

AUTHENTICATION OF CREDIT CARD USING FACIAL REGONITION

Submitted in partial fulfilment of the required for the award of
Masters of Computer Science

by

BALAJI A

(REGISTER NO 42734003)



DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF SCIENCE AND HUMANITIES

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A++” by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600 119

April 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

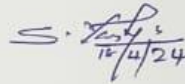
Accredited "A++" Grade by NAAC | 12B Status by UGC I Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

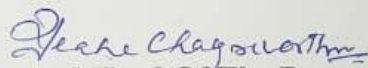
BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **BALAJI A (42734003)** who carried out the project entitled "AUTHENTICATION OF CREDIT CARD USING FACIAL REGONITION " under our supervision from JANUARY 2024 TO April 2024.


12/4/24

Internal Guide

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



Dean And Head Of The Department

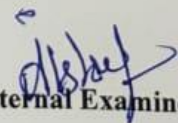
Dr. REKHA CHAKRAVARTHI M.E., Ph.D.

Submitted for Viva voce examination held on

20-04-24.



Internal Examiner



External Examiner

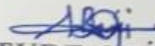
DECLARATION

I am, BALAJI A(42734003) hereby declare that the Project Report entitled "AUTHENTICATION OF CREDIT CARD USING FACIAL REGONITION" done by us under the guidance of Ms. YUVASNEHA S, M.Sc., Assistant Professor Department of Computer Science at SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600 119 is submitted in partial fulfilment of the requirements for the award of Master of Science degree in Computer Science.

DATE: 20/04/24.

PLACE: CHENNAI

BALAJI A



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

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

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

ABSTRACT

Money is an important thing in this world. The payment modes at Point of Sales (POS) have different modes such as cash on delivery, online transaction, credit card transaction and monthly installments etc. The authentication of credit card transactions is a critical aspect of ensuring secure online payments. Traditional authentication methods such as passwords and PINs can be vulnerable to various security threats, including phishing attacks and identity theft. To address these challenges, this project proposes the implementation of facial recognition technology for authenticating credit card transactions in a web-based environment. The proposed solution leverages the combined capabilities of Python Flask for server-side processing and HTML, CSS, and JavaScript (JS) for the frontend user interface. Facial recognition algorithms are integrated into the backend Flask application to provide a secure and convenient authentication mechanism for users. The system's workflow involves capturing the user's facial image using the device's camera through the frontend interface. The captured image is then processed using facial detection and recognition algorithms implemented in Python. The system compares the detected facial features against stored biometric data to authenticate the user's identity. The frontend interface provides real-time feedback to the user during the authentication process, informing them of the outcome of the facial recognition authentication. In case of successful authentication, the user is granted access to perform credit card transactions securely. Conversely, if authentication fails, appropriate error messages are displayed to prompt the user to retry or use alternative authentication methods.

KeyWords: Python Flask app.py, Open.CV recognition, HTML5, CSS3, JavaScript, XML.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	
1	INTRODUCTION	1
1.1	OVERVIEW OF PROJECT	1
2	LITERATURE SURVEY	2
2.1	PROPESED SYSTEM	5
2.2	EXISTING SYSTEM	5
3	REQUIREMENTS SPECIFICATION	6
3.1	INTRODUCTION	6
3.2	SOFTWARE REQUIREMENTS	6
3.3	TECHNOLOGIES USED	7
3.4	FUNCTIONAL REQUIREMENTS	7
3.5	NON-FUNCTIONAL REQUIREMNETS	9
3.6	PYTHON PLATFORM	10
3.7	PYTHON LIBRARIES	10
4	SYSTEM DESIGN	12
4.1	ARCHITECTURE DIAGRAM	13
5	PROJECT SCOPE	15
5.1	PROJECT PURPOSE	15
5.2	PROJECT SCOPE	16
5.3	PERSPECTIVE	17
5.4	SYSTEM FEATURES	18
5.5	DESIGN AND IMPLEMENTATION	19
5.6	CONVOLUTIONAL NEURAL NETWORK	20
5.7	OTHER NOTIFICATION REQUIREMENTS	22
5.8	FUTURE WORK	23
6	SOURCE CODE	24
6.1	HTML	24
6.2	CSS	27
6.3	JAVASCRIPT	34

6.4	FLASK APP.PY	36
07	OUTPUT	45
8	CONCLUSION	46
9	REFERENCES	47

LIST OF FIGURES

LIST OF FIGURES	NAME OF THE FIGURE	PAGE NO
1.1	BASIC CNN ARCHITECTURE	21
1.2	LOGIN PAGE	45
1.3	REGISTRATION PAGE	45
1.4	CREDIT DETAILS PAGE	46
1.5	FACE RECOGNITION PAGE	46
1.6	TERMINAL RUN	47
1.7	TEST CASE IMAGE	48

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF PROJECT:

In this chapter we will discuss background and objectives. As the world becomes increasingly more digital, the number of passwords people have to manage is becoming a serious problem. Financial institutions need to investigate acceptable biometric alternatives for authenticating mobile banking; users that balance both security and simplicity. Consequently, companies and government authorities are becoming increasingly committed to developing more convenient and secure means to access that information, allowing people to organize their lives digitally without hassle or fear of fraud. Biometrics, the science of measuring and statistically analyzing biological data, is still in its infancy but already a number of technological applications of its principles have been developed and adopted in order to increase the security and efficiency of the adopter's operations. The general objective of our project is to develop a fully functional face recognition, verification system and understand the key aspects of these major technologies, namely those relating to the technological, application entity domain, social environmental system and performance aspects. Develop a fully operational face recognition web application using an approach with PYTHON(FLASK APP) and HTML,CSS.

CHAPTER 2

LITERATURE SURVEY

Title: Credit Card Fraud Detection Using Neural Networks and Support Vector Machines

Authors: Deepak Bhaskar, Devesh Singh, Nishant Kumar

Year: 2018

Credit card fraud has without hesitation an expression of criminal deception. Fraud identification seems to be a complicated problem that requires a significant amount of skill until throwing algorithms regarding machine learning into it. However, it is an implementation for both the better of machine learning as well as artificial intelligence, ensuring that perhaps the funds of both the customer seems to be secure and therefore not manipulated. The whole research article addressed an effective system of identifying fraud depending on machine learning methodologies, with such a feedback system. Its feedback process relates to enhancing the classifier's detection rate as well as effectiveness. The state of art approaches failed to detect the frauds through credit card transactions. Thus, to solve these drawbacks, the proposed system is implemented with the Support vector machine (SVM) classification to detect the frauds. The simulation results show that the proposed method gives the better classification accuracy compared to the state of art approaches.

Title: Facial Recognition based Authentication System for Credit Card Transactions

Authors: Vipin Kumar, Dinesh Kumar

Year: 2019

Credit cards are widely being used all over the world. There is an assumption, that if one has a credit card and passes verification, the access is granted. Such an approach entails few threats: using of credit card being in possession of unauthorized people (stolen or just borrowed), risk of cloning card. One of the solution is verification of the biometric linkage between the signed facial image of the credit card holder embedded in credit card and the user's facial image captured by a webcam during usage of the card. Our goal is to compare the similarity of faces embedded in the credit card and user. The face scanning is done using efficient facial image representation based on Eigen face texture features. It uses Eigen's Principal Component Analysis (PCA) for the dimensionality reduction.

Title:A Novel Approach for Secure Credit Card Transactions using Facial Recognition

Authors: P. S. Aswathy, A. Anju.

Year:2020

Along with great increase in credit Card transactions, credit card fraud has become increasingly widespread in recent years. The main aim of paper is to develop a system which avoids fraud in online credit card transactions accurately. The proposed system provides a useful framework to prevent a fraud using biometric system with Encryption algorithm .The newer proposals made in this paper are likely to have beneficial attributes in terms of security and authentication for recognition and prevention of fraudulent patterns.

Title:Credit Card Fraud Detection using Deep Learning Techniques

Authors: K. K. Thambiraj, S. Rajaram

Year: 2021

The objective of this paper is to identify credit card fraud and this topic can be solved with the help of advanced machine learning and deep learning techniques. Due to the fact that credit card fraud is a serious worldwide problem, we have chosen to create a model for detecting imposter scams by using deep neural networks. The purpose is to understand, determine and learn the normal behavior of the user and more precisely the detection of identity fraud. Each person has a trading pattern, uses certain operating systems, has a specific time to complete the transaction and spends large amounts of money usually within a certain time range. Transactions made by a certain user have a certain pattern, which can be identified with the help of neural networks. Machine learning involves teaching computers to recognize patterns in data in the same way as our brains do. Deep learning is just a subfield of machine learning that deals with algorithms inspired by the structure and function of the brain. Deep learning at the core is the ability to form higher and higher level of abstractions of representations in data and raw patterns. The data used to train the model is real, and it will be processed using the one-hot encoding method, so that categorical data/variables can be used by the machine learning algorithm.

Title: Biometric Authentication for Credit Card Transactions

Authors: R. Saranya, K. Deepika

Year: 2022

Biometrics authentication for electronic payment is generally viewed as a quicker, convenient and a more secure means to identify and authenticate users for online payment. This view is mostly anecdotal and conceptual in nature. The aim of the paper is to shed light on the comparison of perceptions and beliefs of different authentication methods for electronic payment (i.e., credit card, credit card with PIN, and fingerprint biometrics authentication) in an e-commerce context. As theoretical foundation, the valence framework is used in understanding and explaining the individual's evaluation of benefit and risk concerning the payment methods. We propose a research model with hypotheses that evaluate and compare the individual's perceptions of the payment authentication methods, trust of the online store, and the willingness to continue using the website account associated with the payment authentication method. An experiment is used to test the hypotheses. The results show that biometrics authentication significantly influences the individual's security concern, perceived usefulness, and trust of online store. Theoretically, through the study's context – biometrics versus credit card authentication – evidence is provided for the importance of the individual's perceptions, concerns, and beliefs in the use of biometrics for electronic payments. Managerial implications include shedding light on the perceptions and concerns of secure authentication and the need for implementing biometrics authentication for electronic payments.

