

VEIN AUTHENTICATION ATM SECURITY IN BANKING USING TRAINED FEATURES

Submitted in partial fulfilment of the required for the award of

Masters of Computer Science

by

KAVYA S A-(42734008)



DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF SCIENCE AND HUMANITIES

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A++” by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600 119

APRIL 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC I Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of KAVYA S A (42734008) who carried out the project entitled "VEIN AUTHENTICATION ATM SECURITY IN BANKING USING TRAINED FETURES" under our supervision from January 2024 TO April 2024.

S. Rajeshwary
Internal Guide

Ms.RAJESHWARY S, M.Sc., Assistant Professor

Rekha Chakravarthi
Dean And Head of The Department

Dr . REKHA CHAKRAVARTHI M.E., Ph.D.

Submitted for Viva voce examination held on

20.04.2024

Sly.v

Internal Examiner

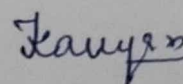
Rekha
External Examiner

DECLARATION

I am, KAVYA S A (42734008) here by declare that the Project Report entitled " VEIN AUTHENDICATION ATM SECURITY IN BANKING USING TRAINED FETURES" done by us under the guidance of Ms. RAJESHWARY S, M.Sc., Assistant Professor Department of Computer Science at SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600 119 is submitted in partial fulfilment of the requirements for the award of Master of Science degree in Computer Science.

DATE: 20-04-2024

PLACE: CHENNAI



KAVYA S A

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr . REKHA CHAKRAVARTHI M.E., Ph.D. Dean of Arts and Science** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my project guide **Ms. RAJESHWARY S, M.Sc., Assistant Professor** Department of Computer Science for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff member of the Department of Computer Science who were helpful in many ways for the completion of the project

KAVYA S A

ABSTRACT

This project presents new approach to develop the appearance of palm vein verification that uses blood vessel patterns as a personal identifying factor. The proposed approach of the vein is hard for replacement, since veins are internal to the human body and the palm vein authentication technology offers a high level of accuracy. The palm-vein-based approach attempts to be more effectively accommodating the potential deformations, revolving and translational changes by encoding the orientation preserving features. An Image Analysis the technique for Vascular Pattern of Hand Palm, which in turn leads towards the Palm Vein Authentication of an individual. It is a contactless authentication technology that offers an easy-to-use, hygienic solution to verify identity. This advanced, vascular pattern recognition solution is highly reliable and in a form factor that is non-intrusive and generates extremely fast authentication. A palm vein recognition system consists of the following steps: Image acquisition from the database and Pre-Processing, Finding of Region of interest, Extraction of Palm Vein pattern Features and Recognition. The aim of the proposed model is to improve the accuracy and response time of palm vein authentication. The proposed model will use neural networks for the final evaluation of the testing sample and training samples to recognize the person. The proposed model will be robust, flexible and accurate than the existing palm vein authentication schemes.

TABLE OF CONTENT

CH.NO	NAME OF THE TOPIC	PAGE NO.
1	INTRODUCTION	1-3
	1.1 Project Overview	1
	1.2 System Description	1
	1.3 Problem Statement	1
	1.4 Existing System	2
	1.5 Proposed System	3
2	LITERATURE SURVEY	4-6
3	SYSTEM DESIGN	7-11
	3.1 UML Design	7
	3.1.1 Use Case Diagram	7
	3.1.2 Sequence Diagram	9
	3.1.3 Activity Diagram	10
4	SYSTEM ARCHITECTURE	12-13
5.	SYSTEM IMPLEMENTATION	14-28
	5.1 Data Flow Diagram	14
	5.2 Data Dictionary	17
	5.3 ER Diagram	18
	5.4 Module Description	19
	5.4.1 Image Acquisition	19
	5.4.2 Pre-Processing	19
	5.4.3 Median Filteration	20

	5.4.4 Image Segmentation	21
	5.4.5 Boundary Detection	22
	5.4.6 Morphological Processing	23
	5.4.7 Regression Tree Classification	25
	5.4.8 Training and Testing	27
	5.4.9 Transaction	28
6	CODING AND RESULTS	29-45
	6.1 Sample Coding	29
	6.2 Screen shots	45
7	CONCLUSION AND FUTURE WORK	54
	7.1 Conclusion	54
	7.2 Future Work	54
8	REFERENCES	55

SL.NO	LIST OF THE FIGURES	PAGE NO
3.1	Use case diagram	08
3.2	Sequence Diagram	10
3.3	Activity Diagram	11
4.1	Architectural Diagram	13
5.1	Level 0 DFD	15
5.2	Level 1 DFD	16
5.3	ER Diagram	18
5.4	Image Acquisition Processing	19
5.5	Gray Scale Conversion	20
5.6	Image Segmentation	21
5.7	Boundary Detection	22
5.8	Boundary detection of the vein image	23
5.9	Morphological Processing	24
5.10	Morphological Processing of vein image	25
5.11	Regression classification	26
5.12	Regression classifier on feature classification	27
5.13	Training and Overview Testing	28
6.1	Feature Extraction	48

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

As veins are internal in the body and have a wealth of differentiating features, attempts to forge an identity are extremely difficult, thereby enabling a high level of security. Additionally, the sensor of the palm vein device can only recognize the pattern if the deoxidized hemoglobin is actively flowing within the individual's veins. The opportunities to implement Palm Secure span a wide range of vertical markets, including security, financial/banking, healthcare, commercial enterprises and educational facilities. Applications for the device include physical admission into secured areas; log-in to PCs or server systems; access to POS, ATMs or kiosks; positive ID control; and other industry-specific applications.

1.2 SYSTEM DESCRIPTION

This thesis will study and propose an implementation of a vein identification system. For such task, this thesis will be defined in different chapters that contain the information and explanations related to the data acquisition, the features and extraction for identification, the study of results, the design of the proposed solution and finally a conclusion that will explain the challenges found around palm vein identification and tips for hardware design. The procedure for which the information is captured, transformed, extracted information about the user and finally compared with a database samples is the palm vein identification process.

1.3 PROBLEM STATEMENT

Now a day's biometric is playing a key role in several fields. A captured image from any kind of sensing element must be processed before the extraction of features. This paper discusses about image processing approach for vein pattern. Hand vein patterns are among the biometric traits being investigated today for identification purposes, attracting interest from both the research community and industry. As an important member of biometric family, the vein patterns rely on the interior biological information of the body, and therefore, cannot be easily damaged, changed or falsified. This paper includes filtering techniques, contrast enhancement strategies

and segmentation processes. During this work, five different image filtering methods are used to remove noise from the image. This paper presents image enhancement operations and their result when applied on multispectral vein image. The noise removal and enhancement operations are much helpful to extract the vein pattern and features.

1.4 EXISTING SYSTEM

In existing system, personal identification number (PIN) is given for each individual to secure their amount. But this seems vulnerable to the risk of exposure and it can easily be forgotten. Fake cards can also be made for stealing the amount from the ATM. Recently many hackers have hacked people's account. For example, palm prints and palm prints are usually frayed; voice, signatures, hand shapes and iris images are easily forged. So we use biometric authentication process. The biometric systems which we come across frequently nowadays are palmprints, voice, hand geometry, palm print, iris or pupil, palm vein, palm vein and many more. Palmprint is an easy technique which we come across more often than the others even at public places not just for security purpose but also for attendance or as a substitute for signatures. Although the other systems have also been used quite regularly for security purpose and none of these are full proof and insusceptible to errors. All these biometric techniques have their pros and cons.

DISADVANTAGES

- Still now security issues are there in the system
- Biometric authentication can be easily known where the user secrets can be revealed easily

1.5 PROPOSED SYSTEM

In the acquired vein minutia images, the redundant points are eliminated. The elimination of redundant points will enhance the matching reliability and reduce computational complexity. Neighborhood elimination technique is employed for the purpose of removing the redundant information while keeping the effective source information for subsequent processing. This technique is applied on the normalized point-set of palm vein. For each point of palm vein, those points that lie within the neighborhood of a certain radius are removed. In order to determine the neighbors of each specific point, the spatial information is used. The main objective of this study is to analyze the most recent techniques applied in finger vein identification. The veins of each person get scanned and classified using the Regression Tree Algorithm. The person can be easily

identified using the implemented image processing system. The system can be added with two levels where the system training and testing process is achieved. At first the concern person images are get trained. After which they are classified and identified using the Regression Tree algorithm.

ADVANTAGES

- Authentication is made better accuracy in proposed system
- Secure ATM transaction is achieved
- The system most efficient and accurate for the user for the best authentication

CHAPTER 2

LITERATURE SURVEY

1. FINGERPRINT IMAGE ENHANCEMENT METHOD BASED ON ADAPTIVE MEDIAN FILTER

Ke Han ; Zizheng Wang ; Zilong Chen

The traditional median filtering method uses a fixed filter window size method to remove the impulse noise in a fingerprint image. If the filtering window size is small, the traditional median filtering method will not filter out the impulse noise completely. If the filtering window size is large, the fingerprint image may become blurred. To solve the problem, a method based on adaptive median filter is proposed for fingerprint image enhancement processing and impulse noise removal in the paper. The use of adaptive median filtering to remove the impulse noise of the fingerprint image mainly involves three steps. First, the size of the adaptive median filter window is initialized, and it is judged whether the center pixel of the filter window in the fingerprint image is impulse noise. Second, the size of the filter window is determined based on the median value, the maximum value, and the minimum value within the filter window. Finally, median filtering is performed on the fingerprint image under the filter window size obtained in the previous steps, and the filter output value is used instead of the window center pixel value. The method is tested on rolled fingerprint images contaminated by impulse noise and fingerprint images contaminated by impulse noise from a crime scene. Experimental results show that the method based on adaptive median filter for fingerprint image enhancement outperforms the traditional median filtering method in filtering impulse noise performance.

2. CONUNDRUM-PASS: A NEW GRAPHICAL PASSWORD APPROACH

Nida Asmat ; Hafiz Syed Ahmed Qasim

Graphical passwords are most widely used as a mechanism for authentication in today's mobile computing environment. This methodology was introduced to enhance security element and overcome the vulnerabilities of textual passwords, pins, or other trivial password methodologies which were difficult to remember and prone to external attacks. There are many graphical password schemes that are proposed over time, however, most of them suffer from

shoulder surfing and could be easily guessed which is quite a big problem. The proposed technique in this paper allows the user to keep the ease-to-use property of the pattern lock while minimizing the risk of shoulder surfing and password guessing. The proposed technique allows the user to divide a picture into multiple chunks and while unlocking, selecting the previously defined chunks results successfully in unlocking the device. This technique can effectively resist the shoulder surfing and smudge attacks, also it is resilient to password guessing or dictionary attacks. The proposed methodology can significantly improve the security of the graphical password system with no cost increase in terms of unlocking time.

3. EXTRACTION OF DORSAL PALM BASILIC AND CEPHALIC HAND VEIN FEATURES FOR HUMAN AUTHENTICATION SYSTEM

D. Sandhiya ; B. Thiyaneswaran

A human authentication in now a day is based on biometrie system. A hand vein biometrics is the upcoming technology. It has higher advantages than the existing biometrics. The hand vein consists of forearm wrist veins, palm veins, finger veins, and anterior positions. The proposed work extracts the dorsal palm vein features. The proposed system uses the hand vein capturing system. The focusing is based IR LED with 760 nm wavelength. The IR focusing system is placed inside the wooden box. The camera is used to capture the image. The captured RGB image is converted into HSV format. The hue, saturation, and brightness components are separated. An illumination correction is applied on the brightness component. The veins are segmented from the dorsal palm using k-means clustering. The GLCM features are extracted from the veins and taken as one part of the feature. Preprocessing steps such as median filter, binary conversion, image complement, and canny edge detection are applied on the captured image. The vein feature points are located in the captured image.

4. A METHOD FOR SECURING USERNAME AND PASSWORD AGAINST THE KEYLOGGER SOFTWARE USING THE LOGISTIC MAP CHAOS FUNCTION

M. Hossein Ahmadzadegan ; Ali-asghar Khorshidvand ; Mehrdad Pezeshki

Secure login by passwords in the sites and applications has always been a major concern by users. Meanwhile, many spywares like Keylogger have been developed that can record the users' input characters and then have access to their passwords. In this method, a capability has been created in which Keylogger Software cannot login the user system as they record input characters. The method is in the way that one-time passwords (OTP) are generated for the users utilizing the Logistic Map Chaos Function; these passwords are expired after being used by the user and they are not valid any more. An Android Software was used to generate one-time passwords. The user generates the one-time password by installing this Android Software on his/her smart phone and enters the password into the computer without any concern of the Keylogger. The results demonstrate the success for this method.

