Develop a webpage that implements steganography, allowing users to hide a secret message inside an image and later extract it. The hidden message could be text.

Introduction

Steganography is the practice of hiding information within digital media, such as images, audio, or video, so that it remains undetectable to unauthorized users. Unlike cryptography, which scrambles data to make it unreadable, steganography hides data in plain sight. This project involves building a **web-based steganography tool** that allows users to embed and retrieve hidden messages from images securely.

Explanation of the Project

Features of the Web App

1. **Hiding a Message:**
    - The user selects an image file.
    - Enters a secret message and a secret key.
    - The message is encrypted and then hidden inside the image pixels.
    - The new image with the hidden message is downloaded.
2. **Extracting the Message:**
    - The user uploads the modified image and enters the correct secret key.
    - The program extracts, decrypts, and displays the hidden message.

Technologies Used

- **HTML, CSS, and JavaScript** for frontend and logic.
- **Canvas API** for manipulating image pixels.
- **CryptoJS (AES Encryption)** for secure message storage

CODE :

index.html

html
CopyEdit

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Secure Steganography Web App</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-js.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/lz-string/1.4.4/lz-string.min.js"></script>
    <link rel="stylesheet" href="styles.css">
</head>
<body class="bg-gray-900 text-white flex justify-center items-center min-h-screen">
    <div class="w-full max-w-2xl p-6 bg-gray-800 rounded-lg shadow-lg text-center">
        <h1 class="text-3xl font-bold mb-4 text-green-400">Secure Steganography Web App</h1>

        <input type="file" id="imageInput" accept="image/*" class="block w-full p-2 bg-gray-700 rounded-lg mb-4">

        <textarea id="message" placeholder="Enter your secret message..." class="w-full p-3 bg-gray-700 rounded-
lg mb-4"></textarea>

        <input type="password" id="key" placeholder="Enter secret key..." class="w-full p-3 bg-gray-700 rounded-lg
mb-4">

        <button onclick="hideMessage()" class="w-full bg-green-500 p-3 rounded-lg hover:bg-green-600 mt-2 font-
semibold">🔒 Hide Message</button>
        <button onclick="extractMessage()" class="w-full bg-blue-500 p-3 rounded-lg hover:bg-blue-600 mt-2 font-
semibold">🔒 Extract Message</button>

        <canvas id="canvas" class="hidden"></canvas>
        <p id="output" class="mt-4 text-lg text-yellow-300 font-semibold"></p>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

script.js (With Compression & Encryption)

