```
// App.css
import React, { useState, useRef } from 'react'; // Add 'useRef' import
import './App.css';
import Modal from './Modal';
import SaveButton from './SaveButton';
import LoadButton from './LoadButton';
import PrintButton from './PrintButton';
import questions from './questions';
import FormattedAnswer from './FormattedAnswer';
import 'react-quill/dist/quill.snow.css';

function App() {
  const [answers, setAnswers] = useState({});
  const [image, setImage] = useState(null);
  const [currentModalQuestionIndex, setCurrentModalQuestionIndex] = useState(nul
l);

  const formattedAnswersRef = useRef(null); // Create a ref for the formatted an
swers container

  const openModal = (index) => {
    if (questions[index].type === 'image') {
        // If it's an image type question, directly open the file picker
        const fileInput = document.createElement('input');
        fileInput.type = 'file';
        fileInput.accept = 'image/*';
        fileInput.onchange = (e) => {
            const file = e.target.files[0];
            if (file) {
                const reader = new FileReader();
                reader.onloadend = () => {
                    setImage(reader.result);
                    setAnswers((prev) => ({
                        ...prev,
                        [index]: reader.result,
                    }));
                };
                reader.readAsDataURL(file);
            }
        };
        fileInput.click();
    } else {
        setCurrentModalQuestionIndex(index);
    }
  };

  const closeModal = () => {
    setCurrentModalQuestionIndex(null);
  };

  const resetAll = () => {
    setAnswers({});  // Clear all answers
    setImage(null);  // Clear the image
  };

  return (
    <div className="app">
      <div className="formatted-answers" ref={formattedAnswersRef}> {/* Attach t
he ref to the container */}
        {questions.map((question, index) => (
          <FormattedAnswer
            key={index}
```

```
            question={question}
            content={answers[index]}
            onClick={() => openModal(index)}
            questionType={question.type}
          />
        ))}
      </div>
      <div className="buttons">
        <SaveButton answers={answers} image={image} setAnswers={setAnswers} setI
mage={setImage} />
        <LoadButton setAnswers={setAnswers} setImage={setImage} />
        <PrintButton contentRef={formattedAnswersRef} />
        <button className="reset-all" onClick={resetAll}>Reset All</button>
      </div>
      <Modal
        key={currentModalQuestionIndex}
        isOpen={currentModalQuestionIndex !== null}
        onClose={closeModal}
        onSave={(editorContent) => {
          if (currentModalQuestionIndex !== null) {
            setAnswers((prev) => ({
              ...prev,
              [currentModalQuestionIndex]: editorContent,
            }));
            closeModal();
          }
        }}
        initialAnswer={answers[currentModalQuestionIndex] || ''}
        questionType={questions[currentModalQuestionIndex]?.type}
        setImage={setImage}
      >
        <h2>{questions[currentModalQuestionIndex]?.header}</h2>
        <p>{questions[currentModalQuestionIndex]?.detail}</p>
      </Modal>
    </div>
  );
}

export default App;

// App.css
/* Global Styles */
body {
    font-family: 'Arial', sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
}

.app {
    max-width: 800px;
    margin: 50px auto;
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.buttons {
    margin-top: 20px;
    display: flex;
    gap: 10px;
```

```
}

/* Question Component Styles */
.question {
    margin-bottom: 20px;
}

.question label {
    display: block;
    margin-bottom: 8px;
    color: #333;
    font-weight: 600;
}

.question input {
    width: 100%;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
    font-size: 16px;
    transition: border 0.3s ease;
}

.question input:focus {
    border-color: #007BFF;
    outline: none;
    box-shadow: 0 0 5px rgba(0, 123, 255, 0.5);
}

/* Button Styles */
button {
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    font-size: 16px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

button:hover {
    opacity: 0.9;
}

/* Save and Load Button Styles */
button + button {
    margin-left: 10px;
}

.SaveButton {
    background-color: #007BFF;
    color: #fff;
}

.LoadButton {
    background-color: #28a745;
    color: #fff;
}

button.reset-all {
    background-color: #FF4136;
