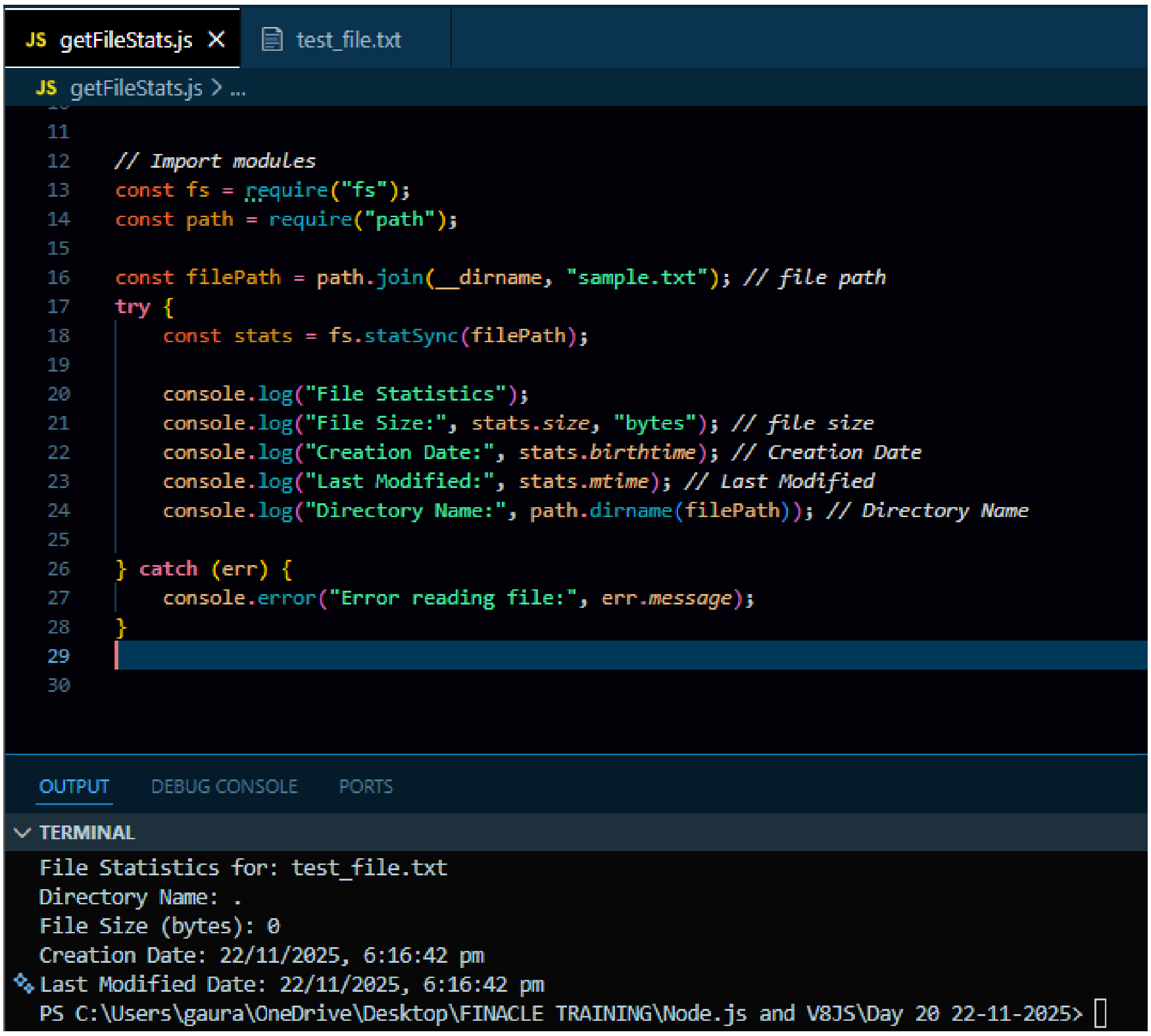
Class Practice Problems

Name: Gaurav Deshmukh

USN: 72232433C

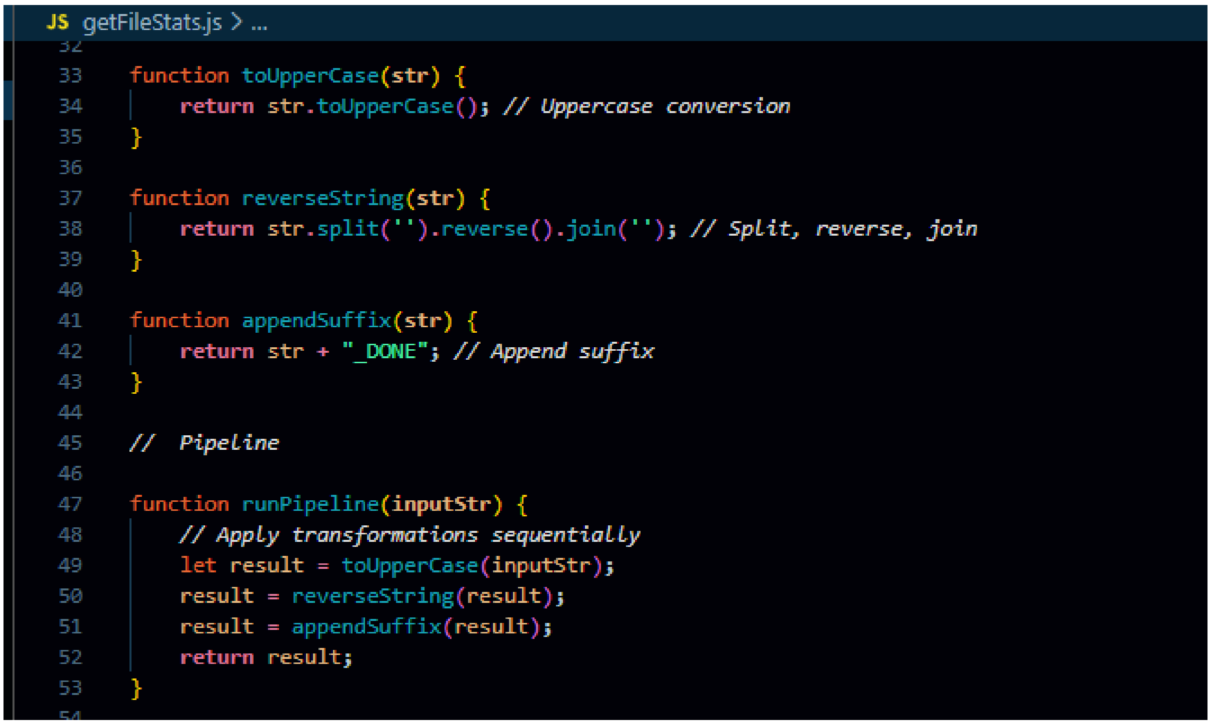
Q. 1. Display File Statistics Using fs and path Modules – 5 Marks Objective: Use Node.js fs and path modules to read a file and display its statistics.



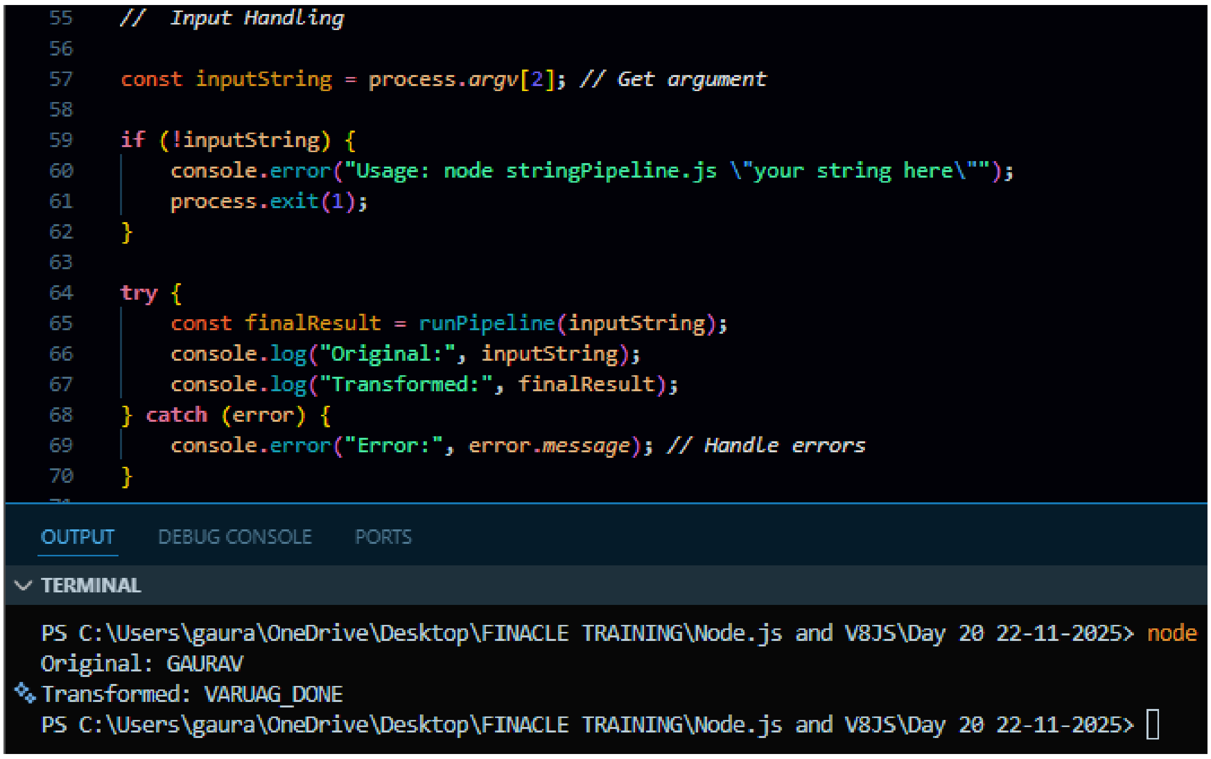
```javascript

// Import modules
const fs = require("fs");
const path = require("path");

const filePath = path.join(__dirname, "sample.txt"); // file path
try {
    const stats = fs.statSync(filePath);

    console.log("File Statistics");
    console.log("File Size:", stats.size, "bytes"); // file size
    console.log("Creation Date:", stats.birthtime); // Creation Date
    console.log("Last Modified:", stats.mtime); // Last Modified
    console.log("Directory Name:", path.dirname(filePath)); // Directory Name

} catch (err) {
    console.error("Error reading file:", err.message);
}