5. FAST ALGORITHM OF FINGERPRINT SINGULARITY REGION ENHANCEMENT

Shi-Qiong Xi ; Wei-Dong Geng

In this paper, a fast fingerprint enhancement algorithm that can improve the singularity enhancement of fingerprint is proposed. It can overcome the problem that the Gabor filter can destroy the ridge structure near the fingerprint singularity point, and the directional Fourier filter can not repair the fingerprint line effect is not obviously. The new method (FS-Gabor) can quickly repair fingerprint lines while retaining the fingerprint structure in the vicinity of the singular points. The experimental results show that the EER of the fingerprint image filtered by the FS-Gabor method is lower than the method of directional Fourier filter and Gabor.

CHAPTER 3

SYSTEM DESIGN

3.1 UML DIAGRAMS

A UML (Unified Modeling Language) diagram is a graphical representation of a system or process using standardized symbols and notation. UML diagrams are widely used in software engineering and other fields to visually depict various aspects of a system, such as its structure, behavior, interactions, and architecture.

There are several types of UML diagrams, each serving a different purpose and focusing on different aspects of the system being modeled. Some common types of UML diagrams include:

Unified Modeling Language (UML) diagram for the proposed automated college timetable scheduling system. Here's an example of a high-level UML class diagram for the system:

3.1.1 Use Case Diagram

A use case diagram is a graphical representation of the interactions between actors (users or external systems) and a system under consideration. It illustrates the various ways users interact with the system to accomplish specific tasks or achieve certain goals.

Key components of a use case diagram include:

Actors: Actors represent the users or external systems interacting with the system. They are typically depicted as stick figures or icons outside the system boundary. Actors can be individuals, other systems, or even time-based triggers.

Use Cases: Use cases represent specific functionalities or tasks that the system performs to fulfill user goals. Each use case describes a particular interaction between the user and the system. Use cases are depicted as ovals within the system boundary, often labeled with brief action-oriented descriptions.

Relationships: Relationships illustrate the connections between actors and use cases. The primary relationship in a use case diagram is the association between an actor and a use case, indicating which actors are involved in which use cases. Relationships can also show dependencies, generalizations, or extensions between use cases.

System Boundary: The system boundary defines the scope of the system being modeled. It encloses all the internal components and use cases of the system, distinguishing them from external actors and systems.

Use case diagrams are valuable tools for understanding and communicating system requirements because they provide a high-level overview of system functionality from a user's perspective. They help stakeholders visualize how users will interact with the system and the tasks they will perform, facilitating discussions about system behavior and requirements..

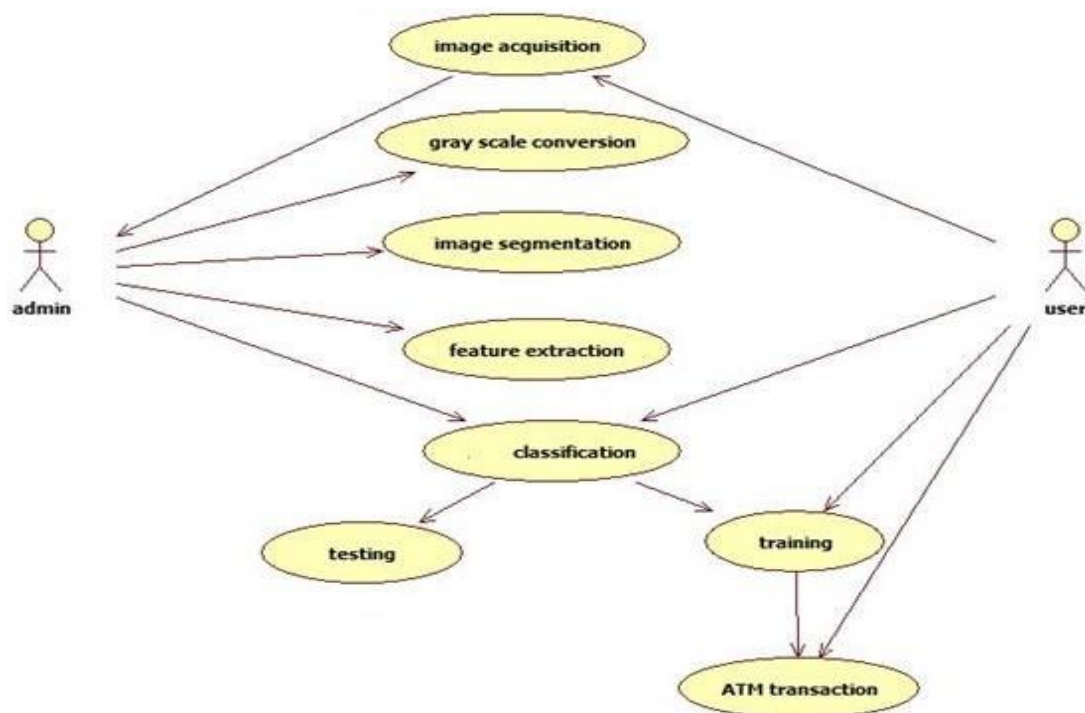


Fig 3.1 Use case diagram

3.1.2 Sequence Diagram

A Sequence Diagram is a type of UML (Unified Modeling Language) diagram that illustrates the interactions between objects or components within a system in a sequential manner. It focuses on the flow of messages or method calls exchanged between these objects over time to accomplish a specific behavior or scenario.

Key components of a sequence diagram include:

Objects/Participants: Represent the entities (objects, components, or actors) involved in the interaction. Each object is typically represented by a box or lifeline vertically aligned along the diagram's left side.

Lifelines: Vertical lines extending downwards from the objects, representing the existence of the objects over time during the interaction.

Messages: Arrows drawn between lifelines, indicating the flow of communication between objects. Messages can be synchronous (e.g., method calls), asynchronous (e.g., signals or events), or replies.

Activation/Execution Occurrences: Represented by narrow rectangles on lifelines, showing the duration during which an object is actively processing a message or performing an operation.

Optional Elements: Other optional elements may include conditionals (such as alternative or loop constructs), iterations, and parallel execution to capture more complex scenarios. This sequence diagram illustrates the sequential flow of interactions between the Administrator, the College Timetable System, and the Staff during the process of generating and approving a timetable.

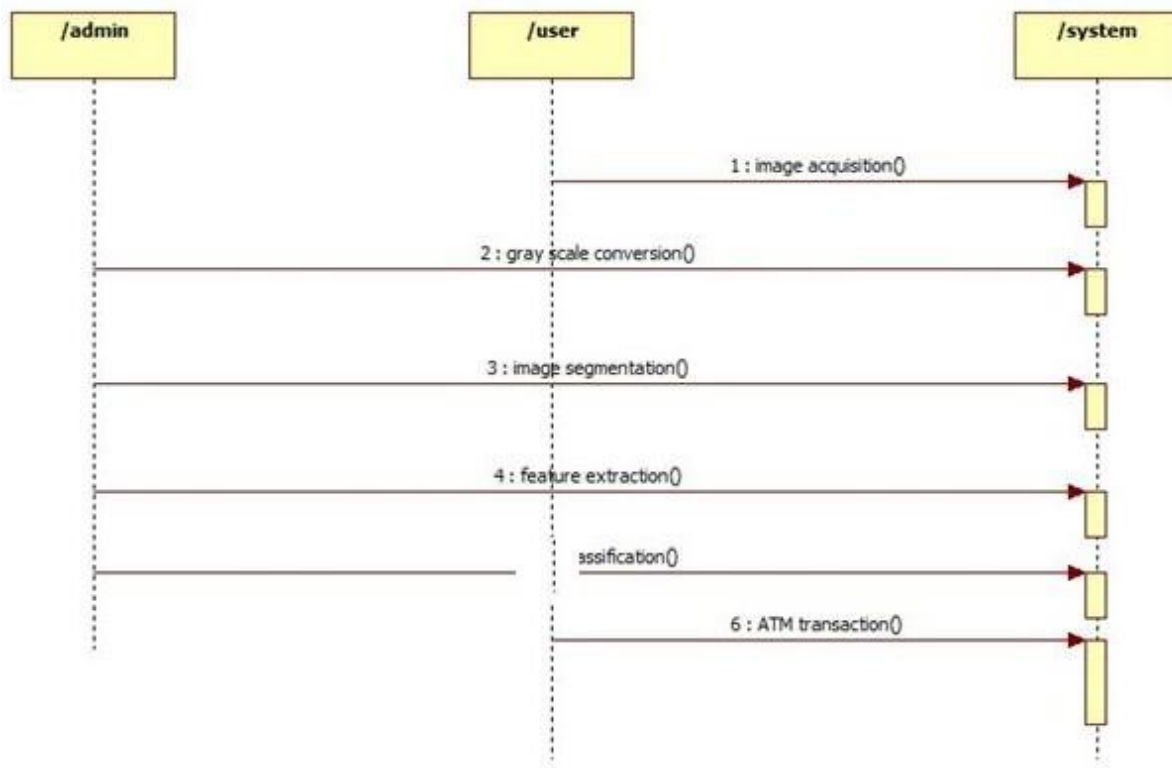


Fig 3.2 Sequence Diagram

3.1.3 Activity Diagram

An Activity Diagram is a type of UML (Unified Modeling Language) diagram that illustrates the flow of activities or actions within a system or a specific business process. It visually represents the sequence of actions, decisions, and transitions from one activity to another, allowing stakeholders to understand the workflow or behavior of a system or process.

Key components of an activity diagram include:

Activities: Represented by rounded rectangles, activities depict specific actions or steps within the system or process. Each activity represents a task that needs to be performed.

Transitions: Arrows connecting activities, indicating the flow of control from one activity to another. Transitions represent the order in which activities are executed, typically triggered by the completion of a preceding activity or by the satisfaction of a condition.

Decision Nodes: Diamond-shaped symbols used to represent decision points or branches in the workflow. Depending on the outcome of a decision, the flow may proceed along different paths.

Merge Nodes: Used to consolidate multiple flow paths back into a single path after a decision or branching point.

Initial Node: Indicates the starting point of the activity diagram. It represents the initiation of the workflow or process

Final Node: Represents the endpoint of the activity diagram, indicating the completion of the workflow or process.

Fork and Join Nodes: Fork nodes split the flow into multiple parallel paths, while join nodes merge parallel paths back into a single flow.

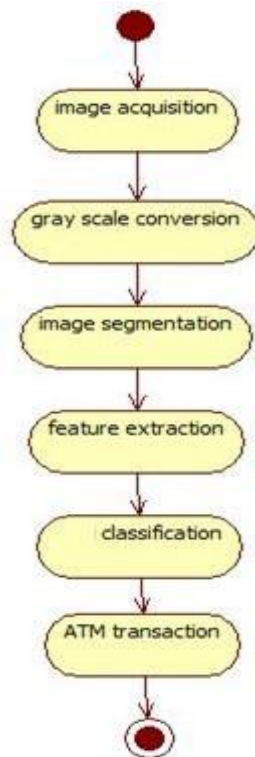


Fig 3.3 Activity Diagram

CHAPTER 4

SYSTEM ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. System architects or solution architects are the people who know what components to choose for the specific use case, make right trade-offs with awareness of the bottlenecks in the overall system. Usually, solution architects who have more years of experience tend to be good at system architecting because system design is an open-ended problem, there is no one correct solution. Mostly it's trial and error experiments done with right trade-offs. So, experience teaches you what components to choose based on the problem at hand but in order to gain that experience you need to start somewhere. That's why I am writing this article for the developers who have never done system designs.

