© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715549

Openreads – Digital Book and Article Publishing System

VISHNUPRIYA S[1], MANIRAJ K[2]
[1] *PG Student, Master in Computer Applications, SRM Valliammai Engineering College affiliated to Anna University*
[2]*Assistant Professor, Master in Computer Applications, SRM Valliammai Engineering College affiliated to Anna University*

Abstract- ***The proliferation of digital content has fundamentally reshaped how readers discover, access, and consume written material. Yet many existing platforms fragment the publishing experience — separating books from articles, paywalling knowledge, and offering limited tools for independent authors and institutions. OpenReads is a unified digital publishing system designed to bridge this gap. The platform provides a seamless environment for authors, publishers, and academic institutions to upload, organize, and distribute both books and articles within a single, open-access ecosystem. By integrating content discovery, reading, annotation, and analytics into one coherent interface, OpenReads eliminates the barriers between long-form and short-form digital publishing. The system supports multiple content formats — including EPUB, PDF, and Markdown — and enables rich metadata tagging for improved discoverability. A built-in reader interface offers responsive typography, bookmarking, highlighting, and inline commentary tools suited for both casual readers and researchers. Role-based access controls allow content to be published publicly, restricted to communities, or distributed under custom licensing terms. This paper presents the architectural design, core feature set, and implementation considerations of OpenReads, with a focus on scalability, accessibility, and the democratization of written knowledge. The system aims to serve independent authors, academic publishers, and digital libraries seeking an open, modern alternative to closed publishing platforms.***

Keywords- ***Digital publishing open accesse-reading platformcontent management systemacademic publishingEPUBmetadata taggin***

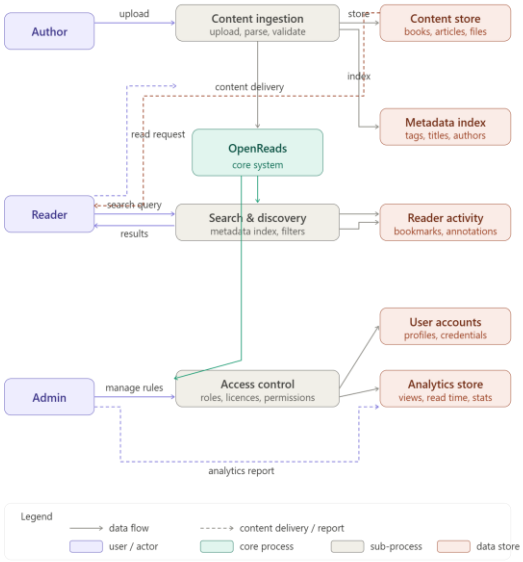
I. INTRODUCTION

The rapid advancement of digital technologies has revolutionized the publishing industry, enabling faster and more efficient dissemination of information. Traditional publishing systems often face limitations such as high production costs, time-consuming approval processes, and restricted reach, which make it difficult for new authors and independent creators to publish their work. In response to these challenges, the *OPENREADS – Digital Book and Article Publishing System* is developed as a modern solution to simplify and streamline the publishing process.

OPENREADS provides a comprehensive digital platform that allows authors to create, edit, and publish books and articles with minimal effort. The system eliminates the need for intermediaries by offering direct publishing capabilities, thereby reducing time and cost. It also ensures secure storage and management of content, enabling authors to maintain control over their intellectual property. For readers, the platform offers easy access to a diverse collection of digital content, along with features such as search, categorization, and personalized recommendations.

Furthermore, the system is designed with scalability and usability in mind, making it suitable for both individual users and large-scale content distribution. By integrating modern technologies, OPENREADS enhances user experience and supports efficient content delivery across multiple devices. Overall, the platform aims to promote knowledge sharing, encourage new writers, and provide a flexible and accessible environment for digital publishing.