    color: #fff;
}
```

```
.print-hide {
    display: none !important;
}

.uploaded-image {
    max-width: 100%;
    height: auto;
    margin-bottom: 20px;
}

.answer-textarea {
    width: 100%;
    padding: 10px;
    border: none;
    background-color: #f4f4f4;
    margin-top: 5px;
    resize: none; /* Disables the resize handle */
    cursor: pointer; /* Indicates it's clickable */
}

// ImageUpload.jsx
function ImageUpload({ setImage }) {
  const handleImageChange = (e) => {
    const file = e.target.files[0];
    if (file) {
      const reader = new FileReader();
      reader.onloadend = () => {
        setImage(reader.result);
      };
      reader.readAsDataURL(file);
    }
  };

  return <input type="file" accept="image/png" onChange={handleImageChange} />;
}

export default ImageUpload;

//LoadButton.jsx
import React from 'react';
import ReactFileReader from 'react-file-reader';

function LoadButton({ setAnswers, setImage }) {
  const handleLoad = (files) => {
    const file = files[0];
    const reader = new FileReader();
    reader.onload = (e) => {
      const data = JSON.parse(e.target.result);
      if (data.image) {
        setImage(data.image);
        delete data.image;
      }
      setAnswers(data);
    };
    reader.readAsText(file);
  };

  return (
    <ReactFileReader handleFiles={handleLoad} fileTypes={['.json']}>
      <button>Load</button>
    </ReactFileReader>
```

```
  );
}

export default LoadButton;

// SaveButton.jsx
import React from 'react';
import { saveAs } from 'file-saver';

function SaveButton({ answers, image, setAnswers, setImage }) {
  const handleSave = () => {
    const dataToSave = { ...answers };
    if (image) {
      dataToSave.image = image;
    }

    // Prompt the user for a filename
    const filename = window.prompt("Enter a filename:", "data.json");
    if (!filename) return;  // If the user cancels or enters nothing, exit the f
unction

    const blob = new Blob([JSON.stringify(dataToSave, null, 2)], { type: 'applic
ation/json' });
    saveAs(blob, filename);
  };

  return <button onClick={handleSave}>Save</button>;
}

export default SaveButton;

// Modal.jsx
import React, { useState, useEffect } from 'react';
import ReactQuill from 'react-quill';
import 'react-quill/dist/quill.snow.css';
import './Modal.css';

function Modal({ isOpen, onClose, onSave, children, initialAnswer, questionType,
 setImage }) {
  const [editorContent, setEditorContent] = useState(initialAnswer || '');

  useEffect(() => {
    setEditorContent(initialAnswer || '');
  }, [initialAnswer]);

  const handleSave = () => {
    onSave(editorContent);
    onClose();
  };

  if (!isOpen) return null;

  return (
    <div className="modal-overlay">
      <div className="modal-content">
        {children}
        <p>Answer:</p>
        <ReactQuill
          value={editorContent}
          onChange={setEditorContent}
          className="modal-quill-editor"
        />
```

```
        <div className="modal-actions">
          <button onClick={onClose}>Cancel</button>
          <button onClick={handleSave}>OK</button>
        </div>
      </div>
    </div>
  );
}

export default Modal;

/* Modal Overlay Styles */
.modal-overlay {
    position: fixed; /* Fixed positioning */
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background-color: rgba(0, 0, 0, 0.7); /* Semi-transparent black */
    display: flex;
    justify-content: center;
    align-items: center;
    z-index: 1000; /* High z-index to ensure it's above other content */
}

/* Modal Content Styles */
.modal-content {
    background-color: #ffffff;
    padding: 20px;
    border-radius: 8px;
    max-width: 80%; /* or whatever maximum width you prefer */
    max-height: 80%; /* to ensure the modal doesn't take up the entire screen he
ight */
    overflow-y: auto; /* to allow scrolling if the content is too tall */
    box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.2); /* Optional: Adds a subtle shad
ow for a more "elevated" look */
}

/* Modal Actions (Buttons) Styles */
.modal-actions {
    margin-top: 20px;
    display: flex;
    justify-content: flex-end;
    gap: 10px; /* Space between buttons */
}

/* Modal Quill Editor Styles */
.modal-quill-editor {
    margin-top: 10px;
}

// PrintButton.jsx
import React from 'react';
import html2pdf from 'html2pdf.js';

function PrintButton({ contentRef }) {
  const handlePrint = () => {
    const content = contentRef.current;
