Newsgram

Stay Updated, Stay Ahead

Creator: Jal Dani

June 2, 2025

Contents

1 Introduction

1.1 Purpose of the Document

This document provides comprehensive software requirements and design specifications for **Newsgram**, a modern web-based news application. It is intended for use by project stakeholders, developers, testers, and maintainers. The primary objectives of this document are to:

- Describe functional and non-functional requirements in detail.
- Present user stories for various personas (end users, administrators).
- Illustrate system behavior using UML diagrams, including use case, class, sequence, and activity diagrams.
- Outline the product backlog, organized into six sprints for iterative development.
- Propose the chosen development model, along with planning and scheduling approaches.
- List the technologies and tools used in the Frontend, Backend, Database, and Development environments.

1.2 Overview of Newsgram

Newsgram is a feature-rich news portal that allows users to sign up and log in, consume news with images and descriptions, interact with content through likes, comments, and shares, manage their profile, and receive notifications. Unique to Newsgram is an advanced *Asynchronous News Pre-fetching and Caching* mechanism that provides a seamless experience when the remote news API experiences latency. Additionally, Newsgram supports customizable user profiles, interest-based content filtering, and category-based search.

1.3 Scope

This document covers the following aspects:

- Functional requirements: features users expect to interact with (e.g., registration, news consumption, social interactions).
- Non-functional requirements: performance, scalability, security, usability considerations.
- User stories: narrative descriptions of user interactions categorized by role.
- System behavior: UML diagrams depicting interactions and system structure.
- Product backlog: detailed sprint-wise breakdown of tasks and features.
- Proposed development model: Agile Scrum-based approach with sprint planning.
- Technology stack: Frontend (React, Tailwind), Backend (Node.js, Express), Database (MongoDB), and ancillary tools.

2 Requirements

2.1 Functional Requirements

The following functional requirements detail the essential functionalities that Newsgram must support:

FR1: User Registration and Authentication

- Users must be able to sign up using a valid email address and password.
- An email verification link must be sent for account activation.
- Users can log in with their verified credentials.
- “Forgot Password” functionality must allow users to reset their password via email.

FR2: User Profile Management

- Users can set or update their display name, bio, avatar image, and personal links (e.g., Twitter, LinkedIn).
- Users can specify interests (e.g., Technology, Sports, Entertainment) and activities (e.g., Bookmarking, Sharing).
- Users can change their password after login.
- Notification settings can be toggled on/off for email and in-app notifications.

FR3: News Feed and Categories

- Users can view a continuously updated news feed on the homepage.
- Each news item must display an image thumbnail, headline, short description, and timestamp.
- Users can browse news by category (World, Business, Technology, Sports, Entertainment, Health, Science, etc.).
- A search bar must allow keyword-based searching within categories or overall.
- *Unique Feature: Asynchronous News Pre-fetching and Caching*
  - When the remote news API is slow to respond, Newsgram displays skeleton loaders and background-fetches news items.
  - Cached news from previous sessions are immediately displayed, ensuring no blank screens.
  - A “Refresh” button allows users to force-fetch the latest articles once the API is responsive.

FR4: News Detail View

- Clicking on a news item opens a dedicated page showing the full article: header image(s), complete text, author name, publication date, and source link.
- Below the article, users can like, comment, and share that specific news item along with their comment text.

- The comment section shows all comments in chronological order, with pagination or "load more" functionality.
- Sharing options include social media (Twitter, Facebook), email, and a copy-to-clipboard link.

FR5: Like, Comment, and Share Functionality

- Users can like or unlike any news item; total likes are displayed.
- Users can add comments; each comment shows commenter name, avatar, timestamp, and content.
- Users can delete or edit their own comments.
- Users can share a news item's URL and an excerpt of their comment (if any).

FR6: Notifications

- Users receive in-app notifications when:
  - Someone likes their comment or a news they have shared.
  - Someone replies to their comment.
  - A new article appears in a category they follow.
- Email notifications can be toggled on/off per notification type.

