P- 2160

ATTENDANCE SYSTEM USING BIOMETRICS

A PROJECT REPORT

Submitted by

HARISHANKAR.V	71204205010
SAI PRADEEP.G	71204205040
SANTHANAM.G.C	71204205041
SARAVANAN.A.K	71204205045

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2008

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"ATTENDANCE SYSTEM USING BIOMETRICS"** is the bonafide work of "HARISHANKAR.V,SAI PRADEEP.G,SANTHANAM.G.C and SARAVANAN.A.K" who carried out the project work under my supervision.

SIGNATURE
Dr.S.Thangasamy

DEAN OF THE DEPARTMENT

Dept of Information Technology

Kumaraguru College of Technology,

Coimbatore – 641 006.

SIGNATURE
S.Vani ,M.E Lecturer

SUPERVISOR

Dept of Information Technology,

Kumaraguru College of Technology,

Coimbatore – 641 006.

The candidates with University Register No 71204205010, 71204205040, 71204205041 and 71204205046 were examined by us in the project viva-voice examination held on 23/04/08

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our sincere thanks to our Chairman **Padmabhushan Arutselvar Dr. N. Mahalingam B.Sc, F.I.E** and Vice chairman **Prof. K. Arumugam B.E., M.S., M.I.E.,** for all their support and ray of strengthening hope extended. We are immensely grateful to our Principal **Dr. Joseph V. Thanikal M.E., Ph.D., PDF., CEPIT.,** and our Project Co-ordinator **Mr. K. R. Baskaran B.E., M.S** for his invaluable support to the outcome of this project.

We are deeply obliged to our project guide **Mrs. S.Vani M.E.,** lecturer, Department of Information Technology for her valuable advice and useful suggestions throughout this project.

We also thank **Mr. E.A.Vimal M.E.,** lecturer , our class advisor, Department of Information Technology who rendered his valuable guidance and support to our project work.

We thank the teaching and non-teaching staff of our department for providing us the technical support in the duration of our project.

We express our humble gratitude and thanks to our beloved parents and family members who have supported and helped us to complete the project and our friends, for lending us valuable tips, support and co-operation throughout the project work.

DECLARATION

We,

HARISHANKAR.V	**71204205010**
SAI PRADEEP.G	**71203205040**
SANTHANAM.G.C	**71204205041**
SARAVANAN.A.K	**71204205046**

hereby declare that the project entitled **"ATTENDANCE SYSTEM USING BIOMETRICS"**, submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of Department of Information Technology, Kumaraguru college of Technology, Coimbatore.

Place: Coimbatore
Date: 9/4/08

V. Harishankar
[HariShankar.V]

G. Saipradeep
[Sai Pradeep.G]

G.C. Santhanam
[Santhanam.G.C]

A.K Saravanan
[Saravanan.A.K]

Project Guided by
S. Vani

[Mrs. S.Vani , M.E]

ABSTRACT

Biometrics technology which uses physical or behavioral characteristics to identify users has come to attract increased attention as a means of reliable personal authentication that helps establish the identity of an actual user. Among various modalities of biometrics, fingerprints are known to have the longest history of actual use in law enforcement applications with proven performance. This project surveys the state of the art in fingerprint identification and image processing technology to implement Attendance Maintenance to various organizations as well as educational Institutes. By using this biometric authentications we provide more security. Authentication is the process of verifying the identity of a user or a computing device in a networked environment.

Biometric authentication is based on an inherent physical trait or a characteristic such as fingerprints, retinal patterns, hand geometry, voice recognition or facial recognition, and offers a high degree of assurance that a person actually is who they claim to be. Fingerprint is the cheapest, fastest, most convenient and most reliable way to identify someone. That's why fingerprint alone has 2/3 of the biometric world market (according to an International Biometric Group independent report). And the tendency, due to scale, easiness and the existing foundation, is that the use of fingerprint will only increase. Cars, cell phones, PDAs, personal computers and dozens of products and devices are using fingerprints more and more. The comparison of the stored image is done with the help of Image Processing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

FC	: Finger Code
AWT	: Abstract Window Toolkit
GUI	: Graphical User Interface
AAD	: Average Absolute Deviation
DPI	: Dots Per Inch
RPL	: Refence Point Locator
IRD	: Inter Ridge Distance
CTT	: Code Testing Technique

1. INTRODUCTION

1.1 GENERAL:

With identity fraud in our society reaching unprecedented proportions and with an increasing emphasis on the emerging automatic personal identification applications, biometrics-based verification, especially fingerprint-based identification, is receiving a lot of attention. There are two major shortcomings of the traditional approaches to fingerprint representation. For a considerable fraction of population, the representations based on explicit detection of complete ridge structures in the fingerprint are difficult to extract automatically. The proposed filter-based algorithm uses a bank of Gabor filters to capture both local and global details in a fingerprint as a compact fixed length Finger Code. The fingerprint matching is based on the Euclidean distance between the two corresponding Finger Codes and hence is extremely fast. We are able to achieve a verification accuracy which is only marginally inferior to the best results of minutiae-based algorithms published in the open literature. The widely used minutiae-based representation does not utilize a significant component of the rich discriminatory information available in the fingerprints. Local ridge structures cannot be completely characterized by minutiae. Further, minutiae-based matching has difficulty in quickly matching two fingerprint images containing different number of unregistered minutiae points. Our system performs better than a state-of-the-art minutiae-based system when the performance requirement of the application system does not demand a

very low false acceptance rate. Finally, we show that the matching performance can be improved by combining the decisions of the matchers based on complementary (minutiae-based and filter-based) fingerprint information.

1.2 PROBLEM DEFINITION:

The main objective of our project is to design a system that prevents offline dictionary attacks mounted by either of the two servers. However, the use of passwords has intrinsic weaknesses. It is a well-known problem that human-user-chosen passwords are inherently weak since most users choose short and easy to remember passwords. In particular, passwords are normally drawn from a relatively small dictionary, so it allows for brute-force dictionary attacks, where an attacker enumerates every possible password in the dictionary to determine the actual password.

Dictionary attacks can be mounted online or offline. In an online dictionary attack, attackers attempt to log in to a server by trying all possible passwords from the dictionary until they find a correct one. In an offline dictionary attack, attackers record a past successful login session between a user and a server and then check all the passwords in the dictionary against the login transcript. Online dictionary attacks can be easily thwarted at the system level by limiting the number of unsuccessful login attempts made by a user. In contrast, offline dictionary attacks are notoriously harder to deal with. As a result, tremendous effort has been dedicated to countering offline dictionary attacks in password systems. The present single server authentication architecture was analyzed and used as a foundation for the proposed system

1.3 SYSTEM STUDY:

System analysis and system design are the main components regarding system development. System analysis is the process of gathering and interpreting facts and it diagnosis the problem. This improves the new system from the existing one. System analysis is a problem solving activity that requires intensive communication between system users and the system developers

1.4 EXISTING SYSTEM:

The popular fingerprint representation schemes have evolved from an intuitive system design tailored for fingerprint experts who visually match the fingerprints. These schemes are either based on predominantly local landmarks (e.g., minutiae-based fingerprint matching systems) or exclusively global information (fingerprint classification based on the Henry system). The minutiae-based automatic identification techniques first locate the minutiae points and then match their relative placement in a given finger and the stored template . A good quality fingerprint contains between 60 and 80 minutiae, but different fingerprints have different number of minutiae. The variable sized minutiae-based representation does not easily lend itself to indexing mechanisms. Further, typical graph-based, and point pattern-based approaches to match minutiae from two fingerprints need to align the unregistered minutiae patterns of different sizes which makes them computationally expensive. Correlation- based techniques match the global patterns of ridges and valleys to determine if the ridges align. The global approach to fingerprint representation is typically used for indexing , and does not offer very good individual discrimination. Further, the indexing

efficacy of existing global representations is poor due to a small number of categories that can be effectively identified and a highly skewed distribution of the population in each category. The natural proportion of fingerprints belonging to categories whorl (whorl and double loop put together), loop (right and left loop put together), and arch (arch and tented arch put together), is 0.279, 0.655, and 0.066, respectively. Both these approaches utilize representations which cannot be easily extracted from poor quality fingerprints.

1.5. PROPOSED SYSTEM:

The proposed scheme of feature extraction tessellates the region of interest of the given fingerprint image with respect to a reference point . A feature vector is composed of an ordered enumeration of the features extracted from the (local) information contained in each sub image (sector) specified by the tessellation. Thus, the feature elements capture the local information and the ordered enumeration of the tessellation captures the invariant global relationships among the local patterns. The local discriminatory information in each sector needs to be decomposed into separate components. Gabor filter banks are a well-known technique to capture useful information in specific band pass channels as well as to decompose this information into biorthogonal components in terms of spatial frequencies. A feature vector, which we call Finger Code, is the collection of all the features (for every sector) in each filtered image. These features capture both the global pattern of ridges and valleys and the local characteristics. Matching is based on the Euclidean distance between the Finger Codes.

2. LITERATURE REVIEW

2.1. JAVA:

Java, the Object Oriented Programming language developed by Sun Microsystems. Java has a good chance to be the first really successful new computer language in several decades. Advanced programmers like it because it has a clean, well-designed definition. Business likes it because it dominates an important new application, Web programming.

SEVERAL IMPORTANT FEATURES:

A Java program runs exactly the same way on all computers. Most other languages allow small differences in interpretation of the standards.

- It is not just the source that is portable. A Java program is a stream of bytes that can be run on any machine. An interpreter program is built into Web browsers, though it can run separately. Java programs can be distributed through the Web to any client computer.
- Java applets are safe. The interpreter program does not allow Java code loaded from the network to access local disk files, other machines on the local network, or local databases. The code can display information on the screen and communicate back to the server from which it was loaded.

A group at Sun reluctantly invented Java when they decided that existing computer languages could not solve the problem of distributing applications over the network. C++ inherited many unsafe practices from the old C

language. Basic was too static and constrained to support the development of large applications and libraries.

Today, every major vendor supports Java. Netscape incorporates Java support in every version of its Browser and Server products. Oracle will support Java on the Client, the Web Server, and the Database Server. IBM looks to Java to solve the problems caused by its heterogeneous product line.

The Java programming language and environment is designed to solve a number of problems in modern programming practice. It has many interesting features that make it an ideal language for software development. It is a high-level language that can be characterized by all of the following buzzwords:

2.2. FEATURES:

Sun describes Java as

- Simple
- Object-oriented
- Distributed
- Robust
- Secure
- Architecture Neutral
- Portable
- Interpreted
- High performance
- Multithreaded
- Dynamic.

- **JAVA IS SIMPLE**

What it means by simple is being small and familiar. Sun designed Java as closely to C++ as possible in order to make the system more comprehensible, but removed many rarely used, poorly understood, confusing features of C++. These primarily include operator overloading, multiple inheritance, and extensive automatic coercions. The most important simplification is that Java does not use pointers and implements automatic garbage collection so that we don't need to worry about dangling pointers, invalid pointer references, and memory leaks and memory management.

