# Tender Management Dashboard - Multi-Feature Component Guide

## 📋 Overview

A complete, production-ready multi-feature dashboard component library for React with Material UI. Each tab features three selectable options: **View Data**, **Add Form Entry**, and **Bulk Upload from Excel**.

## 🎯 Key Features

### Per-Tab Features

1. **Feature Selection Card** - Initial screen with 3 buttons
2. **View Data Table** - Display all records with search, sort, filter, export
3. **Form Entry** - Add new records with validation
4. **Bulk Upload** - Upload multiple records from Excel with template download

### Core Capabilities

✅ Dynamic mode switching (select → view/form/upload)
✅ Back button to return to selection
✅ Reusable components across all 7 forms
✅ Configuration-driven setup
✅ Excel template generation and validation
✅ Real-time search and filtering
✅ Pagination and sorting
✅ CSV export functionality
✅ Fully responsive design
✅ Loading states and error handling

## 📁 Project Structure

```
tender-dashboard/
├── src/
│   ├── components/
│   │   ├── common/
│   │   │   ├── Header.jsx
│   │   │   ├── LoadingSpinner.jsx
│   │   │   └── SnackbarNotification.jsx
│   │   ├── cards/
│   │   │   └── FeatureSelectionCard.jsx        # Selection card (View/Form/Upload)
│   │   ├── forms/
```

```
│   │   │   ├── FormField.jsx                    # Reusable field component
│   │   │   ├── LostDomesticLeadsForm.jsx
│   │   │   ├── DomesticOrderForm.jsx
│   │   │   ├── BudgetaryQuotationForm.jsx
│   │   │   ├── LeadSubmittedForm.jsx
│   │   │   ├── DomesticLeadsV2Form.jsx
│   │   │   ├── ExportLeadsForm.jsx
│   │   │   └── CRMLeadsForm.jsx
│   │   ├── tables/
│   │   │   ├── DataTable.jsx                    # Generic reusable table
│   │   │   ├── LostDomesticLeadsTable.jsx
│   │   │   ├── DomesticOrderTable.jsx
│   │   │   ├── BudgetaryQuotationTable.jsx
│   │   │   ├── LeadSubmittedTable.jsx
│   │   │   ├── DomesticLeadsV2Table.jsx
│   │   │   ├── ExportLeadsTable.jsx
│   │   │   └── CRMLeadsTable.jsx
│   │   ├── upload/
│   │   │   ├── ExcelUpload.jsx                  # Excel upload handler
│   │   │   └── excelUtils.js                    # Excel read/write utilities
│   │   └── tabs/
│   │       └── TabContent.jsx                   # Main content wrapper
│   ├── config/
│   │   ├── tabsConfig.js                        # Tab definitions
│   │   ├── fieldsConfig.js                      # Field specifications
│   │   └── constants.js                         # Enums and constants
│   ├── App.jsx                                 # Main application
│   └── index.js
├── public/
│   └── index.html
└── package.json
```

## 📋 Component Descriptions

### 1. FeatureSelectionCard.jsx

**Purpose:** Shows 3 interactive options when tab opens

**Features:**

- View Data button (shows table)
- Add New Entry button (shows form)
- Bulk Upload button (shows Excel upload)
- Hover animations
- Color-coded buttons
- Responsive grid layout

**Props:**

```
&lt;FeatureSelectionCard
  formName="Lost Domestic Leads"
  onSelectView={() =&gt; setMode('view')}
  onSelectForm={() =&gt; setMode('form')}
  onSelectUpload={() =&gt; setMode('upload')}
/&gt;
```

### 2. DataTable.jsx

**Purpose:** Generic reusable table for all data types

**Features:**

- Sortable columns
- Search/filter across all fields
- Pagination (5, 10, 25, 50 rows)
- CSV export
- View detail dialog
- Edit/Delete actions
- Column type handling (date, currency, status)
- Hover effects

**Props:**

```
&lt;DataTable
  data={data}
  columns={[
    { key: 'id', label: 'ID', sortable: true },
    { key: 'name', label: 'Name', sortable: true },
    { key: 'value', label: 'Value', type: 'currency' },
    { key: 'date', label: 'Date', type: 'date' }
  ]}
  onDelete={handleDelete}
  onEdit={handleEdit}
  title="Lost Domestic Leads"
/&gt;
```

### 3. ExcelUpload.jsx

**Purpose:** Handle bulk Excel file uploads

**Features:**

- Drag & drop file upload
- File type validation
- Size validation (5MB max)

- Sample template download
- Progress indication
- Error handling
- Success notification
- Field mapping validation

**Props:**

```
&lt;ExcelUpload
  fields={formFields}
  formType="lost-domestic-leads"
  title="Bulk Upload - Lost Domestic Leads"
  onUpload={handleBulkUpload}
/&gt;
```

### 4. excelUtils.js

**Exported Functions:**

- `readExcelFile(file)` - Parse Excel to JSON
- `validateExcelData(data, fields)` - Validate data
- `generateSampleExcel(fields, formType)` - Create template
- `exportToExcel(data, columns, filename)` - Export to Excel
- `exportToCSV(data, columns, filename)` - Export to CSV