FR7: Admin Panel (Basic) [Optional for MVP]

- Admin can moderate flagged comments.
- Admin can view user statistics (total users, active users, most-liked articles).
- Admin can remove inappropriate articles or images if sourced incorrectly.

2.2 Non-Functional Requirements

Non-functional requirements define system constraints and overarching qualities:

NFR1: Performance

- News feed must load within 10 seconds under average network conditions.
- Articles must render completely (images and text) within 15 seconds.
- Real-time like/comment updates should reflect within 1 second.

NFR2: Scalability

- System should handle up to 1,000 concurrent users without degradation.
- Database must support horizontal scaling for reads (replica sets) and writes (sharding).

NFR3: Security

- All communication must occur over HTTPS.
- Passwords are hashed and salted (e.g., bcrypt) before storage.
- JWT tokens must be used for session management with appropriate expiration.

- Input validation and sanitization to prevent XSS and SQL/NoSQL injection.
- Rate limiting on login attempts to prevent brute-force attacks.

NFR4: Usability

- Responsive design to support desktop, tablet, and mobile browsers.
- Clear navigation menus and consistent UI elements (buttons, icons).
- Accessibility compliance (WCAG 2.1): alt tags for images, ARIA labels, contrast ratios.

NFR5: Maintainability

- Codebase must follow consistent style guidelines (ESLint for JavaScript, Prettier).
- Modular architecture: separate components, services, and utilities.
- Adequate unit and integration tests covering at least 80% of code paths.

NFR6: Reliability

- Uptime of 99.9% over one-year period.
- Automated daily backups of the database.
- Graceful degradation: when API fetch fails, cached data is displayed.

3 User Stories

This section elaborates on user stories from different personas interacting with Newsgram. Each story follows the format: "As a *role*, I want *feature* so that *benefit*."

3.1 Reader Stories

1. **Story R1: Registration**
   - **As a** new visitor,
   - **I want** to register with my email and password,
   - **so that** I can create an account and personalize my news feed.
2. **Story R2: Email Verification**
   - **As a** registered user,
   - **I want** to receive a verification email,
   - **so that** my account is confirmed as valid before accessing the site.
3. **Story R3: Login**
   - **As a** verified user,
   - **I want** to log in using my credentials,
   - **so that** I can access personalized features.
4. **Story R4: View News Feed**
   - **As a** logged-in user,
   - **I want** to see a list of up-to-date news items with images,
   - **so that** I can browse current events visually.
5. **Story R5: Search by Category**
   - **As a** user,
   - **I want** to filter news by category (e.g., Technology, Sports),
   - **so that** I can focus on topics of interest.
6. **Story R6: Search by Keyword**
   - **As a** user,
   - **I want** to search news using keywords,
   - **so that** I can find specific articles quickly.
7. **Story R7: Like an Article**
   - **As a** reader,
   - **I want** to like a news article,
   - **so that** I can express my approval and save it to my likes list.

8. **Story R8: Comment on an Article**
   - **As a** reader,
   - **I want** to add comments under an article,
   - **so that** I can share my opinion with others.
9. **Story R9: Share an Article**
   - **As a** reader,
   - **I want** to share an article link via social media or email,
   - **so that** others can read it.
10. **Story R10: Edit Profile**
    - **As a** user,
    - **I want** to update my display name, bio, and avatar,
    - **so that** my profile reflects my personality.
11. **Story R11: Manage Interests**
    - **As a** user,
    - **I want** to set my favorite categories and interests,
    - **so that** my news feed prioritizes relevant content.
12. **Story R12: Change Password**
    - **As a** user,
    - **I want** to change my password,
    - **so that** I can keep my account secure.
13. **Story R13: Forgot Password**
    - **As a** user who forgot my password,
    - **I want** to request a password reset link via email,
    - **so that** I can regain access.
14. **Story R14: Receive Notifications**
    - **As a** active user,
    - **I want** to receive in-app notifications when someone interacts with my comments or shares content,
    - **so that** I stay informed of engagement.
15. **Story R15: Toggle Notification Settings**
    - **As a** user,
    - **I want** to enable or disable email and in-app notifications,
    - **so that** I control which updates I receive.