```

```
File Statistics for: test_file.txt
Directory Name: .
File Size (bytes): 0
Creation Date: 22/11/2025, 6:16:42 pm
Last Modified Date: 22/11/2025, 6:16:42 pm
PS C:\Users\gaura\OneDrive\Desktop\FINACLE TRAINING\Node.js and V8JS\Day 20 22-11-2025>
```

Q. 2. Function to run the transformation pipeline : Uppercase -> Reverse -> Append Suffix



```javascript
function toUpperCase(str) {
    return str.toUpperCase(); // Uppercase conversion
}

function reverseString(str) {
    return str.split('').reverse().join(''); // Split, reverse, join
}

function appendSuffix(str) {
    return str + "_DONE"; // Append suffix
}

//  Pipeline

function runPipeline(inputStr) {
    // Apply transformations sequentially
    let result = toUpperCase(inputStr);
    result = reverseString(result);
    result = appendSuffix(result);
    return result;
}
```

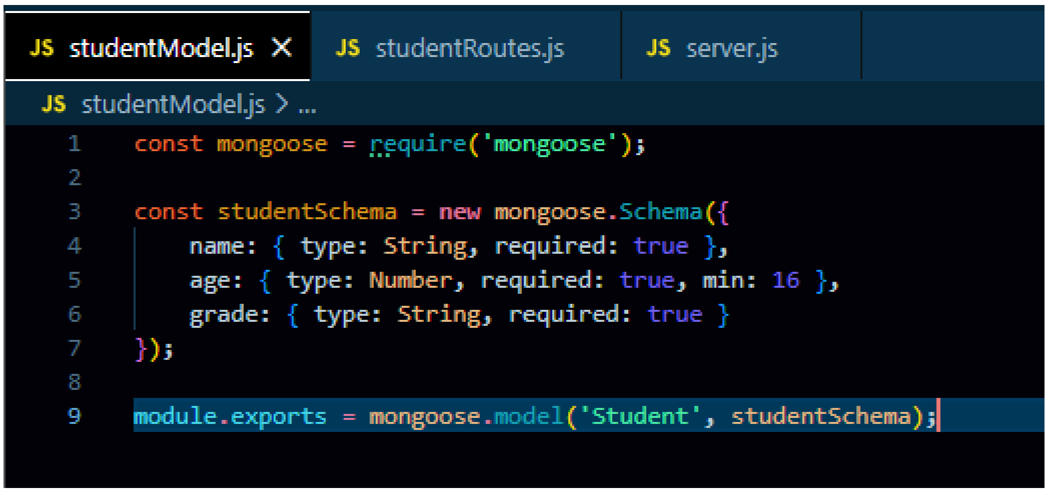


```javascript
//  Input Handling

const inputString = process.argv[2]; // Get argument

if (!inputString) {
    console.error("Usage: node stringPipeline.js \"your string here\"");
    process.exit(1);
}

try {
    const finalResult = runPipeline(inputString);
    console.log("Original:", inputString);
    console.log("Transformed:", finalResult);
} catch (error) {
    console.error("Error:", error.message); // Handle errors
}
```

```
PS C:\Users\gaura\OneDrive\Desktop\FINACLE TRAINING\Node.js and V8JS\Day 20 22-11-2025> node
Original: GAURAV
Transformed: VARUAG_DONE
PS C:\Users\gaura\OneDrive\Desktop\FINACLE TRAINING\Node.js and V8JS\Day 20 22-11-2025>
```

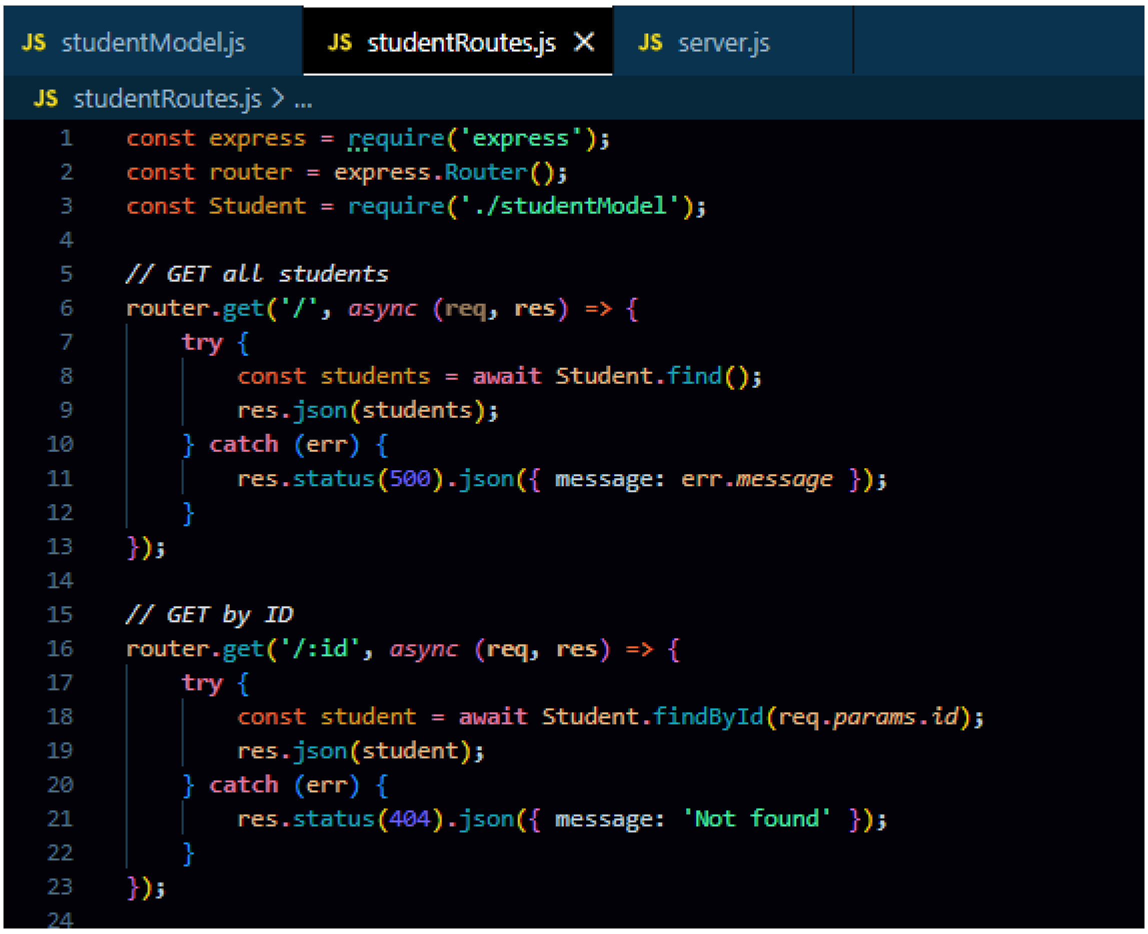
Q. 3. Student Management System (CRUD) .

