
Day 3 — JavaScript Deep Dive + Backend Foundations

MCA – SEM : - 2 **Assi. Date. :- 04-03-2026**

Day 3 — Practical Task **NAME:- Bhautik khandhala**

Technologies Used

- Node.js
- Express.js
- MongoDB
- Mongoose
- JWT Authentication
- Bcryptjs
- Postman (API Testing)

Project Folder Structure:

```
project
│
├── config
│     db.js
│
├── controllers
│     userController.js
│     productController.js
│
├── models
│     User.js
│     Product.js
│
├── routes
│     userRoutes.js
│     productRoutes.js
│
├── middleware
│     authMiddleware.js
│
└── server.js
```

Database Configuration (db.js)

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

const connectDB = async () => {
  try {
    await mongoose.connect("mongodb://localhost:27017/studentDB");
    console.log("MongoDB Connected");
  } catch (err) {
    console.log(err);
    process.exit(1);
  }
};

module.exports = connectDB;
```

The `db.js` file establishes the connection between the Node.js application and MongoDB using Mongoose. The `connectDB()` function connects to the database `studentDB`.

User Model (User.js)

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

const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
```

```
    required: true,
    minlength: 6
  },
  role: {
    type: String,
    enum: ["admin", "user"],
    default: "user"
  }
}, { timestamps: true });
module.exports = mongoose.model("User", userSchema);
```

Product Model (Product.js)

```
const mongoose = require("mongoose");
const productSchema = new
  mongoose.Schema({ name: {
    type: String,
    required: true
  },
  price: {
    type: Number,
    required: true,
    min: 0
  },
  description: String,
  stock: {
    type: Number,
    default: 0
```

```
  }
}, { timestamps: true });
module.exports = mongoose.model("Product", productSchema);
```

User Controller (userController.js)

```
const User = require("../models/User");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");
exports.getdata = async (req, res) => {
    res.status(200).json({ message: "welcome to user controller"
}); }
exports.register = async (req, res) => {
    console.log("testing");
  try {
    // res.status(500).json({ message: "message" });
    console.log(req.body);
    const { name, email, password } = req.body;
    const existingUser = await User.findOne({ email });
    if(existingUser)
      return res.status(400).json({ message: "Email already existing"
    }); const hashedPassword = await bcrypt.hash(password, 10);
    const user = await User.create({
      name,
      email,
      password: hashedPassword
    });
```

```
    res.status(201).json(user);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
exports.login = async (req, res) => {
  try {
    const { email, password } = req.body;
    const user = await User.findOne({ email });
    if (!user)
      return res.status(400).json({ message: "Invalid credentials" });
    const isMatch = await bcrypt.compare(password,
    user.password); if (!isMatch)
      return res.status(400).json({ message: "Invalid credentials"
    }); const token = jwt.sign({ id: user._id }, "secretkey", {
      expiresIn: "1d"
    });
    res.json({ token });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
```

Product Controller (productController.js)

```
const Product = require("../models/Product");

exports.createProduct = async (req, res) => {
  try {
    const product = await Product.create(req.body);
```

```
    res.status(201).json(product);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};

exports.getProducts = async (req, res) => {
  const products = await Product.find();
  res.json(products);
};

exports.getProduct = async (req, res) => {
  const product = await Product.findById(req.params.id);
  res.json(product);
};

exports.updateProduct = async (req, res) => {
  const product = await Product.findByIdAndUpdate(
    req.params.id,
    req.body,
   { new: true }
  );
  res.json(product);
};

exports.deleteProduct = async (req, res) => {
  await Product.findByIdAndDelete(req.params.id);
  res.json({ message: "Product deleted" });
};
```

Middleware (authMiddleware.js)

```
const jwt = require("jsonwebtoken");

const User = require("../models/User");

const authMiddleware = async(req, res, next) => {

    try {

        const token = req. headers.authorization;

        if(!token) {

            return res.status(401).json({ message: "Access denied. No token provided."

        }); }

        const decoded = jwt.verify(token, "secretkey");

        const user = await User.findById(decoded.id).select("-password");

        if(!user) {

            return res.status(401).json({ message: "User not found" });

        }

        req.user = user;

        next();

    } catch (error) {
        res.status(401).json({ message: "Invalid Token" });

    }

};

module.exports = authMiddleware;
```

