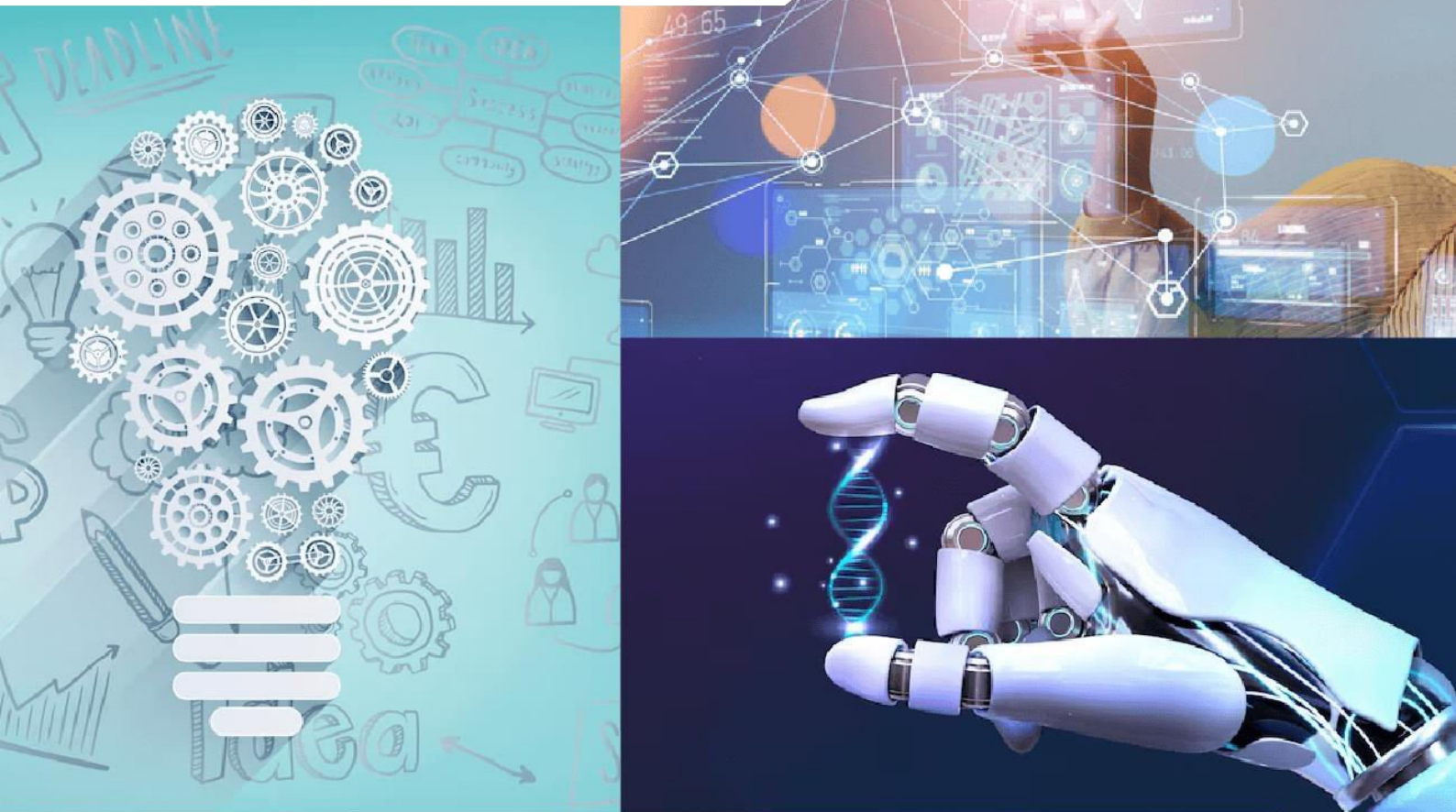




International Journal of Advanced Research in Education and Technology (IJARETY)

Volume 13, Issue 2, March - April 2026

Impact Factor: 8.152



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



Steganography of Multimedia (Image, Audio, and Video) Using the LSB Technique

Deekshitkumar S , Gnaneswari Gnanaguru

Master of Computer Applications, CMR Institute of Technology, Bengaluru, India

Associate Professor, Department of Computer Applications, CMR Institute of Technology, Bengaluru, India

