Xuất sắc, Master Tu 👨‍✈️ — đây chính là phần **trái tim** của một hệ thống *tạp chí khoa học điện tử hiện đại*:

"Kho lưu trữ bài báo và cơ sở dữ liệu học thuật thống nhất."

Tôi sẽ hướng dẫn ngài xây dựng **chức năng Lưu trữ & Quản lý CSDL bài báo** theo kiến trúc thực chiến, hiện đại, đảm bảo:

- quản lý được **toàn bộ bài báo**,
- liên kết với **các số tạp chí (Issue)**,
- hỗ trợ **tìm kiếm, phân loại, gợi ý, trích xuất và phát hiện trùng lặp**,
- hoạt động ổn định trên **Next.js 14 + Prisma + PostgreSQL**.

🧭 1️⃣ Mục tiêu tổng thể

Chức năng	Mô tả ngắn gọn
🗃️ **Lưu trữ bài báo**	Quản lý tất cả bài viết theo metadata, file PDF, DOI, từ khóa, tác giả, lĩnh vực.
📚 **Liên kết số tạp chí**	Mỗi bài thuộc về một "Issue" (tập – số – năm).
🔍 **Tìm kiếm và tra cứu**	Full-text search (title, abstract, author, keyword).
🧠 **Gợi ý bài tương tự**	Tự động gợi ý bài cùng chủ đề, lĩnh vực, hoặc từ khóa.

Chức năng	Mô tả ngắn gọn
🧬 **Phát hiện trùng lặp**	So sánh nội dung, phát hiện trùng hoặc tương đồng cao.
🔒 **Quản trị**	Quản lý CRUD cho editor, reviewer, researcher.

🧩 2️⃣ Mô hình cơ sở dữ liệu (Prisma Schema)

```
model Issue {
  id          String   @id @default(cuid())
  volume      Int
  number      Int
  year        Int
  coverImage  String?
  publishedAt DateTime
  description String?
  articles    Article[]
}

model Article {
  id          String   @id @default(cuid())
  title       String
  abstract    String
```

```
  authors       String
  orgs          String?
  keywords      String[]
  category      String?
  doi           String?   @unique
  pdfUrl        String?
  issueId       String?
  issue         Issue?    @relation(fields: [issueId], references: [id])
  similarityHash String?  // phục vụ phát hiện trùng lặp
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt
}
```

💡 **Ưu điểm:**

- Dữ liệu rõ ràng, truy vấn 2 chiều (Issue ↔ Article).
- Sẵn sàng cho gợi ý, tìm kiếm, và đối chiếu trùng lặp.

⚙️ 3️⃣ **API cần thiết**

API	**Mục tiêu**
POST /api/articles	Thêm bài báo mới (metadata + file PDF).
GET /api/articles	Lấy danh sách bài báo (phân trang, lọc).

API	Mục tiêu
GET /api/articles/[id]	Chi tiết bài báo.
GET /api/articles/search?q=	Tìm kiếm toàn văn / từ khóa / tác giả.
POST /api/articles/similarity	So khớp bài mới với kho hiện tại (trùng lặp).
GET /api/issues	Danh sách số tạp chí.

💾 4️⃣ Tích hợp lưu trữ file PDF

1. **Upload → Lưu trong thư mục /public/pdfs/** (hoặc AWS S3, Google Cloud Storage).
2. **Lưu đường dẫn pdfUrl trong Article.**

Ví dụ API upload:

```
import { writeFile } from 'fs/promises';


export async function POST(req: Request) {
  const data = await req.formData();
  const file = data.get('file') as File;
  const bytes = await file.arrayBuffer();
  await writeFile(`./public/pdfs/${file.name}`, Buffer.from(bytes));
  return Response.json({ url: `/pdfs/${file.name}` });
}
```

🔍 5️⃣ Tìm kiếm & Tra cứu

🧱 Cách 1 – PostgreSQL Full-Text Search

```
SELECT * FROM "Article"
WHERE to_tsvector('simple', title || ' ' || abstract || ' ' || authors)
    @@ plainto_tsquery('simple', 'quốc phòng');
```

🧠 Cách 2 – ElasticSearch hoặc Meilisearch

Nếu > 100 000 bài, nên dùng search engine riêng. Next.js gọi API để lấy kết quả tức thì.

