|  |
| --- |
| "Daily.dev challenge": Viết code backend nhằm tạo trang tổng hợp thông tin(tương tư như trang daily.dev,...) từ nhiều nguồn khác nhau.  Mô tả công việc:  Tìm source (source kiếm từ RSS ví dụ: Báo mới, tuổi trẻ, dân trí,...) của các news, đọc source đó và lưu source xuống database và hiển thị lên trang tổng hợp.  Đăng ký source, mỗi khi mở trang tổng hợp thì phần mềm sẽ đi tìm từng source và insert từng source đó vào database. Sau đó từ database lấy những source của những news mà người dùng thích nhất lên trang tin chính. Hoặc hiển thị lên những news mà chủ đề người dùng lựa chọn.  Hiện tại chỉ dừng lại ở mức độ backend và project nhỏ nên chưa quan tâm đến bảo mật và phần front end giao diện. |
| **1. Hiểu rõ yêu cầu:**   * **Mục tiêu:** Tạo một nền tảng tổng hợp tin tức từ nhiều nguồn khác nhau, tương tự như daily.dev. * **Chức năng chính:**   + Lấy dữ liệu từ các nguồn RSS.   + Xử lý dữ liệu và lưu vào database.   + Hiển thị dữ dụng lên trang tổng hợp.   + Cho phép người dùng đăng ký theo dõi các nguồn tin yêu thích.   + Hiển thị tin tức theo chủ đề người dùng chọn. * **Công nghệ:** SQL Server, C#, các framework của C#.   **2. Phân tích các thành phần:**   * **Các module chính:**   + **Module lấy dữ liệu:**     - Lấy danh sách các nguồn RSS.     - Đọc nội dung từ từng nguồn RSS.     - Xử lý dữ liệu (parse, extract thông tin cần thiết).   + **Module lưu trữ dữ liệu:**     - Thiết kế cấu trúc database.     - Lưu trữ dữ liệu vào database.   + **Module logic nghiệp vụ:**     - Xử lý các yêu cầu của người dùng (đăng ký, tìm kiếm).     - Lọc dữ liệu theo yêu cầu của người dùng.   + **Module giao tiếp với frontend:**     - (Trong tương lai) Cung cấp API để frontend gọi.   3. **Design Database:**   * **Dim\_Source:** Chứa thông tin về các nguồn tin tức (RSS Feed) mà người dùng có thể thêm hoặc chọn lọc. * **Dim\_User:** Chứa thông tin về người dùng trên hệ thống. * **Fact\_News:** Chứa thông tin về các bài viết, đây là trung tâm cho các dữ liệu tin tức. * **Fact\_Article\_Interaction:** Quản lý tương tác của người dùng với bài viết (bookmark, lịch sử đọc, upvote, comment). * **Dim\_Tag:** Chứa các tags mà người dùng có thể theo dõi hoặc bài viết có liên quan. * **Dim\_Category:** Nếu muốn phân loại tin tức theo các danh mục lớn (ví dụ: Công nghệ, Đời sống, Giáo dục). * **Fact\_Bookmark:** Chứa thông tin về các bài viết mà người dùng đã lưu (bookmarked). * **Fact\_History:** Chứa thông tin về lịch sử đọc bài viết của người dùng. * **Dim\_Date:** Giúp phân tích các dữ liệu theo các khoảng thời gian khác nhau (ngày, tháng, năm, quý, tuần). * **News\_Tag:** Một bài viết có thể có nhiều thẻ, và một thẻ có thể gắn với nhiều bài viết. * **User\_Source:** Một người dùng có thể theo dõi nhiều nguồn tin, và một nguồn tin có thể được nhiều người dùng theo dõi. * **User\_ Tag:** Một người dùng có thể quan tâm đến nhiều thẻ, và một thẻ có thể được nhiều người quan tâm.   **4. Quy trình hoạt động:**   1. **Khởi động ứng dụng:** Đọc danh sách các nguồn RSS từ database. 2. **Lấy dữ liệu:**    * Lặp qua từng nguồn RSS, gọi hàm lấy dữ liệu.    * Xử lý dữ liệu và tạo các đối tượng Article. 3. **Lưu trữ dữ liệu:**    * Kiểm tra xem bài viết đã tồn tại trong database chưa.    * Nếu chưa, insert vào bài viết. 4. **Hiển thị dữ liệu:**    * Lấy dữ liệu từ database theo yêu cầu của người dùng (mới nhất, theo nguồn, theo chủ đề).    * Trả về dữ liệu cho frontend. |