ABSTRACT: In today's digital world, protecting sensitive information during online communication has become essential across education, corporate environments, and social platforms. While encryption makes data unreadable, it often draws attention as suspicious ciphered content. Steganography offers a smarter approach by hiding the very existence of secret messages within everyday multimedia files like images, audio, and video. This paper presents a unified system that embeds AES-encrypted text into digital media using the Least Significant Bit (LSB) technique. The initial embedding position is determined by a password-based MD5 hash, making it incredibly hard to detect. The system features a full graphical user interface built in Python using Pillow, NumPy, librosa, OpenCV, and Tkinter. Real-time capacity checks and progress tracking enable users to encode and decode across all three media types with ease. The system achieves PSNR > 39 dB for images, SNR > 48 dB for audio, and supports up to 150 MB per minute with HD video.

KEYWORDS: Steganography, LSB Technique, AES Encryption, Multimedia Security, Data Hiding, Image Steganography, Audio Steganography, Video Steganography, Python, Covert Communication

I. INTRODUCTION

In today's technology-driven era, most communication, transactions, and file exchanges take place through digital networks. With the growing dependence on online connections, protecting shared data has become critical. Steganography hides the very existence of a secret message within ordinary digital files such as images, audio, and video — derived from the Greek words meaning 'covered writing'.

Unlike encryption, where secrets are scrambled but still visible, steganography conceals the presence of communication entirely. Data slips through public networks without drawing attention, shielded by imperceptibility rather than complexity. Hidden messages are embedded using tiny changes that cannot be seen or heard by any observer.

This paper presents a unified multimedia steganography system built with Python, combining Pillow for image processing, librosa for audio manipulation, OpenCV for video handling, and Tkinter for the user interface. By weaving AES encryption into the LSB hiding technique, the method proves strong, adaptable, and safe for modern covert communication.

II. LITERATURE SURVEY

Steganography and data hiding have been studied extensively in both academic and industrial research. Early work mostly dealt with image steganography using simple LSB substitution, where the lowest bits of pixel values are modified to store hidden data. These techniques were vulnerable to compression, image processing, and statistical attacks.

Subsequent research introduced frequency-domain methods such as DCT and DWT, where data is embedded in transformed coefficients rather than raw pixels. Audio steganography followed, hiding data in timing shifts and wave phases. Video steganography scatters messages across frames to exploit the large capacity of moving media.

A common modern approach combines AES encryption with LSB hiding so that even if the hidden data is extracted, it remains unreadable without the correct key. Password-based MD5 hashing derives pseudo-random start positions, making patterns harder to detect. The proposed system addresses the gaps of existing tools by combining multi-media support, strong encryption, and an intuitive GUI.

III. PROPOSED METHODOLOGY

The system follows a modular standalone desktop architecture comprising a Tkinter GUI controller, media-specific processing modules, an AES handler, and file I/O operations.

