// src/models/orderModel.js

let orders = []; // In-memory data store

module.exports = {

createOrder(orderData, callback) {

// Simulate saving the order with status 'accepted'

const newOrder = {

id: orders.length + 1,

...orderData,

status: 'accepted'

};

orders.push(newOrder);

// Simulate asynchronous acceptance with setTimeout

setTimeout(() => {

// Once "accepted", call the callback

callback(newOrder);

}, 500);

},

processOrder(orderId) {

// Return a Promise to process the order

return new Promise((resolve, reject) => {

const order = orders.find(o => o.id === orderId);

if (!order) {

return reject('Order not found');

}

// Simulate async processing (e.g., cooking, packaging)

setTimeout(() => {

order.status = 'processed';

resolve();

}, 2000);

});

},

getOrderStatus(orderId) {

// Return a Promise to fetch order status asynchronously

return new Promise((resolve) => {

setTimeout(() => {

const order = orders.find(o => o.id === orderId);

if (!order) {

return resolve(null);

}

resolve(order.status);

}, 500);

});

}

};

// src/controllers/orderController.js

Const Order = require(‘../models/orderModel’);

Const orderEvents = require(‘../events/orderEvents’);

Exports.acceptNewOrder = (req, res) => {

Const orderData = req.body;

// Callback function to confirm the order was accepted

Function orderAcceptedCallback(order) {

// Respond to the client once the order is accepted

Res.status(201).json({

Message: ‘Order accepted successfully’,

Order: order

});

}

// 1. Accept the order (using callback)

Order.createOrder(orderData, orderAcceptedCallback);

// 2. Process the order asynchronously (using Promise)

Order.processOrder(orderData.id)

.then(() => {

// 3. Emit an event when order is processed/ready

orderEvents.emit(‘orderReady’, orderData.id);

})

.catch((err) => {

Console.error(‘Error processing order:’, err);

});

};

Exports.getOrderStatus = async (req, res) => {

Try {

Const orderId = parseInt(req.params.id, 10);

// 4. Retrieve order status using async/await

Const status = await Order.getOrderStatus(orderId);

If (!status) {

Return res.status(404).json({ message: ‘Order not found’ });

}

Res.json({ orderId, status });

} catch (error) {

Res.status(500).json({ message: ‘Internal server error’, error });

}

};