Project Title- Secure Data Hiding in Images Using Steganography

Presented By- Bhavesh Rai

College Name & Department-Babasaheb Bhimrao Ambedkar University Lucknow And Department Of Information Technology

Project Outline

1. **Problem Statement-**

   Defining the challenges this project aims to solve.

2. **Technology Used-**

   Highlighting the key libraries and platforms employed.

3. **Wow Factor-**

   Showcasing the unique and innovative features.

4. **End Users-**

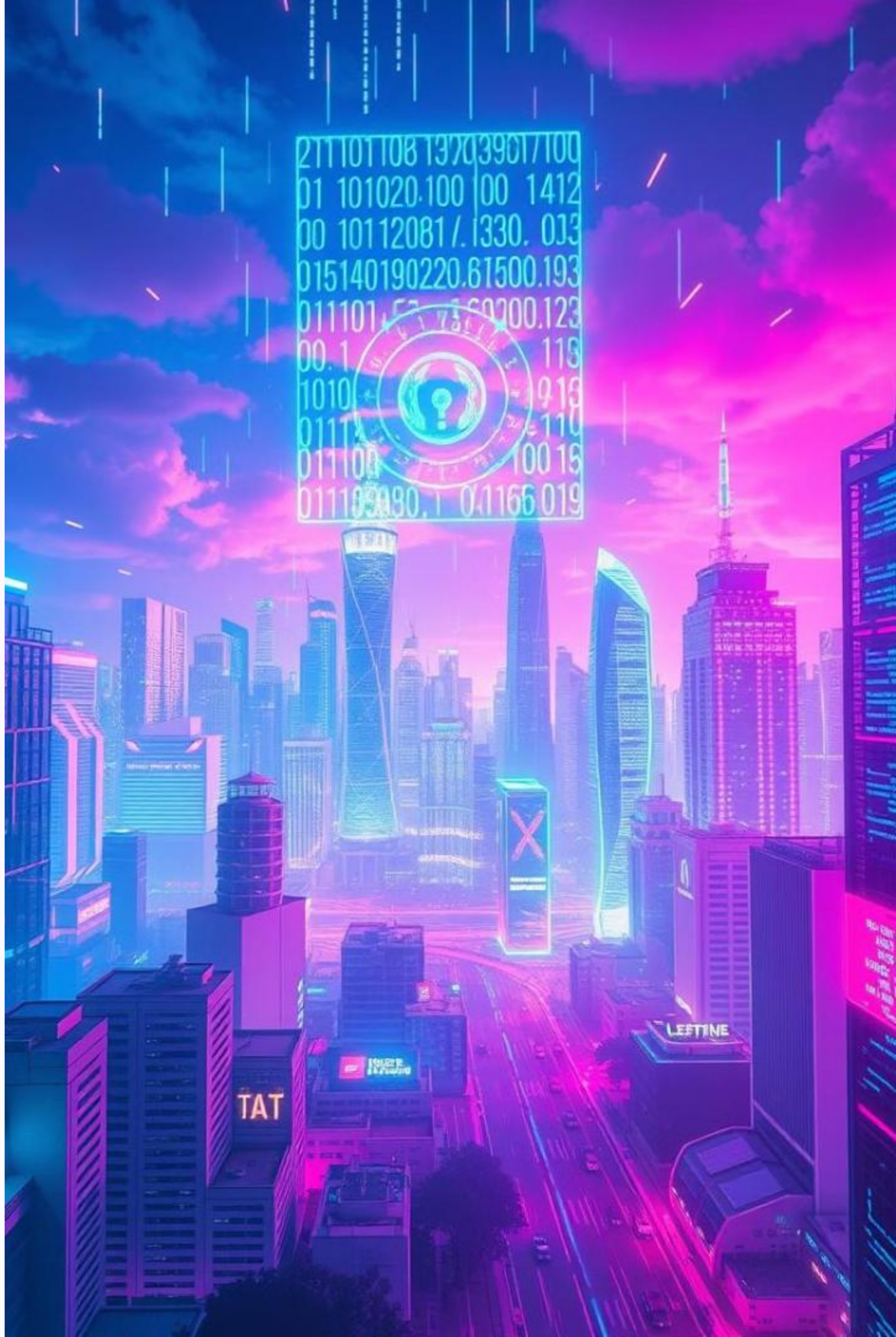
   Identifying the target demographic.