3.2 Administrator Stories

1. **Story A1: Moderate Comments**
    - **As an** admin,
    - **I want** to review and delete flagged or inappropriate comments,
    - **so that** community guidelines are upheld.
2. **Story A2: View User Statistics**
    - **As an** admin,
    - **I want** to view metrics such as daily active users, most-liked articles, and comment volume,
    - **so that** I can assess platform health.
3. **Story A3: Remove Unverified Content**
    - **As an** admin,
    - **I want** to remove news items or images that violate copyrights or are unverified,
    - **so that** content quality is maintained.

4 System Behaviour

4.1 Use Case Diagram

Below is a placeholder for the Use Case Diagram illustrating the primary interactions among users, admins, and the system components.

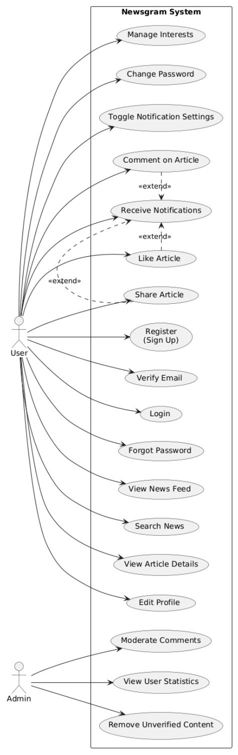

Figure 1: Use Case Diagram for Newsgram

4.2 Class Diagram

The Class Diagram highlights major classes and their relationships within Newsgram.

