# MERN Stack Training

**Company:** Sensation Software Solutions
**Student Name:** Tajinder Kaur
**Training Duration:** 6 Months
**Days:** 85

---

### Objective of the Day

The main objective of **Day 85** was to implement **Product Pagination and Search Enhancements** to improve site performance, usability, and navigation. As the number of products increased, it became essential to efficiently manage large datasets and allow users to quickly find desired products.

The goal was to ensure that pagination, search, and filters worked together smoothly, providing a seamless browsing experience similar to modern e-commerce platforms.

---

### Requirement Analysis

At the start of the day, I reviewed the Shop page and identified the limitations:

- All products were loading at once.
- Search functionality was basic.
- Navigation became difficult with a large product list.

To resolve these issues, I planned to implement server-side pagination combined with improved frontend search behavior.

---

### Backend Development – Pagination Logic

### Pagination API Design

Pagination was implemented in the backend using query parameters:

- `page` – Current page number
- `limit` – Number of products per page

Example API:

```
GET /products?page=2&limit=10
```

The backend calculated:

- Number of products to skip
- Total number of pages
- Current page information

This reduced server load and improved response time.

---

### Search Integration

Search functionality was enhanced to work alongside pagination by passing a `search` query parameter. The backend filtered products before applying pagination logic, ensuring accurate results.

---

### Frontend Development – Pagination UI

### Pagination Component

A reusable **Pagination component** was created with:

- Next and Previous buttons
- Page number buttons
- Active page highlighting

The component dynamically updated based on API response data.

---

### Shop Page Enhancements

The Shop page was updated to:

- Fetch products page-wise
- Update URL parameters dynamically
- Maintain search state during navigation

This ensured better SEO and browser navigation support.

---

### Live Search Improvements

Search input was enhanced to provide **instant results while typing**:

- Debouncing technique was used to reduce API calls.
- Search queries updated URL parameters.
- Results refreshed dynamically without full page reload.

---

### Styling and UI Polish

Pagination buttons and search UI were styled using **Tailwind CSS**:

- Clear button spacing
- Hover and active states
- Mobile-friendly layout

Special care was taken to maintain design consistency across the application.

---

### Hands-on Practice and Testing

Extensive testing was performed:

- Tested pagination with large datasets.
- Verified search and pagination integration.
- Tested responsiveness on different devices.

Edge cases such as empty results and last page navigation were also handled properly.

---

### Performance Optimization

By implementing pagination:

- Initial page load time was reduced.
- Server load was optimized.
- User experience improved significantly.

---

### Challenges Faced

- Synchronizing search and pagination state.
- Handling URL parameter updates.
- Ensuring smooth UI transitions.

All challenges were resolved using structured state management and clean component design.

---

### Conclusion

Day 85 marked the completion of **advanced product navigation features** in the GadgetShop project. The integration of pagination and enhanced search significantly improved usability and performance.