

# MERN Stack Training

**Company:** Sensation Software Solutions

**Student Name:** Tajinder Kaur

**Training Duration:** 6 Months

**Days:** 71

---

## Objective of the Day

The primary objective for Day 71 was to **set up backend CRUD architecture** for the GadgetShop project. This included preparing the foundational structure for product management features and configuring protected API endpoints. The focus was on enabling the server to handle **Create, Read, Update, and Delete (CRUD)** operations through REST APIs, while ensuring authorization through middleware.

---

## Work Detailed Summary

On this day, I specifically worked on **Node.js Express backend** to establish a minimal yet scalable CRUD system for the products module:

### 1 Backend Route Setup

- Created a new file named **products.js** inside the `routes` folder.
- Defined REST API routes to handle different product operations.
- Imported necessary controllers and middleware for request parsing.

### 2 Defined Product CRUD Endpoints

Operation	Endpoint	Description
Read (All)	<b>GET /products</b>	Fetch list of all products from DB
Create	<b>POST /products</b>	Add a new product to the system
Update	<b>PUT /products/:id</b>	Modify existing product details
Delete	<b>DELETE /products/:id</b>	Remove product by ID

Each route was structured with status checks and JSON response formatting to ensure frontend-side readability and integration readiness.

### 3 Secured Critical Operations

- Implemented **authentication middleware** to ensure only authorized users can:
  - Create products
  - Edit products
  - Delete products

- Kept GET requests public for product visibility.

## API Testing

- Used Postman to test each endpoint individually.
- Sent sample data payloads to validate server request handling.
- Debugged minor validation conflicts and ensured status codes were accurate (200/201/400/404).

---

## Hands-On Practice & Learning

- Understood the difference between **middleware-level validation** and **route-level validation**.
- Practiced sending **JSON request bodies** and identifying malformed requests.
- Compared behavior between **synchronous** vs **asynchronous** database interaction.

---

## Challenges & Solutions

Challenge	Solution Applied
Ensuring auth only triggers on protected routes	Implemented middleware selectively for POST/PUT/DELETE
Handling invalid ID requests	Added fallback responses with descriptive error messages
Structuring scalable route file	Grouped code by endpoint category and added comments

---

## Conclusion

Day 71 successfully resulted in a **fully prepared backend CRUD structure**, secured by authentication middleware. This served as a baseline for the next phases of frontend integration. The foundation laid today will support product data manipulation throughout the app lifecycle.