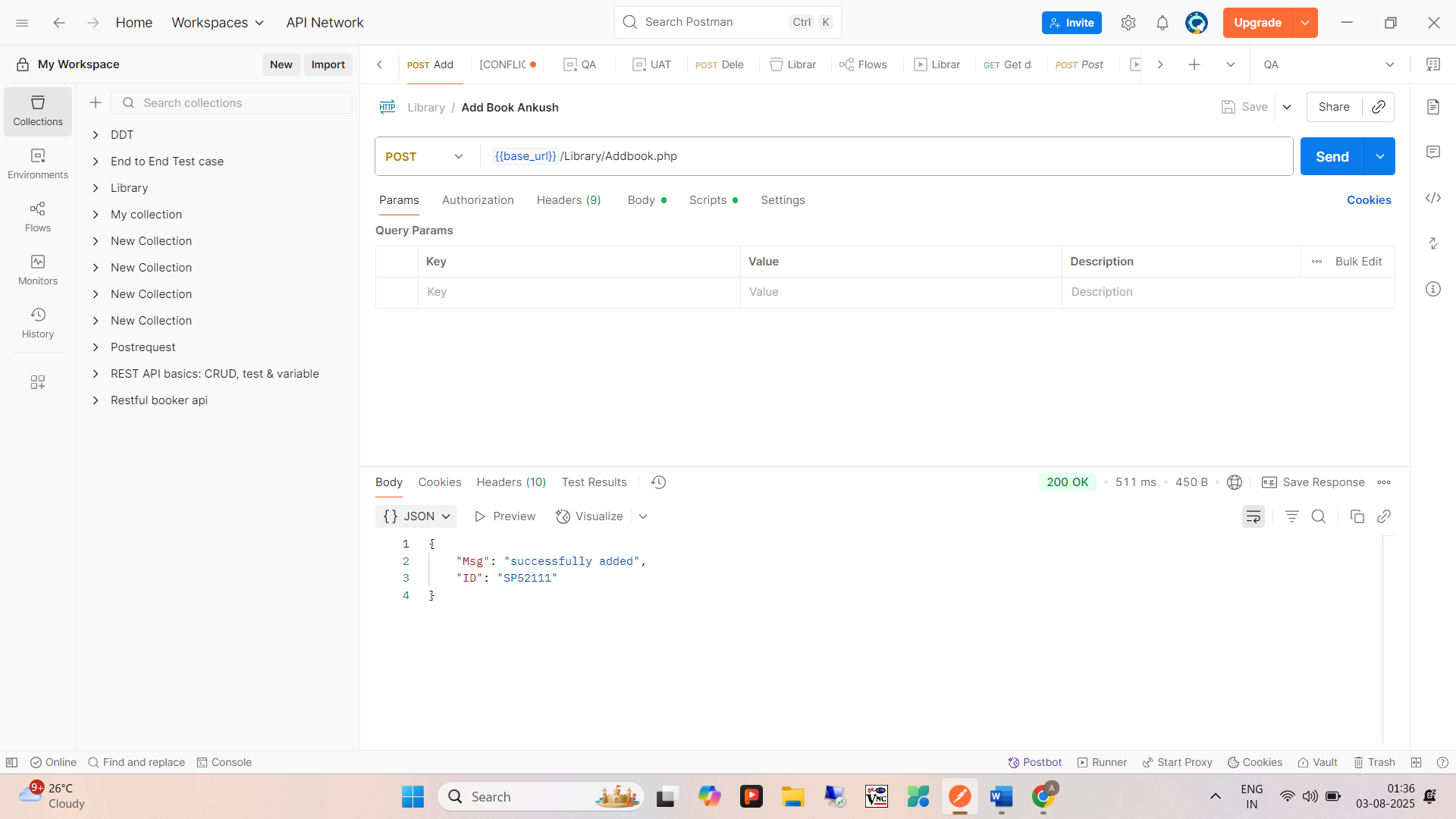
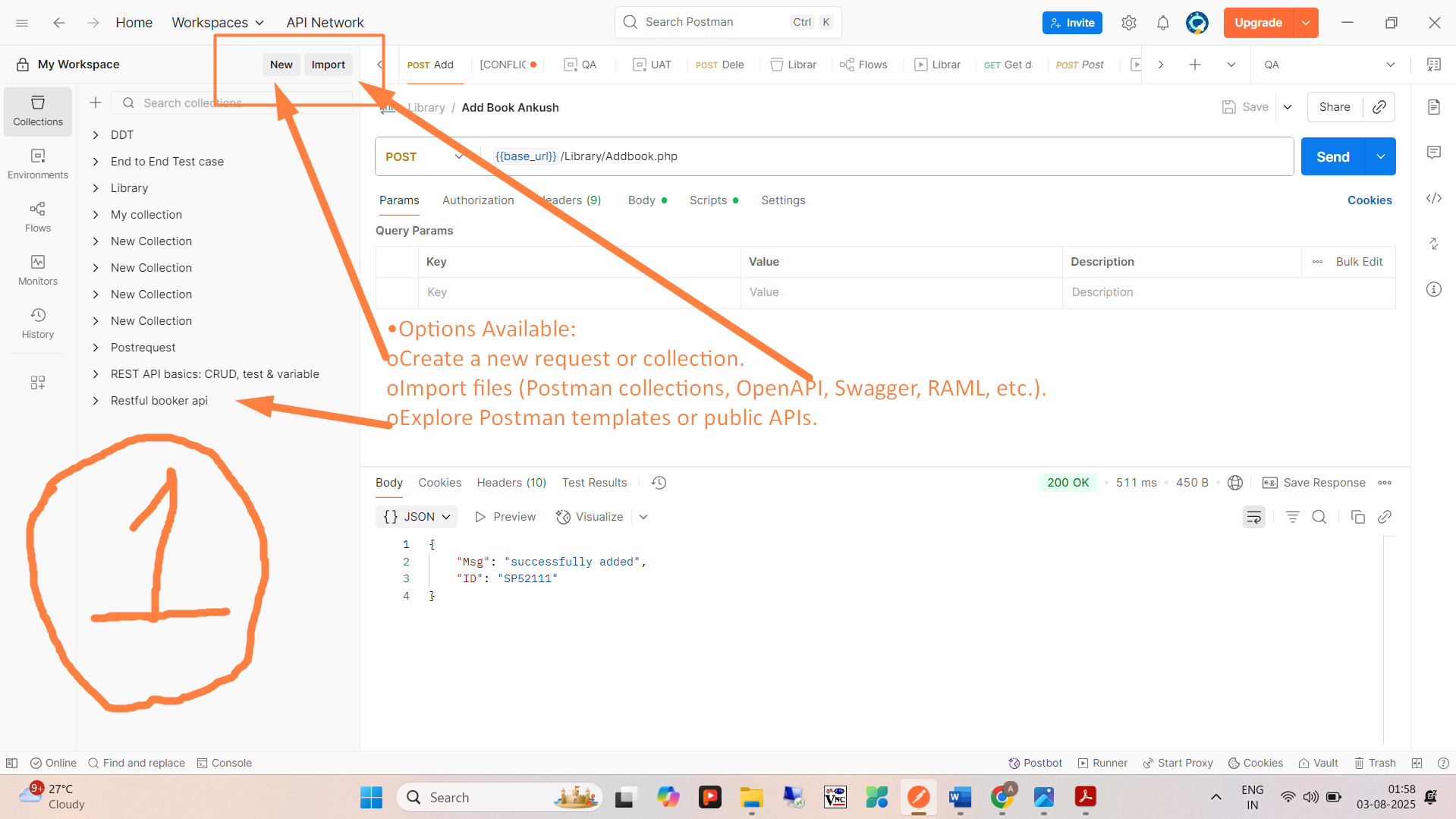
🔧 **Postman UI Section-wise Detailed Documentation:-**



