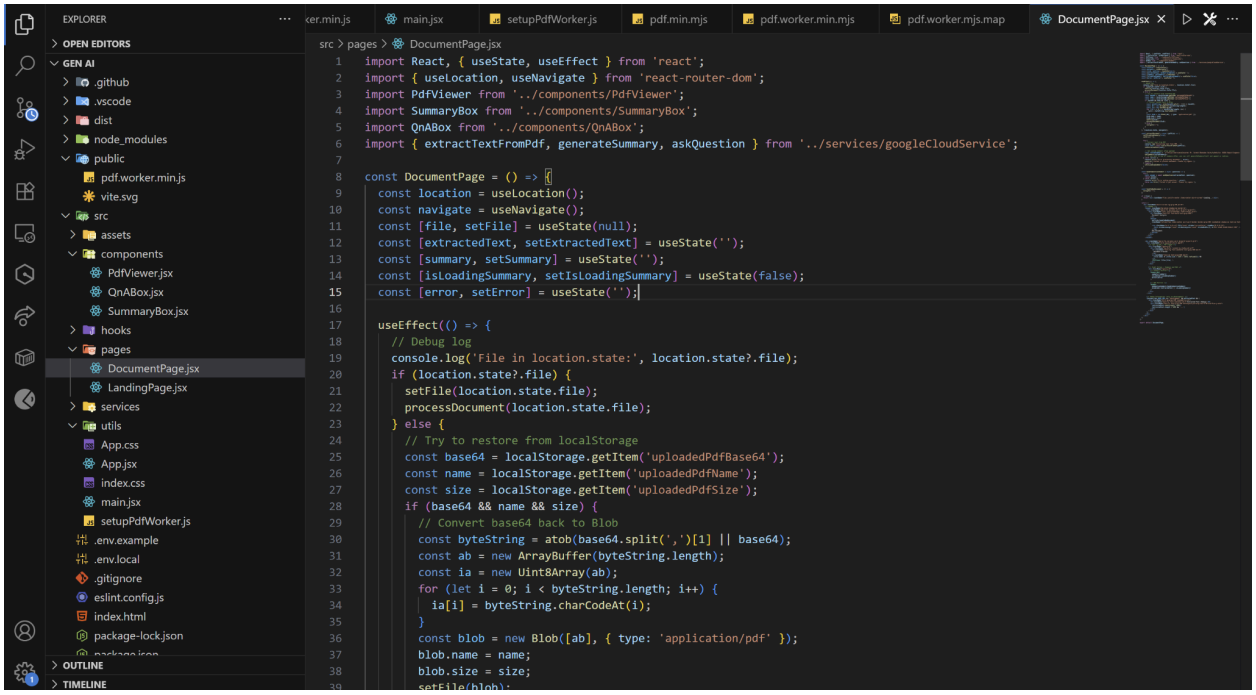
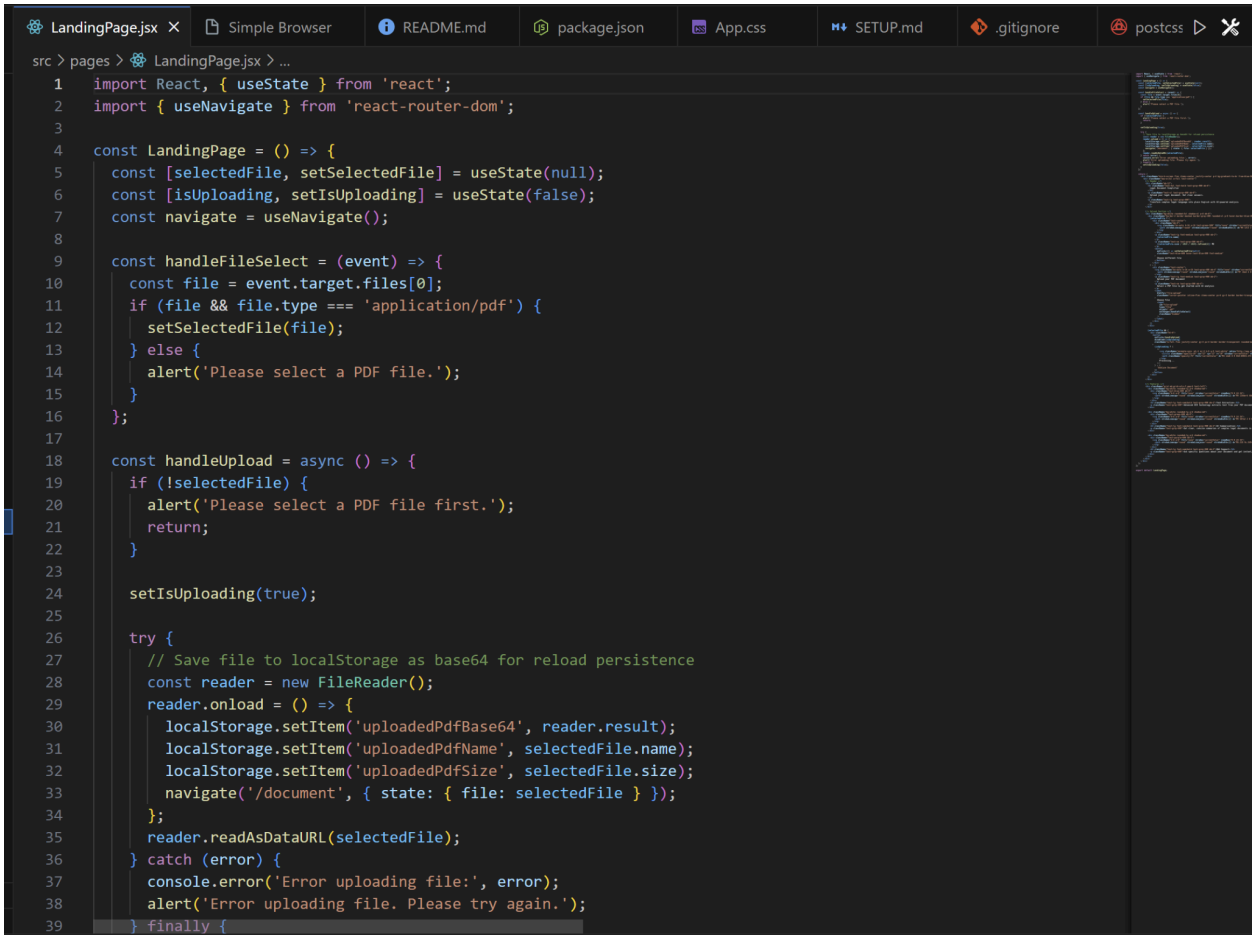
