🗓️Day 3: URL Tracking and Avoiding Revisits with Hashtable

📅User Story: Prevent Re-crawling of URLs

💼As a Developer, I want to:

- Implement a mechanism to track visited URLs during the crawl process
- Ensure that the crawler avoids revisiting any URL it has already processed
- Maintain clean and efficient traversal of web pages using DFS (Depth-First Search)

📅Tasks:

1. **Implement a Hashtable for Visited URLs:**
2. Design and integrate a hashtable (or map/set) to record URLs that have been visited.
3. **Check Before Visit:**
4. Before visiting a URL in the DFS function, check if it exists in the hashtable.
5. Only proceed if the URL hasn't been visited yet.
6. **Mark as Visited:**
7. Once a URL is visited, immediately mark it in the hashtable to prevent duplicate traversal.
8. **Integrate with Crawl Function:**
9. Update the existing crawl logic to consult the hashtable at every step.
10. **Enforce Global Limits:**
11. Ensure the crawler respects a maximum recursion depth (maxDepth).
12. Limit total links crawled (maxLinks).
13. **Skip Counting Duplicates:**
14. Ensure URLs skipped due to duplication are not counted towards the maximum link limit.

🗜️Outcome:

- A robust DFS-based crawler that:

- Extracts and normalizes links.
- Uses a hashtable to track and skip already visited URLs.
- Efficiently explores the web without redundant crawling.
- Enforces depth and link count constraints correctly.

🔗Integration Point:

- This user story directly builds on the previously implemented DFS crawler and enhances it with URL tracking for better performance and correctness.

🕐Success Criteria:

- Crawler completes without re-visiting any URL.
- Duplicate URLs do not affect global limits.
- All links are normalized and tracked effectively.
- Logs show visited and skipped URLs clearly.