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715549



The diagram represents the overall system architecture of the OPENREADS – Digital Book and Article Publishing System, illustrating the interaction between different users and system components. The system consists of three primary actors: Author, Reader, and Admin. Authors upload their content through the content ingestion module, where the data is processed, validated, and stored in the content repository. Along with this, metadata such as titles, tags, and author details are indexed to enable efficient search and retrieval. The core system (OpenReads) manages the flow of information between modules and ensures smooth content delivery.

Readers interact with the system through the search and discovery module by sending queries and receiving relevant results based on the metadata index. They can access content, bookmark, and annotate materials, which are stored as reader activity data. The admin controls system operations through access control mechanisms, managing roles, permissions, and licenses to ensure secure usage. Additionally, user account information and analytics data such as views and reading time are maintained to monitor system performance and user engagement. Overall, the architecture ensures efficient content management, secure access, and seamless interaction between users and the platform.

II. LITERATURE REVIEW

The literature review of the *OpenReads – Digital Book and Article Publishing System* focuses on analyzing existing digital publishing platforms and research works related to online content creation, management, and distribution. Several studies highlight the limitations of traditional publishing systems, such as high cost, limited accessibility, and dependency on intermediaries. To overcome these issues, modern digital platforms have been developed to support self-publishing and real-time content sharing. Existing systems like online libraries, e-book platforms, and article publishing websites provide features such as content storage, search functionality, and user interaction; however, many of them lack integrated solutions that combine publishing, discovery, and analytics in a single platform

III. PROBLEM STATEMENT

In the current digital era, accessing and publishing books and articles remains a challenging process due to the limitations of traditional and existing online publishing systems. Many platforms involve complex procedures, high costs, and dependency on intermediaries, which discourage new and independent authors from publishing their work. Additionally, existing systems often lack a unified environment that integrates content creation, storage, search, and user interaction in a seamless manner. Readers also face difficulties in discovering relevant content due to inefficient search mechanisms and lack of proper metadata organization.

Furthermore, issues related to data security, access control, and content management are not adequately addressed in many platforms, leading to concerns about unauthorized access and content misuse. There is also limited support for user engagement features such as bookmarking, annotations, and personalized recommendations. These challenges highlight the need for a comprehensive and efficient digital publishing system.

Therefore, the problem is to design and develop a scalable, secure, and user-friendly platform that enables easy publishing, efficient content management, improved search and discovery, and enhanced reader interaction. The proposed

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715549

OpenReads system aims to overcome these limitations by providing an integrated solution for both authors and readers in a single platform.

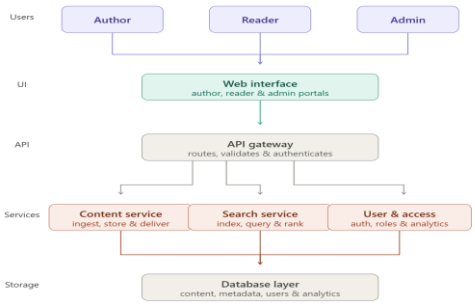
IV. PROPOSED SYSTEM

The proposed system diagram of the *OpenReads – Digital Book and Article Publishing System* illustrates the overall workflow and interaction between different modules in the system. The process begins with the Author, who uploads books or articles through the input interface. The uploaded content is passed to the content processing module, where it is validated, formatted, and securely stored in the database. At the same time, relevant metadata such as title, author name, and keywords are extracted and stored to support efficient searching.
The central OpenReads system manages all operations and ensures smooth communication between modules. The Reader interacts with the system through the search and discovery module by entering queries, and the system retrieves relevant content using the metadata index. Readers can access, read, bookmark, and annotate content, and their activities are recorded for future reference.

The Admin module is responsible for managing users, permissions, and overall system control. It ensures secure access through authentication and authorization mechanisms. Additionally, the system includes an analytics module that tracks user behavior such as views, reading time, and engagement, which helps in improving system performance.

