Front-End & Back-End

웹 클라이언트 개발 명세서

문서번호 : DS-103

VER1.0

```jsx
import './css/App.css';
import { Route, Routes } from 'react-router-dom';
import { Link } from 'react-router-dom'; // React Router의 Link 컴포넌트 임포트
import Main from './components/Main.js';
import Contact from './components/Contact.js';
import Donation from './components/Donation.js';
import Packing from './components/Packing.js';
import VirusScan from './components/VirusScan.js';
import UserGuide from './components/UserGuide.js';
import TermsOfUse from './components/TermsOfUse.js';
import PrivacyPolicy from './components/PrivacyPolicy.js';
import PreVision from './images/Pre-Vision.png';
function App() {
  return (
    <>
      {/* 네비게이션 바 */}
      <div className="nav">
        <Link className="nav__logo" to="/"><img className="nav__logo__img" src={PreVision} alt="TeamLogo" /></Link>
        <div className="nav__right">
          <Link to="/user-guide" className="nav__right__a"><h2>User Guide </h2></Link>
          <Link to="/virus-scan" className="nav__right__a"><h2>Virus Scan </h2></Link>
          <Link to="/packing" className="nav__right__a"><h2>Packing</h2></Link>
        </div>
      </div>

      {/* 각 페이지 컴포넌트화 */}
      <div className="content">
        <Routes>
          <Route path="/" element={<Main/>} />
          <Route path="/contact" element={<Contact/>} />
          <Route path="/donation" element={<Donation/>} />
          <Route path="/Packing" element={<Packing/>} />
          <Route path="/virus-scan" element={<VirusScan/>} />
          <Route path="/user-guide" element={<UserGuide/>} />
          <Route path="/terms" element={<TermsOfUse/>} />
          <Route path="/privacy" element={<PrivacyPolicy/>} />
        </Routes>
      </div>

      {/* 푸터 */}
      <footer className="footer">
        <div className="footer__nav">
          <a href="https://github.com/" className="footer__link" target="_blank" rel="noopener noreferrer">Team Github</a>
          <Link to="/contact" className="footer__link">Contact Us</Link>
          <Link to="/donation" className="footer__link">Donation</Link>
        </div>
        <div className="footer__legal">
          <p><Link to="/terms">Terms of Use</Link> | <Link to="/privacy">Privacy Policy</Link></p>
          <p>&copy; Copyright 2024. All Rights Reserved. Hosted by Pre-Vision</p>
        </div>
      </footer>
    </>
  );
}

export default App;
```

목적	App.js 파일은 네비게이션 바, 각 페이지의 라우팅, 그리고 푸터를 포함한 전체 웹사이트의 기본 레이아웃과 구조를 관리
파일명	App.js

Ver 1.0

```jsx
import React from 'react';
import { Link } from 'react-router-dom'; // React Router의 Link 컴포넌트 임포트
import '../css/Main.css';
import vstImg1 from '../images/virus-total_1.png'
import vstImg2 from '../images/virus-total_2.png';
import vstImg3 from '../images/virus-total_3.png';
import vstImg4 from '../images/virus-total_4.png';
import vstImg5 from '../images/virus-total_5.png';
import lockerImg from '../images/locker.jpg';
import malwarefoundImg from '../images/malwarefound.jpg';

function Main() {
    return (
        <>
            <div className="slider">
                <div className="slide">
                    <a href='https://www.virustotal.com/gui/home/upload' target="_blank" rel="noopener noreferrer">
                        <img src={vstImg1} alt="Image 1" />
                    </a>
                </div>
                <div className="slide">
                <a href='https://www.virustotal.com/gui/home/upload' target="_blank" rel="noopener noreferrer">
                        <img src={vstImg3} alt="Image 3" />
                    </a>
                </div>
                <div className="slide">
                <a href='https://www.virustotal.com/gui/home/upload' target="_blank" rel="noopener noreferrer">
                        <img src={vstImg4} alt="Image 4" />
                    </a>
                </div>
                <div className="slide">
                <a href='https://www.virustotal.com/gui/home/upload' target="_blank" rel="noopener noreferrer">
                        <img src={vstImg2} alt="Image 2" />
                    </a>
                </div>
                <div className="slide">
                <a href='https://www.virustotal.com/gui/home/upload' target="_blank" rel="noopener noreferrer">
                        <img src={vstImg5} alt="Image 5" />
                    </a>
                </div>
            </div>
```

```jsx
<div className="main-service">
    <h2 className="main-service__title">Main Service</h2>
    <h1 className="main-service__heading">바이러스를 분석하고 암호화하세요.</h1>
    <div className="main-service__content">
        <div className="main-service__box">
            <h3 className="main-service__subtitle">Virus Analyze</h3>
            <h2 className="main-service__text">실행파일에 숨어있는<br />바이러스 및 멀웨어를 분석</h2>
            <p>실행파일을 업로드하면 분석해서 파일구조를 보여드려요!</p>
            <br />
            <Link to="/virus-scan" className="main-service__a"><h2>Get Started!</h2></Link>
        </div>
        <div className="main-service__image">
            <img src={malwarefoundImg} alt="Service Image 1" />
        </div>
        <div className="main-service__image">
            <img src={lockerImg} alt="Service Image 2" />
        </div>
        <div className="main-service__box">
            <h3 className="main-service__subtitle">Encryption</h3>
            <h2 className="main-service__text">실행파일의 악의적인 변조를<br />막기위해 파일을 암호화</h2>
            <p>실행파일을 암호화해서 악의적인 변조를 막으세요!</p>
            <br />
            <Link to="/packing" className="main-service__a"><h2>Get Started!</h2></Link>
        </div>
    </div>
</div>
</>
);
}

export default Main;
```

