Asynchronous Programming Dengan Asycn/Await di Node.js

Pengertian Async/Await:

Async/Await adalah sintaks yang lebih modern untuk menangani operasi asynchronous di JavaScript. Async/await dibangun di atas Promise dan menyediakan cara yang lebih mudah dan lebih bersih untuk menulis kode asynchronous yang tampak seperti kode synchronous. Kata kunci async menandai sebuah fungsi sebagai asynchronous, dan kata kunci await digunakan untuk menunggu hasil dari Promise.

Karakteristik Async/Await di Node.js:

1. Berbasis Promise: Async/await bekerja dengan Promise, sehingga fungsi async selalu mengembalikan Promise.
2. Syntax yang Bersih: Mengurangi kompleksitas dan memungkinkan penulisan kode asynchronous yang terlihat seperti synchronous.
3. Error Handling: Menggunakan blok try-catch untuk menangani error, menjadikannya lebih intuitif.
4. Debugging yang Lebih Mudah: Lebih mudah untuk melacak stack trace dan error.
5. Sequential dan Parallel Execution: Dapat menjalankan operasi secara berurutan atau parallel dengan mudah.

Deskripsi Modul:

Pada praktikum ini, mahasiswa akan melanjutkan pengembangan aplikasi backend sistem reservasi restoran menggunakan Node.js. Fokus utama modul ini adalah implementasi Async/Await pattern untuk mengelola pesanan, termasuk pembuatan pesanan baru dengan validasi meja dan item menu, pengambilan daftar pesanan untuk menampilkan semua pesanan yang ada, serta pembaruan status pesanan, seperti mengubah status dari pending ke completed. Modul ini dirancang untuk membantu mahasiswa memahami cara menggunakan Async/Await dalam menangani operasi asynchronous, seperti akses database, validasi data, dan manajemen status pesanan.

Tujuan Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu:

1. Memahami konsep Async/Await dan perbedaannya dengan callback dan promise.

2. Mengimplementasikan operasi asynchronous menggunakan Async/Await.
3. Membuat endpoint untuk manajemen pesanan (create, read, update).
4. Melakukan validasi data dan penanganan error dengan Async/Await.
5. Mengintegrasikan manajemen pesanan dengan modul lain (seperti manajemen meja dan menu).

Kesimpulan

Modul ini dirancang untuk membantu mahasiswa memahami dan mengimplementasikan Async/Await dalam konteks manajemen pesanan. Dengan mengikuti modul ini, mahasiswa akan memahami cara kerja Async/Await, mampu membuat endpoint untuk manajemen pesanan, menerapkan validasi data dan penanganan error dengan baik, serta mengintegrasikan manajemen pesanan dengan modul lain, seperti meja dan menu. Dengan demikian, mahasiswa akan memiliki pemahaman yang kuat tentang pengembangan aplikasi backend menggunakan Node.js dan Async/Await.

Praktikum 4: Implementasi Order Management dengan Async/Await

Langkah 1: Setup Model Order

1. Buat file orderModel.js di folder src/models/orderModel.js
2. Tambahkan kode berikut di dalam file orderModel.js

```
const orderItemSchema = new mongoose.Schema({
    menuId: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'Menu',
        required: true
    },
    quantity: {
        type: Number,
        required: true,
        min: 1
    }
});
```

3. Tambahkan kode berikut di dalam file orderModel.js juga

```
const orderSchema = new mongoose.Schema({
    tableNumber: {
        type: Number,
        required: true
    },
    items: [orderItemSchema],
    total: {
        type: Number,
        required: true
    },
    status: {
        type: String,
        enum: ['pending', 'completed', 'cancelled'],
        default: 'pending'
    },
    createdAt: {
        type: Date,
        default: Date.now
    },
```

```
    updatedAt: {
        type: Date,
        default: Date.now
    }
});
```

Langkah 2: Implementasi Order Controller

1. Buat file orderController.js di dalam folder src/controller/orderController.js
2. Import model menu, meja dan order
3. Tambahkan kode created order berikut di file orderController.js

```
const createOrder = async (req, res) => {
    const { tableNumber, items } = req.body;
    try {
        // Cek apakah meja tersedia
        const meja = await Meja.findOne({ tableNumber, status: 'available' });
        if (!meja) {
            return res.status(400).json({ success: false, error: 'Meja tidak
tersedia atau sedang dipesan' });
        }
        // Cek apakah semua item menu valid
        const menuItems = await Menu.find({ _id: { $in: items.map(item =>
item.menuId) } });
        if (menuItems.length !== items.length) {
            return res.status(400).json({ success: false, error: 'Beberapa item
menu tidak valid' });
        }
        // Hitung total harga
        const total = items.reduce((acc, item) => {
            const menuItem = menuItems.find(menu => menu._id.equals(item.menuId));
            return acc + (menuItem.price * item.quantity);
        }, 0);
        // Buat pesanan
        const order = new Order({
            tableNumber,
            items,
            total,
            status: 'pending'
        });
        const savedOrder = await order.save();
        // Update status meja menjadi 'occupied'
        await Meja.findOneAndUpdate({ tableNumber }, { status: 'occupied' });
        res.status(201).json({ success: true, data: savedOrder });
```

```
    } catch (error) {
        res.status(500).json({ success: false, error: error.message });
    }
};
```

4. Tambahkan kode get all order berikut di file orderController.js

```
const getAllOrders = async (req, res) => {
    try {
        const orders = await Order.find().sort({ createdAt: -1 });
        res.status(200).json({ success: true, data: orders });
    } catch (error) {
        res.status(500).json({ success: false, error: error.message });
    }
};
```