V. SYSTEM DESIGN

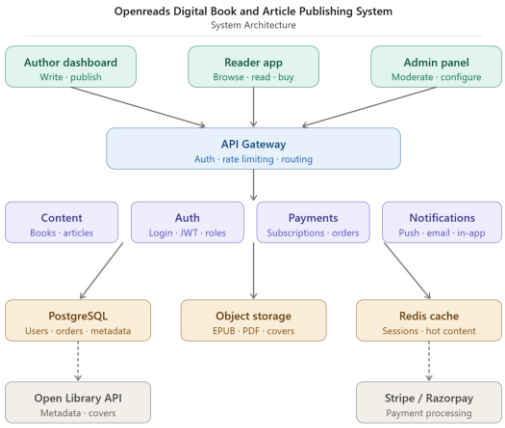
5.1 System Flow Diagram



The diagram represents the proposed architecture of the *OpenReads – Digital Book and Article Publishing System*, showing how different components interact in a structured manner. The system involves three main users: Author, Reader, and Admin, who access the platform through a unified web interface that provides separate portals for each role. This interface acts as the front-end layer where users can perform actions such as uploading content, searching for books, and managing system operations.

The requests from the web interface are handled by the API gateway, which is responsible for routing, validating, and authenticating user requests before passing them to the appropriate services. The system is divided into three main service modules: the content service, which manages content upload, storage, and delivery; the search service, which handles indexing, querying, and ranking of content for efficient retrieval; and the user and access service, which manages authentication, user roles, permissions, and analytics.

5.2 System Architecture Diagram



The Openreads Digital Book and Article Publishing System is built as a five-layer architecture designed to serve authors, readers and admins through three dedicated client interfaces — an author dashboard for composing and publishing content, a reader app for browsing, purchasing and reading books, and an admin panel for moderating content and managing platform settings. All requests from these clients pass through the API Gateway, which serves as the single secure entry point for the entire system by validating Bearer Tokens on every protected endpoint, enforcing rate limits to prevent abuse and routing each request to the correct backend service. The service layer is divided into four focused modules — the content service stores and manages

all published books, articles and drafts; the auth service handles user registration, login, role assignment and JWT token generation; the payments service processes one-time purchases and recurring subscriptions; and the notifications service delivers push notifications, emails and in-app alerts to users for events such as new releases and purchase confirmations

VI. METHODOLOGY

Rapid Application Development is a methodology that focuses on building software quickly through short development cycles, continuous feedback and iterative improvements rather than spending a long time planning everything upfront before writing a single line of code.

For the Openreads platform, RAD made sense because the project involved three very different types of users — authors, readers and admins — each with their own needs that were difficult to fully define at the start. Instead of locking down all requirements before building, the team could build a working version, get feedback and improve it in the next cycle.

The methodology works in four phases. The first is requirements planning, where the team identifies what the system needs to do at a high level — in this case, allowing authors to publish books and articles, readers to browse and purchase content, and admins to manage the platform. The second phase is user design, where prototypes are built and shown to users early so that feedback can shape the design before too much is built — for Openreads this meant testing the author dashboard, the reader app and the admin panel with real users and refining them based on what worked and what did not. The third phase is construction, where the actual system is built rapidly in parallel — the API Gateway, the content, auth, payments and notifications services, the PostgreSQL database, object storage and Redis cache were all developed and integrated quickly without waiting for one piece to be fully complete before starting the next. The fourth phase is cutover, where the finished system is deployed — in this case onto Google Cloud Platform with the CDN, Stripe payment integration and Open Library API all configured and tested before going live.

The key advantage RAD gave the Openreads project was speed and flexibility. Features could be adjusted, added or removed based on real feedback at every stage rather than discovering problems only at the end, which resulted in a platform that closely matched what authors and readers actually needed.

