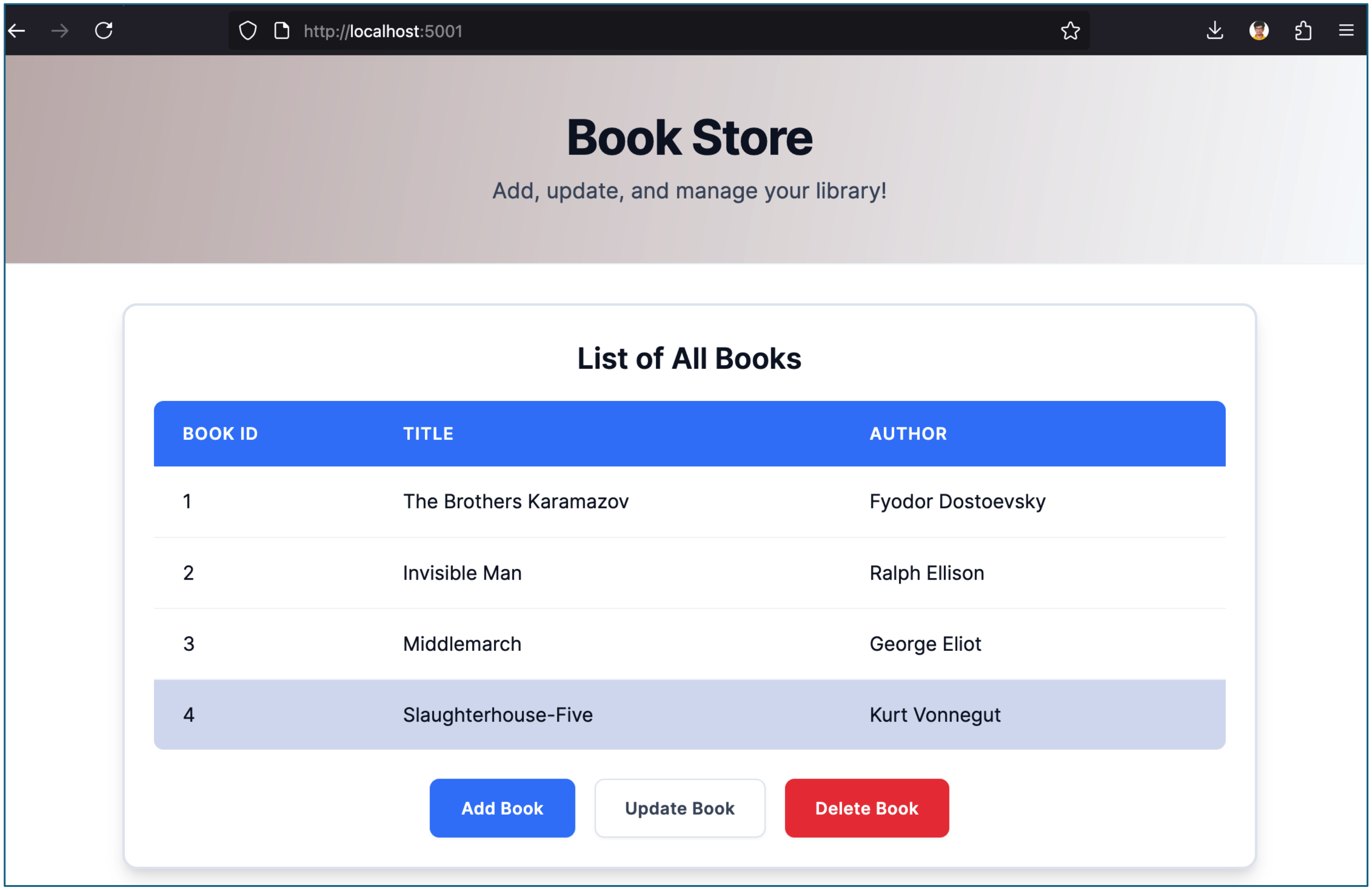
HW2 – Prajwal Dambalkar (018318196)

Part One

GitHub link: https://github.com/PrajwalDambalkar/Distributed-Learnings/tree/main/Assignments/HW2

I. HTML & CSS (4 Points) – Artist Library

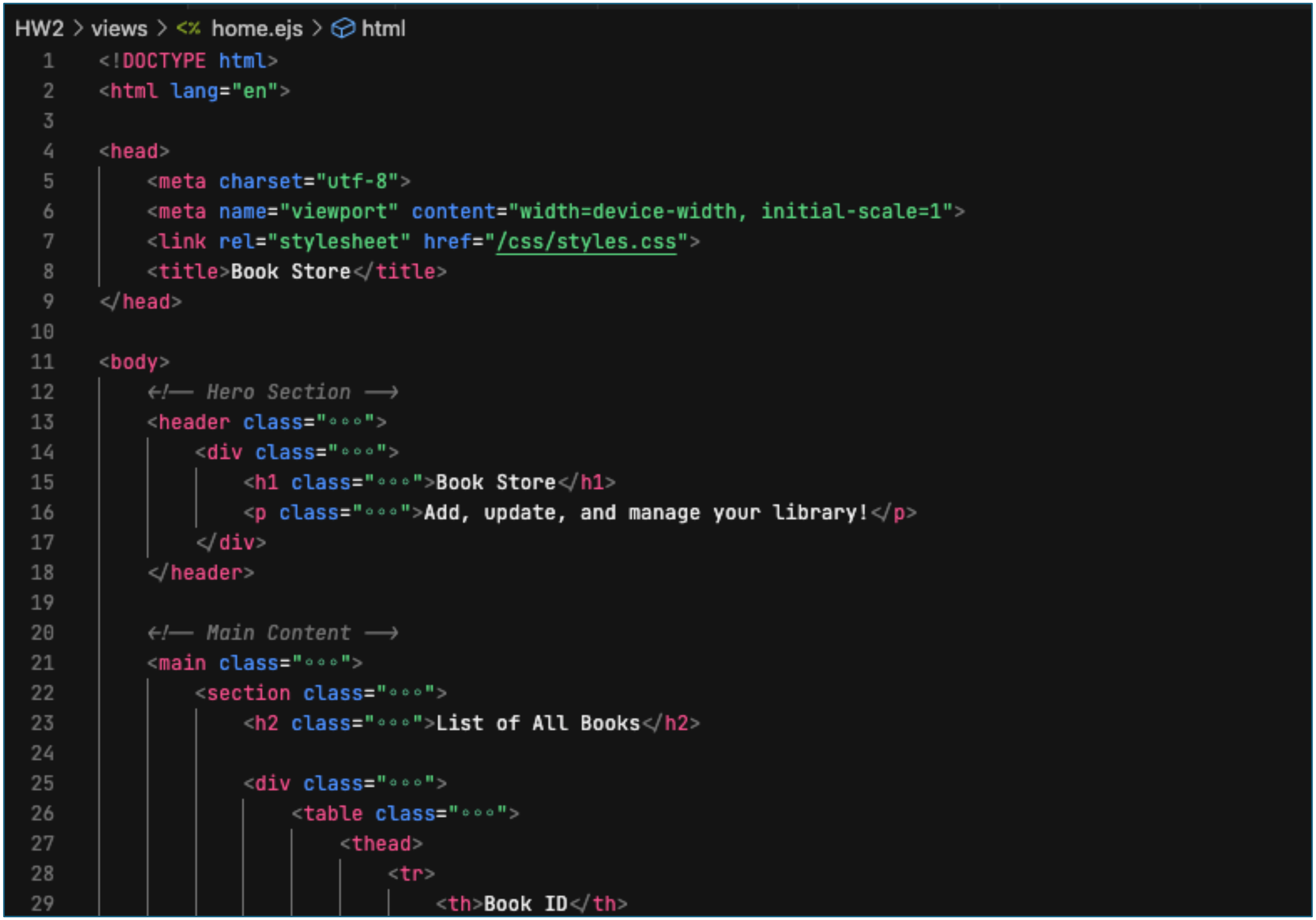


Book Store

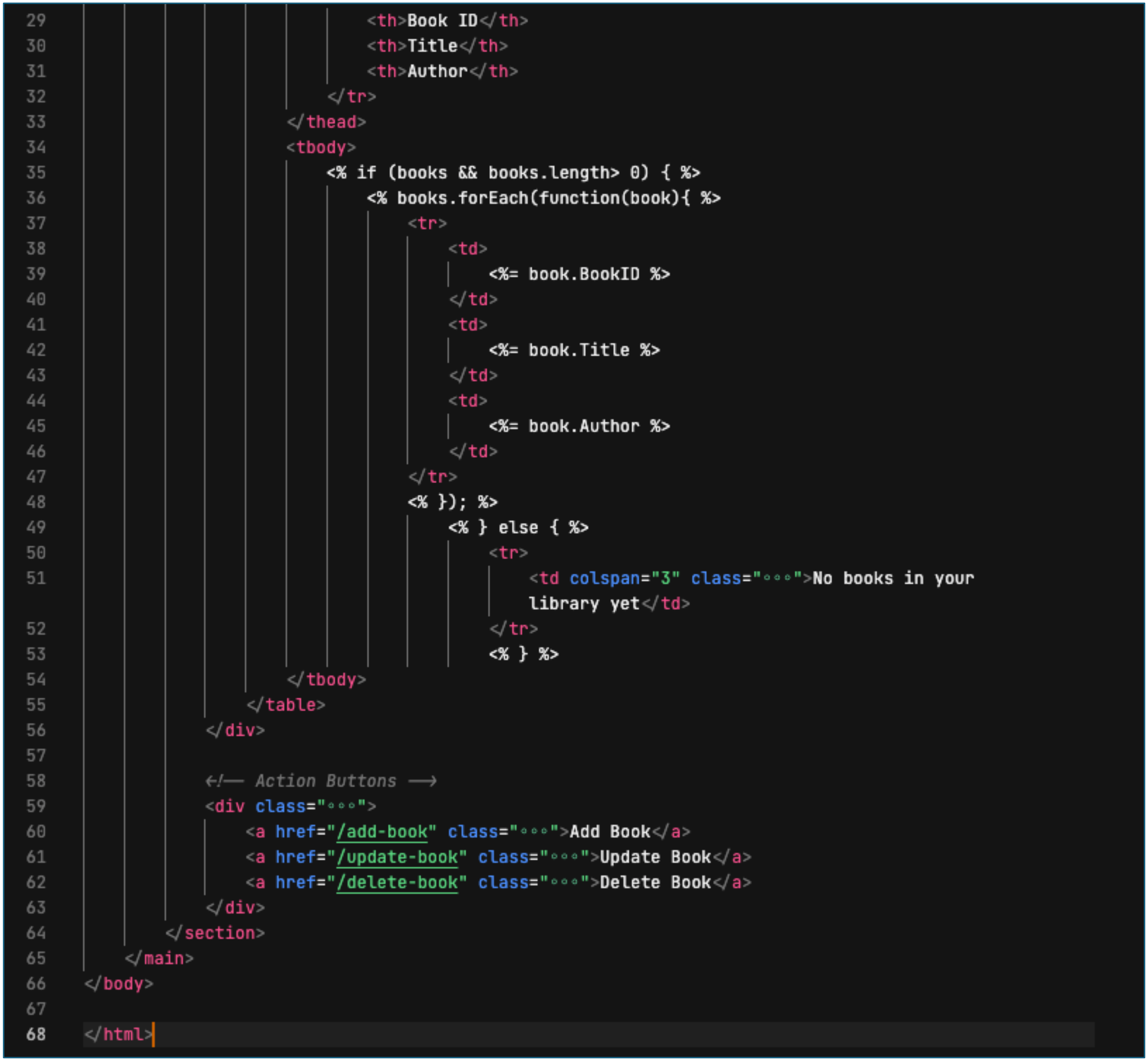
Add, update, and manage your library!

List of All Books

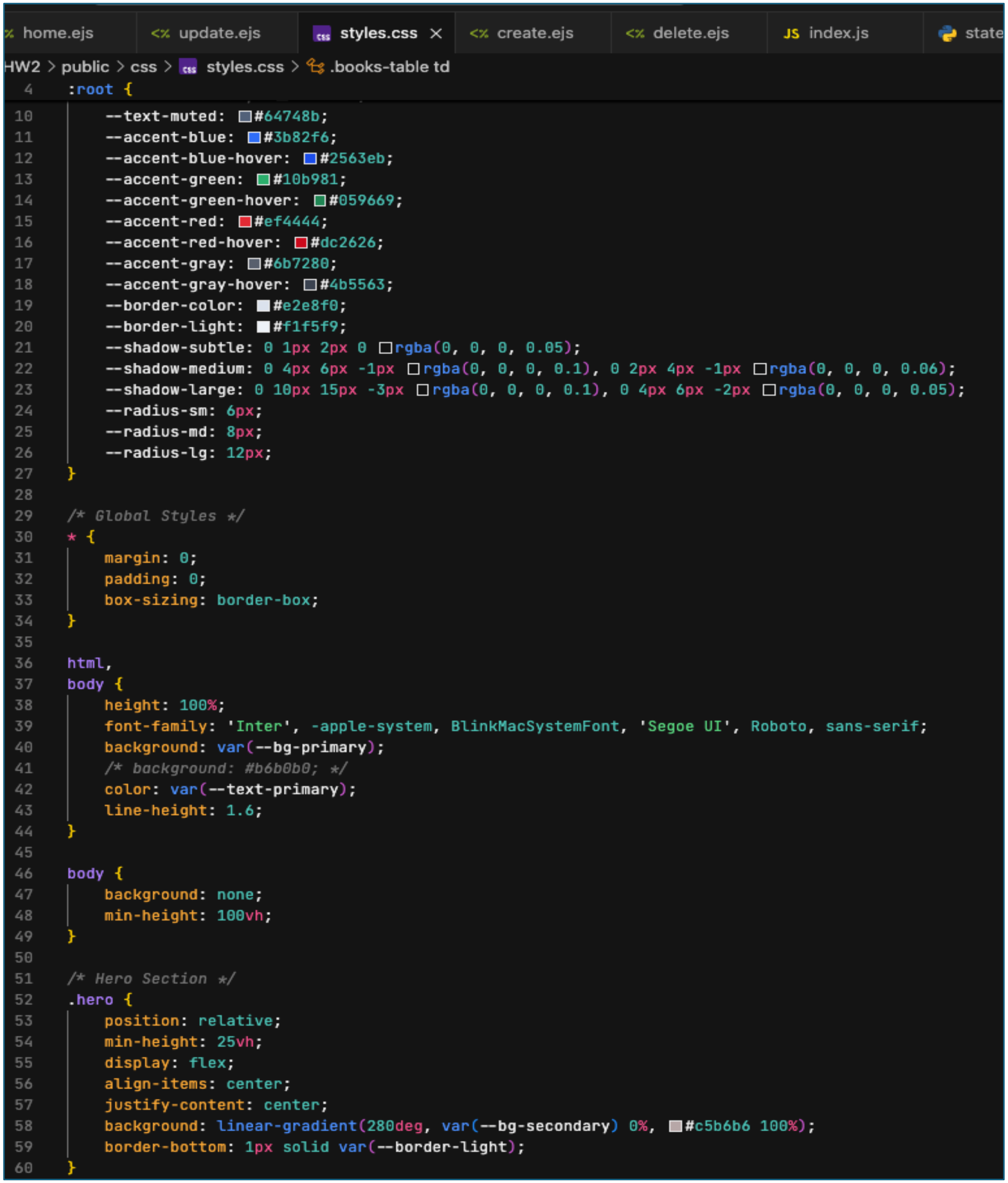
BOOK ID	TITLE	AUTHOR
1	The Brothers Karamazov	Fyodor Dostoevsky
2	Invisible Man	Ralph Ellison
3	Middlemarch	George Eliot
4	Slaughterhouse-Five	Kurt Vonnegut