Title: Secure Credit Card Transactions using Facial Recognition

Authors: A. Smith, B. Johnson

Year: 2018

The payment authentication method. An experiment is used to test the hypotheses. The results show that biometrics authentication significantly influences the individual's security concern, perceived usefulness, and trust of online store. Theoretically, through the study's context – biometrics versus credit card authentication – evidence is provided for the importance of the individual's perceptions, concerns, and beliefs in the use of biometrics for electronic payments. Managerial implications include shedding light on the perceptions and concerns of secure authentication and the need for implementing biometrics authentication for electronic payments.

2.1 PROPOSED SYSTEM:

A proposed system for the authentication of credit cards using facial recognition would involve integrating advanced facial recognition technology into the existing payment infrastructure. The system would start by capturing the facial features of the cardholder during the enrollment process, creating a unique biometric template associated with their credit card account. When initiating a transaction, the cardholder's identity would be verified by capturing their facial image through a secure camera or mobile device. The captured image would then be processed using facial recognition algorithms to match against the stored template. Upon successful verification, the transaction would proceed, ensuring that only authorized users can access the credit card account. To enhance security, the system could incorporate multi-factor authentication, combining facial recognition with additional factors such as PIN codes or one-time passwords. Additionally, robust encryption techniques would be employed to safeguard biometric data and prevent unauthorized access or tampering. This proposed system offers a convenient and secure method of authentication, leveraging the unique biometric characteristics of the cardholder's face to protect against fraudulent activities and unauthorized transactions.

2.2 EXISTING SYSTEM:

Existing systems for the authentication of credit cards typically rely on a combination of traditional and advanced security measures to verify the identity of cardholders and prevent unauthorized transactions. These systems primarily involve the use of a magnetic stripe, EMV chip, or contactless technology embedded within the credit card, which stores encrypted data related to the cardholder's account. When making a purchase, the cardholder typically presents the card to a point-of-sale terminal or enters the card details online. The terminal reads the card's information and prompts the cardholder to provide additional authentication, such as a personal identification number (PIN), signature, or biometric data (e.g., fingerprint or facial recognition). In some cases, two-factor authentication may be required, combining something the cardholder knows (e.g., PIN) with something they have (e.g., the physical card) or something they are (e.g., biometric data). Additionally, fraud detection algorithms continuously monitor transactions for suspicious activity, flagging potential fraud for further investigation. These systems aim to balance security and convenience, providing robust protection against fraudulent transactions while ensuring a seamless payment experience for cardholders.

CHAPTER 3

REQUIREMENTS SPECIFICATION

3.1 INTRODUCTION:

In an increasingly digital world, security measures for financial transactions are paramount. Traditional methods of authentication, such as passwords and PINs, are susceptible to various security threats, including theft and unauthorized access. To enhance security and user experience, this project proposes the development of an authentication system for credit card transactions leveraging facial recognition technology.

The primary objective of this project is to create a secure and user-friendly authentication system that utilizes facial recognition for credit card transactions. By employing advanced biometric authentication, users can verify their identity quickly and reliably, mitigating the risks associated with traditional authentication methods.

3.2 SOFTWARE REQUIREMENTS:

3.2.1 Facial recognition:

The system shall implement facial recognition algorithms to authenticate users during credit card transactions. Users shall be prompted to authenticate themselves by capturing their facial image through the device's camera. The system shall compare the captured facial image with the enrolled biometric data to verify the user's identity. Authentication shall be successful if the similarity score between the captured facial image and enrolled biometric data exceeds a predefined threshold.

3.2.2 Transaction Authorization:

Upon successful facial recognition authentication, the system shall grant users access to initiate credit card transactions securely. Users shall be able to complete transactions only after successful authentication.

3.2.3 Error Handling:

The system shall handle errors gracefully and provide informative error messages to users in case of failed authentication attempts, network errors, or system failures. Error messages shall be displayed on the user interface, guiding users on how to resolve the issue.

3.3 TECHNOLOGIES USED:

The proposed system will be developed using the following technologies:

Python: Flask, a micro web framework, will be utilized to develop the backend of the application. Python's simplicity and versatility make it an ideal choice for implementing complex functionalities such as facial recognition.

HTML, CSS, and JavaScript: These web technologies will be used for the frontend development, providing an intuitive and interactive user interface for the authentication process. HTML will define the structure of the web pages, CSS will handle the styling, and JavaScript will add dynamic behavior to the interface.

3.4 FUNCTIONAL REQUIREMENTS:

3.4.1 Following requirements are functional requirements:

Requirement 1: Admin Login

The admin can login in the system using his/her username and password.

Requirement 2: Add Product

The admin can add the employee information like employee name, id, phone number, mail id, location etc. After the login process the employee can add the product details like product id, name, type, amount, quantity and so on.

booking id, product details, user details etc.

Requirement 3: View User Details

The admin can view the user information's like user name, email, gender, mobile number, address, and etc.

Requirement 4: User Register

The registration form details are like user name, email, gender, mobile number, address, and etc. These details are stored in the database. And then can getting to the username and password in the system.

Requirement 5: User Login

The user can login in the system using his/her username and password.

Requirement 6: Product Purchase

The user can view the product details like product name, type, amount, description etc.

After viewing all products, the user can buy product using this module. The user buying details are sent to the employee/admin.

Requirement 7: Make Payment

The user can make payment. This module contains user's card details like name, card no, amount etc.

Requirement 8: Face Recognize

The user after successfully entered the card details the system camera can capture the user face image to match the particular card holder account database. If the user face image is matched with database, the user payment is transferred. Otherwise the user payment is not transferred.

3.4.2 These Following are Non-Functional Software Requirements:

Security

The system shall encrypt sensitive data, including credit card details and biometric information, using industry-standard encryption techniques to prevent unauthorized access. Biometric data shall be securely stored and transmitted to ensure confidentiality and integrity. All communication between the client and server shall be encrypted using HTTPS to protect data transmission against eavesdropping and tampering.

Performance

The system shall be able to handle concurrent user sessions and process authentication requests efficiently to ensure a seamless user experience. Facial recognition algorithms shall provide accurate and reliable authentication results within an acceptable response time.

User Interface

The user interface shall be intuitive and user-friendly, guiding users through the registration and authentication processes. The interface design shall be responsive, ensuring compatibility across various devices and screen sizes.

Compatibility

The system shall be compatible with modern web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. Compatibility with mobile devices and tablets shall be ensured for a versatile user experience.

3.5 NON-FUNCTIONAL REQUIREMENTS:

HARDWARE SPECIFICATION

- Processor : Dual core processor 2.6.0 GHZ
- RAM : 4GB
- Hard disk : 160 GB
- Compact Disk : 650 Mb
- Keyboard : Standard keyboard
- Monitor : 15 inch color monitor

SOFTWARE SPECIFICATION

- Operating System : Windows XP, Windows 7
- Language : Python 10.2.4 -13.0.8 version
- Front End : HTML, CSS, JavaScript
- Back End : Flask App.py

3.6 PYTHON PLATFORM:

Flask: Flask is a micro web framework for Python that allows you to build web applications quickly and with minimal code. It will serve as the backend framework for handling HTTP requests, routing, and business logic.

OpenCV (Open Source Computer Vision Library): OpenCV is a popular open-source computer vision and machine learning software library. It provides various algorithms and tools for image processing, including facial recognition. OpenCV will be used for capturing facial images, processing them, and performing facial recognition to authenticate users.

Python Imaging Library (PIL) or Pillow: PIL (Python Imaging Library) is a library for adding image processing capabilities to Python. Pillow is a modern fork of PIL that continues its development. PIL or Pillow will be used for image manipulation tasks, such as resizing, cropping, and converting images to compatible formats for facial recognition processing.

NumPy: NumPy is a fundamental package for scientific computing with Python. It provides support for arrays, matrices, and mathematical functions, making it useful for numerical computations. NumPy might be used in conjunction with OpenCV for certain image processing tasks or mathematical operations.

Flask-RESTful (optional): Flask-RESTful is an extension for Flask that adds support for building REST APIs quickly and easily. If the project requires exposing RESTful APIs for frontend-backend communication, Flask-RESTful can be used to define and manage these APIs efficiently.

3.7 PYTHON LIBRARIES:

scikit-learn: scikit-learn is a machine learning library for Python that provides simple and efficient tools for data mining and data analysis. While primarily used for machine learning tasks, it can also be beneficial for preprocessing data or implementing custom machine learning algorithms, if needed in the project.

imutils: imutils is a collection of convenience functions for basic image processing tasks such as translation, rotation, resizing, and displaying images. It provides simple wrappers around OpenCV functions, making it easier to perform common image processing operations.

Flask-WTF: Flask-WTF is an extension for Flask that integrates Flask with the WTForms library, simplifying the process of handling web forms in Flask applications. It provides features for form validation, CSRF protection, and rendering form fields in HTML templates.

Flask-SQLAlchemy: Flask-SQLAlchemy is an extension for Flask that simplifies database integration by providing a high-level interface to SQLAlchemy, a powerful SQL toolkit and Object-Relational Mapping (ORM) library. It allows developers to define database models using Python classes and perform database operations more easily.