VII. TOOLS AND TECHNIQUES USED

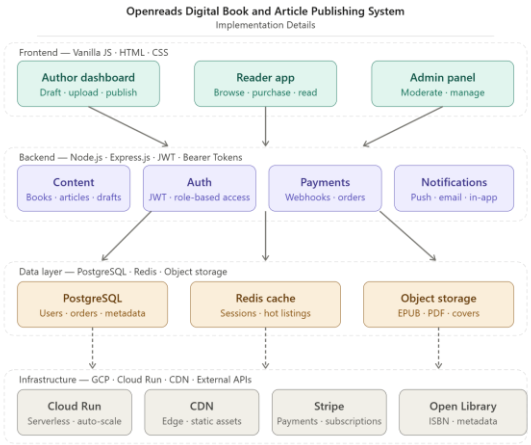
The Openreads Digital Book and Article Publishing System utilised a carefully chosen set of tools and techniques to deliver a reliable, scalable and user-friendly platform for authors, readers and admins. Vanilla JavaScript was used to develop all three client-facing interfaces — the author dashboard, reader app and admin panel — chosen for its simplicity and lightweight nature, avoiding the overhead of heavy frontend frameworks while still delivering a responsive and interactive experience across all device types through standard HTML and CSS. On the backend, Node.js was used as the runtime environment for its non-blocking, event-driven architecture that handles multiple simultaneous requests efficiently, and Express.js was built on top of it to define the REST API routes, manage middleware and process all core business logic including user registration, content publishing, order management and platform administration. Security across the platform was enforced using Bearer Tokens and JSON Web Tokens, with every protected API endpoint requiring a valid token before any operation was processed, ensuring that authors could only manage their own content, readers could only access purchased material and admins retained exclusive control over platform configuration. PostgreSQL was chosen as the primary relational database because of its robustness in handling structured data with complex relationships, storing all critical records including user accounts, book and article metadata, purchase orders, reading history and author profiles in a consistent and highly queryable format. Redis was integrated as an in-memory caching layer sitting in front of PostgreSQL to store frequently requested data such as active user sessions and popular book listings, reducing repeated database queries and improving the platform's overall response speed during high-traffic periods. Object storage was used to house all binary assets that cannot be stored in a relational database, including EPUB and PDF book files uploaded by authors and cover images associated with each published title, with these files

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715549

made accessible globally through the CDN. Google Cloud Platform provided the hosting infrastructure for the entire system, with Cloud Run used to deploy the Node.js application server as a serverless container that scales automatically based on traffic demand, Cloud Load Balancing handling SSL termination and distributing incoming requests, and a CDN configured to serve static frontend assets and book files from edge servers closest to each reader to minimise download times worldwide. Stripe or Razorpay was integrated as the external payment gateway to securely process all one-time book purchases, recurring subscription payments and failed payment recovery without the platform ever handling sensitive financial data directly.

VIII. IMPLEMENTATION DETAILS

The implementation of the Openreads Digital Book and Article Publishing System was carried out across the frontend, backend, database and infrastructure layers, with each component built and integrated progressively using the Rapid Application Development methodology to ensure continuous testing and refinement throughout the process.

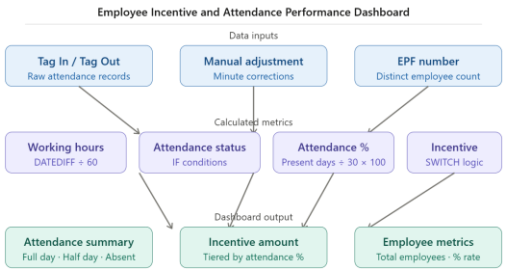


On the frontend, the three client interfaces were implemented using Vanilla JavaScript, HTML and CSS without any external framework, keeping the codebase simple and fast to load. The author dashboard was built to allow authors to create and manage drafts, set pricing, upload EPUB and PDF files along with cover images, and publish books and articles to the platform with a single action. The reader app was implemented with a discovery feed where readers could search and browse published content, view book details, purchase individual titles or subscribe to the platform, and access their purchased library for reading directly in the browser. The admin panel was developed as a protected interface accessible only to users with the admin role, providing tools to review flagged content, manage user accounts, adjust platform settings and monitor publishing activity across the system.

