

## **Lab Mid**

**Program: BS Software Engineering  
Department of Computer Science**



**Submitted to: Ma'am Yasmeen  
Submitted by: Misbah Jabbar  
Roll no: SP22-BSE-011**

**Course Title: Advanced Web Technologies**

**COMSATS University Islamabad, Vehari  
Campus.**

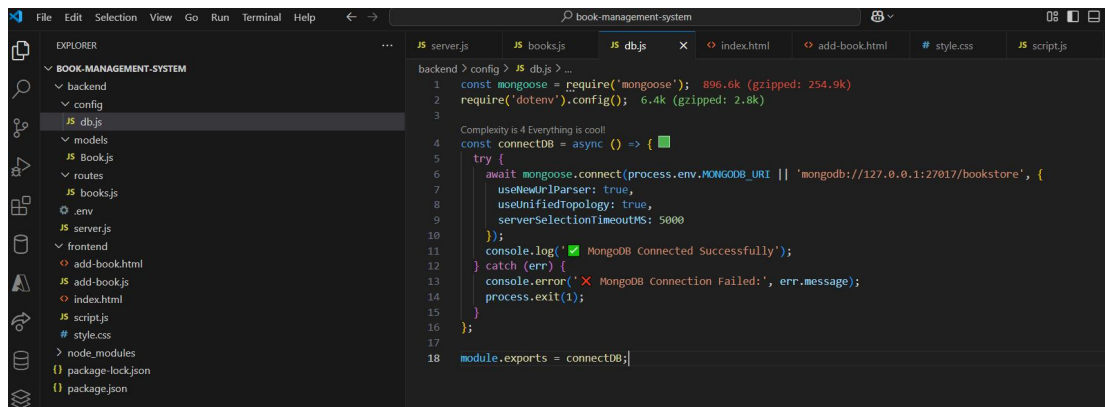
## Question:no:01

**You have been hired as a Full Stack Developer. Your task is to develop a complete web application that allows users to view a list of available books, search for books by author, and add new books to the system. You are required to build both the frontend and the backend components of this application.**

## Solution:

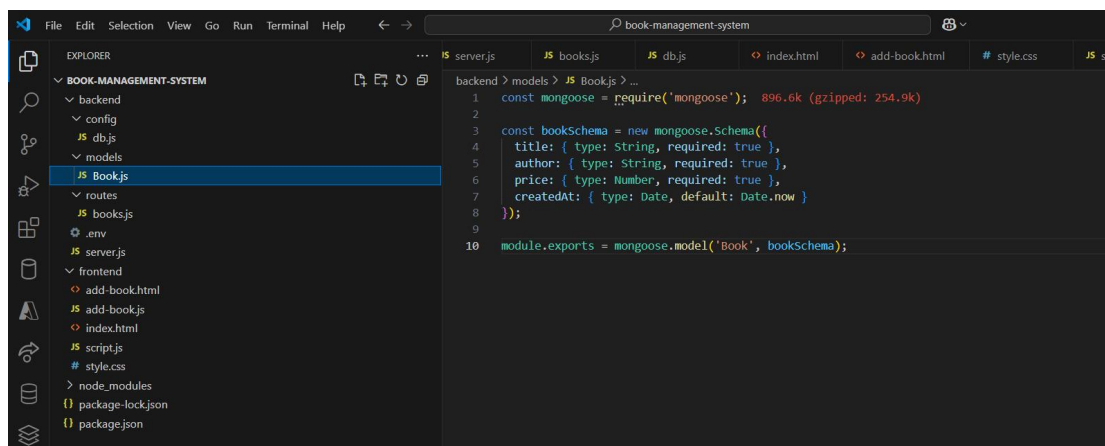
### Backend System

#### Config(db.js):



```
1 const mongoose = require('mongoose'); 896.6k (gzipped: 254.9k)
2 require('dotenv').config(); 6.4k (gzipped: 2.8k)
3
4 // Complexity is 4 Everything is cool!
5
6 const connectDB = async () => {
7   try {
8     await mongoose.connect(process.env.MONGODB_URI || 'mongodb://127.0.0.1:27017/bookstore', {
9       useNewUrlParser: true,
10      useUnifiedTopology: true,
11      serverSelectionTimeoutMS: 5000
12    });
13    console.log('MongoDB Connected Successfully');
14   } catch (err) {
15     console.error('MongoDB connection Failed:', err.message);
16     process.exit(1);
17   }
18 }
19
20 module.exports = connectDB;
```

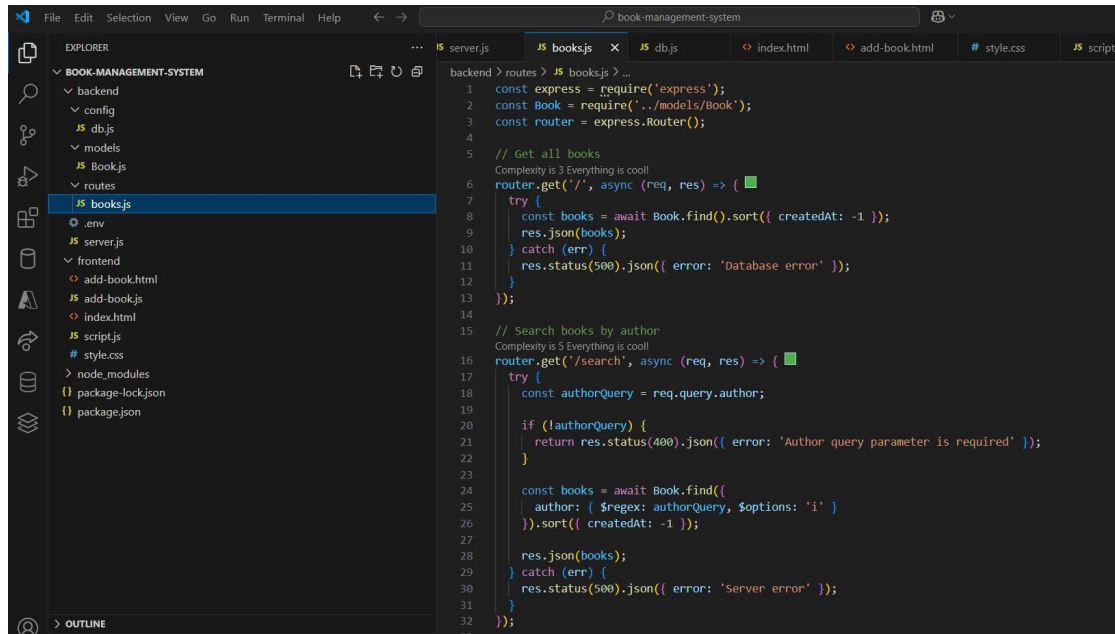
#### Model(Book.js)



```
1 const mongoose = require('mongoose'); 896.6k (gzipped: 254.9k)
2
3 const bookSchema = new mongoose.Schema({
4   title: { type: String, required: true },
5   author: { type: String, required: true },
6   price: { type: Number, required: true },
7   createdAt: { type: Date, default: Date.now }
8 });
9
10 module.exports = mongoose.model('Book', bookSchema);
```