5. **Result- Results show how well the hidden data is embedded without visibly altering the image.**

6. **Conclusion-This project demonstrates that data can be securely hidden within images, with a focus on both maintaining image quality and ensuring data security.**

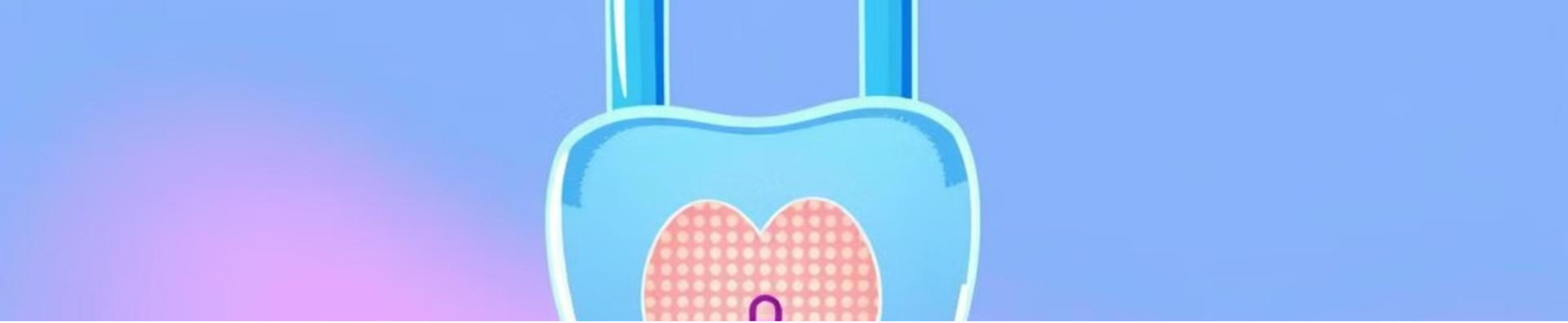
7. **Git-Hub Link-**
   Project Link.

8. **Future Scope-Advanced Algorithms, More Data Types,**
   Cloud Integration.



Secure Data Hiding in Images Using Steganography

Steganography is the art of hiding information within data. This project explores secure data hiding in images. It uses steganography to protect sensitive information.



Problem Statement

Protecting sensitive data from unauthorized access is crucial. Traditional encryption methods can attract attention. Steganography offers a way to conceal data. This project implements a secure steganographic system.

