Frontend Code

`package.json`

```
{
    "name": "vizsprints-frontend",
    "private": true,
    "version": "0.0.0",
    "type": "module",
    "scripts": {
        "dev": "vite",
        "build": "vite build",
        "preview": "vite preview"
    },
    "dependencies": {
        "react": "^18.2.0",
        "react-dom": "^18.2.0",
        "recharts": "^2.10.3",
        "axios": "^1.6.2"
    },
    "devDependencies": {
        "@types/react": "^18.2.43",
        "@types/react-dom": "^18.2.17",
        "@vitejs/plugin-react": "^4.2.1",
        "autoprefixer": "^10.4.16",
        "postcss": "^8.4.32",
        "tailwindcss": "^3.3.6",
        "vite": "^5.0.8"
    }
}
```

`vite.config.js`

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vitejs.dev/config/
export default defineConfig({
    plugins: [react()],
    server: {
        proxy: {
            '/api': {
                target: 'http://localhost:5000',
                changeOrigin: true,
            },
        },
    },
})
```

`tailwind.config.js`

```
/** @type {import('tailwindcss').Config} */
export default {
```

```
    content: [
        "./index.html",
        "./src/**/*.{js,ts,jsx,tsx}",
    ],
    theme: {
        extend: {
            colors: {
                primary: '#3b82f6',
                secondary: '#64748b',
                dark: '#0f172a',
                light: '#f8fafc',
            },
        },
    },
    plugins: [],
}
```

`postcss.config.js`

```
export default {
    plugins: {
        tailwindcss: {},
        autoprefixer: {},
    },
}
```

`index.html`

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>VizSprints Dashboard</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

`src/main.jsx`

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'

ReactDOM.createRoot(document.getElementById('root')).render(
    <React.StrictMode>
        <App />
    </React.StrictMode>,
)
```

`src/App.jsx`

```
import React from 'react';
import Dashboard from './components/Dashboard';

function App() {
    return (
        <div className="min-h-screen bg-slate-50">
            <nav className="bg-white shadow-sm border-b border-slate-200">
                <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
                    <div className="flex justify-between h-16">
                        <div className="flex items-center">
                            <span className="text-2xl font-bold text-primary">VizSprints</span>
                        </div>
                        <div className="flex items-center space-x-4">
                                        <div  className="text-sm  text-slate-500">Product  Analytics
Dashboard</div>
                        </div>
                    </div>
                </div>
            </nav>

            <main className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-8">
                <Dashboard />
            </main>
        </div>
    );
}

export default App;
```

`src/index.css`

```
@tailwind base;
@tailwind components;
@tailwind utilities;

body {
  @apply bg-slate-900 text-slate-200;
}
```

`src/api.js`

```
import axios from 'axios';

const api = axios.create({
    baseURL: '/api',
    headers: {
        'Content-Type': 'application/json',
    },
});

export default api;
```

