# Procurement Processing - Implementation Guide

**Date:** October 27, 2025
**Status:** ⚠ **Backend Complete - Frontend Needs Implementation**

---

## Requirements

Once a requisition is approved by HOD, it should move to Procurement. Procurement needs to:

1. ☑ **Add/Edit Unit Prices** - Enter unit price for each item
2. ☑ **Auto-calculate Total Cost** - Quantity × Unit Price = Total
3. ☑ **Select Vendor from Dropdown** - Choose vendor for each item
4. ☑ **Add Line Items** - Add new items to the requisition
5. ☑ **Remove Line Items** - Delete items from the requisition
6. ☑ **Edit Form Fields:**
    - Urgency (Low, Medium, High, Urgent)
    - Date Required
    - Delivery Location
7. ☑ **Complete Procurement** - Mark as done and move forward

---

## ☑ Backend Implementation (COMPLETE)

### New Endpoints Added:

#### 1. Update Requisition Form Fields

```
PUT /api/requisitions/:id/update-fields
Authorization: Bearer <token>
Content-Type: application/json

{
  "urgency": "High",
  "required_date": "2025-11-15",
  "delivery_location": "Lusaka"
}
```

**Authorization:** `procurement`, `admin`
**Response:**

```
{
  "success": true,
  "message": "Requisition fields updated successfully"
}
```

---

#### 2. Delete Line Item

```
DELETE /api/requisitions/:id/items/:item_id
Authorization: Bearer <token>
```

**Authorization:** `procurement`, `admin`
**Response:**

```
{
  "success": true,
  "message": "Item deleted successfully"
}
```

**Note:** Automatically recalculates grand total after deletion.

---

#### 3. Add Line Item (Already Exists)

```
POST /api/requisitions/:id/items
Authorization: Bearer <token>
Content-Type: application/json

{
  "item_name": "Mouse",
  "quantity": 10,
  "specifications": "Wireless, USB",
  "unit_price": 150.00,
  "vendor_id": 2
}
```

**Authorization:** `initiator`, `admin`, `procurement`
**Calculates:** `total_price = unit_price * quantity`

---

#### 4. Update Line Item (Already Exists)

```
PUT /api/requisitions/:id/items/:item_id
Authorization: Bearer <token>
Content-Type: application/json

{
  "item_name": "Laptop",
  "quantity": 5,
  "unit_price": 5500.00,
  "vendor_id": 1,
  "specifications": "Dell Latitude"
}
```

**Authorization:** `initiator`, `admin`, `procurement`, `hod`
**Calculates:** `total_price = unit_price * quantity`
**Updates:** Grand total automatically

---

#### 5. Complete Procurement (Already Exists)

```
PUT /api/requisitions/:id/procurement
Authorization: Bearer <token>
Content-Type: application/json

{
  "user_id": 3,
  "comments": "All vendors confirmed and pricing finalized"
}
```

**Authorization:** `procurement`, `admin`
**Changes Status:** `hod_approved` → `procurement_completed`

---

#### 6. Get Vendors List (Already Exists)

```
GET /api/vendors
Authorization: Bearer <token>
```

**Response:**

```
[
  { "id": 1, "name": "Tech Solutions Ltd", "contact": "..." },
  { "id": 2, "name": "Office Supplies Co", "contact": "..." },
  { "id": 3, "name": "Hardware Plus", "contact": "..." }
]
```

---

## ⚠ Frontend Implementation (NEEDED)

### Overview:

The frontend needs a **Procurement Processing Modal** that appears when procurement views an HOD-approved requisition.

### Modal Features:

1. **Editable Form Fields:**
   - Delivery Location (dropdown)
   - Urgency (dropdown)
   - Required Date (date picker)
2. **Line Items Table (Editable):**
   - Item Name
   - Quantity (editable)
   - Specifications (editable)
   - Unit Price (editable input field) ← **Key feature**
   - Vendor (dropdown selection) ← **Key feature**
   - Total Price (auto-calculated: quantity × unit price)
   - Delete button (✕) for each row
3. **Add Item Button:**
   - "+Add Line Item" button below table
   - Opens a form to add new item
4. **Grand Total Display:**
   - Sum of all line item totals
   - Updates automatically
5. **Complete Button:**
   - "Complete Procurement" button
   - Marks requisition as done

