Research Papers Manager

سیستم مدیریت مقالات تحقیقاتی دانشگاه

این سوال از شما می‌خواهد که یک **سیستم مدیریت مقالات تحقیقاتی دانشگاه** را برای مدیریت مقالات تحقیقاتی پیاده‌سازی کنید. سیستم از **MongoDB** برای ذخیره‌سازی دائمی پروفایل کاربران، متادیتا مقالات، و روابط استنادی، و از **Redis** برای کش کردن نتایج جستجو، ردیابی تعداد بازدیدهای بی‌درنگ مقالات، و پیاده‌سازی جدول هش برای بررسی در دسترس بودن نام کاربری به صورت بی‌درنگ در زمان ثبت‌نام استفاده می‌کند. این سوال روی ورکفلو دیتابیس شامل طراحی اسکیما، ایندکس‌گذاری، کوئری‌نویسی، و ادغام Redis برای بهینه‌سازی عملکرد تمرکز دارد. احراز هویت کاربران مینیمال است و فقط شامل ثبت‌نام و ورود با نام کاربری یکتا می‌شود.

الزامات سیستم

الزامات عملکردی

۱. **مدیریت کاربران**:
- **ثبت‌نام**: کاربران با نام کاربری یکتا، نام، ایمیل، رمز عبور، و دپارتمان ثبت‌نام می‌کنند.
  - بررسی در دسترس بودن نام کاربری به صورت بی‌درنگ با استفاده از جدول هش Redis.
  - اگر نام کاربری قبلاً گرفته شده باشد، خطا برگردانده شود؛ در غیر این صورت، کاربر ذخیره شده و نام کاربری به عنوان گرفته شده علامت‌گذاری شود.
- **ورود**: کاربران با نام کاربری و رمز عبور وارد سیستم شده و شناسه کاربر (user ID) برای ردیابی سشن دریافت می‌کنند.
- احراز هویت مینیمال است و فقط برای وصل کردن مقالات به کاربران استفاده می‌شود.

۲. **مدیریت مقالات**:
- کاربران احراز هویت شده می‌توانند مقالات را با متادیتا زیر آپلود کنند:
  - `title` : رشته (اجباری، حداکثر 200 کاراکتر)
  - `authors` : لیست (1 تا 5 رشته، هر کدام حداکثر 100 کاراکتر)
  - `abstract` : رشته (اجباری، حداکثر 1000 کاراکتر)
  - `publication_date` : رشته تاریخ ISO (مثال: "15-05-2023")
  - `journal_conference` : رشته (حداکثر 200 کاراکتر)
  - `keywords` : لیست (1 تا 5 رشته، هر کدام حداکثر 50 کاراکتر)
  - `citations` : لیست (0 تا 5 شناسه مقاله معتبر از مجموعه Papers)
- مقالات به شناسه کاربر آپلودکننده مرتبط می‌شوند.

۳. **قابلیت جستجو**:
- جستجوی متنی در عنوان، چکیده، و کلمات کلیدی با استفاده از جستجوی متنی MongoDB.
- پشتیبانی از مرتب‌سازی بر اساس تاریخ انتشار (صعودی یا نزولی) یا relevance (امتیاز متنی).
- بازگشت تمام نتایج منطبق (بدون صفحه‌بندی).

- کش کردن نتایج جستجو در Redis برای کاهش لود MongoDB.

۴. **مدیریت استنادها**:

- ذخیره روابط استنادی (مقاله A به مقاله B ارجاع می‌دهد) در یک مجموعه اختصاصی.
- محاسبه تعداد استنادها برای هر مقاله (تعداد مقالاتی که به آن ارجاع داده‌اند).

۵. **متریک‌های بی‌درنگ**:

- ردیابی تعداد بازدیدهای مقالات در Redis، افزایش در هر بازدید.
- سینک کردن تعداد بازدیدها با MongoDB هر 10 دقیقه.

۶. **وارد کردن داده‌ها**:

- ارائه اسکریپتی برای پر کردن MongoDB با:
  - 100 کاربر با نام‌های کاربری یکتا، نام، ایمیل، رمزهای عبور هش‌شده، و دپارتمان‌ها.
  - 1,000 مقاله با متادیتا تصادفی و شناسه‌های بارگذارنده.
  - استنادهای تصادفی (0 تا 5 برای هر مقاله) که به مقالات دیگر ارجاع دارند.

۷. **APIها**:

- پیاده‌سازی پنج endpoint RESTful برای ثبت‌نام، ورود، آپلود مقاله، جستجو، و جزئیات مقاله، با تعاملات دیتابیس مشخص.

