**News Aggregator Documentation**

**1. Title Page**

* **Project Title:** My News Aggregator
* **Student Names:  
  20221289** Martyn Orupabo **20220240** Usman Ali Alhassan **20220997** Markmanuel Zibediomuze Markbere **20220369** Usman Ma’aji Arfo **20232657** Suhaib Sulema Bodinga
* **Date:** 8th May, 2025
* **Course/University:** Nile University, Web Application Development(SEN 311)

**2. Table of Contents**

1. Title Page
2. Table of Contents
3. Project Overview
4. Requirements
5. System Architecture
6. Design
7. Core Functionalities
8. Implementation Details
9. Testing
10. Deployment
11. Challenges and Solutions
12. Future Improvements
13. Conclusion
14. References

**3. Project Overview**

**Description:** A responsive web application that aggregates curated RSS/Atom news feeds into a single interface, allowing users to browse, search, and paginate through multiple sources with a modern UI featuring dark/light mode and lazy‑loaded images.

**Target Audience:**

* News enthusiasts who prefer a consolidated view of multiple sources.
* Students and researchers needing quick access to updates from various feeds.

**Key Features:**

* Dynamic feed list driven by feeds.json.
* Pagination, search, and customized snippets.
* Dark/Light mode toggle with persisted preference.
* Lazy‑loaded images and responsive Bootstrap cards.

**4. Requirements**

**Hardware/Software:**

* Any modern desktop or mobile device with a web browser.
* Python 3.8+ installed.

