**TRƯỜNG CÔNG NGHỆ THÔNG TIN PHENIKAA**

**KHOA HỆ THỐNG THÔNG TIN**

**⸎⸎⸎⸎⸎**

A logo of a university

AI-generated content may be incorrect.

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN LẬP TRÌNH CHO THIẾT BỊ DI ĐỘNG**

**Đề tài: Xây dựng ứng dụng đọc sách "BookReader"**

**trên nền tảng Flutter**

Thông tin sinh viên : Lê Ngọc Khánh - 23010546

Giảng viên hướng dẫn : Th.S Nguyễn Xuân Quế

Lớp tín chỉ : [Lập trình cho thiết bị di động-1-1-25(N04)](https://canvas.phenikaa-uni.edu.vn/courses/19672)

**HÀ NỘI, THÁNG 11/2025**

Mục Lục

[Lời mở đầu 3](#_Toc213875760)

[Chương 1: Giới thiệu đề tài và cơ sở lý thuyết 4](#_Toc213875761)

[**1.1.** **Giới thiệu đề tài** 4](#_Toc213875762)

[**1.2.** **Cơ sở lý thuyết** 5](#_Toc213875763)

[Chương 2: Đặc tả yêu cầu phần mềm 7](#_Toc213875767)

[**2.1. Yêu cầu chức năng** 7](#_Toc213875768)

[**2.2. Yêu cầu phi chức năng** 10](#_Toc213875774)

[Chương 3: Thiết kế kiến trúc phần mềm 11](#_Toc213875779)

[**3.1. Thiết kế kiến trúc tổng thể** 11](#_Toc213875780)

[**3.2. Thiết kế dữ liệu** 13](#_Toc213875783)

[**3.3. Thiết kế lớp dịch vụ** 15](#_Toc213875791)

[**3.4. Thiết kế chi tiết các module chính** 16](#_Toc213875795)

[Chương 4: Thiết kế giao diện 23](#_Toc213875799)

[**4.1. Sơ đồ cấu trúc Widget** 23](#_Toc213875800)

[**4.2. Giao diện các màn hình** 24](#_Toc213875801)

[Chương 5: Tổng kết và hướng phát triển 32](#_Toc213875809)

[**5.1. Kết quả đạt được** 32](#_Toc213875810)

[**5.2. Hạn chế hiện tại** 32](#_Toc213875811)

[**5.3. Đề xuất cải tiến** 33](#_Toc213875815)

# 

# **Lời mở đầu**

Trong bối cảnh công nghệ thông tin và thiết bị di động phát triển vượt bậc, nhu cầu học tập, nghiên cứu và giải trí của con người cũng đã có những thay đổi rõ rệt. Việc đọc sách, một hình thức tiếp thu tri thức truyền thống, đang dần được số hóa mạnh mẽ. Các ứng dụng đọc sách trên điện thoại thông minh và máy tính bảng ngày càng trở nên phổ biến nhờ tính tiện lợi, cho phép người dùng mang theo cả một thư viện số cá nhân bên mình.

Báo cáo này trình bày chi tiết quá trình thực hiện đề tài Bài tập lớn môn học: "Phát triển ứng dụng đọc sách di động BookReader". Nội dung báo cáo tập trung vào việc áp dụng các kiến thức về lập trình di động đa nền tảng với framework Flutter để xây dựng một ứng dụng hoàn chỉnh, đáp ứng các nhu cầu cơ bản của người dùng.

Em xin chân thành cảm ơn thầy Nguyễn Xuân Quế đã tận tình hướng dẫn và cung cấp các kiến thức nền tảng quý báu để em có thể hoàn thành đề tài này.

