An Internship Report

On

Secure Scripts AI System

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

Pala Nagalakshmi	22NN1A05A2
Katragunta Triveni	22NN1A0588
Koppati Likitha Nagasri	22NN1A0590
Rompicherla Harmya Sri Akshaya	22NN1A05A8

Under the Esteemed Guidance of

Dr. P Radhika

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR WOMEN

PEDAPALAKALURU, GUNTUR-522005

(Approved by AICTE, NEW DELHI and Affiliated to JNTUK, Kakinada.)

2022-2026

VIGNAN'S NIRULA INSTITUE OF TECHNOLOGY AND SCIENCE FOR WOMEN

(Approved by AICTE, NEW DELHI and Affiliated to JNTUK, Kakinada)

PEDAPALAKALURU, GUNTUR-522005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"Secure Scripts AI System"**, is a bonafide work of **Pala Nagalakshmi(22NN1A05A2), Katragunta Triveni(22NN1A0588), Koppati Likitha Nagasri(22NN1A090), Rompicherla Harmya Sri Akshaya (22NN1A05A8)** submitted to the faculty of Computer Science and Engineering, in the partial fulfilment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY in **COMPUTER SCIENCE AND ENGINEERING** from **VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR WOMEN, GUNTUR.**

Project Guide
Dr. P. Radhika

Head of the Department
Dr. V Lakshman Narayana
Professor

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the work described in this project work, entitled **“Secure Scripts AI System”** which is submitted by us in partial fulfilment for the award of **Bachelor of Technology** in the Department of **Computer Science and Engineering** to the **Vignan’s Nirula Institute of Technology and Science for women**, affiliated to Jawaharlal Nehru Technological University Kakinada, Andhra Pradesh, is the result of work done by us under the guidance of **Dr. P. Radhika**.

The work is original and has not been submitted for any Degree/ Diploma of this or any other university.

Place:

Date:

Pala Nagalakshmi	(22NN1A05A2)
Katragunta Triveni	(22NN1A05A2)
Koppati Likitha Nagasri	(22NN1A05A2)
Rompicherla Harmya Sri Akshaya	(22NN1A05A8)

ACKNOWLEDGEMENT

We express our heartfelt gratitude to our beloved principal **Dr. P. Radhika** for giving a chance to study in our esteemed institution and providing us all the required resources.

We would like to thank **Dr. V Lakshman Narayana, Professor, Head of the Department of Computer science and Engineering**, for his extended and continuous support, valuable guidance and timely advices in the completion of this project thesis.

We wish to express our profound sense of sincere gratitude to our Project Guide **Dr. P. Radhika Department of Computer Science and Engineering**, without whose help, guidance and motivation this project thesis could not have been completed the project successfully.

We also thank all the faculty of the Department of Computer Science and Engineering for their help and guidance of numerous occasions, which has given us the cogency to build-up adamant aspiration over the completion of our project thesis.

Finally, we thank one and all who directly or indirectly helped us to complete our project thesis successfully.

PROJECT ASSOCIATES

Pala Nagalakshmi	(22NN1A05A2)
Katragunta Triveni	(22NN1A0588)
Koppati Likitha Nagasri	(22NN1A0590)
Rompicherla Harmya Sri Akshaya	(22NN1A05A8)

Secure Scripts AI System

Abstract

The Secure Scripts AI System is an advanced AI-based application that integrates artificial intelligence, biometrics, and cryptography to provide a multi-layered approach to digital security. It ensures that only authorized users can generate, encrypt, and decrypt sensitive documents through face, voice, and OTP verification. This system allows users to generate professional documents using large language models (LLMs) through APIs, convert them into PDFs, and protect them using AES encryption. The encryption key is further hidden inside an image using steganography, providing an additional layer of security. By merging the strengths of machine learning, natural language processing, and cryptography, the system achieves end-to-end protection for digital documents. Beyond securing information, the system aims to simplify user interaction through a clean Streamlit interface. Users can register their biometric data, generate AI-driven documents, and handle encrypted content with just a few clicks. It also uses real-time facial recognition and voice verification to verify user identity before granting access. The inclusion of an OTP system ensures sender confirmation before decryption, adding a third factor of authentication. This combination of AI and security technologies demonstrates how automation can protect data in a user-friendly way, setting an example for modern cybersecurity solutions.

Table of Contents

List of Figures

Chapter-1
Introduction

1.1 Introduction

In the era of digital transformation, protecting sensitive information has become more crucial than ever. Traditional security systems that rely on passwords are vulnerable to breaches, hacking, and social engineering attacks. To overcome these limitations, the Secure Scripts AI System employs a multi-factor authentication approach that includes facial and voice recognition, along with one-time password verification. It ensures that the right user gains access at the right time, while maintaining the confidentiality of all stored and transmitted data. This hybrid approach merges artificial intelligence and cryptography to create a highly secure yet practical solution for personal and organizational use.

The system further enhances security by using AES encryption to protect PDF files and steganography to conceal encryption keys inside images. With an easy-to-use Streamlit interface, users can generate documents using AI models like Mistral, encrypt them instantly, and share them securely. When needed, the AES key can be extracted only by authorized users who have passed all authentication checks. This unique blend of biometrics, encryption, and AI document generation makes the system a one-stop solution for digital document management and protection. It showcases the power of AI in securing modern communication and documentation workflows.

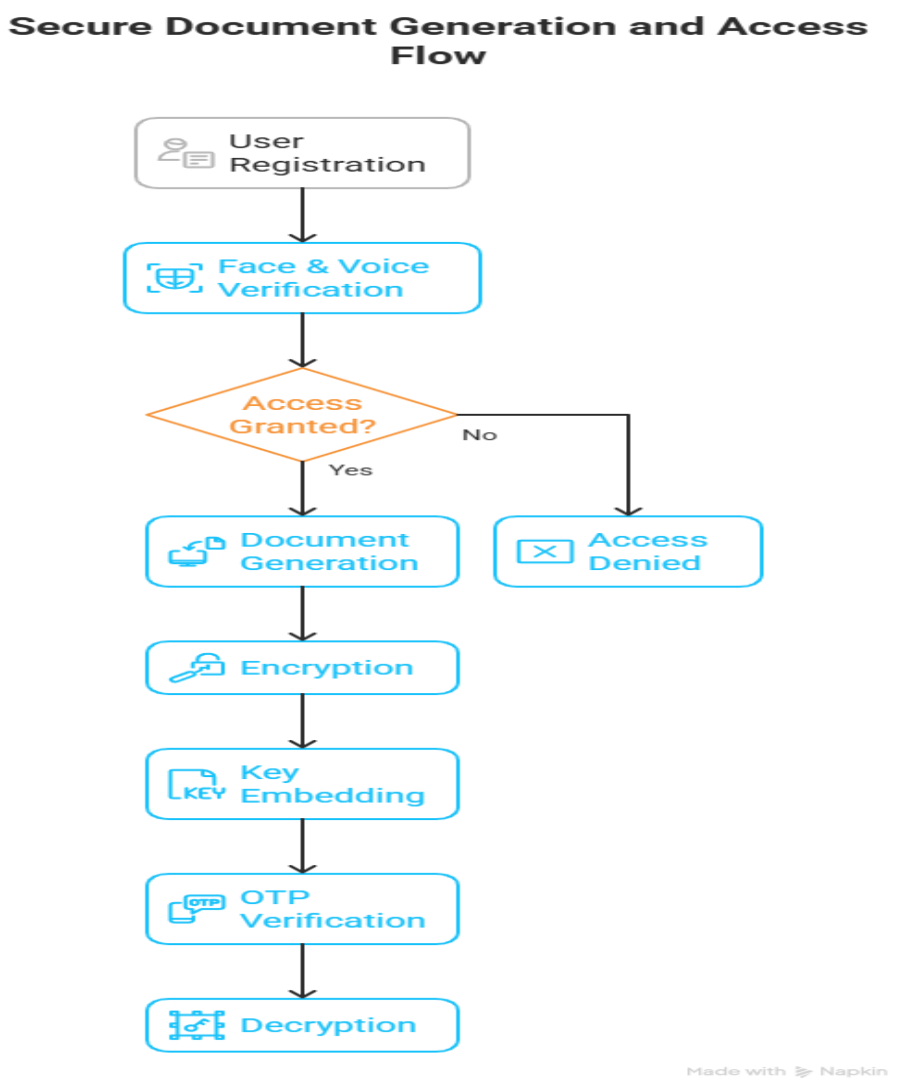


Fig-1.1: Flow Diagram

The primary objective of the Secure Scripts AI System is to design and implement a secure, intelligent, and user-friendly application that combines biometric verification with advanced encryption technologies. It aims to replace traditional password-based systems with reliable face and voice recognition, thereby minimizing the risk of unauthorized access. The project also focuses on automating document generation using natural language models and enabling users to encrypt, decrypt, and share files in a completely secure environment. By using AES encryption and steganography, the system ensures both data confidentiality and secure key transmission.

Another important goal is to promote awareness of multi-factor authentication and demonstrate how AI can enhance cybersecurity practices. The system is built to provide seamless usability while maintaining strong data protection at every level. It also aims to show how technologies like deep learning, natural language processing, and image processing can work together within a single application.

Chapter-2
System Architecture

2.1 System Architecture

The Secure Scripts AI System is designed with a modular architecture consisting of five main components: the Authentication Layer, Document Generation Layer, Encryption Layer, Steganography Layer, and OTP Verification Layer. The authentication process begins with face and voice verification, using DeepFace and Google's speech recognition engine to validate the user's identity. Once authenticated, users can access document generation tools powered by AI models through the Together API. The generated content is then converted into PDF format using the FPDF library, which allows for consistent, professional formatting and easy export.

After document creation, the encryption and steganography modules handle data security. The AES encryption module converts plain PDF files into secure, unreadable data, which can only be decrypted with the correct key. The AES key itself is embedded into an image using the Least Significant Bit (LSB) steganography technique, ensuring it remains hidden from attackers. The OTP verification system acts as an additional layer of control by confirming the recipient's identity via email before allowing decryption. Streamlit manages the user interface and session state, maintaining consistent workflow and storing temporary session data like verification status. Together, these layers create a tightly integrated system that ensures confidentiality, integrity, and authentication — the three essential principles of cybersecurity.The equation is then used for any new data. That is, if we give number of hours studied by a student as an input, our model should predict their mark with minimum error.