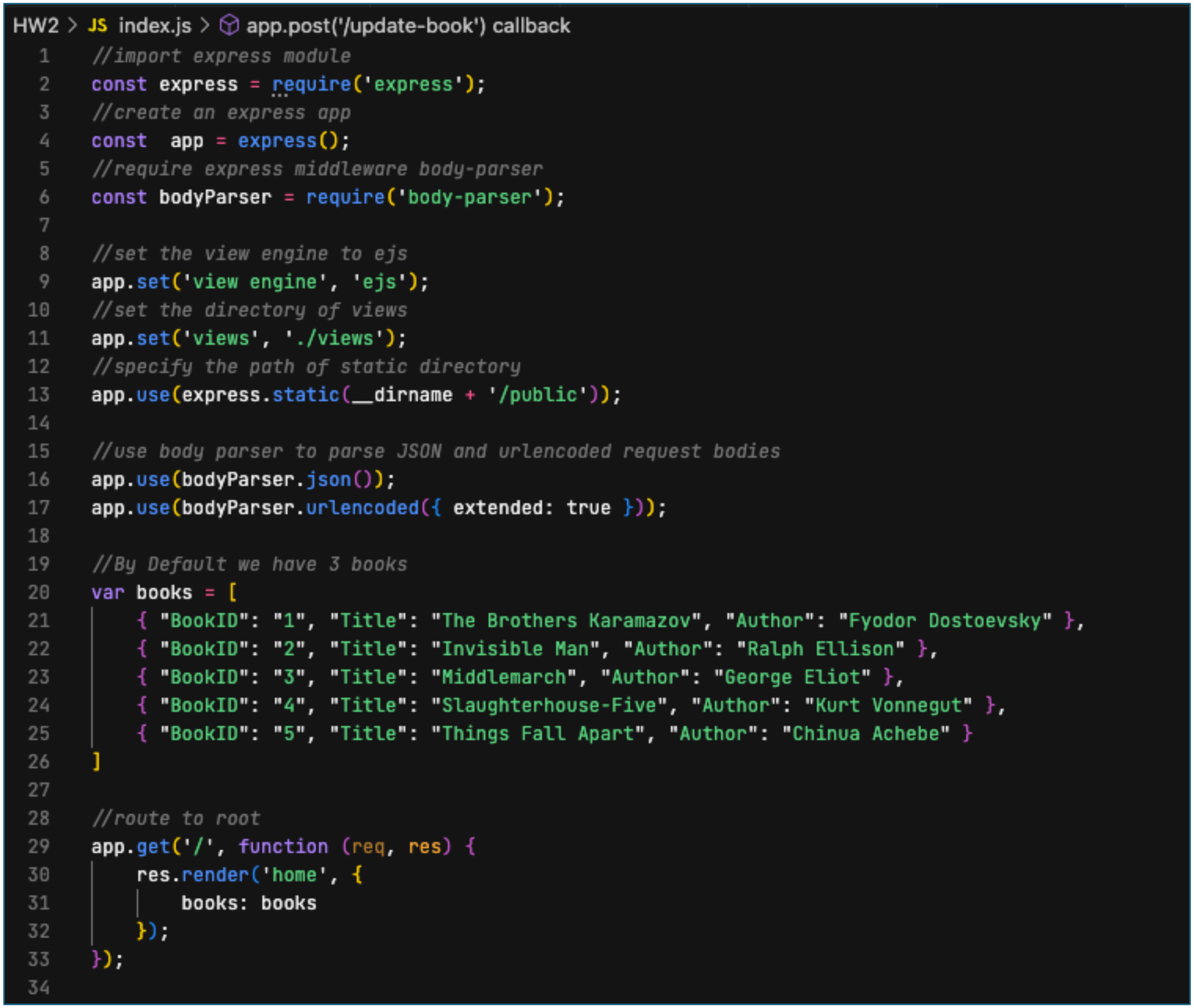
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/css/styles.css">
    <title>Book Store</title>
</head>

<body>
    <!-- Hero Section -->
    <header class="◦◦◦">
        <div class="◦◦◦">
            <h1 class="◦◦◦">Book Store</h1>
            <p class="◦◦◦">Add, update, and manage your library!</p>
        </div>
    </header>

    <!-- Main Content -->
    <main class="◦◦◦">
        <section class="◦◦◦">
            <h2 class="◦◦◦">List of All Books</h2>

            <div class="◦◦◦">
                <table class="◦◦◦">
                    <thead>
                        <tr>
                            <th>Book ID</th>
```



```
                            <th>Book ID</th>
                            <th>Title</th>
                            <th>Author</th>
                        </tr>
                    </thead>
                    <tbody>
                        <% if (books && books.length> 0) { %>
                            <% books.forEach(function(book){ %>
                                <tr>
                                    <td>
                                        <%= book.BookID %>
                                    </td>
                                    <td>
                                        <%= book.Title %>
                                    </td>
                                    <td>
                                        <%= book.Author %>
                                    </td>
                                </tr>
                                <% }); %>
                                    <% } else { %>
                                        <tr>
                                            <td colspan="3" class="◦◦◦">No books in your library yet</td>
                                        </tr>
                                        <% } %>
                    </tbody>
                </table>
            </div>

            <!-- Action Buttons -->
            <div class="◦◦◦">
                <a href="/add-book" class="◦◦◦">Add Book</a>
                <a href="/update-book" class="◦◦◦">Update Book</a>
                <a href="/delete-book" class="◦◦◦">Delete Book</a>
            </div>
        </section>
    </main>
</body>

</html>
```



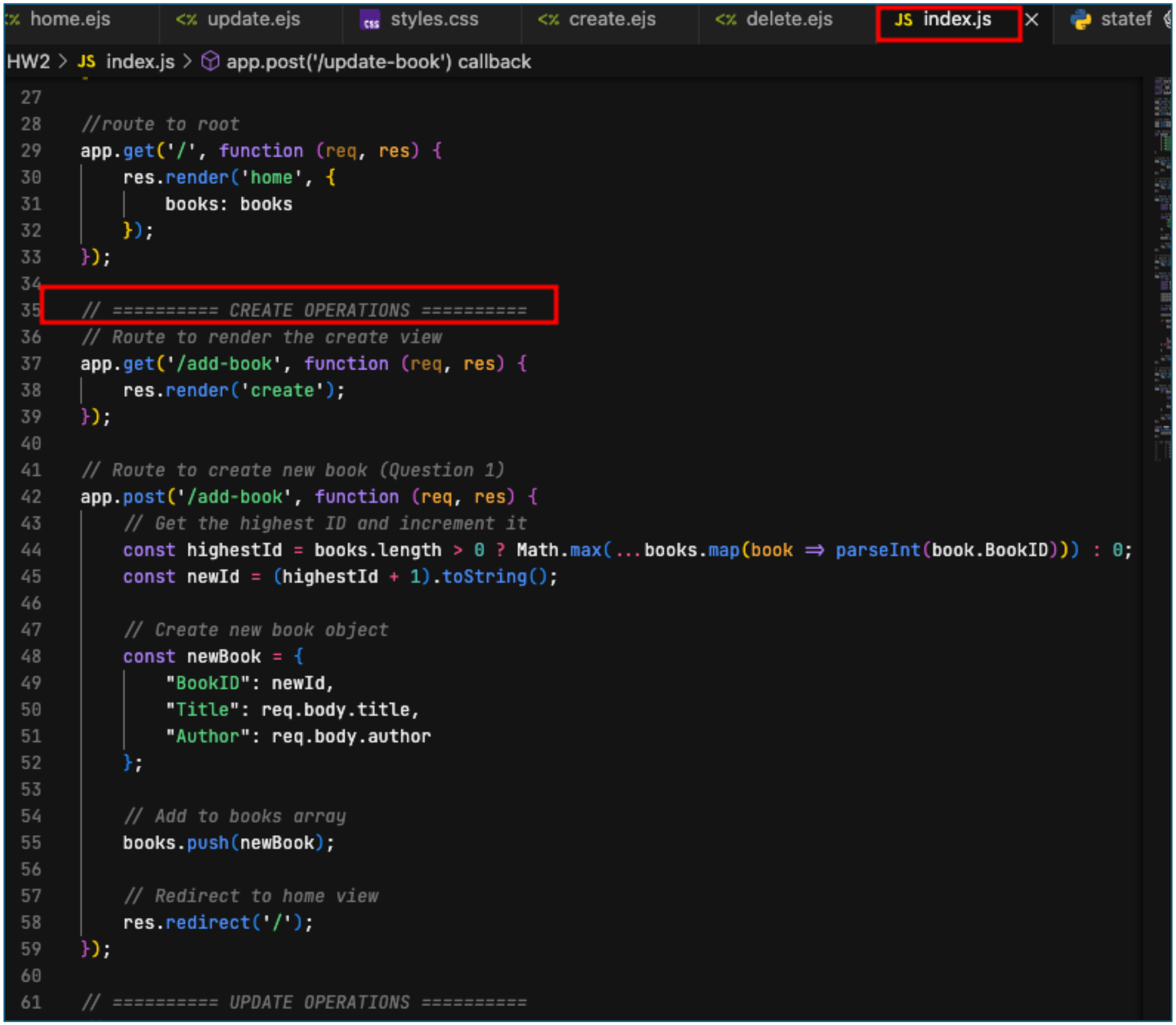
```
:root {
    --text-muted: #64748b;
    --accent-blue: #3b82f6;
    --accent-blue-hover: #2563eb;
    --accent-green: #10b981;
    --accent-green-hover: #059669;
    --accent-red: #ef4444;
    --accent-red-hover: #dc2626;
    --accent-gray: #6b7280;
    --accent-gray-hover: #4b5563;
    --border-color: #e2e8f0;
    --border-light: #f1f5f9;
    --shadow-subtle: 0 1px 2px 0 rgba(0, 0, 0, 0.05);
    --shadow-medium: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06);
    --shadow-large: 0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0, 0, 0, 0.05);
    --radius-sm: 6px;
    --radius-md: 8px;
    --radius-lg: 12px;
}

/* Global Styles */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

html,
body {
    height: 100%;
    font-family: 'Inter', -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
    background: var(--bg-primary);
    /* background: #b6b0b0; */
    color: var(--text-primary);
    line-height: 1.6;
}

body {
    background: none;
    min-height: 100vh;
}

/* Hero Section */
.hero {
    position: relative;
    min-height: 25vh;
    display: flex;
    align-items: center;
    justify-content: center;
    background: linear-gradient(280deg, var(--bg-secondary) 0%, #c5b6b6 100%);
    border-bottom: 1px solid var(--border-light);
}
```



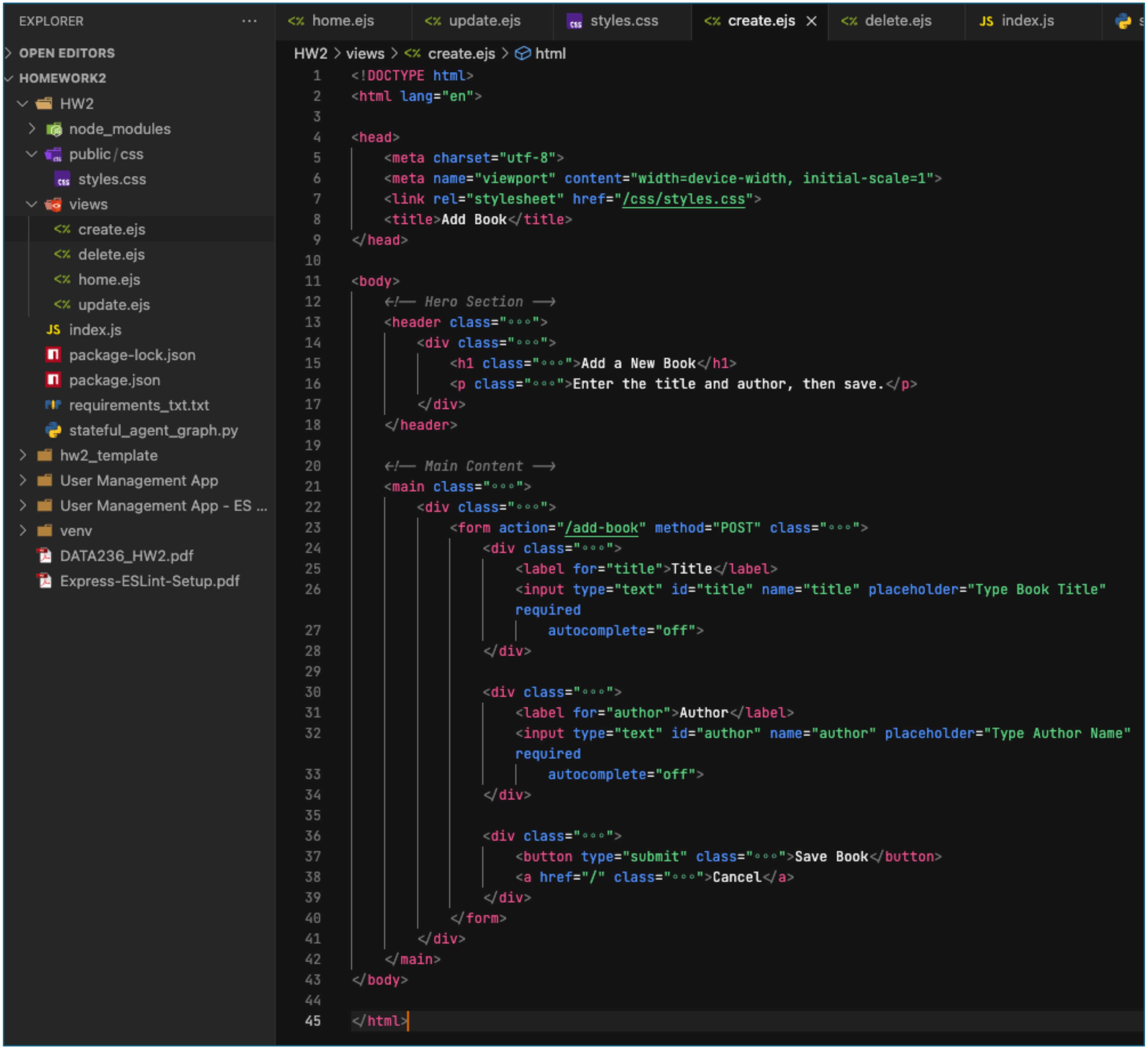
```javascript
//import express module
const express = require('express');
//create an express app
const  app = express();
//require express middleware body-parser
const bodyParser = require('body-parser');