Protection from Unauthorized Access

Concealing Data Effectively

Secure Implementation

Technology Used

Python

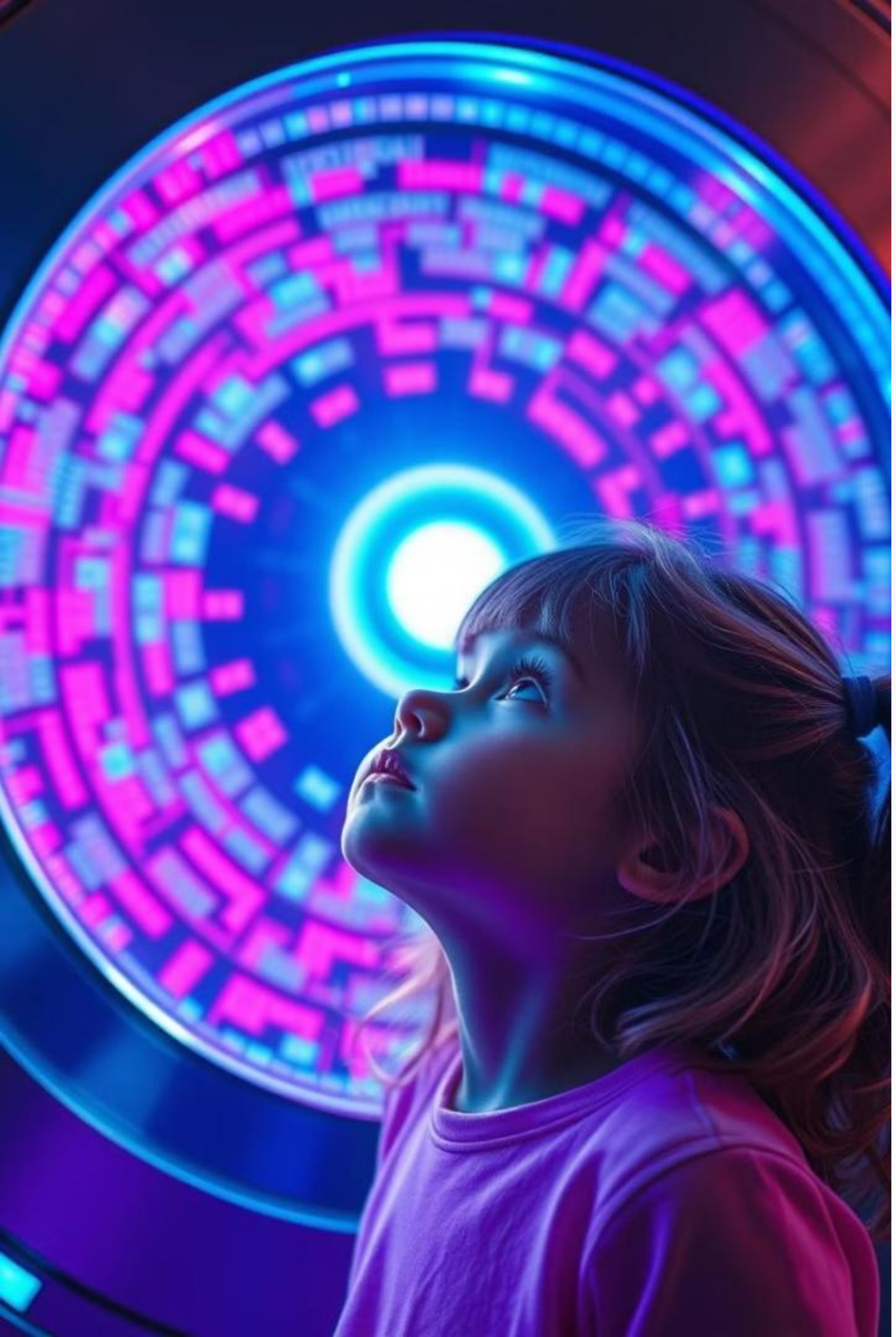
Primary programming language.

OpenCV (cv2)

For loading, manipulating, and saving images.

Steganography

The technique of hiding a message within an image.



Wow Factors

High Capacity

Can hide large amounts of data within images.

Secure Encryption

Utilizes strong encryption algorithms to protect data.

User-Friendly

Easy-to-use interface for encoding and decoding.

End Users

Journalists

For secure communication and data transfer.

Security Professionals

To hide sensitive information in penetration testing.