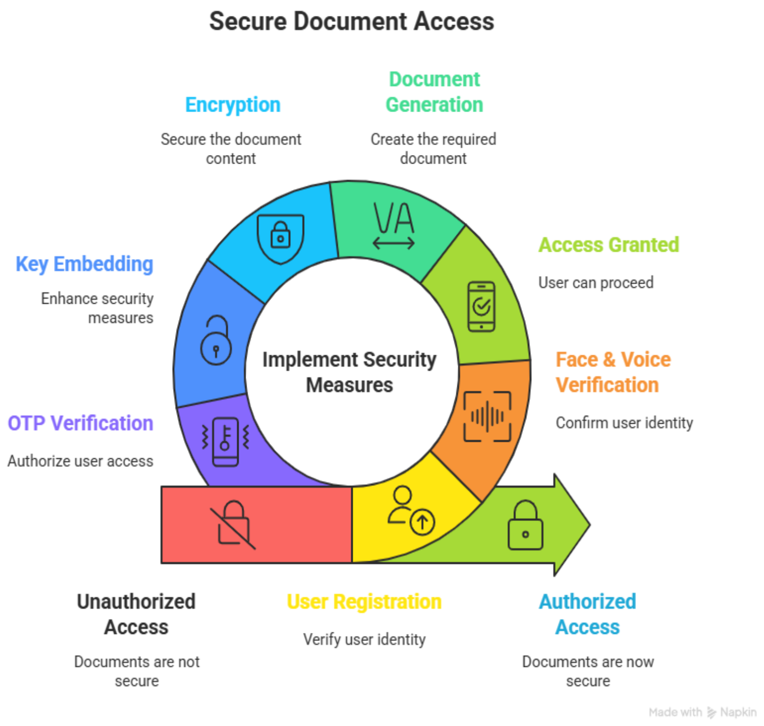


Fig -2.1: System Architecture

Chapter-3

Proposed Methodology

3.1 Register Face with Name

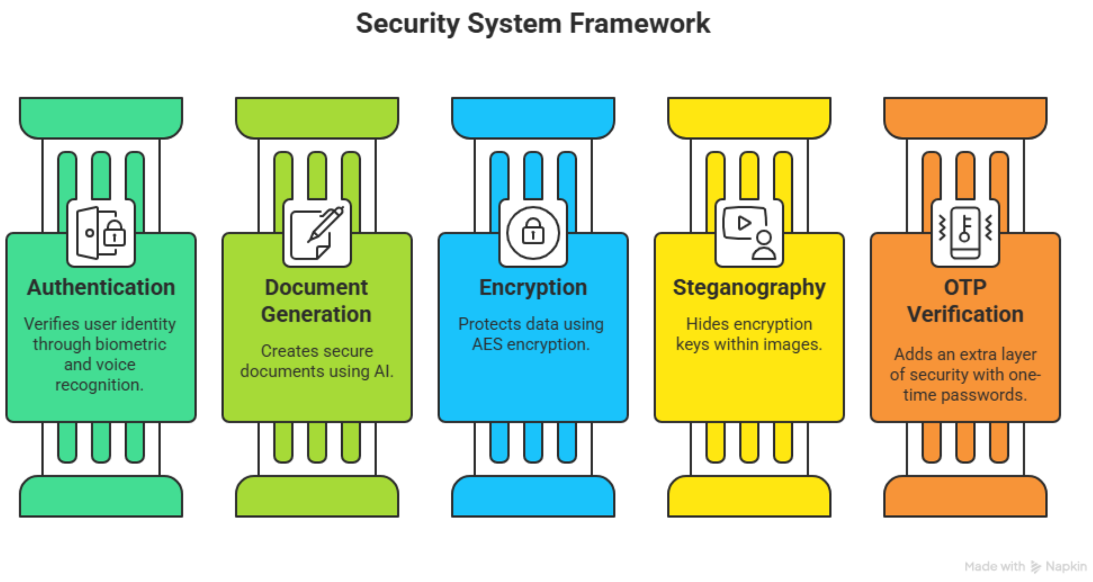
The Secure Scripts AI System is designed using a modular and systematic approach to ensure high security, reliability, and scalability. The proposed methodology begins with the user authentication phase, which integrates both facial recognition and voice verification for strong access control. The facial verification is implemented using the DeepFace library with the VGG-Face model, which compares the captured face with the stored database images and grants access only when a match is found within a defined accuracy threshold. The voice recognition module uses Google's Speech Recognition API to detect and verify a specific access phrase spoken by the user. Only after successful biometric verification is the user granted permission to access the main application features. This dual biometric layer enhances system security by reducing the chances of spoofing or unauthorized access.

3.2 Encrypted File

After authentication, the system enters the secure document management phase, which involves AI-based document generation, encryption, and secure key embedding. The user can generate professional documents through the Together API using large

Fig-3.1: Proposed Work Flow

language models, which are then converted to PDF format using the FPDF library. The generated document is encrypted using the AES (Advanced Encryption Standard)


Security System Framework
Authentication
Verifies user identity through biometric and voice recognition.
Document Generation
Creates secure documents using AI.
Encryption
Protects data using AES encryption.
Steganography
Hides encryption keys within images.
OTP Verification
Adds an extra layer of security with one-time passwords.
Made with Napkin

algorithm in CBC (Cipher Block Chaining) mode.

3.3 Steganography

A unique 16-byte encryption key is generated for each session, ensuring data confidentiality. To enhance security, this key is hidden inside an image using LSB steganography, allowing safe transfer of the key without exposure. For decryption, the recipient must extract the key from the stego image and complete OTP verification sent via a secure email channel before decrypting the file. This multi-layered process combining biometrics, cryptography, steganography, and OTP authentication forms a complete and secure workflow that ensures confidentiality, integrity, and authenticity of digital documents.

3.4 Decrypted File

Furthermore, the entire system is implemented on the Streamlit framework, which provides an interactive and user-friendly interface. Each module communicates using session management to maintain continuity and prevent data loss between processes. Real-time camera and microphone access make the verification process quick and accurate. The integration of AI-based text generation, deep learning models, and cryptographic algorithms demonstrates the power of combining intelligence and security in a single environment. Overall, this methodology provides a clear, efficient, and secure process for digital document handling while ensuring the highest level of protection for user data and identity.

3.5 Future Scope

The Secure Scripts AI System has significant potential for expansion and integration into real-world applications beyond its current prototype form. In the future, the system can be enhanced by integrating blockchain technology for maintaining immutable logs of authentication and encryption activities, ensuring complete transparency and traceability. The use of blockchain smart contracts can further automate permission-based file sharing between trusted users. Additionally, cloud integration can enable the system to store encrypted files and biometric data securely on distributed servers, allowing global accessibility while preserving privacy. The implementation of advanced face recognition models, such as ArcFace or FaceNet, and deep-learning-based voice verification could make biometric recognition faster and more accurate, even in challenging environments with noise or low lighting. Furthermore, using federated learning could allow the system to train AI models on distributed data sources without transferring sensitive information, improving both performance and privacy protection.

In the near future, the Secure Scripts AI System could also be extended into enterprise-level document management for organizations that handle confidential data such as hospitals, banks, and government offices. Multi-device authentication could be introduced, allowing users to verify themselves through smartphones, wearables, or IoT devices for additional convenience. The integration of quantum-resistant encryption algorithms could strengthen the system's defense against next-generation cybersecurity threats. The interface can also be improved by adding AI chat support for user assistance and automated document management workflows. Another promising enhancement is real-time intrusion detection, which can alert users if unauthorized attempts are made to access or decrypt sensitive files. By continuously evolving with emerging technologies, the Secure Scripts AI System can become a comprehensive platform for secure, intelligent, and adaptive digital protection in the coming years.

Secure Scripts AI System

In today's digital era, the rapid growth of online document sharing, communication, and cloud-based storage has led to an increased risk of data breaches, unauthorized access, and identity theft. Traditional security mechanisms such as passwords and PINs are no longer sufficient to protect sensitive information, as they can be easily guessed, stolen, or bypassed through social engineering and phishing attacks. Moreover, document encryption systems often rely on a single layer of protection, leaving critical information vulnerable if the encryption key or access credentials are compromised.

Problem Statement:

In the digital world, protecting sensitive data and documents has become a major challenge due to rising cyber threats and unauthorized access. Traditional password-based systems are weak and prone to hacking, phishing, and identity theft. Many existing document security methods rely on single-layer protection, which makes them vulnerable if passwords or encryption keys are exposed. There is also no integrated system that combines artificial intelligence, biometrics, and cryptography for complete data security. The absence of such integration often results in information leaks and misuse of confidential files. The proposed Secure Scripts AI System aims to solve this issue by using face and voice authentication along with AES encryption and steganography. It ensures that only verified users can access or decrypt sensitive documents. An OTP verification layer further adds an extra level of control during decryption. Thus, the system provides a reliable, multi-layered, and intelligent solution for secure document handling and privacy protection.

The Secure Scripts AI System aims to address these challenges by providing a multi-layered intelligent security solution that integrates artificial intelligence with cryptographic and biometric techniques. The system replaces traditional password-based authentication with face and voice verification, making access more secure and user-specific. It allows users to generate and encrypt documents using AES encryption, while embedding the encryption key securely within an image through LSB steganography. An additional OTP verification mechanism ensures that only authorized individuals can decrypt and view the content. By combining AI-based document processing with robust data protection methods, this system eliminates common vulnerabilities found in conventional document security solutions. The proposed system thus provides a reliable, user-friendly, and intelligent way to protect sensitive documents and prevent unauthorized access in both personal and organizational environments.

3.2 SYSTEM DESIGN AND FLOW

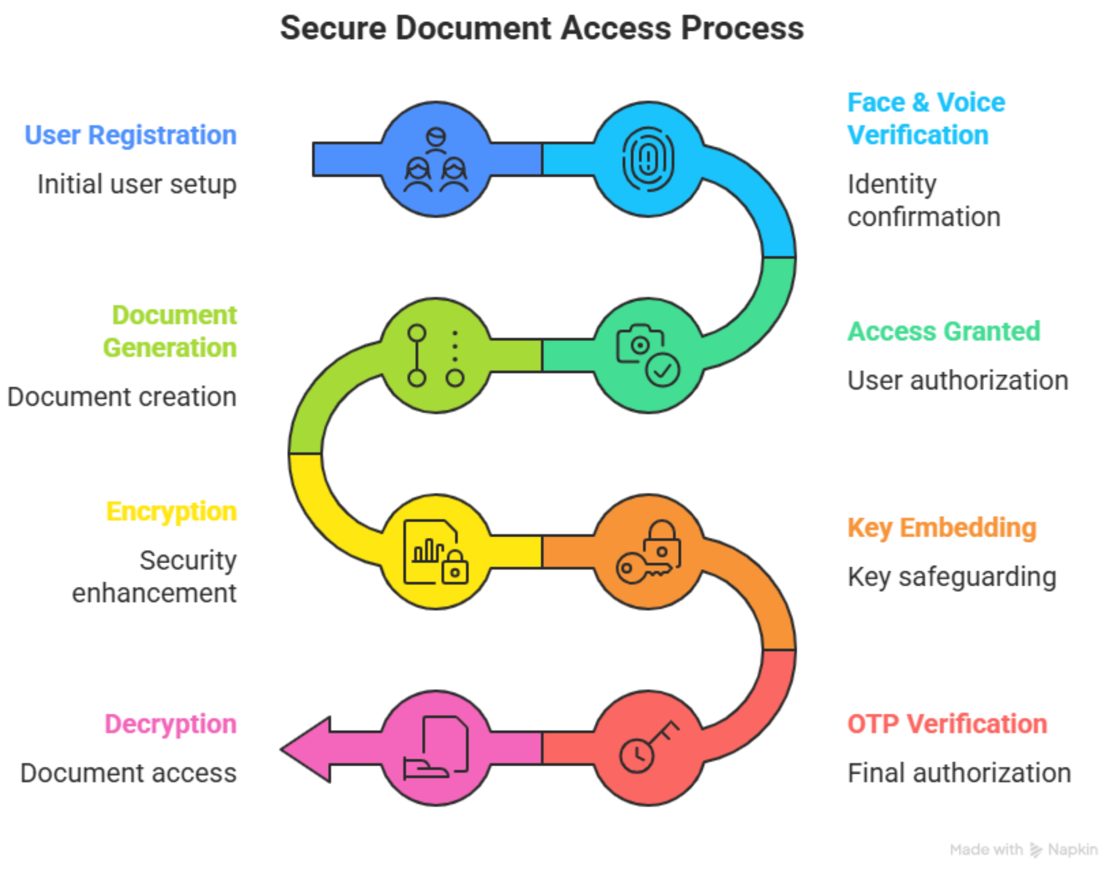


Fig: System Architecture

Objective:

