Frontend Quotes and Adjudication Component Code

Add this component before line 6090 in frontend/app.js

```
// ============================================
// QUOTES AND ADJUDICATION MANAGEMENT
// ============================================

function QuotesAndAdjudication({ user, setView, loadData }) {
  const [requisitions, setRequisitions] = useState([]);
  const [selectedReq, setSelectedReq] = useState(null);
  const [quotes, setQuotes] = useState([]);
  const [adjudication, setAdjudication] = useState(null);
  const [loading, setLoading] = useState(true);
  const [uploading, setUploading] = useState(false);
  const [showUploadForm, setShowUploadForm] = useState(false);
  const [showAdjudicationForm, setShowAdjudicationForm] = useState(false);

  const [uploadForm, setUploadForm] = useState({
    vendor_id: '',
    vendor_name: '',
    quote_number: '',
    quote_amount: '',
    currency: 'ZMW',
    notes: '',
    file: null
  });

  const [adjForm, setAdjForm] = useState({
    recommended_vendor_id: '',
    recommended_vendor_name: '',
    recommended_amount: '',
    currency: 'ZMW',
    summary: '',
    evaluation_criteria: '',
    technical_compliance: '',
    pricing_analysis: '',
    delivery_terms: '',
    payment_terms: '',
    recommendation_rationale: ''
  });

  // Load requisitions that need quotes
  useEffect(() => {
    loadRequisitions();
  }, []);

  const loadRequisitions = async () => {
    setLoading(true);
    try {
      const data = await api.getRequisitions();
      // Filter requisitions that are approved by procurement or in procurement review
      const filtered = data.filter(r =>
        r.status === 'pending_finance' || r.status === 'pending_md' || r.status === 'approved'
      );
      setRequisitions(filtered);
    } catch (error) {
      console.error('Error loading requisitions:', error);
    } finally {
      setLoading(false);
    }
  };

  const loadQuotesForReq = async (reqId) => {
    try {
      const quotesData = await api.getQuotes(reqId);
      setQuotes(quotesData);

      // Load adjudication if exists
      try {
        const adjData = await api.getAdjudication(reqId);
        setAdjudication(adjData);
      } catch (e) {
        setAdjudication(null);
      }
    } catch (error) {
      console.error('Error loading quotes:', error);
      setQuotes([]);
    }
  };

  const handleSelectRequisition = async (req) => {
    setSelectedReq(req);
    await loadQuotesForReq(req.id);
    setShowUploadForm(false);
    setShowAdjudicationForm(false);
  };

  const handleFileChange = (e) => {
    const file = e.target.files[0];
    if (file && file.type === 'application/pdf') {
      setUploadForm({ ...uploadForm, file });
    } else {
      alert('Please select a PDF file');
      e.target.value = '';
    }
  };

  const handleUploadQuote = async (e) => {
    e.preventDefault();
    if (!uploadForm.file) {
      alert('Please select a PDF file');
      return;
    }

    setUploading(true);
    try {
      const formData = new FormData();
      formData.append('quoteFile', uploadForm.file);
      formData.append('vendor_id', uploadForm.vendor_id);
      formData.append('vendor_name', uploadForm.vendor_name);
      formData.append('quote_number', uploadForm.quote_number);
      formData.append('quote_amount', uploadForm.quote_amount);
      formData.append('currency', uploadForm.currency);
      formData.append('notes', uploadForm.notes);

      await api.uploadQuote(selectedReq.id, formData);
      alert('Quote uploaded successfully!');

      // Reset form
      setUploadForm({
        vendor_id: '',
        vendor_name: '',
        quote_number: '',
        quote_amount: '',
        currency: 'ZMW',
        notes: '',
        file: null
      });
      document.querySelector('input[type="file"]').value = '';

      // Reload quotes
      await loadQuotesForReq(selectedReq.id);
      setShowUploadForm(false);
    } catch (error) {
      alert('Error uploading quote: ' + error.message);
    } finally {
      setUploading(false);
    }
  };

  const handleDeleteQuote = async (quoteId) => {
    if (!confirm('Are you sure you want to delete this quote?')) return;

    try {
      await api.deleteQuote(quoteId);
      alert('Quote deleted successfully');
      await loadQuotesForReq(selectedReq.id);
    } catch (error) {
      alert('Error deleting quote: ' + error.message);
    }
  };

  const handleSubmitAdjudication = async (e) => {
    e.preventDefault();

    if (quotes.length < 3) {
      alert('Please upload all 3 vendor quotes before creating adjudication');
      return;
    }

    setUploading(true);
    try {
      await api.createAdjudication(selectedReq.id, adjForm);
      alert('Adjudication created successfully!');
      await loadQuotesForReq(selectedReq.id);
      setShowAdjudicationForm(false);
    } catch (error) {
      alert('Error creating adjudication: ' + error.message);
    } finally {
      setUploading(false);
    }
  };

  const canUploadQuotes = user.role === 'procurement' || user.role === 'admin';
  const canViewQuotes = user.role === 'procurement' || user.role === 'finance' || user.role === 'md' || user.role === 'admin';

  if (loading) {
    return React.createElement('div', { className: "flex items-center justify-center p-8" },
      React.createElement('p', { className: "text-gray-600" }, "Loading requisitions...")