- **JAVA IS OBJECT ORIENTED**

This means that the programmer can focus on the data in his application and the interface to it. In Java, everything must be done via method invocation for a Java object. We must view our whole application as an object; an object of a particular class. .

- **JAVA IS DISTRIBUTED**

Java is designed to support applications on networks. Java supports various levels of network connectivity through classes in java. net. For instance, the URL class provides a very simple interface to networking. If we want more control over the downloading data than is through simpler URL methods, we would use a URLConnection object which is returned by a URL URL. open

Connection() method. Also, you can do your own networking with the Socket and Server Socket classes.

- **JAVA IS ROBUST**

Java is designed for writing highly reliable or robust software. Java puts a lot of emphasis on early checking for possible problems, later dynamic (runtime) checking, and eliminating situations that are error prone. The removal of pointers eliminates the possibility of overwriting memory and corrupting data.

- **JAVA IS SECURE**

Java is intended to be used in networked environments. Toward that end, Java implements several security mechanisms to protect us against malicious code that might try to invade your file system. Java provides a firewall between a networked application and our computer.

- **JAVA IS ARCHITECTURE-NEUTRAL**

Java program are compiled to an architecture neutral byte-code format. The primary advantage of this approach is that it allows a Java application to run on any system that implements the Java Virtual Machine. This is useful not only for the networks but also for single system software distribution. With the multiple flavors of Windows 95 and Windows NT on the PC, and the new PowerPC Macintosh, it is becoming increasing difficult to produce software that runs on all platforms.

- **JAVA IS PORTABLE**

The portability actually comes from architecture-neutrality. But Java goes even further by explicitly specifying the size of each of the primitive data types to eliminate implementation-dependence. The Java system itself is quite portable. The Java compiler is written in Java, while the Java run-time system is written in ANSI C with a clean portability boundary.

- **JAVA IS INTERPRETED**

The Java compiler generates byte-codes. The Java interpreter executes the translated byte codes directly on system that implements the Java Virtual Machine. Java's linking phase is only a process of loading classes into the environment.

- **JAVA IS HIGH PERFORMANCE**

Compared to those high-level, fully interpreted scripting languages, Java is high-performance. If the just-in-time compilers are used, Sun claims that the performance of byte-codes converted to machine code are nearly as good as native C or C++. Java, however, was designed to perform well on very low-power CPUs.

- **JAVA IS MULTITHREADED**

Java provides support for multiple threads of execution that can handle different tasks with a Thread class in the java.lang Package. The thread class supports methods to start a thread, run a

thread, stop a thread, and check on the status of a thread. This makes programming in Java with threads much easier than programming in the conventional single-threaded C and C++ style.

- **JAVA IS DYNAMIC**

Java language was designed to adapt to an evolving environment. It is a more dynamic language than C or C++. Java loads in classes, as they are needed, even from across a network. This makes an upgrade to software much easier and effectively.With the compiler, first we translate a program into an intermediate language called Java byte codes ---the platform-independent codes interpreted by the interpreted on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happen s just once; interpretation occurs each time the program is executed.Java byte codes can be thought as the machine code instructions for the Java Virtual Machine (JVM). Every Java interpreter, whether it's a development tool or a web browser that can run applets, is an implementation of the Java Virtual Machine.

THE JAVA PLATFORM:

A platform is the hardware or software environment in which a program runs. Most Platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's software –only platform that runs on top of other hardware-based platforms.

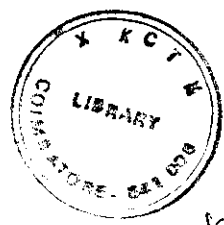
The Java platform has two components:

1. The Java Virtual Machine (JVM).
2. The Java Application Programming Interfaces (Java API).

The JVM has been explained above. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI). The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages.

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.



2.3. OVERVIEW OF THE SWING:

The Swing package is part of Java Foundation Classes (JFC) in the Java platform. The JFC encompasses a group of features to help people build GUIs; Swing provides all the components from buttons to split panes and tables.

The Swing package was first available as an add-on to JDK 1.1. Prior to the introduction of the Swing package, the Abstract Window Toolkit (AWT) components provided all the UI components in the JDK1.0 and 1.1 platforms. Although the Java2 Platform still supports the AWT components, we strongly encourage using Swing components instead. You can identify Swing components because their names start with J. The AWT button class, for example, is named Button, whereas the Swing button class is named JButton. In addition, the AWT components are in the java.awt package, whereas the swing components are in the javax.swing package.

As a rule, programs should not use "heavyweight" AWT components alongside Swing components. Heavyweight components include all the ready-to-use AWT components, such as Menu and ScrollPane, and all components that inherit from the AWT canvas and Panel classes. When Swing components (and all other "lightweight" components) overlap with heavyweight components, the heavyweight component is always painted on top.

SWING FEATURES AND CONCEPTS:

It introduces Swing's features and explains all the concepts we need to be able to use Swing components effectively.

SWING CONCEPTS AND CONTAINMENT HIERACHY:

Swing provides many standard GUI components such as buttons, lists, menus and textareas, which we combine to create our program's GUI. It also includes containers such as windows and tool bars.

LAYOUT MANAGEMENT:

Containers use layout managers to determine the size and position of the components they contain. Borders affect the layout of Swing GUIs by making Swing components larger. We can also use invisible components to affect layout.

EVENT HANDLING:

Event handling is how programs respond to external events, such as the user pressing a mouse button. Swing programs perform all their painting and event handling in the event-dispatching thread.

PAINTING:

Painting means drawing the components on-screen. Although it's easy to customize a component's painting, most programs don't do anything more complicated than customizing a component's border.

THREADS AND SWING:

If we do something to a visible component that might depend on or affect its state, then we need to do it from the event-dispatching thread. This isn't an issue for many simple programs, which generally refer to components only in event -handling code. However, other programs need to use the invoke Later method to execute component-related calls in the event-dispatching thread.

MORE SWING FEATURES AND CONCEPTS:

These are some of the major concepts we need to know to build Swing GUIs—the containment hierarchy, layout management, event handling, painting, and threads. These are some of the other important Swing features. They are

- Features that JComponent provides
- Icons
- Actions
- Pluggable Look and Feel support
- Support for assistive technologies

FEATURES THAT JCOMPONENTS PROVIDE:

Except for the top-level containers, all components that begin with J inherit from the JComponent class. They get many features from JComponent, such as the ability to have borders, tool tips, and a configurable look and feel. They also inherit many convenient methods.

ICONS:

Many Swing components—notably buttons and labels—can display images. We specify these images as Icon objects.

ACTIONS:

With Action objects, the Swing API provides special support for sharing data and state between two or more components that can generate action events. For example, if we have a button and a menu item that perform the same function, consider using an Action object to coordinate the text, icon, and enabled state for the two components.

PLUGGABLE LOOK AND FEEL:

This Feature enables the user to switch the look- and-feel of Swing components without restarting the application. The Swing library supports cross-platform look-and-feel---also Java look-and-feel--- that remains the same across all platforms wherever the program runs. The native look-and-feel is native to whatever particular system on which the program happens to be running,

including Windows and Motif. The Swing library provides an API that gives great flexibility in determining the look-and-feel of applications. It also enables you to create our own look-and-feel.

2.4.SOME SWING COMPONENTS OF JAVA:

JScrollPane:

Provides a scrollable view of a lightweight component. It is used to display a child component with a built-in scrolling facility. The scrolling of a child component, when its size is larger than the available view port, is performed in horizontal or vertical directions by using the scrollbars associated with the scroll pane. Scroll panes are very easy to implement because the adjustment events fired by the scrollbars are already taken care of by the scrollpane object. A Swing scroll pane is an object of type JScrollPane that extends from the class JComponent.

JButton:

Swing buttons are represented by the objects of class JButton, and each button is basically an implementation of a push-type button. Unlike AWT buttons, Swing buttons can be displayed with text labels as well as icons. We can also set different icon for different states of the buttons by supporting methods.

JFrame:

An extended version of java.awt.Frame that adds support for the JFC/Swing components architecture. The JFrame class is slightly incompatible with Frame. Like all other JFC/Swing top-level containers, a JFrame contains a JRootPane as its only child. The content pane provided by the root pane should, as a rule, contain all the non-menu components displayed by the JFrame.

2.5. DATABASE DESIGN:

TABLE NAME : student

Filed Name	Data Type
Regno	Text(50)
Name	Text(50)
Dept	Text(50)
Address	Text(250)
Facepath	Text(250)

TABLE NAME : Intime

Filed Name	Data Type
Regno	Text(50)
Time	Text(50)
Date	Text(50)

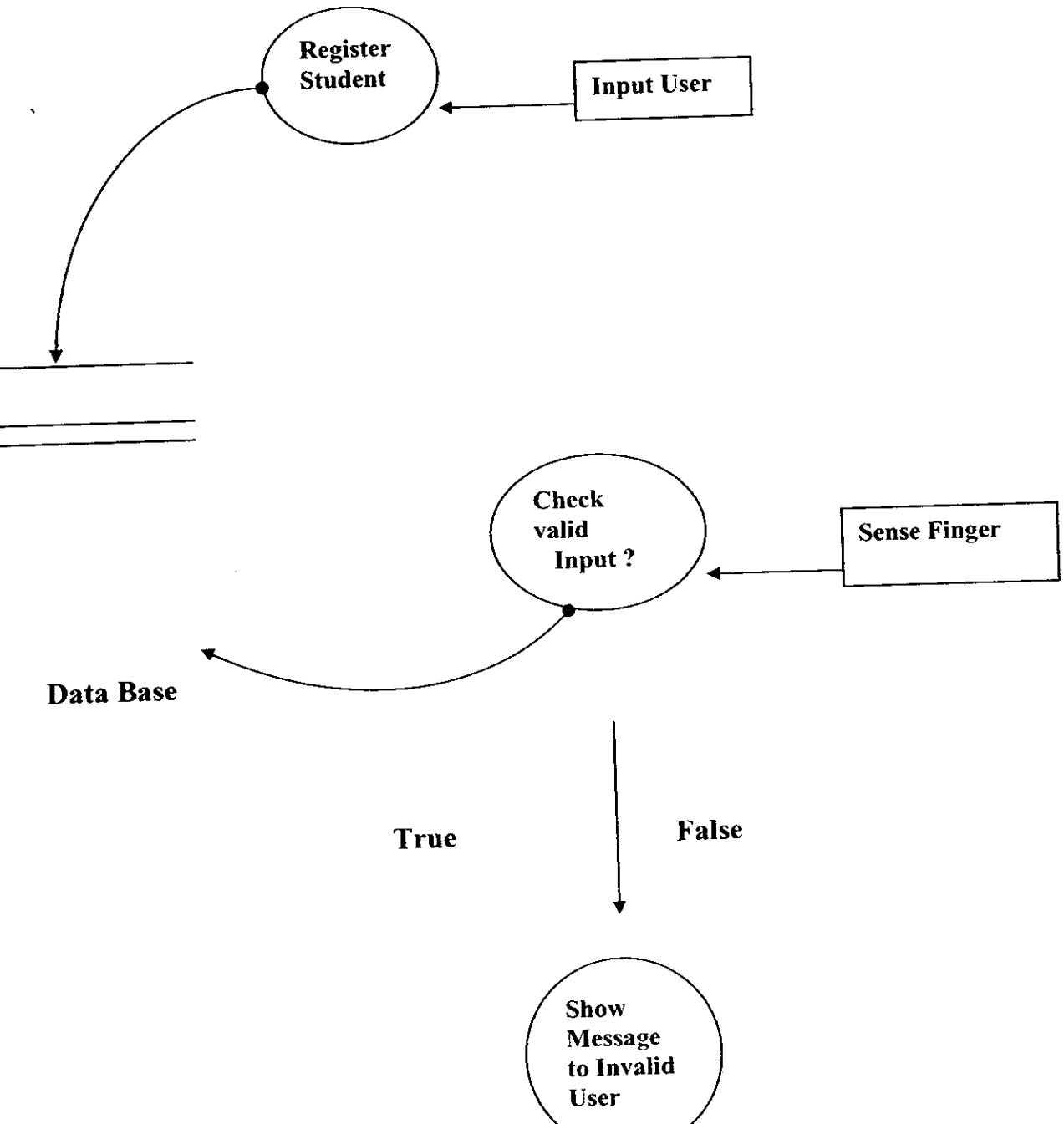
TABLE NAME : Outtime

Filed Name	Data Type
Regno	Text(50)
Time	Text(50)
Date	Text(50)