5. Ekspor tiap fungsi secara individual

Langkah 3: Implementasi Routes

1. Buat file orderRoute.js di dalam folder src/routes/orderRoute.js
2. Tambahkan kode berikut di file orderRoute.js

```
const express = require('express');
const router = express.Router();
const orderController = require('../controllers/orderController');

router.post('/createOrders', orderController.createOrder);
router.get('/orders', orderController.getAllOrders);

module.exports = router;
```

3. Import directory menuRoute.js ke file app.js

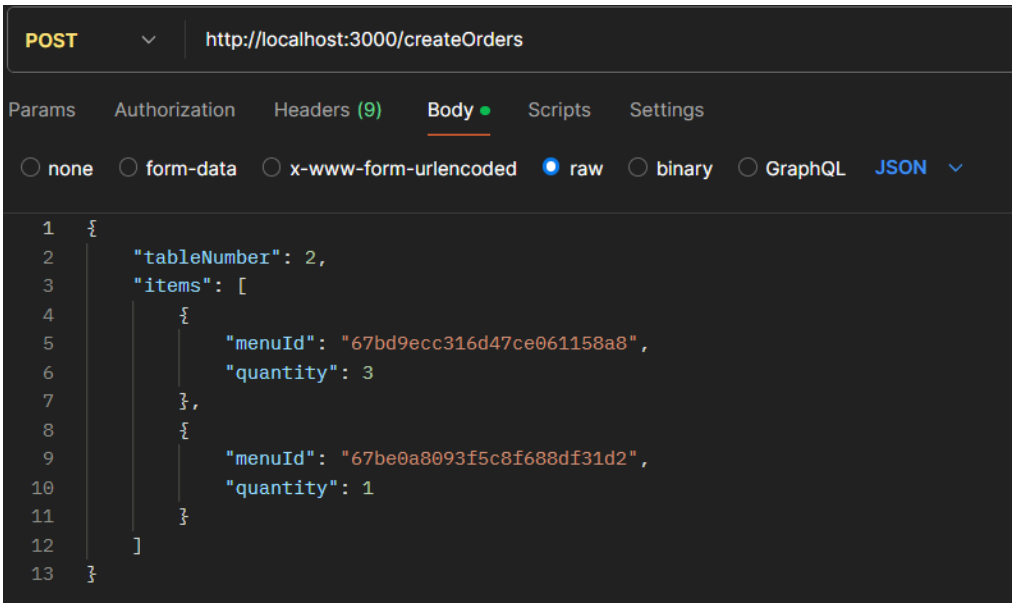
```
const orderRoutes = require('./src/routes/orderRoutes');
```

4. Selanjutnya tambahkan kode berikut di file app.js

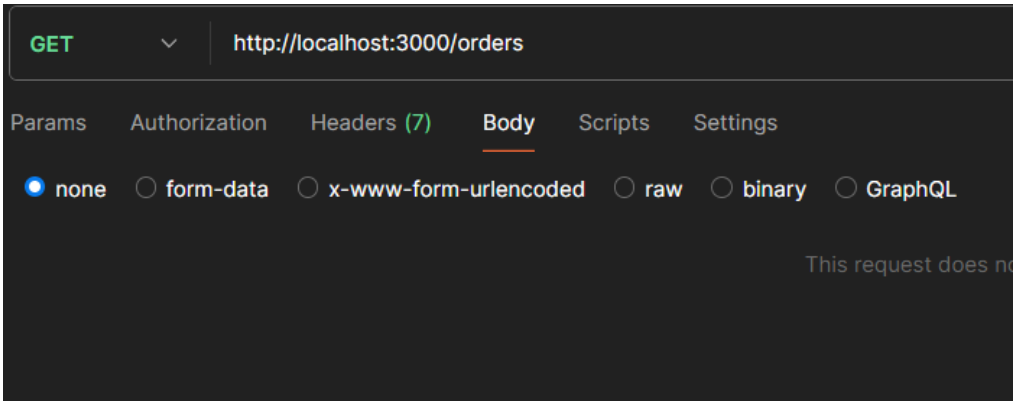
```
app.use('/', orderRoutes);
```

Langkah 4: Testing dengan Postman

1. Sebelum melakukan pengujian pada order pastikan anda sudah menambahkan beberapa data untuk menu dan meja
2. Create Order (POST/ http://localhost:{port}createOrder)



3. Get All Order (GET/ http://localhost:{port}/order)



Tase Case

1. Buatlah API untuk update status order yang akan mengubah status meja menjadi available dengan clue berikut

   Controllers

```
const updateOrderStatus = async (___, ___) => {
    const { ___ } = ___.params; // TODO: Ambil orderId dari parameter URL
    const { ___ } = ___.body; // TODO: Ambil status dari request body

    try {
        // TODO: Cari dan update pesanan berdasarkan orderId
        const order = await Order.___(
            ___,
            { ___ }, // TODO: Update status pesanan
            { ___ } // TODO: Kembalikan dokumen yang sudah diupdate
        );

        // TODO: Periksa apakah pesanan ditemukan
        if (!___) {
            return ___.status(___).json({ success: false, error: 'Pesanan tidak ditemukan' });
        }

        // TODO: Jika status pesanan selesai, kembalikan status meja menjadi 'available'
        if (___ === '___') {
            await Meja.___(
                { tableNumber: ___.___ }, // TODO: Cari meja berdasarkan tableNumber
                { status: '___' } // TODO: Update status meja menjadi 'available'
            );
        }

        // TODO: Kirim respons sukses dengan data pesanan yang sudah diupdate
        ___.status(___).json({ success: true, data: ___ });
    } catch (___) {
        // TODO: Tangani error dan kirim respons error
        ___.status(___).json({ success: false, error: ___.message });
    }
};
```

Routes

```
router.get("/order/:__________/status", orderController.__________); // TODO:
Lengkapi parameter dan fungsi yang dipanggil
```