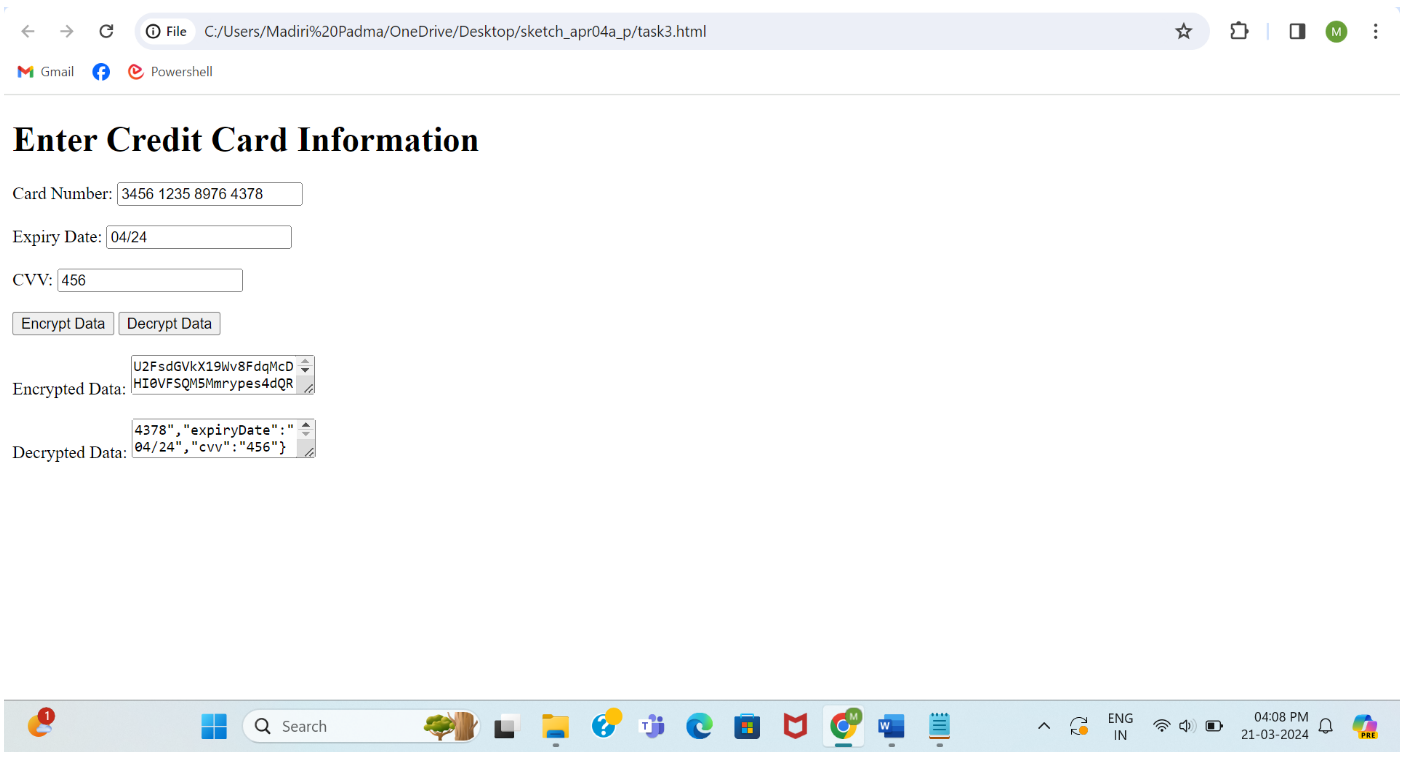
INTERMEDIATE Project:

