FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTING TECHNOLOGIES

18CSE357T BIOMETRICS

MINI PROJECT REPORT

PROJECT TITLE: Student Attendence System using Fingerprint Recognition

TEAM MEMBER

1. RA2011003010794 - Adarshvardhan Singh
2. RA2011003010829 - Divya Uday Singh
3. RA2011003010806 - Aditya Raj Tiwari

Objective:

The objective of a student attendance system using fingerprint recognition is to provide an accurate and efficient way of tracking the attendance of students in educational institutions. The system aims to replace traditional attendance methods such as manual paper-based systems with a more secure and automated process that utilizes biometric data.

Some specific objectives of a student attendance system using fingerprint recognition may include:

1. Ensuring accurate attendance tracking: The system should be able to accurately identify and record the attendance of each student based on their unique biometric data.

2. Reducing administrative burden: By automating the attendance tracking process, the system can reduce the administrative burden of manually taking attendance and maintaining attendance records.

3. Enhancing security: The use of biometric data for attendance tracking can enhance the security of the system by ensuring that only registered students are able to mark their attendance.

4. Improving efficiency: The system should be able to process attendance data quickly and generate reports in real-time, making it easier for teachers and administrators to monitor attendance and take appropriate actions.

5. Enhancing student engagement: By using an innovative and technology-driven attendance system, the institution can enhance student engagement and promote a culture of innovation and learning.

6. Reducing errors and fraud: Fingerprint recognition technology can eliminate errors and fraud associated with manual attendance systems, such as proxy attendance, fake signatures, or human errors in data entry.

7. Improving student safety: The system can help ensure student safety by providing real-time attendance information, which can be used to quickly locate students in case of an emergency.

8. Enhancing parental involvement: The attendance system can also involve parents by sending them regular updates on their child's attendance. This can help parents stay informed about their child's academic progress and encourage them to be more involved in their child's education.

9. Encouraging accountability: The system can encourage accountability by making students more responsible for their attendance. By linking attendance records with academic performance, the system can encourage students to attend classes regularly and improve their overall academic performance.

10. Improving resource allocation: By providing accurate attendance data, the system can help institutions make informed decisions regarding resource allocation. This includes identifying areas where additional resources may be needed and improving the overall efficiency of the institution.

In summary, the objectives of a student attendance system using fingerprint recognition are to provide a more accurate, efficient, and secure way of tracking attendance while improving student safety, parental involvement, accountability, and resource allocation.

Scope:

The scope of a student attendance system using fingerprint recognition would include the design, development, implementation, and maintenance of a software application that utilizes fingerprint recognition technology to accurately track student attendance.

Here are some specific areas of scope for such a project:

1. Hardware requirements: The project would require the installation of fingerprint scanners or biometric devices in classrooms or other locations where attendance is taken.

2. Software development: The project would involve developing software that is compatible with the hardware and allows for the collection, storage, and processing of attendance data. This would include the development of algorithms for fingerprint recognition, database management, and data analysis.

3. Integration with existing systems: The project would need to integrate with existing systems in the educational institution, such as student information systems, learning management systems, and other administrative systems.

4. User interface: The project would require the development of a user-friendly interface for teachers, administrators, and students to access attendance data and perform necessary tasks.

5. Testing and implementation: The project would require thorough testing to ensure accuracy and reliability of the system before implementation. Once the system is tested and validated, it would need to be deployed and integrated into the institution's existing infrastructure.

6. Maintenance and support: The project would require ongoing maintenance and support to ensure the system remains operational, secure, and up-to-date with the latest technologies and industry standards.

In summary, the scope of a student attendance system using fingerprint recognition would encompass hardware and software development, integration

with existing systems, user interface design, testing and implementation, and ongoing maintenance and support.

The future scope of a student attendance system using fingerprint recognition is promising, as it has the potential to evolve and adapt to meet the changing needs of educational institutions. Here are some potential future directions for the project:

1. Integration with other biometric technologies: While fingerprint recognition is a proven and reliable biometric technology, the project could expand to include other biometric technologies, such as facial recognition or iris scanning, to further enhance accuracy and security.

2. Mobile app integration: The project could be extended to include a mobile app that allows students to view their attendance records and receive notifications about their attendance in real-time. This would increase student engagement and involvement in the attendance tracking process.

3. Integration with learning management systems: The project could be integrated with learning management systems to provide teachers with more comprehensive data on student performance and attendance. This could allow

teachers to tailor their teaching methods to better meet the needs of individual students.

