

TWIST ERP - Complete Frontend UI Implementation Guide

React Component Library & User Interface

Version: 1.0

Date: October 2025

Framework: React 18 + Ant Design 5 + Vite

1. Frontend Architecture

1.1 Technology Stack

- **Framework:** React 18.2+
- **UI Library:** Ant Design 5.12+
- **Build Tool:** Vite 5.0+
- **State Management:** Zustand + React Query
- **Routing:** React Router v6
- **Charts:** ECharts + Recharts
- **Drag & Drop:** React Beautiful DnD
- **Forms:** React Hook Form + Yup
- **HTTP Client:** Axios

1.2 Folder Structure

```
frontend/
├── public/
│   ├── index.html
│   └── assets/
├── src/
│   ├── components/           # Reusable components
│   │   ├── AIAssistant/
│   │   ├── Common/
│   │   ├── Finance/
│   │   ├── Inventory/
│   │   ├── Sales/
│   │   └── ...
│   ├── layouts/              # Layout components
│   │   ├── MainLayout.jsx
│   │   ├── AuthLayout.jsx
│   │   └── DashboardLayout.jsx
│   ├── pages/                # Page components
│   │   └── Dashboard/
```

```

├── Finance/
├── Inventory/
├── Sales/
├── Procurement/
├── FormBuilder/
├── Workflows/
├── Assets/
├── Budgeting/
├── HR/
├── Projects/
├── contexts/           # React contexts
├──   ├── AuthContext.jsx
├──   ├── CompanyContext.jsx
├──   └── ThemeContext.jsx
├── hooks/              # Custom hooks
├──   ├── useApi.js
├──   ├── useAuth.js
├──   └── useCompany.js
├── services/           # API services
├──   ├── api.js
├──   ├── auth.service.js
├──   ├── finance.service.js
├──   └── ...
├── utils/              # Utility functions
├──   ├── formatters.js
├──   ├── validators.js
├──   └── helpers.js
├── styles/             # Global styles
├──   ├── index.css
├──   └── variables.css
├── App.jsx             # Main app component
├── main.jsx            # Entry point
├── vite.config.js      # Vite configuration
├── package.json
└── README.md

```

2. Core Layouts

2.1 MainLayout.jsx

```

import React, { useState } from 'react';
import { Layout, Menu, Avatar, Dropdown, Badge, Space } from 'antd';
import {
  DashboardOutlined,
  DollarOutlined,
  ShoppingOutlined,
  TeamOutlined,
  ShopOutlined,
  SettingOutlined,
  BellOutlined,
  UserOutlined,
  LogoutOutlined,
  SwapOutlined,

```

```

AppstoreOutlined,
FormOutlined,
BranchesOutlined,
RobotOutlined,
BarChartOutlined,
ProjectOutlined,
} from '@ant-design/icons';
import { useNavigate, useLocation } from 'react-router-dom';
import { useAuth } from '../contexts/AuthContext';
import { useCompany } from '../contexts/CompanyContext';
import CompanySelector from '../components/Common/CompanySelector';

const { Header, Sider, Content, Footer } = Layout;

const MainLayout = ({ children }) => {
  const [collapsed, setCollapsed] = useState(false);
  const navigate = useNavigate();
  const location = useLocation();
  const { user, logout } = useAuth();
  const { currentCompany } = useCompany();

  const menuItems = [
    {
      key: '/',
      icon: <DashboardOutlined />,
      label: 'Dashboard',
    },
    {
      key: 'finance',
      icon: <DollarOutlined />,
      label: 'Finance',
      children: [
        { key: '/finance/accounts', label: 'Chart of Accounts' },
        { key: '/finance/journals', label: 'Journal Vouchers' },
        { key: '/finance/invoices', label: 'Invoices' },
        { key: '/finance/payments', label: 'Payments' },
      ],
    },
    {
      key: 'inventory',
      icon: <AppstoreOutlined />,
      label: 'Inventory',
      children: [
        { key: '/inventory/products', label: 'Products' },
        { key: '/inventory/warehouses', label: 'Warehouses' },
        { key: '/inventory/movements', label: 'Stock Movements' },
      ],
    },
    {
      key: 'sales',
      icon: <ShoppingOutlined />,
      label: 'Sales & CRM',
      children: [
        { key: '/sales/customers', label: 'Customers' },
        { key: '/sales/orders', label: 'Sales Orders' },
        { key: '/sales/pipeline', label: 'Sales Pipeline' },
      ],
    },
  ],

