3 Project Inputs and Outputs

3.1 Input Details, Outputs – Screenshots

1. Input HTML Code for main webpage of Cipher Shield –

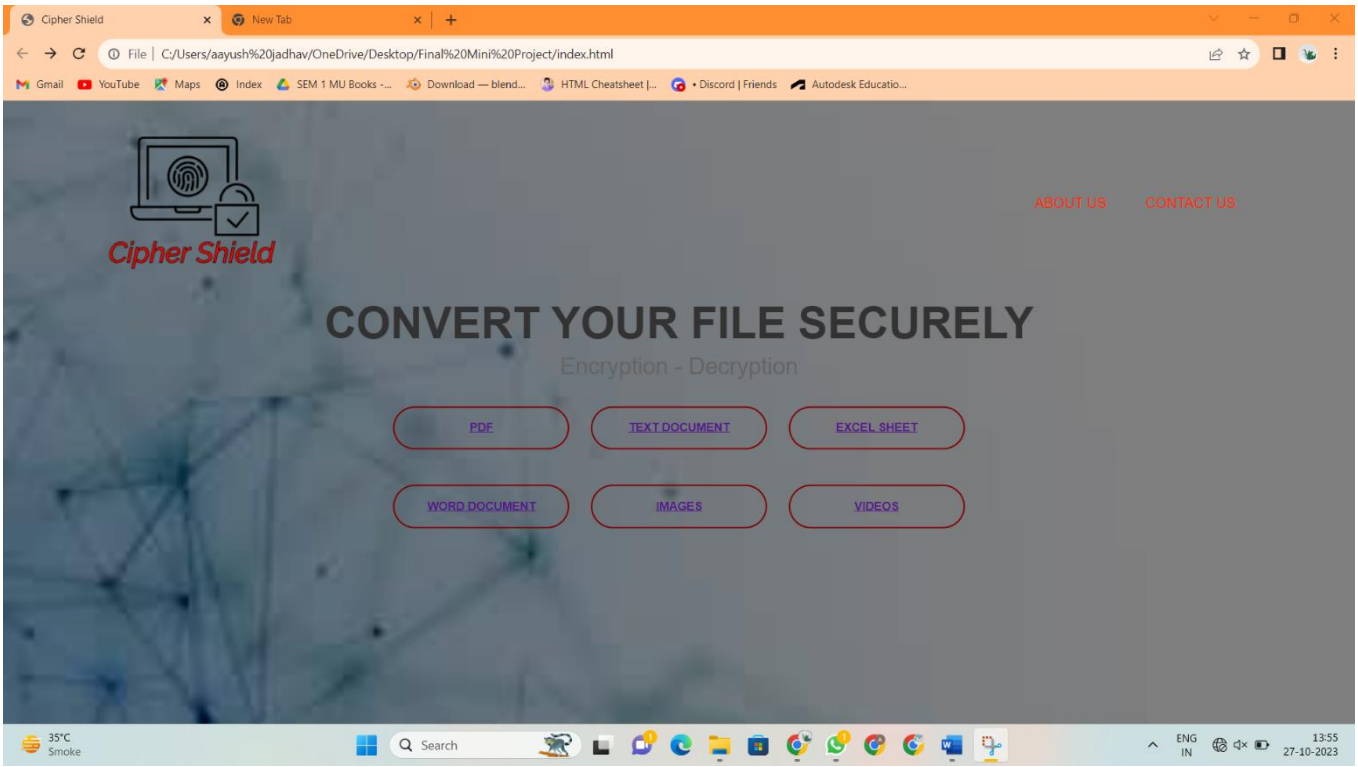
```
<!DOCTYPE html>
<html>
  <head>
    <title> Cipher Shield </title>
    <link rel="stylesheet" href="style.css">
    <style>
      /* CSS for the heading */
      .content h1 {
        font-family: Arial, sans-serif;
        font-size: 50px; /* Adjust the font size for the heading */
        font-weight: bold; /* Optionally set the font weight for the heading */
        color: #333; /* Set the text color for the heading */
      }

      /* CSS for the paragraph */
      .content p {
        font-family: "Your Paragraph Font", Arial, sans-serif; /* Specify the font for the
paragraph */
        font-size: 26px; /* Adjust the font size for the paragraph */
        font-weight: normal; /* Optionally set the font weight for the paragraph */
        color: #666; /* Set the text color for the paragraph */
      }
    </style>
  </head>
  <body>
    <div class="banner">
      <div class="navbar">
        <img src="logo.png" class="logo">
        <ul>
          <li><a href="about_us.html">ABOUT US</a></li>
          <li><a href="contact_us.html">CONTACT US</a></li>
        </ul>
      </div>

      <div class="content">
        <h1>CONVERT YOUR FILE SECURELY</h1>
        <p>Encryption - Decryption</p>

        <div>
          <button type="button"><span></span><a href="pdf.html">PDF</a></button>
          <button type="button"><span></span><a href="text.html">TEXT DOCUMENT</a></button>
          <button type="button"><span></span><a href="excel.html">EXCEL SHEET</a></button><br>
          <button type="button"><span></span><a href="word.html">WORD DOCUMENT</a></button>
          <button type="button"><span></span><a href="image.html">IMAGES</a></button>
          <button type="button"><span></span><a href="video.html">VIDEOS</a></button>
        </div>
      </div>
    </div>
  </body>
</html>
```