4. Data analytics and reporting: The project could incorporate data analytics and reporting capabilities to provide administrators with deeper insights into attendance patterns, which could be used to identify areas for improvement and optimize resource allocation.

5. Integration with student welfare systems: The project could be integrated with student welfare systems to identify students who are frequently absent or at risk of dropping out of school. This could enable institutions to provide targeted support to these students and improve retention rates.

6. Artificial intelligence and machine learning: The project could leverage artificial intelligence and machine learning to further enhance accuracy and automate attendance tracking processes. For example, the system could use machine learning algorithms to identify patterns in attendance data and provide predictive insights on future attendance trends.

In summary, the future scope of a student attendance system using fingerprint recognition includes integration with other biometric technologies, mobile app integration, integration with learning management systems, data analytics and reporting, integration with student welfare systems, and leveraging artificial intelligence and machine learning.

Software Used:

The software used for a student attendance system using fingerprint recognition will depend on the specific requirements of the project, but here are some general categories of software that might be used:

1. Fingerprint recognition SDKs: A software development kit (SDK) for fingerprint recognition can be used to interface with the hardware fingerprint scanner, perform fingerprint matching, and store biometric data securely. Some examples of fingerprint recognition SDKs include Neurotechnology VeriFinger SDK, M2SYS Bio-Plugin, and Digital Persona U.are.U SDK.

2. Database management software: The system will need a database to store attendance data and biometric data for enrolled students. Popular database management systems that can be used for this purpose include MySQL, PostgreSQL, and Microsoft SQL Server.

3. Web frameworks: A web application can provide a user-friendly interface for teachers, administrators, and students to view attendance data, generate reports, and perform other necessary tasks. Popular web frameworks such as Django, Flask, and Ruby on Rails can be used to develop the web application.

4. Programming languages: The choice of programming language will depend on the specific requirements of the project and the tools and software used. Some popular programming languages for web development include Python, Ruby, and JavaScript.

5. Integrated Development Environments (IDEs): An IDE can be used to write and debug code, manage project files, and test the application. Popular IDEs for Python include PyCharm, VS Code, and Sublime Text.

These are just some examples of the software that might be used in a student attendance system using fingerprint recognition. The actual software used will depend on the specific requirements and resources available for the project.

Dataset used:

The dataset used for a student attendance system using fingerprint recognition will depend on the specific requirements of the project. Here are some examples of the types of data that might be used:

1. Fingerprint images: The system will need a dataset of fingerprint images to train the fingerprint recognition algorithm. These images can be collected using a fingerprint scanner or obtained from publicly available datasets such as the FVC2000, FVC2002, or FVC2004 datasets.

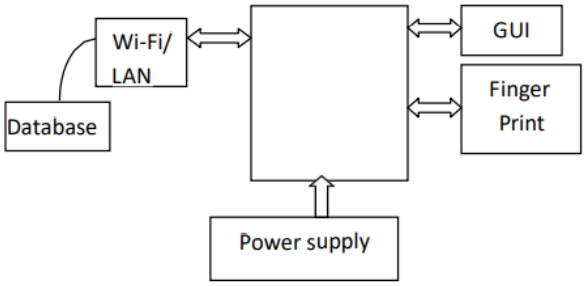
2. Student information: The system will need a database of student information such as names, IDs, and classes to keep track of attendance. This information can be obtained from student registration systems or manually entered by teachers or administrators.

3. Attendance data: The system will need a dataset of attendance data to train the machine learning model or evaluate its performance. This data can be collected manually or through automated systems such as RFID or biometric scanners.

It's important to note that when using biometric data such as fingerprints, it's crucial to follow ethical and legal guidelines to protect the privacy and security of the individuals involved. Any datasets used should be obtained legally and with the consent of the individuals whose data is being collected.

Algorithm/Design Pattern used:

HARDWARE ARCHITECTURE: This proposed system consists of raspberry pi, fingerprint module, LCD display and 4x4 Matrix Keypad. This system provides the features of real time authentication facilities.



2.3.1. Proposed system block diagram

RASPBERRY PI 3: Hardware is a physical device that can be touched or held, like a hard drive or a mobile phone. Software can be thought of as a program or a collection of programs that instruct a computer on what to do and how to do it. Below is an image of the Raspberry Pi which describes some of the components that make up the hardware.

