Find That Book

A Technical Challenge

Overview

Build a small application for a library discovery workflow. Given a messy, plain-text blob (title and/or author and/or keywords), find which book it most likely refers to and present the results in a simple modern UI backed by a .NET 8 Web API. This project should show your full-stack skills, real LLM usage, text processing, clean architecture, and product thinking.

Functionality

Reader queries are inconsistent: sometimes they include just an author, sometimes a title, sometimes both with extra noise (e.g., mark huckleberry, twilight meyer, tolkien hobbit illustrated deluxe 1937). Your service should:

1. Accept plain text queries (messy blob)
    a. Sparse: only a title or only an author (e.g., dickens, tale two cities)
    b. Dense/noisy: author + title + extra tokens (e.g., tolkien hobbit illustrated deluxe 1937)
    c. Ambiguous: partial names or character hints (e.g., mark huckleberry, austen bennet)
2. Use AI to extract useful information
    a. Field extraction/normalization -> produce hypotheses like { title?, author?, keywords[] } from the blob
    b. Short "why it matched" explanation for each returned candidate (1–2 sentences), grounded in fields you fetched
    c. Optional re-ranking of top candidates into a final ordered list
3. Search Open Library (https://openlibrary.org/dev/docs/api/search) for candidates
    a. Use /search.json (works/editions) and fetch details as needed:
        i. /works/{work_id}.json
        ii. /authors/{author_id}.json
        iii. /authors/{author_id}/works.json
    b. Normalize inputs (lowercase, punctuation/diacritics, partials).
        i. Handle variants like subtitles (e.g., The Hobbit vs. There and Back Again).
        ii. De-duplicate to canonical works and resolve primary authors from `works.authors`
4. Apply a matching hierarchy
    a. Exact/normalized title + primary author match (strongest)
    b. Exact/normalized title + contributor-only author (lower rank)
    c. Near-match title + author match (candidate)
    d. Author-only fallback -> return top works by that author
    e. If no clear winner, return up to 5 ordered candidates with explanations

Data quality note

Open Library may list contributors (illustrators, editors, adaptors) in author_name alongside the primary author. Your matcher should prioritize primary authors (from canonical work records) and treat contributors as lower signal. Explanations should highlight this (e.g., “Exact title; Tolkien is primary author; Dixon listed as adaptor.”).

Return results

For each candidate, return:

- Book details: title, primary author(s), first publish year, Open Library IDs/links, cover image if available
- A one-sentence explanation (“why this book”) that cites concrete fields

Output is expected to be an ordered list of potential matches; numeric confidence is not required. For each item in the list, return:

```
```
{
  "title": "The Hobbit",
  "author": "J.R.R. Tolkien",
  "first_publish_year": 1937,
  "explanation": "Exact title match; Tolkien is primary author; Dixon listed as adaptor."
}
```
```

Evaluation Criteria

We will evaluate your submission based on:

- Code quality, organization, and architecture
- API design
- Approach to integrating AI
- Error handling and edge cases
- Testing strategy
- Communication
- Creative problem solving

Submission Guidelines

- Share a GitHub repository link with your code.
- Include a README with:
    - Set up and running instructions
    - Overview of your implementation
    - Any assumptions or design decisions you made
    - Description of features implemented
    - Explanation of your testing strategy
    - Future improvements you would make with more time
- AI API setup:
    - By default, use Gemini (https://ai.google.dev/gemini-api/docs/api-key) (free tier available)
    - If you choose a different model, you must clearly document this choice in your README.
    - Provide clear instructions in your README for setting up keys so we can run your solution with ease.
- Deploy a live demo (if possible)

Time Expectation

This challenge is designed to take approximately 4-6 hours, but we have left some requirements open-ended. We respect your time and don't expect a perfectly polished product - focus on showing your technical skills and approach. Please be prepared to present your project.

We encourage you to reach out if you have questions about the requirements, consider us your stakeholder. We're excited to see your creativity and technical skills in action!