2. Input CSS Code for Designing main webpage of Cipher Shield –

```
*{
  margin: 0;
  padding: 0;
  font-family: sans-serif;
 }

.banner{
   width: 100%;
   height: 100vh;
   background-image: linear-gradient(rgba(0,0,0,0.50),rgba(0,0,0,0.50)),url(bg1.jpeg);
   background-size: cover;
   background-position: center;
}

.navbar{
   width: 85%;
   margin: auto;
   padding: 35px 0;
   display: flex;
   align-items: center;
   justify-content: space-between;
}

.logo{
   width: 200px;
   cursor: pointer;
}

.navbar ul li{
    list-style: none;
    display: inline-block;
    margin: 0 20px;
    position: relative;
}

.navbar ul li a{
   text-decoration: none;
   color: #ff2400;
}

.navbar ul li::after{
   content: '';
   height: 3px;
   width: 0;
   background: #810000;
   position: absolute;
   left: 0;
   bottom: -10px;
   transition: 0.5s;
}

.navbar ul li:hover::after{
   width: 100%;
}

.content{
   width: 100%;
   position: absolute;
   top: 50%;
```

```
   transform: translateY(-60%);
   text-align: center;
   color: #fff;
}

.content h1{
   font-size: 100px;
   margin-top: 50px;
}

.content{
   margin: 20px auto;
   font-weight: 100;
   line-height: 50px;
}
content p {
        font-family: "Your Chosen Font", Arial, sans-serif; /* Specify the font you want to use
*/
        font-size: 36px; /* Adjust the font size */
        font-weight: bold; /* Optionally set the font weight */
        color: #333; /* Set the text color */
      }


button{
   width: 200px;
   padding: 15px 0;
   text-align: center;
   margin: 20px 10px;
   border-radius: 25px;
   font-weight: bold;
   border: 2px solid #810000;
   background: transparent;
   color: #ff2400;
   cursor: pointer;
   position: relative;
   overflow: hidden;
}

span{
   background: #810000;
   height: 100%;
   width: 0;
   border-radius: 25px;
   position: absolute;
   left: 0;
   bottom: 0;
   z-index: -1;
   transition: 0.5s;
}

button:hover span{
   width: 100%;
}

button:hover{
   border: none:
}
```

