

TOUCHLESS TOKEN GENERATION SYSTEM

A PROJECT REPORT

Submitted by

NIKILA MANOJ (TKM21MCA-2030)

to

The APJ Abdul Kalam Technological University

In partial fulfillment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATION



**Thangal Kunju Musaliar College of Engineering
Kerala**

DEPARTMENT OF COMPUTER APPLICATION

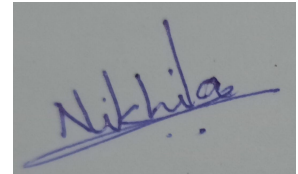
MAY 2023

DECLARATION

I undersigned hereby declare that the project report on **TOUCHLESS TOKEN GENERATION SYSTEM**, submitted for partial fulfillment of the requirements for the award of the degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under the supervision of Prof. Jasmin M R. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Kollam

16-05-2023



NIKILA MANOJ

DEPARTMENT OF COMPUTER APPLICATION
TKM COLLEGE OF ENGINEERING



CERTIFICATE

This is to certify that the report entitled **TOUCHLESS TOKEN GENERATION SYSTEM** submitted by **NIKILA MANOJ** (TKM21MCA-2030) to the APJ Abdul Kalam Technological University in partial fulfillment of the Masters degree in Computer Application is a bonafide record of the project work carried out by her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor

Head of the Department

External Examiner

Acknowledgement

First and foremost I thank GOD almighty and my parents for the success of this project. I owe sincere gratitude and heartfelt thanks to everyone who shared their precious time and knowledge for the successful completion of my project.

I am extremely grateful to **Dr. Fousia M Shamsudeen**, Head of the Department, Department of Computer Application, for providing me with the best facilities.

I would like to express my sincere gratitude to the project coordinator, **Prof. Vaheetha Salam**, Department of Computer Application, for their invaluable guidance, support, and encouragement throughout the entire duration of this project.

I would like to thank my project guide **Prof. Jasmin M R**, Department of Computer Application, who motivated me throughout the project.

I would like to express my heartfelt gratitude to the advisor, **Prof. Natheera Beevi M**, Department of Computer Application, for the unwavering support throughout this project.

I profusely thank all other faculty members in the department and all other members of TKM College of Engineering, for their guidance and inspiration throughout my course of study. I owe my thanks to my friends and all others who have directly or indirectly helped me in the successful completion of this project.

NIKILA MANOJ

ABSTRACT

TOUCHLESS TOKEN GENERATION SYSTEM is a software solution that generates tokens without physical contact or manual interaction. It helps in reducing wait times, increasing customer satisfaction, and reducing the interaction between people or objects. In the current scenario, it is equally important to have an automated system and avoid the spread of diseases through unsanitary touch. The proposed system can be used to overcome the existing manual token dispenser by employing a webcam or a built-in camera for capturing hand gestures and hand tip detection using computer vision to generate tokens. The system makes use of machine learning classification algorithms such as RandomForest, DecisionTree, NaiveBayes, SVM, and KNN to train the model. The SVM classifier achieved an accuracy rate of 98.47%, which was the highest among all the classifiers. Based on hand gestures, the computer can be controlled virtually and can perform token generation without the use of a physical object. The system uses computer vision library to fetch input frames from the video capturing the hands. It uses Mediapipe framework to detect hand gestures thereby marking significant landmarks. Each time a hand gesture is detected the token count increases. Hence, the proposed system eliminates the need for physical contact reducing the risk of spreading infection and contamination. Tokens can be generated quickly and easily making the process faster and more convenient for users.

Contents

List of Figures	iii
List of Abbreviations	v
1 Introduction	1
1.1 Problem Statement	2
1.2 Objective	2
1.3 Existing System	3
1.3.1 Limitations of Existing System	3
1.4 Proposed system	4
2 Literature Survey	5
2.1 Purpose of the Literature Review	5
2.2 Related Works	6
3 Methodology	12
3.1 Algorithm	12
3.2 System Architecture	13
3.2.1 Dataset	14
3.2.2 Image Processing	14
3.2.3 Building and training the model	15
3.2.4 Testing and Deployment	22
3.3 System Specifications	25
3.3.1 Software Specification	25
3.3.2 Software Description	25
3.4 Hardware and experimental environment	27

4	RESULT AND DISCUSSION	28
4.1	Training and Validation Results	28
4.1.1	Training and Validation Accuracy Graphs	30
4.1.2	Graph of Comparison of the models used.	33
4.2	Performance Metrics for Validation Phase	36
4.2.1	Confusion Matrix	36
5	CONCLUSION	41
5.1	Future Enhancement	42
	REFERENCE	43
	APPENDIX	45

List of Figures

3.1	Training the system	13
3.2	Deployment of the system	13
3.3	Indian Sign Language dataset	14
3.4	Image Processing using Mediapipe	15
3.5	RandomForest	17
3.6	Decision Tree	18
3.7	Naive Bayes	19
3.8	Support Vector machine	20
3.9	KNN	21
3.10	Hand landmarks identified by Mediapipe	23
4.1	Comparison of models	29
4.2	Training and Testing Accuracy of Random Forest	30
4.3	Training and Testing Accuracy of Decision Tree	31
4.4	Training and Testing Accuracy of Naive Bayes	31
4.5	Training and Testing Accuracy of SVM	32
4.6	Training and Testing Accuracy of KNN	32
4.7	Comparison of Testing accuracy of models	33
4.8	Comparison of Testing accuracy of models	34
4.9	Comparison of Testing accuracy of models	35
4.10	Confusion Matrix of Random Forest	36
4.11	Confusion Matrix of Decision Tree	37
4.12	Confusion Matrix of Naive Bayes	38
4.13	Confusion Matrix of SVM	39
4.14	Confusion Matrix of KNN	40

A.1	Home Page	45
A.2	About Page	46
A.3	Token Generation page	46

List of Abbreviations

DT - Decision Tree

GNB - Gaussian Naive Bayes

KNN - K-Nearest Neighbors

OpenCV - Open Source Computer Vision

RF - Random Forest

SVM - Support Vector Machine

Chapter 1

Introduction

TOUCHLESS TOKEN GENERATION SYSTEM, is a system that enables users to generate tokens without physically touching any devices. Instead, the system uses hand gestures captured by the built-in camera to authenticate the user and generate the required token. The World Health Organization has reported that approximately 3 million individuals worldwide have been infected with COVID-19 while in hospital settings due to unsanitary conditions and unsafe contact. Therefore, it is crucial to establish an environment that enables individuals to visit places where large crowds gather, such as hospitals, while minimizing the risk of contracting other illnesses. In order to prevent the transmission of contact-based diseases and maintain social distancing, a software system has been developed for generating tokens.

By using hand gestures, this technology eliminates the need for physical contact and provides a more secure method of medical security. The system captures the unique characteristics of the user's hand gestures to generate the token or ticket. Some of the benefits of touchless token generation systems include increased convenience, improved security, and reduced risk of germ transmission. These systems can be used in a wide range of applications, such as hospitals and medical clinics, banks and financial institutions, government offices, retails and supermarkets etc.

With the use of computer vision technologies, a model is developed for the purpose of generating tokens. Mediapipe framework's hand detection pipeline is used to detect and trace hand landmarks which provides 21 specific key points for each detected hand. The extracted data is used to train the model using different classification algorithms-RandomForest(RF), DecisionTree(DT), NaiveBayes, Support Vector Machine (SVM), and K-nearest neighbors (KNN). Classification algorithms in the computer vision field enable machines to interpret and

understand visual data in ways that were once only possible for humans. During the testing phase, the images of hand gestures captured using OpenCV are pre-processed and fed into a saved classification model to determine the expected gesture. The model uses this input to predict the corresponding gesture. The token number is incremented if a hand gesture is detected and maintained for a specified duration, such as 5 seconds.

1.1 Problem Statement

The problem statement of the project is:

- The system employs advanced computer vision and machine learning technologies to develop a token generation platform to generate tokens by interpreting hand gestures captured in real-time video feeds.

1.2 Objective

The objectives of the project are to achieve the following:

- To develop a reliable and efficient system for generating tokens without physical contact, and ensure public health protection.
- To develop a machine learning model to accurately recognize and classify hand gestures in real-time.
- To train the system to recognize a wide range of hand gestures accurately.
- To design a system that can handle high-traffic areas and queues, increasing efficiency and reducing wait times.

1.3 Existing System

The existing system of token generation system is a straightforward and low-tech approach that relies more on manual processes. The existing token generation system can be categorized into two main approaches: a straightforward and manual-based approach, and a low-tech solution that is, a token dispenser.

- **Manual Handwritten Token**

Manual handwritten tokens are generated by individuals, typically receptionists or staff members, who physically write or print the token information on pieces of paper. These tokens serve as unique identifiers for customers or individuals in a queue or waiting area. The details of the person, such as their name or other relevant information, may be included along with the token number.

- **Token Dispenser**

A token dispenser is an automated device used to generate and distribute tokens in a queue management system. It is designed to streamline the process and eliminate the need for manual token generation. When a customer or individual arrives, they interact with the token dispenser, which typically features a touchscreen interface or buttons. The customer selects their desired service or purpose, and the dispenser generates a token.

1.3.1 Limitations of Existing System

- **Hygiene concerns:** The main disadvantage of traditional token systems can pose a risk during times like COVID-19 as multiple people handle the system, increasing the chances of contamination and potential virus transmission.
- **Time-consuming:** The process of existing token systems is time-consuming, leading to longer waiting times and potential customer dissatisfaction.
- **Accessibility:** Existing token generation system may not accommodate individuals with disabilities or older age people, causing difficulties in handling tokens.

1.4 Proposed system

The touchless token generation system using hand gestures is particularly advantageous during a pandemic like COVID-19 due to its ability to address hygiene concerns and minimize the risk of virus transmission. By eliminating the need for physical contact, such as touching ticket machines or handling physical tokens, the system reduces the potential for contaminated surfaces and person-to-person contact.

Users can simply use hand gestures, gesturing towards a camera, to generate tokens. This touchless interaction promotes social distancing and reduces the likelihood of virus spread in crowded environments. It eliminates the need for customers to physically interact with shared surfaces, such as touchscreens or buttons, thereby minimizing the risk of cross-contamination.

Moreover, the touchless token generation system enhances the overall customer experience by providing a seamless and convenient process. Users can generate their tokens quickly and effortlessly through intuitive hand gestures, eliminating the need for physical queues or waiting in close proximity to others. This helps in reducing crowding and maintaining safe distances between individuals.

From an operational perspective, the touchless system improves efficiency by automating the token generation process. It eliminates manual tasks, reduces waiting times, and optimizes resource allocation. This streamlined approach contributes to a smoother and more organized flow of customers, which is particularly crucial during a pandemic when managing crowd control and minimizing congestion is essential.

To prevent the transmission of contagious diseases and uphold social distancing measures, it is crucial to develop a software system that can generate tokens. By implementing this solution, people can safely engage in necessary activities during future outbreaks resembling the COVID-19 pandemic.

Chapter 2

Literature Survey

The literature review is the comprehensive study and interpretation of literature that relates to a selected topic. When doing a literature review, research questions are defined, and then relevant literature is sought for and analyzed to address these issues. By reanalyzing the study's data, it is possible to acquire fresh insights, which is an advantage of literature reviews. A literature review is both a summary and an explanation of the complete and current state of information on a topic as contained in academic books and journal articles. There are two types of literature reviews you may be required to write in college: one is written as a stand-alone assignment in a course, while the other is done as an introduction to or preparation for a longer piece of writing, typically a thesis or research report. The primary objective and perspective of your review, as well as the hypothesis or thesis argument you develop, depend on the type of review you are writing. The distinctions between these two types by reading published literature reviews or the introductory chapters of theses and dissertations in your subject area can be learned. Note the framework of their arguments and the manner in which they approach the issues.

2.1 Purpose of the Literature Review

1. It chooses top-notch research papers or studies that are pertinent, significant, important, and valid and summarises them into a single comprehensive report to provide readers with quick access to information on a certain issue.
2. By requiring them to describe, assess, and compare original research in this particular field, it gives researchers who are starting their research in a new area a great place to start.

3. It makes sure that researchers don't repeat already completed studies.
4. It can indicate potential directions for future research or suggest topics to concentrate on.
5. It emphasises the important findings.
6. It points up gaps, discrepancies, and inconsistencies in the literature.
7. It offers a helpful critique of the methods and strategies used by other researchers.

2.2 Related Works

Various studies pertaining to hand gesture classification and computer vision are listed below:

Mehmood Nawaz et.al[1] proposes a new approach for hand gesture recognition using Electrical Impedance Tomography (EIT) images. EIT is a medical imaging technique that uses electrical measurements to produce images of the human body's internal structures. In this paper, the authors use EIT to capture images of hand gestures, which are then processed and classified using machine learning algorithms. The proposed approach is intended to be used in human-computer interaction applications, where users can interact with computers or other devices using hand gestures. The paper presents the results of experiments conducted on a dataset of EIT images, demonstrating the effectiveness of the proposed approach in accurately recognizing hand gestures. The authors discuss the potential of EIT-based hand gesture recognition systems and their applications in various fields, such as virtual reality, gaming, and rehabilitation.

Cheok Ming Jin et.al[2] describes a study that aims to address the difficulty that deaf and verbally-challenged individuals face daily by proposing a sign language translator on the smartphone platform. The proposed framework uses established image processing techniques to recognize images of several sign language gestures. The framework initially segments the hand gesture from its background using Canny edge detection and seeded region growing. Feature points are then extracted using the Speeded Up Robust Features (SURF) algorithm, and these features are derived through Bag of Features (BoF). The framework uses Support Vector Machine (SVM) to classify the gesture image dataset, which is then used to recognize future sign language gesture inputs. The proposed framework has been successfully implemented on smartphone platforms, and experimental results show that it is able to recognize and translate 16

different American Sign Language gestures with an overall accuracy of 97.13. This framework can help improve communication between deaf and hearing individuals and provide better accessibility for the deaf community.