🧬 6️⃣ Gợi ý và Phát hiện trùng lặp

A. Hash-based

```
import { createHash } from 'crypto';
const hash = createHash('sha256')
  .update(article.abstract + article.title)
  .digest('hex');
```

So sánh hash với kho hiện có. Nếu khoảng cách Levenshtein < 0.1 → trùng lặp.

B. Semantic Similarity (AI)

Dùng Sentence Transformers (Python hoặc JS):

```
from sentence_transformers import SentenceTransformer, util
model = SentenceTransformer('all-MiniLM-L6-v2')
```

sim = util.cos_sim(model.encode(a1), model.encode(a2))

Nếu sim > 0.9 → nghi trùng hoặc đạo văn.

🖥️ 7️⃣ Giao diện trên Next.js

- **Trang /archives**: hiển thị tất cả Issue (bìa, năm).
- **Trang /archive/[issueId]**: danh sách bài báo trong số.
- **Trang /article/[id]**: chi tiết bài, nút tải PDF, gợi ý bài liên quan.
- **Trang /search**: hộp tìm kiếm toàn bộ kho bài báo.

🛡️ 8️⃣ Bảo mật và phân quyền

Vai trò	Quyền
Editor	Thêm / sửa / xuất bản Article & Issue
Reviewer	Xem, duyệt, nhận xét
Researcher	Chỉ xem, tìm kiếm
Guest	Truy cập public

🔁 9️⃣ Quy trình vận hành

1️⃣ Tác giả gửi bài.

2️⃣ Biên tập duyệt → gán vào Issue.

3️⃣ Khi xuất bản Issue → Article tự động hiển thị public.

4️⃣ Bài được lưu vĩnh viễn trong CSDL & có DOI.

5️⃣ Hệ thống AI chạy định kỳ kiểm tra trùng lặp.

🧠 10 Tóm tắt kiến trúc

Lớp	Công nghệ	Vai trò
Front-end	Next.js 14 (App Router + Server Components)	Giao diện web
API	Next.js API Routes / tRPC	Xử lý logic
Database	PostgreSQL + Prisma	Lưu trữ metadata
File Storage	S3 / local / CDN	PDF
Search	PostgreSQL FTS / ElasticSearch	Tra cứu
AI Layer	Sentence Transformers	Gợi ý & Trùng lặp

Nếu ngài muốn, tôi có thể **tạo bộ mã khung hoàn chỉnh (Next.js + Prisma + PostgreSQL)** cho chức năng này, bao gồm:

- Schema Prisma
- API Routes (articles + issues + search + similarity)
- Trang giao diện hiển thị và tìm kiếm
- Logic phát hiện trùng lặp cơ bản

Ngài có muốn tôi **tạo luôn bộ mã triển khai mẫu "Kho Lưu trữ Bài báo Học thuật"** cho hệ thống của ngài không, Master Tu?

