

## EXPERIMENT NO: 07

**Aim:** Connect a Node.js application for Book Store to a MongoDB database. Implement CRUD operations to interact with the database using the MongoDB Node.js driver.

### Introduction to MongoDB

MongoDB is a NoSQL database that stores data in a document-oriented format (JSON-like BSON format). Unlike relational databases, it does not require a predefined schema, making it flexible for handling dynamic data

### Connecting Backend with Frontend

To connect the backend (Node.js + Express + MongoDB) with the frontend (React or any other framework), follow these steps:

#### Set Up the Backend (Node.js + Express + MongoDB)

- Install dependencies:

#### Set Up the Frontend

- Install Axios for API calls:

### Understanding Key Tools

#### Nodemon

- Nodemon is a tool that automatically restarts the server when changes are detected in the backend code.

#### Body-Parser

- body-parser is a middleware that parses JSON and form data from incoming requests.
- Express now includes JSON parsing, so instead of installing body-parser, just use:

#### MongoDB Compass

- MongoDB Compass is a GUI tool for MongoDB that allows you to:
  - View, insert, update, and delete data.

- Run queries without writing long MongoDB shell commands.
- Connect to both local and cloud MongoDB databases.

#### bookControllers.js

<pre>import Book from "../models/bookModel.js";  export const addBook = async (req, res) =&gt; {   try {     const bookData = new Book(req.body);     const { title } = bookData;      const bookExist = await Book.findOne({ title });     if (bookExist) {       return res.status(400).json({ message: "Book already exists." });     }      const savedBook = await bookData.save();     res.status(201).json(savedBook);    } catch (error) {     res.status(500).json({ error: "Internal Server Error." });   } };  export const getBooks = async (req, res)</pre>	<pre>     }     res.status(200).json(books);   } catch (error) {     res.status(500).json({ error: "Internal Server Error." });   } };  export const updateBook = async (req, res) =&gt; {   try {     const id = req.params.id;     const bookExist = await Book.findById(id);      if (!bookExist) {       return res.status(404).json({ message: "Book not found!" });     }      const updatedBook = await Book.findByIdAndUpdate(id, req.body, { new: true });     res.status(200).json(updatedBook);    } catch (error) {     res.status(500).json({ error: "Internal</pre>
--	---

<pre>Server Error." });     } };  export const deleteBook = async (req, res) =&gt; {     try {         const books = await Book.find();         if (books.length === 0) {             return res.status(404).json({ message: "No books found!" });         }         try {             const id = req.params.id;             const bookExist = await Book.findById(id);              if (!bookExist) {                 return res.status(404).json({ message: "Book not found!" });             }              await Book.findByIdAndDelete(id);             res.status(200).json({ message: "Book deleted successfully!" });          } catch (error) {             res.status(500).json({ error: "Internal</pre>	<pre>Server Error." });     } };</pre>
--	--

### bookModel.js

<pre>import mongoose from "mongoose";  const bookSchema = new mongoose.Schema({   title: {     type: String,     required: true   },   author: {     type: Number,     type: String,     required: true     required: true   } }, { timestamps: true });  export default mongoose.model("Book", bookSchema);</pre>	<pre>}, description: {   type: String,   required: true }, price: {   type: Number,   required: true }, publishedYear: {</pre>
--	--

### bookRoutes.js

<pre>import express from 'express'; import { getBooks, addBook, updateBook, deleteBook } from '../controllers/bookController.js';  const router = express.Router();  router.put("/:id", updateBook);</pre>	<pre>// Create a new book router.post("/", addBook);  // Get all books router.get("/", getBooks);  // Update a book by ID</pre>
--	---

<pre>// Delete a book by ID router.delete("/:id", deleteBook);  export default router;</pre>	
--	--

.env

PORT=5000

MONGO\_URI="mongodb://localhost:27017/bookstore"

index.js

<pre>import express from "express"; import mongoose from "mongoose"; import dotenv from "dotenv"; import cors from "cors"; // Added CORS for frontend communication import bookRoutes from "./routes/bookRoutes.js";  dotenv.config();  const app = express(); const PORT = process.env.PORT    5000; const MONGO_URI = process.env.MONGO_URI;  // Debug Logs  .catch((error) =&gt; {</pre>	<pre>.catch((error) =&gt; {   console.error(" Database Connection Error:", error.message);   process.exit(1); // Exit process on failure }); console.log("Starting Server..."); console.log("PORT:", PORT); console.log("MONGO_URI:", MONGO_URI);  // Middleware app.use(cors()); // Enable Cross-Origin Requests app.use(express.json()); // Built-in JSON body parser</pre>
---	---

```
    console.error(" Database Connection
Error:", error.message);
    process.exit(1); // Exit process on failure
  });
// Default Route
app.get("/", (req, res) => {
  res.send(" Welcome to the Book Store
API!");
});

/ Book Routes
app.use("/api/books", bookRoutes);

// Database Connection
mongoose
  .connect(MONGO_URI, {
    useNewUrlParser: true, useUnifiedTopology:
    true })
  .then(() => {
    console.log(" Database Connected
Successfully!");
    app.listen(PORT, () => {
      console.log(` Server is running at
http://localhost:${PORT}`);
    });
  })
```