//set the view engine to ejs
app.set('view engine', 'ejs');
//set the directory of views
app.set('views', './views');
//specify the path of static directory
app.use(express.static(__dirname + '/public'));

//use body parser to parse JSON and urlencoded request bodies
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

//By Default we have 3 books
var books = [
    { "BookID": "1", "Title": "The Brothers Karamazov", "Author": "Fyodor Dostoevsky" },
    { "BookID": "2", "Title": "Invisible Man", "Author": "Ralph Ellison" },
    { "BookID": "3", "Title": "Middlemarch", "Author": "George Eliot" },
    { "BookID": "4", "Title": "Slaughterhouse-Five", "Author": "Kurt Vonnegut" },
    { "BookID": "5", "Title": "Things Fall Apart", "Author": "Chinua Achebe" }
]

//route to root
app.get('/', function (req, res) {
    res.render('home', {
        books: books
    });
});

```

II. HTTP, Express, NodeJS (6 Points)

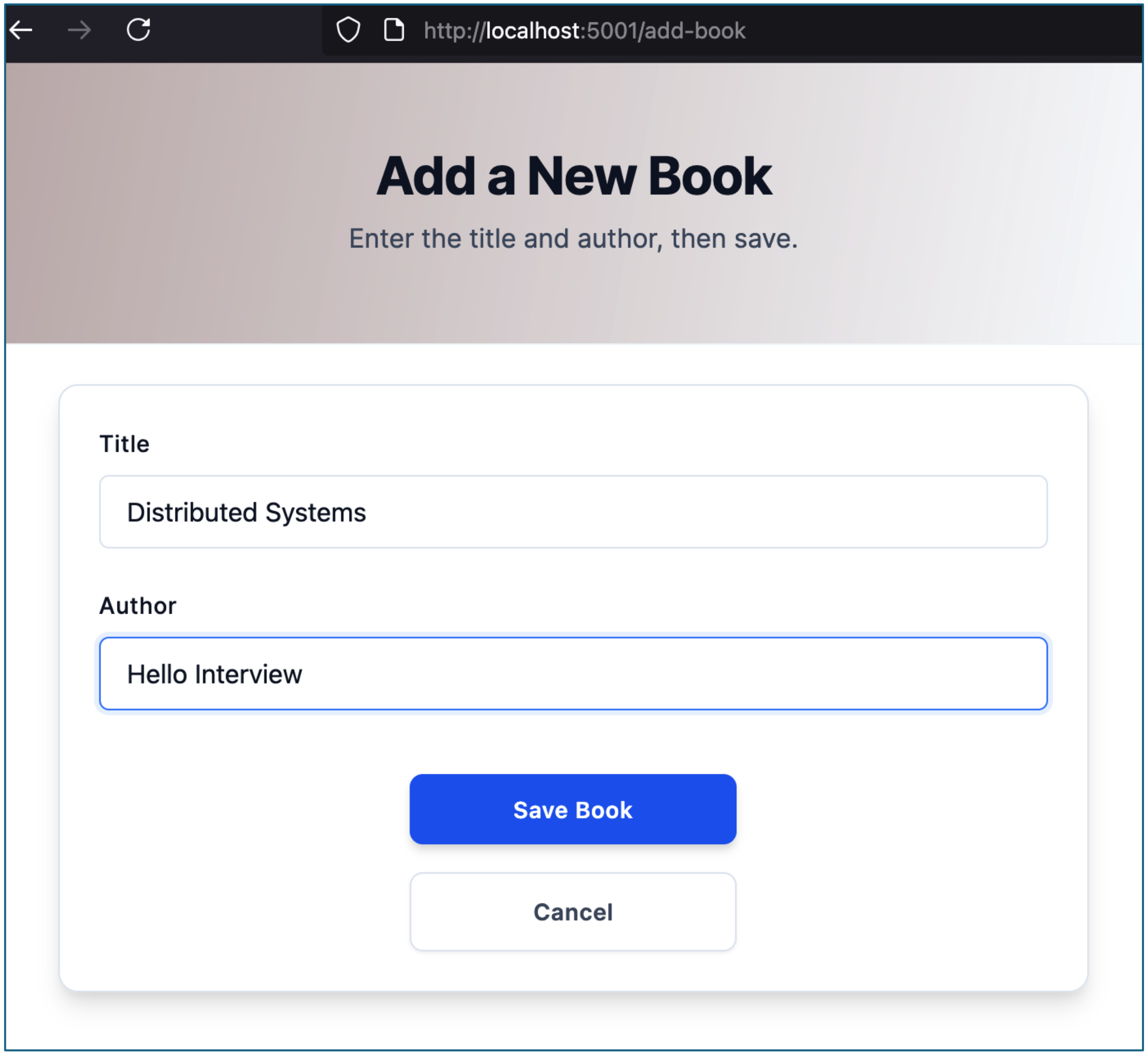
1. **Write the code to add a new book. The user should be able to enter the Book Title and Author Name. Once the user submits the required data, the book should be added, and the user should be redirected to the home view showing the updated list of books. (2 points)**



```javascript

//route to root
app.get('/', function (req, res) {
    res.render('home', {
        books: books
    });
});

// ========== CREATE OPERATIONS ==========
// Route to render the create view
app.get('/add-book', function (req, res) {
    res.render('create');
});

// Route to create new book (Question 1)
app.post('/add-book', function (req, res) {
    // Get the highest ID and increment it
    const highestId = books.length > 0 ? Math.max(...books.map(book => parseInt(book.BookID))) : 0;
    const newId = (highestId + 1).toString();

    // Create new book object
    const newBook = {
        "BookID": newId,
        "Title": req.body.title,
        "Author": req.body.author
    };

    // Add to books array
    books.push(newBook);

    // Redirect to home view
    res.redirect('/');
});

// ========== UPDATE OPERATIONS ==========
```



```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/css/styles.css">
    <title>Add Book</title>
</head>

<body>
    <!-- Hero Section -->
    <header class="◦◦◦">
        <div class="◦◦◦">
            <h1 class="◦◦◦">Add a New Book</h1>
            <p class="◦◦◦">Enter the title and author, then save.</p>
        </div>
    </header>

    <!-- Main Content -->
    <main class="◦◦◦">
        <div class="◦◦◦">
            <form action="/add-book" method="POST" class="◦◦◦">
                <div class="◦◦◦">
                    <label for="title">Title</label>
                    <input type="text" id="title" name="title" placeholder="Type Book Title" required
                        autocomplete="off">
                </div>

                <div class="◦◦◦">
                    <label for="author">Author</label>
                    <input type="text" id="author" name="author" placeholder="Type Author Name" required
                        autocomplete="off">
                </div>

                <div class="◦◦◦">
                    <button type="submit" class="◦◦◦">Save Book</button>
                    <a href="/" class="◦◦◦">Cancel</a>
                </div>
            </form>
        </div>
    </main>
</body>

