Below are the full contents of each file you need to replace.

**1. client/src/pages/print-quote.tsx**

tsx

**import** { useEffect, useState } **from** 'react';

**import** frameImage **from** '@assets/image\_1757577550193.png';

**import** emetLogo **from** '@assets/image\_1757577759606.png';

**import** techDiagram **from** '@assets/image\_1757577458643.png';

*// Comprehensive types for quote data with proper field mapping*

**interface** QuoteData {

quote: {

subject: string;

customer: string;

salesPerson: string;

terms: string | number;

currency: string;

};

bomItems: Array<{

no: string | number;

partNumber: string;

productDescription: string;

quantity: number;

unitPrice?: number;

totalPrice?: number;

}>;

costItems: Array<{

description?: string;

productDescription?: string;

quantity: number;

unitPrice: number;

totalPrice: number;

isDiscount: boolean;

}>;

columnVisibility: {

no: boolean;

partNumber: boolean;

productDescription: boolean;

qty: boolean;

unitPrice: boolean;

totalPrice: boolean;

};

bomEnabled: boolean;

costsEnabled: boolean;

date: string;

version: string;

contact: {

name?: string;

salesPersonName?: string;

phone: string;

email: string;

};

}

**export** **default** **function** PrintQuote() {

**const** [quoteData, setQuoteData] = useState<QuoteData | **null**>(**null**);

**const** [loadingError, setLoadingError] = useState<string | **null**>(**null**);

*// Enhanced data loading with better error handling*

useEffect(() => {

**const** loadQuoteData = () => {

**try** {

**const** savedData = localStorage.getItem('quoteData');

**if** (savedData) {

**const** parsedData = JSON.parse(savedData);

**if** (!parsedData.quote) {

setLoadingError('Invalid quote data structure');

**return**;

}

**const** mappedData: QuoteData = {

quote: {

subject: parsedData.quote.subject || parsedData.quoteSubject || 'Untitled Quote',

customer: parsedData.quote.customer || parsedData.customerCompany || 'Customer Name',

salesPerson: parsedData.quote.salesPerson || parsedData.salesPersonName || 'Sales Representative',

terms: parsedData.quote.terms || parsedData.paymentTerms || 30,

currency: parsedData.quote.currency || parsedData.currency || 'USD'

},

bomItems: (parsedData.bomItems || []).map((item: any, i: number) => ({

no: item.no || i + 1,

partNumber: item.partNumber || '',

productDescription: item.productDescription || item.description || '',

quantity: item.quantity || 1,

unitPrice: item.unitPrice,

totalPrice: item.totalPrice

})),

costItems: (parsedData.costItems || []).map((item: any) => ({

description: item.description || item.productDescription || '',

productDescription: item.productDescription || item.description || '',

quantity: item.quantity || 1,

unitPrice: item.unitPrice || 0,

totalPrice: item.totalPrice || 0,

isDiscount: item.isDiscount || false

})),

columnVisibility: parsedData.columnVisibility || {

no: true, partNumber: true, productDescription: true,

qty: true, unitPrice: false, totalPrice: false

},

bomEnabled: parsedData.bomEnabled ?? true,

costsEnabled: parsedData.costsEnabled ?? true,

date: parsedData.date || **new** Date().toISOString().split('T')[0],

version: parsedData.version || '1',

contact: {

name: parsedData.contact?.name || parsedData.contact?.salesPersonName || parsedData.salesPersonName,

salesPersonName: parsedData.contact?.salesPersonName || parsedData.contact?.name || parsedData.salesPersonName,

phone: parsedData.contact?.phone || '+972-XXX-XXXX',

email: parsedData.contact?.email || 'info@emetdorcom.com'

}

};

setQuoteData(mappedData);

} **else** {

setLoadingError('No quote data found. Please create a quote first.');

}

} **catch** (error) {

setLoadingError('Failed to load quote data. Please try again.');

}

};

loadQuoteData();

}, []);

*// Enhanced printing logic*

useEffect(() => {

**if** (!quoteData || loadingError) **return**;

**const** waitForPrint = **async** () => {

**try** {

**const** images = [frameImage, emetLogo, techDiagram];

**const** loadImg = (src: string) =>

**new** Promise<**void**>(resolve => {

**const** img = **new** Image();

img.onload = () => resolve();

img.onerror = () => resolve();

img.src = src;

setTimeout(resolve, 5000);

});

**await** Promise.all(images.map(loadImg));

**await** Promise.all(Array.from(document.images).map(img => img.complete ? Promise.resolve() : **new** Promise(r => { img.onload = r; img.onerror = r; setTimeout(r,3000);})));

