Ex No 4 **WRITE CUSTOM APPLICATIONS WITH NODE.JS AND MONGO DB**

Date:

Aim

To design and develop a custom CRUD application using Node.js as backend and MongoDB.

Algorithm

Step 1: Start

Step 2: Initialize Node.js Project

- Use the command: npm init -y
- Install required packages: npm install express mongoose dotenv

Step 3: Create Project Structure

- Create folders:
  - models/ → to define schema
  - routes/ → to define API routes
  - controllers/ *(optional)* → for logic separation

Step 4: Configure MongoDB Connection

- Create .env file to store connection string
- Connect MongoDB using mongoose.connect() in the main file

Step 5: Define Data Schema

- In models/, create a Mongoose schema with relevant fields
- Export the schema model

Step 6: Create RESTful API Endpoints

- Define CRUD routes:
  - **POST** for inserting data
  - **GET** for retrieving data
  - **PUT** for updating data
  - **DELETE** for deleting data

Step 7: Use Express Middleware

- Use express.json() to parse incoming JSON data

Step 8: Handle Database Operations

- Use Mongoose methods like save(), find(), findByIdAndUpdate(), findByIdAndDelete() inside route functions

Step 9: Start the Server

- Use app.listen(PORT) to run the server

Step 10: Test the Application

- Use Postman or any API testing tool to test the endpoints

Step 11: Stop

Concepts Involved

Node.js

Node.js is a powerful, open-source, and cross-platform JavaScript runtime environment built on Chrome's V8 engine. It allows you to run JavaScript code outside the browser, making it ideal for building scalable server-side and networking applications.

- JavaScript was mainly used for frontend development earlier. With Node.js (Introduced in 2009), JavaScript became a backend language as well.
- Non-blocking, event-driven architecture for high performance.
- Supports the creation of REST APIs, real-time applications, and microservices.
- Comes with a rich library of modules through npm (Node Package Manager).

What is Node.js?

- Node.js is not a programming language like Python, Java or C/C++. Node.js is a runtime, similar to Java virtual machine, that converts JavaScript code into machine code. It is , widely used by thousands of developers around the world to develop I/O intensive web applications like video streaming sites, single-page applications, and other web applications.
- With Node.js, it is possible to use JavaScript as a backend. With JavaScript already being a popular choice for frontend development, application development around MERN (MongoDB, Express, React and Node.js.) and MEAN (MongoDB, Express, Angular and Node.js) stacks is being increasingly employed by developers.

What Can You Build With Node.js?

- Node.js uses an **event-driven**, **non-blocking** model.
- It can handle many connections at once without waiting for one to finish before starting another. This makes it great for real-time apps and high-traffic websites. Here are some examples of what you can build with Node.js:
  - Web servers and websites
  - REST APIs
  - Real-time apps (like chat)
  - Command-line tools
  - Working with files and databases
  - IoT and hardware control

Applications of Node.js

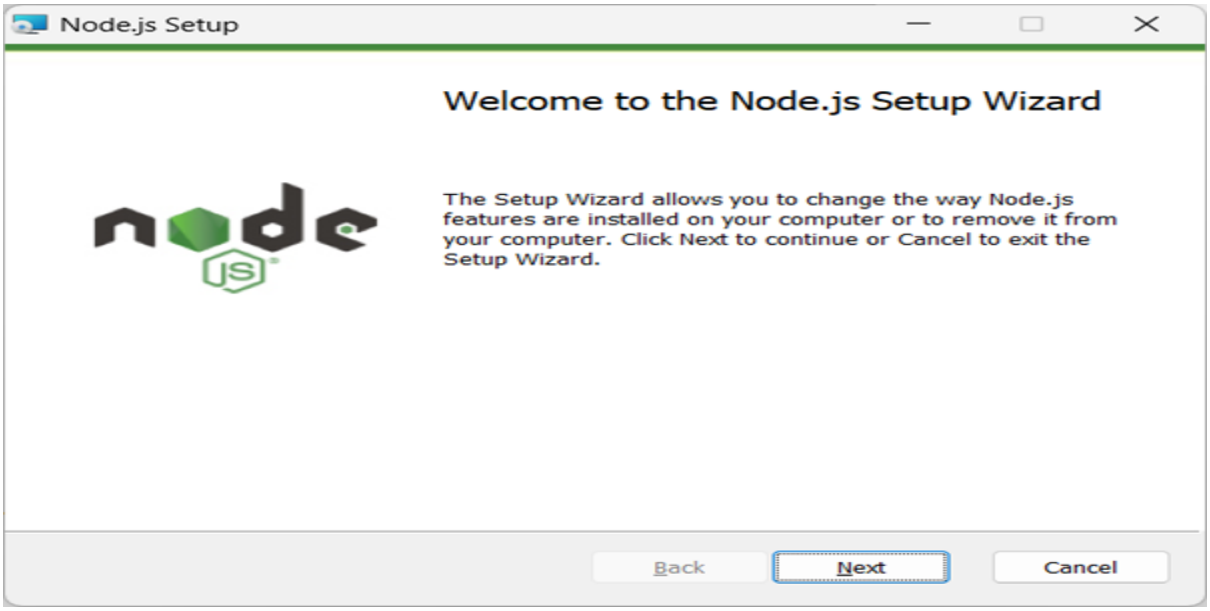
Node.js is used for building different type of applications. Some of the application types are listed below.