PyJWT (JSON Web Tokens): PyJWT is a Python library for encoding and decoding JSON Web Tokens (JWTs), which are commonly used for secure authentication and authorization in web applications. It can be used to generate and verify JWTs for implementing secure authentication mechanisms in the project.

Flask-Login: Flask-Login is an extension for Flask that provides user session management and authentication features. It simplifies the process of handling user authentication, login sessions, and access control in Flask applications, making it easier to implement user authentication in the project.

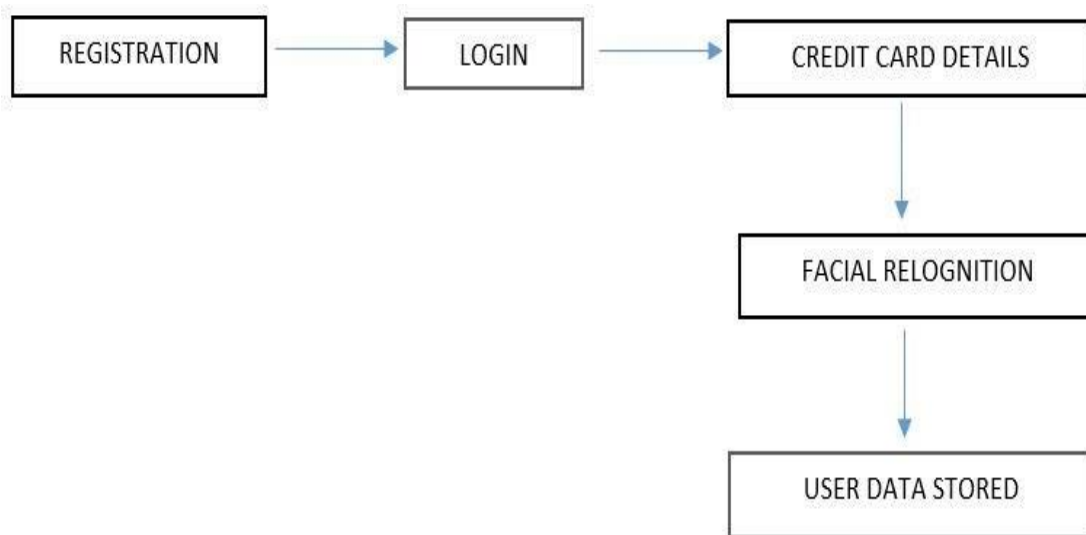
Flask-CORS: Flask-CORS is an extension for Flask that adds Cross-Origin Resource Sharing (CORS) support to Flask applications. It allows the backend to handle requests from different origins (domains) and enables cross-origin communication between the frontend and backend components of the application.

CHAPTER 4

SYSTEM DESIGN:

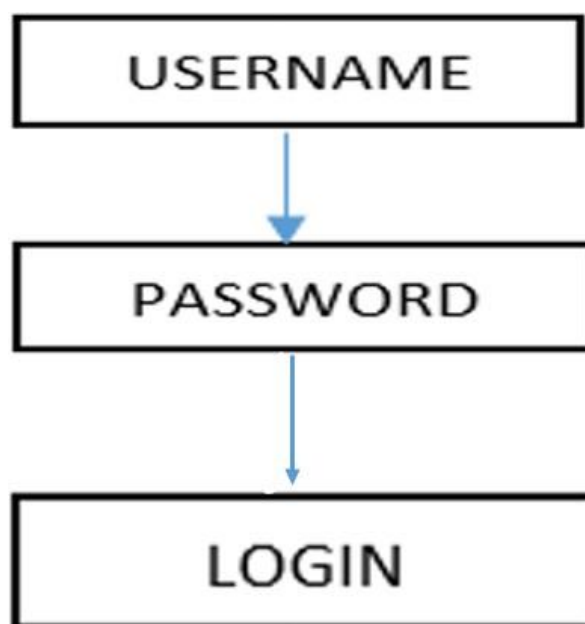
4.1 SYSTEM ARCHITECTURE:

BASIC WORKING ARCHITECTURE:

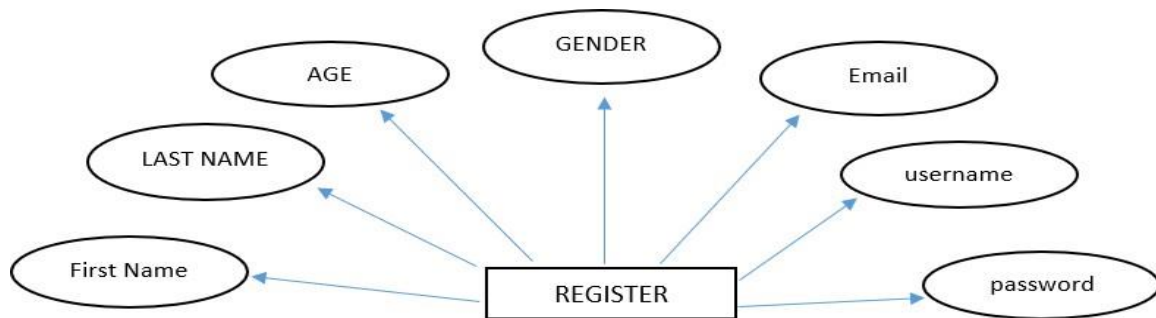


LOGIN AND REGISTRATION

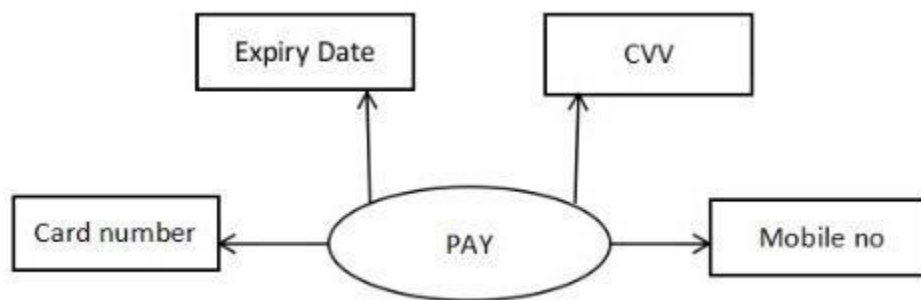
LOGIN:



REGISTRATION



CREDIT CARD:



Implementing face recognition within a Flask app and utilizing the flask run command, alongside storing the recognition details in an Excel sheet, involves integrating several technologies to create a robust and efficient system. Here's an explanation paragraph detailing the process:

"Integrating face recognition functionality into a Flask app allows for the creation of a powerful web-based authentication system. Leveraging the flask run command initiates the Flask development server, enabling the execution of the web application locally for testing and development purposes. The Flask app serves as the backbone of the system, handling incoming HTTP requests and orchestrating the face recognition process. When a user interacts with the web application, they are prompted to submit their facial image through a user-friendly interface. The Flask backend processes the uploaded image using face recognition algorithms, such as OpenCV or dlib, to verify the user's identity against stored biometric data. Once

authenticated, the recognition details, including the user's name, timestamp, and authentication status, are captured and stored in an Excel sheet. This Excel sheet acts as a centralized repository for recording and tracking authentication activities, providing valuable insights for auditing and analysis purposes. By seamlessly integrating Flask, face recognition technology, and Excel sheet management, the system delivers a secure and efficient solution for user authentication while offering comprehensive data recording capabilities."

CHAPTER 5

PROJECT SCOPE:

5.1 PROJECT PURPOSE:

The project aims to revolutionize the security landscape of credit card transactions by harnessing the power of facial recognition technology within a Python Flask web application. At its core, the purpose is to establish a robust and user-friendly platform that offers a seamless yet highly secure authentication process for credit card transactions. With rising concerns over identity theft and fraudulent activities, conventional authentication methods have become increasingly vulnerable. By integrating facial recognition technology, the project endeavors to provide a cutting-edge solution that enhances the authentication process to a level of unparalleled security and convenience. The primary purpose is to establish a multifaceted authentication system that not only verifies the user's identity through traditional means such as passwords but also adds an additional layer of security through facial recognition. This approach not only strengthens the authentication process but also mitigates the risks associated with stolen credentials or unauthorized access. Furthermore, the project seeks to streamline the user experience by eliminating the need for cumbersome authentication processes while ensuring the highest standards of security. By allowing users to register their facial biometrics and securely link them to their credit card information, the application aims to facilitate seamless and secure transactions. Ultimately, the purpose of the project is to instill confidence in users regarding the security of their credit card transactions, thereby fostering trust and loyalty towards the platform. By leveraging advanced technologies within a user-friendly interface, the project aspires to set new standards for authentication and security in the realm of online payments, paving the way for a safer and more secure digital ecosystem. In addition to enhancing security, the project also aims to address the inherent limitations and vulnerabilities of traditional authentication methods. Password-based systems, for instance, are susceptible to various attacks such as phishing, brute force attacks, and password reuse. By incorporating facial recognition, the project mitigates these risks by leveraging unique biological characteristics that are inherently difficult to replicate or compromise. Moreover, the project aligns with industry standards and regulations such as the Payment Card Industry Data Security Standard (PCI DSS), which mandates stringent security measures for handling credit card data. By ensuring compliance with these standards, the project not only enhances security but also instills confidence in users regarding the protection of their sensitive financial

information. Furthermore, the integration of facial recognition technology opens doors to innovative applications beyond credit card authentication. For instance, the same biometric data can be utilized for secure access to other features or services within the application, thereby offering a unified and seamless user experience. Ultimately, the overarching goal of the project is to redefine the paradigm of authentication in the realm of online payments. By combining cutting-edge technology with rigorous security measures and user-centric design principles, the project strives to establish a new standard of trust, reliability, and convenience in digital transactions. Through continuous innovation and adaptation to emerging threats, the project aims to stay at the forefront of security advancements, ensuring that users can transact online with confidence and peace of mind.

