Simple Book Store API Case Study

Project Structure

```
bookstore/
│── package.json
│── server.js
│── config/
│   └── db.js
│── models/
│   └── Book.js
│── controllers/
│   └── bookController.js
│── routes/
│   └── bookRoutes.js
```

Step 1: Initialize Project

```
mkdir bookstore
cd bookstore
```

```
npm init -y

npm install express mongoose
```

Step 2: Database Connection

File: config/db.js

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



const connectDB = async () => {

  try {

    await mongoose.connect("mongodb://127.0.0.1:27017/bookstore", {

      useNewUrlParser: true,

      useUnifiedTopology: true,

    });

    console.log(" MongoDB Connected");

  } catch (err) {

    console.error("MongoDB connection failed:", err.message);

    process.exit(1);

  }

};
```

```
module.exports = connectDB;
```

Step 3: Book Model

File: models/Book.js

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


const bookSchema = new mongoose.Schema({
  title: { type: String, required: true, unique: true },
  author: { type: String, required: true },
  price: { type: Number, required: true }
}, { timestamps: true });


module.exports = mongoose.model("Book", bookSchema);
```

Step 4: Book Controller (CRUD Functions)

File: controllers/bookController.js

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

```
// Add a new book

exports.addBook = async (req, res) => {

  try {

    const { title, author, price } = req.body;

    const book = new Book({ title, author, price });

    await book.save();

    res.status(201).json({ message: " Book Added", book });

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

};



// List all books

exports.listBooks = async (req, res) => {

  try {

    const books = await Book.find();

    res.json(books);

  } catch (err) {

    res.status(500).json({ error: err.message });
  }
};

// Find book by title
exports.findBook = async (req, res) => {
  try {
    const { title } = req.params;
    const book = await Book.findOne({ title });
    if (!book) return res.status(404).json({ message: "Book not found" });
    res.json(book);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

// Update book price
exports.updatePrice = async (req, res) => {
  try {
    const { title } = req.params;
    const { price } = req.body;

    const book = await Book.findOneAndUpdate(

      { title },

      { price },

      { new: true }

    );

    if (!book) return res.status(404).json({ message: "Book not found" });

    res.json({ message: "💲 Price Updated", book });

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

};
```

Step 5: Book Routes

File: routes/bookRoutes.js

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

const router = express.Router();

const bookController = require("../controllers/bookController");
```

```
// Routes

router.post("/books", bookController.addBook);       // Add book

router.get("/books", bookController.listBooks);      // List all

router.get("/books/:title", bookController.findBook); // Find by title

router.put("/books/:title", bookController.updatePrice); // Update price


module.exports = router;
```

Step 6: Server Setup

File: server.js

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

const connectDB = require("./config/db");

const bookRoutes = require("./routes/bookRoutes");


const app = express();

const PORT = 5000;


// Middleware

app.use(express.json());
```

```
// Routes

app.use("/api", bookRoutes);



// Connect DB & Start server

connectDB();



app.listen(PORT, () => {

  console.log(` Server running on http://localhost:${PORT}`);

});
```

 Step 7:Run the Project

```
node server.js
```

Server will start at: http://localhost:5000

API Endpoints (Test in Postman)

1. **Add Book (POST)**
2. POST http://localhost:5000/api/books
3. {
4.   "title": "Bhagavad Gita",

5. "author": "AC. Bhaktivedanta Swami Srila Prabhupad",
6. "price": 300
7. }
8. **List All Books (GET)**
9. GET http://localhost:5000/api/books
10. **Find Book by Title (GET)**
11. GET http://localhost:5000/api/books/Bhagavad Gita
12. **Update Book Price (PUT)**
13. PUT http://localhost:5000/api/books/Bhagavad Gita
14. {
15. "price": 350
16. }