studentModel.js

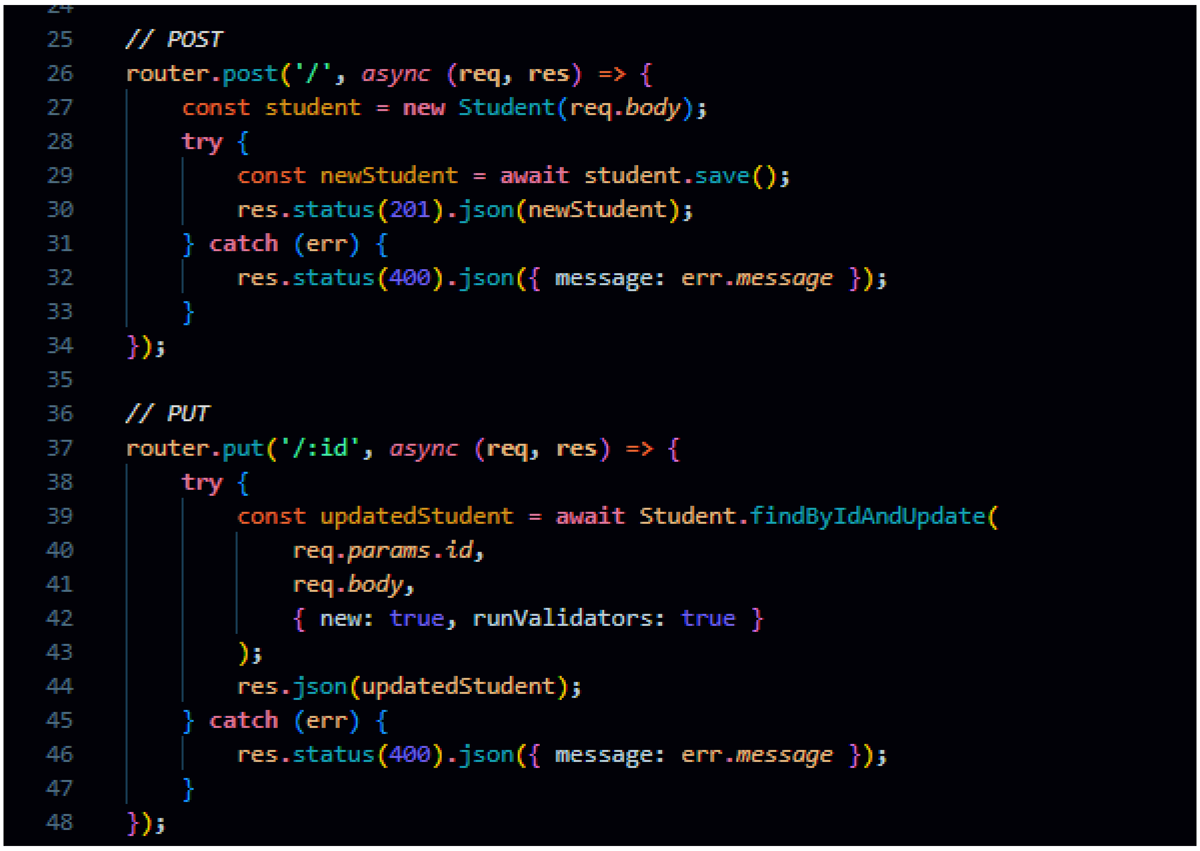


```javascript
const mongoose = require('mongoose');

const studentSchema = new mongoose.Schema({
    name: { type: String, required: true },
    age: { type: Number, required: true, min: 16 },
    grade: { type: String, required: true }
});

module.exports = mongoose.model('Student', studentSchema);
```

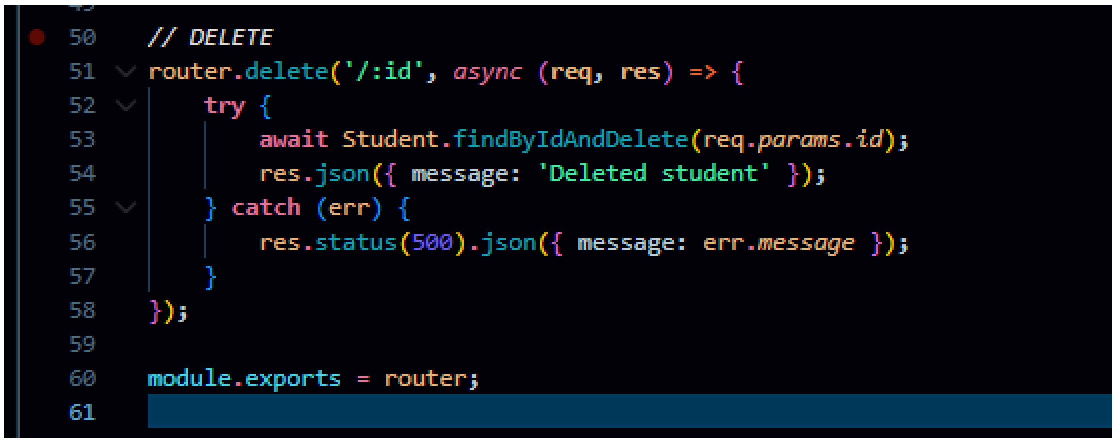
studentRoutes.js



```javascript
const express = require('express');
const router = express.Router();
const Student = require('./studentModel');

// GET all students
router.get('/', async (req, res) => {
    try {
        const students = await Student.find();
        res.json(students);
    } catch (err) {
        res.status(500).json({ message: err.message });
    }
});

// GET by ID
router.get('/:id', async (req, res) => {
    try {
        const student = await Student.findById(req.params.id);
        res.json(student);
    } catch (err) {
        res.status(404).json({ message: 'Not found' });
    }
});

```

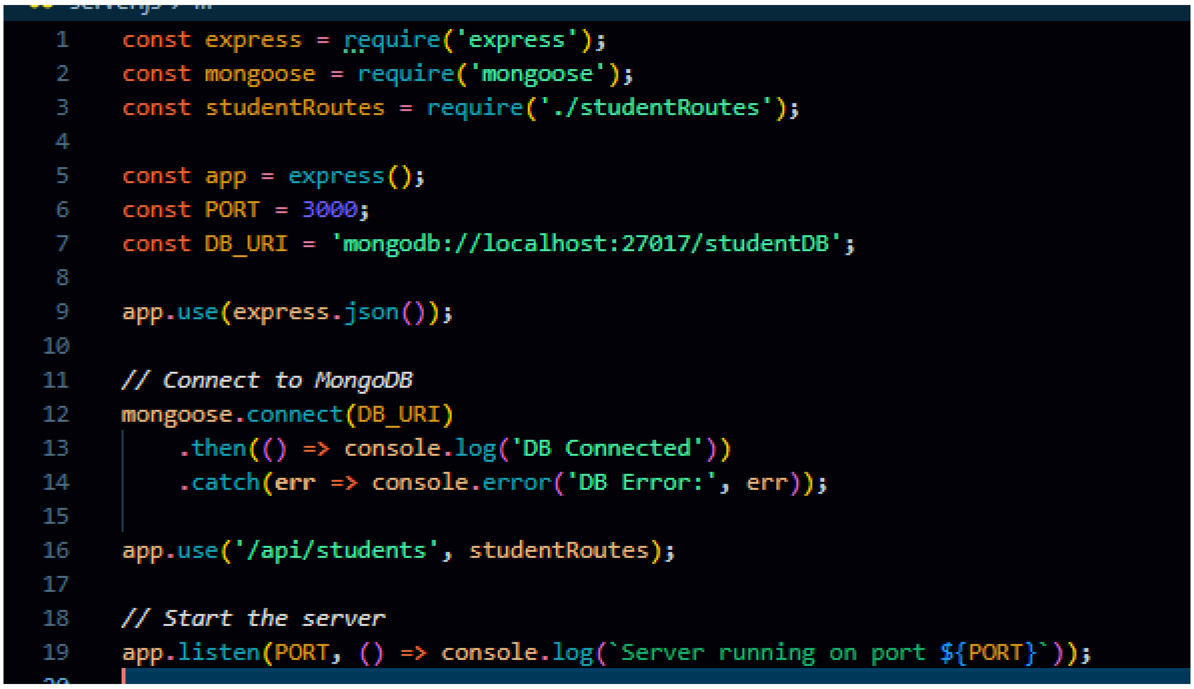


```
// POST
router.post('/', async (req, res) => {
    const student = new Student(req.body);
    try {
        const newStudent = await student.save();
        res.status(201).json(newStudent);
    } catch (err) {
        res.status(400).json({ message: err.message });
    }
});

// PUT
router.put('/:id', async (req, res) => {
    try {
        const updatedStudent = await Student.findByIdAndUpdate(
            req.params.id,
            req.body,
            { new: true, runValidators: true }
        );
        res.json(updatedStudent);
    } catch (err) {
        res.status(400).json({ message: err.message });
    }
});
```



```
// DELETE
router.delete('/:id', async (req, res) => {
    try {
        await Student.findByIdAndDelete(req.params.id);
        res.json({ message: 'Deleted student' });
    } catch (err) {
        res.status(500).json({ message: err.message });
    }
});

module.exports = router;
```

