AI APPROACH TO PREDICT BLOOD TYPE USING FINGERPRINT

A PROJECT REPORT

Submitted by

ANBUSELVAM V	**721220205002**
INDHUMATHI M	**721220205014**
KARRUPASAMY B	**721220205023**
NARRENDRAN P	**721220205034**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KARPAGAM INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY:: CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"AI APPROACH TO PREDICT BLOOD GROUP USING FINGERPRINT"** is the bonafide work of **ANBUSELVAM V(721220205002), INDHUMATHI M(721220205014), KARRUPASAMY B (721220205024), NARRENDRAN P(721220205034)** who carried out the project work under my supervision.

SIGNATURE

Dr. D. Bhanu., M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Professor,

Department of Information Technology
Karpagam Institute of Technology
Coimbatore-641105

SIGNATURE

Ms. S. GOKILA M.E

SUPERVISOR

Assistant Professor,

Karpagam Institute of Technology
Department of Information Technology
Coimbatore - 641105

Submitted for the university project Viva-voce examination conducted at

Karpagam Institute of Technology, Coimbatore, on ……

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

With genuine humility, we are obediently thankful to God Almighty Without him, this work would have never been a reality.

We express our profound gratitude to our respected Chairman **Dr. R. VASANTHAKUMAR,** for giving this opportunity to pursue this course. At this pleasing moment of having successfully completed the Project work.

We wish to acknowledge sincere gratitude and heartfelt thanks to our respected Principal **Dr. P. MANIMARAN M.E., Ph.D.** for having us given the adequate support and opportunity for completing the project work successfully.

We express our deep sense of gratitude and sincere thanks to our beloved Vice Principal and Head of the Department **Dr. D. BHANU M.E., Ph.D.** who has been a spark for enlightening our knowledge.

Our profound gratitude goes to our project coordinator **Mr. K. SRIRAM KUMAR M.E., (Ph.D.**) and our project guide **Ms. S. GOKILA M.E.,** and review members and all the faculty members of the Department of Information Technology for the invaluable knowledge they have imparted on us.

We express our gratitude all the staff members of the Department of Information Technology for their support throughout the project.

Our humble gratitude and heartiest thanks goes to our family members and friends for their encouragement and support throughout the course of this project.

ABSTRACT

Fingerprints are hailed as the most reliable means of individual identification, especially in legal proceedings where fingerprint evidence holds unparalleled trustworthiness. Two crucial factors attest to the efficacy of fingerprints: firstly, the ridges formed during fetal development persistently align throughout a person's life until skin decomposition; and secondly, no two fingerprints, whether of the same individual or different individuals, are identical—they invariably vary in pattern and ridge characteristics. This distinctiveness renders fingerprints widely regarded as conclusive evidence in courtrooms. This study introduces an innovative approach for blood group identification leveraging fingerprints and cutting-edge machine learning techniques. Fingerprint patterns, renowned for their unique and enduring nature, serve as a pivotal biometric identifier. In this research, Convolutional Neural Networks (CNNs) – VGG 19(Visual Geometry Group 19), a specialized subset of advanced machine learning employed to discern intricate features from fingerprint images for predicting blood groups.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	**III**
	LIST OF FIGS	**VII**
	LIST OF ABBREVIATIONS	**VIII**
1	**INTRODUCTION**	**9**
	1.1 DEVELOPMENT OVERVIEW	9
2	**BACKGROUNG AND HISTORY**	**11**
	2.1 RELATED WORK	11
	2.1.1 MEDICAL DIAGNOSTICS THROGH FINGERPRINT ANALYSIS	11
	2.1.2 BIOMETRIC SECURITY AND PRIVACY	11
	2.1.3 POINT-OF-CARE DIAGNOSTICS	13
3	**TECHNICAL STRUCTURE**	**14**
	3.1 HARDWARE SET UP	14
	3.2 SOFTWARE DEVELOPMENT	14
	3.2.1 IMAGE PROCESSING	14
	3.2.2 BLOOD GROUP PREDICTION MODEL	15
	3.2.3 USER INTERFACE	16
	3.3 DATA COLLECTION AND ANNOTATION	16
	3.3.1 DATASET COLLECTION	16
	3.3.2 ANNOTATION PROCESS	17
4	**MODEL TRAINING AND EVALUATION**	**19**
	4.1 MODEL TRAINING	19
	4.2 VALIDATION	19
	4.2.1 TEST SET EVALUATION	20
	4.3 CNN ALGORITHM	20
	4.4 DATA PROCESSING PIPELINE	21
	4.4.1 IMAGE CAPTURE	21

	4.4.2 PREPROCESSING	21
5	**FEATURE EXTRACTION**	**22**
	5.1 BLOODGROUP PREDICTION	23
	5.2 POST-PROCESSING	25
	5.3 DEPLOYMENT AND INTEGRATION	25
	5.3.1 APPLICATION DEVELOPMENT	25
	5.4 COMPATIBILITY TESTING	30
	5.5 REAL-WORLD TESTING AND VALIDATION	31
6	**TOOLS OF IMPLEMENTATION**	**34**
	6.1 VGG 19	34
	6.2 VGG 19 AND RESNET COMPARISIONS	35
	6.3 FLASK (BACKEND)	36
	6.4 HTML, CSS (FRONTEND)	37
	6.5CONVOLUTIONAL NEURAL NETWORK (ALGORITHM)	38
7	**APPLICATION IMPLEMENTATION**	**40**
8	**SYSTEM TESTING**	**42**
	6.1 UNIT TESTING	42
9	**CONCLUSION**	**43**
	APPENDIX I SOURCE CODES	**44**
	APPENDIX II SCREENSHOTS	**54**
	APPENDIX III REFERENCES	**59**
	APPENDIX IV CONFERENCE CERTIFICATES	**60**

FIG.NO.	LIST OF FIGS	PAGE.NO.
1.1.1	Fingerprint patterns	9
3.3.1.1	Dataset collections	17
3.3.2.1	Fingerprint count and its blood type	18
4.1.1	Fingerprint collections for Model Training	19
5.1	Minutiae feature extraction of fingerprint	23
5.1.1	System Architecture	25
5.5.1	Accuracy Analysis	32
5.5.2	Loss Analysis	33
5.5.3	Accuracy Calculations	33
6.1.1	VGG 19 Architecture	36
A.2.1	Fingerprint Scanning or uploading fingerprint image	55
A.2.2	Uploading fingerprint images	55
A.2.3	Blood group analysis result for “A+’ blood type	56
A.2.4	Blood group analysis result for “AB-” blood type	56
A.2.5	Blood group analysis result for “B-” blood type	57
A.2.6	Blood group analysis result for “O-” blood type	57
A.2.7	Blood group analysis result for “A-” blood type	58
A.2.8	Blood group analysis result for “AB+” blood type	58
A.2.9	Blood group analysis result for “O+” blood type	59
A.2.10	Blood group analysis result for “B+” blood type	59

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CNN	Convolutional Neural Networks
PIN	Personnel Identification Number
PII	Personally Identifiable Number
GDPR	General Data Protection Regulation
HIPAA	Health Insurance Portability and Accountability Act
EHR	Electronic Health Records
VGG 19	Visual Geometry Group 19
JPEG	Joint Photographic Expert Graphics
PNG	Portable Network Graphics
LBP	Local Binary Patterns
HOG	Histogram of Oriented Graphics
URL	Uniform Resource Locator
SQL	Structured Query Language
HTML	Hypertext Markup Language
RESTful	Representational State Transfer
API	Application Programming Interface
UI	User Interface
PR	Pull Request
MR	Merge Request
ORM	Object Relational Mapping
HTTP	Hypertext Transport Protocol
RBAC	Role Based Access Control
AWS	Amazon Web Service
SDK	Software Development Kit
IDE	Integrated Development Environment
SGD	Stochastic Gradient Descent
CSS	Cascading Style Sheet

CHAPTER 1

INTRODUCTION

1.1 DEVELOPMENT OVERVIEW

Blood group identification is crucial in various medical settings, including blood transfusions, organ transplantation, and prenatal care. The four main blood groups—A, B, AB, and O—are determined by the presence or absence of specific antigens on the surface of red blood cells. Additionally, each blood group can be further classified based on the presence of Rh factor (positive or negative). Traditional methods of blood typing involve serological tests, which require blood samples and specialized laboratory equipment. While these methods are generally accurate, they are not always convenient, especially in emergency situations or resource-limited settings. Repeated blood sampling may pose risks to patients, such as discomfort, infection, or complications. Fingerprints are unique to each individual and remain relatively stable throughout a person's life. Advances in biometric technology have enabled the use of fingerprints for various applications, including identification, access control, and forensic analysis. By analyzing the distinctive patterns of ridges and valleys (loops, whorls, arches) present in fingerprints, it is possible to extract valuable information about an individual's identity and potentially other biological characteristics.

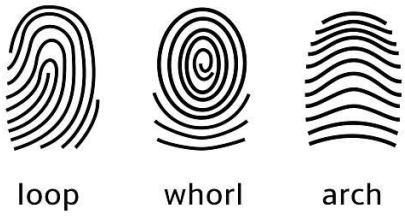

Fig 1.1.1 Fingerprint patterns

The project aims to develop an innovative approach utilizing artificial intelligence (AI) algorithms to detect an individual's blood group through fingerprint analysis. Traditional methods of blood typing often involve invasive procedures and time-consuming laboratory tests. The integration of AI-driven blood typing into existing healthcare infrastructures holds the promise of enhancing epidemiological surveillance, enabling early detection and containment of infectious diseases with blood group correlations. The integration of AI and biometric data emerges as a potent avenue, promising a revolutionary approach to blood group identification. The convergence of AI and biometric data heralds a watershed moment in the field of blood group identification, promising to transcend the limitations of traditional methodologies and redefine the contours of medical diagnostics. As this visionary project unfolds, its impact reverberates not only within the confines of clinical practice but also across the broader landscape of public health, catalyzing a paradigm shift towards more accessible, efficient, and equitable healthcare solutions. By leveraging AI and biometric data, this project seeks to provide a non-invasive, rapid, and cost-effective solution for blood group identification.

CHAPTER 2

BACKGROUND AND HISTORY

2.1 RELATED WORK

2.1.1 Medical diagnostics through fingerprint analysis

The concept of using fingerprints for blood group prediction is a relatively recent development, spurred by advancements in biometric technology and the growing capabilities of AI algorithms. Prior to the 21st century, research primarily focused on the use of fingerprints for identification purposes rather than medical diagnostics. In the early 2000s, with the advancement of biometric technologies and the increasing availability of large-scale databases, researchers began exploring the potential of biometrics for healthcare applications. Studies investigated the correlation between various biometric traits and physiological characteristics, including blood groups. During this period, a few studies started exploring the feasibility of predicting blood groups based on fingerprint patterns. Researches focused on analyzing the structural and textural features of fingerprints to identify potential correlations with blood group types. Initial results showed promising but limited success, with challenges related to accuracy and reliability. So that Researchers began applying AI algorithms, such as convolutional neural networks (CNNs), to extract complex patterns from fingerprint images and predict blood groups more accurately. Studies leveraged large datasets of fingerprint images paired with blood group information to train and validate predictive models. Advances in image processing, feature extraction, and pattern recognition techniques further improved the accuracy and reliability of blood group prediction using fingerprints.

2.1.2 Biometric security and privacy

Fingerprint biometrics can be used for secure authentication, ensuring that only authorized individuals can access blood group prediction systems or view their own blood group information. This authentication mechanism enhances

overall system security and prevents unauthorized access. Fingerprint biometrics provide a reliable means of non-repudiation, as an individual's fingerprints are inherently tied to their identity. This helps prevent impersonation or fraudulent activities in blood group prediction procedures. Fingerprint biometrics offer a high level of security due to the uniqueness of each individual's fingerprint patterns. This uniqueness enhances the accuracy of blood group prediction, as it ensures that each fingerprint can potentially serve as a distinct identifier. In some implementations, fingerprint biometrics can be combined with other authentication factors, such as passwords or PINs, to create a multi-factor authentication system. This enhances security by requiring multiple credentials for access, reducing the risk of unauthorized entry. Blood group prediction systems should only collect and store the minimum amount of fingerprint data necessary for the intended purpose. This principle of data minimization reduces the risk of privacy breaches and unauthorized access by limiting the exposure of sensitive biometric information. Individuals must provide informed consent before their fingerprint data is collected and used for blood group prediction. Transparent communication about how the data will be used, who will have access to it, and the measures taken to protect privacy helps build trust and ensures compliance with privacy regulations. Fingerprint data can be anonymized by removing personally identifiable information (PII) associated with the individual, such as names or identification numbers. Anonymization helps protect the privacy of individuals by preventing their identities from being directly linked to their fingerprint data. Blood group prediction systems must comply with relevant data protection regulations and standards, such as GDPR, HIPAA, or local privacy laws. Compliance ensures that individuals' rights regarding the collection, processing, and storage of their biometric data are respected and protected.