    );
  }

  return React.createElement('div', { className: "space-y-6" },
    React.createElement('div', { className: "bg-white rounded-lg shadow-sm border p-6" },
      React.createElement('h2', { className: "text-2xl font-bold text-gray-800 mb-4" }, "Quotes & Adjudication Management"),

      // Requisitions list
      !selectedReq && React.createElement('div', null,
        React.createElement('p', { className: "text-gray-600 mb-4" },
          canUploadQuotes
            ? "Select a requisition to upload vendor quotes and create adjudication"
            : "Select a requisition to view quotes and adjudication"
        ),
        React.createElement('div', { className: "space-y-3" },
          requisitions.length === 0
            ? React.createElement('p', { className: "text-center text-gray-500 py-8" }, "No requisitions available for quotes")
            : requisitions.map(req =>
                React.createElement('div', {
                  key: req.id,
                  onClick: () => handleSelectRequisition(req),
                  className: "p-4 border rounded-lg hover:bg-gray-50 cursor-pointer transition-colors"
                },
                  React.createElement('div', { className: "flex justify-between items-start" },
                    React.createElement('div', null,
                      React.createElement('h3', { className: "font-semibold text-gray-900" }, req.req_number || req.id),
                      React.createElement('p', { className: "text-sm text-gray-600" }, req.title || req.description),
                      React.createElement('div', { className: "flex gap-2 mt-2" },
                        React.createElement('span', { className: "text-xs px-2 py-1 bg-blue-100 text-blue-700 rounded" },
req.department),
                        req.has_quotes && React.createElement('span', { className: "text-xs px-2 py-1 bg-green-100 text-green-700
rounded" }, "✓ Quotes"),
                        req.has_adjudication && React.createElement('span', { className: "text-xs px-2 py-1 bg-purple-100 text-
purple-700 rounded" }, "✓ Adjudication")
                      )
                    ),
                    React.createElement('button', {
                      className: "text-blue-600 hover:text-blue-800"
                    }, "View →")
                  )
                )
              )
        )
      ),

      // Selected requisition view
      selectedReq && React.createElement('div', null,
        React.createElement('div', { className: "flex items-center justify-between mb-6" },
          React.createElement('div', null,
            React.createElement('button', {
              onClick: () => {
                setSelectedReq(null);
                setQuotes([]);
                setAdjudication(null);
              },
              className: "text-blue-600 hover:text-blue-800 mb-2"
            }, "← Back to List"),
            React.createElement('h3', { className: "text-xl font-bold text-gray-900" }, selectedReq.req_number || selectedReq.id),
            React.createElement('p', { className: "text-sm text-gray-600" }, selectedReq.title || selectedReq.description)
          ),
          canUploadQuotes && quotes.length < 3 && React.createElement('button', {
            onClick: () => setShowUploadForm(!showUploadForm),
            className: "px-4 py-2 bg-blue-600 text-white rounded-lg hover:bg-blue-700"
          }, showUploadForm ? "Cancel" : "Upload Quote")
        ),

        // Upload form
        showUploadForm && canUploadQuotes && React.createElement('form', {
          onSubmit: handleUploadQuote,
          className: "bg-blue-50 p-6 rounded-lg border border-blue-200 mb-6"
        },
          React.createElement('h4', { className: "font-semibold text-gray-900 mb-4" }, `Upload Quote ${quotes.length + 1} of 3`),
          React.createElement('div', { className: "grid grid-cols-2 gap-4 mb-4" },
            React.createElement('div', null,
              React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Vendor Name *"),
              React.createElement('input', {
                type: "text",
                required: true,
                value: uploadForm.vendor_name,
                onChange: (e) => setUploadForm({ ...uploadForm, vendor_name: e.target.value }),
                className: "w-full px-4 py-2 border border-gray-300 rounded-lg"
              })
            ),
            React.createElement('div', null,
              React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Quote Number"),
              React.createElement('input', {
                type: "text",
                value: uploadForm.quote_number,
                onChange: (e) => setUploadForm({ ...uploadForm, quote_number: e.target.value }),
                className: "w-full px-4 py-2 border border-gray-300 rounded-lg"