---

### Implementation Steps:

#### Step 1: Detect When to Show Procurement Mode

In the `RequisitionModal` component, check if:

- User role is `procurement`
- Requisition status is `hod_approved`

```
const isProcurementMode = user.role === 'procurement' && requisition.status === 'hod_approved';
```

---

**Step 2: Add State for Procurement**

```
const [vendors, setVendors] = useState([]);
const [editedReqFields, setEditedReqFields] = useState({
    urgency: requisition.urgency,
    required_date: requisition.required_date,
    delivery_location: requisition.delivery_location
});
const [procurementComments, setProcurementComments] = useState('');
```

---

**Step 3: Fetch Vendors on Modal Open**

```
useEffect(() => {
    if (isProcurementMode) {
        fetchVendors();
    }
}, [isProcurementMode]);

const fetchVendors = async () => {
    const token = localStorage.getItem('authToken');
    const response = await fetch(`${API_URL}/vendors`, {
        headers: { 'Authorization': `Bearer ${token}` }
    });
    const data = await response.json();
    setVendors(data);
};
```

---

**Step 4: Render Editable Form Fields**

```
{isProcurementMode && (
    <div className="mb-4">
        <h6>Requisition Details (Editable)</h6>
        <div className="row">
            <div className="col-md-4">
                <label>Delivery Location</label>
                <select
                    className="form-control"
                    value={editedReqFields.delivery_location}
                    onChange={(e) => setEditedReqFields({
                        ...editedReqFields,
                        delivery_location: e.target.value
                    })}
                >
                    <option value="Kitwe">Kitwe</option>
                    <option value="Lusaka">Lusaka</option>
                    <option value="Solwezi">Solwezi</option>
                </select>
            </div>

            <div className="col-md-4">
                <label>Urgency</label>
                <select
                    className="form-control"
                    value={editedReqFields.urgency}
                    onChange={(e) => setEditedReqFields({
                        ...editedReqFields,
                        urgency: e.target.value
                    })}
                >
                    <option value="Low">Low</option>
                    <option value="Medium">Medium</option>
                    <option value="High">High</option>
                    <option value="Urgent">Urgent</option>
                </select>
            </div>

            <div className="col-md-4">
                <label>Required Date</label>
                <input
                    type="date"
                    className="form-control"
                    value={editedReqFields.required_date}
                    onChange={(e) => setEditedReqFields({
                        ...editedReqFields,
                        required_date: e.target.value
                    })}
                />
            </div>
        </div>
        <button
            className="btn btn-sm btn-secondary mt-2"
            onClick={saveRequisitionFields}
        >
            Save Changes
        </button>
    </div>
)}
```