A). Side bar Section:

**1. Home/Launchpad**

* **Where to find**: Opens by default when Postman launches.
* **Purpose**:
  + Quick access to recent workspaces, APIs, collections, environments.
  + Create new requests, collections, or APIs.
* **Options Available**:
  + Create a new request or collection.
  + Import files (Postman collections, OpenAPI, Swagger, RAML, etc.).
  + Explore Postman templates or public APIs.



**2. Sidebar Navigation (Left Panel)**

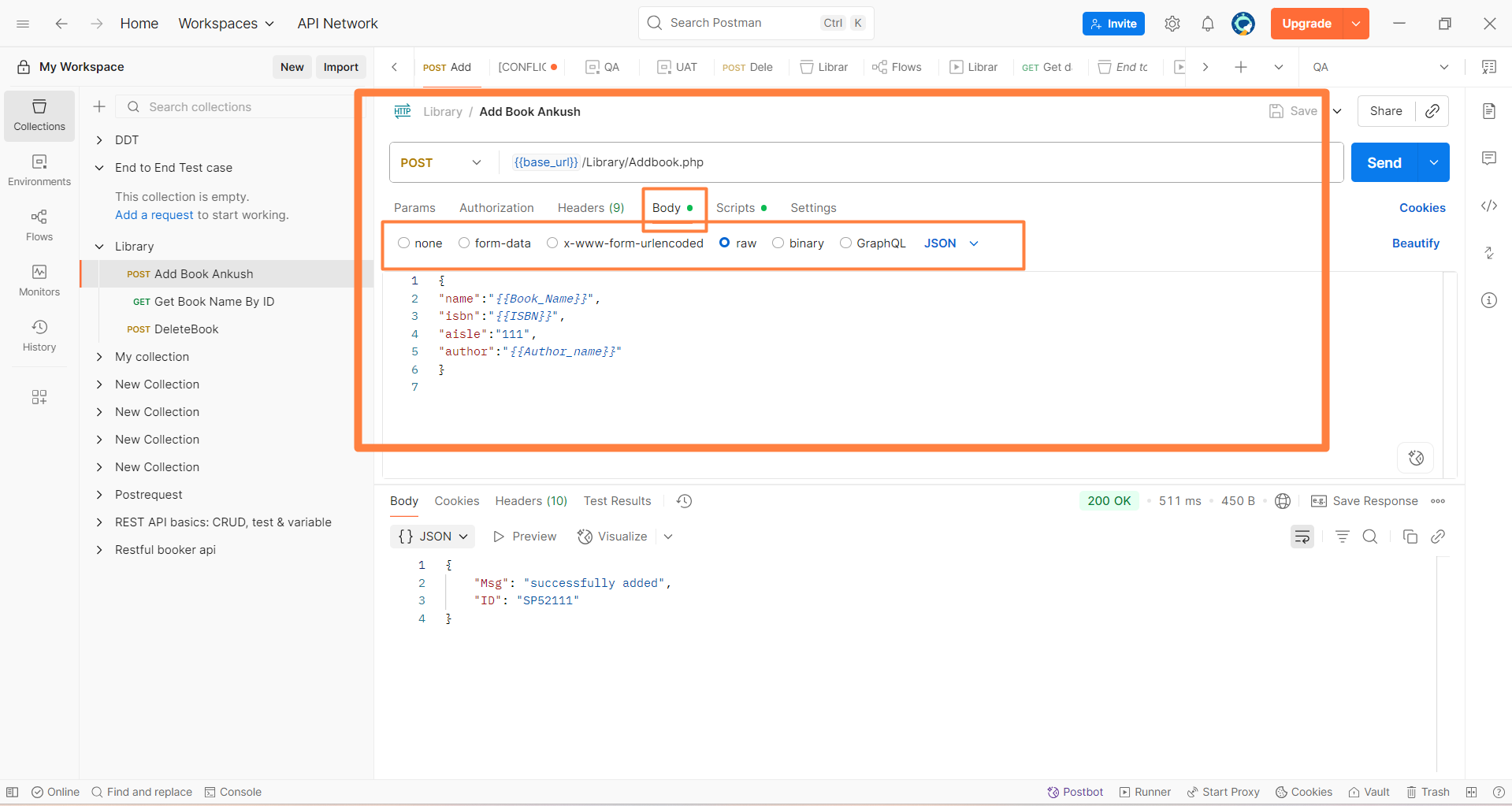
* **Sections**:
  + **Workspaces**: Collaborative space for collections, environments, etc.
  + **Collections**: Stores grouped requests.
  + **APIs**: Manage, define, and document APIs.
  + **Environments**: Manage variables like baseUrl, token, etc.
  + **Mock Servers**: Simulate APIs.
  + **Monitors**: Schedule request runs.
  + **Flows**: Visual API workflows.
* **Purpose**: Organize and switch between core functionalities easily.