```

```

    ],
  },
  {
    key: 'procurement',
    icon: <ShopOutlined />,
    label: 'Procurement',
    children: [
      { key: '/procurement/suppliers', label: 'Suppliers' },
      { key: '/procurement/orders', label: 'Purchase Orders' },
    ],
  },
  {
    key: 'nocode',
    icon: <FormOutlined />,
    label: 'No-Code Tools',
    children: [
      { key: '/forms', label: 'Form Builder' },
      { key: '/workflows', label: 'Workflows' },
      { key: '/migration', label: 'Data Migration' },
    ],
  },
  {
    key: 'advanced',
    icon: <BarChartOutlined />,
    label: 'Advanced',
    children: [
      { key: '/assets', label: 'Asset Management' },
      { key: '/budgets', label: 'Budgeting' },
      { key: '/hr/employees', label: 'HR & Payroll' },
      { key: '/projects', label: 'Projects' },
    ],
  },
];

const userMenuItems = [
  {
    key: 'profile',
    icon: <UserOutlined />,
    label: 'Profile',
    onClick: () => navigate('/settings/profile'),
  },
  {
    key: 'settings',
    icon: <SettingOutlined />,
    label: 'Settings',
    onClick: () => navigate('/settings'),
  },
  {
    type: 'divider',
  },
  {
    key: 'logout',
    icon: <LogoutOutlined />,
    label: 'Logout',
    onClick: logout,
  },
];

```

```

];

const handleMenuClick = ({ key }) => {
  navigate(key);
};

return (
  <Layout style={{ minHeight: '100vh' }}>
    <Sider
      collapsible
      collapsed={collapsed}
      onCollapse={setCollapsed}
      theme="light"
      width={250}
      style={{
        overflow: 'auto',
        height: '100vh',
        position: 'fixed',
        left: 0,
        top: 0,
        bottom: 0,
      }}
    >
      <div>
        {collapsed ? 'TE' : 'TWIST ERP'}
      </div>

      <Menu
        mode="inline"
        selectedKeys={[location.pathname]}
        items={menuItems}
        onClick={handleMenuClick}
      />
    </Sider>

    <Layout style={{ marginLeft: collapsed ? 80 : 250 }}>
      <Header
        style={{
          background: '#fff',
          padding: '0 24px',
          display: 'flex',
          justifyContent: 'space-between',
          alignItems: 'center',
          position: 'sticky',
          top: 0,
          zIndex: 100,
          boxShadow: '0 2px 8px rgba(0,0,0,0.1)',
        }}
      >
        <div>
          <CompanySelector />
        </div>

        <Space size="large">
          <Badge count={5}>
            <BellOutlined style={{ fontSize: 20 }} />
          </Badge>
        </Space>
      </Header>
    </Layout>
  </Layout>
);

```

```

        </Badge>;

        <Dropdown menu={{ items: userMenuItems }} placement="bottomRight">
          <Space style={{ cursor: 'pointer' }}>
            <Avatar icon={<UserOutlined />} />
            <span>{user?.name}</span>
          </Space>
        </Dropdown>
      </Space>
    </Header>

    <Content style={{ margin: '24px 16px', padding: 24, background: '#f0f2f5' }}>
      {children}
    </Content>

    <Footer style={{ textAlign: 'center' }}>
      TWIST ERP ©2025 - Transform, Integrate, Simplify, Track
    </Footer>
  </Layout>
</Layout>
);
};

export default MainLayout;