استک تکنولوژی

- **دیتابیس**: MongoDB (نسخه 7.0 یا بالاتر) برای ذخیره دائمی. استفاده از PyMongo (Python) یا Mongoose (Node.js).
- **کش**: Redis (نسخه 7.2 یا بالاتر) برای کش، تعداد بازدیدها، و جدول هش نام کاربری. استفاده از redis-py (Python) یا ioredis (Node.js).
- **زبان برنامه‌نویسی**: Python (با Flask) یا Node.js (با Express.js). Python توصیه می‌شود.
- **فریم‌ورک وب**: استفاده از Flask (Python) یا Express.js (Node.js) برای APIها.
- **احراز هویت**: احراز هویت ساده مبتنی بر سشن (بازگشت شناسه کاربر در ورود، استفاده در هدر `X-User-ID`).
- **تولید داده**: استفاده از Faker (Python، نسخه 22.0.0) یا faker-js (Node.js، نسخه 8.4.1).
- **هش رمز عبور**: استفاده از bcrypt (Python، نسخه 4.1.2) یا bcryptjs (Node.js، نسخه 2.4.3).

توجه: استفاده از دیگر زبان‌های برنامه‌نویسی و باقی فریم‌ورک‌ها مجاز است.

اسکیمای دیتابیس

مجموعه‌های MongoDB

۱. **Users**:

- `_id` : ObjectId (شناسه یکتا)
- `username` : String (یکتا، 3–20 کاراکتر، حروف و اعداد با زیرخط)
- `name` : String (حداکثر 100 کاراکتر)

- `email` : String (حداکثر 100 کاراکتر، فرمت ایمیل معتبر)
- `password` : String (هش‌شده با bcrypt)
- `department` : String (حداکثر 100 کاراکتر)
- **Indexes**:
  - ایندکس یکتا روی `username` : `db.users.createIndex({ "username": 1 }, { unique: true })`

۲. **Papers**:

- `_id` : ObjectId (شناسه یکتا)
- `title` : String (اجباری، حداکثر 200 کاراکتر)
- `authors` : [String] (رشته، هر کدام حداکثر 100 کاراکتر 1–5)
- `abstract` : String (اجباری، حداکثر 1000 کاراکتر)
- `publication_date` : Date (فرمت ISO، مثال: ISODate("2023-05-15"))
- `journal_conference` : String (حداکثر 200 کاراکتر)
- `keywords` : [String] (رشته، هر کدام حداکثر 50 کاراکتر 1–5)
- `uploaded_by` : ObjectId (ارجاع به مجموعه Users)
- `views` : Number (پیش‌فرض 0، همگام‌سازی از Redis)
- **Indexes**:
  - ایندکس متنی روی `title`, `abstract`, `keywords` : `db.papers.createIndex({ "title": "text", "abstract": "text", "keywords": "text" })`

۳. **Citations**:

- `_id` : ObjectId (شناسه یکتا)
- `paper_id` : ObjectId (ارجاع به مقاله ارجاع‌دهنده، Papers)
- `cited_paper_id` : ObjectId (ارجاع به مقاله ارجاع‌شده، Papers)
- **Indexes**:
  - ایندکس روی `cited_paper_id` : `db.citations.createIndex({ "cited_paper_id": 1 })`

استفاده از Redis

۱. **جدول هش نام‌های کاربری**:

- **کلید**: `usernames`
- **ساختار**: هش که فیلد آن `username` و مقدار آن `1` (نشانه گرفته شدن) است.
- **مثال**: `HSET usernames johndoe123 1`
- **عملیات**:
  - بررسی در دسترس بودن: `HEXISTS usernames <username>`
  - اگر `0` باشد، نام کاربری در دسترس است؛ کاربر را درج کرده و `HSET usernames <username> 1` را تنظیم کنید.

- اگر `1` باشد، خطای 409 برگردانید ("نام کاربری گرفته شده است").

۲. **کش نتایج جستجو**:

- **فرمت کلید**: `search:<search_term>:<sort_by>:<order>`
- **مثال**: `search:machine learning:publication_date:desc`
- **TTL**: 300 ثانیه (5 دقیقه)
- **داده**: رشته JSON از نتایج جستجو (آرایه‌ای از اشیاء مقاله)
- **عملیات**:
  - بررسی کلید با `GET` .
  - اگر وجود دارد، JSON را تجزیه کرده و برگردانید.
  - اگر وجود ندارد، MongoDB را کوئری کرده، JSON را با `SETEX` ذخیره کنید، و برگردانید.