**3. Top Bar**

* **Elements**:
  + **New**: Create requests, collections, environments, mock servers, etc.
  + **Import**: Upload files (Postman collection, Swagger, etc.).
  + **Runner**: Execute request collections in sequence.
  + **Search**: Find elements within workspace.
  + **Account/Profile**: Settings, account switch, or upgrade plans.
* **Usage Tip**: Shortcut to start or import your API testing quickly.

**4. Request Builder (Middle Panel)**

* **Components**:
  + **Method Dropdown**: GET, POST, PUT, DELETE, etc.
  + **Request URL**: Endpoint URL.
  + **Params Tab**: Add query parameters.
  + **Authorization**: Auth types (Bearer, Basic, OAuth 2.0).
  + **Headers**: Custom request headers.
  + **Body**: Raw, JSON, Form-data, x-www-form-urlencoded, etc.
  + **Pre-request Script**: JS to run before request.
  + **Tests**: JS tests for response validation.
* **Send Button**: Trigger the request.



**🧩 Where is the "Body" Tab in Postman?**

When you choose methods like POST, PUT, or PATCH, Postman enables a tab called **"Body"**—used to **send data to the server**.

**✅ Types of Body Data Input in Postman (with Examples)**

**1️⃣ none**

* 🚫 No body is sent with the request.
* 📌 Use when the API doesn’t expect any body, like GET or DELETE.

**2️⃣ form-data**

* 📦 Used for **multipart/form-data** requests (usually for file uploads or form submissions).
* You can send **key-value pairs** (some keys can be files).
* Automatically sets Content-Type: multipart/form-data.

**🧪 Example: Upload a file with user ID**

| **Key** | **Type** | **Value** |
| --- | --- | --- |
| userId | Text | 123 |
| file | File | resume.pdf |

➡️ Used in POST /upload

**3️⃣ x-www-form-urlencoded**

* Like HTML form submits.
* Sends key-value pairs encoded in the body (like key1=value1&key2=value2).
* Automatically sets Content-Type: application/x-www-form-urlencoded.

**🧪 Example:**

| **Key** | **Value** |
| --- | --- |
| username | ankushagle |
| password | secret123 |

➡️ Used in login APIs or legacy systems

**4️⃣ raw**

* Used when you want to **send JSON, XML, Text, HTML, etc.** directly in the body.
* You must **select the format** from the dropdown (Text / JSON / XML / HTML).
* You **manually write** the body content.

**🧪 Example: Send JSON to create user**

json

CopyEdit

{

"name": "Ankush",

"email": "ankush@example.com",

"role": "QA"

}

* Content-Type should be: application/json

➡️ Used in most REST APIs (POST, PUT, PATCH)

**5️⃣ binary**

* Used to send a **file directly** as binary data (e.g., an image or PDF) in the request body.
* You upload only the **file**, no form fields.
* Content-Type is usually application/octet-stream.

**🧪 Example:**

* PUT /updateProfilePicture
* Select profile.png in the binary tab

**5. Response Section (Bottom Panel)**

* **What it shows**:
  + Status code (200 OK, 404 Not Found, etc.).
  + Response time & size.
  + Response body in JSON, XML, HTML, etc.
  + Cookies tab.
  + Headers tab.
  + Test Results (if written in "Tests" tab).
* **Purpose**: Analyze API response, debug, and validate.