목적	주요 서비스인 바이러스 분석 및 파일 암호화 기능을 소개하며, 사용자가 VirusTotal 웹사이트를 방문하여 바이러스 파일을 업로드하거나, 웹사이트 내에서 바이러스 스캔과 암호화 서비스를 시작할 수 있도록 유도
파일명	Main.js

```jsx
import React from 'react';
import axios from 'axios';
import '../css/Contact.css';
import { useState } from 'react';

function Contact() {
    const [formData, setFormData] = useState({
        name: '',
        email: '',
        message: ''
    });

    const handleChange = (e) => {
        setFormData({
            ...formData,
            [e.target.name]: e.target.value,
        });
    };

    const handleSubmit = async (e) => {
        e.preventDefault();
        try {
            // axios로 POST 요청을 보내기
            const response = await axios.post('/send-email', formData);
            if (response.data.success) {
                alert('문의 사항이 성공적으로 전송되었습니다.');
            } else {
                alert('문의 사항 전송에 실패했습니다.');
            }
        } catch (error) {
            console.error('Error:', error);
            alert('문의 사항 전송 중 오류가 발생했습니다.');
        }
    };

    return (
```

```jsx
    return (
        <div className="container">
            <div className="contact">
                <h1>Contact us</h1>
                <p>Contact the administrator if you have any questions.</p>
                <span>(It takes 1-2 business days.)</span>

                <form className="contact-form" onSubmit={handleSubmit}>
                    <label htmlFor="name">Name</label>
                    <input
                        type="text"
                        id="name"
                        name="name"
                        placeholder="이름을 입력하세요"
                        value={formData.name}
                        onChange={handleChange}
                        required
                    />

                    <label htmlFor="email">E-mail</label>
                    <input
                        type="email"
                        id="email"
                        name="email"
                        placeholder="이메일을 입력하세요"
                        value={formData.email}
                        onChange={handleChange}
                        required
                    />

                    <label htmlFor="message">Message</label>
                    <textarea
                        id="message"
                        name="message"
                        rows="6"
                        placeholder="문의 내용을 입력하세요"
                        value={formData.message}
                        onChange={handleChange}
                        required
                    ></textarea>

                    <button type="submit">Submit</button>
                </form>
            </div>
        </div>
    );
}

export default Contact;

```

목적	사용자가 이름, 이메일, 문의 내용을 입력하여 폼을 제출하면, 해당 데이터를 Axios를 통해

	Express 서버의 /send-email 경로로 POST 요청을 보내 이메일 문의를 처리함.
파일명	Contact.js

```jsx
import React, { useState } from 'react';
import DatePicker from 'react-datepicker';
import 'react-datepicker/dist/react-datepicker.css';
import '../css/Donation.css';

function Donation() {
    const [startDate, setStartDate] = useState(null);

    return (
        <div className="container">
            <div className="container__head">
                <h1>Donation</h1>
                <p>Your support helps create a better world.</p>
            </div>

            <div className="container__main">
                <section className="bank-details">
                    <h2>Donation account</h2>
                    <p className='bank-details__p'>Bank :
                        <span className='bank-details__p2'> KakaoBank</span>
                    </p>
                    <p className='bank-details__p'>Account number :
                        <span className='bank-details__p2'> 3333-03-9932704</span>
                    </p>
                    <p className='bank-details__p'>Account holder :
                        <span className='bank-details__p2'> Kimsunwoo</span>
                    </p>
                </section>

                <section className="donation-confirmation">
                    <h2>Donation confirmation</h2>
                    <form className="donation-form">
                        <label htmlFor="name">Name</label>
                        <input type="text" id="name" name="name" placeholder="Please enter your name" required />

                        <label htmlFor="email">E-mail</label>
                        <input type="email" id="email" name="email" placeholder="Please enter your email" required />

                        <label htmlFor="transfer-amount">Donation amount</label>
                        <input type="number"
                            id="transfer-amount"
                            name="transfer-amount"
                            placeholder="Please enter the donation amount"
                            step="10000"
                            required />

```

```
46
47                      <label htmlFor="transfer-date">Donation date</label>
48                      <DatePicker
49                          className='datePicker'
50                          selected={startDate}
51                          onChange={(date) => setStartDate(date)}
52                          dateFormat="yyyy-MM-dd" // 날짜 형식을 영어로 표시
53                          placeholderText="Please enter the donation date"
54                          required
55                      />
56
57                      <button type="submit">submit</button>
58                  </form>
59              </section>
60          </div>
61
62          <div className="container__foot">
63              <br />
64              <p>&copy; 2024 Pre-Vision</p>
65          </div>
66      </div>
67  );
68  }
69
70  export default Donation;
71
```

목적	사용자는 이름, 이메일, 기부 금액, 그리고 기부 날짜를 입력한 후, 제출 버튼을 통해 기부 확인을 완료할 수 있다.
파일명	Donation.js

```jsx
import React, { useState } from 'react';
import '../css/Packing.css';
import axios from 'axios';
import PackPopup from './PackPopup.js';

function Packing() {
    const [files, setFiles] = useState([]); // 업로드한 파일 리스트
    const [uploadStatus, setUploadStatus] = useState(''); // 업로드 상태 메시지 관리
    const [isUploading, setIsUploading] = useState(false); // 업로드 중 여부 상태 관리
    const [isPopupVisible, setIsPopupVisible] = useState(false); // 팝업창 표시 여부 상태 관리
    const [popupData, setPopupData] = useState([]); // 팝업에 표시할 데이터 상태 추가
    const [tooltipVisible, setTooltipVisible] = useState(false); // 툴팁 표시 여부 상태
    const [tooltipPosition, setTooltipPosition] = useState({ x: 0, y: 0 }); // 툴팁 위치

    // 파일 선택 시 호출되는 함수
    const handleFileSelect = (e) => {
        const selectedFiles = e.target.files;
        handleFiles(selectedFiles);
    };

    // 파일 드래그 앤 드롭 시 호출되는 함수
    const handleDrop = (e) => {
        e.preventDefault();
        const droppedFiles = e.dataTransfer.files;
        handleFiles(droppedFiles);
    };

    // 드래그 중인 상태에서의 처리
    const handleDragOver = (e) => {
        e.preventDefault();
    };

    // 드래그 앤 드롭 존에 마우스를 올릴 때 툴팁 표시
    const handleMouseOver = (e) => {
        setTooltipVisible(true);
        setTooltipPosition({ x: e.clientX, y: e.clientY });
    };

    // 드래그 앤 드롭 존에서 마우스가 벗어날 때 툴팁 숨김
    const handleMouseOut = () => {
        setTooltipVisible(false);
    };

    // 드래그 앤 드롭 존에서 마우스가 이동할 때 툴팁 위치 갱신
    const handleMouseMove = (e) => {
        setTooltipPosition({ x: e.clientX, y: e.clientY });
    };