              })
            ),
            React.createElement('div', null,
              React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Quote Amount *"),
              React.createElement('input', {
                type: "number",
                step: "0.01",
                required: true,
                value: uploadForm.quote_amount,
                onChange: (e) => setUploadForm({ ...uploadForm, quote_amount: e.target.value }),
                className: "w-full px-4 py-2 border border-gray-300 rounded-lg"
              })
            ),
            React.createElement('div', null,
              React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Currency"),
              React.createElement('select', {
                value: uploadForm.currency,
                onChange: (e) => setUploadForm({ ...uploadForm, currency: e.target.value }),
                className: "w-full px-4 py-2 border border-gray-300 rounded-lg"
              },
                React.createElement('option', { value: "ZMW" }, "ZMW"),
                React.createElement('option', { value: "USD" }, "USD"),
                React.createElement('option', { value: "EUR" }, "EUR"),
                React.createElement('option', { value: "ZAR" }, "ZAR")
              )
            )
          ),
          React.createElement('div', { className: "mb-4" },
            React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Quote PDF File *"),
            React.createElement('input', {
              type: "file",
              accept: ".pdf",
              required: true,
              onChange: handleFileChange,
              className: "w-full px-4 py-2 border border-gray-300 rounded-lg"
            })
          ),
          React.createElement('div', { className: "mb-4" },
            React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Notes"),
            React.createElement('textarea', {
              rows: "2",
              value: uploadForm.notes,
              onChange: (e) => setUploadForm({ ...uploadForm, notes: e.target.value }),
              className: "w-full px-4 py-2 border border-gray-300 rounded-lg"
            })
          ),
          React.createElement('button', {
            type: "submit",
            disabled: uploading,
            className: "w-full bg-blue-600 text-white py-2 rounded-lg hover:bg-blue-700 disabled:bg-gray-400"
          }, uploading ? "Uploading..." : "Upload Quote")
        ),

        // Quotes list
        React.createElement('div', { className: "mb-6" },
          React.createElement('h4', { className: "font-semibold text-gray-900 mb-3" }, `Vendor Quotes (${quotes.length}/3)`),
          quotes.length === 0
            ? React.createElement('p', { className: "text-gray-500 text-center py-4" }, "No quotes uploaded yet")
            : React.createElement('div', { className: "space-y-3" },
                quotes.map((quote, idx) =>
                  React.createElement('div', {
                    key: quote.id,
                    className: "p-4 border rounded-lg bg-gray-50"
                  },
                    React.createElement('div', { className: "flex justify-between items-start" },
                      React.createElement('div', null,
                        React.createElement('h5', { className: "font-semibold text-gray-900" }, `Quote ${idx + 1}:
${quote.vendor_name}`),
                        React.createElement('p', { className: "text-sm text-gray-600" }, `Amount: ${quote.currency}
${parseFloat(quote.quote_amount).toLocaleString()}`),
                        quote.quote_number && React.createElement('p', { className: "text-sm text-gray-600" }, `Quote #:
${quote.quote_number}`),
                        React.createElement('p', { className: "text-xs text-gray-500 mt-1" }, `Uploaded: ${new
Date(quote.uploaded_at).toLocaleString()}`)
                      ),
                      React.createElement('div', { className: "flex gap-2" },
                        React.createElement('button', {
                          onClick: () => api.downloadQuote(quote.id),
                          className: "px-3 py-1 bg-green-600 text-white text-sm rounded hover:bg-green-700"
                        }, "Download PDF"),
                        canUploadQuotes && !adjudication && React.createElement('button', {
                          onClick: () => handleDeleteQuote(quote.id),
                          className: "px-3 py-1 bg-red-600 text-white text-sm rounded hover:bg-red-700"
                        }, "Delete")
                      )
                    )
                  )
                )
              )
        ),

        // Adjudication section
        canUploadQuotes && quotes.length === 3 && !adjudication && React.createElement('div', { className: "border-t pt-6" },