Guangyu Jia et.al[3] proposes a method for accurately classifying electromyogram (EMG) signals that can be used in various applications such as prosthetic hand control, sign language, grasp recognition, and human-machine interaction. The proposed method combines unsupervised and supervised learning methods to classify a dataset consisting of 10 classes of hand gestures. Clustering methods, including subtractive clustering and modified fuzzy c-means (FCM) algorithms, are employed to obtain the initial partition of the inputs. Based on the grouping information obtained from clustering, a two-step supervised learning approach is proposed, which includes a top classifier and three sub-classifiers integrated with a windowing method and majority voting. The experimental results show that the proposed method achieves 100 test accuracy and strong robustness compared to conventional machine learning approaches. The proposed method has the potential for various industrial and healthcare applications such as movement intention detection, grasp recognition, and dexterous prostheses control.

Athira Devaraj et.al[4] proposed a method for hand gesture recognition using machine learning algorithms. The system uses an accelerometer and a gyroscope sensor mounted on a wristband to collect hand gesture data from the user. A total of 12 hand gestures were chosen for classification, including letters and numbers in the American Sign Language alphabet. The collected sensor data was preprocessed, and a feature set was extracted using time-domain and frequency-domain analysis. Several machine learning classifiers were evaluated for the classification task, including k-Nearest Neighbor, Support Vector Machine, Random Forest, and Naive Bayes. The experimental results showed that the Random Forest classifier performed the best, achieving an overall classification accuracy of 98.5. The proposed system can be used for human-computer interaction, virtual reality control, and assistive technology for people with disabilities.

U Sairam et.al[5] propose a solution for natural user interface (NUI) using hand gestures to control a computer mouse without touching it. The system design is based on TensorFlow and uses computer vision techniques such as object detection, object tracking, and gesture recognition to track hand movements captured through a webcam using RGB color scheme. The proposed system allows users to perform mouse operations like left click, right-click,

scroll, drag, and move by using different hand gestures. The implementation is done using Python, TensorFlow, and OpenCV libraries for real-time computer vision. The proposed system can be useful in situations where touch input is not desirable, such as during a pandemic or for people with disabilities.

Mithelan Devanandan et.al[6] provide solutions to automatically learn and practice cricket, especially using the advancement in computer vision technologies. To classify cricket shot images based on human body keypoints extracted using MediaPipe, the authors developed a Random Forest model. The experimental results demonstrated that the proposed model achieved an F1-score of 87 percent, outperforming the existing solution by a margin of 5 percent. The objective of this study is to classify a cricket shot image into one of six categories, namely cut, cover drive, straight drive, pull, leg glance, and scoop. To achieve this, the study utilizes MediaPipe to extract the posture details of the batsman, which are then used as features to train a Random Forest model. After predicting the shot type, the study employs a similarity estimation approach to compare the image with a collection of cricket images of international players, ultimately identifying the most similar image and associated similarity score. This study also creates a mobile application that enables individuals who play cricket to evaluate, enhance, and monitor their batting performance.

Safyzan Salim et.al[7] proposed a work that focuses on developing a machine learning-based sign language recognition system using MediaPipe and Google Teachable Machine. The system aims to simplify the data processing of acquired sensor data and improve the accuracy of gesture classification. The study involves experiments with different machine learning algorithms, data preprocessing techniques, and feature extraction methods. The results of the experiments demonstrate that the proposed system achieves high accuracy in detecting American Sign Language(ASL) gestures. This work has the potential to improve the development of wearable-based sign language recognition systems and enhance communication for individuals with hearing impairments.

Zhen Zhai et.al[8] describes a system that uses the Mediapipe framework to recognize hand gestures for telerehabilitation purposes. Telerehabilitation is a form of remote rehabilitation where patients can receive therapy at home through telecommunication technology. The system utilizes a camera to capture the hand gestures of the patient, and the Mediapipe framework processes the images to recognize the gestures. The recognized gestures are then used to control a virtual rehabilitation environment. The paper discusses the design and development of the

system, including the hardware and software components used. The authors also evaluate the performance of the system and compare it to other similar systems. Overall, the paper presents a promising approach for using hand gesture recognition to enhance telerehabilitation programs, making therapy more accessible and convenient for patients.

Ashish Sharma et.al[9] proposes a novel approach for recognizing and classifying American Sign Language (ASL) hand gestures using image analysis and classification techniques. The authors used a dataset of ASL images of a person's hand taken under different environmental conditions. The paper compares the performance of several image preprocessing techniques such as Histogram of Gradients, Principal Component Analysis, and Local Binary Patterns. The authors also proposed a novel approach that involves using Canny edge detection, ORB (Oriented FAST and Rotated BRIEF) feature extraction, and bag of word technique for image classification. To evaluate the performance of the proposed approach, the authors used several classifiers such as Random Forests, Support Vector Machines, Naïve Bayes, Logistic Regression, K-Nearest Neighbours, and Multilayer Perceptron. They compared the accuracy of their proposed model with existing models and found that their novel approach achieved significantly higher accuracy. The proposed novel approach shows promising results and outperforms existing models, demonstrating the importance of developing new and innovative techniques for solving real-world problems.

Raheja et.al[10] discusses the development of a sign language recognition system for Indian sign language, which is aimed at helping people who cannot speak or hear properly. The system is based on dynamic hand gesture recognition techniques and is designed to work in real-time scenarios. The process of sign language recognition involves capturing a video, which is then pre-processed by converting it to the HSV color space and segmenting it based on skin pixels. In addition, depth information is used to obtain more accurate results. Hu-Moments and motion trajectory are then extracted from the image frames, and the classification of gestures is done using Support Vector Machine (SVM). The system was tested using a webcam as well as the MS Kinect, and the results showed that it can accurately recognize Indian sign language gestures in real-time scenarios. Such a system can be helpful in teaching and communication for hearing-impaired persons. The paper highlights the importance of developing sign language recognition systems that can help people who cannot speak or hear properly. The proposed system uses dynamic hand gesture recognition techniques and is designed to work in real-time scenarios, making it a valuable tool for teaching and communication.

Isaiah Jassen Tupal et.al[11] describes the development of a vision-based hand-tracking system for non-face-to-face interaction in human-computer interaction (HCI). The system aims to improve HCI by allowing the user to track their hand movements, which will act as a pen, for creating texts, drawings, and other tasks. The system uses OpenCV and Mediapipe to track the user's hand movements using the computer's camera. The system provides a vision-based board where the user can draw on and the pen or marker will be the user's hand. The system is accurate enough to be a feasible application for handwriting and basic drawings, according to the results of the study. The paper highlights the potential of vision-based hand-tracking systems for non-face-to-face interaction in HCI. Such systems have the potential to provide a reusable writing surface for creating texts, drawings, and other tasks, which can improve the user experience and reduce the need for physical controllers such as a mouse or a keyboard.

Bidyut Jyoti Boruah et.al[12] describes the development of an interactive learning-aid tool based on a vision-based hand gesture recognition system. The system uses MediaPipe for hand gesture recognition and a virtual-mouse-based object-controlling system to control various virtual objects created using Unity. The proposed system has been tested using six hand gestures and it has been found that the system can be used effectively for controlling various virtual objects. The use of such technology-dependent tools in e-learning can increase communication and interaction between the teacher and the student. The paper highlights the potential of vision-based hand gesture recognition systems for interactive learning-aid tools. Such systems have the potential to enhance the learning experience by providing a natural and simple way of communicating and interacting with virtual objects.

Subhangi Adhikary et.al[13] propose a vision-based system for the recognition of words used in Indian Sign Language (ISL) using MediaPipe. Indian Sign Language is a form of communication used by the speech and hearing-impaired community in India, and it is based on gestures of the hands, arms, face, and head. The proposed system uses MediaPipe for hand detection, tracking, and recognition of hand gestures used in ISL a Random Forest Classifier for classification. The system aims to bridge the communication gap between the speech and hearing-impaired community and others. An accuracy of 97.4 was achieved in recognizing ISL signs. The results indicate that the integration of MediaPipe with machine learning algorithms can be effectively employed to correctly recognize signs of ISL.

Divyansh Bisht et.al[14] propose that Sign language has become more widespread, but communication between non-signers and mute/deaf individuals without a translator is still

challenging. Existing methods for Sign Language Recognition (SLR) use arm sensors or computer vision, but they either require additional wearable devices or are computationally intensive and less accurate. This paper proposes a vision-based, lightweight, web-based sign language interpretation system that allows two-way communication for all groups of people. The system uses Mediapipe to extract hand features and a random forest classifier for classification with an accuracy of 94.69. The model is trained on American Sign Language alphabets and includes text-to-speech, speech-to-text, and auto-correct features for ease of deployment and to support communication between deaf-mute, hard-of-hearing, visually impaired, and non-signers.

Priya K et.al[15] propose a hand landmark distance-based sign language recognition system using MediaPipe. The deaf and hard-of-hearing community often use sign language as a mode of communication with others. Research in sign language recognition has been ongoing, with both sensor-based and vision-based methods being developed. While sensor-based methods have traditionally been more accurate, vision-based methods are becoming more common due to their cost-effectiveness. This study focuses on recognizing sign language words using hand pictures captured by a web camera. The system extracts hand landmarks from the input image using MediaPipe and calculates the distances between landmarks. These distances are then used as features for training a machine learning model based on Random Forest Classifier. The proposed system is tested on a dataset of Indian Sign Language (ISL) gestures and achieves an accuracy of 91.9. The system also performs well on real-time gesture recognition using a webcam. The proposed system can be used as an alternative to traditional sensor-based methods for sign language recognition.

Chapter 3

Methodology

TOUCHLESS TOKEN GENERATION SYSTEM is a software solution developed using advanced computer vision technologies to generate tokens using hand gestures without physically touching any objects. The model is trained with different machine learning classification algorithms using a hand gesture dataset.

3.1 Algorithm

The model follows a set of procedures which includes:

1. Load the dataset
2. Processing of images
3. Building and training the model
4. Testing and deployment
 - Reading frames from the live feed
 - Tracking and drawing landmarks
 - Count Increment
 - Deploying the model

3.2 System Architecture

Touchless token generation system allows users to generate a token by making specific hand gestures. The system architecture of the touchless token generation system includes computer vision library that analyze the data captured from the camera to recognize specific hand gestures associated with generating tokens. The system also includes machine learning models that are trained to recognize a wide range of hand gestures and improve the system's accuracy over time. Once the hand gesture is recognized, the system generates a token.

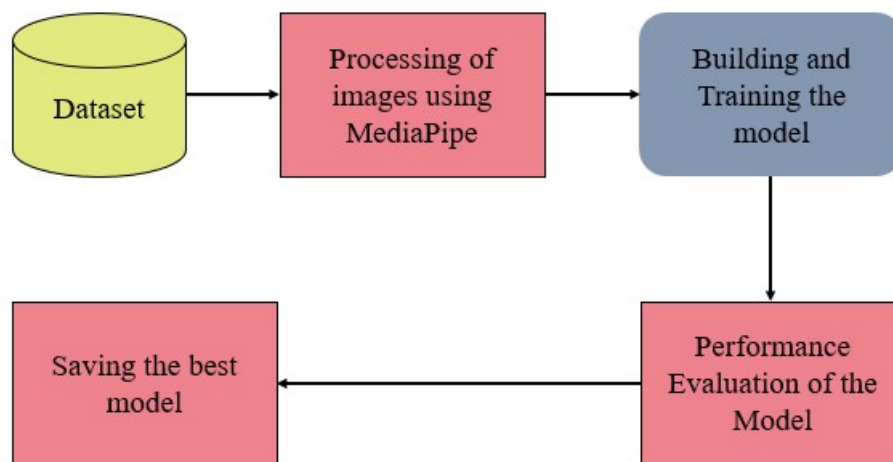


Figure 3.1: Training the system

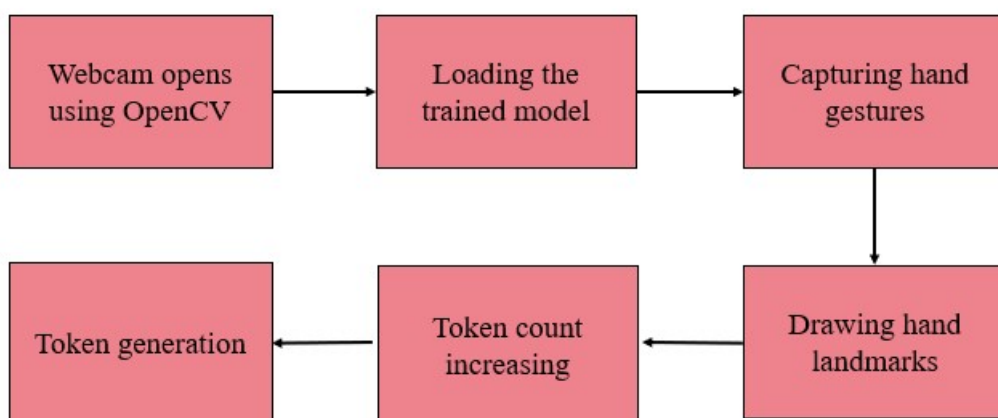


Figure 3.2: Deployment of the system

3.2.1 Dataset

The dataset used in the project is the Indian-Sign-Language dataset. ISL uses hand signs similar to British Sign Language and is similar to International Sign Language. It contains a total of 28,700 images. The dataset is classified into different classes with alphabets starting from A to Z and numbers 0-9.