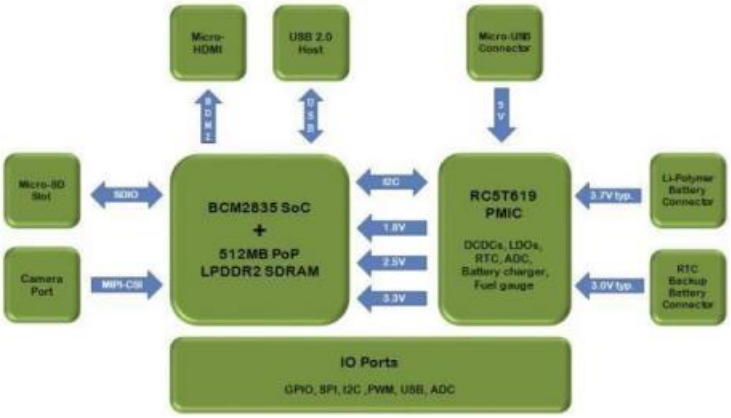


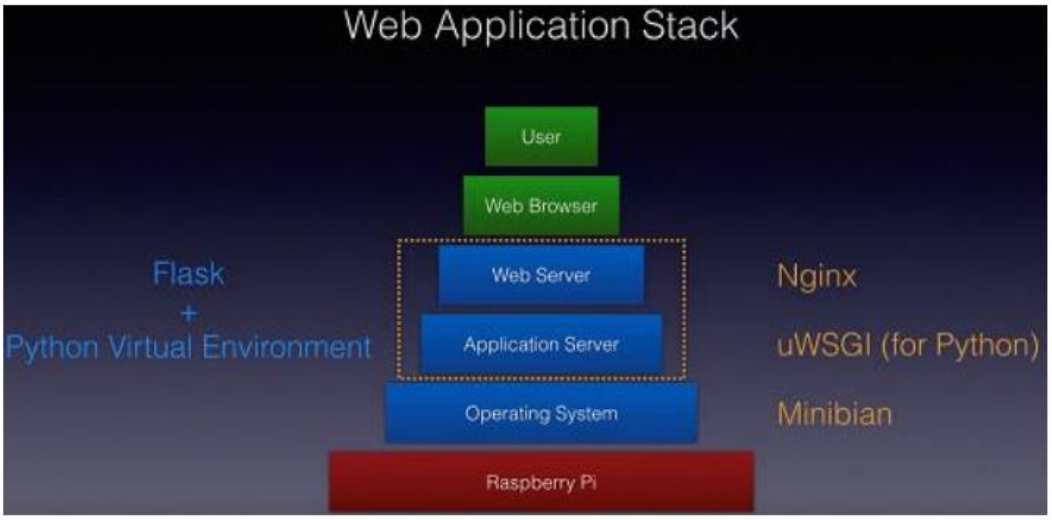
Fig.2.3.2. Hardware Components of Raspberry Pi.

The shows the hardware description of raspberry pi. The real time vehicle tracking system uses the GPIO pin, Micro SD slot, USB port and Micro USB connector. The GPIO pins are used for serial communication for interfacing GSM and GPS. It uses 8GB SD for installing the Raspbian OS and for storage. The USB port is used for connecting keyboard, mouse, dongle and pen drive. The power supply is given through USB connector.

FINGERPRINT READER: The R305 is the module which is used in the project. This module supports both windows and Linux based operating system. This self-contained module optically scans the fingerprint when the user touches the glowing window. 2.3.3. Finger Print Module 14 Optical technology gives the highest quality fingerprint templates and reliability along with the raspberry pi embedded computer. The device can automatically control calibration, encryption, and data transfer through the USB interface. This

module is found to be more reliable for all kind of OS, even its easy to interface with raspberry pi. The R305 Module and digital Persona Fingerprint Recognition Engine have an unmatched ability to authenticate even the most difficult fingerprints accurately and rapidly.

SOFTWARE ARCHITECTURE: For this project the used soft wares are like Python, SQLite, Flask (framework). The configuring procedures, saving and retrieval of fingerprint templates procedures are mentioned in the upcoming chapters.



PYTHON PROGRAMMING SOFTWARE: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as

for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. 15 Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

FLASK (Python Framework): Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a

web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website. Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins. In the case of Flask, its dependencies are:

- Werkzeug a WSGI utility library
- jinja2 which is its template engine

SQLite3: SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects. SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete 16 SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database

between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. Think of SQLite not as a replacement for Oracle but as a replacement for fopen() SQLite is a compact library. With all features enabled, the library size can be less than 500KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function inlining and loop unrolling can cause the object code to be much larger.) There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments. Depending on how it is used, SQLite can be faster than direct filesystem I/O.