```

```
49      // 파일 크기 및 확장자 제한을 처리하는 함수임
50      const handleFiles = (fileList) => {
51          const maxFileSize = 300 * 1024 * 1024; // 300MB로 제한
52          const allowedExtensions = ['exe', 'dll'];
53
54          // 파일을 하나만 허용함
55          const file = fileList[0];
56          const fileExtension = file.name.split('.').pop().toLowerCase();
57
58          if (file.size > maxFileSize) {
59              alert('파일은 300MB를 초과합니다.');
60              return;
61          }
62
63          if (!allowedExtensions.includes(fileExtension)) {
64              alert('허용되지 않는 파일 형식입니다. exe 또는 dll 파일만 업로드할 수 있습니다.');
65              return;
66          }
67
68          setFiles([file]); // 파일을 상태에 저장
69          setTooltipVisible(false); // 파일이 선택되면 툴팁을 숨김
70      };
71
72      // 파일을 서버에 업로드하는 함수
73      const uploadFileToServer = async () => {
74          if (files.length === 0) {
75              alert('업로드할 파일이 없습니다.');
76              return;
77          }
78
79          const formData = new FormData();
80          formData.append('file', files[0]); // 첫 번째 파일만 업로드
81
82          try {
83              setIsUploading(true); // 업로드 시작 시 로딩 상태 활성화
84              setUploadStatus(''); // 상태 초기화
85
86              // 파일을 Express 서버로 POST 요청
87              const response = await axios.post('/upload-file-pack', formData, {
88                  headers: {
89                      'Content-Type': 'multipart/form-data',
90                  },
91              });
92              // 팝업에서 pe_info 데이터와 함께 표시
93              setPopupData(response.data.peInfo);
94              setUploadStatus(`파일이 성공적으로 업로드되었습니다: ${response.data.filename}`);
95          } catch (error) {
96              setUploadStatus('파일 업로드 중 오류가 발생했습니다.');
97              console.error('Error uploading file:', error);
98          } finally {
99              setIsUploading(false); // 업로드가 완료되면 로딩 상태 비활성화
100         }
101     };
102
```

```
102
103     return (
104         <div className="packing-container">
105             <h1>🔒</h1>
106             <h2>안전하게 파일을 암호화하여 데이터를 보호하세요!</h2>
107
108             {/* 파일이 업로드되었는지 확인 후 업로드된 파일이 없을 때만 파일 선택 창을 보여줌 */}
109             {files.length === 0 ? (
110                 <div
111                     id="drop-zone"
112                     className="drop-zone"
113                     onDragOver={handleDragOver}
114                     onDrop={handleDrop}
115                     onClick={() => document.getElementById('file-input').click()}
116                     onMouseOver={handleMouseOver}
117                     onMouseOut={handleMouseOut}
118                     onMouseMove={handleMouseMove}
119                 >
120                     <p>Drag & Drop your files here or click to select files</p>
121                     <input
122                         type="file"
123                         id="file-input"
124                         style={{ display: 'none' }}
125                         onChange={handleFileSelect}
126                     />
127                 </div>
128             ) : (
129                 <button className='uploadBtn' onClick={uploadFileToServer} disabled={isUploading}>
130                     Get Started!
131                 </button>
132             )}
133
134             {/* 업로드 중일 때 로딩 UI 표시 */}
135             {isUploading && (
136                 <div className="loading-spinner">
137                     <p>업로드 및 스캔 중... 잠시만 기다려 주세요.</p>
138                     <div className="spinner"></div>
139                 </div>
140             )}
141
142             <ul id="file-list">
143                 {files.map((file, index) => {
144                     // 파일 이름이 60글자를 넘으면 잘라내고 '...'을 추가
145                     const shortenedName = file.name.length > 60 ? file.name.slice(0, 60) + '...' : file.name;
146                     return <p key={index}>{`${shortenedName} (${(file.size / (1024 * 1024)).toFixed(2)} MB)`}</p>;
147                 })}
148             </ul>
149
```

```
            {/* 업로드 상태 메시지 */}
            {uploadStatus && <p>{uploadStatus}</p>}

            {/* 팝업을 띄우는 버튼 추가 */}
            {uploadStatus && (
                <button onClick={() => setIsPopupVisible(true)} className="open-popup-btn">
                    파일 구조 보기
                </button>
            )}

            {/* 팝업 컴포넌트 */}
            <PackPopup isVisible={isPopupVisible} onClose={() => setIsPopupVisible(false)} data={popupData} />

            {/* 툴팁 추가 */}
            {tooltipVisible && (
                <div
                    style={{
                        position: 'fixed',
                        top: tooltipPosition.y + 15,
                        left: tooltipPosition.x + 15,
                        backgroundColor: '#333',
                        color: 'white',
                        padding: '5px',
                        borderRadius: '5px',
                        fontSize: '12px',
                        zIndex: 1000,
                    }}
                >
                    Max size: 300MB
                </div>
            )}
        </div>
    );
}

export default Packing;

```

목적	.exe 또는 .dll 파일을 업로드하고, 서버로 전송하여 파일을 암호화, 업로드된 파일은 서버에서 처리된 후, 파일 정보와 함께 사용자가 해당 파일 구조를 팝업에서 볼 수 있도록 지원
파일명	Packing.js

```jsx
import React from 'react';
import '../css/PackPopup.css';
import axios from 'axios';
import exeImg from '../images/exe.png';
import lockImg from '../images/lock.png';
import arrowGif from '../images/arrow.gif';
import { useState } from 'react';

// 팝업 컴포넌트 정의
function PackPopup({ isVisible, onClose, data }) {
    const [isChildPopupVisible, setIsChildPopupVisible] = useState(false); // child 팝업창을 만들기 위한 state

    // 팝업이 보이지 않으면 아무것도 렌더링하지 않음
    if (!isVisible) {
        return null;
    }

    const handleDownload = async (fileId, filename) => {
        try {
            // axios를 사용하여 파일 다운로드 요청
            const response = await axios.get(`/download-file/${fileId}`, {
                responseType: 'blob', // 서버로부터 binary 파일 데이터를 받을 수 있게 설정
            });

            const blob = new Blob([response.data], { type: 'application/octet-stream' }); // 받은 데이터를 Blob으로 변환
            const link = document.createElement('a');
            const url = window.URL.createObjectURL(blob);

            link.href = url;
            link.download = `Protected${filename}`;
            link.click();

            // 다운로드 후 URL 객체 해제
            window.URL.revokeObjectURL(url);
        } catch (error) {
            console.error('Error downloading the file:', error);
        }
    };

    const ShowChildPopup = () => {
        setIsChildPopupVisible(true); // 버튼을 누르면 내부 팝업 상태를 true로 변경
    };

    const CloseChildPopup = () => {
        setIsChildPopupVisible(false); // 내부 팝업을 닫기 위한 함수
    };

    function renderJson(obj) {
        return (
            <ul className="json">
                {Object.entries(obj).map(([key, value], index) => (
                    <li key={index}>
                        <span className="json-key">"{key}"</span>:{" "}
                        <span className="json-value">
                            {typeof value === 'object' && value !== null ? renderJson(value) : JSON.stringify(value)}
                        </span>
                    </li>
                ))}
            </ul>
        );
    }