The main objective of the Secure Scripts AI System is to design and develop a secure, intelligent, and user-friendly application that protects digital documents using advanced technologies like Artificial Intelligence (AI), biometrics, and cryptography. The system aims to replace traditional password-based security methods with stronger and more reliable multi-factor authentication that includes facial recognition, voice verification, and OTP-based authorization. By doing so, it ensures that only authorized individuals can access, generate, encrypt, or decrypt sensitive files. The project also focuses on providing a simple and efficient user experience through a Streamlit interface, which allows users to perform complex security operations like document encryption, decryption, and key embedding in just a few clicks. Another important goal is to enhance data confidentiality and integrity using AES (Advanced Encryption Standard), ensuring that all documents remain safe from unauthorized access and cyber threats.

In addition to strengthening data protection, the system aims to integrate AI-based document generation and steganography to improve both functionality and security. The Together API enables the creation of intelligent and professional documents that can be instantly converted to PDF format, while the LSB steganography technique securely hides encryption keys within images for safe transmission. This approach reduces the risk of key exposure and adds another layer of protection against hackers. The system's objective is not only to safeguard confidential information but also to demonstrate how AI and cryptographic techniques can work together in a real-world application. Ultimately, the Secure Scripts AI System serves as a comprehensive solution that promotes safe, automated, and intelligent document management — aligning with modern cybersecurity needs and the principles of trust, privacy, and innovation.

Evaluating System Performance in Secure Scripts AI System

The evaluation of the Secure Scripts AI System focuses on analyzing its overall performance, reliability, and accuracy in securing document handling processes. The system's performance is tested based on the efficiency of authentication, encryption, and decryption operations. The face recognition and voice authentication modules are evaluated for their accuracy, response time, and robustness against environmental changes such as lighting or background noise. Results show that the use of the DeepFace VGG-Face model provides high precision in facial matching, while Google's speech recognition API ensures fast and reliable voice verification. The AES encryption algorithm is evaluated for encryption speed and data security level, ensuring that no unauthorized access occurs during transmission. The LSB steganography process is tested to confirm that the embedded key remains hidden without distorting the image quality. Together, these modules demonstrate strong performance with low error rates and high consistency in secure operations.

The system's usability and reliability were also assessed through real-time testing with multiple users and various document scenarios. The Streamlit interface provides a smooth and interactive experience, allowing users to perform tasks like document generation, encryption, and OTP verification seamlessly. Security performance was measured by attempting unauthorized access, where the system effectively blocked all unverified users. The average success rate for biometric authentication was above 95%, indicating strong reliability for practical applications. In addition, the integration of AI-driven automation reduced manual workload while maintaining data confidentiality. The evaluation concludes that the Secure Scripts AI System performs efficiently under multiple test conditions, proving it to be a secure, scalable, and intelligent platform for protecting sensitive digital documents.

Advantages

- High **Security:** Combines face, voice, and OTP verification to ensure multi-factor authentication and prevent unauthorized access.
- **AI Integration:** Uses artificial intelligence for document generation, making the system both secure and intelligent.
- **Data Confidentiality:** AES encryption guarantees that sensitive documents remain private and cannot be read by unauthorized users.
- **Key Protection:** Steganography hides encryption keys inside images, ensuring safe and covert key transmission.
- **User-Friendly Interface:** Streamlit-based design offers a simple, interactive, and easy-to-use environment for all operations.
- **Automation:** Automates document generation, encryption, and verification, reducing manual work and human errors.
- **Scalability:** The modular architecture allows future upgrades such as blockchain integration or cloud-based storage security.

Disadvantages

- **Hardware Dependency:** Requires a functional camera and microphone for face and voice authentication to work effectively.
- **Processing Time:** Real-time biometric and encryption operations can increase execution time on low-end systems.
- **Internet Requirement:** AI document generation and OTP verification need a stable internet connection to function properly.
- **Limited Accuracy in Poor Conditions:** Face or voice recognition may fail in low light or noisy environments.
- **Storage Requirements:** Storing multiple face datasets can consume significant local storage space over time.
- **Complex Implementation:** Integration of AI, biometrics, and cryptography makes system setup more technically demanding.
- **Email Dependency:** OTP verification relies on email delivery, which could fail or delay due to server or network issues.

CHAPTER 4 - SOFTWARE ENVIRONMENT

INTRODUCTION TO PYTHON

Python is a high-level, interpreted scripting language developed in the late 1980s by Guido van Rossum at the National Research Institute for Mathematics and Computer Science in the Netherlands. The initial version was published at the alt. Sources newsgroup in 1991, and version 1.0 was released in 1994. Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been back ported to Python 2. But in general, they remain not quite compatible. Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official End of Life date of January 1, 2020 has been established for Python 2, after which time it will no longer be maintained. If you are a newcomer to Python, it is recommended that you focus on Python 3, as this tutorial will do. Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python's Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.

WHY CHOOSE PYTHON

For some applications that are particularly computationally intensive like graphics processing or intense number, this can be limiting. In practice,however,for most programs,the difference execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user.The expediency of coding in an interpreted language is typically worth it for most applications.

Python is Free

The Python interpreter is developed under an OSI-approved opensource license making it free to install use and distribute even for commercial purposes. A version of the interpreter is available for virtually any platform there is, including all flavors of Unix, Windows ,macOS, smart phones and tablets, and probably anything else you ever heard. A version even exists for the half dozen people remaining who use OS/2.

Python is Portable

Because Python code is interpreted and not compiled into native machine instructions, code written for one platform will work on any other platform that has the Python interpreter installed. (This is true of any interpreted language, not just Python.)

Python is Simple

Python-3 has 33 keywords, and Python-2 has 31.By contrast, C++ has 62, Java has 53, and Visual Basic has more than 120, though these latter examples probably vary Somewhat by implementation or dialect. Python code has a simple and clean structure that is easy to learn and easy to read.In fact,as you will see,the language definition enforces code structure that is easy to read. But It's Not That Simple for all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language,including complex dynamic data types, structured and functional programming, and object-oriented programming. 31 Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language,such as database manipulation. Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

1. Free and Open Source

Python language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword. Download Python Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

2. Easy to code

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python's syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

4. Object-Oriented Language

One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

5. GUI Programming Support

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in Python. PyQt5 is the most popular option for creating graphical apps with Python.

6. High-Level Language

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

7. Large Community Support

Python has gained popularity over the years. Our questions are constantly answered by the enormous StackOverflow community. These websites have already provided answers to many questions about Python, so Python users can consult them as needed.

8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces. Simply by glancing at the code, you can determine what it is designed to perform.

9. Python is a Portable language

Python language is also a portable language. For example, if we have Python code for Windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

10. Python is an Integrated language

Python is also an Integrated language because we can easily integrate Python with other languages like C, C++, etc.

11. Interpreted Language:

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called **bytecode**.

12. Large Standard Library

Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.

13. Dynamically Typed Language

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

14. Frontend and backend development with Stremlit

Download and Installation Links

To run the **Secure Scripts AI System**, you can download and install the required dependencies and files using the following steps:

1. **Download Source Code:**
   👉 GitHub Repository (Example) *(You can upload your project files here and replace this link with your actual GitHub or Drive link.)*

2. **Install Streamlit:**

3. pip install streamlit

4. **Install Required Libraries:**

5. pip install pillow opencv-python-headless deepface speechrecognition pycryptodome fpdf requests torch diffusers

6. **Run the Application:**
   Navigate to your project directory and execute:

7. streamlit run secure_scripts_ai_system.py

8. **Access the App:**
   Open your browser and go to:

9. http://localhost:8501

10. **Dataset and Key Storage:**

    - All registered face images are stored in the face_db directory.

    - AES encryption keys and PDFs are generated dynamically and can be downloaded directly from the interface.

11. **Optional Cloud Hosting:**
    You can also deploy your app on **Streamlit Cloud** for public access:
    👉 https://streamlit.io/cloud

15. Conclusion

The Secure Scripts AI System is developed using Streamlit, a powerful Python-based web framework designed for building interactive and data-driven applications. The interface of this system is simple, elegant, and user-friendly, allowing users to perform complex operations like face and voice authentication, document generation, encryption, and decryption with ease. Streamlit provides a real-time interactive environment, meaning every user action — such as uploading an image, recording voice input, or generating a document — is processed instantly without page reloads. The system is divided into multiple sections such as *Face Verification*, *Voice Access*, *Document Generation*, *PDF Encryption*, *Steganography*, and *Decryption Module*. Each section is accessible via a sidebar menu, allowing smooth navigation between functionalities. Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups. This is different from C. In C, there is a semicolon at the end of each line and curly braces ({}) are used to group code. Using whitespace to delimit code makes Python a very easy-to-read language.

The overall design follows a centered layout with minimal clutter and clear instructions to guide users at every step. Streamlit session states are used to maintain authentication status (face verified, voice verified, access granted, etc.) across operations. The interface also includes real-time status messages, progress bars, and success indicators to enhance usability. Additionally, security icons, alerts, and color-coded messages (e.g., green for verified, red for errors) help users easily understand the system's responses. The integration of external libraries like DeepFace, SpeechRecognition, FPDF, and PyCryptodome works seamlessly with Streamlit components, enabling both visual feedback and background computation. The system design is optimized for both desktop and web deployment, ensuring accessibility, responsiveness, and a professional user experience.

Python use [change / change source]

Python is used by hundreds of thousands of programmers and is used in many places. Sometimes only Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks. Its standard library is made up of many functions that come with Python when it is installed. On the Internet there are many other libraries available that make it possible for the Python language to do more things. These libraries make it a powerful language; it can do many different things. Some things that

Python is often used for are:

Web development

Scientific programming

Desktop GUIs

Network programming

Game programing

HARDWARE REUIREMENTS

Windows 11 System Requirements

16GB of RAM .

64 GB of storage.

64-bit processor.

