Analytics Dashboard - Implementation Guide

Current Status

☑ **Backend:** 7 analytics endpoints ready and running
☑ **Chart.js:** Library loaded and ready
⌛ **Frontend:** Need to build analytics dashboard component

Quick Implementation Summary

What's Been Completed:

1. **7 Analytics Endpoints** running on http://localhost:3001:
    - `/api/analytics/overview` - Executive metrics
    - `/api/analytics/spending-trend` - Time series data
    - `/api/analytics/department-breakdown` - Department spending
    - `/api/analytics/approval-flow` - Funnel data
    - `/api/analytics/duration` - Processing times
    - `/api/analytics/status-distribution` - Status breakdown
    - `/api/analytics/top-vendors` - Vendor rankings
2. **Chart.js 4.4.0** - Loaded in index.html
3. **Theme System** - Light/dark mode ready

What Needs to Be Built:

The current `Reports` component (starting around line 3150 in app.js) shows basic statistics. We need to replace/enhance it with:

1. **Enhanced Analytics Component** with:
    - API integration for all 7 endpoints
    - Interactive Chart.js visualizations
    - Date range filtering
    - Department filtering
    - Period selection (daily/weekly/monthly)
2. **Chart Components:**
    - Spending Trend Line Chart
    - Department Pie Chart
    - Status Donut Chart
    - Approval Funnel Chart
    - Duration Bar Chart
    - Vendor Rankings
3. **Filter Panel:**
    - Date range picker
    - Department selector
    - Period toggle
    - Export buttons

Implementation Approach

Option 1: Replace Existing Reports Component (Recommended)

Location: `frontend/app.js` around line 3150

Steps:

1. Add analytics API methods to the `api` object
2. Create new `AnalyticsDashboard` component
3. Replace `Reports` component or add new menu item
4. Build chart rendering functions
5. Add filtering logic

Option 2: Create Separate Analytics View

Steps:

1. Add new menu item "Analytics" (separate from "Reports")
2. Create `AnalyticsDashboard` component
3. Finance/MD/Admin only access
4. Full-featured with all charts

Code Structure Needed

1. API Methods (Add to `api` object)

```
// Around line 700-800 in app.js, add to api object:

analytics: {
  getOverview: async (filters = {}) => {
    const params = new URLSearchParams(filters);
    const res = await fetchWithAuth(`${API_URL}/analytics/overview?${params}`);
    if (!res.ok) throw new Error('Failed to fetch overview');
    return res.json();
  },

  getSpendingTrend: async (filters = {}) => {
    const params = new URLSearchParams(filters);
    const res = await fetchWithAuth(`${API_URL}/analytics/spending-trend?${params}`);
    if (!res.ok) throw new Error('Failed to fetch spending trend');
    return res.json();
  },

  getDepartmentBreakdown: async (filters = {}) => {
    const params = new URLSearchParams(filters);
    const res = await fetchWithAuth(`${API_URL}/analytics/department-breakdown?${params}`);
    if (!res.ok) throw new Error('Failed to fetch department breakdown');
    return res.json();
  },

  getApprovalFlow: async (filters = {}) => {
    const params = new URLSearchParams(filters);
    const res = await fetchWithAuth(`${API_URL}/analytics/approval-flow?${params}`);
    if (!res.ok) throw new Error('Failed to fetch approval flow');
    return res.json();
  },

  getDuration: async (filters = {}) => {
    const params = new URLSearchParams(filters);
    const res = await fetchWithAuth(`${API_URL}/analytics/duration?${params}`);
    if (!res.ok) throw new Error('Failed to fetch duration');
    return res.json();
  },

  getStatusDistribution: async (filters = {}) => {
    const params = new URLSearchParams(filters);
    const res = await fetchWithAuth(`${API_URL}/analytics/status-distribution?${params}`);
    if (!res.ok) throw new Error('Failed to fetch status distribution');
    return res.json();
  },

  getTopVendors: async (filters = {}) => {
    const params = new URLSearchParams(filters);
    const res = await fetchWithAuth(`${API_URL}/analytics/top-vendors?${params}`);
    if (!res.ok) throw new Error('Failed to fetch top vendors');
    return res.json();
  }
}
```