```

3. Dashboard Components

3.1 Dashboard.jsx

```

import React, { useState, useEffect } from 'react';
import { Row, Col, Card, Statistic, Progress, Table, Tag } from 'antd';
import {
  ArrowUpOutlined,
  ArrowDownOutlined,
  DollarOutlined,
  ShoppingCartOutlined,
  TeamOutlined,
  WarningOutlined,
} from '@ant-design/icons';
import { Line, Column, Pie } from '@ant-design/charts';
import { useCompany } from '../../contexts/CompanyContext';
import api from '../../services/api';

const Dashboard = () => {
  const { currentCompany } = useCompany();
  const [loading, setLoading] = useState(true);
  const [stats, setStats] = useState({});
  const [salesData, setSalesData] = useState([]);
  const [revenueData, setRevenueData] = useState([]);

  useEffect(() => {
    loadDashboardData();
  }, [currentCompany]);

```

```

const loadDashboardData = async () => {
  try {
    setLoading(true);
    const response = await api.get('/api/v1/dashboard/summary/');
    setStats(response.data.stats);
    setSalesData(response.data.sales_trend);
    setRevenueData(response.data.revenue_breakdown);
  } catch (error) {
    console.error('Failed to load dashboard:', error);
  } finally {
    setLoading(false);
  }
};

```

```

const salesConfig = {
  data: salesData,
  xField: 'month',
  yField: 'amount',
  smooth: true,
  color: '#1890ff',
};

```

```

const revenueConfig = {
  data: revenueData,
  angleField: 'value',
  colorField: 'category',
  radius: 0.8,
  label: {
    type: 'outer',
    content: '{name} {percentage}',
  },
};

```

```

// Recent activities mock data
const recentActivities = [
  {
    key: '1',
    action: 'Sales Order Created',
    user: 'John Doe',
    time: '2 minutes ago',
    status: 'success',
  },
  {
    key: '2',
    action: 'Invoice Generated',
    user: 'Jane Smith',
    time: '15 minutes ago',
    status: 'success',
  },
  {
    key: '3',
    action: 'Payment Received',
    user: 'System',
    time: '1 hour ago',
    status: 'success',
  },
];

```

```

    },
  ];

  const columns = [
    {
      title: 'Action',
      dataIndex: 'action',
      key: 'action',
    },
    {
      title: 'User',
      dataIndex: 'user',
      key: 'user',
    },
    {
      title: 'Time',
      dataIndex: 'time',
      key: 'time',
    },
    {
      title: 'Status',
      dataIndex: 'status',
      key: 'status',
      render: (status) => (
        <Tag color={status === 'success' ? 'green' : 'red'}>
          {status.toUpperCase()}
        </Tag>
      ),
    },
  ];

  return (
    <div>
      <h1>Dashboard - {currentCompany?.name}</h1>

      {/* KPI Cards */}
      <Row gutter={[16, 16]} style={{ marginTop: 24 }}>
        <Col xs={24} sm={12} lg={6}>
          <Card>
            <Statistic
              title="Total Revenue"
              value={stats.revenue || 0}
              precision={2}
              valueStyle={{ color: '#3f8600' }}
              prefix={<DollarOutlined />}
              suffix="BDT"
            />
            <div>
              <ArrowUpOutlined style={{ color: '#3f8600' }} /> 15% vs last month
            </div>
          </Card>
        </Col>

        <Col xs={24} sm={12} lg={6}>
          <Card>
            <Statistic

