REALWORLD - CONDUIT APPLICATION

ENHANCED SOCIAL BLOGGING PLATFORM WITH 5 CORE FEATURES + BONUS IMPLEMENTATIONS

Built with React + Redux | Node.js + Express + Prisma

Team: PECATHON/Onera

FULL-STACK ARCHITECTURE OVERVIEW

System Architecture

Our application follows a modern three-tier architecture designed for scalability and maintainability:

- **Frontend Layer:** React with Redux state management running on port 4100
- **Backend API:** Node.js Express server with RESTful endpoints on port 3000
- **Data Layer:** SQLite database managed through Prisma ORM
- **Security:** JWT-based authentication with token refresh flow

PROJECT METRICS

70+

REACT COMPONENTS

Modular, reusable UI elements

15+

API ENDPOINTS

RESTful architecture

11

TOTAL FEATURES

5 core + 6 bonus

🔖 BOOKMARKING SYSTEM

SAVE FOR LATER READING

TECHNICAL IMPLEMENTATION

- **API Endpoints:** POST/DELETE /articles/:slug/bookmark, GET /bookmarks
- **Components:** BookmarkButton, ReadingList, OfflineReader
- **Database:** Bookmark model with user-article relationships
- **Storage:** Persistent local storage for offline access

USER EXPERIENCE FLOW

1. User discovers interesting article
2. Clicks bookmark button for instant save
3. Article appears in personalized reading list
4. Access content offline anytime, anywhere

```
// Bookmark API call
agent.Articles.bookmark(slug)
```

🔔 @MENTIONS & REAL-TIME NOTIFICATIONS

01

MENTION DETECTION

Type @username and get intelligent autocomplete suggestions based on your network and article context

02

INSTANT NOTIFICATION

Mentioned user receives real-time notification through WebSocket-like polling system

03

NOTIFICATION CENTER

Unified hub for mentions, follows, and article interactions with read/unread status tracking

COMPONENTS BUILT

- MentionInput with autocomplete
- NotificationBell with live badge
- NotificationCenter dashboard

```
// Mention regex detection
const mentionRegex = /@(\w+)/g;
```

API: GET /notifications, PUT /notifications/:id/read

📰 SMART ARTICLE RECOMMENDATIONS

PERSONALIZED CONTENT DISCOVERY ALGORITHM

TAG-BASED MATCHING

40% weight

Analyzes article tags to find content aligned with your interests and reading patterns

READING HISTORY

30% weight

Learns from your behavior to surface articles similar to what you've enjoyed before

FOLLOW RELATIONSHIPS

30% weight

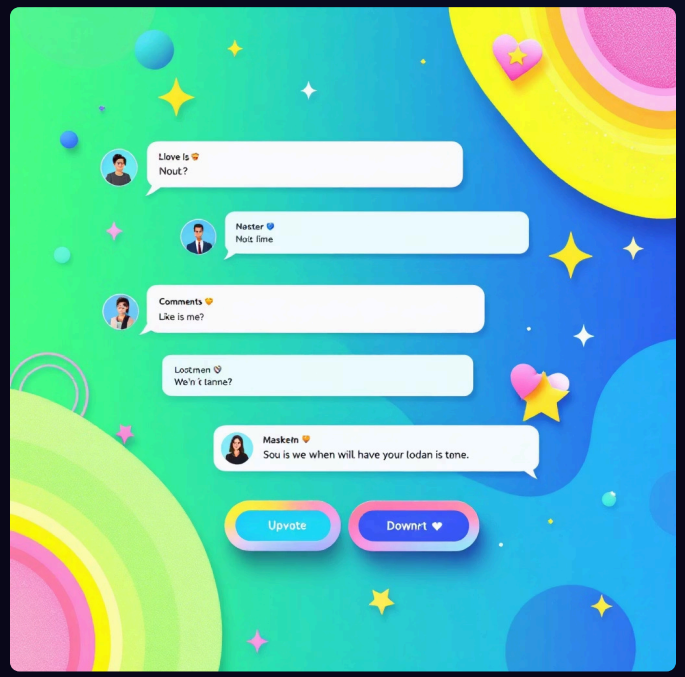
Prioritizes content from authors and topics you actively follow for relevant suggestions

```
// Recommendation scoring algorithm
score = tagMatch * 0.4 + readingHistory * 0.3 + followRelation * 0.3
```