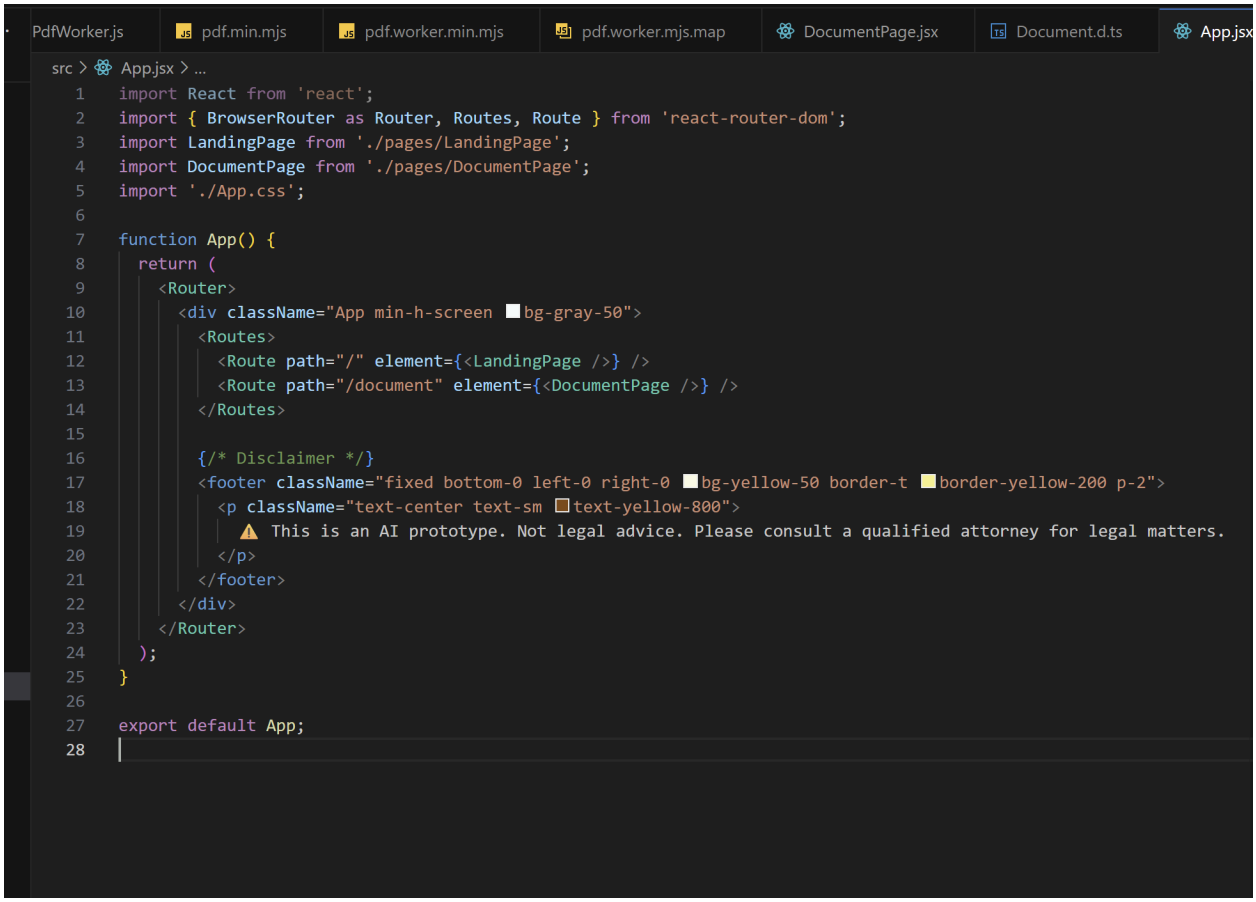