2. Chart Helper Functions

```
// Add before component definitions

// Get theme-aware colors
const getChartColors = () => {
  const theme = getTheme();
  return {
    primary: '#0070AF',
    secondary: '#58A6D0',
    light: '#D0E3F2',
    success: '#10B981',
    warning: '#F59E0B',
    danger: '#EF4444',
    info: '#3B82F6',
    grid: theme === 'dark' ? 'rgba(255,255,255,0.1)' : 'rgba(0,0,0,0.1)',
    text: theme === 'dark' ? '#F1F5F9' : '#1E293B'
  };
};

// Create line chart
const createLineChart = (canvasId, data, options = {}) => {
  const ctx = document.getElementById(canvasId);
  if (!ctx) return null;

  const colors = getChartColors();

  return new Chart(ctx, {
    type: 'line',
    data: {
      labels: data.labels,
      datasets: data.datasets.map((dataset, index) => ({
        ...dataset,
        borderColor: dataset.borderColor || [colors.primary, colors.warning, colors.danger][index],
        backgroundColor: dataset.backgroundColor || 'transparent',
        tension: 0.4
      }))
    },
    options: {
      responsive: true,
      maintainAspectRatio: false,
      plugins: {
        legend: {
          labels: { color: colors.text }
        },
        tooltip: {
          mode: 'index',
          intersect: false
        }
      },
      scales: {
        y: {
          beginAtZero: true,
          grid: { color: colors.grid },
          ticks: { color: colors.text }
        },
        x: {
          grid: { color: colors.grid },
          ticks: { color: colors.text }
        }
      },
      ...options
    }
  });
};

// Create pie/doughnut chart
const createPieChart = (canvasId, data, type = 'pie') => {
  const ctx = document.getElementById(canvasId);
  if (!ctx) return null;

  const colors = getChartColors();
  const chartColors = [
    colors.primary,
    colors.secondary,
    colors.light,
    colors.success,
    colors.warning,
    colors.danger,
    colors.info
  ];

  return new Chart(ctx, {
    type: type,
    data: {
      labels: data.labels,
      datasets: [{
        data: data.values,
        backgroundColor: chartColors,
        borderWidth: 2,
        borderColor: 'var(--bg-primary)'
      }]
    },
    options: {
      responsive: true,
      maintainAspectRatio: false,
      plugins: {
        legend: {
          position: 'right',
          labels: { color: colors.text, padding: 15 }
        },
        tooltip: {
          callbacks: {
            label: (context) => {
              const label = context.label || '';
              const value = context.parsed || 0;
              const total = context.dataset.data.reduce((a, b) => a + b, 0);
              const percentage = ((value / total) * 100).toFixed(1);
              return `${label}: ${value.toLocaleString()} (${percentage}%)`;
            }
          }
        }
      }
    }
  });
};

// Create bar chart
const createBarChart = (canvasId, data, options = {}) => {
  const ctx = document.getElementById(canvasId);
  if (!ctx) return null;

  const colors = getChartColors();

  return new Chart(ctx, {
    type: 'bar',
    data: {
      labels: data.labels,
      datasets: data.datasets.map((dataset, index) => ({
        ...dataset,
        backgroundColor: dataset.backgroundColor || colors.primary,
        borderColor: dataset.borderColor || colors.primary,
        borderWidth: 1
      }))
    },
    options: {
      responsive: true,
      maintainAspectRatio: false,
      plugins: {
        legend: {
          labels: { color: colors.text }
        }
      },
      scales: {
        y: {
          beginAtZero: true,
          grid: { color: colors.grid },
          ticks: { color: colors.text }
        },
        x: {
          grid: { display: false },
          ticks: { color: colors.text }
        }
      },
      ...options
    }
  });
};
```