`src/components/Dashboard.jsx`

```
import React, { useEffect, useState } from 'react';
import api from '../api';
import MetricsCard from './MetricsCard';
import CohortHeatmap from './CohortHeatmap';
import FunnelChart from './FunnelChart';
import ABTestResults from './ABTestResults';
import UserSessions from './UserSessions';
import KPITimeSeries from './KPITimeSeries';

const Dashboard = () => {
    const [metrics, setMetrics] = useState(null);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
        const fetchMetrics = async () => {
            try {
                const response = await api.get('/metrics');
                setMetrics(response.data);
                setLoading(false);
            } catch (err) {
                console.error('Error fetching metrics:', err);
                setError('Failed to load dashboard data');
                setLoading(false);
            }
        };

        fetchMetrics();
    }, []);

    if (loading) {
        return (
            <div className="flex justify-center items-center h-64">
                                <div  className="animate-spin  rounded-full  h-12  w-12  border-b-2
border-primary"></div>
            </div>
        );
    }

    if (error) {
        return (
            <div className="bg-red-900/20 border-l-4 border-red-400 p-4">
                <div className="flex">
                    <div className="ml-3">
                        <p className="text-sm text-red-300">{error}</p>
                    </div>
                </div>
            </div>
        );
    }
```

```
    return (
        <div className="space-y-6">
            {/* Key Metrics Grid */}
            <div className="grid grid-cols-1 gap-5 sm:grid-cols-2 lg:grid-cols-3">
                    <MetricsCard title="Total Users" value={metrics.total_users.toLocaleString()}
subtext="Registered users" />
                  <MetricsCard title="Active Users" value={metrics.active_users.toLocaleString()}
subtext="Last 30 days" />
                      <MetricsCard title="Conversion Rate" value={`${metrics.conversion_rate}%`}
subtext="Task completion" />
            </div>

            {/* KPI Time Series */}
            <div className="bg-slate-800 p-6 rounded-lg shadow">
                <h3 className="text-lg font-medium leading-6 text-white mb-4">KPI Trends</h3>
                <KPITimeSeries />
            </div>

            {/* Charts Section */}
            <div className="grid grid-cols-1 gap-6">
                <div className="bg-slate-800 p-6 rounded-lg shadow">
                        <h3 className="text-lg font-medium leading-6 text-white mb-4">Conversion
Funnel</h3>
                    <FunnelChart />
                </div>
            </div>

            {/* Detailed components stacked */}
            <div className="grid grid-cols-1 gap-6">
                <div className="bg-slate-800 p-6 rounded-lg shadow">
                     <h3 className="text-lg font-medium leading-6 text-white mb-4">Cohort Retention
Analysis</h3>
                    <CohortHeatmap />
                </div>
                <div className="bg-slate-800 p-6 rounded-lg shadow">
                          <h3 className="text-lg font-medium leading-6 text-white mb-4">A/B Test
Results</h3>
                    <ABTestResults />
                </div>
                <div className="bg-slate-800 p-6 rounded-lg shadow">
                       <h3 className="text-lg font-medium leading-6 text-white mb-4">User Activity
Tracking</h3>
                    <UserSessions />
                </div>
            </div>
        </div>
    );
};

export default Dashboard;
```

`src/components/MetricsCard.jsx`

```
import React from 'react';
```

```
const MetricsCard = ({ title, value, subtext, trend }) => {
    return (
                <div  className="bg-slate-800  overflow-hidden  rounded-lg  shadow  hover:shadow-md
transition-shadow duration-300">
            <div className="p-5">
                <div className="flex items-center">
                    <div className="flex-1">
                        <dt className="text-sm font-medium text-slate-400 truncate">{title}</dt>
                        <dd className="mt-1 text-3xl font-semibold text-white">{value}</dd>
                    </div>
                </div>
                <div className="mt-4">
                    <div className={`text-sm ${trend >= 0 ? 'text-green-400' : 'text-red-400'}`}>
                        {subtext}
                    </div>
                </div>
            </div>
        </div>
    );
};

export default MetricsCard;
```