EXPLORER
OPEN EDITORS
GEN AI
.github
.vscode
dist
node_modules
public
pdf.worker.min.js
vite.svg
src
assets
components
PdfViewer.jsx
QnABox.jsx
SummaryBox.jsx
hooks
pages
DocumentPage.jsx
LandingPage.jsx
services
utils
App.css
App.jsx
index.css
main.jsx
setupPdfWorker.js
.env.example
.env.local
.gitignore
eslint.config.js
index.html
package-lock.json
OUTLINE
TIMELINE
main.jsx
setupPdfWorker.js
pdf.min.mjs
pdf.worker.min.mjs
pdf.worker.mjs.map
DocumentPage.jsx
src > pages > DocumentPage.jsx
import React, { useState, useEffect } from 'react';
import { useLocation, useNavigate } from 'react-router-dom';
import PdfViewer from '../components/PdfViewer';
import SummaryBox from '../components/SummaryBox';
import QnABox from '../components/QnABox';
import { extractTextFromPdf, generateSummary, askQuestion } from '../services/googleCloudService';
const DocumentPage = () => {
const location = useLocation();
const navigate = useNavigate();
const [file, setFile] = useState(null);
const [extractedText, setExtractedText] = useState('');
const [summary, setSummary] = useState('');
const [isLoadingSummary, setIsLoadingSummary] = useState(false);
const [error, setError] = useState('');
useEffect(() => {
// Debug log
console.log('File in location.state:', location.state?.file);
if (location.state?.file) {
setFile(location.state.file);
processDocument(location.state.file);
} else {
// Try to restore from localStorage
const base64 = localStorage.getItem('uploadedPdfBase64');
const name = localStorage.getItem('uploadedPdfName');
const size = localStorage.getItem('uploadedPdfSize');
if (base64 && name && size) {
// Convert base64 back to Blob
const byteString = atob(base64.split(',')[1] || base64);
const ab = new ArrayBuffer(byteString.length);
const ia = new Uint8Array(ab);
for (let i = 0; i < byteString.length; i++) {
ia[i] = byteString.charCodeAt(i);
}
const blob = new Blob([ab], { type: 'application/pdf' });
blob.name = name;
blob.size = size;