3. Output –

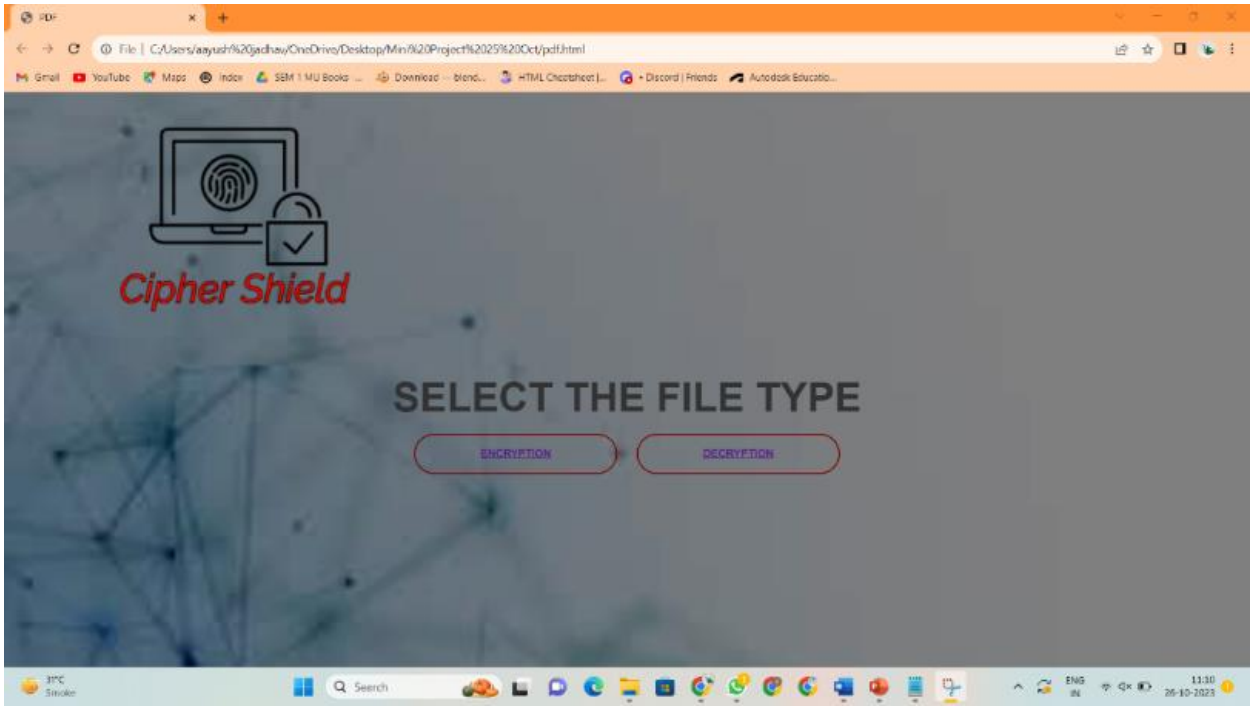


Main Webpage of Cipher Shield Website

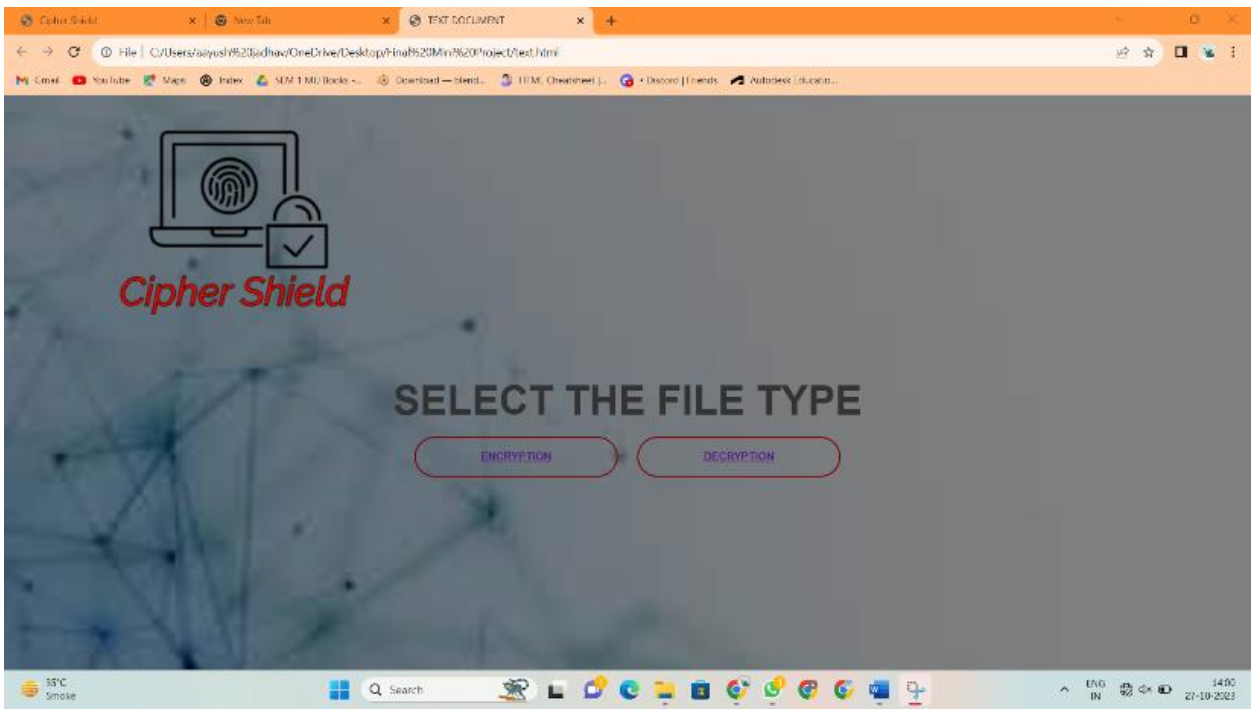
4. Input HTML Code for webpage of PDF, Text Document –