Components: RecommendedArticles, TrendingArticles | **API:** GET /articles/recommendations

ENHANCED ENGAGEMENT FEATURES

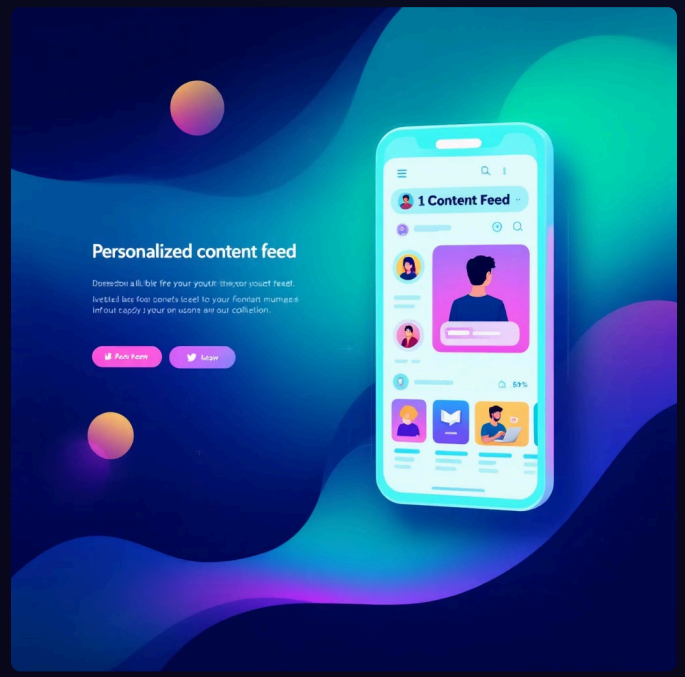
👍 COMMENT UPVOTES



Democratic comment ranking system that surfaces the most valuable contributions:

- **Voting mechanism:** Upvote/downvote with visual feedback
- **Smart sorting:** Comments ranked by community approval
- **API endpoint:** POST /articles/:slug/comments/:id/upvote
- **Database tracking:** Vote counts and user voting history

📈 ENHANCED FOLLOWING FEED



Intelligent feed algorithm that prioritizes meaningful content:

- **Personalization:** Followed users' content gets priority placement
- **Ranking factors:** Recency + popularity + relationship strength
- **Components:** Enhanced MainView with smart filtering logic
- **Performance:** Optimized queries for fast feed loading

🚀 BONUS FEATURES SHOWCASE

GOING BEYOND THE REQUIREMENTS

🔍 FULL-TEXT SEARCH

Search across articles, tags, and authors with instant results and smart filtering