          React.createElement('div', { className: "flex justify-between items-center mb-4" },
            React.createElement('h4', { className: "font-semibold text-gray-900" }, "Create Adjudication Summary"),
            React.createElement('button', {
              onClick: () => setShowAdjudicationForm(!showAdjudicationForm),
              className: "px-4 py-2 bg-purple-600 text-white rounded-lg hover:bg-purple-700"
            }, showAdjudicationForm ? "Cancel" : "Create Adjudication")
          ),

          showAdjudicationForm && React.createElement('form', {
            onSubmit: handleSubmitAdjudication,
            className: "bg-purple-50 p-6 rounded-lg border border-purple-200 space-y-4"
          },
            React.createElement('div', { className: "grid grid-cols-2 gap-4" },
              React.createElement('div', null,
                React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Recommended Vendor *"),
                React.createElement('select', {
                  required: true,
                  value: adjForm.recommended_vendor_id,
                  onChange: (e) => {
                    const quote = quotes.find(q => q.vendor_id == e.target.value);
                    setAdjForm({
                      ...adjForm,
                      recommended_vendor_id: e.target.value,
                      recommended_vendor_name: quote ? quote.vendor_name : '',
                      recommended_amount: quote ? quote.quote_amount : '',
                      currency: quote ? quote.currency : 'ZMW'
                    });
                  },
                  className: "w-full px-4 py-2 border border-gray-300 rounded-lg"
                },
                  React.createElement('option', { value: "" }, "-- Select Vendor --"),
                  quotes.map(q => React.createElement('option', { key: q.id, value: q.vendor_id },
                    `${q.vendor_name} - ${q.currency} ${parseFloat(q.quote_amount).toLocaleString()}`
                  ))
                )
              ),
              React.createElement('div', null,
                React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Recommended Amount *"),
                React.createElement('input', {
                  type: "number",
                  step: "0.01",
                  required: true,
                  readOnly: true,
                  value: adjForm.recommended_amount,
                  className: "w-full px-4 py-2 border border-gray-300 rounded-lg bg-gray-100"
                })
              )
            ),
            React.createElement('div', null,
              React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Executive Summary *"),
              React.createElement('textarea', {
                rows: "3",
                required: true,
                value: adjForm.summary,
                onChange: (e) => setAdjForm({ ...adjForm, summary: e.target.value }),
                className: "w-full px-4 py-2 border border-gray-300 rounded-lg",
                placeholder: "Brief summary of the procurement process and quotes received..."
              })
            ),
            React.createElement('div', null,
              React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Evaluation Criteria"),
              React.createElement('textarea', {
                rows: "2",
                value: adjForm.evaluation_criteria,
                onChange: (e) => setAdjForm({ ...adjForm, evaluation_criteria: e.target.value }),
                className: "w-full px-4 py-2 border border-gray-300 rounded-lg",
                placeholder: "Criteria used to evaluate quotes (price, quality, delivery, etc.)..."
              })
            ),
            React.createElement('div', null,
              React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Technical Compliance"),
              React.createElement('textarea', {
                rows: "2",
                value: adjForm.technical_compliance,
                onChange: (e) => setAdjForm({ ...adjForm, technical_compliance: e.target.value }),
                className: "w-full px-4 py-2 border border-gray-300 rounded-lg",
                placeholder: "Assessment of technical specifications and compliance..."
              })
            ),
            React.createElement('div', null,
              React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Pricing Analysis"),
              React.createElement('textarea', {
                rows: "3",
                value: adjForm.pricing_analysis,
                onChange: (e) => setAdjForm({ ...adjForm, pricing_analysis: e.target.value }),
                className: "w-full px-4 py-2 border border-gray-300 rounded-lg",
                placeholder: "Comparison of the 3 vendor quotes and pricing breakdown..."