Compass

My Queries

CONNECTIONS (2)

Search connections

- localhost:27017
  - admin
  - bookstore
    - books
  - config
  - local
- localhost:27017
  - admin
  - config
  - local

localhost:27017 > bookstore > books

Documents 10 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

Explain Reset Find Options

ADD DATA EXPORT DATA UPDATE DELETE

25 1 - 10 of 10

```
{
  "_id": ObjectId('67ea2f4f39d69b4193f29d80'),
  "title": "The Catcher in the Rye",
  "author": "J.D. Salinger",
  "description": "A novel about teenage rebellion.",
  "price": 350,
  "publishedYear": 1951
}
```

```
{
  "_id": ObjectId('67ea2f4f39d69b4193f29d81'),
  "title": "Pride and Prejudice",
  "author": "Jane Austen",
  "description": "A classic romance novel.",
  "price": 299,
  "publishedYear": 1813
}
```

```
{
  "_id": ObjectId('67ea2f4f39d69b4193f29d82'),
  "title": "To Kill a Mockingbird",
  "author": "Harper Lee",
  "description": "A novel about racism and injustice.",
  "price": 399,
  "publishedYear": 1960
}
```

POST http://localhost:5000/api/books

Params Authorization Headers (8) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON**

```
1 {
2   "title": "Harry Potter and the Sorcerer's Stone",
3   "author": "J.K. Rowling",
4   "description": "A young wizard discovers his magical heritage.",
5   "price": 499,
6   "publishedYear": 1997
7 }
```

Body Cookies Headers (8) Test Results ↺

{ } JSON ▾ ▶ Preview 📄 Visualize ▾

```
1  {
2    "title": "Harry Potter and the Sorcerer's Stone",
3    "author": "J.K. Rowling",
4    "description": "A young wizard discovers his magical heritage.",
5    "price": 499,
6    "publishedYear": 1997,
7    "_id": "67ea2fb2bdc730c8559d4fe4",
8    "createdAt": "2025-03-31T06:01:22.462Z",
9    "updatedAt": "2025-03-31T06:01:22.462Z",
10   "__v": 0
11 }
```

 http://localhost:5000/api/books

 Save ▾ Share

GET ▾ http://localhost:5000/api/books

Send ▾

Params Authorization Headers (6) Body Scripts Settings

Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▾

Beautify

1



Body Cookies Headers (8) Test Results

200 OK • 9 ms • 1.96 KB •

{ } JSON ▾ ▶ Preview Visualize ▾



```
1  [
2    {
3      "_id": "67ea2f4f39d69b4193f29d80",
4      "title": "The Catcher in the Rye",
5      "author": "J.D. Salinger",
6      "description": "A novel about teenage rebellion.",
7      "price": 350,
8      "publishedYear": 1951
9    },
10   {
11     "_id": "67ea2f4f39d69b4193f29d81",
12     "title": "Pride and Prejudice",
13     "author": "Jane Austen",
14     "description": "A classic romance novel.",
15     "price": 299,
16     "publishedYear": 1813
17   },
18   {
```

```
    "_id": "67ea2f4f39d69b4193f29d82",
    "title": "To Kill a Mockingbird",
    "author": "Harper Lee",
    "description": "A novel about racism and injustice.",
    "price": 399,
    "publishedYear": 1960
  },
  {
    "_id": "67ea2f4f39d69b4193f29d83",
    "title": "1984",
    "author": "George Orwell",
    "description": "A dystopian novel on totalitarianism.",
    "price": 250,
    "publishedYear": 1949
  }
}
```

http://localhost:5000/api/books/67ea2f4f39d69b4193f29d86

Save ▾ Share

PUT ▾ http://localhost:5000/api/books/67ea2f4f39d69b4193f29d86

Send ▾

Params Authorization Headers (8) Body ● Scripts Settings

Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▾


Beautify

```
1  {
2    "title": "War and Peace (Updated)",
3    "author": "Leo Tolstoy",
4    "description": "An updated description of the historical epic.",
5    "price": 550,
6    "publishedYear": 1870
7  }
```

```
{
  "_id": "67ea2f4f39d69b4193f29d86",
  "title": "War and Peace (Updated)",
  "author": "Leo Tolstoy",
  "description": "An updated description of the historical epic.",
  "price": 550,
  "publishedYear": 1870,
  "updatedAt": "2025-03-31T06:07:07.207Z"
}
```