LandingPage.jsx
Simple Browser
README.md
package.json
App.css
SETUP.md
.gitignore
postcss
src > pages > LandingPage.jsx > ...
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
const LandingPage = () => {
const [selectedFile, setSelectedFile] = useState(null);
const [isUploading, setIsUploading] = useState(false);
const navigate = useNavigate();
const handleFileSelect = (event) => {
const file = event.target.files[0];
if (file && file.type === 'application/pdf') {
setSelectedFile(file);
} else {
alert('Please select a PDF file.');
}
};
const handleUpload = async () => {
if (!selectedFile) {
alert('Please select a PDF file first.');
return;
}
setIsUploading(true);
try {
// Save file to localStorage as base64 for reload persistence
const reader = new FileReader();
reader.onload = () => {
localStorage.setItem('uploadedPdfBase64', reader.result);
localStorage.setItem('uploadedPdfName', selectedFile.name);
localStorage.setItem('uploadedPdfSize', selectedFile.size);
navigate('/document', { state: { file: selectedFile } });
};
reader.readAsDataURL(selectedFile);
} catch (error) {
console.error('Error uploading file:', error);
alert('Error uploading file. Please try again.');
} finally {


PdfWorker.js
pdf.min.mjs
pdf.worker.min.mjs
pdf.worker.mjs.map
DocumentPage.jsx
Document.d.ts
App.jsx
src > App.jsx > ...
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import LandingPage from './pages/LandingPage';
import DocumentPage from './pages/DocumentPage';
import './App.css';
function App() {
return (
<Router>

<Routes>
<Route path="/" element={<LandingPage />} />
<Route path="/document" element={<DocumentPage />} />
</Routes>
{/* Disclaimer */}
<footer className="fixed bottom-0 left-0 right-0 bg-yellow-50 border-t border-yellow-200 p-2">

⚠ This is an AI prototype. Not legal advice. Please consult a qualified attorney for legal matters.