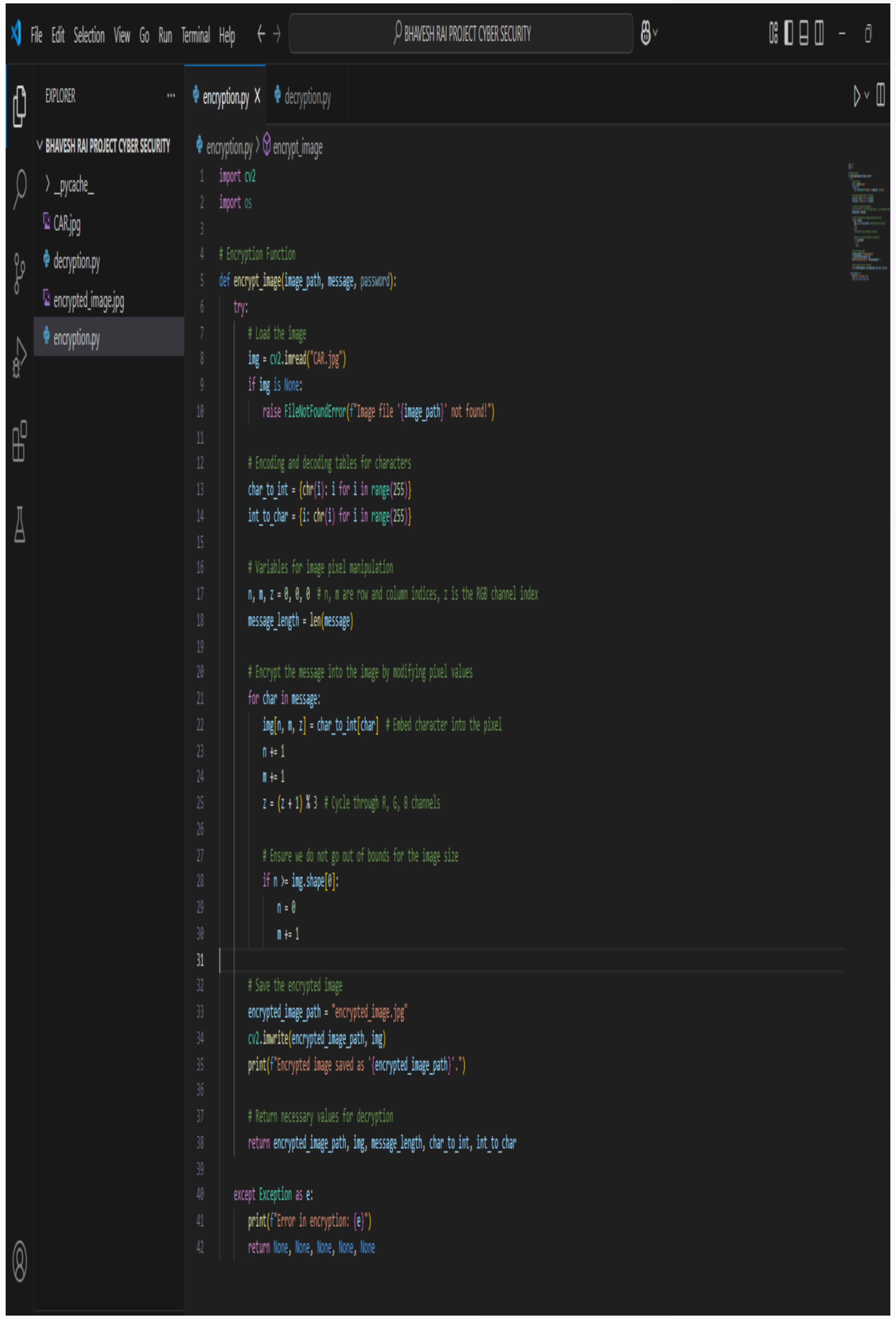
Privacy Advocates

To protect personal data from surveillance.