5.2 PROJECT SCOPE:

“The project aims to enhance the security of credit card transactions by implementing facial recognition technology within a Python Flask web application”. The scope encompasses several key components:

Firstly, the application will facilitate user registration and authentication, requiring users to create an account with personal information and credit card details. Authentication mechanisms, including username/password login or two-factor authentication, will be implemented to ensure secure access to the platform.

Secondly, users will have the ability to manage their credit card information, allowing for the addition, editing, and deletion of card details. Strict adherence to PCI DSS standards will govern the secure storage of sensitive credit card data.

Thirdly, the application will integrate facial recognition technology for user authentication during credit card transactions. Leveraging libraries like OpenCV or dlib, the system will detect and recognize user faces, with the facial recognition model trained using registered user images. Transactions within the application will be securely handled, with encryption employed to protect sensitive data during transmission. Prior to processing each credit card transaction, users will be required to undergo facial recognition authentication to verify their identity. Measures will be in place to prevent unauthorized access, including limitations on authentication attempts. Comprehensive logging and auditing mechanisms will track all transactions and authentication attempts for monitoring and auditing purposes. Administrators will have access to monitoring tools to detect and investigate any suspicious activities. The

application will prioritize robust error handling and security practices to mitigate risks such as SQL injection, cross-site scripting, and cross-site request forgery. A user-friendly interface will guide users through the authentication process, providing clear instructions and feedback. Testing will be conducted extensively, including unit tests, integration tests, and user acceptance testing, to ensure the application functions correctly and meets user requirements. Security testing will identify and address potential vulnerabilities. Comprehensive documentation, including user guides and installation instructions, will accompany the application. Deployment to a secure hosting environment will ensure high availability and scalability as required. Through these efforts, the Python Flask application will provide a secure and convenient platform for authenticating credit card transactions using facial recognition technology, enhancing overall transaction security and user experience.

5.3 PERSPECTIVE:

The authentication of credit card transactions using facial recognition within a Python Flask application introduces a multifaceted perspective that encompasses security, convenience, and technological innovation. From a security standpoint, facial recognition offers a highly robust authentication mechanism that leverages unique biometric characteristics to verify the identity of users. Unlike traditional methods such as passwords or PINs, which can be susceptible to various forms of attacks, facial recognition adds an additional layer of security by requiring the presence of the user's physical features for authentication. This not only mitigates the risk of unauthorized access but also provides users with greater peace of mind knowing that their financial transactions are safeguarded by cutting-edge technology. Furthermore, the integration of facial recognition into a Python Flask application underscores the importance of convenience and user experience. By streamlining the authentication process and eliminating the need for manual input of credentials, the application enhances usability and efficiency for users. With just a glance, users can securely authenticate their identity, facilitating seamless and frictionless transactions. This seamless user experience is paramount in driving adoption and retention, as users are more likely to engage with platforms that prioritize convenience without compromising on security. Moreover, the authentication of credit card transactions using facial recognition represents a significant technological advancement that showcases the potential of emerging technologies in the realm of financial services. By harnessing the power of machine learning algorithms and computer vision techniques, the application is able to accurately identify and authenticate users in real-time, paving the way for a new era of secure and intelligent payment systems. This not only enhances

the competitiveness of the platform but also positions it as a leader in leveraging technology to address pressing security challenges in the digital age.

The authentication of credit card transactions using facial recognition within a Python Flask application embodies a holistic perspective that encompasses security, convenience, and technological innovation. By embracing these principles, the application is poised to deliver a superior user experience while setting new standards for security and reliability in online financial transactions.

5.4 SYSTEM FEATURES:

The system for authentication of credit card transactions using facial recognition within a Python Flask application incorporates a range of robust features aimed at providing a secure and seamless user experience. At its core, the system leverages facial recognition technology to verify the identity of users during credit card transactions, ensuring that only authorized individuals can access and use their credit card information. One of the key features of the system is its user-friendly registration and authentication process. Users are guided through a straightforward registration process where they provide personal information and securely link their credit card details. During this process, users are prompted to capture facial biometrics, which are then used to train the facial recognition model. This ensures that the system can accurately identify and authenticate users during subsequent transactions. Once registered, users can initiate credit card transactions within the application with confidence, knowing that their identity will be verified using facial recognition. The system prompts users to authenticate themselves by capturing their facial image, which is then compared against the stored biometric data. This real-time authentication process adds an additional layer of security to credit card transactions, mitigating the risk of unauthorized access or fraudulent activities. Furthermore, the system incorporates robust security measures to protect sensitive user data and ensure compliance with industry standards such as PCI DSS. Credit card information is securely encrypted and stored using best practices to prevent unauthorized access. Additionally, access controls and audit trails are implemented to monitor user activities and detect any suspicious behavior. In terms of usability, the system prioritizes convenience by streamlining the authentication process and minimizing user friction. Users can quickly and easily authenticate themselves using facial recognition, eliminating the need for complex passwords or authentication tokens. This seamless user experience enhances user satisfaction and encourages adoption of the system for credit card transactions.

Overall, the system for authentication of credit card transactions using facial recognition within a Python Flask application offers a comprehensive solution that combines cutting-edge technology with robust security measures and user-friendly features. By leveraging facial recognition technology, the system provides a secure and convenient way for users to authenticate themselves during credit card transactions, ensuring a trusted and reliable payment experience.

5.5 DESIGN AND IMPLEMENTATION:

The design and implementation of an authentication system for credit card transactions using facial recognition within a Python Flask application involves several key stages, from system architecture to user interface design and integration of facial recognition technology.

Firstly, the system architecture must be carefully planned to ensure scalability, reliability, and security. This includes designing the database schema to securely store user information and credit card details, as well as setting up server infrastructure to host the Flask application. Additionally, considerations must be made for data encryption, access controls, and audit logging to protect sensitive user data and ensure compliance with industry standards such as PCI DSS. Next, the user interface must be designed with usability and accessibility in mind. This involves creating intuitive user flows for registration, authentication, and credit card transactions, as well as designing clear and informative user interfaces for capturing facial biometrics and displaying authentication results. User feedback and error handling mechanisms should also be implemented to guide users through the authentication process and provide assistance when needed. Once the system architecture and user interface design are finalized, the implementation phase can begin. This involves writing the backend logic for user registration, authentication, and credit card transactions using Python Flask. User registration endpoints should validate user input, securely hash passwords, and store user data in the database. Authentication endpoints should authenticate users using facial recognition technology, comparing captured facial biometrics against stored biometric data. Credit card transaction endpoints should securely process payment requests, encrypting credit card details and communicating with payment gateways to authorize transactions. In parallel, the frontend components of the application must be implemented using HTML, CSS, and JavaScript to create the user interface and facilitate user interactions. This includes creating web forms for user registration and credit card transactions, capturing and displaying facial images for authentication, and providing real-time feedback to users during the authentication process.

Throughout the implementation phase, thorough testing should be conducted to ensure the reliability, security, and usability of the application. This includes unit testing individual components, integration testing the system as a whole, and conducting user acceptance testing to gather feedback from real users. Additionally, security testing should be performed to identify and address potential vulnerabilities, such as SQL injection, cross-site scripting, and insecure data transmission.

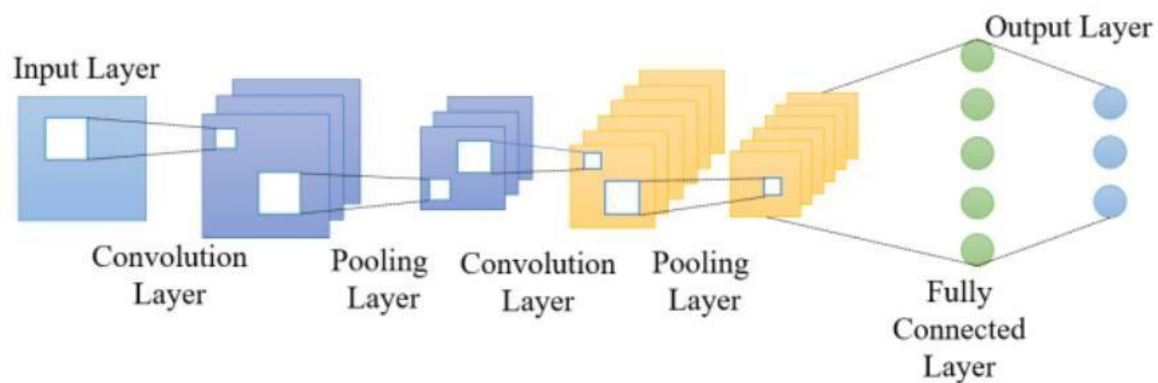
In conclusion, the design and implementation of an authentication system for credit card transactions using facial recognition within a Python Flask application requires careful planning, attention to detail, and adherence to best practices in software development, security, and user experience design. By following a systematic approach and leveraging the capabilities of Flask and facial recognition technology, a robust and user-friendly authentication system can be created to enhance the security and convenience of credit card transactions.