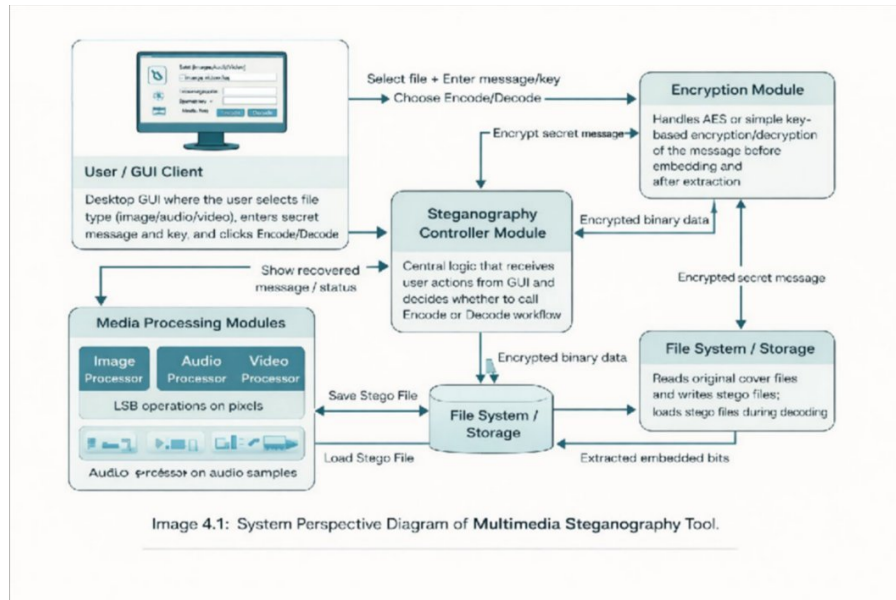


Fig 1: System Perspective Diagram of Multimedia Steganography Tool

3.1 System Architecture

The system operates as a standalone desktop application. Users access all core functions through a single intuitive window with three primary groups: Encode, Decode, and utility features (Help, Exit, About). During encoding, AES-256 encryption is applied first, the message is converted to binary, and each bit is embedded into the LSB of successive pixel channels, audio sample bytes, or video frame channels. A delimiter string '####' marks the end of the payload.

Fig 2: System Architecture of Steganography Tool

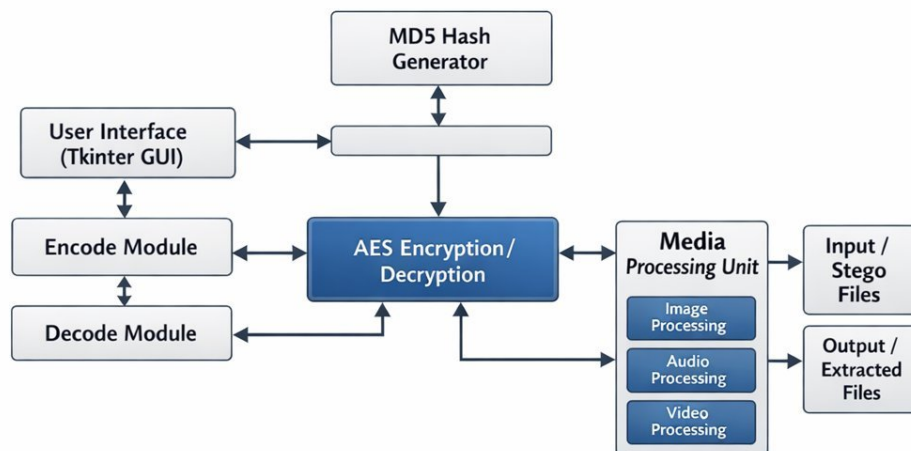


Fig 2: System Architecture of Steganography Tool

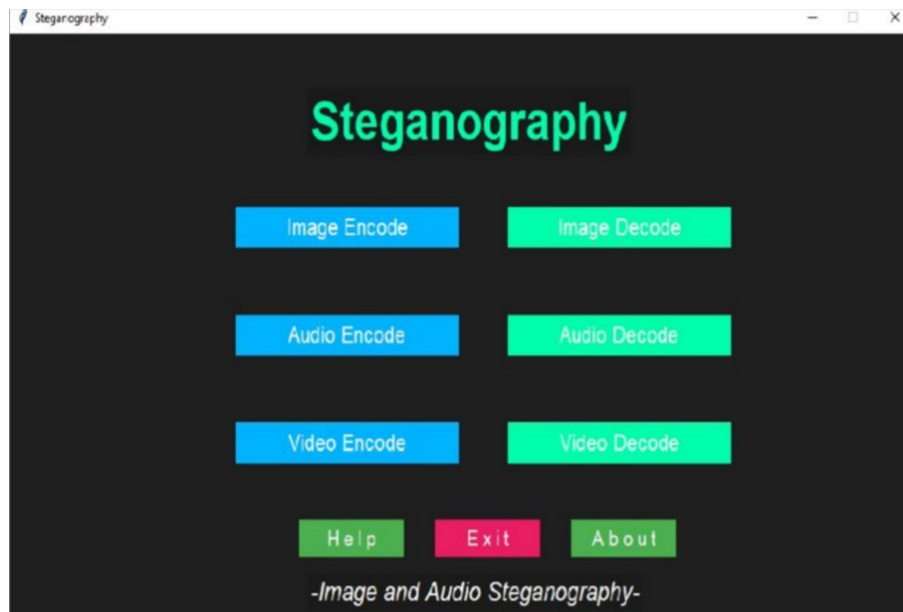


Fig 3: Main Application Window – Steganography Tool

3.2 Data Flow and Validation

All inputs are validated before embedding. Image files must be PNG or JPG; audio must be WAV; video must be MP4 or AVI. Message size is checked against the file's maximum capacity. Unsupported types are rejected with descriptive error messages. Temporary files are auto-deleted and the application runs without network access.