**Libraries/Frameworks/APIs:**

* Flask
* Flask-Caching
* Feedparser
* Bootstrap
* python-dateutil

**5. System Architecture**

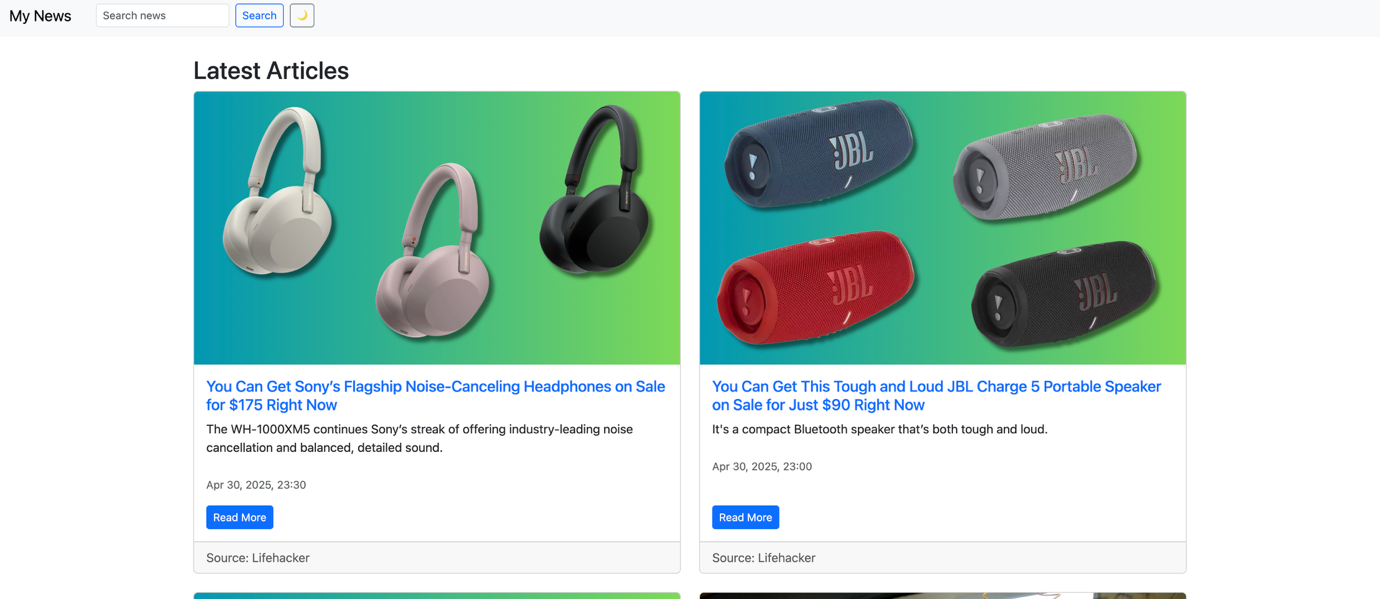
**Diagram:**

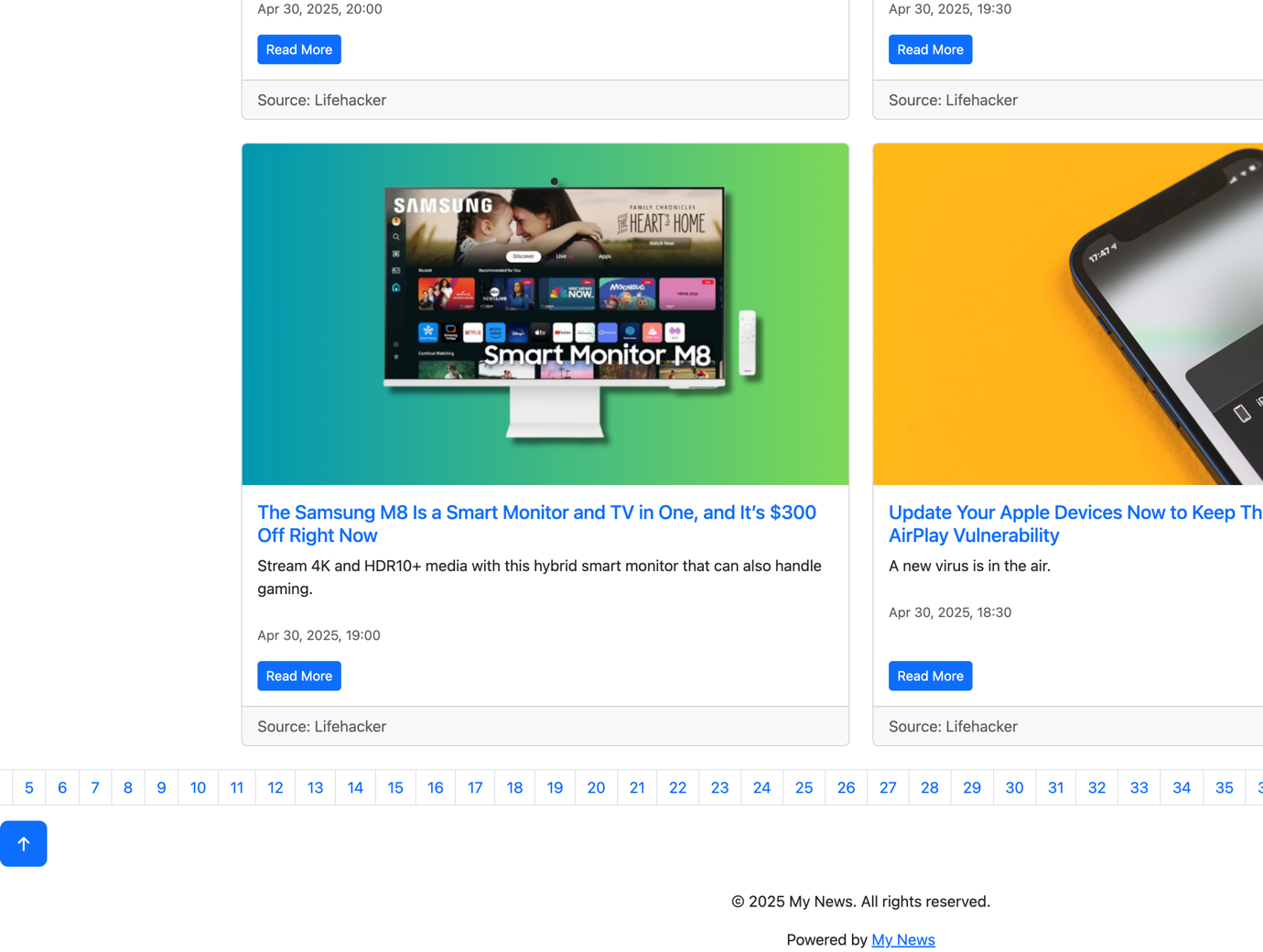