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <link rel="stylesheet" href="sub_style.css">
    <style>
      /* CSS for the heading */
      .content h1 {
        font-family: Arial, sans-serif;
        font-size: 50px; /* Adjust the font size for the heading */
        font-weight: bold; /* Optionally set the font weight for the heading */
        color: #333; /* Set the text color for the heading */
      }
      .navbar {
        display: flex;
        justify-content: space-between;
        align-items: center;
      }
      .logo {
        width: 300px; /* Adjust the width as needed */
        cursor: pointer;
        margin-right: auto; /* Push the logo to the left corner */
      }
    </style>
  </head>
  <body>
    <div class="banner">
      <div class="navbar">
        <img src="logo.png" class="logo">
      </div>
      <div class="content">
        <h1>SELECT THE FILE TYPE</h1>
        <div>
          <button type="button"><span></span><a
href="pdfE.html">ENCRYPTION</a></button>
          <button type="button"><span></span><a
href="pdfD.html">DECRYPTION</a></button>
        </div>
      </div>
    </div>
  </body>
</html>
```

5. Output –



Webpage of PDF Tab



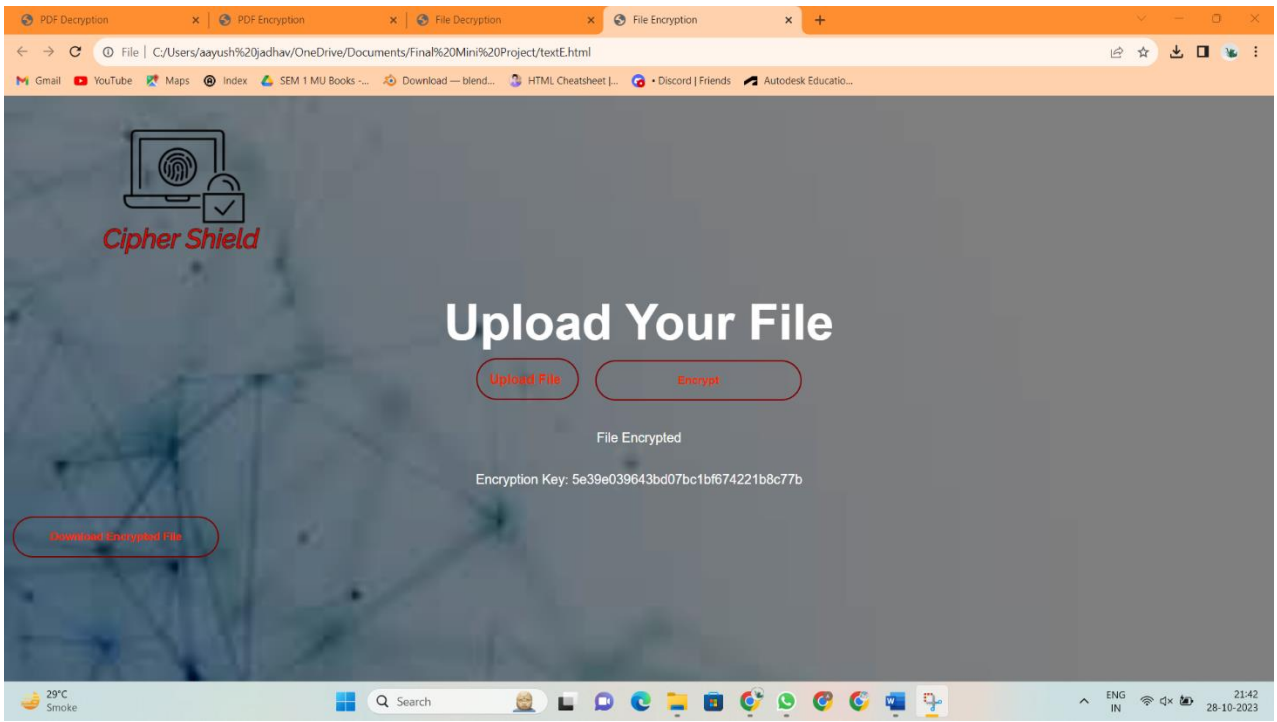
Webpage of Text Document Tab

6. Input JavaScript Code to Encrypt a Text Document –

```
<!DOCTYPE html>
<html>
<head>
    <title>File Encryption</title>
    <link rel="stylesheet" href="sub_style2.1.css">
</head>
<body>
    <div class="banner">
        <div class="navbar">
            <img src="logo.png" class="logo">
        </div>
        <div class="content">
            <h1>Upload Your File</h1>
            <label for="file-input" class="upload-button">
                <span></span>Upload File
                <input type="file" id="file-input" accept=".txt" required>
            </label>
            <button onclick="encryptFile()">Encrypt</button>
            <p id="result"></p>
            <p id="key-display" style="display: none">Encryption Key: <span id="key-
value"></span></p>
            <button id="download-button" onclick="downloadEncryptedFile()"
style="display: none; margin-left: 10px;">Download Encrypted File</button>
        </div>
    </div>
    <script>
        let encryptionKey; // Variable to store the encryption key
        let encryptedArray;

        async function encryptFile() {
            const fileInput = document.getElementById('file-input');
            const resultElement = document.getElementById('result');
            const keyDisplay = document.getElementById('key-display');
            const keyValue = document.getElementById('key-value');
            const downloadButton = document.getElementById('download-button');

            if (fileInput.files.length > 0) {
                const file = fileInput.files[0];
                const reader = new FileReader();

                reader.onload = async (e) => {
                    const fileContent = e.target.result;

                    // Generate a random encryption key
                    encryptionKey = new Uint8Array(16); // 128 bits (16 bytes)
                    crypto.getRandomValues(encryptionKey);

                    // Convert the key to a hexadecimal string for display
                    const keyString = Array.from(encryptionKey).map(byte =>
byte.toString(16).padStart(2, '0')).join('');

                    // Import the key as a CryptoKey
                    const importedKey = await crypto.subtle.importKey(
```

```
                        'raw', // Key format
                        encryptionKey,
                        'AES-GCM',
                        true, // Extractable
                        ['encrypt']
                    );

                    // Encode the file content as a base64 string
                    const encodedData = btoa(fileContent);

