RSS Fireside

A Project Report Submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

Of

Bachelor of Technology

in

Computer and Communication Engineering

by

Aadit Agrawal (230953344), Prathamesh Aggarwal (230953314), Faisal Aziz (230953354)

Under the guidance of

Dr. Akshay K C
Assistant Professor – Senior Scale
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India

Mrs. Swetha Pai
Assistant Professor
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Unit of MAHE, Manipal

March 2025

ABSTRACT

The Fireside RSS Reader is a comprehensive web-based application designed to streamline the consumption and organization of online content through RSS (Really Simple Syndication) feeds. This application addresses the challenges of information overload and scattered content sources by providing a centralized platform for users to subscribe to, categorize, and interact with content from various publishers. Fireside implements a sophisticated database architecture that manages users, feeds, publications, and interaction data while offering personalized recommendations based on user behavior and preferences. Key features include multi-source feed aggregation, content categorization, interactive note-taking, and an intelligent recommendation system that leverages both taxonomy and folksonomy approaches. The application's architecture ensures scalability and performance through normalized database design while maintaining an intuitive user experience. This project demonstrates the effective integration of content management systems with personalized information delivery mechanisms to enhance digital content consumption.

ACM Classification Keywords:

[Information systems]: Information retrieval; Content analysis and feature selection; Personalization; Recommender systems

[Human-centered computing]: Interactive systems and tools; User interface management systems; Social recommendation

[Software and its engineering]: Software organization and properties; Software system structures

[Information systems]: World Wide Web; Web applications; Web mining

Sustainable Development Goal [SDG]: Industry, Innovation and Infrastructure (Goal 9) - Building resilient infrastructure, promoting inclusive and sustainable industrialization, and fostering innovation through improved information access and digital infrastructure.

Table of Contents

Chapter 1: Introduction ………………………………………………………4
Chapter 2: Literature Survey/ Background …………………………………4
Chapter 3: Objectives/ Problem Statement ………………………………….5
Chapter 4: Data Design ……………………………………………………7
Chapter 5: Methodology …………………………………………………..21
Chapter 6: Results ………………………………………………………25
Chapter 7: Conclusion and Future Work …………………………………...33
Chapter 8: References ……………………………………………………..34

1. Introduction

The Fireside RSS Reader is a modern web application designed to streamline the consumption of online content by aggregating RSS and Atom feeds into a single, user-friendly interface. The application enables registered users to seamlessly subscribe to various feeds, view aggregated articles, and manage their subscriptions. This project focuses on designing a robust backend database to handle user accounts, feed metadata, feed items, and the associated relationships between these entities. The design challenges include ensuring data integrity, eliminating redundancy, and guaranteeing optimal query performance through proper normalization and reduction techniques

2. Literature Survey/Background

The development of RSS (Really Simple Syndication) readers has been a significant evolution in web content aggregation, enabling users to efficiently access updates from multiple sources in a standardized format. RSS technology, first formalized in the late 1990s, was designed to facilitate the syndication of web content, allowing users and applications to receive timely updates without the need to manually check individual websites [1].

The foundational RSS 2.0 specification, detailed in the RSS Advisory Board’s white paper, established a simple XML-based structure that has since been widely adopted for news, blogs, and other frequently updated sites [2].

Over the years, numerous academic and industry studies have examined the architecture and usability of RSS readers. Early research focused on the technical aspects of feed parsing, data storage, and user interface design, highlighting the importance of efficient data normalization and schema reduction for scalable performance. Subsequent works explored the integration of security best practices, such as secure authentication and session management, which are now considered essential for any web-based aggregator.

Recent advancements in web frameworks (e.g., React, Next.js) and database systems (e.g., MariaDB) have further enhanced the capabilities of RSS readers, enabling real-time data fetching, responsive interfaces, and robust backend management. The literature also emphasizes the need for rigorous normalization—from 1NF to BCNF—to ensure data integrity and minimize redundancy in relational database design.

In summary, the background research underscores the critical role of standards compliance, secure design, and normalized data structures in building reliable and user-friendly RSS reader applications.

3. Objectives/ Problem Statement

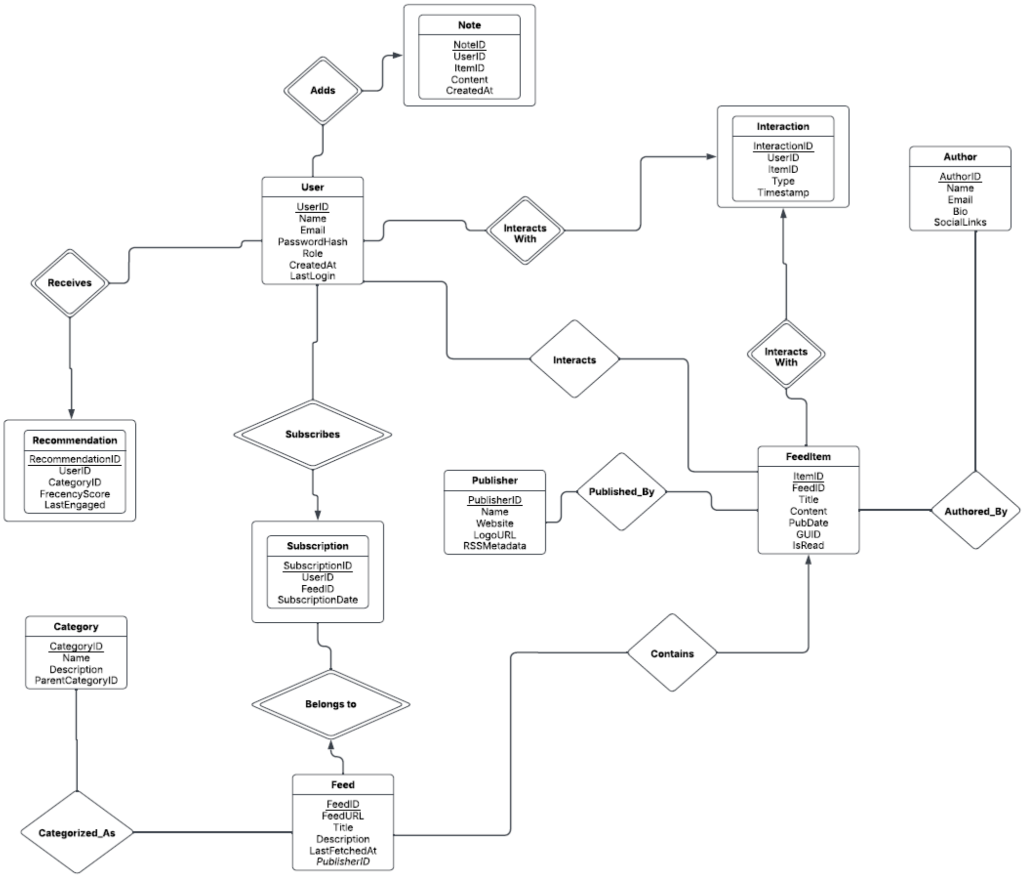
- To design and implement a robust, scalable, and secure database for the Fireside RSS Reader web application, enabling efficient aggregation and management of RSS/Atom feeds for users.
- To ensure seamless user authentication and secure session management, using industry-standard password hashing and secure, HTTP-only cookies for session tokens.
- To provide users with the ability to subscribe to, manage, and view multiple RSS/Atom feeds from a unified dashboard, supporting core operations such as adding, deleting, and updating subscriptions.
- To structure the database schema using entity-relationship modelling, reduction, and rigorous normalization (up to BCNF), thereby minimizing redundancy, ensuring data integrity, and supporting efficient query performance.
- To implement support for additional features such as tracking user interactions with feed items (e.g., marking as read/unread, notes, recommendations), and maintaining relationships between feeds, authors, publishers, and categories.
- To meet nonfunctional requirements including fast API response times, high reliability, maintainability, and adherence to security best practices

(e.g., input validation, prevention of SQL injection and XSS, secure handling of external feed URLs).

- To ensure the system is portable across standard Node.js hosting environments, interoperable with standard RSS/Atom formats, and maintainable for future enhancements or scaling.
- To address business rules such as independent user subscription lists, centralized feed metadata, and per-user feed item status tracking.
- To provide a foundation for future enhancements, such as background feed refreshing, improved user-specific read status, and advanced recommendation features.