API Routes

User Routes (userRoutes.js)

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

const router = express.Router();
```

const userController =

require("../controllers/userController"); router.post("/register",

userController.register);

router.post("/login", userController.login);

// router.post("/login", userController.login);

module.exports = router;

Product Routes

(productRoutes.js) const express =

require("express");

const router = express.Router();

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

const authMiddleware = require("../middleware/authMiddleware");

router.post("/",authMiddleware, productController.createProduct);

router.get("/",authMiddleware, productController.getProducts);

router.get("/:id",authMiddleware, productController.getProduct);

router.put("/:id",authMiddleware, productController.updateProduct);

router.delete("/:id",authMiddleware,

productController.deleteProduct); module.exports = router;

Postman Testing

Register User

Request: POST /api/users/register

Body:

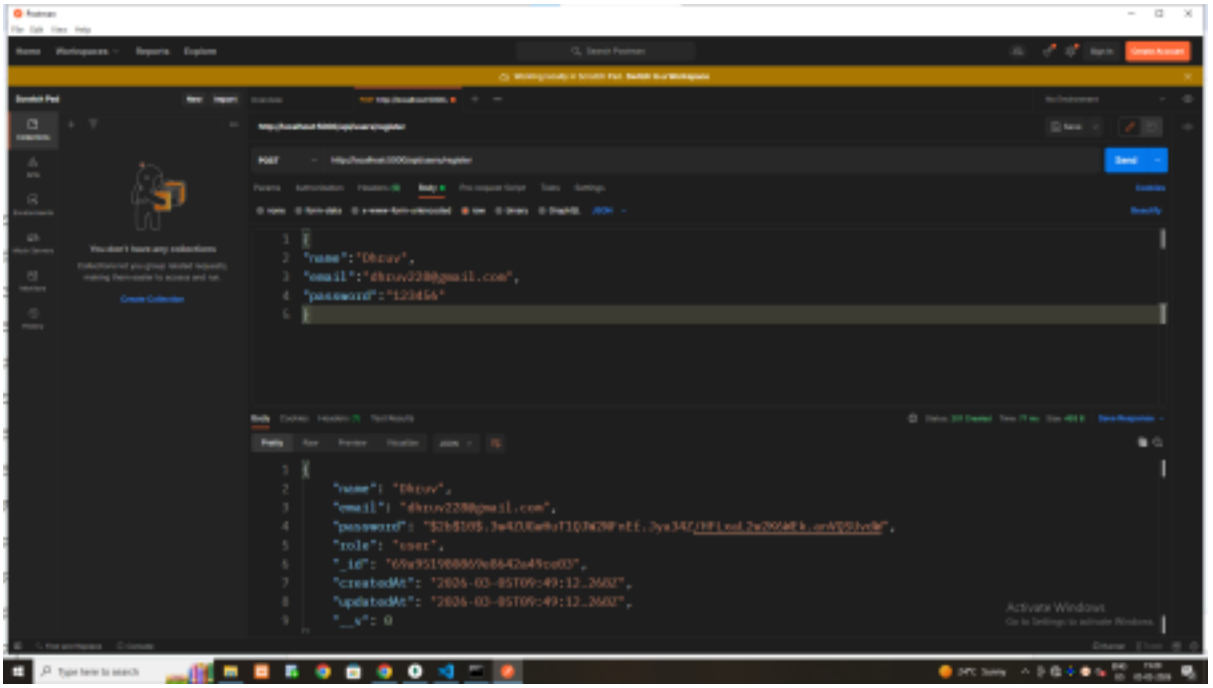
{

"name":"Dhruv",

"email":"dhruv228@gmail.com",

"password":"123456"

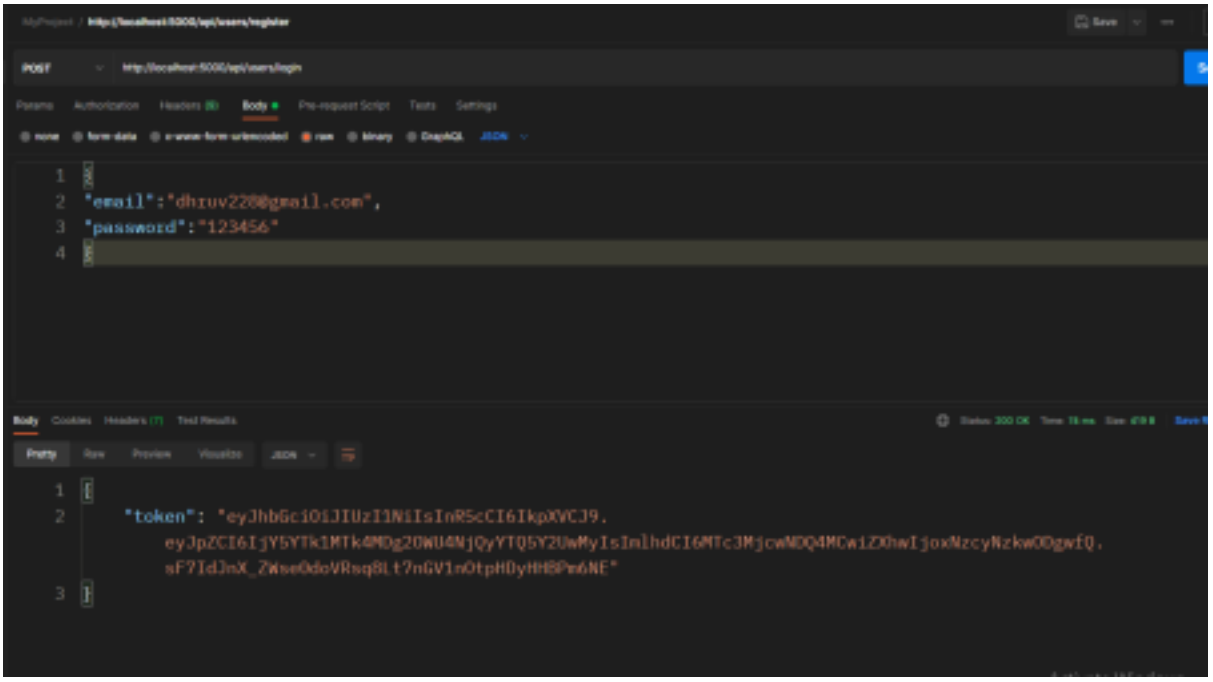
}



Login User

Request: POST /api/users/login
Response:

token: JWT_TOKEN



Create Product

Header: Authorization: TOKEN

Body:

{

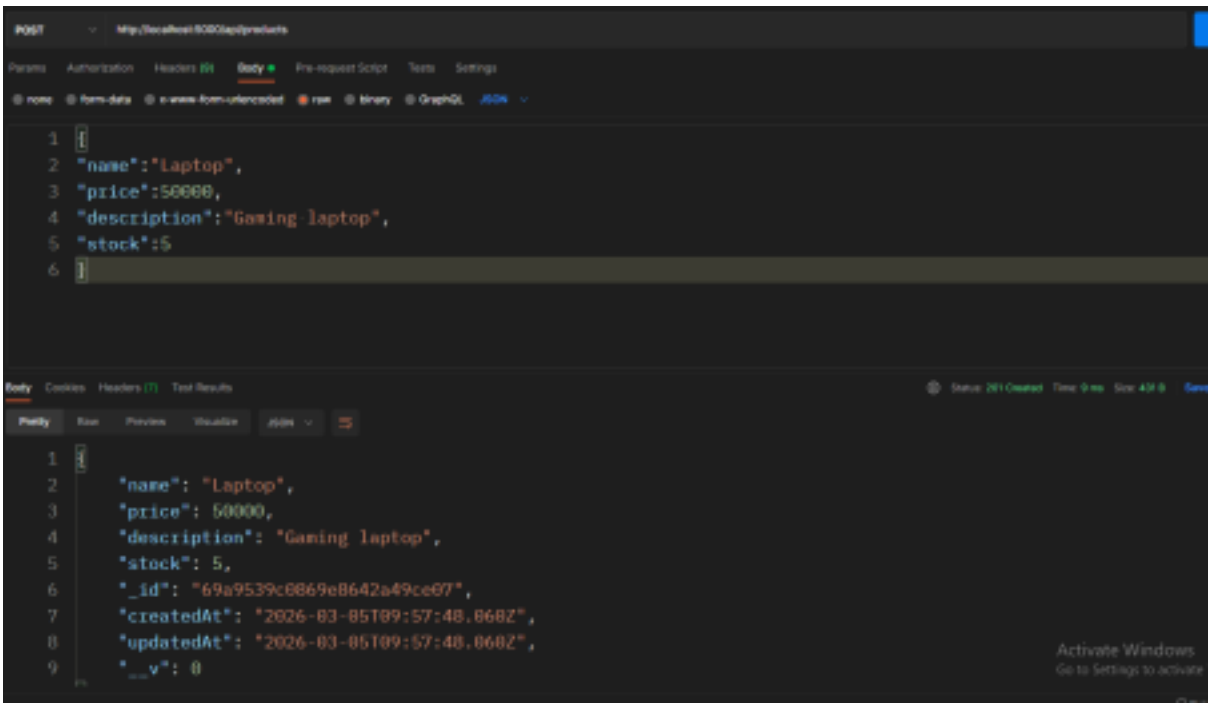
"name":"Laptop",

"price":50000,

"description":"Gaming laptop",

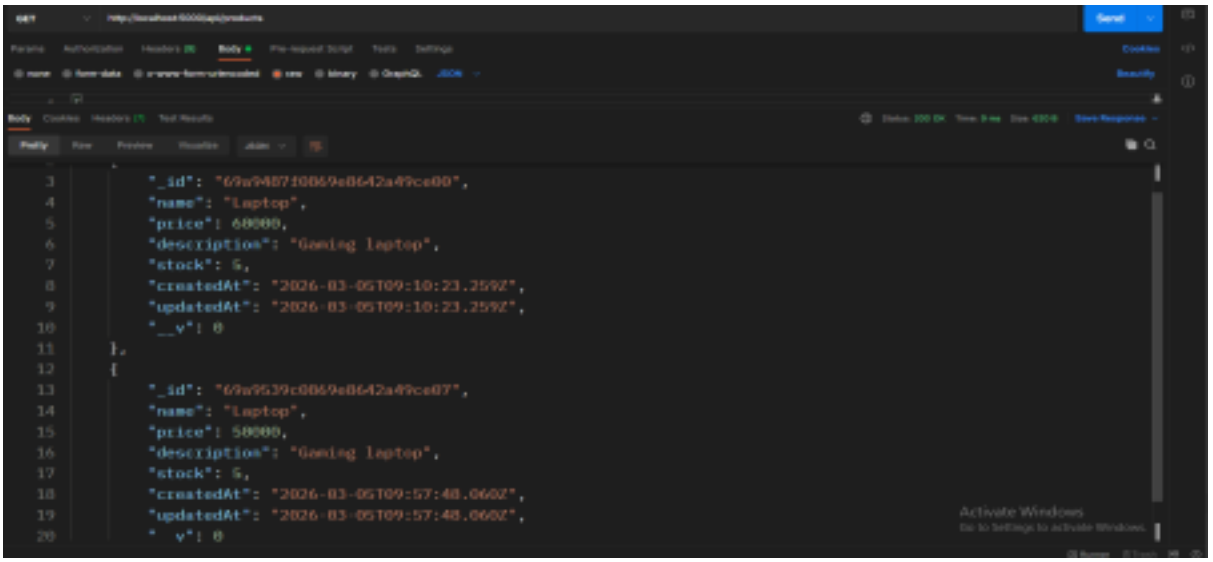
"stock":5

}



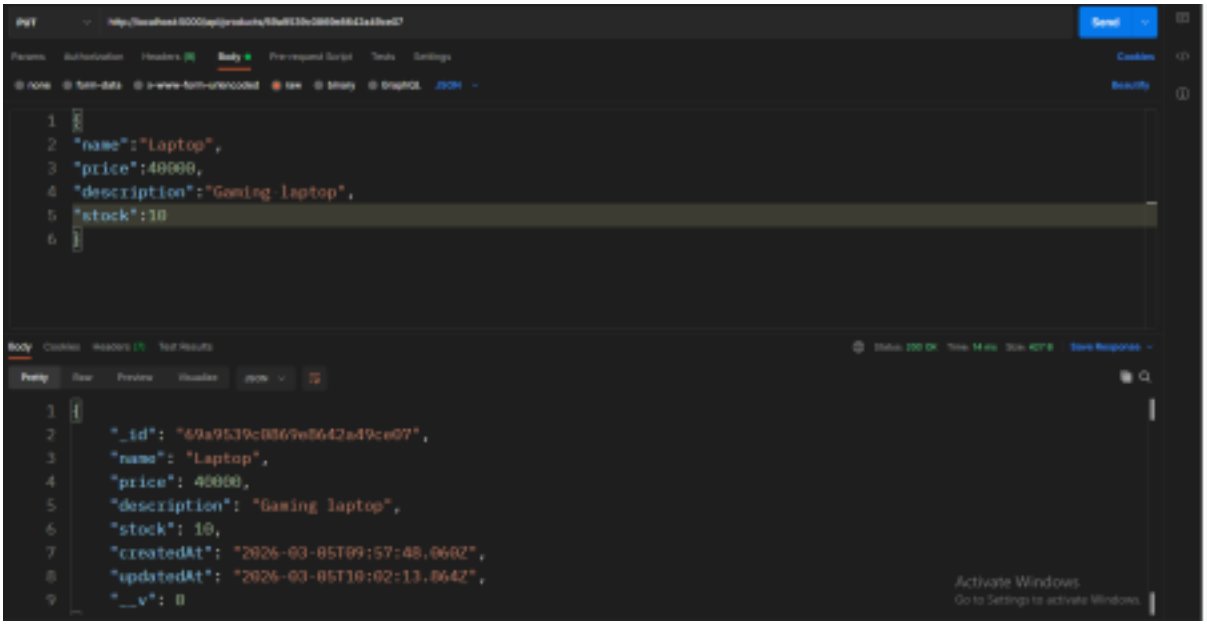