                    // Encrypt the file using the Web Crypto API
                    const encryptedData = await crypto.subtle.encrypt(
                        { name: 'AES-GCM', iv: encryptionKey },
                        importedKey, // Use the imported key
                        new TextEncoder().encode(encodedData)
                    );

                    encryptedArray = new Uint8Array(encryptedData);

                    resultElement.textContent = 'File Encrypted';
                    keyDisplay.style.display = 'block';
                    keyValue.textContent = keyString;
                    downloadButton.style.display = 'block';
                };

                reader.readAsText(file);
            } else {
                resultElement.textContent = 'Please select a file.';
                keyDisplay.style.display = 'none';
                downloadButton.style.display = 'none';
            }
        }

        function downloadEncryptedFile() {
            if (encryptedArray) {
                // Replace 'encrypted_file.enc' with your desired file name
                const fileName = 'encrypted_file.enc';

                // Create a Blob with the encrypted data
                const encryptedBlob = new Blob([encryptedArray], { type:
'application/octet-stream' });

                // Create a link and trigger the download
                const a = document.createElement('a');
                a.href = URL.createObjectURL(encryptedBlob);
                a.download = fileName;
                a.click();
            }
        }
    </script>
</body>
</html>
```

7. Output –



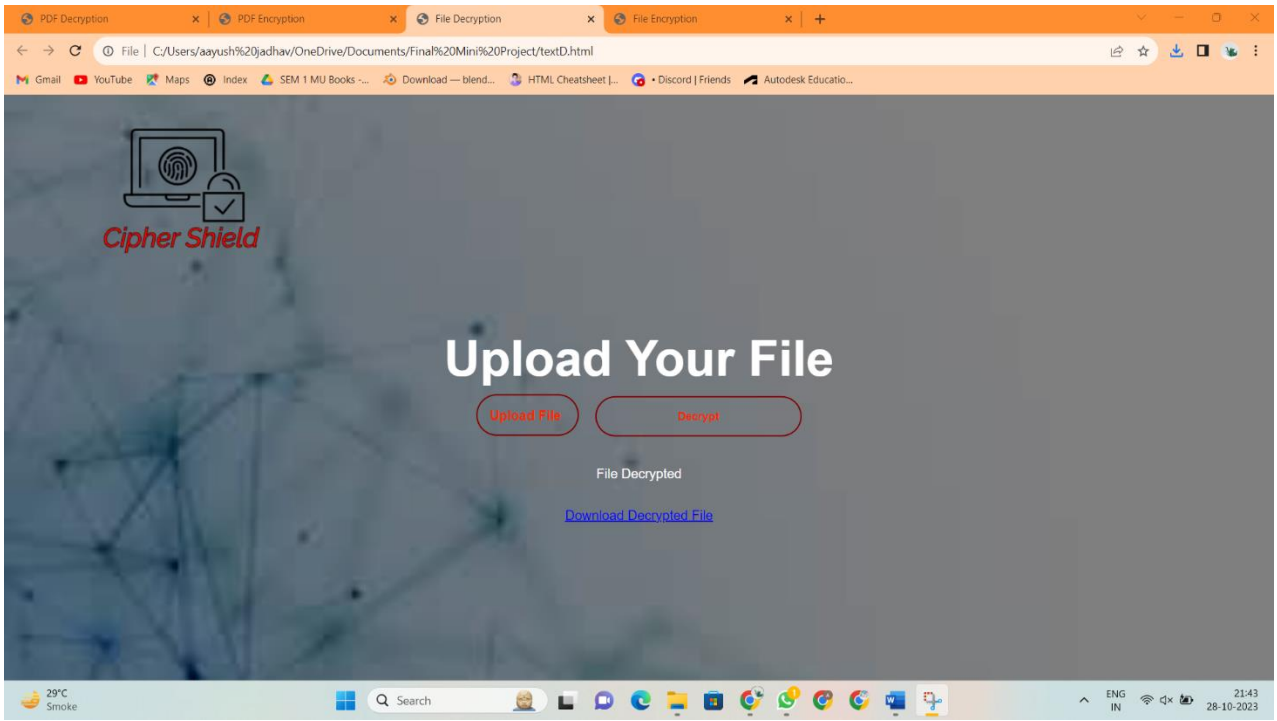
Text Document tab after encrypting the text file

8. Input JavaScript Code to Decrypt a Text Document –

```
<!DOCTYPE html>
<html>
<head>
    <title>File Decryption</title>
    <link rel="stylesheet" href="sub_style2.1.css">
</head>
<body>
    <div class="banner">
      <div class="navbar">
        <img src="logo.png" class="logo">
    </div>
    <div class="content">
    <h1>Upload Your File</h1>
    <label for="file-input" class="upload-button">
      <span></span>Upload File
      <input type="file" id="file-input" accept=".enc" required>
    </label>
    <button onclick="decryptFile()">Decrypt</button>
    <p id="result"></p>
    <p id="key-display" style="display: none">Encryption Key: <span id="key-
value"></span></p>
    <a id="download-link" style="display: none" download="decrypted_file.txt">Download
Decrypted File</a>
   </div>

