Requirement Document

Project Name: Article Manager

Team Name: Gen-Z

---

1. BUSINESS REQUIREMENTS

1.1 INTRODUCTION

The "Gen-Z" team is building **Article Manager**, a web-based application designed to manage educational content for subjects like **Arts**, **Mathematics**, and **Technology**. The platform will enable different types of users (students, tutors, and administrators) to interact with educational articles, helping students access categorized, searchable information and allowing staff to manage the content dynamically.

1.2 STATEMENT OF PROBLEM OR NEED

Centralized Resource Hub:

- Centralized, categorized, and searchable reference materials in a digital format.

Content Management:

- Tutors and admins can add, edit, and delete articles.

Information Accessibility:

- Students can explore and search through categorized articles.

User Empowerment:

- Teachers and admins are empowered with tools for content creation and curation.

Cross-Platform Access:

- Accessible on both mobile and desktop via a single-page React application.

Security & Privacy:

- Secure access based on roles with password hashing and JWT-based authentication.

Continuous Improvement:

- Easy update of content and adaptable system design for future enhancements.

1.3 BUSINESS REQUIREMENTS

1.3.1 STAKEHOLDERS

- Students
- Tutors
- School Administrators
- Board of Trustees

1.3.2 CLIENTS

- End Users: Students, Tutors, and Administrators

1.3.3 FUNCTIONAL REQUIREMENTS

Students:

- Browse articles by category (Art, Mathematics, Technology)
- Search articles by keyword in title

Tutors:

- Add new articles
- Modify existing articles

Administrators:

- Add, modify, and delete articles

1.3.4 QUALITY REQUIREMENTS

Non-Functional:

- **Reliability:** High availability and consistency in results
- **Usability:** Intuitive UI for all users
- **Security:** Password hashing, JWT tokens, role-based access
- **Performance:** Fast response (< 2 seconds)
- **Scalability:** Handle increasing user load gracefully

---

2. BUSINESS SOLUTION

2.1 OPTIONS CONSIDERED

MongoDB:

- Flexible, schema-less, and scalable
- Ideal for document-based articles
- Future-proof with support for rich content (video/images)

MySQL:

- Mature and widely adopted
- Less suited for unstructured or multimedia content

Single Page Web App (React):

- Enables access via various devices
- Keyboard-friendly and easily accessible

Mobile App:

- Considered but discarded due to potential legal and usability issues in school environments

Node.js + Express + Mongoose:

- Simplifies development
- Good ecosystem for educational tools

2.2 RECOMMENDED SOLUTION

Final Stack:

- Frontend: React
- Backend: Node.js + Express
- Database: MongoDB + Mongoose

Justification:

- High flexibility and ease of growth
- Ideal for dynamic educational resources
- Adaptable to multimedia requirements

---

3. SOLUTION REQUIREMENTS

3.1 CONTEXT DIAGRAM

[To be added in the design document]

3.2 USER ROLES

- **Students:** View and search articles

- **Tutors:** Add/edit articles
- **Administrators:** Full CRUD permissions
- **Board of Trustees:** Read-only insights (optional)

3.3 DOMAIN MODEL

Entities:

- **User:** ID, Name, Email, Role
- **Article:** ID, Title, Category, Type, Metadata (born, year, etc.), Content
- **Subject:** ID, Name

3.4 USER STORIES

- As a student, I want to browse articles by category.
- As a student, I want to search articles by keyword in title.
- As a tutor, I want to add or edit articles.
- As an administrator, I want to add, edit, and delete articles.

3.5 NON-FUNCTIONAL REQUIREMENTS

3.5.1 SECURITY

- **Encryption:** HTTPS and SSL enforcement
- **Authentication:** JWT + bcrypt
- **Authorization:** Role-based access
- **Backup:** Weekly database backups

3.5.2 OTHER

- **Performance:** < 2 seconds response
- **Scalability:** Horizontal & vertical scaling
- **Accessibility:** WCAG-compliant
- **Availability:** 99.9% uptime
- **Testing:** Unit, Integration, and User Acceptance Tests

4. SCOPE

4.1 ITERATION 1: Planning and Architecture

- Define tech stack and system design
- Prepare database schema and ER model

4.2 ITERATION 2: Backend Development

- Create MongoDB collections for articles and users
- Import SampleData.xlsx into the database
- Implement validation and business logic

4.3 ITERATION 3: Article Features and Queries

- Create CRUD API endpoints
- Implement search and filter logic by category/title
- Add role-based logic for access

4.4 ITERATION 4: UI/UX + Auth

- Design frontend components using React
- Develop login/register pages with role logic
- Connect frontend with backend using Axios

---

5. NON-FUNCTIONAL REQUIREMENTS SUMMARY

- **Availability:** 24/7
- **Load Handling:** High concurrency support
- **Response Time:** Under 2 seconds
- **User Experience:** Modern, intuitive UI/UX
- **Data Protection:** Robust authorization + daily backups