</html>
```



http://localhost:5001/add-book

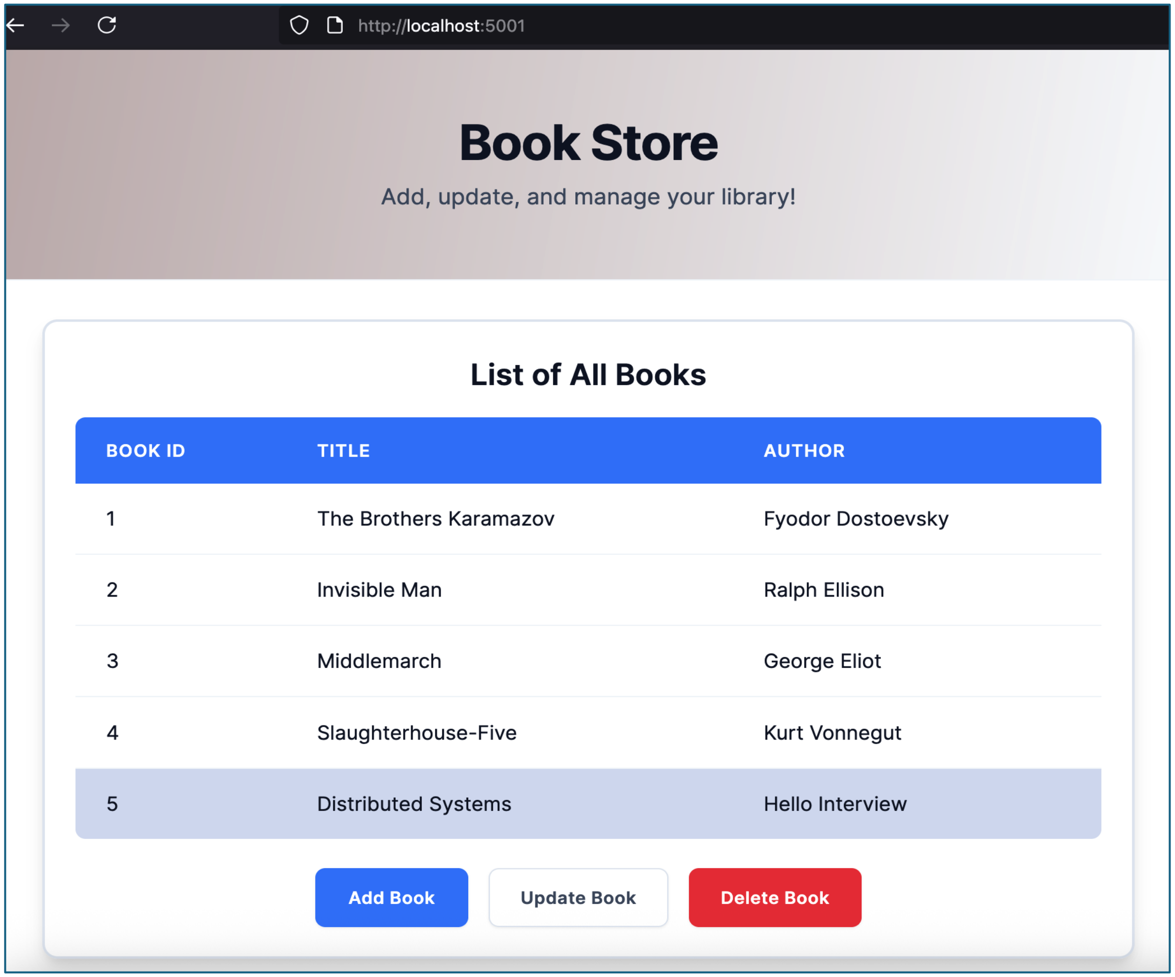
Add a New Book

Enter the title and author, then save.

Title: Distributed Systems

Author: Hello Interview

The page gets redirected to the homepage, and the list is updated, where the new book gets the latest ID.



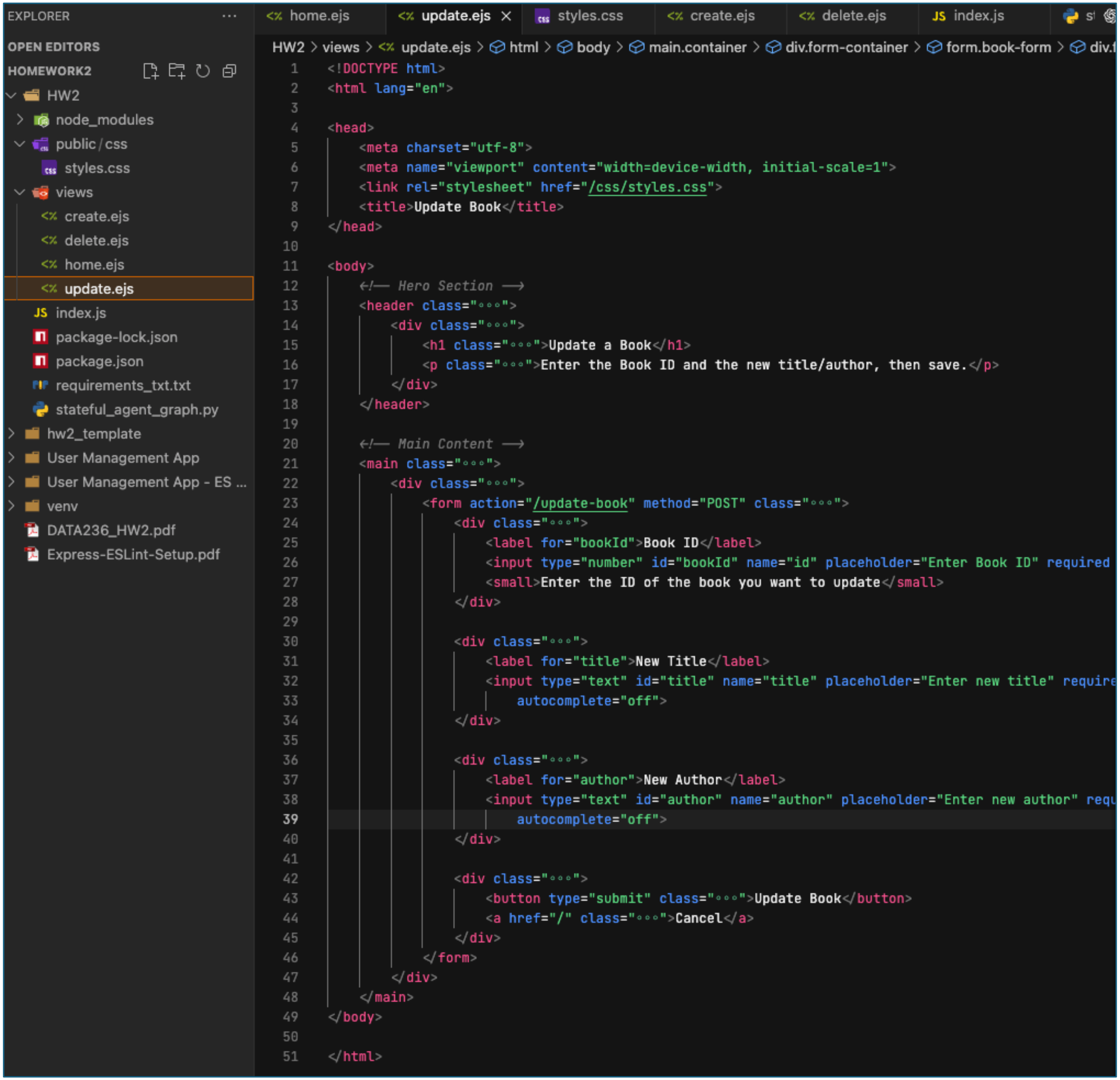
Book Store

Add, update, and manage your library!

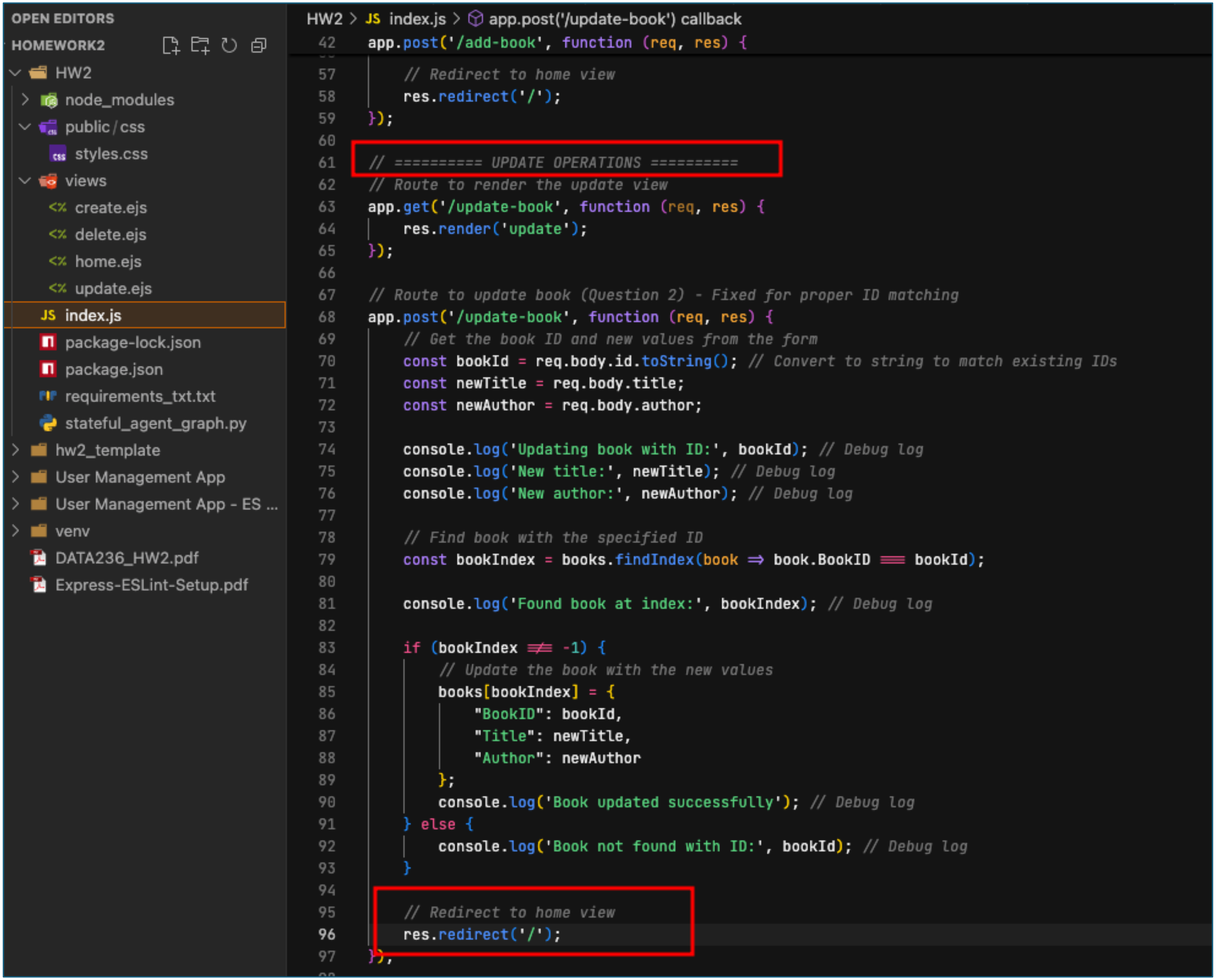
List of All Books

BOOK ID	TITLE	AUTHOR
1	The Brothers Karamazov	Fyodor Dostoevsky
2	Invisible Man	Ralph Ellison
3	Middlemarch	George Eliot
4	Slaughterhouse-Five	Kurt Vonnegut
5	Distributed Systems	Hello Interview

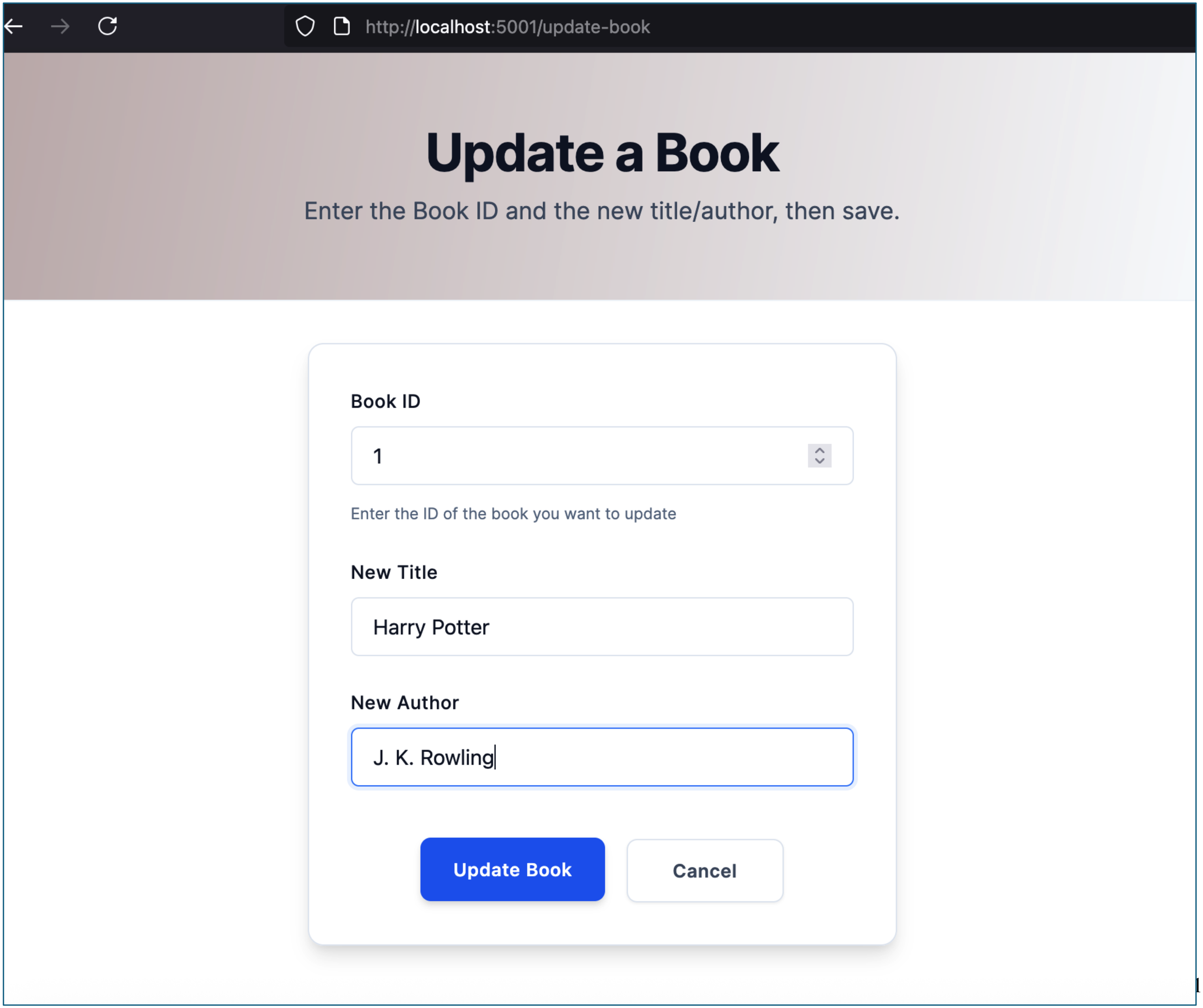
2. **Write the code to update the book with ID 1 to title:" Harry Potter", Author Name: "J.K Rowling". After submitting the data, redirect to the home view and show the updated data in the list of books. (2 points)**



```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/css/styles.css">
    <title>Update Book</title>
</head>

<body>
    <!-- Hero Section -->
    <header class="◦◦◦">
        <div class="◦◦◦">
            <h1 class="◦◦◦">Update a Book</h1>
            <p class="◦◦◦">Enter the Book ID and the new title/author, then save.</p>
        </div>
    </header>

    <!-- Main Content -->
    <main class="◦◦◦">
        <div class="◦◦◦">
            <form action="/update-book" method="POST" class="◦◦◦">
                <div class="◦◦◦">
                    <label for="bookId">Book ID</label>
                    <input type="number" id="bookId" name="id" placeholder="Enter Book ID" required
                    <small>Enter the ID of the book you want to update</small>
                </div>

                <div class="◦◦◦">
                    <label for="title">New Title</label>
                    <input type="text" id="title" name="title" placeholder="Enter new title" require
                        autocomplete="off">
                </div>

                <div class="◦◦◦">
                    <label for="author">New Author</label>
                    <input type="text" id="author" name="author" placeholder="Enter new author" requ
                        autocomplete="off">
                </div>

                <div class="◦◦◦">
                    <button type="submit" class="◦◦◦">Update Book</button>
                    <a href="/" class="◦◦◦">Cancel</a>
                </div>
            </form>
        </div>
    </main>
</body>

</html>
```



```
app.post('/add-book', function (req, res) {
    // Redirect to home view
    res.redirect('/');
});

// ========== UPDATE OPERATIONS ==========
// Route to render the update view
app.get('/update-book', function (req, res) {
    res.render('update');
});

// Route to update book (Question 2) - Fixed for proper ID matching
app.post('/update-book', function (req, res) {
    // Get the book ID and new values from the form
    const bookId = req.body.id.toString(); // Convert to string to match existing IDs
    const newTitle = req.body.title;
    const newAuthor = req.body.author;

    console.log('Updating book with ID:', bookId); // Debug log
    console.log('New title:', newTitle); // Debug log
    console.log('New author:', newAuthor); // Debug log

    // Find book with the specified ID
    const bookIndex = books.findIndex(book => book.BookID === bookId);

    console.log('Found book at index:', bookIndex); // Debug log

    if (bookIndex !== -1) {
        // Update the book with the new values
        books[bookIndex] = {
            "BookID": bookId,
            "Title": newTitle,
            "Author": newAuthor
        };
        console.log('Book updated successfully'); // Debug log
    } else {
        console.log('Book not found with ID:', bookId); // Debug log
    }

