Journal of Informatics, Computer Science, Data Science And Arificial Intelligence
Volume 4 No. 1 June 2026, ISSN : 3031-9145



A COMPREHENSIVE NATURAL SCIENCE SUMMARY E-BOOK APPLICATION USING LINEAR SEARCH ALGORITHM

Wibi Ananda[1], Mhd Zulfansyuri Siambaton[2], Rachmat Aulia[3]

[1,2,3]Informatics Engineering Study Program, Faculty of Engineering, Universitas Islam Sumatera Utara, [1)]wibiananda71@gmail.com, [2)]zulfansyuri@ft.uisu.ac.id, [3)]rachmataulia@gmail.com

ARTICLE INFO

ARTICLE HISTORY:
Received :26 January 2026
Revised : 07 Mei 2026
Accepted : 09 June 2026

Keywords: *Android; E-Book; Flipbook; Jetpack Compose; Linear Search*

ABSTRACT

Reference summary books for natural science, known as Rangkuman Pengetahuan Alam Lengkap (RPAL), are mostly distributed as printed books or static PDF files that offer little interaction and slow elementary school students in finding topics. This study designs and builds SiPERAL, an interactive Android e-book that turns a static PDF into a two-page flipbook and adds keyword search. Using the Research and Development method with the Waterfall model, the application was developed in Kotlin and Jetpack Compose. PdfRenderer renders the pages, HorizontalPager and Modifier.graphicsLayer create a three-dimensional page-flip animation, and the Linear Search algorithm matches keywords against the chapter list. Black box testing in two iterations found three failures (case-sensitive search, audio lifecycle, and page-boundary handling), which were fixed until all seven scenarios of the second iteration passed. The application shows a book-like visual effect, plays chapter audio explanations, and finds relevant chapters regardless of letter case, showing that a declarative interface combined with a simple search algorithm can improve the interactivity and accessibility of science learning media.

INTRODUCTION

Digital transformation in education calls for learning media that match the characteristics of elementary school students as digital natives, who prefer dynamic and interactive visual content because they grew up with digital communication technology (Prensky, 2001). Supplementary material such as the Comprehensive Natural Science Summary (*Rangkuman Pengetahuan Alam Lengkap*, RPAL) is nevertheless still limited to printed books or static PDF documents. Content that is delivered mainly as text, without interactivity, tends to be less effective and to lower students' interest in learning (Nuraini et al., 2024). Learning media should therefore act as an effective intermediary between the source of a message and its receiver

(Arsyad, 2014), and static material needs to be converted into a more immersive digital format.

A major technical obstacle of ordinary PDF-based e-books is the inefficiency of finding specific information in a document that holds hundreds of pages. Students who are used to instant access to information (Van Slyke, 2003) often struggle when they must flip or scroll through the pages one by one, and time that should be spent on understanding the material is spent on locating it. This condition calls for an automatic search facility that improves access to the material and supports active learning.

Several earlier works address parts of this problem (Table 1). Rizzi (2023) and Chirag (2023) show how PdfRenderer and Jetpack Compose can display PDF pages, but both present the pages as a simple vertical list without page-flip animation, text search, or audio. Subbarayudu et al. (2021) compare variants of Linear Search, yet stay at the level of algorithm analysis and do not integrate the algorithm into a mobile e-book. KITABOO (2024) is a complete commercial platform for ePub content with multimedia features, but it is not built on PDF rendering, does not offer a realistic three-dimensional flipbook, and is an enterprise solution. Based on this review, none of these works combines real-time PDF rendering, a two-page flipbook interface, a three-dimensional page-flip animation, audio feedback, and keyword search in one application built with a modern declarative interface toolkit.

Table 1. Comparison of related work and the present study

Study	Technology	Limitation / feature
Rizzi (2023)	PdfRenderer, Jetpack Compose, LazyColumn	Vertical single-page list; no spread view, page flip, search, or audio
Chirag (2023)	Jetpack Compose, Kotlin, PdfRenderer, AndroidView	Basic viewer; no animation, search, or specific search algorithm
Subbarayudu et al. (2021)	Linear Search, Two-Way Linear Search, LRMW	Theoretical analysis; no mobile or e-book integration
KITABOO (2024)	ePub, multimedia, cloud, DRM	ePub rather than PDF; no 3D flipbook; enterprise (paid) solution
This study	**PdfRenderer, Jetpack Compose, HorizontalPager, graphicsLayer, Linear Search**	**Two-page spread, 3D page flip, chapter search, audio**