2.1.3 Point-of-care diagnostics

Ensuring seamless integration of the point-of-care diagnostic device with existing healthcare information systems and workflows. Enabling data exchange and interoperability to facilitate seamless communication between the device and electronic health records (EHRs) for comprehensive patient management. Implementation of robust data security measures to protect the privacy and confidentiality of patient information collected during blood group identification. Encryption of data transmission and storage, implement access controls, and adhere to data protection regulations to safeguard sensitive biometric and health data. Providing training to healthcare professionals on how to use the point-of-care diagnostic device effectively. Deploying the device in healthcare facilities, ensuring proper maintenance, calibration, and quality control procedures are followed to maintain its performance and accuracy over time. Performing clinical trials to evaluate the device's usability, acceptability, and clinical utility in real-world healthcare settings. Gathering feedback from healthcare professionals and end-users to identify any usability issues or areas for improvement. Conducting rigorous validation studies to assess the performance of the point-of-care diagnostic device. Evaluation of its accuracy, sensitivity, specificity, and reliability compared to traditional laboratory-based blood typing methods. Validation should involve diverse populations to ensure the device's effectiveness across different demographics. Integrating AI algorithms into the device's software to analyze fingerprint images and predict the blood group of the individual. These algorithms should be optimized for real-time processing and accuracy, ensuring reliable blood group identification within seconds. Continuously monitoring and evaluating the performance of the point-of-care diagnostic device in real-world settings.

CHAPTER 3
TECHNICAL STRUCTURE

3.1 Hardware setup

Fingerprint sensors are electronic devices designed to capture and digitally record an individual's fingerprint image. These sensors are commonly used for biometric authentication and security purposes. They work by scanning the unique patterns present in the ridges and valleys of a person's fingerprint and converting them into a digital representation. Sensor Surface is the area where the user places their finger for scanning. It typically consists of a thin layer of silicon or glass with built-in sensing elements. Fingerprint sensors utilize various sensing technologies such as capacitive, optical, ultrasonic, or thermal to capture the fingerprint image. Capacitive sensors, for example, detect the electrical capacitance variations caused by the ridges and valleys of the fingerprint. Once the ingerrint is scanned, the sensor's image processing module analyzes the captured data (ridges, whorls, arches) to extract unique features and create a digital template of the fingerprint.

3.2 Software Development

3.2.1 Image Processing

The process begins with capturing high-resolution fingerprint images using a digital fingerprint scanner or another imaging device capable of capturing detailed fingerprint patterns. Removing noise and artifacts from the fingerprint image to enhance clarity and improve the quality of feature extraction. Adjusting the contrast and brightness of the image to ensure consistent illumination across different fingerprint samples. Standardizing the size and orientation of fingerprint images to facilitate feature extraction and comparison across different samples. Applying post-processing techniques to refine the predicted blood type results, such as filtering out unlikely predictions or correcting misclassifications. Also it

includes presenting the final blood type predictions alongside the corresponding fingerprint images in a user-friendly format for interpretation by healthcare professionals or end-users.

3.2.2 Blood group prediction model

VGG-19 is a deep convolutional neural network (CNN) architecture consisting of 19 layers, including 16 convolutional layers and 3 fully connected layers. The convolutional layers extract hierarchical features from the input images, while the fully connected layers perform classification based on these features. Preprocess the fingerprint images to ensure uniform size, orientation, and quality. Applying techniques such as normalization, resizing, and noise reduction to standardize the images for input to the VGG-19 model. Fine-tuning the pre-trained model on the fingerprint dataset by adjusting the weights during training to adapt the model to the specific task of blood group prediction.

VGG-19 has been pretrained on large-scale image datasets like ImageNet, which means the lower layers have already learned general features useful for various image recognition tasks. With its capability to discern images across 1000 distinct object categories, VGG19 demonstrates proficiency in classifying various entities ranging from animals to diverse objects and beyond.

Transfer learning allows leveraging these pre learned features and fine-tuning the network on a smaller dataset of fingerprint images for blood group prediction. This can expedite training and improve performance, especially with limited data availability. VGG-19, renowned for its robustness, consistently yields precise outcomes across diverse datasets. Through meticulous training on ImageNet and its uniform architecture with 3x3 convolutions, it excels in minimizing discrepancies and false identifications, making it a preferred choice for demanding image classification tasks. Its simplicity and uniformity contribute to its ease of comprehension and implementation while still achieving commendable performance in image classification endeavors

3.2.3 User Interface

Home screen displays a welcoming message that introduces the user to the application's purpose. Clear instructions on how to proceed with fingerprint scanning for blood group prediction are given. A prominent button to initiate the blood group prediction process is placed. A designated area where users can place their finger for scanning. A Scan button to start the fingerprint scanning process. A progress Indicator is given to provide visual feedback indicating the progress of the scanning process. A notification message informing the user to wait while the system processes the fingerprint is given.

3.3 DATA COLLECTION AND ANNOTATION

4.4.1 Dataset Collection

The loop fingerprint pattern emerged as the most prevalent, constituting 49.4% of cases, followed closely by the whorl pattern at 44.9%, while the arch pattern was least common at 5.7%. An in-depth analysis revealed a statistically significant association ($p < 0.001$) between gender and fingerprint pattern. Specifically, the loop pattern predominated among females (54.6%), whereas males exhibited a higher prevalence of the whorl pattern (50.0%). Moreover, the whorl pattern was found to be most frequent among individuals with AB+ and O- blood groups, while the loop pattern prevailed among those with A+, A-, B+, B-, and O+ blood groups. Utilizing the Chi-square test, our analysis underscored a significant correlation between different fingerprint patterns and the blood groups of the subjects ($p < 0.001$). To elucidate these findings, we delved into the intricate relationship between fingerprint patterns and various demographic and physiological factors. Our comprehensive investigation encompassed a thorough examination of the implications of fingerprint patterns on individual characteristics such as age, ethnicity, and medical history. Additionally, we explored the potential implications of genetic predispositions and environmental factors on the formation and prevalence of specific fingerprint patterns.

Furthermore, our study employed advanced statistical methodologies to discern nuanced trends and patterns within the dataset. Through rigorous regression analyses and multivariate modeling techniques, we sought to uncover hidden associations and elucidate the underlying mechanisms driving the observed correlations. By conducting stratified analyses and subgroup comparisons, we aimed to provide a nuanced understanding of the complex interplay between fingerprint patterns and demographic variables. We embarked on qualitative inquiries to glean insights from the subjective experiences and perspectives of individuals with diverse fingerprint patterns. Leveraging advancements in biometric authentication systems, machine learning algorithms, and image processing techniques, we sought to enhance the accuracy and efficiency of fingerprint identification methods.

			Blood group							
			A−	A+	AB+	B−	B+	O−	O+	Total
Fingerprint Pattern	Arch	Count	0	33	1	0	15	10	55	114
		% within blood group	0.0%	28.9%	0.9%	0.0%	13.2%	8.8%	48.2%	100.0%
	Loop	Count	12	181	35	11	194	16	539	988
		% within blood group	1.2%	18.3%	3.5%	1.1%	19.6%	1.6%	54.6%	100.0%
	Whorl	Count	8	166	54	9	151	24	486	898
		% within blood group	0.9%	18.5%	6.0%	1.0%	16.8%	2.7%	54.1%	100.0%
Total		Count	20	380	90	20	360	50	1,080	2,000
		% within blood group	1.0%	19.0%	4.5%	1.0%	18.0%	2.5%	54.0%	100.0%

Fig 3.3.1.1 Dataset collections

3.3.2 Annotation process

This project uses Convolutional Neural Networks (CNNs) and VGG 19 model to predict blood types from fingerprint images, achieving high accuracy. It involves nearly 6000 samples and highlights the potential of CNN models in biometric identification tasks. It focuses on reducing human error in blood group detection through image processing techniques, improving efficiency and accuracy in medical diagnostics. Investigates the relationship between blood types and fingerprint patterns and finds correlations between specific blood groups and fingerprint patterns. Evaluates the relationship between gender, blood groups, and fingerprint patterns.

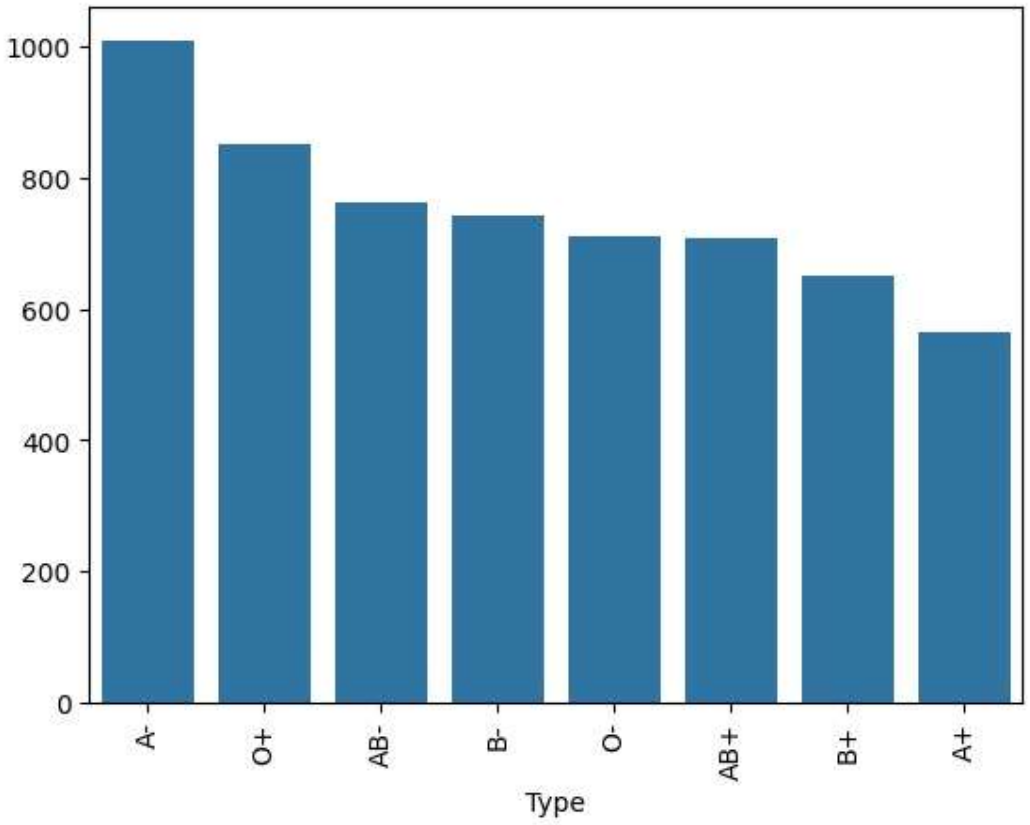

Fig: 3.3.2.1 Fingerprint count and its blood type

CHAPTER 4

MODEL TRAINING AND EVALUATION

4.1 MODEL TRAINING

VGG-19 model trained on the training data set using the techniques stochastic gradient descent (SGD). The performance of the model monitored on the validation set to prevent overfitting and adjust hyperparameters accordingly. Iteration of the training process until satisfactory performance is achieved (80%)

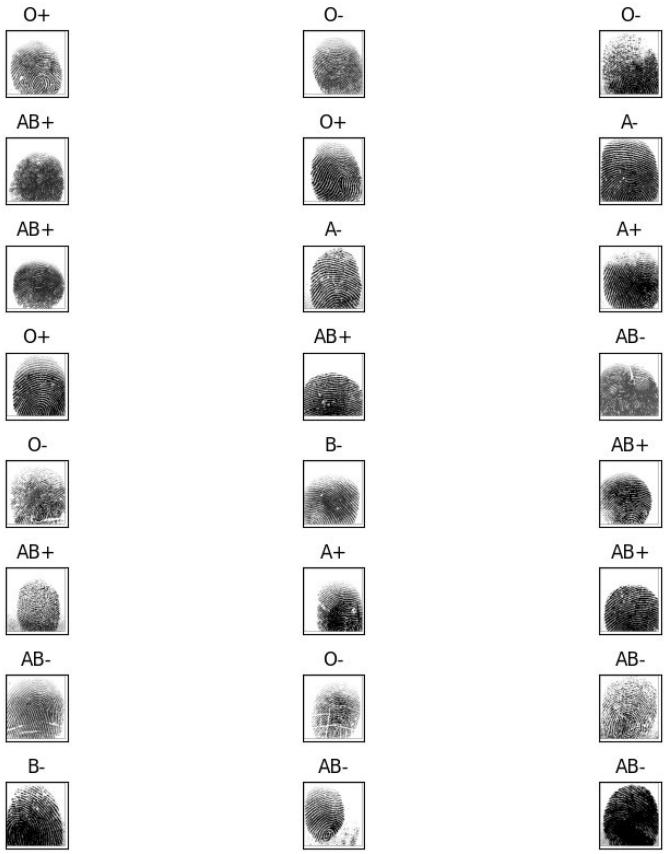

Fig:4.1.1 Fingerprint collections for Model Training

4.2 Validation

Performed cross-validation using nearly 100 samples to validate the model's performance across multiple folds of the dataset. This ensures the consistency in the performance of the model and independency on the particular partitioning of the data.