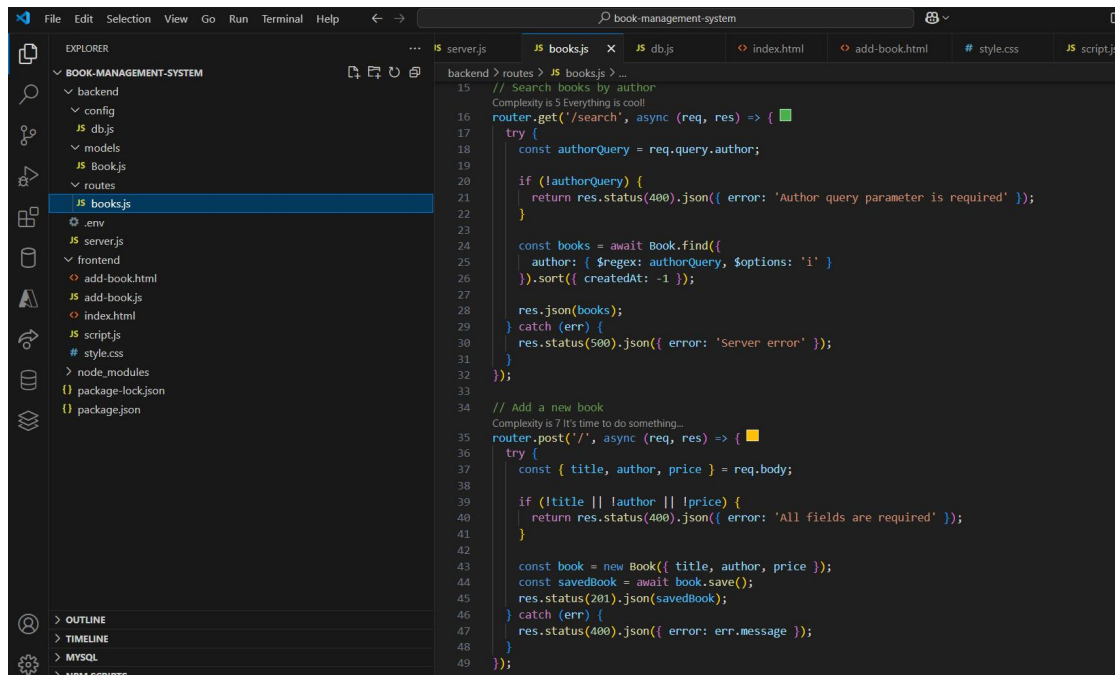
# Routes(books.js)

A)



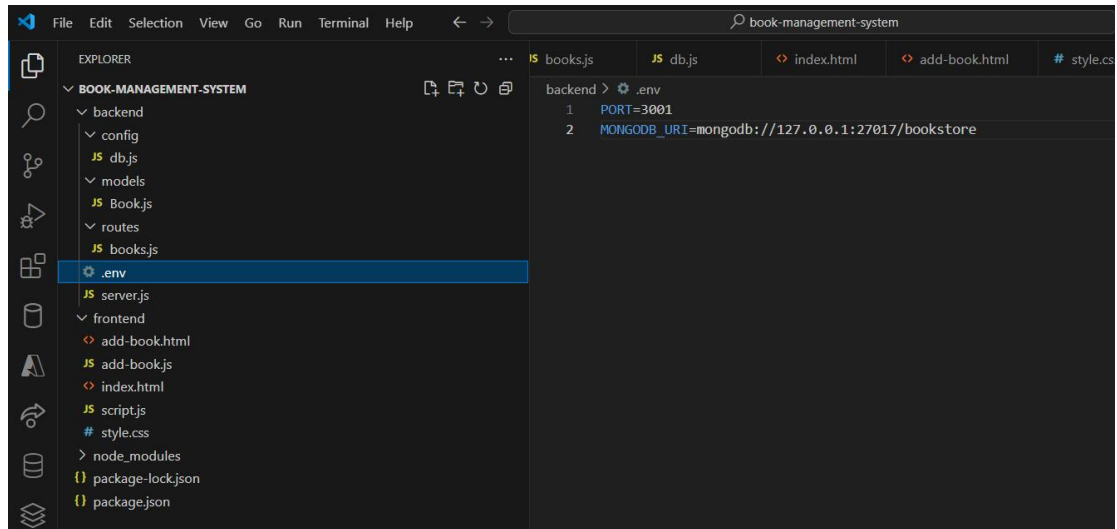
```
1 const express = require('express');
2 const Book = require('../models/Book');
3 const router = express.Router();
4
5 // Get all books
6 Complexity is 3 Everything is cool!
7 router.get('/', async (req, res) => {
8   try {
9     const books = await Book.find().sort({ createdAt: -1 });
10    res.json(books);
11  } catch (err) {
12    res.status(500).json({ error: 'Database error' });
13  }
14});
15
16 // Search books by author
17 Complexity is 5 Everything is cool!
18 router.get('/search', async (req, res) => {
19   try {
20     const authorQuery = req.query.author;
21
22     if (!authorQuery) {
23       return res.status(400).json({ error: 'Author query parameter is required' });
24     }
25
26     const books = await Book.find({
27       author: { $regex: authorQuery, $options: 'i' }
28     }).sort({ createdAt: -1 });
29
30     res.json(books);
31   } catch (err) {
32     res.status(500).json({ error: 'Server error' });
33   }
34});
```

B)



```
15 // Search books by author
16 Complexity is 5 Everything is cool!
17 router.get('/search', async (req, res) => {
18   try {
19     const authorQuery = req.query.author;
20
21     if (!authorQuery) {
22       return res.status(400).json({ error: 'Author query parameter is required' });
23     }
24
25     const books = await Book.find({
26       author: { $regex: authorQuery, $options: 'i' }
27     }).sort({ createdAt: -1 });
28
29     res.json(books);
30   } catch (err) {
31     res.status(500).json({ error: 'Server error' });
32   }
33 });
34
35 // Add a new book
36 Complexity is 7 It's time to do something...
37 router.post('/', async (req, res) => {
38   try {
39     const { title, author, price } = req.body;
40
41     if (!title || !author || !price) {
42       return res.status(400).json({ error: 'All fields are required' });
43     }
44
45     const book = new Book({ title, author, price });
46     const savedBook = await book.save();
47     res.status(201).json(savedBook);
48   } catch (err) {
49     res.status(400).json({ error: err.message });
50   }
51 });
```

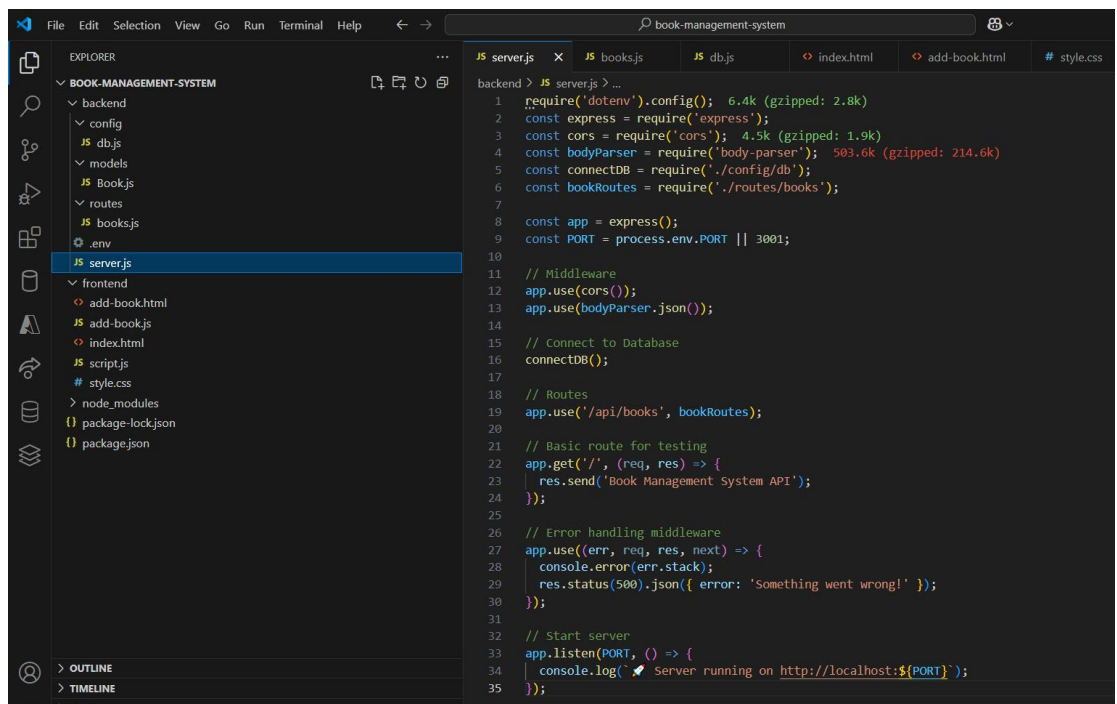
## Env File:



The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows a project named 'BOOK-MANAGEMENT-SYSTEM' with a file tree including 'backend', 'config', 'models', 'routes', and 'books.js'. The '.env' file is selected in the Explorer. The Editor shows the contents of the '.env' file:

```
backend > .env
1 PORT=3001
2 MONGODB_URI=mongodb://127.0.0.1:27017/bookstore
```

## Server.js



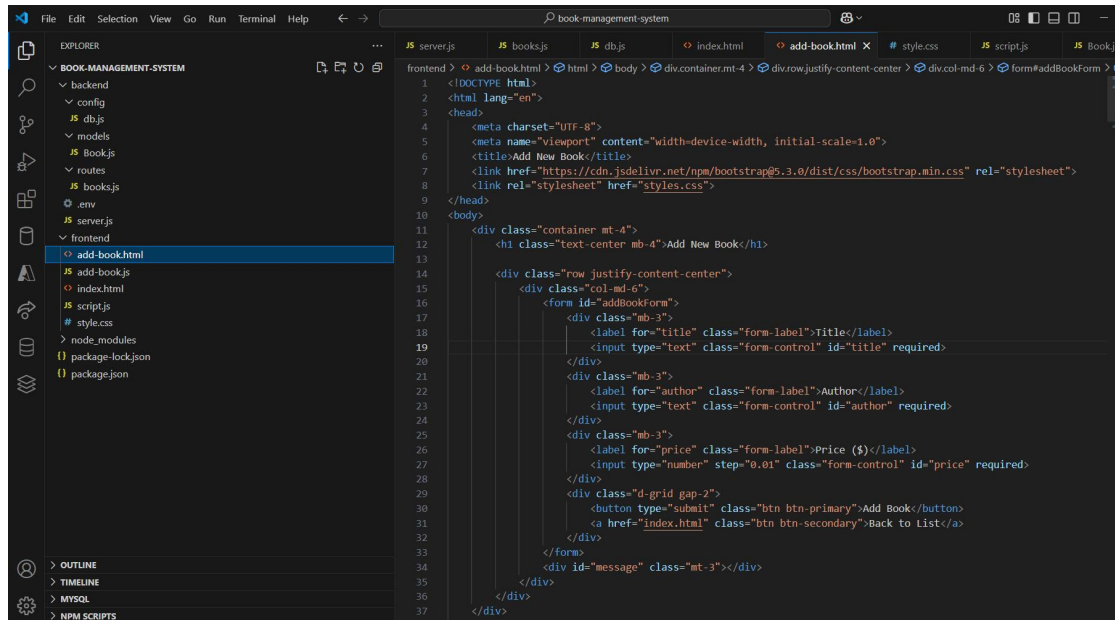
The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows the same project structure as the previous screenshot, but the 'server.js' file is selected in the Explorer. The Editor shows the contents of the 'server.js' file:

```
backend > JS server.js > ...
1 require('dotenv').config(); 6.4k (gzipped: 2.8k)
2 const express = require('express');
3 const cors = require('cors'); 4.5k (gzipped: 1.9k)
4 const bodyParser = require('body-parser'); 503.6k (gzipped: 214.6k)
5 const connectDB = require('./config/db');
6 const bookRoutes = require('./routes/books');
7
8 const app = express();
9 const PORT = process.env.PORT || 3001;
10
11 // Middleware
12 app.use(cors());
13 app.use(bodyParser.json());
14
15 // Connect to Database
16 connectDB();
17
18 // Routes
19 app.use('/api/books', bookRoutes);
20
21 // Basic route for testing
22 app.get('/', (req, res) => {
23   res.send('Book Management System API');
24 });
25
26 // Error handling middleware
27 app.use((err, req, res, next) => {
28   console.error(err.stack);
29   res.status(500).json({ error: 'Something went wrong!' });
30 });
31
32 // Start server
33 app.listen(PORT, () => {
34   console.log('Server running on http://localhost:${PORT}');
35 });
```

## Frontend System

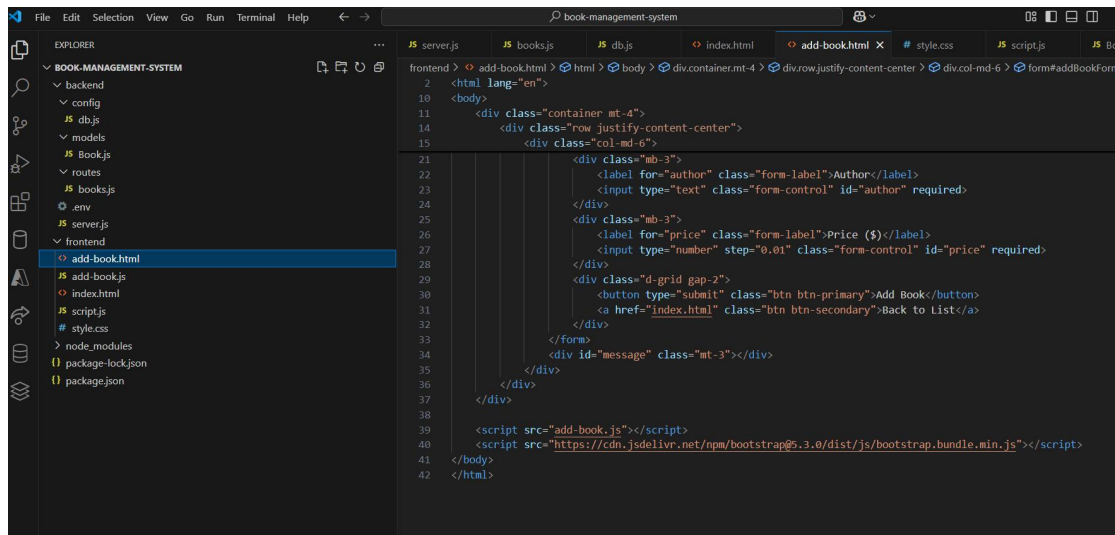
### Add-book.html:

A)



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Add New Book</title>
7   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
8   <link rel="stylesheet" href="styles.css">
9 </head>
10 <body>
11   <div class="container mt-4">
12     <h1 class="text-center mb-4">Add New Book</h1>
13
14     <div class="row justify-content-center">
15       <div class="col-md-6">
16         <form id="addBookForm">
17           <div class="mb-3">
18             <label for="title" class="form-label">Title</label>
19             <input type="text" class="form-control" id="title" required>
20           </div>
21           <div class="mb-3">
22             <label for="author" class="form-label">Author</label>
23             <input type="text" class="form-control" id="author" required>
24           </div>
25           <div class="mb-3">
26             <label for="price" class="form-label">Price ($)</label>
27             <input type="number" step="0.01" class="form-control" id="price" required>
28           </div>
29           <div class="d-grid gap-2">
30             <button type="submit" class="btn btn-primary">Add Book</button>
31             <a href="index.html" class="btn btn-secondary">Back to List</a>
32           </div>
33         </form>
34       <div id="message" class="mt-3"></div>
35     </div>
36   </div>
37 </body>
```

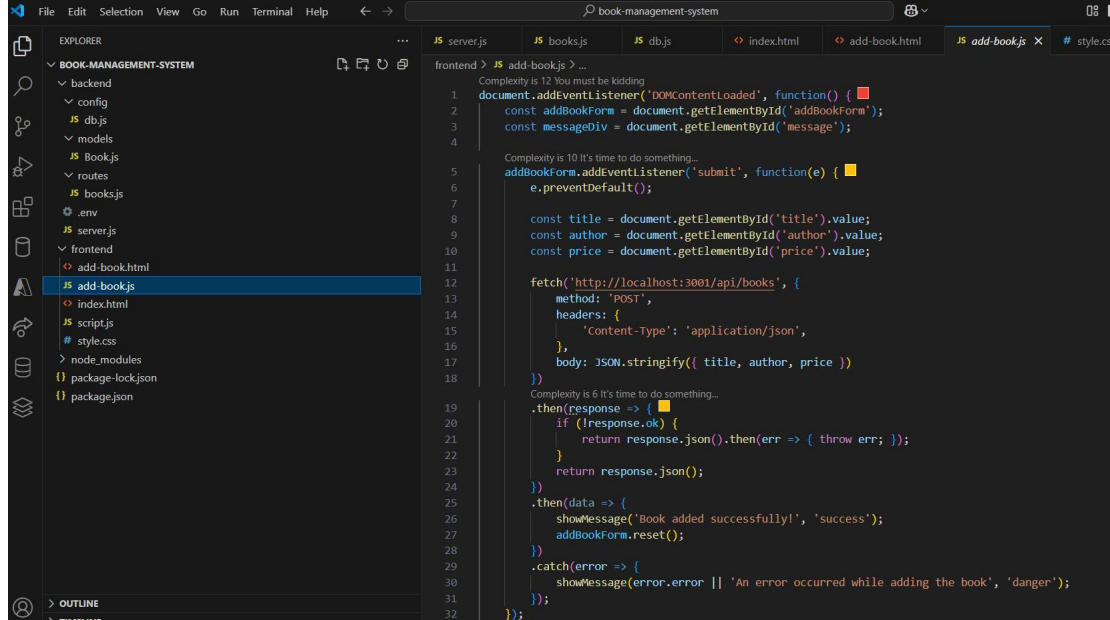
B)