```

```
63      return (
64          <>
65              {/* 오버레이 추가: 배경을 검게 만들고 클릭 불가하게 */}
66              <div className={`popup-overlay ${isVisible ? 'show' : ''}`}></div>
67              {/* 팝업 오버레이 및 내용 */}
68              <div className={`popup-container ${isVisible ? 'show' : ''}`}>
69                  <div className="popup-header">
70                      <h2>File Information</h2>
71                      <button className="popup-close" onClick={onClose}>✕</button>
72                  </div>
73                  <div className="info-data">
74                      <div className="file-section-container">
75                          <div>
76                              <h3>Normal Files</h3>
77                              <img className='exeImg' src={exeImg} alt="exe" />
78                          </div>
79
80                          {/* 가운데에 화살표 이미지 추가 */}
81                          <div className="arrow-section">
82                              <img className="arrowImg" src={arrowGif} alt="arrow" />
83                          </div>
84
85                          <div>
86                              <h3>Encrypted Files</h3>
87                              <img className='lockImg' src={lockImg} alt="lock" />
88                          </div>
89                      </div>
90                      <br />
91                      <br />
92                      <div className="download-section">
93                          {/* 다운로드 버튼 추가 */}
94                          {data.encryptedFilesData.length > 0 ? (
95                              data.encryptedFilesData.map((file, index) => (
96                                  <div key={index}>
97                                      <p>Protected{file.original_filename}</p>
98                                      <br />
99                                      <button className='showPopupBtn' onClick={ShowChildPopup}>Protected File Info</button>
100                                     <button className='downloadBtn'
101                                         onClick={() => handleDownload(file.gridfs_file_id, file.original_filename)}>
102                                         Download Protected File
103                                     </button>
104                                 </div>
105                             ))
106                         ) : (
107                             <p>No matching encrypted file data found</p>
108                         )}
109                     </div>
110                 </div>
111             </div>
112             {/* 내부 팝업 */}
113             {isChildPopupVisible && (
114                 <div className={`child-popup-overlay ${isChildPopupVisible ? 'show' : ''}`}>
115                     <div className={`child-popup-container ${isChildPopupVisible ? 'show' : ''}`}>
116                         <div className="child-popup-header">
117                             <h2>Protected File Info</h2>
118                             <button className="popup-close" onClick={CloseChildPopup}>✕</button>
119                         </div>
120                         <div className="child-info-data">
121                             {data.encryptedFileInfo.length > 0 ? (
122                                 data.encryptedFileInfo.map((item, index) => (
123                                     <div key={index}>{renderJson(item)}</div>
124                                 ))
125                             ) : (
126                                 <p>No encrypted file info available</p>
127                             )}
128                         </div>
129                     </div>
130                 </div>
131             )}
132         </>
133     );
134 }
135
136 export default PackPopup;
137
```