javascript
CopyEdit

```
function hideMessage() {
    const fileInput = document.getElementById("imageInput");
    const message = document.getElementById("message").value.trim();
    const key = document.getElementById("key").value.trim();

    if (fileInput.files.length === 0 || message === "" || key === "") {
        alert("Please select an image, enter a message, and provide a secret key.");
        return;
    }

    // Compress and Encrypt message
    const compressedMessage = LZString.compress(message);
    const encryptedMessage = CryptoJS.AES.encrypt(compressedMessage, key).toString();

    const reader = new FileReader();
    reader.onload = function(event) {
        const img = new Image();
        img.src = event.target.result;
        img.onload = function() {
            const canvas = document.getElementById("canvas");
            const ctx = canvas.getContext("2d");
            canvas.width = img.width;
            canvas.height = img.height;
            ctx.drawImage(img, 0, 0);
            const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);
            const data = imageData.data;

            // Convert message to binary
            let binaryMessage = "";
            for (let i = 0; i < encryptedMessage.length; i++) {
                let binaryChar = encryptedMessage.charCodeAt(i).toString(2).padStart(8, '0');
                binaryMessage += binaryChar;
            }
            binaryMessage += "00000000"; // End marker

            let messageIndex = 0;
            for (let i = 0; i < data.length; i += 4) {
                if (messageIndex < binaryMessage.length) {
                    data[i] = (data[i] & 0xFE) | parseInt(binaryMessage[messageIndex], 2);
                    messageIndex++;
                }
            }

            ctx.putImageData(imageData, 0, 0);

            // Save the new image
            const downloadLink = document.createElement("a");
            downloadLink.href = canvas.toDataURL("image/png");
            downloadLink.download = "secure_message.png";
            document.body.appendChild(downloadLink);
            downloadLink.click();
            document.body.removeChild(downloadLink);

            alert("Message hidden successfully!");
        };
    };
    reader.readAsDataURL(fileInput.files[0]);
}

function extractMessage() {
    const fileInput = document.getElementById("imageInput");
    const key = document.getElementById("key").value.trim();

    if (fileInput.files.length === 0 || key === "") {
        alert("Please select an image and enter the secret key.");
        return;
    }

    const reader = new FileReader();
    reader.onload = function(event) {
        const img = new Image();
        img.src = event.target.result;
        img.onload = function() {
            const canvas = document.getElementById("canvas");
            const ctx = canvas.getContext("2d");
            canvas.width = img.width;
            canvas.height = img.height;
            ctx.drawImage(img, 0, 0);

            const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);
            const data = imageData.data;
            let binaryMessage = "";
            for (let i = 0; i < data.length; i += 4) {
                binaryMessage += (data[i] & 1).toString();
            }

            let extractedMessage = "";
            for (let i = 0; i < binaryMessage.length; i += 8) {
                let byte = binaryMessage.slice(i, i + 8);
                if (byte === "00000000") break;
                extractedMessage += String.fromCharCode(parseInt(byte, 2));
            }

            // Decrypt message
            try {
                const decryptedMessage = CryptoJS.AES.decrypt(extractedMessage,
key).toString(CryptoJS.enc.Utf8);
                const decompressedMessage = LZString.decompress(decryptedMessage);

                if (decompressedMessage) {
                    document.getElementById("output").innerText = "Hidden Message: " + decompressedMessage;
                } else {
                    document.getElementById("output").innerText = "Incorrect Key! Cannot decrypt.";
                }
            } catch (error) {
                document.getElementById("output").innerText = "Incorrect Key! Cannot decrypt.";
            }
        };
    };
    reader.readAsDataURL(fileInput.files[0]);
}
```

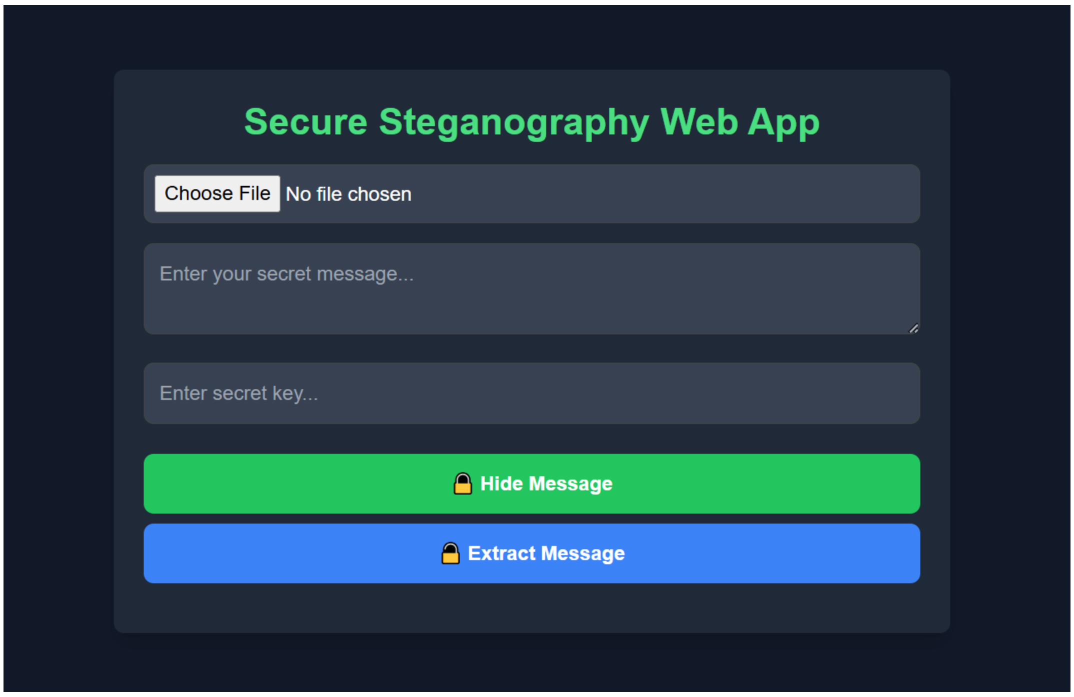
styles.css

```
css
CopyEdit
body {
    font-family: 'Arial', sans-serif;
    background-color: #121212;
    color: white;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

input, textarea {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    background: #2A2A2A;
    border: 1px solid #444;
    color: white;
    border-radius: 6px;
}
```