The system architecture is mainly based on the fuzzy nature analytics where the learner behavior while learning a course and his style of learning is efficiently predicted using the intuitionistic fuzzy logic. The observer system can be designed with the modeling of various attributes like knowledge creditability, learner aggregation, learner objects and so

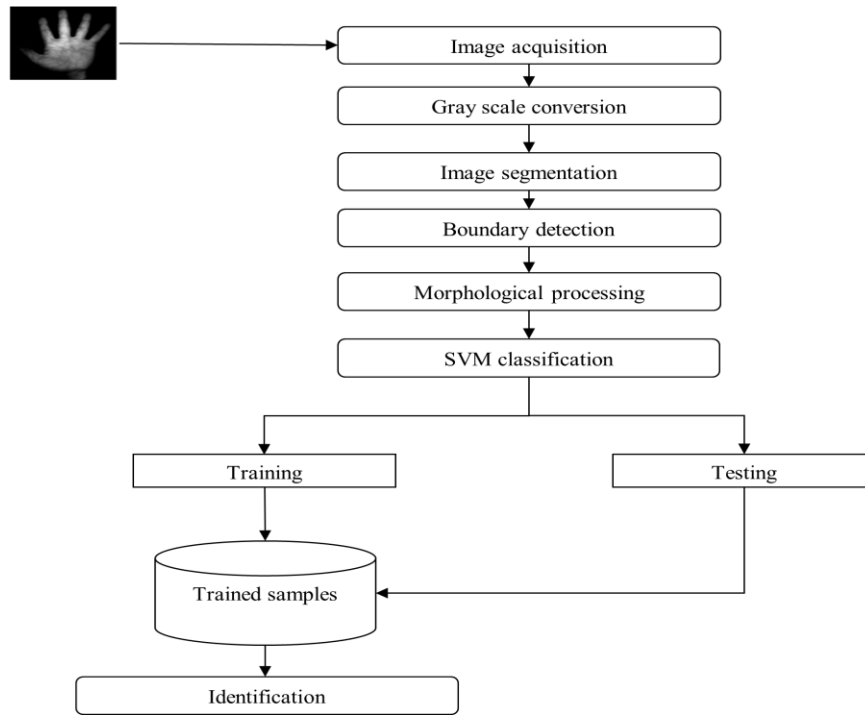


Fig 4.1 Architecture Diagram

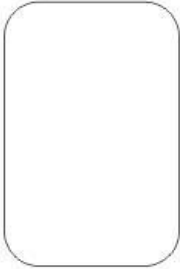
CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 DATA FLOW DIAGRAM

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

Data flow Symbols:

Symbol	Description
	An entity . A source of data or a destination for data.
	A process or task that is performed by the system.
	A data store , a place where data is held between processes.
	A data flow .

LEVEL 0

DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

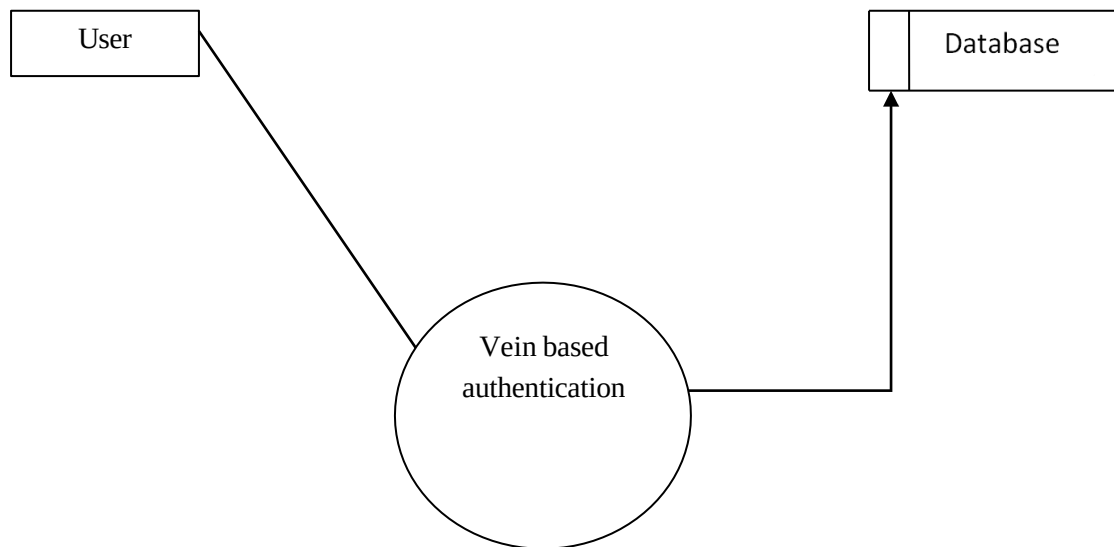


Fig 5.1 level 0 DFD

LEVEL 1

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub-processes. A level 1 data flow diagram (DFD) is more detailed than a level 0 DFD but not as detailed as a level 2 DFD. It breaks down the main processes into sub processes that can then be analyzed and improved on a more intimate level.

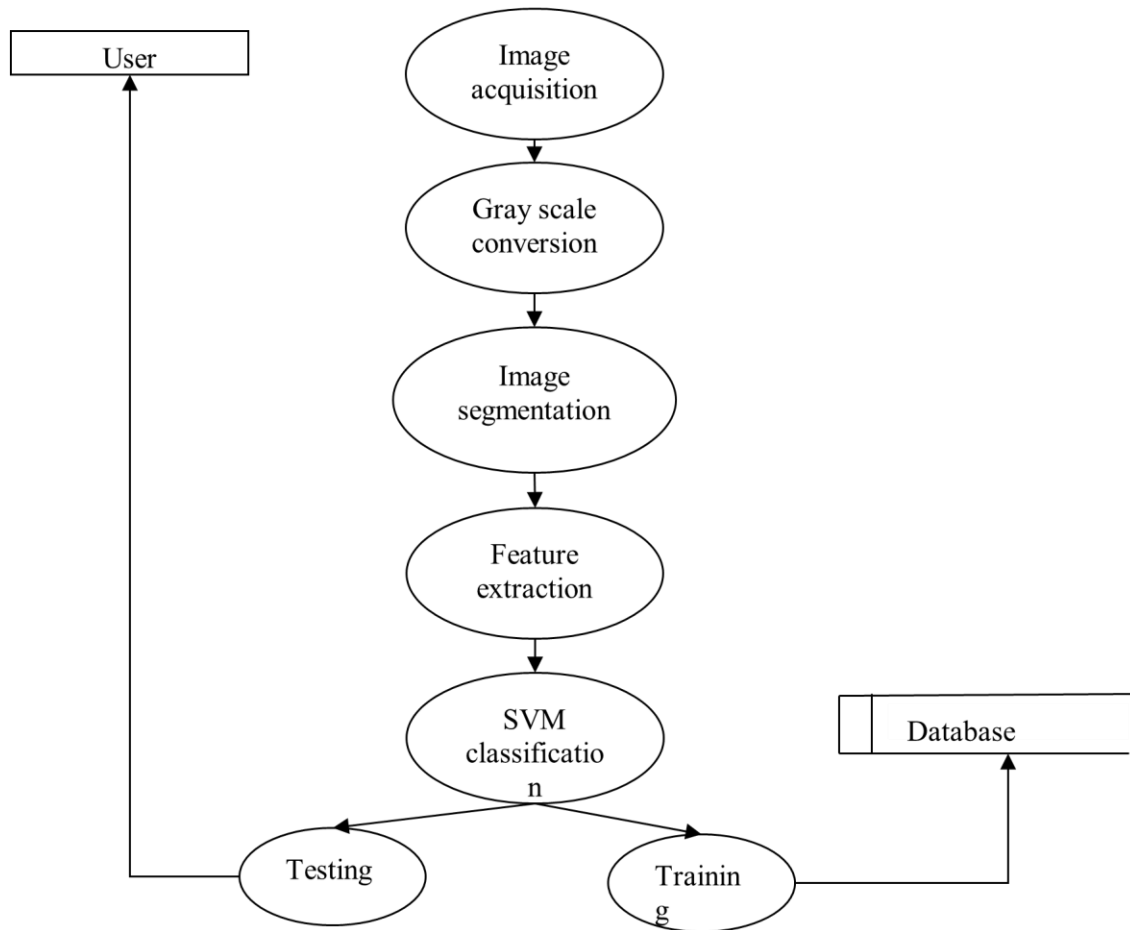


Fig 5.2 Level 1 DFD

5.2 DATA DICTIONARY

User Register table

Field	Type	Null	Default
id	int(100)	Yes	NULL
name	varchar(100)	Yes	NULL
contact	varchar(100)	Yes	NULL
email	varchar(100)	Yes	NULL
address	varchar(100)	Yes	NULL
accnumber	varchar(100)	Yes	NULL
password	varchar(100)	Yes	NULL
cdate	varchar(100)	Yes	NULL
histo	varchar(100)	Yes	NULL
sca	varchar(100)	Yes	NULL
svm	varchar(100)	Yes	NULL
status	varchar(100)	Yes	NULL
report	varchar(100)	Yes	NULL
balance	varchar(100)	Yes	NULL

Mini Statement table

Field	Type	Null	Default
accnum	varchar(255)	Yes	NULL
process	varchar(255)	Yes	NULL
amount	varchar(255)	Yes	NULL
receiver	varchar(255)	Yes	NULL
status	varchar(255)	Yes	NULL
report	varchar(255)	Yes	NULL

5.3 ER DIAGRAM

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology (IT) system.

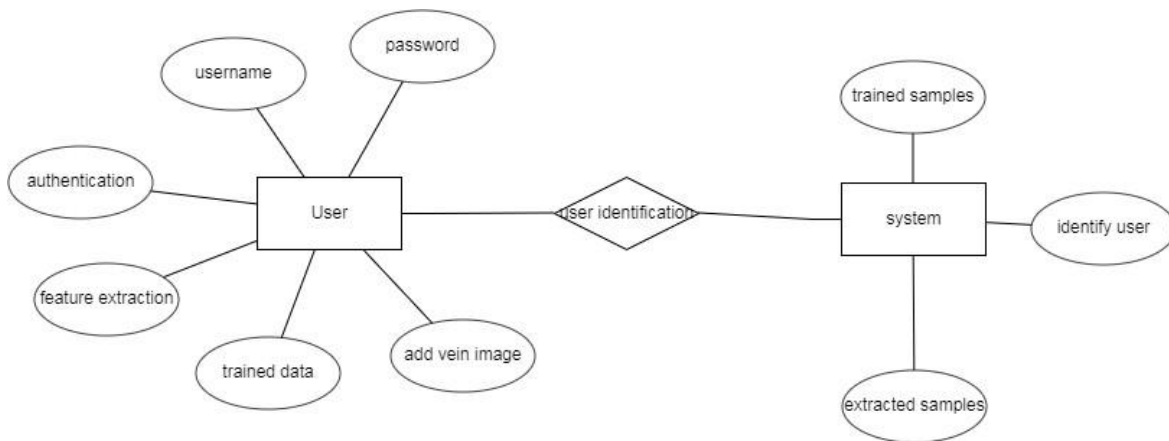


Fig 5.3 ER Diagram

5.4 MODULE DESCRIPTION

5.4.1 IMAGE ACQUISITION

The hand of a person is put over the detecting component to get the basic highlights of the vein designs. Veins are found underneath the skin and in this way, it is exceptionally troublesome to get the vein design in unmistakable light. To capture the vein pictures, a CCD camera with close infrared is utilized. Picture procurement is the primary fundamental step in FVR in which the finger vein picture is captured by utilizing NIR (near infra-red) light within the brightening exchange strategy. The securing gadget comprises of an NIR gathering portion for situation of the finger, and a charge-coupled gadget (CCD) preprocessor camera is at that point utilized to get an image. The gotten picture is included for the picture input and finds the individual.

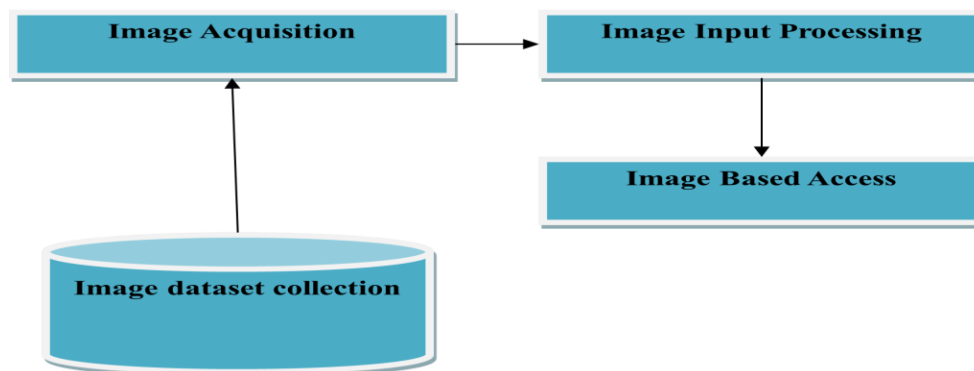


Fig 5.4 Image Acquisition Processing

5.4.2 PRE-PROCESSING

Gray Scale conversion

Converting an RGB image to grayscale, we have to take the RGB values for each pixel and make as output a single value reflecting the brightness of that pixel. One such approach is to take the average of the contribution from each channel.

$$\text{Gray Scale} = (R+G+B)/3 \quad (6.1)$$

The luminance of a pixel value of a grayscale image ranges from 0 to 255. The conversion of a color image into a grayscale image is converting the RGB values (24 bit) into grayscale value (8 bit). Various image processing techniques and software applications convert color image to grayscale image.