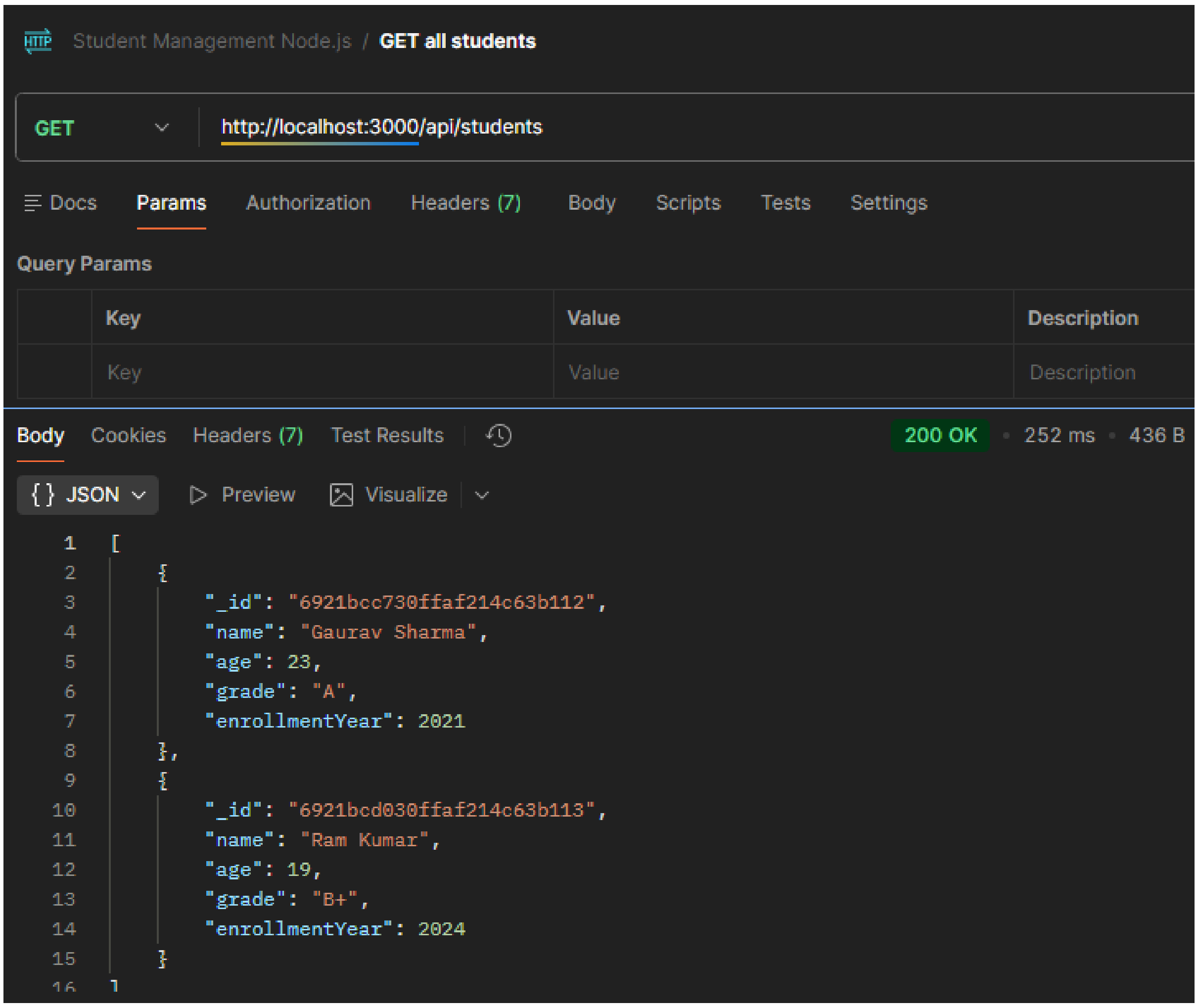
Server.js



```javascript
const express = require('express');
const mongoose = require('mongoose');
const studentRoutes = require('./studentRoutes');

const app = express();
const PORT = 3000;
const DB_URI = 'mongodb://localhost:27017/studentDB';

app.use(express.json());

// Connect to MongoDB
mongoose.connect(DB_URI)
    .then(() => console.log('DB Connected'))
    .catch(err => console.error('DB Error:', err));

app.use('/api/students', studentRoutes);

// Start the server
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

POSTMAN:

(1) GET all students



http://localhost:3000/api/students

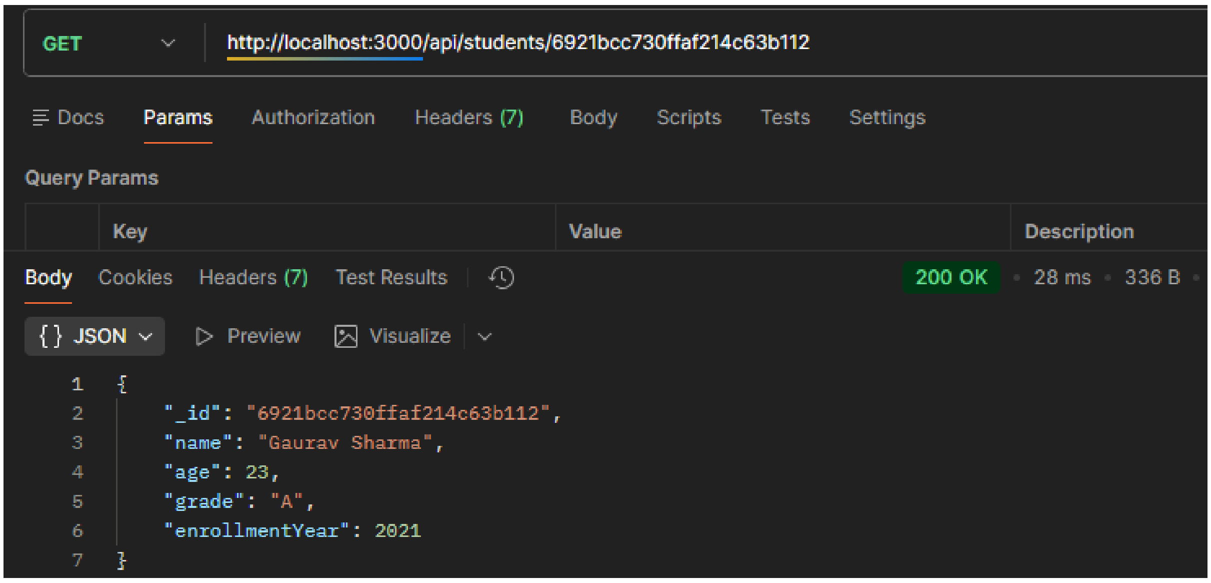
Query Params

Key	Value	Description
Key	Value	Description

200 OK • 252 ms • 436 B

```
[
    {
        "_id": "6921bcc730ffaf214c63b112",
        "name": "Gaurav Sharma",
        "age": 23,
        "grade": "A",
        "enrollmentYear": 2021
    },
    {
        "_id": "6921bcd030ffaf214c63b113",
        "name": "Ram Kumar",
        "age": 19,
        "grade": "B+",
        "enrollmentYear": 2024
    }
]
```

(2) GET student by ID



http://localhost:3000/api/students/6921bcc730ffaf214c63b112

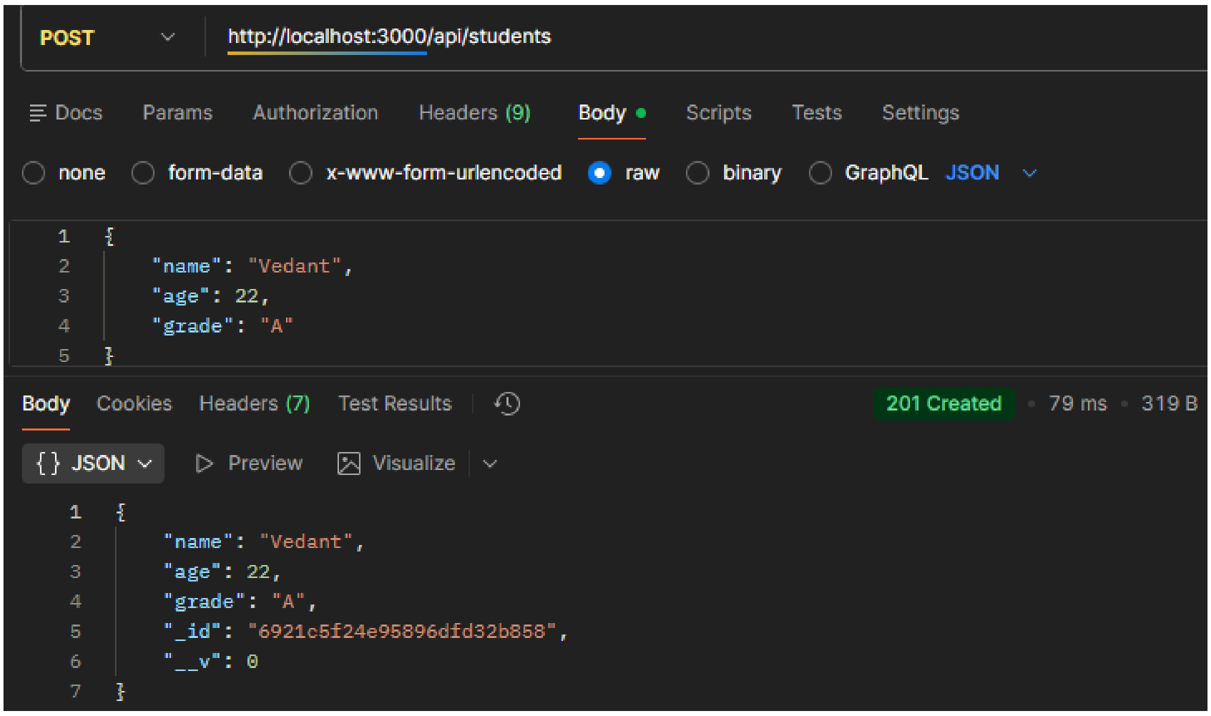
Query Params

Key	Value	Description

200 OK • 28 ms • 336 B