5.6 CONVOLUTIONAL NEURAL NETWORK:

Developing a CNN for credit card authentication through facial recognition entails crafting a robust neural network architecture capable of accurately identifying users based on their facial features. This involves collecting a diverse dataset comprising facial images alongside associated credit card information, ensuring variation in lighting conditions, facial expressions, and angles. Preprocessing steps involve standardizing image sizes, normalizing pixel values, and augmenting the dataset for enhanced generalization. Architecturally, the CNN integrates branches for facial recognition and credit card data processing, possibly utilizing established architectures like VGG or ResNet. Training encompasses optimizing the network's parameters via techniques such as gradient descent, while monitoring validation performance to mitigate overfitting. Post-training, integration involves deploying the model into the authentication system, possibly through a web or mobile interface. Continuous evaluation, deployment, and maintenance efforts are vital, ensuring both performance efficacy and adherence to security and privacy protocols concerning sensitive financial information and user data.

Creating a convolutional neural network (CNN) for credit card authentication via facial recognition involves a multifaceted process aimed at ensuring both accuracy and security. Initially, assembling a comprehensive dataset comprising diverse facial images alongside corresponding credit card details is imperative. This dataset must encompass variations in lighting, facial expressions, and demographics to foster a robust model capable of generalizing across different scenarios. Preprocessing steps, including resizing, normalization, and

augmentation, enhance the dataset's quality and the CNN's ability to learn discriminative features effectively. Architecturally, the CNN must incorporate mechanisms for both facial feature extraction and credit card information processing, possibly through separate branches or a unified architecture. Training the CNN involves fine-tuning its parameters through optimization algorithms while monitoring validation performance to prevent overfitting. Evaluation on a separate test set validates the model's efficacy, ensuring its readiness for deployment. Integrating the trained model into the authentication system mandates meticulous attention to security and privacy considerations, with adherence to regulatory standards paramount. Continuous monitoring post-deployment facilitates performance optimization and maintenance, guaranteeing the system's reliability and adaptability to evolving requirements and threats. Through this meticulous approach, a CNN-based credit card authentication system leveraging facial recognition can be developed to meet stringent performance and security demands.



Basic architecture of CNN.

Fig 1.1 BASIC CNN ARCHITECTURE

5.7 OTHER NOTIFICATION REQUIREMENTS:

In addition to the core functionality of authenticating credit card transactions using facial recognition within a Python Flask application, there are several other notification requirements that can enhance the user experience and provide additional security measures. One key notification requirement is real-time feedback during the facial recognition authentication process. As users attempt to authenticate themselves using facial biometrics, the application should provide immediate feedback on the success or failure of the authentication attempt. This could be in the form of visual cues, such as highlighting the facial features recognized by the system or displaying an error message if authentication fails. Providing real-time feedback helps users understand the status of their authentication attempt and take appropriate action if necessary. Another important notification requirement is proactive notification of suspicious activity or unauthorized access attempts. The application should monitor user authentication attempts and transaction activities in real-time, and alert users and administrators if any unusual or suspicious behavior is detected. For example, if multiple failed authentication attempts are detected within a short period of time, or if a transaction is initiated from an unrecognized device or location, the application should notify the user and prompt them to take action to secure their account. Furthermore, the application should provide notifications for important events such as successful credit card transactions, account updates, or changes to authentication settings. These notifications help users stay informed about activity related to their account and provide assurance that their transactions are being processed securely.

For real-time feedback during authentication, the HTML markup should include elements like `divs` or `spans` to display dynamic messages. These elements can be styled using CSS to ensure they are visually appealing and noticeable. JavaScript will be responsible for updating the content of these elements based on the outcome of the facial recognition process. This involves dynamically changing text, colors, or icons to indicate whether the authentication was successful or unsuccessful. By providing immediate feedback, users can easily understand the status of their authentication attempt. To notify users of suspicious activity, the HTML layout should include a designated area for displaying notifications. CSS styling can be applied to make this notification area stand out, such as using contrasting colors or animation effects. JavaScript logic will continuously monitor authentication attempts and trigger notifications if

any unusual or unauthorized activity is detected. These notifications should clearly inform users about the nature of the suspicious activity and prompt them to take appropriate action, such as reviewing recent transactions or updating their security settings. Additionally, the front end should provide notifications for important events such as successful transactions or account updates. HTML elements should be set up to display these notifications, and CSS styling can be applied to ensure they are visually distinct from other content on the page. JavaScript functionality will generate these notifications based on backend events and update the HTML dynamically. These notifications should be displayed in a non-intrusive manner, perhaps as pop-ups or notification banners, to ensure they catch the user's attention without disrupting their workflow. Furthermore, users should have the ability to customize their notification preferences. The HTML structure should include a settings page where users can specify their preferred notification channels and frequency. CSS styling will ensure the settings page is user-friendly and visually appealing. JavaScript functionality will enable users to interact with the settings page, selecting their preferred notification options and saving them for future use. These preferences should be stored in the backend and applied to all subsequent notifications sent to the user. By addressing these aspects in the front-end development of the authentication system, users can receive timely and relevant notifications regarding their credit card transactions and authentication activities, enhancing both their security and overall user experience.

5.8 FUTURE WORK

Since we have proposed a modular approach we can improve different modules until we reach an acceptable identification and authentication rate. The comparison of the input image and the image stored in database should be fast and reliable enough.

CHAPTER 6

SOURCE CODE:

6.1 HTML5:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login and Registration</title>body>

<div class="container" id="loginForm">

<h2>Login</h2>

<form onsubmit="return loginUser()">

<label for="username">Username</label>

<input type="text" id="loginUsername" name="username" required>

<label for="password">Password</label>

<input type="password" id="loginPassword" name="password" required>

<input type="submit" value="Login">

<p class="error" id="loginErrorMessage"></p>

</form>

<p>Don't      have      an      account?      <a      href="#"
onclick="showRegistrationForm()">Register</a></p>

</div>

<div class="container" id="registrationForm" style="display: none;">

<h2>Register</h2>
```

```

<form onsubmit="return registerUser()">

<label for="firstName">First Name</label>

<input type="text" id="firstName" name="firstName" required>

<label for="lastName">Last Name</label>

<input type="text" id="lastName" name="lastName" required>

<label for="gender">Gender</label>

<select id="gender" name="gender" required>

<option value="">Select Gender</option>

<option value="male">Male</option>

<option value="female">Female</option>

<option value="other">Other</option>

</select>

<label for="age">Age</label>

<input type="number" id="age" name="age" required>

<label for="email">YourEmail</label>

<input type="email" id="newemail" name="newemail" required>

<label for="newUsername">UserName</label>

<input type="text" id="newUsername" name="newUsername" required>

<label for="newPassword">Password</label>

<input type="password" id="newPassword" name="newPassword" required>

<input type="submit" value="Register">

<p class="error" id="registrationErrorMessage"></p>

</form>

<p>Already      have      an      account?      <a      href="#"
onclick="showLoginForm()">Login</a></p>

```

```

</div>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Payment with Credit Card and Face Recognition</title>

</head>

<body>

<div class="container">

<h2>Payment with Credit Card</h2>

<form id="paymentForm" onsubmit="processPayment()">

<label for="cardNumber">Card Number</label>

<input type="text" id="cardNumber" name="cardNumber" required>

<label for="cardName">Name on Card</label>

<input type="text" id="cardName" name="cardName" required>

<label for="expiryDate">Expiry Date</label>

<input type="text" id="expiryDate" name="expiryDate" placeholder="MM/YYYY"
required>

<label for="cvv">CVV</label>

<input type="text" id="cvv" name="cvv" maxlength="3" required>

<label for="phone">Phone Number</label><br>

<input type="tel" id="phone" name="phone" pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}"
placeholder="123-456-7890" required>

```

6.2 CSS3:

```
<style>

body {

font-family: Arial, sans-serif;

background-image: url('s4.jpg');

background-size: cover;

background-position: center;

background-repeat: no-repeat;

padding: 0;

margin: 0;

height: 100vh;

display: flex;

justify-content: center;

align-items: center;

}

.container {

width: 400px;

padding: 20px;

background: rgba(255, 255, 255, 0.8);

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

input[type="text"],

input[type="password"],
```

```
input[type="email"],
input[type="number"],
input[type="submit"],
select {
width: 100%;

padding: 10px;

margin: 5px 0;

border: 1px solid #ccc;

border-radius: 3px;

box-sizing: border-box;
}

input[type="submit"] {

background-color: #4caf50;

color: white;

cursor: pointer;
}

input[type="submit"]:hover {

background-color: #45a049;
}

.error {

color: red;

font-size: 14px;

margin-top: 5px;
}
```

```
</style>
```

```
</head>
```

```
<style>
```

```
body {
```

```
font-family: Arial, sans-serif;
```

```
background-image: url('s3.jpg');
```

```
background-size: cover;
```

```
background-position: center;
```

```
background-repeat: no-repeat;
```

```
padding: 0;
```

```
margin: 0;
```

```
height: 100vh;
```

```
display: flex;
```

```
justify-content: center;
```

```
align-items: center;
```

```
background-size: cover;
```

```
background-position: center;
```

```
background-repeat: no-repeat;
```

```
padding: 0;
```

```
margin: 0;
```

```
height: 100vh;
```

```
display: flex;
```

```
justify-content: center;
```

```
align-items: center;

}

.container {

width: 400px;

padding: 20px;

background: rgba(255, 255, 255, 0.8);

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

input[type="text"],

input[type="password"],

input[type="number"],

input[type="tel"],

input[type="submit"] {

width: 100%;

padding: 10px;

margin: 5px 0;

border: 1px solid #ccc;

border-radius: 3px;

box-sizing: border-box;