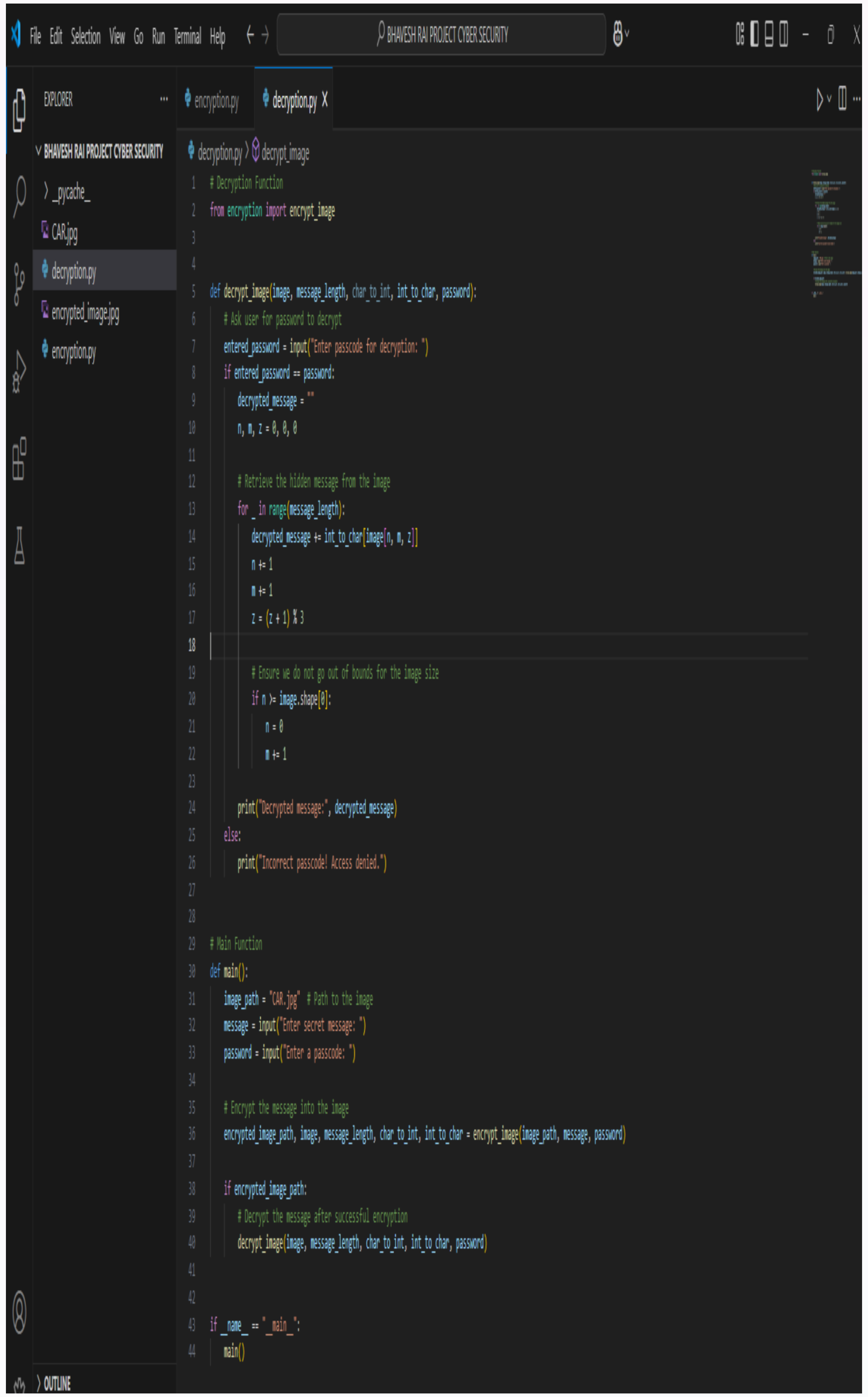
IT Professionals

Secure sensitive documents

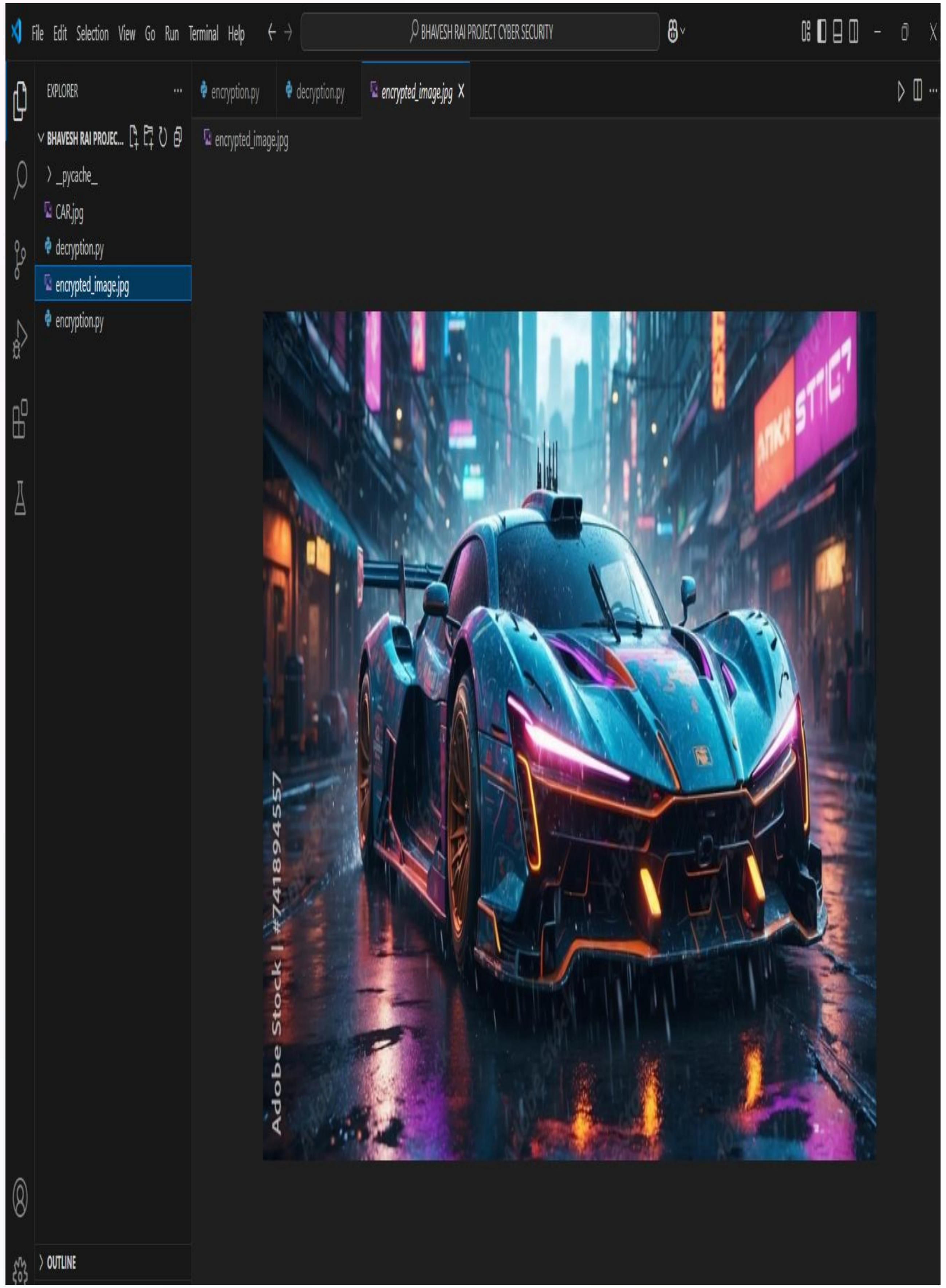
Results



```python
import cv2
import os

# Encryption Function
def encrypt_image(image_path, message, password):
    try:
        # Load the image
        img = cv2.imread("CAR.jpg")
        if img is None:
            raise FileNotFoundError(f"Image file '{image_path}' not found!")

        # Encoding and decoding tables for characters
        char_to_int = {chr(i): i for i in range(255)}
        int_to_char = {i: chr(i) for i in range(255)}

        # Variables for image pixel manipulation
        n, m, z = 0, 0, 0  # n, m are row and column indices, z is the RGB channel index
        message_length = len(message)

        # Encrypt the message into the image by modifying pixel values
        for char in message:
            img[n, m, z] = char_to_int[char]  # Embed character into the pixel
            n += 1
            m += 1
            z = (z + 1) % 3  # Cycle through R, G, B channels

            # Ensure we do not go out of bounds for the image size
            if n >= img.shape[0]:
                n = 0
                m += 1

        # Save the encrypted image
        encrypted_image_path = "encrypted_image.jpg"
        cv2.imwrite(encrypted_image_path, img)
        print(f"Encrypted image saved as '{encrypted_image_path}'.")

        # Return necessary values for decryption
        return encrypted_image_path, img, message_length, char_to_int, int_to_char

    except Exception as e:
        print(f"Error in encryption: {e}")
        return None, None, None, None, None
```



```python
# Decryption Function
from encryption import encrypt_image


def decrypt_image(image, message_length, char_to_int, int_to_char, password):
    # Ask user for password to decrypt