TABLE NAME : entry

Filed Name	Data Type
Regno	Text(50)
Facepath	Text(50)

2.6.DFD DIAGRAM:



2.7.HARDWARE SPECIFICATIONS:

Processor : Pentium IV

Speed : 500 MHz

Main Memory : 256 MB RAM

Hard Disk : 40 GB

Sensor : Finger Print Sensor

Floppy Disk : 1.44 MB

Display : 14' SVGA Color Monitor

Keyboard : 104 Keys Keyboard

Mouse : Samsung Scroll Mouse

2.8.SOFTWARE SPECIFICATIONS:

Operating System : Windows (Service Pack 2)

Software : JAVA ,EV TOOLS

Front End Tool : SWINGS

Back End : MS ACCESS

3. DETAILS OF METHODOLOGY EMPLOYED

3.1. FILTER-BASED FEATURE EXTRACTION

It is desirable to obtain representations for fingerprints which are scale, translation, and rotation invariant. Scale invariance is not a significant problem since most fingerprint images could be scaled as per the dpi specification of the sensors. The rotation and translation invariance could be accomplished by establishing a reference frame based on the intrinsic fingerprint characteristics which are rotation and translation invariant. It is also possible to establish many frames of reference based upon several landmark structures in a fingerprint to obtain multiple representations.

At the expense of additional processing and storage cost, the multiple representations offer robust matching performance when extraction algorithm fails to detect one or more frames of reference. In the proposed feature extraction scheme, translation is handled by a single reference point location during the feature extraction stage. The present implementation of feature extraction assumes that the fingerprints are vertically oriented. In reality, the fingerprints in our database are not exactly vertically oriented; the fingerprints may be oriented up to different orientation angles away from the assumed vertical orientation. This image rotation is partially handled by a cyclic rotation of the feature values in the Finger Code in the matching stage in future implementations, the image rotation will be correctly handled by automatically determining the fingerprint orientation from the image data. The current scheme of feature extraction identifies the region of interest in the given fingerprint image with respect to the point of reference.

The four main steps in our feature extraction algorithm are

1) Determine a reference point and region of interest for the fingerprint image;

2) Calculate the region of interest around the reference point;

3) Filter the region of interest in eight different directions using a bank of Gabor filters (eight directions are required to completely capture the local ridge characteristics in a fingerprint while only four directions are required to capture the global configuration [6]);

4) Compute the average absolute deviation from the mean
of gray values in individual sectors in filtered images
to define the feature vector or the Finger Code.

In the current implementation, we have used the AAD features which give slightly better performance than variance features . Although AAD features perform reasonably well, we believe that a significantly better performance can be achieved by using more discriminative features.

Let I(x,y) denote the gray level at pixel (x,y) in an M* N fingerprint image and let (xc,yc) denote the reference point .The region of interest is defined as the collection of a, the sectors Si , where the i th sector Si is computed in term of parameter (r,0) as follows

Si={ (x,y) / b(Ti+1) = r < b(Ti+2) , ? i = ? < ? i+1,1= x = N,1= y <M}

is the width of each band, is the number of sectors considered in each band, and , where is the number of concentric bands considered around the reference point for feature extraction. These parameters depends upon the image resolution and size. In our first experiment with MSU_DBI database (image size = pixels, scanned at 500 dpi), we considered five concentric bands for feature extraction. Each band is 20-pixels wide , and segmented

into sixteen sectors. A 20-pixel wide band captures an area spanning about one ridge and valley pair, on an average, in a 500 dpi fingerprint image. A band with a width of 20 pixels is necessary to capture a single minutia in a sector, allowing our low-level features to capture this local information. If the sector width is more than 20 pixels, then the local information may be modulated by more global information. The innermost band is not used for feature extraction because the flow field in a region around a very high curvature point has poor coherence. Thus, absolute deviations of oriented Gabor responses to this region would be expected to be unreliable matching features. Thus, we have a total of sectors and the region of interest is a circle of radius 120 pixels, centered at the reference point. Eighty features for each of the eight filtered images provide a total of 640 features per fingerprint image. Each feature can be quantized into 256 values and requires 1 byte of storage, so the entire feature vector requires only 640 bytes of storage. In our second experiment with NIST 9 database (image size = pixels, scanned at 500 dpi), we used seven concentric bands giving us an 896 byte Finger Code. It is difficult to rely on feature extraction based on explicit detection of structural features in fingerprints, especially in poor quality images. Features based on statistical properties of images are likely to degrade gracefully with the image quality deterioration. For this study, we use grayscale variance-based features.

The average absolute deviation of the gray levels from the mean value in an image sector is indicative of the overall ridge activity in that sector which we claim to be useful for fingerprint verification. As mentioned above our matcher based on this simple statistical feature performs well and we expect to achieve significantly better accuracies with more discriminative attributes.

3.2. REFERENCE POINT LOCATION:

Fingerprints have many conspicuous landmark structures and a combination of them could be used for establishing a reference point.We define the reference point of a fingerprint as the point of maximum curvature of the concave ridges in the fingerprint image. Many previous approaches to determination of a reference point critically relied on the local features like Poincaré index or some other similar properties of the orientation field. While these methods work well in good quality fingerprint images, they fail to correctly localize reference points in poor quality fingerprints with cracks and scars, dry skin, or poor ridge and valley contrast. Recently, Hong and Jain have attempted to judiciously combine the orientation field information with available ridge details in a fingerprint . However, this method does not reliably handle poor quality fingerprints when the orientation field is very noisy and can be misled by poor structural cues in the presence of finger cracks. In order that a reference point algorithm gracefully handles local noise in a poor quality fingerprint, the detection should necessarily consider a large neighborhood in the fingerprint. On the other hand, for an accurate localization of the reference point, the approach should be sensitive to the local variations in a small neighborhood. To meet these conflicting requirements of an accurate and reliable localization, we propose a new method of reference point determination based on multiple resolution analysis of the orientation fields.

Let us first define the orientation field, for a fingerprint image. The orientation field, is defined as a image, where represents the local ridge orientation at pixel. Local ridge orientation is usually specified for a block rather than at every pixel; an image is divided into a set of nonoverlapping

blocks and a single orientation is defined for each block .Note that there is an ambiguity of in fingerprint orientation, i.e., local ridges oriented at and ridges oriented at cannot be differentiated from each other. A number of methods have been developed to estimate the orientation field in a fingerprint .

The least mean square orientation estimation algorithm has the following steps.

1) Divide the input image, into nonoverlapping blocks of size .

2) Compute the gradients and at each pixel .

Depending on the computational requirement, the gradient operator may vary from the simple Sobel operator to the more complex Marr–Hildreth operator

3) Estimate the local orientation of each block centered at pixel using the following equations :

$$V_x(i,j) = \sum_{u=i-w/2}^{i+w/2} \sum_{v=j-w/2}^{j+w/2} 2\partial_x(u,v)\partial_y(u,v) \qquad (6)$$

$$V_y(i,j) = \sum_{u=i-w/2}^{i+w/2} \sum_{v=j-w/2}^{j+w/2} (\partial_x^2(u,v) - \partial_y^2(u,v)) \qquad (7)$$

$$O(i,j) = \frac{1}{2}\tan^{-1}\left(\frac{V_y(i,j)}{V_x(i,j)}\right) \qquad (8)$$

Mathematically, it represents the direction that is orthogonal to the dominant direction of the *Fourier spectrum* of the window.

A summary of our reference point location algorithm is presented below.

1) Estimate the orientation field as described above using a window size of .

2) Smooth the orientation field in a local neighborhood. Let the smoothed orientation field be represented as . In order to perform smoothing (low-pass filtering), the orientation image needs to be converted into a *continuous vector field* . Although this successfully detects the reference point in most of the cases, including double loops , the present implementation is not very precise and consistent for the arch type fingerprints.

3) Find the maximum value in and assign its coordinate to the core, i.e., the reference point.

4) For a fixed number of times, repeat steps 1–6 by using a window size of , where and restrict the search for the reference point in step 6 in a local neighborhood of the detected reference point. In our experiments, we used three iterations with, and pixels, respectively, and hence the precision of the detected reference point is 5 pixels.