# **Chương 1: Giới thiệu đề tài và cơ sở lý thuyết**

## **Giới thiệu đề tài**

Ngày nay, điện thoại thông minh đã trở thành vật dụng không thể thiếu trong cuộc sống hàng ngày. Cùng với sự bùng nổ của nội dung số, thói quen đọc sách của người dùng đã có sự dịch chuyển lớn. Thay vì mang theo những cuốn sách giấy cồng kềnh, người dùng có xu hướng tìm đến các ứng dụng đọc sách di động.

Sự tiện lợi của việc có thể truy cập hàng ngàn đầu sách mọi lúc, mọi nơi, trên một thiết bị duy nhất đã thúc đẩy nhu cầu về các ứng dụng thư viện số cá nhân. Một ứng dụng đọc sách tốt không chỉ cung cấp nội dung mà còn phải mang lại trải nghiệm đọc thoải mái, khả năng quản lý sách khoa học và các tính năng cá nhân hóa, giúp người dùng dễ dàng theo dõi hành trình tri thức của mình.

Mục tiêu của dự án là xây dựng một ứng dụng đọc sách di động đa nền tảng với tên gọi "BookReader". Ứng dụng tập trung vào việc cung cấp một trải nghiệm đọc mượt mà và các công cụ quản lý thư viện cá nhân hiệu quả.

Các chức năng chính bao gồm:

* Xây dựng giao diện người dùng thân thiện, hiện đại và dễ sử dụng.
* Phát triển các tính năng cốt lõi: tìm kiếm sách, duyệt sách theo danh mục.
* Tích hợp trình đọc sách (PDF) với các tùy chỉnh cơ bản.
* Phát triển hệ thống quản lý thư viện cá nhân, cho phép người dùng "yêu thích" và xem lại các sách đã đánh dấu.
* Xây dựng tính năng "thử thách" để theo dõi tiến độ và tạo động lực cho người dùng.
* Người dùng có thể chuyển đổi giao diện giữa tiếng việt và tiếng anh.

Để đảm bảo tính khả thi trong khuôn khổ bài tập lớn, đề tài tập trung vào các phạm vi sau:

* Nền tảng: Ứng dụng được phát triển bằng Flutter, có khả năng biên dịch và chạy trên hệ điều hành di động Android.
* Người dùng: Ứng dụng tập trung hoàn toàn vào trải nghiệm của người dùng cá nhân, không bao gồm các chức năng quản trị.
* Dữ liệu: Do không xây dựng hệ thống backend, toàn bộ dữ liệu được quản lý cục bộ trên thiết bị của người dùng hoặc sử dụng dữ liệu giả được định nghĩa sẵn trong code.

## **Cơ sở lý thuyết**

### **1.2.1. Giới thiệu về Flutter và ngôn ngữ Dart**

Flutter là một bộ công cụ giao diện người dùng mã nguồn mở do Google phát triển. Nó cho phép các nhà phát triển xây dựng các ứng dụng đẹp mắt, có hiệu năng cao cho di động, web và máy tính để bàn từ một cơ sở mã duy nhất. Các ưu điểm chính của Flutter được áp dụng trong đề tài bao gồm:

* Đa nền tảng: Chỉ cần viết code một lần và triển khai trên nhiều nền tảng, giúp tiết kiệm thời gian và chi phí phát triển.
* Hiệu năng cao: Flutter biên dịch trực tiếp ra mã máy, mang lại hiệu năng gần như tương đương với ứng dụng gốc.
* Hot Reload: Tính năng "Hot Reload" cho phép lập trình viên thấy ngay lập tức các thay đổi về code trên giao diện mà không cần khởi động lại ứng dụng, giúp tăng tốc độ phát triển và sửa lỗi.

Dart là ngôn ngữ lập trình được Google tối ưu hóa để xây dựng giao diện người dùng. Đây là ngôn ngữ lập trình hướng đối tượng, có cú pháp rõ ràng, quen thuộc (tương tự C/Java/C#) và hỗ trợ biên dịch AOT lẫn JIT. Dart cung cấp nền tảng vững chắc cho Flutter, đặc biệt là trong việc xử lý bất đồng bộ và quản lý bộ nhớ hiệu quả.