3. Analytics Dashboard Component Structure

```
function AnalyticsDashboard({ user }) {
  const [loading, setLoading] = useState(true);
  const [filters, setFilters] = useState({
    dateFrom: '',
    dateTo: '',
    department: '',
    period: 'monthly'
  });

  const [overview, setOverview] = useState(null);
  const [spendingTrend, setSpendingTrend] = useState(null);
  const [departments, setDepartments] = useState(null);
  const [approvalFlow, setApprovalFlow] = useState(null);
  const [duration, setDuration] = useState(null);
  const [statusDist, setStatusDist] = useState(null);
  const [topVendors, setTopVendors] = useState(null);

  const [charts, setCharts] = useState({});

  useEffect(() => {
    loadAnalytics();
  }, [filters]);

  useEffect(() => {
    // Destroy charts on unmount
    return () => {
      Object.values(charts).forEach(chart => {
        if (chart) chart.destroy();
      });
    };
  }, []);

  const loadAnalytics = async () => {
    setLoading(true);
    try {
      const filterParams = {};
      if (filters.dateFrom) filterParams.dateFrom = filters.dateFrom;
      if (filters.dateTo) filterParams.dateTo = filters.dateTo;
      if (filters.department) filterParams.department = filters.department;
      if (filters.period) filterParams.period = filters.period;

      const [
        overviewData,
        trendData,
        deptData,
        flowData,
        durationData,
        statusData,
        vendorData
      ] = await Promise.all([
        api.analytics.getOverview(filterParams),
        api.analytics.getSpendingTrend(filterParams),
        api.analytics.getDepartmentBreakdown(filterParams),
        api.analytics.getApprovalFlow(filterParams),
        api.analytics.getDuration(filterParams),
        api.analytics.getStatusDistribution(filterParams),
        api.analytics.getTopVendors({ ...filterParams, limit: 5 })
      ]);

      setOverview(overviewData);
      setSpendingTrend(trendData);
      setDepartments(deptData);
      setApprovalFlow(flowData);
      setDuration(durationData);
      setStatusDist(statusData);
      setTopVendors(vendorData);

      // Render charts after data loads
      setTimeout(() => renderCharts(), 100);

    } catch (error) {
      console.error('Error loading analytics:', error);
      alert('Failed to load analytics data');
    } finally {
      setLoading(false);
    }
  };

  const renderCharts = () => {
    // Destroy existing charts
    Object.values(charts).forEach(chart => {
      if (chart) chart.destroy();
    });

    const newCharts = {};

    // Spending Trend Line Chart
    if (spendingTrend && spendingTrend.data) {
      newCharts.spendingTrend = createLineChart('spendingTrendChart', {
        labels: spendingTrend.data.map(d => d.period),
        datasets: [
          {
            label: 'Approved',
            data: spendingTrend.data.map(d => d.approved),
            borderColor: '#10B981'
          },
          {
            label: 'Pending',
            data: spendingTrend.data.map(d => d.pending),
            borderColor: '#F59E0B'
          },
          {
            label: 'Rejected',
            data: spendingTrend.data.map(d => d.rejected),
            borderColor: '#EF4444'
          }
        ]
      });
    }

    // Department Pie Chart
    if (departments && departments.length > 0) {
      newCharts.departments = createPieChart('departmentChart', {
        labels: departments.map(d => d.department),
        values: departments.map(d => d.total_amount || 0)
      }, 'doughnut');
    }

    // Status Distribution
    if (statusDist && statusDist.length > 0) {
      newCharts.status = createPieChart('statusChart', {
        labels: statusDist.map(s => s.status),
        values: statusDist.map(s => s.count)
      }, 'doughnut');
    }

    // Duration Bar Chart
    if (duration) {
      newCharts.duration = createBarChart('durationChart', {
        labels: ['HOD', 'Procurement', 'Finance', 'MD'],
        datasets: [{
          label: 'Average Days',
          data: [
            duration.hod_stage || 0,
            duration.procurement_stage || 0,
            duration.finance_stage || 0,
            duration.md_stage || 0
          ],
          backgroundColor: [
            duration.hod_stage > 2 ? '#EF4444' : '#10B981',
            duration.procurement_stage > 3 ? '#EF4444' : '#10B981',
            duration.finance_stage > 1 ? '#EF4444' : '#10B981',
            duration.md_stage > 1 ? '#EF4444' : '#10B981'
          ]
        }]
      }, {
        indexAxis: 'y'
      });
    }

    setCharts(newCharts);
  };

  // Component JSX structure here...
  // (see next section)
}
```

