Error Handling di Node.js

Pengertian Error Handling:

Error handling adalah proses menangani kesalahan yang terjadi selama eksekusi program. Dalam Node.js, error handling sangat penting untuk memastikan aplikasi tetap stabil dan dapat memberikan feedback yang jelas kepada pengguna ketika terjadi kesalahan. Tanpa penanganan error yang baik, aplikasi dapat mengalami crash atau menghasilkan perilaku yang tidak diinginkan.

Jenis-jenis Error di Node.js:

1. Operational Errors: Kesalahan yang terjadi selama runtime, seperti kegagalan koneksi database, invalid input, atau resource yang tidak ditemukan.
2. Programmer Errors: Kesalahan yang disebabkan oleh bug dalam kode, seperti syntax error, logical error, atau penggunaan API yang salah.

Karakteristik Error Handling di Node.js:

1. Try-Catch Block: Digunakan untuk menangkap error yang terjadi dalam blok kode synchronous.
2. Error-First Callback: Pola callback di Node.js yang menggunakan parameter pertama sebagai error object.
3. Promise Rejection: Error handling dalam Promise menggunakan .catch() atau try-catch dengan async/await.
4. Global Error Handling: Menangkap error yang tidak tertangkap di level aplikasi menggunakan process.on('uncaughtException') dan process.on('unhandledRejection').

Deskripsi Modul:

Pada praktikum ini, mahasiswa akan mempelajari cara menangani error dalam aplikasi Node.js. Fokus utama modul ini adalah implementasi error handling yang baik dalam konteks sistem reservasi restoran. Mahasiswa akan belajar cara menangani error yang terjadi selama operasi database, validasi input, dan operasi asynchronous lainnya. Modul ini juga akan membahas penggunaan middleware untuk menangani error di level aplikasi.

Tujuan Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu:

1. Memahami jenis-jenis error yang dapat terjadi dalam aplikasi Node.js.
2. Mengimplementasikan error handling menggunakan try-catch, error-first callback, dan Promise rejection.
3. Membuat middleware untuk menangani error di level aplikasi.
4. Menerapkan global error handling untuk menangkap error yang tidak tertangkap.
5. Mengintegrasikan error handling dengan modul lain dalam sistem reservasi restoran.

Kesimpulan

Modul ini dirancang untuk membantu mahasiswa memahami dan mengimplementasikan error handling yang baik dalam aplikasi Node.js. Dengan mengikuti modul ini, mahasiswa akan memahami cara menangani error yang terjadi selama runtime, membuat aplikasi yang lebih stabil, dan memberikan feedback yang jelas kepada pengguna ketika terjadi kesalahan.

Praktikum 5: Implementasi Error Handling di Node.js

Langkah 1: Setup Error Handling Middleware

1. Buat folder middleware di dalam src sehingga struktur folder menjadi seperti dibawah:

```
restaurant-reservation/
├── node_modules
├── src/
│   ├── routes/
│   ├── controllers/
│   ├── models/
│   ├── config/
│   ├── middleware/
├── app.js
├── server.js
├── .env
├── .gitignore
├── package-lock.json
├── package.json
```

2. Buat file errorHandler.js di folder src/middleware/errorHandler.js
3. Tambahkan kode berikut di dalam file errorHandler.js

```
const errorHandler = (err, req, res, next) => {
    console.error(err.stack);

    // Default status code dan pesan error
    const statusCode = err.statusCode || 500;
    const message = err.message || 'Internal Server Error';

    res.status(statusCode).json({
        success: false,
        error: message
    });
};

module.exports = errorHandler;
```

Langkah 2: Implementasi Error Handling di Controller

1. Buka file orderController.js di folder src/controllers/orderController.js.
2. Tambahkan error handling pada fungsi createOrder

```
const createOrder = async (req, res, next) => {
    const { tableNumber, items } = req.body;
    try {
        // Cek apakah meja tersedia
        const meja = await Meja.findOne({ tableNumber, status: 'available' });
        if (!meja) {
            const error = new Error('Meja tidak tersedia atau sedang dipesan');
            error.statusCode = 400;
            throw error;}
        // Cek apakah semua item menu valid
        const menuItems = await Menu.find({ _id: { $in: items.map(item =>
item.menuId) } });
        if (menuItems.length !== items.length) {
            const error = new Error('Beberapa item menu tidak valid');
            error.statusCode = 400;
            throw error;}
        // 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) {
        next(error); // Lanjutkan ke error handling middleware
    }
};
```

Langkah 3: Implementasi Global Error Handling

1. Buka file app.js di folder utama proyek.
2. Import error handling dan tambahkan kode berikut (tambahkan setelah route)

```
const errorHandler = require('./src/middleware/errorHandler');


app.use(errorHandler);
```

Langkah 4: Testing dengan Postman

1. Gunakan endpoint create order untuk melakukan pengujian error handle ini
2. Create order dengan meja tidak tersedia (menghasilkan status error code 400)
3. Create order dengan item menu tidak valid (menghasilkan status error code 400)
4. Global error handling dengan mencoba endpoint yang tidak ada (menghasilkan status error code 404)