**6. Collection Sidebar Details**

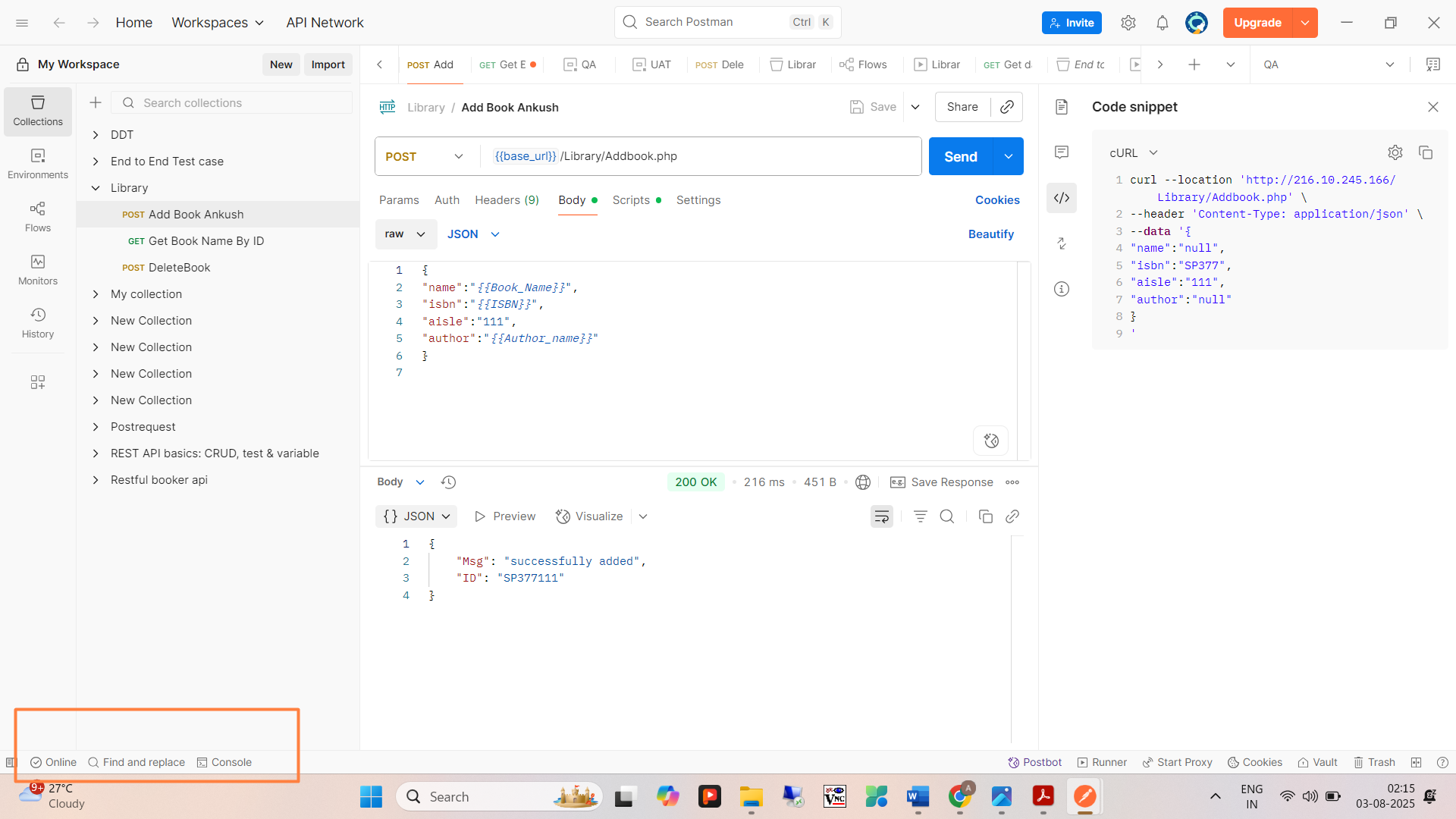
* **Inside a Collection**:
  + Add folders to organize requests.
  + Each request can be saved with method, URL, headers, scripts.
  + Collection-level pre-request scripts and tests.
* **Use Case**: Grouping related endpoints like Login, Signup, Logout.