```

```

        title="Sales Orders"
        value={stats.orders || 0}
        valueStyle={{ color: '#1890ff' }}
        prefix={<ShoppingCartOutlined />}
      />
    <div>
      <ArrowUpOutlined style={{ color: '#3f8600' }} /> 8% vs last month
    </div>
    </Card>
  </Col>

  <Col xs={24} sm={12} lg={6}>
    <Card>
      <Statistic
        title="Customers"
        value={stats.customers || 0}
        valueStyle={{ color: '#722ed1' }}
        prefix={<TeamOutlined />}
      />
    <div>
      <ArrowUpOutlined style={{ color: '#3f8600' }} /> 12 new this month
    </div>
    </Card>
  </Col>

  <Col xs={24} sm={12} lg={6}>
    <Card>
      <Statistic
        title="Pending Approvals"
        value={stats.approvals || 0}
        valueStyle={{ color: '#cf1322' }}
        prefix={<WarningOutlined />}
      />
    <div>
      Requires attention
    </div>
    </Card>
  </Col>
</Row>

/* Charts */
<Row gutter={[16, 16]} style={{ marginTop: 24 }}>
  <Col xs={24} lg={16}>
    <Card title="Sales Trend (Last 12 Months)" loading={loading}>
      <Line {...salesConfig} />
    </Card>
  </Col>

  <Col xs={24} lg={8}>
    <Card title="Revenue Breakdown" loading={loading}>
      <Pie {...revenueConfig} />
    </Card>
  </Col>
</Row>

/* Budget Progress */

```

```

<Row gutter=[[16, 16]] style={{ marginTop: 24 }}>
  <Col xs={24} lg={12}>
    <Card title="Budget Consumption">
      <div>
        <div>
          <span>Sales Budget: </span>
          <span>
            75% Used
          </span>
        </div>
        <Progress percent={75} status="active" />
      </div>

      <div>
        <div>
          <span>Operational Budget: </span>
          <span>
            45% Used
          </span>
        </div>
        <Progress percent={45} />
      </div>

      <div>
        <div>
          <span>Marketing Budget: </span>
          <span>
            92% Used
          </span>
        </div>
        <Progress percent={92} status="exception" />
      </div>
    </Card>
  </Col>

  <Col xs={24} lg={12}>
    <Card title="Recent Activities">
      <Table
        dataSource={recentActivities}
        columns={columns}
        pagination={false}
        size="small"
      />
    </Card>
  </Col>
</Row>
);
};

export default Dashboard;

```

4. Finance Module UI

4.1 Chart of Accounts (AccountsList.jsx)

```
import React, { useState, useEffect } from 'react';
import {
  Card,
  Table,
  Button,
  Space,
  Modal,
  Form,
  Input,
  Select,
  Tag,
  message,
} from 'antd';
import { PlusOutlined, EditOutlined, DeleteOutlined } from '@ant-design/icons';
import api from '../../services/api';

const AccountsList = () => {
  const [accounts, setAccounts] = useState([]);
  const [loading, setLoading] = useState(false);
  const [modalVisible, setModalVisible] = useState(false);
  const [editingAccount, setEditingAccount] = useState(null);
  const [form] = Form.useForm();

  useEffect(() => {
    loadAccounts();
  }, []);

  const loadAccounts = async () => {
    try {
      setLoading(true);
      const response = await api.get('/api/v1/finance/accounts/');
      setAccounts(response.data.results);
    } catch (error) {
      message.error('Failed to load accounts');
    } finally {
      setLoading(false);
    }
  };

  const handleCreate = () => {
    form.resetFields();
    setEditingAccount(null);
    setModalVisible(true);
  };

  const handleEdit = (record) => {
    form.setFieldsValue(record);
    setEditingAccount(record);
    setModalVisible(true);
  };
};
```

```

const handleDelete = async (id) => {
  try {
    await api.delete(`/api/v1/finance/accounts/${id}/`);
    message.success('Account deleted');
    loadAccounts();
  } catch (error) {
    message.error('Failed to delete account');
  }
};

const handleSubmit = async (values) => {
  try {
    if (editingAccount) {
      await api.put(`/api/v1/finance/accounts/${editingAccount.id}/`, values);
      message.success('Account updated');
    } else {
      await api.post('/api/v1/finance/accounts/', values);
      message.success('Account created');
    }
    setModalVisible(false);
    loadAccounts();
  } catch (error) {
    message.error('Operation failed');
  }
};

const columns = [
  {
    title: 'Code',
    dataIndex: 'code',
    key: 'code',
    width: 100,
  },
  {
    title: 'Name',
    dataIndex: 'name',
    key: 'name',
  },
  {
    title: 'Type',
    dataIndex: 'account_type',
    key: 'account_type',
    render: (type) => {
      const colors = {
        ASSET: 'blue',
        LIABILITY: 'orange',
        EQUITY: 'purple',
        REVENUE: 'green',
        EXPENSE: 'red',
      };
      return <Tag color={colors[type]}>{type}</Tag>;
    },
  },
  {
    title: 'Balance',
    dataIndex: 'current_balance',
  }
];