Catch the Quantum Wave... Password: spinor

```
from fastapi import FastAPI, UploadFile, Form
from pydantic import BaseModel
from typing import List, Optional
from datetime import datetime
import hashlib
import os

app = FastAPI(title="Kho Lưu Trữ Bài Báo Học Thuật")

# ---------- Dữ liệu mô phỏng (có thể thay bằng CSDL thực tế) ----------
articles_db = []
issues_db = []

# ---------- Mô hình dữ liệu ----------
class Issue(BaseModel):
    id: str
    volume: int
    number: int
    year: int
    coverImage: Optional[str] = None
```

```
    publishedAt: datetime
    description: Optional[str] = None


class Article(BaseModel):
    id: str
    title: str
    abstract: str
    authors: str
    orgs: Optional[str]
    keywords: List[str]
    category: Optional[str]
    doi: Optional[str]
    pdfUrl: Optional[str]
    issueId: Optional[str]
    similarityHash: Optional[str]
    createdAt: datetime
    updatedAt: datetime


# ---------- Chức năng API ----------


@app.post("/api/issues")
```

```
def create_issue(issue: Issue):
    issues_db.append(issue.dict())
    return {"message": "Issue created successfully", "issue": issue}


@app.get("/api/issues")
def list_issues():
    return issues_db


@app.post("/api/articles")
def create_article(title: str = Form(...), abstract: str = Form(...), authors: str = Form(...), orgs: str = Form(None), keywords: str = Form(...), category: str = Form(None), doi: str = Form(None), issueId: str = Form(None), file: UploadFile = None):
    # Luu file PDF nếu có
    pdf_path = None
    if file:
        pdf_dir = "pdfs"
        os.makedirs(pdf_dir, exist_ok=True)
        pdf_path = os.path.join(pdf_dir, file.filename)
        with open(pdf_path, "wb") as f:
            f.write(file.file.read())
```

```
# Tạo hash phát hiện trùng lặp
similarity_data = title + abstract + authors
similarityHash = hashlib.sha256(similarity_data.encode()).hexdigest()

article = Article(
    id=hashlib.md5(similarity_data.encode()).hexdigest(),
    title=title,
    abstract=abstract,
    authors=authors,
    orgs=orgs,
    keywords=keywords.split(","),
    category=category,
    doi=doi,
    pdfUrl=pdf_path,
    issueId=issueId,
    similarityHash=similarityHash,
    createdAt=datetime.utcnow(),
    updatedAt=datetime.utcnow()
)

articles_db.append(article.dict())
```

```
    return {"message": "Article added successfully", "article": article}


@app.get("/api/articles")
def list_articles():
    return articles_db


@app.get("/api/articles/search")
def search_articles(q: str):
    results = [a for a in articles_db if q.lower() in a["title"].lower() or q.lower() in a["abstract"].lower() or q.lower() in a["authors"].lower()]
    return {"results": results, "count": len(results)}


@app.post("/api/articles/similarity")
def detect_similarity(title: str = Form(...), abstract: str = Form(...), authors: str = Form(...)):
    new_hash = hashlib.sha256((title + abstract + authors).encode()).hexdigest()
    matches = []
    for a in articles_db:
        if a["similarityHash"] == new_hash:
            matches.append(a)
    return {"similar": matches, "count": len(matches)}
```

```
# ---------- Khởi chạy ví dụ ----------

# Chạy bằng lệnh: uvicorn this_file_name:app –reload
```

```
import React, { useState, useEffect } from "react";
import { Card, CardContent } from "@/components/ui/card";
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { Textarea } from "@/components/ui/textarea";

export default function ArticleRepository() {
  const [articles, setArticles] = useState([]);
  const [query, setQuery] = useState("");
  const [loading, setLoading] = useState(false);
  const [uploading, setUploading] = useState(false);

  // Form dữ liệu bài báo mới
  const [newArticle, setNewArticle] = useState({
    title: "",
    abstract: "",
    authors: "",
    orgs: "",
    keywords: "",
    category: "",
    doi: "",
    file: null,
  });

  useEffect(() => {
    fetchArticles();
  }, []);
```

```
  const fetchArticles = async () => {
    setLoading(true);
    const res = await fetch("/api/articles");
    const data = await res.json();
    setArticles(data);
    setLoading(false);
  };

  const handleSearch = async () => {
    if (!query.trim()) {
      fetchArticles();
      return;
    }
    setLoading(true);
    const res = await
fetch(`/api/articles/search?q=${encodeURIComponent(query)}`);
    const data = await res.json();
    setArticles(data.results);
    setLoading(false);
  };

  const handleUpload = async (e) => {
    e.preventDefault();
    setUploading(true);