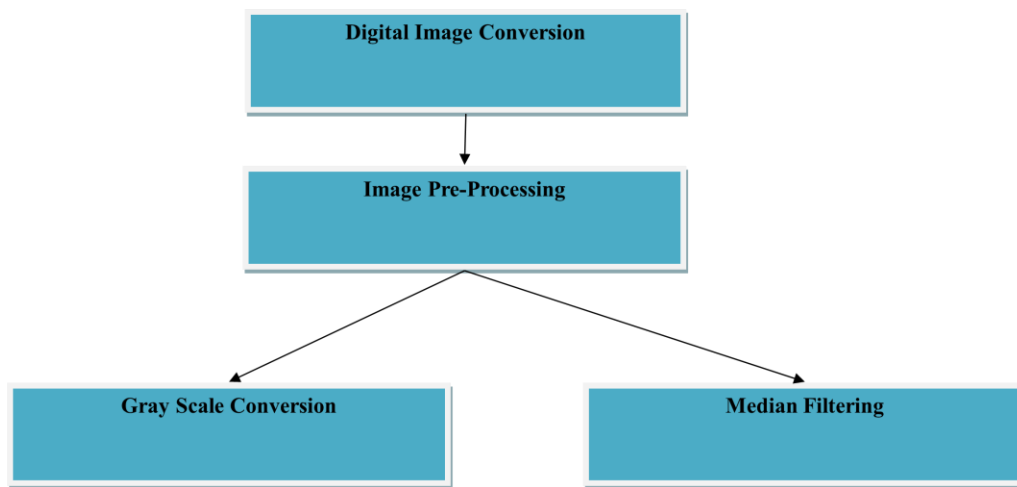


Fig 5.5 Gray Scale Conversion

5.4.3 MEDIAN FILTERING

In median filtering, they utilized the low pass Gaussian channel (3 * 3) and high pass (11 * 11) to get a great picture vein patterns. Others have favored to utilize the Middle channel (5 * 5) instead of Gaussian within the case of the Gaussian filter remove a few required edges, as an elective the Median filter that moderates the edges.

On the other hand the median filtering utilized the two channels: Low pass Gaussian components to decrease the picture having a high frequency (dull pixels) and Middle (5 * 5) to remove speckles (the little employments that show up within the picture surface). Another method has been adjusted in; the ridge let transform in arrange to kill commotion and highlight

the veins. In [8], they utilized Coordinate channel, Wiener channel, smoothing filter to dispense with commotion and get clear veins.

5.4.4 IMAGE SEGMENTATION

In computer vision, image segmentation is the process of portioning a digital image into multiple segments (set of pixels). The original image can be changed into something meaningful and easier way to analyze. Here Gaussian bi-lateral filter is used for enhancement. A bilateral filter is a non-linear, edge –preserving and noise reducing smoothing filter for images. It replaces the intensity values from nearby pixels . The utilization of a fuzzy segmentation method avoids identifying the finger vein contour to isolate it from the finger vein pattern during the extraction. This project proposes for such a task an algorithm similar to the one found

Usually the acquisition of the vein pattern is limited to an area inside the finger. However, the experimentation during this project showed that including the finger shape as one more finger vein, fusing it with the vein closer to the border of the finger, improved the end user experience, increasing the probability that a genuine user was identified.

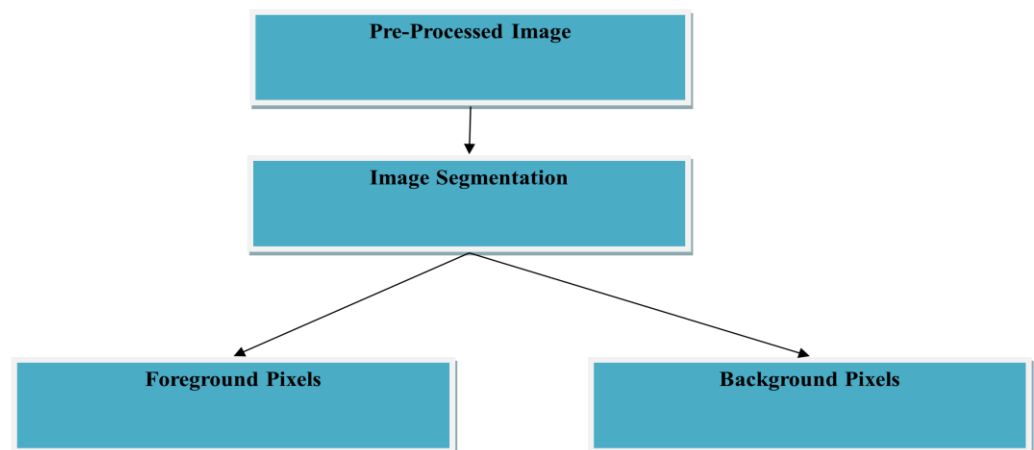


Fig 5.6 Image Segmentation

5.4.5 BOUNDARY DETECTION

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision. The boundary of the user palm which is segmented will be made here to identify the user. The user will make the transaction processing which are stored. The stored data processing can be made with the implementation of the user verification.

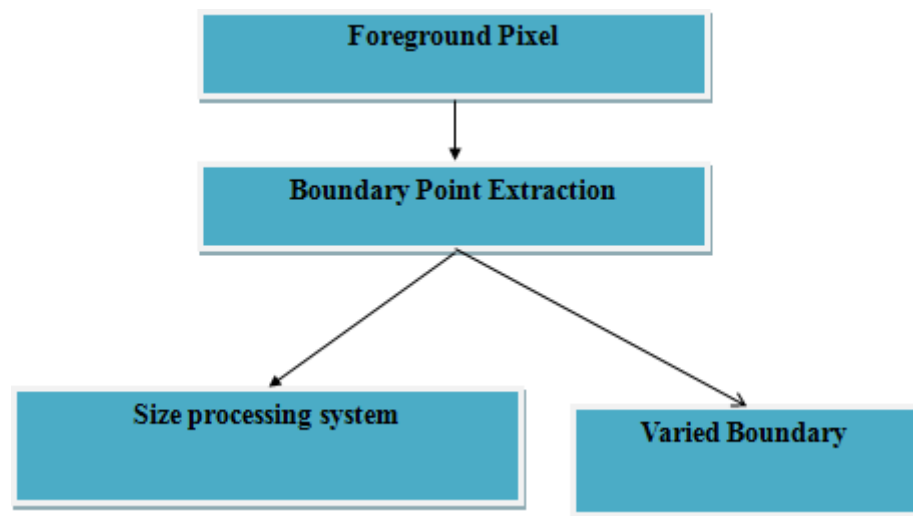


Fig 5.7 Boundary Detection

Boundary detection is the process of edge location of the image to find out the edge point and the co-ordinates. The segmentation gives out the regions of the image provided which tends in the detection over the exact path in the computer vision. Thus the exact path gives out the detection of the layers in the image. The boundary of the client palm which is portioned will be made here to distinguish the client. The client will make the exchange preparing which are put away. The put away information preparing can be made with the execution of the client verification.



Fig 5.8 Boundary detection of the vein image

An internal line of the finger is to begin with identified to separate the finger vein picture by two parts, after that two edge discovery formats and a arrangement of advances such as addition, fit, etc. are utilized to distinguish and settle the off-base edges of the finger. Besides, considering that the shapes of the brighter finger joint locale are sporadic, numerous sliding windows counting rectangle, disk, precious stone and oval are created, individually to distinguish the reference line of the finger joint. At last, a form similitude distance-based strategy is presented to assess the execution of different sliding windows. The exploratory comes about appear that the proposed edge discovery strategy can 100% effectively identify the edges of the fingers in our finger vein picture database. And for different discovery windows, the circle window is more reasonable for the location of the finger joint reference line.

5.4.6 MORPHOLOGICAL PROCESSING

Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. A morphological operation on a binary image creates a new binary image in which the pixel has a non-zero value only if the test is successful

at that location in the input image. The vein based authentication will be analyzed with the implemented process.

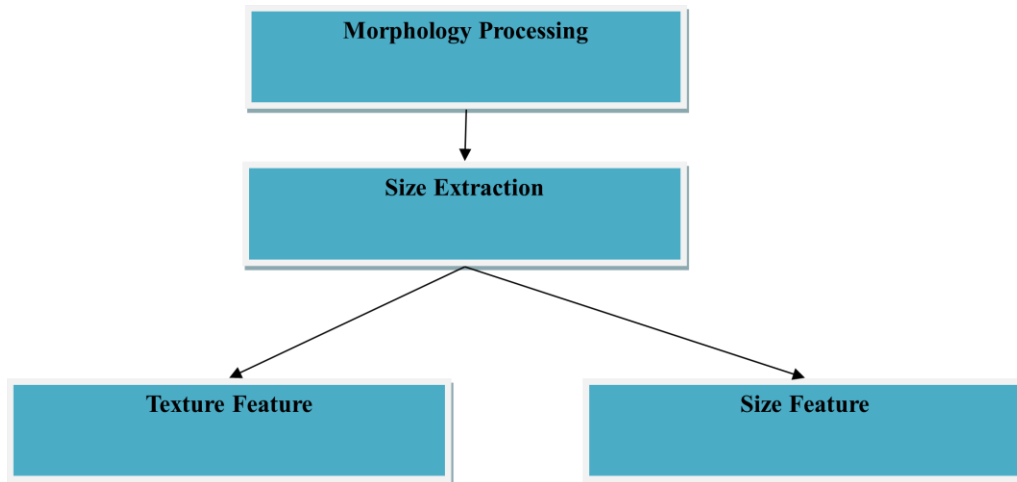


Fig 5.9 Morphological Processing

Morphological processing is the collection over the morphological size variations of the image which are gathered. This module lends to differentiate the pattern of the vein in the picture of each authentication levels. The process starts from the non- zero value then the pixel variation with the brightness and the intensity levels. Parallel image matching will be given where the test will be done using the verified picture processing steps. Thus the variation will be known with their actual size of matching.

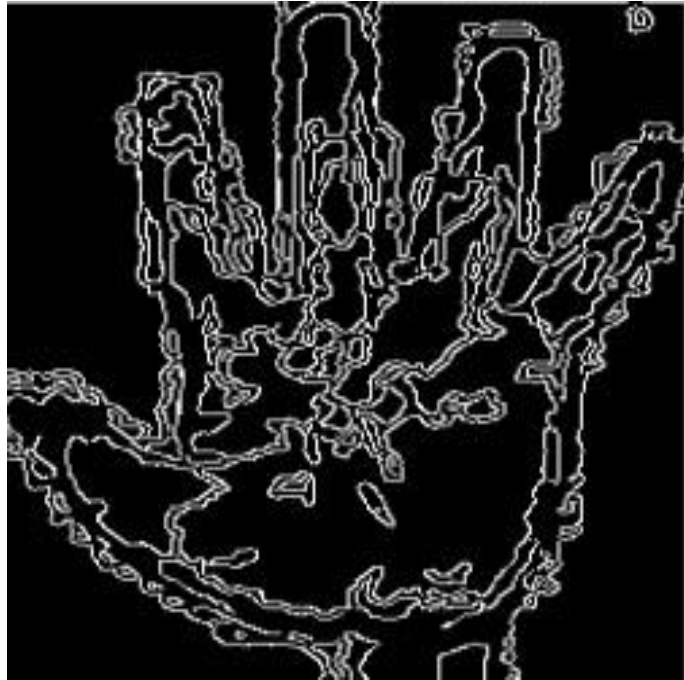


Fig 5.10 Morphological Processing of vein image

Highlights extraction from pictures of dorsal hand veins takes place after essential pre-processing stage. So, four pictures per each individual are given to the framework as input at that point pre-processing stages performed on every single picture. In another step, an interesting and common feature apportioned to all four pictures and some of random points will be chosen for each person separately. After the essential pre-processing arrange, per add up to number of input pictures of dorsal hand veins for each individual, value I is tallied as an irregular esteem.