4. Component Render Structure

```
return React.createElement('div', {
  className: "space-y-6",
  style: { minHeight: '100vh' }
},
  // Header
  React.createElement('div', {
    className: "flex items-center justify-between"
  },
    React.createElement('h2', {
      className: "text-3xl font-bold transition-colors",
      style: { color: 'var(--text-primary)' }
    }, "📊 Analytics Dashboard"),
    React.createElement(ThemeToggle)
  ),

  // Filter Panel
  React.createElement('div', {
    className: "p-6 rounded-lg transition-colors",
    style: {
      backgroundColor: 'var(--bg-primary)',
      borderColor: 'var(--border-color)',
      boxShadow: 'var(--shadow-md)'
    }
  },
    React.createElement('div', { className: "grid grid-cols-1 md:grid-cols-4 gap-4" },
      // Date From
      React.createElement('div', null,
        React.createElement('label', {
          className: "block text-sm font-medium mb-2 transition-colors",
          style: { color: 'var(--text-secondary)' }
        }, "Date From"),
        React.createElement('input', {
          type: "date",
          value: filters.dateFrom,
          onChange: (e) => setFilters({...filters, dateFrom: e.target.value}),
          className: "w-full px-3 py-2 border rounded-lg transition-colors"
        })
      ),
      // Date To
      React.createElement('div', null,
        React.createElement('label', {
          className: "block text-sm font-medium mb-2 transition-colors",
          style: { color: 'var(--text-secondary)' }
        }, "Date To"),
        React.createElement('input', {
          type: "date",
          value: filters.dateTo,
          onChange: (e) => setFilters({...filters, dateTo: e.target.value}),
          className: "w-full px-3 py-2 border rounded-lg transition-colors"
        })
      ),
      // Department
      React.createElement('div', null,
        React.createElement('label', {
          className: "block text-sm font-medium mb-2 transition-colors",
          style: { color: 'var(--text-secondary)' }
        }, "Department"),
      React.createElement('select', {
        value: filters.department,
        onChange: (e) => setFilters({...filters, department: e.target.value}),
        className: "w-full px-3 py-2 border rounded-lg transition-colors"
      },
        React.createElement('option', { value: "" }, "All Departments"),
        React.createElement('option', { value: "Operations" }, "Operations"),
        React.createElement('option', { value: "Finance" }, "Finance"),
        React.createElement('option', { value: "IT" }, "IT"),
        React.createElement('option', { value: "HR" }, "HR"),
        React.createElement('option', { value: "Procurement" }, "Procurement")
      )
    ),
    // Period
    React.createElement('div', null,
      React.createElement('label', {
        className: "block text-sm font-medium mb-2 transition-colors",
        style: { color: 'var(--text-secondary)' }
      }, "Period"),
      React.createElement('select', {
        value: filters.period,
        onChange: (e) => setFilters({...filters, period: e.target.value}),
        className: "w-full px-3 py-2 border rounded-lg transition-colors"
      },
        React.createElement('option', { value: "daily" }, "Daily"),
        React.createElement('option', { value: "weekly" }, "Weekly"),
        React.createElement('option', { value: "monthly" }, "Monthly")
      )
    )
  )
),

// KPI Cards
overview && React.createElement('div', { className: "grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4" },
  // Total Spend
  React.createElement('div', {
    className: "p-6 rounded-lg transition-colors",
    style: {
      backgroundColor: '#D0E3F2',
      borderColor: '#0070AF'
    }
  },
    React.createElement('p', {
      className: "text-sm font-medium mb-2",
      style: { color: '#0070AF' }
    }, "Total Spend"),
    React.createElement('p', {
      className: "text-3xl font-bold",
      style: { color: '#0070AF' }
    }, `ZMW ${(overview.totalSpend || 0).toLocaleString()}`)
  ),
  // Avg Processing Time
  React.createElement('div', {
    className: "p-6 rounded-lg transition-colors",
    style: {
      backgroundColor: '#D1FAE5',
      borderColor: '#10B981'
    }
  },
    React.createElement('p', {
      className: "text-sm font-medium mb-2",
      style: { color: '#10B981' }
    }, "Avg Processing Time"),
    React.createElement('p', {
      className: "text-3xl font-bold",
      style: { color: '#10B981' }
    }, `${(overview.avgProcessingTime || 0).toFixed(1)} days`)
  ),
  // Approval Rate
  React.createElement('div', {
    className: "p-6 rounded-lg transition-colors",
    style: {
      backgroundColor: '#FEF3C7',
      borderColor: '#F59E0B'
    }
  },
    React.createElement('p', {
      className: "text-sm font-medium mb-2",
      style: { color: '#F59E0B' }
    }, "Approval Rate"),
    React.createElement('p', {