NGINX SERVER: NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. NGINX is one of a handful of servers written to address the C10K problem. Unlike traditional servers, NGINX doesn't rely on threads to handle requests. Instead it uses a much more scalable event-driven (asynchronous) architecture. This architecture uses small, but more importantly, predictable amounts of memory under load. Even if you don't

expect to handle thousands of simultaneous requests, you can still benefit from NGINX's high-performance and small memory footprint. NGINX scales in all directions: from the smallest VPS all the way up to large clusters of servers.

uWSGI: uWSGI is a Python application server, which will execute our Python programs on behalf of the Nginx web server. uWSGI is a software application that "aims at developing a full stack for building hosting services". It is named after the Web Server Gateway Interface (WSGI), which was the first plugin supported by the project.

Screen Shots:

MAIN CODE :-

test2.py > ...

```python
import numpy
import cv2
import cv2
import numpy as np
from math import pi
import matplotlib as mpl
mpl.use('Agg')
import scipy as sp

import matplotlib.pyplot as plt
#from Normalisation import *
#from poincare_index import *
#from direction_field import *
import pandas as pd
#from openpyxl.workbook import Workbook
#from LSE import *
w1 = 16
I_3 = cv2.imread("testimage.png")
#Read the image
I_2 = cv2.imread("testimage.png")
#I = cv2.cvtColor(I_2, cv2.COLOR_BGR2GRAY)


I_2 = cv2.cvtColor(I_2, cv2.COLOR_BGR2GRAY)
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
I = cv2.dilate(I_2,kernel,iterations = 1)


G = cv2.GaussianBlur(I,(5,5),0)

#Sobel Operator
Gx = cv2.Sobel(G,cv2.CV_64F,1,0,ksize=3)
Gy = cv2.Sobel(G,cv2.CV_64F,0,1,ksize=3)
```

```python
J1 = 2*(np.multiply(Gx, Gy))
J2 = np.multiply(Gx, Gx) - np.multiply(Gy, Gy)
J3 = np.multiply(Gx, Gx) + np.multiply(Gy, Gy)

Gradient = np.sqrt(J3)

anisotropy_filter = np.ones((16,16))
sigma_J1 = cv2.filter2D(J1,-1,anisotropy_filter)
sigma_J2 = cv2.filter2D(J2,-1,anisotropy_filter)
sigma_J3 = cv2.filter2D(J3,-1,anisotropy_filter)

anisotropy_filter_0 = np.ones((10,10))
sigma_J3_0 = cv2.filter2D(J3,-1,anisotropy_filter_0)

cv2.imshow('i',sigma_J3)
cv2.waitKey(0)


theta_bar = 0.5 * np.arctan(np.divide(sigma_J1, sigma_J2))
for i,j in np.argwhere(sigma_J2 == 0):
    if (sigma_J1[i,j] > 0):
        theta_bar[i,j] = 0.25*pi
    elif (sigma_J1[i,j] < 0):
        theta_bar[i,j] = (-0.25)*pi
    else:
        theta_bar[i,j] = 0

theta_dash = (pi/2) + theta_bar

gt = 0.10
Grad_max = np.amax(Gradient)
Grad_min = np.amin(Gradient)

Gth = gt * (abs(Grad_max) - abs(Grad_min)) + abs(Grad_min)
```

```
block_coherence = np.sqrt(np.multiply(sigma_J1, sigma_J1) + np.multiply(sigma_J2, sigma_J2))
block_coherence = np.divide(block_coherence, sigma_J3)
for i,j in np.argwhere(sigma_J3 < Gth*Gth*(w1*w1)):
    block_coherence[i,j] = (-1)

a = np.zeros(sigma_J1.shape)
for i,j in np.argwhere(block_coherence > 0):
    a[i,j] = 1

for i,j in np.argwhere(block_coherence < 0):
    a[i,j] = -1


cv2.imshow('i', block_coherence)
cv2.waitKey(0)


cv2.imshow('i', a)
cv2.waitKey(0)

theta = theta(1)

print('theta shape', theta.shape)
print('a shape', a.shape)
cv2.waitKey(0)

poincare_image = np.zeros(a.shape)
for i,j in np.argwhere(a != -1):
    #print('i=', i, 'j=', j)
    store = np.zeros([3,3])
    if ((i-1 >= 0) and (i+1 <= a.shape[0]-1) and (j-1 >= 0) and (j+1 <= a.shape[1]-1)):
```