**Interactions:**

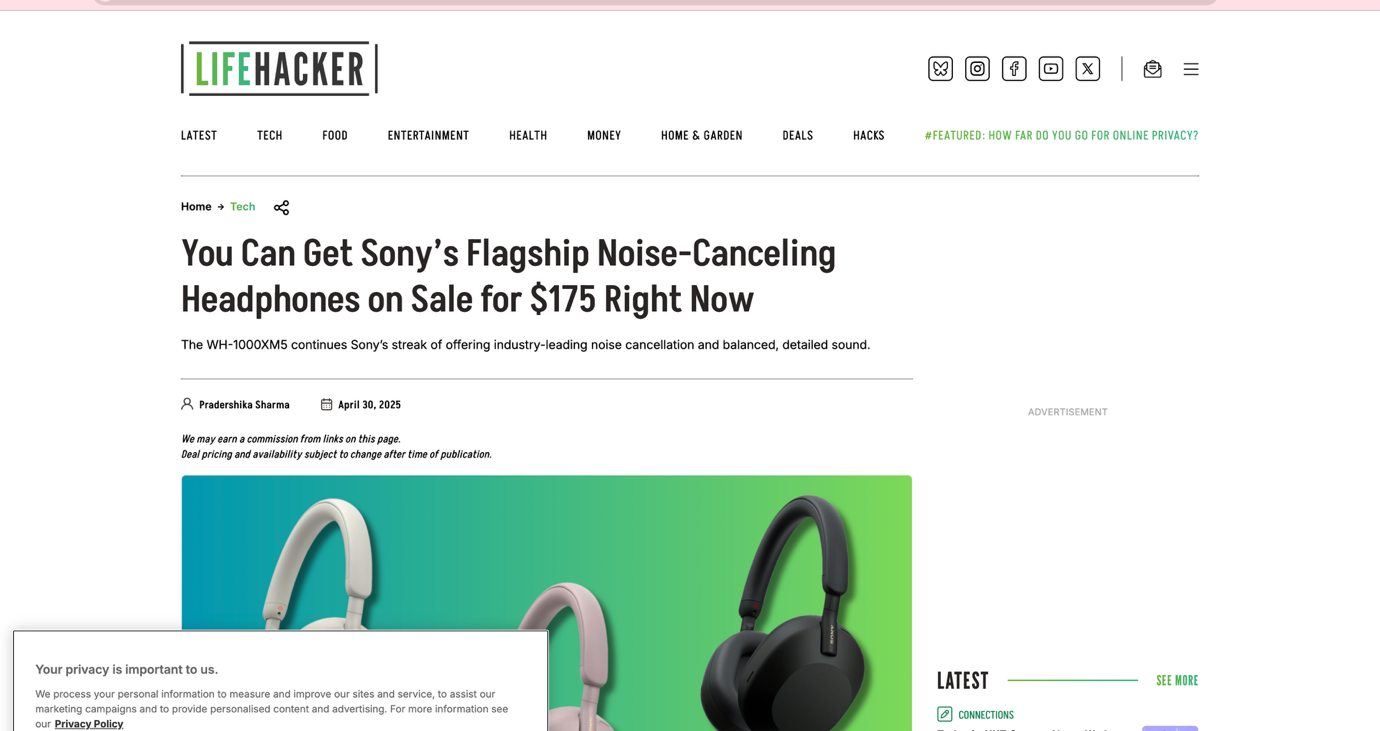
* Frontend renders via Jinja, calls routes.
* Routes fetch aggregated articles from cache‑enabled parser.
* Parser reads feeds.json, requests remote RSS URLs, parses entries.