    const formData = new FormData();
```

```
    for (const key in newArticle) {
      if (key === "file" && newArticle.file) {
        formData.append("file", newArticle.file);
      } else {
        formData.append(key, newArticle[key]);
      }
    }

    const res = await fetch("/api/articles", {
      method: "POST",
      body: formData,
    });

    const data = await res.json();
    setUploading(false);
    if (data.article) {
      alert("Thêm bài báo thành công!");
      fetchArticles();
      setNewArticle({ title: "", abstract: "", authors: "", orgs: "", keywords: "",
category: "", doi: "", file: null });
    } else {
      alert("Lỗi khi thêm bài báo!");
    }
  };

  return (
    <div className="container mx-auto p-6 space-y-6">
```

```
<h1 className="text-3xl font-bold">Kho Lưu Trữ Bài Báo Học Thuật</h1>

{/* --- Tìm kiếm --- */}
<div className="flex space-x-2">
  <Input
    placeholder="Tìm kiếm bài báo, tác giả hoặc từ khóa..."
    value={query}
    onChange={(e) => setQuery(e.target.value)}
  />
  <Button onClick={handleSearch}>Tìm kiếm</Button>
</div>

{/* --- Biểu mẫu thêm bài báo mới --- */}
<Card className="p-6 bg-gray-50">
  <h2 className="text-xl font-semibold mb-4">Thêm Bài Báo Mới</h2>
  <form onSubmit={handleUpload} className="space-y-3">
    <Input placeholder="Tiêu đề bài báo" value={newArticle.title}
onChange={(e) => setNewArticle({ ...newArticle, title: e.target.value })} required
/>
    <Textarea placeholder="Tóm tắt" value={newArticle.abstract}
onChange={(e) => setNewArticle({ ...newArticle, abstract: e.target.value })}
required />
    <Input placeholder="Tác giả" value={newArticle.authors} onChange={(e)
=> setNewArticle({ ...newArticle, authors: e.target.value })} required />
    <Input placeholder="Cơ quan công tác" value={newArticle.orgs}
onChange={(e) => setNewArticle({ ...newArticle, orgs: e.target.value })} />
    <Input placeholder="Từ khóa (phân cách bằng dấu phẩy)"
value={newArticle.keywords} onChange={(e) => setNewArticle({ ...newArticle,
keywords: e.target.value })} />
```

```
        <Input placeholder="Lĩnh vực nghiên cứu" value={newArticle.category}
onChange={(e) => setNewArticle({ ...newArticle, category: e.target.value })} />

        <Input placeholder="DOI" value={newArticle.doi} onChange={(e) =>
setNewArticle({ ...newArticle, doi: e.target.value })} />

        <Input type="file" accept="application/pdf" onChange={(e) =>
setNewArticle({ ...newArticle, file: e.target.files[0] })} />

        <Button type="submit" disabled={uploading}>{uploading ? "Đang tải lên..."
: "Thêm bài báo"}</Button>

      </form>

    </Card>


    {/* --- Danh sách bài báo --- */}

    {loading && <p>Đang tải dữ liệu...</p>}

    <div className="grid md:grid-cols-2 lg:grid-cols-3 gap-4">

      {articles.map((article) => (

        <Card key={article.id} className="shadow-md hover:shadow-lg transition-
all">

          <CardContent className="p-4">

            <h2 className="text-lg font-semibold mb-2">{article.title}</h2>

            <p className="text-sm text-gray-700 mb-2">{article.authors}</p>

            <p className="text-sm text-gray-600 line-clamp-
3">{article.abstract}</p>

            <div className="mt-2 flex flex-wrap gap-1">

              {article.keywords.map((kw) => (

                <span key={kw} className="bg-gray-200 text-gray-700 px-2 py-1
rounded text-xs">

                  {kw}

                </span>

              ))}
```

```
            </div>
            {article.pdfUrl && (
              <a href={article.pdfUrl} target="_blank" className="block text-blue-600 text-sm mt-3 hover:underline">
                Xem bài báo (PDF)
              </a>
            )}
          </CardContent>
        </Card>
      ))}
    </div>

    {articles.length === 0 && !loading && (
      <p className="text-gray-600 italic">Không có bài báo nào phù hợp với tìm kiếm.</p>
    )}
  </div>
 );
}
```

```
import React, { useState, useEffect } from "react";
import { Card, CardContent } from "@/components/ui/card";
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { Textarea } from "@/components/ui/textarea";

export default function ArticleRepository() {
  const [articles, setArticles] = useState([]);
  const [query, setQuery] = useState("");
  const [loading, setLoading] = useState(false);
  const [uploading, setUploading] = useState(false);

  // Form dữ liệu bài báo mới
  const [newArticle, setNewArticle] = useState({
    title: "",
    abstract: "",
    authors: "",
    orgs: "",
    keywords: "",
    category: "",
    doi: "",
    file: null,