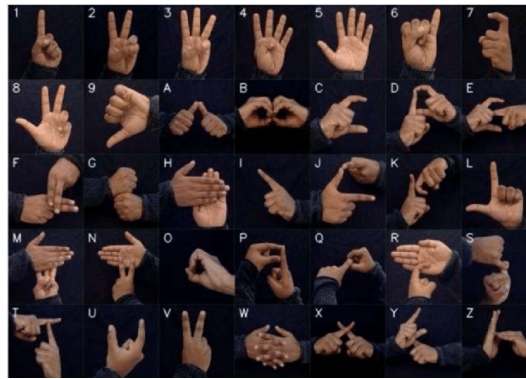


Figure 3.3: Indian Sign Language dataset

3.2.2 Image Processing

Image processing is essential in the hand gesture recognition system because it is the first step in detecting and extracting features from the image that represent the hand gestures. Preprocessing the image to improve its quality and make it easier to extract relevant features. This can involve various techniques, such as adjusting the contrast and brightness, removing noise, and resizing the image to a standard size.

The image processing steps performed here include converting the image from BGR to RGB format, flipping the image in the Y-axis, and then passing the processed image through the MediaPipe Hands solution. The MediaPipe Hands solution is then applied to the image to detect hand landmarks. Mediapipe has certain parameters such as static image mode, max num hands, and min detection confidence which helps to customize the behavior of the Mediapipe Hands model according to the specific requirements of the application. The Hands class from Mediapipe is initialized with the static image mode=True, max num hands=2, and min detection confidence=0.7. If the output contains landmarks, the function extracts them as a string and splits them into a list. It then removes any unnecessary data from the list and formats the remaining data as a list of floating-point numbers.

If the landmarks are successfully extracted, a function loops through each feature value and

writes it to the CSV file, separated by commas. After all the values have been written, the function writes the label for the hand gesture to the CSV file. The CSV file is used to train machine learning models for hand gesture recognition.

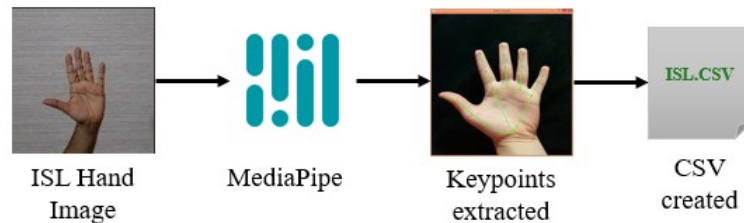


Figure 3.4: Image Processing using Mediapipe

3.2.3 Building and training the model

Training a touchless token generation system using hand gestures is possible with the use of computer vision techniques and machine learning algorithms. The system is trained using the CSV file that was created. Machine learning classifiers such as RandomForest, Decision Tree, NaiveBayes, K-Nearest Neighbor and Support Vector Machine(SVM) are used to train the model. Using the 'train test split' the dataset is split into two separate sets - a training set and a testing set.

The RandomForestClassifier object is created with the criterion='entropy' parameter which specifies that the decision trees in the random forest should use the entropy criterion to measure the quality of a split. The next step is to train the model using the training data. During training, the model builds multiple decision trees based on random subsets of the training data and then combines their predictions to make a final classification.

In the second model, the decision tree is being trained with the 'entropy' criterion, which measures the amount of disorder in the data and seeks to create splits that maximize the information gain at each step. The random state parameter is set to 'zero' to ensure that the same random numbers are generated every time the code is run, resulting in reproducible results. After training the model using the training data, its performance is evaluated on the test data to determine how well it can generalize to new, unseen data.

In the third one, a new instance of the Gaussian Naive Bayes(GNB) class is created, which specifies that the input features are continuous variables that follow a Gaussian (normal) distribution. The GNB classifier is then trained using the training data. During training, the

algorithm estimates the mean and variance of each input feature for each class label in the training set. These estimates are then used to calculate the likelihood of a new data point belonging to each class label using Bayes' theorem. The GNB classifier assumes that the input features are independent of each other given the class label, which is a simplifying assumption known as the "naive Bayes" assumption. After training the model, its performance is evaluated on the test data.

In the next model, an SVM classifier is trained on the training set using the SVC class. The SVC class provides a range of kernels for SVM, including the radial basis function (RBF) kernel, which is used in this case. The RBF kernel is a commonly used kernel in SVM that is effective in capturing non-linear relationships in the data. Two important hyperparameters that need to be set for the SVM classifier are C and gamma. The C parameter controls the trade-off between achieving a low training error and a low testing error by regulating the extent to which the SVM model will allow the margins to be violated by the training data. C is set to 50, which indicates a relatively low tolerance for misclassification. The gamma parameter controls the shape of the decision boundary and the extent to which a single training example influences the decision boundary. gamma is set to 0.1, which indicates a relatively small influence of individual training examples on the decision boundary. After configuring the SVM classifier, the fit method is called to train the model on the training data.

In the fifth model, a KNN classifier is trained on the training data and evaluated on the testing data. The number of neighbors (n neighbors) is set to 5, which means that the algorithm will consider the 5 nearest data points when making predictions. The 'Minkowski' metric is used to measure the distance between data points. The power parameter is set to 2, which means that the Euclidean distance is used. The KNN classifier is trained on the training data by storing it in memory and computing the distances between the new data points and the training data. The K nearest neighbors are then selected, and their labels are used to make predictions for the new data points. Finally, the performance of the KNN classifier is evaluated on the testing data.

The model with the highest testing accuracy is selected and saved to use it later to test and deploy the system. The dump function from 'joblib' is used to save the trained classifier object into a pickle file.

Random Forest

Random Forest(RF) is an ensemble learning algorithm that is used for both classification and regression tasks. In Random Forest, a large number of decision trees are trained on random subsets of the training data, using a technique called bagging (bootstrap aggregating). Each decision tree in the forest is trained independently on a randomly sampled subset of the training data, with replacement. This means that some data points may appear in multiple subsets, while others may be excluded from some subsets. Each tree is constructed by repeatedly splitting the data based on the values of a randomly selected subset of the features until the data is divided into subsets that are as pure as possible.

To make a prediction on a new, unseen data point, the prediction of each decision tree in the forest is combined to produce a final prediction. In classification tasks, the predicted label is the mode label predicted by the individual trees. Random Forest has several advantages over other machine learning algorithms. It is less prone to overfitting than single decision trees because the ensemble of trees reduces the variance of the model.

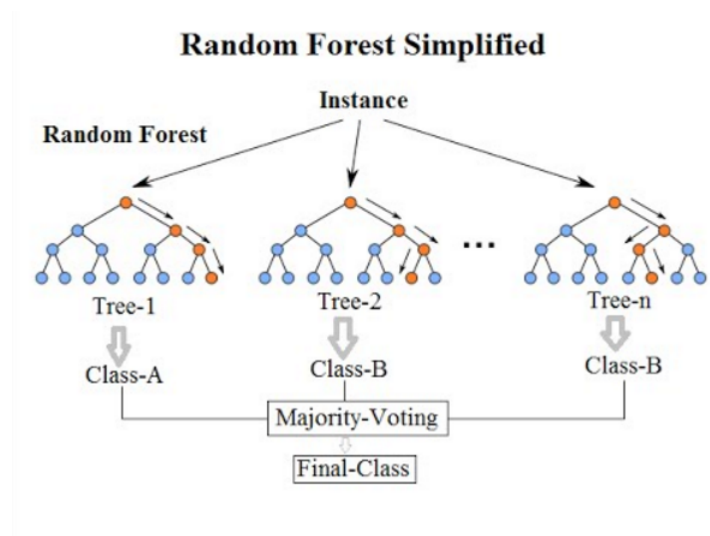


Figure 3.5: RandomForest

Decision Tree

A Decision Tree (DT) is a popular machine learning algorithm that is widely used for both classification and regression tasks. The basic idea behind a decision tree is to recursively split the data into subsets based on the most significant features that can best separate the classes or predict the target variable.

In a classification problem, the decision tree algorithm selects the feature that best separates the classes using a statistical measure such as entropy or Gini impurity. It then splits the data based on that feature, creating a new node in the tree. The splits result in a tree-like structure where each internal node represents a test on an attribute, each branch represents the outcome of a test, and each leaf node represents a class label or a numerical value. This process is repeated for each subset of data until a stopping criterion is reached, such as a maximum depth or a minimum number of samples per leaf.

The decision tree algorithm is known for its interpretability and ease of use. It allows the user to visually interpret and understand the underlying decision-making process. Decision trees can also handle both categorical and numerical data, as well as missing values, making it a versatile algorithm. However, decision trees are prone to overfitting, where the model fits too closely to the training data, resulting in poor generalization to new data. Different methods, such as pruning and setting a minimum number of samples required to split, can be used to prevent overfitting.

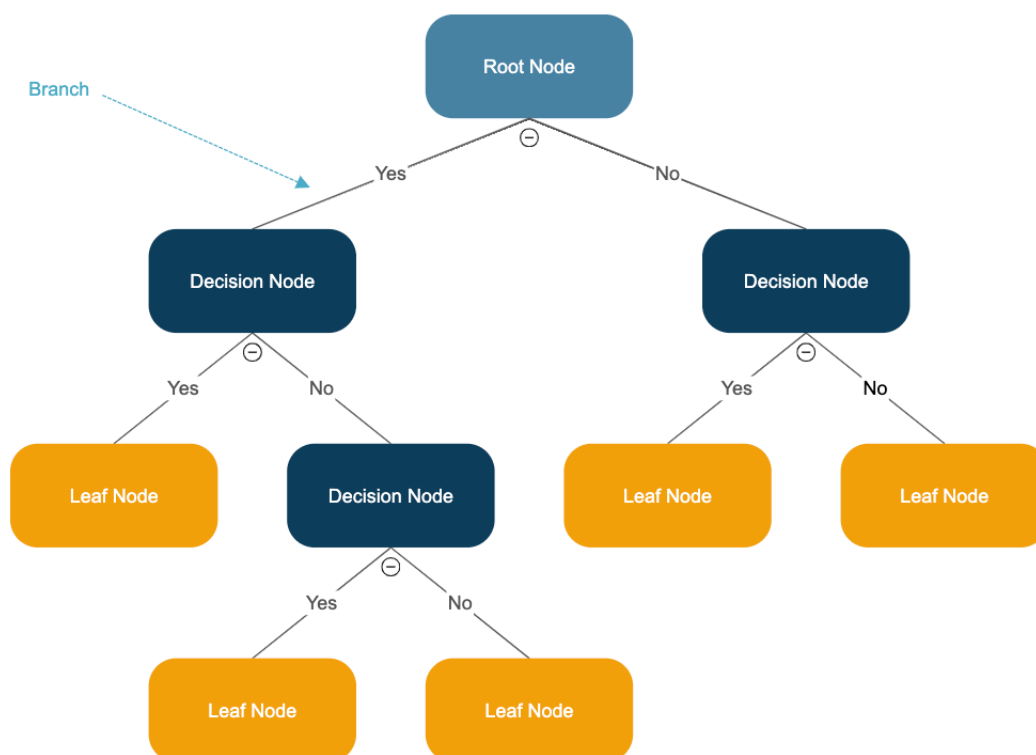


Figure 3.6: Decision Tree

NaiveBayes

Naive Bayes is a probabilistic machine learning algorithm used for classification tasks. The basic idea behind the algorithm is to use Bayes' theorem to compute the probability of each class given the input features and select the class with the highest probability. Naive Bayes assumes that the features are independent of each other given the class, hence the "naive" assumption. This assumption simplifies the computation of probabilities and allows the algorithm to work well even with high-dimensional data. Gaussian distribution, also known as the normal distribution, is a continuous probability distribution that is widely used in statistics and machine learning.

The algorithm starts by computing the prior probability of each class, which is the proportion of data points in the training set that belong to each class. It then computes the likelihood of each feature given each class, which is the probability of observing that feature given the class. This can be computed using different probability distributions, such as the Gaussian distribution for continuous data or the multinomial distribution for discrete data. Once the prior probability and likelihoods are computed, the algorithm applies Bayes' theorem to compute the posterior probability of each class given the input features. The class with the highest posterior probability is selected as the predicted class for the input data.

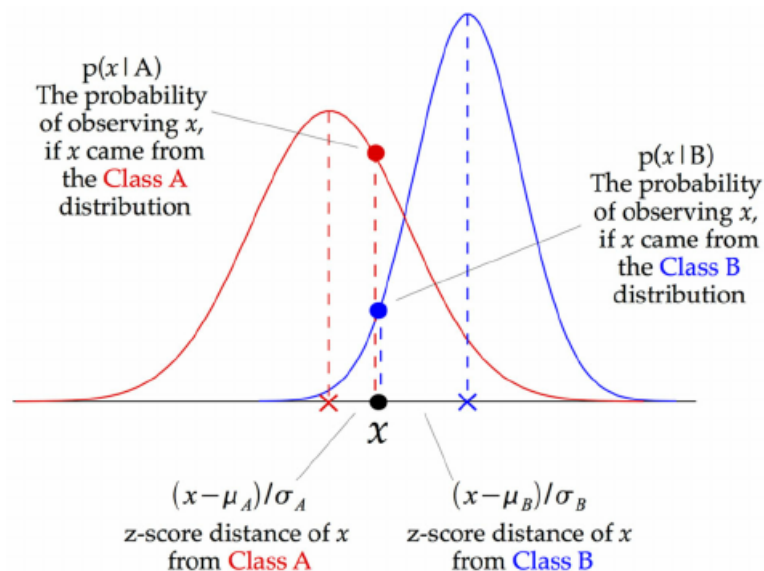


Figure 3.7: Naive Bayes

Support Vector Machine

SVM stands for Support Vector Machines, which is a powerful and versatile machine learning algorithm used for both classification and regression tasks. SVM is particularly useful when dealing with high-dimensional data or when the number of features is greater than the number of samples. The basic idea behind SVM is to find the optimal hyperplane that can separate the different classes of data points in the feature space. The hyperplane is a line that separates the data points of one class from another. The SVM algorithm finds the hyperplane that maximizes the margin, which is the distance between the hyperplane and the nearest data points of each class. This margin maximization ensures that the classifier is more robust and generalizes better to new data points.