5.4.7 Regression Tree classification

Regression Tree Classification is a supervised learning models with associated learning algorithms that analyze data used for classifications and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories. Regression Tree training algorithms builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

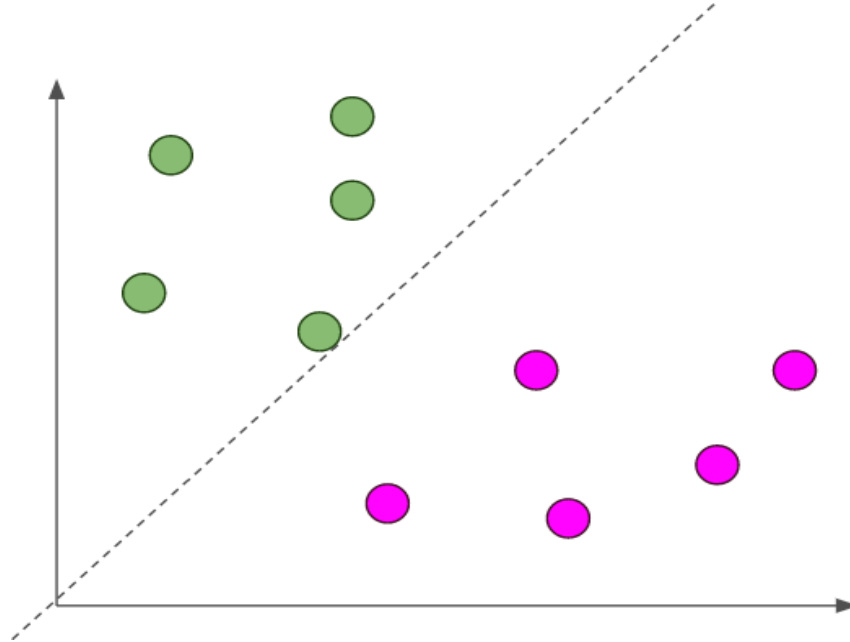


Fig 5.11 Regression Classification

As Regression model is a representation of the examples as points in space are mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible in the implemented system. New examples are then mapped into that same space and predicted to a category based on which side of the gap they fall on.

To compare the shape of the extricated veins (s) measure (s) of reference utilized the Hausdorff remove, the test comes about appeared that all the vein design pictures in the database have been accurately recognized. Used Regression classifier, the mistake rate of Distant and FRR was still decreased. Others11 have utilized the strategy of minutiae triangulation, which can frame triangles from minutiae (bifurcation focuses and conclusion focuses). At that point a score is generated by computing the number of triplet's bifurcation and conclusion focuses of the input picture, and compared with the number of the triplets and conclusion focuses that are put away in the database.

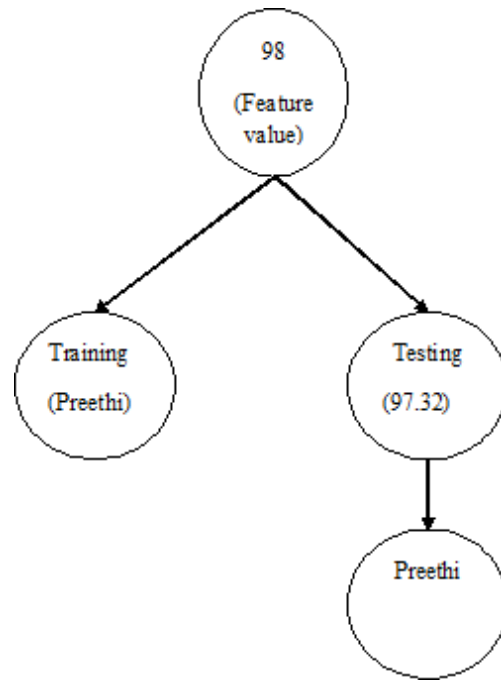


Fig 5.12 Regression classifier on feature classification

The regression classifier known with template matching of the given image based on the classified tree analysis. Here the extracted feature points will be given with different layout source picture. The matching approach will be on the co-ordinate points which are extracted at the boundary detection. The whole part is taken with the histogram patterning of all area which gives the intensity value, morphological value etc. The test and the given image analyzed will be completely trained with the extracted feature pattern and then the coordinates are matched. The format pattern might be known with the highlight pattern may be noticed. Mainly the system will give a coordinating value of the testing and the training phase system. The Tree maintained with the left and right sub node will give out the functional analysis.

5.4.8 Training and Testing

From the extracted features the user features are classified and registered with their name. If the user vein is making visible then the tested values will be matched with trained values. Here the database is get known and the user will get an access to their account.

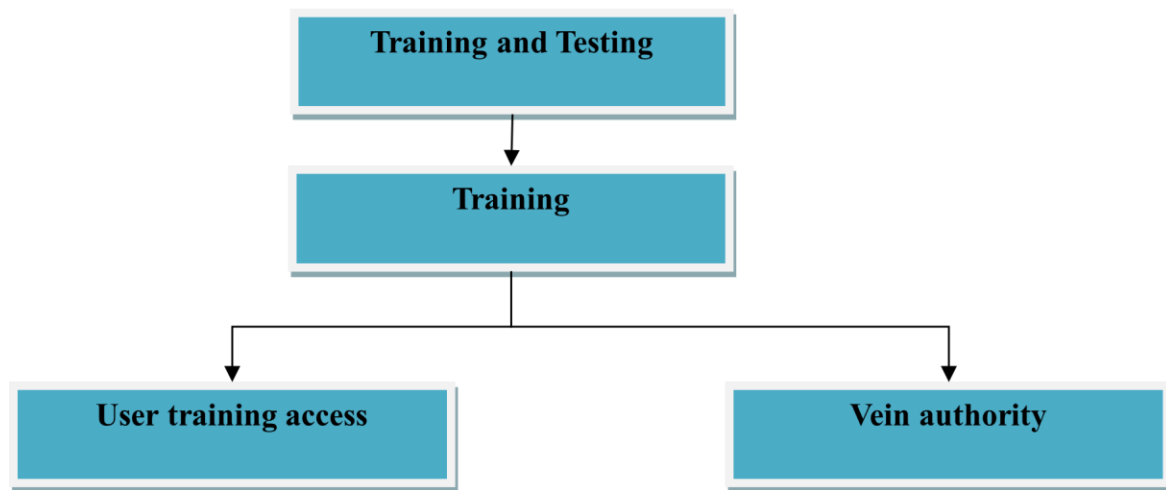


Fig 5.13 Training and Overview Testing

5.4.9 Transaction

In the transaction pattern the user will get an account number where they will add the transfer and deposit amount. This may be helpful for the user to secure the access from the unknown third party access.

CHAPTER 6

CODING AND RESULTS

6.1 SAMPLE CODING

Home

```
import datetime
import os
import shutil
from skimage.io import imread
import numpy as np
import matplotlib.pyplot as plt
from tkinter.ttk import Treeview
import time
import cv2
from tkinter import Tk, messagebox, ttk
from tkinter import *
from scipy.fftpack import dct, idct
from tkinter.filedialog import askopenfilename
from tkinter.messagebox import askyesno
from PIL import Image, ImageTk
import numpy as np
import ar_master
from m_bpnn import BPNNNetwork
import tkinter.simpledialog
mm= ar_master.master_flask_code()
class ar_vein():
    image=""
    path=""
    gray=""
    fname=""
```

feature_value=0

```

username=""
def __init__(self):
    self.master = 'ar_master'
    self.title = 'Vein Authentication based Banking'
    self.titlec = 'VEIN AUTHENTICATION BASED BANKING'
    self.background_color = '#2F4F4F'
    self.text_color = '#FFF'
    self.background_image = 'images/background_hd.jpg'
    self.account_no=""
def get_title(self):
return self.title
def get_titlec(self):
return self.titlec
def get_background_color(self):
return self.background_color
def get_text_color(self):
return self.text_color
def get_background_image(self):
return self.background_image
def get_account_no(self):
return self.account_no
def set_account_no(self,acc):
    self.account_no=acc
def home_window(self):
    home_window_root=Tk()
    w = 950
    h = 600
ws = home_window_root.winfo_screenwidth()
hs = home_window_root.winfo_screenheight()
x = (ws / 2) - (w / 2)
y = (hs / 2) - (h / 2)

```

```

home_window_root.geometry('%dx%d+%d+%d' % (w, h, x, y))
self.bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
home_window_root.title(self.title)
home_window_root.resizable(False, False)
bg = ImageTk.PhotoImage(file=self.background_image)
canvas = Canvas(home_window_root, width=200, height=300)
canvas.pack(fill="both", expand=True)
    canvas.create_image(0, 0, image=bg, anchor=NW)
    canvas.create_text(470, 40, text=self.title, font=("Times New Roman", 24),
fill=self.text_color)
def clickHandler(event):
tt = ar_vein
    tt.user_login(event)
image = Image.open('images/admin.png')
img = image.resize((150, 150))
my_img = ImageTk.PhotoImage(img)
image_id = canvas.create_image(350, 170, image=my_img)
canvas.tag_bind(image_id, "<1>", clickHandler)
###
def clickHandler1(event):
tt = ar_vein
    tt.user_registration(event)
image1 = Image.open('images/users1.png')
img1 = image1.resize((150, 150))
my_img1 = ImageTk.PhotoImage(img1)
image_id1 = canvas.create_image(350, 370, image=my_img1)
canvas.tag_bind(image_id1, "<1>", clickHandler1)
###
admin_id = canvas.create_text(570, 170, text="LOGIN", font=("Times New Roman", 24),
fill=self.text_color)
canvas.tag_bind(admin_id, "<1>", clickHandler)

```

```

###
admin_id1 = canvas.create_text(570, 370, text="REGISTRATION", font=("Times New
Roman", 24), fill=self.text_color)
canvas.tag_bind(admin_id1, "<1>", clickHandler1)
home_window_root.mainloop()
def user_registration(self):
    user_registration_root = Toplevel()
    user_registration_root.attributes('-topmost', 'true')
    get_data = ar_vein()
    w = 950
    h = 600
ws = user_registration_root.winfo_screenwidth()
hs = user_registration_root.winfo_screenheight()
    x = (ws / 2) - (w / 2)
    y = (hs / 2) - (h / 2)
    user_registration_root.geometry('%dx%d+%d+%d' % (w, h, x, y))
    self.bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
    user_registration_root.title(get_data.get_title())
    user_registration_root.resizable(False, False)
bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
canvas = Canvas(user_registration_root, width=200, height=300)
canvas.pack(fill="both", expand=True)
    canvas.create_image(0, 0, image=bg, anchor=NW)
    admin_id2 = canvas.create_text(500, 70, text="USER REGISTRATION", font=("Times
New Roman", 24), fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(200, 200, text="PATH", font=("Times New Roman", 24),
fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(200, 325, text="FILE", font=("Times New Roman", 24),
fill=get_data.get_text_color())
    e1 = Entry(user_registration_root, font=('times', 15, 'bold '),width=40)
    canvas.create_window(480, 200, window=e1)

```

```

ws = median_filter_root.winfo_screenwidth()
hs = median_filter_root.winfo_screenheight()
    x = (ws / 2) - (w / 2)
    y = (hs / 2) - (h / 2)
    median_filter_root.geometry('%dx%d+%d+%d' % (w, h, x, y))
    self.bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
    median_filter_root.title(get_data.get_title())
    median_filter_root.resizable(False, False)
bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
canvas = Canvas(median_filter_root, width=200, height=300)
canvas.pack(fill="both", expand=True)
    canvas.create_image(0, 0, image=bg, anchor=NW)
    admin_id2 = canvas.create_text(500, 70, text="NOISE REMOVE", font=("Times New
Roman", 24),
fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(200, 195, text="GRAYSCALE IMAGE", font=("Times
New Roman", 15),
fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(500, 195, text="DCT FILTER", font=("Times New
Roman", 15),
fill=get_data.get_text_color())
def select_image():
    canvas.itemconfig(image_id2, state='normal')
    canvas.update()
    time.sleep(1)
def next_image():
    median_filter_root.destroy()
tt = ar_vein()
    tt.boundary_detection()
    image1 = Image.open('data/greyscale.png')

```

```

img1 = image1.resize((250, 250))
my_img = ImageTk.PhotoImage(img1)
image_id1 = canvas.create_image(300, 350, image=my_img)

def dct2(a):
    return dct(dct(a.T, norm='ortho').T, norm='ortho')
def dct1(a):
    return idct(idct(a.T, norm='ortho').T, norm='ortho')
im = (imread(r'data/input.png'))
imF = dct2(im)
    im1 = dct1(imF)
dd = np.allclose(im, im1)
np.allclose(im, im1)
path=('data\dct.png')
cv2.imwrite(path, im1)
    image1 = Image.open('data/dct.png')
    img1 = image1.resize((250, 250))
    my_img1 = ImageTk.PhotoImage(img1)
global image_id2
    image_id2 = canvas.create_image(600, 350, image=my_img1, state='hidden')
    b1 = Button(median_filter_root, text='DCT Filter', command=lambda: select_image(),
font=('times', 15, ' bold '), width=20)
b1.pack()
    canvas.create_window(300, 525, window=b1)
    b2 = Button(median_filter_root, text='Next', command=next_image, font=('times', 15, ' bold
'), width=20)
    canvas.create_window(600, 525, window=b2)
    median_filter_root.mainloop()

def boundary_detection(self):
    boundary_detection_root = Toplevel()
    boundary_detection_root.attributes('-topmost', 'true')
    get_data = ar_vein()