```
2 <html lang="en">
10 <body>
11   <div class="container mt-4">
14     <div class="row justify-content-center">
15       <div class="col-md-6">
21         <div class="mb-3">
22           <label for="author" class="form-label">Author</label>
23           <input type="text" class="form-control" id="author" required>
24         </div>
25         <div class="mb-3">
26           <label for="price" class="form-label">Price ($)</label>
27           <input type="number" step="0.01" class="form-control" id="price" required>
28         </div>
29         <div class="d-grid gap-2">
30           <button type="submit" class="btn btn-primary">Add Book</button>
31           <a href="index.html" class="btn btn-secondary">Back to List</a>
32         </div>
33       </div>
34       <div id="message" class="mt-3"></div>
35     </div>
36   </div>
37 </body>
38
39 <script src="add-book.js"></script>
40 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
41
42 </html>
```

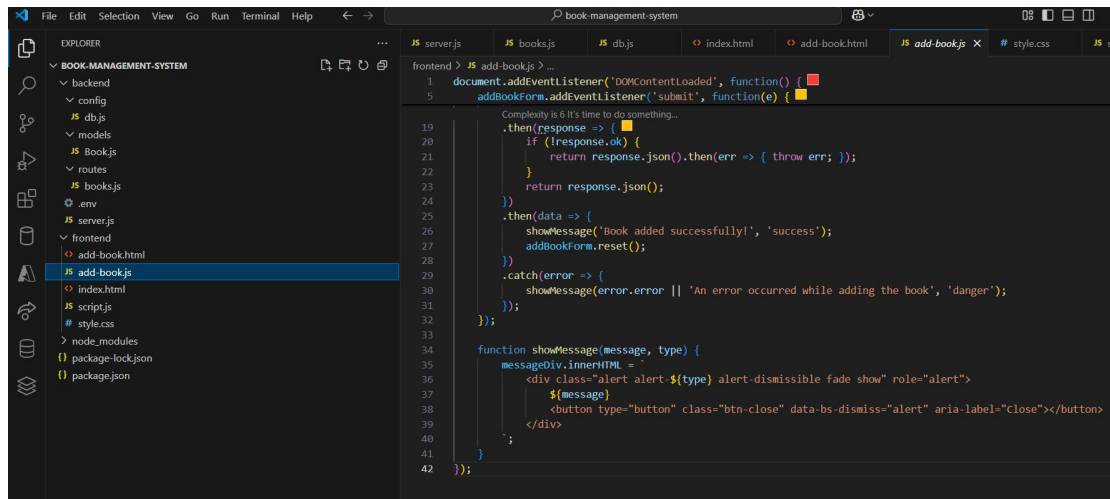
## Add-book.js:

A)



```
1 document.addEventListener('DOMContentLoaded', function() {
2   const addBookForm = document.getElementById('addBookForm');
3   const messageDiv = document.getElementById('message');
4
5   // Complexity is 10. It's time to do something...
6   addBookForm.addEventListener('submit', function(e) {
7     e.preventDefault();
8
9     const title = document.getElementById('title').value;
10    const author = document.getElementById('author').value;
11    const price = document.getElementById('price').value;
12
13    fetch('http://localhost:3001/api/books', {
14      method: 'POST',
15      headers: {
16        'Content-Type': 'application/json',
17      },
18      body: JSON.stringify({ title, author, price })
19    })
20    .then(response => {
21      if (!response.ok) {
22        return response.json().then(err => { throw err; });
23      }
24      return response.json();
25    })
26    .then(data => {
27      showMessage('Book added successfully!', 'success');
28      addBookForm.reset();
29    })
30    .catch(error => {
31      showMessage(error.error || 'An error occurred while adding the book', 'danger');
32    });
33  });
34
35  function showMessage(message, type) {
36    messageDiv.innerHTML = `
37      <div class="alert alert-${type} alert-dismissible fade show" role="alert">
38        ${message}
39        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
40      </div>
41    `;
42  }
```

B)



```
1 document.addEventListener('DOMContentLoaded', function() {
2   const addBookForm = document.getElementById('addBookForm');
3   const messageDiv = document.getElementById('message');
4
5   // Complexity is 10. It's time to do something...
6   addBookForm.addEventListener('submit', function(e) {
7     e.preventDefault();
8
9     const title = document.getElementById('title').value;
10    const author = document.getElementById('author').value;
11    const price = document.getElementById('price').value;
12
13    fetch('http://localhost:3001/api/books', {
14      method: 'POST',
15      headers: {
16        'Content-Type': 'application/json',
17      },
18      body: JSON.stringify({ title, author, price })
19    })
20    .then(response => {
21      if (!response.ok) {
22        return response.json().then(err => { throw err; });
23      }
24      return response.json();
25    })
26    .then(data => {
27      showMessage('Book added successfully!', 'success');
28      addBookForm.reset();
29    })
30    .catch(error => {
31      showMessage(error.error || 'An error occurred while adding the book', 'danger');
32    });
33  });
34
35  function showMessage(message, type) {
36    messageDiv.innerHTML = `
37      <div class="alert alert-${type} alert-dismissible fade show" role="alert">
38        ${message}
39        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
40      </div>
41    `;
42  }
```

# Index.html:

A)

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Book Management System</title>
7   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
8   <link rel="stylesheet" href="styles.css">
9 </head>
10 <body>
11   <div class="container mt-4">
12     <h1 class="text-center mb-4">Book Management System</h1>
13
14     <div class="row mb-4">
15       <div class="col-md-6">
16         <input type="text" id="searchInput" class="form-control" placeholder="Search by author...">
17       </div>
18       <div class="col-md-6 text-end">
19         <a href="add-book.html" class="btn btn-primary">Add New Book</a>
20       </div>
21     </div>
22
23     <div class="table-responsive">
24       <table class="table table-striped table-hover">
25         <thead class="table-dark">
26           <tr>
27             <th>ID</th>
28             <th>Title</th>
29             <th>Author</th>
30             <th>Price ($)</th>
31           </tr>
32         </thead>
33         <tbody id="booksTableBody">
34           <!-- Books will be loaded here -->
35         </tbody>
36       </table>
37     </div>
38   </div>
39
40   <script src="script.js"></script>
41   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
42 </body>
43 </html>
```