### **1.2.2. Kiến trúc Widget trong Flutter**

**"Everything is a Widget":** Triết lý cốt lõi của Flutter là "Tất cả mọi thứ đều là Widget". Một widget là một mô tả bất biến về một phần của giao diện người dùng. Không chỉ các thành phần hữu hình như button hay text là widget, mà ngay cả các thành phần bố cục (Row, Column), căn chỉnh (Center), hay khoảng đệm (Padding) cũng đều là widget. Các widget này được lồng vào nhau để tạo thành "Cây Widget" (Widget Tree), mô tả toàn bộ giao diện của ứng dụng.

Trong dự án, hai loại widget cơ bản đã được sử dụng:

StatelessWidget: Là các widget tĩnh, có cấu hình không thay đổi sau khi được tạo. Chúng chỉ phụ thuộc vào thông tin được truyền vào khi khởi tạo.

StatefulWidget (Widget có trạng thái): Là các widget động, có thể thay đổi giao diện dựa trên dữ liệu bên trong nó hoặc dựa trên tương tác của người dùng.

### **1.2.3. Quản lý trạng thái**

Khái niệm "State": Trong lập trình ứng dụng, "State" được hiểu là bất kỳ dữ liệu nào có thể thay đổi theo thời gian và có ảnh hưởng đến giao diện người dùng (ví dụ: một giá trị trong bộ đếm, một danh sách dữ liệu, trạng thái đăng nhập...).

Quản lý trạng thái cục bộ: Đây là phương pháp quản lý trạng thái cơ bản nhất, thường được sử dụng cho các trạng thái chỉ liên quan đến một widget đơn lẻ. Trong StatefulWidget, Flutter cung cấp hàm setState(). Khi dữ liệu trạng thái thay đổi, việc gọi setState() sẽ thông báo cho framework biết rằng widget đó cần được "vẽ" lại (rebuild) để cập nhật giao diện theo dữ liệu mới.

# **Chương 2: Đặc tả yêu cầu phần mềm**

Chương này trình bày chi tiết các yêu cầu của ứng dụng "BookReader". Nội dung chương tập trung vào việc mô tả các chức năng mà hệ thống cần phải thực hiện (yêu cầu chức năng) và các tiêu chuẩn mà hệ thống phải tuân thủ (yêu cầu phi chức năng). Đây là tài liệu làm cơ sở cho việc thiết kế, phát triển và kiểm thử phần mềm.

## **2.1. Yêu cầu chức năng**

Yêu cầu chức năng mô tả các hành vi cụ thể của hệ thống, định nghĩa những gì hệ thống sẽ làm. Các yêu cầu được nhóm lại thành các module (Use Case - UC) để dễ quản lý và theo dõi.

A diagram of a person with text

AI-generated content may be incorrect.

*Hình 1. Sơ đồ Use Case*

### **2.1.1. Nhóm chức năng xem và tìm kiếm (UC1)**

Nhóm chức năng này cho phép người dùng tìm kiếm, duyệt và xem thông tin về các đầu sách có trong ứng dụng.

**UC1.1 - Tìm kiếm sách:** Cung cấp một ô nhập liệu cho phép người dùng tìm kiếm sách dựa trên từ khóa. Hệ thống sẽ lọc và hiển thị danh sách các sách có tên sách hoặc tên tác giả chứa từ khóa, với kết quả được cập nhật tự động ngay khi người dùng thay đổi nội dung tìm kiếm.

**UC1.2 - Lọc sách theo danh mục:** Cho phép người dùng lọc danh sách sách theo một danh mục cụ thể. Hệ thống hiển thị một danh sách các danh mục có sẵn; khi người dùng chọn một danh mục, chỉ các sách thuộc danh mục đó được hiển thị. Người dùng có thể chọn "Tất cả" để xem lại toàn bộ sách.