**await** **new** Promise(r => requestAnimationFrame(r));

window.print();

} **catch** {

window.print();

}

};

setTimeout(waitForPrint, 200);

}, [quoteData, loadingError]);

**const** formatDate = (d: string) => **new** Date(d).toLocaleDateString('en-GB') || d;

**const** formatCurrency = (amt: number, cur='USD') => {

**try** {

**const** map:{[k:string]:any}={USD:{code:'USD',locale:'en-US'},NIS:{code:'ILS',locale:'he-IL'},EUR:{code:'EUR',locale:'de-DE'}};

**const** cfg = map[cur]||map.USD;

**return** **new** Intl.NumberFormat(cfg.locale,{style:'currency',currency:cfg.code}).format(amt);

} **catch** {

**return** amt+' '+cur;

}

};

**if** (loadingError) **return** <div>Error: {loadingError}</div>;

**if** (!quoteData) **return** <div>Loading quote data...</div>;

**const** { quote, bomItems, costItems, columnVisibility, bomEnabled, costsEnabled, date, version, contact } = quoteData;

**const** grandTotal = costItems.reduce((s,i)=>s+(i.isDiscount?-i.totalPrice:i.totalPrice),0);

**return** (

<div className="print-container">

<style jsx>{`

@media print {

.print-container { font-family:Arial,sans-serif; font-size:12px; }

.page-break { page-break-before:always; }

}

`}</style>

{*/\* Page 1 \*/*}

<div className="page-break">

<img src={emetLogo} alt="Logo" style={{height:64,display:'block',margin:'16px auto'}}/>

<h1 style={{textAlign:'center',fontSize:48}}>Quotation For</h1>

<h2 style={{textAlign:'center',fontSize:32,borderBottom:'4px solid #2563EB',margin:'16px'}}>

{quote.subject}

</h2>

<p style={{textAlign:'center',marginTop:32}}>

{quote.salesPerson} | {formatDate(date)} | Ver {version}

</p>

</div>

{*/\* Page 2 \*/*}

<div className="page-break">

<img src={emetLogo} alt="Logo" style={{height:48}}/>

<h3>Intro</h3>

<p>EMET Dorcom **is** one **of** the most successful...</p>

<img src={techDiagram} alt="Diagram" style={{width:'100%',marginTop:16}}/>

</div>

{*/\* Page 3: BOM & Costs \*/*}

{(bomEnabled || costsEnabled) && (

<div className="page-break">

<img src={emetLogo} alt="Logo" style={{height:48}}/>

{bomEnabled && (

<>

<h3>BOM</h3>

<table style={{width:'100%',borderCollapse:'collapse',fontSize:10}}>

<thead>

<tr>

{columnVisibility.no && <th style={{border:'1px solid #ccc',padding:4}}>NO</th>}

{columnVisibility.partNumber && <th style={{border:'1px solid #ccc',padding:4}}>PN</th>}

{columnVisibility.productDescription && <th style={{border:'1px solid #ccc',padding:4}}>Product Description</th>}

{columnVisibility.qty && <th style={{border:'1px solid #ccc',padding:4}}>QTY</th>}

{columnVisibility.unitPrice && <th style={{border:'1px solid #ccc',padding:4}}>Unit Price</th>}

{columnVisibility.totalPrice && <th style={{border:'1px solid #ccc',padding:4}}>Total Price</th>}

</tr>

</thead>

<tbody>

{bomItems.map((item, i) => (

<tr key={i} style={{backgroundColor:i%2?'#f9fafb':'#fff'}}>

{columnVisibility.no && <td style={{border:'1px solid #ccc',padding:4}}>{item.no}</td>}

{columnVisibility.partNumber && <td style={{border:'1px solid #ccc',padding:4}}>{item.partNumber}</td>}

{columnVisibility.productDescription && <td style={{border:'1px solid #ccc',padding:4}}>{item.productDescription}</td>}

{columnVisibility.qty && <td style={{border:'1px solid #ccc',padding:4}}>{item.quantity}</td>}

{columnVisibility.unitPrice && <td style={{border:'1px solid #ccc',padding:4}}>{item.unitPrice!=**null**?formatCurrency(item.unitPrice,quote.currency):''}</td>}

{columnVisibility.totalPrice && <td style={{border:'1px solid #ccc',padding:4}}>{item.totalPrice!=**null**?formatCurrency(item.totalPrice,quote.currency):''}</td>}