3.3. FILTERING :

Fingerprints have local parallel ridges and valleys, and well-defined local frequency and orientation . Properly tuned Gabor filters, can remove noise, preserve the true ridge and valley structures, and provide information contained in a particular orientation in the image. A minutia point can be viewed as an anomaly in locally parallel ridges and it is this information that we are attempting to capture using the Gabor filters. Before filtering the fingerprint image, we normalize the region of interest in each sector separately to a constant mean and variance. Normalization is performed to remove the effects of sensor noise and gray level deformation due to finger pressure differences. Let denote the gray value at pixel, and , the estimated mean and variance of sector , respectively, and , the normalized gray-level

value at pixel . For all the pixels in sector , the normalized image is defined as

$$N_i(x, y) = \begin{cases} M_0 + \sqrt{\dfrac{V_0 \times (I(x, y) - M_i)^2)}{V_i}}, & \text{if } I(x, y) > M_i \\ M_0 - \sqrt{\dfrac{V_0 \times (I(x, y) - M_i)^2)}{V_i}}, & \text{otherwise} \end{cases} \quad (16)$$

An even symmetric Gabor filter has the following general form in the spatial domain:

$$G(x, y; f, \theta) = \exp\left\{\frac{-1}{2}\left[\frac{x'^2}{\delta_{x'}^2} + \frac{y'^2}{\delta_{y'}^2}\right]\right\} \cos(2\pi f x') \quad (17)$$

$$x' = x \sin\theta + y \cos\theta \quad (18)$$

$$y' = x \cos\theta - y \sin\theta \quad (19)$$

convolution process significantly while maintaining the information content as the convolution with small values of the filter mask does not contribute significantly to the overall convolution. We also make use of the symmetry of the filter to speed up the convolution. However, convolution with Gabor filters is still the major contributor to the overall feature extraction time. In our experiments, we set the filter frequency to the average ridge frequency ( ), where is the average inter-ridge distance. The average inter-ridge distance is approximately 10 pixels in a 500 dpi fingerprint image. If is too large, spurious ridges are created in the filtered image whereas if is too small, nearby ridges are merged into one. We have used eight different values for (0 , 22.5 , 45 , 67.5 , 90 , 112.5 , 135 , and 157.5) with respect to

the -axis. The normalized region of interest in a fingerprint image is convolved with each of these eight filters to produce a set of eight filtered images. A fingerprint convolved with a -oriented filter accentuates those ridges which are parallel to the -axis and smoothes the ridges in the other directions. Filters tuned to other directions work in a similar way. These eight directional-sensitive filters capture most of the global ridge directionality information as well as the local ridge characteristics present in a fingerprint. We illustrate this through reconstructing a fingerprint image by adding together all the eight filtered images. The reconstructed image is very similar to the original image and only been slightly blurred (degraded) due to lack of orthogonality among the filters. At least four directional filters are required to capture the entire global ridge information in a fingerprint, but eight directional filters are required to capture the local characteristics. So, while four directions are sufficient for classification , eight directions are needed for matching. Our empirical results support our claim, we could get better accuracy by using eight directions for matching as compared to only four directions. By capturing both the global and local information, the verification accuracy is improved although there is some redundancy among the eight filtered images. If and (standard deviations of the Gaussian envelope) values are too large, the filter is more robust to noise, but is more likely to smooth the image to the extent that the ridge and valley details in the fingerprint are lost. If and values are too small, the filter is not effective in removing the noise. The values for and were empirically determined and each is set to 4.0 (about half the average inter-ridge distance).

than variance features as used .The 640-dimensional feature vectors (Finger Codes) for fingerprint images of two different fingers from the MSU_DBI database are shown as gray level images with eight disks, each disk corresponding to one filtered image in. The gray level in a sector in a disk represents the feature value for that sector in the corresponding filtered image. Note that appear to be visually similar as are, but the corresponding disks for two different fingers look very different.

3.4. MATCHING :

Fingerprint matching is based on finding the Euclidean distance between the corresponding Finger Codes. The translation invariance in the Finger Code is established by the reference point. However, in our present implementation, features are not rotationally invariant. An approximate rotation invariance is achieved by cyclically rotating the features in the Finger Code itself. A single step cyclic rotation of the features in the Finger Code described by corresponds to a feature vector which would be obtained if the image were rotated by. A rotation by steps corresponds to a rotation of the image. A positive rotation implies clockwise rotation while a negative rotation implies counterclockwise rotation. The Finger Code obtained after steps of rotation is given by

$$V_{i\theta}^{R} = V_{i'\theta'} \tag{21}$$

$$i' = (i + k + R) \bmod k + (i \operatorname{div} k) \times k \tag{22}$$

$$\theta' = (\theta + 180^{\circ} + 22.5^{\circ} \times R) \bmod 180^{\circ} \tag{23}$$

where k (=16) is the number of sectors in a band, $i \in [0, 1, 2, \cdots 79]$, and $\theta \in [0^{\circ}, 22.5^{\circ}, 45^{\circ}, 67.5^{\circ}, 90^{\circ}, 112.5^{\circ}, 135^{\circ}, 157.5^{\circ}]$.

For each fingerprint in the database, we store five templates corresponding to the following five rotations of the corresponding Finger Code. The input Finger Code is matched with the five templates stored in the database to obtain five different matching scores. The minimum matching score corresponds to the best alignment of the input fingerprint with the database fingerprint. Since a single cyclic rotation of the features in the Finger Code corresponds to a rotation of in the original image, we can only generate those representations of the fingerprint which are in steps of. Due to the nature of the tessellation, our features are invariant to only small perturbations that are within. Therefore, we generate another feature vector for each fingerprint during the time of registration which corresponds to a Rotation of. The original image is rotated by an angle of and its Finger Code is generated. Five templates corresponding to the various rotations of this Finger Code are also stored in the database. Thus, the database contains ten templates for each fingerprint. These ten templates correspond to all the rotations on the fingerprint image in steps of. As a result, we have generated Finger Codes for every rotation of the fingerprint image. This takes care of the rotation while matching the input Finger Code with the stored templates. The final matching distance score is taken as the minimum of the ten scores, i.e., matching of the input Finger Code with each of the ten templates. This minimum score corresponds to the best alignment of the two fingerprints being matched. Since the template generation for storage in the database is an off-line process, the verification time still depends on the time taken to generate a single template.

3.5.FUNDAMENTALS OF IMAGE PROCESSING :

3.5.1.VECTORS :

In the simplest case, the number of unknowns equals the number of equations. For example, here are a two equations in two unknowns:

$$2x - y = 1$$

$$x + y = 5.$$

There are at least two ways in which we can think of solving these equations for x and y. The first is to consider each equation as describing a line, with the solution being at the intersection of the lines: in this case the point (2, 3), Figure 0.1. This solution is termed a "row" solution because the equations are considered in isolation of one another. This is in contrast to a "column" solution in which the equations are rewritten in vector form:

$$\begin{pmatrix} 2 \\ 1 \end{pmatrix} x + \begin{pmatrix} -1 \\ 1 \end{pmatrix} y = \begin{pmatrix} 1 \\ 5 \end{pmatrix}.$$

3.5.2. MATRICES :

In solving n linear equations in n unknowns there are three quantities to consider. For example consider again the following set of equations:

$$2u + v + w = 5$$

$$4u - 6v + 0w = -2$$

$$-2u + 7v + 2w = 9.$$

On the right is the column vector:

$$\begin{pmatrix} 5 \\ -2 \\ 9 \end{pmatrix}.$$

and on the left are the three unknowns that can also be written as a column vector:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix}.$$

Matrices, like vectors, can be added and scalar multiplied. Not surprising, since we may think of a vector as a skinny matrix: a matrix with only one column. Consider the following 3×3 matrix:

$$A = \begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix}.$$

The matrix cA, where c is a scalar value, is given by:

$$cA = \begin{pmatrix} ca_1 & ca_2 & ca_3 \\ ca_4 & ca_5 & ca_6 \\ ca_7 & ca_8 & ca_9 \end{pmatrix}.$$

And the sum of two matrices, A = B + C, is given by:

$$\begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix} = \begin{pmatrix} b_1 + c_1 & b_2 + c_2 & b_3 + c_3 \\ b_4 + c_4 & b_5 + c_5 & b_6 + c_6 \\ b_7 + c_7 & b_8 + c_8 & b_9 + c_9 \end{pmatrix}.$$

3.5.3. DIGITAL FILTER DESIGN :

It is often the case that such filters pass certain frequencies and attenuate others (e.g., a lowpass, bandpass, or highpass filters).

The design of such filters consists of four basic steps:

1. choose the desired frequency response
2. choose the length of the filter
3. define an error function to be minimized
4. choose a minimization technique and solve

The choice of frequency response depends, of course, on the designers particular application, and its selection is left to their discretion. We will however provide some general guidelines for choosing a frequency response that is amenable to a successful design. In choosing a filter size there are two conflicting goals, a large filter allows for a more accurate match to the desired frequency response,however a small filter is desirable in order to minimize computational demands .

4.PERFORMANCE EVALUATION:

System testing is the procedure for enduring that the developed system is working at a good sense regarding the actual objectives and their needs. The main tragedy behind the testing process is performed in order to find errors. The main of the testing is to vanish all the errors in the developed system.

4.1 TESTING STRATEGIES:

There are general strategies for testing software:

◆ UNIT TESTING:

The unit testing involves testing of every module concerning the project .This testing is processed to examine the enhancing functionality of the project.

This testing had been processed and followed at the 2 modules in this stipulated project. The modules named as customer module and admin module. Customer module is tested by attaining authentication activity and shopping process is done completely. The admin module so as enters by providing his authentication process and finally views or does any modification process.

◆ FUNCTION TESTING :

The function testing generally focuses on any requirements for test that can be traced directly to use business functions and business rules. The main goal of this test is to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules.

The function testing plays a vital role in this project at the area of online bookstore which serves as an simple application. The functional testing enhances the process of shopping cart and its function of adding books, discarding, updating and other process are well evaluated.

◆ CODE TESTING :

The code testing examines the logic of the program. To endure this test every path of the program is tested.

The main logic used in this project is creation of web services and that web services is mainly communicated with WSDL language. The website created can communicate with any number of websites. All the combined logics are well tested using the code testing technique.

◆ CONFIGURATION TESTING :

Configuration testing verifies the operation of the testing on different software and hardware configurations. The configuration testing is used at the area of configuring a web service.

4.2 SYSTEM IMPLEMENTATION:

Implementation is the process of converting a new or a revised system design into an operational one. This is the final and important phase in the system life cycle. This phase is considered to be the most crucial stage in the development of the successful system.

The primary objective implementation is to write source code and internal documentation, so that the conformance of the code to its specification can easily be verified. The implementation phase involves the following major tasks:

- Careful planning.
- Investigation of the system and constraints.
- Design of the methods to achieve the change over.
- Evaluation of change over method.

5. CONCLUSION

We have developed a novel filter-based representation technique. The technique exploits both the local and global characteristics in a fingerprint image to verify an identity. Each fingerprint image is filtered in a number of directions and a fixed-length feature vector is extracted in the central region of the fingerprint. The feature vector (Finger- Code) is compact and requires only 640 (or 896, depending on image size) bytes. The matching stage computes the Euclidean distance between the templates. Finger Code and the input Finger- Code. On MSU_DBI database of 2672 fingerprints from 167 different subjects, four impressions per finger, we are able to achieve a verification accuracy which is only marginally inferior to the performance of a state-of-the-art minutiae-based fingerprint matcher. Our system, however, performs better than the minutiae-based system when the system performance requirements are less demanding on FAR (for example, in accessing ATM machines). Similar performance is obtained on the more challenging NIST 9 database. The minutiae-based technique is a very mature technology while the proposed filter-based approach is being developed and refined. We expect to improve the performance significantly by developing algorithms to overcome the following main shortcomings of our technique: 1) the reference point cannot be located accurately in noisy images and 2) the matching scheme is not able to tolerate large deformation in the ridge pattern due to finger pressure differences .About 99% of the total compute time for verification (3 s on a SUN ULTRA 10) is taken by the convolution of the input image with eight Gabor filters.