4.2.1 Test Set Evaluation

Evaluation of the trained model on the test set to assess its performance in detecting blood groups from fingerprint images. Calculation of evaluation metrics such as accuracy, precision, recall, and F1-score to quantify the model's performance. Visualization of the model's predictions and analyzing any misclassifications to identify areas for improvement. Done fine-tune of the model further based on the evaluation results and domain-specific insights. Explored various techniques such as data augmentation, regularization, and hyperparameter tuning to improve the model's performance. Optimization of the model for deployment on resource-constrained devices, if necessary, by reducing its size and computational complexity.

4.3 CNN Algorithm

Convolutional Neural Networks (CNNs) are a class of deep learning algorithms specifically designed for processing structured grid-like data, such as images. CNNs comprise multiple layers of convolutional filters that extract features from input images. Each filter performs a convolution operation over the input image, producing feature maps that highlight different patterns or textures**.** Pooling layers down sample the feature maps obtained from convolutional layers, reducing their spatial dimensions while preserving important features. Common pooling operations include max pooling and average pooling, which retain the maximum or average value within each pooling window respectively. CNNs exploit parameter sharing to reduce the number of learnable parameters and improve computational efficiency. In convolutional layers, the same set of filter weights is applied across different spatial locations of the input image, allowing the network to capture spatial hierarchies of features. CNNs learn hierarchical representations of features through multiple layers of abstraction. Lower layers typically capture low-level features such as edges and textures, while higher layers extract more complex and abstract features relevant to the task. CNN

architectures often include one or more fully connected layers at the end of the network, which combine the extracted features from convolutional and pooling layers to perform classification or regression tasks. These layers connect every neuron in one layer to every neuron in the subsequent layer. CNNs trained on large-scale datasets can be leveraged for transfer learning. Pre-trained CNN models can be fine-tuned on smaller datasets or specific tasks, allowing them to adapt to new domains with limited labeled data.

4.4 Data Processing Pipeline

A data processing pipeline is a series of steps or stages through which raw data is transformed, cleaned, and manipulated to extract valuable insights or prepare it for further analysis.

4.4.1 Image Capture

Place the finger firmly on the surface of the fingerprint scanner ensuring that the entire fingerprint area is in contact with the scanner. Maintain a steady position and avoid excessive movement during the image capture process. Activation of the fingerprint scanner or imaging device to initiate the image capture process. The scanner will illuminate the finger with light and capture multiple images of the fingerprint from different angles to ensure comprehensive coverage.

4.4.2 Preprocessing

After capturing the fingerprint images, performing a quality assessment to ensure that the images are clear, well-defined, and free from artifacts or distortions. Checking for factors such as image resolution, contrast, sharpness, and overall visibility of fingerprint ridges and valleys. Save the captured fingerprint images in a secure and standardized format, such as JPEG or PNG, for further processing and analysis. Organizing the images into labeled datasets, associating each image with the corresponding subject's information.

CHAPTER 5

FEATURE EXTRACTION

Feature extraction plays a crucial role in capturing the distinctive characteristics of fingerprints that correlate with different blood groups. Features based on the ridge patterns present in the fingerprint images. These features can include ridge density, ridge width, ridge count, ridge curvature, and ridge orientation. Ridge density refers to the number of ridges per unit area and can be computed by counting the number of ridge pixels within a predefined region. Ridge width provides information about the thickness of the ridges and can be calculated by measuring the width of ridges at different points along their length. Ridge count represents the number of ridges intersecting a line of a specific length and orientation and can be used to characterize the fingerprint pattern. Extraction of features based on the spatial distribution and characteristics of minutiae points. Utilize the orientation field of the fingerprint image, which represents the local ridge orientations at each pixel. Extract features based on the orientation field, such as dominant orientation angles, coherence of ridge orientations, and orientation entropy. Orientation angles identify the predominant ridge orientations within the fingerprint and can be computed using techniques like gradient-based orientation estimation. Coherence of ridge orientations measures the consistency or alignment of ridge orientations across the fingerprint image. Orientation entropy quantifies the uniformity or randomness of ridge orientations, providing information about the complexity of the fingerprint pattern. LBP(Local Binary Patterns) computes binary patterns based on the intensity variations within local image neighborhoods and can be used to characterize the texture of fingerprint ridges. Gabor filters extract features based on the responses of a set of Gabor kernels tuned to different frequencies and orientations, capturing texture information at multiple scales. HOG (Histogram of Oriented Gradients) computes histograms of gradient orientations within local image regions and has been widely used for texture analysis in fingerprint recognition. These features provide

complementary information to ridge- and minutiae-based features and can help differentiate between different blood groups based on subtle variations in fingerprint characteristics.

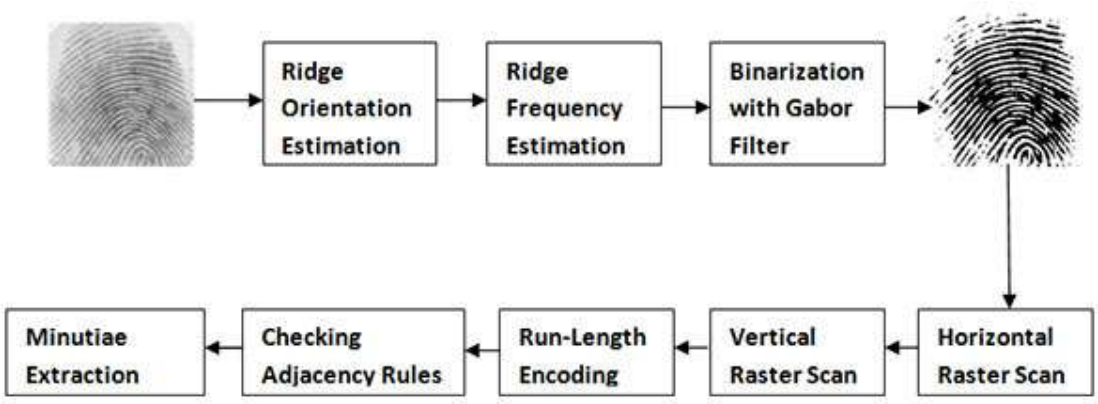

Fig 5.1 Minutiae feature extraction of fingerprint

5.1 Blood Group Prediction

The VGG-19 model, which has been trained on a large dataset of fingerprint images and blood group labels is integrated into the mobile application. This involves converting the trained model into a format suitable for deployment on mobile devices. The mobile application provides a user-friendly interface for individuals to scan their fingerprints using the device's built-in fingerprint sensor or camera. Users are prompted to place their finger on the sensor or capture a photo of their fingerprint. The captured fingerprint image is processed within the mobile application to enhance its quality and extract relevant features. This may involve techniques such as image normalization, noise reduction, and feature extraction. The preprocessed fingerprint image is fed into the integrated VGG-19 model for blood group prediction. In the mobile application designed for blood group prediction using fingerprints, the integrated VGG-19 model undertakes a pivotal role in analyzing and interpreting the unique features extracted from the fingerprint images. Once a user submits their fingerprint through the application's interface, the image undergoes a series of

preprocessing steps to optimize its quality and ensure consistency. These steps encompass techniques like normalization and noise reduction, aimed at refining the fingerprint image to a standardized format suitable for analysis. Subsequently, the preprocessed fingerprint image is fed into the VGG-19 model, where a sophisticated network of convolutional layers diligently scrutinizes the intricate patterns and structures inherent to the fingerprint. Through a process of feature extraction and hierarchical representation learning, the model discerns subtle nuances and discriminative characteristics embedded within the fingerprint. Leveraging its learned patterns and associations gleaned from extensive training on a diverse dataset, the VGG-19 model generates predictions regarding the most likely blood group corresponding to the input fingerprint. Upon completion of the prediction process, the mobile application promptly presents the outcome to the user within its intuitive interface. Users receive immediate feedback regarding their predicted blood group, empowering them with valuable health-related information at their fingertips. This seamless interaction between the user and the application underscores the accessibility and convenience afforded by the integration of AI technology into mobile devices. The model analyzes the features extracted from the fingerprint image and predicts the most likely blood group based on learned patterns and associations. The preprocessed fingerprint image is fed into the integrated VGG-19 model for blood group prediction. The model analyzes the features extracted from the fingerprint image and predicts the most likely blood group based on learned patterns and associations. The predicted blood group is displayed to the user within the mobile application interface. The predictive capabilities of the VGG-19 model serve as a testament to the advancements in artificial intelligence and its potential to revolutionize healthcare practices. The VGG-19 model operates autonomously within the mobile application, it does so with utmost privacy and security considerations. Stringent measures are in place to safeguard users' sensitive data, ensuring that their privacy remains paramount throughout the blood group prediction process.

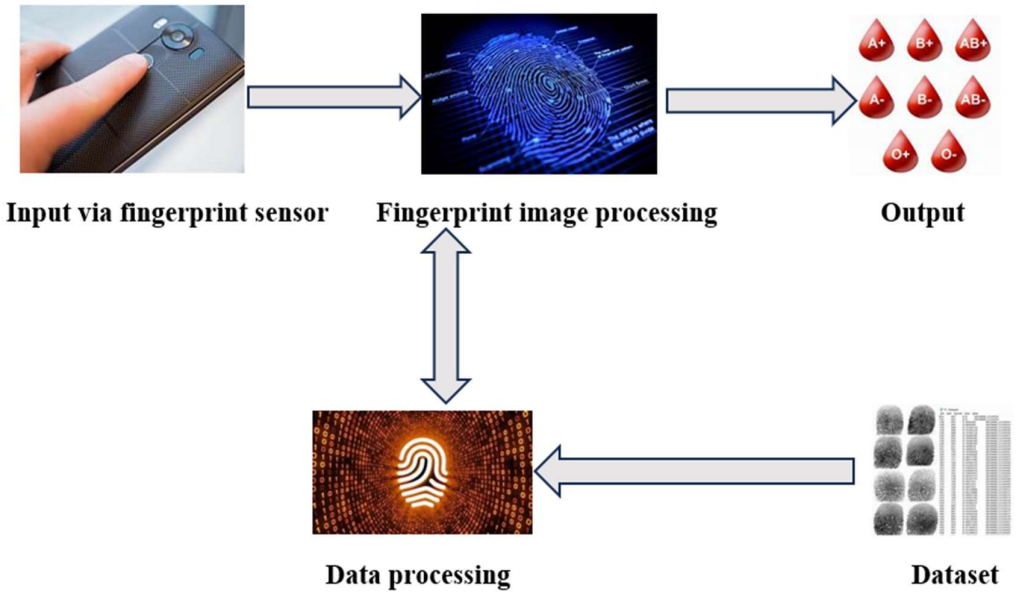

Fig 5.1.1 System Architecture

5.2 Post Processing

The mobile application includes a feedback mechanism where users can provide feedback on the accuracy of the predictions. This feedback can be used to continuously improve the model's performance through iterative updates and refinements. The application includes error handling mechanisms to handle unexpected scenarios or failures during the blood group prediction process. This involves displaying informative error messages to users and providing guidance on troubleshooting steps. Ensures the mobile application adheres to data privacy and security standards to protect users' sensitive information, such as fingerprint images and blood group prediction.