</tr>

))}

</tbody>

</table>

</>

)}

{costsEnabled && (

<>

<h3 style={{marginTop:16}}>Costs</h3>

<table style={{width:'100%',borderCollapse:'collapse',fontSize:10}}>

<thead>

<tr>

<th style={{border:'1px solid #ccc',padding:4}}>Description</th>

<th style={{border:'1px solid #ccc',padding:4}}>QTY</th>

<th style={{border:'1px solid #ccc',padding:4}}>Unit Price</th>

<th style={{border:'1px solid #ccc',padding:4}}>Total Price</th>

</tr>

</thead>

<tbody>

{costItems.map((item,i)=>(

<tr key={i} style={{backgroundColor:i%2?'#f9fafb':'#fff'}}>

<td style={{border:'1px solid #ccc',padding:4}}>

{item.isDiscount?'-':''}{item.description||item.productDescription}

</td>

<td style={{border:'1px solid #ccc',padding:4}}>{item.quantity}</td>

<td style={{border:'1px solid #ccc',padding:4}}>{formatCurrency(item.unitPrice,quote.currency)}</td>

<td style={{border:'1px solid #ccc',padding:4}}>

{item.isDiscount?'-':''}{formatCurrency(item.totalPrice,quote.currency)}

</td>

</tr>

))}

</tbody>

</table>

<p style={{textAlign:'right',marginTop:8,fontWeight:'bold'}}>Grand Total: {formatCurrency(grandTotal,quote.currency)}</p>

</>

)}

</div>

)}

{*/\* Page 4 \*/*}

<div className="page-break">

<img src={emetLogo} alt="Logo" style={{height:48}}/>

<h3>Payment Terms</h3>

<p><strong>Terms:</strong> {quote.terms} days</p>

<p><strong>Currency:</strong> {quote.currency}</p>

</div>

{*/\* Page 5 \*/*}

<div className="page-break">

<img src={emetLogo} alt="Logo" style={{height:48}}/>

<h3>Contact Information</h3>

<p><strong>Sales Person:</strong> {contact.name||contact.salesPersonName||quote.salesPerson}</p>

<p><strong>Phone:</strong> {contact.phone}</p>

<p><strong>Email:</strong> {contact.email}</p>

<p><strong>Company:</strong> EMET Dorcom</p>

<p><strong>Customer:</strong> {quote.customer}</p>

</div>

</div>

);

}

**2. client/src/components/QuotePreview.tsx**

tsx

**import** { Card, CardContent } **from** "@/components/ui/card";

**import** { Button } **from** "@/components/ui/button";

**import** { ZoomIn, ZoomOut, Download, FileText } **from** "lucide-react";

**import** { useState } **from** "react";

**import** emetLogo **from** "@assets/image\_1757577759606.png";

**import** **type** { ColumnVisibility, ContactInfo } **from** "@shared/schema";

**interface** BomItem {

no: number;

partNumber: string;

productDescription: string;

quantity: number;

unitPrice?: number;

totalPrice?: number;

}

**interface** CostItem {

productDescription: string;

quantity: number;

unitPrice: number;

totalPrice: number;

isDiscount: boolean;

}

**interface** QuotePreviewProps {

quoteSubject: string;

customerCompany: string;

customerLogo?: string;

salesPersonName: string;

date: string;

version: string;

paymentTerms: string;

currency: string;

bomEnabled: boolean;

costsEnabled: boolean;

columnVisibility: ColumnVisibility;

contactInfo: ContactInfo;

bomItems: BomItem[];

costItems: CostItem[];

onSectionClick?: (section: string) => **void**;

}

**export** **default** **function** QuotePreview({

quoteSubject,

customerCompany,

customerLogo,

salesPersonName,

date,

version,

paymentTerms,

currency,

bomEnabled,

costsEnabled,

columnVisibility,

contactInfo,

bomItems,

costItems,

onSectionClick,

}: QuotePreviewProps) {

**const** [zoom, setZoom] = useState(100);

**const** [isExporting, setIsExporting] = useState(false);

**const** formatCurrency = (amount: number) => {

**const** map: any = {

USD: { code: "USD", locale: "en-US" },

NIS: { code: "ILS", locale: "he-IL" },

EUR: { code: "EUR", locale: "de-DE" },

ILS: { code: "ILS", locale: "he-IL" },

};

**const** cfg = map[currency] || map.USD;

**return** **new** Intl.NumberFormat(cfg.locale, {

style: "currency",

currency: cfg.code,

}).format(amount);

};