      className: "text-3xl font-bold",
      style: { color: '#F59E0B' }
    }, `${overview.totalRequisitions > 0 ? ((overview.completedRequisitions / overview.totalRequisitions) * 100).toFixed(1) : 0}%`)
  ),
  // Active Requisitions
  React.createElement('div', {
    className: "p-6 rounded-lg transition-colors",
    style: {
      backgroundColor: '#E0E7FF',
      borderColor: '#3B82F6'
    }
  },
    React.createElement('p', {
      className: "text-sm font-medium mb-2",
      style: { color: '#3B82F6' }
    }, "Active Requisitions"),
    React.createElement('p', {
      className: "text-3xl font-bold",
      style: { color: '#3B82F6' }
    }, overview.pendingRequisitions || 0)
  )
),

// Charts Grid
!loading && React.createElement('div', { className: "grid grid-cols-1 lg:grid-cols-2 gap-6" },
  // Spending Trend
  React.createElement('div', {
    className: "p-6 rounded-lg transition-colors",
    style: {
      backgroundColor: 'var(--bg-primary)',
      borderColor: 'var(--border-color)',
      boxShadow: 'var(--shadow-md)'
    }
  },
    React.createElement('h3', {
      className: "text-xl font-bold mb-4 transition-colors",
      style: { color: 'var(--text-primary)' }
    }, "📈 Spending Trend"),
    React.createElement('div', { style: { height: '300px' } },
      React.createElement('canvas', { id: "spendingTrendChart" })
    )
  ),

  // Department Breakdown
  React.createElement('div', {
    className: "p-6 rounded-lg transition-colors",
    style: {
      backgroundColor: 'var(--bg-primary)',
      borderColor: 'var(--border-color)',
      boxShadow: 'var(--shadow-md)'
    }
  },
    React.createElement('h3', {
      className: "text-xl font-bold mb-4 transition-colors",
      style: { color: 'var(--text-primary)' }
    }, "🏢 Department Spending"),
    React.createElement('div', { style: { height: '300px' } },
      React.createElement('canvas', { id: "departmentChart" })
    )
  ),

  // Status Distribution
  React.createElement('div', {
    className: "p-6 rounded-lg transition-colors",
    style: {
      backgroundColor: 'var(--bg-primary)',
      borderColor: 'var(--border-color)',
      boxShadow: 'var(--shadow-md)'
    }
  },
    React.createElement('h3', {
      className: "text-xl font-bold mb-4 transition-colors",
      style: { color: 'var(--text-primary)' }
    }, "📊 Status Distribution"),
    React.createElement('div', { style: { height: '300px' } },
      React.createElement('canvas', { id: "statusChart" })
    )
  ),

  // Duration Analysis
  React.createElement('div', {
    className: "p-6 rounded-lg transition-colors",
    style: {
      backgroundColor: 'var(--bg-primary)',
      borderColor: 'var(--border-color)',
      boxShadow: 'var(--shadow-md)'
    }
  },
    React.createElement('h3', {
      className: "text-xl font-bold mb-4 transition-colors",
      style: { color: 'var(--text-primary)' }
      }, "⏱ Avg Duration by Stage"),
      React.createElement('div', { style: { height: '300px' } },
        React.createElement('canvas', { id: "durationChart" })
      )
    )
  ),

  // Loading State
  loading && React.createElement('div', {
    className: "flex items-center justify-center py-12"
  },
    React.createElement('div', { className: "text-center" },
      React.createElement('div', {
        className: "animate-spin rounded-full h-12 w-12 border-b-2 mx-auto mb-4",
        style: { borderColor: '#0070AF' }
      }),
      React.createElement('p', {
        className: "transition-colors",
        style: { color: 'var(--text-secondary)' }
      }, "Loading analytics...")
    )
  )
);
```

Integration Steps

1. **Add API methods** to the `api` object (around line 700-800)
2. **Add chart helper functions** before component definitions (around line 990)
3. **Create `AnalyticsDashboard` component** (around line 3150, after Reports)
4. **Update App component** to render AnalyticsDashboard when `view === 'analytics'`
5. **Add menu item** in Sidebar for Analytics (or replace Reports)

Testing

1. Login as Finance Manager (sarah.banda / password123)
2. Click "Analytics" or "Reports" in sidebar
3. Verify charts render correctly
4. Test filters (date range, department, period)
5. Check light/dark mode compatibility
6. Verify data accuracy

Next Steps After Implementation

1. Add export functionality (PDF/Excel)
2. Add drill-down click handlers
3. Implement approval funnel visualization
4. Add vendor analytics
5. Create comparative period analysis
6. Add alert thresholds
7. Implement scheduled reports

Current Files

- **Backend:** `backend/server.js` (Analytics endpoints lines 1938-2326)
- **Frontend:** `frontend/app.js` (Need to add AnalyticsDashboard component)
- **HTML:** `frontend/index.html` (Chart.js already loaded line 16)

Status: Ready for frontend implementation
Estimated Time: 2-3 hours for full dashboard
Priority: High - Significant business value for Finance/MD