Pengenalan Asynchronous Programming Pada Node.js untuk Backend Development

Pengertian:

Node.js adalah platform runtime JavaScript berbasis server yang banyak digunakan untuk pengembangan backend. Salah satu keunggulan utama Node.js adalah kemampuannya dalam menangani proses yang berjalan secara bersamaan tanpa harus menunggu satu per satu, sehingga sangat cocok untuk aplikasi dengan banyak permintaan, seperti sistem reservasi restoran. Dalam kursus ini, kita akan membangun sistem reservasi restoran secara bertahap menggunakan tiga cara berbeda untuk menangani proses yang berjalan bersamaan: Callbacks, Promises, dan Async/Await.

Deskripsi Modul:

Pada praktikum ini, mahasiswa akan memulai pengembangan aplikasi menggunakan Node.js yang berfokus pada pembuatan sistem reservasi restoran. Langkah awal yang akan dilakukan adalah setup project Node.js, dimulai dari inisialisasi project menggunakan npm (Node Package Manager), membuat struktur direktori yang terorganisir, dan mengkonfigurasi file package.json untuk mengelola dependensi. Praktikum pada modul pertama ini akan fokus pada persiapan dan setup project, yang akan menjadi fondasi untuk mempelajari dan menerapkan konsep asynchronous programming pada modul-modul selanjutnya. Dengan persiapan yang baik, mahasiswa akan lebih mudah memahami dan mengimplementasikan fitur-fitur asynchronous dalam pengembangan sistem reservasi restoran.

Persyaratan Sistem untuk Mengikuti Praktikum:

Spesifikasi Hardware:

1. OS: Windows 10/11, macOS 10.15+ (Catalina atau lebih baru), atau Linux (Ubuntu 20.04 LTS atau lebih baru)
2. CPU: Intel Core i5 / AMD Ryzen 5 atau lebih tinggi
3. RAM: 8GB minimum (16GB atau lebih direkomendasikan untuk performa optimal)
4. Storage: Minimal 20GB ruang kosong (SSD sangat disarankan untuk kecepatan yang lebih baik)

Software yang diperlukan:

1. Node.js: v16.x LTS atau lebih baru (v18.x atau terbaru direkomendasikan untuk fitur terbaru dan dukungan jangka panjang)
2. MongoDB Atlas: Untuk menyimpan dan mengelola data
3. Visual Studio Code
4. Postman: Untuk pengujian API secara lebih lengkap dan interaktif

Sebelum memulai praktikum, pastikan Node.js dan editor kode sudah terinstal dengan baik. Selain itu, perbarui sistem operasi dan perangkat lunak ke versi terbaru agar sesuai dengan teknologi yang akan digunakan. Koneksi internet yang stabil juga dibutuhkan untuk mengunduh berbagai paket dan dependensi yang diperlukan. Dengan persiapan ini, mahasiswa dapat mengikuti praktikum ini dengan lebih mudah dan lancar.

Praktikum 1: Setup Project dan Pengenalan Dasar

Langkah 1: Membuat Struktur Project

1. Buat folder project baru

```
mkdir restaurant-reservation
cd restaurant-reservation
```

2. Inisialisasi project Node.js

```
npm init -y
```

3. Buat stuktur folder

```
mkdir src
mkdir src\routes src\controller src\models src\config
```

Struktur Folder

```
restaurant-reservation/
├── src/
│   ├── routes/
│   ├── controllers/
│   ├── models/
│   ├── config/
```

4. Buat file utama

```
restaurant-reservation/
├── src/
│   ├── routes/
│   ├── controllers/
│   ├── models/
│   ├── config/
│   │   ├── database.js
├── app.js
├── server.js
```

```
├── .env
├── .gitignore
├── package.json
```

Langkah 2: Instalasi Dependencies

1. Install dependencies utama

```
npm install express mongodb mongoose dotenv
```

Struktur folder akan menjadi seperti dibawah:

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

2. Install dependencies development

```
npm install --save-dev nodemon
```

3. Edit file package.json untuk menambahkan script:

```
"scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
}
```

Edit main pada package.json dari index.js menjadi app.js

```
"main": "app.js"
```

Full package.json

```
{
  "name": "restaurant-reservation",
  "version": "1.0.0",
  "description": "1. **Buat folder
proyek**  \r    ```sh\r    npm init -y",
  "main": "app.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
 },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "dotenv": "^16.4.7",
    "express": "^4.21.2",
    "mongodb": "^6.13.1",
    "mongoose": "^8.10.1"
  },
  "devDependencies": {
    "nodemon": "^3.1.9"
  }
}
```

Langkah 3: Membuat Server Sederhana

1. Buat file **.env**

```
PORT=3000

MONGODB_URL=mongodb+srv://<username>:<password>@<cluster-url>
<nama_database>?retryWrites=true&w=majority
```

2. Buat dan edit file **app.js**

```
// app.js
const express = require('express');
const dotenv = require('dotenv');

// Load environment variables
dotenv.config();

// Create Express app
const app = express();

// Middleware
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Test route
app.get('/test', (req, res) => {
  res.json({message: 'Welcome to Restaurant Reservation API'});
});

module.exports = {app};
```

3. Buat dan edit file **Server.js**

```
const {app} = require('./app');
const connectDB = require('./src/config/database');


const PORT = process.env.PORT || 3000;


// Start server
const server = app.listen(PORT, async () => {
  // Connect to database
  await connectDB();
  console.log(`Server running on port ${PORT}`);
});
module.exports = {server};
```

Langkah 4: Konfigurasi Koneksi Database Dan Run Server

1. Edit file **src/config/database.js:**

```
const mongoose = require('mongoose');
require('dotenv').config();


const connectDB = async () => {
    try {
        await mongoose.connect(process.env.MONGODB_URL);
        console.log('✅ MongoDB Connected Successfully');
    } catch (error) {
        console.error('❌ Error connecting to MongoDB Atlas:',
error.message);
        process.exit(1);
    }
};
module.exports = connectDB;
```

2. Jalankan server

```
npm run dev
```

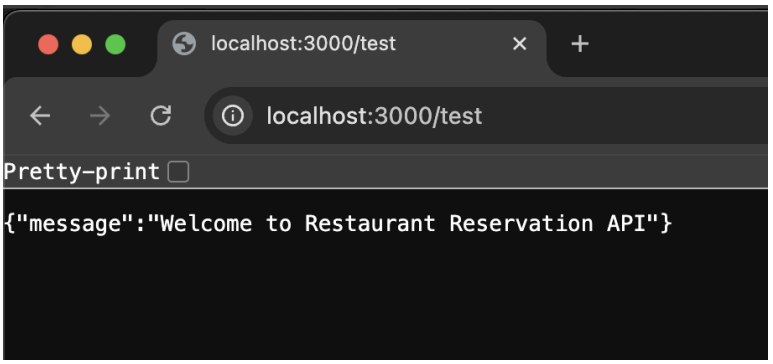
3. Jika berhasil akan seperti ini

```
PS B:\restaurant-reservation> npm run dev

> restaurant-reservation@1.0.0 dev
> nodemon app.js

[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
✅ MongoDB Connected Successfully
Server running on port 3000
```

4. Buka browser http://localhost:(port) /test



Task

1. Buat koneksi ke MongoDB menggunakan mongoose di file **database.js**
2. Buat agar respons dari test route menjadi seperti dibawah:

```
{"status":"success","message":"Welcome to Restaurant Reservation API","version":"1.0.0"}
```