Asynchronous Programming Dengan Callback di Node.js

Pengertian Callback:

Callback adalah fungsi yang diberikan sebagai argumen ke fungsi lain dan dieksekusi setelah fungsi utama selesai menjalankan tugasnya. Dalam konteks Node.js, callback sangat penting untuk menangani operasi asynchronous seperti membaca file, mengakses database, atau melakukan HTTP request.

Karakteristik callback di nodejs**:**

1. Error-First Pattern: Callback di Node.js mengikuti pola "error-first", di mana parameter pertama selalu merupakan error object (jika ada error) dan parameter kedua adalah hasil sukses.
2. Non-blocking: Callback memungkinkan kode untuk tetap berjalan tanpa harus menunggu operasi yang memakan waktu selesai.
3. Asynchronous: Eksekusi callback tidak menghalangi eksekusi kode lainnya.

Deskripsi Modul:

Pada praktikum ini, mahasiswa akan melanjutkan pengembangan aplikasi backend menggunakan Node.js dengan fokus pada pembuatan sistem reservasi restoran. Modul kedua ini akan membahas implementasi callback pattern dalam mengelola menu restoran. Mahasiswa akan mempelajari cara menangani operasi asynchronous menggunakan callback, memahami konsep non-blocking I/O, serta mengimplementasikan operasi Create dan Get All menu items menggunakan pendekatan callback yang didukung oleh Promise.

Perlu dicatat bahwa pada Mongoose versi terbaru, penggunaan callback murni tidak lagi didukung. Oleh karena itu, mahasiswa harus menggunakan pendekatan async-await atau .then().catch() untuk menangani operasi database secara asinkron. Meskipun demikian, praktikum ini tetap menerapkan callback pattern dengan cara membungkus Promise di dalam callback, sehingga mahasiswa tetap memahami cara kerja callback dalam menangani operasi asinkron tanpa menggunakan callback murni secara langsung.

Tujuan Pembelajaran:

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu:

1. Memahami konsep callback dan perbedaannya dengan pendekatan asynchronous lainnya seperti Promise dan Async/Await.

2. Mengimplementasikan operasi asynchronous menggunakan callback dalam konteks manajemen menu restoran.
3. Membuat endpoint untuk manajemen menu (create, read) dengan menggunakan callback.
4. Memahami pola error-first callback dan cara menangani error dalam operasi asynchronous.
5. Mengintegrasikan manajemen menu dengan modul lain dalam sistem reservasi restoran.

Kesimpulan:

Modul ini dirancang untuk membantu mahasiswa memahami dan mengimplementasikan callback dalam konteks manajemen menu restoran. Dengan mengikuti modul ini, mahasiswa akan memahami cara kerja callback, mampu membuat endpoint untuk manajemen menu, menerapkan pola error-first callback, serta mengintegrasikan manajemen menu dengan modul lain dalam sistem reservasi restoran. Dengan demikian, mahasiswa akan memiliki pemahaman yang kuat tentang pengembangan aplikasi backend menggunakan Node.js dan callback.

Praktikum 2: Implementasi Menu Management dengan Callback

Langkah 1: Setup Model Menu

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

```
const mongoose = require('mongoose');


const menuSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  description: String,
  price: {
    type: Number,
    required: true
  },
  category: {
    type: String,
    enum: ['appetizer', 'main', 'dessert', 'beverage']
  },
  isAvailable: {
    type: Boolean,
    default: true
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});
module.exports = mongoose.model('Menu', menuSchema);
```

Langkah 2: Implementasi Menu Controller

1. Buat file menuController.js di dalam folder src/controllers/menuController.js
2. Tambahkan kode berikut di file menuController.js

```
const Menu = require("../models/menuModel");

// Wrapper untuk mengubah Promise menjadi callback style
function withCallback(promise, callback) {
  promise
    .then(data => callback(null, data))
    .catch(err => callback(err));
}
// Fungsi untuk membuat menu item
const createMenuItem = (req, res) => {
  const newMenuItem = new Menu(req.body);


  withCallback(newMenuItem.save(), (err, savedItem) => {
    if (err) return res.status(500).json({ error: err.message });
    res.status(201).json(savedItem);
  });
};
// Fungsi untuk mengambil semua menu item
const getAllMenuItems = (req, res) => {
  withCallback(Menu.find({}), (err, items) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(items);
  });
};
// Ekspor fungsi-fungsi sebagai objek
module.exports = {
  getAllMenuItems,
  createMenuItem
}
```

