const { createMachine, createActor, assign } = require("xstate");  
const axios = require("axios");  
  
const apiMachine = createMachine({  
 id: "apiRequest",  
 initial: "loading", // Start in loading state directly  
 context: {  
 data: null,  
 error: null,  
 endpoint: null, // Dynamically set  
 },  
 states: {  
 loading: {  
 invoke: {  
 src: "fetchData",  
 onDone: { target: "success", actions: "setData" },  
 onError: { target: "failure", actions: "setError" },  
 },  
 },  
 success: {  
 type: "final",  
 },  
 failure: {  
 type: "final",  
 },  
 },  
}, {  
 actions: {  
 setData: assign((context, event) => ({ data: event.output, error: null })),  
 setError: assign((context, event) => ({ error: event.error })),  
 },  
 services: {  
 fetchData: async (context) => {  
 const response = await axios.get(context.endpoint);  
 return response.data;  
 },  
 },  
});  
  
module.exports = { apiMachine, createActor };

const express = require("express");  
const { createActor } = require("xstate");  
const { apiMachine } = require("./apiMachine");  
  
const app = express();  
app.use(express.json());  
  
app.get("/api/proxy", async (req, res) => {  
 const endpoint = req.query.url;  
  
 if (!endpoint) return res.status(400).json({ error: "URL is required" });  
  
 try {  
 const service = createActor(apiMachine.provide({  
 context: { endpoint } // Inject endpoint into machine  
 }));  
  
 service.start(); // Start the machine  
 await service.waitFor((state) => state.done); // Wait for final state  
  
 const state = service.getSnapshot(); // Get final state  
  
 if (state.matches("success")) {  
 res.json(state.context.data);  
 } else {  
 res.status(500).json({ error: state.context.error });  
 }  
  
 service.stop(); // Destroy the state machine  
 } catch (error) {  
 res.status(500).json({ error: "Unexpected error occurred" });  
 }  
});  
  
app.listen(3000, () => console.log("Server running on port 3000"));