    // Redirect to home view
    res.redirect('/');
});
```



http://localhost:5001/update-book

Update a Book

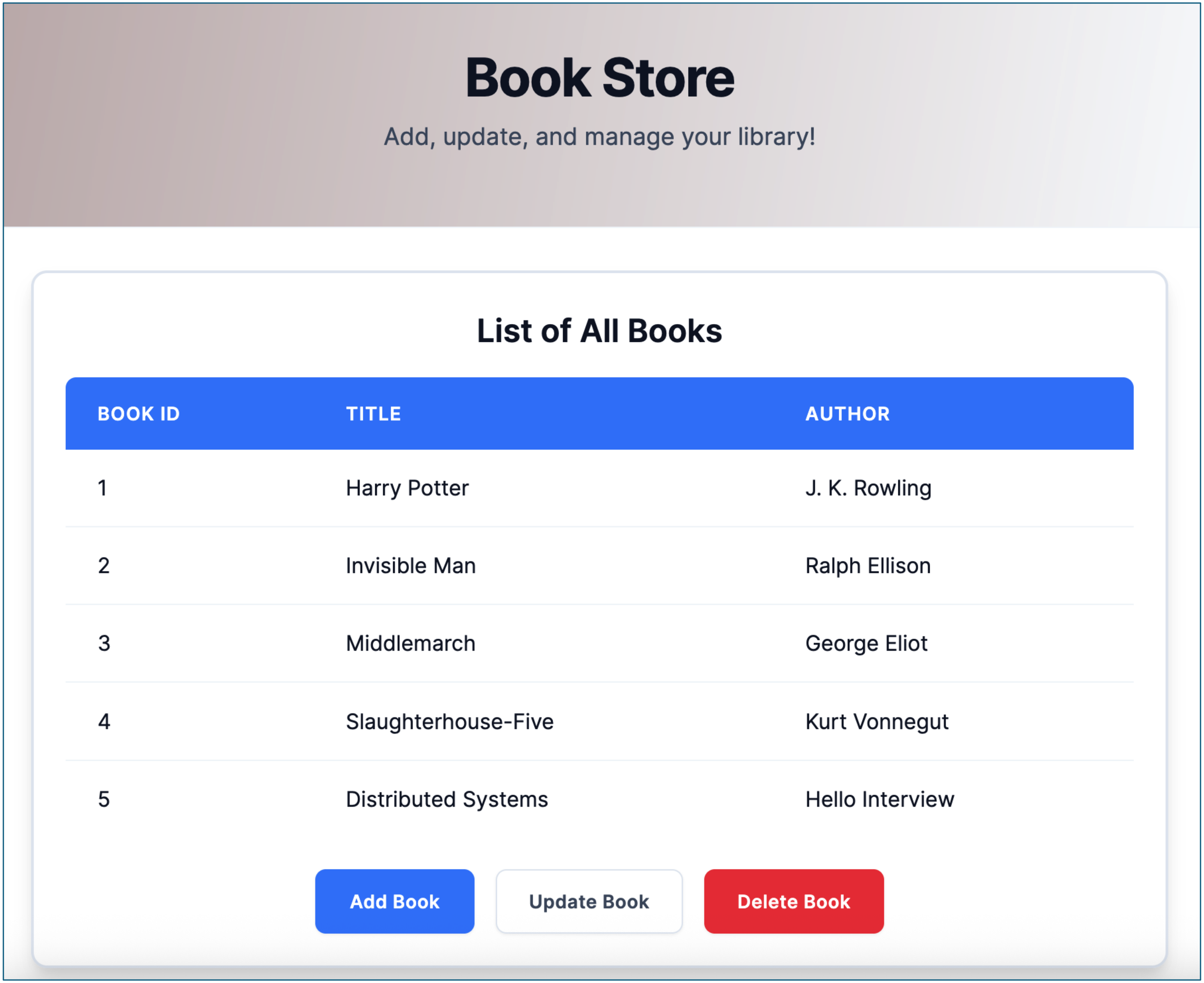
Enter the Book ID and the new title/author, then save.

Book ID: 1

Enter the ID of the book you want to update

New Title: Harry Potter

New Author: J. K. Rowling



Book Store

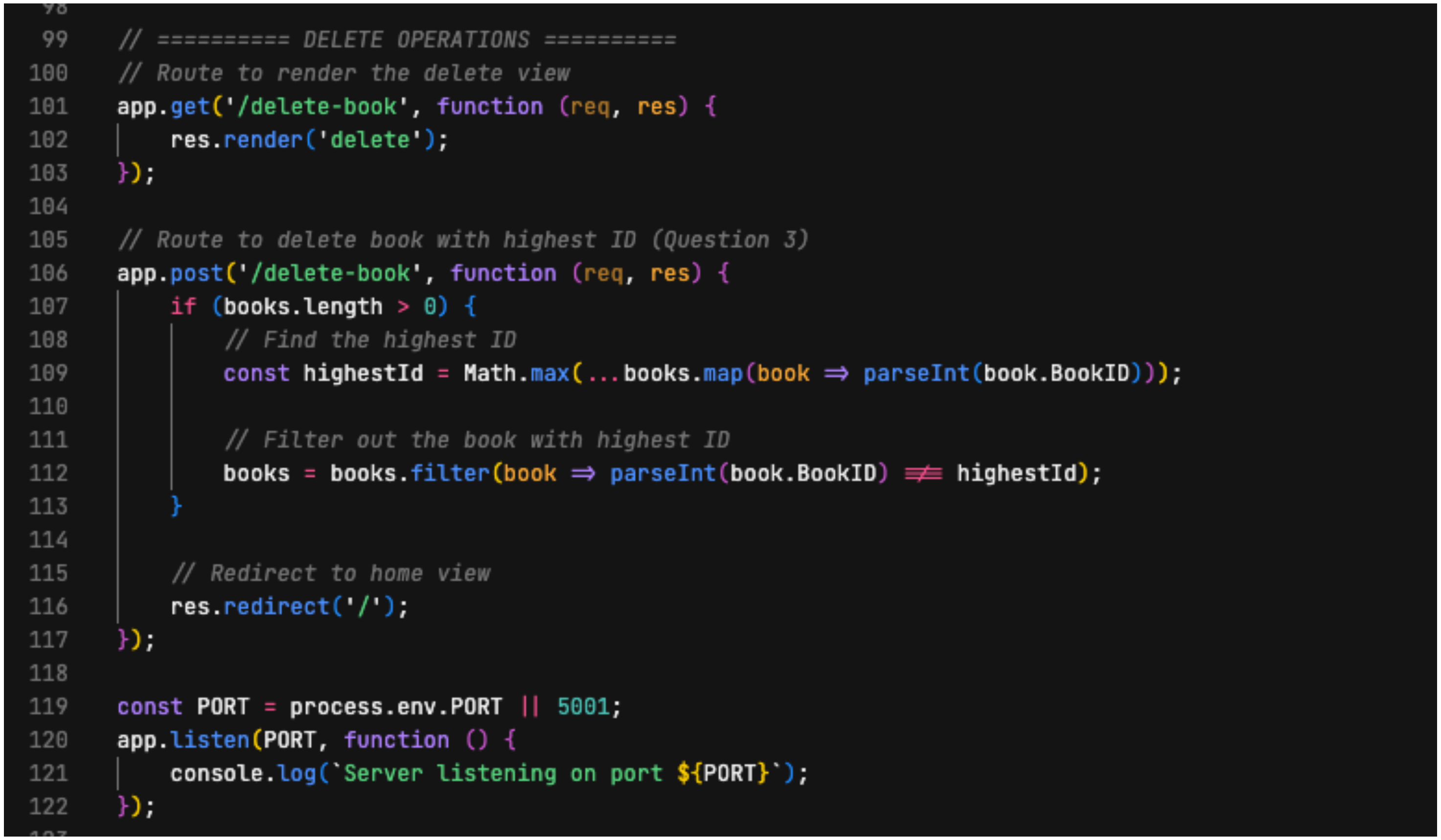
Add, update, and manage your library!

List of All Books

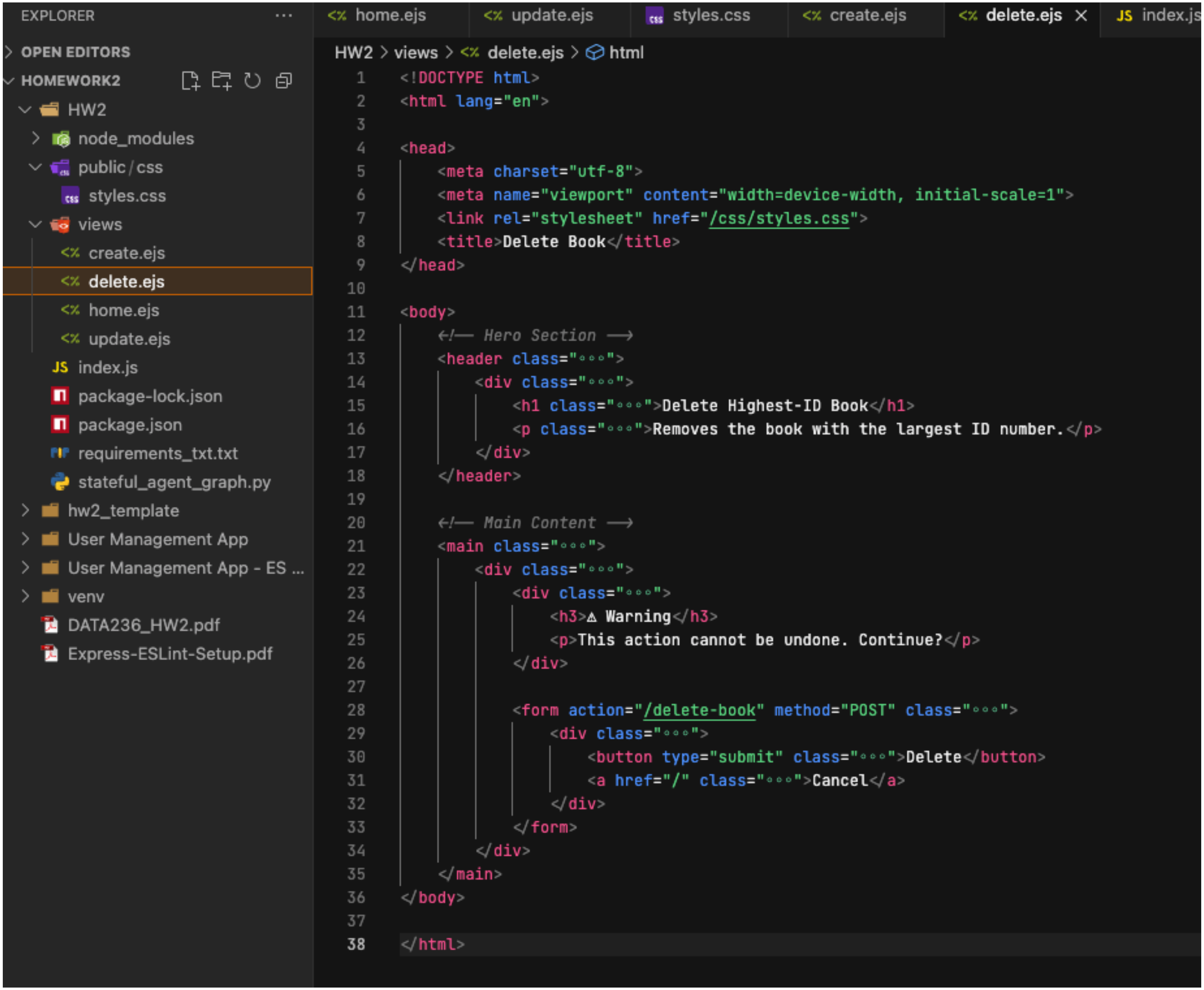
BOOK ID	TITLE	AUTHOR
1	Harry Potter	J. K. Rowling
2	Invisible Man	Ralph Ellison
3	Middlemarch	George Eliot
4	Slaughterhouse-Five	Kurt Vonnegut
5	Distributed Systems	Hello Interview

3. **Write the code to delete the book with the highest ID. After submitting the data, redirect to the home view and show the updated data in the list of books. (2 points)**

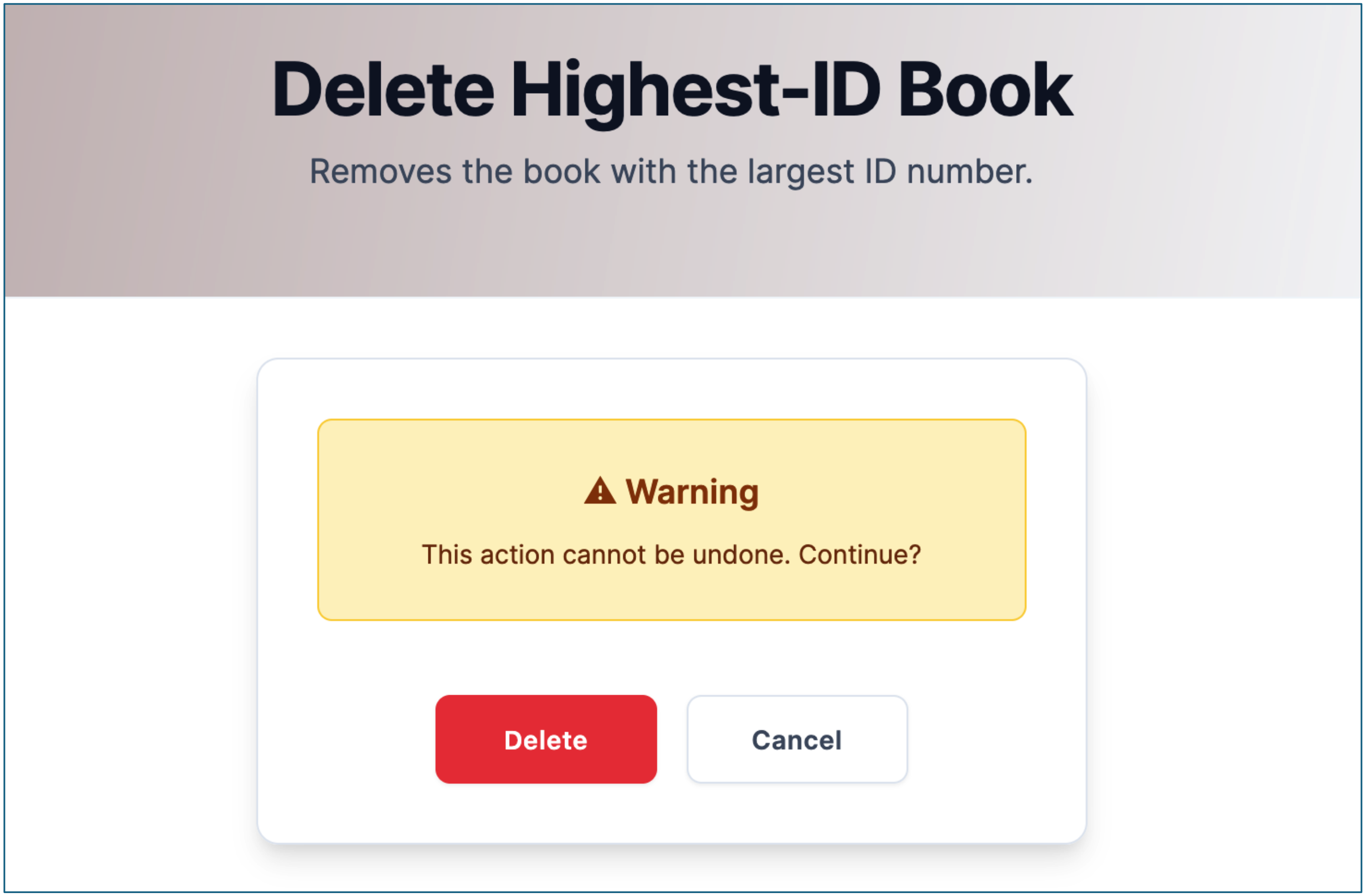
The question was slightly confusing, so I wrote the code to delete the record with the highest ID. Could've written code to delete any ID, too, but it was not mentioned.