목적	파일 정보는 일반 파일과 암호화된 파일로 나뉘어 시각적으로 구분되며, 파일 다운로드 기능을 제공하고 사용자는 파일 보호 정보 팝업

	을 열어 상세 데이터를 확인할 수 있다.
파일명	PackPopup.js

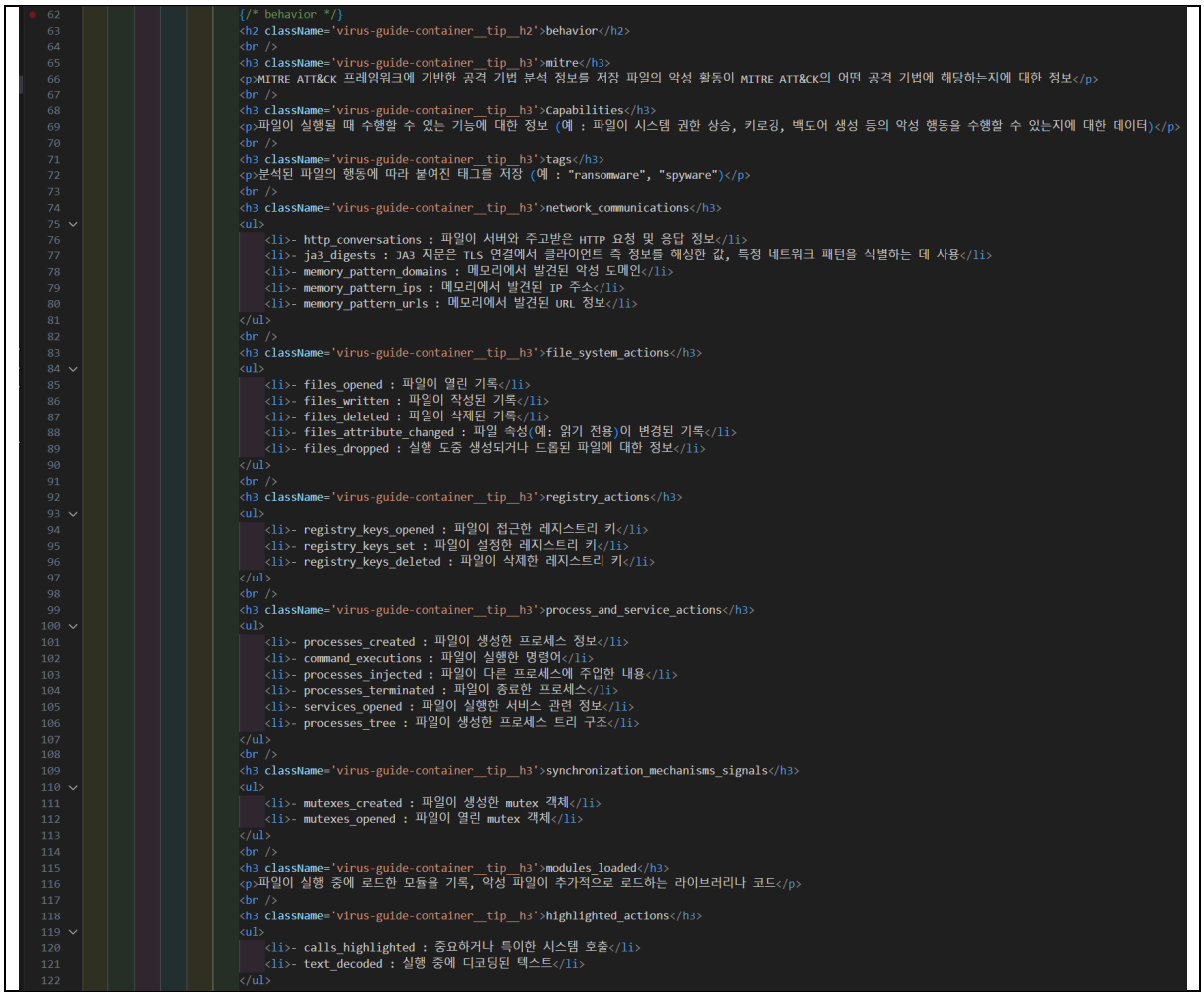
```jsx
import React from 'react';
import '../css/PrivacyPolicy.css';

function PrivacyPolicy() {
    return (
        <div className='policy-container'>
            <h2>개인정보 처리방침</h2>
            <br />
            <br />
            <h3>1. 수집하는 개인정보 항목</h3>
            <ul>
                <li>본 서비스는 다음과 같은 개인정보를 수집합니다.</li>
                <li> - 이름</li>
                <li> - 이메일 주소</li>
                <li> - 업로드한 파일</li>
                <li> - IP 주소</li>
            </ul>
            <br />
            <h3>2. 개인정보의 수집 및 이용 목적</h3>
            <ul>
                <li>IP 주소 수집 : 악성 스크립트가 포함된 파일이 업로드되어 공격을 받는 경우, 해당 사용자가 어떤 IP에서 파일을 업로드했는지 추적하기 위해 수집합니다.</li>
                <li>이메일 주소 수집 : 사용자가 문의를 했을 때 해당 이메일 주소로 회신하기 위해 수집합니다.</li>
            </ul>
            <br />
            <h3>3. 개인정보의 보유 및 이용 기간</h3>
            <p>사용자가 서비스를 이용한 후에는 수집된 개인정보를 자동으로 즉시 파기합니다.</p>
            <br />
            <h3>4. 개인정보의 안정성 확보 조치</h3>
            <p>
                본 서비스는 사용자의 개인정보 보호를 위해 SSL/TLS 인증서를 적용하여 데이터를 암호화하고 안전하게 전송합니다.
                추가적으로 방화벽, 접근 권한 제어, 및 데이터 접근 로그 관리를 통해 개인정보의 안전성을 확보하고 있습니다.
            </p>
            <br />
            <h3>5. 개인정보 처리방침의 변경</h3>
            <p>본 서비스와 관련된 모든 콘텐츠와 소프트웨어에 대한 지적 재산권은 회사에 귀속되며, 사용자는 이를 허가 없이 복제, 배포, 수정할 수 없습니다.</p>
        </div>
    );
}

export default PrivacyPolicy;
```

목적	서비스에서 수집하는 개인정보 항목, 이용 목적, 보유 및 이용 기간, 그리고 개인정보 보호 조치 등을 명시하는 개인정보 처리방침 페이지를 구성
파일명	PrivacyPolicy.js

```jsx
import React from 'react';
import '../css/TermsOfUse.css';

function TermsOfUse() {
    return (
        <div className='terms-container'>
            <h2>이용약관</h2>
            <br />
            <h3>1. 서비스 개요</h3>
            <p>
                본 서비스는 사용자가 업로드한 실행 파일의 구조를 분석하고, 악성코드 여부를 탐지하여 사용자에게 정보를 제공하는 도구입니다.
                이를 통해 사용자는 자신의 파일에 존재할 수 있는 잠재적인 보안 위협을 확인할 수 있습니다.
                </p>
            <br />
            <h3>2. 사용자 의무</h3>
            <p>사용자는 본 서비스를 이용함에 있어 다음의 사항을 준수해야 합니다:</p>
            <ul>
                <li>타인의 권리(지적 재산권 포함)를 침해하지 않아야 합니다.</li>
                <li>법령에 위배되는 목적으로 서비스를 이용하지 않아야 합니다.</li>
            </ul>
            <br />
            <h3>3. 제한 사항</h3>
            <ul>
                <li>본 서비스는 악성코드를 탐지하기 위해 제공되는 도구이며, 악의적인 파일이나 악성 스크립트를 주입하는 행위는 엄격히 금지됩니다.</li>
                <li>본 서비스를 통해 악성 파일을 고의적으로 암호화하여 악용하는 행위는 금지됩니다.</li>
                <li>서비스는 정상적인 보안 목적 외의 어떠한 악의적인 목적에도 사용될 수 없습니다.</li>
            </ul>
            <br />
            <h3>4. 개인정보 보호</h3>
            <p>
                본 서비스는 사용자가 업로드한 파일의 메타 데이터를 수집할 수 있으며, 이러한 데이터는 서비스 제공 및 보안 연구 목적으로만 사용됩니다.
                개인정보에 대한 자세한 사항은 별도의 개인정보 처리방침을 참조하십시오.
                </p>
            <br />
            <h3>5. 지적 재산권</h3>
            <p>본 서비스와 관련된 모든 콘텐츠와 소프트웨어에 대한 지적 재산권은 회사에 귀속되며, 사용자는 이를 허가 없이 복제, 배포, 수정할 수 없습니다.</p>
            <br />
            <h3>6. 면책 조항</h3>
            <p>
                본 서비스는 사용자의 편의를 위해 제공되는 정보형 도구이며, 이를 악용하여 발생하는 모든 문제에 대해서 자사는 어떠한 책임도 지지 않습니다.
                모든 책임은 사용자 본인의 행동에 의해 발생하며, 사용자는 본인의 행동에 대한 법적 책임을 지게 됩니다.
                </p>
            <br />
            <h3>7. 분쟁 해결</h3>
            <p>본 약관과 관련하여 발생하는 모든 분쟁은 대한민국 법률에 따르며, 본 서비스의 제공자가 소재한 관할 법원의 전속 관할로 합니다.</p>
            <br />
        </div>
    );
}

export default TermsOfUse;

