```
import React, { useEffect, useState, useCallback } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { createSlice, configureStore } from '@reduxjs/toolkit';
import { motion } from 'framer-motion';
import { Sparkline } from '@sparkcharts/react';
import { ArrowUp, ArrowDown, TrendingUp, TrendingDown } from 'lucide-react';

// ================================
// Mock WebSocket & Data
// ================================

// Simulate a WebSocket connection with random data updates
class MockWebSocket {
  listeners: { [event: string]: ((data: any) => void)[] } = {};

  on(event: string, callback: (data: any) => void) {
    if (!this.listeners[event]) {
      this.listeners[event] = [];
    }
    this.listeners[event].push(callback);
  }

  // Simulate sending updates every 1-2 seconds
  simulateUpdates() {
    setInterval(() => {
      const updatedData = generateRandomUpdates();
```

```
            if (this.listeners['message']) {
                this.listeners['message'].forEach(callback => callback(updatedData));
            }
        }, 1500); // Changed to 1500ms for better visibility
    }

    close() {
        clearInterval(this.intervalId); // Clear interval on close
    }

    private intervalId: NodeJS.Timeout; // Store the interval ID
}

// Initial static data
const initialCryptoData = [
    {
        id: 'bitcoin',
        name: 'Bitcoin',
        symbol: 'BTC',
        price: 64893.45,
        change1h: 0.85,
        change24h: 4.22,
        change7d: 8.15,
        marketCap: 1274000000000,
        volume24h: 35890000000,
        supply: 19672325,
        maxSupply: 21000000,
        chartData: [64000, 64200, 64500, 64750, 65000, 64900, 64893], // 7 days
        logo: 'https://assets.coingecko.com/coins/images/1/large/bitcoin.png?1696501438',
    },
    {
        id: 'ethereum',
        name: 'Ethereum',
        symbol: 'ETH',
        price: 3524.12,
        change1h: -0.32,
        change24h: 2.88,
        change7d: 5.62,
        marketCap: 423500000000,
        volume24h: 18560000000,
        supply: 120234567,
        maxSupply: null,
        chartData: [3450, 3475, 3500, 3520, 3550, 3540, 3524],
        logo: 'https://assets.coingecko.com/coins/images/279/large/ethereum.png?1696501628',
    },
    {
        id: 'tether',
        name: 'Tether',
        symbol: 'USDT',
        price: 1.00,
        change1h: 0.05,
        change24h: -0.02,
        change7d: 0.10,

        marketCap: 110200000000,

        volume24h: 65430000000,

        supply: 110000000000,

        maxSupply: null,

        chartData: [0.998, 0.999, 1.00, 1.001, 1.002, 1.001, 1.00],

        logo: 'https://assets.coingecko.com/coins/images/325/large/Tether-logo.png?1696501661',

    },

    {

        id: 'binancecoin',

        name: 'Binance Coin',

        symbol: 'BNB',

        price: 602.55,

        change1h: 1.20,

        change24h: 6.70,

        change7d: 12.30,

        marketCap: 92850000000,

        volume24h: 2567000000,

        supply: 153847333,

        maxSupply: null,

        chartData: [590, 595, 600, 605, 610, 608, 602],

        logo: 'https://assets.coingecko.com/coins/images/825/large/bnb-
icon2_2x.png?1696502070',

    },

    {

        id: 'solana',
        name: 'Solana',
        symbol: 'SOL',
        price: 172.88,
        change1h: -0.75,
        change24h: 9.40,
        change7d: 18.50,
        marketCap: 78500000000,
        volume24h: 10450000000,
        supply: 453215876,
        maxSupply: null,
        chartData: [165, 168, 170, 173, 175, 174, 172],
        logo: 'https://assets.coingecko.com/coins/images/4128/large/solana.png?1696504226',
    },
];

// Function to generate random updates
const generateRandomUpdates = () => {
    return initialCryptoData.map(item => ({
        id: item.id,
        price: +(item.price + (Math.random() - 0.5) * (item.price * 0.05)).toFixed(2), // Price
changes by +/- 5%
        change1h: +(item.change1h + (Math.random() - 0.5) * 1).toFixed(2),     // Changes by +/- 1
        change24h: +(item.change24h + (Math.random() - 0.5) * 3).toFixed(2),   // Changes by +/- 3
        change7d: +(item.change7d + (Math.random() - 0.5) * 5).toFixed(2),     // Changes by +/- 5
        volume24h: +(item.volume24h + (Math.random() - 0.5) * (item.volume24h *
0.1)).toFixed(0), // Volume changes by +/- 10%
```

```
        chartData: [...item.chartData.slice(1), +(item.price + (Math.random() - 0.5) * (item.price *
0.05)).toFixed(2)], // Add new price, remove oldest
    }));
};

// ================================
// Redux Setup
// ================================

// Create a Redux slice for crypto data
const cryptoSlice = createSlice({
    name: 'crypto',
    initialState: initialCryptoData,
    reducers: {
        updateCryptoData: (state, action) => {
            action.payload.forEach((updatedItem: any) => {
                const index = state.findIndex(item => item.id === updatedItem.id);
                if (index !== -1) {
                    state[index] = { ...state[index], ...updatedItem };
                }
            });
        },
    },
});

// Export the action
```

```
export const { updateCryptoData } = cryptoSlice.actions;

// Create the Redux store
const store = configureStore({
  reducer: {
    crypto: cryptoSlice.reducer,
  },
});

// Selector for getting crypto data.  Good practice for performance.