**const** formatDate = (d: string) => **new** Date(d).toLocaleDateString("en-GB") || d;

**const** grandTotal = costItems.reduce((s, i) => s + (i.isDiscount ? -i.totalPrice : i.totalPrice), 0);

**const** handleZoom = (dir: "in" | "out") => {

setZoom((z) => Math.max(50, Math.min(150, z + (dir === "in" ? 25 : -25))));

};

**const** handleExport = **async** () => {

setIsExporting(true);

**const** data = {

quote: { subject: quoteSubject, customer: customerCompany, salesPerson: salesPersonName, terms: paymentTerms, currency },

bomItems,

costItems,

columnVisibility,

bomEnabled,

costsEnabled,

date,

version,

contact: contactInfo,

};

localStorage.setItem("quoteData", JSON.stringify(data));

window.open("/print", "\_blank", "width=800,height=600");

setTimeout(() => setIsExporting(false), 1000);

};

**return** (

<Card className="h-full flex flex-col">

<CardContent className="flex-1 p-0">

<div className="flex justify-between p-4 border-b bg-gray-50">

<div className="flex items-center gap-2">

<Button size="sm" onClick={() => handleZoom("out")} disabled={zoom <= 50}><ZoomOut /></Button>

<span>{zoom}%</span>

<Button size="sm" onClick={() => handleZoom("in")} disabled={zoom >= 150}><ZoomIn /></Button>

</div>

<Button size="sm" onClick={handleExport} disabled={isExporting}>

<FileText /><span className="ml-1">{isExporting ? "Exporting..." : "Browser PDF"}</span>

</Button>

</div>

<div

className="overflow-auto bg-gray-100 flex-1"

style={{ transform: `scale(${zoom / 100})`, transformOrigin: "top center" }}

>

{*/\* Cover Page Preview \*/*}

<div className="min-h-[297mm] w-[210mm] mx-auto bg-white shadow mb-4 relative">

<img src={emetLogo} alt="Logo" className="h-24 mx-auto mt-8" />

<h1 className="text-center text-6xl font-bold mt-12">Quotation For</h1>

<h2 className="text-center text-4xl font-semibold text-blue-600 mt-4 border-b-4 pb-4">{quoteSubject}</h2>

{customerLogo && <img src={customerLogo} alt="Customer Logo" className="h-12 mx-auto mt-4" />}

<p className="text-center mt-8">{salesPersonName} | {formatDate(date)} | Ver {version}</p>

</div>

{*/\* Extend for remaining pages as needed \*/*}

</div>

</CardContent>

</Card>

);

}

**3. server/routes.ts**  
Add or replace the /api/generate-pdf route inside your existing registerRoutes function:

ts

app.post("/api/generate-pdf", **async** (req, res) => {

**try** {

**const** { quoteData, filename } = req.body;

**if** (!quoteData || !filename) {

**return** res.status(400).json({ error: "quoteData and filename required" });

}

**const** data = {

quote: {

subject: quoteData.quote.subject || quoteData.quoteSubject || "Untitled Quote",

customer: quoteData.quote.customer || quoteData.customerCompany || "Customer",

salesPerson: quoteData.quote.salesPerson || quoteData.salesPersonName || "SalesRep",

terms: quoteData.quote.terms || quoteData.paymentTerms || "30",

currency: quoteData.quote.currency || quoteData.currency || "USD",

},

bomItems: quoteData.bomItems || [],

costItems: quoteData.costItems || [],

date: quoteData.date || **new** Date().toISOString().split("T")[0],

version: quoteData.version || "1",

contact: quoteData.contact || {},

};

**const** PDFDocument = **require**("pdfkit");

**const** doc = **new** PDFDocument({ size: "A4", margin: 40 });

**const** chunks: Buffer[] = [];

doc.on("data", (c) => chunks.push(c));

doc.on("end", () => {

**const** buffer = Buffer.concat(chunks);

res

.setHeader("Content-Type", "application/pdf")

.setHeader("Content-Disposition", `attachment; filename="${filename}"`)

.send(buffer);

});

*// Simple PDF for demo; extend styling as needed*

doc.fontSize(24).text("Professional Quote", { align: "center" });

doc.moveDown();

doc.fontSize(14).text(`Subject: ${data.quote.subject}`);

doc.text(`Customer: ${data.quote.customer}`);

doc.text(`Sales Person: ${data.quote.salesPerson}`);

doc.text(`Terms: ${data.quote.terms} days`);

doc.text(`Currency: ${data.quote.currency}`);

doc.end();

} **catch** (err) {

res.status(500).json({ error: err.message });

}

});

**Finally**, add pdfkit to your dependencies:

bash

npm install pdfkit @types/pdfkit