```

목적	서비스의 이용약관을 명시하고 사용자가 서비스를 이용할 때 따라야 할 규칙과 의무를 설명
파일명	TermsOfUse.js

```jsx
import React from 'react';
import '../css/UserGuide.css';
import virusScanGuide from '../images/virus-scan-guide.gif';
import packingGuide1 from '../images/packing-guide_1.gif';
import packingGuide2 from '../images/packing-guide_2.gif';
import packingGuide3 from '../images/packing-guide_3.gif';

function UserGuide() {
    return (
        <div>
            {/* UserGuide */}
            <div className="guide-container">
                <p className='guide-header'>사용자를 위한 시작 가이드👻</p>
                <div className="virus-guide-container">
                    <h1>Virus Scan👾</h1>
                    <br />
                    <br />
                    <h2>1. 파일을 업로드하고 구조를 확인해보세요.</h2>
                    <br />
                    <div className='guide-gif'>
                        <img src={virusScanGuide} alt='virus-scan-guide' />
                    </div>
                    <br />
                    <h2>💡사용자를 위한 데이터 구조 확인하기 Tip</h2>
                    <br />
                    <div className='virus-guide-container__tip'>
                        <h2 className='virus-guide-container__tip__h2'>Details</h2>
                        <br />
                        <h3 className='virus-guide-container__tip__h3'>Hash</h3>
                        <ul>
                            <li>- md5 : 파일의 MD5 해시값</li>
                            <li>- sha1 : 파일의 SHA1 해시값</li>
                            <li>- sha256 : 파일의 SHA256 해시값</li>
                            <li>- vhash : 파일의 VHash 해시값</li>
                            <li>- auth_hash : 인증 해시값</li>
                            <li>- imphash : Import Hash 해시값</li>
                            <li>- ssdeep : SSDEEP 해시값</li>
                            <li>- tlsh : TLSH 해시값</li>
                        </ul>
                        <br />
                        <h3 className='virus-guide-container__tip__h3'>file_info</h3>
                        <ul>
                            <li>- md5 : 파일의 MD5 해시값</li>
                            <li>- file_type : 파일의 유형</li>
                            <li>- magic : 파일의 매직 넘버 또는 식별자</li>
                            <li>- file_size : 파일 크기</li>
                            <li>- PEID_packer : 파일에 사용된 패커 정보</li>
                            <li>- first_seen_time : 파일이 처음 발견된 시간</li>
                            <li>- name : 파일의 이름</li>
                        </ul>
                        <br />
                        <h3 className='virus-guide-container__tip__h3'>signature</h3>
                        <p>파일 서명 데이터를 저장, 파일이 디지털 서명되어 있거나 인증된 경우 그 정보를 저장</p>
                        <br />
                        <h3 className='virus-guide-container__tip__h3'>pe_info</h3>
                        <p>PE(Portable Executable)파일에 대한 정보 PE 파일은 주로 Windows 운영 체제에서 실행되는 파일, PE 구조와 관련된 추가 정보</p>
                        <br />
                        <h3 className='virus-guide-container__tip__h3'>dot_net_assembly</h3>
                        <p>.NET 어셈블리 파일에 대한 정보를 저장 .NET 파일이 실행되는 동안 사용되는 메타데이터 및 코드 모듈 정보</p>
                        <br />
```



```
{/* behavior */}
<h2 className='virus-guide-container__tip__h2'>behavior</h2>
<br />
<h3 className='virus-guide-container__tip__h3'>mitre</h3>
<p>MITRE ATT&CK 프레임워크에 기반한 공격 기법 분석 정보를 저장 파일의 악성 활동이 MITRE ATT&CK의 어떤 공격 기법에 해당하는지에 대한 정보</p>
<br />
<h3 className='virus-guide-container__tip__h3'>Capabilities</h3>
<p>파일이 실행될 때 수행할 수 있는 기능에 대한 정보 (예 : 파일이 시스템 권한 상승, 키로깅, 백도어 생성 등의 악성 행동을 수행할 수 있는지에 대한 데이터)</p>
<br />
<h3 className='virus-guide-container__tip__h3'>tags</h3>
<p>분석된 파일의 행동에 따라 붙여진 태그를 저장 (예 : "ransomware", "spyware")</p>
<br />
<h3 className='virus-guide-container__tip__h3'>network_communications</h3>
<ul>
    <li>- http_conversations : 파일이 서버와 주고받은 HTTP 요청 및 응답 정보</li>
    <li>- ja3_digests : JA3 지문은 TLS 연결에서 클라이언트 측 정보를 해싱한 값, 특정 네트워크 패턴을 식별하는 데 사용</li>
    <li>- memory_pattern_domains : 메모리에서 발견된 악성 도메인</li>
    <li>- memory_pattern_ips : 메모리에서 발견된 IP 주소</li>
    <li>- memory_pattern_urls : 메모리에서 발견된 URL 정보</li>
</ul>
<br />
<h3 className='virus-guide-container__tip__h3'>file_system_actions</h3>
<ul>
    <li>- files_opened : 파일이 열린 기록</li>
    <li>- files_written : 파일이 작성된 기록</li>
    <li>- files_deleted : 파일이 삭제된 기록</li>
    <li>- files_attribute_changed : 파일 속성(예: 읽기 전용)이 변경된 기록</li>
    <li>- files_dropped : 실행 도중 생성되거나 드롭된 파일에 대한 정보</li>
</ul>
<br />
<h3 className='virus-guide-container__tip__h3'>registry_actions</h3>
<ul>
    <li>- registry_keys_opened : 파일이 접근한 레지스트리 키</li>
    <li>- registry_keys_set : 파일이 설정한 레지스트리 키</li>
    <li>- registry_keys_deleted : 파일이 삭제한 레지스트리 키</li>
</ul>
<br />
<h3 className='virus-guide-container__tip__h3'>process_and_service_actions</h3>
<ul>
    <li>- processes_created : 파일이 생성한 프로세스 정보</li>
    <li>- command_executions : 파일이 실행한 명령어</li>
    <li>- processes_injected : 파일이 다른 프로세스에 주입한 내용</li>
    <li>- processes_terminated : 파일이 종료한 프로세스</li>
    <li>- services_opened : 파일이 실행한 서비스 관련 정보</li>
    <li>- processes_tree : 파일이 생성한 프로세스 트리 구조</li>
</ul>
<br />
<h3 className='virus-guide-container__tip__h3'>synchronization_mechanisms_signals</h3>
<ul>
    <li>- mutexes_created : 파일이 생성한 mutex 객체</li>
    <li>- mutexes_opened : 파일이 열린 mutex 객체</li>
</ul>
<br />
<h3 className='virus-guide-container__tip__h3'>modules_loaded</h3>
<p>파일이 실행 중에 로드한 모듈을 기록, 악성 파일이 추가적으로 로드하는 라이브러리나 코드</p>
<br />
<h3 className='virus-guide-container__tip__h3'>highlighted_actions</h3>
<ul>
    <li>- calls_highlighted : 중요하거나 특이한 시스템 호출</li>
    <li>- text_decoded : 실행 중에 디코딩된 텍스트</li>
</ul>
```

```
                    <br />
                    <h3 className='virus-guide-container__tip__h3'>system_property_lookups</h3>
                    <p>파일이 조회한 시스템 속성 정보(예 :  파일이 시스템 버전, 사용자 정보, 설치된 소프트웨어 정보를 확인하려는 시도 등)</p>
                    <br />
                </div>
            </div>
            <div className="packing-guide-container">
                <h1>Packing🔒</h1>
                <br />
                <br />
                <h2>1. 보호하고 싶은 파일을 업로드해주세요.</h2>
                <br />
                <div className='guide-gif'>
                    <img src={packingGuide1} alt='packing-guide1' />
                </div>
                <br />
                <h2>2. 파일 보호화가 진행되는 동안 기다려주세요.</h2>
                <br />
                <div className='guide-gif'>
                    <span className='packing-guide-container__forward'>⏩6X</span>
                    <img src={packingGuide2} alt='packing-guide2' />
                </div>
                <br />
                <h2>3. 보호된 파일을 다운받아 안전하게 사용하세요.</h2>
                <br />
                <div className='guide-gif'>
                    <img src={packingGuide3} alt='packing-guide3' />
                </div>
                <br />
                <h2>💡사용자를 위한 데이터 구조 확인하기 Tip</h2>
                <br />
                <div className='packing-guide-container__tip'>
                    <h2 className='packing-guide-container__tip__h2'>Encrypted_filename</h2>
                    <p>암호화된 파일의 파일 경로 (예:
                        "20240909-002_protected.exe")</p>
                    <br />
                    <h2 className='packing-guide-container__tip__h2'>encrypted_upload_time</h2>
                    <p>암호화된 파일의 업로드 시간</p>
                    <br />
                    <h2 className='packing-guide-container__tip__h2'>upload_ip</h2>
                    <p>파일을 업로드한 IP 주소</p>
                    <br />
                    <h2 className='packing-guide-container__tip__h2'>pe_info </h2>
                    <p>암호화된 파일의 PE 분석 정보가 포함된 객체</p>
                    <br />
                    <h2 className='packing-guide-container__tip__h2'>encrypted</h2>
                    <p>암호화된 섹션 정보가 포함된 객체</p>
                    <br />
                </div>
            </div>
        </div>
    );
}

export default UserGuide;