This study therefore designs and builds SiPERAL, an interactive RPAL e-book for Android that uses Jetpack Compose to turn a static PDF into a two-page flipbook with

a realistic three-dimensional page-flip animation, and that applies the Linear Search algorithm to its keyword search feature. Linear Search examines each element sequentially until the target is found (Cormen et al., 2009). It is justified here because the number of searchable items is limited, so no heavy preprocessing is required (Knuth, 1998). The research questions are: (1) how to build an interactive Android RPAL e-book that turns a static PDF into a dynamic two-page flipbook; (2) how to integrate PdfRenderer with HorizontalPager to obtain a realistic page-flip animation through Modifier.graphicsLayer; and (3) how Linear Search can support information retrieval in the RPAL material. The scope is limited to the Android platform and to a single PDF embedded in the application assets, without content management or download features.

RESEARCH METHODS

Research approach and development model

This study uses the Research and Development (R&D) method, which produces a particular product and tests its effectiveness (Sugiyono, 2019). The product is an Android RPAL e-book application with a Linear Search feature and a three-dimensional page-flip animation. Development follows the Waterfall model, chosen for its systematic and sequential stages. In requirement analysis, the problems of students who use conventional RPAL books are analyzed and translated into functional requirements. In system design, the flowcharts, the user experience, and the interface layout are prepared as a blueprint. In implementation, the design is coded in Kotlin with Jetpack Compose. In testing, the application is validated with black box testing.

Tools and materials

The hardware consists of a laptop with an Intel Core i5 processor, 16 GB of RAM, and a 512 GB SSD, together with an Android smartphone for testing on a real device. The software consists of Microsoft Windows 11, Android Studio, the Kotlin language, and the Jetpack Compose toolkit with the PdfRenderer, MediaPlayer, and AudioManager APIs. The application targets Android 13 to 15 (Google Developers, 2024). The materials are the RPAL PDF document of 340 pages as the main e-book content, an MP3 page-turn sound effect, chapter audio explanation files, the Android developer documentation, and scientific literature on Linear Search and interactive e-books.

System requirements and architecture

The functional requirements are that the system loads and renders a PDF file; shows it as a two-page spread; changes pages with a three-dimensional page-flip animation when swiped; plays a sound effect when a page turns; provides autoplay with play and pause, zoom in and out, replay and skip; offers a volume control dialog; and searches

chapters by keyword. The architecture has six main components. *MainActivity* hosts the main composable. *SimplePdfViewerScreen* manages the main state (*PagerState*, play state, and zoom state) and the *PdfRenderer*. *PdfPageImage* renders a specific PDF page into a bitmap. *HorizontalPager* provides swipe navigation, and *Modifier.graphicsLayer* applies a rotation around the Y axis for the flip effect. *MediaPlayer* plays the page-turn sound whenever *pagerState.currentPage* changes, and *AudioManager* controls the system volume through a slider dialog. Because *PdfRenderer* needs a file descriptor, the PDF is first copied from the assets folder to the cache storage (Figure 1).

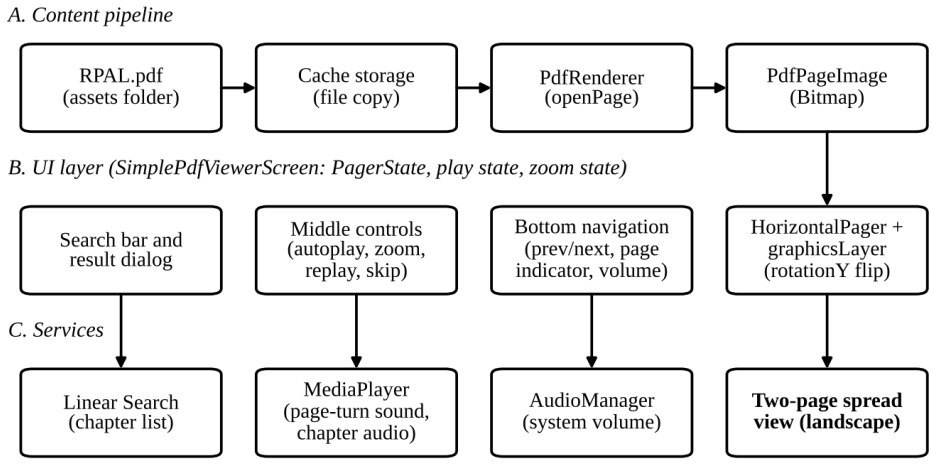


Figure 1. Architecture of SiPERAL: content pipeline, UI layer, and services

The interface is a single landscape screen with five areas: a header with the book title, a search bar, a content area with two PDF pages and a spine in the middle, middle controls (replay, zoom, play or pause, and skip), and a bottom navigation bar with previous and next buttons, the page indicator, and the volume button.