- **Streaming applications:** Node.js can easily handle real-time data streams, where it is required to download resources on-demand without overloading the server or the user's local machine. Node.js can also provide quick data synchronization between the server and the client, which improves user experience by minimizing delays using the Node.js event loop.
- **Single page apps:** Node.js is an excellent choice for SPAs because of its capability to efficiently handle asynchronous calls and heavy input/output(I/O) workloads. Data driven SPAs built with Express.js are fast, efficient and robust.
- **Realtime applications:** Node.js is ideal for building lightweight real-time applications, like messaging apps interfaces, chatbot etc. Node.js has an event- based architecture, as a result has an excellent WebSocket support. It facilitates real-time two-way communication between the server and the client.
- **APIs:** At the heart of Node.js is JavaScript. Hence, it becomes handling JSON data is easier. You can therefore build REST based APIs with Node.js.

These are some of the use cases of Node.js. However, its usage is not restricted to these types. Companies are increasingly employing Node.js for variety of applications.

Node.js Environment Setup

Assuming that you are working with Windows 10/Windows 11 powered computer, download the 64-bit installer for Windows: https://nodejs.org/dist/v20.9.0/node-v20.9.0-x64.msi, and start the installation b double-clicking the downloaded file.



The installation takes you through a few steps of the installation wizard. It also adds the installation directory of Node.js executable to the system path.
To verify if Node.js has been successfully installed, open the command prompt and type node -v. If Node.js is installed successfully then it will display the version of the Node.js installed on your machine.

Mongo DB

The **MongoDB** is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database. MongoDB features are flexible data models that allows the storage of unstructured data. This provides full support indexing, replication, capabilities and also user friendly APIs. The **MongoDB** is a multipurpose dataset that is used for modern application development and cloud environments. This scalable architecture enables us to handle system demands and also adding more nodes to distribute the load.

MongoDB Basic Commands

We have a list of standard MongoDb commands to interact with the database, These commands are CREATE, READ, INSERT, UPDATE, DELETE, DROP and AGGREGATE can be classified into following groups based on their nature −

Command	Description
CREATE	Creates a new table in the database and other objects in the database.
INSERT	Inserts collection name in existing database.
DROP	Deletes an entire table or specified objects in the database.
UPDATE	Updates the document into a collection.

Postman or Web Client

Postman and WebClient are both tools for interacting with APIs, but they serve different purposes and operate at different levels. Postman is a standalone API client, primarily used for testing and exploring APIs through a graphical user interface. WebClient, on the other hand, is a reactive HTTP client library built into Spring Boot, used within applications to make API calls programmatically.

Postman:

- Purpose: Primarily for API testing and exploration, offering a user-friendly interface to send requests, view responses, and manage collections of API calls.
- Level: User-level tool, separate from application code.
- Features: Graphical user interface, request building, response inspection, collections, test scripting, and collaboration features.
- Pros: Easy to use for testing, helps with debugging, good for exploring APIs.
- Cons: Can be resource-intensive with large collections, GUI-centric (limited CLI features).

Web Client:

- Purpose: To make HTTP requests from within an application, particularly useful for microservices communication and handling asynchronous operations.
- Level: Library integrated into application code.
- Features: Reactive, non-blocking operations, fluent API for request building, supports both synchronous and asynchronous programming models.
- Pros: Efficient for making API calls within applications, good for building scalable and high-performance applications.
- Cons: Requires coding to use, not designed for manual exploration like Postman.

Packages

Mongoose, Express.js, and dotenv are common packages used together in Node.js applications, particularly when building web APIs with a MongoDB database.

Express.js:

- This is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It simplifies the process of handling HTTP requests, routing, middleware, and creating RESTful APIs.

Mongoose:

- This is an Object Data Modeling (ODM) library for MongoDB and Node.js. Mongoose provides a straightforward, schema-based solution to model application data. It allows developers to define models, perform CRUD (Create, Read, Update, Delete) operations on MongoDB documents, and manage relationships between data.

dotenv:

- This package loads environment variables from a .env file into process.env. It is primarily used to manage sensitive information like database connection strings, API keys, or port numbers, keeping them separate from the codebase and allowing for easy configuration changes across different environments (development, production, etc.).

Installation: All three packages are typically installed using npm:

npm install express mongoose dotenv

dotenv setup: A .env file is created in the project root to store environment variables (e.g., MONGO_URI, PORT). In the main application file (e.g., app.js or server.js), dotenv is initialized at the very top to load these variables:

```
require('dotenv').config();
```

Mongoose connection: Mongoose uses the MongoDB URI from process.env to establish a connection to the database:

```
const mongoose = require('mongoose');
mongoose.connect(process.env.MONGO_URI)
    .then(() => console.log('MongoDB connected'))
    .catch(err => console.error(err));
```

Express server: Express is used to create and configure the web server, define routes, and handle requests. Mongoose models are then used within these routes to interact with the MongoDB database.

```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;

// ... define routes and use Mongoose models ...

app.listen(port, () => {
console.log(`Server running on port ${port}`);
});
```

RESULT:

Thus, A custom application was successfully created using Node.js and MongoDB. The project performed basic CRUD operations through RESTful APIs and was tested using Postman. The same development process can be applied across various domains by changing the data schema.