---

**Step 5: Render Editable Items Table**

```
<table className="table">
    <thead>
        <tr>
            <th>Item</th>
            <th>Qty</th>
            <th>Specifications</th>
            <th>Unit Price</th>
            <th>Vendor</th>
            <th>Total</th>
            {isProcurementMode && <th>Actions</th>}
        </tr>
    </thead>
    <tbody>
        {editedItems.map((item, index) => (
            <tr key={item.id}>
                <td>
                    {isProcurementMode ? (
                        <input
                            className="form-control form-control-sm"
                            value={item.item_name}
                            onChange={(e) => updateItemField(index, 'item_name', e.target.value)}
                        />
                    ) : item.item_name}
                </td>
```

```
                <td>
                    {isProcurementMode ? (
                        <input
                            type="number"
                            className="form-control form-control-sm"
                            style={{width: '80px'}}
                            value={item.quantity}
                            min="1"
                            onChange={(e) => updateItemField(index, 'quantity', e.target.value)}
                        />
                    ) : item.quantity}
                </td>

                <td>
                    {isProcurementMode ? (
                        <input
                            className="form-control form-control-sm"
                            value={item.specifications || ''}
                            onChange={(e) => updateItemField(index, 'specifications', e.target.value)}
                        />
                    ) : (item.specifications || '-')}
                </td>

                <td>
                    {isProcurementMode ? (
                        <input
                            type="number"
                            className="form-control form-control-sm"
                            style={{width: '100px'}}
                            value={item.unit_price || ''}
                            min="0"
                            step="0.01"
                            placeholder="0.00"
                            onChange={(e) => updateItemField(index, 'unit_price', e.target.value)}
                        />
                    ) : (item.unit_price ? `K${item.unit_price.toFixed(2)}` : '-')}
                </td>

                <td>
                    {isProcurementMode ? (
                        <select
                            className="form-control form-control-sm"
                            value={item.vendor_id || ''}
                            onChange={(e) => updateItemField(index, 'vendor_id', e.target.value)}
                        >
                            <option value="">Select Vendor</option>
                            {vendors.map(v => (
                                <option key={v.id} value={v.id}>{v.name}</option>
                            ))}
                        </select>
                    ) : (item.vendor_name || '-')}
                </td>

                <td>
                    {calculateItemTotal(item)}
                </td>

                {isProcurementMode && (
                    <td>
                        <button
                            className="btn btn-sm btn-danger"
                            onClick={() => deleteItem(item.id)}
                        >
                            ×
                        </button>
                    </td>
                )}
            </tr>
        ))}
    </tbody>
    <tfoot>
        <tr>
            <th colSpan={isProcurementMode ? 5 : 4} className="text-end">
                Grand Total:
            </th>
            <th>K{calculateGrandTotal().toFixed(2)}</th>
            {isProcurementMode && <th></th>}
        </tr>
    </tfoot>
</table>

{isProcurementMode && (
    <button className="btn btn-success btn-sm" onClick={addNewItem}>
        + Add Line Item
    </button>
```

```
)}
```

---

**Step 6: Helper Functions**

```
const updateItemField = (index, field, value) => {
    const updated = [...editedItems];
    updated[index][field] = value;

    // Auto-calculate total if quantity or unit_price changed
    if (field === 'quantity' || field === 'unit_price') {
        const qty = parseInt(updated[index].quantity) || 0;
        const price = parseFloat(updated[index].unit_price) || 0;
        updated[index].total_price = qty * price;
    }

    setEditedItems(updated);
};

const calculateItemTotal = (item) => {
    const qty = parseInt(item.quantity) || 0;
    const price = parseFloat(item.unit_price) || 0;
    return `K${(qty * price).toFixed(2)}`;
};

const calculateGrandTotal = () => {
    return editedItems.reduce((sum, item) => {
        const qty = parseInt(item.quantity) || 0;
        const price = parseFloat(item.unit_price) || 0;
        return sum + (qty * price);
    }, 0);
};

const saveItemChanges = async (item) => {
    const token = localStorage.getItem('authToken');
    const response = await fetch(`${API_URL}/requisitions/${requisition.id}/items/${item.id}`, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`
        },
        body: JSON.stringify({
            item_name: item.item_name,
            quantity: item.quantity,
            specifications: item.specifications,
            unit_price: item.unit_price,
            vendor_id: item.vendor_id
        })
    });

    if (response.ok) {
        alert('Item updated successfully');
        onRefresh();
    }
};

const deleteItem = async (itemId) => {
    if (!confirm('Are you sure you want to delete this item?')) return;

    const token = localStorage.getItem('authToken');
    const response = await fetch(`${API_URL}/requisitions/${requisition.id}/items/${itemId}`, {
        method: 'DELETE',
        headers: { 'Authorization': `Bearer ${token}` }
    });

    if (response.ok) {
        alert('Item deleted successfully');
        onRefresh();
    }
};

const addNewItem = async () => {
    const itemName = prompt('Enter item name:');
    if (!itemName) return;

    const quantity = prompt('Enter quantity:', '1');
    if (!quantity) return;

    const token = localStorage.getItem('authToken');
    const response = await fetch(`${API_URL}/requisitions/${requisition.id}/items`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`
        },
        body: JSON.stringify({
```

```
            item_name: itemName,
            quantity: parseInt(quantity),
            specifications: '',
            unit_price: 0,
            vendor_id: null
        })
    });

    if (response.ok) {
        alert('Item added successfully');
        onRefresh();
    }
};

const saveRequisitionFields = async () => {
    const token = localStorage.getItem('authToken');
    const response = await fetch(`${API_URL}/requisitions/${requisition.id}/update-fields`, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`
        },
        body: JSON.stringify(editedReqFields)
    });

    if (response.ok) {
        alert('Fields updated successfully');
        onRefresh();
    }
};
```