### 5. TabContent.jsx

**Purpose:** Main wrapper managing all modes for a single tab

**Features:**

- State management for select/view/form/upload modes
- Back button to return to selection
- Loading indicator
- Component switching logic
- Refresh mechanism

**Props:**

```
&lt;TabContent
  formType="lost-domestic-leads"
  formName="Lost Domestic Leads"
  tableComponent={LostDomesticLeadsTable}
  formComponent={LostDomesticLeadsForm}
  tableColumns={columns}
  tableData={data}
```

```
  formFields={fields}
  onFormSubmit={handleSubmit}
  onTableDataDelete={handleDelete}
/&gt;
```

## 🔄 User Flow

```
Tab Opens
    ↓
FeatureSelectionCard Displayed (3 buttons)
    ├─ Click "View Data"
    │  ↓
    │  DataTable Opens
    │  ├─ Search/Filter/Sort/Export
    │  ├─ View Details
    │  ├─ Edit/Delete
    │  └─ Back Button → FeatureSelectionCard
    │
    ├─ Click "Add Entry"
    │  ↓
    │  Form Opens
    │  ├─ Fill Fields
    │  ├─ Validation
    │  ├─ Submit
    │  └─ Back Button → FeatureSelectionCard
    │
    └─ Click "Bulk Upload"
       ↓
       ExcelUpload Opens
       ├─ Download Template
       ├─ Drag/Drop or Select File
       ├─ Validate
       ├─ Process
       └─ Back Button → FeatureSelectionCard
```

## 📊 Excel Upload Workflow

### Template Generation

1. User clicks "Bulk Upload"
2. Component loads with "Download Template" button
3. User downloads pre-filled Excel with:
   - Column headers matching form fields
   - 1 sample row with example values
   - Proper formatting and data types

### File Upload

1. User fills Excel with data
2. Selects file via drag-drop or file picker
3. File validated:
    - Extension check (.xlsx, .xls)
    - Size check (max 5MB)
    - Headers validation
4. Data validated:
    - Required fields present
    - Data types match (text, number, date)
    - Format validation
5. Success/Error shown with details

### Data Integration

```
const handleBulkUpload = async (data) =&gt; {
  for (const record of data) {
    // Each record is a validated object
    await submitForm(record);
  }
  // Show success message
  // Refresh data table
};
```

### Component Styling

All components use Material UI with customizable themes:

**Color Scheme:**

- Primary: #667eea (Blue)
- Secondary: #764ba2 (Purple)
- Success: #66bb6a (Green)
- Warning: #ffa726 (Orange)
- Error: #ef5350 (Red)

**Responsive Breakpoints:**

- xs: 0px - 600px
- sm: 600px - 960px
- md: 960px - 1280px
- lg: 1280px - 1920px

- xl: 1920px+

### 🔗 Integration with Your Forms

#### Step 1: Configure Tab

```
// config/tabsConfig.js
{
  id: 1,
  label: 'Lost Domestic Leads',
  icon: 'TrendingDownIcon',
  formType: 'lost-domestic-leads'
}
```

#### Step 2: Create Table Wrapper

```
// components/tables/LostDomesticLeadsTable.jsx
import DataTable from './DataTable';

const LostDomesticLeadsTable = ({ data, columns, onDelete }) =&gt; {
  return (
    &lt;DataTable
      data={data}
      columns={LOST_DOMESTIC_LEADS_COLUMNS}
      onDelete={onDelete}
      title="Lost Domestic Leads"
    /&gt;
  );
};

const LOST_DOMESTIC_LEADS_COLUMNS = [
  { key: 'serialNumber', label: 'Serial Number' },
  { key: 'tenderName', label: 'Tender Name' },
  { key: 'customer', label: 'Customer' },
  { key: 'valueWithoutGST', label: 'Value (ex. GST)', type: 'currency' },
  { key: 'valueWithGST', label: 'Value (inc. GST)', type: 'currency' },
  { key: 'reasonLosing', label: 'Reason' },
  { key: 'createdAt', label: 'Created', type: 'date' }
];
```

#### Step 3: Use in App

```
&lt;TabContent
  formType="lost-domestic-leads"
  formName="Lost Domestic Leads"
  tableComponent={LostDomesticLeadsTable}
  formComponent={LostDomesticLeadsForm}
  tableColumns={COLUMNS}
  tableData={data}
  formFields={fields}
```

```
  onFormSubmit={handleSubmit}
  onTableDataDelete={handleDelete}
/&gt;
```

## 🔌 API Integration Points

### Fetch Data (View Mode)

```
GET /api/lost-domestic-leads
Response: [{ id, serialNumber, tenderName, ... }, ...]
```

### Submit Form (Add Mode)

```
POST /api/lost-domestic-leads
Body: { serialNumber, tenderName, customer, ... }
Response: { success: true, data: { id, ... } }
```

### Bulk Upload (Upload Mode)

```
POST /api/lost-domestic-leads/bulk-upload
Body: [{ serialNumber, tenderName, ... }, ...]
Response: { success: true, count: 10 }
```