Fig 3: Data Flow Diagram of Steganography Process

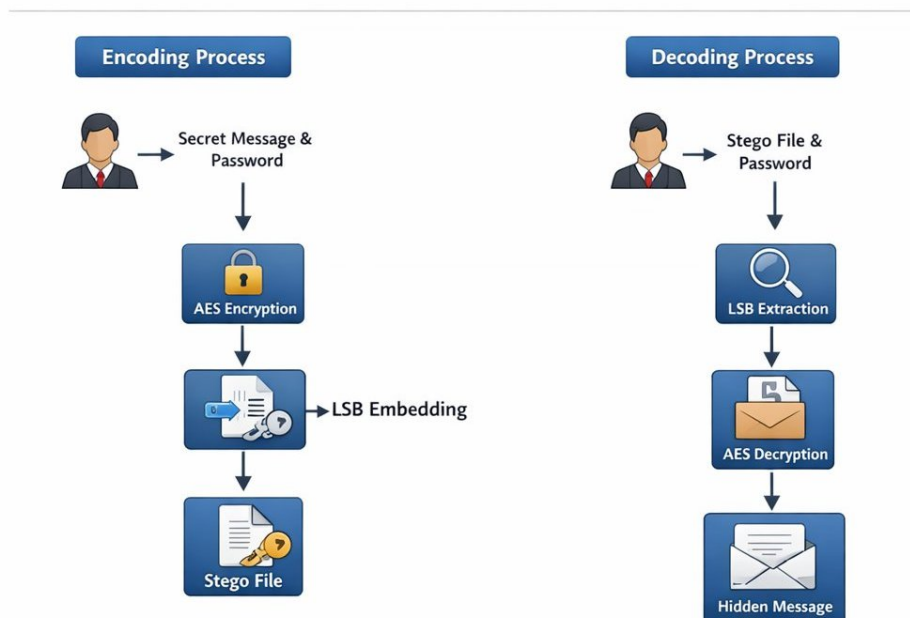


Fig 4: Data Flow Diagram of Steganography Process

3.3 Encoding Workflow

The user selects the media type, loads a cover file, enters the secret text and password. The system verifies capacity, AES-256 encrypts the message, computes the MD5-based start index, and embeds encrypted bits into LSBs from that index. The stego file is saved and a success dialog confirms completion.

3.4 Decoding Workflow

The user loads the stego file and enters the password. The MD5 hash recomputes the start index. LSBs are extracted and assembled into a bit stream. The system detects the '###' delimiter, then AES decryption recovers the original message which is displayed to the user.

3.5 Technology Stack

Python 3.8+ is the core language. Pillow handles image pixel read/write. NumPy enables LSB array manipulation. OpenCV manages video frame extraction and reconstruction. wave and librosa provide WAV audio sample access. PyCryptodome implements AES-256 encryption. hashlib generates MD5-based start indices. Tkinter provides the cross-platform GUI.

IV. RESULTS AND DISCUSSION

The system was evaluated on Windows 11 and Ubuntu 22.04 through functional, security, and performance testing. All ten functional test cases passed successfully across all three media types.