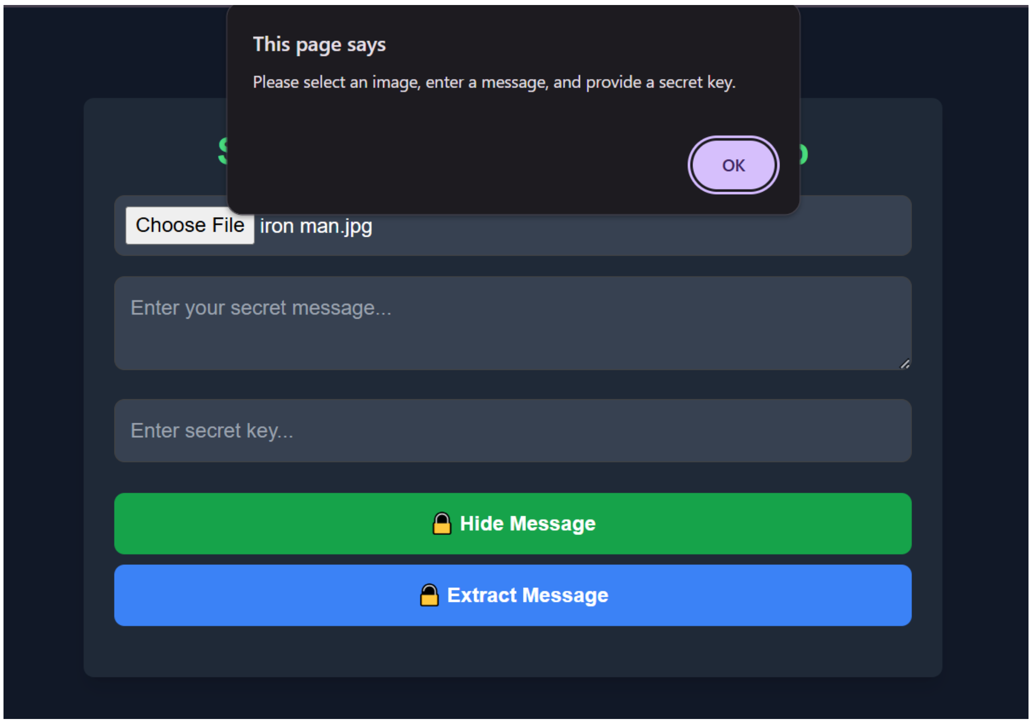
SCREENSHOTS:

1. Blank Page (Initial State)



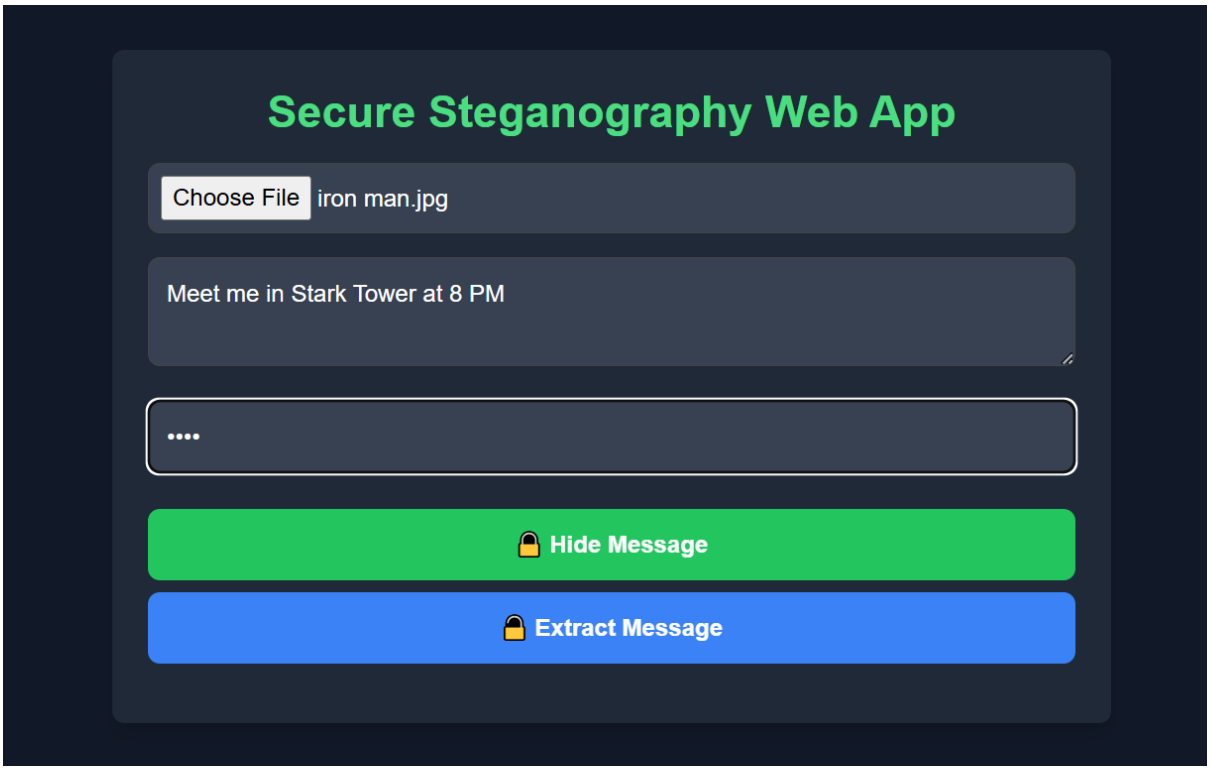
Secure Steganography Web App
Choose File No file chosen
Enter your secret message...
Enter secret key...
🔒 Hide Message
🔒 Extract Message

2. Click on "Hide" Without Entering Details


This page says

Please select an image, enter a message, and provide a secret key.