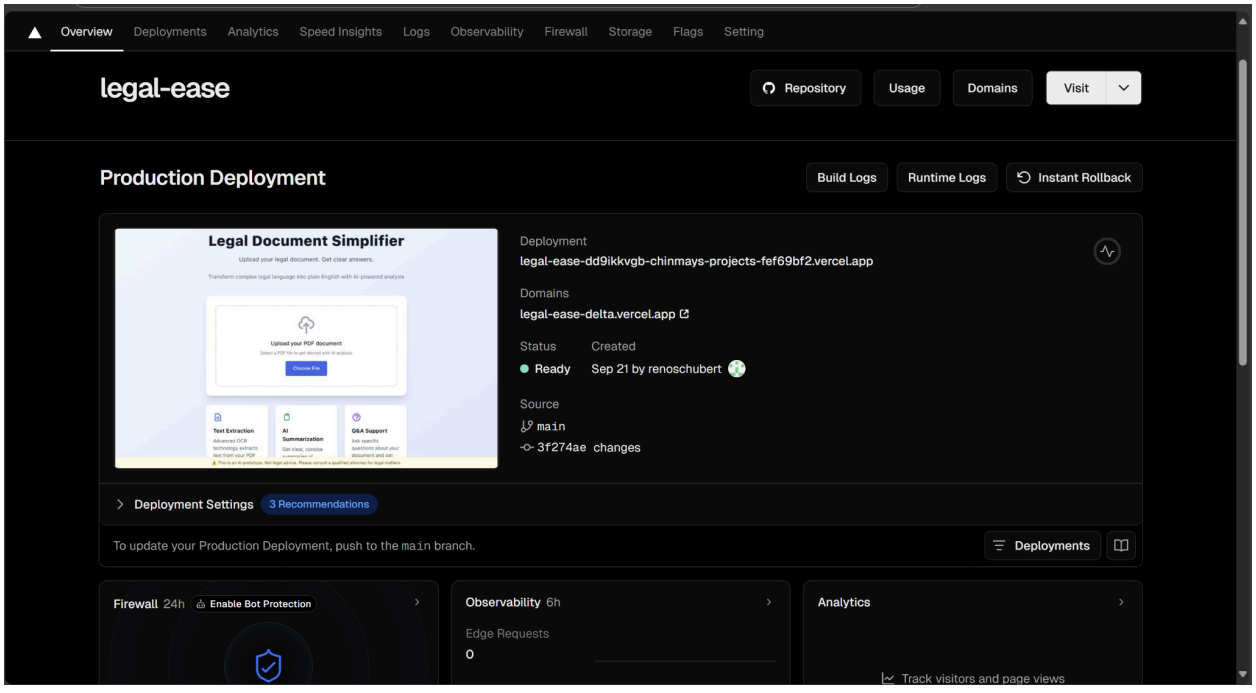
</footer>

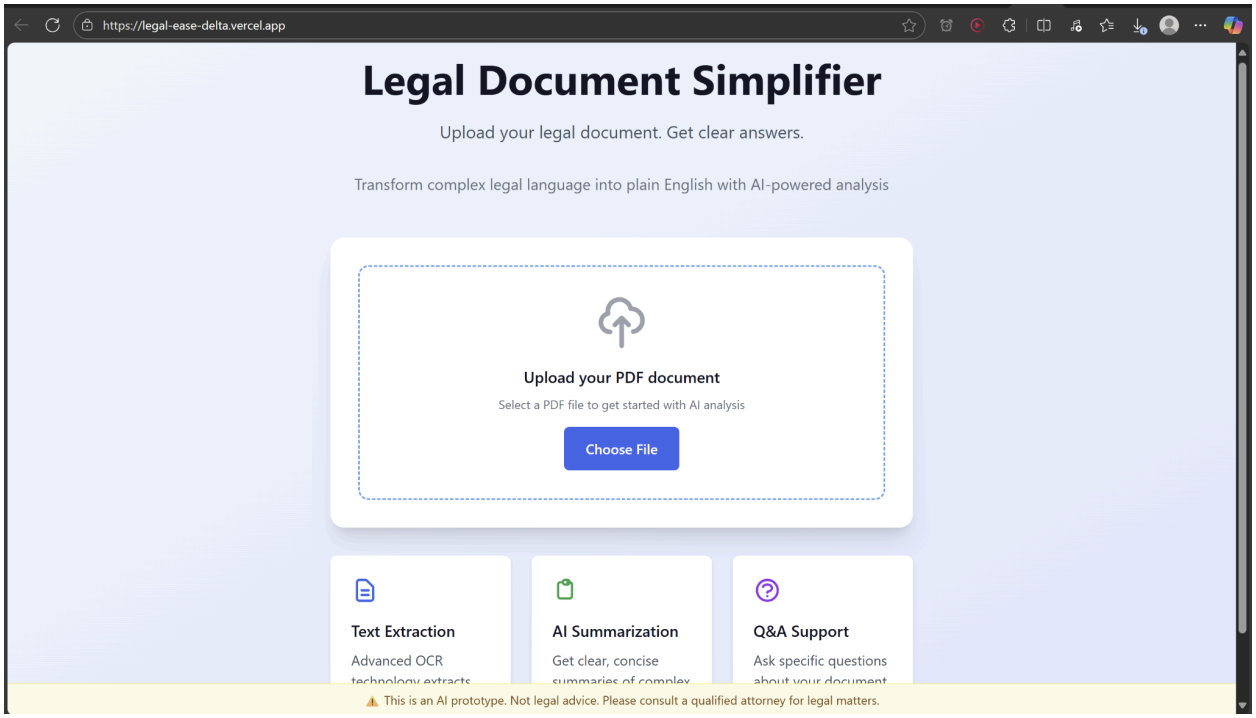
</Router>
);
}
export default App;