6. FUTURE ENHANCEMENT

A fingerprint is unique and permanent to a given individual, so it can serve as his or her identification.Fingerprint identification is inherently voluntary. Yourfinger cannot be read unless you place it on thefingerprint reader. In fingerprint applications, fakefingerprint attack is a serious concern. Building vitalitydetection mechanisms in the fingerprint recognitionsystem hardware and software is one of the ways to solve . Fingerprint devices can incorporatevitality detection by measuring optical, electric, orthermal properties of human skin or other biomedical characteristics, for instance, pulse. Finally, nowadays fingerprint systems are non-intrusive and non-threatening.

The quality of the initial authentication process determines the level of trust that can be placed in the certificates. The solution we proposed is based on fingerprint identification and smart card, the main idea is that fingerprint verification technology is combined with smart card to protect the private key, the card matches the cardholder's fingerprint and the template in it and furthermore, integrate user's private key with his fingerprint which relates user's key with his information. The fingerprint verification method used in smart card is based on minutia extraction and matching.Next we will give the workflow in the system. Our proposal is to integrate the user's fingerprint with his/her private key. RSA is used as the signature algorithm here.Parameters of the system: Let p and q are large prime number, n=p*q. The legal user is Alice.

Our protocol has several features and improvements over the existing smart-card based remote authentication

APPENDIX

SOURCE CODE :

LOGIN FORM :

```
public class LoginForm extends javax.swing.JFrame {
    String uname;
    String pwd;
    public LoginForm() {
        initComponents();
    }
    private void initComponents() {//GEN-BEGIN:initComponents
        java.awt.GridBagConstraints gridBagConstraints;

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jPasswordField1 = new javax.swing.JPasswordField();
        jButton1 = new javax.swing.JButton();
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
```

```
});

jPanel1.setLayout(new java.awt.GridBagLayout());
jPanel1.setBackground(new java.awt.Color(0, 153, 153));
jLabel1.setText("LOGIN FORM");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
gridBagConstraints.ipadx = 61;
jPanel1.add(jLabel1, gridBagConstraints);
jLabel2.setText("UserName");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
gridBagConstraints.ipadx = 67;
gridBagConstraints.insets = new java.awt.Insets(12, 10, 8, 8);
jPanel1.add(jLabel2, gridBagConstraints);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
gridBagConstraints.ipadx = 99;
gridBagConstraints.ipady = 8;
gridBagConstraints.insets = new java.awt.Insets(12, 10, 8, 8);
jPanel1.add(jTextField1, gridBagConstraints);
jLabel3.setText("Password");
gridBagConstraints = new java.awt.GridBagConstraints();
```

```
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.fill = java.awt.GridBagConstraints.VERTICAL;
gridBagConstraints.ipadx = 70;
gridBagConstraints.insets = new java.awt.Insets(12, 10, 8, 8);
jPanel1.add(jLabel3, gridBagConstraints);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 4;
gridBagConstraints.ipadx = 97;
gridBagConstraints.ipady = 5;
gridBagConstraints.insets = new java.awt.Insets(12, 10, 8, 8);
jPanel1.add(jPasswordField1, gridBagConstraints);
jButton1.setText("Submit");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 6;
gridBagConstraints.insets = new java.awt.Insets(12, 10, 8, 8);
jPanel1.add(jButton1, gridBagConstraints);

getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);
```

```
            getRootPane().setDefaultButton(jButton1);
        pack();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
        uname=jTextField1.getText();
        pwd=jPasswordField1.getText();
            try
        {
        if(uname.equals("admin") && pwd.equals("admin"))
        {
        SelectOption s=new SelectOption();
        s.show();
        this.hide();
        }
        }catch(NullPointerException npe)
        {

        }
    }
    private void exitForm(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_exitForm
        System.exit(0);
    }
        public static void main(String args[]) {
        new LoginForm().show();
```

```
}
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JPanel jPanel1;
private javax.swing.JPasswordField jPasswordField1;
private javax.swing.JTextField jTextField1;
 }
```

SELECT OPTION :

```
public class SelectOption extends javax.swing.JFrame {
   public SelectOption() {
      initComponents();
   }
   private void initComponents() {
      java.awt.GridBagConstraints gridBagConstraints;
      buttonGroup1 = new javax.swing.ButtonGroup();
      jPanel1 = new javax.swing.JPanel();
      jRadioButton1 = new javax.swing.JRadioButton();
      jRadioButton2 = new javax.swing.JRadioButton();
      jToggleButton1 = new javax.swing.JToggleButton();
      jToggleButton2 = new javax.swing.JToggleButton();
      jLabel1 = new javax.swing.JLabel();

       addWindowListener(new java.awt.event.WindowAdapter() {
```

```
        public void windowClosing(java.awt.event.WindowEvent evt) {
            exitForm(evt);
        }
    });

    jPanel1.setLayout(new java.awt.GridBagLayout());
    jPanel1.setBackground(new java.awt.Color(0, 153, 153));
    jRadioButton1.setBackground(new java.awt.Color(0, 153, 153));
    jRadioButton1.setText("Enroll New Student");
    buttonGroup1.add(jRadioButton1);
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 2;
    gridBagConstraints.ipadx = 12;
    gridBagConstraints.insets = new java.awt.Insets(22, 16, 21, 19);
    jPanel1.add(jRadioButton1, gridBagConstraints);
jRadioButton1.getAccessibleContext().setAccessibleParent(jRadioButton1);
    jRadioButton2.setBackground(new java.awt.Color(0, 153, 153));
    jRadioButton2.setText("Entery Attendance");
    buttonGroup1.add(jRadioButton2);
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 2;
    gridBagConstraints.insets = new java.awt.Insets(22, 16, 21, 19);
    jPanel1.add(jRadioButton2, gridBagConstraints);
jRadioButton2.getAccessibleContext().setAccessibleParent(jRadioButton2);
```

```
        jToggleButton1.setText("Back");
        jToggleButton1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButton1ActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.insets = new java.awt.Insets(22, 16, 21, 19);
        jPanel1.add(jToggleButton1, gridBagConstraints);

        jToggleButton2.setText("Next");
        jToggleButton2.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButton2ActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.insets = new java.awt.Insets(22, 16, 21, 19);
```

```
    jPanel1.add(jToggleButton2, gridBagConstraints);

    jLabel1.setText("SELECT OPTION ");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.ipadx = 72;
    gridBagConstraints.insets = new java.awt.Insets(7, 16, 21, 19);
    jPanel1.add(jLabel1, gridBagConstraints);

    getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);

    pack();
  }
  private void
jToggleButton2ActionPerformed(java.awt.event.ActionEventevt)
 {
          if(jRadioButton1.isSelected())
        {
        StudentDetails sd=new StudentDetails();
        sd.show();
        this.hide();
        }
        else
         {
        Attendance a=new Attendance();
        a.show();
        this.hide();
        }
```

```
    }
    private void
jToggleButton1ActionPerformed(java.awt.event.ActionEventevt)
 {
        this.hide();
        LoginForm lf=new LoginForm();
        lf.show();
    }
    private void exitForm(java.awt.event.WindowEvent evt)
 {
      System.exit(0);
    }
    public static void main(String args[]) {
      new SelectOption().show();
    }

    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JToggleButton jToggleButton1;
    private javax.swing.JToggleButton jToggleButton2;
 }
```

SERIAL INPUT :

```
import java.io.*;
import java.util.*;
import javax.comm.*;
import javax.swing.*;
import java.sql.*;
import java.util.*;
import java.text.SimpleDateFormat;

public class SerialInput implements Runnable,SerialPortEventListener {
        int flag=0;
        static Enumeration portList;
    static CommPortIdentifier portId;
    InputStream                 inputStream;
    static SerialPort serialPort;
    static OutputStream outputStream;
    String output;
    String out;
    byte[] readBuffer ;

    Thread readThread;
    StringBuffer sb=new StringBuffer();
        String facePath,Table;
        public SerialInput() {

                portList = CommPortIdentifier.getPortIdentifiers();
```

```
    while (portList.hasMoreElements()) {
            portId = (CommPortIdentifier) portList.nextElement();
        if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
            if (portId.getName().equals("COM1"))
{
                                try {
                            serialPort = (SerialPort)
portId.open("SimpleReadApp", 2000);
                        } catch (PortInUseException e) {}
                        try {
                                inputStream = serialPort.getInputStream();
                                outputStream = serialPort.getOutputStream();
                        } catch (IOException e) {}
                                    try {
                                serialPort.addEventListener(this);
                                    } catch (TooManyListenersException e) {}
                        serialPort.notifyOnDataAvailable(true);
                        try {
                                serialPort.setSerialPortParams(9600,
                                SerialPort.DATABITS_8,
                                SerialPort.STOPBITS_1,
                                SerialPort.PARITY_NONE);
                        } catch (UnsupportedCommOperationException e) {}
                        readThread = new Thread(this);
                        readThread.start(); }
                                    }
                        }     }
```

```
public void serialEvent(SerialPortEvent event) {
switch(event.getEventType()) {
      case SerialPortEvent.BI:
      case SerialPortEvent.OE:
      case SerialPortEvent.FE:
      case SerialPortEvent.PE:
      case SerialPortEvent.CD:
      case SerialPortEvent.CTS:
      case SerialPortEvent.DSR:
      case SerialPortEvent.RI:
      case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
        break;
  case SerialPortEvent.DATA_AVAILABLE:
              try {
              try {
                    readThread.sleep(1000);
              }
              catch(Exception e) {}
            while (inputStream.available() > 0) {
                int size=inputStream.available();
                readBuffer = new byte[size];
          int numBytes = inputStream.read(readBuffer);
                output=new String(readBuffer);
            System.out.println(output);
                //sb.append(output);
                int len=output.length();
                System.out.println("length is"+len);
```

```
                char[] c=new char[len];
                c=output.toCharArray();
                int counter=0;
                for(int i=25;i<29;i++){
                        sb.append(c[i]);
                System.out.println("counter="+counter+"value="+c[i]);
                        counter++;
                }

                System.out.println("name is="+sb.toString().trim());
                try {
 StudentDetails.jTextField2.setText(sb.toString().trim());
                        }catch(Exception e){System.out.println(e);}

                                }
        outputStream.flush();
          }
          catch (IOException e) {
                System.out.println(e);
          }
                }
          }
public void run() {
          try {
                try {
                                System.out.println("entered");
```

```
                    DataOutputStream o=new
DataOutputStream(outputStream);
                    System.out.println("entered");

                     int temp=2;
                           o.writeInt(0x7e000000);
                     o.writeInt(0x12000000);
                     o.writeInt(0x00);
                     o.writeInt(0x00);
                     o.writeInt(0x00);
                   o.writeInt(0x00);
                   o.writeInt(0x12000000);

                   if(temp==3)
                   {
                      JOptionPane.showMessageDialog(null, "TimeOut",
  "TimeOut", JOptionPane.ERROR_MESSAGE);
                    }


                 } catch (Exception e) {}

                 } catch (Exception e) {}
     }
```