5.3 Deployment and Integration

5.3.1 Application Development

The project combines AI algorithms with mobile technology to provide users with a convenient and efficient method for determining their blood types. The fusion of artificial intelligence (AI) and mobile applications has paved the

way for innovative solutions. Our project focuses on developing a mobile application that utilizes AI algorithms to analyze fingerprint images captured by mobile devices and predict the user's blood type. This application aims to provide users with quick and accurate blood type identification, enhancing healthcare accessibility and convenience. By using Flask as the framework, Flutter for the frontend, and Python for the backend, our mobile application delivers a seamless and intuitive user experience, supported by robust data processing and machine learning algorithms. With SQL serving as the database and Git facilitating version control, prioritizes data integrity, scalability, and collaborative development to ensure the success and sustainability of our endeavor.

Framework: Flask

Flask provides a lightweight and flexible framework for building web applications, making it an ideal choice for our backend development. Its simplicity and extensibility allow for rapid prototyping and easy integration with other technologies. Flask provides a flexible and intuitive mechanism for defining URL routes and mapping them to specific functions or views. This URL routing mechanism enables developers to define the endpoints for different functionalities of the blood type detection application, such as user authentication, fingerprint image upload, blood type prediction, and result retrieval. Flask is highly modular, allowing developers to choose and integrate third-party extensions and libraries as needed for specific functionalities. Although the frontend of the blood type detection application is implemented using Flutter, Flask's template rendering capabilities can be leveraged for rendering dynamic HTML content in scenarios where server-side rendering is required. Flask follows a minimalist design philosophy, providing only the essential components needed to build web applications. Flask seamlessly integrates with the broader Python ecosystem, allowing developers to leverage a rich ecosystem of libraries and frameworks for various tasks, including data processing, machine learning, and image analysis. This integration enables the blood type detection application to utilize Python-

based AI algorithms for fingerprint analysis and blood type prediction. Its modular architecture promotes code reusability, maintainability, and scalability, enabling developers to build custom solutions tailored to their requirements. Flask's ability to design RESTful APIs enables seamless communication between the frontend Flutter application and the backend Python code responsible for blood type prediction and fingerprint analysis. Flask's lightweight nature ensures that the backend remains efficient and responsive, even when handling complex computations or processing large volumes of data.

Front End: HTML, CSS

HTML (Hypertext Markup Language) is used to create the structure of the web application. It defines the layout and components of the user interface. Semantic HTML tags are utilized to enhance accessibility and search engine optimization (SEO). For instance, <header>, <nav>, <section>, <footer>, etc., will be employed appropriately. HTML forms enable users to upload fingerprint images for analysis. Input fields for personal information such as name and age also included. CSS is used to style buttons for better visual appeal and user interaction. Hover effects, transitions, and animations are added to enhance the user experience. CSS media queries are used to ensure that the web application is responsive and adapts to different screen sizes and devices. CSS (Cascading Style Sheets) is utilized to style the elements created with HTML. This includes defining colors, fonts, margins, padding, borders, and other visual properties. CSS is used to add background images, gradients, and patterns to enhance the aesthetic appeal of the web application.

Back End: Python

Python serves as the backbone of our application's back end, powering the data processing, machine learning, and API development. Python's clean and readable syntax makes it well-suited for developing complex algorithms and logic. It’s simple and concise syntax allows developers to express ideas and concepts more clearly, facilitating collaboration and code maintainability. Python

libraries such as TensorFlow, PyTorch, OpenCV, and scikit-learn are instrumental in implementing fingerprint analysis, AI algorithms, and data manipulation tasks. Python's versatility extends to data processing and analysis tasks. The libraries in the python enables to preprocess fingerprint images, extract relevant features, and perform statistical analysis on blood type prediction results. Its versatility, readability, and extensive libraries make it well-suited for implementing complex algorithms and business logic. Python's seamless integration with external tools and technologies simplifies the development process. Python scripts can interface with fingerprint sensor APIs, external databases, and cloud services, enabling smooth data exchange and interoperability.

Version Control: Git

Git enables the creation of a centralized repository to store the project's source code, documentation, and other development assets. This repository serves as a single source of truth for all project collaborators, ensuring consistency and synchronization across distributed teams. Git facilitates collaborative development and version control, enabling seamless coordination among team members and efficient management of codebase changes. Git provides a robust set of features that facilitate collaborative development, code management, and version tracking. Its branching and merging capabilities streamline the development workflow and ensure code quality and stability. Git's branching and merging capabilities allow developers to work on separate features or tasks in isolation without affecting the main codebase. Branches can be created, switched between, and merged back into the main branch (e.g., master) when the changes are ready for integration. This promotes parallel development, facilitates code review, and minimizes conflicts. Git maintains a comprehensive history of all changes made to the codebase through commits. Each commit represents a snapshot of the project at a specific point in time, including the author, timestamp, and a descriptive message. The commit history provides valuable context, traceability, and accountability, enabling developers to track the evolution of the

project and understand the rationale behind each change. Git supports code review workflows by allowing developers to share their changes with team members for review and feedback. Pull requests (PRs) or merge requests (MRs) can be created to propose changes, solicit feedback, and ensure code quality before merging into the main branch. Reviewers can provide comments, suggestions, and approvals directly within the Git platform, streamlining the review process and promoting collaboration. In collaborative development environments, conflicts may arise when multiple developers make conflicting changes to the same file or code segment. Git provides tools and mechanisms for resolving conflicts gracefully, allowing developers to reconcile differences, choose between conflicting changes, and merge divergent code paths. By facilitating conflict resolution, Git helps maintain code integrity and prevent data loss during collaborative development. Git allows developers to create lightweight or annotated tags to mark significant milestones, releases, or versions of the project. Tags provide a convenient way to reference specific points in the commit history and denote stable or noteworthy releases. By tagging releases, Git facilitates version management, release tracking, and dependency management in the project. Git facilitates remote collaboration by providing mechanisms for sharing, cloning, and synchronizing repositories across distributed teams. Developers can work on their local copies of the repository, make changes offline, and synchronize their work with the central repository using push and pull operations.

Data Processing:

- The model is trained on large datasets of fingerprint images annotated with blood types, ensuring robust performance and generalization.
- Data preprocessing includes organizing and formatting the fingerprint dataset for training, validation, and testing purposes.
- Advanced image processing techniques extract key features from the fingerprint images, including ridge patterns, minutiae points, and orientation fields.

Machine Learning:

- Confidence thresholding, error handling, and user feedback mechanisms enhances the reliability and usability of the blood type prediction process.
- The VGG-19 model is utilized for blood type prediction, leveraging its deep convolutional layers and hierarchical feature learning capabilities.
- Through careful hyperparameter optimization, the model's performance can be optimized, leading to improved accuracy and convergence speed during training.

5.4 Compatability Testing

Compatibility testing is crucial to ensure a consistent and reliable user experience across a wide range of platforms and configurations.

Browser Compatibility:

Verification of the web-based components render correctly and function as intended across different browsers and versions, addressing any cross-browser inconsistencies or issues.

Fingerprint Sensor Compatibility:

Ensuring the integration with the mobile device's fingerprint sensor to ensure compatibility with different sensor models, manufacturers, and hardware specifications. Ensuring that the application accurately captures fingerprint images, processes them for blood type identification, and provides reliable results across a variety of fingerprint sensor implementations.

Performance Compatibility:

Evaluation of the application's performance under varying load conditions, including peak usage scenarios and concurrent user interactions. Ensuring that the application produces results with higher accuracy and minimalization of false predictions.

5.5 Real World Testing and Validation

This process aims to validate the effectiveness of the application in practical use cases and ensure that it meets the requirements and expectations of end-users.

Field Testing:

Engage healthcare professionals, patients, and other stakeholders to participate in field tests, providing feedback on the application's functionality, usability, and effectiveness in blood type identification.

Accuracy Validation:

Validation of the accuracy of blood type predictions generated by the application through comparative analysis with established blood typing methods, such as laboratory tests or medical records. Collecting fingerprint samples from a representative population and compare the application's predictions against known blood types to assess its accuracy and reliability. Calculation of performance metrics, including sensitivity, specificity, positive predictive value, and negative predictive value, to quantitatively evaluate the application's performance.

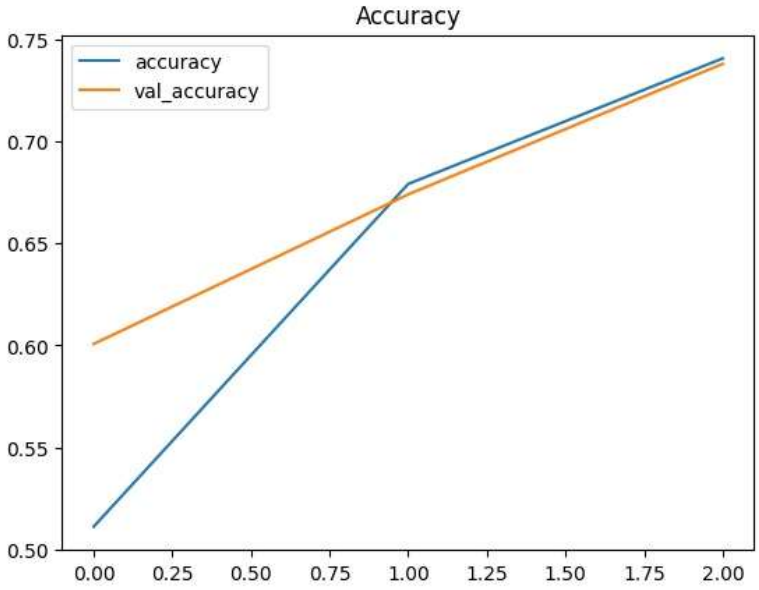

Fig: 5.5.1 Accuracy Analysis

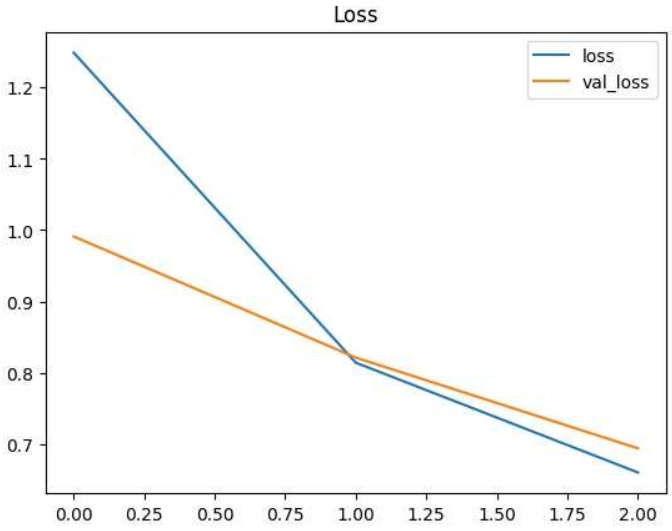

Fig:5.5.2 Loss Analysis

Performance Testing:

Evaluate the application's performance under real-world usage scenarios, including peak load conditions, concurrent user interactions, and network fluctuations. Measure response times, latency, and throughput to assess the application's scalability, reliability, and responsiveness. Identify performance bottlenecks, resource constraints, and optimization opportunities to enhance the application's performance and stability.

	precision	recall	f1-score	support
A+	0.91	0.80	0.85	138
A-	0.72	0.84	0.77	258
AB+	0.67	0.86	0.75	195
AB-	0.90	0.43	0.59	189
B+	0.63	0.82	0.71	160
B-	0.95	0.75	0.84	186
O+	0.88	0.59	0.71	213
O-	0.56	0.82	0.67	161
accuracy			0.74	1500
macro avg	0.78	0.74	0.74	1500
weighted avg	0.78	0.74	0.74	1500

Fig: 5.5.3 Accuracy Calculations

Usability Testing:

Assessed the application's usability by observing users as they perform common tasks, such as capturing fingerprint images, initiating blood type predictions, and interpreting results. Evaluated the intuitiveness of the user interface, clarity of instructions, and ease of navigation to determine the application's overall usability. Incorporated user feedback and usability metrics, such as task completion rates and error rates, to iteratively refine the user experience and interface design.

