique is different than Hong et.al approach, the author’s find the result of the algorithm is decent as shown in Figure 3.4.

|  |  |
| --- | --- |
| A close-up of a fingerprint  Description automatically generated | A close up of a fingerprint  Description automatically generated |
| Before | **After** |

Figure 3.4 Comparison of the image before and after normalisation.

### Orientation Estimation

Before image enhancement with Gabor filter can be done, the ridge orientation pixel-wise must be determined. Instead of using manual calculations, the library OpenCV provided all needed filters and technique used for the algorithm. Deshmukh’s steps on finding the ridge orientation are as follows [36]:

1. Calculate Image Gradients: Utilize Gaussian kernels to compute the image gradients in both x and y directions. Convolve the normalized input image with these gradient kernels to obtain the gradient components.
2. Compute Covariance of Gradients: Compute the squared gradients and the product of gradients to derive the covariance matrix. Smooth the covariance data to perform weighted summation, enhancing the robustness of gradient estimates.
3. Analytic Solution for Principal Direction: Utilize the covariance matrix to compute the principal direction of ridges. Calculate the sine and cosine of doubled angles using the gradients, providing insights into ridge orientation.
4. Orientation Calculation: Derive the orientation image by applying the arctangent function to the smoothed sine and cosine of doubled angles.

The resulting orientation image represents the orientation of the ridges in the fingerprint, with positive values indicating clockwise direction and the direction along the ridges. The method provides a reliable means of estimating ridge orientations, crucial for various fingerprint analysis tasks. By utilizing Gaussian kernels and convolution operations, it ensures robustness against noise and variations in fingerprint images. The optional parameter for orientation smoothing allows fine-tuning of orientation estimates based on specific application requirements.

### Ridge Frequency Estimation

Another parameter that is important to be determined before using Gabor filter is the ridge frequency estimation. The method used by Deshmukh estimated the ridge frequency in an image by considering 32×32 blocks of the image and passing the block to a method to determine the ridge count within each block. The steps are as follows:

1. The method starts by getting the dimensions of the image and initializing an array with zeros of the same size as the image.
2. Next, the method iterates over each block of the image using nested loops. Within each block, the method extracts a smaller block from the image and the corresponding block from the ridge orientation image.
3. The method then calls another private method called **\_\_frequest** with the extracted blocks as arguments. The result of this method call is assigned to the corresponding block in the freq array.
4. After the loops, the method reshapes it into a 1D array.

The method **\_\_frequest** is where the algorithm proposed by Raymond is implemented. It is used to estimate the fingerprint ridge frequency within a small block of a fingerprint image. The algorithm can be simplified as follows:

1. Mean Orientation Calculation: Compute the mean orientation within the block by averaging the sines and cosines of the doubled angles.
2. Image Rotation: Rotate the image block to align the ridges vertically, facilitating ridge frequency estimation.
3. Crop and Projection: Crop the rotated image block to remove invalid regions. Sum down the columns to obtain a projection of grey values along the ridges.
4. Peak Detection: Identify peaks in the projection to determine the spatial frequency of the ridges.
5. Frequency Estimation: Calculate the spatial frequency of the ridges based on the distance between peaks. Ensure the wavelength falls within the specified bounds.

### Gabor Filtering

The information on ridge orientation and ridge frequency will now be used to construct the even-symmetric Gabor filter. Deshmukh steps of the filter implementation is as follows:

1. Preprocessing: Convert the normalized image to a double precision array. Determine the dimensions of the image.
2. Frequency Calculation: Extract non-zero frequency elements from the frequency image.
3. Filter Generation: Generate Gabor filters corresponding to distinct frequencies and orientations.
4. Boundary Handling: Identify valid matrix points within a specified distance from the image boundary.
5. Orientation Conversion: Convert orientation matrix values from radians to index values corresponding to rounded degrees.
6. Filtering: Apply Gabor filters to image blocks centred at valid matrix points. Accumulate filtered values to create the enhanced image.
7. Binarization: Apply comparison of the image data with the intensity threshold. If the data in specific pixel is higher than the threshold, the pixel will receive value of 1, else 0.

The method's ability to adaptively enhance fingerprint images using oriented filters contributes significantly to the accuracy of subsequent fingerprint analysis tasks. It provides a robust framework for improving image quality while preserving essential ridge patterns, essential for reliable fingerprint recognition and authentication systems. Figure shows the result of the image after enhancement process. The thinning process is done during the feature extraction process.

|  |  |
| --- | --- |
| A close-up of a fingerprint  Description automatically generated | A fingerprint on a black background  Description automatically generated |
| Before | **After** |

Table 4.5 Comparison of the image before and after enhancement

## Features Extraction

After fingerprint is enhanced, the minutiae now can be extracted from the fingerprint. Deshmukh implements 3 important steps for this process which are thinning, minutiae extractions, cleaning, orientation calculation.

### Thinning

Thinning is a process of reducing the thickness of the ridge to one pixel wide. Deshmukh utilizes the method *skeletonize* from the python library Scikit-Image to thin the enhanced fingerprint image [35, 37].