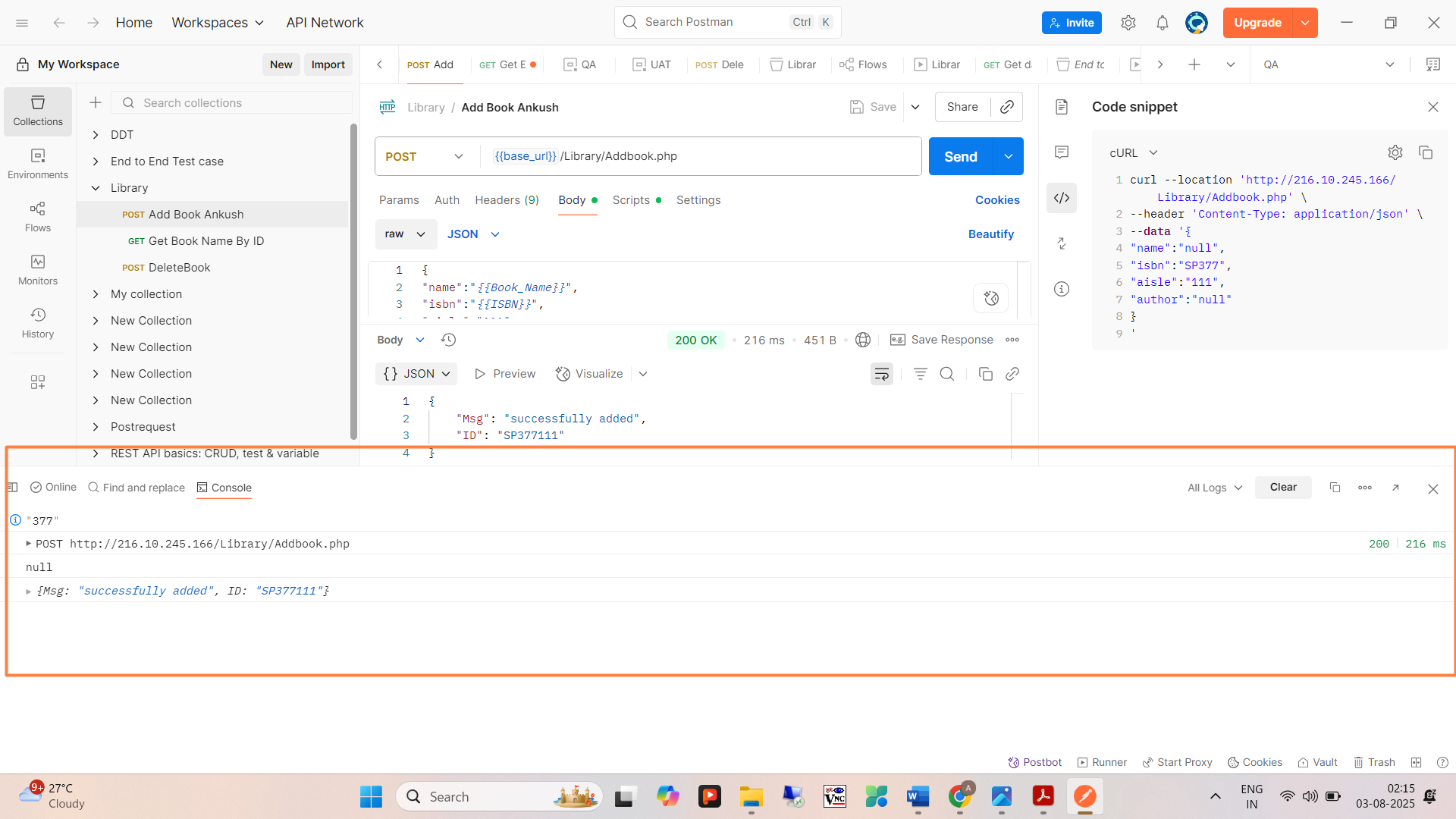
**7. Environment & Global Variables**

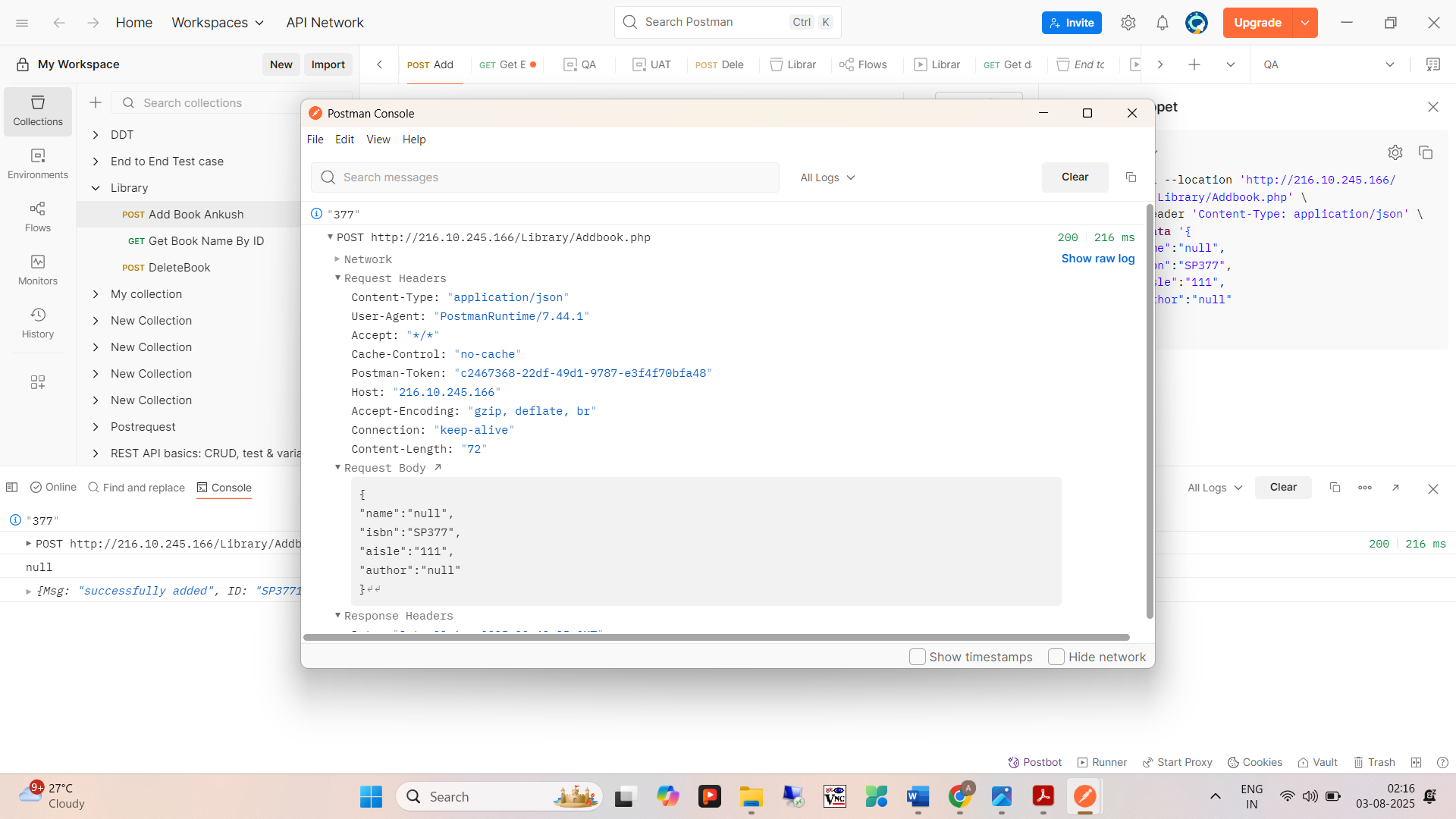
* **Where to find**: Top right (Eye icon or Environment dropdown).
* **Environment**:
  + Defined sets of key-value pairs.
  + Example: baseUrl = https://dev.api.com
* **Globals**: Variables available in any environment.
* **Use**: Switch between DEV/QA/PROD easily without changing requests manually.

**8. Console**

* **Shortcut**: Ctrl + Alt + C
* **What it does**:
  + Logs request and response data.
  + Helps in debugging test scripts or failed requests.





s

**9. Settings (Gear Icon)**

* **Found at top-right corner**.
* **Main Tabs**:
  + General: Theme, request timeout, language.
  + Request: Behavior of redirects, SSL certs.
  + Proxy: For setting up custom proxy.
  + Certificates: Add client SSL certs.
  + Data: Clear cache, cookies, history.

**10. Collection Runner**

* **Where**: Top-left sidebar → Run icon or Runner tab.
* **Use Case**: Run multiple requests in a sequence (with data files like CSV or JSON).
* **Options**:
  + Select iteration count.
  + Data file upload.
  + Run summary report.

**11. Monitors**

* **Use**: Schedule requests or test suites to run hourly, daily, etc.
* **How to use**:
  + Create monitor → Choose collection → Select environment → Set frequency → Get alerts (email, Slack, etc.).

**12. Mock Servers**

* **Purpose**: Test API integration even before backend is ready.
* **Steps**:
  + Create mock server.
  + Define sample response for a URL.
  + Share mock URL with frontend/devs.

**13. Flows**

* **New Visual Tool**: Automate API interactions using a visual drag-and-drop flow.
* **Use Case**: Chain APIs without coding logic manually.