    <script>
        let decryptionKey; // Variable to store the decryption key

        async function decryptFile() {
            const fileInput = document.getElementById('file-input');
            const resultElement = document.getElementById('result');
            const keyDisplay = document.getElementById('key-display');
            const keyValue = document.getElementById('key-value');
            const downloadLink = document.getElementById('download-link');

            if (fileInput.files.length > 0) {
                const file = fileInput.files[0];
                const reader = new FileReader();

                reader.onload = async (e) => {
                    const fileContent = e.target.result;

                    const key = prompt('Enter the encryption key (32 hexadecimal
characters):');
                    if (!key || key.length !== 32) {
                        return;
                    }

                    try {
                        // Convert the key from a hexadecimal string to a Uint8Array
                        const keyArray = new Uint8Array(16);
                        for (let i = 0; i < 16; i++) {
                            keyArray[i] = parseInt(key.substr(i * 2, 2), 16);
```

```
                    }

                    // Import the key as a CryptoKey
                    const importedKey = await crypto.subtle.importKey(
                        'raw', // Key format
                        keyArray,
                        'AES-GCM',
                        true, // Extractable
                        ['decrypt']
                    );

                    // Decrypt the file using the Web Crypto API
                    const decryptedData = await crypto.subtle.decrypt(
                        { name: 'AES-GCM', iv: keyArray },
                        importedKey, // Use the imported key
                        new Uint8Array(fileContent)
                    );

                    const decryptedText = new TextDecoder().decode(decryptedData);

                    // Decode the decrypted base64 data
                    const decodedData = atob(decryptedText);

                    resultElement.textContent = 'File Decrypted';
                    keyDisplay.style.display = 'none';
                    keyValue.textContent = ''; // Clear the displayed key
                    const decryptedBlob = new Blob([decodedData], { type:
'text/plain' });
                    downloadLink.href = URL.createObjectURL(decryptedBlob);
                    downloadLink.style.display = 'block';
                } catch (error) {
                    resultElement.textContent = 'Decryption failed. Please check
the key.';
                    keyDisplay.style.display = 'none';
                    keyValue.textContent = ''; // Clear the displayed key
                    downloadLink.style.display = 'none';
                }
            };

            reader.readAsArrayBuffer(file);
        } else {
            resultElement.textContent = 'Please select a file.';
            keyDisplay.style.display = 'none';
            keyValue.textContent = ''; // Clear the displayed key
            downloadLink.style.display = 'none';
        }
    }
</script>
</body>
</html>
```

9. Output –



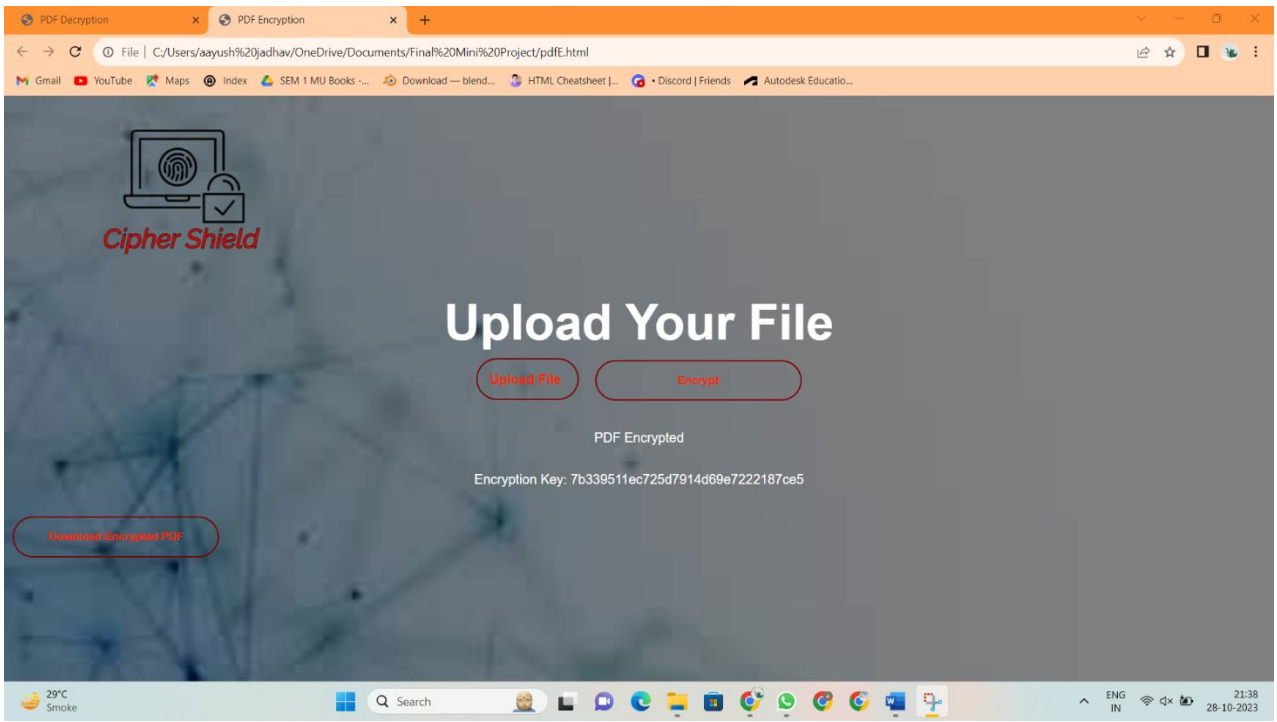
Text Document tab after decrypting the text file

10. Input JavaScript Code to Encrypt a PDF –