}

input[type="submit"] {

background-color: #4caf50;

color: white;
```

```
cursor: pointer;

}

input[type="submit"]:hover {

background-color: #45a049;

}

</style>
```

6.3 JAVASCRIPT:

```
<script>

// Simulated database

let users = [];

function showRegistrationForm() {

  document.getElementById('loginForm').style.display = 'none';

  document.getElementById('registrationForm').style.display = 'block';

}

function showLoginForm() {

  document.getElementById('registrationForm').style.display = 'none';

  document.getElementById('loginForm').style.display = 'block';

}

function loginUser() {

  let username = document.getElementById('loginUsername').value;

  let password = document.getElementById('loginPassword').value;

  let user = users.find(u => u.username === username && u.password === password);

  if (user) {

    alert("Login successful!");

    return true;

  } else {

    document.getElementById('loginErrorMessage').textContent = "Invalid username or password.";

    return false;

  }

}
```

```

}

}

function registerUser() {

let firstName = document.getElementById('firstName').value;

let lastName = document.getElementById('lastName').value;

let gender = document.getElementById('gender').value;

let age = document.getElementById('age').value;

let email = document.getElementById('newemail').value;

let username = document.getElementById('newUsername').value;

let password = document.getElementById('newPassword').value;

// Check if username already exists

if (users.some(u => u.username === username)) {

document.getElementById('registrationErrorMessage').textContent = "Username
already exists.";

return false;

}


// Simulate server-side password strength checking

if (password.length < 8) {

document.getElementById('registrationErrorMessage').textContent = "Password must
be at least 8 characters long.";

return false;

}


// Simulate server-side registration

```

```

users.push({firstName: firstName, lastName: lastName, gender: gender, age: age,
email: email, username: username, password: password});

alert("Registration successful!");

showLoginForm();

return false;

}

function loginUser() {

let username = document.getElementById('loginUsername').value;

let password = document.getElementById('loginPassword').value;

let user = users.find(u => u.username === username && u.password === password);

if (user) {

// Redirect to the payment page

window.location.href = "credit.html";

return false; // Prevent form submission

} else {

document.getElementById('loginErrorMessage').textContent = "Invalid username or
password.";

return false;

}

}

</script>

</body>

</html>

<script>

```

```

function processPayment() {

  // Simulate face recognition

    // Simulate payment processing

    let cardNumber = document.getElementById('cardNumber').value;

    let cardName = document.getElementById('cardName').value;

    let expiryDate = document.getElementById('expiryDate').value;

    let phone = document.getElementById('phone').value;

    let cvv = document.getElementById('cvv').value;

    if (cardNumber.length !== 16 || isNaN(cardNumber)) {

      alert("Invalid card number.");

      return false;

    }


    if (cvv.length !== 3 || isNaN(cvv)) {

      alert("Invalid CVV.");

      return false;

    }

    alert("GO TO TERMINAL!USER DETAILS:\nCard Number: " +
cardNumber.substring(0, 4) + " ** ** " + cardNumber.substring(12) + "\nName on
Card: " + cardName + "\nExpiry Date: " + expiryDate );

    return false;

  }

</script>

```

6.4 FLASK APP.PY:

```
import cv2

import os

from flask import Flask, request, render_template

from datetime import date

from datetime import datetime

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

import pandas as pd

import joblib


# Defining Flask App

app = Flask(__name__)

nimgs = 10


# Saving Date today in 2 different formats

datetoday = date.today().strftime("%m_%d_%y")

datetoday2 = date.today().strftime("%d-%B-%Y")

# Initializing VideoCapture object to access WebCam

face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')


# If these directories don't exist, create them

if not os.path.isdir('Entries'):

    os.makedirs('Entries')
```

```

if not os.path.isdir('static'):

    os.makedirs('static')

if not os.path.isdir('static/faces'):

    os.makedirs('static/faces')

if f'Entries-{datetoday}.csv' not in os.listdir('Entries'):

    with open(f'Entries/Entries-{datetoday}.csv', 'w') as f:

        f.write('Name,Roll,Time')


# get a number of total registered users

def totalreg():

    return len(os.listdir('static/faces'))

# extract the face from an image

def extract_faces(img):

    try:

        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        face_points = face_detector.detectMultiScale(gray, 1.2, 5, minSize=(20, 20))

        return face_points

    except:

        return []


# Identify face using ML model

def identify_face(facearray):

    model = joblib.load('static/face_recognition_model.pkl')

    return model.predict(facearray)

```

```
# A function which trains the model on all the faces available in faces folder
```

```
def train_model():
```

```
    faces = []
```

```
    labels = []
```

```
    userlist = os.listdir('static/faces')
```

```
    for user in userlist:
```

```
        for imgname in os.listdir(f'static/faces/{user}')
```

```
            img = cv2.imread(f'static/faces/{user}/{imgname}')
```

```
            resized_face = cv2.resize(img, (50, 50))
```

```
            faces.append(resized_face.ravel())
```

```
            labels.append(user)
```

```
    faces = np.array(faces)
```

```
    knn = KNeighborsClassifier(n_neighbors=5)
```

```
    knn.fit(faces, labels)
```

```
    joblib.dump(knn, 'static/face_recognition_model.pkl')
```

```
# Extract info from today's Entries file in Entries folder
```

```
def extract_Entries():
```

```
    df = pd.read_csv(f'Entries/Entries-{datetoday}.csv')
```

```
    names = df['Name']
```

```
    rolls = df['Roll']
```

```
    times = df['Time']
```

```
    l = len(df)
```

```

    return names, rolls, times, l

# Add Entries of a specific user

def add_Entries(name):

    username = name.split('_')[0]

    userid = name.split('_')[1]

    current_time = datetime.now().strftime("%H:%M:%S")

    df = pd.read_csv(f'Entries/Entries-{datetoday}.csv')

    if int(userid) not in list(df['Roll']):

        with open(f'Entries/Entries-{datetoday}.csv', 'a') as f:

            f.write(f'\n{username},{userid},{current_time}')

## A function to get names and rol numbers of all users

def getallusers():

    userlist = os.listdir('static/faces')

    names = []

    rolls = []

    l = len(userlist)

    for i in userlist:

        name, roll = i.split('_')

        names.append(name)

        rolls.append(roll)

    return userlist, names, rolls, l

## A function to delete a user folder

```

```

def deletefolder(duser):

    pics = os.listdir(duser)

    for i in pics:

        os.remove(duser+'/'+i)

    os.rmdir(duser)

##### ROUTING FUNCTIONS #####

# Our main page

@app.route('/')

def home():

    names, rolls, times, l = extract_Entries()

    return render_template('home.html', names=names, rolls=rolls, times=times, l=l,
totalreg=totalreg(), datetoday2=datetoday2)

## List users page

@app.route('/listusers')

def listusers():

    userlist, names, rolls, l = getallusers()

    return render_template('listusers.html', userlist=userlist, names=names, rolls=rolls,
l=l, totalreg=totalreg(), datetoday2=datetoday2)

## Delete functionality

@app.route('/deleteuser', methods=['GET'])

def deleteuser():

    duser = request.args.get('user')

    deletefolder('static/faces/'+duser)

    ## if all the face are deleted, delete the trained file...

```

```

if os.listdir('static/faces/')==[]:

    os.remove('static/face_recognition_model.pkl')

try:

    train_model()

except:

    pass

userlist, names, rolls, l = getallusers()

return render_template('listusers.html', userlist=userlist, names=names, rolls=rolls,
l=l, totalreg=totalreg(), datetoday2=datetoday2)

# Our main Face Recognition functionality.

# This function will run when we click on Take Entries Button.

@app.route('/start', methods=['GET'])

def start():

    names, rolls, times, l = extract_Entries()

    if 'face_recognition_model.pkl' not in os.listdir('static'):

        return render_template('home.html', names=names, rolls=rolls, times=times, l=l,
totalreg=totalreg(), datetoday2=datetoday2, mess='There is no trained model in the
static folder. Please add a new face to continue.')

    ret = True

    cap = cv2.VideoCapture(0)

    while ret:

        ret, frame = cap.read()

        if len(extract_faces(frame)) > 0:

            (x, y, w, h) = extract_faces(frame)[0]

```

```

cv2.rectangle(frame, (x, y), (x+w, y+h), (86, 32, 251), 1)

cv2.rectangle(frame, (x, y), (x+w, y-40), (86, 32, 251), -1)

face = cv2.resize(frame[y:y+h, x:x+w], (50, 50))

identified_person = identify_face(face.reshape(1, -1))[0]

add_Entries(identified_person)

cv2.putText(frame, f'{identified_person}', (x+5, y-5),

            cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

cv2.imshow('Entries', frame)

if cv2.waitKey(1) == 27:

    break

cap.release()

cv2.destroyAllWindows()

names, rolls, times, l = extract_Entries()

return render_template('home.html', names=names, rolls=rolls, times=times, l=l,
totalreg=totalreg(), datetoday2=datetoday2)

```

```

# A function to add a new user.

```

```

# This function will run when we add a new user.

```

```

@app.route('/add', methods=['GET', 'POST'])

```

```

def add():

```

```

    newusername = request.form['newusername']

```

```

    newuserid = request.form['newuserid']

```

```

    userimagefolder = 'static/faces/'+newusername+'_'+str(newuserid)

```

```

    if not os.path.isdir(userimagefolder):

```

```

        os.makedirs(userimagefolder)

i, j = 0, 0

cap = cv2.VideoCapture(0)

while 1:

    _, frame = cap.read()

    faces = extract_faces(frame)

    for (x, y, w, h) in faces:

        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 20), 2)

        cv2.putText(frame, f'Images Captured: {i}/{nimgs}', (30, 30),

                    cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 20), 2, cv2.LINE_AA)

    if j % 5 == 0:

        name = newusername+'_'+str(i)+'.jpg'

        cv2.imwrite(userimagefolder+'/'+name, frame[y:y+h, x:x+w])

        i += 1

        j += 1

    if j == nimgs*5:

        break

    cv2.imshow('Adding new User', frame)

    if cv2.waitKey(1) == 27:

        break

cap.release()

cv2.destroyAllWindows()

print('Training Model')

train_model()

```

```
names, rolls, times, l = extract_Entries()

return render_template('home.html', names=names, rolls=rolls, times=times, l=l,
totalreg=totalreg(), datetoday2=datetoday2)

# Our main function which runs the Flask App

if __name__ == '__main__':

    app.run(debug=True)
```

CHAPTER 7

OUTPUT

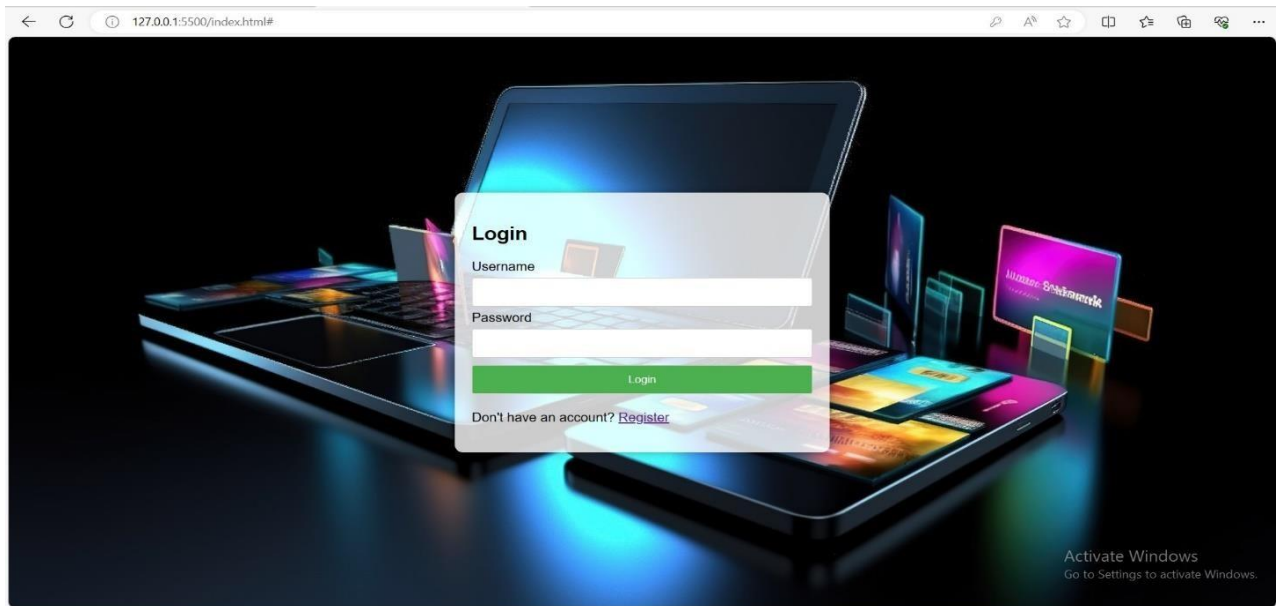


Fig 1.2 LOGIN PAGE

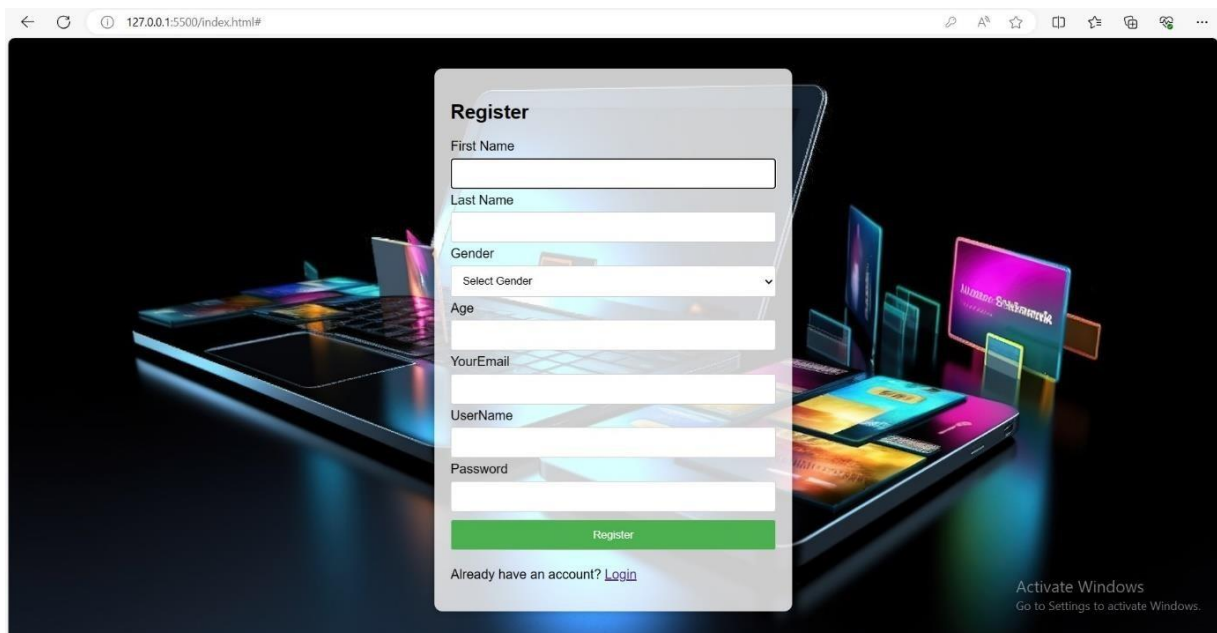


Fig 1.3 REGISTRATION PAGE

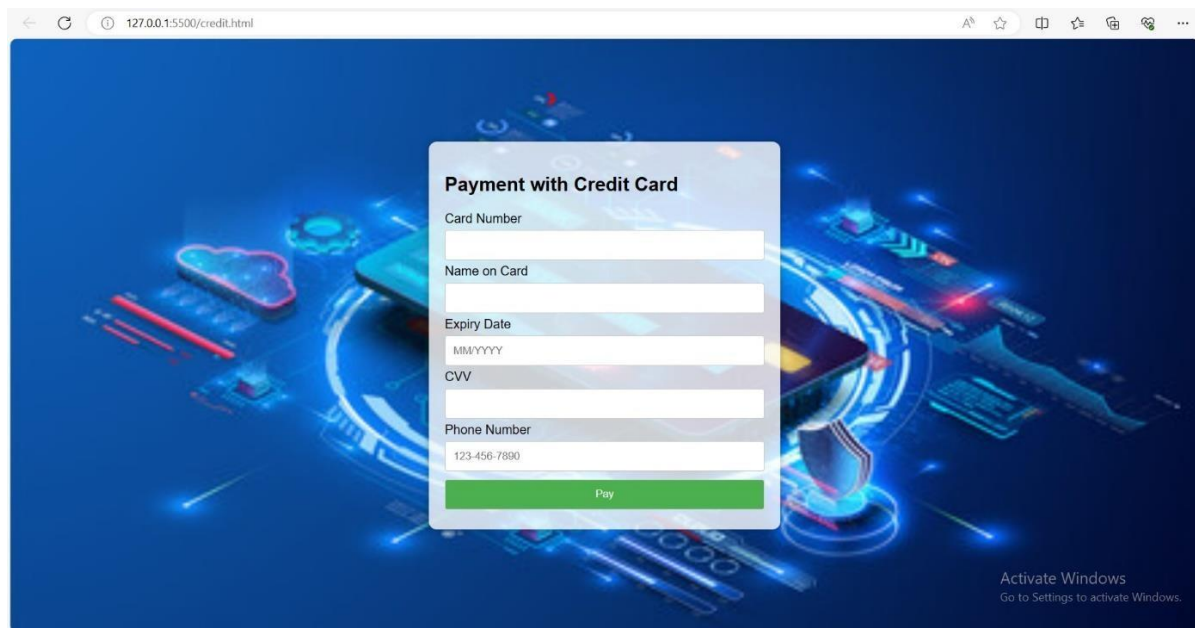


Fig 1.4 CREDIT DETAILS PAGE

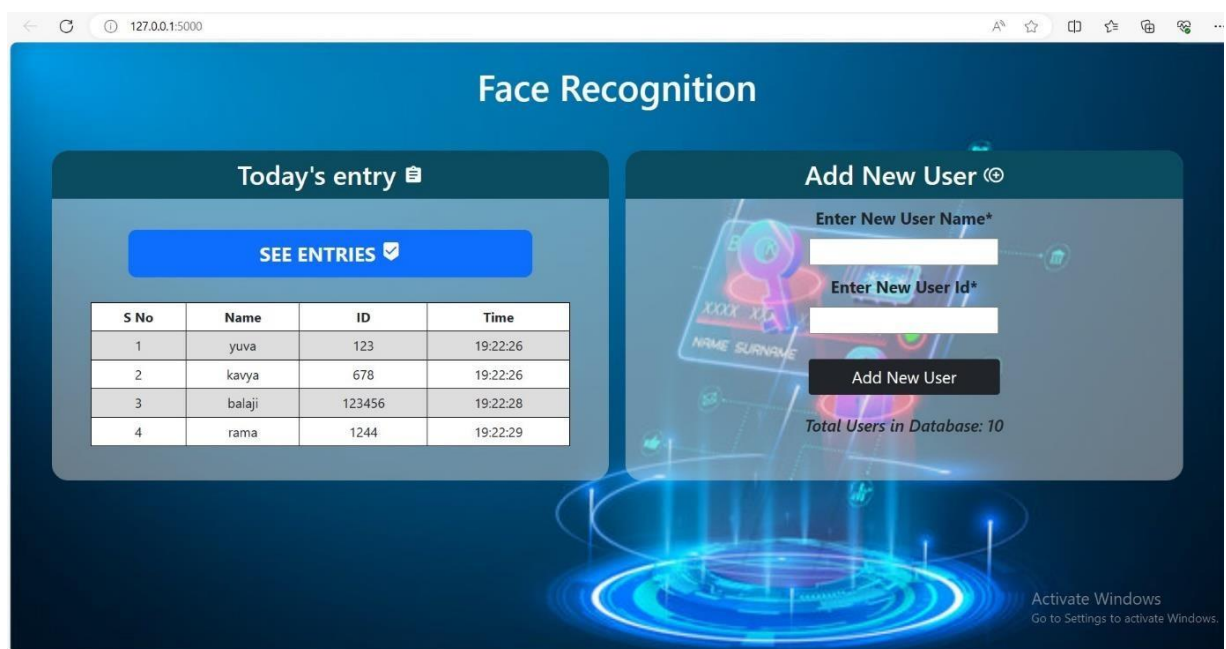


Fig 1.5 FACE RECOGNITION PAGE

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Face-Recognition\face-recognition-website> flask run
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [18/Apr/2024 10:10:49] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Apr/2024 10:11:18] "GET / HTTP/1.1" 200 -
```

```
request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args) # type: ignore[no-any-return]
    File "C:\Face-Recognition\face-recognition-website\app.py", line 179, in start
    cv2.imshow('Entries', frame)
cv2.error: OpenCV(4.9.0) D:\a\opencv-python\opencv-python\opencv\modules\highgui\src\window.cpp:971: error: (-215:Assertion failed) size.width>0 && size.height>0 in function 'cv::imshow'

127.0.0.1 - - [18/Apr/2024 10:31:57] "GET /start HTTP/1.1" 500 -
[ WARN:10@351.521] global cap_msmf.cpp:475 'anonymous-namespace':::SourceReaderCB::OnReadSample videoio(MSMF): OnReadSample() is called with error status: -1072873821
[ WARN:10@351.526] global cap_msmf.cpp:487 'anonymous-namespace':::SourceReaderCB::OnReadSample videoio(MSMF): async ReadSample() call is failed with error status: -1072873821
[ WARN:90@351.900] global cap_msmf.cpp:1768 CvCapture_MSMF::grabFrame videoio(MSMF): can't grab frame. Error: -1072873821
[2024-04-18 10:32:05.176] ERROR in app: Exception on /start [GET]
Traceback (most recent call last):
  File "C:\Users\LENOVO\AppData\Local\Programs\Python\Python312\Lib\site-packages\flask\app.py", line 1463, in wsgi_app
    response = self.full_dispatch_request()
               ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\LENOVO\AppData\Local\Programs\Python\Python312\Lib\site-packages\flask\app.py", line 872, in full_dispatch_request
    rv = self.handle_user_exception(e)
         ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\LENOVO\AppData\Local\Programs\Python\Python312\Lib\site-packages\flask\app.py", line 870, in full_dispatch_request
    rv = self.dispatch_request()
         ^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\LENOVO\AppData\Local\Programs\Python\Python312\Lib\site-packages\flask\app.py", line 855, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args) # type: ignore[no-any-return]
```