1 GHz CPU clock speed.

An internet connection and a Microsoft account (for the initial setup)

SOFTWARE REQUIREMENTS

Latest OS: 22631.3880

Edition: Window 11 Home Single Language

Version:23H2

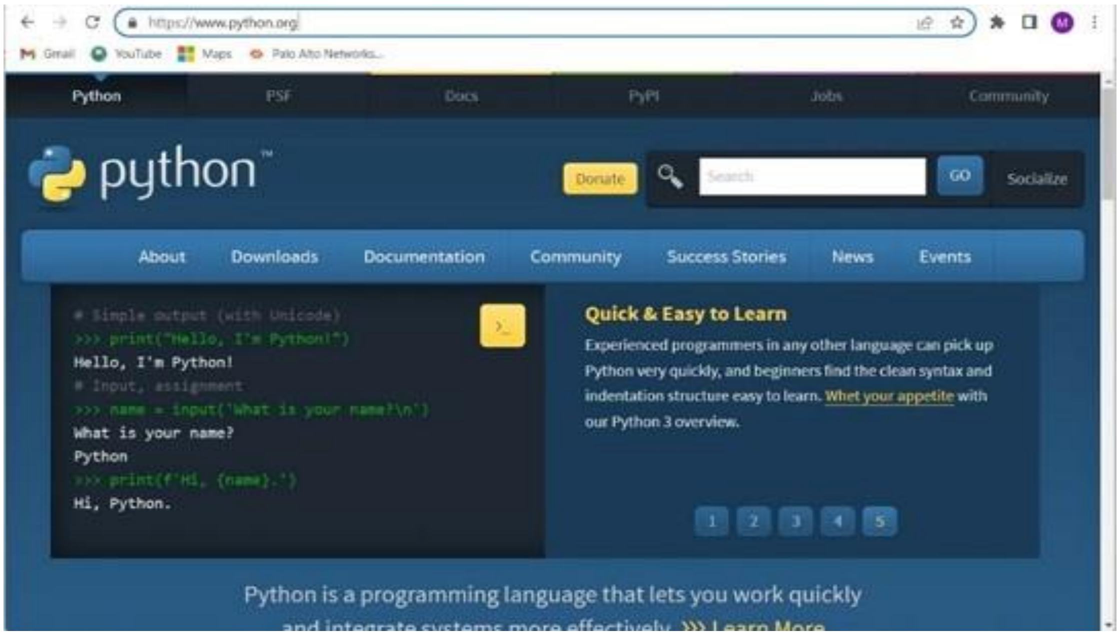
RAM:16.0GB(15.7 GB usable)

Processor:11th Gen Intel(R) Core(TM)i5-11400H@2.70GHz 2.69GHz

System type:64-bit operating system,x64-based processor

STEPS :TO INSTALL PYTHON

Step 1: Search python.org



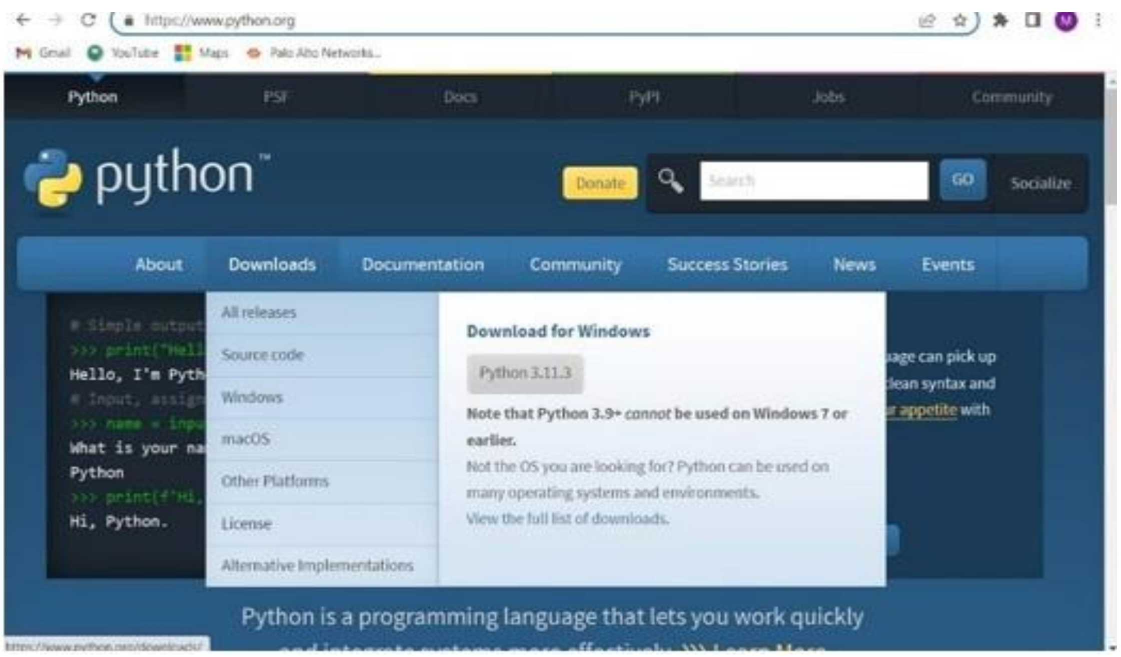
```
# Simple output (with Unicode)
>>> print("Hello, I'm Python!")
Hello, I'm Python!
# Input, assignment
>>> name = input('What is your name?\n')
What is your name?
Python
>>> print(f'Hi, {name}.')
Hi, Python.
```

Quick & Easy to Learn

Experienced programmers in any other language can pick up Python very quickly, and beginners find the clean syntax and indentation structure easy to learn. Whet your appetite with our Python 3 overview.

Python is a programming language that lets you work quickly
and integrate systems more effectively. >>> Learn More

Step 2: Go to downloads and select windows



Download for Windows

Python 3.11.3

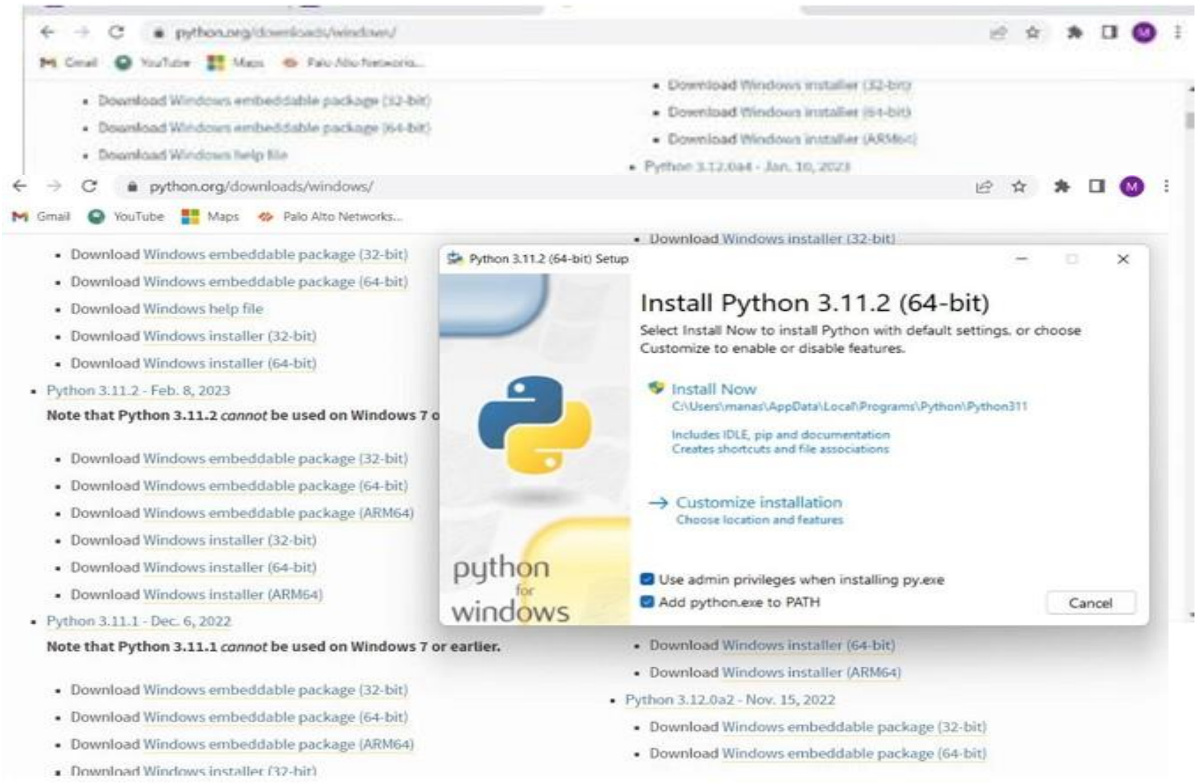
Note that Python 3.9+ *cannot* be used on Windows 7 or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

Python is a programming language that lets you work quickly

Step 3: Download Windows installer(64-bit)



- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (64-bit)
- Download Windows help file
- Download Windows installer (32-bit)
- Download Windows installer (64-bit)
- Python 3.11.2 - Feb. 8, 2023

Note that Python 3.11.2 *cannot* be used on Windows 7 o

- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (64-bit)
- Download Windows embeddable package (ARM64)
- Download Windows installer (32-bit)
- Download Windows installer (64-bit)
- Download Windows installer (ARM64)
- Python 3.11.1 - Dec. 6, 2022

Note that Python 3.11.1 *cannot* be used on Windows 7 or earlier.

- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (64-bit)
- Download Windows embeddable package (ARM64)
- Download Windows installer (32-bit)

Python 3.11.2 (64-bit) Setup

Install Python 3.11.2 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\manas\AppData\Local\Programs\Python\Python311

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Use admin privileges when installing py.exe
☑ Add python.exe to PATH

Cancel

- Download Windows installer (64-bit)
- Download Windows installer (ARM64)
- Python 3.12.0a2 - Nov. 15, 2022
- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (64-bit)

Step 4: Now select python.exe to path and install the IDLE

SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement. configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results. Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to conducted as two distinct phases.