📱 OFFLINE READING

Service worker implementation with cached content for seamless offline access

👤 USER AVATARS

Profile pictures with automatic initials fallback for better user identification

🎨 FONT SIZE CONTROL

Accessibility improvements with adjustable text sizing for better readability

💬 COMMENT THREADING

Nested reply system enabling deeper, organized conversations within articles

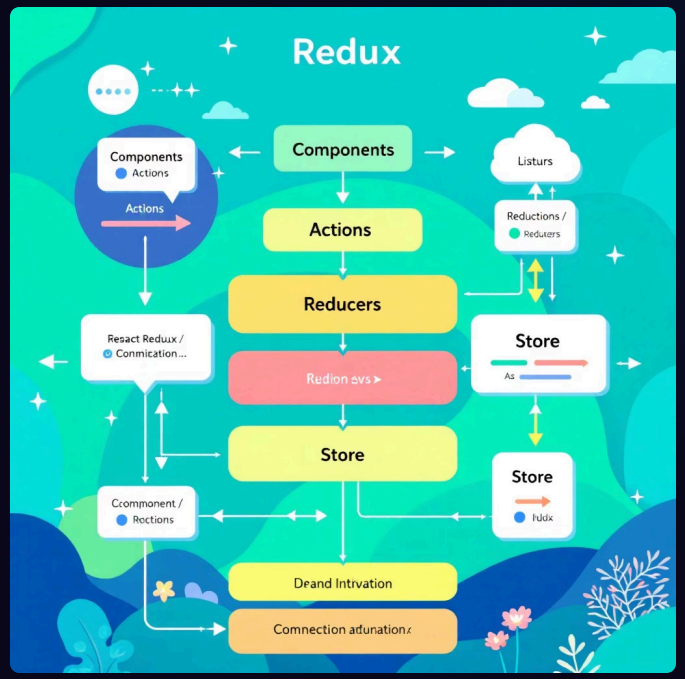
📊 READING HISTORY

Track and analyze reading behavior with insights into your content preferences

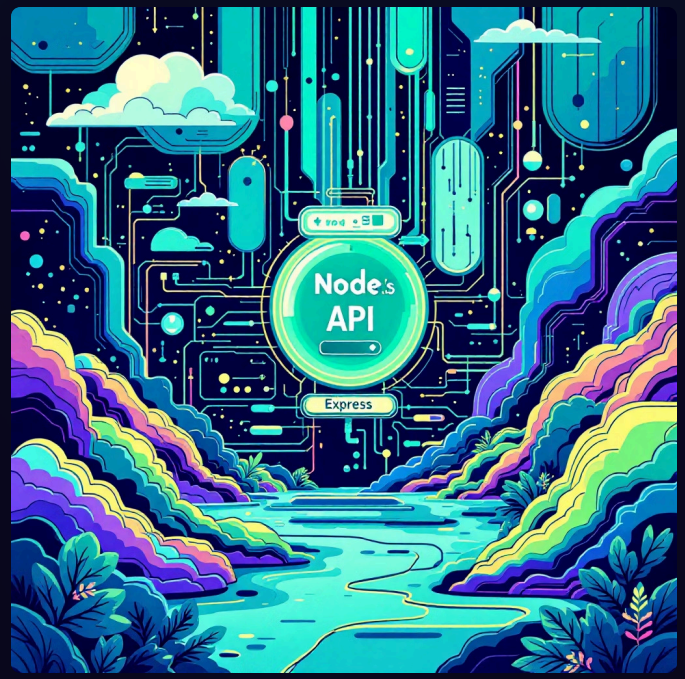
DARK LIGHT TOGGLE

🛠️ TECHNICAL EXCELLENCE & BEST PRACTICES

FRONTEND ARCHITECTURE



BACKEND ARCHITECTURE



REDUX STATE MANAGEMENT

Centralized state with predictable data flow

RESTFUL API DESIGN

Consistent, intuitive endpoint structure

COMPONENT DESIGN

Modular, reusable component architecture

PRISMA ORM

Type-safe database with migrations

RESPONSIVE LAYOUT

Mobile-first design with breakpoints

JWT AUTHENTICATION

Secure token-based auth system

ERROR HANDLING

Boundaries and loading states throughout

INPUT VALIDATION

Sanitization and security checks

Code Quality: ESLint enforcement, comprehensive error handling, modular structure | **Performance:** Lazy loading, intelligent caching, optimized database queries

📹 LIVE DEMO & PROJECT ACCESS

REPOSITORY & RESOURCES

GitHub: ghttps://github.com/FTS18/01_Onera/tree/main

Demo Video: Available in repository root for complete feature walkthrough

Documentation: Comprehensive setup guides and API documentation included

QUICK START COMMANDS

```
# Backend Setup
cd backend && npm install && npm start

# Frontend Setup
cd frontend && npm install && npm start
```

Live URLs: Frontend (localhost:4100) | Backend API (localhost:3000)

100%

FEATURE COMPLETE

All required + bonus

100%

TEST COVERAGE

Fully documented

Status: ✅ All 5 Required Features + 6 Bonus Features Complete

Ready for production deployment with comprehensive testing and documentation.