```javascript
// ========== DELETE OPERATIONS ==========
// Route to render the delete view
app.get('/delete-book', function (req, res) {
    res.render('delete');
});

// Route to delete book with highest ID (Question 3)
app.post('/delete-book', function (req, res) {
    if (books.length > 0) {
        // Find the highest ID
        const highestId = Math.max(...books.map(book => parseInt(book.BookID)));

        // Filter out the book with highest ID
        books = books.filter(book => parseInt(book.BookID) !== highestId);
    }

    // Redirect to home view
    res.redirect('/');
});

const PORT = process.env.PORT || 5001;
app.listen(PORT, function () {
    console.log(`Server listening on port ${PORT}`);
});
```



```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/css/styles.css">
    <title>Delete Book</title>
</head>

<body>
    <!-- Hero Section -->
    <header class="◦◦◦">
        <div class="◦◦◦">
            <h1 class="◦◦◦">Delete Highest-ID Book</h1>
            <p class="◦◦◦">Removes the book with the largest ID number.</p>
        </div>
    </header>

    <!-- Main Content -->
    <main class="◦◦◦">
        <div class="◦◦◦">
            <div class="◦◦◦">
                <h3>⚠ Warning</h3>
                <p>This action cannot be undone. Continue?</p>
            </div>

            <form action="/delete-book" method="POST" class="◦◦◦">
                <div class="◦◦◦">
                    <button type="submit" class="◦◦◦">Delete</button>
                    <a href="/" class="◦◦◦">Cancel</a>
                </div>
            </form>
        </div>
    </main>
</body>

</html>
```

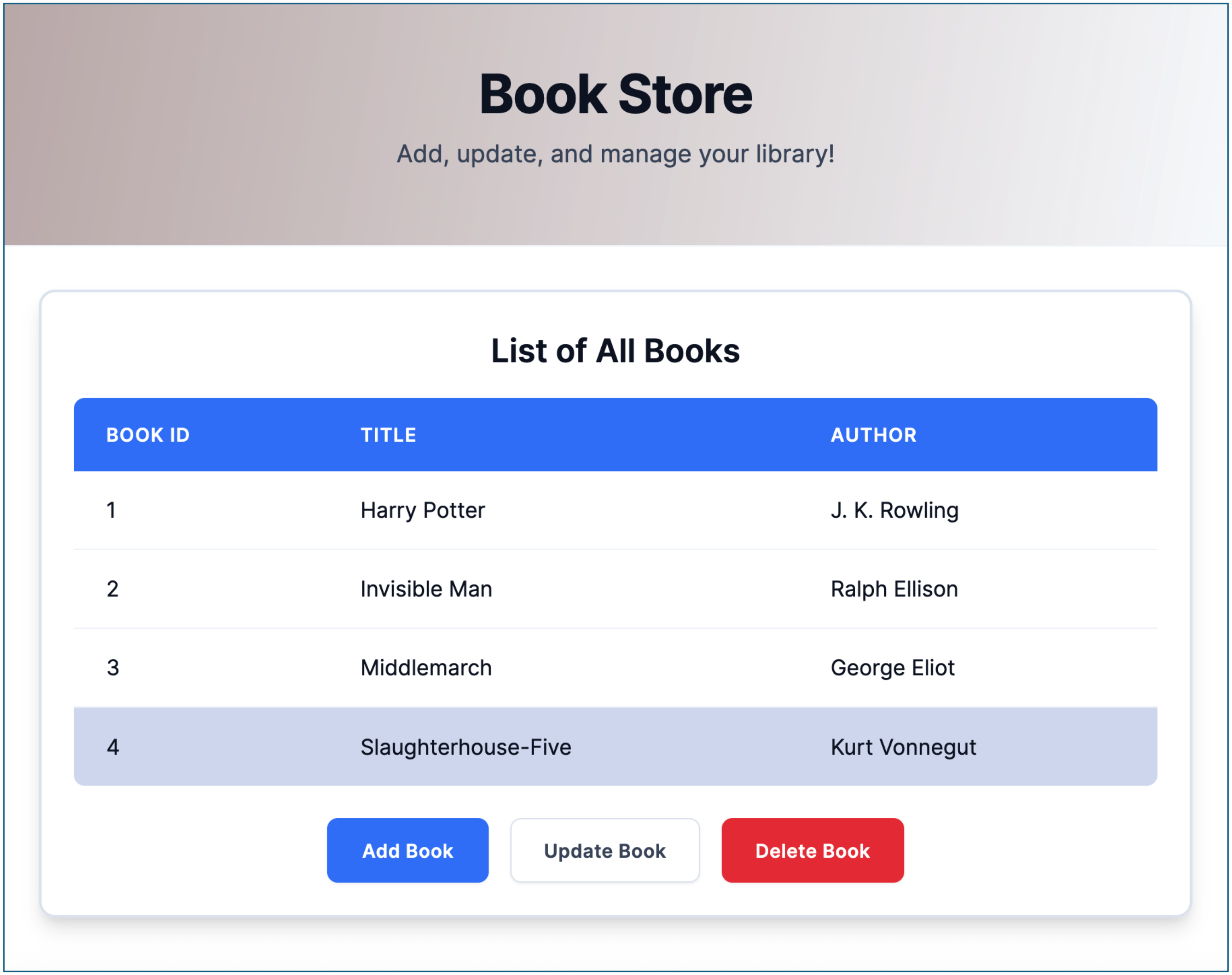


Delete Highest-ID Book

Removes the book with the largest ID number.

⚠ Warning

This action cannot be undone. Continue?



Book Store

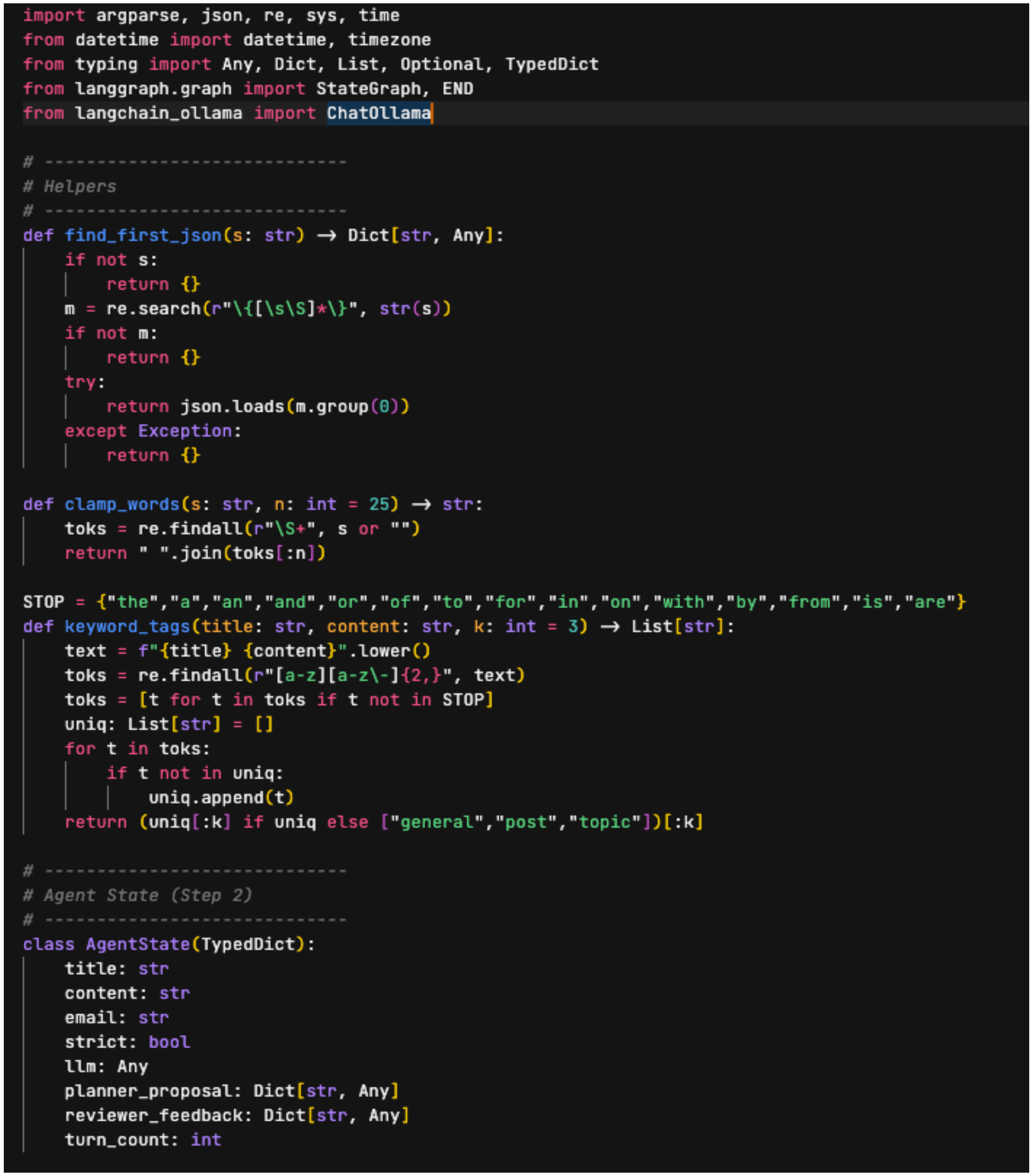
Add, update, and manage your library!

List of All Books

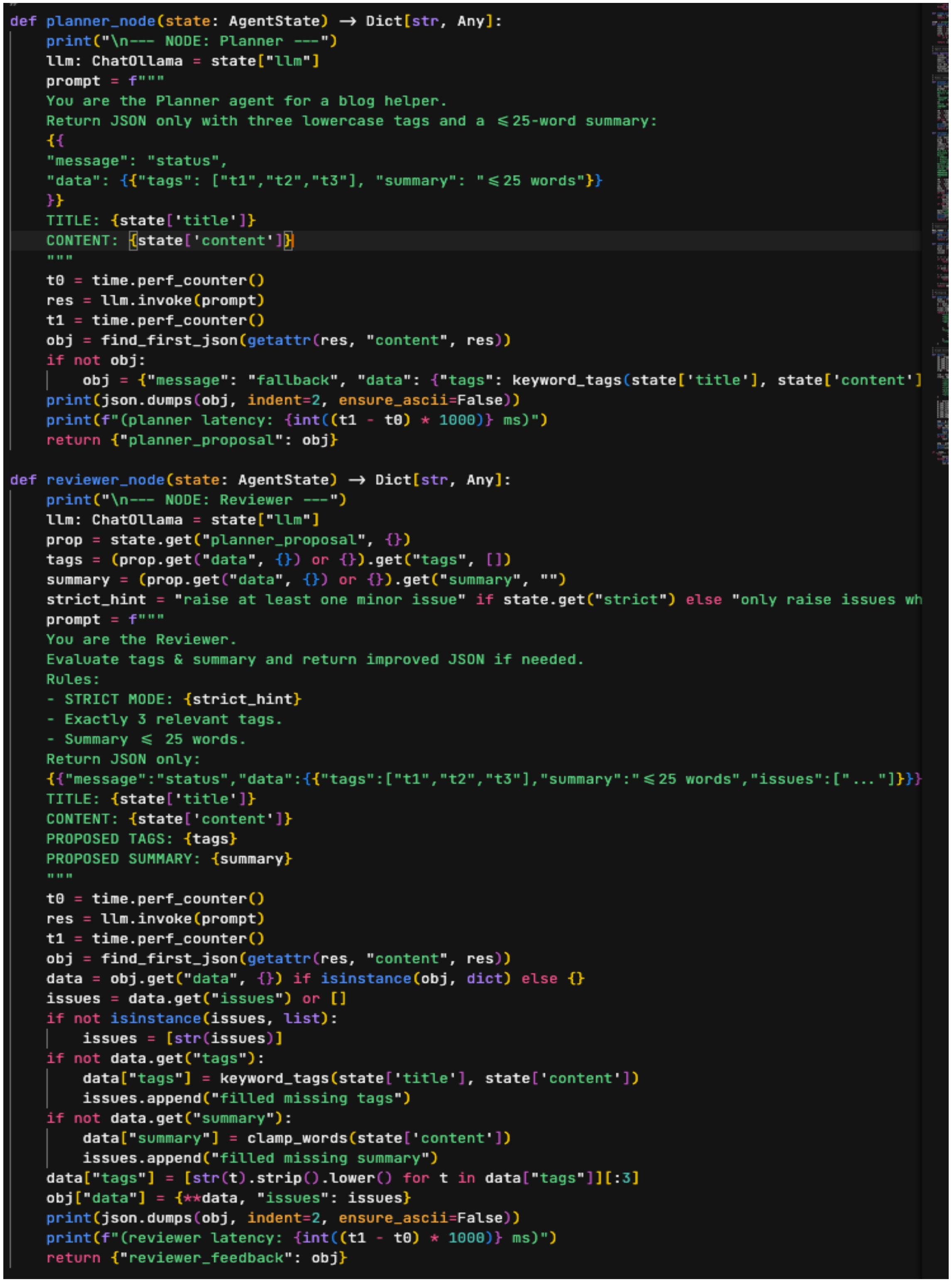
BOOK ID	TITLE	AUTHOR
1	Harry Potter	J. K. Rowling
2	Invisible Man	Ralph Ellison
3	Middlemarch	George Eliot
4	Slaughterhouse-Five	Kurt Vonnegut

Part Two: Agentic AI

Step 1: Understanding the Core Concepts
Step 2: Setting up the State

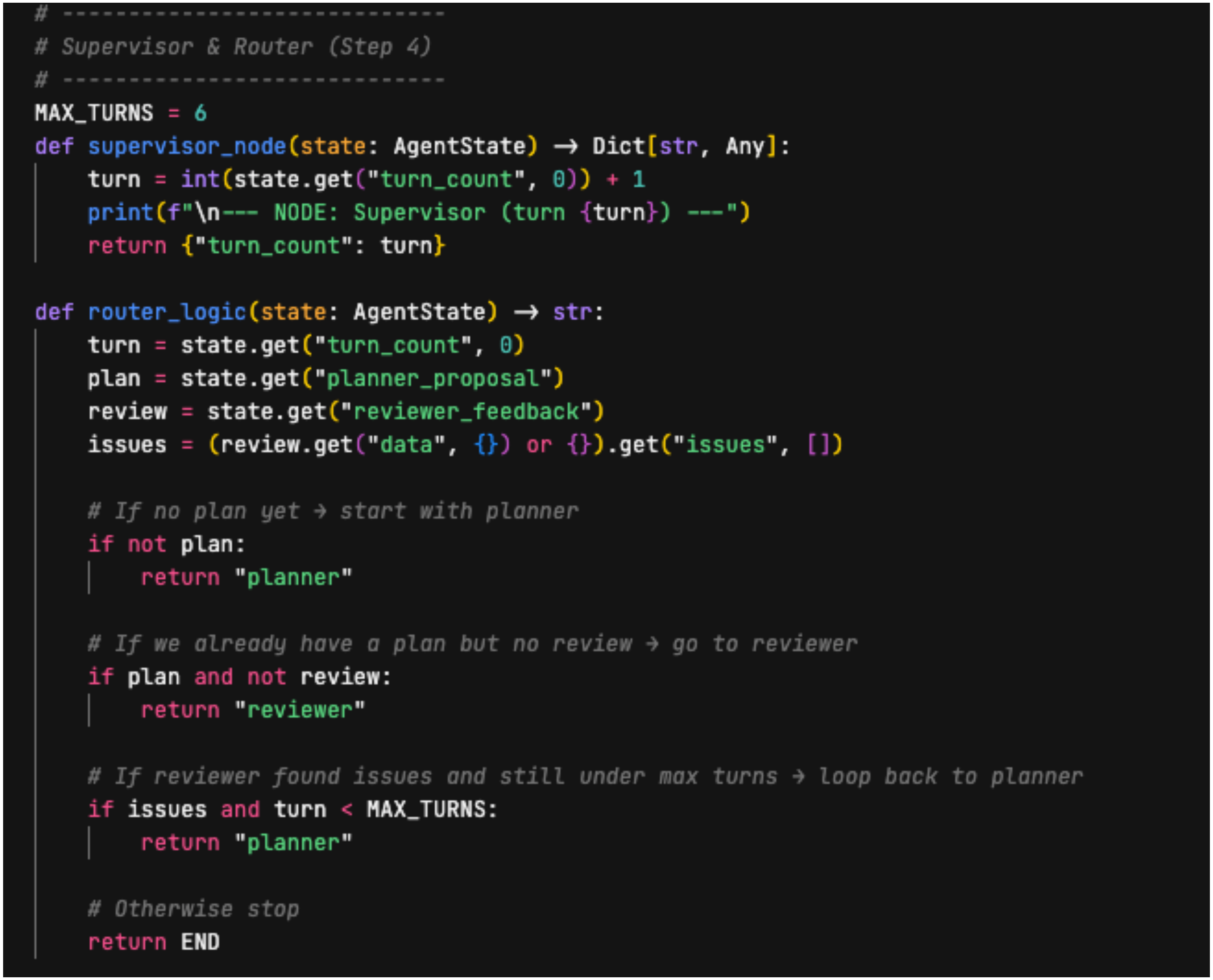