  });

  useEffect(() => {
    fetchArticles();
  }, []);
```

```
  const fetchArticles = async () => {
    setLoading(true);
    const res = await fetch("/api/articles");
    const data = await res.json();
    setArticles(data);
    setLoading(false);
  };

  const handleSearch = async () => {
    if (!query.trim()) {
      fetchArticles();
      return;
    }
    setLoading(true);
    const res = await
fetch(`/api/articles/search?q=${encodeURIComponent(query)}`);
    const data = await res.json();
    setArticles(data.results);
    setLoading(false);
  };

  const handleUpload = async (e) => {
    e.preventDefault();
    setUploading(true);

    const formData = new FormData();
```

```
    for (const key in newArticle) {
      if (key === "file" && newArticle.file) {
        formData.append("file", newArticle.file);
      } else {
        formData.append(key, newArticle[key]);
      }
    }

    const res = await fetch("/api/articles", {
      method: "POST",
      body: formData,
    });

    const data = await res.json();
    setUploading(false);
    if (data.article) {
      alert("Thêm bài báo thành công!");
      fetchArticles();
      setNewArticle({ title: "", abstract: "", authors: "", orgs: "", keywords: "",
category: "", doi: "", file: null });
    } else {
      alert("Lỗi khi thêm bài báo!");
    }
  };

  return (
    <div className="container mx-auto p-6 space-y-6">
```

```
<h1 className="text-3xl font-bold">Kho Lưu Trữ Bài Báo Học Thuật</h1>

{/* --- Tìm kiếm --- */}
<div className="flex space-x-2">
  <Input
    placeholder="Tìm kiếm bài báo, tác giả hoặc từ khóa..."
    value={query}
    onChange={(e) => setQuery(e.target.value)}
  />
  <Button onClick={handleSearch}>Tìm kiếm</Button>
</div>

{/* --- Biểu mẫu thêm bài báo mới --- */}
<Card className="p-6 bg-gray-50">
  <h2 className="text-xl font-semibold mb-4">Thêm Bài Báo Mới</h2>
  <form onSubmit={handleUpload} className="space-y-3">
    <Input placeholder="Tiêu đề bài báo" value={newArticle.title}
onChange={(e) => setNewArticle({ ...newArticle, title: e.target.value })} required
/>
    <Textarea placeholder="Tóm tắt" value={newArticle.abstract}
onChange={(e) => setNewArticle({ ...newArticle, abstract: e.target.value })}
required />
    <Input placeholder="Tác giả" value={newArticle.authors} onChange={(e)
=> setNewArticle({ ...newArticle, authors: e.target.value })} required />
    <Input placeholder="Cơ quan công tác" value={newArticle.orgs}
onChange={(e) => setNewArticle({ ...newArticle, orgs: e.target.value })} />
    <Input placeholder="Từ khóa (phân cách bằng dấu phẩy)"
value={newArticle.keywords} onChange={(e) => setNewArticle({ ...newArticle,
keywords: e.target.value })} />
```

```
        <Input placeholder="Lĩnh vực nghiên cứu" value={newArticle.category}
onChange={(e) => setNewArticle({ ...newArticle, category: e.target.value })} />

        <Input placeholder="DOI" value={newArticle.doi} onChange={(e) =>
setNewArticle({ ...newArticle, doi: e.target.value })} />

        <Input type="file" accept="application/pdf" onChange={(e) =>
setNewArticle({ ...newArticle, file: e.target.files[0] })} />

        <Button type="submit" disabled={uploading}>{uploading ? "Đang tải lên..."
: "Thêm bài báo"}</Button>

      </form>

    </Card>


    {/* --- Danh sách bài báo --- */}

    {loading && <p>Đang tải dữ liệu...</p>}

    <div className="grid md:grid-cols-2 lg:grid-cols-3 gap-4">

      {articles.map((article) => (

        <Card key={article.id} className="shadow-md hover:shadow-lg transition-
all">

          <CardContent className="p-4">

            <h2 className="text-lg font-semibold mb-2">{article.title}</h2>

            <p className="text-sm text-gray-700 mb-2">{article.authors}</p>

            <p className="text-sm text-gray-600 line-clamp-
3">{article.abstract}</p>

            <div className="mt-2 flex flex-wrap gap-1">

              {article.keywords.map((kw) => (

                <span key={kw} className="bg-gray-200 text-gray-700 px-2 py-1
rounded text-xs">

                  {kw}

                </span>

              ))}
```

```
            </div>
            {article.pdfUrl && (
              <a href={article.pdfUrl} target="_blank" className="block text-blue-600 text-sm mt-3 hover:underline">
                Xem bài báo (PDF)
              </a>
            )}
          </CardContent>
        </Card>
      ))}
    </div>

    {articles.length === 0 && !loading && (
      <p className="text-gray-600 italic">Không có bài báo nào phù hợp với tìm kiếm.</p>
    )}
  </div>
 );
}
```