3. Entering Details (Message and Key)



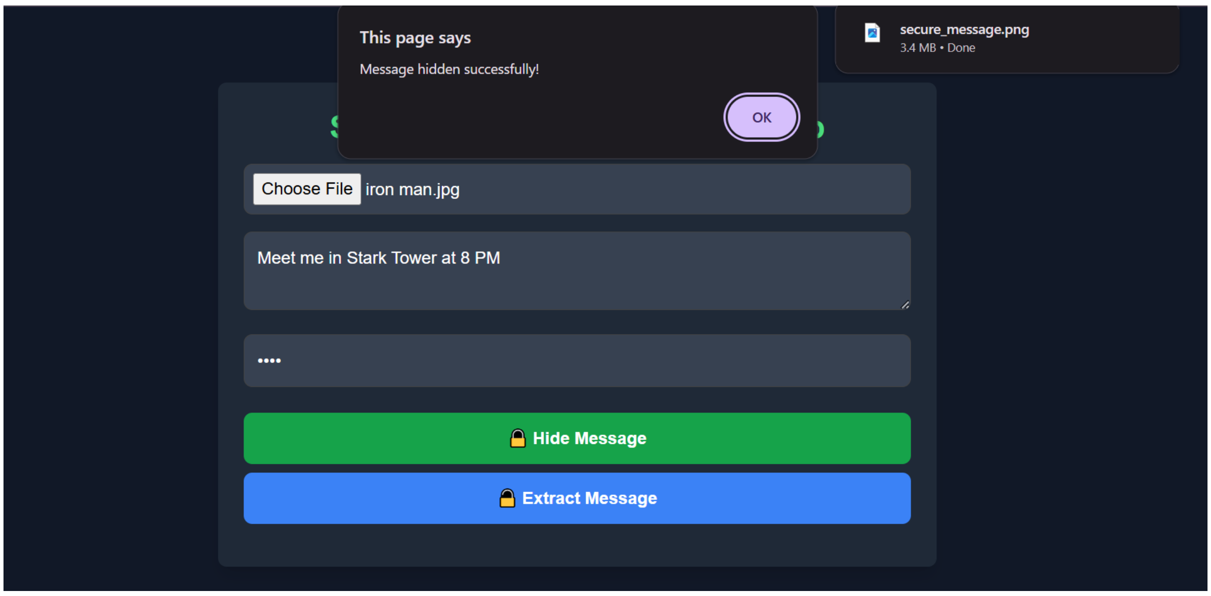
Secure Steganography Web App

Choose File iron man.jpg

Meet me in Stark Tower at 8 PM

••••

4. After Clicking "Hide" (Successful Hiding)



This page says

Message hidden successfully!