**Step 7: Complete Procurement Button**

```
{isProcurementMode && (
    <div className="mb-4">
        <h6>Procurement Comments</h6>
        <textarea
            className="form-control"
            rows="3"
            placeholder="Add comments about procurement process..."
            value={procurementComments}
            onChange={(e) => setProcurementComments(e.target.value)}
        />
    </div>
)}

// In modal footer:
{isProcurementMode && (
    <button
        className="btn btn-success"
        onClick={completeProcurement}
    >
        Complete Procurement
    </button>
)}
```

```
const completeProcurement = async () => {
    // Save all items first
    for (const item of editedItems) {
        await saveItemChanges(item);
    }

    // Save requisition fields
    await saveRequisitionFields();

    // Mark procurement as complete
    const token = localStorage.getItem('authToken');
    const response = await fetch(`${API_URL}/requisitions/${requisition.id}/procurement`, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`
        },
        body: JSON.stringify({
            user_id: user.id,
            comments: procurementComments
        })
    });

    if (response.ok) {
        alert('Procurement completed successfully!');
        onRefresh();
        onClose();
    }
};
```

## Testing Workflow

### Step 1: Create & Submit Requisition

```
Login: john.banda / password123
Create requisition with 2 items
Submit for approval
```

### Step 2: HOD Approves

```
Login: mary.mwanza / password123
Approve the requisition
Status changes: pending_hod → hod_approved
```

### Step 3: Procurement Processing

```
Login: james.phiri / password123
View the HOD-approved requisition
Should see:
  ☑ Editable form fields (urgency, date, location)
  ☑ Editable items table
  ☑ Unit price input fields
  ☑ Vendor dropdown for each item
  ☑ Delete button for each item
  ☑ "+ Add Line Item" button
  ☑ Auto-calculated totals
  ☑ "Complete Procurement" button
```

**Actions to Test:**

1. Change urgency from "High" to "Urgent"
2. Change delivery location
3. Enter unit price for Item 1: K5,500.00
4. Select vendor for Item 1: "Tech Solutions Ltd"
5. Total should auto-calculate: 5 × 5,500 = K27,500.00
6. Add unit price for Item 2: K150.00
7. Select vendor for Item 2: "Office Supplies Co"
8. Total should auto-calculate: 5 × 150 = K750.00
9. Grand Total: K27,500 + K750 = K28,250.00
10. Add new item: "USB Cable", Qty: 10, Price: K50, Vendor: "Hardware Plus"
11. Delete an item (test delete functionality)
12. Add procurement comments: "All items sourced and priced"
13. Click "Complete Procurement"

**Expected Result:**

- Status changes: hod_approved → procurement_completed
- All changes saved to database
- PDF can now be downloaded with all details

---

## Status Summary

| Feature | Backend | Frontend |
|---|---|---|
| Update form fields | ☑ Done | ⚠ Needed |
| Add unit prices | ☑ Done | ⚠ Needed |
| Auto-calculate totals | ☑ Done | ⚠ Needed |
| Select vendors | ☑ Done | ⚠ Needed |
| Add line items | ☑ Done | ⚠ Needed |
| Delete line items | ☑ Done | ⚠ Needed |
| Edit item details | ☑ Done | ⚠ Needed |
| Complete procurement | ☑ Done | ⚠ Needed |

---

## Files Modified

### Backend:

- `backend/server.js` - Added 2 new endpoints:
  - `PUT /api/requisitions/:id/update-fields`
  - `DELETE /api/requisitions/:id/items/:item_id`

### Frontend:

- `frontend/index.html` - **Needs modification:**
  - Add procurement mode detection
  - Add editable form fields
  - Add editable items table with pricing
  - Add vendor dropdowns
  - Add add/delete item buttons
  - Add complete procurement button

---

## Next Steps

1. **Implement Procurement Modal** in `frontend/index.html`
2. **Test End-to-End Workflow**
3. **Verify PDF Generation** includes all procurement details

**All backend APIs are ready and working!**

The frontend just needs to integrate these APIs into a user-friendly procurement processing interface.

---

**Last Updated:** October 27, 2025