CHAPTER 6
TOOLS AND IMPLEMENTATION

6.1 VGG 19

VGG-19 is employed to extract high-level features from fingerprint images captured by the mobile application. The deep convolutional layers of VGG-19 automatically learn hierarchical representations of features, capturing patterns and structures present in the fingerprint images. By passing fingerprint images through the layers of VGG-19, meaningful feature representations are obtained, which serve as input to the blood type prediction model. VGG-19 acts as a feature extractor, leveraging its pre-trained weights and architecture to learn generic feature representations from a large corpus of images. Through the process of representation learning, VGG-19 learns to encode relevant visual information from fingerprint images into a compact and informative feature space. The learned representations capture discriminative characteristics of fingerprint patterns that are indicative of blood types, facilitating accurate prediction. Transfer learning is utilized to fine-tune the pre-trained VGG-19 model on the specific task of blood type prediction using fingerprint images. By fine-tuning the model's parameters on a task-specific dataset, VGG-19 adapts its learned features to better discriminate between different blood types based on fingerprint characteristics. Transfer learning enables the project to leverage the knowledge and expertise encoded in VGG-19's pre-trained weights while customizing the model for the specific requirements of blood type detection. The output of VGG-19's feature extraction layers serves as input to the blood type prediction model, which is typically implemented using fully connected layers and softmax activation. The extracted features are transformed into a probability distribution over the different blood types, allowing the model to predict the most likely blood type associated with the input fingerprint image. By combining the feature extraction capabilities of VGG-19 with the predictive power of the blood type prediction model, the project achieves accurate and reliable blood type identification from fingerprints.

VGG-19's performance in feature extraction and blood type prediction is evaluated using validation datasets, assessing metrics such as accuracy, precision, recall, and F1-score. Optimization techniques, such as fine-tuning hyperparameters and adjusting model architecture, may be applied to enhance VGG-19's performance and generalization ability on the blood type prediction task. The project iteratively evaluates and refines VGG-19's functionalities to achieve optimal performance and robustness in real-world scenarios.

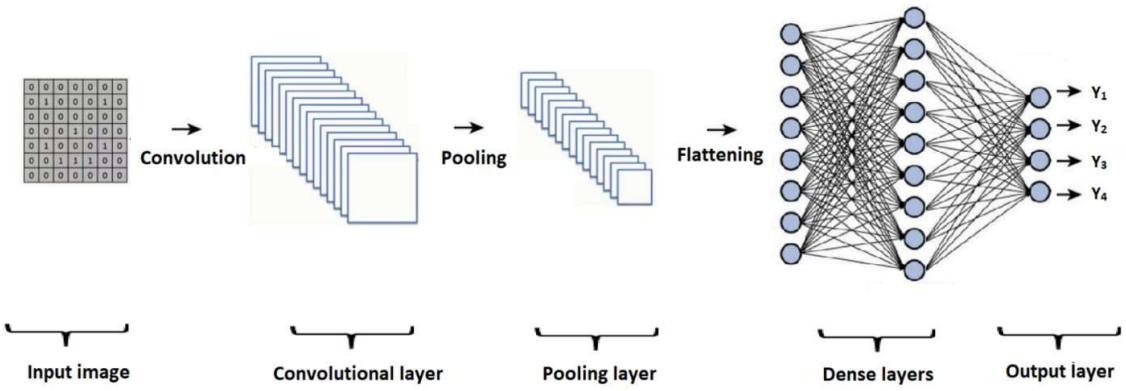

Fig: 6.1.1 VGG 19 Architecture

6.2 VGG 19 and ResNet comparisions:

- **Depth:**

VGG 19 has 19 layers, while ResNet has 50 or more layers.

- **Training Speed:**

VGG 19 tends to have slower training speed due to its deeper architecture, while ResNet is faster due to skip connections.

- **Feature Capture:**

VGG 19 may capture more detailed features from images, whereas ResNet is effective in capturing relevant features with fewer parameters.

- **Complexity:**

VGG 19 has a relatively simpler architecture, making it easier to understand and implement, while ResNet's architecture is more complex with residual

connections, requiring a deeper understanding for implementation and troubleshooting.

- **Implementation:**

VGG 19 is easier to understand and implement, while ResNet requires a deeper understanding for implementation.

- **Fine-Tuning:**

VGG 19 requires more fine-tuning on specific tasks, but ResNet achieves good performance with minimal adjustments.

- **Computational Cost:**

VGG 19 is more computationally expensive due to the larger number of parameters and computations, whereas ResNet is more computationally efficient, especially for deeper architectures, due to reduced parameter redundancy and computational complexity.

- **Accuracy:**

VGG 19 achieves a higher accuracy rate compared to ResNet, which has comparatively lesser accuracy.

6.3 FLASK(Backend)

Flask serves as the backend framework responsible for handling HTTP requests, executing business logic, and communicating with the frontend Flutter application. Install Flask using pip, the Python package manager. Set up a virtual environment to manage dependencies and isolate the project's environment.

Define configuration settings for Flask, including database connection details, secret keys for session management, and debug mode settings. Organize the project's backend codebase into a structured directory layout. Create separate modules or packages for different components of the application, such as routes, models, controllers, and utilities. Design RESTful API endpoints to define the interface for communication between the frontend Flutter application and the backend Flask server. Define routes for handling various HTTP methods (GET,

POST, PUT, DELETE) and corresponding actions (e.g., fetching blood type predictions, submitting fingerprint images). Choose a database management system (DBMS) such as MySQL for storing user data, fingerprint images, and blood type predictions. Integrating Flask with the chosen DBMS using Object-Relational Mapping (ORM) library, for simplified database operations and query management. Implement authentication and authorization mechanisms to secure access to protected endpoints and ensure data privacy. Utilize Flask extensions such as Flask-Login or Flask-JWT for session management, user authentication, and role-based access control (RBAC).

Defining route handlers using Flask's routing mechanism to process incoming HTTP requests and execute corresponding business logic. Implementing algorithms for fingerprint analysis, blood type prediction, and result generation within Flask route handlers. Implementing error handling mechanisms to gracefully handle exceptions, validation errors, and other unexpected conditions. Utilize Flask's error handlers to return appropriate HTTP error responses with descriptive error messages. Unit tests using frameworks such as PyTest to validate the functionality of Flask route handlers and business logic. Conducting integration tests to ensure seamless interaction between the frontend Flutter application and the backend Flask server via RESTful API endpoints. Deploying the Flask backend to a production environment using deployment platforms such as AWS. Configuring environment variables, security settings, and scalability options to optimize performance and ensure reliability in production environments.

6.4 HTML, CSS(Frontend)

HTML, or Hypertext Markup Language, serves as the foundation for structuring the web application, delineating its layout and components within the user interface. Employing semantic HTML tags like <header>, <nav>, <section>, and <footer> enhances accessibility and bolsters search engine

optimization (SEO). The inclusion of HTML forms empowers users to upload fingerprint images for analysis, while also incorporating input fields for personal details such as name and age.

CSS, or Cascading Style Sheets, steps in to imbue buttons with visual allure and enhance user interaction. Leveraging CSS, hover effects, transitions, and animations are seamlessly integrated to elevate the overall user experience. Furthermore, CSS media queries are harnessed to ensure responsiveness across diverse screen sizes and devices.

Within this framework, CSS not only styles the elements crafted with HTML but also extends its capabilities to define colors, fonts, margins, padding, borders, and other visual attributes. Background images, gradients, and patterns are skillfully applied using CSS to enrich the aesthetic appeal of the web application.

6.5 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are pivotal in analyzing fingerprint images and predicting associated blood types. Designed specifically for image processing and classification, CNNs are ideal for tasks like fingerprint analysis. VGG-19, selected for its balance between computational efficiency and performance, offers various trade-offs in model size, accuracy, and computational cost. Initializing the CNN model with pre-trained weights from large-scale image datasets such as ImageNet leverages transfer learning, accelerating training and improving convergence in blood type prediction. Fine-tuning the pre-trained CNN model involves retraining the last few layers on a dataset of fingerprint images annotated with corresponding blood types, adapting learned features to the specific characteristics of fingerprint patterns relevant for blood type identification. Preprocessing fingerprint images before feeding them into the CNN model enhances their suitability for analysis. Preprocessing steps may involve resizing images to a consistent input size, normalizing pixel values,

and employing augmentation techniques to increase the diversity of training data. Training the CNN model using a dataset of labeled fingerprint images and corresponding blood types utilizes techniques like mini-batch stochastic gradient descent (SGD), learning rate scheduling, and regularization (e.g., dropout) to optimize model parameters and prevent overfitting. Validation on a separate dataset assesses the trained CNN model's performance and generalization ability.

Using evaluation metrics such as accuracy, precision, recall to quantify the model's predictive performance in blood type identification. Integrate the trained CNN model into the mobile application's backend infrastructure, allowing it to perform real-time blood type prediction from fingerprint images captured by the device's camera. Optimize the CNN model for deployment on mobile devices, considering constraints such as computational resources, memory usage, and inference speed.

CHAPTER 7
APPLICATION IMPLEMENTATION

The application entails various stages, including setting up the development environment, constructing backend and frontend components, integrating the fingerprint sensor, AI model development, testing, and deployment. Begin by installing and configuring essential development tools and frameworks like Visual Studio Code for IDE, Flutter SDK, and Python environment. Establish version control using Git and initiate a repository on a hosting platform such as GitHub for the project. Organize a Flask project structure with directories allocated for routes, models, and utilities. Implement RESTful API endpoints for user authentication, fingerprint image upload, blood type prediction, and result retrieval. Incorporate a MySQL database for storing user data, fingerprint images, and blood type predictions. Develop functions utilizing Python and OpenCV libraries for image preprocessing, feature extraction, and blood type classification. Create a Flutter project structure encompassing screens, widgets, and services. Design and implement user interface screens for fingerprint image capture, blood type prediction, and result display utilizing Flutter's widget library. Integrate logic for capturing fingerprint images via the device's fingerprint sensor API and uploading them to the backend server. Implement API calls to interact with the Flask backend for blood type prediction and result retrieval.

Design and create user interface screens using CSS and HTML for capturing fingerprint images, predicting blood type, and displaying results. Implement logic to capture fingerprint images using the device's fingerprint sensor API and upload them to the backend server. Integrate API calls to communicate with the Flask backend for blood type prediction and result retrieval. Utilize platform-specific APIs like the Android Fingerprint API for seamless integration of the fingerprint sensor. Develop functionality to prompt users to authenticate using the fingerprint sensor before capturing fingerprint

images, handling authentication success and failure events appropriately. Choose the VGG-19 pre-trained convolutional neural network model for blood type prediction and fine-tune it using transfer learning techniques with fingerprint image datasets. Implement functions to load the trained model, preprocess input images, and make predictions using TensorFlow. Perform unit tests for backend functions, API endpoints, and frontend components using PyTest.

Conduct integration testing to verify smooth communication between frontend and backend components through RESTful API calls. Test the application on both physical devices and emulators/simulators to confirm functionality, usability, and performance across various scenarios. Deploy the Flask backend on a cloud platform and establish continuous integration/continuous deployment (CI/CD) pipelines to automate the build, test, and deployment phases. With successful development and deployment, the application delivers a convenient and dependable tool for blood type identification using fingerprint analysis to users.

CHAPTER 8

SYSTEM TESTING

System testing for the project involves evaluating the entire system as a whole to ensure that all components function together seamlessly and meet the project's objectives.

8.1 UNIT TESTING:

Flask API:

PyTest provides a simple syntax and powerful features for writing and executing tests. This includes testing route handlers, request processing logic, error handling, and response formatting. This ensures that tests focus solely on the functionality of the API endpoints.

Frontend (HTML, CSS):

Tested the integration between HTML forms and AI algorithm for image uploading and analysis. Tested the integration between CSS styles and HTML elements to ensure proper styling across different browsers and devices.

AI Model:

TensorFlow provides testing utilities and frameworks for evaluating the performance and accuracy of machine learning models. Verifies the model outputs the expected blood type given sample fingerprint images.

Integration Testing:

Integration test validates the interaction between different parts of the system, such as the frontend, backend, AI model and tests ensure that the application functions correctly as a whole.

CHAPTER 9
CONCLUSION

This project represents a significant advancement in the field of biometric analysis and healthcare technology. Through the integration of artificial intelligence algorithms with fingerprint recognition systems, the project has demonstrated promising results in accurately determining an individual's blood group based on their fingerprint patterns. The development and implementation of this innovative approach hold immense potential for various applications, including medical diagnostics, personalized healthcare, and forensic investigations. It provides a non-invasive and efficient method for blood group identification, the project addresses the need for rapid and reliable diagnostic tools in clinical settings, especially in resource-constrained environments where traditional blood typing methods may be inaccessible or time-consuming.