4. Data Design

4.1 ER Diagram



4.2 Reduction Schemas

Step 1: Identify Strong Entities

Users
Attributes:

- UserID (Primary Key)
- Name
- Email
- PasswordHash
- Role
- CreatedAt
- LastLogin

Feeds
Attributes:

- FeedID (Primary Key)
- FeedURL
- Title
- Description
- LastFetchedAt
- PublisherID (Optional foreign key reference)

FeedItems
Attributes:

- ItemID (Primary Key)
- FeedID (Foreign Key to Feeds)
- Title
- Content
- PubDate
- GUID
- IsRead

Authors
Attributes:

- AuthorID (Primary Key)
- Name
- Email
- Bio
- SocialLinks

Publishers
Attributes:

- PublisherID (Primary Key)
- Name
- Website
- LogoURL
- RSSMetadata

Categories
Attributes:

- CategoryID (Primary Key)
- Name
- Description
- ParentCategoryID

Step 2: Identify Weak Entities

Subscriptions
Attributes:

- SubscriptionID (Primary Key)
- UserID (Foreign Key to Users)
- FeedID (Foreign Key to Feeds)
- SubscriptionDate

Interactions
Attributes:

- InteractionID (Primary Key)
- UserID (Foreign Key to Users)
- ItemID (Foreign Key to FeedItems)
- Type (e.g., like, share, note)
- Timestamp

Notes
Attributes:

- NoteID (Primary Key)
- UserID (Foreign Key to Users)
- ItemID (Foreign Key to FeedItems)
- Content
- CreatedAt

Recommendations
Attributes:

- RecommendationID (Primary Key)
- UserID (Foreign Key to Users)
- CategoryID (Foreign Key to Categories)
- FrecencyScore
- LastEngaged

Step 3: Define Many-to-Many Junction Tables

FeedItemAuthors

Composite Primary Key: (ItemID, AuthorID)
Purpose: Associate FeedItems with multiple Authors.

FeedItemPublishers

Composite Primary Key: (ItemID, PublisherID)
Purpose: Associate FeedItems with multiple Publishers if needed.

Feed_Categories

Composite Primary Key: (FeedID, CategoryID)
Purpose: Link Feeds with multiple Categories.

User_FeedItems (Optional)

Composite Primary Key: (UserID, ItemID)
Purpose: Capture additional relationships between Users and FeedItems (e.g., saved or bookmarked items).

Step 4: Address Multivalued Attributes

Handling the Role Attribute in Users:

Since a user can have multiple roles, decompose this attribute into two separate tables:

1. **Users:** (UserID, Name, Email, PasswordHash, CreatedAt, LastLogin)
2. **User_Role:** (UserID, Role)

Step 5: Implement One-to-Many (1:N) Relationships

Subscriptions: A user may have multiple subscriptions.
Implementation: Include UserID (foreign key) in the Subscriptions table.

FeedItems: A feed contains many items.
Implementation: Include FeedID (foreign key) in the FeedItems table.

Interactions and Notes: A user can interact with or add notes to many feed items.
Implementation: Include both UserID and ItemID as foreign keys in the Interactions and Notes tables, respectively.

Recommendations: A user may have recommendations for several categories.
Implementation: Include UserID (and CategoryID) as foreign keys in the Recommendations table.

Step 6: Implement Many-to-Many (M:N) Relationships

FeedItemAuthors: Create a junction table with a composite key (ItemID, AuthorID).

FeedItemPublishers: Create a junction table with a composite key (ItemID, PublisherID).

Feed_Categories: Create a junction table with a composite key (FeedID, CategoryID).

User_FeedItems (Optional): Create a junction table with a composite key (UserID, ItemID) to capture additional relationships if necessary.

Step 7: Finalize the List of Tables

Users
Attributes: (UserID, Name, Email, PasswordHash, CreatedAt, LastLogin)

User_Role
Attributes: (UserID, Role)

Feeds

Attributes: (FeedID, FeedURL, Title, Description, LastFetchedAt, [Optional: PublisherID])

FeedItems
Attributes: (ItemID, FeedID, Title, Content, PubDate, GUID, IsRead)

Authors
Attributes: (AuthorID, Name, Email, Bio, SocialLinks)

Publishers
Attributes: (PublisherID, Name, Website, LogoURL, RSSMetadata)

Categories
Attributes: (CategoryID, Name, Description, ParentCategoryID)

Subscriptions
Attributes: (SubscriptionID, UserID, FeedID, SubscriptionDate)

Interactions
Attributes: (InteractionID, UserID, ItemID, Type, Timestamp)

Notes
Attributes: (NoteID, UserID, ItemID, Content, CreatedAt)

Recommendations
Attributes: (RecommendationID, UserID, CategoryID, FrecencyScore, LastEngaged)

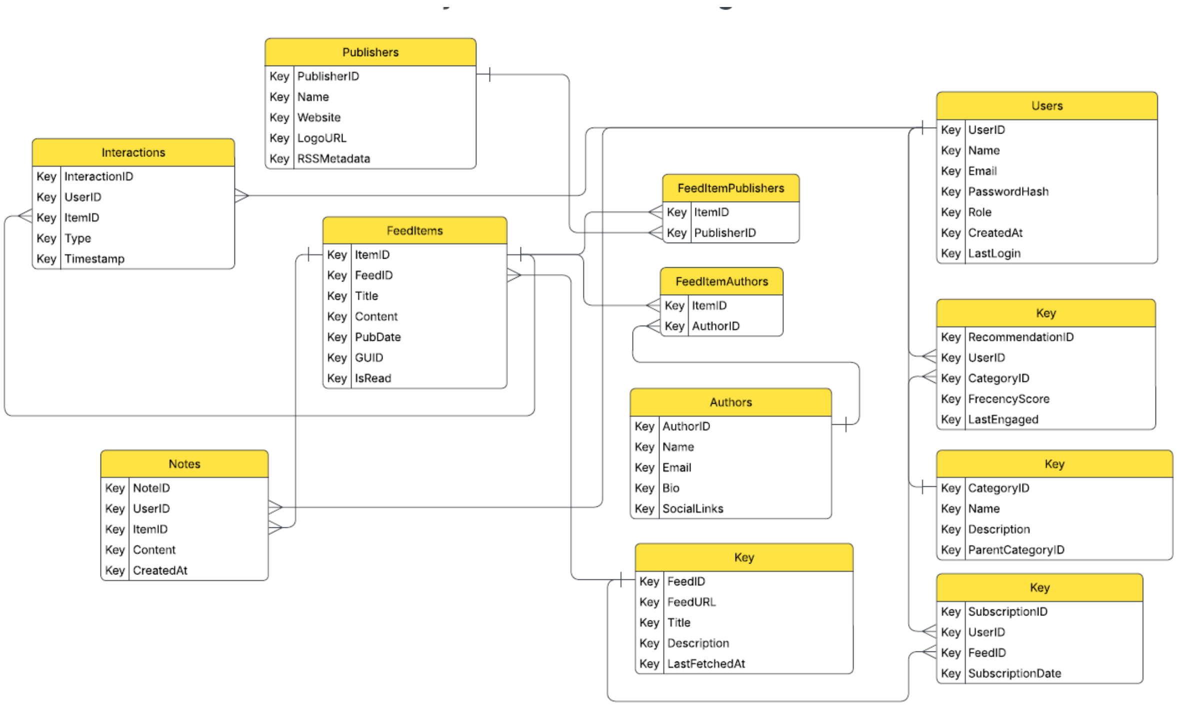
FeedItemAuthors
Composite Key: (ItemID, AuthorID)

FeedItemPublishers
Composite Key: (ItemID, PublisherID)

Feed_Categories
Composite Key: (FeedID, CategoryID)

User_FeedItems (Optional)

Composite Key: (UserID, ItemID)



4.3 Normalization

Define the Universal Relation

Start with a single comprehensive relation named FireSide_DB that includes all the attributes: UserID, UserName, Email, PasswordHash, Role, CreatedAt, LastLogin, FeedID, FeedURL, FeedTitle, FeedDescription, LastFetchedAt, ItemID, FeedID, ItemTitle, Content, PubDate, GUID, IsRead, AuthorID, AuthorName, AuthorEmail, Bio, SocialLinks, PublisherID, PublisherName, Website, LogoURL, RSSMetadata, CategoryID, CategoryName, CategoryDescription, ParentCategoryID, SubscriptionID, SubscriptionDate, InteractionID, InteractionType, InteractionTimestamp, NoteID, NoteContent, NoteCreatedAt, RecommendationID, FrecencyScore, LastEngaged.