SVM also has the ability to handle non-linearly separable data by using kernel functions to transform the input data into a higher-dimensional space where the classes are linearly separable. While using a kernel function to transform the feature space into a higher-dimensional space a hyperplane can be found. Common kernel functions used in SVM include the linear kernel, polynomial kernel, and radial basis function (RBF) kernel. The SVM algorithm is also capable of handling multi-class classification problems by combining multiple binary SVM classifiers. In addition to classification, SVM can be used for regression tasks by predicting a continuous output variable instead of a binary class label. SVMs can be effective for multi-class classification problems, particularly when combined with appropriate techniques such as the "one-vs-all" or "one-vs-one" approach.

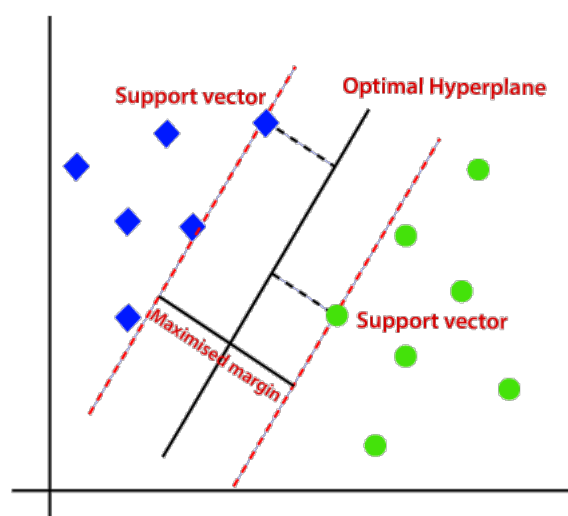


Figure 3.8: Support Vector machine

K-Nearest Neighbor

K-Nearest Neighbors (KNN) is a machine learning algorithm used for both classification and regression tasks. The basic idea behind the algorithm is to predict the class or value of a new data point based on the classes or values of its nearest neighbors in the training set.

In KNN classification, the algorithm finds the K nearest neighbors of the new data point in the training set based on a distance metric such as Euclidean distance or Manhattan distance. 'Minkowski' distance is a commonly used metric to calculate the distance between two data points. Minkowski distance is a generalized form of Euclidean distance and Manhattan distance. It takes a parameter ' p ', which determines the order of the distance metric. When $p=1$, it represents the Manhattan distance and when $p=2$, it represents the Euclidean distance. The class of the new data point is then assigned based on the majority class among its K nearest neighbors. K is a positive integer that represents the number of neighbors to consider. The value of K is a hyperparameter that can be chosen based on the problem and the size of the dataset.

The KNN algorithm has some advantages over other algorithms. Firstly, it is simple to implement and easy to understand. Secondly, it is a versatile algorithm that can be used for both classification and regression problems. Thirdly, it can handle multi-class problems with ease. Lastly, the KNN algorithm can be used for problems where the data distribution is unknown or where the decision boundary is very irregular.

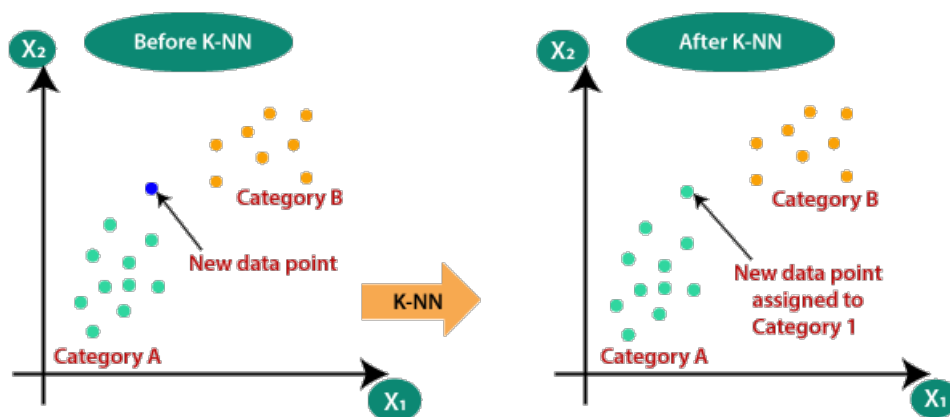


Figure 3.9: KNN

3.2.4 Testing and Deployment

To test and deploy the system, the saved model is loaded into the code, and frames are read from the live feed using OpenCV. Landmarks are tracked and drawn using Mediapipe, and the count is incremented accordingly.

1. Reading frames from the live feed

Live video frames are read from the default system webcam using OpenCV's 'VideoCapture' function. Once the frames are captured, the images undergo several image manipulations and processing operations. Firstly, the color space of the image is converted from RGB to BGR using the 'cv2.cvtColor()' function. This conversion is necessary because OpenCV uses BGR color space by default. Secondly, the image is flipped horizontally using the 'cv2.flip()' function. This operation is done to correct the orientation of the image so that it matches what the user would expect to see. Overall, these operations prepare the image data for further processing or analysis.

OpenCV

OpenCV (Open Source Computer Vision) is an open-source computer vision and machine learning library that provides a wide range of algorithms for image and video processing. OpenCV is written in C++ and has interfaces for several programming languages, including Python and Java. OpenCV provides various functionalities such as image and video capturing, image processing, feature detection, object detection, and machine learning. It can be used for a variety of applications such as face recognition, object tracking, motion detection, augmented reality, and robotics. OpenCV has become a popular library due to its ease of use, high performance, and flexibility, making it suitable for both research and commercial applications.

2. Tracking and drawing hand landmarks

Mediapipe is used to track and store landmarks from the captured live video frames. The detected hand landmarks are then visualized on the image using 'mp.drawing.draw_landmarks()' function provided by MediaPipe. The while loop is used to keep the video capture running until it is stopped or an error occurs. These landmarks are then extracted and processed using the same method that was used to train the model. Once the data is cleaned, the pre-trained

model is loaded into the code using 'joblib.load()'. The cleaned hand landmarks data is passed into the model, and the model is used to predict the gesture label. Overall, the landmarks tracked by Mediapipe are used to make predictions using a pre-trained model.

Mediapipe

Mediapipe is an open-source, cross-platform framework used for building multi-modal machine learning pipelines for mobile, edge, and cloud devices. It was developed by Google and is used for developing pipelines for different use cases. Mediapipe provides a set of pre-built models for different tasks, including hand tracking, object detection, face detection, and more. It provides an easy-to-use and highly customizable framework for building computer vision and machine learning pipelines, which are highly optimized for performance on various platforms. It consists of a set of pre-built, reusable, and highly optimized building blocks for different tasks, such as face detection, hand tracking, object detection, and more.

One of the most popular and widely used features of Mediapipe is hand tracking. The hand tracking pipeline in Mediapipe takes as input a video or image stream and produces a sequence of hand landmarks for each detected hand in the input frame. These landmarks represent the key points on the hand, such as the fingertips, palm, wrist, and joints between the fingers. MediaPipe returns a total of 21 key points for each detected hand.

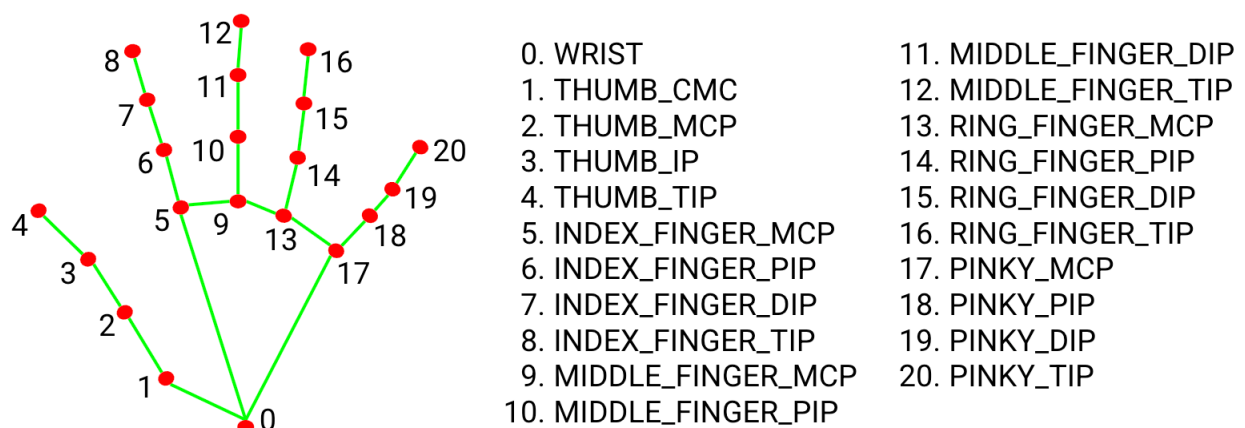


Figure 3.10: Hand landmarks identified by Mediapipe

Mediapipe hand tracking pipeline consists of multiple stages that are optimized for real-time processing on mobile and edge devices. These stages include:

1. Hand landmarks estimation: Once a hand is detected, the hand landmarks estimation stage is used to estimate the location of different landmarks on the hand.

2. Hand tracking: The hand tracking stage is used to track the hand landmarks over time as the hand moves in the input video or image stream.

3.Count Increment

The number of times a certain gesture has been made by the user within a certain time period is counted. If the model predicts the same gesture for 40 consecutive frames, a ticket is generated and displayed. This is done by initializing a counter and setting a time window during which the counter will increase whenever the predicted gesture is the same as the previous prediction. When the counter reaches a certain threshold, a ticket is generated and the process starts again. This allows the system to recognize when a certain gesture has been made repeatedly and trigger a response, such as generating a ticket.

4.Deploying the model

The model is deployed using flask. An interface is created where the users are required to show specific hand gestures of their choice and the token is displayed.

3.3 System Specifications

The application development architecture recognized for this project is specified in this section on the basis of requirements.

3.3.1 Software Specification

The software used for the project includes:

1. Python
2. Google Colaboratory
3. Flask

3.3.2 Software Description

1. Python

Python is a high-level, interpreted programming language known for its simple syntax and readability. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages in the world. Python has a wide range of applications, including web development, scientific computing, data analysis, artificial intelligence, machine learning, and automation. It is also known for its large standard library, which includes modules for a variety of tasks such as string processing, regular expressions, and file I/O. When a Python script is run, the interpreter evaluates the code and converts it into machine-readable bytecode. Python is an object-oriented programming language, meaning it allows users to handle and manipulate objects or data structures to create and run programs. Python is a reliable and popular language, constantly evolving to meet the needs of developers.

Features:

- Interpreted: Python is an interpreted language, meaning it does not require a compilation step before execution.
- Dynamically-typed: Python is dynamically-typed, which means that variable types are inferred at runtime, rather than being specified in the code.

- Object-oriented: Python is an object-oriented language, meaning it supports encapsulation, inheritance, and polymorphism.
- Easy-to-learn: Python has a simple and intuitive syntax, making it easy to learn for beginners.

2. Google Colaboratory

Google Colaboratory, also known as "Colab," is a cloud-based platform developed by Google that allows users to write, run, and share Python code via a Jupyter notebook interface. It is designed to provide a free and easy-to-use environment for data analysis, machine learning, and artificial intelligence tasks, with access to powerful hardware resources such as GPUs and TPUs. Colab provides a range of pre-installed libraries and frameworks commonly used in data science and machine learning, such as NumPy, Pandas, TensorFlow, and MediaPipe, and also supports the use of external libraries and packages. Users can store and access their code and data files on Google Drive, and collaborate with others by sharing notebooks and commenting on code. Additionally, Colab allows users to run code on remote servers, making it useful for tasks that require large amounts of computing power or memory.

Features:

- Free access to a virtual machine with a pre-installed environment for data analysis and machine learning.
- Ability to write and execute Python code in a Jupyter Notebook-style environment, which allows for interactive coding, documentation, and visualization.
- Access to Google's high-performance hardware, such as GPUs and TPUs, which can significantly speed up machine learning tasks and reduce training time.
- Integration with Google Drive, which allows for easy sharing and collaboration on projects with others.

3. Flask

Flask is a micro web framework written in Python. It is designed to be lightweight and modular, providing developers with the flexibility to customize and extend it to meet their specific needs. Flask includes built-in support for creating web applications, including handling HTTP requests and responses, routing, and templating. Its simplicity and ease

of use make it a popular choice for building small to medium-sized web applications, APIs, and prototypes. Flask can also be easily integrated with other Python libraries and frameworks, making it a versatile tool for web development.

Features:

- Easy to get started: Flask is designed to be easy to learn and use, making it a great choice for beginners who want to build web applications quickly.
- Lightweight and flexible: Flask is a micro-framework, which means it is lightweight and flexible, allowing developers to add only the features they need for their applications.
- Built-in development server: Flask comes with a built-in development server that makes it easy to test and debug your application.
- Modular design: Flask is designed to be modular, with a core framework that provides basic functionality, and extensions that can be used to add additional features and functionality.

3.4 Hardware and experimental environment

The hardware used for the experiments includes Windows 11 Pro OS, 64-bit operating system, x64-based processor, AMD Ryzen 3 3250U with Radeon Graphics @2.60 GHz, 8 GB RAM. The experimental environment was prepared by using Python 3.10 programming language. The framework used is Keras with TensorFlow and Mediapipe.

Chapter 4

RESULT AND DISCUSSION

A touchless token generation system using hand gestures is a technology that allows users to generate tokens without physical contact or the use of traditional methods such as token dispenser. Instead, this system uses hand gestures to generate a unique token.