`src/components/KPITimeSeries.jsx`

```
import React, { useEffect, useState } from 'react';
import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer, Legend } from
'recharts';
import api from '../api';

const KPITimeSeries = () => {
    const [data, setData] = useState([]);
    const [metric, setMetric] = useState('dau'); // 'dau' or 'signups'
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        const fetchData = async () => {
            try {
                const response = await api.get('/kpi-time-series');
                setData(response.data);
                setLoading(false);
            } catch (err) {
                console.error('Error fetching KPI data:', err);
                setLoading(false);
            }
        };

        fetchData();
    }, []);

            if   (loading)   return   <div   className="h-64   flex   items-center   justify-center
text-slate-400">Loading...</div>;
```

```
    const formatXAxis = (tickItem) => {
        const date = new Date(tickItem);
        return `${date.getMonth() + 1}/${date.getDate()}`;
    };

    return (
        <div className="w-full">
            <div className="flex justify-end mb-4">
                <div className="inline-flex rounded-md shadow-sm" role="group">
                    <button
                        type="button"
                        onClick={() => setMetric('dau')}
                         className={`px-4 py-2 text-sm font-medium rounded-l-lg border ${metric ===
'dau'
                                ? 'bg-blue-600 text-white border-blue-600'
                                                : 'bg-slate-700 text-slate-300 border-slate-600
hover:bg-slate-600'
                            }`}
                    >
                        DAU
                    </button>
                    <button
                        type="button"
                        onClick={() => setMetric('signups')}
                         className={`px-4 py-2 text-sm font-medium rounded-r-lg border ${metric ===
'signups'
                                ? 'bg-blue-600 text-white border-blue-600'
                                                : 'bg-slate-700 text-slate-300 border-slate-600
hover:bg-slate-600'
                            }`}
                    >
                        Signups
                    </button>
                </div>
            </div>

            <div className="h-72">
                <ResponsiveContainer width="100%" height="100%">
                    <LineChart
                        data={data}
                        margin={{ top: 5, right: 20, left: 10, bottom: 5 }}
                    >
                        <CartesianGrid strokeDasharray="3 3" stroke="#334155" vertical={false} />
                        <XAxis
                            dataKey="date"
                            stroke="#94a3b8"
                            tickFormatter={formatXAxis}
                            minTickGap={30}
                        />
                        <YAxis stroke="#94a3b8" />
                        <Tooltip
                                contentStyle={{ backgroundColor: '#1e293b', borderColor: '#334155',
color: '#f8fafc' }}
                            itemStyle={{ color: '#f8fafc' }}
```

```
                        labelStyle={{ color: '#94a3b8' }}
                    />
                    <Legend />
                    <Line
                        type="monotone"
                        dataKey={metric}
                        name={metric === 'dau' ? 'Daily Active Users' : 'Signups per Day'}
                        stroke={metric === 'dau' ? '#3b82f6' : '#10b981'}
                        strokeWidth={2}
                        dot={false}
                        activeDot={{ r: 6 }}
                    />
                </LineChart>
            </ResponsiveContainer>
        </div>
    </div>
  );
};

export default KPITimeSeries;
```

`src/components/FunnelChart.jsx`

```
import React, { useEffect, useState } from 'react';
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer, Cell }
from 'recharts';
import api from '../api';

const FunnelChart = () => {
  const [data, setData] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await api.get('/funnel');
        setData(response.data.funnel);
        setLoading(false);
      } catch (err) {
        console.error('Error fetching funnel data:', err);
        setLoading(false);
      }
    };

    fetchData();
  }, []);

      if (loading) return <div className="h-64 flex items-center justify-center
text-slate-400">Loading...</div>;

  return (
    <div className="h-80 w-full">
      <ResponsiveContainer width="100%" height="100%">
        <BarChart
```

```
                    data={data}
                    layout="vertical"
                    margin={{
                        top: 5,
                        right: 30,
                        left: 40,
                        bottom: 5,
                    }}
                >
                      <CartesianGrid strokeDasharray="3 3" horizontal={true} vertical={false}
stroke="#334155" />
                    <XAxis type="number" hide />
                    <YAxis
                        dataKey="stage"
                        type="category"
                        width={100}
                        tick={{ fontSize: 12, fill: '#94a3b8' }}
                        tickLine={false}
                        axisLine={false}
                    />
                    <Tooltip
                        cursor={{ fill: '#334155' }}
                        content={({ active, payload, label }) => {
                            if (active && payload && payload.length) {
                                const data = payload[0].payload;
                                return (
                                       <div className="bg-slate-800 p-3 rounded-lg shadow-lg border
border-slate-700">
                                         <p className="font-medium text-white">{label}</p>
                                                       <p className="text-blue-400 text-sm">Users:
{data.users}</p>
                                                  <p className="text-slate-400 text-xs">Conversion:
{data.conversion_from_previous}%</p>
                                                      <p className="text-red-400 text-xs">Drop-off:
{data.drop_off}%</p>
                                    </div>
                                );
                            }
                            return null;
                        }}
                    />
                    <Bar dataKey="users" radius={[0, 4, 4, 0]}>
                        {data.map((entry, index) => (
                             <Cell key={`cell-${index}`} fill={index === 0 ? '#3b82f6' : '#60a5fa'}
fillOpacity={1 - (index * 0.15)} />
                        ))}
                    </Bar>
                </BarChart>
            </ResponsiveContainer>
        </div>
    );
};

export default FunnelChart;
```