```

```

w = 950
h = 600
ws = boundary_detection_root.winfo_screenwidth()
hs = boundary_detection_root.winfo_screenheight()
x = (ws / 2) - (w / 2)
y = (hs / 2) - (h / 2)
boundary_detection_root.geometry('%dx%d+%d+%d' % (w, h, x, y))
self.bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
boundary_detection_root.title(get_data.get_title())
boundary_detection_root.resizable(False, False)
bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
canvas = Canvas(boundary_detection_root, width=200, height=300)
canvas.pack(fill="both", expand=True)
    canvas.create_image(0, 0, image=bg, anchor=NW)
    # canvas.create_text(390, 20, text=get_data.get_title(), font=("Times New Roman", 24),
fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(500, 70, text="BOUNDARY DETECTION", font=("Times
New Roman", 24),
fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(250, 195, text="DCT FILTER", font=("Times New
Roman", 15),
fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(550, 195, text="DETECTION", font=("Times New
Roman", 15),
fill=get_data.get_text_color())
def select_image():
    canvas.itemconfig(image_id2, state='normal')
    canvas.update()
    time.sleep(1)
def next_image():
    boundary_detection_root.destroy()

```

```

tt = ar_vein()
    tt.morphological_processing()
    image1 = Image.open('data/dct.png')
    img1 = image1.resize((250, 250))
    my_img = ImageTk.PhotoImage(img1)
    image_id1 = canvas.create_image(300, 350, image=my_img)
im = (imread(r'data/input.png'))
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 20, 30)
    edges_high_thresh = cv2.Canny(gray, 60, 120)
    im1 = np.hstack((gray, edges, edges_high_thresh))
path = ('data\\boundary.png')
cv2.imwrite(path, edges_high_thresh)
    image1 = Image.open('data/boundary.png')
    img1 = image1.resize((250, 250))
    my_img1 = ImageTk.PhotoImage(img1)
global image_id2
    image_id2 = canvas.create_image(600, 350, image=my_img1, state='hidden')
    b1 = Button(boundary_detection_root, text='BOUNDARY DETECTION',
command=lambda: select_image(),
font=('times', 15, 'bold '), width=20)
b1.pack()
    canvas.create_window(300, 525, window=b1)
    b2 = Button(boundary_detection_root, text='Next', command=next_image, font=('times', 15,
' bold '), width=20)
    canvas.create_window(600, 525, window=b2)
    boundary_detection_root.mainloop()
def morphological_processing(self):
    morphological_processing_root = Toplevel()
    morphological_processing_root.attributes('-topmost', 'true')
    get_data = ar_vein()

```

```

w = 950
h = 600
ws = morphological_processing_root.winfo_screenwidth()
hs = morphological_processing_root.winfo_screenheight()
x = (ws / 2) - (w / 2)
y = (hs / 2) - (h / 2)
morphological_processing_root.geometry('%dx%d+%d+%d' % (w, h, x, y))
self.bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
morphological_processing_root.title(get_data.get_title())
morphological_processing_root.resizable(False, False)
bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
canvas = Canvas(morphological_processing_root, width=200, height=300)
canvas.pack(fill="both", expand=True)
    canvas.create_image(0, 0, image=bg, anchor=NW)
    admin_id2 = canvas.create_text(500, 70, text="MORPHOLOGICAL PROCESS",
font=("Times New Roman", 24),
fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(250, 195, text="DETECTION", font=("Times New
Roman", 15),
fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(550, 195, text="MORPHOLOGICAL", font=("Times New
Roman", 15),
fill=get_data.get_text_color())
def select_image():
    canvas.itemconfig(image_id2, state='normal')
    canvas.update()
    time.sleep(1)
def next_image():
    morphological_processing_root.destroy()
tt = ar_vein()
    tt.feature_extraction()

```

```

image1 = Image.open('data/boundary.png')

user_login_root.title(get_data.get_title())
user_login_root.resizable(False, False)
bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
canvas = Canvas(user_login_root, width=200, height=300)
canvas.pack(fill="both", expand=True)
    canvas.create_image(0, 0, image=bg, anchor=NW)
    admin_id2 = canvas.create_text(500, 70, text="USER LOGIN", font=("Times New
Roman", 24), fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(200, 200, text="PATH", font=("Times New Roman", 24),
fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(200, 325, text="FILE", font=("Times New Roman", 24),
fill=get_data.get_text_color())
    e1 = Entry(user_login_root, font=('times', 15, 'bold '),width=40)
    canvas.create_window(480, 200, window=e1)
    e2 = Entry(user_login_root, font=('times', 15, 'bold '),width=40)
    canvas.create_window(480, 325, window=e2)
def select_image():
    dir = r'data'
    if not os.path.exists(dir):
        os.mkdir(dir)
    for f in os.listdir(dir):
        os.remove(os.path.join(dir, f))
    if not os.path.exists(dir):
        os.makedirs(dir)
        csv_file_path = askopenfilename( parent=user_login_root)
    fpath = os.path.dirname(os.path.abspath(csv_file_path))
    fname = (os.path.basename(csv_file_path))
    fsize = os.path.getsize(csv_file_path)
    e1.delete(0, END)

```

```

e
if password==password1:
messagebox.showinfo("Result", "Login Success", parent=user_login_root)
        user_login_root.destroy()
tt = ar_vein()
        tt.user_home(account_no)
else:
messagebox.showerror("Result", "Login Failed", parent=user_login_root)
        b1 = Button(user_login_root,text='Select Image',command=select_image ,font=('times', 15,
' bold '), width=20)
        canvas.create_window(300, 425, window=b1)
        b2 = Button(user_login_root, text='Next',command=next_image ,font=('times', 15, ' bold '),
width=20)
        canvas.create_window(580, 425, window=b2)
        user_login_root.mainloop()
def user_home(self,account_no):
        user_home_root =Toplevel()
        w = 950
        h = 600
ws = user_home_root.winfo_screenwidth()
hs = user_home_root.winfo_screenheight()
        x = (ws / 2) - (w / 2)
        y = (hs / 2) - (h / 2)
        user_home_root.geometry("%dx%d+%d+%d" % (w, h, x, y))
        self.bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
        user_home_root.title(self.title)
        user_home_root.resizable(False, False)
bg = ImageTk.PhotoImage(file=self.background_image)
canvas = Canvas(user_home_root, width=200, height=300)
canvas.pack(fill="both", expand=True)
        canvas.create_image(0, 0, image=bg, anchor=NW)

```

```

        canvas.create_text(470, 40, text=self.titlec, font=("Times New Roman", 24),
fill=self.text_color)
def deposit(event):
    tt = ar_vein
        tt.user_deposit(event,account_no)
image = Image.open('images/deposit.png')
img = image.resize((100, 100))
    my_img = ImageTk.PhotoImage(img)
    image_id = canvas.create_image(350, 170, image=my_img)
    canvas.tag_bind(image_id, "<1>", deposit)
    admin_id = canvas.create_text(520, 170, text="DEPOSIT", font=("Times New Roman",
15), fill=self.text_color)
        canvas.tag_bind(admin_id, "<1>", deposit)

def withdraw(event):
    tt = ar_vein
        tt.user_withdraw(event,account_no)
image2 = Image.open('images/withdrw.png')
img2 = image2.resize((100, 100))
    my_img2 = ImageTk.PhotoImage(img2)
    image_id2 = canvas.create_image(350, 270, image=my_img2)
    canvas.tag_bind(image_id2, "<1>", withdraw)

    admin_id1 = canvas.create_text(520, 270, text="WITHDRAW", font=("Times New
Roman", 15),fill=self.text_color)
        canvas.tag_bind(admin_id1, "<1>", withdraw)
#####
def transaction(event):
    tt = ar_vein
        tt.user_transaction(event,account_no)

```

```

image1 = Image.open('images/transaction.png')
img1 = image1.resize((100, 100))
my_img1 = ImageTk.PhotoImage(img1)
image_id1 = canvas.create_image(350, 370, image=my_img1)
canvas.tag_bind(image_id1, "<1>", transaction)
admin_id = canvas.create_text(520, 370, text="TRANSACTION", font=("Times New
Roman", 15), fill=self.text_color)
canvas.tag_bind(admin_id, "<1>", transaction)
# #####
def mini(event):
    tt = ar_vein
        tt.user_ministatement(event,account_no)
    admin_id1 = canvas.create_text(520, 470, text="MINI STATEMENT", font=("Times New
Roman", 15),fill=self.text_color)
    canvas.tag_bind(admin_id1, "<1>", mini)
    image3 = Image.open('images/mini.png')
    img3 = image3.resize((100, 100))
    my_img3 = ImageTk.PhotoImage(img3)
    image_id3 = canvas.create_image(350, 480, image=my_img3)
    canvas.tag_bind(image_id3, "<1>", mini)
    user_home_root.mainloop()
def user_deposit(self,account_no):
    user_deposit_root = Toplevel()
    get_data = ar_vein()
    user_deposit_root.attributes('-topmost', 'true')
    w = 550
    h = 350
ws = user_deposit_root.winfo_screenwidth()
hs = user_deposit_root.winfo_screenheight()
    x = (ws / 2) - (w / 2)
    y = (hs / 2) - (h / 2)

```

```

bal = int(data[0][0]) - int(amount)
    mm.insert_query(
        "update user_details set balance=" + str(bal) + " where account_no=" +
str(account_no) + "")
    admin_id2 = canvas.create_text(280, 70, text="TRANSACTION", font=("Times New
Roman", 24), fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(180, 140, text="Amount", font=("Times New Roman",
24), fill=get_data.get_text_color())
    admin_id2 = canvas.create_text(180, 180, text="Account No", font=("Times New Roman",
24), fill=get_data.get_text_color())
    e1 = Entry(user_transaction_root, font=('times', 15, 'bold '), width=10)
    canvas.create_window(320, 140, window=e1)
    e2 = Entry(user_transaction_root, font=('times', 15, 'bold '), width=10)
    canvas.create_window(320, 180, window=e2)
def select_image():
amount=int(e1.get())
receiver=int(e2.get())
    qry="select balance from user_details where account_no=" + str(account_no) + ""
data=mm.select_direct_query(qry)
xx=int(data[0][0])
if amount<=xx:
bal = int(data[0][0]) - int(amount)
    mm.insert_query(
        "update user_details set balance=" + str(bal) + " where account_no=" +
str(account_no) + "")
process = "Debit"
date = datetime.datetime.today()
time = date.strftime("%I:%M %p")
    mm.insert_query("insert into mini values(" + str(process) + "," + str(amount) + "," +
str(receiver) + "," + str(date) + "," + str(time) + "," + str(account_no) + ")")

```

```

#####

qry = "select balance from user_details where account_no=" + str(receiver) + ""
data = mm.select_direct_query(qry)
if len(data) > 0:
    bal = int(data[0][0]) + int(amount)

    user_transaction_root.geometry('%dx%d+%d+%d' % (w, h, x, y))
    self.bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
    user_transaction_root.title(get_data.get_title())
    user_transaction_root.resizable(False, False)
    bg = ImageTk.PhotoImage(file='images/background_hd.jpg')
    canvas = Canvas(user_transaction_root, width=200, height=300)
    canvas.pack(fill="both", expand=True)
    canvas.create_image(0, 0, image=bg, anchor=NW)
    admin_id2 = canvas.create_text(310, 70, text="MINI STATEMENT", font=("Times New
Roman", 24), fill=get_data.get_text_color())
    fram = Frame(canvas)
    fram.place(x=60, y=100, width=550, height=230)
    scrollbarx = Scrollbar(fram, orient=HORIZONTAL)
    scrollbarx.pack(side=BOTTOM, fill=X)
    scrollbary = Scrollbar(fram, orient=VERTICAL)
    scrollbary.pack(side=RIGHT, fill=Y)
    tree = Treeview(fram, columns=("process", "amount", "accnumber", "date", "time"),
yscrollcommand=scrollbary.set, xscrollcommand=scrollbarx.set)
    scrollbarx.config(command=tree.xview)
    scrollbary.config(command=tree.yview)
    tree.heading('process', text="process", anchor=W)
    tree.heading('amount', text="amount", anchor=W)
    tree.heading('accnumber', text="accnumber", anchor=W)
    tree.heading('date', text="date", anchor=W)
    tree.heading('time', text="time", anchor=W)