Design of the Linear Search algorithm

The search feature works on the list of chapter and sub-chapter titles of the RPAL book. When the user submits a keyword, the system (1) normalizes the keyword to lowercase so that the search is case-insensitive; (2) scans the chapter list from the first item to the last; (3) adds every chapter whose lowercase title contains the keyword to the result list; and (4) shows the result dialog if the list is not empty, or a short “not found” message otherwise. Selecting a result jumps directly to the corresponding page. Table 2 simulates the algorithm on a list of five chapter titles (Ecosystem, Energy, Force, Living Things, and Solar System) with the keyword “Gaya” (Force).

Table 2. Simulation of Linear Search with the keyword "Gaya" (Force)

Iteration (i)	Data[i]	Keyword (x)	Data[i] = x	Action
1	*Ekosistem*	*Gaya*	False	Continue
2	*Energi*	*Gaya*	False	Continue
3	*Gaya*	*Gaya*	True	Stop (found)
4	*Makhluk Hidup*	*Gaya*	–	Not executed
5	*Tata Surya*	*Gaya*	–	Not executed

The time complexity depends on the position of the target. In the best case the target is the first element and only one comparison is needed, O(1). In the worst case the target is the last element or is absent, and all N elements are compared, O(n). On average, the number of comparisons is

$$C_{\text{avg}} = (N + 1) / 2 \quad (1)$$

where C_{avg} is the expected number of comparisons and N is the number of searchable items. Table 3 applies these cases to $N = 50$ items. The average is 25.5, or about 26 comparisons, and the worst case is 50 comparisons, a workload that is very light for a mobile processor.

Table 3. Time complexity of Linear Search for N = 50 searchable items

Case	Position of the target	Complexity	Comparisons
Best case	First element (index 0)	O(1)	1
Average case	Random position	O(n)	(N + 1) / 2 ≈ 26
Worst case	Last element or absent	O(n)	50

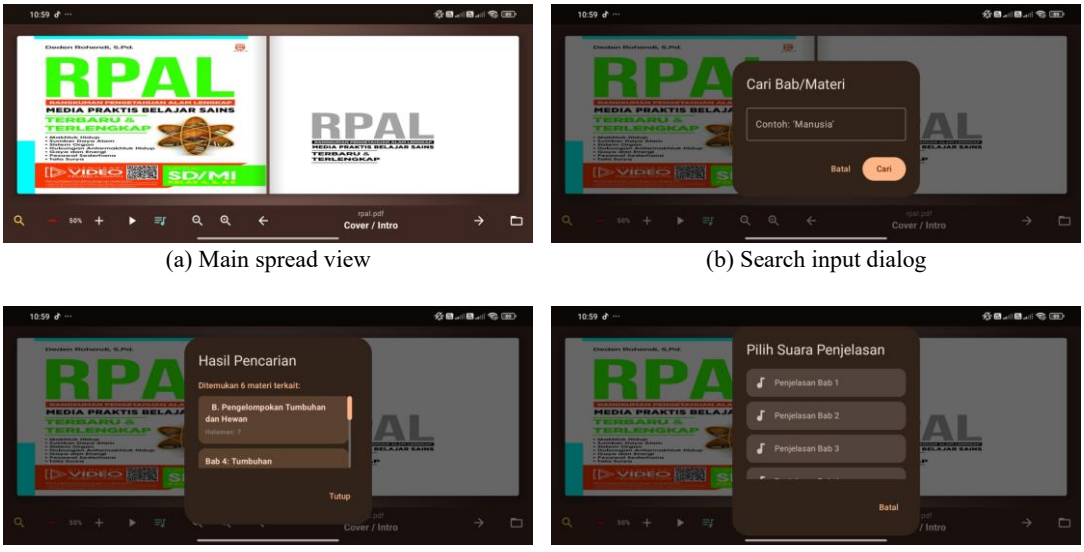
Testing

The application is tested with black box testing, which checks the input and output of each function without inspecting the internal code. Testing is done in two iterations. The first iteration runs on the early prototype to identify bugs, and the second runs after debugging. The scenarios cover opening the application, page navigation, search with a valid and with a non-existent keyword, the audio lifecycle, volume control, and zoom.