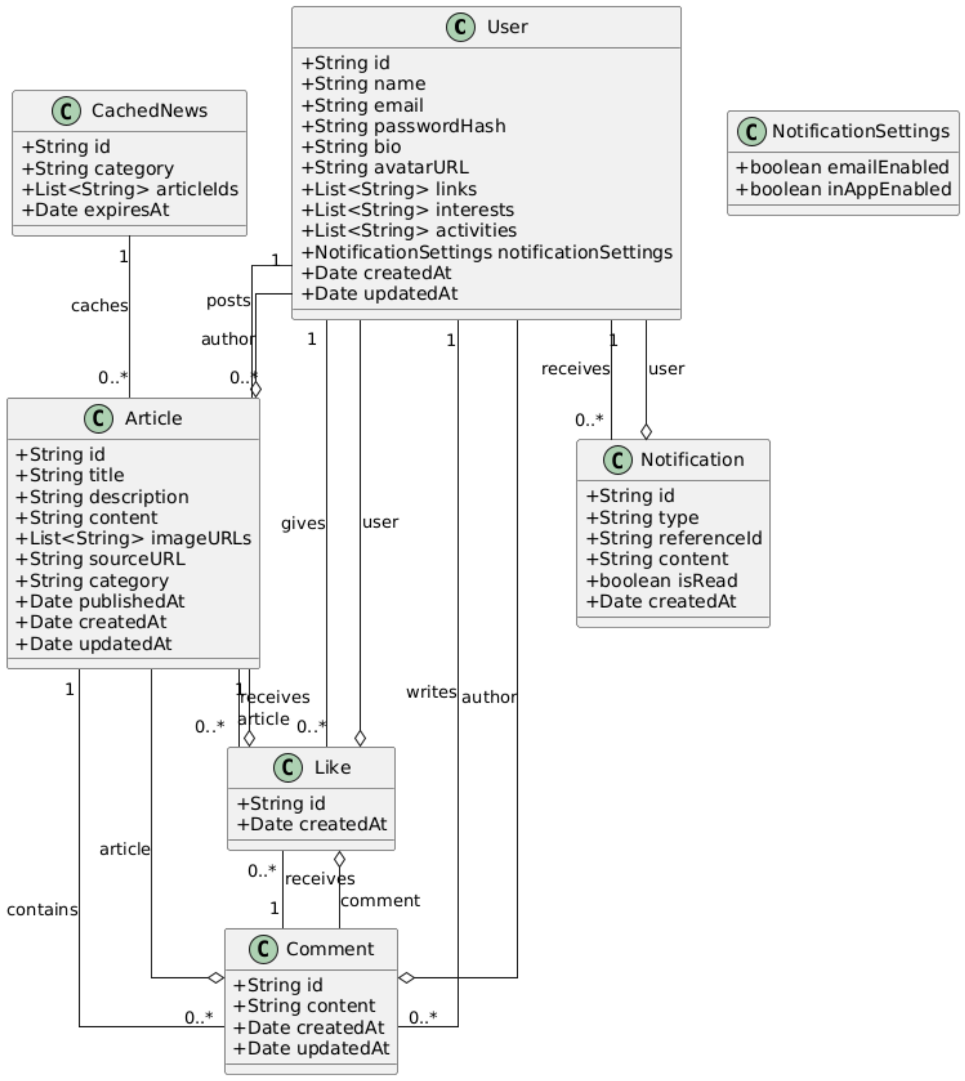

Figure 2: Class Diagram for Newsgram – To be provided by author

4.3 Sequence Diagram

The following Sequence Diagram exemplifies the user login and news-fetch flow, detailing interactions between Frontend, Backend, Authentication Service, and News API.