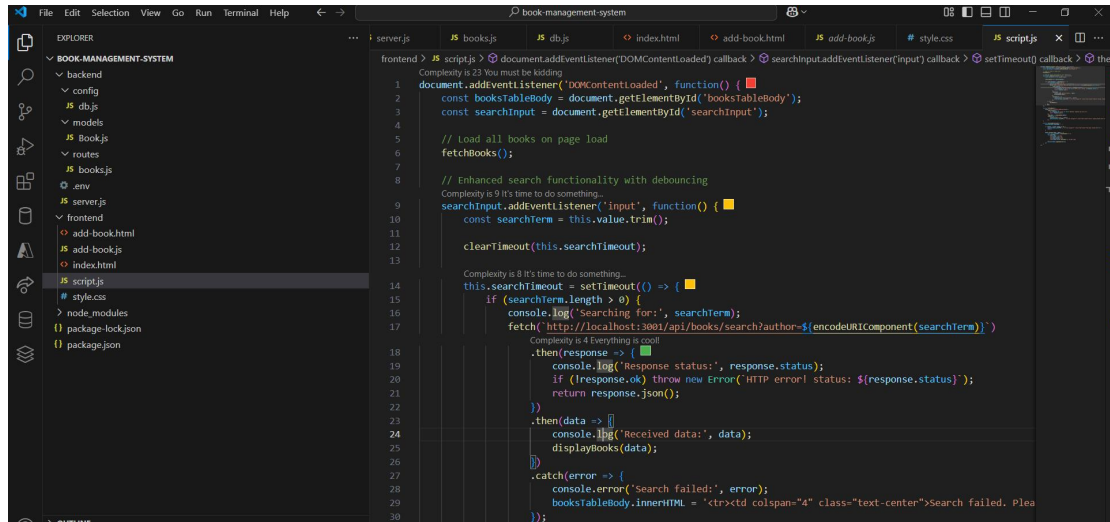
B)

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Book Management System</title>
7   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
8   <link rel="stylesheet" href="styles.css">
9 </head>
10 <body>
11   <div class="container mt-4">
12     <h1 class="text-center mb-4">Book Management System</h1>
13
14     <div class="row mb-4">
15       <div class="col-md-6">
16         <input type="text" id="searchInput" class="form-control" placeholder="Search by author...">
17       </div>
18       <div class="col-md-6 text-end">
19         <a href="add-book.html" class="btn btn-primary">Add New Book</a>
20       </div>
21     </div>
22
23     <div class="table-responsive">
24       <table class="table table-striped table-hover">
25         <thead class="table-dark">
26           <tr>
27             <th>ID</th>
28             <th>Title</th>
29             <th>Author</th>
30             <th>Price ($)</th>
31           </tr>
32         </thead>
33         <tbody id="booksTableBody">
34           <!-- Books will be loaded here -->
35         </tbody>
36       </table>
37     </div>
38   </div>
39
40   <script src="script.js"></script>
41   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
42 </body>
43 </html>
```



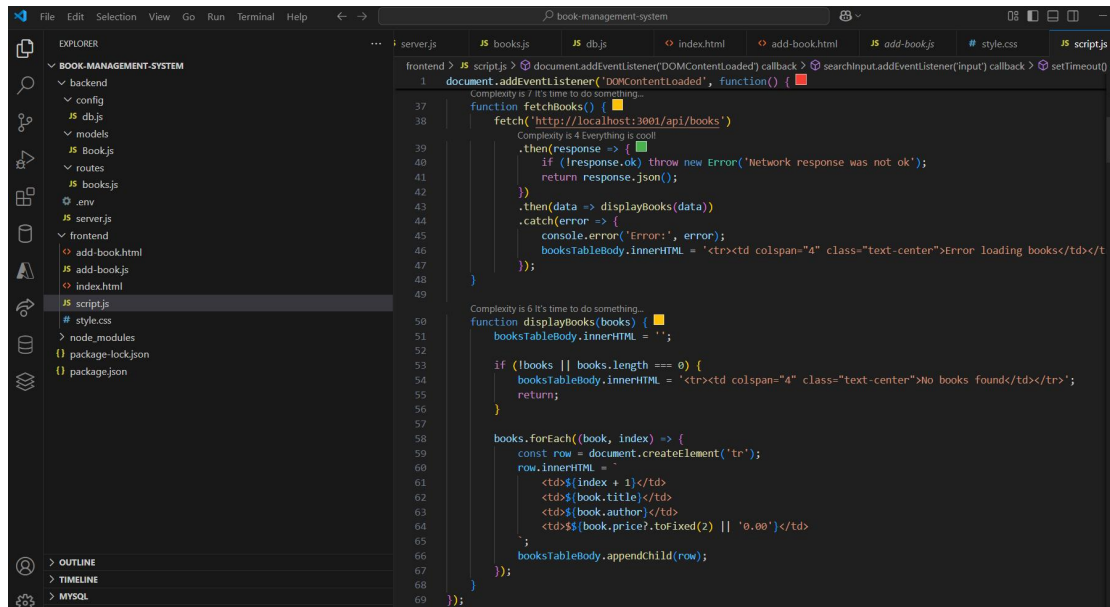
# Script.js:

A)



```
1 document.addEventListener('DOMContentLoaded', function() {  
2   const booksTableBody = document.getElementById('booksTableBody');  
3   const searchInput = document.getElementById('searchInput');  
4  
5   // Load all books on page load  
6   fetchBooks();  
7  
8   // Enhanced search functionality with debouncing  
9   searchInput.addEventListener('input', function() {  
10     const searchTerm = this.value.trim();  
11  
12     clearTimeout(this.searchTimeout);  
13  
14     // Complexity is 8 It's time to do something...  
15     this.searchTimeout = setTimeout(() => {  
16       if (searchTerm.length > 0) {  
17         console.log('Searching for:', searchTerm);  
18         fetch('http://localhost:3001/api/books/search?author=${encodeURIComponent(searchTerm)}')  
19           .then(response => {  
20             console.log('Response status:', response.status);  
21             if (!response.ok) throw new Error('HTTP error! status: ${response.status}');  
22             return response.json();  
23           })  
24           .then(data => {  
25             console.log('Received data:', data);  
26             displayBooks(data);  
27           })  
28           .catch(error => {  
29             console.error('Search failed:', error);  
30             booksTableBody.innerHTML = `<tr><td colspan="4" class="text-center">Search failed. Please try again.</td></tr>`;  
31           });  
32       }  
33     }, 300);  
34   });  
35 }  
36  
37 // Complexity is 23 You must be kidding  
38 function fetchBooks() {  
39   fetch('http://localhost:3001/api/books')  
40     .then(response => {  
41       if (!response.ok) throw new Error('Network response was not ok');  
42       return response.json();  
43     })  
44     .then(data => displayBooks(data))  
45     .catch(error => {  
46       console.error('Error:', error);  
47       booksTableBody.innerHTML = `<tr><td colspan="4" class="text-center">Error loading books</td></tr>`;  
48     });  
49 }  
50  
51 // Complexity is 6 It's time to do something...  
52 function displayBooks(books) {  
53   booksTableBody.innerHTML = `<tr><td colspan="4" class="text-center">No books found</td></tr>`;  
54   return;  
55 }  
56  
57 books.forEach((book, index) => {  
58   const row = document.createElement('tr');  
59   row.innerHTML = `<tr><td>${index + 1}</td><td>${book.title}</td><td>${book.author}</td><td>${book.price?.toFixed(2) || '0.00'}</td></tr>`;  
60   booksTableBody.appendChild(row);  
61 }  
62 }  
63 }  
64 }  
65 }  
66 }  
67 }  
68 }  
69 }  
70 }  
71 }  
72 }  
73 }  
74 }  
75 }  
76 }  
77 }  
78 }  
79 }  
80 }  
81 }  
82 }  
83 }  
84 }  
85 }  
86 }  
87 }  
88 }  
89 }  
90 }  
91 }  
92 }  
93 }  
94 }  
95 }  
96 }  
97 }  
98 }  
99 }  
100 }
```

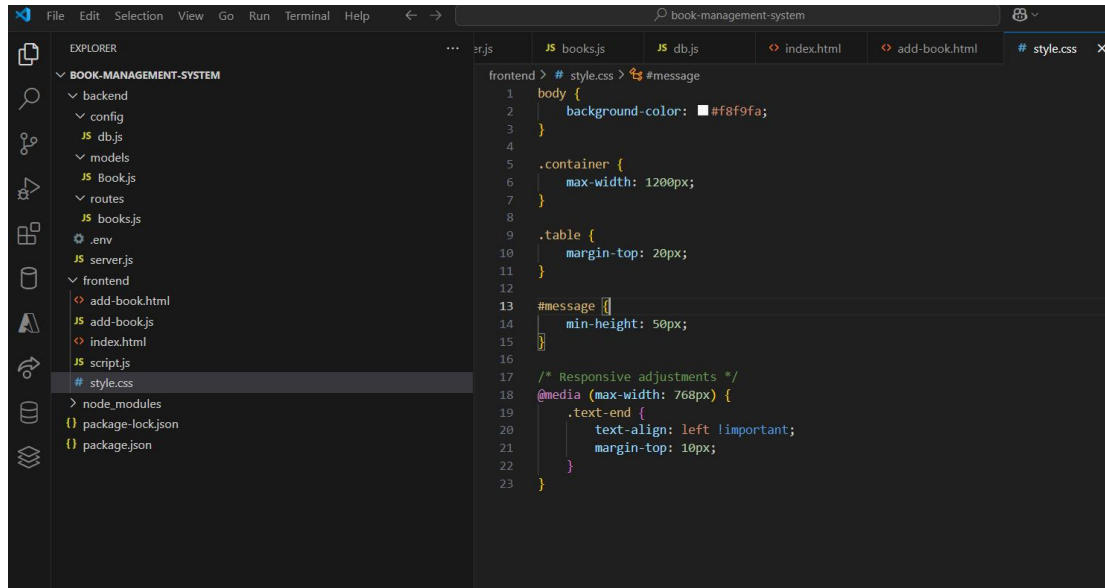
B)