```python
import argparse, json, re, sys, time
from datetime import datetime, timezone
from typing import Any, Dict, List, Optional, TypedDict
from langgraph.graph import StateGraph, END
from langchain_ollama import ChatOllama

# ----------------------------
# Helpers
# ----------------------------
def find_first_json(s: str) -> Dict[str, Any]:
    if not s:
        return {}
    m = re.search(r"\{[\s\S]*\}", str(s))
    if not m:
        return {}
    try:
        return json.loads(m.group(0))
    except Exception:
        return {}

def clamp_words(s: str, n: int = 25) -> str:
    toks = re.findall(r"\S+", s or "")
    return " ".join(toks[:n])

STOP = {"the","a","an","and","or","of","to","for","in","on","with","by","from","is","are"}
def keyword_tags(title: str, content: str, k: int = 3) -> List[str]:
    text = f"{title} {content}".lower()
    toks = re.findall(r"[a-z][a-z\-]{2,}", text)
    toks = [t for t in toks if t not in STOP]
    uniq: List[str] = []
    for t in toks:
        if t not in uniq:
            uniq.append(t)
    return (uniq[:k] if uniq else ["general","post","topic"])[:k]

# ----------------------------
# Agent State (Step 2)
# ----------------------------
class AgentState(TypedDict):
    title: str
    content: str
    email: str
    strict: bool
    llm: Any
    planner_proposal: Dict[str, Any]
    reviewer_feedback: Dict[str, Any]
    turn_count: int
```

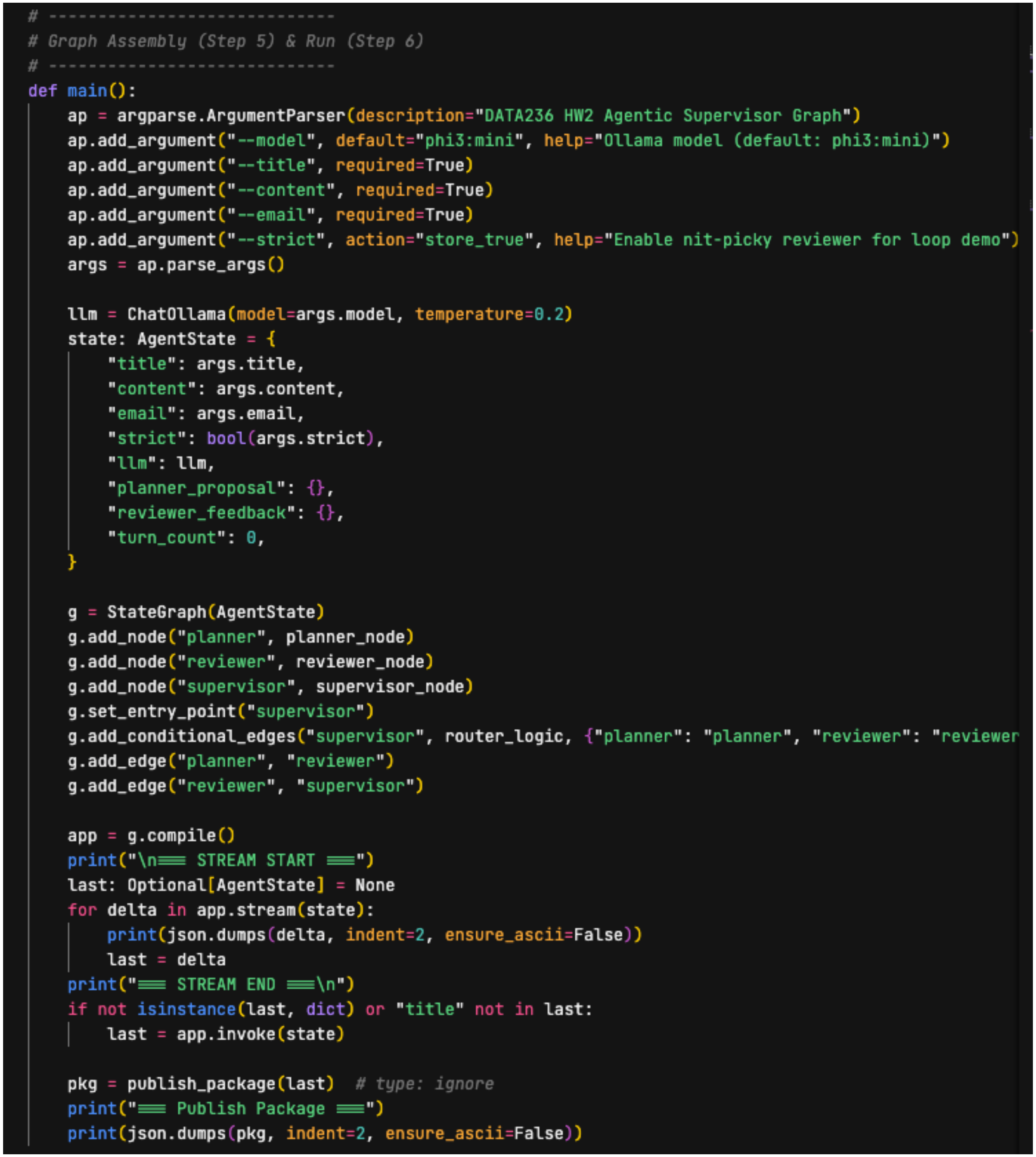


```python
def planner_node(state: AgentState) → Dict[str, Any]:
    print("\n--- NODE: Planner ---")
    llm: ChatOllama = state["llm"]
    prompt = f"""
    You are the Planner agent for a blog helper.
    Return JSON only with three lowercase tags and a ≤25-word summary:
    {{
    "message": "status",
    "data": {{"tags": ["t1","t2","t3"], "summary": "≤25 words"}}
    }}
    TITLE: {state['title']}
    CONTENT: {state['content']}
    """
    t0 = time.perf_counter()
    res = llm.invoke(prompt)
    t1 = time.perf_counter()
    obj = find_first_json(getattr(res, "content", res))
    if not obj:
        obj = {"message": "fallback", "data": {"tags": keyword_tags(state['title'], state['content']
    print(json.dumps(obj, indent=2, ensure_ascii=False))
    print(f"(planner latency: {int((t1 - t0) * 1000)} ms)")
    return {"planner_proposal": obj}

def reviewer_node(state: AgentState) → Dict[str, Any]:
    print("\n--- NODE: Reviewer ---")
    llm: ChatOllama = state["llm"]
    prop = state.get("planner_proposal", {})
    tags = (prop.get("data", {}) or {}).get("tags", [])
    summary = (prop.get("data", {}) or {}).get("summary", "")
    strict_hint = "raise at least one minor issue" if state.get("strict") else "only raise issues wh
    prompt = f"""
    You are the Reviewer.
    Evaluate tags & summary and return improved JSON if needed.
    Rules:
    - STRICT MODE: {strict_hint}
    - Exactly 3 relevant tags.
    - Summary ≤ 25 words.
    Return JSON only:
    {{"message":"status","data":{{"tags":["t1","t2","t3"],"summary":"≤25 words","issues":["..."]}}}}
    TITLE: {state['title']}
    CONTENT: {state['content']}
    PROPOSED TAGS: {tags}
    PROPOSED SUMMARY: {summary}
    """
    t0 = time.perf_counter()
    res = llm.invoke(prompt)
    t1 = time.perf_counter()
    obj = find_first_json(getattr(res, "content", res))
    data = obj.get("data", {}) if isinstance(obj, dict) else {}
    issues = data.get("issues") or []
    if not isinstance(issues, list):
        issues = [str(issues)]
    if not data.get("tags"):
        data["tags"] = keyword_tags(state['title'], state['content'])
        issues.append("filled missing tags")
    if not data.get("summary"):
        data["summary"] = clamp_words(state['content'])
        issues.append("filled missing summary")
    data["tags"] = [str(t).strip().lower() for t in data["tags"]][:3]
    obj["data"] = {**data, "issues": issues}
    print(json.dumps(obj, indent=2, ensure_ascii=False))
    print(f"(reviewer latency: {int((t1 - t0) * 1000)} ms)")
    return {"reviewer_feedback": obj}
```

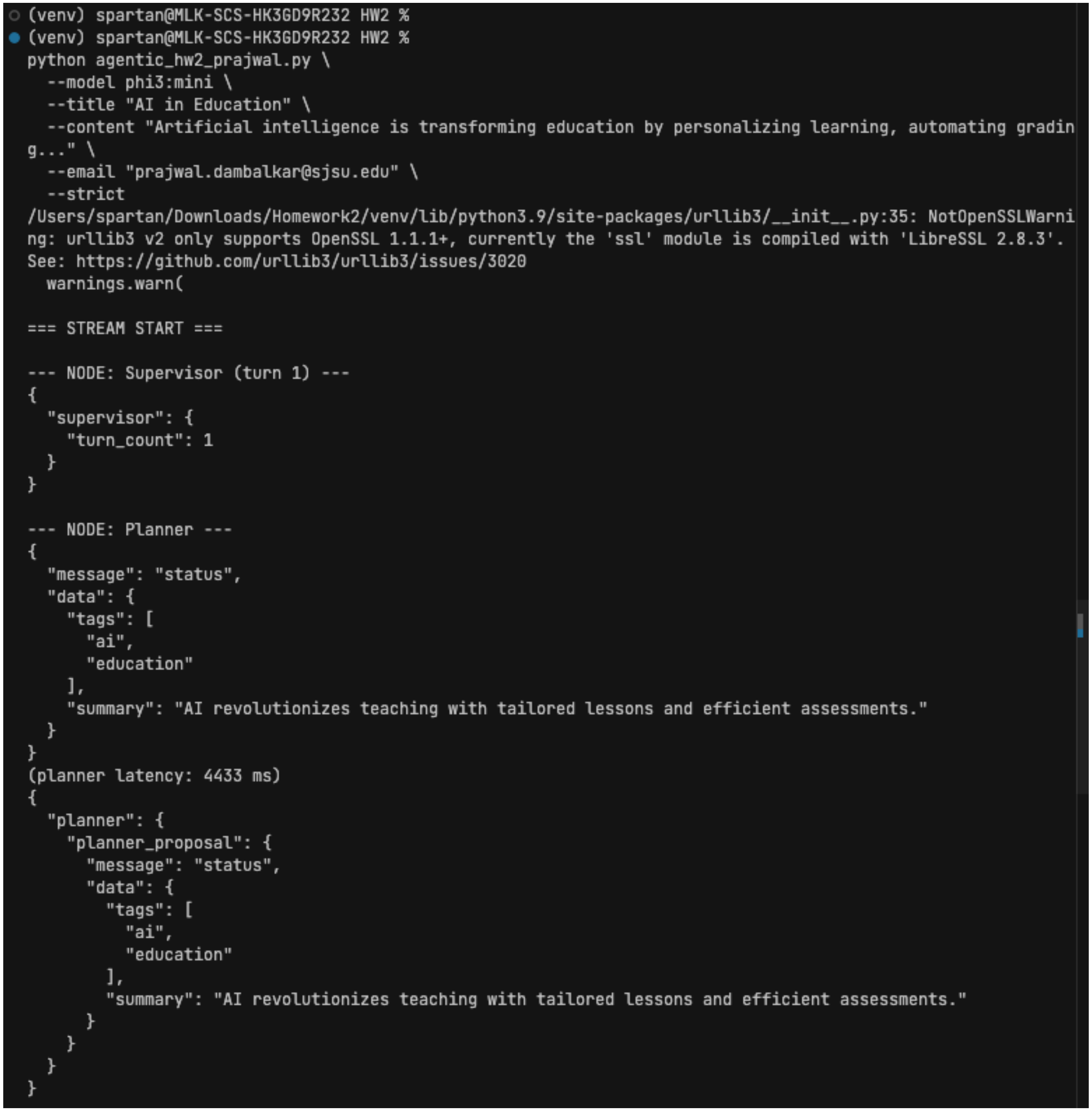
Step 3: Supervisor Logic
Step 4: Graph Assembly



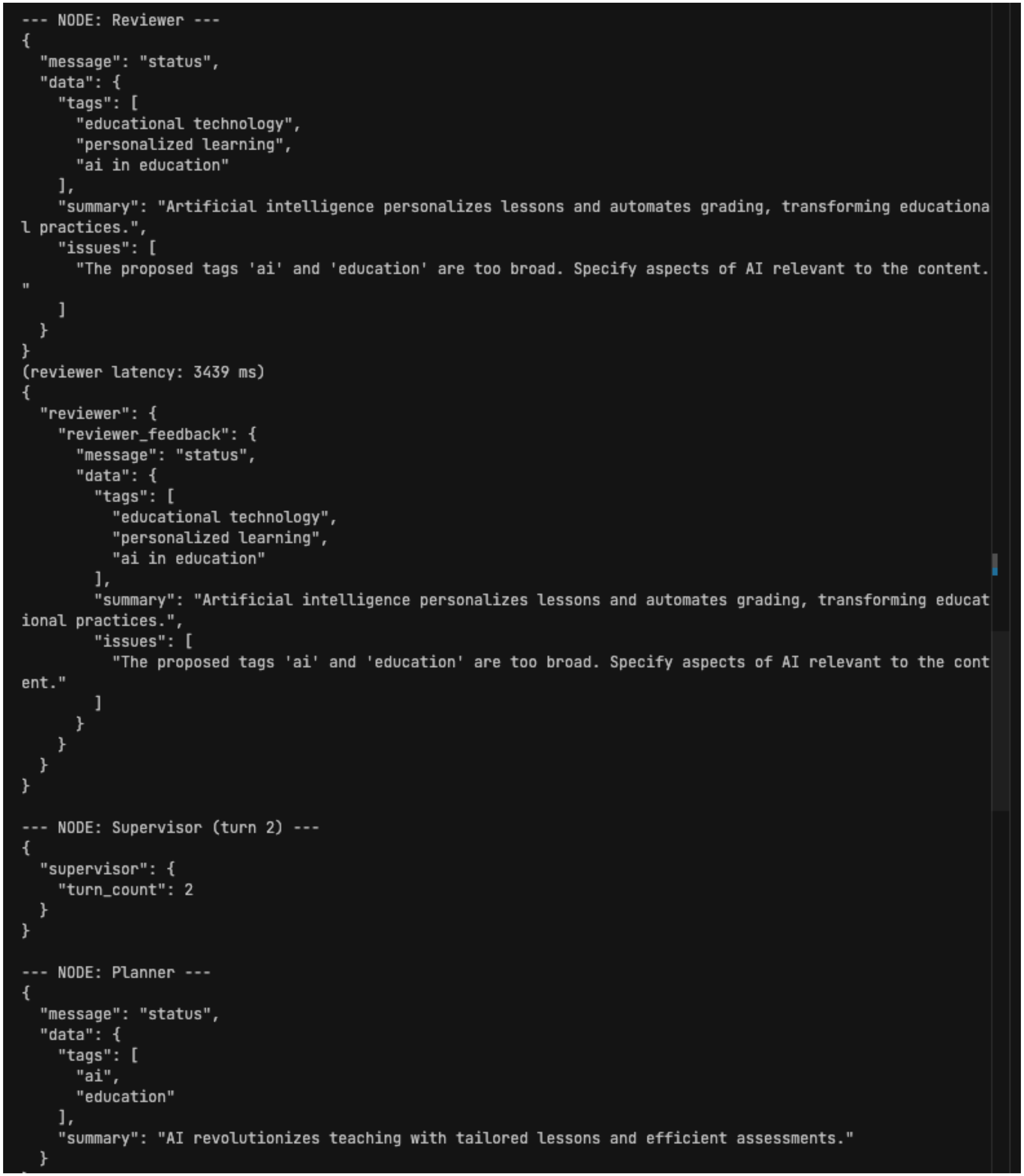
```python
# ----------------------------
# Supervisor & Router (Step 4)
# ----------------------------
MAX_TURNS = 6
def supervisor_node(state: AgentState) -> Dict[str, Any]:
    turn = int(state.get("turn_count", 0)) + 1
    print(f"\n--- NODE: Supervisor (turn {turn}) ---")
    return {"turn_count": turn}