CHAPTER -5
PYTHON CODE

VIEW CODE :

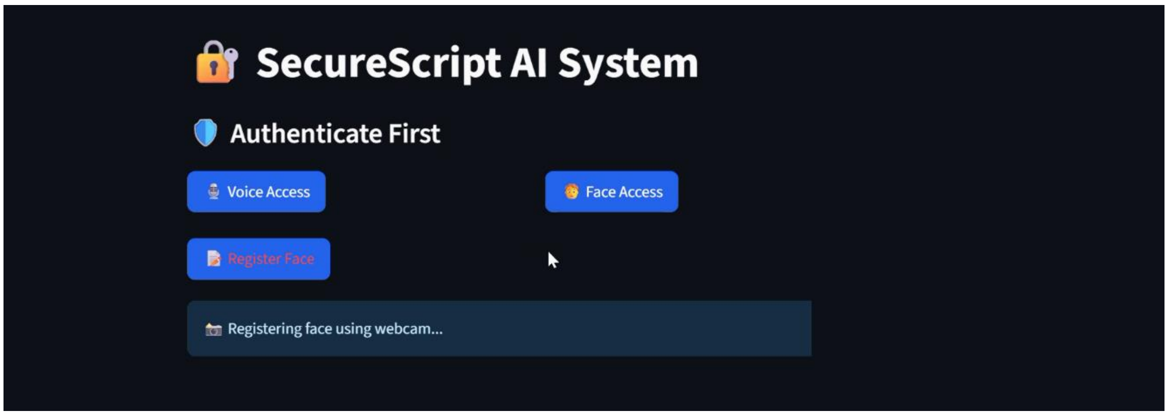
```
 import streamlit as st
from PIL import Image, UnidentifiedImageError
import io, os, random, smtplib, requests, time
from email.mime.text import MIMEText
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import speech_recognition as sr
import cv2
from deepface import DeepFace
from fpdf import FPDF
from diffusers import StableDiffusionPipeline
import torch
import numpy as np

for key in [
    "face_verified", "voice_verified", "access_granted", "otp_sent", "otp_verified",
    "generated_key", "pdf_data", "encrypt_email", "otp_code", "otp_sent_time"
]:
    if key not in st.session_state:
        if key in ["face_verified", "voice_verified", "access_granted", "otp_sent",
"otp_verified"]:
            st.session_state[key] = False
        else:
            st.session_state[key] = None

st.set_page_config(page_title="Secure Scripts AI System", layout="centered",
page_icon="🔐")
st.title("🔐 Secure Scripts AI System")

TOGETHER_API_KEY =
"16fe845937edfc5513d5a3adca83f7d53f207fe64aa32da3672e2fa224fa0579"
TOGETHER_API_URL = "https://api.together.xyz/v1/chat/completions"
HEADERS = {
    "Authorization": f"Bearer {TOGETHER_API_KEY}",
    "Content-Type": "application/json"
}

FACE_DB = "face_db"
os.makedirs(FACE_DB, exist_ok=True)

def generate_document(prompt: str, model: str = "mistralai/Mixtral-8x7B-Instruct-v0.1") -
> str:
    payload = {
        "model": model,
        "messages": [
            {"role": "system", "content": "You generate clean, professional documents from
prompts."},
            {"role": "user", "content": prompt}
        ],
        "max_tokens": 800,
        "temperature": 0.7
    }
    try:
        response = requests.post(TOGETHER_API_URL, headers=HEADERS,
json=payload)
        response.raise_for_status()
        return response.json()["choices"][0]["message"]["content"].strip()
    except Exception as e:
        return f"⚠ Error: {e}"

import textwrap
from fpdf import FPDF
import os

def text_to_pdf(text: str) -> bytes:
    pdf = FPDF()
    pdf.add_page()

    font_path = "DejaVuSans.ttf"
    if os.path.exists(font_path):
        pdf.add_font("DejaVu", "", font_path, uni=True)
        pdf.set_font("DejaVu", size=12)
    else:
        pdf.set_font("Arial", size=12)

    page_width = pdf.w - 2 * pdf.l_margin

    for line in text.split("\n"):
        safe_lines = textwrap.wrap(
            line,
            width=90,
            break_long_words=True,
            break_on_hyphens=False
        )
        for safe_line in safe_lines:
            pdf.multi_cell(page_width, 10, safe_line)

    # ✅ Return as pure bytes
    return bytes(pdf.output(dest="S"))


def capture_image():
    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    cap.release()
    if not ret:
        return None
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    return Image.fromarray(rgb)

def save_face_image(name, img):
    user_dir = os.path.join(FACE_DB, name)
    os.makedirs(user_dir, exist_ok=True)
    count = len(os.listdir(user_dir))
    img.save(os.path.join(user_dir, f"{count}.jpg"))

if not os.path.exists("face_db"):
    os.makedirs("face_db")

st.set_page_config(page_title="Secure Face Auth", layout="centered")
def capture_image():
    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    cap.release()
    if not ret:
        return None
    rgb_img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    return rgb_img

def save_face(name, image):
    path = os.path.join("face_db", f"{name}.jpg")
    cv2.imwrite(path, cv2.cvtColor(image, cv2.COLOR_RGB2BGR))

def verify_face(image):
    for filename in os.listdir("face_db"):
        if filename.endswith(".jpg"):
            path = os.path.join("face_db", filename)
            try:
                result = DeepFace.verify(
                    img1_path=image,
                    img2_path=path,
                    model_name='VGG-Face',
                    detector_backend='opencv',
                    enforce_detection=True
                )
                if result["verified"] and result["distance"] < 0.35:  # More strict matching
                    return True, os.path.splitext(filename)[0]
            except Exception as e:
                print(f"Verification failed with {filename}: {e}")
    return False, None

def verify_voice():
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        st.info("🎙 Say: 'your secret phrase'")
        try:
            audio = recognizer.listen(source, timeout=5)
            return "unlock secure system" in recognizer.recognize_google(audio).lower()
        except:
            return False

def generate_aes_key():
    return get_random_bytes(16)

def encrypt_pdf(data: bytes, key: bytes) -> bytes:
    cipher = AES.new(key, AES.MODE_CBC)
    return cipher.iv + cipher.encrypt(pad(data, AES.block_size))

def decrypt_pdf(encrypted_data: bytes, key: bytes) -> bytes:
    iv = encrypted_data[:16]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(encrypted_data[16:]), AES.block_size)

def embed_key_in_image(image_bytes, aes_key):
    try:
        img = Image.open(io.BytesIO(image_bytes)).convert("RGB")
    except UnidentifiedImageError:
        st.error("Not a valid image.")
        return None
    key_bin = ''.join(format(byte, '08b') for byte in aes_key) + '11111110'
    pixels, new_pixels, idx = list(img.getdata()), [], 0
    for pixel in pixels:
        r, g, b = pixel
        if idx < len(key_bin): r = (r & ~1) | int(key_bin[idx]); idx += 1
        if idx < len(key_bin): g = (g & ~1) | int(key_bin[idx]); idx += 1
        if idx < len(key_bin): b = (b & ~1) | int(key_bin[idx]); idx += 1
        new_pixels.append((r, g, b))
    img.putdata(new_pixels)
    out = io.BytesIO(); img.save(out, format="PNG")
    return out.getvalue()

def extract_key_from_image(img_bytes):
    try:
        img = Image.open(io.BytesIO(img_bytes)).convert("RGB")
    except UnidentifiedImageError:
        st.error("Invalid image.")
        return b""
    binary = ''.join(str(value & 1) for pixel in img.getdata() for value in pixel)
    byte_list = [binary[i:i+8] for i in range(0, len(binary), 8)]
    result = bytearray()
    for b in byte_list:
        if b == '11111110': break
        result.append(int(b, 2))
    return bytes(result)

def generate_otp():
    return str(random.randint(100000, 999999))

def send_otp_via_email(recipient, otp, decryptor_email=None):
    sender = "nagalakshmipala06@gmail.com"
    password = "pcccxlorrrzoyeif"
    body = f"🔐 Your OTP is: {otp}\n\n Decryption attempt by:\n{decryptor_email}" if
decryptor_email else f"🔐 Your OTP is: {otp}"
    message = MIMEText(body)
    message['Subject'] = "OTP for SecureScript Decryption"
    message['From'] = sender
    message['To'] = recipient
    with smtplib.SMTP_SSL("smtp.gmail.com", 465) as server:
        server.login(sender, password)
        server.send_message(message)

if not st.session_state["face_verified"] or not st.session_state["voice_verified"]:

   st.subheader(" Register Face")
name = st.text_input("Enter Name for Registration")

if st.button("Register Face"):
    if name.strip() == "":
        st.warning(" Please enter a valid name.")
    else:
        image = capture_image()
        if image is not None:
            save_face(name, image)
            st.image(image, width=150, caption="Registered Face")
            st.success(f" Face registered for {name}")
        else:
            st.error("Could not capture image. Try again.")

st.subheader("Verify Face")

if st.button("Verify Face"):
    image = capture_image()
    if image is not None:
        st.image(image, width=150, caption="Face to Verify")

        temp_path = "temp_verify.jpg"
        cv2.imwrite(temp_path, cv2.cvtColor(image, cv2.COLOR_RGB2BGR))
        matched, person = verify_face(temp_path)
        if matched:
            st.success(f"Face verified! Welcome, {person}")
            st.session_state["face_verified"] = True
        else:
            st.error("Face not recognized.")
        os.remove(temp_path)
    else:
        st.error(" Could not capture image. Try again.")

    st.subheader(" Voice Access")
if st.button("Verify Voice"):
    if verify_voice():
        st.session_state["voice_verified"] = True
        st.success(" Voice verified!")
    else:
        st.error(" Voice not recognized.")

if st.session_state["face_verified"] and st.session_state["voice_verified"]:
    st.session_state["access_granted"] = True

is_authenticated = st.session_state["access_granted"]

if is_authenticated:
    st.success("Access Granted.")
    menu = st.sidebar.selectbox("Choose Action", [
        "📄 Generate Document", "🗁 Upload PDF", "🔐 Encrypt PDF",
        "🕒 Stego Image", "🕵 Extract AES Key", "🗂 Decrypt PDF"])

    if menu == "📄 Generate Document":
        prompt = st.text_area("Enter prompt")
        if st.button("Generate") and prompt:
            content = generate_document(prompt)
            st.session_state.pdf_data = text_to_pdf(content)
            st.session_state.generated_key = generate_aes_key()
        if st.session_state.pdf_data:
            st.download_button(" Download PDF", st.session_state.pdf_data, "doc.pdf")
        if st.session_state.generated_key:
            st.download_button("🔑 Download Key", st.session_state.generated_key,
"aes.key")

    elif menu == "🗁 Upload PDF":
        file = st.file_uploader("Upload PDF")
        if file:
            key = generate_aes_key()
            st.download_button("🔑 Download Key", key, "aes.key")
```

```
elif menu == "🍪 Encrypt PDF":
    email = st.text_input("Sender Email")
    pdf = st.file_uploader("Upload PDF")
    key = st.file_uploader("Upload AES Key")
    if email: st.session_state.encrypt_email = email
    if pdf and key:
        encrypted = encrypt_pdf(pdf.read(), key.read())
        st.download_button("Download Encrypted PDF", encrypted, "enc.enc")

elif menu == "⏱ Stego Image":
    opt = st.radio("Choose", ["Upload", "Generate"])
    key_file = st.file_uploader("AES Key")
    if opt == "Upload":
        img = st.file_uploader("Upload Image")
        if img and key_file:
            result = embed_key_in_image(img.read(), key_file.read())
            if result:
                st.download_button("Download Stego Image", result, "stego.png")
    elif opt == "Generate":
        prompt = st.text_input("AI Image Prompt")
        if st.button("Generate AI Image") and prompt and key_file:
            pipe = StableDiffusionPipeline.from_pretrained(
                "runwayml/stable-diffusion-v1-5", torch_dtype=torch.float16
            ).to("cuda")
            image = pipe(prompt).images[0]
            img_bytes = io.BytesIO(); image.save(img_bytes, "PNG")
            stego = embed_key_in_image(img_bytes.getvalue(), key_file.read())
            st.image(image)
            if stego:
                st.download_button("Download Stego Image", stego, "ai_stego.png")

elif menu == "🕵 Extract AES Key":
    img = st.file_uploader("Stego Image")
```

```
        if img:
            key = extract_key_from_image(img.read())
            st.download_button("Extracted Key", key, "extracted.key")

    elif menu == "🗂 Decrypt PDF":
        decryptor_email = st.text_input("Your Email")
        current_time = time.time()
        if st.session_state.otp_sent_time is None or current_time -
st.session_state.otp_sent_time > 120:
            if st.button("Send OTP to Sender"):
                if not decryptor_email:
                    st.error("Enter email.")
                elif not st.session_state.encrypt_email:
                    st.error("No sender email found.")
                else:
                    otp = generate_otp()
                    st.session_state.otp_code = otp
                    st.session_state.otp_sent_time = current_time
                    send_otp_via_email(st.session_state.encrypt_email, otp, decryptor_email)
                    st.success(f"OTP sent to sender: {st.session_state.encrypt_email}")
        else:
            left = 120 - int(current_time - st.session_state.otp_sent_time)
            st.info(f"Wait {left} sec before resending OTP.")

        user_otp = st.text_input("Enter OTP", type="password")
        if st.button("Verify OTP") and user_otp == st.session_state.otp_code:
            st.session_state.otp_verified = True
            st.success("OTP Verified.")
        elif user_otp:
            st.error("Incorrect OTP.")

        if st.session_state.otp_verified:
            enc = st.file_uploader("Encrypted File", type=["enc"])
            key = st.file_uploader("AES Key", type=["key"])
            if enc and key:
                try:
                    decrypted = decrypt_pdf(enc.read(), key.read())
                    st.download_button("Download Decrypted PDF", decrypted,
"decrypted.pdf")
                except Exception as e:
                    st.error(f"Decryption failed: {e}")
else:
    st.info("🔐 Authenticate using Face and Voice to proceed.")
```