Identify Functional Dependencies (FDs)

List the dependencies that describe how the non-key attributes relate to the keys:

FD1: UserID → UserName, Email, PasswordHash, Role, CreatedAt, LastLogin

FD2: FeedID → FeedURL, FeedTitle, FeedDescription, LastFetchedAt

FD3: ItemID → FeedID, ItemTitle, Content, PubDate, GUID, IsRead

FD4: AuthorID → AuthorName, AuthorEmail, Bio, SocialLinks

FD5: PublisherID → PublisherName, Website, LogoURL, RSSMetadata

FD6: CategoryID → CategoryName, CategoryDescription, ParentCategoryID

FD7: SubscriptionID → UserID, FeedID, SubscriptionDate

FD8: InteractionID → UserID, ItemID, InteractionType, InteractionTimestamp

FD9: NoteID → UserID, ItemID, NoteContent, NoteCreatedAt

FD10: RecommendationID → UserID, CategoryID, FrecencyScore, LastEngaged

Achieve First Normal Form (1NF)

Rules:

- Each attribute must contain only atomic (indivisible) values.
- There should be no repeating groups, and each row must be uniquely identifiable.

Violation & Resolution:
The Role attribute in the Users table is multi-valued if a user has multiple roles.

Decompose: Separate this into two tables:

- **Users:** (UserID, UserName, Email, PasswordHash, CreatedAt, LastLogin)
- **User_Role:** (UserID, Role)

This ensures that every attribute holds a single, atomic value.

Achieve Second Normal Form (2NF)

Requirement:

- The relation must first be in 1NF.
- Every non-key attribute must be fully functionally dependent on the entire primary key (especially critical for tables with composite keys).
- Identify Partial Dependencies:

- For example, partial dependencies may exist in tables where some attributes depend only on part of a composite key.

Decompose: Break the universal relation into smaller relations, ensuring that non-key attributes fully depend on their respective primary keys:

Users: UserID, UserName, Email, PasswordHash, CreatedAt, LastLogin

User_Role: UserID, Role

Feeds: FeedID, FeedURL, FeedTitle, FeedDescription, LastFetchedAt

FeedItems: ItemID, FeedID, ItemTitle, Content, PubDate, GUID, IsRead

Authors: AuthorID, AuthorName, AuthorEmail, Bio, SocialLinks

Publishers: PublisherID, PublisherName, Website, LogoURL, RSSMetadata

Categories: CategoryID, CategoryName, CategoryDescription, ParentCategoryID

Subscriptions: SubscriptionID, UserID, FeedID, SubscriptionDate

Interactions: InteractionID, UserID, ItemID, InteractionType, InteractionTimestamp

Notes: NoteID, UserID, ItemID, NoteContent, NoteCreatedAt

Recommendations: RecommendationID, UserID, CategoryID, FrecencyScore, LastEngaged

Additionally, create junction tables for many-to-many relationships:

FeedItemAuthors: ItemID, AuthorID

FeedItemPublishers: ItemID, PublisherID

Feed_Categories: FeedID, CategoryID

User_FeedItems: UserID, ItemID

This decomposition removes any partial dependency by ensuring each table's attributes relate fully to its primary key.

Achieve Third Normal Form (3NF)

Requirement:

- The relation must be in 2NF.
- There must be no transitive dependencies—non-key attributes should not depend on other non-key attributes.

Verification:

- For instance, in the Feeds table, FeedID directly determines FeedURL, FeedTitle, FeedDescription, and LastFetchedAt without any intermediate dependency.
- Similar checks are performed for FeedItems, Authors, Publishers, etc.
- Confirm that non-key attributes are directly and solely dependent on the primary key.
- With this, all transitive dependencies are eliminated.

Achieve Boyce-Codd Normal Form (BCNF)

Requirement:

For every functional dependency, X→Y, the determinant X must be a superkey.

Verification:

In the Users table: UserID (a superkey) determines all other attributes.

In the Feeds table: FeedID is the superkey for FeedURL, FeedTitle, FeedDescription, and LastFetchedAt.

The same applies to FeedItems (ItemID as superkey), Authors (AuthorID as superkey), Publishers (PublisherID as superkey), and Categories (CategoryID as superkey).
Each table is verified to ensure all functional dependencies meet the BCNF condition.

Final Normalized Relations (After BCNF)

The complete set of relations—now free from redundancy, partial, and transitive dependencies—are as follows:

Users: UserID, UserName, Email, PasswordHash, CreatedAt, LastLogin

User_Role: UserID, Role

Feeds: FeedID, FeedURL, FeedTitle, FeedDescription, LastFetchedAt

FeedItems: ItemID, FeedID, ItemTitle, Content, PubDate, GUID, IsRead

Authors: AuthorID, AuthorName, AuthorEmail, Bio, SocialLinks

Publishers: PublisherID, PublisherName, Website, LogoURL, RSSMetadata

Categories: CategoryID, CategoryName, CategoryDescription, ParentCategoryID

Subscriptions: SubscriptionID, UserID, FeedID, SubscriptionDate

Interactions: InteractionID, UserID, ItemID, InteractionType, InteractionTimestamp

Notes: NoteID, UserID, ItemID, NoteContent, NoteCreatedAt

Recommendations: RecommendationID, UserID, CategoryID, FrecencyScore, LastEngaged

FeedItemAuthors: ItemID, AuthorID

FeedItemPublishers: ItemID, PublisherID

Feed_Categories: FeedID, CategoryID

User_FeedItems: UserID, ItemID

5. Methodology

Project Fireside is a modern, personalized RSS feed aggregator and reader built with Next.js and MariaDB. This document provides a comprehensive overview of the system architecture and implementation details.

5.1 Architecture Overview

Project Fireside follows a layered architecture pattern with clear separation of concerns:

1. Presentation Layer: Next.js-based frontend with React components
2. Application Layer: API routes and middlewares
3. Business Logic Layer: Service modules for authentication, feed processing
4. Data Access Layer: Database interaction modules
5. Database Layer: MariaDB relational database

5.2 System Components

5.2.1 Frontend Components

The frontend is built with Next.js and organized into reusable React components that provide a responsive user interface for:

- Authentication (login/signup forms)
- Feed discovery and management
- Article reading and interaction
- User dashboard with reading statistics

5.2.2 Backend Services

The backend consists of several core services:

- Authentication Service: User registration, login, and session management
- Feed Processing Service: Fetching, parsing, and processing RSS feeds
- Recommendation System: Personalized content recommendations based on user behavior
- Data Management: Database operations and state management

5.2.3 Database Schema

The database uses a relational schema with tables for:

- Users and authentication
- Feeds and categories
- Feed items (articles)
- User interactions and preferences
- Content publishers and authors

5.3 Authentication Flow

1. Users register/login via the login form component
2. Credentials are validated against the database
3. Session tokens are issued and stored in cookies
4. Protected routes check for valid sessions via middleware

5.4 Feed Processing Flow

1. Users add RSS feed URLs via the add-feed form
2. The system fetches and parses the feed using the RSS parser
3. Feed metadata and items are stored in the database
4. New content is associated with publishers, authors, and categories
5. Items are presented to users based on subscriptions and preferences

5.5 Content Recommendation System

1. User interactions (reads, likes, saves) are tracked
2. Engagement patterns are analyzed by category and publisher
3. A recommendation algorithm calculates relevance scores
4. Personalized content is presented in the user's feed

5.6 Database Operations

1. Connection pooling for efficient database access
2. Transactions for data integrity
3. Normalized schema design to minimize redundancy
4. Indexes for optimized query performance

5.6.1 Frontend

- Next.js 14+ (React framework)
- React components with TypeScript
- Modern UI components with responsive design