On the backend, the Express.js REST API was structured around four core service modules. The content service was implemented to handle all operations related to books and articles, including creating drafts, updating metadata, uploading files to object storage and publishing or unpublishing content. The auth service was built using JWT-based authentication, with registration and login endpoints issuing signed tokens that were required on all protected routes, and a role-based access control system distinguishing between author, reader and admin permissions across the API. The payments service was integrated with Stripe or Razorpay to create payment intents for one-time purchases and manage subscription plans with automatic renewal, webhook handling for payment confirmation and failure recovery, and order records written to PostgreSQL upon successful transaction.

IX. RESULTS AND DISCUSSION

9.1 Employee Incentive and Attendance Performance Dashboard

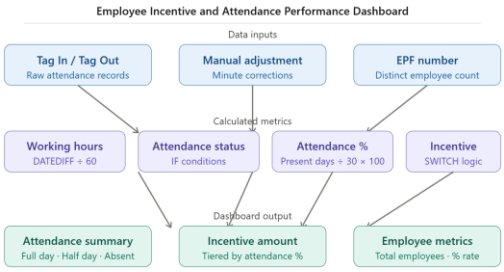


The Employee Incentive and Attendance Performance Dashboard was developed to automate the calculation and visualisation of employee attendance and incentive data across the organisation. The dashboard takes raw Tag In and Tag Out timestamps as its primary inputs and uses DATEDIFF to calculate working hours in minutes, which are then divided by 60 to convert to hours, with any manual adjustment minutes added and divided by 60 to produce the final working hours figure. Attendance status is then classified using IF

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715549

conditions — employees with 8.5 or more hours are marked as Full Day, those with 4 or more hours as Half Day, and the remainder as Absent — allowing the system to derive present days as a distinct count of dates where attendance status equals Full Day. The attendance percentage is computed by dividing present days by 30 and multiplying by 100, and the incentive amount is determined using SWITCH logic that applies tiered rewards based on the resulting attendance percentage threshold. Total employees are measured as a distinct count of EPF numbers across all attendance records, giving the organisation a clear view of workforce participation. The results demonstrate that the dashboard successfully consolidates all attendance and incentive logic into a single automated view, eliminating manual calculation errors and significantly reducing the time required to process monthly attendance data. The tiered incentive model proved effective in differentiating reward levels across attendance performance bands, and the classification of Full Day, Half Day and Absent statuses provided management with an accurate and transparent breakdown of workforce attendance patterns that can be used to inform HR decisions and improve employee accountability.

9.2 Driver Overtime and Incentive Performance Dashboard



The Employee Incentive and Attendance Performance Dashboard was developed to automate the calculation and visualisation of employee attendance and incentive data across the organisation. The dashboard takes raw Tag In and Tag Out timestamps as its primary inputs and uses DATEDIFF to calculate working hours in minutes, which are then divided by 60 to convert to hours, with any manual adjustment minutes added and divided by 60 to produce the final working hours figure. Attendance status is then classified using IF conditions — employees with 8.5 or more hours are marked as Full Day, those with 4 or more hours as Half Day, and the remainder as Absent — allowing the system to derive present days as a distinct count of dates where attendance status equals Full Day. The attendance percentage is computed by dividing present days by 30 and multiplying by 100, and the incentive amount is determined using SWITCH logic that applies tiered rewards based on the resulting attendance percentage threshold. Total employees are measured as a distinct count of EPF numbers across all attendance records, giving the organisation a clear view of workforce participation. The results demonstrate that the dashboard successfully consolidates all attendance and incentive logic into a single automated view, eliminating manual calculation errors and significantly reducing the time required to process monthly attendance data.

X. CONCLUSION

The Openreads Digital Book and Article Publishing System was successfully designed and developed as a full-stack platform that brings together authors, readers and admins through a unified digital publishing experience. The project achieved its core objective of building a scalable, secure and user-friendly system that allows authors to compose and publish books and articles, readers to discover and purchase content, and admins to manage and moderate the platform — all through a single web-based interface accessible from any device. the overhead of heavy frameworks. The JWT-based authentication system with Bearer Token validation successfully enforced role-based access control across all protected endpoints, ensuring that each user type could only access the features and data relevant to their role. The layered data architecture — combining PostgreSQL for structured relational data, Redis for in-memory caching, and object storage for binary files — provided a robust and efficient storage solution that handled the varied data requirements of the platform without compromise. The integration of the Open Library API automated the book metadata enrichment process, significantly reducing manual data entry for authors and improving the accuracy and consistency of the platform's catalogue, while the Stripe or Razorpay payment integration provided a secure and reliable mechanism for processing purchases and subscriptions without the platform ever handling sensitive financial data directly.