Fig 1.6 TERMIALNAL RUN

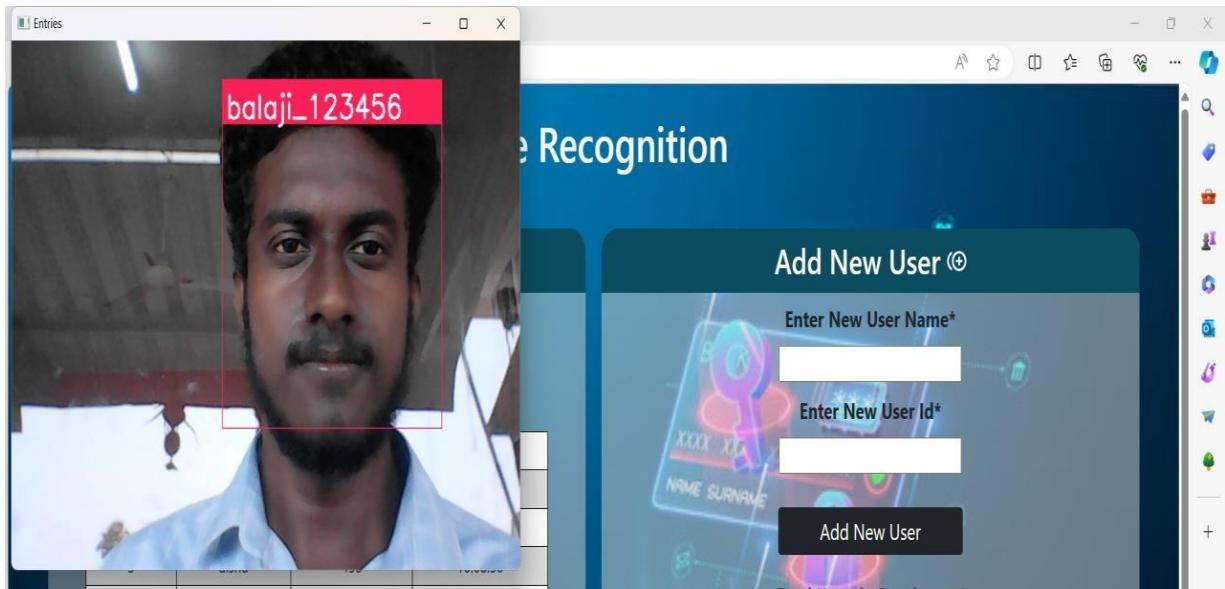


Fig 1.7 TEST CASE IMAGE

CHAPTER 8

CONCLUSION

In conclusion, the implementation of credit card authentication using facial recognition technology, facilitated by a Flask app with HTML, CSS, and JavaScript, represents a significant advancement in online security and user authentication methods. By harnessing the power of facial recognition algorithms within a Flask backend, coupled with intuitive frontend interfaces developed using HTML, CSS, and JavaScript, this system offers users a seamless and secure authentication experience. The integration of Flask as the backend framework enables efficient handling of HTTP requests and responses, while also providing a robust platform for integrating facial recognition algorithms for user authentication. The HTML, CSS, and JavaScript frontend components enhance user interaction by providing intuitive interfaces for capturing facial images and facilitating the authentication process. Furthermore, the use of facial recognition technology adds an additional layer of security to the credit card authentication process, mitigating the risks associated with traditional authentication methods such as passwords or PINs. By leveraging biometric data unique to each individual, the system enhances security while also offering convenience and ease of use for users.

In summary, the authentication of credit card transactions using facial recognition technology, powered by a Flask app and complemented by HTML, CSS, and JavaScript, represents a sophisticated and effective solution for enhancing online security and user authentication in the digital age.

CHAPTER 9

REFERENCES

- [1]. Sebastian Zientek, Wojciech Wodo, “Biometric linkage between identity document card and its holder based on real-time facial recognition”, July 28-30, 2017.
- [2]. Matthew A. Turk and Alex P. Pentland, “Face Recognition Using Eigenfaces”, 1991.
- [3]. Timo Ahonen, Matti Pietikainen, “Face Description with Local Binary Patterns: Application to Face Recognition”, IEEE, DECEMBER 2006.
- [4]. Fatma zohra CHELALI. A. DJERABA and R. DJERIDI, “Linear Discriminant Analysis for Face Recognition”, IEEE, 2009
- [5]. Method for secure credit card transaction, Nader Nassar, Grant Miller, International Conference, 2013.
- [6]. Credit card fraud detection based on transaction behaviour, John Richard, Larry A. Veal, TENCON, 2017.
- [7]. Credit fraud card detection, Andrea, Giacomini, Olivier Cealun, IEEE International Conference, 2018.
- [8]. Credit card fraudulent Transaction Detection, IEEE International Conference, 2018.
- [9]. Credit card fraud detection using machine learning, John Williams, 7th IEEE International Conference, 2017.
- [10]. Deep learning detecting fraud in credit card transaction, Abhimanyu Roy, Loreto Alonzi, Peter Beling, System and information, 2018.
- [11]. Credit card fraud detection system, V. Filippov, System and information, 2008.
- [12]. Random forest for credit card fraud detection, Lutao Zheng, Shuo Wang, IEEE 15th International conference, 2018.
- [13]. Adaptive credit card fraud detection System, KK Sherly, IEEE International conference,

2018.

[14].Detecting credit card fraud using periodic features, Alejandro, Bjorn, IEEE International Conference, 2015.

[15].MN BORHAN,"Design Of The High Speed And Reliable Source Coupled Logic Multiplexer",Journal of VLSI Circuits And Systems 1 (01), 2019

[16].Mv Ngo Tien HoA,High Speed And Reliable Double Edge Triggered D- Flip-Flop For Memory

Applications",Journal of VLSI Circuits And Systems, 1 (01), 2019