`src/components/CohortHeatmap.jsx`

```
import React, { useEffect, useState } from 'react';
import api from '../api';

const CohortHeatmap = () => {
    const [data, setData] = useState([]);
    const [maxMonths, setMaxMonths] = useState(0);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        const fetchData = async () => {
            try {
                const response = await api.get('/cohorts');
                setData(response.data.cohorts);
                setMaxMonths(response.data.max_months);
                setLoading(false);
            } catch (err) {
                console.error('Error fetching cohort data:', err);
                setLoading(false);
            }
        };

        fetchData();
    }, []);

            if (loading) return <div className="h-64 flex items-center justify-center
text-slate-400">Loading...</div>;

    const getBackgroundColor = (value) => {
        if (value === 0) return 'bg-slate-800';
        if (value < 20) return 'bg-blue-950';
        if (value < 40) return 'bg-blue-900';
        if (value < 60) return 'bg-blue-700';
        if (value < 80) return 'bg-blue-600';
        return 'bg-blue-500';
    };

    return (
        <div className="overflow-x-auto">
            <table className="min-w-full divide-y divide-slate-700">
                <thead className="bg-slate-700">
                    <tr>
                            <th className="px-3 py-2 text-left text-xs font-medium text-slate-300
uppercase tracking-wider sticky left-0 bg-slate-700">Cohort</th>
                            <th className="px-3 py-2 text-left text-xs font-medium text-slate-300
uppercase tracking-wider">Users</th>
                        {[...Array(maxMonths + 1)].map((_, i) => (
                                <th key={i} className="px-3 py-2 text-center text-xs font-medium
text-slate-300 uppercase tracking-wider">
                                Month {i}
                            </th>
                        ))}
```

```
                    </tr>
                </thead>
                <tbody className="bg-slate-800 divide-y divide-slate-700">
                    {data.map((row) => (
                        <tr key={row.cohort}>
                                <td className="px-3 py-2 whitespace-nowrap text-sm font-medium
text-white sticky left-0 bg-slate-800">
                                {row.cohort}
                            </td>
                            <td className="px-3 py-2 whitespace-nowrap text-sm text-slate-300">
                                {row.size}
                            </td>
                            {[...Array(maxMonths + 1)].map((_, i) => {
                                const value = row[`month_${i}`];
                                return (
                                    <td
                                        key={i}
                                                 className={`px-3 py-2 whitespace-nowrap text-sm
text-center ${getBackgroundColor(value)}`}
                                        title={`${value}% retention`}
                                    >
                                        {value > 0 ? `${value}%` : '-'}
                                    </td>
                                );
                            })}
                        </tr>
                    ))}
                </tbody>
            </table>
        </div>
    );
};

export default CohortHeatmap;
```