Images from the dataset are read and processed using Mediapipe to access the hand landmark of the hand images. It provides the processed hand landmark data as a list of floating-point numbers or a zero-filled array if no hand landmarks were detected in the image. A total of 21 key points are returned for each detected hand image in each folder. The extracted landmarks are written into a CSV file with 63 columns. Each row in the CSV file represents an instance, and each column corresponds to a specific landmark coordinate.

Using the file, the model is trained using different machine learning classification algorithms- RandomForest, DecisionTree, NaiveBayes, SVM, and KNN. As the SVM model has demonstrated high efficiency, it has been saved and utilized for testing purposes. To validate the prediction accuracy, the real-time data is inputted and classified, and upon successful classification, the token count is incremented.

4.1 Training and Validation Results

Training and validating results of classification machine learning algorithms involve assessing the performance and accuracy of the trained models on both the training data and unseen validation data. The training and validation phases are crucial steps in the machine learning workflow to assess the performance and generalization ability of classification algorithms. The initial model utilized in the analysis is a RandomForest Classifier, which achieved a testing

accuracy of 92.25%. Subsequently, a DecisionTree Classifier was employed and yielded an accuracy of 90.80%. The third classification model employed was NaiveBayes, which achieved an accuracy of 84.00%. In the subsequent investigation with an SVM classifier, an impressive accuracy of 98.47% was attained. Lastly, the KNN classifier demonstrated a notable accuracy rate of 97.94%. The SVM classifier showcased the highest accuracy rate of 98.47%, hence, the model was saved in pickle format for further process.

The previous study [2] shows that it is able to recognize and translate 16 different Sign Language gestures with an overall accuracy of 97.13% using SVM. The accuracy analysis indicates that the SVM model in this study surpasses the SVM model described in [2] with an exceptional accuracy of 98.47%, the model proves to be highly accurate in recognizing and translating sign language gestures more. The research [6] demonstrates an 87% accuracy in image recognition when employing the RandomForest algorithm. However, this study reports a higher recognition rate of 92.25% for image recognition using the same algorithm.

In this study, the proposed method outperformed the SVM by 1.34% and random forest by 5.25% in accurately recognizing and classifying images comparing the previous ones[2,4]. The results indicate that the new approach offers enhanced performance and accuracy in the field of image recognition.

The following table shows the training score, testing score, and f1 score of the different machine learning classifiers used in the respective study.

SL.NO	Model	Training_accuracy	Testing_accuracy	F1_score
1	Random Forest	0.926638	0.922572	0.922572
2	Decision Tree	0.921056	0.908043	0.908043
3	Naïve Bayes	0.851193	0.840007	0.840007
4	SVM	0.986355	0.984762	0.984762
5	KNN	0.980684	0.979447	0.979447

Figure 4.1: Comparison of models

4.1.1 Training and Validation Accuracy Graphs

Bar plots can be used to visually represent training and testing accuracy. The training accuracy graph shows the performance of the model on the training data, while the testing accuracy graph indicates the model's performance on a separate set of testing data.

The following plots provide a concise representation of the performance of different algorithms used.

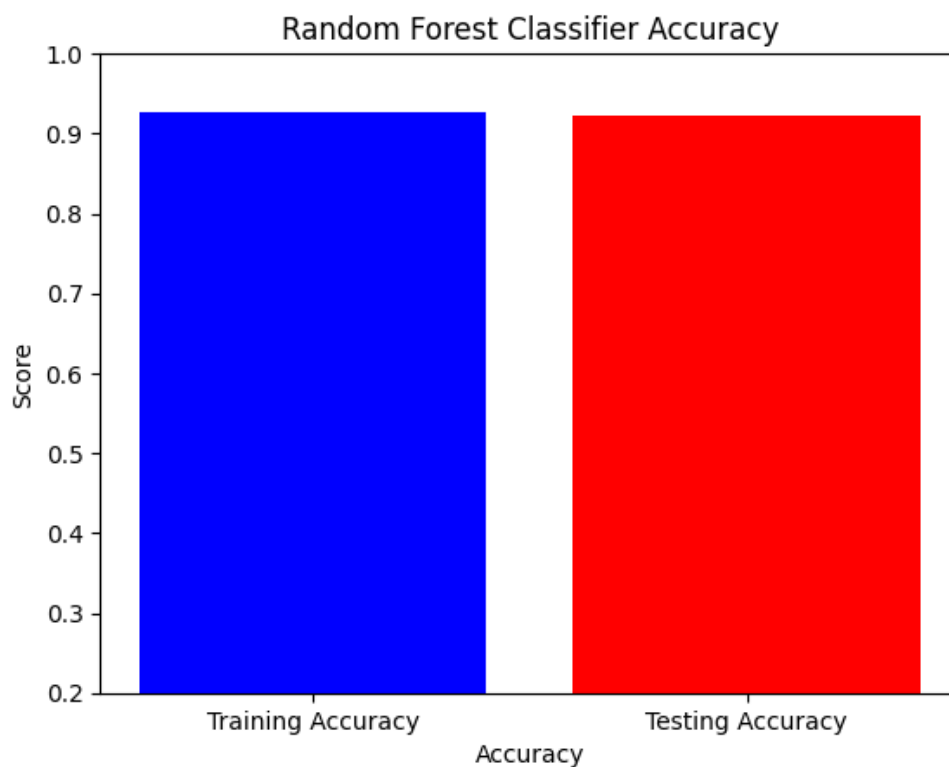


Figure 4.2: Training and Testing Accuracy of Random Forest

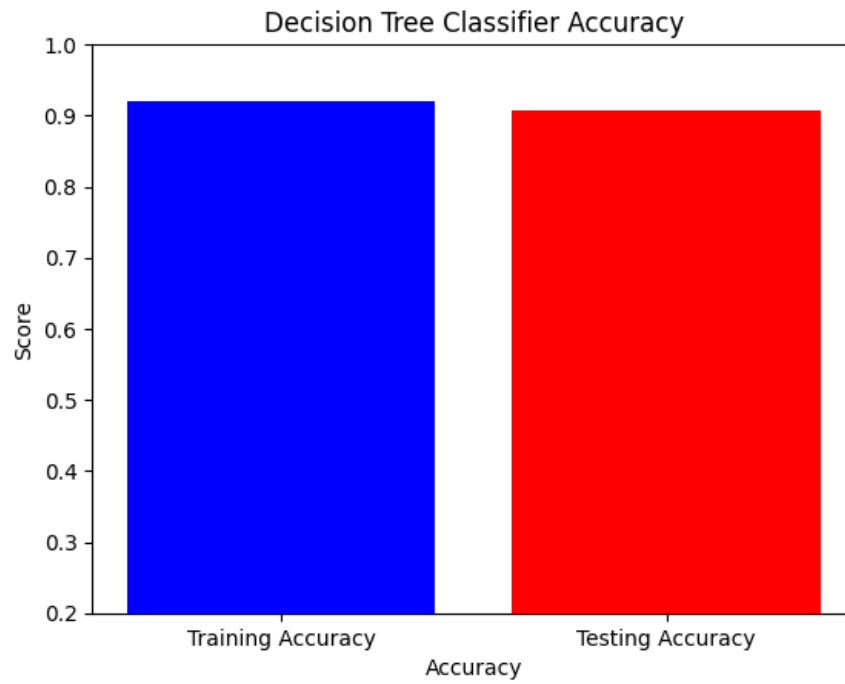


Figure 4.3: Training and Testing Accuracy of Decision Tree

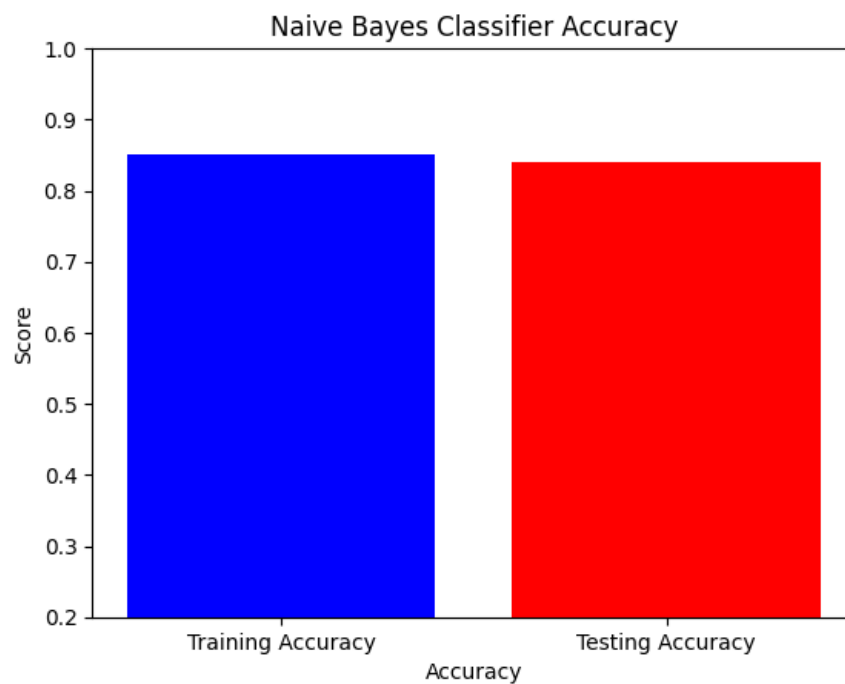


Figure 4.4: Training and Testing Accuracy of Naive Bayes

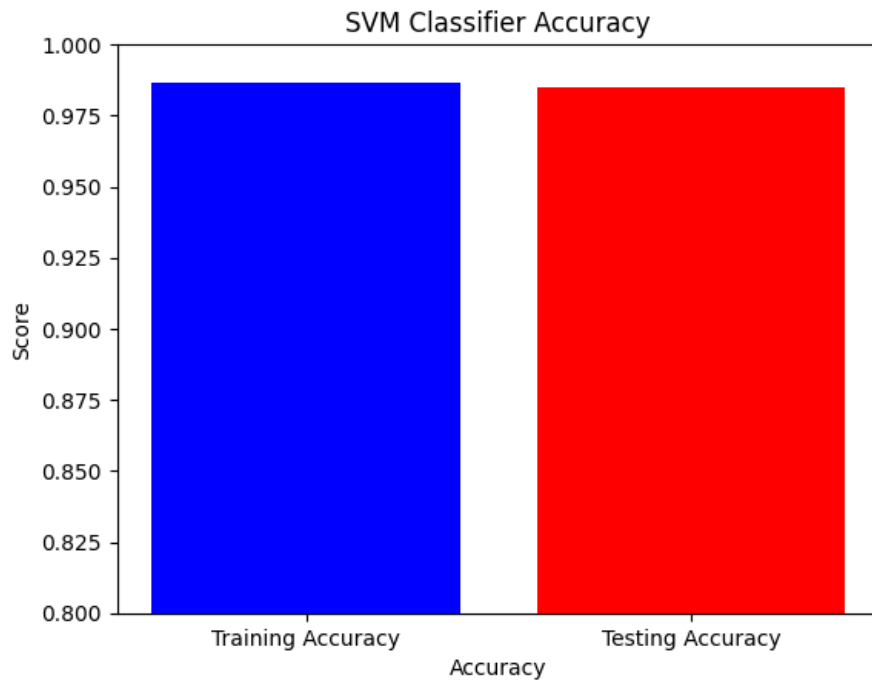


Figure 4.5: Training and Testing Accuracy of SVM

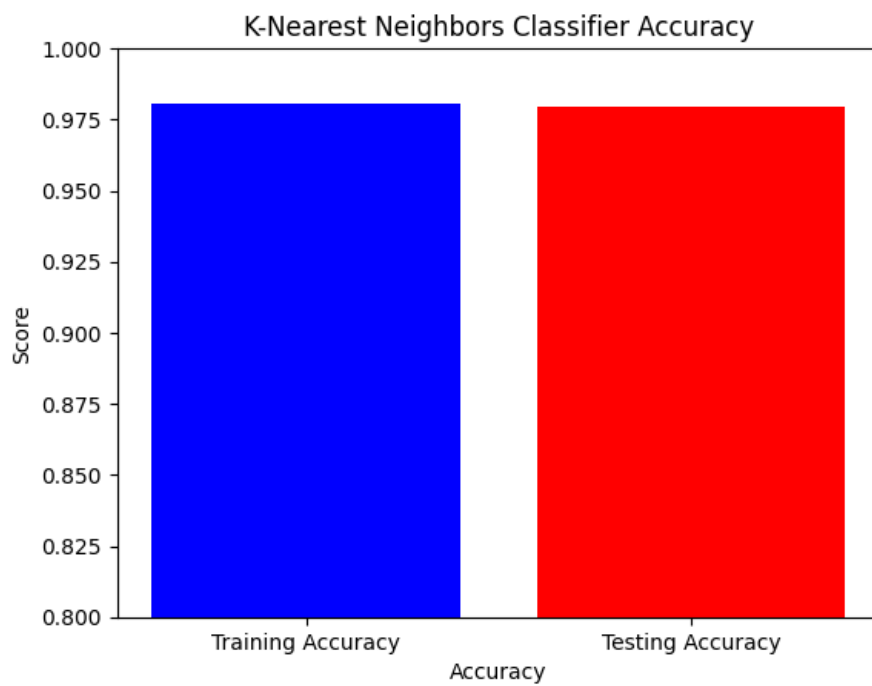


Figure 4.6: Training and Testing Accuracy of KNN

4.1.2 Graph of Comparison of the models used.

1. Bar Plot

The bar plot presents a visual comparison of various machine learning classification algorithms. Each bar represents the performance of a specific algorithm based on a chosen evaluation metric or metrics. The height of each bar reflects the metric's value, indicating the algorithm's relative effectiveness. By visually contrasting the bars, one can easily determine which algorithm performs better or worse in terms of the chosen metrics. It serves as a valuable tool for evaluating and selecting the most appropriate machine learning classification algorithm.