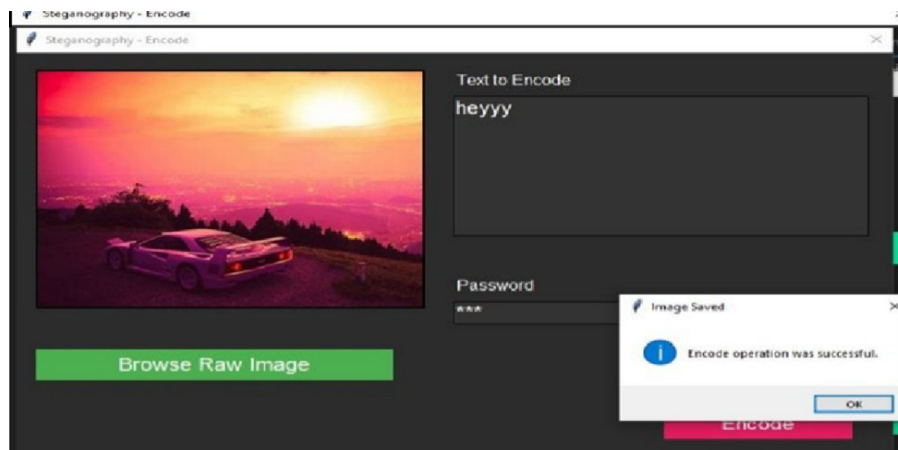


Fig 5: Image Steganography – Encode with Success Confirmation

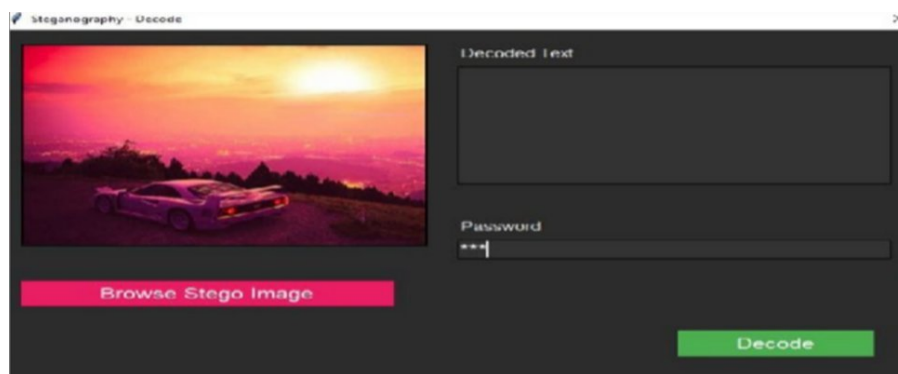


Fig 6: Image Steganography – Decode Interface

Performance: a 5 MB JPG encoding 1000 characters took ~2 seconds; a 10 MB WAV hiding 2000 characters took ~8 seconds; a 50 MB MP4 processing 5000 characters took ~25 seconds. CPU stayed below 20 percent with no memory leaks.