```
1 document.addEventListener('DOMContentLoaded', function() {  
2   const booksTableBody = document.getElementById('booksTableBody');  
3   const searchInput = document.getElementById('searchInput');  
4  
5   // Load all books on page load  
6   fetchBooks();  
7  
8   // Enhanced search functionality with debouncing  
9   searchInput.addEventListener('input', function() {  
10     const searchTerm = this.value.trim();  
11  
12     clearTimeout(this.searchTimeout);  
13  
14     // Complexity is 8 It's time to do something...  
15     this.searchTimeout = setTimeout(() => {  
16       if (searchTerm.length > 0) {  
17         console.log('Searching for:', searchTerm);  
18         fetch('http://localhost:3001/api/books/search?author=${encodeURIComponent(searchTerm)}')  
19           .then(response => {  
20             console.log('Response status:', response.status);  
21             if (!response.ok) throw new Error('HTTP error! status: ${response.status}');  
22             return response.json();  
23           })  
24           .then(data => {  
25             console.log('Received data:', data);  
26             displayBooks(data);  
27           })  
28           .catch(error => {  
29             console.error('Search failed:', error);  
30             booksTableBody.innerHTML = `<tr><td colspan="4" class="text-center">Search failed. Please try again.</td></tr>`;  
31           });  
32       }  
33     }, 300);  
34   });  
35 }  
36  
37 // Complexity is 23 You must be kidding  
38 function fetchBooks() {  
39   fetch('http://localhost:3001/api/books')  
40     .then(response => {  
41       if (!response.ok) throw new Error('Network response was not ok');  
42       return response.json();  
43     })  
44     .then(data => displayBooks(data))  
45     .catch(error => {  
46       console.error('Error:', error);  
47       booksTableBody.innerHTML = `<tr><td colspan="4" class="text-center">Error loading books</td></tr>`;  
48     });  
49 }  
50  
51 // Complexity is 6 It's time to do something...  
52 function displayBooks(books) {  
53   booksTableBody.innerHTML = `<tr><td colspan="4" class="text-center">No books found</td></tr>`;  
54   return;  
55 }  
56  
57 books.forEach((book, index) => {  
58   const row = document.createElement('tr');  
59   row.innerHTML = `<tr><td>${index + 1}</td><td>${book.title}</td><td>${book.author}</td><td>${book.price?.toFixed(2) || '0.00'}</td></tr>`;  
60   booksTableBody.appendChild(row);  
61 }  
62 }  
63 }  
64 }  
65 }  
66 }  
67 }  
68 }  
69 }  
70 }  
71 }  
72 }  
73 }  
74 }  
75 }  
76 }  
77 }  
78 }  
79 }  
80 }  
81 }  
82 }  
83 }  
84 }  
85 }  
86 }  
87 }  
88 }  
89 }  
90 }  
91 }  
92 }  
93 }  
94 }  
95 }  
96 }  
97 }  
98 }  
99 }  
100 }
```



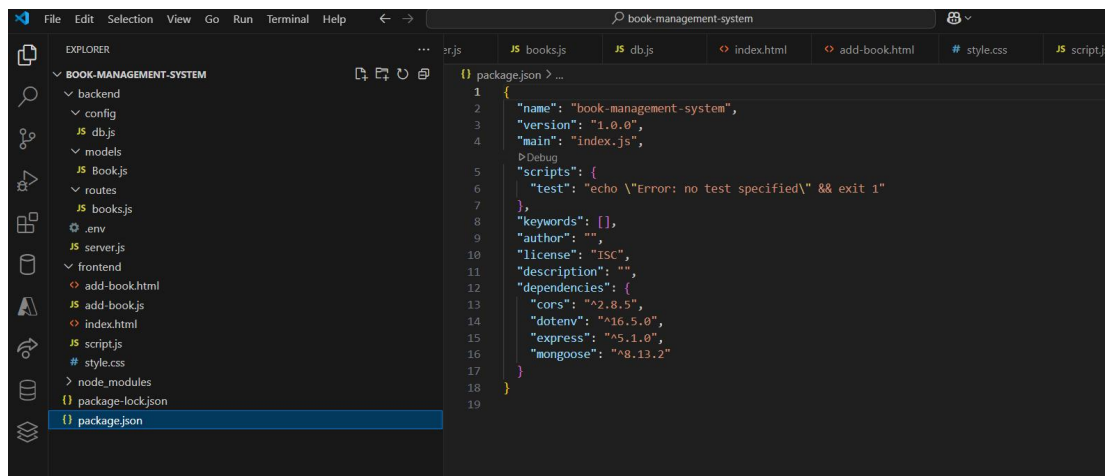
## Styles.css:



The screenshot shows the Visual Studio Code editor with a project named 'book-management-system'. The Explorer sidebar on the left shows the file structure, with 'style.css' selected under the 'frontend' directory. The main editor area displays the content of 'style.css'.

```
frontend > # style.css > #message
1  body {
2      background-color: #f8f9fa;
3  }
4
5  .container {
6      max-width: 1200px;
7  }
8
9  .table {
10     margin-top: 20px;
11 }
12
13 #message {
14     min-height: 50px;
15 }
16
17 /* Responsive adjustments */
18 @media (max-width: 768px) {
19     .text-end {
20         text-align: left !important;
21         margin-top: 10px;
22     }
23 }
```

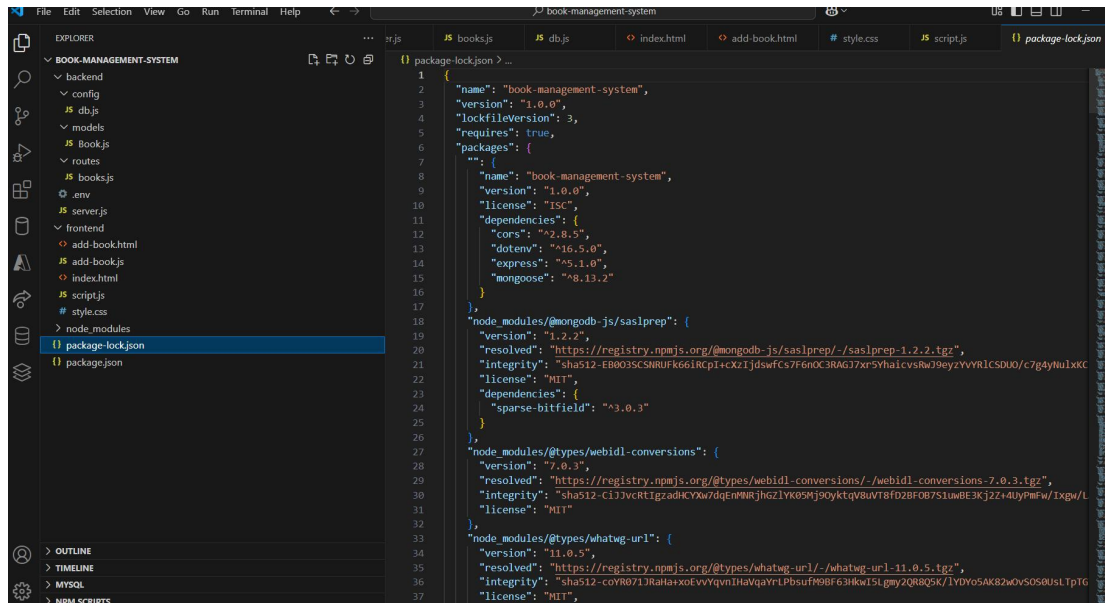
## Package.json:



The screenshot shows the Visual Studio Code editor with the same project. The Explorer sidebar shows 'package.json' selected under the root directory. The main editor area displays the content of 'package.json'.