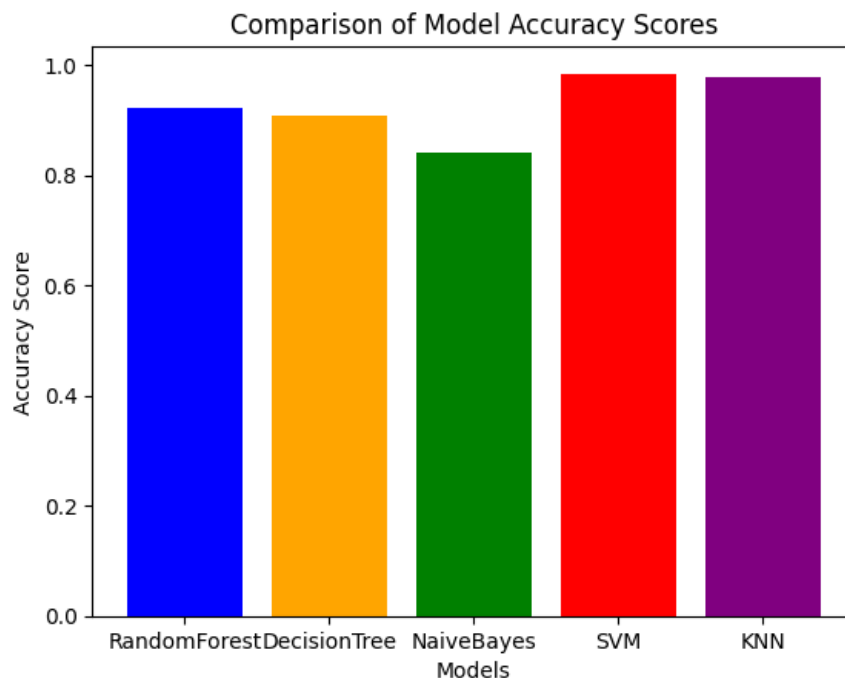


Figure 4.7: Comparison of Testing accuracy of models

2. Heatmap

The heatmap provides a graphical representation of the performance comparison among different machine learning classification algorithms. It visualizes the evaluation metrics across the algorithms in a color-coded manner, allowing for easy identification of patterns and variations in performance. In the heatmap, each cell represents the performance of a specific algorithm on a particular evaluation metric. Each cell's color intensity or shading indicates the metric's magnitude or value, with darker or more intense colors typically representing better performance. In the heatmap, each cell represents the performance

of a specific algorithm on a particular evaluation metric. Each cell's color intensity or shading indicates the metric's magnitude or value, with darker or more intense colors typically representing better performance.

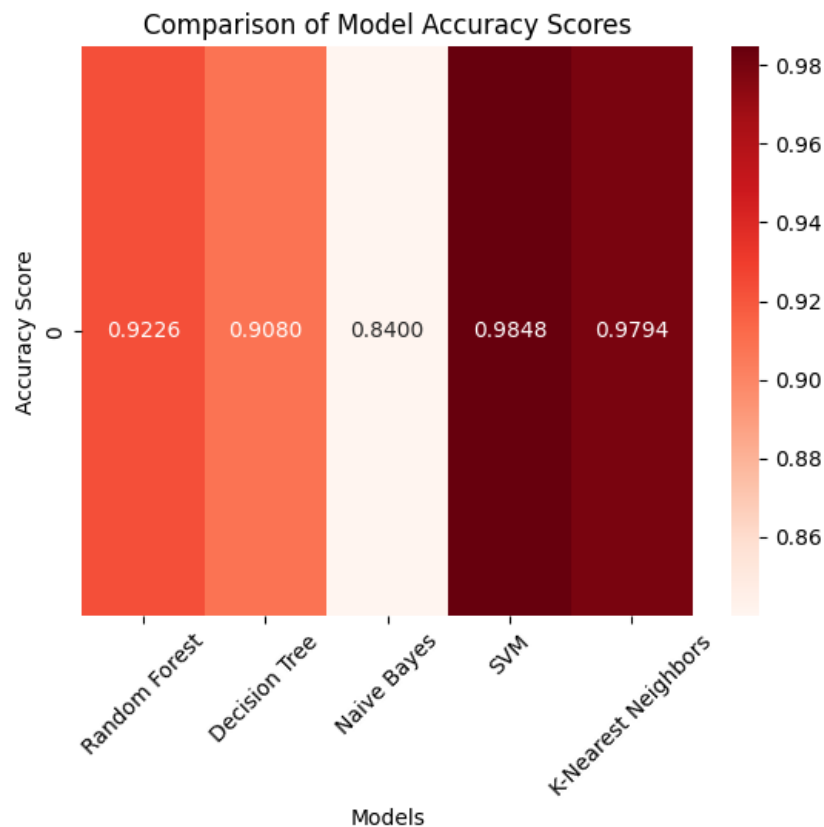


Figure 4.8: Comparison of Testing accuracy of models

3. Line Plot

A line graph—also known as a line plot or a line chart—is a graph that uses lines to connect individual data points. A line graph displays quantitative values over a specified time interval. A line plot can be used to visualize the accuracy scores of different classification algorithms. Each algorithm is represented on the x-axis, and the corresponding accuracy score is plotted on the y-axis. In a line plot, each data point is represented by a marker or symbol, and these points are connected by lines to show the overall trend or pattern of the data. The line segments visually represent how the values change or progress as the independent variable varies. The line graph connects the data points, showing the trend or progression of accuracy scores across the different algorithms. Line plots provide a concise and intuitive way to visualize data and understand the relationship between variables in a graphical manner. They can reveal

how the performance metric changes with respect to different factors. They provide a clear overview of how various models perform and assist in choosing the most suitable one for a specific task.

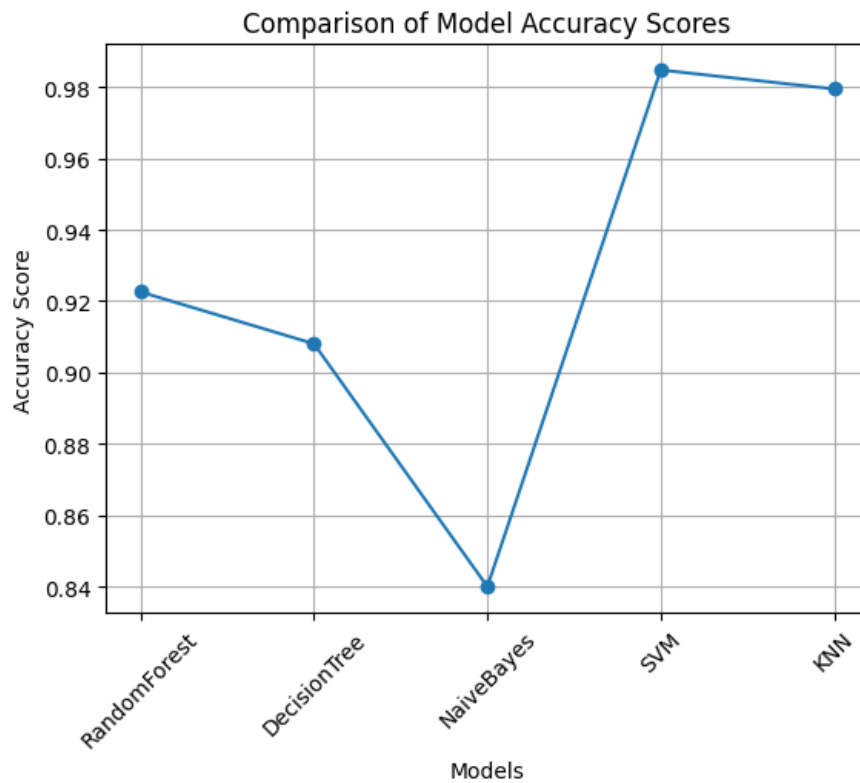


Figure 4.9: Comparison of Testing accuracy of models

4.2 Performance Metrics for Validation Phase

Performance metrics for the validation phase are used to evaluate the effectiveness and generalization of a machine learning model during the validation process. These metrics provide insights into how well the model is performing on unseen data and help assess its ability to make accurate predictions in real-world scenarios. These performance metrics collectively help assess the strengths and weaknesses of the model during the validation phase.

4.2.1 Confusion Matrix

A confusion matrix provides a tabular representation of predicted labels against actual labels. It shows the number of true positives, true negatives, false positives, and false negatives. From the confusion matrix, various metrics like accuracy, precision, and recall can be derived.

The provided figures illustrate the confusion matrices of the utilized models.