| **1. My Feed (Trang Chính)**   * **Lấy danh sách các bài viết**:   + Truy vấn từ **Fact\_News** theo sở thích của người dùng (sử dụng thông tin từ **User\_Tag**, **User\_Source**).   + Tích hợp bộ lọc để hiển thị các bài viết mới nhất theo nguồn hoặc thẻ mà người dùng quan tâm.   + Hỗ trợ phân trang và tải thêm bài viết.   + **Thống kê**: Cập nhật và truy vấn số lượng like, comment, view cho từng bài viết từ các trường trong **Fact\_News**.   + **Gợi ý bài viết mới**:Đưa ra các gợi ý bài viết từ các nguồn hoặc thẻ mà người dùng chưa theo dõi.   **2. Custom Feed**   * **Lựa chọn và tùy chỉnh nguồn tin**:   + Người dùng có thể thêm hoặc xóa các nguồn tin (**Dim\_Sourse**) và các thẻ (**Dim\_Tag**) mà họ muốn theo dõi.   + Liên kết giữa người dùng và các nguồn tin được cập nhật trong **User\_Source** và giữa người dùng và các thẻ trong **User\_Tag**. * **Lưu và áp dụng cấu hình feed cá nhân**:   + Lưu cấu hình feed tùy chỉnh của người dùng để áp dụng mỗi khi họ mở trang.   + Truy vấn danh sách bài viết từ các nguồn và thẻ mà người dùng đã thiết lập.   **3. Explore (Khám Phá)**   * **Hiển thị các bài viết và nguồn tin nổi bật**:   + Lấy danh sách các bài viết phổ biến từ các nguồn khác nhau (**Fact\_News**), sử dụng các tiêu chí như số lượng view, like, và comment. * **Gợi ý nguồn tin**: Hiển thị các nguồn tin nổi bật từ **Dim\_Sourse** mà người dùng chưa theo dõi. * **Phân loại theo chủ đề**:Lấy danh sách bài viết theo các chủ đề (danh mục) từ **Dim\_Category**.   **4. Discussions (Thảo Luận)**   * **Quản lý bình luận**: Thêm, xóa, và lấy danh sách bình luận từ **Fact\_Article\_Interaction** với loại tương tác là "Comment". * **Tương tác với bình luận**: Người dùng có thể like hoặc reply bình luận, cập nhật số lượng like cho các bình luận.   **5. Tags**   * **Quản lý thẻ**:   + Lấy danh sách các thẻ có sẵn từ **Dim\_Tag**.   + Lọc bài viết theo thẻ từ **Fact\_News** dựa trên **News\_Tag**. * **Theo dõi và bỏ theo dõi thẻ**: Cho phép người dùng theo dõi hoặc bỏ theo dõi các thẻ, cập nhật **User\_Tag**.   **6. Sources**   * **Quản lý nguồn tin**:   + Hiển thị danh sách các nguồn tin đã đăng ký trong hệ thống từ **Dim\_Sourse**.   + Người dùng có thể theo dõi hoặc bỏ theo dõi các nguồn tin. * **Cập nhật và kiểm tra nguồn tin**: Tự động cập nhật bài viết từ nguồn tin thông qua RSS và lưu xuống cơ sở dữ liệu.   **7. Leaderboard**   * **xếp hạng bài viết và nguồn tin**:   + Truy vấn các bài viết được tương tác nhiều nhất (view, like, comment) từ **Fact\_News**.   + Hiển thị các nguồn tin có nhiều bài viết hoặc bài viết được tương tác nhiều nhất.   **8. Submit a Link**   * **Gửi bài viết mới**: Người dùng có thể gửi bài viết hoặc liên kết để xem xét và đăng tải. Các liên kết này sẽ được thêm vào **Fact\_News** sau khi được quản trị viên kiểm duyệt. * **Kiểm duyệt liên kết**: Quản trị viên có thể duyệt các bài viết do người dùng gửi, quyết định đăng tải hoặc từ chối.   **9. Bookmarks (Dấu trang)**   * **Quản lý bookmark**:   + Người dùng có thể lưu và quản lý các bài viết đã bookmark, lưu dữ liệu vào **Bookmark\_Fact**.   + Lấy danh sách các bài viết đã được bookmark của người dùng.   **10. History (Lịch sử đọc)**   * **Quản lý lịch sử đọc**:   + Lấy danh sách các bài viết mà người dùng đã đọc từ **History\_Fact**.   + Xóa hoặc quản lý lịch sử đọc của người dùng. |