```

```

tree.column('#0', width=0)
tree.column('#1', width=100)
tree.column('#2', width=100)
tree.column('#3', width=100)
tree.column('#4', width=100)
tree.pack()
#Create a button to update the canvas image
button= ttk.Button(win, text="Update",
command=lambda:update_image())
button.pack()
#Open an Image in a Variable
image1 = Image.open('data/input.png')
my_img1 = ImageTk.PhotoImage(image1)

image1 = Image.open('data/greyscale.png')
my_img2 = ImageTk.PhotoImage(image1)

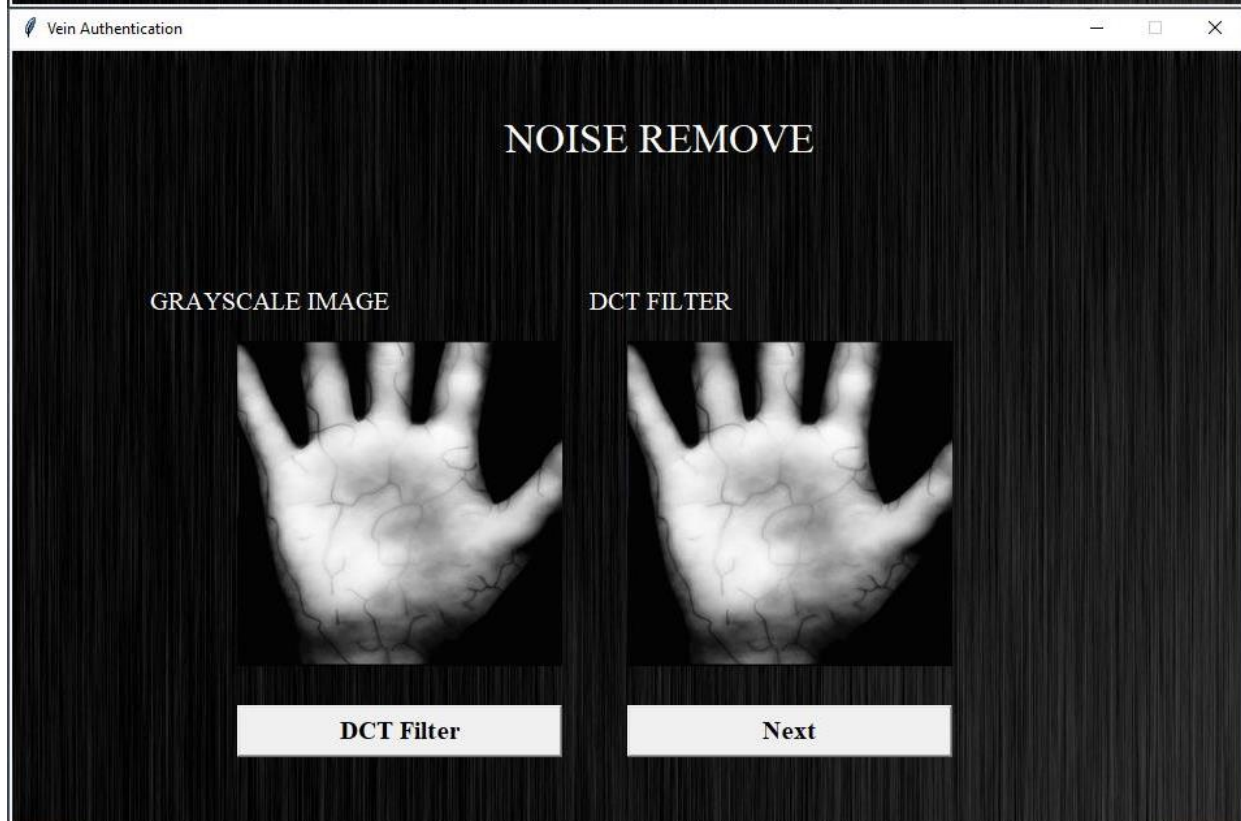
img2= PhotoImage(file="data\\greyscale.png")

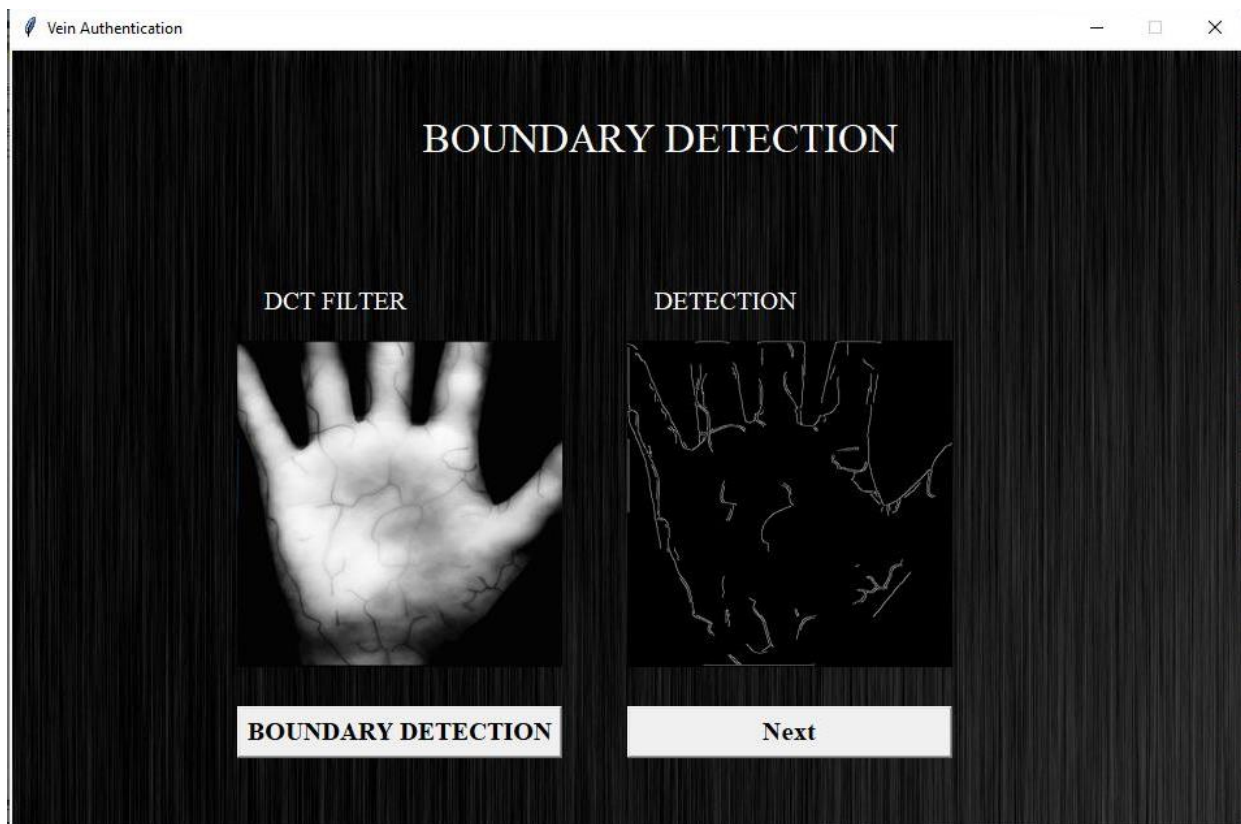
#Add image to the canvas
image_container =canvas.create_image(0,0, anchor="nw",image=my_img1)
win.mainloop()