```
{
    "_id": "6921bcc730ffaf214c63b112",
    "name": "Gaurav Sharma",
    "age": 23,
    "grade": "A",
    "enrollmentYear": 2021
}
```

(3) POST new student



http://localhost:3000/api/students

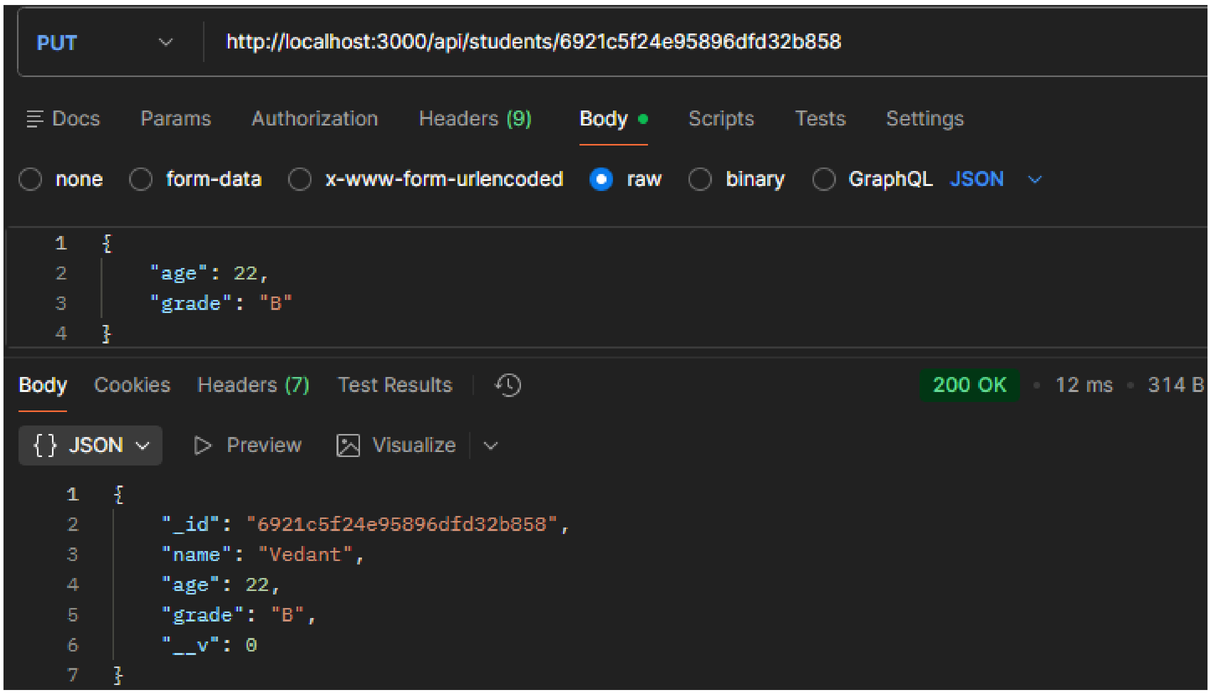
Body: none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "name": "Vedant",
    "age": 22,
    "grade": "A"
}
```

201 Created • 79 ms • 319 B

```
{
    "name": "Vedant",
    "age": 22,
    "grade": "A",
    "_id": "6921c5f24e95896dfd32b858",
    "__v": 0
}
```

(4) PUT – update student



http://localhost:3000/api/students/6921c5f24e95896dfd32b858

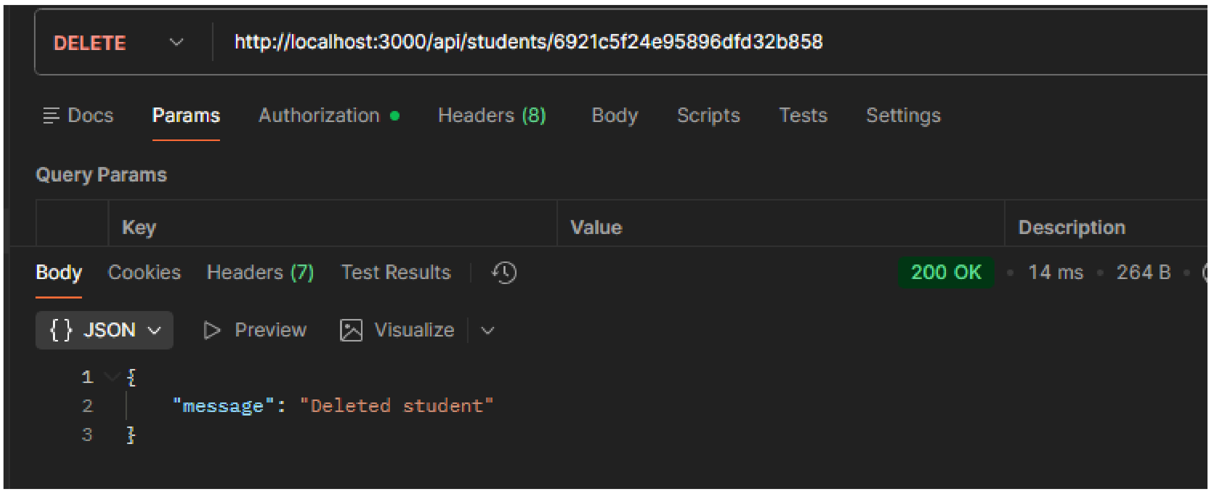
Body: none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "age": 22,
    "grade": "B"
}
```

200 OK • 12 ms • 314 B

```
{
    "_id": "6921c5f24e95896dfd32b858",
    "name": "Vedant",
    "age": 22,
    "grade": "B",
    "__v": 0
}
```

(5) DELETE student by ID

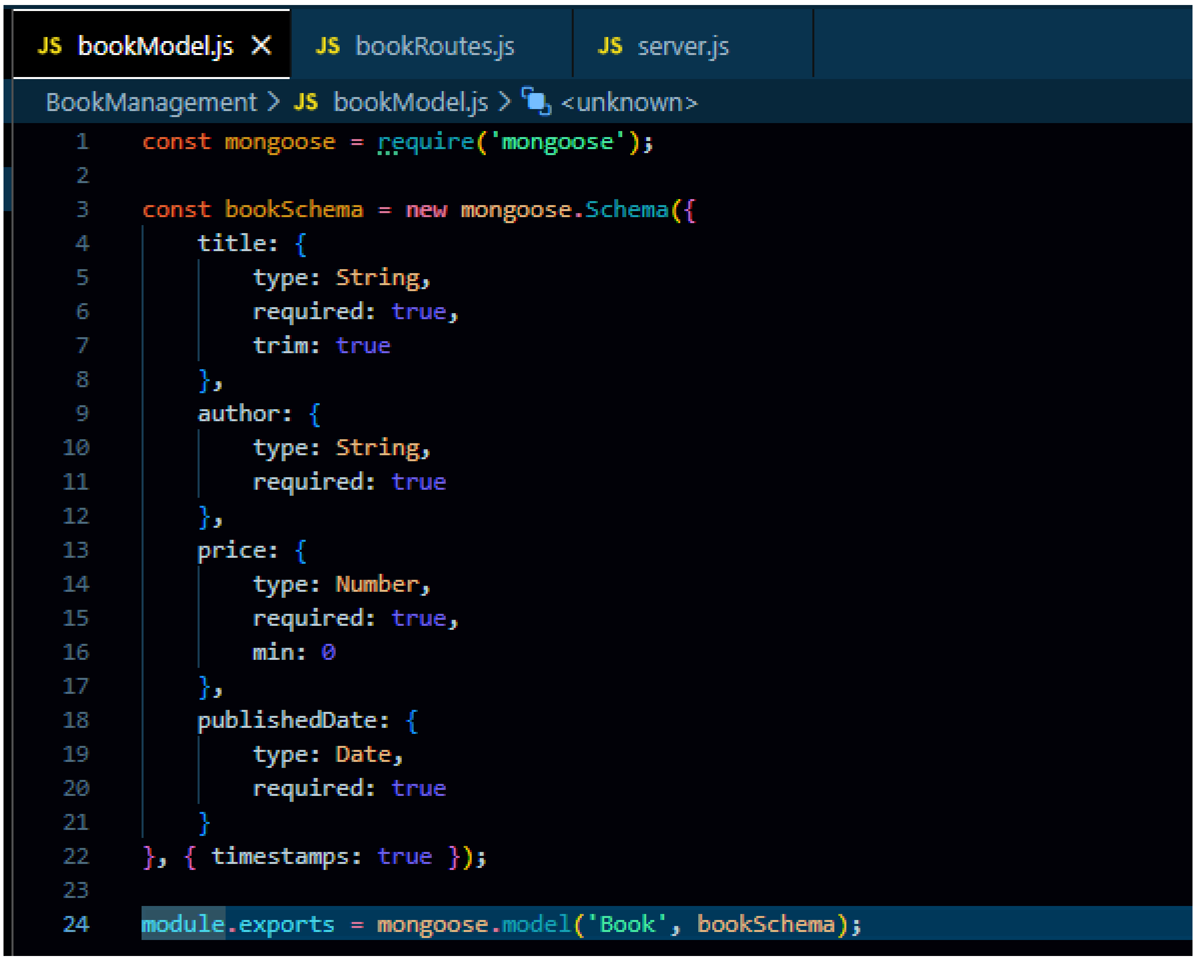


