Perfect — this screenshot is basically a **cheat-sheet** for how to **identify application entry points** in a web application. Let me break it down into **clean notes** for you:

**🔑 How to Identify Application Entry Points**

Application entry points = places where **user input** or **requests** interact with the application. These are often attack vectors.

**1. Request / Response**

* Inspect HTTP **Requests** and **Responses**.
* Look at headers, methods, response codes.
* Tools: Browser DevTools, Burp Suite, OWASP ZAP.

**2. GET / POST**

* **GET parameters** (in URL):  
  http://site.com/page.php?id=10&cat=books
* **POST parameters** (in body):  
  username=admin&password=1234
* Both can contain vulnerable input fields.

**3. Parameters & Special Characters**

* Look for parameters (id=, page=, q=).
* Special characters like &, ~, ;, \ may cause unusual behavior.
* Try modifying values to see response changes.

**4. HTTP Status Codes (Indicators)**

* **200** = OK (valid endpoint)
* **302** = Redirect (can reveal hidden pages)
* **404** = Not Found (check for path guessing)
* **500/501** = Server error (may leak stack traces)

**5. Cookies**

* Stored on client → sent with requests.
* May contain:
  + Session tokens (PHPSESSID=...)
  + Authentication details
  + Tracking IDs
* Modifying cookies can expose hidden functionality.

**6. Attack Surface Detector**

* Automated tool that identifies possible **attack surfaces**:
  + Endpoints
  + Parameters
  + Hidden APIs

**7. Burp Suite (MITM)**

* Acts as a **man-in-the-middle proxy**.
* Lets you intercept, modify, and replay requests.
* Useful for finding hidden parameters and vulnerabilities.

**8. ZAP Proxy (OWASP ZAP)**

* Alternative to Burp Suite (open-source).
* Can spider a site and map out all entry points.

✅ **Summary Flow for Entry Point Discovery:**

1. Crawl the site with **Burp/ZAP** → find endpoints.
2. Check **requests/responses** for hidden parameters.
3. Test **GET/POST values** and **special chars**.
4. Inspect **cookies** and session handling.
5. Monitor **HTTP codes** (200/302/500 etc.).
6. Cross-check with **attack surface detector tools**.

Do you want me to make you a **visual workflow diagram** of this (like a mindmap/flowchart) so you can remember the process faster?