    entered_password = input("Enter passcode for decryption: ")
    if entered_password == password:
        decrypted_message = ""
        n, m, z = 0, 0, 0

        # Retrieve the hidden message from the image
        for _ in range(message_length):
            decrypted_message += int_to_char[image[n, m, z]]
            n += 1
            m += 1
            z = (z + 1) % 3

            # Ensure we do not go out of bounds for the image size
            if n >= image.shape[0]:
                n = 0
                m += 1

        print("Decrypted message:", decrypted_message)
    else:
        print("Incorrect passcode! Access denied.")


# Main Function
def main():
    image_path = "CAR.jpg"  # Path to the image
    message = input("Enter secret message: ")
    password = input("Enter a passcode: ")

    # Encrypt the message into the image
    encrypted_image_path, image, message_length, char_to_int, int_to_char = encrypt_image(image_path, message, password)

    if encrypted_image_path:
        # Decrypt the message after successful encryption
        decrypt_image(image, message_length, char_to_int, int_to_char, password)


if __name__ == "__main__":
    main()
```



Conclusion

The implemented system offers a secure and reliable way to protect sensitive data. The project addresses the problem statement effectively.

- Data hiding achieved.
- Secure communication.
- Reliable system.

Git-Hub Link

- https://github.com/Bhavesh140299/CyberSecurity_project_Aicte.git text

Future Scope

1. **Advanced Algorithms**

   Implement sophisticated steganographic techniques.

2. **More Data Types**

   Extend to audio, video, and other file formats.

3. **Cloud Integration**

   Integrate with cloud storage for secure data storage.

Thank You

Thank you for your time and attention.