**UC1.3 - Xem danh sách sách:** Hiển thị danh sách sách ở nhiều khu vực. Tại trang chủ, sách được trình bày dưới dạng các danh sách cuộn ngang. Tại trang Khám phá, tất cả sách được hiển thị dưới dạng lưới có thể cuộn dọc.

**UC1.4 - Xem chi tiết sách:** Khi người dùng chọn một cuốn sách từ bất kỳ danh sách nào, hệ thống sẽ điều hướng đến màn hình chi tiết, hiển thị đầy đủ thông tin: ảnh bìa, tên sách, tác giả, mô tả, số trang, và đánh giá.

### **2.1.2. Nhóm chức năng đọc sách (UC2)**

Nhóm chức năng này tập trung vào trải nghiệm đọc sách cốt lõi của người dùng.

**UC2.1 - Mở trình đọc sách:**

Tại màn hình chi tiết sách, khi người dùng nhấn "Bắt đầu đọc", hệ thống sẽ khởi chạy trình đọc file PDF (hoặc định dạng khác) để hiển thị nội dung cuốn sách.

**UC2.2 - Điều hướng trang:**

Trong trình đọc, người dùng có thể ấn vào dấu mũi tên để chuyển đến trang tiếp theo hoặc trang trước đó, chọn số trang để chuyển tới.

**UC2.3 – Đánh dấu trang sách:**

Hệ thống tự động ghi nhớ trang sách mà người dùng đang đọc dở. Mỗi khi người dùng chuyển trang, số trang hiện tại sẽ được lưu lại. Khi người dùng mở lại một cuốn sách đã đọc, ứng dụng sẽ tự động điều hướng đến trang đã được lưu trước đó.

### **2.1.3. Nhóm chức năng thư viện cá nhân (UC3)**

Nhóm chức năng này cho phép người dùng cá nhân hóa bộ sưu tập sách của riêng mình.

**UC3.1 - Quản lý sách yêu thích:**

Người dùng có thể thêm một cuốn sách vào danh sách "Yêu thích" từ màn hình chi tiết, hoặc xóa sách khỏi danh sách này. Hệ thống cung cấp một tab "Yêu thích" trong Thư viện để hiển thị tất cả các sách đã được thêm.

**UC3.2 - Quản lý sách đang đọc dở:**

Chức năng này liên kết với UC2.3, cung cấp một tab "Đã đánh dấu" trong Thư viện. Nó hiển thị danh sách các sách đang đọc dở cùng với số trang hiện tại, cho phép người dùng tiếp tục đọc một cách nhanh chóng.

### **2.1.4. Nhóm chức năng thử thách và thành tích (UC4)**

Nhóm chức năng này nhằm tạo động lực và trò chơi hóa trải nghiệm đọc sách.

**UC4.1 - Theo dõi mục tiêu đọc:**

Màn hình Thử thách hiển thị một thanh tiến trình trực quan, cho phép người dùng theo dõi số lượng sách đã đọc so với mục tiêu tháng đã đề ra.

**UC4.2 - Xem thử thách:**

Hiển thị danh sách các thử thách đọc sách đang diễn ra (ví dụ: "Đọc 5 sách trong tháng", "Chuỗi 7 ngày đọc liên tục") cùng với tiến độ hoàn thành của từng thử thách.

**UC4.3 - Xem thành tích:**

Cung cấp một bộ sưu tập các huy hiệu mà người dùng có thể đạt được. Hệ thống sẽ đánh dấu những huy hiệu đã được mở khóa khi người dùng hoàn thành các cột mốc nhất định.

### **2.1.5. Nhóm chức năng cài đặt và hồ sơ (UC5)**

Nhóm chức năng này cho phép người dùng tùy chỉnh ứng dụng và xem các thông tin liên quan.