APPENDIX I

SOURCE CODE

BACKEND CODE:

```
from flask import Flask, render_template, request
from flask import Flask, render_template, request
import os
import numpy as np
from tensorflow.keras.preprocessing import image  # Add this import statement
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.applications import VGG19
from tensorflow.keras.models import Model
from tensorflow.keras.applications.imagenet_utils import preprocess_input
app = Flask(__name__)
# Define the path to the directory containing blood group images
file_path = '/home/karuppasamy/Downloads/archive (1)/dataset_blood_group'
# Check if the directory exists
if not os.path.isdir(file_path):
    print("Error: Directory '{}' does not exist.".format(file_path))
    exit()
# Get the list of blood group names from the directory
name_class = sorted(os.listdir(file_path))
print("List of blood group names:", name_class)

# Assigning blood group names to indices
blood_group_names = {index: blood_group for index, blood_group in
enumerate(name_class)}
print("Blood group indices with names:", blood_group_names)
# Load the trained model
model = None
```

```
def load_model():
    global model
    try:
        # Load VGG19 model
        vgg19_base = VGG19(weights='imagenet', include_top=False,
input_shape=(256, 256, 3))
        # Remove the last layers of VGG-19
        output = vgg19_base.layers[-1].output
        # Add Global Average Pooling layer
        output = GlobalAveragePooling2D()(output)
        # Create a new model with VGG-19 base and global average pooling
        pretrained_model = Model(inputs=vgg19_base.input, outputs=output)
        # Freeze the layers
        for layer in pretrained_model.layers:
            layer.trainable = False
        inputs = pretrained_model.input
        x = Dense(128, activation='relu')(pretrained_model.output)
        x = Dense(128, activation='relu')(x)

        outputs = Dense(len(name_class), activation='softmax')(x)
        model = Model(inputs=inputs, outputs=outputs)
        model.compile(
            optimizer='adam',
            loss='categorical_crossentropy',
            metrics=['accuracy']
        )
        # Save the model weights
        model.save_weights("model_blood_group_detection.h5")
        print("Model loaded successfully.")
```

```
    except Exception as e:
        print("Error loading model:", e)
# Function to analyze the image and predict the blood group with confidence scores
def analyze_image(file_path):
    try:
        img = image.load_img(file_path, target_size=(256, 256))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        x = preprocess_input(x)
        result = model.predict(x)[0]  # Get the prediction result for the first image (batch size = 1)
        print(result)
        value=(result*100).astype('int')
        print(value)
        confidence_scores = {blood_group_names[i]: score for i, score in enumerate(result)}
        predicted_blood_group = max(confidence_scores, key=confidence_scores.get)
        print(predicted_blood_group)
        return predicted_blood_group
    except Exception as e:
        print("Error analyzing image:", e)
        return 'Unknown', {}
# Route for the home page
@app.route('/', methods=['GET', 'POST'])
def home():
    if request.method == 'POST':
        if 'file' not in request.files:
```

```
            return render_template('index.html', message='No file part')
        file = request.files['file']
        if file.filename == '':
            return render_template('index.html', message='No selected file')
        if file:
            # Save the uploaded file to the 'uploads' directory
            file_path = os.path.join('uploads', file.filename)
            file.save(file_path)

            # Analyze the uploaded image
            predicted_blood_group = analyze_image(file_path)
            return render_template('result.html',
blood_group=predicted_blood_group)
    return render_template('index.html')

if __name__ == '__main__':
    load_model()  # Load the model when the application starts
    app.run(debug=True)
```

HOME PAGE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blood Group Identification</title>
    <style>
      body {
            font-family: Arial, sans-serif;
```

```
        background-image: url('https://scontent.xx.fbcdn.net/v/t1.15752-9/438115853_454271750589199_1055235032389431579_n.jpg?stp=dst-jpg_p403x403&_nc_cat=107&ccb=1-7&_nc_sid=5f2048&_nc_ohc=UoCh76MBkXEQ7kNvgFumuEs&_nc_ad=z-m&_nc_cid=0&_nc_ht=scontent.xx&oh=03_Q7cD1QHwFcop6XGMGkyXg5xfhR073bzkD-dvZq4gG5-Hj6McLQ&oe=665C1E36');
        background-size: 2000px 1200px;
        background-position: center;
        background-repeat: no-repeat; /* Prevent repetition */
        margin: 0;
        padding: 0;
    }

    .container {
        max-width: 800px;
        margin: 220px auto;
        padding: 20px;
        background-color: rgba(255, 255, 255, 0.8); /* Semi-transparent white background for better readability */
        border-radius: 8px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
    h1 {
        text-align: center;
        margin-bottom: 30px;
    }
    form {
        text-align: center;
        margin-bottom: 20px;
```

```
        }
        label {
            display: block;
            margin-bottom: 10px;
        }
        input[type="file"] {
            display: block;
            margin: 0 auto;
            margin-bottom: 20px;
        }
        input[type="submit"] {
            padding: 10px 20px;
            background-color: #007bff;
            color: #fff;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            transition: background-color 0.3s;
        }
        input[type="submit"]:hover {
            background-color: #0056b3;
        }
        p.message {
            text-align: center;
            color: #ff0000;
            margin-top: 20px;
        }
    </style>
</head>
```

```
<body>
    <div class="container">
        <h1>Blood Group Identification</h1>
        <form method="POST" enctype="multipart/form-data">
            <label for="file">Upload Image:</label>
            <input type="file" name="file" accept=".png, .jpg, .jpeg, .BMP">
            <input type="submit" value="Upload Image">
        </form>
        <hr>
        <form method="POST">
            <label for="fingerprint">Scan Fingerprint:</label>
            <button type="button" onclick="scanFingerprint()">Scan
Fingerprint</button><br><br>
            <input type="hidden" name="fingerprintData" id="fingerprintData">
            <input type="submit" value="Submit Fingerprint">
        </form>
        {% if message %}
            <p class="message">{{ message }}</p>
        {% endif %}
    </div>

    <script>
        function scanFingerprint() {
            // Placeholder function for scanning fingerprint
            // You can implement your logic for fingerprint scanning here
            alert("Fingerprint scanned successfully!");
            document.getElementById("fingerprintData").value =
"fingerprint_data_placeholder";
        }
```

```
  </script>
</body>
</html>
```

RESULT PAGE:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Analysis Result</title>
  <style>
    body {
  font-family: Arial, sans-serif;
  background-image: url('https://scontent.xx.fbcdn.net/v/t1.15752-
9/438115853_454271750589199_1055235032389431579_n.jpg?stp=dst-
jpg_p403x403&_nc_cat=107&ccb=1-
7&_nc_sid=5f2048&_nc_ohc=UoCh76MBkXEQ7kNvgFumuEs&_nc_ad=z-
m&_nc_cid=0&_nc_ht=scontent.xx&oh=03_Q7cD1QHwFcop6XGMGkyXg5x
fhR073bzkD-dvZq4gG5-Hj6McLQ&oe=665C1E36');
  background-size: 2000px 1200px;
  background-position: center;
  background-repeat: no-repeat; /* Prevent repetition */
  margin: 500px 0 0 0; /* 50px top margin, 0 right, 0 bottom, 0 left */
  padding: 0;
}

      .container {
          max-width: 800px;
```

```
            margin: 10% auto;
            background-color: rgba(255, 255, 255, 0.8);
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
            text-align: center;
            color: #333;
        }
        p {
            font-size: 18px;
            color: #666;
        }
        .result {
            font-weight: bold;
            color: #00aaff;
            text-transform: uppercase;
            font-size: 20px;
            margin-top: 10px;
        }

        @media screen and (max-width: 600px) {
            .container {
                margin: 20% auto;
            }
        }
    </style>
</head>
```

```
<body>
    <div class="container">
        <h1>Analysis Result</h1>
        <p>Predicted Blood Group: <span class="result">{{ blood_group
}}</span></p>
    </div>
</body>
</html>
```