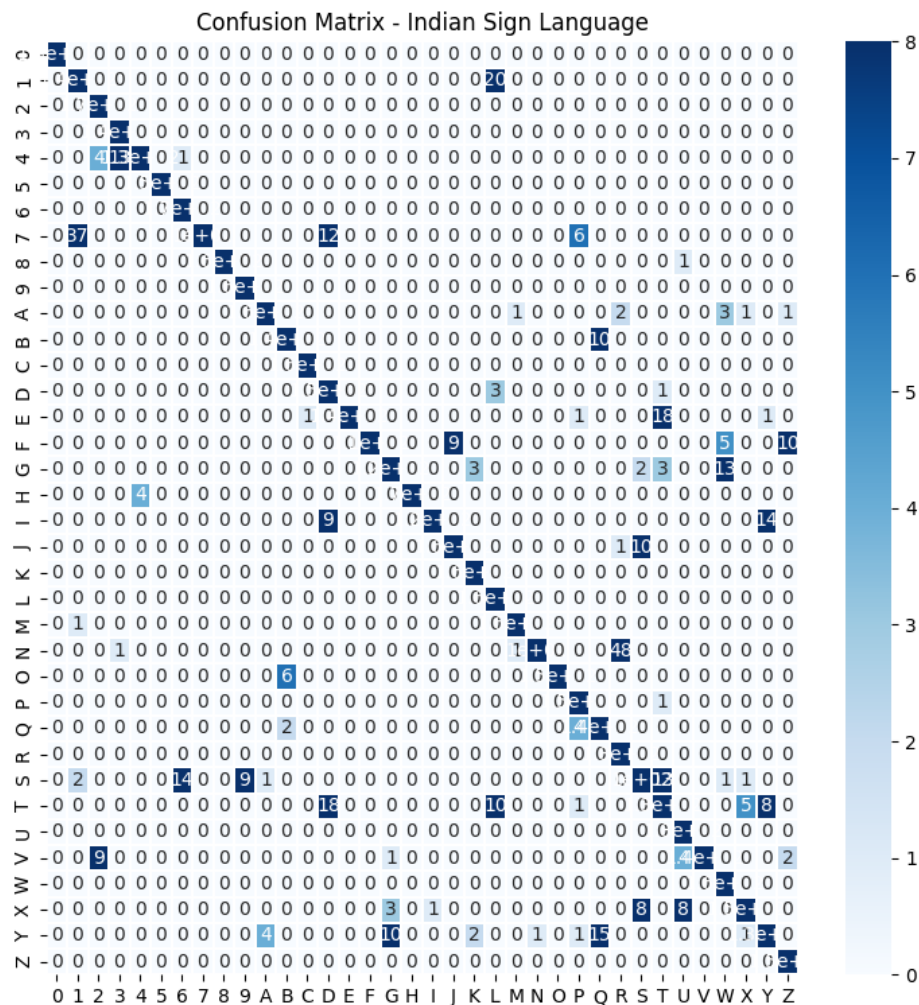


Figure 4.10: Confusion Matrix of Random Forest

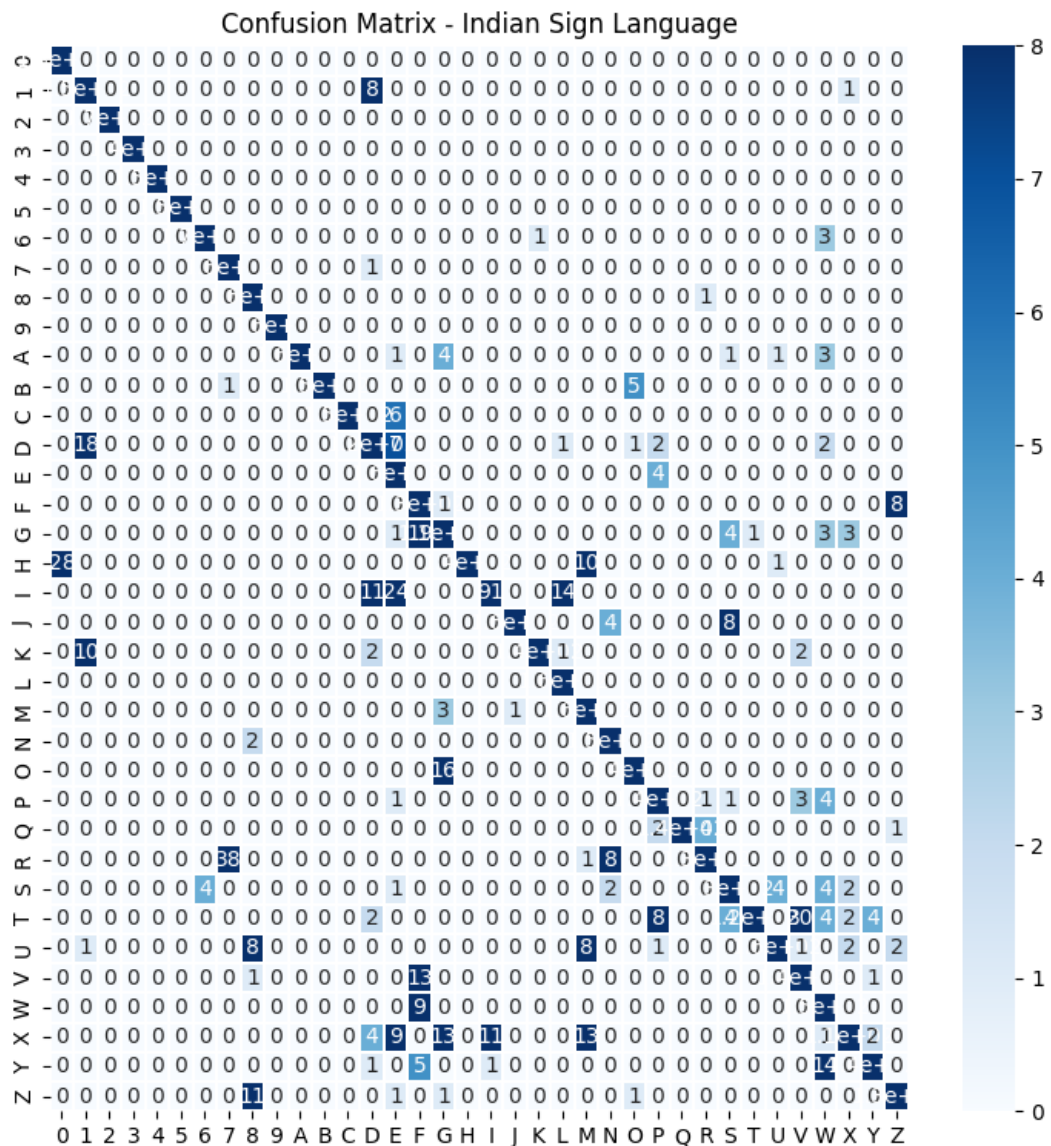


Figure 4.11: Confusion Matrix of Decision Tree

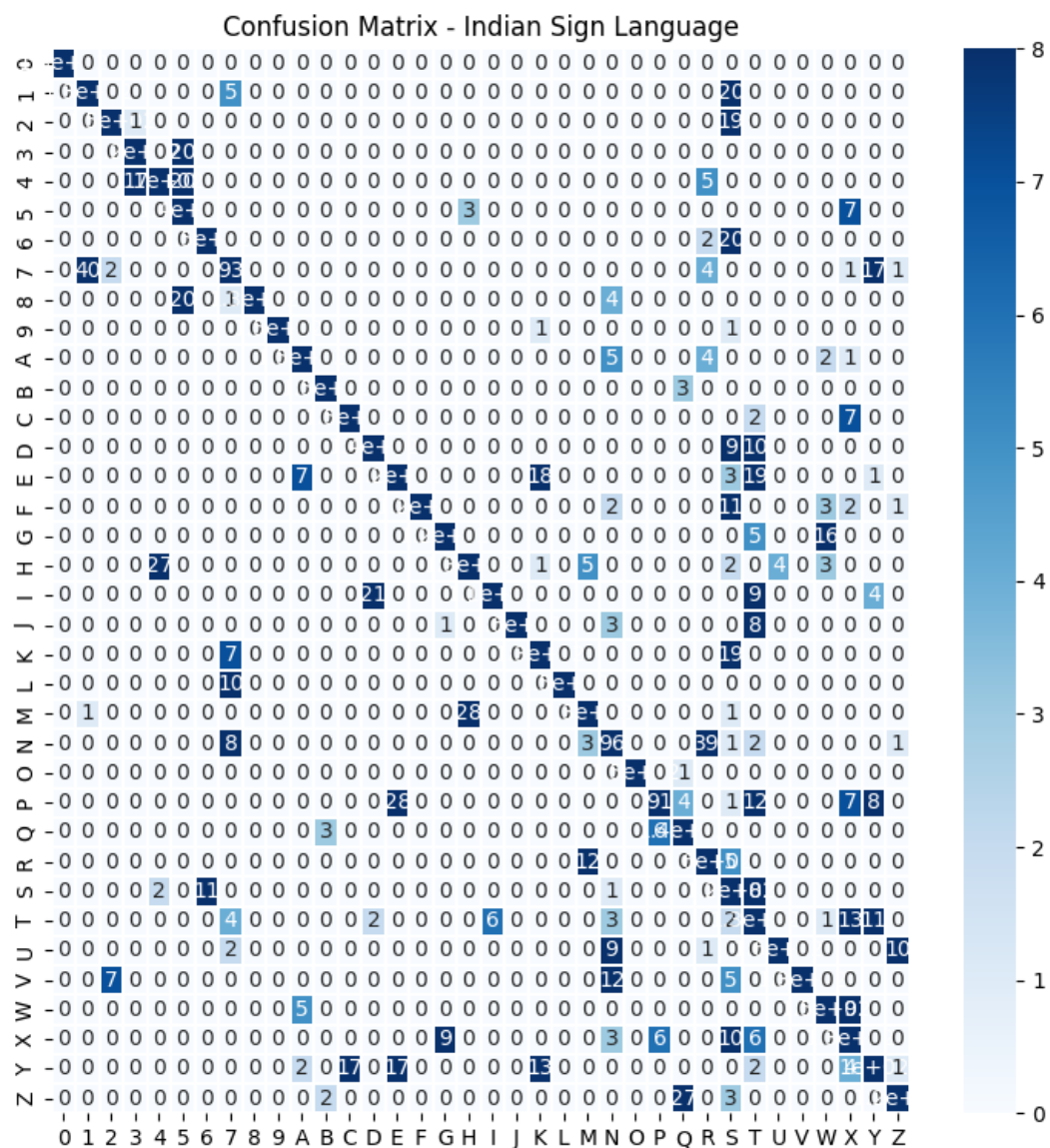


Figure 4.12: Confusion Matrix of Naive Bayes

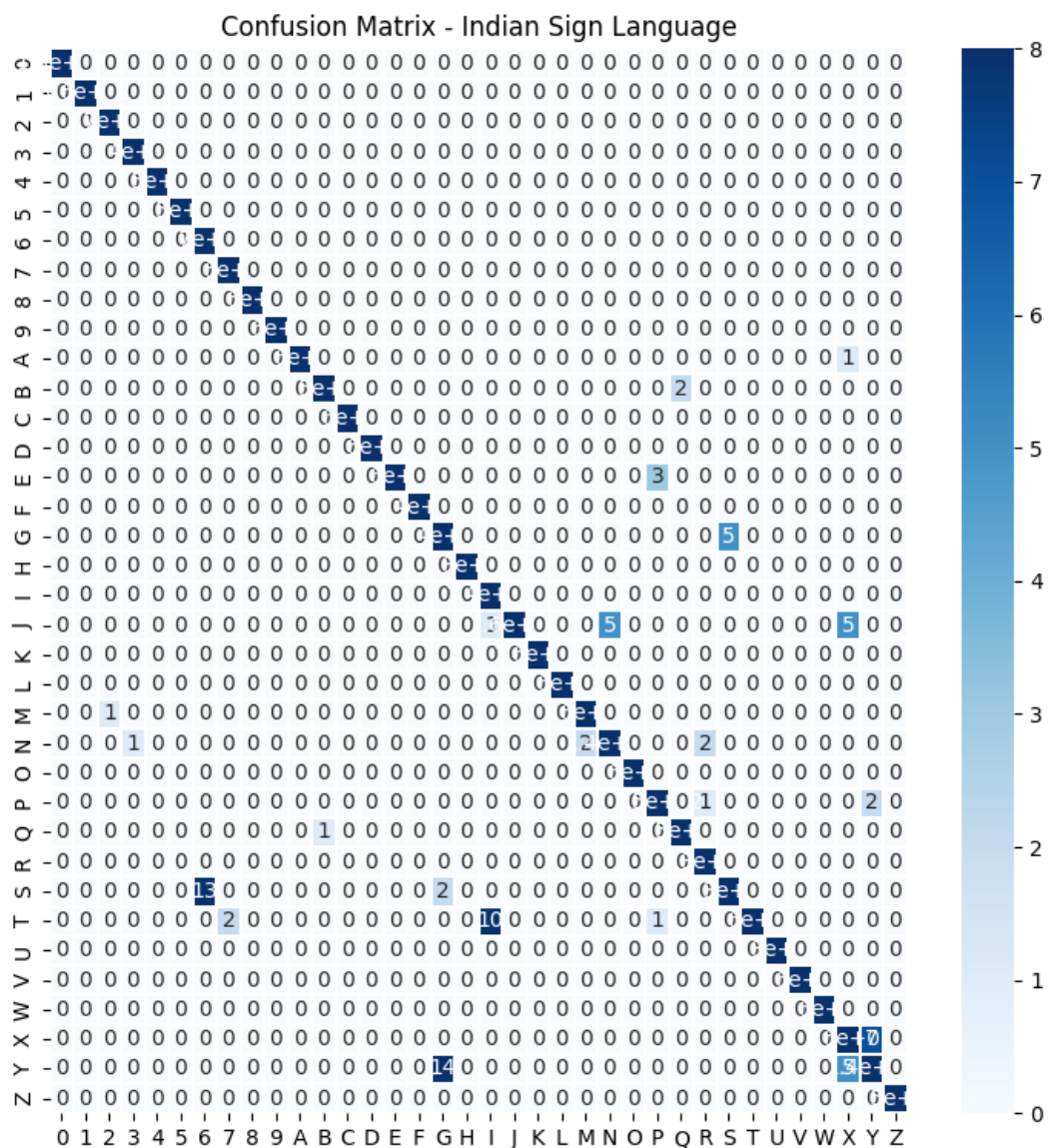


Figure 4.13: Confusion Matrix of SVM

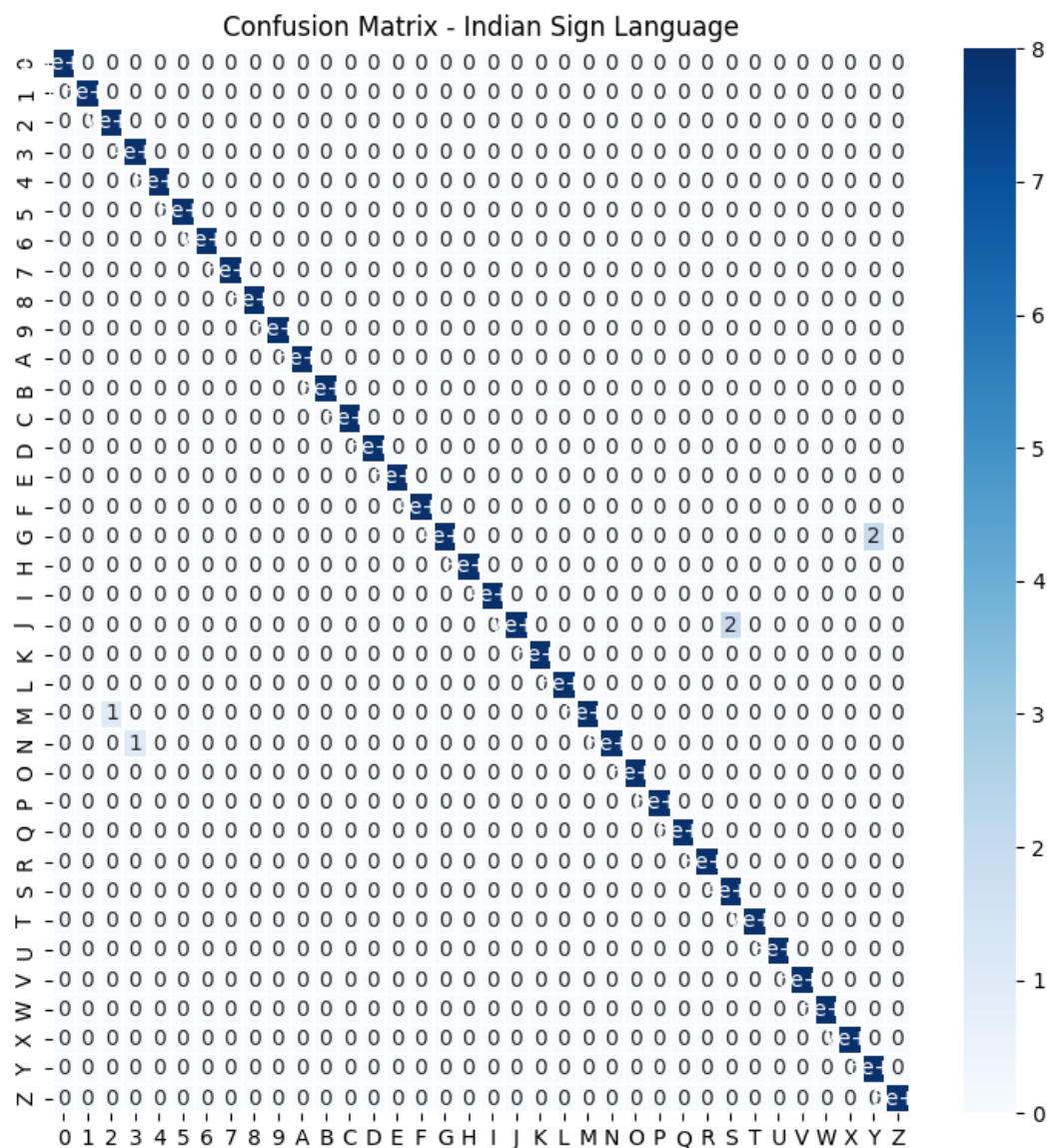


Figure 4.14: Confusion Matrix of KNN

Chapter 5

CONCLUSION

The touchless token generation system using hand gestures is a promising approach to provide a more hygienic and convenient way of accessing secure systems. The experimental results demonstrate that the system is capable of accurately recognizing hand gestures in real-time with a high degree of accuracy, even under varying lighting conditions and different camera angles. The system has good robustness and can be used in various environments. To use the system, the user would stand in front of a camera that is capable of detecting hand gestures, and then perform a specific gesture that has been programmed to generate a token. The system would then recognize the gesture and generate a unique token for that user. The system uses a combination of computer vision and machine learning techniques to recognize specific hand gestures and generate tokens. It imports the necessary libraries, reads in the image data, processes it using the MediPipe solution, and then stored which is used to train the model. Classification models like Random Forest, Decision Tree, Naive Bayes, SVM, and KNN are trained on the training data and evaluated on the testing data. The SVM classifier achieved an accuracy rate of 98.47%, which was the highest among the tested classifiers. Real-time data is captured using computer vision library, preprocessed, and then classified. A ticket shall be generated when a hand gesture is captured.