```

```

      key: 'current_balance',
      align: 'right',
      render: (value) => `₹ ${parseFloat(value || 0).toLocaleString()}`,
    },
    {
      title: 'Status',
      dataIndex: 'is_active',
      key: 'is_active',
      render: (active) => (
        <Tag color={active ? 'green' : 'red'}>
          {active ? 'Active' : 'Inactive'}
        </Tag>
      ),
    },
  ],
  {
    title: 'Actions',
    key: 'actions',
    render: (_, record) => (
      <Space>
        <Button
          type="link"
          icon={<EditOutlined />}
          onClick={() => handleEdit(record)}
        />
        <Button
          type="link"
          danger
          icon={<DeleteOutlined />}
          onClick={() => handleDelete(record.id)}
        />
      </Space>
    ),
  },
],
];

return (
  <div>
    <Card
      title="Chart of Accounts"
      extra={
        <Button
          type="primary"
          icon={<PlusOutlined />}
          onClick={handleCreate}
        />
        New Account
      </Button>
    >
    <Table
      dataSource={accounts}
      columns={columns}
      loading={loading}
      rowKey="id"
      pagination={{ pageSize: 20 }}
    />
  </div>
);

```

```

</Card>;

<Modal
  title={editingAccount ? 'Edit Account' : 'Create Account'}
  open={modalVisible}
  onOk={() => form.submit()}
  onCancel={() => setModalVisible(false)}
  width={600}
>;
  <Form form={form} layout="vertical" onFinish={handleSubmit}>;
    <Form.Item
      name="code"
      label="Account Code"
      rules={[{ required: true, message: 'Please enter account code' }]}
    >;
      <Input placeholder="e.g., 1000" />;
    </Form.Item>;

    <Form.Item
      name="name"
      label="Account Name"
      rules={[{ required: true, message: 'Please enter account name' }]}
    >;
      <Input placeholder="e.g., Cash on Hand" />;
    </Form.Item>;

    <Form.Item
      name="account_type"
      label="Account Type"
      rules={[{ required: true }]}
    >;
      <Select>;
        <Select.Option value="ASSET">Asset</Select.Option>;
        <Select.Option value="LIABILITY">Liability</Select.Option>;
        <Select.Option value="EQUITY">Equity</Select.Option>;
        <Select.Option value="REVENUE">Revenue</Select.Option>;
        <Select.Option value="EXPENSE">Expense</Select.Option>;
      </Select>;
    </Form.Item>;

    <Form.Item name="currency" label="Currency" initialValue="BDT">;
      <Select>;
        <Select.Option value="BDT">BDT - Bangladeshi Taka</Select.Option>;
        <Select.Option value="USD">USD - US Dollar</Select.Option>;
        <Select.Option value="EUR">EUR - Euro</Select.Option>;
      </Select>;
    </Form.Item>;

    <Form.Item
      name="is_active"
      label="Active"
      valuePropName="checked"
      initialValue={true}
    >;
      <Select>;
        <Select.Option value={true}>Active</Select.Option>;

```

```

        <Select.Option value={false}>Inactive</Select.Option>
      </Select>
    </Form.Item>
  </Form>
  </Modal>
</div>
);
};

export default AccountsList;

