SOMNOG 7

Somali Network Operators Group

WORKSHOPS AND CONFERENCE

22 - 26 December, 2024

THEME:

Digital Public Infrastructure: Laying the Foundations for Somalia's Digital Future

Bashir Ahmed Mohamud

Software developer at **Tabaarakict solution**

Node.js Overview

- **Node.js** is a runtime that allows you to execute JavaScript on the server side.
- Built on Chrome's V8 JavaScript engine.
- Used to create scalable, event-driven, and non-blocking I/O applications.

Setting Up the Environment

- 1. Install Node.js from nodejs.org.
- 2. Install MongoDB locally or set up MongoDB Atlas for cloud storage.
- 3. Install a code editor like VS Code.
- 4. Verify installations:
- - Node.js: `node -v`
- - MongoDB: `mongod --version`.

Building a Node.js Server


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

const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello World\n');
});

server.listen(3000, () => {
  console.log('Server running at http://localhost:3000/');
});
```

Express.js

- A web application framework for Node.js.
- Simplifies the creation of robust web applications and APIs.
- Provides middleware for handling requests, responses, and routing.

Key Features:

- Minimal and flexible.
- Middleware support.
- Powerful routing system.

.

Building RESTful APIs with Express.js

- 1. Create routes:
- - GET: Fetch data.
- - POST: Add data.
- - PUT: Update data.
- - DELETE: Remove data.

Example: Express "Hello World"


```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(3000, () => {
  console.log('Server running on http://localhost:3000/');
});
```

MongoDB

- NoSQL database that stores data in JSON-like documents.
- Highly scalable and flexible.
- Works well with JavaScript and Node.js.

Key Features:

- Document-oriented.
- Schema-less (or schema-flexible).
- Supports rich querying and indexing.

Setting Up MongoDB

- 1. Install Mongoose for MongoDB: `npm install mongoose`.
- 2. Connect to MongoDB:
- ```javascript
- const mongoose = require('mongoose');
- mongoose.connect('mongodb://localhost:27017/mydb')

Advanced MongoDB Queries

- 1. Aggregation:
- - Example: Group by age:
- ```javascript
- User.aggregate([{ $group: { _id: '$age', count: { $sum: 1 } } }]);
- ```
- 2. Lookup (joins):
- ```javascript
- Orders.aggregate([{ $lookup: { from: 'users', localField: 'userId', foreignField: '_id', as: 'userDetails' } }]);

Building Track Registration System: Project Setup & Steps

Initialize Project

- Create a project folder and initialize with `npm init -y`.
- Install dependencies: `express`, `mongoose`, and `body-parser`.

Define File Structure

Set Up Components:

- **index.js**: Set up Express app, connect to MongoDB, define routes.
- **Database Connection**: Configure MongoDB connection in `db/connect.js`.
- **Models**: Create Mongoose schema for track details in `models/Track.js`.
- **Controllers**: Define CRUD operations for tracks in `controllers/trackController.js`.
- **Routes**: Set up API endpoints for track registration in `routes/trackRoutes.js`.

Run Project:

- Start server with `npm run dev or node index.js`.
- Test CRUD API using Postman or `curl`.

Thank You!