```
http://localhost:3000/api/students/6921c5f24e95896dfd32b858
```

```
{
    "message": "Deleted student"
}
```

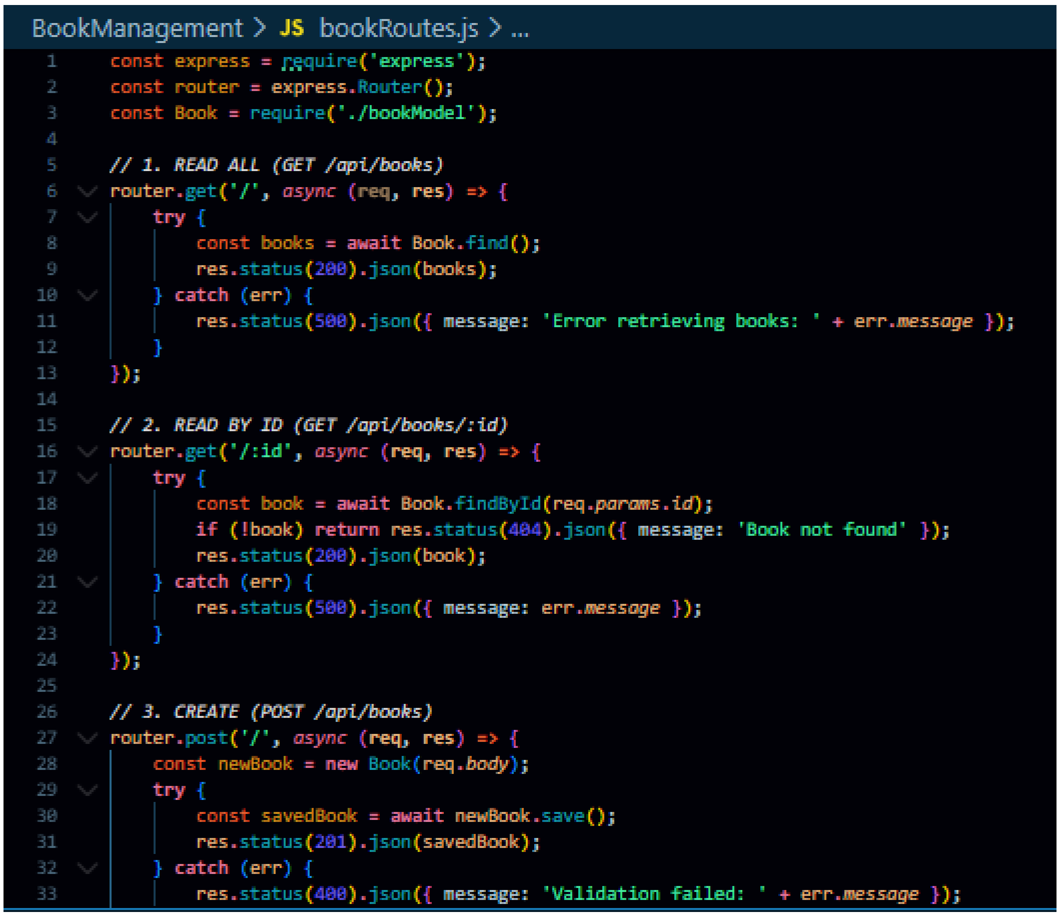
Q . 4. Book Management System (CRUD).

bookModel.js



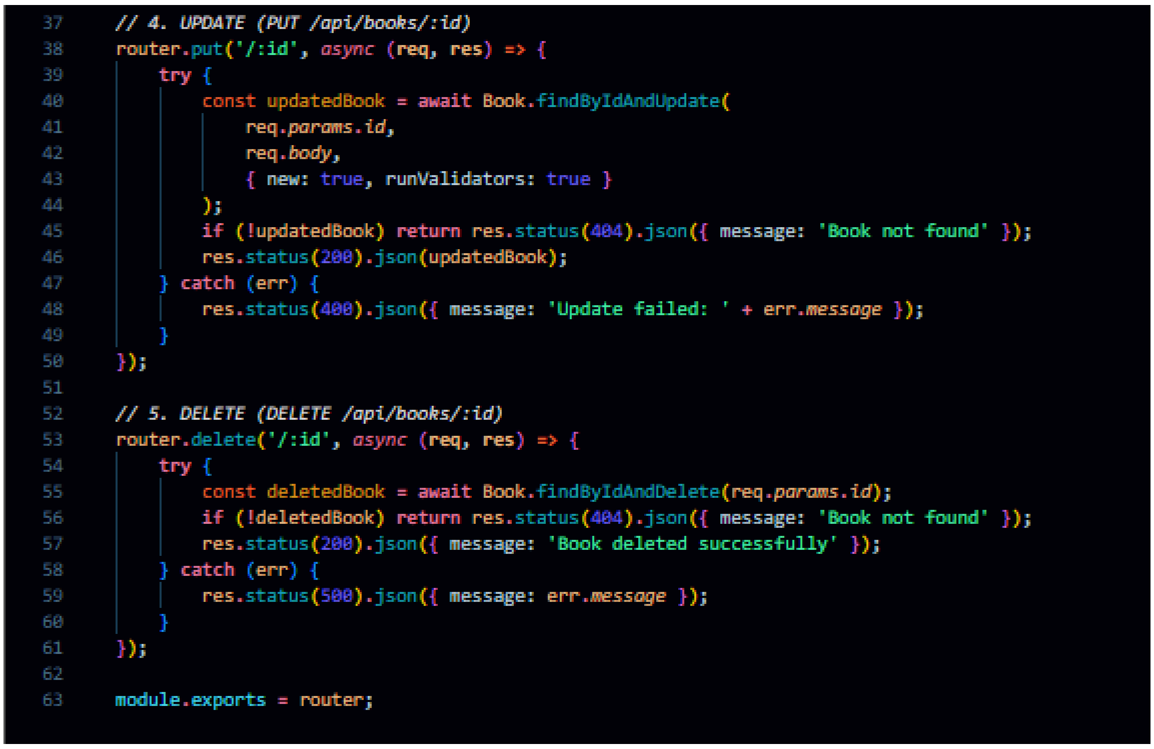
```javascript
const mongoose = require('mongoose');

const bookSchema = new mongoose.Schema({
    title: {
        type: String,
        required: true,
        trim: true
    },
    author: {
        type: String,
        required: true
    },
    price: {
        type: Number,
        required: true,
        min: 0
    },
    publishedDate: {
        type: Date,
        required: true
    }
}, { timestamps: true });

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

bookRoutes.js



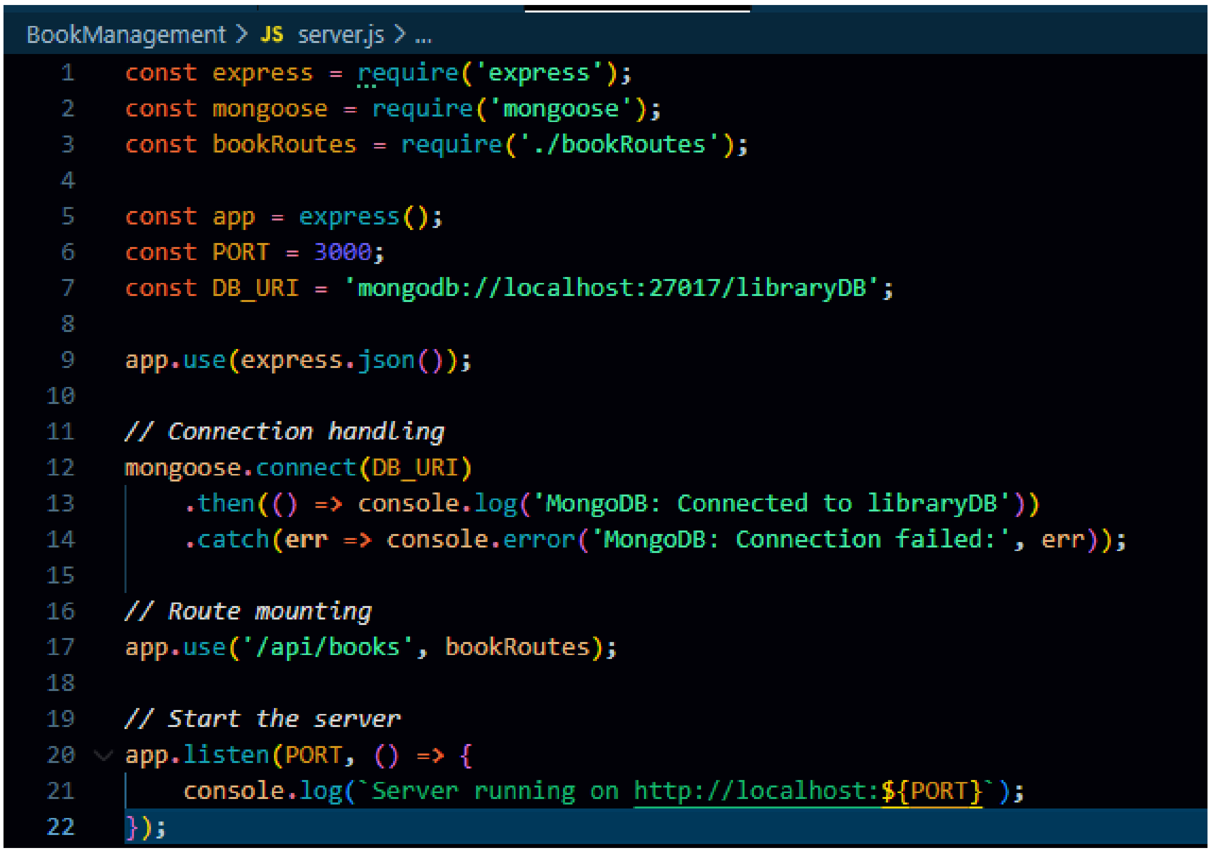
BookManagement > JS bookRoutes.js > ...

```javascript
const express = require('express');
const router = express.Router();
const Book = require('./bookModel');

// 1. READ ALL (GET /api/books)
router.get('/', async (req, res) => {
    try {
        const books = await Book.find();
        res.status(200).json(books);
    } catch (err) {
        res.status(500).json({ message: 'Error retrieving books: ' + err.message });
    }
});

// 2. READ BY ID (GET /api/books/:id)
router.get('/:id', async (req, res) => {
    try {
        const book = await Book.findById(req.params.id);
        if (!book) return res.status(404).json({ message: 'Book not found' });
        res.status(200).json(book);
    } catch (err) {
        res.status(500).json({ message: err.message });
    }
});

// 3. CREATE (POST /api/books)
router.post('/', async (req, res) => {
    const newBook = new Book(req.body);
    try {
        const savedBook = await newBook.save();
        res.status(201).json(savedBook);
    } catch (err) {
        res.status(400).json({ message: 'Validation failed: ' + err.message });
```



```javascript
// 4. UPDATE (PUT /api/books/:id)
router.put('/:id', async (req, res) => {
    try {
        const updatedBook = await Book.findByIdAndUpdate(
            req.params.id,
            req.body,
            { new: true, runValidators: true }
        );
        if (!updatedBook) return res.status(404).json({ message: 'Book not found' });
        res.status(200).json(updatedBook);
    } catch (err) {
        res.status(400).json({ message: 'Update failed: ' + err.message });
    }
});

// 5. DELETE (DELETE /api/books/:id)
router.delete('/:id', async (req, res) => {
    try {
        const deletedBook = await Book.findByIdAndDelete(req.params.id);
        if (!deletedBook) return res.status(404).json({ message: 'Book not found' });
        res.status(200).json({ message: 'Book deleted successfully' });
    } catch (err) {
        res.status(500).json({ message: err.message });
    }
});

module.exports = router;
```