Task 2 Credit Card Encryption and Decryption

```
<!DOCTYPE html>
<html>
<head>
   <title>Credit Card Encryption and Decryption</title>
   <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-
js.min.js"></script>
</head>
<body>
   <h1>Enter Credit Card Information</h1>
   <form id="creditCardForm">
      <label for="cardNumber">Card Number:</label>
      <input type="text" id="cardNumber" name="cardNumber" required><br><br>

      <label for="expiryDate">Expiry Date:</label>
      <input type="text" id="expiryDate" name="expiryDate" placeholder="MM/YY"
required><br><br>

      <label for="cvv">CVV:</label>
      <input type="text" id="cvv" name="cvv" required><br><br>

      <button type="button" onclick="encryptData()">Encrypt Data</button>
      <button type="button" onclick="decryptData()">Decrypt
Data</button><br><br>

      <label for="encryptedData">Encrypted Data:</label>
      <textarea id="encryptedData" name="encryptedData"
readonly></textarea><br><br>

      <label for="decryptedData">Decrypted Data:</label>
      <textarea id="decryptedData" name="decryptedData"
readonly></textarea><br><br>
   </form>

   <script>
      var key = "ThisIsASecretKey"; // Note: In real applications, generate a secure
key
```

```
        function encryptData() {
            var cardNumber = document.getElementById("cardNumber").value;
            var expiryDate = document.getElementById("expiryDate").value;
            var cvv = document.getElementById("cvv").value;

            var data = {
                cardNumber: cardNumber,
                expiryDate: expiryDate,
                cvv: cvv
            };

            var encrypted = CryptoJS.AES.encrypt(JSON.stringify(data), key).toString();
            document.getElementById("encryptedData").value = encrypted;
        }

        function decryptData() {
            var encryptedData = document.getElementById("encryptedData").value;

            try {
                var decrypted = CryptoJS.AES.decrypt(encryptedData,
key).toString(CryptoJS.enc.Utf8);
                document.getElementById("decryptedData").value = decrypted;
            } catch (e) {
                document.getElementById("decryptedData").value = "Decryption failed.
Please check the encrypted data.";
            }
        }
    </script>
</body>
</html>
```

Output



C:/Users/Madiri%20Padma/OneDrive/Desktop/sketch_apr04a_p/task3.html

Enter Credit Card Information

Card Number: 3456 1235 8976 4378

Expiry Date: 04/24

CVV: 456

Encrypt Data Decrypt Data

Encrypted Data:
```
U2FsdGVkX19Wv8FdqMcD
HI0VFSQM5Mmrypes4dQR
```

Decrypted Data:
```
4378","expiryDate":"
04/24","cvv":"456"}
```

04:08 PM
21-03-2024

Task 1 Web-Based Facial Authentication System

HTML CODE

```
<!DOCTYPE html>
<html>
<head>
    <title>Web-Based Facial Authentication System</title>
    <script src="https://docs.opencv.org/4.5.5/opencv.js"></script>
</head>
<body>
    <h1>Facial Authentication</h1>
    <video id="video" width="640" height="480" autoplay></video><br>
    <canvas id="canvas" width="640" height="480"></canvas><br>
    <button onclick="startDetection()">Start Detection</button>
    <button onclick="stopDetection()">Stop Detection</button>
    <p id="result"></p>

    <script>
        let video = document.getElementById('video');
        let canvas = document.getElementById('canvas');
        let ctx = canvas.getContext('2d');
        let resultElement = document.getElementById('result');
        let isDetecting = false;
        let faces = new cv.RectVector();

        // Load the face detection model
        cv['onRuntimeInitialized'] = () => {
          let faceCascade = new cv.CascadeClassifier();
          faceCascade.load('haarcascade_frontalface_default.xml');

          // Start the video stream and detect faces
          function detectFaces() {
            if (!isDetecting) return;
            ctx.drawImage(video, 0, 0, canvas.width, canvas.height);
            let frame = new cv.Mat(canvas.height, canvas.width, cv.CV_8UC4);
            cv.imshow(frame, canvas);
            cv.cvtColor(frame, frame, cv.COLOR_RGBA2GRAY, 0);
            faceCascade.detectMultiScale(frame, faces);
            if (faces.size() > 0) {
              resultElement.textContent = 'Face Detected';
            } else {
              resultElement.textContent = 'No Face Detected';
            }
            frame.delete();
            requestAnimationFrame(detectFaces);
          }

          // Start face detection
          function startDetection() {
            isDetecting = true;
            navigator.mediaDevices.getUserMedia({ video: true })
              .then((stream) => {
                video.srcObject = stream;
                video.play();
                detectFaces();
              })
              .catch((error) => {
                console.error('Error accessing webcam:', error);
              });
          }

          // Stop face detection
          function stopDetection() {
```

```
            isDetecting = false;
            video.srcObject.getTracks().forEach((track) => {
                track.stop();
            });
        }

        // Load the video and start face detection
        video.addEventListener('loadeddata', () => {
            console.log('Video loaded');
        });

        // Clean up on page close
        window.addEventListener('beforeunload', () => {
            if (isDetecting) {
                stopDetection();
            }
        });
    };
  </script>
</body>
</html>
```