```

6.2 SAMPLE SCREENSHOT







Vein Authentication

FEATURE EXTRACTION

Feature Value : 116125725

MORPHOLOGICAL



feature_extraction

ACCOUNT NO

Name

Contact

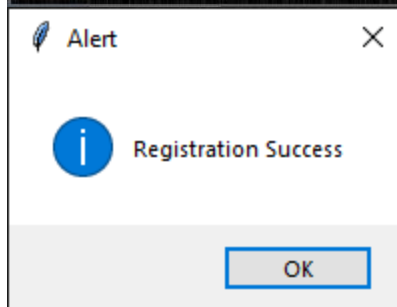
Email

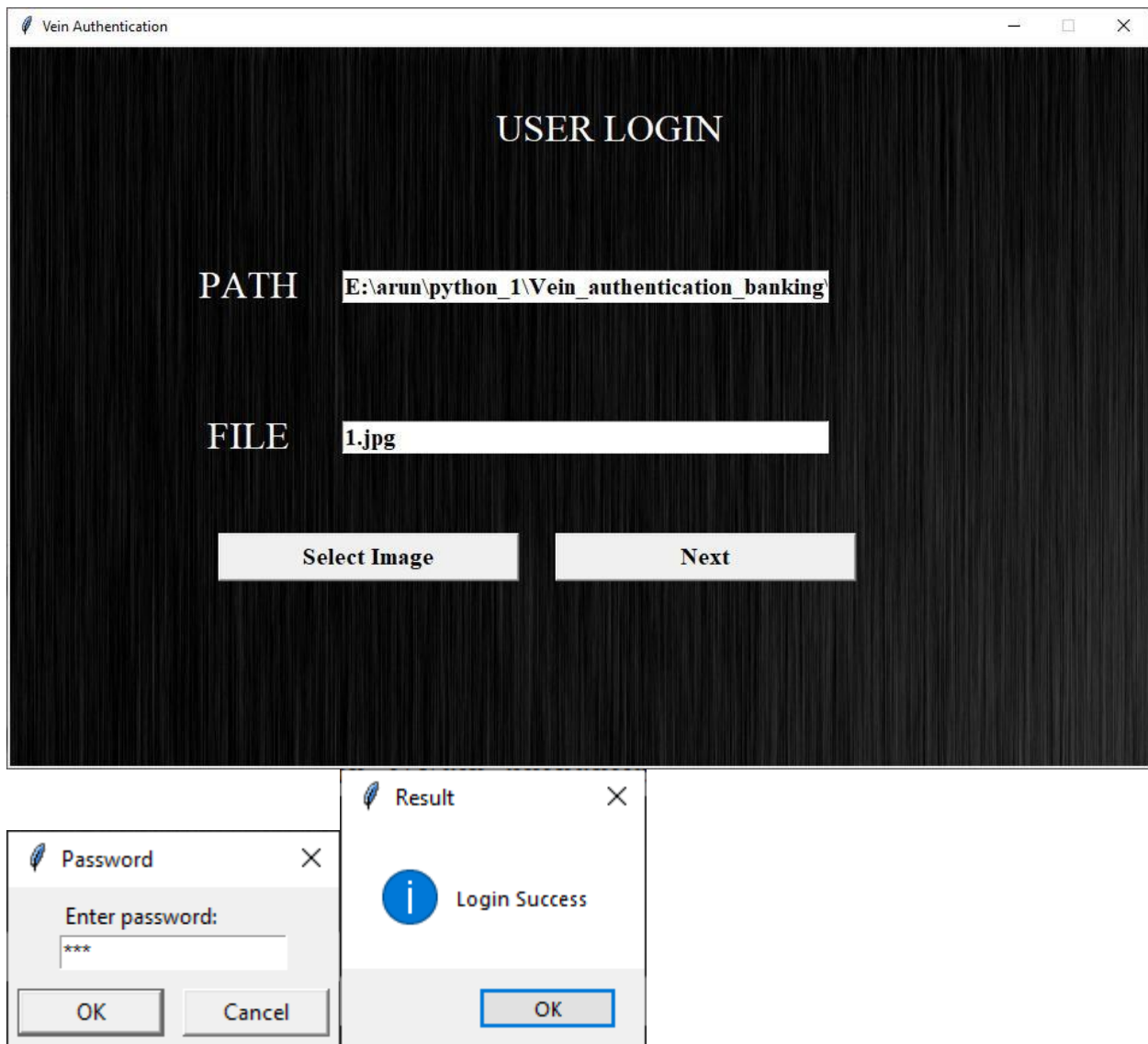
Address

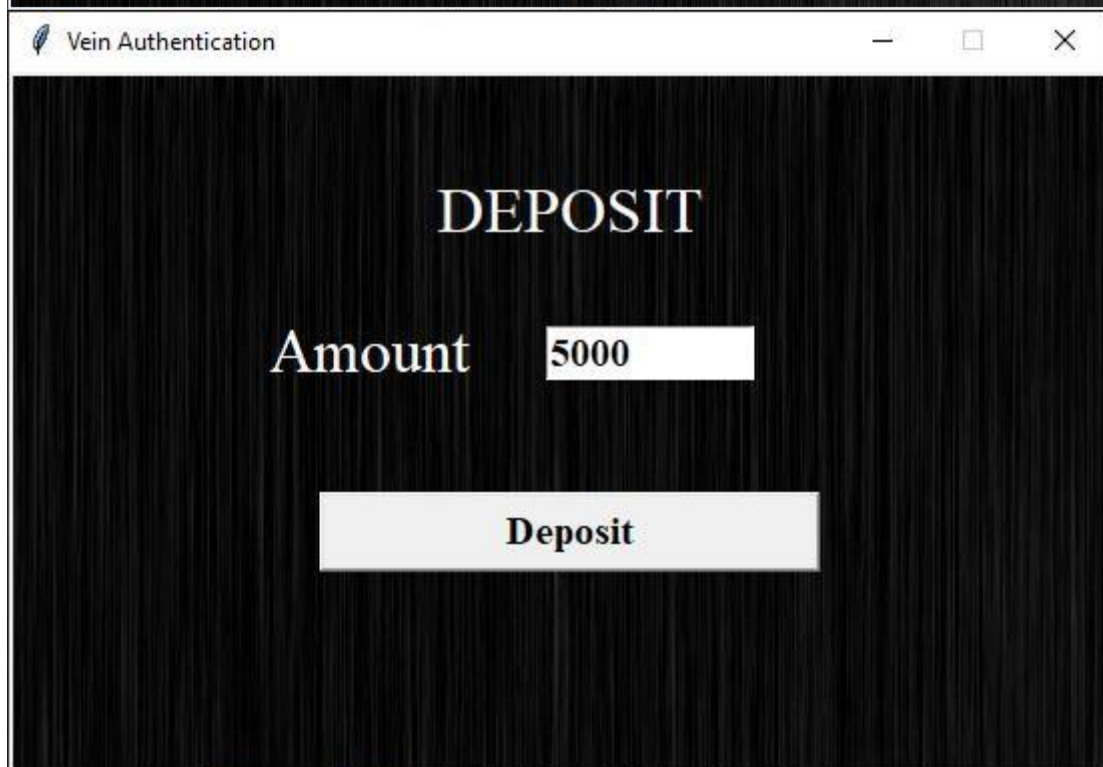
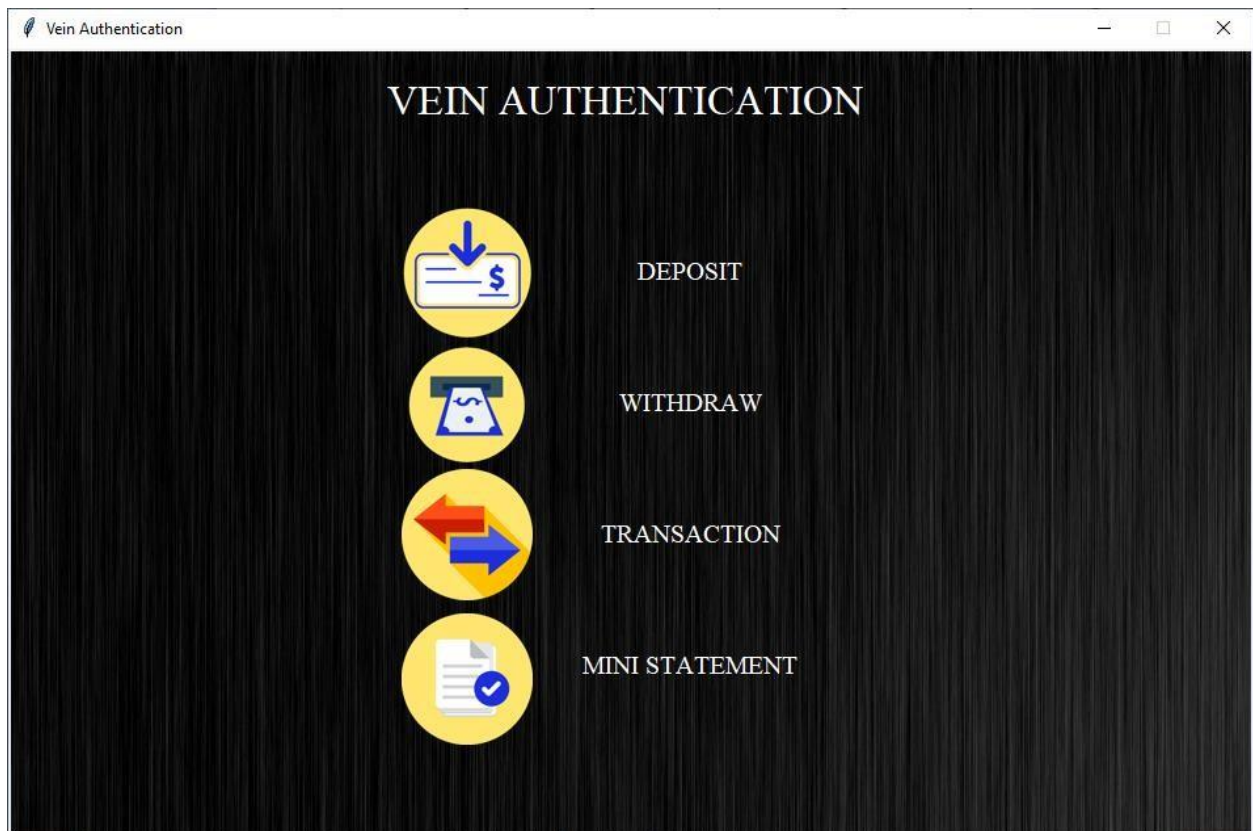
Username

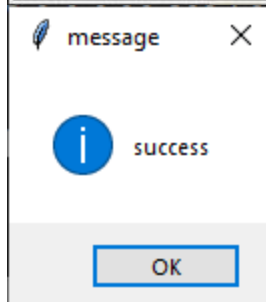
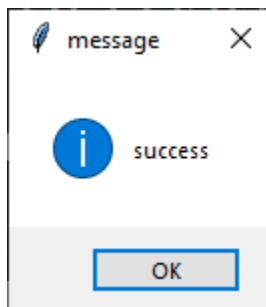
Password

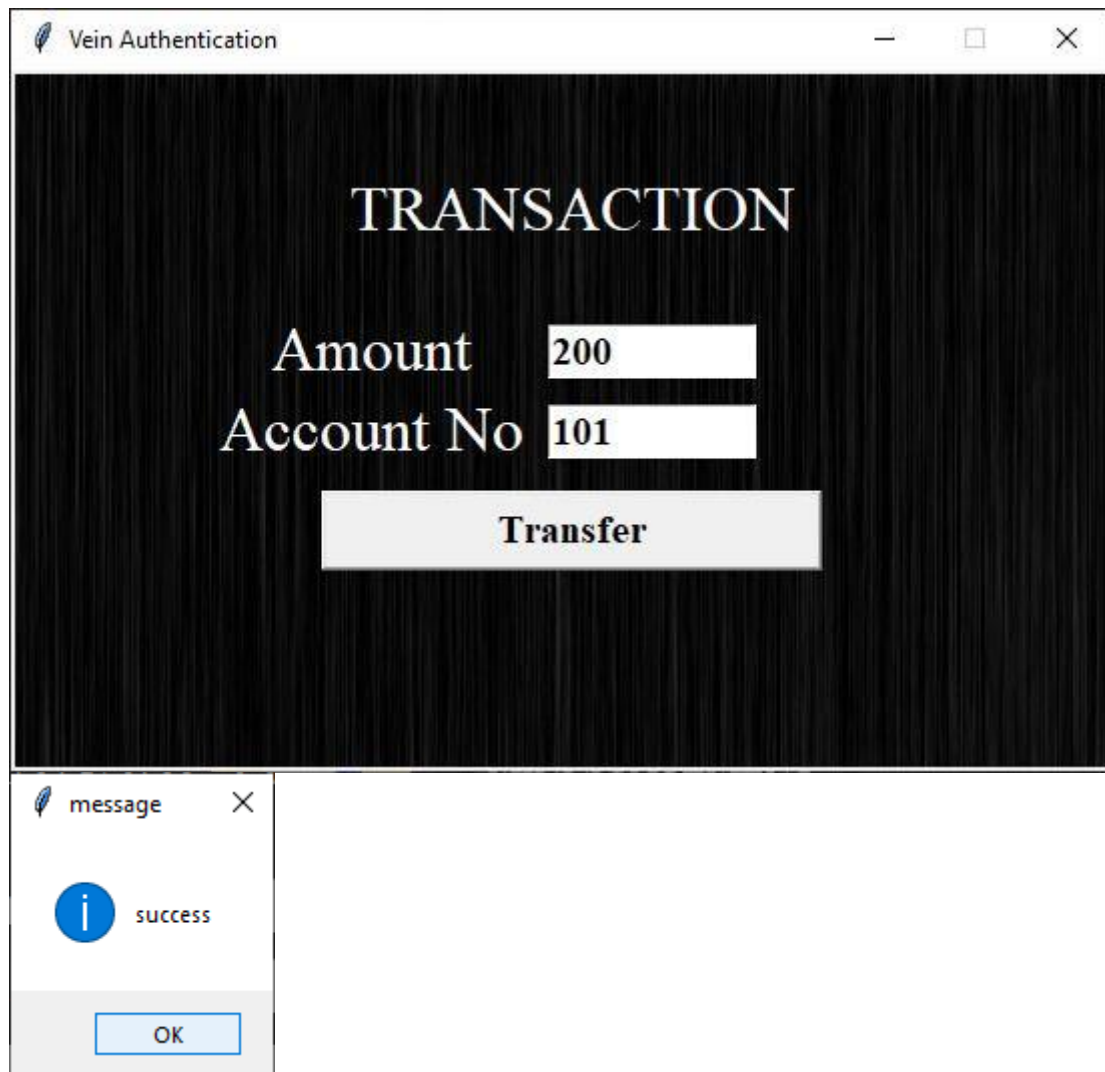
Register











Vein Authentication

MINI STATEMENT

process	amount	accnumber	date	time
Debit	200	101	2023-03-04 16:18:	04:18 PM
Withdraw	500	-	2023-03-04 16:17:	04:17 PM
Deposit	5000	-	2023-03-04 16:17:	04:17 PM

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

In previous days, ATM cards are used for money transaction. Usually people insert an ATM card and get their money. We are hearing more news that, hackers were hacking people's money. These kinds of cyber- crimes increase by these years and also by using infrared detecting technology, they detect the pin number and they steal people's money. In order to overcome this disadvantage, we developed a unique method. The system has are using palm vein detection for money transaction and also for depositing amount. As each and every user has unique palm vein, there is no chance for strangers to transact money unknowingly. This intersection spot will be taken by the system itself, taking the intersecting point as the mid-point. Thus palm vein authentication by means of this new method will ensure high level of security. The reduced feature set method proposed has high accuracy and improved performance.

7.2 FUTURE WORK

In future systems the implementation can be known with the further identification of the disease with hyper parameters. The search of the disease with the identification can be made more relevance system. Google maps can be also added to get nearby hospitals in emergency cases.

REFERENCES

1. Tiago Salgado de Magalhães Taveira-Gomes, 2017, Reinforcement Learning for primary care appointment scheduling, Faculdade de Engenharia da Universidade do Porto Mestrado de Engenharia da Informação
2. T. Cayirli and E. Veral, 2003 “Outpatient scheduling in health care: a review of literature,” *Prod. Oper. Manag.*, vol. 12, no. 4, pp. 519–549,
3. D. Gupta and B. Denton, 2008 “Appointment scheduling in health care: Challenges and opportunities,” *IIE Trans.*, vol. 40, no. 9, pp. 800–819
4. Xiaojun Zhang, 2012, Developing an online patient Appointment Scheduling system based on web services architecture, Chinese Academy of Sciences EET ALAPAMI 2012 Conference Proceedings
5. S. Sri Gowthem, & K.P. Kaliyamurthi, 2015, Smart Appointment Reservation System, *International Journal of Innovative Research in Science, Engineering and Technology*, Vol. 4, Issue .
6. Xiaojun Zhang, Dr. Ping Yu, Dr. Jun Yan, Hongxiang Hu, and Dr. Niraj Gourea, 2012,
7. Peng Zhao, Ilhoi Yoo, Jaie Lavoie, Beau James Lavoie, and Eduardo Simoes, 2017, Web-Based Medical Appointment Systems: A Systematic Review Vol 19, No 4 (2017): April
8. D van Brenk, 2016, Reducing Waiting Times In The Pre-Anaesthetic Clinic Of Vu University Medical Center, Master Thesis university of Twente