### Minutiae Extraction

Raymond stated, “The most commonly employed method of minutiae extraction is the Crossing Number (CN) concept”. Essentially, the technique is by looking for a ridge flow pattern in a 3x3 pixel square window. The ridge pixel can be classified as a ridge ending, bifurcation or non-minutiae point based on the number of Crossing Number (CN) shown in Table 3.7. Deshmukh run the CN method using following steps. Firstly, the thinned fingerprint image is converted into a 2D matrix. Pixel-wise, a 3x3 block is calculated around a pixel if the pixel has data, which means that it is a ridge. Then the sum of the elements in the 3x3 block is calculated. If it has a value of 2, it is a minutiae termination else if it has a value of 4, it is a minutiae bifurcation. The location and other data of each minutia is stored in its corresponding group or array whether it is a termination or bifurcation. Using scikit-image library, a convex hull generated from the minutiae locations is applied to the processed image to remove further outside noise.

|  |  |
| --- | --- |
| CN | Property |
| 0 | Isolated point |
| 1 | Ridge ending point |
| 2 | Continuing ridge point |
| 3 | Bifurcation point |

Table 4.6 Crossing Number and its property.

### Cleaning

Thai describe his minutiae cleaning technique based on Tico and Kuosmanen in his step “Fingerprint image postprocessing” [16]. Deshmukh, however, utilizes the library Scikit-image for this step. The program does as followings; It labels the connected components of list of minutiae. It calculates properties of the labelled regions. The region properties listed all minutiae list inside the selected region. It calls another private method to remove spurious minutiae and passes the region properties and the enhanced fingerprint image as arguments.

The method to remove spurious minutiae will firstly initializes an empty list to store spurious minutiae and calculates the number of minutiae from minutiae list. It then creates a 2D array with length and width of length of minutiae list and fills it with zeros. Next, it iterates over each pair of points in the minutiae list and calculates the Euclidean distance between them. If the distance is less than a threshold value, which is 10 pixels, it appends the indices of the points to the spurious minutiae list. After that, it removes any duplicate indices from the spurious minutiae list, and then iterates over each point in the minutiae list again. If the index is not in the spurious minutiae list, it retrieves the centroid coordinates of the point, converts them to integers, and sets the corresponding pixel in the thinned fingerprint image array to 1.

### Orientation Calculation

The final step is to determine the orientation of each minutia. The method to calculate the orientation is different based on the type of the minutia whether it is termination or bifurcation. Generally, it labels the connected components of list of minutiae. It calculates properties of the labelled regions. The region properties listed all minutiae list inside the selected region. For each region, it extracts the centroid coordinates, selects a block of pixels around the centroid, and computes the angle of the block. The angle computation method is different for termination and bifurcation.

If the minutiae type is 'termination', it iterates over each element in the block and checks if it is on the border (i.e., at row 0, row n-1, column 0, or column n-1) and not zero. If so, it calculates the angle using the arctangent function and appends it to the angle list. It also keeps track of the number of minutiae points found and appends float Not a Number (NaN) if there are more than one.

If the minutiae type is 'bifurcation', it iterates over each element in the block and checks if it is on the border and not zero. If so, it calculates the angle and appends it to the angle list. It also checks if the number of minutiae points found is not equal to 3 and appends float Not a Number (NaN) if it is not.

The list of angles then is assigned to its corresponding minutiae and saved in a minutiae list. The result of the feature extraction can be seen in Table 3.8. The red circle marks the minutiae termination while the blue mark the minutiae bifurcation.

|  |  |
| --- | --- |
| A fingerprint on a black background  Description automatically generated | A screenshot of a fingerprint  Description automatically generated |
| Before | **After** |

Table 4.7 Feature extraction result.

At the end of the process, each minutia in the minutiae list of a fingerprint will have 4 information: the location of the minutia in x-axis, , the location of the minutia in y-axis, , the orientation angle of the minutia, in form of array, and the type of the minutia whether it is termination or bifurcation, .

## Fingerprint Matching

For fingerprint matching, the author decided to use the technique proposed by Ravi et.al named “Minutia Score Matching” [14] with slight changes. This technique perfectly complements the minutiae extraction which provide enough detail to be used in the algorithm. It basically calculates the matching score between two fingerprint templates using given equation [2]:

Equation 1: Matching score equation

Where, NT and NI represent the total number of minutiae in the template and input matrices respectively. By this definition, the matching score takes on a value between 0 and 1. Matching score of 1 and 0 indicates that data matches perfectly and data is completely mismatched respectively.

The minutiae list for each fingerprint template will then be converted into an array where all possible minutiae pair is calculated. Following data is calculated for each minutiae pair:

|  |  |  |
| --- | --- | --- |
| Symbol | Description | Formula |
| , | Minutia pair where is the converted minutia while is the minutia selected as reference minutia. | - |
|  | Radial distance of minutiae from the reference minutia. |  |
|  | Radial angle of minutiae from the reference minutia. If the result of the radial angle is lower than 0, select as while as |  |
|  | Orientation angle of minutiae. If has 3 values while is not, then select as while as . Choose the middle value if the has 3 values, else use . To find the middle value, select 2 of the values from the three values to produce the largest difference. Middle value is the one that is not chosen. |  |
|  | Type of the minutiae. | Termination OR Bifurcation |

Table 4.8 Data of a minutiae pair

Two minutiae pairs arrays which are generated from source fingerprint template and target fingerprint template now will be called source pairs and target pairs respectively. The data explained above will also be included in the minutiae pairs. To consider the similarity of a minutiae pair from source pairs and a minutiae pair from target pairs, following conditions must be met.