ATTENDANCE :

```
import java.awt.geom.AffineTransform;
import java.awt.image.ImageObserver;
import java.sql.*;
import java.awt.Graphics2D;
import java.applet.Applet;
import java.awt.*;
import java.awt.Image.*;
import javax.imageio.ImageIO;
import java.io.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.awt.image.PixelGrabber;
import java.awt.image.*;
public class Attendance extends javax.swing.JFrame {



    String regno,facepath;
  JFileChooser chooser;
  public Attendance() {
    initComponents();
  }

  private void initComponents() {//GEN-BEGIN:initComponents
```

```
java.awt.GridBagConstraints gridBagConstraints;

jPanel1 = new javax.swing.JPanel();
jLabel1 = new javax.swing.JLabel();
jPanel2 = new javax.swing.JPanel();
jLabel2 = new javax.swing.JLabel();
jTextField1 = new javax.swing.JTextField();
jLabel3 = new javax.swing.JLabel();
jTextField2 = new javax.swing.JTextField();
jToggleButton1 = new javax.swing.JToggleButton();
jLabel4 = new javax.swing.JLabel();
jTextField3 = new javax.swing.JTextField();
jToggleButton2 = new javax.swing.JToggleButton();
jToggleButton3 = new javax.swing.JToggleButton();
jToggleButton4 = new javax.swing.JToggleButton();
jLabel5 = new javax.swing.JLabel();
jComboBox1 = new javax.swing.JComboBox();

addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
    }
});

jPanel1.setBackground(new java.awt.Color(0, 153, 153));
jPanel1.setPreferredSize(new java.awt.Dimension(50, 30));
jLabel1.setText("ATTENDANCE ENTRY");
```

```
        jPanel1.add(jLabel1);
        getContentPane().add(jPanel1, java.awt.BorderLayout.NORTH);
        jPanel2.setLayout(new java.awt.GridBagLayout());
        jPanel2.setBackground(new java.awt.Color(0, 153, 153));
        jLabel2.setText("RegNo");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.ipadx = 67;
        gridBagConstraints.ipady = 5;
        gridBagConstraints.anchor =
    java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(13, 11, 9, 11);
        jPanel2.add(jLabel2, gridBagConstraints);
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.ipadx = 58;
        gridBagConstraints.ipady = 9;
        gridBagConstraints.insets = new java.awt.Insets(13, 11, 9, 11);
        jPanel2.add(jTextField1, gridBagConstraints);
        jLabel3.setText("Select Face");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.ipadx = 67;
        gridBagConstraints.ipady = 5;
        gridBagConstraints.anchor =
     java.awt.GridBagConstraints.NORTHWEST;

        gridBagConstraints.insets = new java.awt.Insets(13, 11, 9, 11);
```

```
    jPanel2.add(jLabel3, gridBagConstraints);
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 1;
    gridBagConstraints.ipadx = 58;
    gridBagConstraints.ipady = 9;
    gridBagConstraints.insets = new java.awt.Insets(13, 11, 9, 11);
    jPanel2.add(jTextField2, gridBagConstraints);
    jToggleButton1.setText("Browse Face");
    jToggleButton1.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jToggleButton1ActionPerformed(evt);
        }
    });

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 2;
    gridBagConstraints.gridy = 1;
    gridBagConstraints.ipady = 4;
    gridBagConstraints.insets = new java.awt.Insets(13, 11, 9, 11);
    jPanel2.add(jToggleButton1, gridBagConstraints);
    jLabel4.setText("Select Finger");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 3;
    gridBagConstraints.ipadx = 67;
```

```
        gridBagConstraints.ipady = 5;
        gridBagConstraints.insets = new java.awt.Insets(13, 11, 9, 11);
        gridBagConstraints.anchor =
    java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.ipadx = 58;
        gridBagConstraints.ipady = 9;
        gridBagConstraints.insets = new java.awt.Insets(13, 11, 9, 11);
        jToggleButton2.setText("Browse Finger");
        jToggleButton2.addActionListener(new
      java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButton2ActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.ipady = 4;
        gridBagConstraints.insets = new java.awt.Insets(13, 11, 9, 11);
        jToggleButton3.setText("Register Attendance");
        jToggleButton3.addActionListener(new

            java.awt.event.ActionListener() {
```

```
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jToggleButton3ActionPerformed(evt);
        }
    });

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 2;
    gridBagConstraints.gridy = 4;
    gridBagConstraints.insets = new java.awt.Insets(13, 11, 9, 11);
    jPanel2.add(jToggleButton3, gridBagConstraints);
    jToggleButton4.setText("Back");
    jToggleButton4.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jToggleButton4ActionPerformed(evt);
        }
    });

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 4;
    gridBagConstraints.insets = new java.awt.Insets(13, 11, 9, 11);
    jPanel2.add(jToggleButton4, gridBagConstraints);
    jLabel5.setText("Time Entry");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 2;
```

```
    gridBagConstraints.ipadx = 71;
    gridBagConstraints.insets = new java.awt.Insets(2, 7, 5, 4);
    jPanel2.add(jLabel5, gridBagConstraints);
    jComboBox1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jComboBox1ActionPerformed(evt);
        }
    });

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 2;
    gridBagConstraints.ipadx = 71;
    gridBagConstraints.insets = new java.awt.Insets(2, 7, 5, 4);
    jPanel2.add(jComboBox1, gridBagConstraints);
            jComboBox1.addItem("TIME IN");
            jComboBox1.addItem("TIME OUT");
    getContentPane().add(jPanel2, java.awt.BorderLayout.CENTER);
    pack();
  }
  private void
jComboBox1ActionPerformed(java.awt.event.ActionEventevt) {
  }
```

```
  private void jToggleButton4ActionPerformed(java.awt.event.ActionEvent
evt) {

    this.hide();
    SelectOption s=new SelectOption();
    s.show();
  }

  private void jToggleButton3ActionPerformed(java.awt.event.ActionEvent
evt) {
 String time = ""+jComboBox1.getSelectedItem();
                regno=jTextField1.getText();
          facepath=jTextField2.getText();
          System.out.println(regno);
          System.out.println(facepath);
    CompareImage cmp =  new CompareImage(facepath,regno);
    if(cmp.toCompare()) {
     SerialCheck s=new SerialCheck(facepath,time);
            JOptionPane.showMessageDialog(this,"Image Matched");
          }
    else
            JOptionPane.showMessageDialog(this,"Image Not Matched");
                  }
```

```
private void jToggleButton2ActionPerformed(java.awt.event.ActionEvent
evt) {
    int returnVal = jf1.showOpenDialog(this);
        if(returnVal == JFileChooser.APPROVE_OPTION)
    {
    f1=jf1.getSelectedFile();
                files=f1.getPath();
                jTextField3.setText(files);
                System.out.println(files);
          }
          try{
     f=new File(files);
    long len=f.length();
    System.out.println(len);
     image11= ImageIO.read(f);
          int ty=image11.getType();
          System.out.println(ty);
         }catch(Exception ee){}
    pixels1=image11.getRGB(0,0,256,256, pixel1,0,256);
          int valuetoc=0;
          for(int in=0;in<256;in++)
          {
          for(int jn=0;jn<256;jn++)
          {
                pixels[jn][in]=pixel1[valuetoc];
                image[jn][in]=pixel1[valuetoc];
                valuetoc++;
```

```
            }
            }
            }
  private void jToggleButton1ActionPerformed(java.awt.event.ActionEvent
evt) {
      chooser = new JFileChooser();
      int returnVal = chooser.showOpenDialog(this);
           if(evt.getSource()==jToggleButton1)
      {
                String path=chooser.getSelectedFile().getAbsolutePath();
        String str[]=path.split("\\.");
       if(str[1].equals("jpg"))
        {
            jTextField2.setText(path);
        }
       else
       {
         JOptionPane.showMessageDialog(this,"Please Select finger jpg
  File","Error",JOptionPane.ERROR_MESSAGE);
              }
            }
                        }
  private void exitForm(java.awt.event.WindowEvent evt) {
        System.exit(0);
      }
```

```
public static void main(String args[]) {
    new Attendance().show();
}



private javax.swing.JComboBox jComboBox1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
private javax.swing.JToggleButton jToggleButton1;
private javax.swing.JToggleButton jToggleButton2;
private javax.swing.JToggleButton jToggleButton3;
private javax.swing.JToggleButton jToggleButton4;

}
```

SERIAL CHECK :

```
import java.io.*;
import java.util.*;
import javax.comm.*;
import javax.swing.*;
import java.sql.*;
import java.util.*;
import java.text.SimpleDateFormat;

public class SerialCheck implements Runnable,SerialPortEventListener {
        int flag=0;
        static Enumeration portList;
    static CommPortIdentifier portId;
    InputStream                 inputStream;
    static SerialPort serialPort;
    static OutputStream outputStream;
    String output;
    String out;
    byte[] readBuffer ;

    Thread readThread;
    StringBuffer sb=new StringBuffer();
        String facePath,Table;
        public SerialCheck(String path,String time) {
                Table = time;
                facePath = path;
```

```
        portList = CommPortIdentifier.getPortIdentifiers();
    while (portList.hasMoreElements()) {
        portId = (CommPortIdentifier) portList.nextElement();
      if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
        if (portId.getName().equals("COM1")) {
                        try {
                    serialPort = (SerialPort)
portId.open("SimpleReadApp", 2000);
                } catch (PortInUseException e) {}
                try {
                    inputStream = serialPort.getInputStream();
                    outputStream = serialPort.getOutputStream();
                } catch (IOException e) {}
                        try {
                    serialPort.addEventListener(this);
                        } catch (TooManyListenersException e) {}
                serialPort.notifyOnDataAvailable(true);
                try {
                    serialPort.setSerialPortParams(9600,
                    SerialPort.DATABITS_8,
                    SerialPort.STOPBITS_1,
                    SerialPort.PARITY_NONE);
                } catch (UnsupportedCommOperationException e) {}
                readThread = new Thread(this);
                readThread.start();

                }
```

```
            }
        }
}
public void serialEvent(SerialPortEvent event) {
switch(event.getEventType()) {
      case SerialPortEvent.BI:
      case SerialPortEvent.OE:
      case SerialPortEvent.FE:
      case SerialPortEvent.PE:
      case SerialPortEvent.CD:
      case SerialPortEvent.CTS:
      case SerialPortEvent.DSR:
      case SerialPortEvent.RI:
      case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
        break;
 case SerialPortEvent.DATA_AVAILABLE:
              try {
              try {
                    readThread.sleep(1000);
              }
              catch(Exception e) {}
          while (inputStream.available() > 0) {
              int size=inputStream.available();
              readBuffer = new byte[size];
        int numBytes = inputStream.read(readBuffer);
              output=new String(readBuffer);
          System.out.println(output);
```

```
int len=output.length();
System.out.println("length is"+len);
char[] c=new char[len];
c=output.toCharArray();
int counter=0;
for(int i=25;i<29;i++){
        sb.append(c[i]);
System.out.println("counter="+counter+"value="+c[i]);
        counter++;
}
System.out.println("Id is="+sb.toString().trim());
try {
                String query = "";
SimpleDateFormat sd = new SimpleDateFormat("hh:mm:ss");
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy");
String str1 = sd.format(new java.util.Date());
String str2 = sdf.format(new java.util.Date());
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection conn=DriverManager.getConnection("jdbc:odbc:bio","","");
    Statement st=conn.createStatement();
    ResultSet rs = st.executeQuery("select * from student where regno =
'"+sb.toString().trim()+"'");
      if(rs.next()) {
      if(Table.equalsIgnoreCase("TIME IN"))

       query="insertintoIntimevalues('"+sb.toString().trim()+"','"+str1+"','"+s
tr2+"')";
```

```
                                    else
    query = "insert into Outtime
values('"+sb.toString().trim()+"','"+str1+"','"+str2+"')";
        st.executeUpdate(query);

        JOptionPane.showMessageDialog(null,"Sucessfully Stored");
                                        new
 StudentInformation(sb.toString());
                                    }
                                    else
         System.out.println("Please Register ur Data");
                                }catch(Exception e){System.out.println(e);}


                                    }
                                outputStream.flush();

                }
                catch (IOException e) {
                        System.out.println(e);
                }
                        }
                }
        public void run() {
                try {
                        try {
                                        System.out.println("entered");
                DataOutputStream o=new DataOutputStream(outputStream);
```

```
                System.out.println("entered");

                int temp=2;
                        o.writeInt(0x7e000000);
                o.writeInt(0x12000000);
                o.writeInt(0x00);
                o.writeInt(0x00);
                o.writeInt(0x00);
            o.writeInt(0x00);
            o.writeInt(0x12000000);
             temp=3;
            if(temp==3)
            {
                JOptionPane.showMessageDialog(null, "TimeOut",
"TimeOut", JOptionPane.ERROR_MESSAGE);
            }

        } catch (Exception e) {}

        } catch (Exception e) {}
  }
  }
```

