📚Article Submission & Management Feature

Complete Documentation

📋Table of Contents

🕐Overview

A complete **Article Submission and Review System** for the **Law Nation** platform featuring:

- Guest article submission (no login required)
- Admin assignment workflow
- Editor review and approval
- Public preview and authenticated full access
- PDF download for logged-in users

🧱Features

1. Article Submission (Guest Access)

- ✅Submit articles without creating an account
- ✅PDF upload (max 10MB)
- ✅Mandatory and optional fields
- ✅Email confirmation on submission

2. Admin Management

- ✅View all submitted articles
- ✅Filter by status, category, author
- ✅Assign articles to editors
- ✅Delete articles

3. Editor Review

- ✅Option A: Direct approval
- ✅Option C: Upload corrected PDF → Approve
- ✅Track revision history

4. Article Access & Visibility

- ✅Public: Preview only (title, abstract, author)
- ✅Logged-in: Full article access + PDF download
- ✅Only approved articles visible to public

5. Email Notifications

- ✅Submission confirmation
- ✅Editor assignment notification
- ✅Author status updates
- ✅Approval notification
- ✅Correction notification

🗂️Database Schema

Article Model

```
model Article {
  id                String        @id @default(cuid())
  authorName        String
  authorEmail       String
  authorPhone       String?
  authorOrganization String?
  title             String
  category          String
  abstract          Text
  keywords          String?
  coAuthors         Text?
  remarksToEditor   Text?
  originalPdfUrl    String
  currentPdfUrl     String
  status            ArticleStatus @default(PENDING_ADMIN_REVIEW)
  assignedEditorId  String?
  assignedEditor    User?         @relation(fields: [assignedEditorId],
references: [id])
  submittedAt       DateTime      @default(now())
  reviewedAt        DateTime?
  approvedAt        DateTime?
  createdAt         DateTime      @default(now())
  updatedAt         DateTime      @updatedAt
  revisions         ArticleRevision[]
```

```
  @@index([status])
  @@index([assignedEditorId])
  @@index([authorEmail])
}
```

ArticleRevision Model

```
model ArticleRevision {
  id         String   @id @default(cuid())
  articleId  String
  article    Article  @relation(fields: [articleId], references: [id],
onDelete: Cascade)
  pdfUrl     String
  uploadedBy String?
  comments   Text?
  createdAt  DateTime @default(now())

  @@index([articleId])
}
```

ArticleStatus Enum

```
enum ArticleStatus {
  PENDING_ADMIN_REVIEW
  ASSIGNED_TO_EDITOR
  PENDING_APPROVAL
  APPROVED
  REJECTED
}
```

🔌API Endpoints

Public Endpoints

Submit Article (Guest)

`POST /api/articles/submit`

- Content-Type: `multipart/form-data`
- Required: pdf, authorName, authorEmail, title, category, abstract

---

Get Article Preview

`GET /api/articles/:id/preview`

Returns limited data (no PDF URLs).

Protected Endpoints (Authentication Required)

List Articles

`GET /api/articles`

Supports filtering, pagination, and role-based access.

Get Full Article

`GET /api/articles/:id`

Only returns **approved** articles.

Download Article PDF

`GET /api/articles/:id/download`

Returns downloadable PDF URL.

Assign Editor (Admin)

`PATCH /api/articles/:id/assign-editor`

Approve Article (Editor)

`PATCH /api/articles/:id/approve`

Upload Corrected PDF (Editor)

`PATCH /api/articles/:id/upload-corrected`

Delete Article (Admin)

`DELETE /api/articles/:id`

🕐Workflow

1. **Guest submits article** → `PENDING_ADMIN_REVIEW`

2. **Admin assigns editor** → `ASSIGNED_TO_EDITOR`
3. **Editor reviews**
4. Option A: Direct approve → `APPROVED`
5. Option C: Upload correction → `PENDING_APPROVAL` → Approve
6. **Article published & visible**

🔐 Access Control

Permissions

- `article.read`
- `article.write`
- `article.delete`

Roles

Role	Capabilities
Guest	Submit & preview
Logged-in User	Full access + download
Editor	Review & approve
Admin	Full control

📧 Email Notifications

Trigger	Recipient
Submission	Author
Editor Assignment	Editor
Status Updates	Author
Approval	Author
Correction	Author

Emails are async and non-blocking.

⚙️ Setup Instructions

1. Install dependencies
2. Configure environment variables
3. Generate Prisma client
4. Create permissions
5. Assign roles

6. Start server

🧪Testing Guide

Includes test cases for:

- Guest submission
- Admin assignment
- Editor approval
- PDF download
- Filtering & pagination

📊Feature Summary

- ✅Guest submission
- ✅Admin & editor workflows
- ✅RBAC permissions
- ✅Email notifications
- ✅Prisma + TypeScript
- ✅PDF storage

🗜️Production Checklist

- SMTP credentials
- Supabase storage
- Permissions setup
- Monitoring & logging
- Frontend integration

📁File Structure

```
backend/
├── prisma/
├── src/
│   ├── modules/article/
│   ├── middlewares/
│   └── utils/
└── .env
```

Status: COMPLETE

All features implemented and ready for testing and deployment.