APPENDIX II

SCREENSHOTS

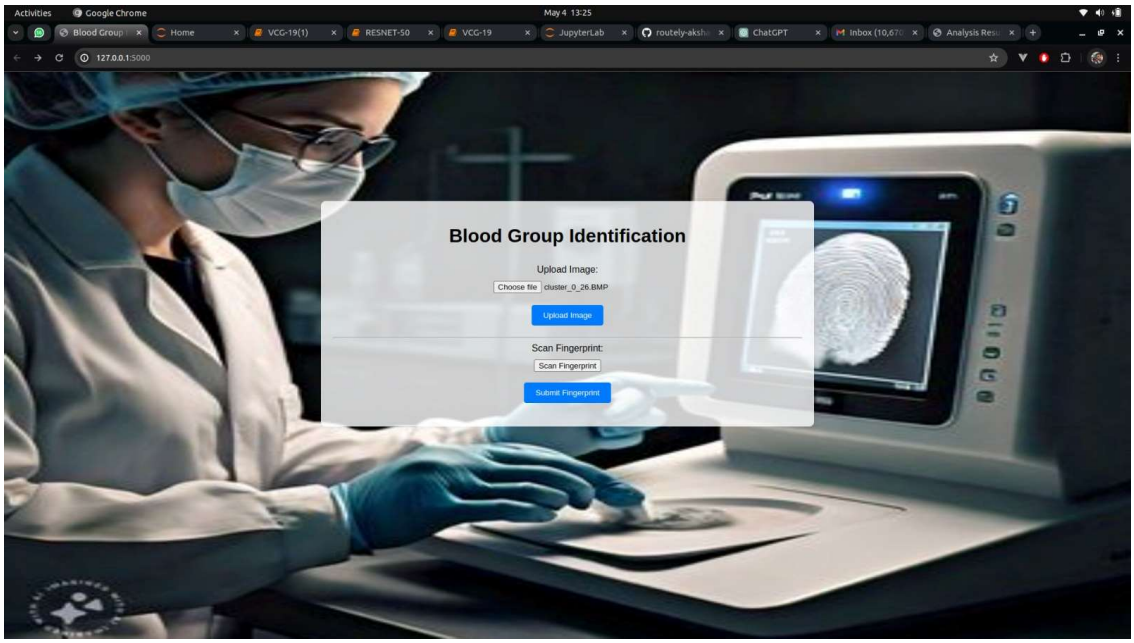

Fig A.2.1 Fingerprint Scanning or uploading fingerprint image

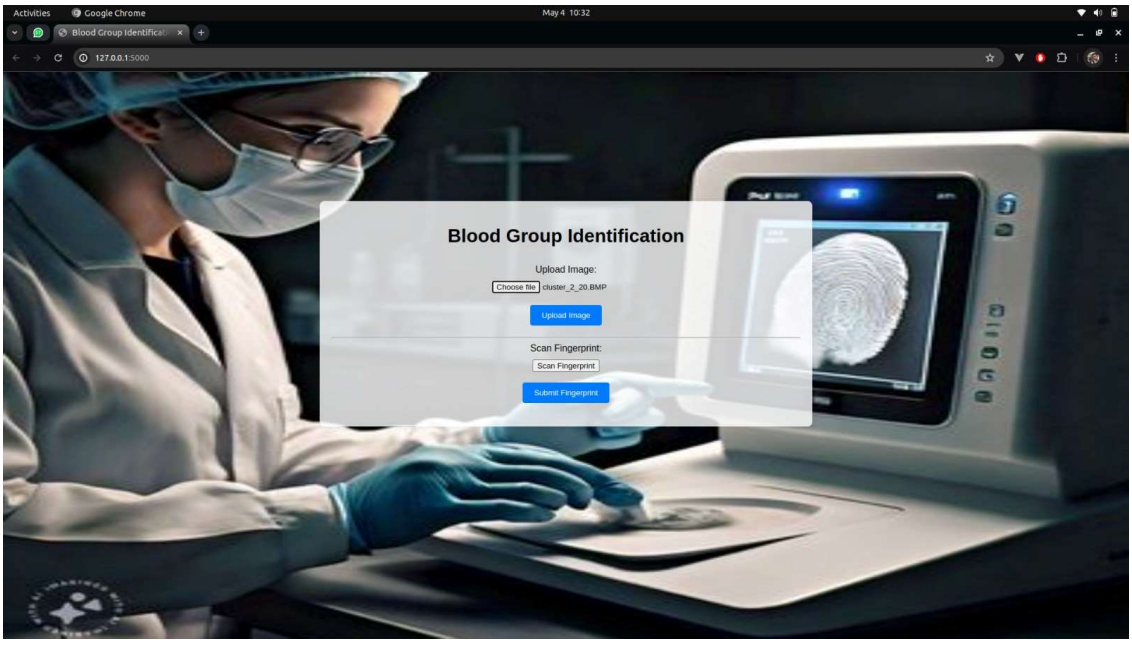

Fig A.2.2 Uploading fingerprint images

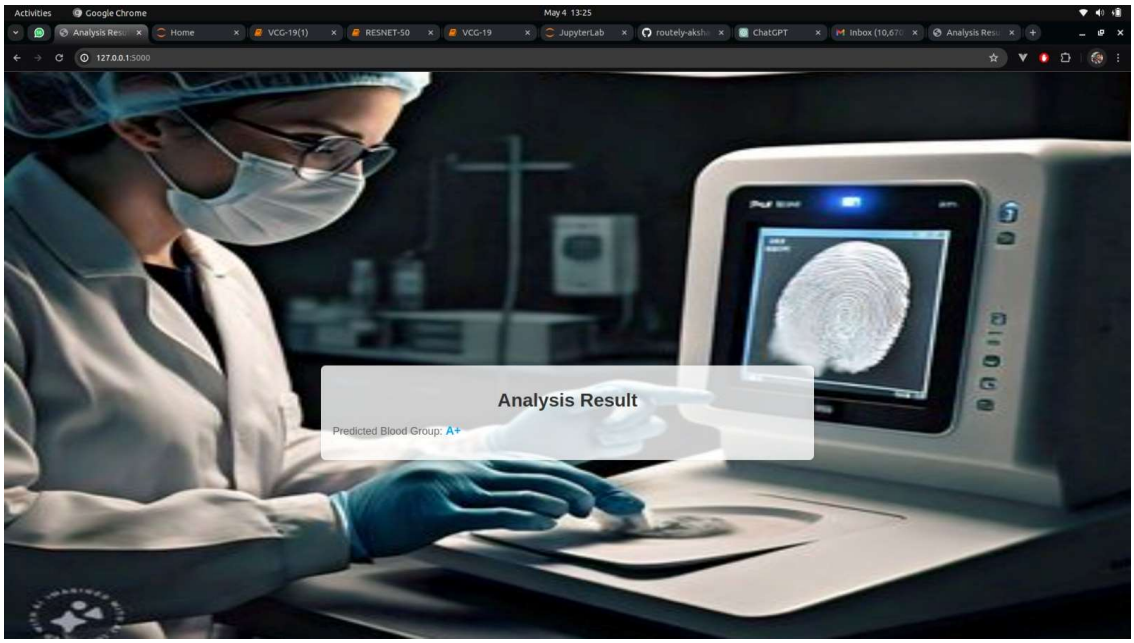

Fig A.2.3 Blood group analysis result for “A+” blood type

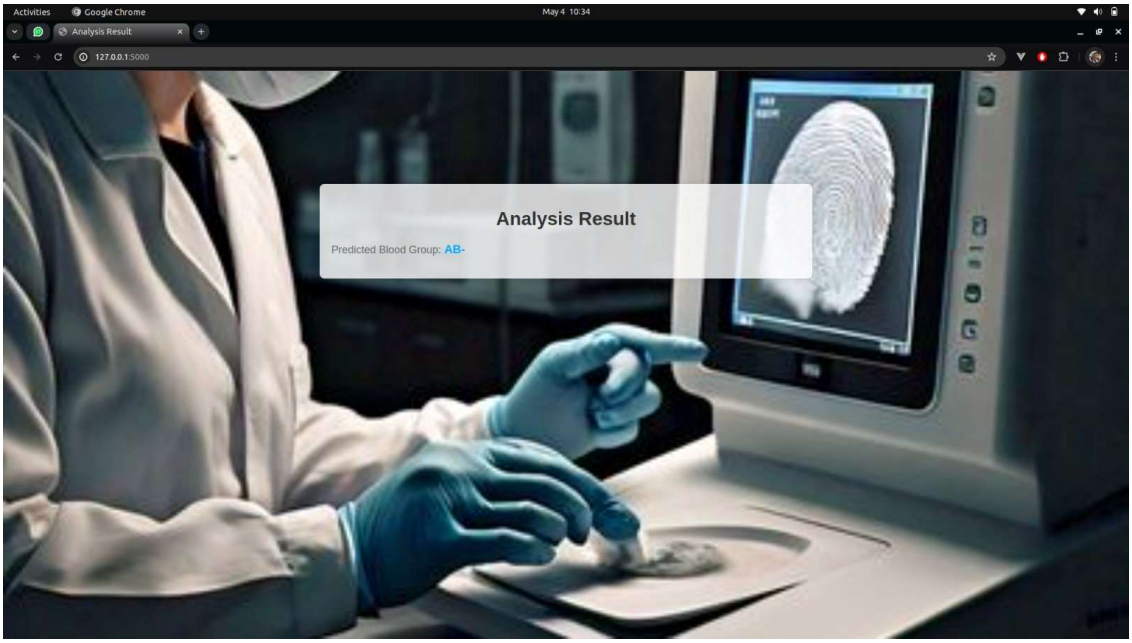

Fig A.2.4 Blood group analysis result for “AB-” blood type

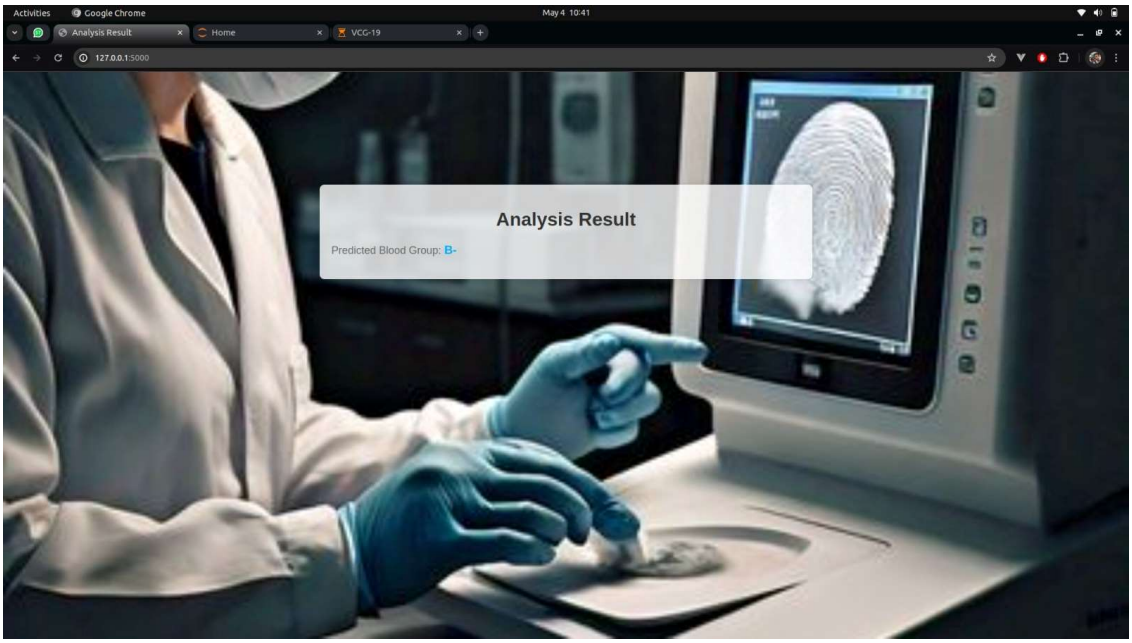

Fig A.2.5 Blood group analysis result for “B-” blood type

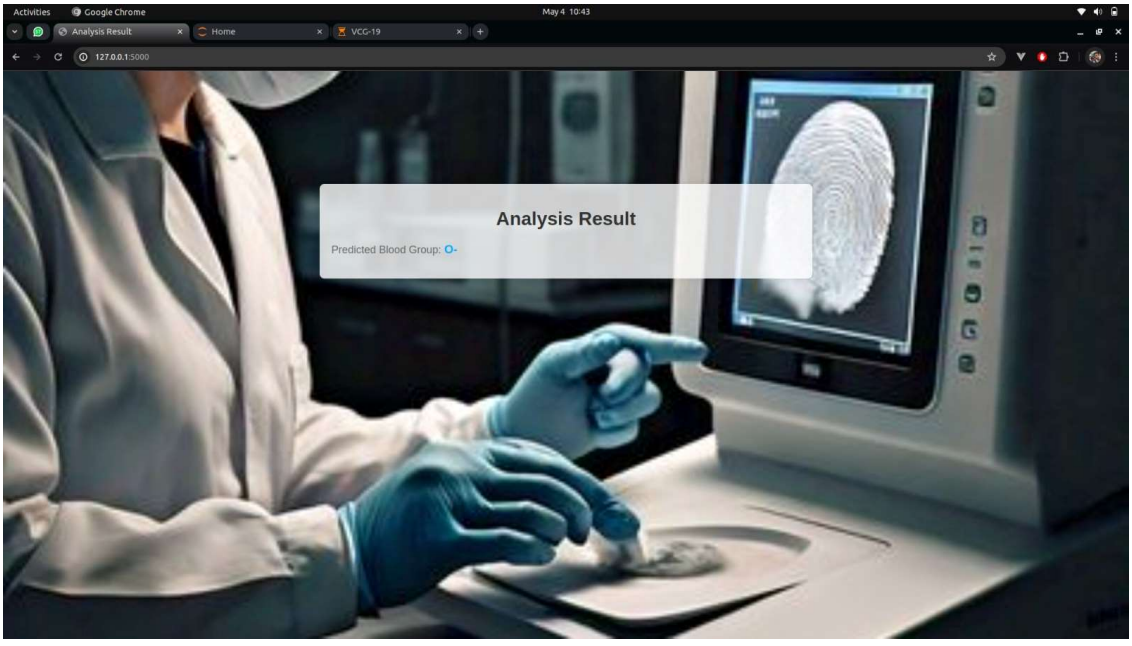

Fig A.2.6 Blood group analysis result for “O-” blood type

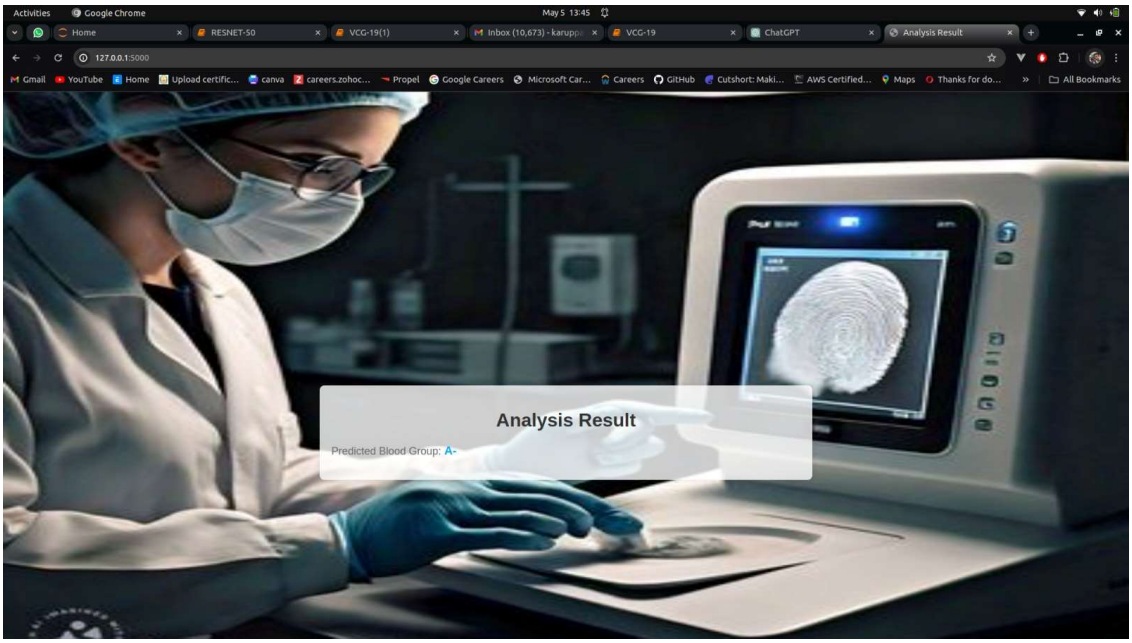

Fig A.2.7 Blood group analysis result for "A-" blood type

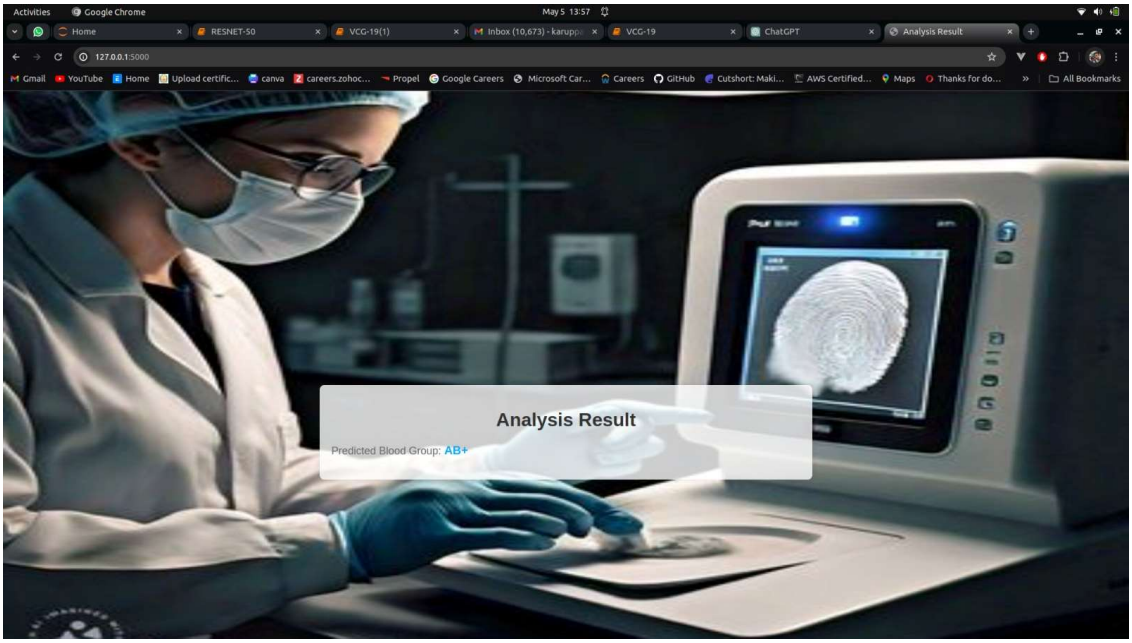

Fig A.2.8 Blood group analysis result for "AB+" blood type

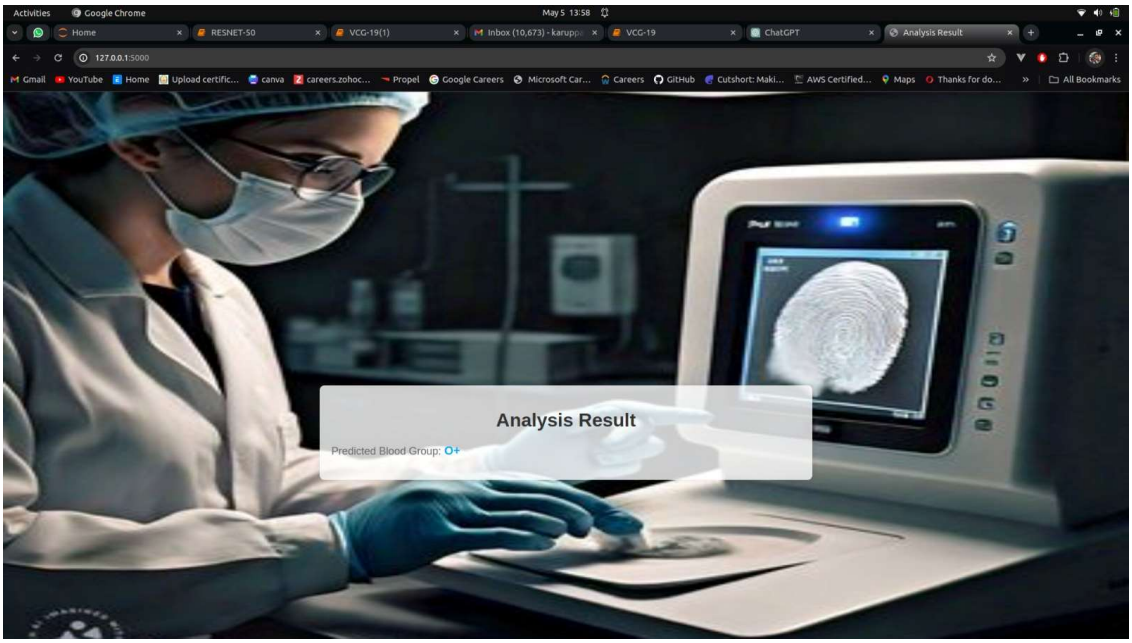

Fig A.2.9 Blood group analysis result for “O+” blood type

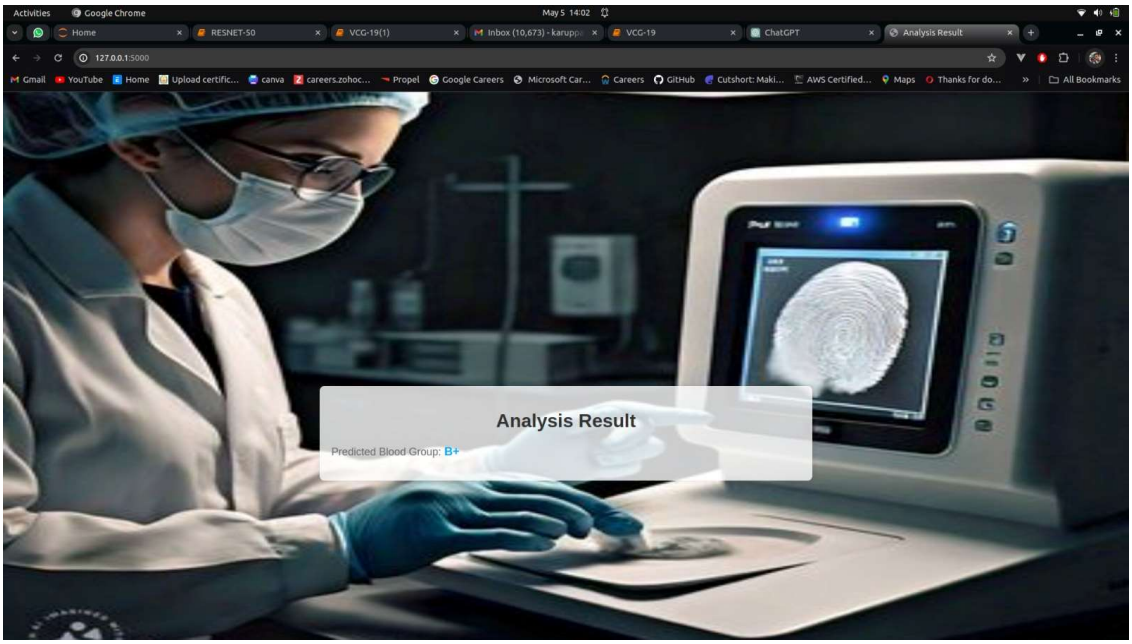

Fig A.2.10 Blood group analysis result for “B+” blood type

APPENDIX III

REFERENCES

[1] Al Habsi, Tariq, Hussein Al Khabori, Sara Al Qasmi, Tasnim Al Habsi, Mohamed Al Mushaiqri, Srijit Das, and Srinivasa Rao Sirasanagandla. "The association between fingerprint patterns and blood groups in the Omani population." Arab Gulf Journal of Scientific Research, 2023.

[2] Takahashi, Ai, Yoshinori Koda, Koichi Ito, and Takafumi Aoki. "Fingerprint feature extraction by combining texture, minutiae, and frequency spectrum using multi-task CNN." In 2020 IEEE International Joint Conference on Biometrics (IJCB), pp. 1-8. IEEE, 2020.

[3] Kukadiya, Urvik, Pratik Trivedi, Ashish Rathva, and Chintan Lakhani. "Study of fingerprint patterns in relationship with blood group and gender in saurashtraregion." International Journal of Anatomy and Research 8, no. 2.3 (2020): 7564-7567.

[4] Smail, Harem Othman, Dlnya Ahmed Wahab, and Zhino Yahia Abdullah. "Relationship between pattern of fingerprints and blood groups." J Adv Lab Res Biol 10, no. 3 (2019): 84-90.

[5] Fernandes, Jose, Sara Pimenta, Filomena O. Soares, and Graca Minas. "A complete blood typing device for automatic agglutination detection based on absorption spectrophotometry." IEEE Transactions on Instrumentation and Measurement 64, no. 1 (2014): 112-119

[6] Ali, Mouad MH, Vivek H. Mahale, Pravin Yannawar, and A. T. Gaikwad. "Fingerprint recognition for person identification and verification based on minutiae matching." In 2016 IEEE 6th international conference on advanced computing (IACC), pp. 332-339. IEEE, 2016.