 http://localhost:5000/api/books/67ea2f4f39d69b4193f29d86


 Save  Share

DELETE  http://localhost:5000/api/books/67ea2f4f39d69b4193f29d86

Send 

Params Authorization Headers (6) Body Scripts Settings

Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON 

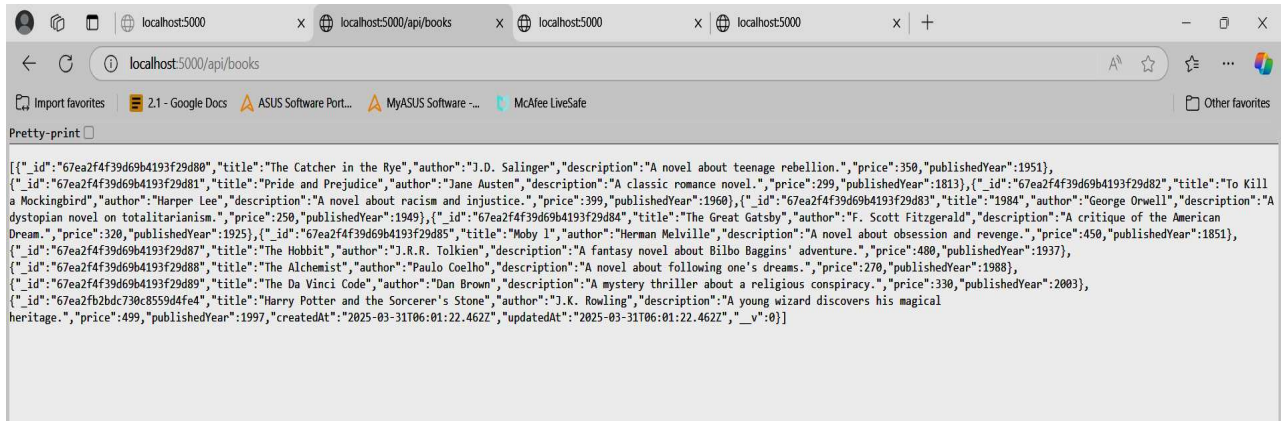
Beautify

1

Body Cookies Headers (8) Test Results 

 JSON   Preview  Visualize 

```
1 {
2   ... "message": "Book deleted successfully!"
3 }
```



```
localhost:5000/api/books

Pretty-print ☐

[{"_id":"67ea2f4f39d69b4193f29d80","title":"The Catcher in the Rye","author":"J.D. Salinger","description":"A novel about teenage rebellion.","price":350,"publishedYear":1951}, {"_id":"67ea2f4f39d69b4193f29d81","title":"Pride and Prejudice","author":"Jane Austen","description":"A classic romance novel.","price":299,"publishedYear":1813}, {"_id":"67ea2f4f39d69b4193f29d82","title":"To Kill a Mockingbird","author":"Harper Lee","description":"A novel about racism and injustice.","price":399,"publishedYear":1960}, {"_id":"67ea2f4f39d69b4193f29d83","title":"1984","author":"George Orwell","description":"A dystopian novel on totalitarianism.","price":250,"publishedYear":1949}, {"_id":"67ea2f4f39d69b4193f29d84","title":"The Great Gatsby","author":"F. Scott Fitzgerald","description":"A critique of the American Dream.","price":320,"publishedYear":1925}, {"_id":"67ea2f4f39d69b4193f29d85","title":"Moby 1","author":"Herman Melville","description":"A novel about obsession and revenge.","price":450,"publishedYear":1851}, {"_id":"67ea2f4f39d69b4193f29d87","title":"The Hobbit","author":"J.R.R. Tolkien","description":"A fantasy novel about Bilbo Baggins' adventure.","price":480,"publishedYear":1937}, {"_id":"67ea2f4f39d69b4193f29d88","title":"The Alchemist","author":"Paulo Coelho","description":"A novel about following one's dreams.","price":270,"publishedYear":1988}, {"_id":"67ea2f4f39d69b4193f29d89","title":"The Da Vinci Code","author":"Dan Brown","description":"A mystery thriller about a religious conspiracy.","price":330,"publishedYear":2003}, {"_id":"67ea2fb2bdc730c8559d4fe4","title":"Harry Potter and the Sorcerer's Stone","author":"J.K. Rowling","description":"A young wizard discovers his magical heritage.","price":499,"publishedYear":1997,"createdAt":"2025-03-31T06:01:22.462Z","updatedAt":"2025-03-31T06:01:22.462Z","_v":0}]
```

Conclusion:

MongoDB is a powerful NoSQL database that seamlessly integrates with Node.js and Express to build scalable backend applications. Tools like Nodemon, body-parser, and MongoDB Compass simplify development, debugging, and database management. By connecting the backend with the frontend, we enable smooth data flow for dynamic web applications.