CHAPTER-6 FINAL RESULT

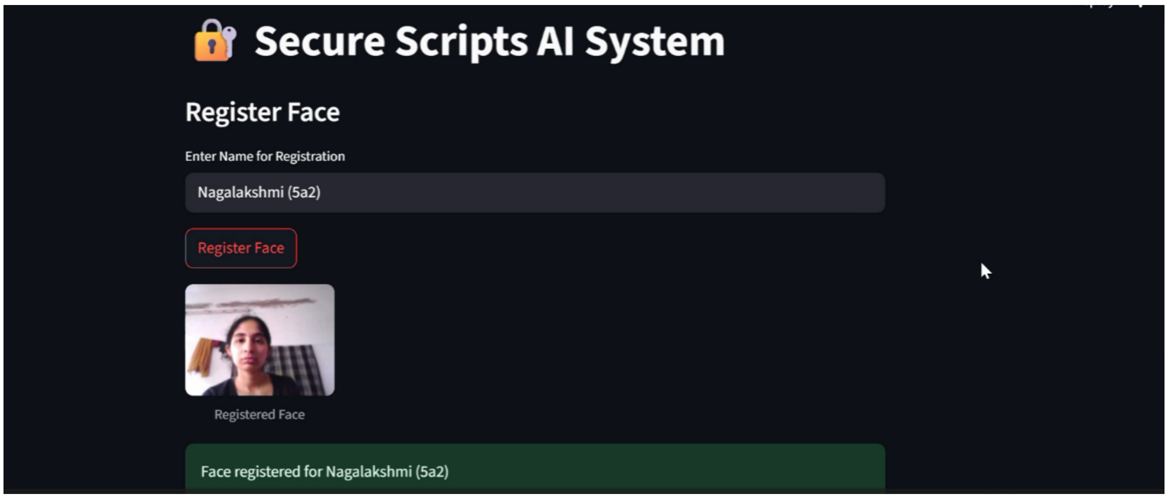
Taking Input



🔐 SecureScript AI System

🛡️ Authenticate First

📷 Registering face using webcam...



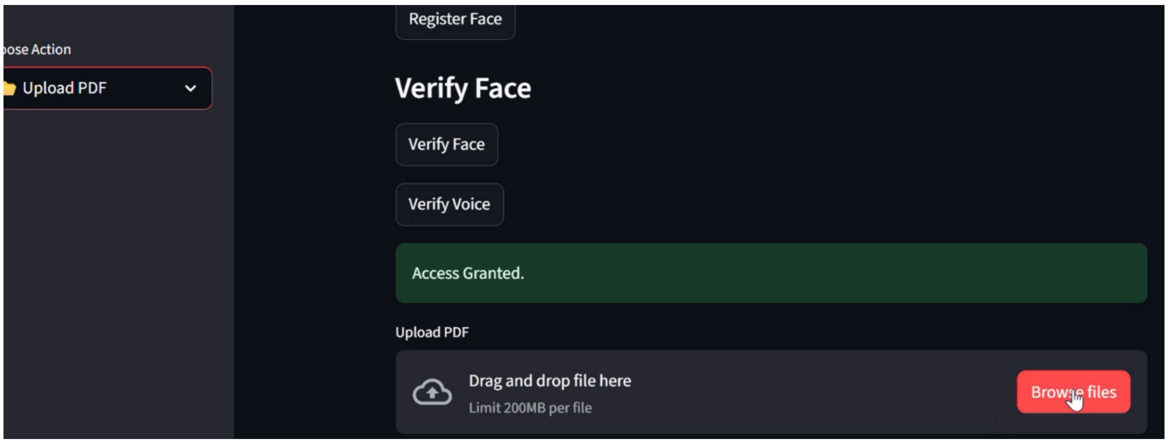
🔐 Secure Scripts AI System

Register Face

Enter Name for Registration

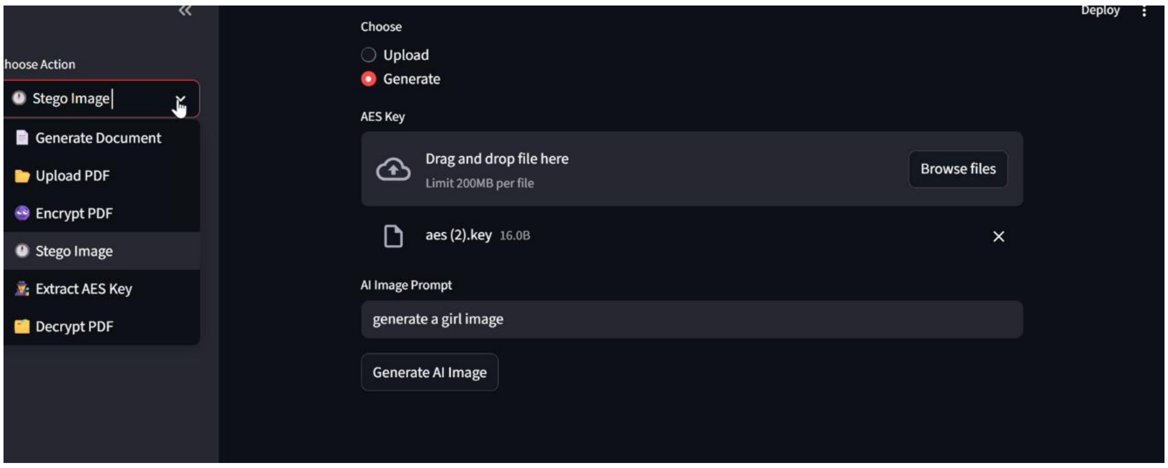
Nagalakshmi (5a2)

Face registered for Nagalakshmi (5a2)

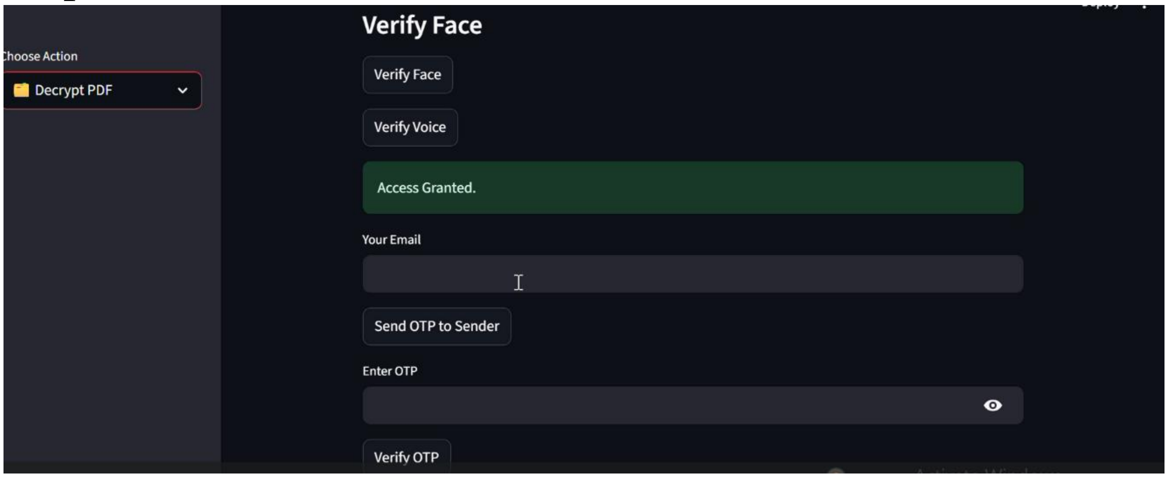


Verify Face

Access Granted.

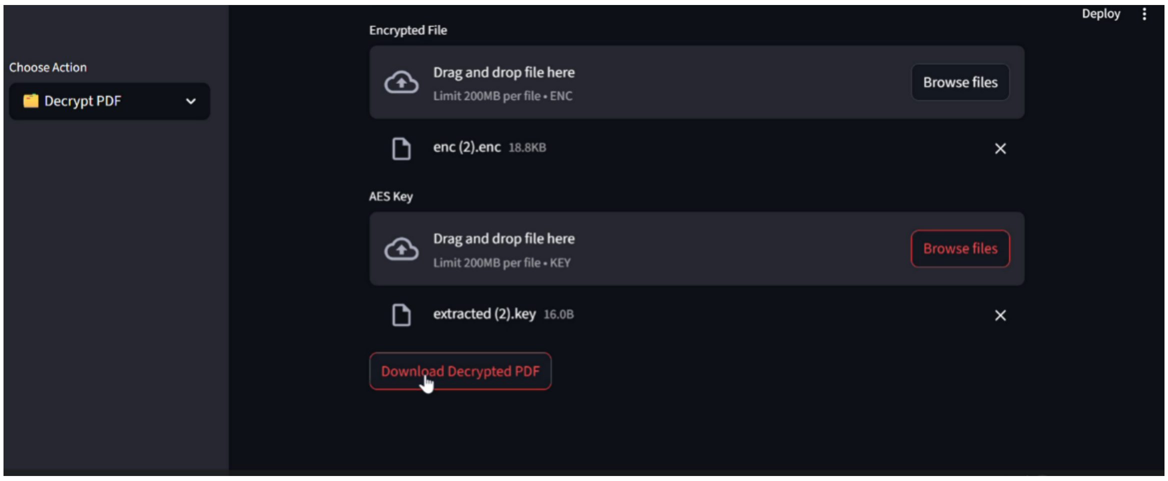


output



Verify Face

Access Granted.



