Cloud Pub/Sub Dashboard

```
import React, { useState, useEffect } from 'react';
import { Activity, Wifi, WifiOff, Plus, Trash2, Bell, BellOff } from 'lucide-react';

export default function PubSubDashboard() {
  const [publishers, setPublishers] = useState([]);
  const [subscribers, setSubscribers] = useState([]);
  const [topics, setTopics] = useState(['temperature', 'humidity', 'pressure', 'motion']);
  const [messages, setMessages] = useState([]);
  const [newPublisher, setNewPublisher] = useState({ name: '', topic: 'temperature' });
  const [newSubscriber, setNewSubscriber] = useState({ name: '', topic: 'temperature' });

  // Simulate device publishing
  useEffect(() => {
    const interval = setInterval(() => {
      publishers.forEach(pub => {
        if (pub.active) {
          const value = generateValue(pub.topic);
          const message = {
            id: Date.now() + Math.random(),
            publisher: pub.name,
            topic: pub.topic,
            value: value,
            timestamp: new Date().toLocaleTimeString()
          };

          setMessages(prev => [message, ...prev].slice(0, 50));

          // Notify subscribers
          subscribers.forEach(sub => {
            if (sub.active && sub.topic === pub.topic) {
              sub.messageCount = (sub.messageCount || 0) + 1;
            }
          });
        }
      });
    }, 3000);

    return () => clearInterval(interval);
  }, [publishers, subscribers]);

  const generateValue = (topic) => {
    switch(topic) {
      case 'temperature':
        return (20 + Math.random() * 10).toFixed(1) + '°C';
      case 'humidity':
        return (40 + Math.random() * 40).toFixed(1) + '%';
      case 'pressure':
        return (990 + Math.random() * 30).toFixed(1) + ' hPa';
      case 'motion':
        return Math.random() > 0.5 ? 'Detected' : 'Clear';
      default:
        return Math.random().toFixed(2);
    }
  };

  const addPublisher = () => {
    if (newPublisher.name.trim()) {
      setPublishers([...publishers, {
```

```
      id: Date.now(),
      name: newPublisher.name,
      topic: newPublisher.topic,
      active: true
    }]);
    setNewPublisher({ name: '', topic: 'temperature' });
  }
};

const addSubscriber = () => {
  if (newSubscriber.name.trim()) {
    setSubscribers([...subscribers, {
      id: Date.now(),
      name: newSubscriber.name,
      topic: newSubscriber.topic,
      active: true,
      messageCount: 0
    }]);
    setNewSubscriber({ name: '', topic: 'temperature' });
  }
};

const togglePublisher = (id) => {
  setPublishers(publishers.map(p =>
    p.id === id ? { ...p, active: !p.active } : p
  ));
};

const toggleSubscriber = (id) => {
  setSubscribers(subscribers.map(s =>
    s.id === id ? { ...s, active: !s.active } : s
  ));
};

const removePublisher = (id) => {
  setPublishers(publishers.filter(p => p.id !== id));
};

const removeSubscriber = (id) => {
  setSubscribers(subscribers.filter(s => s.id !== id));
};

const getTopicColor = (topic) => {
  const colors = {
    temperature: 'bg-red-500',
    humidity: 'bg-blue-500',
    pressure: 'bg-purple-500',
    motion: 'bg-green-500'
  };
  return colors[topic] || 'bg-gray-500';
};

return (
  <div className="min-h-screen bg-gradient-to-br from-slate-900 via-slate-800 to-slate-900 text-white p
    <div className="max-w-7xl mx-auto">
      {/* Header */}
      <div className="mb-8">
        <div className="flex items-center gap-3 mb-2">
          <Activity className="w-8 h-8 text-cyan-400" />
          <h1 className="text-3xl font-bold">IoT Pub/Sub Dashboard</h1>
        </div>
        <p className="text-slate-400">Real-time publish/subscribe message broker</p>
      </div>
```

```
...
  );
}
```