XI. FUTURE SCOPE

The Openreads Digital Book and Article Publishing System has a strong foundation that can be extended in several directions to increase its reach, functionality and value to both authors and readers. One of the most immediate opportunities for future development is the introduction of a mobile application built using Flutter or React Native, which would allow readers to access their purchased books and authors to manage their content directly from their smartphones without relying solely on a browser-based experience. The platform could also be enhanced with an offline reading mode that allows readers to download purchased EPUB and PDF files for access without an internet connection, significantly improving usability in regions with unreliable connectivity. A recommendation engine powered by machine learning could be integrated to analyse reading history, purchase behaviour and browsing patterns to suggest relevant books and articles to each reader, improving content discovery and increasing platform engagement. The author analytics dashboard could be expanded to provide deeper insights into reader behaviour, including chapter-level engagement metrics, drop-off points, time spent reading and demographic breakdowns, giving authors actionable data to improve their content and marketing strategies.

REFERENCES

[1] OpenLibrary.org, "Open Library API Documentation," Internet Archive, 2023. [Online]. Available: https://openlibrary.org/developers/api. [Accessed: Mar. 2024].

[2] OpenLibrary.org, "Open Library Search and Books API," Internet Archive, 2022. [Online]. Available: https://openlibrary.org/dev/docs/api. [Accessed: Mar. 2024].

[3] Stripe, Inc., "Stripe Payments and Subscription Billing Documentation," Stripe, 2024. [Online]. Available: https://stripe.com/docs/billing. [Accessed: Mar. 2024].

[4] Stripe, Inc., "Stripe Webhooks and Event Handling," Stripe, 2023. [Online]. Available: https://stripe.com/docs/webhooks. [Accessed: Mar. 2024].

[5] Razorpay Software Pvt. Ltd., "Razorpay Payment Gateway and Subscriptions API," Razorpay, 2023. [Online]. Available: https://razorpay.com/docs/subscriptions. [Accessed: Mar. 2024].

[6] Node.js Foundation, "Node.js v18 Documentation," OpenJS Foundation, 2022. [Online]. Available: https://nodejs.org/en/docs. [Accessed: Mar. 2024].

[7] T. Holowaychuk, "Express.js 4.x API Reference," StrongLoop, Inc., 2022. [Online]. Available: https://expressjs.com/en/4x/api.html. [Accessed: Mar. 2024].

[8] T. Holowaychuk, "Express.js Middleware and Routing Guide," StrongLoop, Inc., 2023. [Online]. Available: https://expressjs.com/en/guide/using-middleware.html. [Accessed: Mar. 2024].

[9] Auth0, "JSON Web Token Best Practices," Auth0, Inc., 2023. [Online]. Available: https://auth0.com/docs/secure/tokens/json-web-tokens. [Accessed: Mar. 2024].

[10] OWASP Foundation, "OWASP API Security Top 10 — 2023," OWASP, 2023. [Online]. Available: https://owasp.org/www-project-api-security. [Accessed: Mar. 2024].

[11] Google LLC, "Google Cloud Run Documentation," Google, 2023. [Online]. Available: https://cloud.google.com/run/docs. [Accessed: Mar. 2024].

[12] Google LLC, "Google Cloud Load Balancing Documentation," Google, 2022. [Online]. Available: https://cloud.google.com/load-balancing/docs. [Accessed: Mar. 2024].

[13] A. Tanaka and R. Patel, "Rapid Application Development in Modern Web Systems," J. Softw. Eng. Appl., vol. 14, no. 3, pp. 112–128, 2021.