    const buttonsContainer = content.querySelector('.buttons');
    if (buttonsContainer) {
        buttonsContainer.classList.add('print-hide');
    }
```

```
    const opt = {
      margin: 10,
      filename: 'SecurityReport.pdf',
      image: { type: 'jpeg', quality: 0.98 },
      html2canvas: { scale: 2 },
      jsPDF: { unit: 'mm', format: 'a4', orientation: 'portrait' },
    };

    html2pdf().from(content).set(opt).save().then(() => {
      buttonsContainer.classList.remove('print-hide');
    });
  };

  return <button onClick={handlePrint}>Print as PDF</button>;
}

export default PrintButton;

// Question.js
import React from 'react';

function Question({ index, question, answer, onClick }) {
  return (
    <div className="question">
      <label>{question.header}:</label>
      <textarea
        readOnly
        value={answer}
        onClick={() => onClick(index)}
        className="answer-textarea"
      />
    </div>
  );
}

export default Question;

// questions.jsx
  const questions = [
    {
      header: "Diagram",
      detail: "Add a diagram",
      type: "image"
    },
    {
      header: "Project/Task Name",
      detail: "What is the name or title of the project or task you worked on th
is week?"
    },
    {
      header: "Objective",
      detail: "What was the primary goal or objective of this project/task?"
    },
    {
      header: "Proposed Architecture Details",
      detail: "Can you provide a brief overview of the proposed architecture?"
    },
    {
      header: "Data Classification",
      detail: "What are the data topics and their respective classifications tha
t the project will handle?"
```

```
    },
    {
      header: "Threat Analysis",
      detail: "Based on the proposed architecture and data classification, what
potential threats have you identified?"
    },
    {
      header: "Security Specifications",
      detail: "What specific safeguards or security measures have you recommende
d to mitigate the identified threats?"
    },
    {
      header: "Gap Analysis",
      detail: "Were there any gaps between the proposed security specifications
and existing security controls? If so, what alternatives did you suggest?"
    },
    {
      header: "Security Exceptions",
      detail: "Did you have to submit any information security policy & standard
s exceptions this week? If so, what were the reasons?"
    },
    {
      header: "Penetration Tests & Assessments",
      detail: "Were any penetration tests or security assessments conducted? Wha
t were the outcomes?"
    },
    {
      header: "Collaborations",
      detail: "Did you collaborate with any Solution Architects, DevOps Teams, o
r other stakeholders? If so, what was the nature of the collaboration?"
    },
    {
      header: "Impact",
      detail: "How does your security review align with the company's goals or o
bjectives? What potential impact does it have on ensuring a safe and consistent
customer experience?"
    },
    {
      header: "Next Steps",
      detail: "What are the upcoming tasks or milestones related to this project
/initiative?"
    }
  ];

  export default questions;

// FormattedAnswer.jsx
import React from 'react';
import ReactQuill from 'react-quill';
import 'react-quill/dist/quill.snow.css';
import './FormattedAnswer.css';

function FormattedAnswer({ question, content, onClick, questionType }) {
  return (
    <div className="formatted-answer" onClick={() => onClick()}>
        <h3>{question.header}</h3>
        <div className="modal-quill-editor">
            {questionType === 'image' ? (
                <div>
                    {content ? (
                        <img src={content} alt="Uploaded Diagram" style={{ maxWi
dth: '100%', height: 'auto' }} />
```

```
                    ) : (
                        "No image uploaded."
                    )}
                </div>
            ) : (
                <ReactQuill
                    value={content || ''}
                    readOnly={true}
                    theme={null}
                />
            )}
        </div>
    </div>
  );
}

export default FormattedAnswer;

/* FormattedAnswer.css */

.formatted-answer {
  border: 1px solid #ddd;
  border-radius: 4px;
  padding: 10px;
  margin-top: 5px;
  background-color: #f4f4f4;
  cursor: pointer;
}

/* Additional styles to make the editor look more like a textarea */
.ql-editor {
  min-height: 300px;
  height: auto;
  overflow-y: auto;
  border: none;
  padding: 0;
}
```