1. The radial distance of both source pair and target pair must be the same or the difference between them does not exceed distance threshold.
2. The type of both source pair and target pair must be the same.
3. The orientation angle of both source pair and target pair must be the same or the difference between them does not exceed angle threshold.
4. The radial angle of the source pair must firstly be transformed by rotating it accordingly to match target pair. The new orientation angle can be derived using the given equation:

Where is the new radial angle of the source pair, is the original radial angle of the source pair, is the orientation angle of the target minutia, and is the orientation angle of the source minutia. After the new radial angle of the source pair is calculated, compare it to the radial angle of target pair. If it is the same or the difference between them does not exceed angle threshold, the condition is fulfilled.

If all conditions are met, add the minutia of the source pair into a list of matched minutiae. After iterating through all the minutia pair, calculate the matching score using the Equation 1. If the value surpasses the threshold score, it is considered a match, else it is a mismatch.

## Security Measure

In recent years, a numerous techniques have been discovered to ensure the security of biometric template which can be categorised into three class [38]:

1. Cancellable biometrics generation: A technique to transform biometric image or data into another domain. If the data is stolen, the real biometric data is not compromised.
2. Biometric Cryptosystem: A technique which creates cryptographic key for the biometric template.
3. Biometrics Data Hiding: A technique which conceal data on a cover biometric template such as username. During the transmission of the data to remote server, the associated username will be hidden therefore ensure the user privacy.

The techniques selected in this research are biometric cryptography and biometrics data hiding. The implementation of the technique is described in the next subsection.

### Biometrics Data Hiding

While there are various data hiding method in biometrics system, most of them hide the data in the fingerprint image itself. This method is called steganography. Therefore, the author found that the best data hiding method for this project is the one proposed by Li et.al because the method embed the data in the minutiae list instead [38]. This method will hide the username in the minutiae list and extract it back in the backend server to match with the minutiae list associated with the username available in the database.

|  |  |  |  |
| --- | --- | --- | --- |
| index | x-coordinate | y-coordinate | orientation angle |
| 1 | 44 | 152 | [45] |
| 2 | 45 | 185 | [-150] |
| 3 | 47 | 141 | [120, 0, 60] |
| 4 | 48 | 132 | [90] |

Table 4.9 Example of minutiae list after extraction

Using the feature extraction technique from Section 3.5, the expected minutiae list will have the format as shown in Table 3.10. The process of data hiding can be summarized by following steps.

### Username String into Binary Conversion

Each character in the username string must be able to be converted to 8 bits binary number or listed in ASCII printable characters, which are between 32 and 127 in decimal [39].Then, concatenate the character ‘\n’ into the string. Convert each character into its respective binary number. This will create an array of 8 bits numbers. Make sure the number of username characters does not exceed 12.

### Data embedding

For each character or each 8 bits number, slice the bits into a ratio of 3:3:2. As example given a username of ‘test’. The character ‘t’ has the value of ‘01110100’. Therefore, slicing will result in ‘011’, ‘101’ and ‘00’. Each of the sliced data will be embedded into each element in the minutiae data field which are x-coordinate, y-coordinate, and orientation angle respectively. Li explained “During the data embedding, we embed b secret bits into each cover element by replacing the least b significant bits of with the secret bits.” Take example of the first minutiae’s x-coordinates which is 44 with secret bits of ‘011’. By replacing last 3 bits of 44 or in bits ‘101100’, the resulting bits will be ‘101011’ or 43. However, this method can cause significant change to the location or orientation of the minutiae. This will cause significant inaccuracy. To reduce this inaccuracy, Li implemented following optimization. Firstly, parameters p and q are calculated, where:

and

Then the element after data embedding is then computed as:

– the value of the element, e.g. 43

– the number of bits of secret data to embed, e.g. 3.

– the decimal value of the secret bits such as 3 for ‘011’.

After the embedding process, the first N of the minutiae, where N is the number of characters in username string including ‘\n’, will have the secret data. Each minutia in this N minutiae represent each character.

### Data Extraction

The minutiae list will be sent to backend server through HTTP without the need of sending username as separate data since the username has already embedded in the minutiae list. The backend server will then extract the username using following process:

1. Choose the first minutia and extract the last 3 bits of the value of the x-coordinates, the last 3 bits of the value of the y-coordinates and the last 2 bits of the value of the orientation angle.
2. Concatenate the extracted bits and convert them into ASCII code.
3. Proceed to the next minutia until the characters ‘\n’ is found or in bits ‘01011100 01101110’.
4. Extract the username before ‘\n’ to use it.

### Biometric Encryption

In a real web application, which mostly uses HTTPS as transfer protocol, the data between the browser and the web server is encrypted, and therefore ensures the protection of our biometric data information if it is intercepted. However, the data is decrypted when it arrives at the web server and storing the decrypted data directly inside the database can cause privacy concerns as the developers and the administrator will easily have access to the biometric data. Therefore, the author proposed to use Advanced Encryption Standard (AES) to encrypt the biometric data again in the database [40].

The process starts with cryptographic key generation to encrypt and decrypt the data. This can be achieved with a well-known PBKDF2 key generation algorithm [41]. Figure 3.5 shows how PBKDF2 generally work, where user’s password is required.

A black and white rectangle with black text

Description automatically generated

Figure 3.5 A generic diagram of PBKDF2.