| **1. Guest (Người dùng chưa đăng nhập)**   * **Xem danh sách bài viết**:   + **Mục tiêu**: Người dùng có thể xem danh sách các bài viết mới nhất hoặc phổ biến từ nhiều nguồn.   + **Mô tả**: Hiển thị các bài viết theo mặc định mà không yêu cầu người dùng đăng nhập.   + **Hành động**: Xem, Lọc bài viết theo thẻ hoặc nguồn. * **Tìm kiếm bài viết**:   + **Mục tiêu**: Người dùng có thể tìm kiếm bài viết theo từ khóa.   + **Mô tả**: Cho phép người dùng không cần đăng nhập để tìm kiếm.   + **Hành động**: Nhập từ khóa và tìm kiếm bài viết. * **Đăng ký tài khoản**:   + **Mục tiêu**: Cho phép người dùng tạo tài khoản để có thêm tính năng cá nhân hóa.   + **Hành động**: Đăng ký tài khoản qua email hoặc đăng nhập qua OAuth (Google, GitHub, etc.). * **Xem các nguồn tin phổ biến**:   + **Mục tiêu**: Khuyến khích người dùng khám phá các nguồn tin chất lượng.   + **Hành động**: Xem danh sách các nguồn tin và số lượng bài viết từ mỗi nguồn.   **2. Registered User (Người dùng đã đăng ký)**   * **Xem và quản lý Feed cá nhân**:   + **Mục tiêu**: Cung cấp một luồng tin tức cá nhân hóa dựa trên sở thích của người dùng.   + **Mô tả**: Người dùng đã đăng ký có thể theo dõi nguồn tin và thẻ yêu thích.   + **Hành động**:     - Xem feed được cá nhân hóa.     - Theo dõi và bỏ theo dõi nguồn tin.     - Theo dõi và bỏ theo dõi thẻ.     - Lọc bài viết theo thẻ hoặc nguồn đã theo dõi. * **Tương tác với bài viết**:   + **Mục tiêu**: Người dùng có thể tương tác với bài viết qua các hành động như like, comment, bookmark.   + **Hành động**: Like, Comment, Bookmark bài viết để lưu lại xem sau. * **Quản lý danh sách bookmark**:   + **Mục tiêu**: Người dùng có thể lưu lại các bài viết để xem sau.   + **Hành động**:     - Xem danh sách các bài viết đã bookmark.     - Thêm/xóa bài viết khỏi danh sách bookmark. * **Quản lý lịch sử đọc**:   + **Mục tiêu**: Người dùng có thể xem lại lịch sử các bài viết đã đọc.   + **Hành động**:     - Xem danh sách các bài viết đã đọc.     - Xóa lịch sử đọc nếu cần. * **Submit bài viết**:   + **Mục tiêu**: Người dùng có thể gửi bài viết hoặc liên kết để xem xét và đăng tải.   + **Hành động**: Gửi liên kết bài viết mới để hệ thống kiểm duyệt. * **Chỉnh sửa thông tin cá nhân và sở thích**:   + **Mục tiêu**: Cho phép người dùng quản lý thông tin tài khoản và sở thích về nguồn tin và thẻ.   + **Hành động**:     - Cập nhật thông tin tài khoản.     - Cập nhật sở thích (thêm/xóa thẻ và nguồn tin).   **3. Admin (Quản trị viên)**   * **Quản lý bài viết**:   + **Mục tiêu**: Quản trị viên có thể duyệt và quản lý các bài viết được đăng tải.   + **Hành động**:     - Duyệt bài viết do người dùng submit.     - Xóa bài viết nếu vi phạm.     - Chỉnh sửa nội dung bài viết nếu cần thiết. * **Quản lý người dùng**:   + **Mục tiêu**: Quản trị viên có thể quản lý thông tin người dùng.   + **Hành động**:     - Cập nhật hoặc khóa tài khoản người dùng vi phạm.     - Xem danh sách và thông tin người dùng. * **Quản lý nguồn tin**:   + **Mục tiêu**: Admin có thể thêm, chỉnh sửa, hoặc xóa các nguồn tin.   + **Hành động**:     - Thêm nguồn tin mới vào hệ thống.     - Cập nhật thông tin nguồn tin hiện có.     - Xóa nguồn tin khỏi hệ thống. * **Quản lý thẻ**:   + **Mục tiêu**: Admin có thể thêm, chỉnh sửa, hoặc xóa các thẻ trong hệ thống.   + **Hành động**: Thêm, Cập nhật, Xóa thẻ. * **Quản lý tương tác và báo cáo vi phạm**:   + **Mục tiêu**: Admin có thể theo dõi các báo cáo vi phạm về bài viết, bình luận từ người dùng.   + **Hành động**:     - Kiểm tra các báo cáo vi phạm.     - Xử lý các bài viết hoặc người dùng vi phạm. |
| **1. NewsController** (Quản lý các bài viết tin tức trong bảng Fact\_News)   * **GET /api/news:**   + Lấy danh sách tất cả các bài viết tin tức.   + Lý do: Cần một endpoint để hiển thị danh sách tin tức lên trang tổng hợp. * **GET /api/news/{id}:**   + Lấy chi tiết của một bài viết theo NewID.   + Lý do: Khi người dùng click vào một tin cụ thể, cần hiển thị chi tiết bài viết. * **POST /api/news:**   + Tạo mới một bài viết tin tức.   + Lý do: Cần cho phép các nguồn tin hoặc admin thêm bài viết mới vào hệ thống. * **PUT /api/news/{id}:**   + Cập nhật một bài viết tin tức theo NewID.   + Lý do: Để chỉnh sửa nội dung bài viết, admin hoặc nguồn tin có thể sử dụng endpoint này. * **DELETE /api/news/{id}:**   + Xóa một bài viết tin tức theo NewID.   + Lý do: Để quản lý tin tức không hợp lệ hoặc cần xóa.   **2. SourceController** (Liên quan đến bảng Dim\_Source, cung cấp chức năng quản lý các nguồn tin**)**   * **GET /api/source:**   + Lấy danh sách tất cả các nguồn tin.   + Lý do: Người dùng có thể chọn lọc bài viết theo nguồn. * **GET /api/source/{id}:**   + Lấy chi tiết một nguồn tin theo SourceID.   + Lý do: Để hiển thị thông tin chi tiết của nguồn tin, đặc biệt khi người dùng muốn theo dõi. * **POST /api/source:**   + Tạo một nguồn tin mới.   + Lý do: Admin hoặc hệ thống cần thêm nguồn tin mới từ các RSS feed khác nhau. * **PUT /api/source/{id}:**   + Cập nhật một nguồn tin theo SourceID.   + Lý do: Cần cập nhật URL của RSS feed hoặc trạng thái nguồn tin. * **DELETE /api/source/{id}:**   + Xóa một nguồn tin theo SourceID.   + Lý do: Nguồn tin không còn hợp lệ hoặc cần xóa khỏi hệ thống.   **3. UserController** (liên quan đến bảng Dim\_User, cung cấp chức năng quản lý người dùng)   * **GET /api/user:**   + Lấy danh sách tất cả người dùng.   + Lý do: Admin có thể xem và quản lý tất cả người dùng trong hệ thống. * **GET /api/user/{id}:**   + Lấy thông tin chi tiết của một người dùng theo UserID.   + Lý do: Để xem thông tin cá nhân của một người dùng cụ thể. * **POST /api/user:**   + Tạo mới người dùng.   + Lý do: Đăng ký người dùng mới vào hệ thống. * **PUT /api/user/{id}:**   + Cập nhật thông tin người dùng.   + Lý do: Cho phép người dùng hoặc admin cập nhật thông tin cá nhân. * **DELETE /api/user/{id}:**   + Xóa một người dùng khỏi hệ thống.   + Lý do: Khi người dùng vi phạm quy định hoặc không còn sử dụng hệ thống.   **4. TagController** (Quản các tags trong bảng Dim\_Tag.)   * **GET /api/tag:**   + Lấy danh sách tất cả các thẻ.   + Lý do: Cho phép người dùng hoặc hệ thống lọc tin tức theo thẻ. * **GET /api/tag/{id}:**   + Lấy thông tin chi tiết của một thẻ.   + Lý do: Để hiển thị chi tiết của một thẻ cụ thể. * **POST /api/tag:**   + Tạo mới một thẻ.   + Lý do: Để thêm thẻ vào hệ thống, đặc biệt khi tạo các chủ đề mới. * **PUT /api/tag/{id}:**   + Cập nhật một thẻ cụ thể.   + Lý do: Admin có thể sửa đổi thông tin thẻ nếu cần thiết**.** * **DELETE /api/tag/{id}:**   + Xóa một thẻ cụ thể.   + Lý do: Khi thẻ không còn cần thiết hoặc không sử dụng.   **5. InteractionController** (Quản lý các tương tác từ người dùng (upvote, bookmark, comment) trong bảng Fact\_Article\_Interaction)   * **GET /api/interaction**: Lấy danh sách tất cả các tương tác từ người dùng. * **Lý do**: Cần có một endpoint để hiển thị và quản lý tất cả các tương tác từ người dùng như upvote, bookmark, hoặc comment. * **GET /api/interaction/{id}**: Lấy thông tin chi tiết về một tương tác cụ thể dựa trên InteractionID. * **Lý do**: Cần hiển thị thông tin về một tương tác của người dùng với bài viết, chẳng hạn như nội dung comment hoặc số lượng upvote. * **POST /api/interaction**: Tạo mới một tương tác (bookmark, comment, upvote) giữa người dùng và bài viết. * **Lý do**: Hệ thống cần thêm các tương tác của người dùng khi họ thực hiện các hành động như bình luận, đánh dấu bài viết, hoặc upvote bài viết. * **PUT /api/interaction/{id}**: Cập nhật một tương tác dựa trên InteractionID. Ví dụ như cập nhật nội dung comment hoặc thay đổi số lượng upvote. * **Lý do**: Admin hoặc người dùng có thể muốn sửa đổi các tương tác trước đó của mình (như chỉnh sửa bình luận hoặc thay đổi đánh dấu bài viết). * **DELETE /api/interaction/{id}**: Xóa một tương tác cụ thể dựa trên InteractionID. * **Lý do**: Cần có khả năng xóa các tương tác không còn hợp lệ hoặc không cần thiết.   **6. BookmarkController** (Quản lý việc bookmark bài viết trong bảng Fact\_Bookmark.)   * **POST /api/bookmark:**   + Thêm một bookmark mới.   + Lý do: Người dùng có thể lưu các bài viết mà họ quan tâm. * **GET /api/bookmark/user/{userId}:**   + Lấy danh sách các bài viết đã được bookmark bởi một người dùng.   + Lý do: Hiển thị tất cả các bài viết mà người dùng đã lưu. * **DELETE /api/bookmark/{id}:**   + Xóa một bookmark.   + Lý do: Người dùng có thể xóa bỏ các bài viết mà họ không còn muốn lưu.   **7. HistoryController** (Quản lý lịch sử đọc bài viết trong bảng Fact\_History)   * **POST /api/history:**   + Thêm lịch sử đọc cho một bài viết.   + Lý do: Ghi nhận thời điểm người dùng đọc bài viết. * **GET /api/history/user/{userId}:**   + Lấy danh sách các bài viết đã đọc của một người dùng.   + Lý do: Hiển thị cho người dùng lịch sử đọc của họ.   **8. CategoryController** (Quản lý danh mục bài viết trong bảng Dim\_Category)   * **GET /api/category:**   + Lấy danh sách tất cả các danh mục.   + Lý do: Người dùng có thể lọc tin tức theo danh mục. * **POST /api/category:**   + Tạo một danh mục mới.   + Lý do: Thêm danh mục mới khi hệ thống có các chủ đề khác nhau. * **PUT /api/category/{id}:**   + Cập nhật một danh mục cụ thể.   + Lý do: Admin có thể chỉnh sửa thông tin danh mục nếu cần. * **DELETE /api/category/{id}:**   + Xóa một danh mục.   + Lý do: Loại bỏ các danh mục không còn sử dụng. |