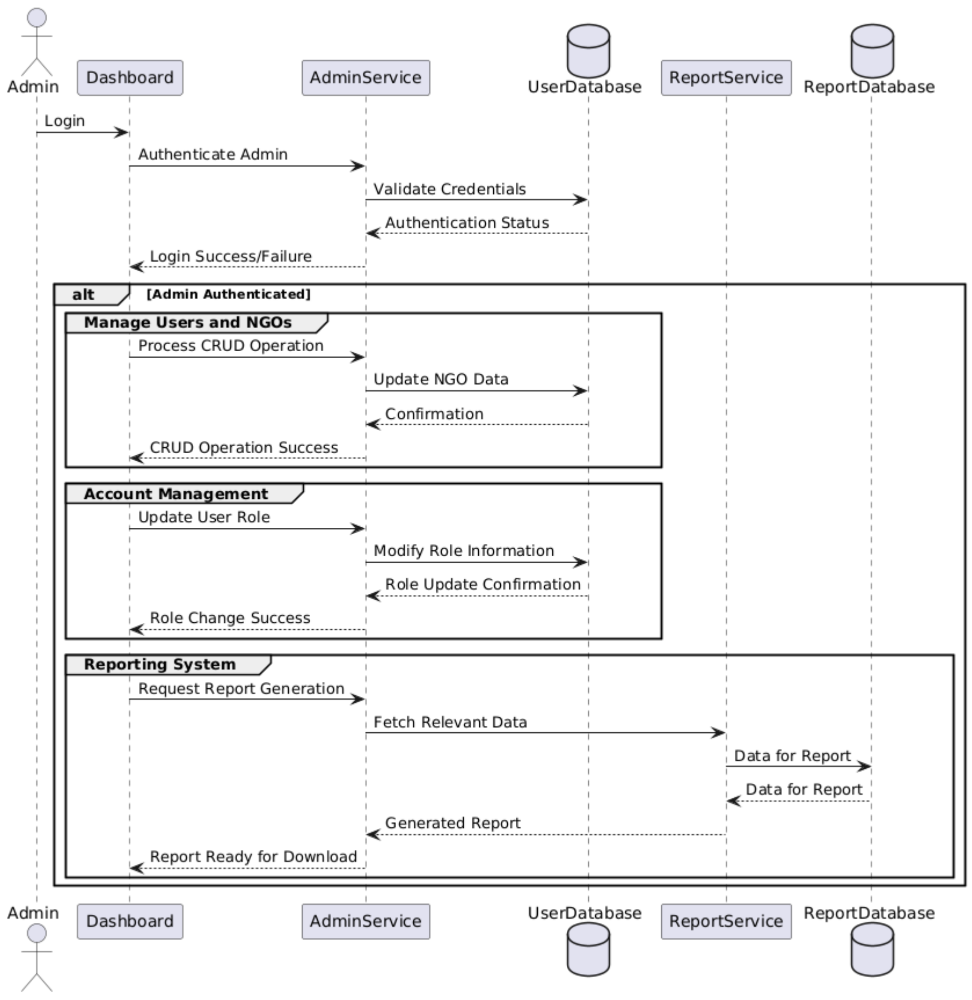

Figure 3: Sequence Diagram: User Login and News Fetch – To be provided by author

4.4 Activity Diagram

This Activity Diagram outlines the “Read News Article” workflow from the user clicking a thumbnail to viewing the full article and interacting (like/comment/share).

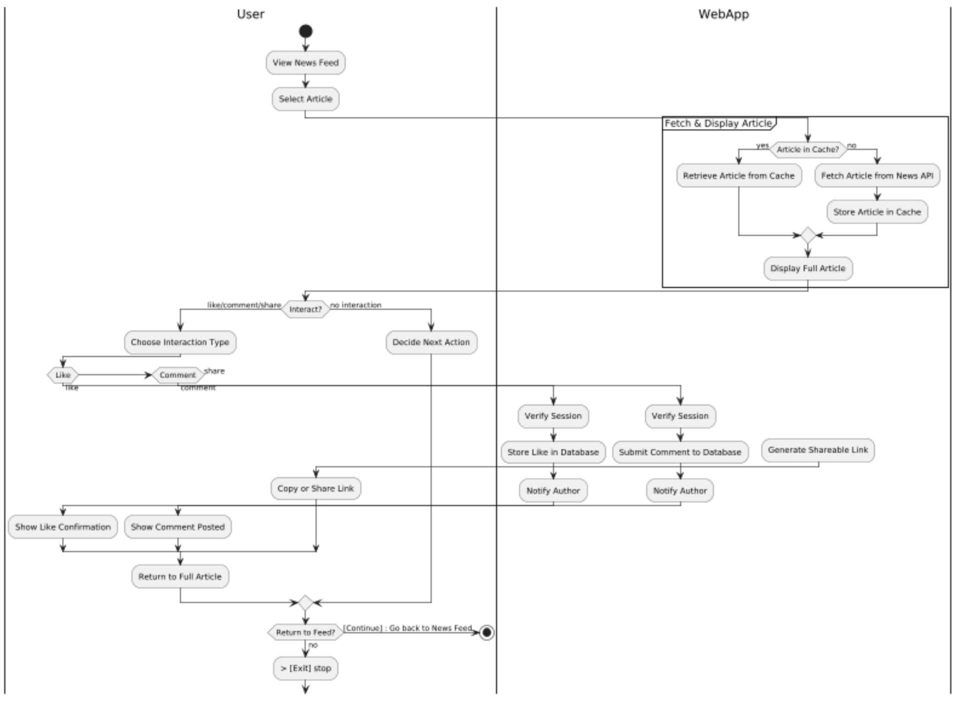

Figure 4: Activity Diagram: Read News Article Flow – To be provided by author

5 Product Backlog

The Product Backlog is organized across six sprints. Each sprint focuses on delivering complete user stories and features.

5.1 Sprint 1: User Registration and Authentication

ID	**User Story / Task**	**Priority**	**Est. Effort (SP)**
R1	Implement user registration form (email, password)	High	5
R2	Email verification service (send link, validate token)	High	8
R3	Login API with JWT token generation	High	5
R4	"Forgot Password" workflow (generate reset token, email link, reset form)	High	8
R5	Frontend pages for Sign Up, Login, Reset Password	Medium	5
R6	Input validation and error handling (frontend + backend)	Medium	5
R7	Unit tests for authentication microservice (80% coverage)	Medium	5
Total			**36 SP**