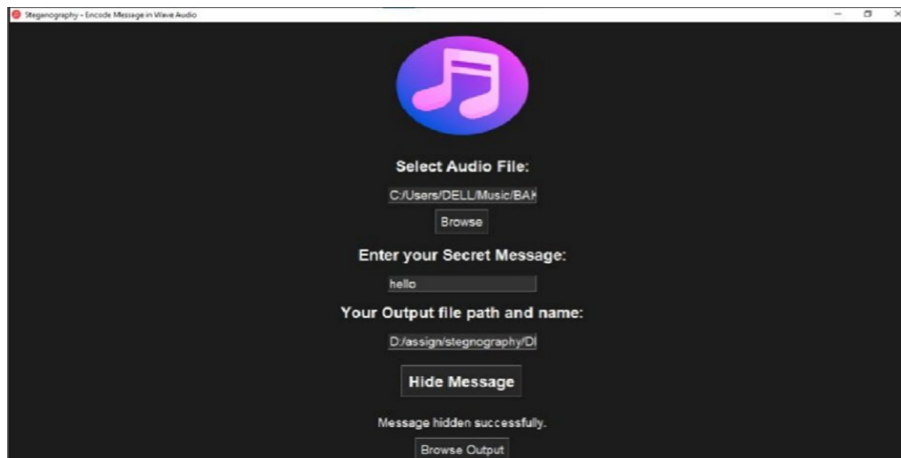


Fig 7: Audio Steganography – Encode Interface

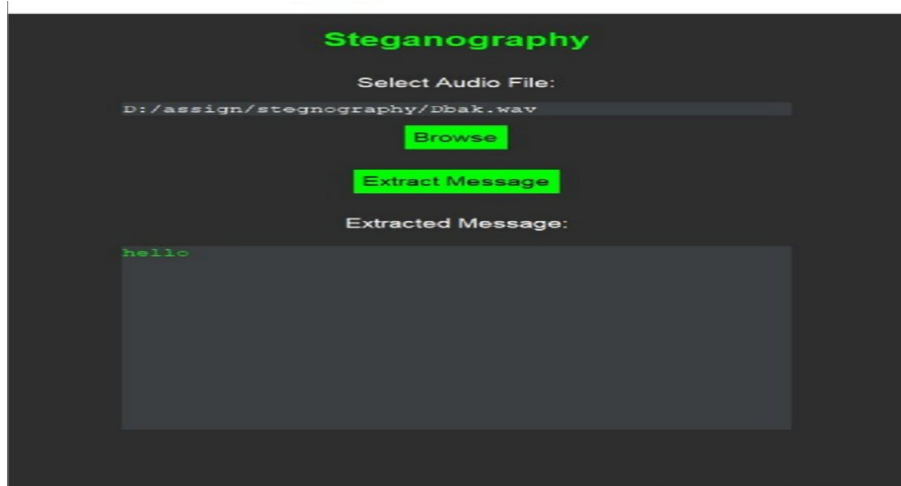


Fig 8: Audio Steganography – Decode with Extracted Message

Image quality was maintained with PSNR > 39 dB and audio with SNR > 48 dB. All stego files passed visual and auditory inspection with no perceptible artefacts. Wrong passwords consistently produced unreadable output. No cleartext passwords are stored at any point.

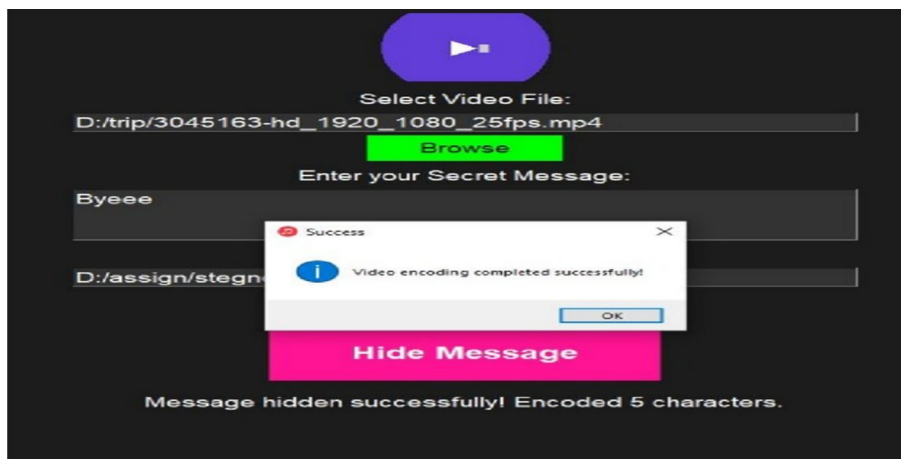


Fig 9: Video Steganography – Encode with Success Dialog



Fig 10: Video Steganography – Decode with Extracted Message

V. CONCLUSION

This paper presented a unified Python-based multimedia steganography system hiding AES-encrypted text inside images, audio, and video using the LSB technique. The system combines AES-256 security, imperceptibility (PSNR > 39 dB, SNR > 48 dB), high capacity (150 MB/min for HD video), and an accessible Tkinter GUI. All ten test cases passed and cross-platform evaluation confirmed reliability.

Future work includes real-time live video streaming, mobile application via React Native or Flutter, multi-layer LSB encoding for higher capacity, and a cloud-based batch dashboard. These enhancements would transform the prototype into a production-ready covert communication platform.

REFERENCES

- [1] R. S. Pressman, Software Engineering, 8th ed. McGraw-Hill, 2014.
- [2] I. Sommerville, Software Engineering, 10th ed. Pearson, 2015.
- [3] A. Dennis et al., Systems Analysis and Design, 6th ed. Wiley, 2018.
- [4] A. N. Vaidyam et al., “Chatbots in mental health,” Canadian Journal of Psychiatry, vol. 64, no. 7, pp. 456–464, 2019.
- [5] Woebot Health, “The role of AI in mental health,” Research Report, 2021.
- [6] Pillow Documentation, 2024. Available: <https://pillow.readthedocs.io>
- [7] OpenCV Documentation, 2024. Available: <https://opencv.org>
- [8] NumPy Documentation, 2024. Available: <https://numpy.org>
- [9] GeeksforGeeks, “Steganography Algorithms Explained,” 2024. Available: <https://www.geeksforgeeks.org>
- [10] R. Anderson, Security Engineering, 3rd ed. Wiley, 2020.

International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 8.152