```

목적	사용자가 서비스의 주요 기능인 **바이러스 스캔**과 **파일 암호화**를 쉽게 이용할 수 있도록 단계별 가이드를 제공. 각 서비스에 대한 사용 절차를 설명하는 애니메이션 GIF를 포함하여, 사용자가 파일 업로드, 분석, 암호화 및 다운로드 과정을 시각적으로 이해할 수 있게 도와줌. 또한, 파일의 해시값 및 파일 시스템 동작 등과 같은 상세 데이터 구조 설명을 제공하여, 사용자가 서비스 결과를 더 잘 이해할 수 있도록 도움.
파일명	UserGuide.js

```jsx
import React, { useState } from 'react';
import '../css/VirusScan.css';
import axios from 'axios';
import ScanPopup from './ScanPopup.js';

function VirusScan() {
    const [files, setFiles] = useState([]);
    const [uploadStatus, setUploadStatus] = useState(''); // 업로드 상태 메시지 관리
    const [isUploading, setIsUploading] = useState(false); // 업로드 중 여부 상태
    const [isPopupVisible, setIsPopupVisible] = useState(false); // 팝업창 표시 여부 상태
    const [popupData, setPopupData] = useState([]); // 팝업에 표시할 데이터 상태 추가
    const [tooltipVisible, setTooltipVisible] = useState(false); // 툴팁 표시 여부
    const [tooltipPosition, setTooltipPosition] = useState({ x: 0, y: 0 }); // 툴팁 위치

    // 파일 선택 시 호출되는 함수
    const handleFileSelect = (e) => {
        const selectedFiles = e.target.files;
        handleFiles(selectedFiles);
    };

    // 파일 드래그 앤 드롭 시 호출되는 함수
    const handleDrop = (e) => {
        e.preventDefault();
        const droppedFiles = e.dataTransfer.files;
        handleFiles(droppedFiles);
    };

    // 드래그 중인 상태에서의 처리
    const handleDragOver = (e) => {
        e.preventDefault();
    };

    // 드래그 앤 드롭 존에 마우스를 올릴 때 툴팁 표시
    const handleMouseOver = (e) => {
        setTooltipVisible(true);
        setTooltipPosition({ x: e.clientX, y: e.clientY });
    };

    // 드래그 앤 드롭 존에서 마우스가 벗어날 때 툴팁 숨김
    const handleMouseOut = () => {
        setTooltipVisible(false);
    };

    // 드래그 앤 드롭 존에서 마우스가 이동할 때 툴팁 위치 갱신
    const handleMouseMove = (e) => {
        setTooltipPosition({ x: e.clientX, y: e.clientY });
    };