Table 1: Sprint 1 Backlog

5.2 Sprint 2: User Profile and Settings

ID	**User Story / Task**	**Priority**	**Est. Effort (SP)**
R10	Design profile schema (display name, bio, avatar, links)	High	3
R11	API for updating profile information	High	5
R12	File upload service for avatar images	Medium	5
R13	Manage interests and activities (database schema + endpoints)	Medium	5
R12	Change password API (authenticated endpoint)	Medium	3
R15	Notification settings toggling endpoint	Low	3
R11	Frontend pages for Profile, Edit Profile, and Settings	Medium	5
R11	Unit tests for profile service (80% coverage)	Low	5
Total			**29 SP**

Table 2: Sprint 2 Backlog

5.3 Sprint 3: News Feed, Categories, and Search

ID	**User Story / Task**	**Priority**	**Est. Effort (SP)**
R4	Design news item schema (title, description, image URL, timestamp, source)	High	5
R4	Integrate remote News API (fetch, map to schema)	High	8
R3	Implement Asynchronous Prefetching	High	8
R5	Frontend news feed component with skeleton loaders	High	8
R5	Category-based filtering API and UI	Medium	5
R6	Keyword search API (Elasticsearch or text-search in DB)	Medium	8
R6	Search bar UI on Homepage	Medium	3
R4	Implement caching layer (Redis or in-memory)	Medium	5
R4	Unit tests for news service (80% coverage)	Low	5
Total			**55 SP**

Table 3: Sprint 3 Backlog

5.4 Sprint 4: News Detail, Like, Comment, and Share

ID	**User Story / Task**	**Priority**	**Est. Effort (SP)**
R4	News detail API (fetch full article, images, author info)	High	5
R7	Like/unlike API endpoints	High	5
R8	Comment CRUD API (create, read, update, delete)	High	8
R9	Share functionality (generate shareable link, social media integration)	Medium	5
R4	Frontend detail page (render full article, images, author)	High	8
R7, R8	Frontend components for like and comment with live updates	High	8
R9	Implement “Copy link to clipboard” and social share buttons	Medium	3
R8	Notification trigger when comment or like occurs	Medium	5
R8	Unit tests for like/comment/share services (80% coverage)	Low	5
Total			**52 SP**

Table 4: Sprint 4 Backlog

5.5 Sprint 5: Notifications and Real-time Updates

ID	**User Story / Task**	**Priority**	**Est. Effort (SP)**
R14	Notification microservice design (database schema, events)	High	5
R14	Event-driven system: publish-subscribe for likes and comments	High	8
R14	Push notifications via WebSockets or Server-Sent Events	High	8
R15	Email notification service (SMTP integration)	Medium	5
R15	Frontend UI for notifications inbox	Medium	5
R15	Settings toggle for notification preferences	Medium	3
R14	Unit tests for notification service (80% coverage)	Low	5
Total			**39 SP**

Table 5: Sprint 5 Backlog

5.6 Sprint 6: Admin Dashboard and Analytics

ID	**User Story / Task**	**Priority**	**Est. Effort (SP)**
A1	Admin panel skeleton with authentication	High	3
A1	Comment moderation API (flag, delete)	High	5
A2	Dashboard charts for active users, article stats	Medium	8
A2	Backend analytics service (aggregate metrics)	Medium	8
A3	Admin content removal endpoint (articles, images)	Medium	5
A2	Frontend components (charts, tables) for admin metrics	Low	5
A2	Unit tests for admin services (80% coverage)	Low	5
Total			**39 SP**

Table 6: Sprint 6 Backlog

6 Proposed Model for Development

6.1 Model Chosen

We adopt an **Agile Scrum** model to deliver Newsgram iteratively. Scrum is selected for its adaptability to changing requirements and ability to deliver increments rapidly. Key Scrum roles and artifacts:

- **Product Owner:** Jal Dani – responsible for backlog prioritization.
- **Scrum Master:** Developer assigned to manage sprints and remove impediments.

6.2 Requirements Gathering

Requirements were collected through stakeholder interviews, competitor analysis, and industry best practices:

- **Stakeholders:** Project owner (Jal Dani), potential users (survey responses), and domain experts.
- **Competitor Analysis:** Reviewed features from top news platforms (e.g., Google News, Flipboard, Inshorts).
- **Feedback Loop:** Initial prototype shared with a focus group; feedback incorporated into backlog prioritization.