```
package.json > ...
1  {
2      "name": "book-management-system",
3      "version": "1.0.0",
4      "main": "index.js",
5      "scripts": {
6          "test": "echo \\Error: no test specified\\ && exit 1"
7      },
8      "keywords": [],
9      "author": "",
10     "license": "ISC",
11     "description": "",
12     "dependencies": {
13         "cors": "^2.8.5",
14         "dotenv": "^16.5.0",
15         "express": "^5.1.0",
16         "mongoose": "^8.13.2"
17     }
18 }
19
```

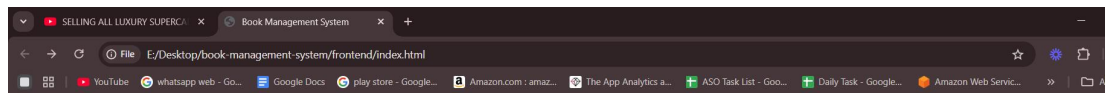
## Package-Lock.json:



## Output

## Frontend:

## Index Page:



## Book Management System

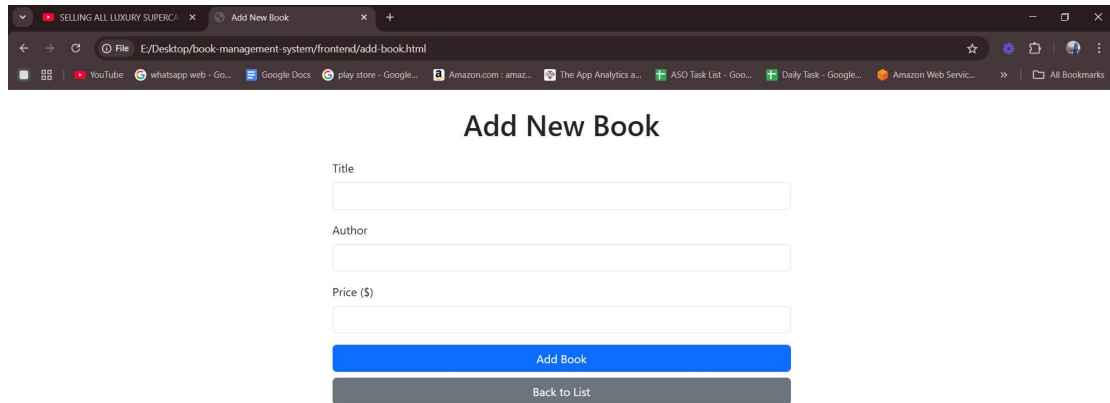
Search by author...

Add New Book

ID	Title	Author	Price (\$)
1	SPM	Adolfo	\$23.99
2	OR	Hamday A.Taha	\$20.99
3	OS	Albert Einstein	\$19.00
4	maths	John Smith	\$20.19
5	The Great Gatsby	F. Scott Fitzgerald	\$10.99
6	Test Book	Test Author	\$19.99

## Add Book Page:

A)



**Add New Book**

Title

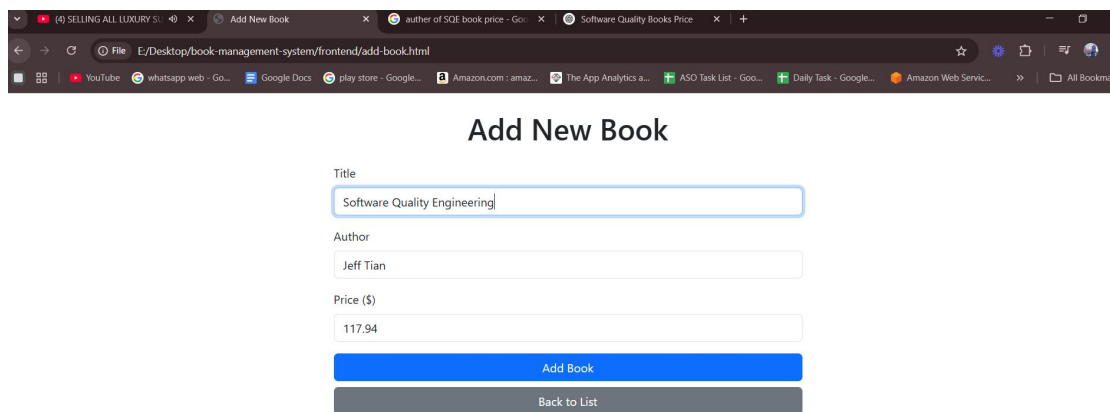
Author

Price (\$)

[Add Book](#)

[Back to List](#)

B)



**Add New Book**

Title

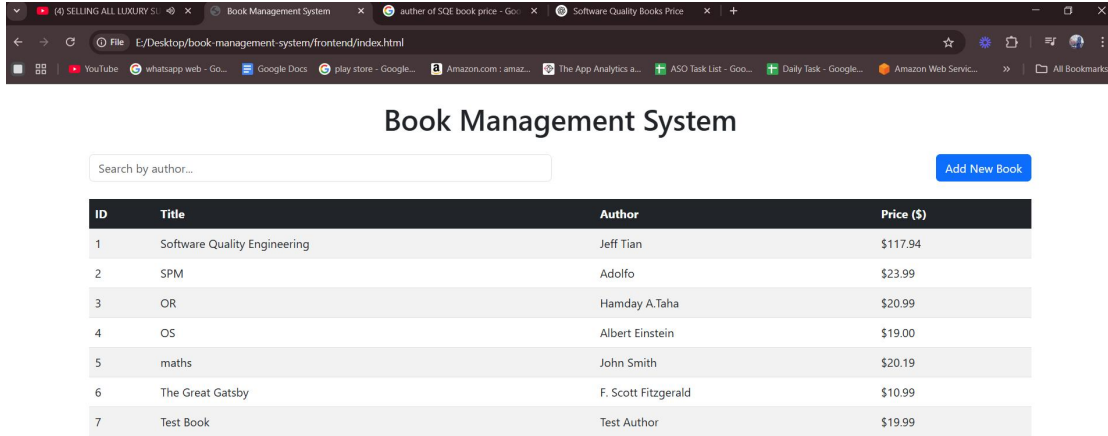
Author

Price (\$)

[Add Book](#)

[Back to List](#)

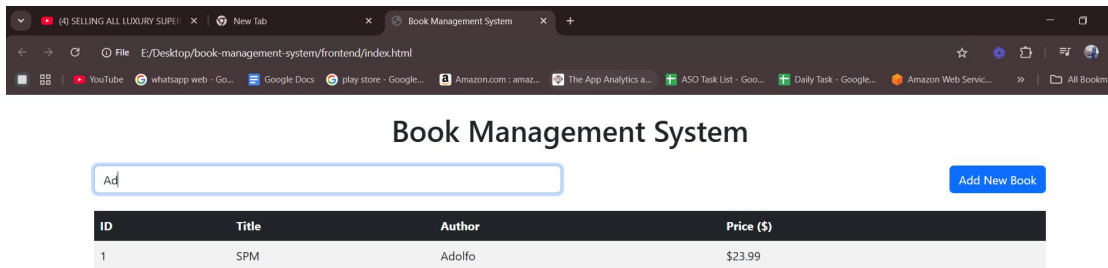
## Index Page while new book add:



The screenshot shows a web browser window with the title "Book Management System". Below the title is a search bar labeled "Search by author..." and a blue button labeled "Add New Book". Below these is a table with 7 rows of book data. The table has columns for ID, Title, Author, and Price (\$).

ID	Title	Author	Price (\$)
1	Software Quality Engineering	Jeff Tian	\$117.94
2	SPM	Adolfo	\$23.99
3	OR	Hamday A.Taha	\$20.99
4	OS	Albert Einstein	\$19.00
5	maths	John Smith	\$20.19
6	The Great Gatsby	F. Scott Fitzgerald	\$10.99
7	Test Book	Test Author	\$19.99

## Search Page:



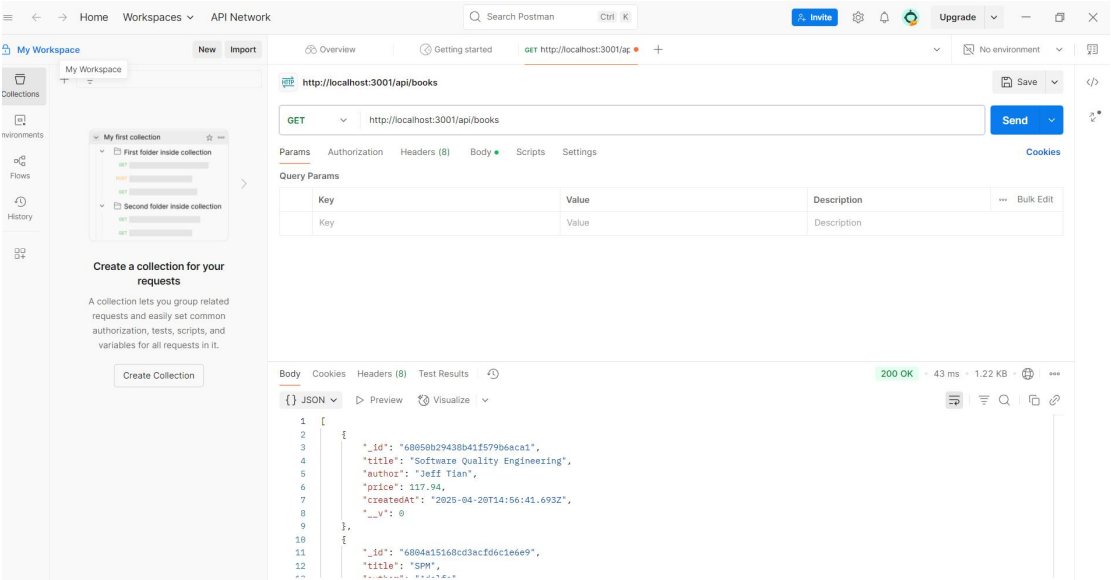
The screenshot shows the same "Book Management System" web page. The search bar now contains the text "Ad". The table below it shows only one row of data, which is the book "SPM" by "Adolfo" priced at "\$23.99".

ID	Title	Author	Price (\$)
1	SPM	Adolfo	\$23.99

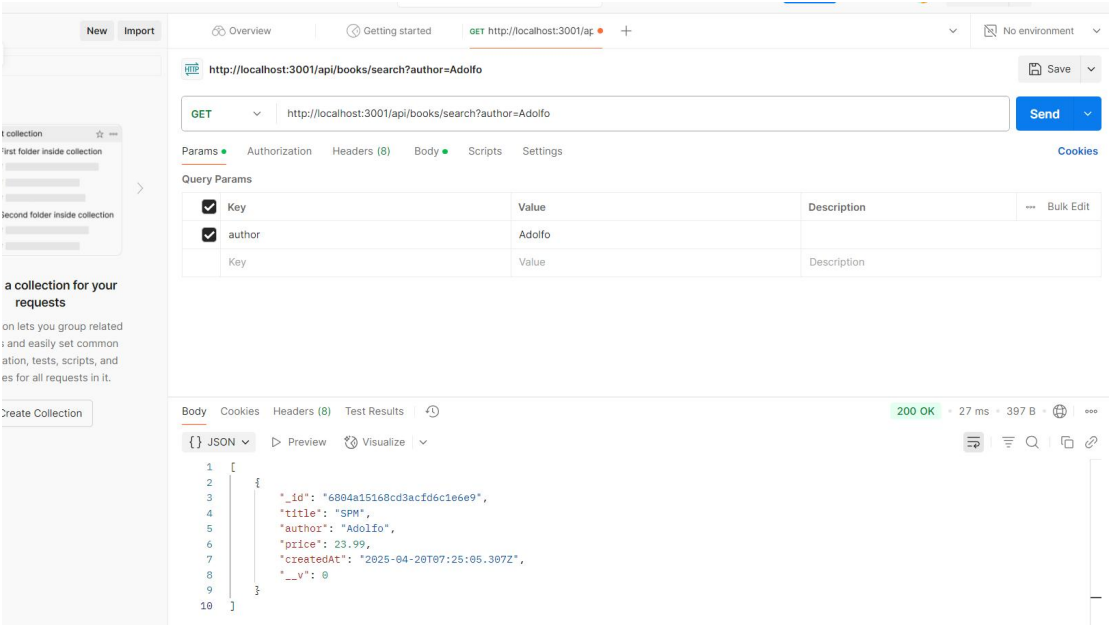
# Backend

## Get Method:

A)



B)



C)

## Get method by Search:

The screenshot shows the Postman interface for a GET request. The URL is `http://localhost:3001/api/books/search?author=Adolfo`. The request is sent, and the response is a 200 OK with a status of 27 ms and 397 B. The response body is a JSON array containing one object:

```
1 [
2   {
3     "_id": "6084a15168cd3acfd6c1e6e9",
4     "title": "SPM",
5     "author": "Adolfo",
6     "price": 23.99,
7     "createdAt": "2025-04-20T07:25:05.307Z",
8     "__v": 0
9   }
10 ]
```

## Post Method:

The screenshot shows the Postman interface for a POST request. The URL is `http://localhost:3001/api/books`. The request is sent, and the response is a 201 Created with a status of 16 ms and 415 B. The response body is a JSON object representing the created book:

```
1 {
2   "title": "L\u00ednea Algebra",
3   "author": "F. Scott",
4   "price": 110.99,
5   "_id": "60850b1d438b41f579b6aca6",
6   "createdAt": "2025-04-20T15:00:13.782Z",
7   "__v": 0
8 }
```

# Mongodb

MongoDB Compass - localhost:27017/bookstore.books

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (1)

Search connections

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

books +

localhost:27017 > bookstore > books

Documents 7 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 - 7 of 7

```
author: "Albert Einstein"
price: 19
createdAt: 2025-04-20T07:01:24.228+00:00
__v: 0
```

```
{
  "_id": "ObjectId('68049dcb32017896db5ff9e9')",
  "title": "DB",
  "author": "Hamday A.Taha",
  "price": 20.99,
  "createdAt": 2025-04-20T07:10:03.099+00:00,
  "__v": 0
}
```

```
{
  "_id": "ObjectId('6804a15168cd3acf6d6c1e6e9')",
  "title": "SPN",
  "author": "Adolfo",
  "price": 23.99,
  "createdAt": 2025-04-20T07:25:05.307+00:00,
  "__v": 0
}
```

```
{
  "_id": "ObjectId('68050b29438b41f579b6aca1')",
  "title": "Software Quality Engineering",
  "author": "Jeff Tian",
  "price": 117.94,
  "createdAt": 2025-04-20T14:56:41.693+00:00,
  "__v": 0
}
```