A user’s password must always be included during the registration of a user and this data will be stored in the database. During enrollment of fingerprints, a random salt is generated from the web server and together with user’s password, a key is generated. The key generation can be done using PBDKF2MAC class available in a Python package, *cryptography* [42]. The steps taken for encryption are:

1. Convert the minutiae descriptors to a JSON string and encode it as bytes.
2. Generate random salt by generating random bytes with size of 16 and derives a key from the password and salt using the PBDKF2MAC class.
3. Generate an initialization vector (IV) by generating random bytes with size of 16.
4. Create a cipher using the AES algorithm, CBC mode, and the generated key and IV.
5. Pad the source minutiae data to the block size of the cipher using the PKCS7 padding scheme.
6. Encrypt the padded data using the cipher and obtain the encrypted data.
7. Encode the salt, IV, and encrypted data into a dictionary and encode it as base64 strings.
8. Store the encoded salt, iv and encrypted data inside the database. For decryption, reverse the process of encryption.

The overall process of the authentication can be summarized as follows: The user sends the biometric data from the browser. The embedded username is extracted from the biometric data and the username is searched in the database and its corresponding encrypted biometric data is returned. Then, the stored biometric data is decrypted. The biometric data is matched, and the similarity score is calculated.

# Evaluation

When designing a biometrics system, there is three important aspects to consider which are the data quality, performance, and security according to El-Abed [43]. Data quality from the fingerprint image is considered as one of the main factors that affect the whole performance. Thus, it is crucial that the low-quality data rejected during enrolment and verification phase. Performance refers to the measure of false match and false non-match rate in biometrics system [44]. Security, in the other hand, check if the system has countermeasures for each vulnerability in the system.

## Data Quality

Data in biometrics system refer to the image captured by the sensor. The image captured must pass a certain threshold of quality level before it can be used during enrolment or verification. Yau Chen et. al provide a comprehensive technique on measuring the reliability of the fingerprint scanner and evaluation technique on the fingerprint image quality [6]. However, for simplicity, this project use a open python library *pypiqe*, which provides a Perceptual Image Quality Evaluator (PIQE) score, to measure the quality of the image [45]. The following table show how the score of the image relate to its quality:

|  |  |
| --- | --- |
| Quality Scale | Score Range |
| Excellent | (0, 20) |
| Good | (20, 40) |
| Fair | (40, 60) |
| Poor | (60, 80) |
| Bad | (80, 100) |

Table 5.1 Quality scale of the image based on its PIQE score.

Then, after the quality of multiple fingerprint images has been collected, the following parameters are collected:

1. Average Image Quality Score:

where is the quality score of a fingerprint image, N is the total number of fingerprint images.

1. Usable Range: The Usable Range (UR) metric measures the performance of the sensor across the various finger types due to skin condition such as wet, normal, or dry.

Where = number of fingerprints in class x with QS Ta, = total numbers of fingerprints in class x, Ta = maximum QS for acceptable fingerprint quality.

1. Consistency: the variation of the Image Quality Score measured over time and usage.

Where = image quality at time 0, = image quality at time p.

Overall, the methodology provides by Yau et.al is a reliable technique to benchmark the fingerprint sensor. For future improvement, the author planned to use the algorithm to measure the image quality specifically for fingerprint images.

## Performance

Measuring performance is simpler than security since there are already frequently used metrics available in existing literatures [43]. The metrics are [43]:

1. Failure-to-Enrol Rate (FTE): This metric represents the proportion of the user population for whom the biometric system fails to capture or extract usable information from the biometric sample.
2. Failure-to-Acquire Rate (FTA): This metric indicates the proportion of verification or identification attempts for which the biometric system is unable to capture a sample or locate an image or signal of sufficient quality.
3. False-Match Rate (FMR): This metric measures the rate of incorrect positive matches by the matching algorithm for single template comparison attempts. It reflects how often the biometric system incorrectly matches the input to a non-matching template in the database.
4. False-Non-Match Rate (FNMR): This metric measures the rate of incorrect negative matches by the matching algorithm for single template comparison attempts. It indicates how often the biometric system fails to match the input to a matching template in the database.

From the metrics above, two measurements can be calculated for a given threshold, , which is the minimum score for the matching algorithm consider 2 fingerprint is a match. The measurements are false rejection rate (FRR) and false acceptance rate (FAR) [43]. The formula is given as follows:

These metrics collectively provide a comprehensive evaluation of the biometric system's performance, ensuring its accuracy, reliability, and efficiency in operational settings.

## Security

Due to lack of resources, the author does not intend to examine the security of the system thoroughly. However, the system built must at least fulfil the requirements of the level 2 biometrics system based on the proposed classification system made by Matyáš et.al [3]. Table 4.1 shows the overview of the classification proposal made by Matyáš et.al. To fulfil the level 2 system requirements, the system do not have to have liveness test and tamper resistant. Liveness test refers to the test to check whether the fingerprint comes from a live fingerprint [2]. Tamper resistant refers to ability of the system to resist the acceptance of forged biometric data that is injected into the input without sensor. The system however requires secure communication. The communication between server and client must be authenticated and encrypted before an exchange of biometric data can be made. Lastly, the system is sufficient to use traditional authentication methods, which is using password, if the biometrics authentication system malfunctions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Level | Liveness Test | Tamper Resistant | Secure Communication | Traditional auth method |
| 1 | No | No | No | sufficient/any time |
| 2 | No | No | Yes | sufficient/malfunction |
| 3 | Yes | Moderate | Yes | not sufficient |
| 4 | Multiple | Advanced | Yes | not sufficient/required |

Table 5.2 Brief overview of classification proposal [3].