.

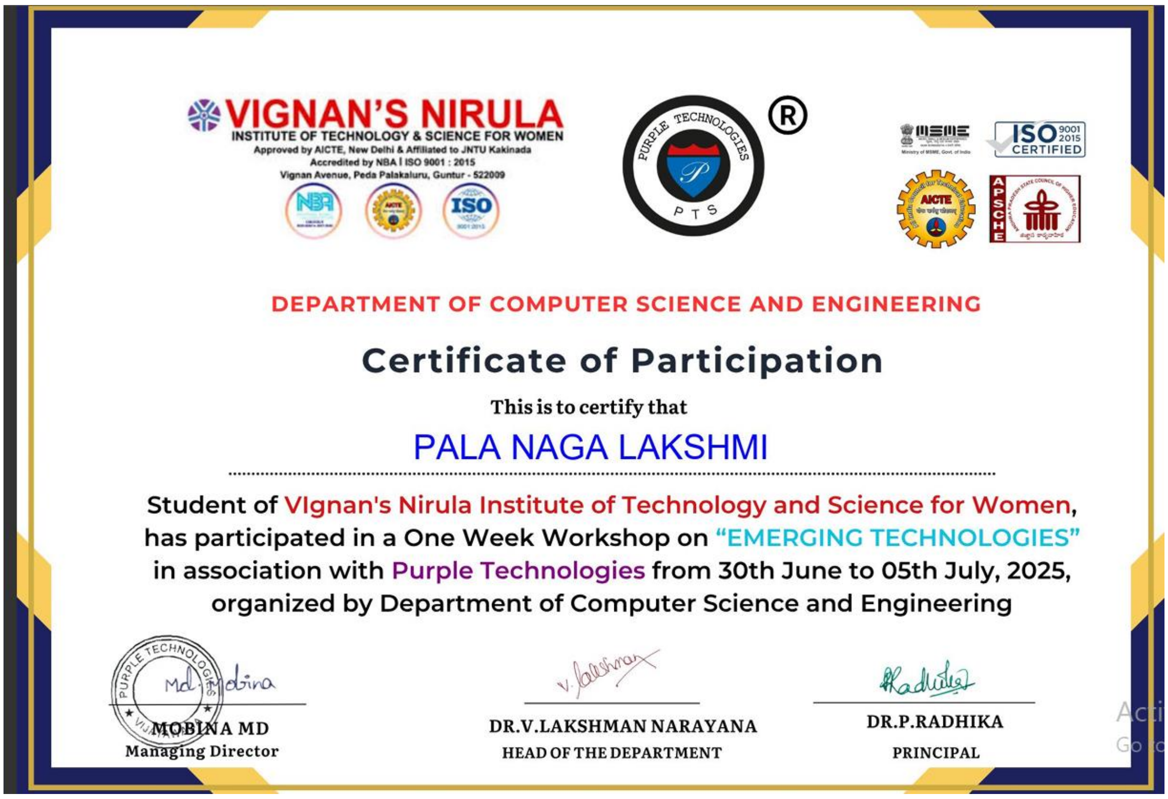
CHAPTER- 7 CONCLUSION

The Secure Scripts AI System successfully integrates Artificial Intelligence (AI), biometrics, and cryptographic security into a single unified platform for secure document handling. The system provides a practical and efficient solution to the growing challenges of digital data protection, ensuring that sensitive information remains confidential and accessible only to authorized users. By combining face recognition, voice authentication, and OTP verification, it establishes a robust multi-factor authentication process that enhances identity validation. Furthermore, the integration of AES encryption guarantees strong data confidentiality, while LSB steganography ensures that the encryption key is securely embedded within an image, adding another layer of security. The system's design through the Streamlit framework makes it highly interactive and user-friendly, allowing even non-technical users to perform secure operations such as document generation, encryption, and decryption effortlessly.

Overall, this project demonstrates how modern AI tools and cryptographic algorithms can be combined to provide next-generation cybersecurity solutions. The Secure Scripts

AI System proves that intelligent automation can significantly reduce the risk of unauthorized access and information leakage. The modular and scalable design of the application also allows future integration of technologies such as blockchain verification, cloud-based encryption, and quantum-safe algorithms. Through this project, it is evident that artificial intelligence not only enhances automation but also plays a critical role in improving data protection and trust in digital systems. Hence, the Secure Scripts AI System stands as a comprehensive, reliable, and innovative approach to secure, intelligent document management in the modern digital era.

Certificates



VIGNAN'S NIRULA
INSTITUTE OF TECHNOLOGY & SCIENCE FOR WOMEN
Approved by AICTE, New Delhi & Affiliated to JNTU Kakinada
Accredited by NBA | ISO 9001 : 2015
Vignan Avenue, Peda Palakaluru, Guntur - 522009

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate of Participation

This is to certify that

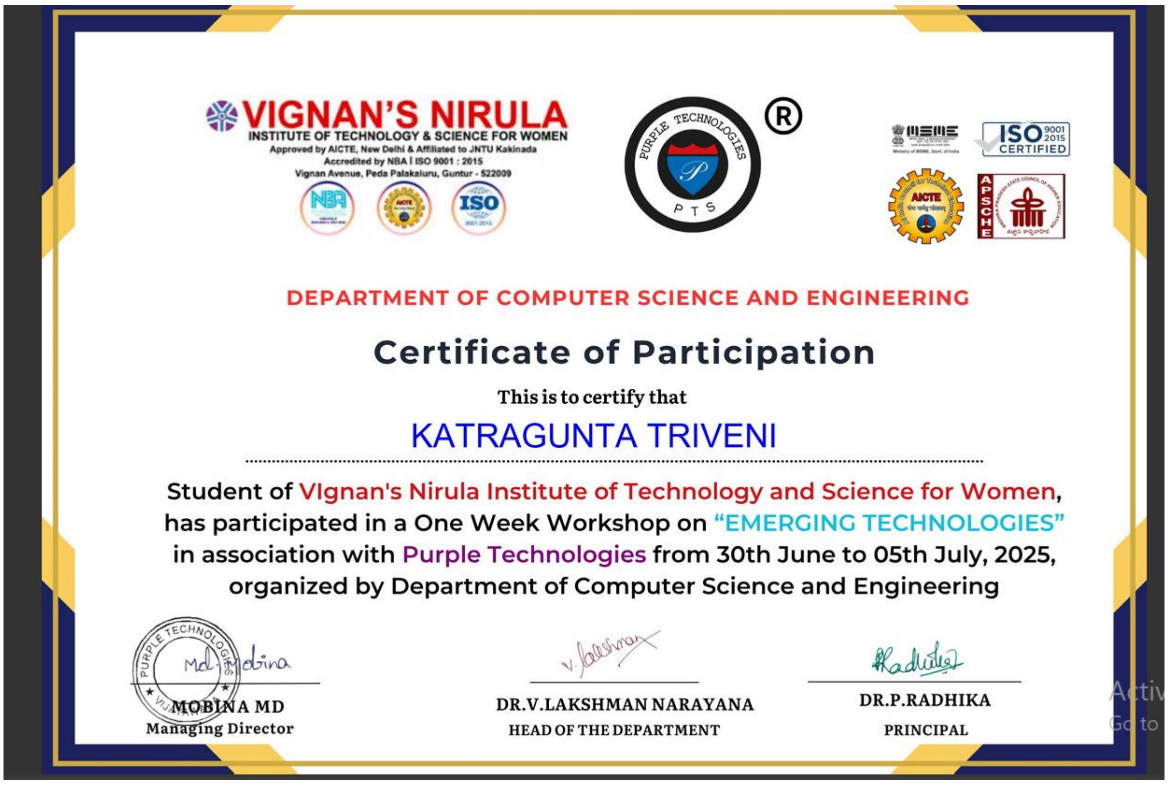
PALA NAGA LAKSHMI

Student of VIgnan's Nirula Institute of Technology and Science for Women, has participated in a One Week Workshop on "EMERGING TECHNOLOGIES" in association with Purple Technologies from 30th June to 05th July, 2025, organized by Department of Computer Science and Engineering

Md. Mobina
MOBINA MD
Managing Director

DR.V.LAKSHMAN NARAYANA
HEAD OF THE DEPARTMENT

DR.P.RADHIKA
PRINCIPAL



VIGNAN'S NIRULA
INSTITUTE OF TECHNOLOGY & SCIENCE FOR WOMEN
Approved by AICTE, New Delhi & Affiliated to JNTU Kakinada
Accredited by NBA | ISO 9001 : 2015
Vignan Avenue, Peda Palakaluru, Guntur - 522009

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate of Participation

This is to certify that

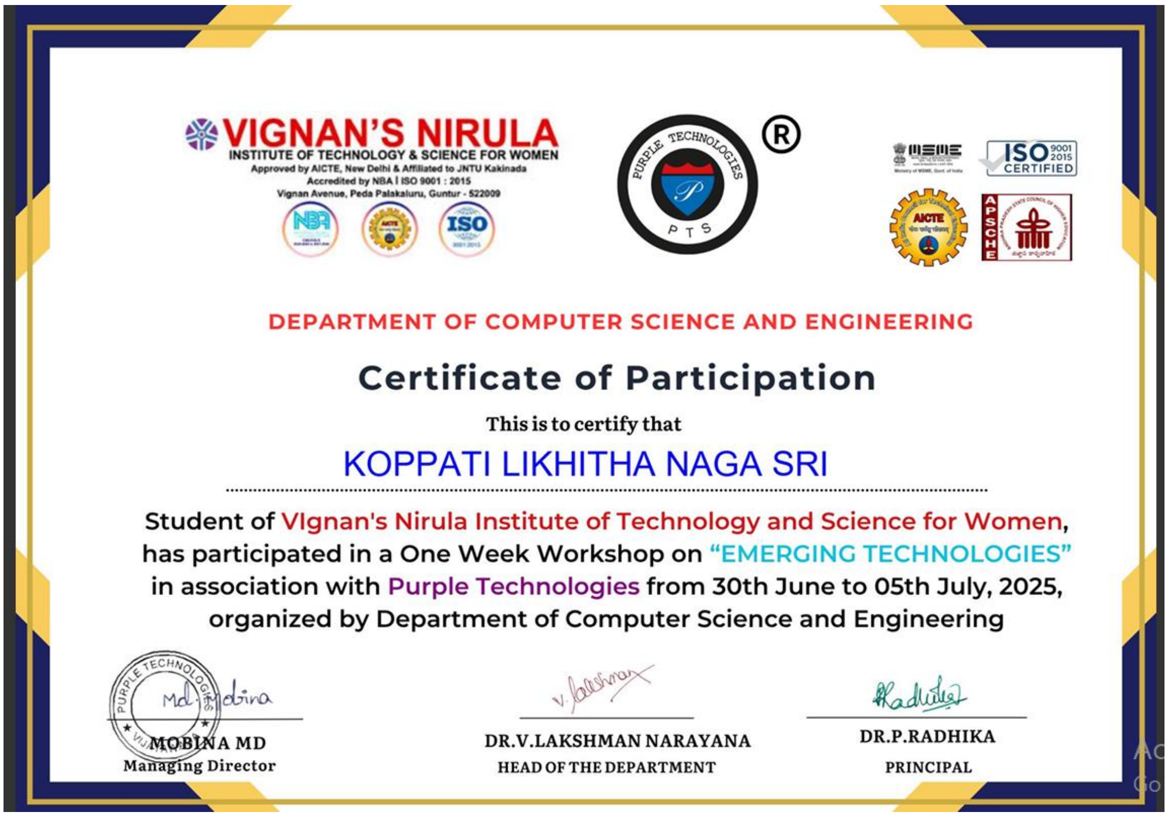
KATRAGUNTA TRIVENI

Student of VIgnan's Nirula Institute of Technology and Science for Women, has participated in a One Week Workshop on "EMERGING TECHNOLOGIES" in association with Purple Technologies from 30th June to 05th July, 2025, organized by Department of Computer Science and Engineering