**UC5.1 - Xem thông tin cá nhân:**

Màn hình Hồ sơ hiển thị thông tin của sinh viên thực hiện dự án, bao gồm tên, lớp, và giảng viên hướng dẫn.

**UC5.2 - Thay đổi ngôn ngữ:**

Hệ thống hỗ trợ đa ngôn ngữ, cho phép người dùng chuyển đổi giao diện hiển thị giữa hai ngôn ngữ: Tiếng Việt và Tiếng Anh.

## **2.2. Yêu cầu phi chức năng**

### **2.2.1. Yêu cầu về hiệu năng**

* Hệ thống phải khởi động trong vòng 5 giây.
* Tất cả các thao tác cuộn (danh sách, chuyển trang PDF) phải đảm bảo mượt mà, đạt tốc độ 60 khung hình/giây (fps).
* Thời gian phản hồi cho các chức năng lọc và tìm kiếm (trên dữ liệu cục bộ) phải dưới 1 giây.

### **2.2.2. Yêu cầu về tính dễ sử dụng**

* Giao diện người dùng phải được thiết kế trực quan, sạch sẽ và nhất quán.
* Các chức năng chính phải dễ dàng truy cập thông qua thanh điều hướng dưới cùng.
* Các biểu tượng và nhãn phải rõ ràng và thống nhất trong toàn bộ ứng dụng.

### **2.2.3. Yêu cầu về tính tương thích**

* Ứng dụng phải hoạt động ổn định trên các thiết bị di động sử dụng hệ điều hành Android (từ phiên bản 5.0 trở lên).
* Giao diện phải có tính co giãn để hiển thị tốt trên các kích thước màn hình khác nhau, từ điện thoại tiêu chuẩn đến máy tính bảng.

### **2.2.4. Yêu cầu về tính bảo mật**

* Toàn bộ dữ liệu cá nhân của người dùng (danh sách yêu thích, tiến độ đọc) phải được lưu trữ an toàn và cục bộ trên thiết bị của người dùng, không được gửi đi nơi khác.

# **Chương 3: Thiết kế kiến trúc phần mềm**

Chương này trình bày các quyết định thiết kế kỹ thuật và kiến trúc cốt lõi của ứng dụng "BookReader". Nội dung chương tập trung vào việc mô tả cấu trúc các thành phần phần mềm, các mô hình được áp dụng, mối quan hệ giữa chúng, và các nguyên tắc thiết kế đã được tuân thủ.

## **3.1. Thiết kế kiến trúc tổng thể**

### **3.1.1. Lựa chọn mô hình kiến trúc**

Để đảm bảo tính dễ bảo trì và khả năng mở rộng, ứng dụng "BookReader" được thiết kế dựa trên mô hình kiến trúc phân lớp. Mô hình này đặc biệt hiệu quả với Flutter, cho phép phân tách rõ ràng logic xử lý giao diện khỏi logic nghiệp vụ và logic truy cập dữ liệu. Kiến trúc của ứng dụng được chia thành 3 lớp logic chính, mỗi lớp có một vai trò và trách nhiệm riêng biệt.

### **3.1.2. Cấu trúc các lớp**

**Lớp trình diễn**

* Trách nhiệm: Là lớp trên cùng, chịu trách nhiệm hiển thị giao diện người dùng và xử lý các tương tác trực tiếp từ người dùng (ví dụ: nhấn nút, cuộn). Lớp này không chứa logic nghiệp vụ phức tạp mà chỉ chuyển tiếp các sự kiện của người dùng đến Lớp Dịch vụ.
* Thành phần chính: Các Widgets, Pages (màn hình), và các thành phần giao diện tái sử dụng (ví dụ: BookCard).
* Vị trí dự án: lib/ui/

features: Các trang chức năng (home, discover, library, challenge, profile, book, category)

shared: Các widget dùng chung (BookCard)

**Lớp dịch vụ**

Trách nhiệm: Là lớp trung gian, chứa toàn bộ logic nghiệp vụ cốt lõi và quản lý trạng thái toàn cục của ứng dụng (ví dụ: danh sách yêu thích, tiến độ đọc). Lớp này thực thi các quy tắc nghiệp vụ và xử lý dữ liệu, độc lập hoàn toàn với giao diện.