### Delete Record

```
DELETE /api/lost-domestic-leads/:id
Response: { success: true }
```

## 📊 Table Column Configuration

```
const columns = [
  {
    key: 'fieldKey',           // Object key
    label: 'Display Label',    // Header text
    type: 'text',              // 'text', 'date', 'currency', 'status'
    sortable: true,            // Enable sorting
    width: '15%'               // Optional width
  }
];
```

### ⚙ Form Fields Configuration

```
const formFields = [
  {
    key: 'serialNumber',
    label: 'Serial Number',
    type: 'text',
    required: true,
    validation: /^[A-Z]{2}-\d{3}$/
  },
  {
    key: 'valueWithGST',
    label: 'Value (with GST)',
    type: 'currency',
    required: true
  },
  {
    key: 'submittedDate',
    label: 'Submission Date',
    type: 'date',
    required: false
  },
  {
    key: 'status',
    label: 'Status',
    type: 'select',
    options: ['Draft', 'Submitted', 'Won', 'Lost']
  }
];
```

### Installation & Setup

#### Dependencies

```
npm install react react-dom @mui/material @emotion/react @emotion/styled
npm install react-hook-form
npm install xlsx (for Excel support)
npm install @mui/icons-material
```

#### Quick Start

```
# 1. Create React app
npx create-react-app tender-dashboard

# 2. Install dependencies
npm install @mui/material @emotion/react @emotion/styled react-hook-form xlsx @mui/icons-

# 3. Copy component files
cp -r components/ src/
cp -r config/ src/
```

```
# 4. Update App.jsx with provided example

# 5. Start development
npm start
```

## 🎨 Customization Guide

### Custom Table Styling

```
&lt;DataTable
  sx={{
    tableHeader: { backgroundColor: '#f5f5f5' },
    tableRow: { '&amp;:hover': { backgroundColor: '#fafafa' } }
  }}
/&gt;
```

### Custom Upload Handler

```
const customUploadHandler = async (validatedData) =&gt; {
  // Custom logic before submission
  const enrichedData = validatedData.map(item =&gt; ({
    ...item,
    importedAt: new Date(),
    status: 'pending'
  }));

  // Send to API
  const response = await fetch('/api/bulk-import', {
    method: 'POST',
    body: JSON.stringify(enrichedData)
  });
};
```

### Custom Column Formatter

```
const columns = [
  {
    key: 'value',
    label: 'Value',
    type: 'currency',
    formatter: (value) =&gt; `₹${(value / 10000000).toFixed(2)}Cr` // In crores
  }
];
```

## ✨ Advanced Features

### Search Persistence

```
const [searchState, setSearchState] = useState({
  tab: 0,
  query: ''
});

// Auto-restore search on tab change
```

### Export Filters

```
// Export only visible/filtered results
const handleExportVisible = () =&gt; {
  exportToCSV(filteredData, columns);
};
```

### Batch Operations

```
// Select multiple rows for bulk actions
const [selectedRows, setSelectedRows] = useState([]);

const handleBatchDelete = () =&gt; {
  selectedRows.forEach(id =&gt; deleteRecord(id));
};
```

## 🐛 Troubleshooting

| Issue | Solution |
| --- | --- |
| Excel not reading | Ensure file is .xlsx/.xls format |
| Validation errors | Check field types match in columns |
| Slow table | Implement virtual scrolling for large datasets |
| Memory leak | Clear data on component unmount |
| Export blank | Ensure columns have `key` property |

## 🧪 Testing

```
// Example test
describe('DataTable', () =&gt; {
  it('should filter data correctly', () =&gt; {
    const { getByPlaceholderText } = render(
      &lt;DataTable data={mockData} columns={columns} /&gt;
```

```
    );

    const searchInput = getByPlaceholderText('Search');
    fireEvent.change(searchInput, { target: { value: 'test' } });

    expect(mockData.filter(...)).toHaveLength(1);
  });
});
```

## 🎯 Deliverables

✅ **FeatureSelectionCard.jsx** - 3-button selection interface
✅ **DataTable.jsx** - Generic table with all features
✅ **ExcelUpload.jsx** - Complete upload handler
✅ **excelUtils.js** - Excel read/write functions
✅ **TabContent.jsx** - Mode management wrapper
✅ **tabsConfig.js** - Configuration file
✅ **App-example.jsx** - Full integration example
✅ **Component Architecture Guide** - File structure
✅ **This PDF** - Complete documentation

## 🚀 Next Steps

1. ✅ Copy components to your project
2. ✅ Configure tabs in `tabsConfig.js`
3. ✅ Create table wrappers for each form type
4. ✅ Connect to your API endpoints
5. ✅ Test all features (view, form, upload)
6. ✅ Customize styling as needed
7. ✅ Deploy to production

## 📞 Support

For questions or issues:

- Review component props in JSDoc comments
- Check config files for structure
- Refer to Material UI documentation
- Test with mock data first

**Status:** Production-Ready ✅
**Version:** 1.0.0
**Last Updated:** December 1, 2025