`src/components/ABTestResults.jsx`

```
import React, { useEffect, useState } from 'react';
import api from '../api';

const ABTestResults = () => {
    const [data, setData] = useState(null);
    const [loading, setLoading] = useState(true);

    // Live Data Filters
    const [sampleSize, setSampleSize] = useState('');
    const [eventLimit, setEventLimit] = useState('');

    // Simulation Mode State
    const [isSimulation, setIsSimulation] = useState(false);

    // Simulation / Stats Parameters
    const [confidenceLevel, setConfidenceLevel] = useState(0.95);
    const [simNA, setSimNA] = useState(1000);
```

```
    const [simNB, setSimNB] = useState(1000);
    const [simConvA, setSimConvA] = useState(10.0);
    const [simConvB, setSimConvB] = useState(12.0);

    useEffect(() => {
        const fetchData = async () => {
            try {
                const params = new URLSearchParams();

                // Common Params
                params.append('confidence_level', confidenceLevel);

                if (isSimulation) {
                    params.append('manual_n_a', simNA);
                    params.append('manual_n_b', simNB);
                    params.append('manual_conv_a', simConvA);
                    params.append('manual_conv_b', simConvB);
                } else {
                    if (sampleSize) params.append('limit', sampleSize);
                    if (eventLimit) params.append('event_limit', eventLimit);
                }

                const url = `/ab-test?${params.toString()}`;
                const response = await api.get(url);
                setData(response.data);
                setLoading(false);
            } catch (err) {
                console.error('Error fetching A/B test data:', err);
                setLoading(false);
            }
        };

        const timeoutId = setTimeout(() => {
            fetchData();
        }, 500); // Debounce API calls

        return () => clearTimeout(timeoutId);
    }, [sampleSize, eventLimit, isSimulation, confidenceLevel, simNA, simNB, simConvA, simConvB]);

        if (loading && !data) return <div className="h-64 flex items-center justify-center
text-slate-400">Loading...</div>;

    const VariantCard = ({ variant, data, isWinner }) => (
           <div className={`p-4 rounded-lg border ${isWinner ? 'border-green-500 bg-slate-700' :
'border-slate-600 bg-slate-800'}`}>
            <div className="flex justify-between items-center mb-4">
                <h4 className="text-lg font-semibold text-white">Variant {variant}</h4>
                      {isWinner && <span className="px-2 py-1 text-xs font-medium text-green-300
bg-green-900/50 rounded-full">Winner</span>}
            </div>
            <dl className="grid grid-cols-2 gap-4">
                <div>
                    <dt className="text-xs text-slate-400">Users</dt>
                    <dd className="text-lg font-medium text-white">{data.total_users}</dd>
```

```
                </div>
                <div>
                        <dt className="text-xs text-slate-400">{isSimulation ? 'Est. Conv Rate' :
'Events/User'}</dt>
                    <dd className="text-lg font-medium text-white">
                                        {isSimulation  ?  `${data.funnel[0].conversion_rate}%`  :
data.avg_events_per_user}
                    </dd>
                </div>
            </dl>

            <div className="mt-4">
                <h5 className="text-sm font-medium text-slate-300 mb-2">Funnel Conversion</h5>
                <div className="space-y-2">
                    {data.funnel.map((stage) => (
                              <div key={stage.stage} className="flex justify-between items-center
text-sm">
                            <span className="text-slate-400 truncate w-24">{stage.stage}</span>
                                      <div  className="flex-1  mx-2  h-2  bg-slate-700  rounded-full
overflow-hidden">
                                <div
                                            className={`h-full rounded-full ${variant === 'A' ?
'bg-blue-400' : 'bg-purple-400'}`}
                                     style={{ width: `${stage.conversion_rate}%` }}
                                />
                            </div>
                                                                <span  className="text-slate-300  w-12
text-right">{stage.conversion_rate}%</span>
                        </div>
                    ))}
                </div>
            </div>
        </div>
    );

    return (
        <div className="space-y-6">
            {/* Header / Controls */}
            <div className="bg-slate-800 p-4 rounded-lg border border-slate-700">
                       <div  className="flex  flex-col  md:flex-row  justify-between  items-start