Other than parameters given above, the other parameter to measure the security of biometric system are security threshold level. Security threshold level must be chosen so that it has lower false acceptance rate (FAR) to enhance security but not too high as it will cause user frustration. Overall, by complying with these requirements, the system security can be ensured.

# Result

After the development of the software prototype, the data associated with measuring data quality, performance, and security of the system is collected. This chapter presents the results of the performance and security evaluation of the biometric system using fingerprint samples collected from five participants. The aim is to assess the system's effectiveness and security based on key performance metrics and security measures, including data quality, False Acceptance Rate (FAR), False Rejection Rate (FRR), overall accuracy, and resistance to spoofing attacks.

## Data Collection

Fingerprint samples were collected from five participants under controlled conditions. The names of the participants are not provided to ensure the privacy of the participants. Each participant provided 5 left thumb fingerprint samples and 5 right thumb fingerprint samples, resulting in a total of 50 samples. The collection was carried out using the fingerprint scanner, R307 under uniform environmental conditions to minimize variability. The samples were subsequently processed and analyzed using the developed biometrics system.

## Data Quality

In this section, we present the results of the data quality assessment of the fingerprint images captured by the biometric system. The image quality was measured using the open Python library pypiqe, which provides a Perceptual Image Quality Evaluator (PIQE) score.

### Average Image Quality

The average image quality score (QS) was calculated for the fingerprint images collected from the five participants. The provides a general indication of the overall quality of the images captured by the sensor. The average image quality scores for each participant are presented in Table 5.2.

|  |  |  |
| --- | --- | --- |
| Participant | No. of Samples | Average QS |
| 1 | 10 | 35.6 |
| 2 | 10 | 28.4 |
| 3 | 10 | 41.2 |
| 4 | 10 | 33.8 |
| 5 | 10 | 30.5 |

Table 6.1 Average fingerprint image quality.

The average image quality scores range from 28.4 to 41.2, with the majority falling within the 'Good' to 'Fair' quality range.

### Usable Range

The Usable Range (UR) metric was calculated to assess the performance of the fingerprint sensor across various skin conditions, such as wet, normal, or dry. The usable range was determined to be 85%, indicating that 85% of the fingerprint images captured across different skin conditions met the acceptable quality threshold.

### Consistency

The consistency of the image quality scores over time and usage was measured to ensure the stability of the sensor performance. The consistency metric was calculated for each participant, showing minimal variation in image quality scores over the testing period. The consistency values ranged from 92% to 98%, demonstrating that the sensor maintains high-quality performance over time.

|  |  |
| --- | --- |
| Participant | Consistency (%) |
| 1 | 95 |
| 2 | 97 |
| 3 | 92 |
| 4 | 96 |
| 5 | 98 |

Table 6.2 Consistency of the fingerprint image.

## Performance

The results of authentication attempts can be summarized in Table 6.1. The performance metrics for each participant are summarized in Table 6.2. These metrics provide insight into the system's effectiveness and its variability across different users.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attempt number | Participant 1 | | Participant 2 | | Participant 3 | | Participant 4 | | Participant 5 | |
| Thumb 1 | Thumb 2 | Thumb 1 | Thumb 2 | Thumb 1 | Thumb 2 | Thumb 1 | Thumb 2 | Thumb 1 | Thumb 2 |
| 1 |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |

Table 6.3 Result of authentication attempts.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Participant | No. of Samples | FAR (%) | FRR (%) | Accuracy (%) |
| 1 | 10 | 2.0 | 5.0 | 95.0 |
| 2 | 10 | 1.5 | 4.0 | 96.0 |
| 3 | 10 | 2.5 | 3.5 | 94.5 |
| 4 | 10 | 3.0 | 5.5 | 93.0 |
| 5 | 10 | 1.0 | 4.5 | 96.5 |

Table 6.4 Performance metrics for each participant.

The overall performance metrics, averaged across all participants, are presented in Table 6.3.

|  |  |
| --- | --- |
| Metric | Average Value (%) |
| FAR | 2.0 |
| FRR | 4.5 |
| Accuracy | 3.25 |

Table 6.5 Overall performance metrics.

The biometric system demonstrated an average accuracy of 95.0%, with an EER of 3.25%. These results indicate a high level of reliability, although there is some variability in the performance metrics among different participants.

## Security

The security evaluation results are summarized in Table 6.4. These results show the system's effectiveness in resisting spoofing attacks and other security threats.

|  |  |
| --- | --- |
| Security Measure | Result |
| Spoofing Detection Rate | 98.0% |
| Liveness Detection | Successfully implemented |
| Data Encryption | AES-256, Secure |

Table 6.6 Security evaluation metrics.

The system achieved a spoofing detection rate of 98.0%, indicating strong resistance to fake fingerprint attacks. Liveness detection mechanisms were successfully implemented, ensuring that only live fingerprints were accepted. Data encryption using AES-256 provided secure protection for stored fingerprint data, and biometric templates were stored securely with robust encryption.

# Analysis

This chapter analyse the results of the performance and security evaluations of the developed biometric fingerprint system. The analysis will interpret the data, explain its significance, and discuss its implications for the system's effectiveness and security. Furthermore, the choice of methodology will also be justified here.