COMPARE IMAGE :

```
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JInternalFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.border.EtchedBorder;
public class CompareImage {
Connection con;
        Statement st;
        ResultSet rs;
        String files,files1;
```

```
String s[],s1;
File f1,f2;
BufferedImage bi,bii;
ImageIcon im,im1;
BufferedImage img,img1;
static int r[] = new int[256],r1[] = new int[256];
int h=0,w=0;
int pixel[];
int pixel1[];
int pixels[][];
int pixels1[][];
int red1=0;
int blue1=0;
int green1=0;
String pathImage,rno;
public CompareImage(String pathString,String Rno) {
        pathImage = pathString;
        rno = Rno;
        System.out.println(pathImage);
        setConnection();
        toInitialize();
        toCompare();
                }
  private void toInitialize() {
                try {
                f1 = new File(pathImage);
bi = ImageIO.read(f1);
```

```
                ImageIcon im = new ImageIcon(bi);
                h = bi.getHeight();
                w = bi.getWidth();
                int[] rgbs = new int[w*h];
                int x = 0,k = 0;
                bi.getRGB(0, 0,w, h, rgbs, 0, w );
               for (int i=0; i<10; i++)
            for (int j=0;j<10;j++) {
                r[k]=rgbs[x];
                 x++;k++;
            }
            }catch (Exception e) {}
          try {
       st = con.createStatement();
       rs =st.executeQuery("select facepath from student where regno
="'"+rno+"'");
                              if(rs.next()) {
                              f2 = new File(rs.getString("facepath"));
                              bii = ImageIO.read(f2);
                              ImageIcon im1 = new ImageIcon(bii);

                               h = bii.getHeight();
                               w = bii.getWidth();
                            int[] rgb = new int[w*h];
                    int x1 = 0,k = 0;
                bii.getRGB(0, 0,w, h, rgb, 0, w );
                for (int i=0; i<10; i++)
```

```
for (int j=0;j<10;j++) {
        r1[k]=rgb[x1];
        x1++;k++;
}
        }

}catch (Exception e) {}
}
public static boolean toCompare() {
int i;
for(i = 0;i < r.length;i++) {
if(r[i]==r1[i]);
      else
              break;
}
if(i == r.length) {
              System.out.println("Images are same");
              return true;
}
else {
System.out.println("Images are not same");
        return false;
}
}
private void setConnection() {
      try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

```
                con = DriverManager.getConnection("jdbc:odbc:bio");
                st = con.createStatement();
                rs = st.executeQuery("select * from student");
            }catch (Exception e) {
                System.out.println("Error in connecting"+e);
            }
        }
}
```

STUDENT DETAILS :

```
import javax.swing.*;
import java.sql.*;
public class StudentDetails extends javax.swing.JFrame {
    JFileChooser chooser;
  public StudentDetails() {
    initComponents();
  }
    private void initComponents() {
    java.awt.GridBagConstraints gridBagConstraints;
    jPanel1 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    jPanel2 = new javax.swing.JPanel();
    jLabel2 = new javax.swing.JLabel();
    jTextField1 = new javax.swing.JTextField();
```

```
jLabel3 = new javax.swing.JLabel();
jTextField2 = new javax.swing.JTextField();
jLabel4 = new javax.swing.JLabel();
jTextField3 = new javax.swing.JTextField();
jLabel5 = new javax.swing.JLabel();
jTextField4 = new javax.swing.JTextField();
jLabel6 = new javax.swing.JLabel();
jTextField5 = new javax.swing.JTextField();
jLabel7 = new javax.swing.JLabel();
jTextField6 = new javax.swing.JTextField();
jToggleButton1 = new javax.swing.JToggleButton();
jToggleButton2 = new javax.swing.JToggleButton();
jToggleButton3 = new javax.swing.JToggleButton();
jToggleButton4 = new javax.swing.JToggleButton();
addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
    }
});
jPanel1.setBackground(new java.awt.Color(0, 153, 153));
jPanel1.setPreferredSize(new java.awt.Dimension(50, 30));
jLabel1.setText("STUDENT DETAILS");
jPanel1.add(jLabel1);
getContentPane().add(jPanel1, java.awt.BorderLayout.NORTH);
jPanel2.setLayout(new java.awt.GridBagLayout());
jPanel2.setBackground(new java.awt.Color(0, 153, 153));
jLabel2.setText("Name");
```

```
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.ipadx = 84;
        gridBagConstraints.ipady = 7;
        gridBagConstraints.anchor =
    java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
        jPanel2.add(jLabel2, gridBagConstraints);
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.ipadx = 114;
        gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
        jPanel2.add(jTextField1, gridBagConstraints);
        jLabel3.setText("RegNo");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.ipadx = 80;
        gridBagConstraints.ipady = 7;
        gridBagConstraints.anchor =
      java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
        jPanel2.add(jLabel3, gridBagConstraints);
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
```

```
        gridBagConstraints.gridy = 2;
        gridBagConstraints.ipadx = 115;
        gridBagConstraints.ipady = 4;
        gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
        jPanel2.add(jTextField2, gridBagConstraints);
        jLabel4.setText("Department");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.ipadx = 57;
        gridBagConstraints.ipady = 7;
        gridBagConstraints.anchor =
java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
        jPanel2.add(jLabel4, gridBagConstraints);
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.ipadx = 115;
        gridBagConstraints.ipady = 4;
        gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
        jPanel2.add(jTextField3, gridBagConstraints);
        jLabel5.setText("Address");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.ipadx = 71;
```

```
        gridBagConstraints.ipady = 7;
        gridBagConstraints.anchor =
java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
        jPanel2.add(jLabel5, gridBagConstraints);
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.ipadx = 115;
        gridBagConstraints.ipady = 4;
        gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
        jPanel2.add(jTextField4, gridBagConstraints);
        jLabel6.setText("Select Face Image");
        jLabel6.setPreferredSize(new java.awt.Dimension(42, 15));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 5;
        gridBagConstraints.ipadx = 28;
        gridBagConstraints.ipady = 7;
        gridBagConstraints.anchor =
java.awt.GridBagConstraints.NORTHWEST;
        gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
        jPanel2.add(jLabel6, gridBagConstraints);
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 5;
        gridBagConstraints.ipadx = 115;
```

```
    gridBagConstraints.ipady = 4;
    gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
    jPanel2.add(jTextField5, gridBagConstraints);
    jLabel7.setText("Select Finger Image");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 6;
    gridBagConstraints.ipadx = 31;
    gridBagConstraints.ipady = 7;
    gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 6;
    gridBagConstraints.ipadx = 115;
    gridBagConstraints.ipady = 4;
    gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
    jToggleButton1.setText("SelectFace");
    jToggleButton1.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
          jToggleButton1ActionPerformed(evt);
        }
      });
      gridBagConstraints = new java.awt.GridBagConstraints();
      gridBagConstraints.gridx = 2;
      gridBagConstraints.gridy = 5;
      gridBagConstraints.ipadx = 3;
```

```
        gridBagConstraints.ipady = 5;
        gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
        jPanel2.add(jToggleButton1, gridBagConstraints);
        jToggleButton2.setText("Sense Finger");
        jToggleButton2.addActionListener(new
    java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButton2ActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.ipadx = 3;
        gridBagConstraints.ipady = 5;
        gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
        jPanel2.add(jToggleButton2, gridBagConstraints);
        jToggleButton3.setText("Back");
        jToggleButton3.addActionListener(new
      java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButton3ActionPerformed(evt);
            }
        });
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 7;
```

```
    gridBagConstraints.ipadx = 17;
    gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
    jPanel2.add(jToggleButton3, gridBagConstraints);
    jToggleButton4.setText("Enroll");
    jToggleButton4.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
          jToggleButton4ActionPerformed(evt);
        }
      });
      gridBagConstraints = new java.awt.GridBagConstraints();
      gridBagConstraints.gridx = 1;
      gridBagConstraints.gridy = 7;
      gridBagConstraints.ipadx = 7;
      gridBagConstraints.insets = new java.awt.Insets(10, 6, 8, 9);
      jPanel2.add(jToggleButton4, gridBagConstraints);
      getContentPane().add(jPanel2, java.awt.BorderLayout.CENTER);
      pack();
    }
    private void
  jToggleButton4ActionPerformed(java.awt.event.ActionEvent evt) {
  FIRST:event_jToggleButton4ActionPerformed
       String Name=jTextField1.getText();
       String RegNo=jTextField2.getText();
       String Dept=jTextField3.getText();
       String Address=jTextField4.getText();
       String FacePath=jTextField5.getText();
```

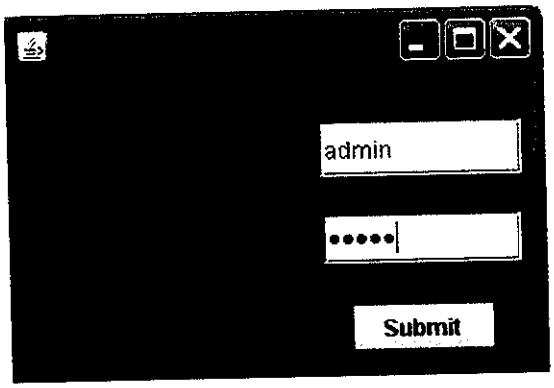
```
    String FingerPath=jTextField6.getText();
    try
    {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      Connection
conn=DriverManager.getConnection("jdbc:odbc:bio","","");
       Statement stmt=conn.createStatement();
       String Query="insert into student
values('"+RegNo+"','"+Name+"','"+Dept+"','"+Address+"','"+FacePath+"','"
+FingerPath+"')";
        int rs=stmt.executeUpdate(Query);
         System.out.println(rs);
      }catch(Exception e){System.out.println(e);}
              }
   private void jToggleButton2ActionPerformed(java.awt.event.ActionEvent
 evt) {
      SerialInput ster = new SerialInput();
    }
    private void jToggleButton1ActionPerformed(java.awt.event.ActionEvent
 evt) {
       chooser = new JFileChooser();
       int returnVal = chooser.showOpenDialog(this);
       if(evt.getSource()==jToggleButton1)
       {
          String path=chooser.getSelectedFile().getAbsolutePath();
          String str[]=path.split("\\.");
         if(str[1].equals("jpg"))
```