* Thành phần chính: Các lớp quản lý (Managers) như FavoritesManager, BookmarksManager và LanguageService
* Vị trí dự án: lib/services/

**Lớp dữ liệu**

* Trách nhiệm: Là lớp dưới cùng, chịu trách nhiệm định nghĩa cấu trúc dữ liệu (Models) và cung cấp dữ liệu cho các lớp bên trên. Trong phiên bản hiện tại, lớp này bao gồm các đối tượng mô hình và nguồn dữ liệu mẫu (sample data).
* Thành phần chính: Các lớp Model (BookModel, CategoryModel,...) và các file dữ liệu (sample\_books.dart).
* Vị trí dự án: lib/data/

Mối quan hệ phụ thuộc giữa các lớp được thiết kế nghiêm ngặt theo một chiều:

* Lớp UI có thể phụ thuộc vào Lớp Service.
* Lớp Service có thể phụ thuộc vào Lớp Data.
* Quy tắc: Một lớp ở tầng thấp hơn không bao giờ được phép biết đến hoặc phụ thuộc vào một lớp ở tầng cao hơn.

A diagram of a user interface

AI-generated content may be incorrect.

*Hình 2. Sơ đồ ba lớp*

## **3.2. Thiết kế dữ liệu**

Lớp dữ liệu định nghĩa cấu trúc cho tất cả các thực thể trong ứng dụng thông qua các lớp Model. Dưới đây là thiết kế chi tiết của các model chính:

### **3.2.1. BookModel**

Đại diện cho đối tượng cốt lõi của ứng dụng là một cuốn sách.

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | String | Mã định danh duy nhất (Primary Key). |
| title | String | Tên sách. |
| author | String | Tên tác giả. |
| coverImageUrl | String | Đường dẫn (local) đến ảnh bìa. |
| pdfPath | String | Đường dẫn (local) đến file PDF của sách. |
| description | String | Mô tả chi tiết về nội dung sách. |
| rating | double | Điểm đánh giá của sách. |
| pageCount | int | Tổng số trang. |
| categoryId | String | Mã tham chiếu đến CategoryModel. |

### *Bảng 1. BookModel*

### **3.2.2. CategoryModel**

Đại diện cho một danh mục (thể loại) sách.

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | String | Mã định danh duy nhất của danh mục. |
| name | String | Tên danh mục. |

### *Bảng 2. CategoryModel*

### **3.2.3. BookmarkModel**

Đại diện cho một đánh dấu trang (tiến độ đọc).

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| bookId | String | Mã sách được đánh dấu. |
| pageNumber | int | Số trang cuối cùng người dùng đã đọc. |

### *Bảng 3. BookmarkModel*

### **3.2.4. Các model khác**

**ChallengeModel**: Đại diện cho một thử thách đọc sách (ví dụ: "Đọc 5 cuốn sách trong tháng")

* id (String): Mã thử thách
* title (String): Tên thử thách
* description (String): Mô tả chi tiết
* current (int): Tiến độ hiện tại (số sách đã đọc)
* target (int): Mục tiêu cần đạt (tổng số sách)
* icon (IconData): Icon đại diện

**BadgeModel**: Đại diện cho một huy hiệu thành tích

* id (String): Mã huy hiệu
* name (String): Tên huy hiệu
* description (String): Mô tả cách đạt được huy hiệu
* unlocked (bool): Trạng thái đã mở khóa hay chưa
* icon (IconData): Icon đại diện

## **3.3. Thiết kế lớp dịch vụ**

Lớp Dịch vụ được thiết kế để quản lý trạng thái và logic nghiệp vụ toàn cục. Để đảm bảo chỉ có một nguồn chân lý cho các trạng thái này trên toàn ứng dụng, các lớp quản lý được triển khai theo mẫu thiết kế Singleton. Điều này đảm bảo rằng mọi màn hình đều truy cập và thay đổi cùng một instance duy nhất, giúp đồng bộ hóa dữ liệu một cách hiệu quả.