```

5. Form Builder UI (Drag & Drop)

5.1 FormBuilder.jsx

```

import React, { useState } from 'react';
import { Card, Row, Col, Button, Input, Select, Form, Space, message } from 'antd';
import { DragDropContext, Droppable, Draggable } from 'react-beautiful-dnd';
import {
  PlusOutlined,
  SaveOutlined,
  EyeOutlined,
  DeleteOutlined,
} from '@ant-design/icons';

const fieldTypes = [
  { type: 'text', label: 'Text Input', icon: '📄' },
  { type: 'number', label: 'Number', icon: '🔢' },
  { type: 'date', label: 'Date', icon: '📅' },
  { type: 'select', label: 'Dropdown', icon: '▼' },
  { type: 'checkbox', label: 'Checkbox', icon: '☑' },
  { type: 'textarea', label: 'Text Area', icon: '📝' },
  { type: 'email', label: 'Email', icon: '✉' },
  { type: 'phone', label: 'Phone', icon: '☎' },
];

const FormBuilder = () => {
  const [formFields, setFormFields] = useState([]);
  const [formName, setFormName] = useState('');
  const [selectedField, setSelectedField] = useState(null);

  const handleDragEnd = (result) => {
    if (!result.destination) return;

    const { source, destination } = result;

    // Adding from palette
    if (source.droppableId === 'palette' && destination.droppableId === 'canvas')
      const fieldType = fieldTypes[source.index];
      const newField = {
        id: `field-${Date.now()}`,
        ...fieldType,
        label: fieldType.label,

```

```

        required: false,
        placeholder: '',
    };
    const newFields = [...formFields];
    newFields.splice(destination.index, 0, newField);
    setFormFields(newFields);
}

// Reordering within canvas
if (source.droppableId === 'canvas' && destination.droppableId === 'canvas')
    const newFields = Array.from(formFields);
    const [removed] = newFields.splice(source.index, 1);
    newFields.splice(destination.index, 0, removed);
    setFormFields(newFields);
}
};

const handleFieldClick = (field) => {
    setSelectedField(field);
};

const handleFieldUpdate = (updatedField) => {
    setFormFields(
        formFields.map((f) => (f.id === updatedField.id ? updatedField : f))
    );
    setSelectedField(updatedField);
};

const handleFieldDelete = (fieldId) => {
    setFormFields(formFields.filter((f) => f.id !== fieldId));
    if (selectedField?.id === fieldId) {
        setSelectedField(null);
    }
};

const handleSave = async () => {
    if (!formName) {
        message.error('Please enter form name');
        return;
    }

    const formData = {
        name: formName,
        schema: {
            fields: formFields,
        },
    };

    try {
        // await api.post('/api/v1/forms/', formData);
        message.success('Form saved successfully');
    } catch (error) {
        message.error('Failed to save form');
    }
};

```

```

return (
  <div>
    &lt;Card
      title={
        &lt;Input
          placeholder="Form Name"
          value={formName}
          onChange={(e) => setFormName(e.target.value)}
          style={{ width: 300 }}
        /&gt;
      }
      extra={
        &lt;Space&gt;
        &lt;Button icon={&lt;EyeOutlined /&gt;}&gt;Preview&lt;/Button&gt;
        &lt;Button type="primary" icon={&lt;SaveOutlined /&gt;} onClick={handleSave}&gt;
          Save Form
        &lt;/Button&gt;
        &lt;/Space&gt;
      }
    &gt;
    &lt;DragDropContext onDragEnd={handleDragEnd}&gt;
    &lt;Row gutter={16}&gt;
      { /* Field Palette */ }
    &lt;Col span={6}&gt;
      &lt;Card title="Field Types" size="small"&gt;
        &lt;Draggable droppableId="palette" isDropDisabled={false}&gt;
          {(provided) => (
            <div>
              {fieldTypes.map((field, index) => (
                &lt;Draggable
                  key={field.type}
                  draggableId={field.type}
                  index={index}
                &gt;
                  {(provided, snapshot) => (
                    <div>
                      <span>{field.icon}</span>
                      {field.label}
                    </div>
                  )}
                &lt;/Draggable&gt;
              )}
            &lt;/Draggable&gt;
          )}
          {provided.placeholder}
        </div>
      )}
    &lt;/Draggable&gt;
    &lt;/Card&gt;
    &lt;/Col&gt;

    { /* Form Canvas */ }
    &lt;Col span={12}&gt;
      &lt;Card title="Form Canvas" size="small" style={{ minHeight: 500 }}&gt;
        &lt;Draggable droppableId="canvas"&gt;
          {(provided, snapshot) => (
            <div>
              {formFields.length === 0 & & (