**6. Design**

**Wireframes/Screenshots:**







**Database Schema:**

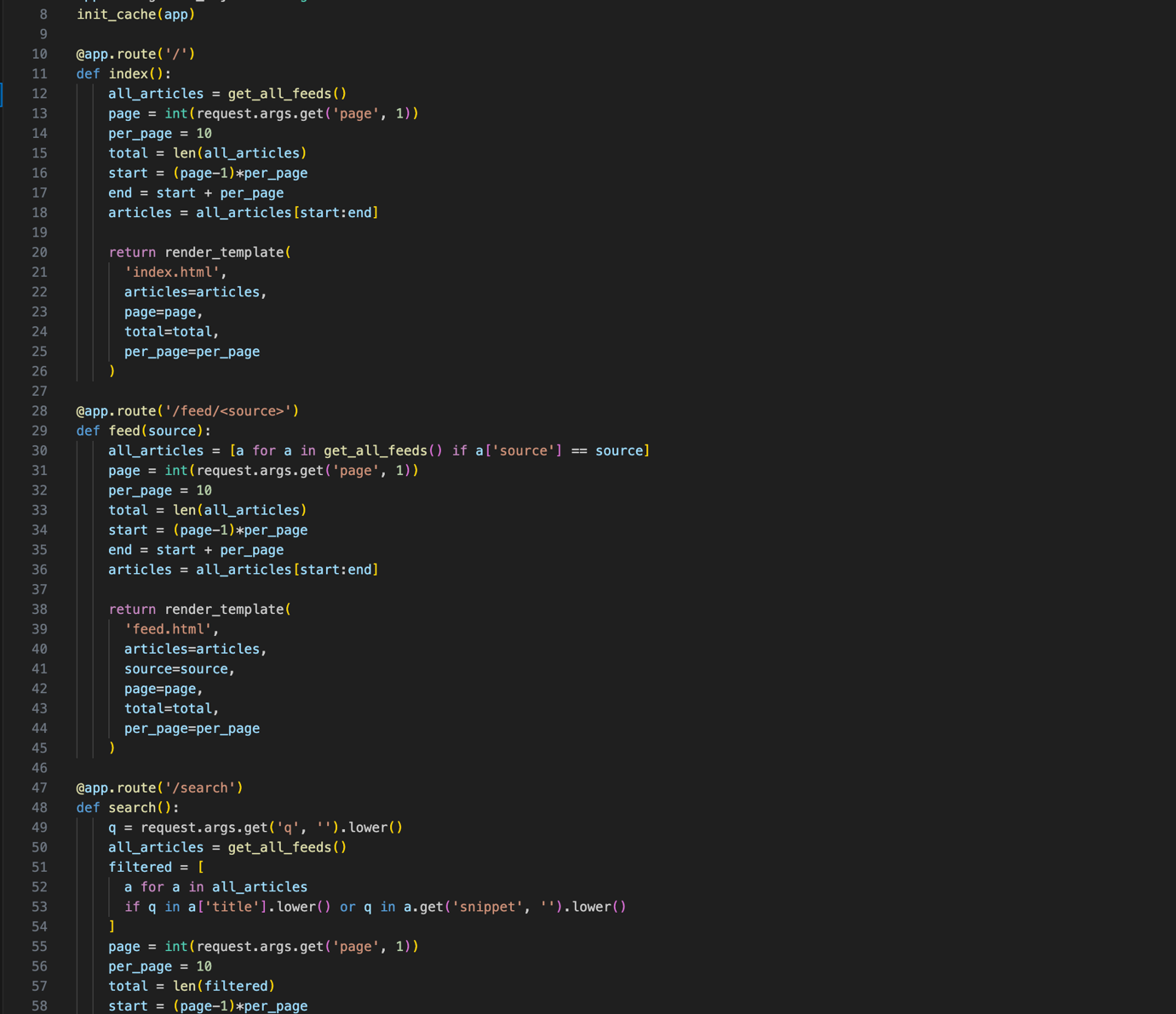
* No persistent database; feed list stored in JSON and cached in memory.