[7] Rastogi, Ashok, MD Abu Bashar, Nishat Ahmed Sheikh, and MD ABU BASHAR. "Relation of Primary FingerprintPatternsWith Gender and Blood Group: A Dermatoglyphic Study From a Tertiary Care Institute in Eastern India."Cureus 15, no. 5 (2023).

APPENDIX IV

CONFERENCE CERTIFICATES

ARJUN COLLEGE OF TECHNOLOGY

Thamaraikulam, Coimbatore - 642 120

ICRASET - 2024

CERTIFICATE

OF PARTICIPATION

This certificate is proudly presented to

Dr/Mr/Mrs/Ms ANBU SELVAM V

TITLE ARTICLE ID - BCS036

AI APPROACH TO DETECT BLOOD TYPE FROM FINGERPRINT

In recognition of his/her participation in the **6th International Conference on Recent Advances in Science, Engineering and Technology** (ICRASET - 2024)

Convenor
Dr. J. Thilagavathi

Principal
Dr. N. Janaki Manohar


ARJUN
COLLEGE OF TECHNOLOGY
INSTITUTION'S
INNOVATION
COUNCIL
(Ministry of Education Initiative)
ARJUN COLLEGE OF TECHNOLOGY
Thamaraikulam, Coimbatore - 642 120
ICRASET - 2024
CERTIFICATE
OF PARTICIPATION
This certificate is proudly presented to
Dr/Mr/Mrs/Ms
INDHUMATHI M
TITLE
ARTICLE ID - BCS036
AI APPROACH TO DETECT BLOOD TYPE FROM FINGERPRINT
In recognition of his/her participation in the
6th International Conference on
Recent Advances in Science, Engineering and
Technology (ICRASET - 2024)
Convenor
Dr. J. Thilagavathi
Principal
Dr. N. Janaki Manohar


ARJUN
COLLEGE OF TECHNOLOGY
INSTITUTION'S
INNOVATION
COUNCIL
(Ministry of Education Initiative)
ARJUN COLLEGE OF TECHNOLOGY
Thamaraikulam, Coimbatore - 642 120
ICRASET - 2024
CERTIFICATE
OF PARTICIPATION
This certificate is proudly presented to
Dr/Mr/Mrs/Ms KARUPPASAMY B
TITLE
ARTICLE ID - BCS036
AI APPROACH TO DETECT BLOOD TYPE FROM FINGERPRINT
In recognition of his/her participation in the
6th International Conference on
Recent Advances in Science, Engineering and
Technology (ICRASET - 2024)
Convenor
Dr. J. Thilagavathi
Principal
Dr. N. Janaki Manohar


ARJUN
COLLEGE OF TECHNOLOGY
INSTITUTION'S
INNOVATION
COUNCIL
ARJUN COLLEGE OF TECHNOLOGY
Thamaraikulam, Coimbatore - 642 120
ICRASET - 2024
CERTIFICATE
OF PARTICIPATION
This certificate is proudly presented to
Dr/Mr/Mrs/Ms
NARRENDRAN P
TITLE
ARTICLE ID - BCS036
AI APPROACH TO DETECT BLOOD TYPE FROM FINGERPRINT
In recognition of his/her participation in the
6th International Conference on
Recent Advances in Science, Engineering and
Technology (ICRASET - 2024)
Convenor
Dr. J. Thilagavathi
Principal
Dr. N. Janaki Manohar