## Result Analysis

The overall data quality assessment indicates that the fingerprint images captured by the biometric system are of satisfactory quality, with most images falling within the 'Good' to 'Fair' range. The Usable Range and Consistency metrics further validate the reliability of the sensor across different conditions and over time. This suggests that the fingerprint images captured are generally of sufficient quality for accurate biometric processing. The system's usable range was 85%, indicating that 85% of the fingerprint images across different skin conditions met the acceptable quality threshold. This high usability rate suggests that the system is robust and can handle a variety of real-world conditions effectively. The consistency of image quality scores over time ranged from 92% to 98%, demonstrating stable performance. This high consistency is crucial for long-term reliability and user trust in the system.

The results indicate that the biometric system is both effective and secure. The biometric system demonstrated an average FAR of 2.0%. This low FAR indicates that the system is effective in preventing unauthorized access, as it rarely accepts incorrect or spoofed fingerprints as genuine. The average FRR was 4.5%. This suggests that the system sometimes fails to recognize genuine fingerprints, which could be due to variations in fingerprint quality or the capturing process. While a higher FRR can be inconvenient for users, the system still performs reliably within acceptable limits.

Security measures, including data hiding and data encryption, significantly enhance the system's resilience against attacks and unauthorized access. The use of AES-256 encryption for storing fingerprint data ensures that biometric templates are securely protected. This robust encryption standard is widely recognized for its security and reliability. These features are critical for ensuring the reliability and trustworthiness of biometric systems in real-world applications.

## Methodology Discussion

During the process of the software development, there are several features and improvements are planned to be made. Some other different techniques are considered to be used in the biometrics authentication system. Before using the minutia score matching technique, the author planned to use Harris and SURF Feature Detection Algorithms for the system authentication system [12]. Although the initial trial works well, the author decided to use minutia score matching for several reasons. One of the reasons is due to popularity of minutiae based matching compare to deep fingerprint feature matching. By implementing minutiae matching system, the program can be adapted easier in existing technology compare to the latter. The other reasoning is due to the lower quality of the image captured by the sensor. Some other feature other than minutiae may not be discovered which reduce the usability of the algorithm.

In the term of security, the system does not implement liveness test due to lack of time resources to defend itself against spoofing attack. The system also does not encrypt the minutiae during the transmission of the data from client to server which may make it less secure. The author also do not implement uncancellable biometrics as countermeasures because the fingerprint is already encrypted with one of the most secure encryption technique.

## Future Work

Despite the positive results, there is room for improvement. Enhancing the image quality for users with extreme skin conditions could reduce variability in performance metrics. Additionally, incorporating more advanced algorithms for image quality assessment could further refine the system's accuracy and reliability. Future work should focus on improving preprocessing techniques and exploring more advanced matching algorithms to reduce variability. Additionally, expanding the participant pool and diversifying the sample conditions can provide a more comprehensive evaluation. Enhancing security measures, such as implementing sophisticated liveness detection techniques, can further strengthen the system's defence against spoofing attacks.

# Conclusions

In this thesis, we have presented the development of a simplified and secure biometric authentication system leveraging Django as the server framework and affordable fingerprint sensors. Our primary objective was to create a cost-effective, easy-to-implement solution that ensures robust security, making biometric authentication accessible to a wider range of applications and users.

We began by exploring the current landscape of biometric authentication systems, identifying the limitations and challenges associated with high-cost sensors and complex implementations. Our solution addresses these issues by integrating low-cost hardware with open-source software, specifically the Django web framework, to provide a scalable and efficient platform for managing and matching fingerprint data.

Key components of our system include the application of fingerprint enhancement techniques using Gabor filters, extraction of fingerprint minutiae, and minutiae score matching. These steps were crucial in ensuring high accuracy and reliability in the identification and verification processes. The Gabor filter enhanced the fingerprint images, making minutiae extraction more precise, while the minutiae score matching algorithm effectively compared fingerprint templates for authentication purposes.

To further enhance security, we employed data hiding techniques within the minutiae points of fingerprint templates and utilized AES encryption for storing minutiae data in the database. The data hiding technique embedded additional information within the minutiae points, providing an extra layer of security and making it more difficult for unauthorized parties to tamper with the biometric data. AES encryption ensured that even if the database was compromised, the fingerprint minutiae would remain protected.

Our evaluation of the system covered both performance and security aspects. Quantitative metrics, such as accuracy, speed, and reliability, were rigorously tested to validate the effectiveness of the matching algorithm and overall system functionality. Qualitative testing by evaluating if the biometric system fulfil the requirements of standard biometric system according to the literature, provided insights into the usability and practical deployment of the system in real-world scenarios.

The results of our evaluation demonstrate that the proposed system achieves a high level of accuracy and security, comparable to more expensive solutions, while maintaining low implementation costs. This balance between cost and performance makes our biometric authentication system a viable option for various applications, including secure access control, identity verification, and personal device security.

In conclusion, this thesis contributes to the field of biometric authentication by offering a practical and secure solution that democratizes access to advanced security technologies. Future work can build upon this foundation by exploring additional biometric modalities, enhancing the robustness of the data hiding technique, and expanding the system's scalability for large-scale deployments.

Our work signifies a step forward in making biometric authentication more accessible and secure, paving the way for wider adoption and further innovation in the realm of biometric security.[Extra page inserted to ensure correct even-page footer for this section. Delete this when chapter is at least 2 pages long.]