Server.js

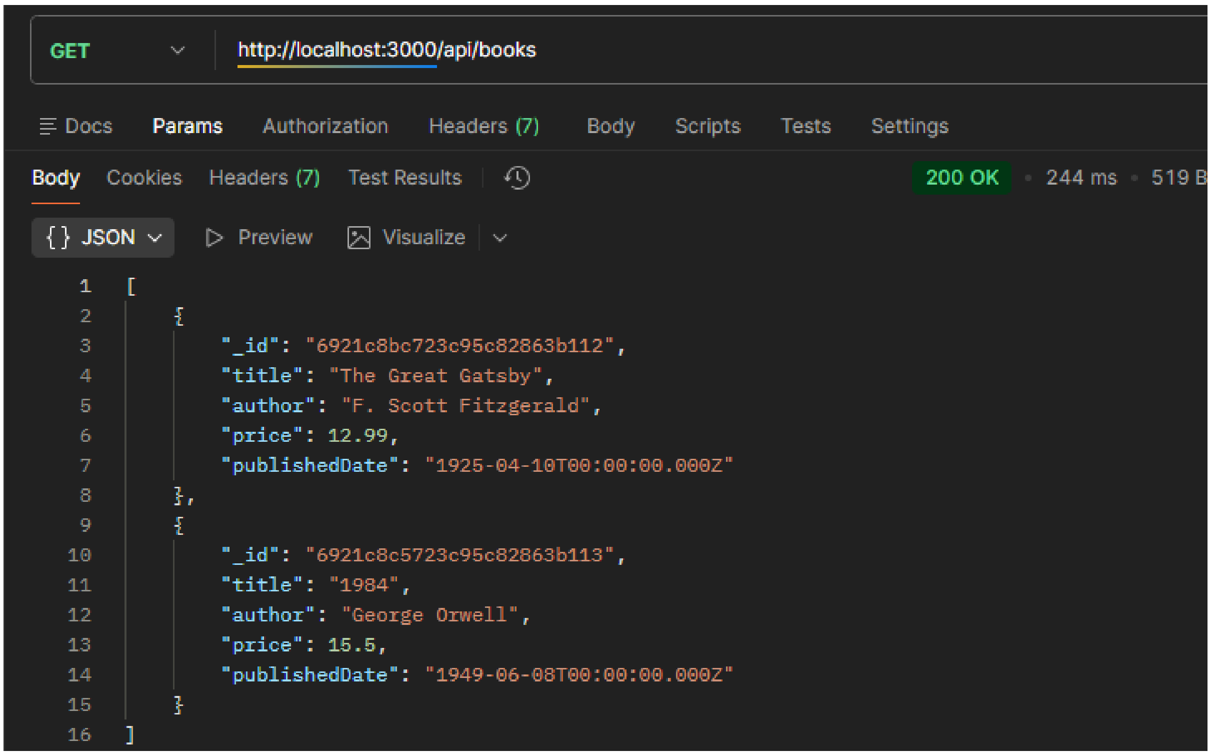


BookManagement > JS server.js > ...

```javascript
const express = require('express');
const mongoose = require('mongoose');
const bookRoutes = require('./bookRoutes');

const app = express();
const PORT = 3000;
const DB_URI = 'mongodb://localhost:27017/libraryDB';

app.use(express.json());

// Connection handling
mongoose.connect(DB_URI)
    .then(() => console.log('MongoDB: Connected to libraryDB'))
    .catch(err => console.error('MongoDB: Connection failed:', err));

// Route mounting
app.use('/api/books', bookRoutes);

// Start the server
app.listen(PORT, () => {
    console.log(`Server running on http://localhost:${PORT}`);
});
```

POSTMAN:

(1) GET all books

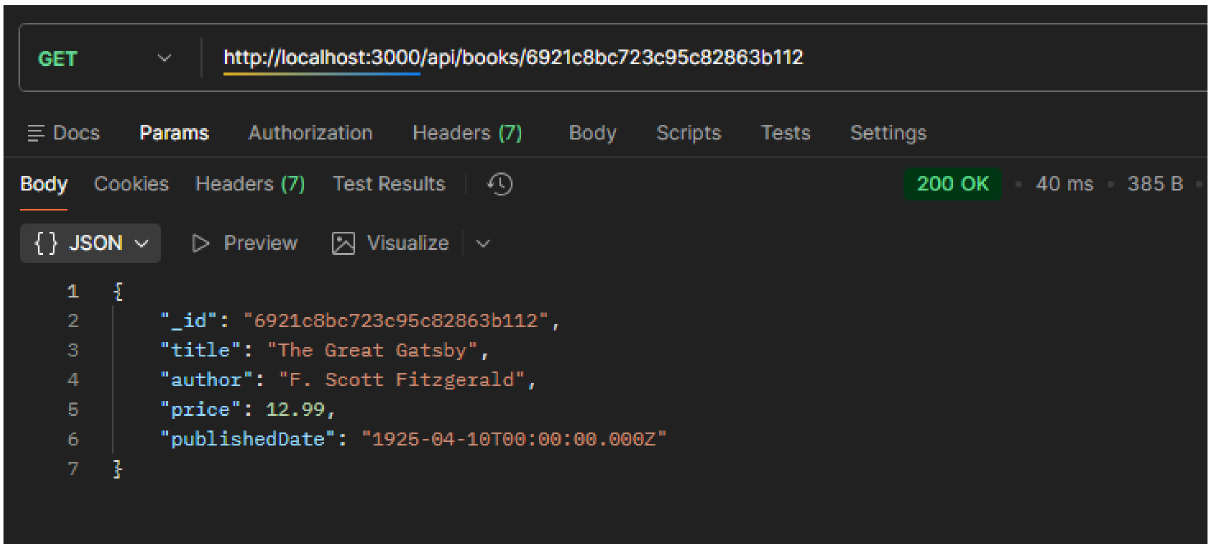


```
http://localhost:3000/api/books
```

```
200 OK • 244 ms • 519 B
```

```
[
    {
        "_id": "6921c8bc723c95c82863b112",
        "title": "The Great Gatsby",
        "author": "F. Scott Fitzgerald",
        "price": 12.99,
        "publishedDate": "1925-04-10T00:00:00.000Z"
    },
    {
        "_id": "6921c8c5723c95c82863b113",
        "title": "1984",
        "author": "George Orwell",
        "price": 15.5,
        "publishedDate": "1949-06-08T00:00:00.000Z"
    }
]
```