              })
            ),
            React.createElement('div', { className: "grid grid-cols-2 gap-4" },
              React.createElement('div', null,
                React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Delivery Terms"),
                React.createElement('textarea', {
                  rows: "2",
                  value: adjForm.delivery_terms,
                  onChange: (e) => setAdjForm({ ...adjForm, delivery_terms: e.target.value }),
                  className: "w-full px-4 py-2 border border-gray-300 rounded-lg"
                })
              ),
              React.createElement('div', null,
                React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Payment Terms"),
                React.createElement('textarea', {
                  rows: "2",
                  value: adjForm.payment_terms,
                  onChange: (e) => setAdjForm({ ...adjForm, payment_terms: e.target.value }),
                  className: "w-full px-4 py-2 border border-gray-300 rounded-lg"
                })
              )
            ),
            React.createElement('div', null,
              React.createElement('label', { className: "block text-sm font-medium text-gray-700 mb-2" }, "Recommendation Rationale
*"),
              React.createElement('textarea', {
                rows: "4",
                required: true,
                value: adjForm.recommendation_rationale,
                onChange: (e) => setAdjForm({ ...adjForm, recommendation_rationale: e.target.value }),
                className: "w-full px-4 py-2 border border-gray-300 rounded-lg",
                placeholder: "Detailed justification for the recommended vendor..."
              })
            ),
            React.createElement('button', {
              type: "submit",
              disabled: uploading,
              className: "w-full bg-purple-600 text-white py-3 rounded-lg font-semibold hover:bg-purple-700 disabled:bg-gray-400"
            }, uploading ? "Submitting..." : "Submit Adjudication")
          )
        ),

        // View adjudication
        adjudication && React.createElement('div', { className: "border-t pt-6" },
          React.createElement('h4', { className: "font-semibold text-gray-900 mb-4 flex items-center gap-2" },
            "Adjudication Summary",
            React.createElement('span', { className: "text-xs px-2 py-1 bg-green-100 text-green-700 rounded" }, "✓ Complete")
          ),
          React.createElement('div', { className: "bg-green-50 p-6 rounded-lg border border-green-200 space-y-4" },
            React.createElement('div', { className: "pb-4 border-b" },
              React.createElement('h5', { className: "font-semibold text-lg text-gray-900 mb-2" }, "Recommended Vendor"),
              React.createElement('p', { className: "text-xl font-bold text-green-700" }, adjudication.recommended_vendor_name),
              React.createElement('p', { className: "text-lg text-gray-900" },
                `${adjudication.currency} ${parseFloat(adjudication.recommended_amount).toLocaleString()}`
              )
            ),
            React.createElement('div', null,
              React.createElement('h6', { className: "font-semibold text-gray-900 mb-1" }, "Executive Summary"),
              React.createElement('p', { className: "text-gray-700" }, adjudication.summary)
            ),
            adjudication.evaluation_criteria && React.createElement('div', null,
              React.createElement('h6', { className: "font-semibold text-gray-900 mb-1" }, "Evaluation Criteria"),
              React.createElement('p', { className: "text-gray-700" }, adjudication.evaluation_criteria)
            ),
            adjudication.technical_compliance && React.createElement('div', null,
              React.createElement('h6', { className: "font-semibold text-gray-900 mb-1" }, "Technical Compliance"),
              React.createElement('p', { className: "text-gray-700" }, adjudication.technical_compliance)
            ),
            adjudication.pricing_analysis && React.createElement('div', null,
              React.createElement('h6', { className: "font-semibold text-gray-900 mb-1" }, "Pricing Analysis"),
              React.createElement('p', { className: "text-gray-700 whitespace-pre-wrap" }, adjudication.pricing_analysis)
            ),
            adjudication.recommendation_rationale && React.createElement('div', null,
              React.createElement('h6', { className: "font-semibold text-gray-900 mb-1" }, "Recommendation Rationale"),
              React.createElement('p', { className: "text-gray-700" }, adjudication.recommendation_rationale)
            ),
            React.createElement('div', { className: "pt-4 border-t text-sm text-gray-600" },
              React.createElement('p', null, `Created by: ${adjudication.created_by_name}`),
              React.createElement('p', null, `Date: ${new Date(adjudication.created_at).toLocaleString()}`)
            )
          )
        )
      )
    )
  );
}
```

Installation Instructions

1. Open `frontend/app.js`
2. Find line 6090 (just before `// Mount the application using React 18 API`)
3. Paste the entire component code above
4. Save the file
5. Refresh the browser (Ctrl+F5)

The component provides:

- Requisition list for quotes/adjudication
- Upload up to 3 vendor quotes (PDF only)
- View/download uploaded quotes
- Create comprehensive adjudication summary with comparison
- View complete adjudication details
- Role-based access (Procurement uploads, Finance/MD view)