def router_logic(state: AgentState) -> str:
    turn = state.get("turn_count", 0)
    plan = state.get("planner_proposal")
    review = state.get("reviewer_feedback")
    issues = (review.get("data", {}) or {}).get("issues", [])

    # If no plan yet → start with planner
    if not plan:
        return "planner"

    # If we already have a plan but no review → go to reviewer
    if plan and not review:
        return "reviewer"

    # If reviewer found issues and still under max turns → loop back to planner
    if issues and turn < MAX_TURNS:
        return "planner"

    # Otherwise stop
    return END
```



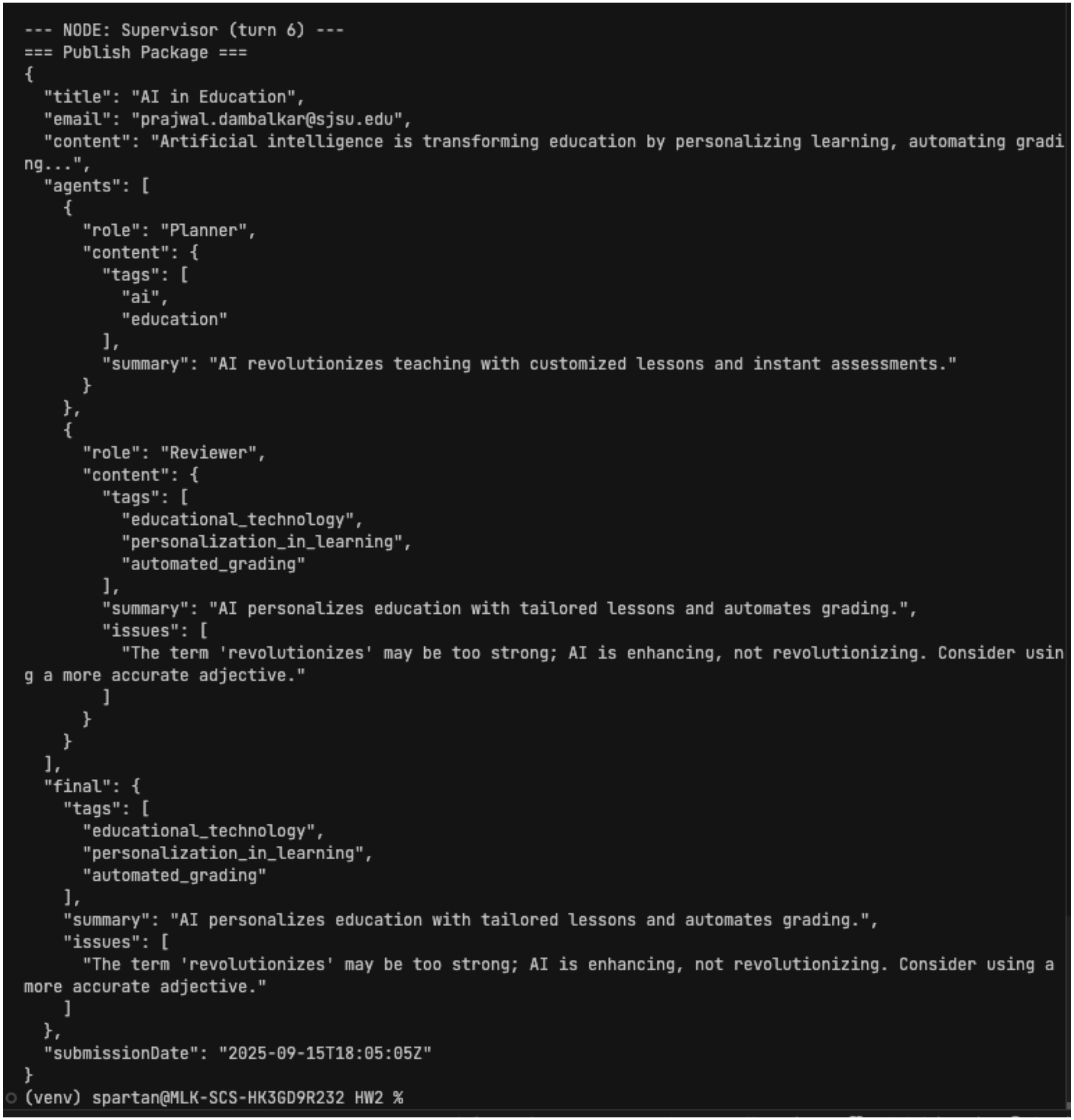
```python
# ----------------------------
# Graph Assembly (Step 5) & Run (Step 6)
# ----------------------------
def main():
    ap = argparse.ArgumentParser(description="DATA236 HW2 Agentic Supervisor Graph")
    ap.add_argument("--model", default="phi3:mini", help="Ollama model (default: phi3:mini)")
    ap.add_argument("--title", required=True)
    ap.add_argument("--content", required=True)
    ap.add_argument("--email", required=True)
    ap.add_argument("--strict", action="store_true", help="Enable nit-picky reviewer for loop demo")
    args = ap.parse_args()

    llm = ChatOllama(model=args.model, temperature=0.2)
    state: AgentState = {
        "title": args.title,
        "content": args.content,
        "email": args.email,
        "strict": bool(args.strict),
        "llm": llm,
        "planner_proposal": {},
        "reviewer_feedback": {},
        "turn_count": 0,
    }

    g = StateGraph(AgentState)
    g.add_node("planner", planner_node)
    g.add_node("reviewer", reviewer_node)
    g.add_node("supervisor", supervisor_node)
    g.set_entry_point("supervisor")
    g.add_conditional_edges("supervisor", router_logic, {"planner": "planner", "reviewer": "reviewer
    g.add_edge("planner", "reviewer")
    g.add_edge("reviewer", "supervisor")

    app = g.compile()
    print("\n=== STREAM START ===")
    last: Optional[AgentState] = None
    for delta in app.stream(state):
        print(json.dumps(delta, indent=2, ensure_ascii=False))
        last = delta
    print("=== STREAM END ===\n")
    if not isinstance(last, dict) or "title" not in last:
        last = app.invoke(state)

    pkg = publish_package(last)  # type: ignore
    print("=== Publish Package ===")
    print(json.dumps(pkg, indent=2, ensure_ascii=False))
```



```
○ (venv) spartan@MLK-SCS-HK3GD9R232 HW2 %
● (venv) spartan@MLK-SCS-HK3GD9R232 HW2 %
  python agentic_hw2_prajwal.py \
    --model phi3:mini \
    --title "AI in Education" \
    --content "Artificial intelligence is transforming education by personalizing learning, automating gradin
  g..." \
    --email "prajwal.dambalkar@sjsu.edu" \
    --strict
  /Users/spartan/Downloads/Homework2/venv/lib/python3.9/site-packages/urllib3/__init__.py:35: NotOpenSSLWarni
  ng: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'.
  See: https://github.com/urllib3/urllib3/issues/3020
    warnings.warn(

  === STREAM START ===

  --- NODE: Supervisor (turn 1) ---
  {
    "supervisor": {
      "turn_count": 1
    }
  }

  --- NODE: Planner ---
  {
    "message": "status",
    "data": {
      "tags": [
        "ai",
        "education"
      ],
      "summary": "AI revolutionizes teaching with tailored lessons and efficient assessments."
    }
  }
  (planner latency: 4433 ms)
  {
    "planner": {
      "planner_proposal": {
        "message": "status",
        "data": {
          "tags": [
            "ai",
            "education"
          ],
          "summary": "AI revolutionizes teaching with tailored lessons and efficient assessments."
        }
      }
    }
  }
```



```
--- NODE: Reviewer ---
{
  "message": "status",
  "data": {
    "tags": [
      "educational technology",
      "personalized learning",
      "ai in education"
    ],
    "summary": "Artificial intelligence personalizes lessons and automates grading, transforming educationa
l practices.",
    "issues": [
      "The proposed tags 'ai' and 'education' are too broad. Specify aspects of AI relevant to the content.
"
    ]
  }
}
(reviewer latency: 3439 ms)
{
  "reviewer": {
    "reviewer_feedback": {
      "message": "status",
      "data": {
        "tags": [
          "educational technology",
          "personalized learning",
          "ai in education"
        ],
        "summary": "Artificial intelligence personalizes lessons and automates grading, transforming educat
ional practices.",
        "issues": [
          "The proposed tags 'ai' and 'education' are too broad. Specify aspects of AI relevant to the cont
ent."
        ]
      }
    }
  }
}

--- NODE: Supervisor (turn 2) ---
{
  "supervisor": {
    "turn_count": 2
  }
}

--- NODE: Planner ---
{
  "message": "status",
  "data": {
    "tags": [
      "ai",
      "education"
    ],
    "summary": "AI revolutionizes teaching with tailored lessons and efficient assessments."
  }
```



```
--- NODE: Supervisor (turn 6) ---
=== Publish Package ===
{
  "title": "AI in Education",
  "email": "prajwal.dambalkar@sjsu.edu",
  "content": "Artificial intelligence is transforming education by personalizing learning, automating gradi
ng...",
  "agents": [
    {
      "role": "Planner",
      "content": {
        "tags": [
          "ai",
          "education"
        ],
        "summary": "AI revolutionizes teaching with customized lessons and instant assessments."
      }
    },
    {
      "role": "Reviewer",
      "content": {
        "tags": [
          "educational_technology",
          "personalization_in_learning",
          "automated_grading"
        ],
        "summary": "AI personalizes education with tailored lessons and automates grading.",
        "issues": [
          "The term 'revolutionizes' may be too strong; AI is enhancing, not revolutionizing. Consider usin
g a more accurate adjective."
        ]
      }
    }
  ],
  "final": {
    "tags": [
      "educational_technology",
      "personalization_in_learning",
      "automated_grading"
    ],
    "summary": "AI personalizes education with tailored lessons and automates grading.",
    "issues": [
      "The term 'revolutionizes' may be too strong; AI is enhancing, not revolutionizing. Consider using a
more accurate adjective."
    ]
  },
  "submissionDate": "2025-09-15T18:05:05Z"
}
○ (venv) spartan@MLK-SCS-HK3GD9R232 HW2 %
```