```

```
// 파일 크기 및 확장자 제한을 처리하는 함수임
const handleFiles = (fileList) => {
    const maxFileSize = 500 * 1024 * 1024; // 500MB로 제한
    const allowedExtensions = ['exe', 'dll'];

    // 파일을 하나만 허용함
    const file = fileList[0];
    const fileExtension = file.name.split('.').pop().toLowerCase();

    if (file.size > maxFileSize) {
        alert('파일은 500MB를 초과합니다.');
        return;
    }

    if (!allowedExtensions.includes(fileExtension)) {
        alert('허용되지 않는 파일 형식입니다. exe 또는 dll 파일만 업로드할 수 있습니다.');
        return;
    }

    setFiles([file]); // 파일을 상태에 저장
    setTooltipVisible(false); // 파일이 선택되면 툴팁을 숨김
};

// 파일을 서버에 업로드하는 함수
const uploadFileToServer = async () => {
    if (files.length === 0) {
        alert('업로드할 파일이 없습니다.');
        return;
    }

    const formData = new FormData();
    formData.append('file', files[0]); // 첫 번째 파일만 업로드

    try {
        setIsUploading(true); // 업로드 시작 시 로딩 상태 활성화
        setUploadStatus(''); // 상태 초기화

        // 파일을 Express 서버로 POST 요청
        const response = await axios.post('/upload-file-scan', formData, {
            headers: {
                'Content-Type': 'multipart/form-data',
            },
        });
        setUploadStatus(`파일이 스캔되었습니다: ${response.data.filename}`);
        // 파일 업로드 성공 후 일치하는 문서를 팝업창에 표시
        setPopupData(response.data.infoDocuments); // 팝업에 표시할 데이터를 상태에 저장

    } catch (error) {
        setUploadStatus('파일 업로드 중 오류가 발생했습니다.');
        console.error('Error uploading file:', error);
    } finally {
        setIsUploading(false); // 업로드가 완료되면 로딩 상태 비활성화
    }
};

```

```
    return (
        <div className="virus-container">
            <h1>👾</h1>
            <h2>파일을 분석하여 즉시 악성코드를 찾아보세요!</h2>

            {/* 파일이 업로드되었는지 확인 후 업로드된 파일이 없을 때만 파일 선택 창을 보여줌 */}
            {files.length === 0 ? (
                <div
                    id="drop-zone"
                    className="drop-zone"
                    onDragOver={handleDragOver} // 드래그 중 이벤트 처리
                    onDrop={handleDrop} // 드래그 앤 드롭 파일 처리
                    onClick={() => document.getElementById('file-input').click()} // 클릭 시 파일 선택 창 열기
                    onMouseOver={handleMouseOver}
                    onMouseOut={handleMouseOut}
                    onMouseMove={handleMouseMove}
                >
                    <p>Drag & Drop your files here or click to select files</p>
                    <input
                        type="file"
                        id="file-input"
                        style={{ display: 'none' }} // 숨겨진 파일 입력
                        onChange={handleFileSelect} // 파일 선택 처리
                    />
                </div>
            ) : (
                <button className='uploadBtn' onClick={uploadFileToServer} disabled={isUploading}>
                    Get Started!
                </button>
            )}

            {/* 업로드 중일 때 로딩 UI 표시 */}
            {isUploading && (
                <div className="loading-spinner">
                    <p>업로드 및 스캔 중... 잠시만 기다려 주세요.</p>
                    <div className="spinner"></div>
                </div>
            )}

            {/* 업로드된 파일 리스트 */}
            <ul id="file">
                {files.map((file, index) => {
                    // 파일 이름이 60글자를 넘으면 잘라내고 '...'을 추가
                    const shortenedName = file.name.length > 60 ? file.name.slice(0, 60) + '...' : file.name;
                    return <p key={index}>{`${shortenedName} (${(file.size / (1024 * 1024)).toFixed(2)} MB)`}</p>;
                })}
            </ul>

            {/* 업로드 상태 메시지 */}
            {uploadStatus && <p>{uploadStatus}</p>}

            {/* 팝업을 띄우는 버튼 추가 */}
            {uploadStatus && (
                <button onClick={() => setIsPopupVisible(true)} className="open-popup-btn">
                    파일 구조 보기
                </button>
            )}

            {/* 팝업 컴포넌트 */}
            <ScanPopup isVisible={isPopupVisible} onClose={() => setIsPopupVisible(false)} data={popupData} />

            {/* 툴팁 추가 */}
            {tooltipVisible && (
                <div
                    style={{
                        position: 'fixed',
                        top: tooltipPosition.y + 15,
                        left: tooltipPosition.x + 15,
                        backgroundColor: '#333',
                        color: 'white',
                        padding: '5px',
                        borderRadius: '5px',
                        fontSize: '12px',
                        zIndex: 1000,
                    }}
                >
                    Max size: 500MB
                </div>
            )}
        </div>
    );
}

export default VirusScan;
```

목적	`.exe` 또는 `.dll` 파일을 업로드하여 서버에서 악성코드를 분석하고, 해당 파일의 구조 정보를 반환받아 확인.
파일명	VirusScan.js

```jsx
import React, { useEffect } from 'react';
import '../css/ScanPopup.css';

// 팝업 컴포넌트 정의
function ScanPopup({ isVisible, onClose, data }) {

    // 팝업이 보이지 않으면 아무것도 렌더링하지 않음
    if (!isVisible) {
        return null;
    }

    function renderJson(obj) {
        return (
            <ul className="json">
                {Object.entries(obj).map(([key, value], index) => (
                    <li key={index}>
                        <span className="json-key">"{key}"</span>:{" "}
                        <span className="json-value">
                            {typeof value === 'object' && value !== null ? renderJson(value) : JSON.stringify(value)}
                        </span>
                    </li>
                ))}
            </ul>
        );
    }
    return (
        <>
            {/* 오버레이 추가: 배경을 검게 만들고 클릭 불가하게 */}
            <div className={`popup-overlay ${isVisible ? 'show' : ''}`}></div>
            {/* 팝업 오버레이 및 내용 */}
            <div className={`popup-container ${isVisible ? 'show' : ''}`}>
                <div className="popup-header">
                    <h2>File Details</h2>
                    <button className="popup-close" onClick={onClose}>✕</button>
                </div>
                <div className="info-data">
                    {/* 데이터가 있을 경우 출력 */}
                    {data.length > 0 ? (
                        data.map((item, index) => (
                            <div key={index}>{renderJson(item)}</div>
                        ))
                    ) : (
                        <p>No matching data found</p> // 데이터가 없을 때 메시지 출력
                    )}
                </div>
            </div>
        </>
    );
}

export default ScanPopup; // Popup 컴포넌트를 외부에서 사용 가능하도록 export

```

목적	서버에서 반환된 파일의 구조 데이터를 JSON 형식으로 렌더링하여 파일 세부 정보를 제공하며, 데이터가 있을 경우 이를 트리 형태로 출력
파일명	ScanPopup.js