```
        {
          jTextField5.setText(path);
        }
        else
        {
       JOptionPane.showMessageDialog(this,"Please Select jpg
File","Error",JOptionPane.ERROR_MESSAGE);
                }
     }
   }
   private void jToggleButton3ActionPerformed(java.awt.event.ActionEvent
evt) {
            this.hide();
      SelectOption s=new SelectOption();
      s.show();
   }
   private void exitForm(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_exitForm
      System.exit(0);
   }
   public static void main(String args[]) {
      new StudentDetails().show();
   }
   private javax.swing.JLabel jLabel1;
   private javax.swing.JLabel jLabel2;
   private javax.swing.JLabel jLabel3;
   private javax.swing.JLabel jLabel4;
```

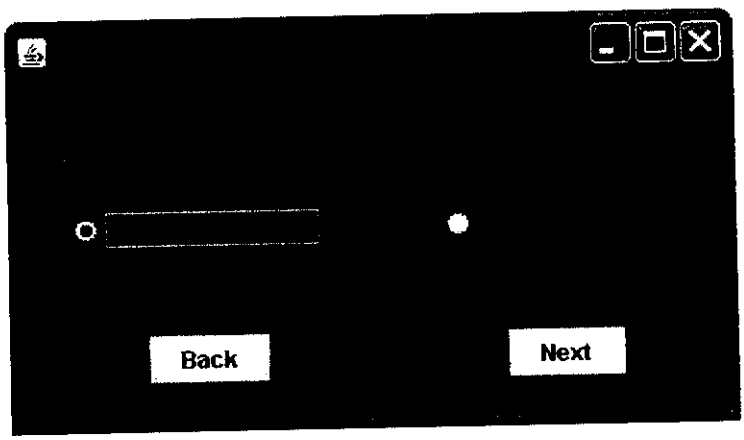
```
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JTextField jTextField1;
    static  javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField5;
    private javax.swing.JTextField jTextField6;
    private javax.swing.JToggleButton jToggleButton1;
    private javax.swing.JToggleButton jToggleButton2;
    private javax.swing.JToggleButton jToggleButton3;
    private javax.swing.JToggleButton jToggleButton4;

}
```

SCREENSHOT:

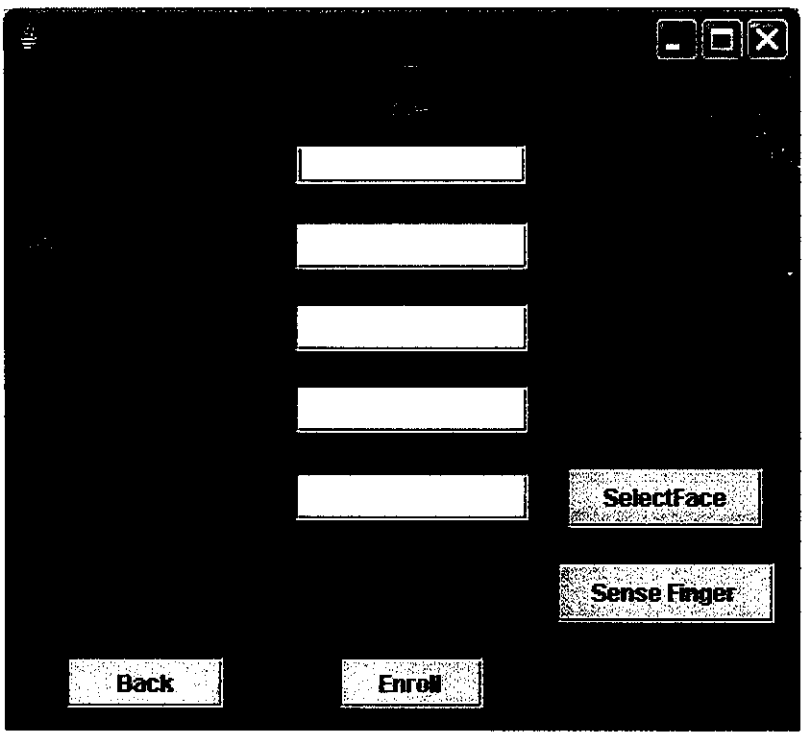
1. LOGIN FORM:



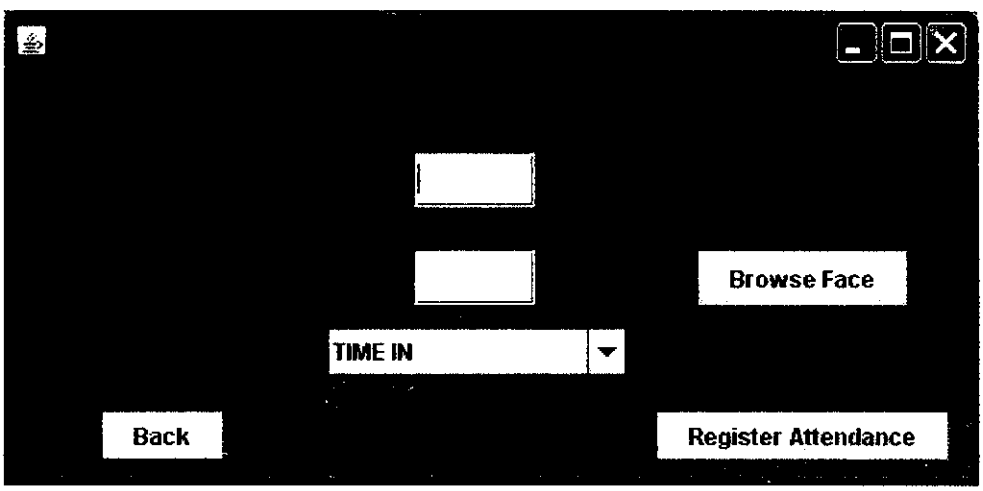
2.SELECT OPTION :



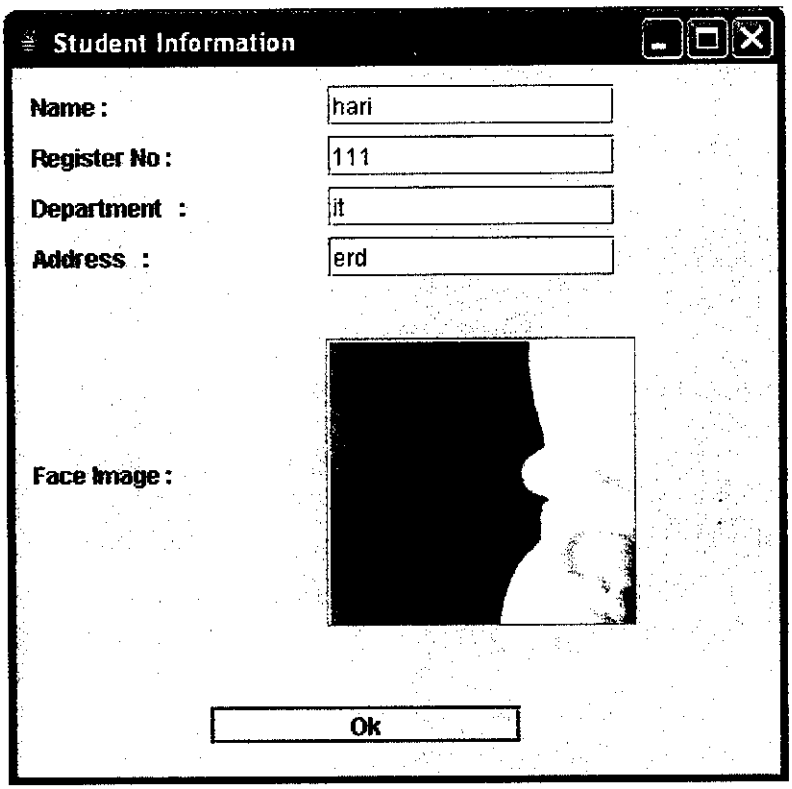
3. STUDENT DETAILS :



4.ATTENDANCE ENTRY :



5. STUDENT INFORMATION :



REFERENCES :

[1] A. K. Jain, L. Hong, S. Pankanti, and R. Bolle, "An identity Authentication system using fingerprints," *Proc. IEEE*, vol. 85, pp. 1365–1388, Sept. 1997.

[2] A. K. Jain, R. M. Bolle, and S. Pankanti, Eds., *Biometrics: Personal Identification is a Networked Society*. Norwell, MA: Kluwer, 1999.

[3] H. C. Lee and R. E. Gaensslen, Eds., *Advances in Fingerprint Technology*, New York: Elsevier, 1991.

[4] L. O'Gorman, "Fingerprint verification," in *Biometrics: Personal Identification in a Networked Society*, A. K. Jain, R. Bolle, and S. Pankanti, Eds. Norwell, MA: Kluwer, 1999, pp. 43–64.

[5] D. Maio and D. Maltoni, "Direct gray-scale minutiae detection in fingerprints," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 27–40, Jan. 1997.

[6] A. K. Jain, S. Prabhakar, and L. Hong, "A multichannel approach to fingerprint classification," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, no. 4, pp. 348–359, 1999.

[7] G. T. Candela, P. J. Grother, C. I. Watson, R. A. Wilkinson, and C. L. Wilson, "PCASYS: A pattern-level classification automation system for fingerprints,", NIST Tech. Rep. NISTIR 5647, Aug. 1995.

[8] L. Hong and A. K. Jain, "Classification of fingerprint images," in *11th Scandinavian Conf. Image Analysis*, Kangerlussuaq, Greenland, June 7–11, 1999.

[9] M. Eshera and K. S. Fu, "A similarity measure between attributed relational graphs for image analysis," in *Proc. 7th Int. Conf. Pattern Recognition*, Montreal, P.Q., Canada, July 30–Aug. 3 1984.

[10] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, no. 4, pp. 377–388, 1996.

[11] A. K. Hrechak and J. A. McHugh, "Automated fingerprint recognition using structural matching," *Pattern Recognit.*, vol. 23, pp. 893–904, 1990.

[12] A. Ranade and A. Rosenfeld, "Point pattern matching by relaxation," *Pattern Recognit.*, vol. 12, no. 2, pp. 269–275, 1993.