md:items-center gap-4 mb-4">
                    <div>
                        <h3 className="text-lg font-semibold text-white flex items-center gap-2">
                            A/B Test Analysis
                                      <span  className={`text-xs  px-2  py-0.5  rounded-full  border
${isSimulation ? 'border-purple-500 text-purple-400' : 'border-blue-500 text-blue-400'}`}>
                                {isSimulation ? 'Calculator Mode' : 'Live Data'}
                            </span>
                        </h3>
                        <p className="text-sm text-slate-400">
                              {isSimulation ? 'Input data from two variations (A and B) to see if
the difference is real.' : 'Analyzing real user events from database.'}
                        </p>
                    </div>
```

```
                        <div className="flex items-center bg-slate-900 rounded-lg p-1 border
border-slate-700">
                    <button
                        onClick={() => setIsSimulation(false)}
                                    className={`px-3 py-1 text-sm rounded-md transition-colors
${!isSimulation ? 'bg-slate-700 text-white' : 'text-slate-400 hover:text-white'}`}
                    >
                        Live Data
                    </button>
                    <button
                        onClick={() => setIsSimulation(true)}
                                    className={`px-3 py-1 text-sm rounded-md transition-colors
${isSimulation ? 'bg-purple-600 text-white' : 'text-slate-400 hover:text-white'}`}
                    >
                        Calculator
                    </button>
                </div>
            </div>

            {/* Parameters Grid */}
            <div className="grid grid-cols-1 md:grid-cols-4 gap-4">
                <div className="space-y-1">
                    <label className="text-xs text-slate-400">Confidence Level</label>
                    <select
                        value={confidenceLevel}
                        onChange={(e) => setConfidenceLevel(parseFloat(e.target.value))}
                           className="w-full bg-slate-900 border border-slate-600 rounded px-2
py-1 text-white text-sm focus:border-blue-500 outline-none"
                    >
                        <option value="0.90">90%</option>
                        <option value="0.95">95%</option>
                        <option value="0.99">99%</option>
                    </select>
                </div>

                {!isSimulation ? (
                    <>
                        <div className="space-y-1">
                                        <label className="text-xs text-slate-400">Max Users
(Sample)</label>
                            <input
                                type="number"
                                value={sampleSize}
                                onChange={(e) => setSampleSize(e.target.value)}
                                placeholder="All"
                                 className="w-full bg-slate-900 border border-slate-600 rounded
px-2 py-1 text-white text-sm focus:border-blue-500 outline-none"
                                />
                        </div>
                        <div className="space-y-1">
                            <label className="text-xs text-slate-400">Max Events</label>
                            <input
                                type="number"
```

```
                                    value={eventLimit}
                                    onChange={(e) => setEventLimit(e.target.value)}
                                    placeholder="All"
                                     className="w-full bg-slate-900 border border-slate-600 rounded
px-2 py-1 text-white text-sm focus:border-blue-500 outline-none"
                                />
                            </div>
                        </>
                    ) : (
                        <>
                            <div className="space-y-1">
                                <label className="text-xs text-slate-400">Sample Size</label>
                                <div className="flex gap-2">
                                    <input
                                        type="number"
                                        value={simNA}
                                        onChange={(e) => setSimNA(parseInt(e.target.value) || 0)}
                                            className="w-full bg-slate-900 border border-slate-600
rounded px-2 py-1 text-white text-sm focus:border-purple-500 outline-none"
                                        placeholder="A"
                                    />
                                    <input
                                        type="number"