Md. Mobina
MOBINA MD
Managing Director

DR.V.LAKSHMAN NARAYANA
HEAD OF THE DEPARTMENT

DR.P.RADHIKA
PRINCIPAL



VIGNAN'S NIRULA
INSTITUTE OF TECHNOLOGY & SCIENCE FOR WOMEN
Approved by AICTE, New Delhi & Affiliated to JNTU Kakinada
Accredited by NBA | ISO 9001 : 2015
Vignan Avenue, Peda Palakaluru, Guntur - 522009

PURPLE TECHNOLOGIES ®

ISO 9001 2015 CERTIFIED

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

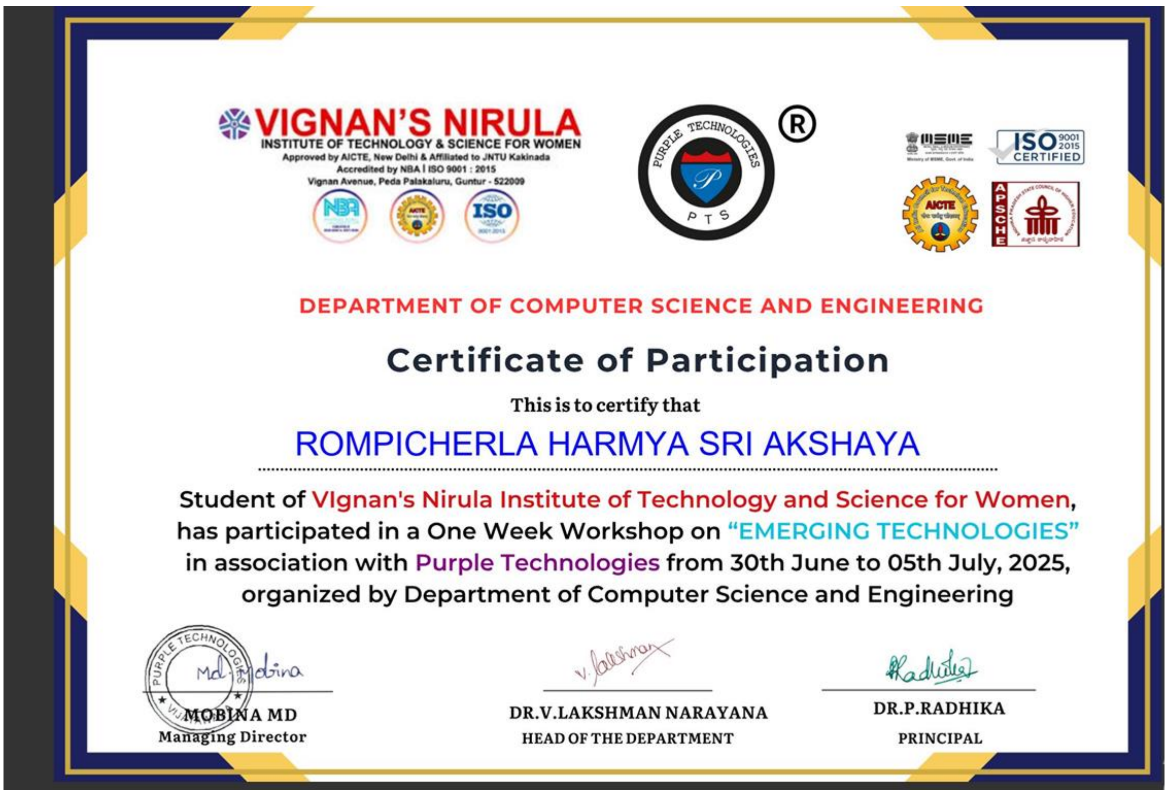
Certificate of Participation

This is to certify that

KOPPATI LIKHITHA NAGA SRI

Student of VIgnan's Nirula Institute of Technology and Science for Women, has participated in a One Week Workshop on "EMERGING TECHNOLOGIES" in association with Purple Technologies from 30th June to 05th July, 2025, organized by Department of Computer Science and Engineering

[illegible]	[illegible]	[illegible]
MOBINA MD	DR.V.LAKSHMAN NARAYANA	DR.P.RADHIKA
Managing Director	HEAD OF THE DEPARTMENT	PRINCIPAL



VIGNAN'S NIRULA
INSTITUTE OF TECHNOLOGY & SCIENCE FOR WOMEN
Approved by AICTE, New Delhi & Affiliated to JNTU Kakinada
Accredited by NBA | ISO 9001 : 2015
Vignan Avenue, Peda Palakaluru, Guntur - 522009

PURPLE TECHNOLOGIES ®

ISO 9001 2015 CERTIFIED

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate of Participation

This is to certify that

ROMPICHERLA HARMYA SRI AKSHAYA

Student of VIgnan's Nirula Institute of Technology and Science for Women, has participated in a One Week Workshop on "EMERGING TECHNOLOGIES" in association with Purple Technologies from 30th June to 05th July, 2025, organized by Department of Computer Science and Engineering

[illegible]	[illegible]	[illegible]
MOBINA MD	DR.V.LAKSHMAN NARAYANA	DR.P.RADHIKA
Managing Director	HEAD OF THE DEPARTMENT	PRINCIPAL

References

[1]Wan, Ming, et al. "Security script arrangement based on enhanced BERT for cooperative defense in networked control systems." Expert Systems with Applications (2025): 129753.

[2]Schmeelk, S., Roth, S., Rooney, J., Tariq, M., Wood, K., Kamen, J., & Dragos, D. (2022, September). Ambient intelligence security checks: Identifying integrity vulnerabilities in industry scripts. In Proceedings of SAI Intelligent Systems Conference (pp. 590-599). Cham: Springer International Publishing.

[3]McNaughton, M., Redford, J., Schaeffer, J., & Szafron, D. (2003, May). Patttern-Based AI Scripting Using ScriptEase. In Conference of the Canadian Society for Computational Studies of Intelligence (pp. 35-49). Berlin, Heidelberg: Springer Berlin Heidelberg.and current wants. This allowed them to subsequently get ready for the grounds drive with plenty of time to spare. Rating marks, the range of prowess, domination, illogical variables, and curiosity were all significant factors considered in the assessment of academic history.

[4]Chowdhury, M. N. U. R., Chowdhury, M. S. S., Neha, F. F., Haque, A., & Hossen, M. S. (2024, September). AI-Driven Secure Coding: Revolutionizing Source Code Defense. In 2024 International Conference on Signal Processing and Advance Research in Computing (SPARC) (Vol. 1, pp. 1-7). IEEE.

[5]Belozerov, V., Barclay, P. J., & Sami, A. (2025). Secure Coding with AI, From Creation to Inspection. arXiv preprint arXiv:2504.20814.

[6]Younas, F., Raza, A., Thalji, N., Abualigah, L., Zitar, R. A., & Jia, H. (2024). An efficient artificial intelligence approach for early detection of cross-site scripting attacks. Decision Analytics Journal, 11, 100466.

[7]Mateo Sanguino, T. D. J. (2024). Enhancing Security in Industrial Application Development: Case Study on Self-Generating Artificial Intelligence Tools. Applied Sciences, 14(9), 3780.

[8]Martin, S. Beyond Manual Scripts: The Role of Artificial Intelligence in Next-Generation Software Testing.

[9]He, Y., Wang, E., Rong, Y., Cheng, Z., & Chen, H. (2025, April). Security of ai agents. In 2025 IEEE/ACM International Workshop on Responsible AI Engineering (RAIE) (pp. 45-52). IEEE.

[10]Tihanyi, N., Bisztray, T., Ferrag, M. A., Jain, R., & Cordeiro, L. C. (2025). How secure is AI-generated code: A large-scale comparison of large language models. Empirical Software Engineering, 30(2), 47.

[11]Ugwa, S. (2024). From Scripts to Intelligence: How AI is Reshaping the Future of Software Testing.

[12]Elashmawi, W. H., Osman, H., Osama, M., & Nader, N. (2023, September). Development Generative AI for Cybersecurity: Evaluating Script Generation and Attack Classification in Penetration Testing. In International Workshop on Advanced Information Security Management and Applications (pp. 48-62). Cham: Springer Nature Switzerland.

[13]Kim, K., Shin, J., Park, J. G., & Kim, J. T. (2025). Performance evaluations of AI-based obfuscated and encrypted malicious script detection with feature optimization. ETRI Journal, 47(4), 753-770.

[14]Chennabasappa, S., Nikolaidis, C., Song, D., Molnar, D., Ding, S., Wan, S., ... & Saxe, J. (2025). Llamafirewall: An open source guardrail system for building secure ai agents. arXiv preprint arXiv:2505.03574.

[15]Perry, N., Srivastava, M., Kumar, D., & Boneh, D. (2023, November). Do users write more insecure code with ai assistants?. In Proceedings of the 2023 ACM SIGSAC conference on computer and communications security (pp. 2785-2799).

[16]Waters, T. E., & Roisman, G. I. (2019). The secure base script concept: An overview. Current Opinion in Psychology, 25, 162-166.

[17] Kaur, J., Garg, U., & Bathla, G. (2023). Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review. Artificial Intelligence Review, 56(11), 12725-12769.

[18]Sugier, J. (2023). Scripting Scenarios of Pedestrian Behavior in a Computer Simulator of Security Monitoring System: A Practitioner's Perspective. Transport and Telecommunication, 24(4), 349-360.

[19]Sikarwar, R., Shakya, H. K., Kumar, A., & Rawat, A. (2024). Advanced security solutions for conversational AI. Conversational Artificial Intelligence, 287-301.

[20]Kaneko, T., Yoshioka, N., & Sasaki, R. (2020, December). STAMP S&S: Safety & Security Scenario for Specification and Standard in the society of AI/IoT. In 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C) (pp. 168-175). IEEE.