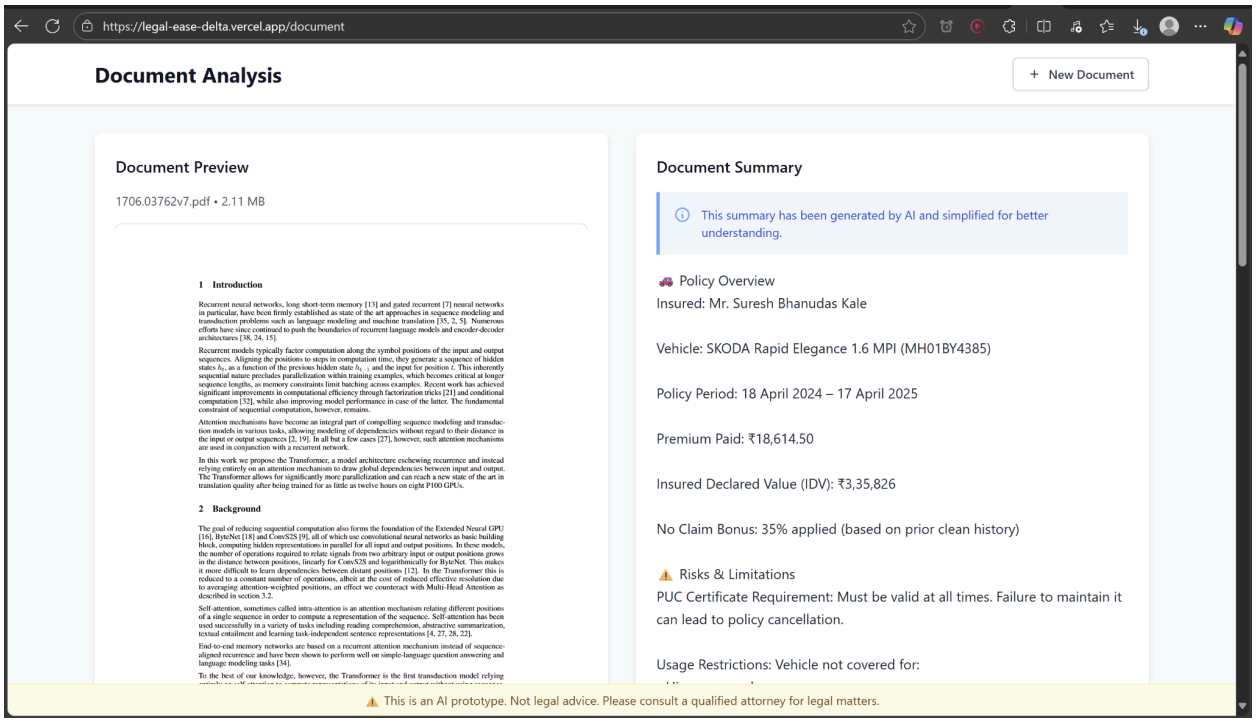
.env.example | .env.local | googleCloudService.js | requirements.md | QnABox.jsx | SummaryBox.jsx | PdfViewer.jsx

src > components > PdfViewer.jsx > ...

```jsx
import React, { useState, useEffect, useRef, useLayoutEffect } from 'react';
import { Document, Page, pdfjs } from 'react-pdf';
pdfjs.GlobalWorkerOptions.workerSrc = '/pdf.worker.min.js';

const PdfViewer = ({ file }) => {
  const [numPages, setNumPages] = useState(null);
  const [pageNumber, setPageNumber] = useState(1);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const viewerRef = useRef(null);
  const [pageWidth, setPageWidth] = useState(null);

  useEffect(() => {
    console.log('PdfViewer received file:', file);
    if (file) {
      console.log('File type:', typeof file);
      if (file instanceof Blob) {
        console.log('File is a Blob. Size:', file.size, 'Type:', file.type);
      }
      if (file instanceof File) {
        console.log('File is a File. Name:', file.name, 'Size:', file.size, 'Type:', file.type);
      }
    } else {
      console.warn('No file received by PdfViewer.');
    }
  }, [file]);

  const onDocumentLoadSuccess = ({ numPages }) => {
    setNumPages(numPages);
    setLoading(false);
    setError(null);
  };

  const onDocumentLoadError = (error) => {
    console.error('Error loading PDF:', error);
    setError('Failed to load PDF document');
    setLoading(false);
  };

```







[link to the websites]