5.6.2 Backend

- Node.js runtime
- Next.js API routes
- TypeScript for type safety

5.6.3 Database

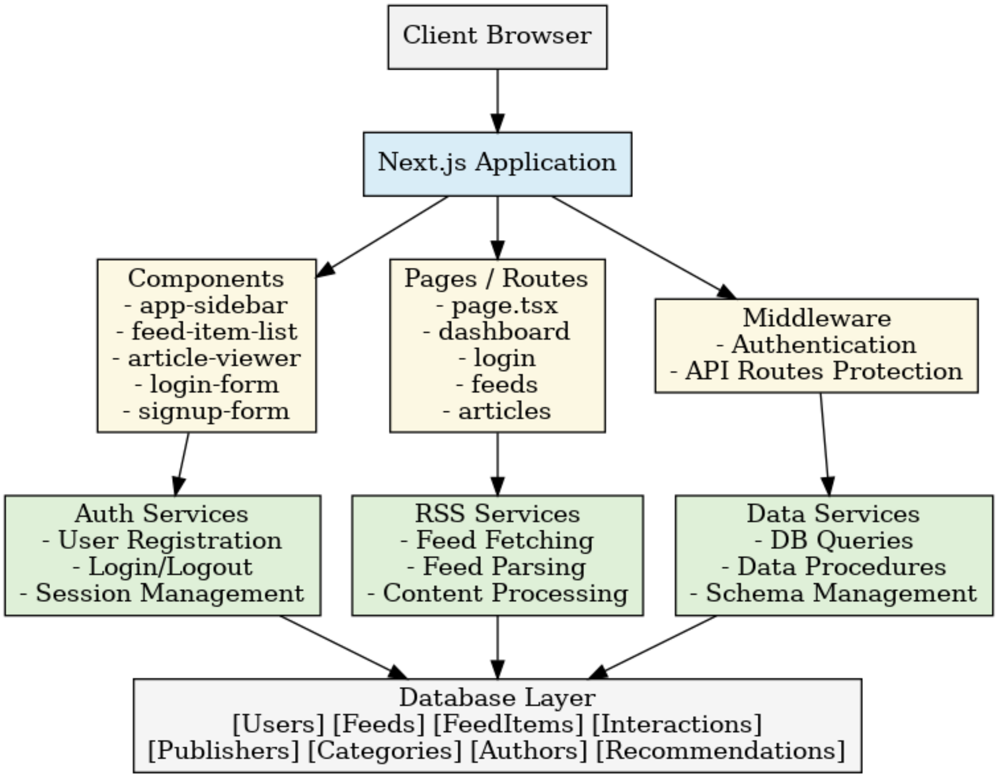
- MariaDB relational database
- Structured schema with referential integrity

5.6.4 DevOps

- Environment configuration via .env files
- Database initialization and seeding scripts
- Cron jobs for scheduled tasks like feed updates

5.6.5 Security Considerations

1. Password hashing with salt for secure storage
2. Session-based authentication with secure cookies
3. Input validation and sanitization
4. SQL injection protection via parameterized queries
5. Role-based access control for protected operations


Client Browser
Next.js Application
Components - app-sidebar - feed-item-list - article-viewer - login-form - signup-form
Pages / Routes - page.tsx - dashboard - login - feeds - articles
Middleware - Authentication - API Routes Protection
Auth Services - User Registration - Login/Logout - Session Management
RSS Services - Feed Fetching - Feed Parsing - Content Processing
Data Services - DB Queries - Data Procedures - Schema Management
Database Layer [Users] [Feeds] [FeedItems] [Interactions] [Publishers] [Categories] [Authors] [Recommendations]

6. Results

The database design and implementation have been tested against the functional and nonfunctional requirements outlined in the SRS document. Key outcomes include:

1. **Performance:** All API endpoints (user authentication, feed addition, subscription retrieval) respond within the specified limits .

2. **Scalability:** The normalized schema supports efficient querying even as the number of feeds and user interactions increases.

3. **Data Integrity:** The reduction and normalization processes have resulted in a stable database design where every non-key attribute is fully functionally dependent on its key, eliminating redundancy and potential anomalies.

4. **Security:** The implementation addresses essential security requirements, including proper user authentication, secure session management, and input sanitization to avoid common vulnerabilities such as SQL injection.

List of Tables

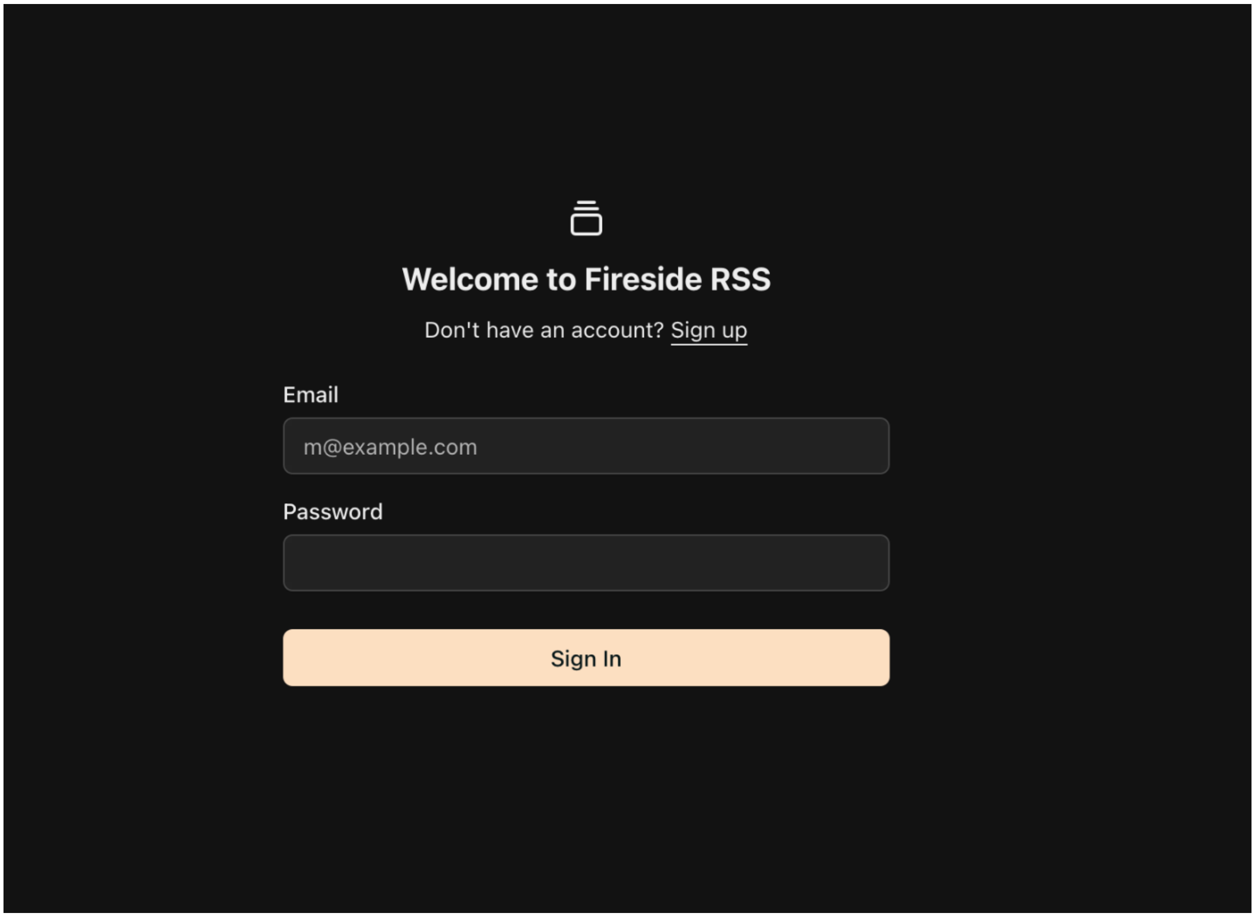
After applying normalization (from 1NF to BCNF), the final set of relations for Project Fireside are:

Table Name	Attributes
Users	UserID, UserName, Email, PasswordHash, CreatedAt, LastLogin
User_Role	UserID, Role
Feeds	FeedID, FeedURL, FeedTitle, FeedDescription, LastFetchedAt
FeedItems	ItemID, FeedID, ItemTitle, Content, PubDate, GUID, IsRead
Authors	AuthorID, AuthorName, AuthorEmail, Bio, SocialLinks
Publishers	PublisherID, PublisherName, Website, LogoURL, RSSMetadata
Categories	CategoryID, CategoryName, CategoryDescription, ParentCategoryID
Subscriptions	SubscriptionID, UserID, FeedID, SubscriptionDate
Interactions	InteractionID, UserID, ItemID, InteractionType, InteractionTimestamp
Notes	NoteID, UserID, ItemID, NoteContent, NoteCreatedAt

Recommendations	RecommendationID, UserID, CategoryID, FrecencyScore, LastEngaged
FeedItemAuthors	ItemID, AuthorID
FeedItemPublishers	ItemID, PublisherID
Feed_Categories	FeedID, CategoryID
User_FeedItems	UserID, ItemID

List of Figures

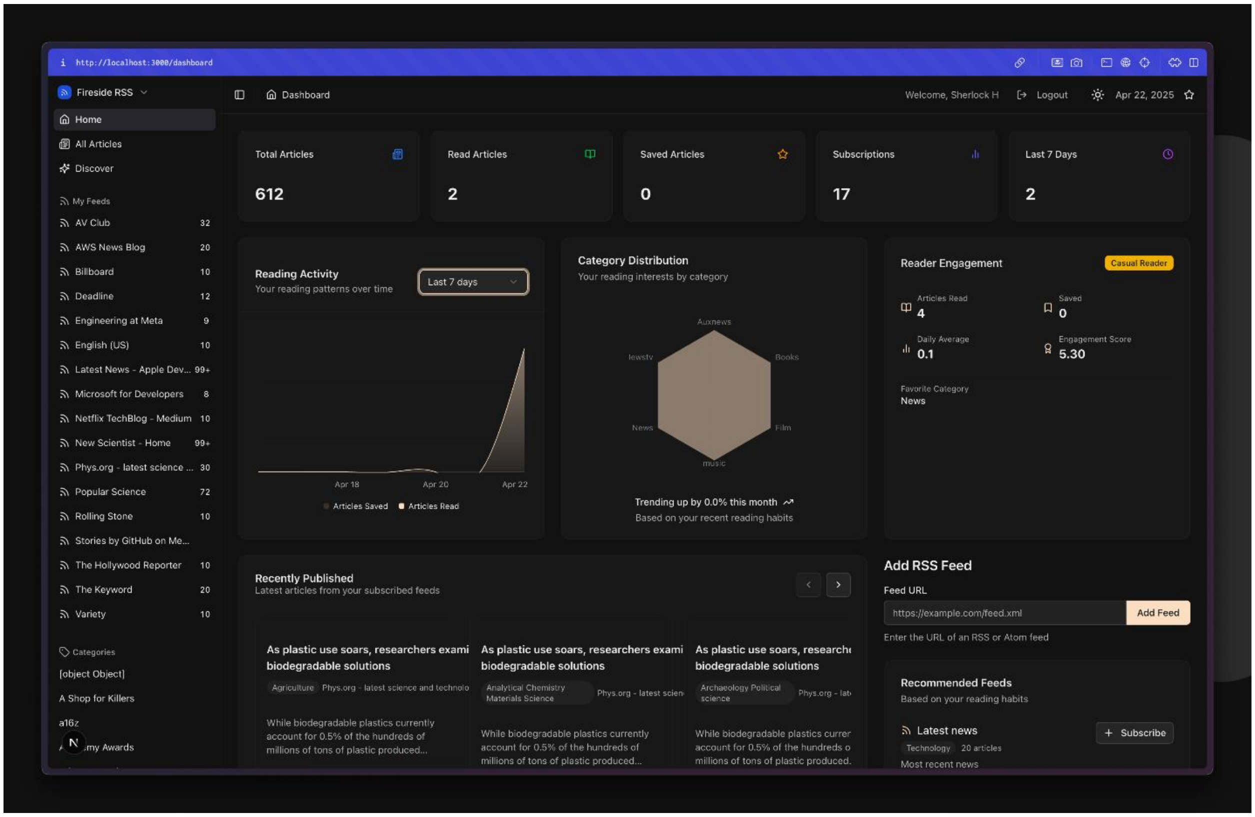
Login Page:



Welcome to Fireside RSS

Don't have an account? Sign up

User Dashboard:



Total Articles 612

Read Articles 2

Saved Articles 0

Subscriptions 17

Last 7 Days 2

Reading Activity
Your reading patterns over time

Category Distribution
Your reading interests by category

Trending up by 0.0% this month
Based on your recent reading habits

Reader Engagement

Articles Read 4

Saved 0

Daily Average 0.1

Engagement Score 5.30

Favorite Category
News

Recently Published
Latest articles from your subscribed feeds

As plastic use soars, researchers exami biodegradable solutions
Agriculture Phys.org - latest science and technolo
While biodegradable plastics currently account for 0.5% of the hundreds of millions of tons of plastic produced...

As plastic use soars, researchers exami biodegradable solutions
Analytical Chemistry Materials Science Phys.org - latest scien
While biodegradable plastics currently account for 0.5% of the hundreds of millions of tons of plastic produced...

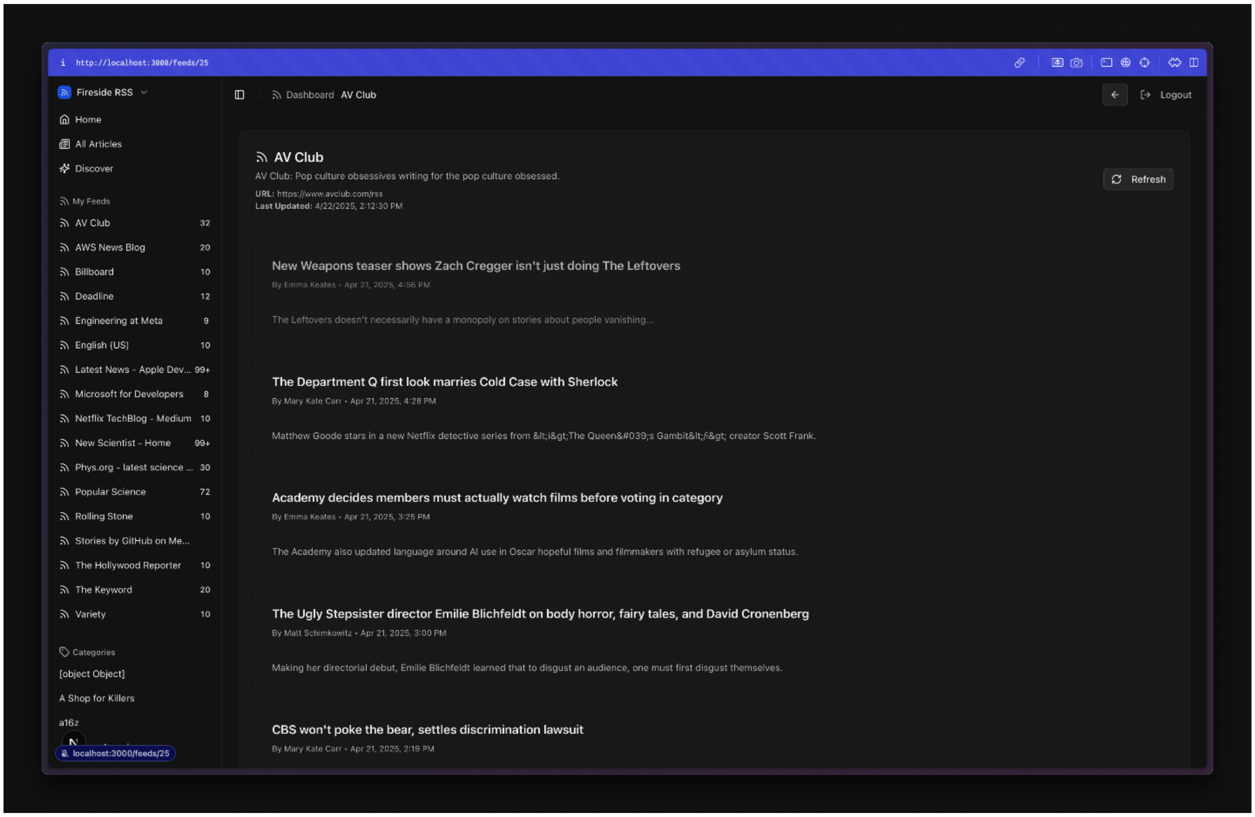
As plastic use soars, researche biodegradable solutions
Archaeology Political science Phys.org - lat
While biodegradable plastics currer account for 0.5% of the hundreds o millions of tons of plastic produced.