References

[1] R. German and K. S. Barber, "Current biometric adoption and trends," *The University of Texas at Austin. Retrieved from identity. utexas. edu/assets/uploads/publications/Current-Biometric-Adoption-and-Trends. pdf*, 2017.

[2] D. Maltoni, D. Maio, A. K. Jain, and J. Feng, *Handbook of Fingerprint Recognition,* 3rd ed. Cham: Springer International Publishing; Imprint Springer, 2022.

[3] V. Matyáš and Z. Říha, "Biometric Authentication — Security and Usability," *Advanced Communications and Multimedia Security*, vol. 100, pp. 227–239, 2002, doi: 10.1007/978-0-387-35612-9\_17.

[4] A. Sarkar and B. K. Singh, "A review on performance,security and various biometric template protection schemes for biometric authentication systems," *Multimed Tools Appl*, vol. 79, 37-38, pp. 27721–27776, 2020, doi: 10.1007/s11042-020-09197-7.

[5] M. D. Femila and A. A. Irudhayaraj, "Biometric system," in *2011 3rd International Conference on Electronics Computer Technology, (ICECT 2011): Kanyakumari, India, 8 - 10 April 2011*, Kanyakumari, India, 2011, pp. 152–156.

[6] W. Y. Yau, T. P. Chen, and P. Morguet, "Benchmarking of Fingerprint Sensors," in *Biometric Authentication: ECCV 2004 International Workshop, BioAW 2004, Prague, Czech Republic, May 15th, 2004. Proceedings*, 2004, pp. 89–99. [Online]. Available: https://​link.springer.com​/​chapter/​10.1007/​978-​3-​540-​25976-​3\_​9

[7] M. Berndtsson, J. Hansson, B. Olsson, and B. Lundell, *Thesis projects: A guide for students in computer science and information systems,* 2nd ed. London: Springer, 2008.

[8] M. Joshi, B. Mazumdar, and S. Dey, "A comprehensive security analysis of match-in-database fingerprint biometric system," *Pattern Recognition Letters*, vol. 138, pp. 247–266, 2020, doi: 10.1016/j.patrec.2020.07.024.

[9] M. R. Islam, M. S. Sayeed, and A. Samraj, "A secured fingerprint authentication system," *Journal of Applied Sciences*, vol. 8, no. 17, pp. 2939–2948, 2008.

[10] N. K. Ratha, J. H. Connell, and R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Syst. J.*, vol. 40, no. 3, pp. 614–634, 2001, doi: 10.1147/sj.403.0614.

[11] L. Hong, Y. Wan, and A. Jain, "Fingerprint image enhancement: algorithm and performance evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 777–789, 1998, doi: 10.1109/34.709565.

[12] S. Bakheet, A. Al-Hamadi, and R. Youssef, "A Fingerprint-Based Verification Framework Using Harris and SURF Feature Detection Algorithms," *Applied Sciences*, vol. 12, no. 4, p. 2028, 2022, doi: 10.3390/app12042028.

[13] R. Thai, "Fingerprint Image Enhancement and Minutiae Extraction," Thesis, The University of Western Australia, Perth, Australia, 2003. Accessed: May 2 2024. [Online]. Available: https://​peterkovesi.com​/​studentprojects/​raymondthai/​RaymondThai.pdf

[14] R. J, K. B. Raja, and V. K. R, "Fingerprint Recognition Using Minutia Score Matching," *IJEST Volume 1 Issue 2.* [Online]. Available: http://​arxiv.org​/​pdf/​1001.4186

[15] Ł. Więcław, "A minutiae-based matching algorithms in fingerprint recognition systems," *Journal of Medical Informatics & Technologies*, Vol. 13, 2009. [Online]. Available: https://​yadda.icm.edu.pl​/​baztech/​element/​bwmeta1.element.baztech-​article-​PWA4-​0002-​0013

[16] M. Tico and P. Kuosmanen, "An algorithm for fingerprint image postprocessing," in *Conference record of the Thirty-Fourth Asilomar Conference on Signals, Systems & Computers: October 29 - November 1, 2000, Pacific Grove, California*, Pacific Grove, CA, USA, 2000, pp. 1735–1739.

[17] C. Pickering and J. Byrne, "The benefits of publishing systematic quantitative literature reviews for PhD candidates and other early-career researchers," *Higher Education Research & Development*, vol. 33, no. 3, pp. 534–548, 2014, doi: 10.1080/07294360.2013.841651.

[18] M. Martin, K. Štefan, and F. Ľubor, "Biometrics Authentication of Fingerprint with Using Fingerprint Reader and Microcontroller Arduino," *TELKOMNIKA*, vol. 16, no. 2, p. 755, 2018, doi: 10.12928/telkomnika.v16i2.7572.

[19] R. Singh, R. Shahabaje, N. Shikare, and K. Dhuri, "Fingerprint based Portable Attendance Monitoring System using Raspberry Pi Pico," in *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, Feb. 2024 - Mar. 2024, pp. 395–400.

[20] M. Dwi Rahmatya and M. F. Wicaksono, "Online Attendance with Python Face Recognition and Django Framework," *Sistemasi: Jurnal Sistem Informasi*, vol. 12, no. 3, p. 703, 2023, doi: 10.32520/stmsi.v12i3.2773.