RESULTS AND DISCUSSION

Implementation

SiPERAL was implemented in Kotlin with Jetpack Compose in Android Studio and run on a physical Android device. Figure 2 shows the resulting interface. The main screen (Figure 2a) shows the RPAL book as a two-page spread in landscape orientation, with a control bar that holds the search button, the volume indicator, play and pause, the audio playlist, zoom in and out, the previous and next buttons, and the folder button. The three-dimensional page-flip effect is obtained by applying *rotationY* and *cameraDistance* in *Modifier.graphicsLayer* according to the *currentPageOffsetFraction* of the *PagerState*, which gives the illusion of a page folding in three-dimensional space while the user swipes. The page-turn sound is played by *MediaPlayer* each time the page changes.



(a) Main spread view (b) Search input dialog

(c) Search result dialog (d) Audio selection dialog

Figure 2. User interface of SiPERAL: (a) main spread view, (b) search input, (c) search results, and (d) audio selection

The search dialog (Figure 2b) takes a keyword and offers “Cari” (search) and “Batal” (cancel) buttons. The result dialog (Figure 2c) states how many related items were found and lists each matching chapter with its page number; tapping a card jumps to that page and starts the audio of the chapter. The audio dialog (Figure 2d) lists the chapter explanations, and the active track is marked with a play icon. Figure 3 shows the implementation of Linear Search. The function lowercases the query, iterates over the chapter list (*mappedChapters*) one by one, and stores every chapter whose lowercase title contains the query. It then opens the result dialog, or shows a short

message when no chapter matches. The complexity is O(*n*), where *n* is the number of chapters in the table of contents, which is below one hundred items.

```
fun performLinearSearch(context: Context, query: String) {
    if (query.isBlank()) return

    // Mengosongkan hasil pencarian sebelumnya
    searchResults.clear()
    val lowerQuery = query.lowercase() // Normalisasi teks ke huruf kecil

    // Proses Linear Search: Iterasi setiap bab dalam daftar
    for (chapter in mappedChapters) {
        // Jika judul bab mengandung kata kunci
        if (chapter.title.lowercase().contains(lowerQuery)) {
            // Tambahkan ke daftar hasil
            searchResults.add(chapter)
        }
    }

    // Logika penanganan hasil
    isSearchDialogOpen = false
    if (searchResults.isNotEmpty()) {
        isSearchResultsOpen = true // Tampilkan dialog hasil
    } else {
        Toast.makeText(context, "Materi tidak ditemukan.", Toast.LENGTH_SHORT)
    }
}
```

Figure 3. Implementation of the Linear Search algorithm in Kotlin

Black box testing

Table 4 summarizes the two test iterations. The first iteration showed three failures. First, searching for “manusia” in lowercase did not find the chapter written “Manusia”, because the comparison was case-sensitive. Second, the audio kept playing in the background after the application was closed. Third, pressing the next button repeatedly at the last page made the application crash with an index out of bounds error. The fixes were to apply *lowercase()* to both the query and the target text, to release the *MediaPlayer* in *onPause()* and *onDispose*, and to check that the current page is smaller than the total number of pages before moving on. In the second iteration, all seven scenarios passed.

Table 4. Black box test results over two iterations

No.	Test scenario	Expected result	Iteration 1	Iteration 2
1	Open the application for the first time	PDF loads from assets; pages 1–2 are shown	–	Pass
2	Page navigation with the left and right arrows	Pages change with flip animation; indicator updates	–	Pass
3	Search with lowercase keyword “manusia”	Chapters containing “Manusia” or “manusia” are listed	Fail: “not found”	Pass
4	Search with non-existent keyword “xyz”	Message “Materi tidak ditemukan” is shown	–	Pass
5	Play audio, then close or minimize the app	Audio stops automatically	Fail: audio kept playing	Pass
6	Volume control (+/–)	Volume indicator and device volume change	–	Pass
7	Zoom in	Page is enlarged and text is clearer	–	Pass
8	Press “Next” repeatedly up to the last page	Button is disabled or ignored at the last page	Fail: crash	–

– = not tested in that iteration.