Add RSS Feed

Feed URL
https://example.com/feed.xml
Enter the URL of an RSS or Atom feed

Recommended Feeds
Based on your reading habits

Latest news
Technology 20 articles
Most recent news



AV Club
AV Club: Pop culture obsessives writing for the pop culture obsessed.
URL: https://www.avclub.com/rss
Last Updated: 4/22/2025, 2:12:30 PM

New Weapons teaser shows Zach Cregger isn't just doing The Leftovers
By Emma Keates • Apr 21, 2025, 4:56 PM
The Leftovers doesn't necessarily have a monopoly on stories about people vanishing...

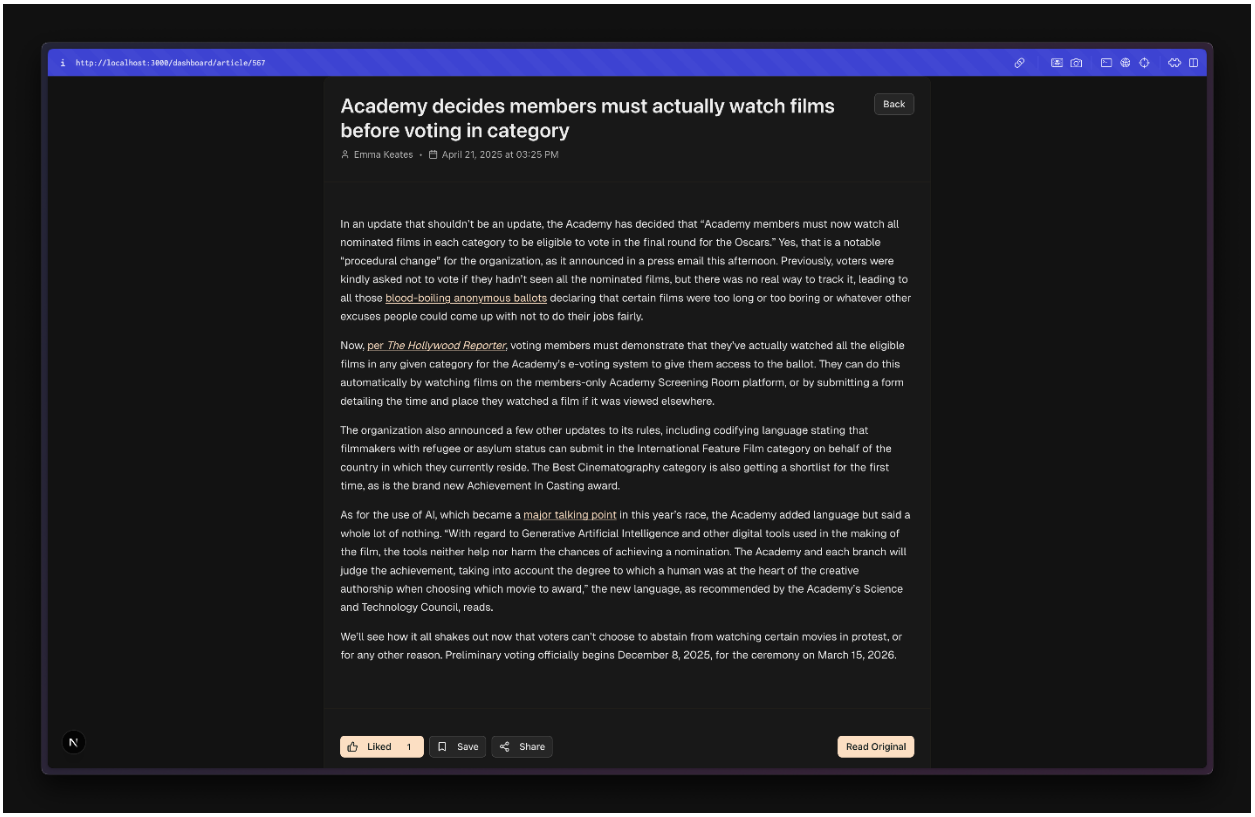
The Department Q first look marries Cold Case with Sherlock
By Mary Kate Carr • Apr 21, 2025, 4:28 PM
Matthew Goode stars in a new Netflix detective series from &lt;i&gt;The Queen&#039;s Gambit&lt;/i&gt; creator Scott Frank.

Academy decides members must actually watch films before voting in category
By Emma Keates • Apr 21, 2025, 3:25 PM
The Academy also updated language around AI use in Oscar hopeful films and filmmakers with refugee or asylum status.

The Ugly Stepsister director Emilie Blichfeldt on body horror, fairy tales, and David Cronenberg
By Matt Schimkowitz • Apr 21, 2025, 3:00 PM
Making her directorial debut, Emilie Blichfeldt learned that to disgust an audience, one must first disgust themselves.

CBS won't poke the bear, settles discrimination lawsuit
By Mary Kate Carr • Apr 21, 2025, 2:19 PM

Article Viewer:



Academy decides members must actually watch films before voting in category

Emma Keates • April 21, 2025 at 03:25 PM

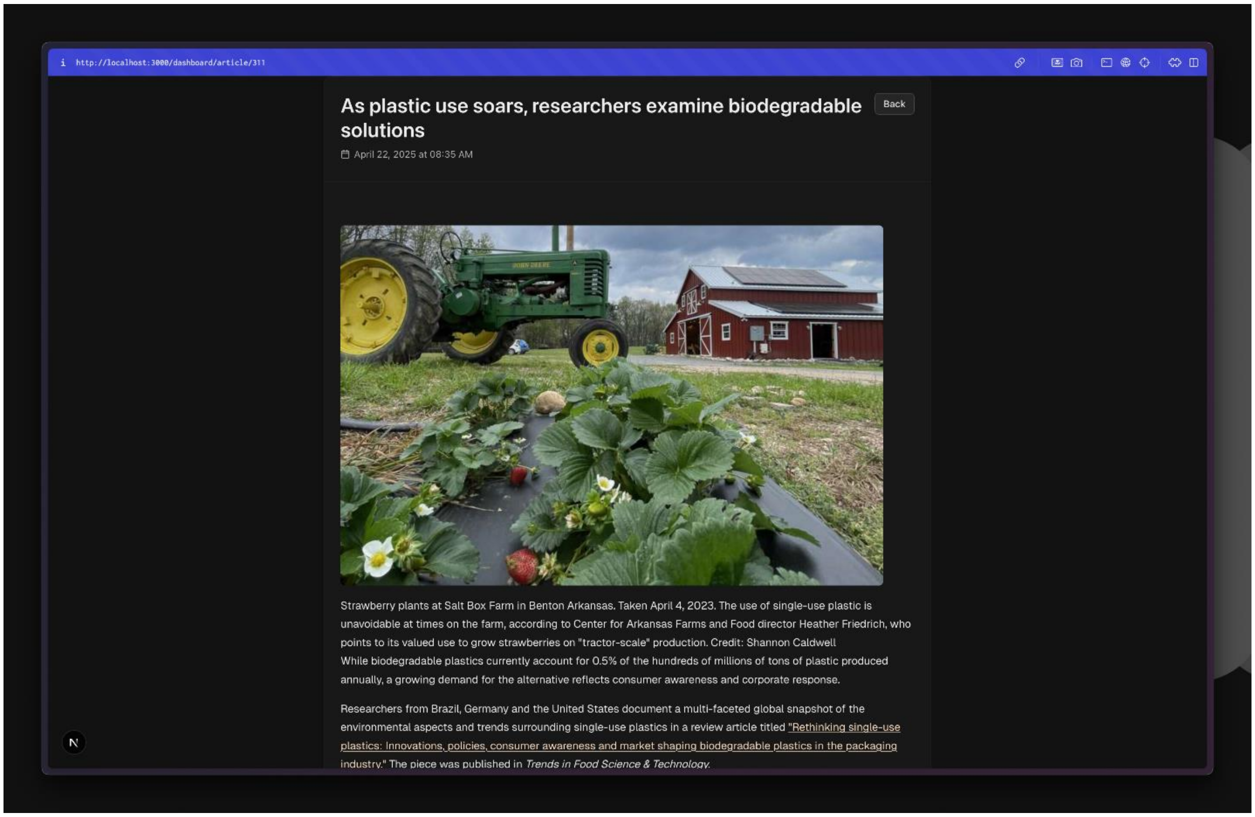
In an update that shouldn't be an update, the Academy has decided that "Academy members must now watch all nominated films in each category to be eligible to vote in the final round for the Oscars." Yes, that is a notable "procedural change" for the organization, as it announced in a press email this afternoon. Previously, voters were kindly asked not to vote if they hadn't seen all the nominated films, but there was no real way to track it, leading to all those blood-boiling anonymous ballots declaring that certain films were too long or too boring or whatever other excuses people could come up with not to do their jobs fairly.