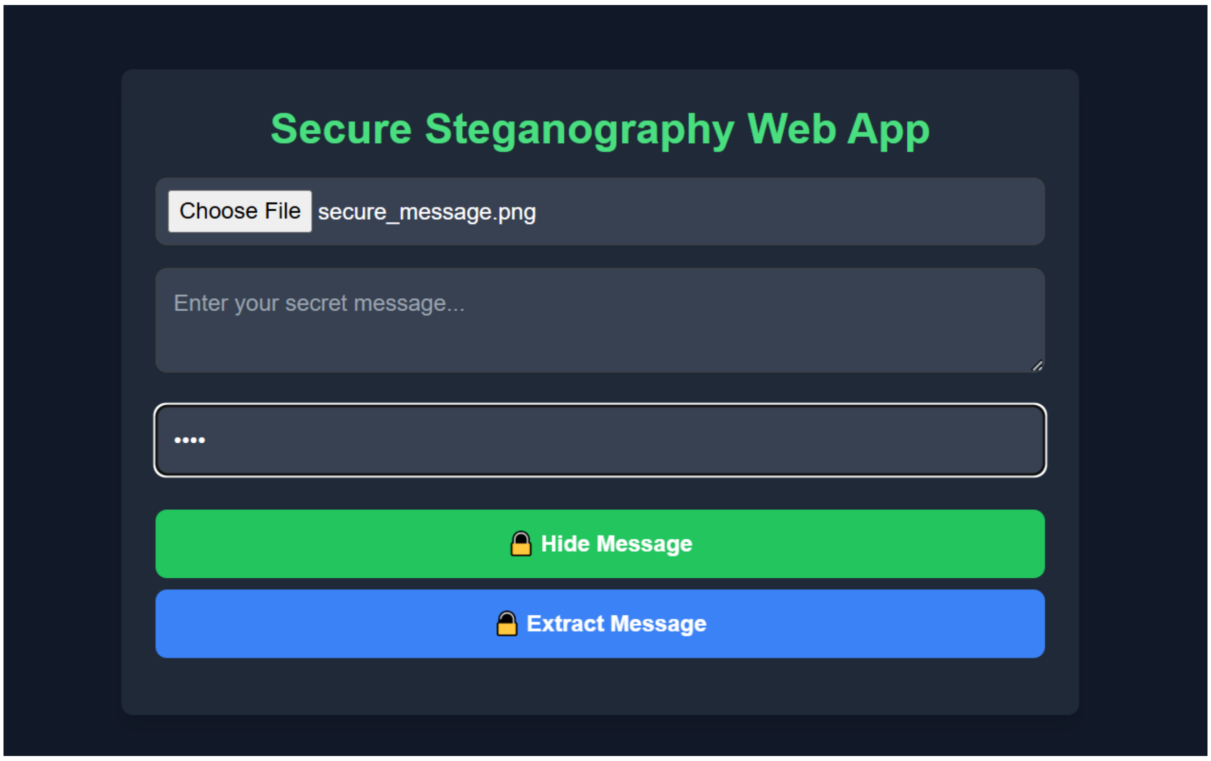
secure_message.png
3.4 MB • Done

Choose File iron man.jpg

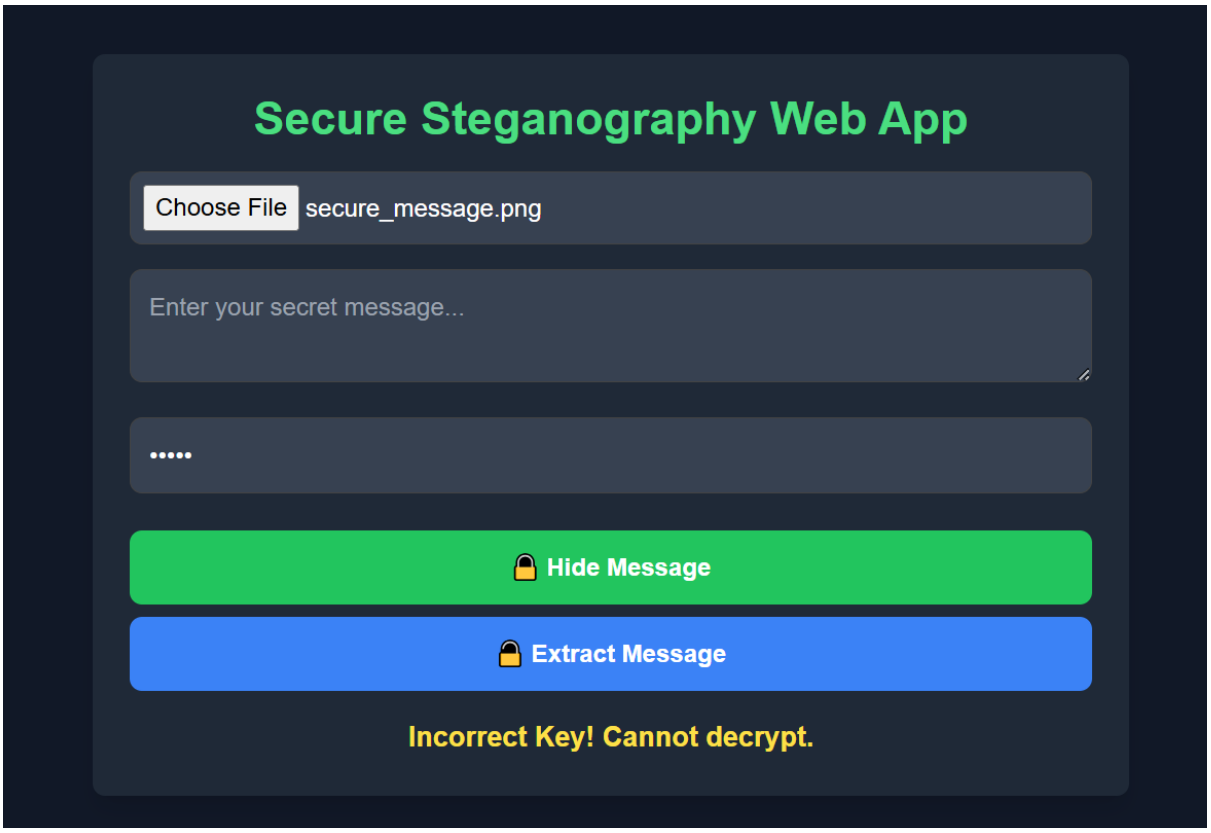
Meet me in Stark Tower at 8 PM

••••

5. Uploading the Secured Image

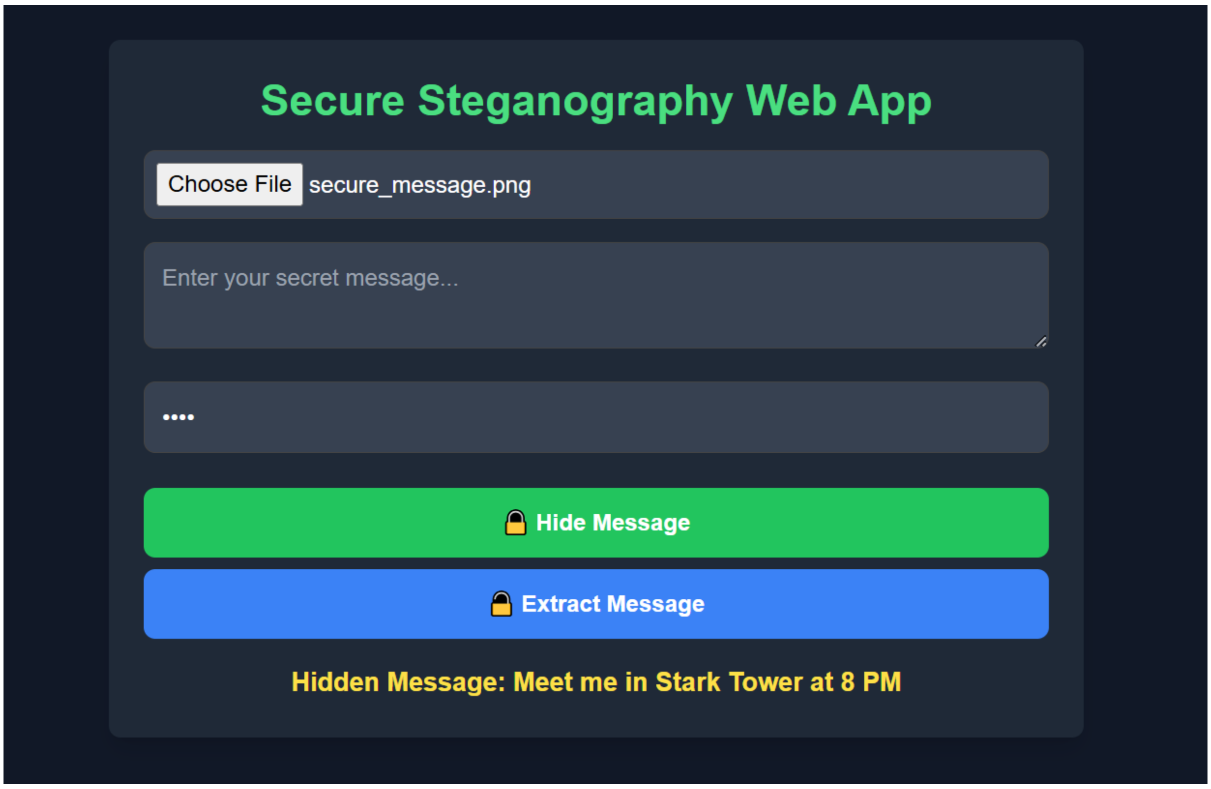


6. Providing the Key (Incorrect Key)



Incorrect Key! Cannot decrypt.

7. Providing the Correct Key and Extracting the Message



secure_message.png

••••

Hidden Message: Meet me in Stark Tower at 8 PM

Conclusion

This project demonstrates how **steganography** can be implemented on a webpage using **JavaScript** and **encryption techniques**. By hiding encrypted messages inside images, we enhance both **data security** and **secrecy**. This approach is useful for **secure communication** and **data privacy** in digital applications.