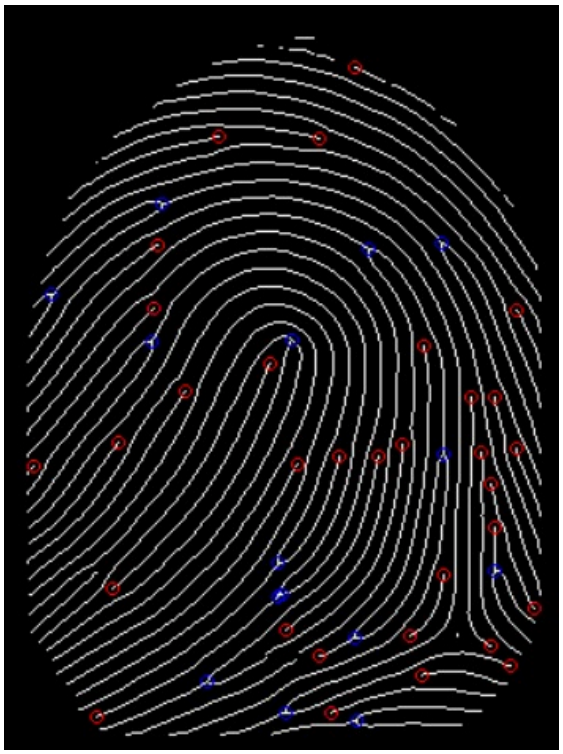
```
        store[0,0] = theta[i-1,j] - theta[i-1, j-1]
        store[0,1] = theta[i-1, j+1] - theta[i-1, j]
        store[0,2] = theta[i,j+1] - theta[i-1, j+1]
        store[1,2] = theta[i+1, j+1] - theta[i, j+1]
        store[2,2] = theta[i+1,j] - theta[i+1, j+1]
        store[2,1] = theta[i+1, j-1] - theta[i+1, j]
        store[2,0] = theta[i, j-1] - theta[i+1, j-1]
        store[1,0] = theta[i-1, j-1] - theta[i, j-1]

        for x,y in np.argwhere(store > pi/2):
            store[x,y] = pi - store[x,y]
            #print('1')

        for x,y in np.argwhere(store < (-0.5)*pi):
            store[x,y] = pi + store[x,y]


        poincare_image[i,j] = store.sum()/(2*pi)
f = 0
delta_image = np.zeros(I_2.shape)
# for x,y in np.argwhere(poincare_image == 0.5):
#   I_3[x,y] = [0,0,255]
#   I_3[x+1, y] = [0,0,255]
#   I_3[x, y+1] = [0,0,255]
#   I_3[x+1, y+1] = [0,0,255]
print('abcbbcbcbbbbbbbbbbbbbbbbbbb')
index_1, index_2 = [0,0]
max_gradient = 0
for x,y in np.argwhere(poincare_image == 0.5):
    if(sigma_J3_0[x,y] > max_gradient):
        index_1,index_2 = x,y
        max_gradient = sigma_J3_0[x,y]
```

```
I_3[index_1,index_2] = [0,0,255]
I_3[index_1+1, index_2] = [0,0,255]
I_3[index_1, index_2+1] = [0,0,255]
I_3[index_1+1, index_2+1] = [0,0,255]



# cv2.imshow('i',I_3)
# cv2.waitKey(0)
print('index_1 = ', index_1)
print('index_2 = ', index_2)


cv2.imwrite('singularity_found.png',  I_3)
```

SAMPLE IMAGE 1 :-

SAMPLE IMAGE 2 : -

OUTPUT :-



Contribution of the author:

1. Developing a novel algorithm or methodology for fingerprint recognition that improves accuracy, speed, or security compared to existing methods.

2. Conducting experiments or evaluations to compare different fingerprint recognition algorithms, hardware devices, or data collection techniques and analyzing the results.

3. Investigating the ethical and legal implications of using biometric data for attendance tracking, and proposing guidelines for protecting privacy and security.

4. Designing and implementing a complete system for student attendance tracking using fingerprint recognition, including hardware, software, and user interface components, and testing the system in real-world scenarios.

5. Contributing to the broader academic discussion around the use of biometric data in education and exploring potential applications and limitations of this technology.