const selectCryptoData = (state: any) => state.crypto;

// ==============================
// Components
// ==============================

// Reusable component for displaying percentage changes with styling
const PercentageChange = ({ value }: { value: number }) => {
  const isPositive = value >= 0;
  return (
    <div className={`flex items-center gap-1 ${isPositive ? 'text-green-500' : 'text-red-500'}`}>
      {isPositive ? (
        <ArrowUp className="w-4 h-4" />
      ) : (
        <ArrowDown className="w-4 h-4" />
      )}
```

```
            <span>{value.toFixed(2)}%</span>
        </div>
    );
};

// Component for displaying the sparkline chart
const MiniChart = ({ data }: { data: number[] }) => {
    const lastValue = data[data.length - 1];
    const firstValue = data[0];
    const isPositive = lastValue >= firstValue;
    return (
        <Sparkline
            data={data}
            width={100}
            height={30}
            stroke={isPositive ? '#16a34a' : '#dc2626'} // Tailwind green-600 and red-600
            fill={isPositive ? 'rgba(22, 163, 74, 0.2)' : 'rgba(220, 38, 38, 0.2)'}
            gradient={false}
        />
    );
};

// Main App Component
const CryptoPriceTracker = () => {
    const cryptoData = useSelector(selectCryptoData);
    const dispatch = useDispatch();
const [ws] = useState(new MockWebSocket()); // Use useState for consistent instance

// Simulate WebSocket updates
useEffect(() => {
  ws.on('message', (updatedData: any) => {
    dispatch(updateCryptoData(updatedData));
  });

  ws.simulateUpdates(); // Start sending mock updates

  return () => {
    ws.close(); // Clean up the interval
  };
}, [dispatch, ws]);

// Memoize the render of each row.  Crucial for performance with Redux.
const renderRow = useCallback((item: any) => {
  return (
    <motion.tr
      key={item.id}
      initial={{ opacity: 0, y: -10 }}
      animate={{ opacity: 1, y: 0 }}
      exit={{ opacity: 0, y: 10 }}
      transition={{ duration: 0.2 }}
    >
      <td className="px-4 py-2">
```

```
                <img src={item.logo} alt={item.name} className="w-8 h-8 rounded-full" />
              </td>
              <td className="px-4 py-2 font-semibold">{item.name}</td>
              <td className="px-4 py-2 text-gray-500">{item.symbol}</td>
              <td className="px-4 py-2 font-mono">${item.price.toFixed(2)}</td>
              <td className="px-4 py-2">
                <PercentageChange value={item.change1h} />
              </td>
              <td className="px-4 py-2">
                <PercentageChange value={item.change24h} />
              </td>
              <td className="px-4 py-2">
                <PercentageChange value={item.change7d} />
              </td>
              <td className="px-4 py-2 font-mono">${(item.marketCap / 1000000000).toFixed(2)}
B</td>
              <td className="px-4 py-2 font-mono">${(item.volume24h / 1000000000).toFixed(2)}
B</td>
              <td className="px-4 py-2 font-mono">{item.supply.toLocaleString()} /
{item.maxSupply ? item.maxSupply.toLocaleString() : '∞'}</td>
              <td className="px-4 py-2">
                <MiniChart data={item.chartData} />
              </td>
            </motion.tr>
          );
    }, []);
```

```
  return (
    <div className="container mx-auto p-4">
      <h1 className="text-3xl font-bold mb-6 text-center text-gray-800">Real-Time Crypto
Price Tracker</h1>
      <div className="shadow-lg rounded-lg overflow-x-auto">
        <table className="min-w-full bg-white rounded-lg">
          <thead className="bg-gray-100">
            <tr>
              <th className="px-4 py-2 text-left">Logo</th>
              <th className="px-4 py-2 text-left">Name</th>
              <th className="px-4 py-2 text-left">Symbol</th>
              <th className="px-4 py-2 text-right">Price</th>
              <th className="px-4 py-2 text-right">1h %</th>
              <th className="px-4 py-2 text-right">24h %</th>
              <th className="px-4 py-2 text-right">7d %</th>
              <th className="px-4 py-2 text-right">Market Cap</th>
              <th className="px-4 py-2 text-right">24h Volume</th>
              <th className="px-4 py-2 text-left">Circulating Supply</th>
              <th className="px-4 py-2 text-center">7D Chart</th>
            </tr>
          </thead>
          <tbody>
            {cryptoData.map(renderRow)}
          </tbody>
        </table>
      </div>
    </div>
  );
};

export default CryptoPriceTracker;
```