```
<!DOCTYPE html>
<html>
<head>
    <title>PDF Encryption</title>
    <link rel="stylesheet" href="sub_style2.1.css">
</head>
<body>
    <div class="banner">
        <div class="navbar">
            <img src="logo.png" class="logo">
        </div>
        <div class="content">
            <h1>Upload Your File</h1>
            <label for="file-input" class="upload-button">
                <span></span>Upload File
                <input type="file" id="file-input" accept=".pdf" required>
            </label>
            <button id="encrypt-button" onclick="encryptPDF()">Encrypt</button>
            <p id="result"></p>
            <p id="key-display" style="display: none">Encryption Key: <span id="key-
value"></span></p>
            <button id="download-button" onclick="downloadEncryptedPDF()"
style="display: none; margin-left: 10px;">Download Encrypted PDF</button>
        </div>
    </div>

    <script>
        let encryptionKey; // Variable to store the encryption key
        let encryptedArray; // Variable to store encrypted data

        async function encryptPDF() {
            const fileInput = document.getElementById('file-input');
            const resultElement = document.getElementById('result');
            const keyDisplay = document.getElementById('key-display');
            const keyValue = document.getElementById('key-value');
            const downloadButton = document.getElementById('download-button');

            if (fileInput.files.length > 0) {
                const file = fileInput.files[0];
                const reader = new FileReader();

                reader.onload = async (e) => {
                    const fileContent = e.target.result;

                    // Generate a random encryption key
                    encryptionKey = new Uint8Array(16); // 128 bits (16 bytes)
                    crypto.getRandomValues(encryptionKey);

                    // Convert the key to a hexadecimal string for display
                    const keyString = Array.from(encryptionKey)
                        .map(byte => byte.toString(16).padStart(2, '0'))
                        .join('');
```

```
                    // Import the key as a CryptoKey
                    const importedKey = await crypto.subtle.importKey(
                        'raw', // Key format
                        encryptionKey,
                        'AES-GCM',
                        true, // Extractable
                        ['encrypt']
                    );

                    // Encrypt the PDF file using the Web Crypto API
                    const encryptedData = await crypto.subtle.encrypt(
                        { name: 'AES-GCM', iv: encryptionKey },
                        importedKey, // Use the imported key
                        new Uint8Array(fileContent)
                    );

                    encryptedArray = new Uint8Array(encryptedData); // Store encrypted
data

                    resultElement.textContent = 'PDF Encrypted';
                    keyDisplay.style.display = 'block';
                    keyValue.textContent = keyString;
                    downloadButton.style.display = 'block';
                };

                reader.readAsArrayBuffer(file);
            } else {
                resultElement.textContent = 'Please select a PDF file.';
                keyDisplay.style.display = 'none';
                downloadButton.style.display = 'none';
            }
        }

        function downloadEncryptedPDF() {
            if (encryptionKey) {
                // Replace 'encrypted_file.enc' with your desired file name
                const fileName = 'encrypted_file.enc';

                // Create a Blob with the encrypted data
                const encryptedBlob = new Blob([encryptedArray], { type:
'application/octet-stream' });