                                        value={simNB}
                                        onChange={(e) => setSimNB(parseInt(e.target.value) || 0)}
                                            className="w-full bg-slate-900 border border-slate-600
rounded px-2 py-1 text-white text-sm focus:border-purple-500 outline-none"
                                        placeholder="B"
                                    />
                                </div>
                            </div>
                            <div className="space-y-1">
                                 <label className="text-xs text-slate-400">Conversion Rate
(%)</label>
                                <div className="flex gap-2">
                                    <input
                                        type="number"
                                        value={simConvA}
                                         onChange={(e) => setSimConvA(parseFloat(e.target.value) ||
0)}
                                            className="w-full bg-slate-900 border border-slate-600
rounded px-2 py-1 text-white text-sm focus:border-purple-500 outline-none"
                                        placeholder="A %"
                                    />
                                    <input
                                        type="number"
                                        value={simConvB}
                                         onChange={(e) => setSimConvB(parseFloat(e.target.value) ||
0)}
                                            className="w-full bg-slate-900 border border-slate-600
rounded px-2 py-1 text-white text-sm focus:border-purple-500 outline-none"
                                        placeholder="B %"
                                    />
                                </div>
```

```
                        </div>
                    </>
                )}
            </div>
        </div>

        {data && (
            <>
                {/* Statistical Significance Banner */}
                        <div  className={`p-4  rounded-lg  flex  items-center  justify-between
${data.stats.significant
                    ? 'bg-green-900/30 border border-green-500/50'
                    : 'bg-slate-800 border border-slate-700'
                    }`}>
                    <div>
                            <div  className={`text-lg  font-bold  ${data.stats.significant  ?
'text-green-400' : 'text-slate-300'}`}>
                            {data.stats.significant
                                ? `Statistically Significant! Variation ${data.lift > 0 ? 'B'
: 'A'} is better.`
                                : "Not Statistically Significant."
                            }
                        </div>
                        <div  className="text-sm  text-slate-400  mt-1">
                                                    P-Value:  <span  className="text-white
font-mono">{data.stats.p_value}</span>
                            <span  className="mx-2">?</span>
                                                                Confidence  Level:  <span
className="text-white">{(data.stats.confidence_level * 100).toFixed(0)}%</span>
                        </div>
                    </div>
                    <div  className="text-right">
                        <div  className="text-sm  text-slate-400">Statistical  Power</div>
                            <div  className={`text-2xl  font-bold  ${data.stats.power  >  0.8  ?
'text-green-400' : 'text-yellow-400'}`}>
                            {Math.round(data.stats.power * 100)}%
                        </div>
                                <div  className="text-xs  text-slate-500">Prob.  of  detecting
effect</div>
                    </div>
                </div>

                {/* Cards */}
                < div  className="flex  justify-end" >
                    <div  className={`text-sm  font-medium  ${data.lift  >  0  ?  'text-green-400'  :
'text-red-400'}`}>
                        {data.lift > 0 ? '+' : ''}{data.lift}% Lift (B vs A)
                    </div>
                </div >

                <div  className="grid  grid-cols-1  md:grid-cols-2  gap-4">
                    <VariantCard
                        variant="A"
                        data={data.variant_A}
```

```
                            isWinner={data.lift < 0 && data.stats.significant}
                        />
                        <VariantCard
                            variant="B"
                            data={data.variant_B}
                            isWinner={data.lift > 0 && data.stats.significant}
                        />
                    </div>
                </>
            )}
        </div >
    );
};

export default ABTestResults;
```