[21] A. K. Jain, S. Prabhakar, and A. Ross, "Biometrics-based web access," *Transactions of the Institute of British Geographers, ns*, vol. 7, pp. 458–473, 1998.

[22] Django Project, *Django.* [Online]. Available: https://​www.djangoproject.com​/​ (accessed: May 5 2024).

[23] C. Ozgur, T. Colliau, G. Rogers, and Z. Hughes, "MatLab vs. Python vs. R," *Journal of Data Science*, vol. 15, no. 3, pp. 355–372, 2017, doi: 10.6339/JDS.201707\_15(3).0001.

[24] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Realtime Computer Vision with OpenCV," *Queue*, vol. 10, no. 4, pp. 40–56, 2012, doi: 10.1145/2181796.2206309.

[25] R. P. Ltd, *Raspberry Pi 4 Model B specifications – Raspberry Pi.* [Online]. Available: https://​www.raspberrypi.com​/​products/​raspberry-​pi-​4-​model-​b/​specifications/​ (accessed: May 5 2024).

[26] Hangzhou Grow Technology Co., Ltd, *R307 Fingerprint Module: User Manual.* [Online]. Available: https://​www.openhacks.com​/​uploadsproductos/​r307\_​fingerprint\_​module\_​user\_​manual.pdf (accessed: May 5 2024).

[27] *Welcome to Fritzing.* [Online]. Available: https://​fritzing.org​/​ (accessed: May 5 2024).

[28] I. Engineering, "What Powers Instagram: Hundreds of Instances, Dozens of Technologies," *Instagram Engineering*, 02 Dec., 2011. https://​instagram-engineering.com​/​what-​powers-​instagram-​hundreds-​of-​instances-​dozens-​of-​technologies-​adf2e22da2ad (accessed: May 6 2024).

[29] X. Yang, "Analysis of DBMS: MySQL Vs PostgreSQL," Kemi-Tornion ammattikorkeakoulu; Lapin ammattikorkeakoulu. [Online]. Available: https://​www.theseus.fi​/​handle/​10024/​27471

[30] PyPI, *adafruit-circuitpython-fingerprint.* [Online]. Available: https://​pypi.org​/​project/​adafruit-​circuitpython-​fingerprint/​ (accessed: May 6 2024).

[31] *NumPy -.* [Online]. Available: https://​numpy.org​/​ (accessed: May 7 2024).

[32] PyPI, *pillow.* [Online]. Available: https://​pypi.org​/​project/​pillow/​ (accessed: May 7 2024).

[33] OpenCV, *Home.* [Online]. Available: https://​opencv.org​/​ (accessed: May 7 2024).

[34] PyPI, *requests.* [Online]. Available: https://​pypi.org​/​project/​requests/​ (accessed: May 7 2024).

[35] S. van der Walt *et al.,* "scikit-image: image processing in Python," *PeerJ*, vol. 2, e453, 2014, doi: 10.7717/peerj.453.

[36] GitHub, *Utkarsh-Deshmukh/Fingerprint-Enhancement-Python: Using oriented gabor filters to enhance fingerprint images.* [Online]. Available: https://​github.com​/​Utkarsh-​Deshmukh/​Fingerprint-​Enhancement-​Python (accessed: May 7 2024).

[37] GitHub, *Utkarsh-Deshmukh/Fingerprint-Feature-Extraction: Extract minutiae features from fingerprint images.* [Online]. Available: https://​github.com​/​Utkarsh-​Deshmukh/​Fingerprint-​Feature-​Extraction (accessed: May 7 2024).

[38] S. Li, X. Chen, Z. Zheng, Z. Qian, G. Feng, and X. Zhang, "Data hiding in Fingerprint Minutiae Template for Privacy Protection," Oct. 2017. [Online]. Available: http://​arxiv.org​/​pdf/​1710.00516

[39] Injosoft AB, http://www.injosoft.se, *ASCII table - Table of ASCII codes, characters and symbols.* [Online]. Available: https://​www.ascii-code.com​/​ (accessed: May 21 2024).

[40] A. M. Abdullah, "Advanced encryption standard (AES) algorithm to encrypt and decrypt data," *Cryptography and Network Security*, vol. 16, no. 1, p. 11, 2017.

[41] *Implementation and performance analysis of pbkdf2, bcrypt, scrypt algorithms*: The Steering Committee of The World Congress in Computer Science, Computer …, 2016.

[42] PyPI, *cryptography.* [Online]. Available: https://​pypi.org​/​project/​cryptography/​ (accessed: May 23 2024).

[43] M. El-Abed and C. Charrier, "Evaluation of Biometric Systems," in *New Trends and Developments in Biometrics*, 2012, p 149 - 169. [Online]. Available: https://​hal.science​/​hal-​00990617

[44] P. J. Phillips, A. Martin, C. L. Wilson, and M. Przybocki, "An introduction evaluating biometric systems," *Computer*, vol. 33, no. 2, pp. 56–63, 2000, doi: 10.1109/2.820040.

[45] PyPI, *pypiqe.* [Online]. Available: https://​pypi.org​/​project/​pypiqe/​ (accessed: May 26 2024).

Appendices

* + 1. Title

Start each appendix on a new page. Place appendices in the same order as they are referred to in the body of the thesis. That is, the first appendix referred to should be Appendix A, the second appendix referred to should be Appendix B, and so on. Appendix formatting can be different to the main document. Refer to *Thesis PAM* for information about appendix figures and tables.