۳. **تعداد بازدیدهای مقاله**:

- **فرمت کلید**: `paper_views:<paper_id>`
- **مثال**: `paper_views:507f1f77bcf86cd799439011`
- **عملیات**:
  - افزایش با `INCR` در هر بازدید.
  - بازیابی با `GET` برای نمایش.
- **همگام‌سازی**: هر 10 دقیقه، `views` در مجموعه Papers را به‌روزرسانی کرده و کلید Redis را به 0 ریست کنید.

اندپوینت‌های API

`/signup`

- **Method**: POST
- **Description**: ثبت‌نام کاربر با نام کاربری یکتا
- **Headers**: None
- **Body**: `{ "username": string, "name": string, "email": string, "password": string, "department": string }`
- **Response**: `201 Created` , `{ "message": "User registered", "user_id": string }`
- **Status Codes**: 201, 400, 409

`/login`

- **Method**: POST
- **Description**: ورود و بازگشت شناسه کاربر
- **Headers**: None
- **Body**: `{ "username": string, "password": string }`
- **Response**: `200 OK` , `{ "message": "Login successful", "user_id": string }`

- **Status Codes**: 200, 400, 401

`/papers` (POST)

- **Method**: POST
- **Description**: بارگذاری مقاله جدید
- **Headers**: `X-User-ID: <user_id>`
- **Body**: `{ "title": string, "authors": [string], "abstract": string, "publication_date": string, "journal_conference": string, "keywords": [string], "citations": [string] }`
- **Response**: `201 Created`, `{ "message": "Paper uploaded", "paper_id": string }`
- **Status Codes**: 201, 400, 401, 404

`/papers` (GET)

- **Method**: GET
- **Description**: جستجوی مقالات بر اساس پرس‌وجو
- **Headers**: None
- **Body**: Query params: `?search=string`, `?sort_by=string` (relevance یا publication_date), `?order=asc|desc`
- **Response**: `200 OK`, `{ "papers": [ { "id": string, "title": string, "authors": [string], "publication_date": string, "journal_conference": string, "keywords": [string] } ] }`
- **Status Codes**: 200, 400

`/papers/{paper_id}`

- **Method**: GET
- **Description**: دریافت جزئیات مقاله با استنادها و بازدیدها
- **Headers**: None
- **Body**: None
- **Response**: `200 OK`, `{ "id": string, "title": string, "authors": [string], "abstract": string, "publication_date": string, "journal_conference": string, "keywords": [string], "citation_count": int, "views": int }`
- **Status Codes**: 200, 404

ورک‌فلو دیتابیس API

1. **POST /signup**:
    - **اعتبارسنجی ورودی**:

- `username` : 3–20 کاراکتر، حروف و اعداد با زیرخط.
- `name` , `email` , `department` : غیرخالی، در محدوده طول.
- `password` : حداقل 8 کاراکتر.

- **بررسی Redis**: `HEXISTS usernames <username>`
  - اگر `1` باشد، خطای 409 برگردانید ("نام کاربری گرفته شده است").
  - اگر `0` باشد، ادامه دهید.
- **عملیات پایگاه داده**:
  - هش کردن رمز عبور با bcrypt.
  - درج در مجموعه Users.
  - تنظیم در Redis: `HSET usernames <username> 1` .
- **پاسخ**: بازگشت 201 با `{ "message": "User registered", "user_id": "<_id>" }` .
- **خطاها**: 400 (ورودی نامعتبر)، 409 (نام کاربری تکراری).

۲. **POST /login**:

- **اعتبارسنجی ورودی**: `username` و `password` غیرخالی.
- **عملیات پایگاه داده**:
  - کوئری Users با `username` .
  - تأیید رمز عبور با bcrypt.
- **پاسخ**: بازگشت 200 با `{ "message": "Login successful", "user_id": "<_id>" }` .
- **خطاها**: 400 (ورودی نامعتبر)، 401 (اعتبارنامه نامعتبر).

۳. **POST /papers**:

- **اعتبارسنجی هدر**: بررسی `X-User-ID` با کوئری در Users.
- **اعتبارسنجی ورودی**:
  - `title` , `abstract` : غیرخالی، در محدوده طول.
  - `authors` , `keywords` : 1–5 آیتم، در محدوده طول.
  - `publication_date` : تاریخ ISO معتبر.
  - `citations` : 0–5 شناسه مقاله معتبر (بررسی در مجموعه Papers).
- **عملیات پایگاه داده**:
  - درج مقاله در Papers و `views: 0` با `uploaded_by` برابر با `_id` کاربر.
  - برای هر شناسه استناد، درج `{ paper_id: new_paper_id, cited_paper_id:` `cited_id }` در Citations.
- **پاسخ**: بازگشت 201 با `{ "message": "Paper uploaded", "paper_id": "<_id>" }` .
- **خطاها**: 400 (ورودی نامعتبر)، 401 (شناسه کاربر نامعتبر)، 404 (شناسه استناد نامعتبر).

