// src/models/orderModel.js

Const { sql, poolPromise } = require(‘../db’);

Module.exports = {

// 1. Create a new order in the DB and use a callback when “accepted”

createOrder(orderData, callback) {

(async () => {

Try {

Const pool = await poolPromise;

// Insert a new row with status ‘accepted’

Const result = await pool.request()

.input(‘item’, sql.VarChar, orderData.item)

.input(‘quantity’, sql.Int, orderData.quantity)

.input(‘status’, sql.VarChar, ‘accepted’)

.query(`

INSERT INTO Orders (item, quantity, status)

OUTPUT INSERTED.\*

VALUES (@item, @quantity, @status)

`);

// The newly inserted row is in result.recordset[0]

Const newOrder = result.recordset[0];

// Simulate a short delay to confirm “accepted”

setTimeout(() => {

callback(newOrder);

}, 500);

} catch (err) {

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

// If there is an error, you can call the callback with null or handle it differently

Callback(null);

}

})();

},

// 2. Process the order (Promise-based)

processOrder(orderId) {

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

try {

const pool = await poolPromise;

// Check if order exists

Const checkResult = await pool.request()

.input(‘id’, sql.Int, orderId)

.query(‘SELECT \* FROM Orders WHERE id = @id’);

If (checkResult.recordset.length === 0) {

Return reject(‘Order not found’);

}

// Simulate async “processing” delay

setTimeout(async () => {

try {

// Update status to ‘processed’

Await pool.request()

.input(‘id’, sql.Int, orderId)

.input(‘status’, sql.VarChar, ‘processed’)

.query(‘UPDATE Orders SET status = @status WHERE id = @id’);

Resolve();

} catch (updateError) {

Reject(updateError);

}

}, 2000);

} catch (err) {

Reject(err);

}

});

},

// 3. Retrieve order status (using async/await in the controller)

getOrderStatus(orderId) {

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

try {

const pool = await poolPromise;

// Simulate a small delay to mimic an async call

setTimeout(async () => {

try {

const result = await pool.request()

.input(‘id’, sql.Int, orderId)

.query(‘SELECT status FROM Orders WHERE id = @id’);

If (result.recordset.length === 0) {

Return resolve(null); // no such order

}

// Return just the status

Resolve(result.recordset[0].status);

} catch (queryError) {

Reject(queryError);

}

}, 500);

} catch (err) {

Reject(err);

}

});

}

};

// src/controllers/orderController.js

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

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

// Accept a new order

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

Const orderData = req.body;

Function orderAcceptedCallback(order) {

If (!order) {

Return res.status(500).json({ message: ‘Failed to accept order’ });

}

Res.status(201).json({

Message: ‘Order accepted successfully’,

Order: order

});

}

// 1. Create the order (SQL insert) and use the callback

Order.createOrder(orderData, orderAcceptedCallback);

// 2. Process the order asynchronously

Order.processOrder(orderData.id)

.then(() => {

// 3. Emit event when order is processed

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

})

.catch(err => {

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

});

};

// Retrieve the order status

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

Try {

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

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 });

}

};