Frontend Development Test Document

Project Overview

You are required to build a small frontend application that fetches and displays research paper data from the provided API. The app should showcase best practices in fetching, displaying, and interacting with data, with a focus on speed, quality, customization, optimization, and clean coding.

API Details

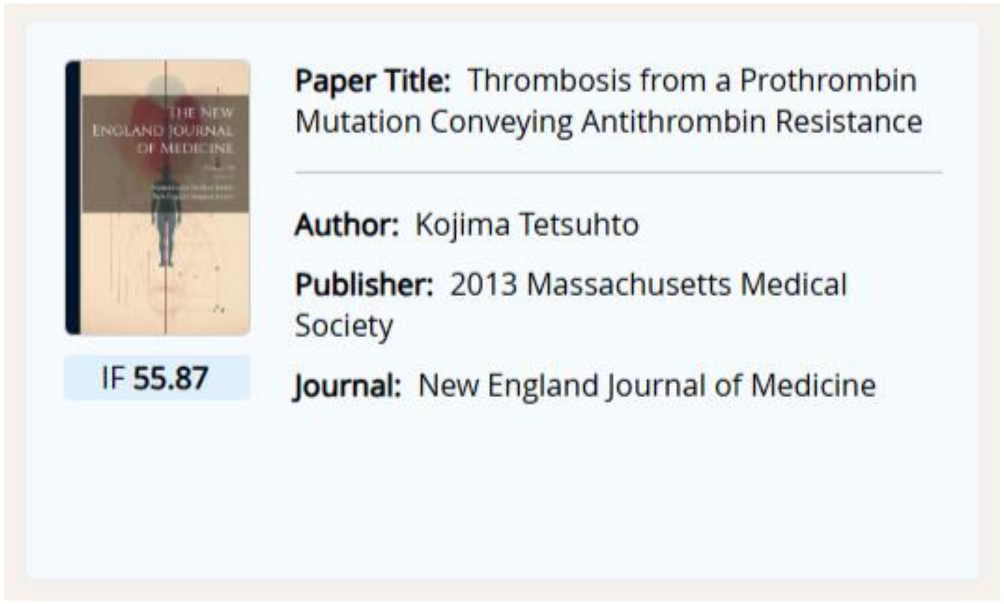
Endpoint:

https://easydash.enago.com/acceptedpapers

Strapi Filtering and other activities:

https://docs-v3.strapi.io/developer-docs/latest/developer-resources/content-api/content-api.html#endpoints

Card UI Design



Paper Title: Thrombosis from a Prothrombin Mutation Conveying Antithrombin Resistance

Author: Kojima Tetsuhto

Publisher: 2013 Massachusetts Medical Society

Journal: New England Journal of Medicine

IF **55.87**

Requirements

1. ***Data Fetching***
   - Fetch all data from the API asynchronously.
   - Handle loading, success, and error states clearly.
2. ***UI Display - Card Components***
   - Display each paper as a **Card UI** with the following details:

  - Title
  - Authors
  - Year
  - Journal Name
  - DOI
  - Impact Factor
  - PDF/Media Links (if any)
- Cards should match the structure of the provided screenshot or a clean professional layout.
- Allow room for UI customization (see #7).

3. ***Search Functionality***
- Add a search bar at the top with:
  - **Text input** for keywords.
  - A **dropdown** to select category to search (e.g., Title, Author, Journal).
- Show filtered results dynamically.

4. ***Sorting***
- Provide sorting controls for:
  - **Title (ASC/DESC)**
  - **Year (ASC/DESC)**
  - **Impact Factor (ASC/DESC)**

5. ***Pagination***
- Implement client-side or server-side pagination.
- Allow changing pages, with a clear indicator for current page and total results.

6. ***Details View***
- Add a button to each card: **"View Details"**.
- On click, show full details of the selected paper in either:
  - A new page (React route), **OR**
  - A popup/modal.
- This must display complete structured metadata.

7. ***Customization & Optimization***
- Design card components with the ability to rearrange elements (e.g., position of media vs text).
- Use **SCSS variables** for colors and theme settings.
- Structure code for **modularity, reusability, and readability**.

- Ensure responsiveness and performance optimization for large datasets.

*8. **Bonus (Optional)***

- Add skeleton loaders during data fetch.
- Implement debounce for search input.
- Add a download button for PDF links if available.

Evaluation Criteria

Criteria	Weight
API Integration & Data Handling	20%
UI/UX & Card Design	20%
Search, Sort, Pagination	20%
Code Quality & Structure	15%
Customization & Variables	10%
Detail View Implementation	10%
Optimization Techniques	5%

Tech Stack (Recommended)

- Next js 14.0.0^ (Preferred)
- SCSS modules for styling

Deliverables

- GitHub repo or zipped project folder
- README with instructions to run the project
- Brief explanation of decisions around architecture, customization, or optimization