Discussion

Flipbook interface. SiPERAL turns a static PDF into an interactive medium. Unlike standard PDF viewers that rely on vertical scrolling, it applies a skeuomorphic design that imitates a physical book through the three-dimensional page flip. Keeping the familiar left page and right page structure is expected to ease the transition of students from printed books to digital ones, and the landscape orientation gives more room for text and pictures so that manual zooming is needed less often. Although bitmaps are rendered from the PDF in real time, no noticeable lag was observed during the page-flip animation, which agrees with the passed navigation test.

Search performance. Because the chapter list is small (below one hundred items), the $O(n)$ cost of Linear Search is negligible, and no perceptible delay was observed between pressing the search button and the appearance of the result dialog. The case-insensitive fix made in the second iteration means that students do not have to remember how a chapter title is capitalized. The feature also changes access to the material from sequential to direct: instead of flipping through many pages to reach a

late chapter, the student jumps to it in one step. A limitation is that the search covers chapter and sub-chapter titles and not the full text of the pages. Full-text search, or a much larger collection, would call for indexing or a more efficient search method.

Audio integration. The chapter audio explanations offer an alternative for students with an auditory learning style or students who tire of reading long text, and the play, pause, volume, and track selection controls let them set their own pace. This multi-sensory feedback is in line with the principles of multimedia learning (Mayer, 2009). On the technical side, the lifecycle test confirmed that the audio now stops when the application is minimized or closed, which prevents memory leaks and unnecessary battery use.

Comparison with other solutions. Compared with a printed book, SiPERAL is more portable and adds automatic search. Compared with the basic PDF viewers described by Rizzi (2023) and Chirag (2023), it adds the two-page flip effect, integrated audio, and navigation organized by chapter instead of page number only. Compared with a commercial platform such as KITABOO (2024), it works directly on PDF content with the native PdfRenderer API.

Limitations. Validation in this study is limited to functional (black box) testing by the developers. User acceptance testing with students and teachers has not yet been carried out, so the effect of the application on learning interest and motivation is still an expectation and not a measured result.

CONCLUSION

This study designed and built SiPERAL, an Android RPAL e-book application based on Jetpack Compose that shows a PDF as a two-page flipbook with an interactive three-dimensional page-flip animation. The Linear Search algorithm applied to the chapter list proved effective in helping users find the desired chapter quickly and accurately, and the supporting features, namely chapter audio, autoplay, zoom, and volume control, make the learning experience more interactive. The results show that a modern declarative interface combined with a simple search algorithm can raise the interactivity and accessibility of science learning media for elementary school students. The main limitations are the title-based search, the use of a single PDF, and the absence of user acceptance testing. Future work should add full-text search or an automatic index for larger documents, interactive quizzes or self-assessment, support for multiple PDF files and for other platforms such as iOS and the web, and a user study with students and teachers to measure the effect on learning.

REFERENCES

Arsyad, A. (2014). *Media pembelajaran*. Jakarta: PT Raja Grafindo Persada.

Chirag, T. (2023). *Reading PDF files made easy with Jetpack Compose*. Medium Engineering Blog. https://medium.com/

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). Cambridge, MA: MIT Press.

Google Developers. (2024). *Android developers documentation: Jetpack Compose & PdfRenderer*. https://developer.android.com/

KITABOO. (2024). *Interactive eBook reader app for education: Features and benefits*. Kitaboo Digital Publishing Platform.

Knuth, D. E. (1998). *The art of computer programming, Volume 3: Sorting and searching* (2nd ed.). Boston, MA: Addison-Wesley Professional.

Mayer, R. E. (2009). *Multimedia learning* (2nd ed.). Cambridge: Cambridge University Press.

Nuraini, et al. (2024). Analisis pengembangan keterampilan praktik dan berpikir ilmiah dalam buku teks sains. *Jurnal Ilmiah Pendidikan Dasar*.

Prensky, M. (2001). Digital natives, digital immigrants part 1. *On the Horizon, 9*(5), 1–6.

Rizzi, G. (2023). *How to show a PDF with Jetpack Compose*. ProAndroidDev. https://proandroiddev.com/

Subbarayudu, B., et al. (2021). Optimized search functionality with linear search algorithm. *International Journal of Computer Science and Mobile Computing, 10*(5).

Sugiyono. (2019). *Metode penelitian kuantitatif, kualitatif, dan R&D*. Bandung: Alfabeta.

Van Slyke, T. (2003). Digital natives, digital immigrants: Some thoughts from the generation gap. *The Technology Source*.