Langkah 3: Implementasi Routes

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

```
const express = require("express");

const menuController =
require("../controllers/menuController");

const router = express.Router();


router.get("/menu", menuController.getAllMenuItems);

router.post("/createMenu", menuController.createMenuItem);


module.exports = router;
```

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

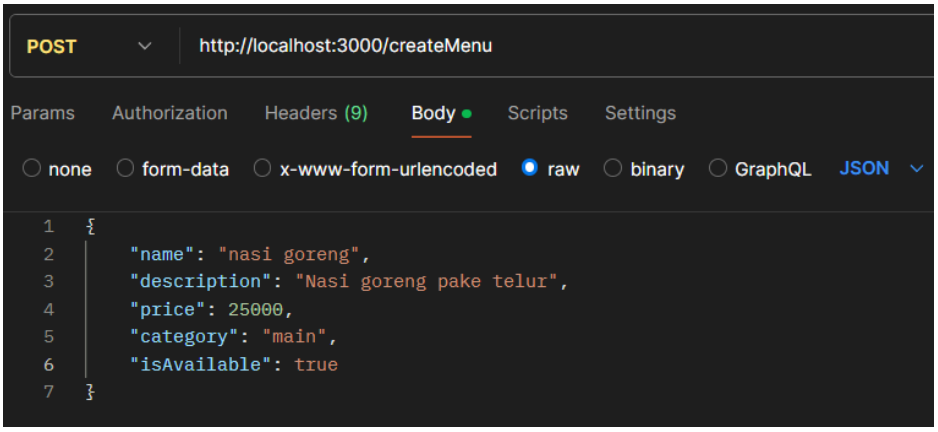
```
const menuRoutes = require('./src/routes/menuRoutes');
```

4. Selanjutnya tambahkan kode berikut di file app.js sebelum module.exports

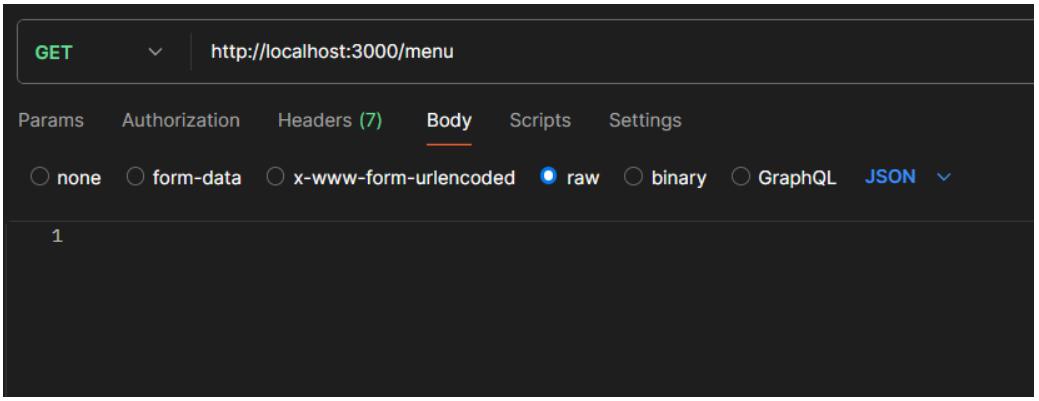
```
// ...

app.get('/test', (req, res) => {

  //...

});

app.use('/', menuRoutes);

module.exports = { app };
```

Langkah 4: Testing dengan Postman

1. Create Menu Item (POST/ http://localhost:{port}/createMenu)



2. Get All Menu Item (GET/ http://localhost:{port}/menu)



Tase Case

1. Buatlah Get Menu by Category dengan clue sebagai berikut

Controllers

```
const getMenuByCategory = (req, res) => {

    const category = _____________; // TODO: Ambil kategori dari
parameter URL



    withCallback(Menu.find({ _____________ }), (err, items) => { //
TODO: Lengkapi query untuk mencari berdasarkan kategori

        if (err) return res.status(500).json({ error: _____________
}); // TODO: Kirimkan pesan error jika terjadi kesalahan



        if (_____________) // TODO: Periksa apakah menu ditemukan

            return res.status(404).json({ error: `Menu with category
'${category}' not found` });



        res.json(_____________); // TODO: Kirimkan hasil query dalam
response

    });

};
```

Routes

```
router.get("/menu/:_____________", menuController._____________); //
TODO: Lengkapi parameter dan fungsi yang dipanggil
```