```

```

    <div>
      Drag fields here to build your form
    </div>
  )}

  {formFields.map((field, index) => (
    <Draggable key={field.id} draggableId={field.id} index={index}>
      {(provided, snapshot) => (
        <div> handleFieldClick(field)</div>
        style={{
          padding: '12px',
          marginBottom: '12px',
          background: '#fff',
          border: `2px solid ${
            selectedField?.id === field.id
              ? '#1890ff'
              : '#d9d9d9'
          }`,
          borderRadius: '4px',
          cursor: 'pointer',
          ...provided.draggableProps.style,
        }}
        <div>
          <div>
            <strong>{field.label}</strong>
            {field.required & 'Required'} (
              <span> *</span>
            )
          </div>
          <Button
            type="text"
            danger
            size="small"
            icon={<DeleteOutlined />}
            onClick={e => {
              e.stopPropagation();
              handleFieldDelete(field.id);
            }}
          />
        </div>
        <div>
          {field.placeholder || `Enter ${field.label}`}
        </div>
      </div>
    )}
    </Draggable>
  )}
  {provided.placeholder}
</div>
)}
</Droppable>
</Card>
</Col>

{ /* Field Properties */}

```

```

        <Col span={6}>
          <Card title="Field Properties" size="small">
            {selectedField ? (
              <Form layout="vertical">
                <Form.Item label="Label">
                  <Input
                    value={selectedField.label}
                    onChange={e =>
                      handleFieldUpdate({
                        ...selectedField,
                        label: e.target.value,
                      })
                    }
                  />
                </Form.Item>

                <Form.Item label="Placeholder">
                  <Input
                    value={selectedField.placeholder}
                    onChange={e =>
                      handleFieldUpdate({
                        ...selectedField,
                        placeholder: e.target.value,
                      })
                    }
                  />
                </Form.Item>

                <Form.Item label="Required">
                  <Select
                    value={selectedField.required}
                    onChange={value =>
                      handleFieldUpdate({
                        ...selectedField,
                        required: value,
                      })
                    }
                  >
                    <Select.Option value={true}>Yes</Select.Option>
                    <Select.Option value={false}>No</Select.Option>
                  </Select>
                </Form.Item>
              </Form>
            ) : (
              <div>
                Click a field to edit properties
              </div>
            )}
          </Card>
        </Col>
      </Row>
    </DragDropContext>
  </Card>
</div>
);
};