(2) GET book by ID

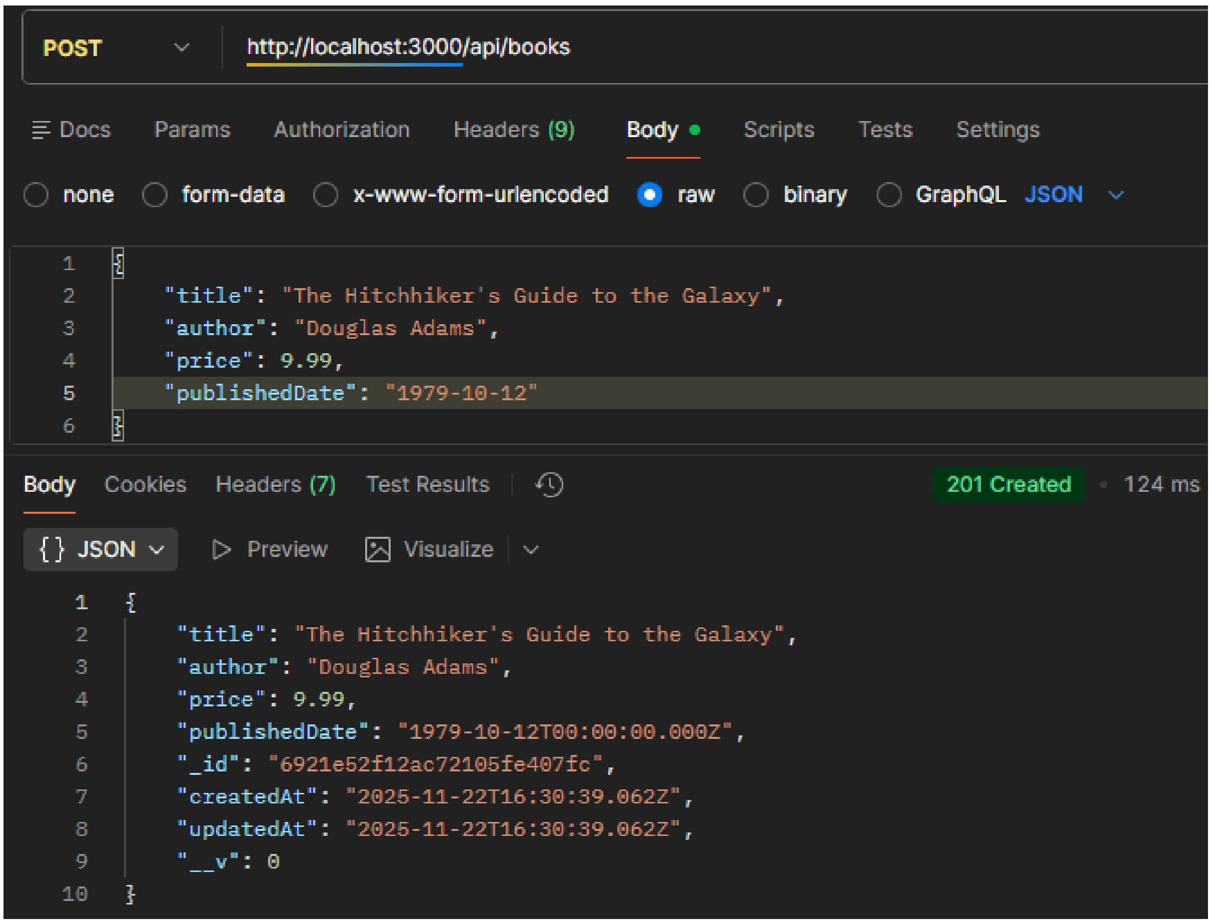


```
http://localhost:3000/api/books/6921c8bc723c95c82863b112
```

```
200 OK • 40 ms • 385 B
```

```
{
    "_id": "6921c8bc723c95c82863b112",
    "title": "The Great Gatsby",
    "author": "F. Scott Fitzgerald",
    "price": 12.99,
    "publishedDate": "1925-04-10T00:00:00.000Z"
}
```

(3) POST new book



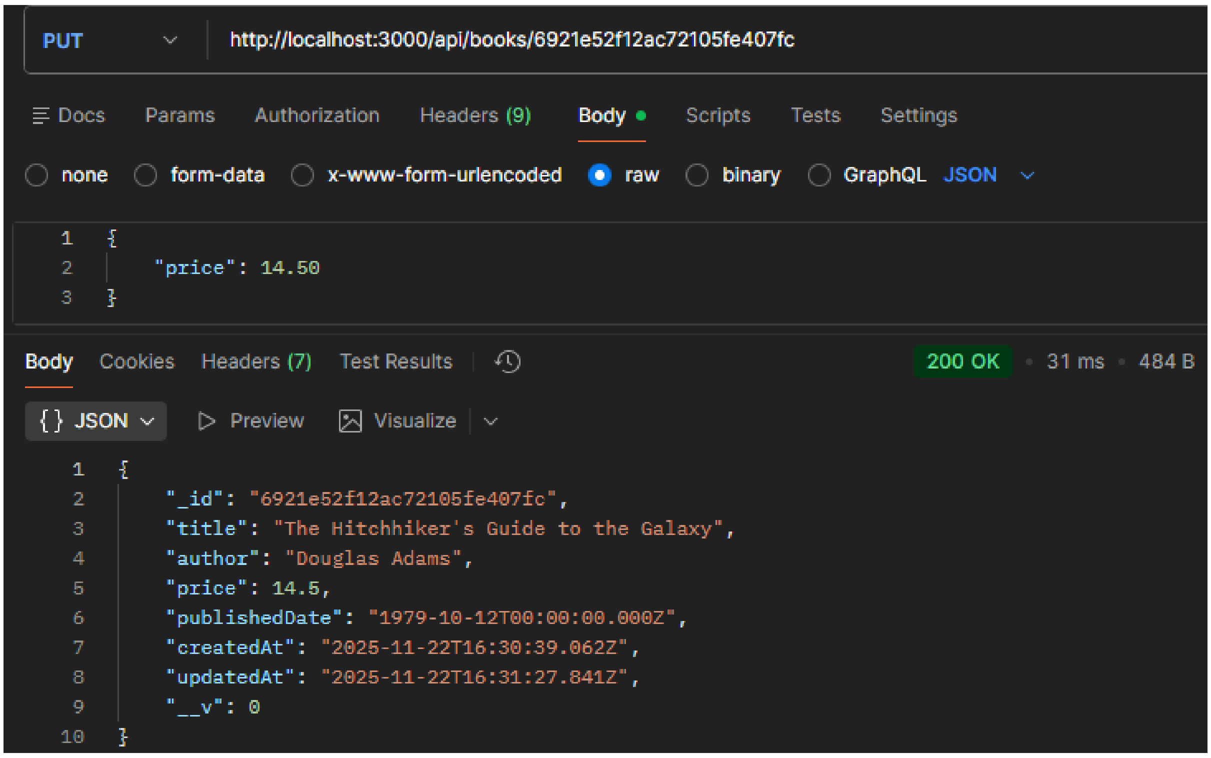
```
http://localhost:3000/api/books
```

```
{
    "title": "The Hitchhiker's Guide to the Galaxy",
    "author": "Douglas Adams",
    "price": 9.99,
    "publishedDate": "1979-10-12"
}
```

201 Created • 124 ms

```
{
    "title": "The Hitchhiker's Guide to the Galaxy",
    "author": "Douglas Adams",
    "price": 9.99,
    "publishedDate": "1979-10-12T00:00:00.000Z",
    "_id": "6921e52f12ac72105fe407fc",
    "createdAt": "2025-11-22T16:30:39.062Z",
    "updatedAt": "2025-11-22T16:30:39.062Z",
    "__v": 0
}
```

(4) PUT update book


PUT http://localhost:3000/api/books/6921e52f12ac72105fe407fc

Docs Params Authorization Headers (9) Body ● Scripts Tests Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

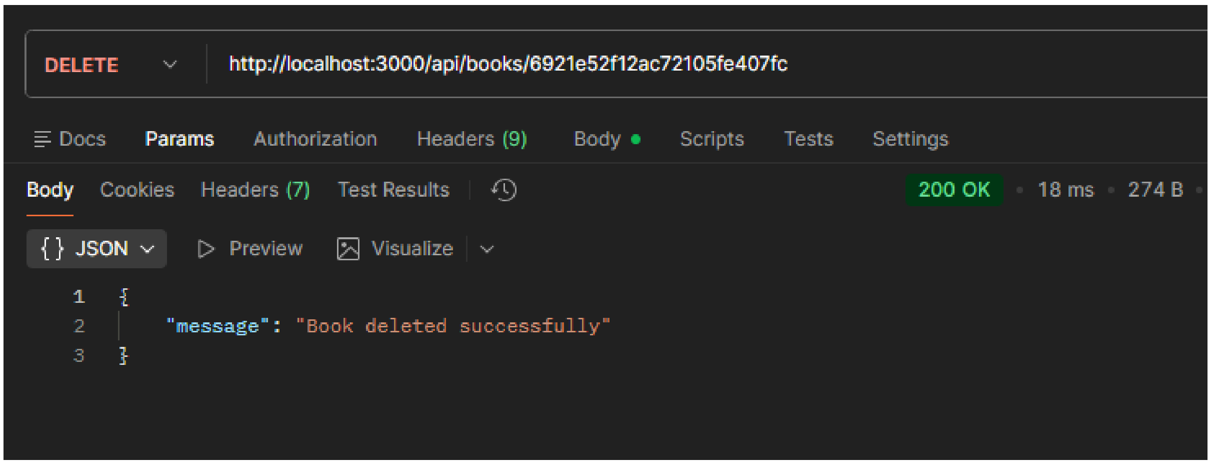
```
{
    "price": 14.50
}
```

Body Cookies Headers (7) Test Results — 200 OK • 31 ms • 484 B

{ } JSON Preview Visualize

```
{
    "_id": "6921e52f12ac72105fe407fc",
    "title": "The Hitchhiker's Guide to the Galaxy",
    "author": "Douglas Adams",
    "price": 14.5,
    "publishedDate": "1979-10-12T00:00:00.000Z",
    "createdAt": "2025-11-22T16:30:39.062Z",
    "updatedAt": "2025-11-22T16:31:27.841Z",
    "__v": 0
}
```

(5) DELETE book by ID


DELETE http://localhost:3000/api/books/6921e52f12ac72105fe407fc

Docs Params Authorization Headers (9) Body ● Scripts Tests Settings

Body Cookies Headers (7) Test Results — 200 OK • 18 ms • 274 B

{ } JSON Preview Visualize

```
{
    "message": "Book deleted successfully"
}
```