? Lumina Library: The "Deep Dive" Guide

"What is actually written in those files?"

Before we look at the code, let's meet the team. Imagine this software is a group of people working together to run a library.

---

1. The Cast of Characters (Plain English)

Here is who everyone is and what they do:

?? `index.html` (The Architect)

Role: Builds the walls and shelves.

Plain English: "I will put a button here. I will put a text box there. I will make a list here."

Relationship: He builds the skeleton but leaves it ugly and gray. He needs the Stylist (`style.css`) to make it look good.

? `style.css` (The Stylist)

Role: Decorates the building.

Plain English: "Make that button blue. Make the text bold. Give the cards a glass effect."

Relationship: She follows the Architect's plan but adds color and beauty.

? `script.js` (The Assistant)

Role: Runs errands and talks to customers.

Plain English: "The user clicked 'Search'? Okay, I'll run to the back office (`app.py`) and ask for the book list."

Relationship: He is the messenger. He takes orders from the User and runs to the Manager (`app.py`).

? `app.py` (The Manager)

Role: Makes decisions and follows rules.

Plain English: "You want to borrow a book? Let me check if we have any left. If yes, I'll approve it."

Relationship: He is the boss. He tells the Assistant (`script.js`) "Yes" or "No". He reads the Record Book (`json`).

? `books.json` (The Record Book)

Role: Remembers everything.

Plain English: "Book #101 is 'Harry Potter'. We have 3 copies."

Relationship: It just sits there holding information. The Manager (`app.py`) reads from it and writes to it.

---

2. Example: How They Talk to Each Other

Imagine you want to Issue a Book. Here is the conversation they have:

1. You (User): Click the "Issue" button on the screen.
2. Assistant (`script.js`): "Hey Manager (`app.py`)! The user wants Book #101."
3. Manager (`app.py`): "Hold on. Let me check the Record Book (`books.json`)."
4. Manager: *Opens book.json*. "Ah, we have 3 copies. Okay, I'll cross one out. Now we have 2."
5. Manager: "Okay Assistant, tell the user it's done!"
6. Assistant: "Great!" *Shows a green 'Success' popup on the screen.*
7. Stylist (`style.css`): "And I'll make sure that popup looks shiny and green!"

---

3. The Code: A Deep Dive

Now that you know who they are, let's look at the actual language they speak.

? `static/index.html` (The Architect)

The Code:

```
<button class="btn-primary" onclick="issueBook()">
    Issue Book
</button>
```

Translation: "Put a button here. Style it like a 'primary' button. When clicked, tell the Assistant to run `issueBook`."

? `static/css/style.css` (The Stylist)

The Code:

```
.btn-primary {
    background-color: #0a84ff; /* Blue */
    border-radius: 10px;       /* Rounded Corners */
}
```

Translation: "Any button named 'primary' should be painted Blue and have smooth corners."

? `static/js/script.js` (The Assistant)

The Code:

```
function issueBook() {
    let bookId = document.getElementById("bookSelect").value;
    fetch("/api/issue", { method: "POST", body: bookId });
}
```

Translation: "When `issueBook` runs: Find the selected book ID. Then run to the Manager at the `/api/issue` desk and give it to him."

? `app.py` (The Manager)

The Code:

```
@app.route('/api/issue', methods=['POST'])
def issue_book():
    if book['available_copies'] > 0:
        book['available_copies'] -= 1
        return "Success"
    else:
        return "Error"
```

Translation: "If the Assistant comes to the `/api/issue` desk: Check if copies > 0. If yes, subtract 1 and say 'Success'. If no, say 'Error'."

? `data/books.json` (The Record Book)

The Code:

```
[
  {
    "id": 101,
    "title": "The Pragmatic Programmer",
    "available_copies": 3
  }
]
```

Translation: "Book #101 exists. We have 3 copies."