۴. **GET /papers**:

- **اعتبارسنجی پارامترهای کوئری**:
  - `search` : رشته اختیاری (پیش‌فرض خالی).
  - `sort_by` : `publication_date` (`relevance` پیش‌فرض) یا `relevance`.

- `order` : `asc` (`desc` (پیش‌فرض) یا `desc`.
- **عملیات Redis**: با `search:<search_term>:<sort_by>:<order>` بررسی کلید `GET`.
  - اگر وجود دارد، JSON را تجزیه کرده و بازگشت 200 با `{ "papers": [...] }`.
- **عملیات MongoDB**:
  - کوئری Papers با `$text: { $search: search_term }`.
  - مرتب‌سازی بر اساس `textScore` (اگر `relevance`) یا `publication_date`.
- **عملیات Redis**: ذخیره نتایج در Redis با `SETEX search:<search_term>:<sort_by>:<order> 300 <JSON>`.
- **پاسخ**: بازگشت 200 با `{ "papers": [...] }`.
- **خطاها**: 400 (پارامترهای کوئری نامعتبر).

۵. **GET /papers/{paper_id}**:

- **عملیات MongoDB**:
  - کوئری Papers با `_id`.
  - شمارش اسناد در Citations است `paper_id` برابر با `cited_paper_id` که.
- **عملیات Redis**:
  - افزایش `INCR` با `paper_views:<paper_id>`.
  - بازیابی تعداد بازدید با `GET` (پیش‌فرض 0 اگر وجود ندارد).
- **پاسخ**: بازگشت 200 با جزئیات مقاله، `citation_count` ، و `views`.
- **خطاها**: 404 (مقاله یافت نشد).

تسک پس‌زمینه

- **هدف**: همگام‌سازی تعداد بازدیدهای Redis با MongoDB هر 10 دقیقه.
- **پیاده‌سازی**:
  - بازیابی تمام کلیدهای `paper_views:*` (استفاده از `KEYS paper_views:*` یا ردیابی شناسه‌های مقالات).
  - برای هر کلید، تعداد را با `GET` بازیابی کنید، فیلد `views` در Papers را با `$inc: { views: count }` به‌روزرسانی کنید، و کلید Redis را با `SET <key> 0` بازنشانی کنید.
- **زمان‌بند**: استفاده از APScheduler (Python، نسخه 3.10.1) یا node-cron (Node.js، نسخه 3.0.2).

اسکریپت تولید و وارد کردن دادهٔ جعلی

- **الزامات**:
  - تولید 100 کاربر:
    - `username` : یکتا، 3–20 کاراکتر، حروف و اعداد با زیرخط.
    - `name` : تصادفی (حداکثر 100 کاراکتر).
    - `email` : ایمیل معتبر تصادفی (حداکثر 100 کاراکتر).
    - `password` : هش‌شده با bcrypt، تصادفی 8–12 کاراکتر.

- `department` : تصادفی (حداکثر 100 کاراکتر).
  - تولید 1,000 مقاله:
    - `title` : جمله تصادفی (6–10 کلمه، حداکثر 200 کاراکتر).
    - `authors` : 1–5 نام تصادفی (هر کدام حداکثر 100 کاراکتر).
    - `abstract` : پاراگراف تصادفی (حداکثر 1000 کاراکتر).
    - `publication_date` : تاریخ تصادفی بین 2015-06-05 و 2025-06-05.
    - `journal_conference` : نام تصادفی (حداکثر 200 کاراکتر).
    - `keywords` : 1–5 کلمه تصادفی (هر کدام حداکثر 50 کاراکتر).
    - `uploaded_by` : شناسه کاربر تصادفی از Users.
    - `views` : 0.
  - تولید استنادها:
    - برای هر مقاله، 0–5 مقاله دیگر به صورت تصادفی انتخاب کنید (بدون خود-استناد).
    - درج در مجموعه Citations.
  - به‌روزرسانی جدول هش Redis `usernames` با تمام نام‌های کاربری (`HSET usernames <username> 1`).
- **کتابخانه**: استفاده از Faker (Python، نسخه 22.0.0) یا faker-js (Node.js، نسخه 8.4.1).

فرمت سابمیت نهایی

- ارسال کل سورس کد
- ارسال ویدیو دمو از پروژه به همراه توضیح مختصر از کدها
- نوشتن کوئری‌های مونگو بدون استفاده از ORM