`src/components/UserSessions.jsx`

```
import React, { useEffect, useState } from 'react';
import api from '../api';

const UserSessions = () => {
    const [data, setData] = useState([]);
    const [loading, setLoading] = useState(true);
    const [sortBy, setSortBy] = useState('total_hours');

    useEffect(() => {
        fetchData();
    }, [sortBy]);

    const fetchData = async () => {
        try {
            const response = await api.get(`/user-sessions?limit=50&sort_by=${sortBy}`);
            setData(response.data.user_sessions);
            setLoading(false);
        } catch (err) {
            console.error('Error fetching user sessions:', err);
            setLoading(false);
        }
    };

    const formatDate = (dateString) => {
        const date = new Date(dateString);
        return date.toLocaleString('en-US', {
            year: 'numeric',
            month: 'short',
            day: 'numeric',
            hour: '2-digit',
            minute: '2-digit'
        });
    };

            if (loading) return <div className="h-64 flex items-center justify-center
text-slate-400">Loading...</div>;
```

```
    return (
        <div className="space-y-4">
            {/* Header with sorting */}
            <div className="flex justify-between items-center">
                <div className="text-sm text-slate-400">
                    Showing top 50 users by activity
                </div>
                <div className="flex gap-2">
                    <label className="text-sm text-slate-400">Sort by:</label>
                    <select
                        value={sortBy}
                        onChange={(e) => setSortBy(e.target.value)}
                              className="bg-slate-700 text-white text-sm rounded px-2 py-1 border
border-slate-600 focus:outline-none focus:border-blue-500"
                    >
                        <option value="total_hours">Total Hours</option>
                        <option value="total_sessions">Total Sessions</option>
                        <option value="last_activity">Last Activity</option>
                    </select>
                </div>
            </div>

            {/* Table */}
            <div className="overflow-x-auto">
                <table className="min-w-full divide-y divide-slate-700">
                    <thead className="bg-slate-700">
                        <tr>
                             <th className="px-4 py-3 text-left text-xs font-medium text-slate-300
uppercase tracking-wider">
                                User ID
                            </th>
                             <th className="px-4 py-3 text-left text-xs font-medium text-slate-300
uppercase tracking-wider">
                                First Activity
                            </th>
                             <th className="px-4 py-3 text-left text-xs font-medium text-slate-300
uppercase tracking-wider">
                                Last Activity
                            </th>
                                        <th className="px-4 py-3 text-center text-xs font-medium
text-slate-300 uppercase tracking-wider">
                                Sessions
                            </th>
                                        <th className="px-4 py-3 text-center text-xs font-medium
text-slate-300 uppercase tracking-wider">
                                Total Hours
                            </th>
                                        <th className="px-4 py-3 text-center text-xs font-medium
text-slate-300 uppercase tracking-wider">
                                Avg Session
                            </th>
                                        <th className="px-4 py-3 text-center text-xs font-medium
text-slate-300 uppercase tracking-wider">
```

```
                                Status
                            </th>
                        </tr>
                    </thead>
                    <tbody className="bg-slate-800 divide-y divide-slate-700">
                        {data.map((user) => (
                                        <tr  key={user.user_id}  className="hover:bg-slate-750
transition-colors">
                                <td className="px-4 py-3 whitespace-nowrap text-sm font-medium
text-white">
                                {user.user_id}
                            </td>
                                        <td  className="px-4  py-3  whitespace-nowrap  text-sm
text-slate-300">
                                {formatDate(user.first_activity)}
                            </td>
                                        <td  className="px-4  py-3  whitespace-nowrap  text-sm
text-slate-300">
                                {formatDate(user.last_activity)}
                            </td>
                                <td className="px-4 py-3 whitespace-nowrap text-sm text-center
text-white">
                                {user.total_sessions}
                            </td>
                                <td className="px-4 py-3 whitespace-nowrap text-sm text-center
font-semibold text-blue-400">
                                {user.total_hours.toFixed(2)}h
                            </td>
                                <td className="px-4 py-3 whitespace-nowrap text-sm text-center
text-slate-300">
                                {user.avg_session_duration.toFixed(2)}h
                            </td>
                            <td className="px-4 py-3 whitespace-nowrap text-center">
                                  <span className={`px-2 py-1 text-xs font-medium rounded-full
${user.status === 'active'
                                    ? 'bg-green-900/50 text-green-300'
                                    : 'bg-slate-700 text-slate-400'
                                    }`}>
                                    {user.status}
                                </span>
                            </td>
                        </tr>
                    ))}
                    </tbody>
                </table>
            </div>

            {data.length === 0 && (
                <div className="text-center py-8 text-slate-400">
                    No user session data available
                </div>
            )}
        </div>
    );
```

```
};

export default UserSessions;
```