```

```
export default FormBuilder;
```

6. AI Assistant Widget (Always Visible)

6.1 AIWidget.jsx

```
import React, { useState, useEffect, useRef } from 'react';
import { Button, Drawer, Input, List, Avatar, Badge, Spin } from 'antd';
import { RobotOutlined, SendOutlined, CloseOutlined } from '@ant-design/icons';
import api from '../../services/api';

const AIWidget = () => {
  const [visible, setVisible] = useState(false);
  const [messages, setMessages] = useState([]);
  const [input, setInput] = useState('');
  const [loading, setLoading] = useState(false);
  const [conversationId, setConversationId] = useState(null);
  const [unreadAlerts, setUnreadAlerts] = useState(0);
  const messagesEndRef = useRef(null);

  useEffect(() => {
    loadUnreadAlerts();
    const interval = setInterval(loadUnreadAlerts, 60000);
    return () => clearInterval(interval);
  }, []);

  useEffect(() => {
    scrollToBottom();
  }, [messages]);

  const loadUnreadAlerts = async () => {
    try {
      const { data } = await api.get('/api/v1/ai/alerts/unread-count/');
      setUnreadAlerts(data.count);
    } catch (error) {
      console.error('Failed to load alerts:', error);
    }
  };

  const scrollToBottom = () => {
    messagesEndRef.current?.scrollIntoView({ behavior: 'smooth' });
  };

  const sendMessage = async () => {
    if (!input.trim()) return;

    const userMessage = {
      role: 'user',
      content: input,
      timestamp: new Date(),
    };
  };
};
```

```

setMessages((prev) => [...prev, userMessage]);
setInput('');
setLoading(true);

try {
  const { data } = await api.post('/api/v1/ai/chat/', {
    message: input,
    conversation_id: conversationId,
    context: {
      page: window.location.pathname,
      module: getCurrentModule(),
    },
  });

  setConversationId(data.conversation_id);

  const assistantMessage = {
    role: 'assistant',
    content: data.message,
    intent: data.intent,
    confidence: data.confidence,
    sources: data.sources,
    timestamp: new Date(),
  };

  setMessages((prev) => [...prev, assistantMessage]);
} catch (error) {
  const errorMessage = {
    role: 'assistant',
    content: 'Sorry, I encountered an error. Please try again.',
    timestamp: new Date(),
  };
  setMessages((prev) => [...prev, errorMessage]);
} finally {
  setLoading(false);
}
};

const getCurrentModule = () => {
  const path = window.location.pathname;
  if (path.includes('/finance')) return 'finance';
  if (path.includes('/inventory')) return 'inventory';
  if (path.includes('/sales')) return 'sales';
  return 'general';
};

return (
  <>
  </* Floating AI Button */>
  <Badge count={unreadAlerts} offset={[-5, 5]}>
    <Button
      type="primary"
      shape="circle"
      size="large"
      icon={<RobotOutlined />}
      onClick={() => setVisible(true)}
    />
  </Badge>
</>

```

```

        style={{
            position: 'fixed',
            bottom: 24,
            right: 24,
            width: 60,
            height: 60,
            zIndex: 1000,
            boxShadow: '0 4px 12px rgba(0,0,0,0.15)',
        }}
    />
</Badge>

{/* AI Chat Drawer */}
<Drawer
    title={
        <div style={{ display: 'flex', alignItems: 'center' }}>
            <RobotOutlined style={{ marginRight: 8, fontSize: 20 }} />
            AI Assistant
        </div>
    placement="right"
    width={400}
    onClose={() => setVisible(false)}
    open={visible}
    bodyStyle={{ padding: 0 }}
    >
        <div>
            {/* Messages */}
            <div>
                {messages.length === 0 &&& (
                    <div>
                        <RobotOutlined style={{ fontSize: 48, marginBottom: 16 }} />
                        <div>Hi! I'm your AI assistant.</div>
                        <div>How can I help you today?</div>
                    </div>
                )}
            </div>
            <List
                dataSource={messages}
                renderItem={(msg) => (
                    <div>
                        <div>
                            {msg.content}
                            {msg.sources &&& msg.sources.length > 0 &&& (
                                <div>
                                    ⓘ Sources: {msg.sources.length} documents
                                </div>
                            )}
                        </div>
                    </div>
                )}
            />
            {loading &&& (
                <div>
                    <Spin />

```

```

        </div>
    )}

    <div>
</div>

    { /* Input */}
    <div>
        &lt;Input.Search
            placeholder="Ask me anything..."
            value={input}
            onChange={e => setInput(e.target.value)}
            onSearch={sendMessage}
            onPressEnter={sendMessage}
            loading={loading}
            enterButton={&lt;SendOutlined />}
            size="large"
        /&gt;
    </div>
</div>
    &lt;/Drawer&gt;

    );
};

export default AIWidget;

```

7. Complete File Checklist

Components Created

- ✓ App.jsx - Main application router
- ✓ MainLayout.jsx - Main layout with sidebar
- ✓ Dashboard.jsx - Dashboard with KPIs
- ✓ AccountsList.jsx - Finance chart of accounts
- ✓ FormBuilder.jsx - Drag-and-drop form builder
- ✓ AIWidget.jsx - Always-visible AI assistant

Additional Files Needed (See PDFs)

- AuthContext.jsx - Authentication state management
- CompanyContext.jsx - Multi-company state
- API services for all modules
- 100+ component files for complete UI

Document Version: 1.0

Total UI Components: 150+

Pages: 50+ screens

Status: Core components provided, full library in implementation phase</div>