Now, per *The Hollywood Reporter*, voting members must demonstrate that they've actually watched all the eligible films in any given category for the Academy's e-voting system to give them access to the ballot. They can do this automatically by watching films on the members-only Academy Screening Room platform, or by submitting a form detailing the time and place they watched a film if it was viewed elsewhere.

The organization also announced a few other updates to its rules, including codifying language stating that filmmakers with refugee or asylum status can submit in the International Feature Film category on behalf of the country in which they currently reside. The Best Cinematography category is also getting a shortlist for the first time, as is the brand new Achievement In Casting award.

As for the use of AI, which became a major talking point in this year's race, the Academy added language but said a whole lot of nothing. "With regard to Generative Artificial Intelligence and other digital tools used in the making of the film, the tools neither help nor harm the chances of achieving a nomination. The Academy and each branch will judge the achievement, taking into account the degree to which a human was at the heart of the creative authorship when choosing which movie to award," the new language, as recommended by the Academy's Science and Technology Council, reads.

We'll see how it all shakes out now that voters can't choose to abstain from watching certain movies in protest, or for any other reason. Preliminary voting officially begins December 8, 2025, for the ceremony on March 15, 2026.



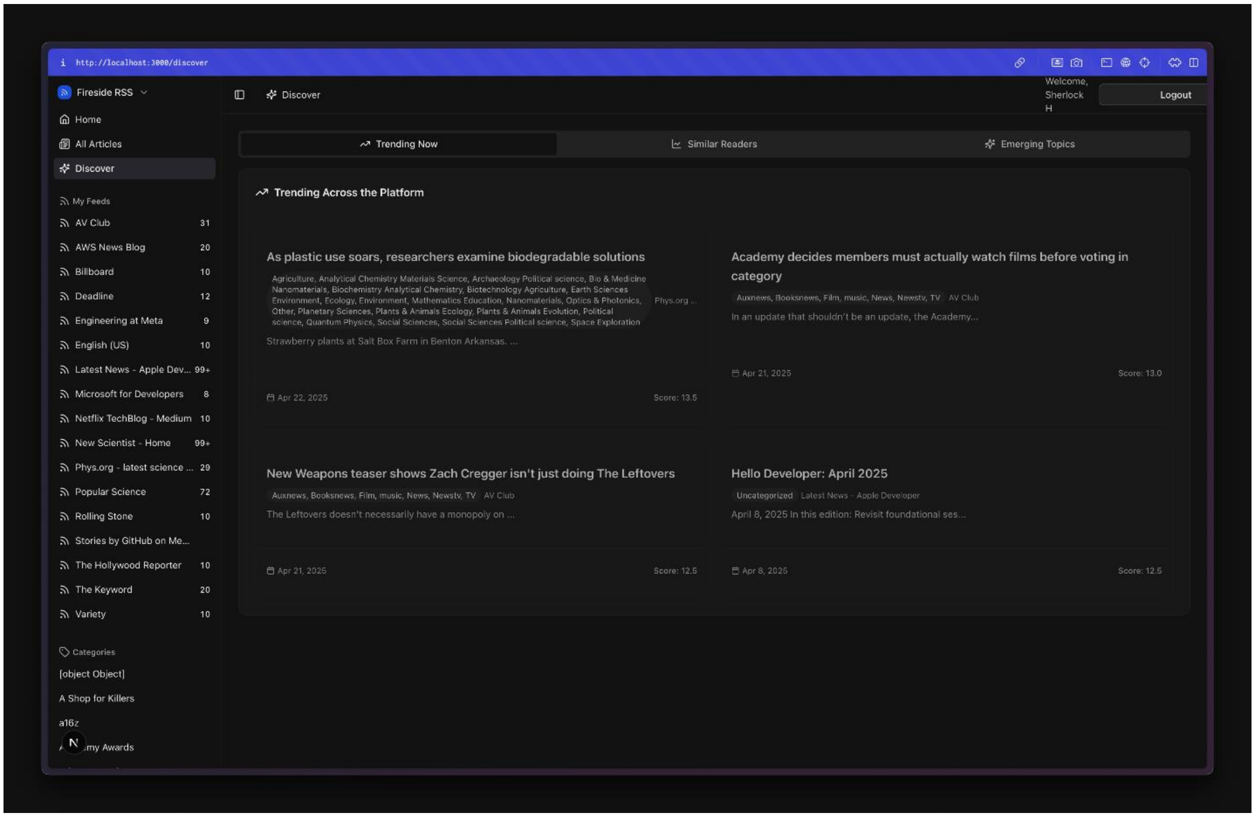
As plastic use soars, researchers examine biodegradable solutions

April 22, 2025 at 08:35 AM

Strawberry plants at Salt Box Farm in Benton Arkansas. Taken April 4, 2023. The use of single-use plastic is unavoidable at times on the farm, according to Center for Arkansas Farms and Food director Heather Friedrich, who points to its valued use to grow strawberries on "tractor-scale" production. Credit: Shannon Caldwell
While biodegradable plastics currently account for 0.5% of the hundreds of millions of tons of plastic produced annually, a growing demand for the alternative reflects consumer awareness and corporate response.

Researchers from Brazil, Germany and the United States document a multi-faceted global snapshot of the environmental aspects and trends surrounding single-use plastics in a review article titled "Rethinking single-use plastics: Innovations, policies, consumer awareness and market shaping biodegradable plastics in the packaging industry." The piece was published in *Trends in Food Science & Technology*.

Discover Page:



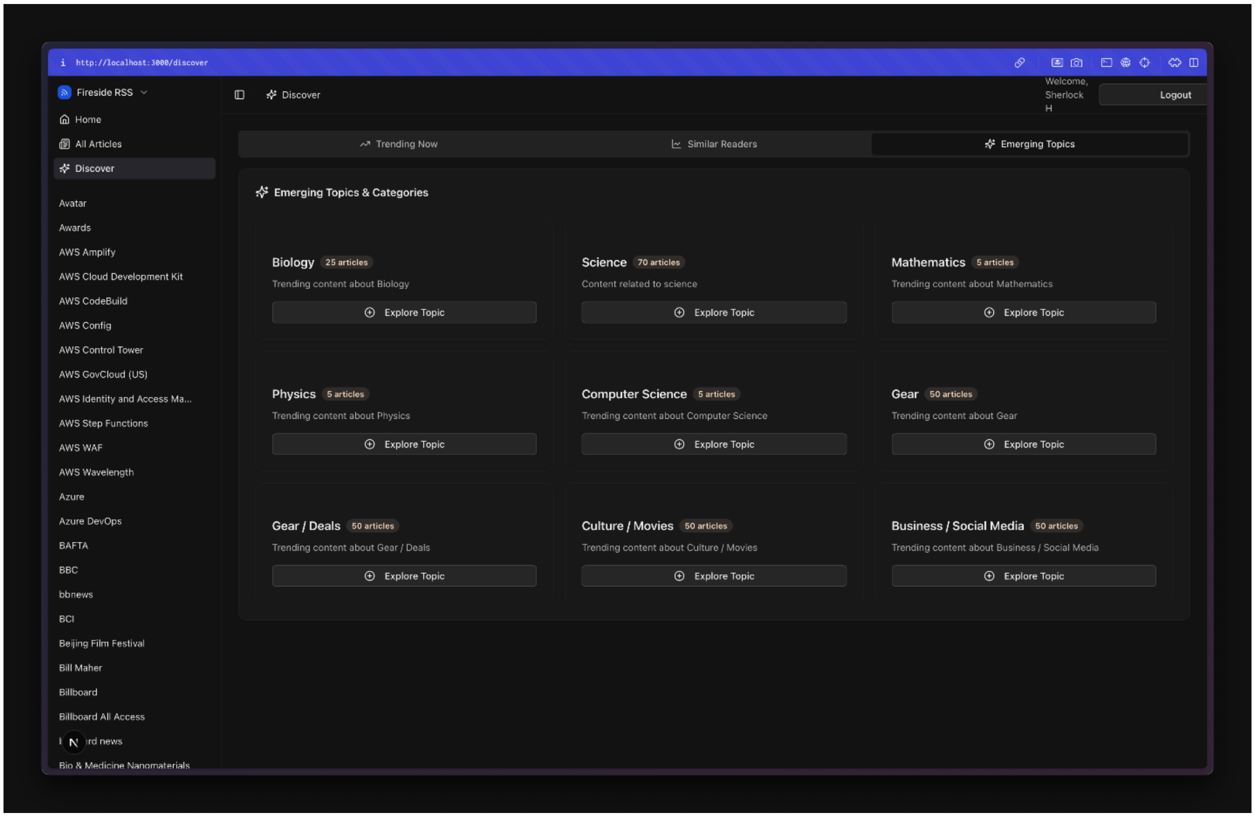
Trending Across the Platform

As plastic use soars, researchers examine biodegradable solutions
Agriculture, Analytical Chemistry Materials Science, Archaeology Political science, Bio & Medicine Nanomaterials, Biochemistry Analytical Chemistry, Biotechnology Agriculture, Earth Sciences Environment, Ecology, Environment, Mathematics Education, Nanomaterials, Optics & Photonics, Other, Planetary Sciences, Plants & Animals Ecology, Plants & Animals Evolution, Political science, Quantum Physics, Social Sciences, Social Sciences Political science, Space Exploration Phys.org ...
Strawberry plants at Salt Box Farm in Benton Arkansas. ...
Apr 22, 2025 Score: 13.5

Academy decides members must actually watch films before voting in category
Auxnews, Booksnews, Film, music, News, Newstv, TV AV Club
In an update that shouldn't be an update, the Academy...
Apr 21, 2025 Score: 13.0

New Weapons teaser shows Zach Cregger isn't just doing The Leftovers
Auxnews, Booksnews, Film, music, News, Newstv, TV AV Club
The Leftovers doesn't necessarily have a monopoly on ...
Apr 21, 2025 Score: 12.5

Hello Developer: April 2025
Uncategorized Latest News - Apple Developer
April 8, 2025 In this edition: Revisit foundational ses...
Apr 8, 2025 Score: 12.5



Emerging Topics & Categories

Biology 25 articles
Trending content about Biology

Science 70 articles
Content related to science

Mathematics 5 articles
Trending content about Mathematics

Physics 5 articles
Trending content about Physics

Computer Science 5 articles
Trending content about Computer Science

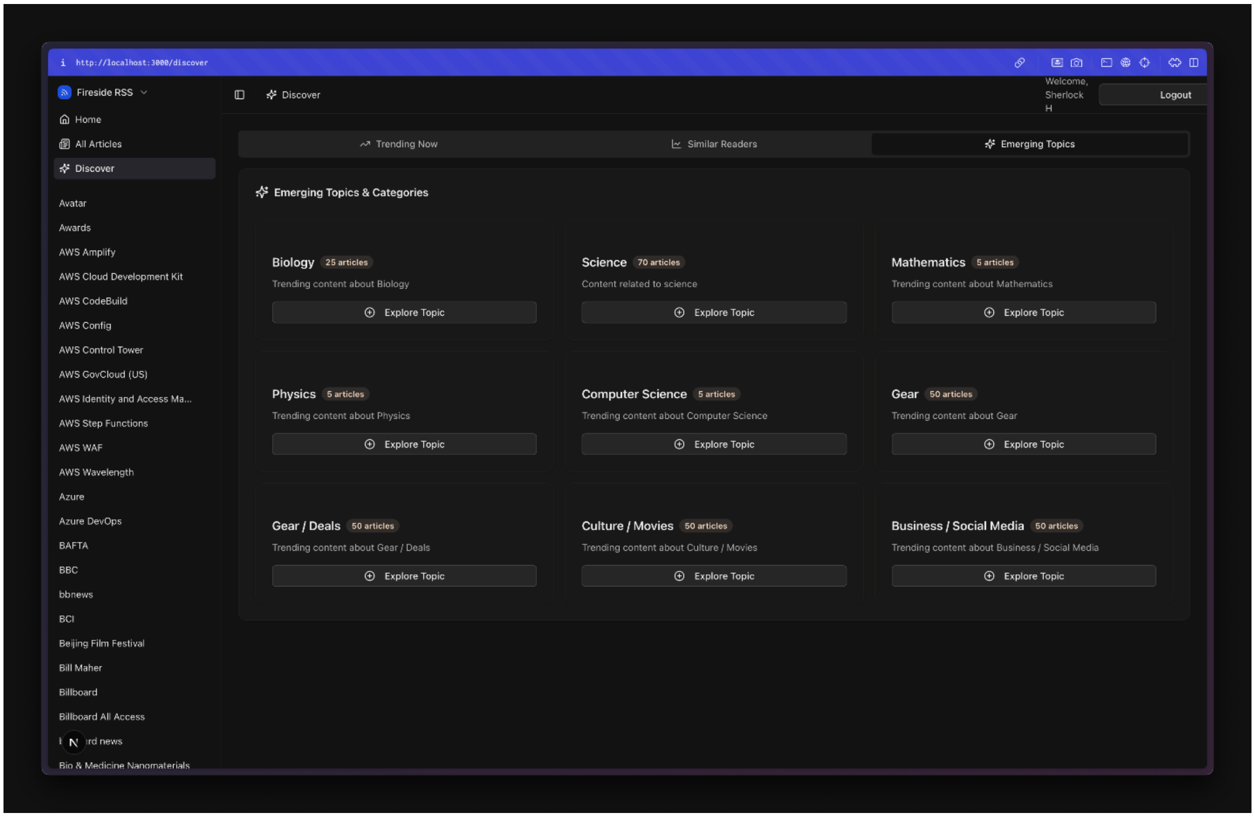
Gear 50 articles
Trending content about Gear

Gear / Deals 50 articles
Trending content about Gear / Deals

Culture / Movies 50 articles
Trending content about Culture / Movies

Business / Social Media 50 articles
Trending content about Business / Social Media

All Articles:



7. Conclusion

The Fireside RSS Reader Database Project successfully meets its design objectives by delivering a robust data model that supports comprehensive feed management, secure user operations, and efficient performance. The systematic reduction and normalization of the schema have ensured high data integrity and maintainability.

Future Work:

Implementing an enhanced user-specific read status mechanism (beyond the global IsRead flag) to support individual user interactions.

Exploring automated background feed refreshing to keep the aggregated content up-to-date.

Incorporating more extensive indexing and caching strategies to further reduce response times under heavy load.

Extending the system's functionality by integrating social features such as sharing or commenting on feed items.

8. References

1. D. Winer, “RSS – Really Simple Syndication: A historical perspective,” RSS Advisory Board, 2003. [Online]. Available: https://www.rssboard.org/rss-history RSS Advisory Board [Accessed: Apr. 22, 2025].

2. D. Winer, “RSS 2.0 Specification,” RSS Advisory Board, 2003. [Online]. Available: https://www.rssboard.org/rss-specification RSS Advisory Board [Accessed: Apr. 22, 2025].