                // Create a link and trigger the download
                const a = document.createElement('a');
                a.href = URL.createObjectURL(encryptedBlob);
                a.download = fileName;
                a.click();
            }
        }
    </script>
</body>
</html>
```

11.Output –



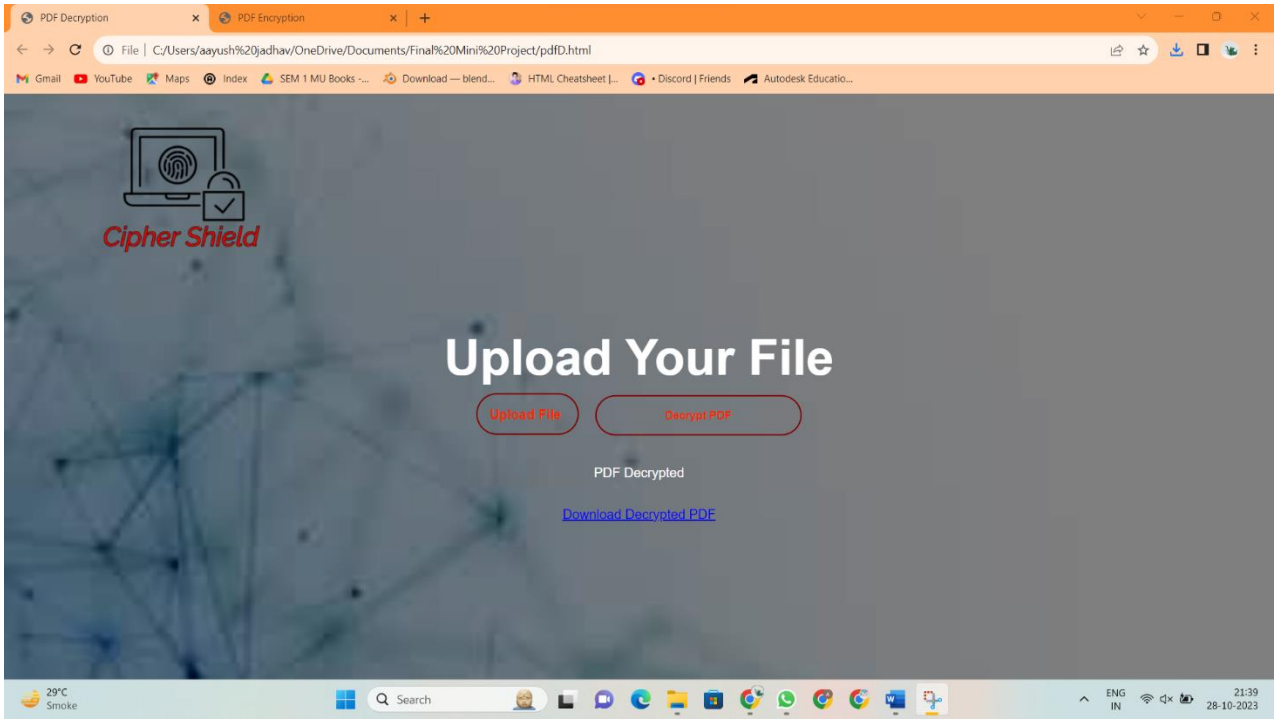
PDF tab after encrypting the pdf

12.Input JavaScript Code to Decrypt a PDF –

```
<!DOCTYPE html>
<html>
<head>
   <title>PDF Decryption</title>
   <link rel="stylesheet" href="sub_style2.1.css">
</head>
<body>
   <div class="banner">
      <div class="navbar">
         <img src="logo.png" class="logo">
      </div>
   <div class="content">
   <h1>Upload Your File</h1>
   <label for="file-input" class="upload-button">
            <span></span>Upload File
            <input type="file" id="file-input" accept=".enc" required>
   </label>

   <button onclick="decryptPDF()">Decrypt PDF</button>
   <p id="result"></p>
   <p id="key-display" style="display: none">Encryption Key: <span id="key-value"></span></p>
   <a id="download-link" style="display: none" download="decrypted_file.pdf">Download Decrypted PDF</a>
  </div>
   <script>
      let decryptionKey; // Variable to store the decryption key

      async function decryptPDF() {
         const fileInput = document.getElementById('file-input');
         const resultElement = document.getElementById('result');
         const keyDisplay = document.getElementById('key-display');
         const keyValue = document.getElementById('key-value');
         const downloadLink = document.getElementById('download-link');

         if (fileInput.files.length > 0) {
            const file = fileInput.files[0];
            const reader = new FileReader();

            reader.onload = async (e) => {
               const fileContent = e.target.result;

               const key = prompt('Enter the encryption key (32 hexadecimal characters):');
               if (!key || key.length !== 32) {
                  return;
               }

               try {
                  // Convert the key from a hexadecimal string to a Uint8Array
                  const keyArray = new Uint8Array(16);
                  for (let i = 0; i < 16; i++) {
                     keyArray[i] = parseInt(key.substr(i * 2, 2), 16);
                  }

                  // Import the key as a CryptoKey
                  const importedKey = await crypto.subtle.importKey(
```

```
                    'raw', // Key format
                    keyArray,
                    'AES-GCM',
                    true, // Extractable
                    ['decrypt']
                );

                // Decrypt the file using the Web Crypto API
                const decryptedData = await crypto.subtle.decrypt(
                    { name: 'AES-GCM', iv: keyArray },
                    importedKey, // Use the imported key
                    new Uint8Array(fileContent)
                );

                // Create a Blob for the decrypted PDF
                const decryptedBlob = new Blob([decryptedData], { type: 'application/pdf' });

                resultElement.textContent = 'PDF Decrypted';
                keyDisplay.style.display = 'none';
                keyValue.textContent = ''; // Clear the displayed key
                downloadLink.href = URL.createObjectURL(decryptedBlob);
                downloadLink.style.display = 'block';
            } catch (error) {
                resultElement.textContent = 'Decryption failed. Please check the key.';
                keyDisplay.style.display = 'none';
                keyValue.textContent = ''; // Clear the displayed key
                downloadLink.style.display = 'none';
            }
        };

        reader.readAsArrayBuffer(file);
    } else {
        resultElement.textContent = 'Please select an encrypted PDF file.';
        keyDisplay.style.display = 'none';
        keyValue.textContent = ''; // Clear the displayed key
        downloadLink.style.display = 'none';
    }
}
</script>
</body>
</html>
```

13.Output –



PDF tab after decrypting the pdf