5.1 Future Enhancement

In the future, the touchless token generation system using hand gestures can be improved by incorporating audio feedback for the generated token. By adding sound effects or other audio cues to indicate when a token has been generated, users can receive immediate feedback and confirmation that their action has been successful. This can enhance the overall user experience of the system and improve the user's confidence in the system's functionality. It is useful in situations where the user may not have visual access to the system, such as in low-light environments or for users with visual impairments.

Another potential future enhancement for a touchless token generation system using hand gestures could involve incorporating the ability to detect fluctuating hand gestures, such as waving hands. This enhancement would introduce an additional level of functionality and interactivity to the system to generate tokens.

REFERENCE

- [1] M. Nawaz, R. W. Chan, A. Malik, T. Khan, and P. Cao, "Hand Gestures Classification Using Electrical Impedance Tomography Images," in *IEEE Sensors Journal*, vol. 22, no. 19, pp. 12923-12931, Oct. 2022.
- [2] C. M. Jin, Z. Omar, and M. H. Jaward, "A mobile application of American sign language translation via image processing algorithms," in *IEEE Region 10 Symposium (TENSYP)*, pp. 104-109, May.2016
- [3] G. Jia, H.-K. Lam, S. Ma, Z. Yang, Y. Xu, and B. Xiao, "Classification of Electromyographic Hand Gesture Signals using Modified Fuzzy C-means Clustering and Two-Step Machine Learning Approach," in *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 6, pp. 1428 - 1435, June. 2020.
- [4] M. F. Wahid, R. Tafreshi, and R. Langari, "A Multi-Window Majority Voting Strategy to Improve Hand Gesture Recognition Accuracies Using Electromyography Signal," in *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27,no. 2, pp. 427 - 436, February.2020.
- [5] U. Sairam, D. K. Reddy Gowra, and S. C. Kopparapu, "Virtual Mouse using Machine Learning and GUI Automation," in *8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Vol. 1, pp. 1112-1117, March.2022.
- [6] Devanandan, M., Rasaratnam, V., Anbalagan, M. K., Asokan, N., Panchendrarajan, R., Tharmaseelan, J."Cricket Shot Image Classification Using Random Forest" in *3rd International Conference on Advancements in Computing (ICAC)*, pp. 425-430, December.2021.
- [7] Salim, S., Jamil, M. M. A., Ambar, R., Roslan, R., Kamardan. "Sign Language Digit Detection with MediaPipe and Machine Learning Algorithm" in *IEEE 12th International*

- Conference on Control System, Computing and Engineering (ICCSCE)*. pp. 180-184. November.2020.
- [8] Zhai, Z. "Gesture Interaction System Design for Telerehabilitation Based on Mediapipe" in *IEEE 2nd International Conference on Software Engineering and Artificial Intelligence (SEAI)* pp. 279-283, June.2022.
- [9] A. Sharma, A. Mittal, S. Singh, V. Awatramani (2020). "Hand Gesture Recognition using Image Processing and Feature Extraction Techniques" in *International Conference on Smart Sustainable Intelligent Computing and Applications under ICITETM2020*, pp. 181-190.
- [10] Raheja, J. L., Anand Mishra, Ankit Chaudhary. "Indian sign language recognition using SVM." in *Pattern Recognition and Image Analysis* 26, pp. 434-441, June.2016,
- [11] I., J. Tupal, M. Cabatuan "Vision-Based Hand Tracking System Development for Non-Face-to-Face Interaction" in *IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, pp.1-6, November.2021
- [12] B., J. Boruah, A., K. Talukdar, K.. K. Sarma "Development of a Learning-aid tool using Hand Gesture Based Human Computer Interaction System" in *Advanced Communication Technologies and Signal Processing (ACTS)*, pp.1-5, December.2021
- [13] S. Adhikary; A., K. Talukdar; K., K. Sarma "A Vision-based System for Recognition of Words used in Indian Sign Language Using MediaPipe" *Sixth International Conference on Image Information Processing (ICIIP)*, pp. 390-394, November.2021.
- [14] D. Bisht, M. Kojage; M. Shukla, Y., P. Patil "Smart Communication System Using Sign Language Interpretation" in *31st Conference of Open Innovations Association (FRUCT)*, pp. 12-20, April.2020
- [15] K. Priya, B.J., Sandesh "Hand Landmark Distance Based Sign Language Recognition using MediaPipe" in *International Conference on Emerging Smart Computing and Informatics (ESCI)*, pp. 1-7, March.2023.

APPENDIX

Screenshots

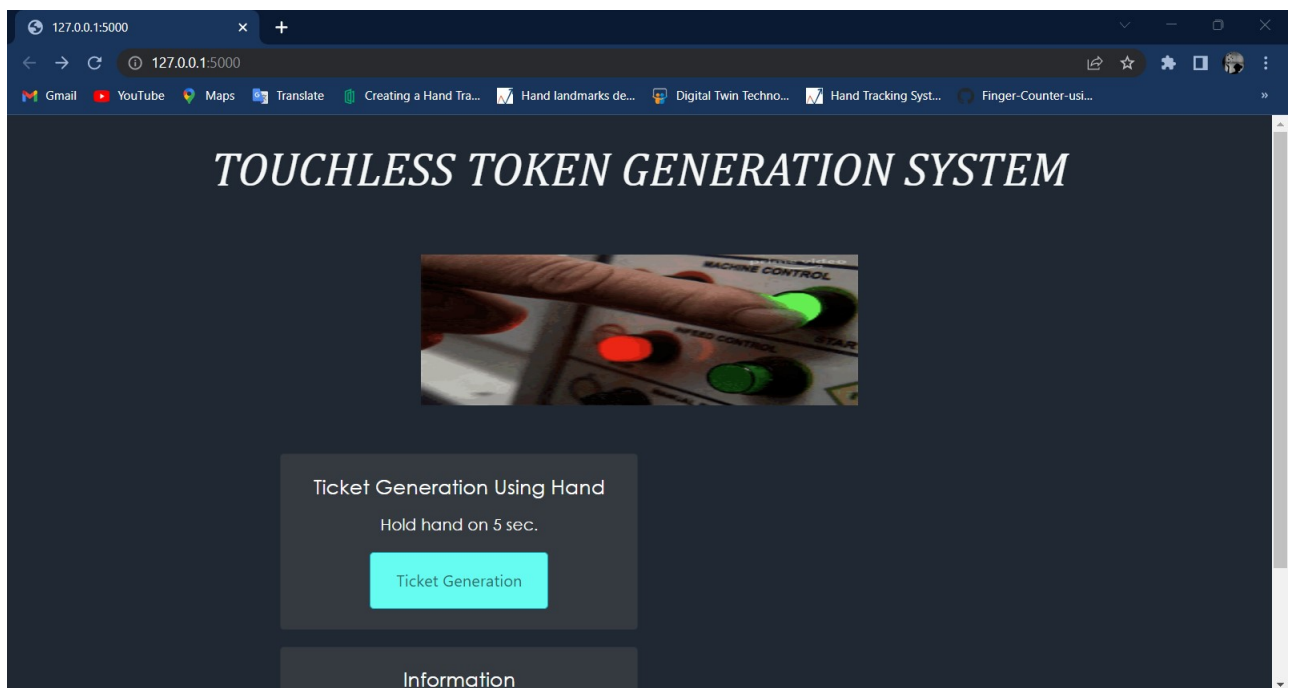


Figure A.1: Home Page

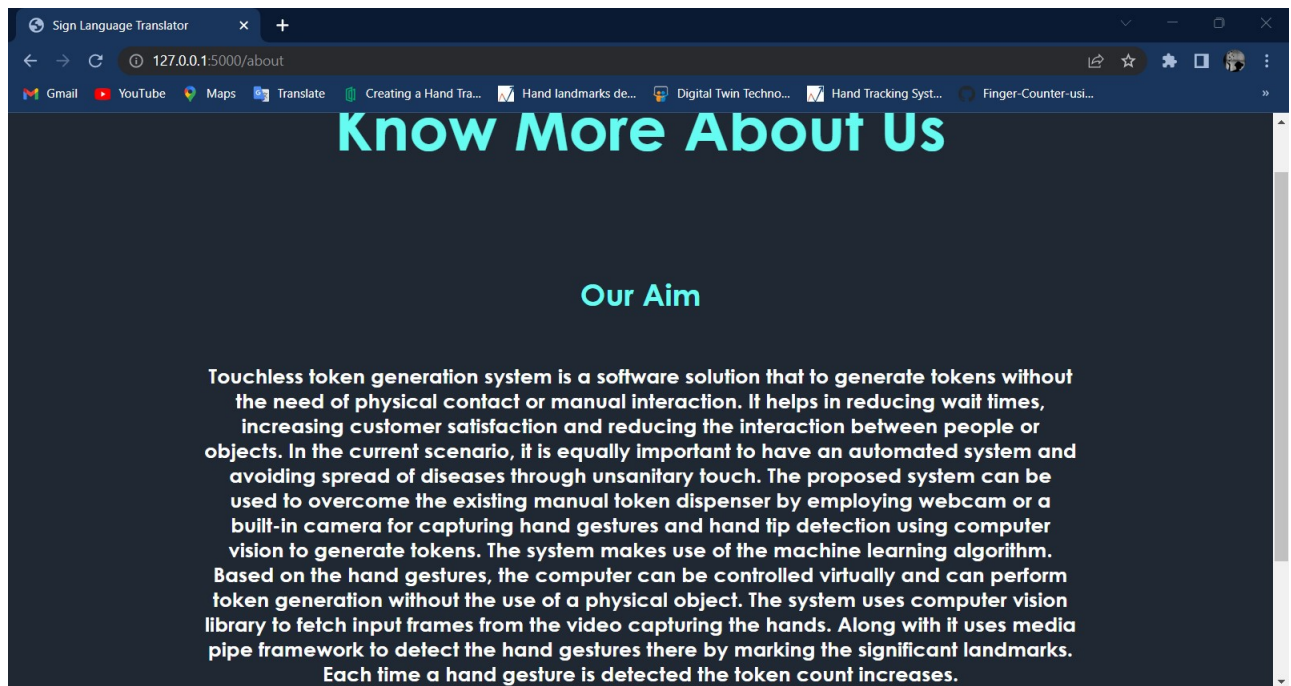


Figure A.2: About Page

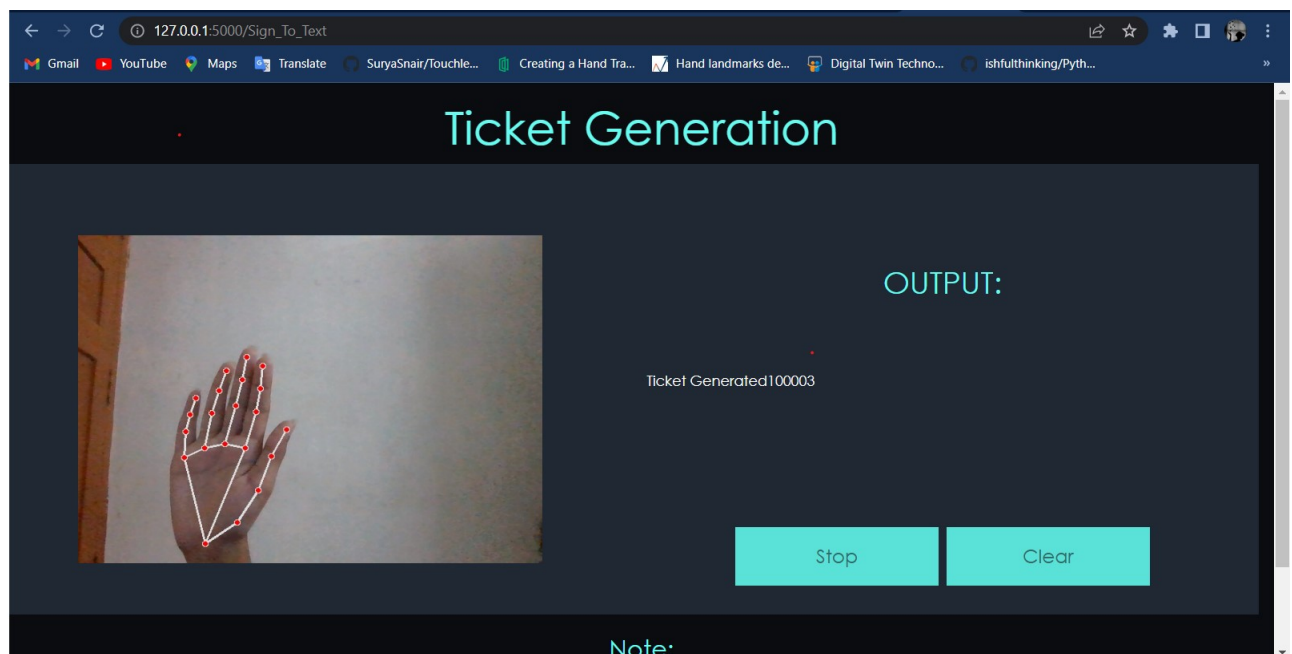


Figure A.3: Token Generation page