Get Products

Request: GET /api/products



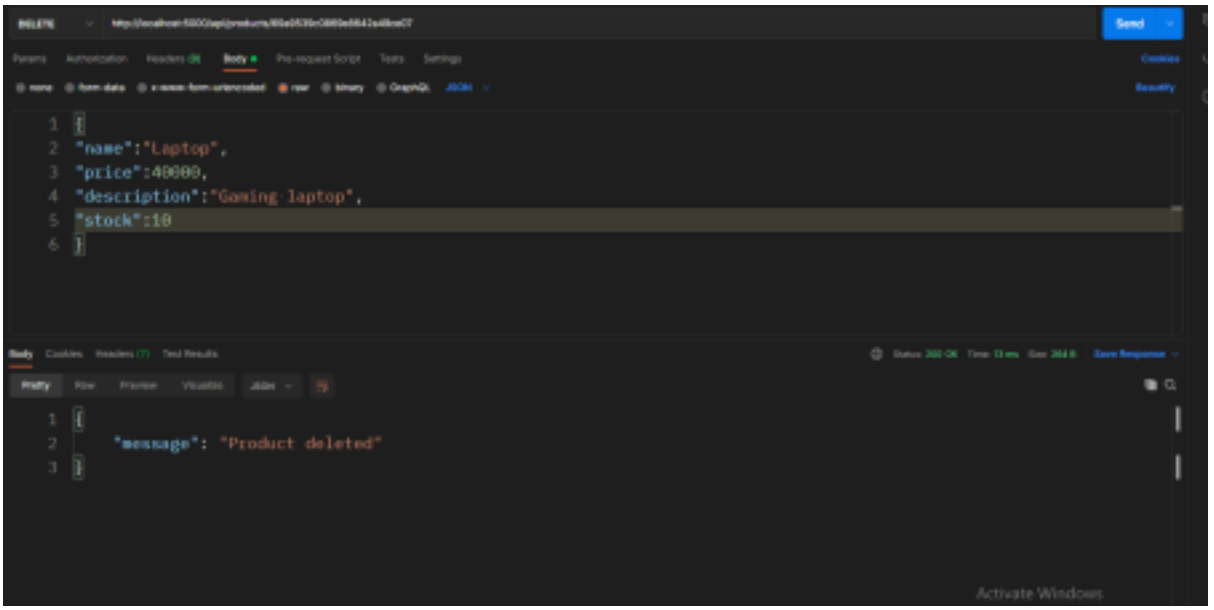
Update Product

Request: PUT /api/products/:id



Delete Product

Request: DELETE /api/products/:id


DELETE
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name":"Laptop",
"price":40000,
"description":"Gaming laptop",
"stock":10
}
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 13 ms
Size: 264 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"message": "Product deleted"
}
Activate Windows