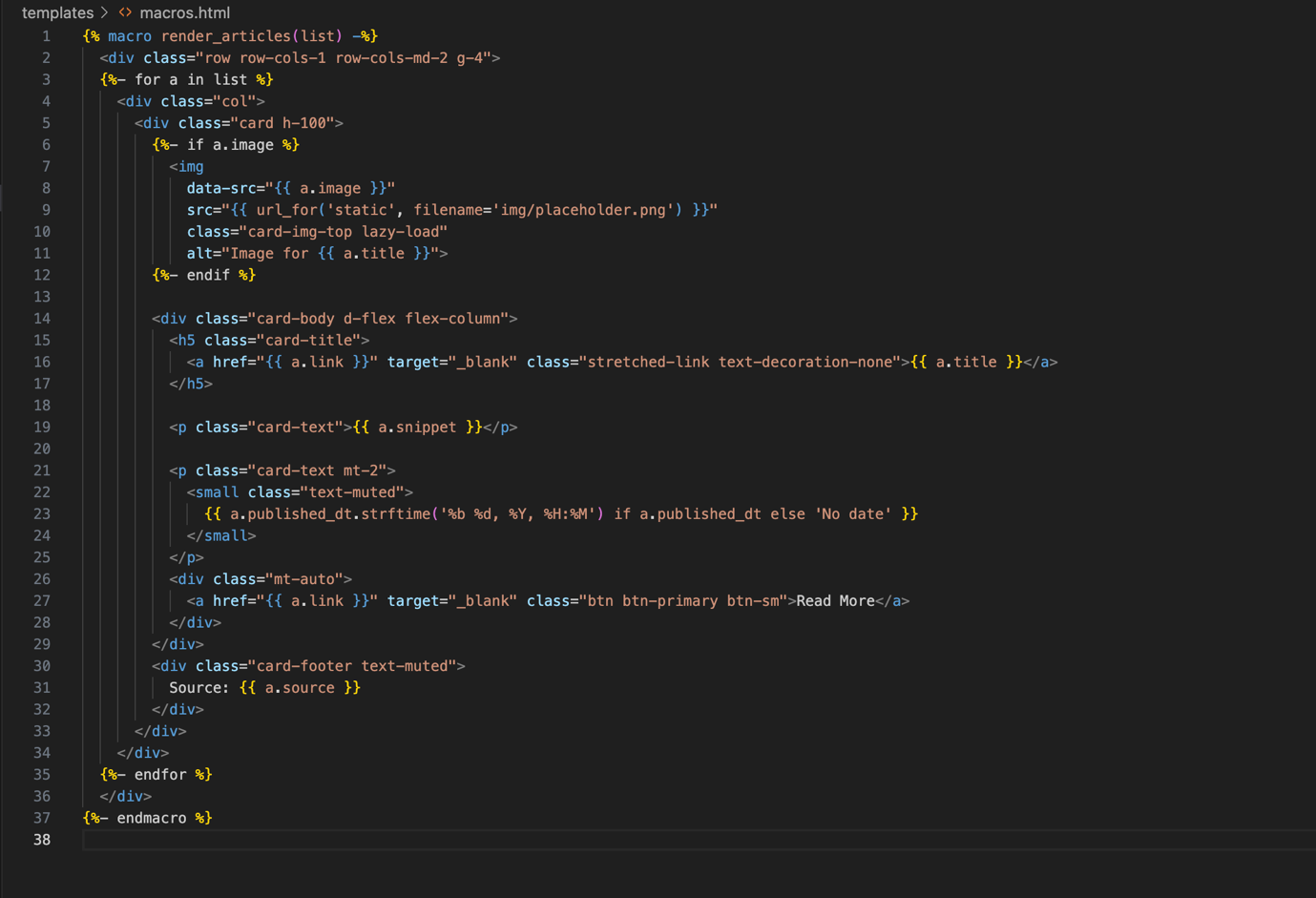
**7. Core Functionalities**

1. **Home (All News):** Displays paginated list of all articles.
2. **Search:** Keyword filter across titles/snippets.
3. **Pagination:** Server‑side paging with horizontal scroll.
4. **Lazy Loading:** IntersectionObserver‑based image loading.

**8. Implementation Details**

**Sample Route (index):**

**Template Snippet (macros.html):**



**Third‑Party Integration:**

* RSS parsing via feedparser.
* Date parsing with python-dateutil.

**9. Testing**

* **Approach:** Manual testing of routes, edge cases (zero results), and responsive layout.
* **Issues:** Fixed division by zero in pagination and invalid template syntax for URL generation.

**10. Deployment**

* **Platform:** Deployed on Heroku.
* **Steps:**
  1. git push heroku main
  2. Set SECRET\_KEY and config vars.
  3. Enable Python buildpack.

**11. Challenges and Solutions**

* **Cache context error:** Moved timeout config into init\_cache.
* **Template syntax errors:** Adjusted pagination links and macro imports.

**12. Future Improvements**

* User login & personalized feeds.
* Database persistence for articles and user settings.
* Real‑time updates with WebSockets.

**13. Conclusion**

This project demonstrates building a dynamic yet lightweight news aggregator using Flask and Jinja, emphasizing usability, performance, and modern UI patterns.

**14. References**

* Flask Documentation: https://flask.palletsprojects.com/
* Feedparser: https://pythonhosted.org/feedparser/
* Bootstrap: https://getbootstrap.com/