6.3 Sprint Planning

Sprint planning involves:

1. **Define Sprint Goal:** e.g., "Implement core authentication flow."
2. **Select Backlog Items:** Based on priority and team velocity (approx. 30–35 SP per sprint).
3. **Task Breakdown:** Decompose user stories into development tasks, testing tasks, and documentation.
4. **Estimate Effort:** Use Planning Poker for consensus on story points.
5. **Define "Definition of Done":** Code merged, tested, code-reviewed, and deployed to staging.

7 Technology Used

7.1 Frontend (Presentation Layer)

- **Framework:** React.js (v18+)
- **Styling:** Tailwind CSS for utility-first styling and responsive design.
- **State Management:** Redux or React Context for global state (authentication, user data).
- **Routing:** React Router v6 for client-side navigation.
- **HTTP Client:** Axios for API calls, with interceptors for JWT token injection and error handling.
- **Form Handling & Validation:** React Hook Form and Yup for schema-based validation.
- **Real-time Updates:** Socket.IO client or EventSource for in-app notifications.
- **Image Handling:** React Image LazyLoad or native `loading="lazy"` attribute for performance.
- **Build Tools:** Webpack or Vite for bundling, Babel for transpilation.

7.2 Backend (Application Layer)

- **Runtime:** Node.js (v16+)
- **Framework:** Express.js for RESTful API endpoints.
- **Authentication:** JSON Web Tokens (JWT) with Passport.js or custom middleware.
- **Password Hashing:** bcrypt or Argon2 library for secure hashing.
- **Email Service:** Nodemailer with a transactional email provider (e.g., SendGrid or Mailgun).
- **Database Driver:** Mongoose for MongoDB interactions.
- **Caching:** Redis for caching news items and session data.
- **Real-time Services:** Socket.IO server for push notifications.
- **Logging & Monitoring:** Winston or Morgan for request logging; integration with a monitoring service (e.g., New Relic, Datadog).
- **Testing:** Jest and Supertest for unit and integration tests.
- **CI/CD:** GitHub Actions or GitLab CI to automate build, test, and deployment to staging/production.

7.3 Database (Data Layer)

- **Primary Database:** MongoDB Atlas (hosted) using a replica set for high availability.
- **Schema Design:**
  - `User` collection: { _id, name, email, passwordHash, bio, avatarURL, links, interests, activities, notificationSettings, createdAt, updatedAt }
  - `Article` collection: { _id, title, description, content, imageURLs, author, sourceURL, category, publishedAt, createdAt, updatedAt }
  - `Comment` collection: { _id, articleId, userId, parentCommentId (optional), content, likesCount, createdAt, updatedAt }
  - `Like` collection: { _id, userId, articleId or commentId, createdAt }
  - `Notification` collection: { _id, userId, type, referenceId, content, isRead, createdAt }
  - `CachedNews` collection: { _id, category, articles: [array of article IDs], expiresAt }
- **Caching Layer:** Redis
  - Key: `news:{category}` → JSON array of latest articles (with TTL of 5 minutes).
  - Key: `session:{token}` → user session data.
- **Search Engine:** Elasticsearch cluster for full-text search across article titles and content.

7.4 Development Tools

- **Version Control:** Git (GitHub repository)
- **Package Management:** npm or Yarn for JavaScript dependencies.
- **Code Quality:** ESLint, Prettier, Husky (pre-commit hooks).
- **Project Management:** Jira or Trello for sprint boards and issue tracking.
- **Containerization:** Docker for consistent development environments; Docker Compose for multi-container services (app + database + cache).
- **Deployment:** Kubernetes (optional) or Docker Swarm; CI/CD pipelines to AWS/GCP/Azure.
- **API Documentation:** Swagger (OpenAPI) for interactive API docs.
- **Diagramming:** draw.io or Lucidchart for initial UML drafts.