Full Stack Development – Lab Program
(AD2604-1)

Program 9: Product Management System using Node.js, Express.js, and MongoDB

Aim: To build a backend-based Product Management System using Node.js, Express.js, and MongoDB, allowing users to add and retrieve product details through a RESTful API. The system uses Mongoose for database interaction and can be tested using Postman for API requests and MongoDB Compass for viewing stored data.

Explanation:

- This program is a backend-based Product Management System built using Node.js, Express.js, and MongoDB to handle product data through a RESTful API.
- It allows users to add and retrieve product details via API requests.
- Express.js is used to create the backend server and handle HTTP requests.
- MongoDB stores product details, and Mongoose manages database interactions with a structured schema.
- CORS is enabled for secure frontend-backend communication.
- The system provides two main API routes:
- POST /products to add a product with details like name, price, and category.
- GET /products to fetch all stored products from the database.
- Postman is used to test the API by sending requests and verifying responses.
- MongoDB Compass is used to view and manage stored product data.
- The server runs on port 5000, listening for API requests.
- This program serves as a backend solution for managing product data and can be integrated with a frontend application.

Code:

# Open a terminal and create a folder for the backend:

```
mkdir backend
cd backend
# Initialize a Node.js project:
npm init -y
# Install the required dependencies
npm install express cors mongoose
```

# Create a new file named index.js inside the backend folder and add the following code:

```
const express = require("express"); // Import Express framework for building web server
const mongoose = require("mongoose"); // Import Mongoose to interact with MongoDB
const cors = require("cors"); // Import CORS to allow cross-origin requests

const app = express(); // Create an instance of Express application
const PORT = 5000; // Define the port number where the server will run

app.use(cors()); // Enable CORS to allow requests from different origins
app.use(express.json()); // Enable parsing of JSON data in request bodies
// MongoDB Connection Setup
// Copy the MongoDB URL from MongoDB Compass
const mongoDBUrl = "mongodb://localhost:2000";

// Function to connect to MongoDB
const connectDB = async () => {
   try {
      await mongoose.connect(mongoDBUrl, {
         useNewUrlParser: true, // Use new URL parser
         useUnifiedTopology: true, // Use modern server discovery and monitoring engine
      });
      console.log("MongoDB Connected"); // Log success message
   } catch (err) {
      console.error("MongoDB Connection Error:", err); // Log error if connection fails
      process.exit(1); // Exit the program in case of failure
   }
};

// Call the function to connect to MongoDB
connectDB();

// Define Product Schema using Mongoose
const ProductSchema = new mongoose.Schema({
   name: String, // Product name (String type)
   price: Number, // Product price (Number type)
   category: String, // Product category (String type)
});

// Create Product model from schema
const Product = mongoose.model("Product", ProductSchema);

// Route to insert a new product
app.post("/products", async (req, res) => {
   try {
      // Extract product details from request body
      const { name, price, category } = req.body;

      // Create a new Product object using the received data
      const newProduct = new Product({ name, price, category });
      await newProduct.save(); // Save the product in the database

      // Send a success response with the saved product details
      res.status(201).json({ message: "Product added", product: newProduct });
   } catch (err) {
      // If any error occurs, send an error response
      res.status(500).json({ error: err.message });
   }
});
```

```
// Route to get all products
app.get("/products", async (req, res) => {
  try {
    const products = await Product.find(); // Fetch all products from the database
    res.json(products); // Send the products as a JSON response
  } catch (err) {
    res.status(500).json({ error: err.message }); // Handle errors
  }
});

// Start the Express server and listen on the specified PORT
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

#Run the Node.js server:

```
node index
```

Output:

How to Test This API

- Add a Product (POST request)

Open Postman and select POST as the request type.
http://localhost:5000/products
Go to the Body tab, select raw, and choose JSON format.
Enter the following JSON data:

```
{
 "name": "Laptop",
 "price": 50000,
 "category": "Electronics"
}
```

Click Send

Response:

```
{
 "message": "Product added",
 "product": {
  "_id": "60d0fe4f5311236168a109ca",
  "name": "Laptop",
  "price": 50000,
  "category": "Electronics",
  "__v": 0
 }
}
```

- Get All Products (GET request)

Open Postman and select GET as the request type.
Enter the request URL:
http://localhost:5000/products
Click Send

Response:

```
[
  {
    "_id": "60d0fe4f5311236168a109ca",
    "name": "Laptop",
    "price": 50000,
    "category": "Electronics",
    "__v": 0
  }
]
```