### **3.3.1. FavoritesManager**

* Trách nhiệm: Quản lý danh sách các sách được người dùng yêu thích.
* Thiết kế kỹ thuật:
  + Sử dụng một Set<String> tên là \_favoriteIds. Việc dùng Set thay vì List đảm bảo hiệu suất kiểm tra nhanh và tự động xử lý việc một bookId không bị trùng lặp.
* Phương thức chính:
  + toggleFavorite(String bookId): Thêm bookId vào Set nếu chưa có, hoặc xóa khỏi Set nếu đã tồn tại.
  + isFavorite(String bookId): Trả về true nếu \_favoriteIds chứa bookId.
  + getFavorites(): Trả về danh sách các bookId yêu thích.

### **3.3.2. BookmarksManager**

* Trách nhiệm: Quản lý tiến độ đọc (bookmark) của người dùng cho từng cuốn sách.
* Thiết kế kỹ thuật:
  + Sử dụng một Map<String, int> tên là \_bookmarks. Cấu trúc Map lý tưởng cho việc này, với key là bookId (String) và value là pageNumber (int).
* Phương thức chính:
  + saveBookmark(String bookId, int pageNumber): Lưu hoặc cập nhật số trang cho một bookId.
  + getBookmark(String bookId): Lấy số trang đã lưu của một bookId. Trả về 0 hoặc giá trị mặc định nếu sách chưa có bookmark.
  + hasBookmark(String bookId): Kiểm tra xem \_bookmarks có chứa key là bookId hay không.

### **3.3.3. LanguageService**

* **Trách nhiệm**: Quản lý ngôn ngữ hiển thị của toàn bộ ứng dụng, cung cấp hệ thống đa ngôn ngữ (Tiếng Việt/English).
* **Thiết kế kỹ thuật**:
  + Sử dụng một biến static bool \_isEnglish để theo dõi ngôn ngữ hiện tại
  + Lưu trữ các chuỗi dịch trong hai Map<String, String>: \_vi (Tiếng Việt) và \_en (English)
  + Mỗi Map chứa các cặp key-value, trong đó key là mã định danh duy nhất (ví dụ: "home\_title") và value là văn bản hiển thị
* **Phương thức chính**:
  + static String get(String key): Lấy văn bản đã dịch theo ngôn ngữ hiện tại. Nếu \_isEnglish = true, trả về \_en[key], ngược lại trả về \_vi[key]
  + static void toggleLanguage(): Chuyển đổi ngôn ngữ bằng cách đảo giá trị \_isEnglish
  + static bool get isEnglish: Getter để kiểm tra ngôn ngữ hiện tại

A diagram of a computer program

AI-generated content may be incorrect.

*Hình 3. Sơ đồ lớp*

## **3.4. Thiết kế chi tiết các module chính**

Để làm rõ cách các lớp tương tác với nhau, phần này sẽ mô tả chi tiết luồng xử lý của hai chức năng quan trọng: "Thêm sách vào danh sách yêu thích" và "Chuyển đổi ngôn ngữ".

### **3.4.1. Thêm một sách vào danh sách yêu thích**

Đây là luồng xử lý phổ biến nhất khi người dùng khám phá một cuốn sách mới và quyết định lưu lại.

Bối cảnh: Người dùng đang ở màn hình BookDetailPage.

Các bước xử lý:

1. Tương tác người dùng: Người dùng nhấn vào biểu tượng hình trái tim (❤️) trên màn hình BookDetailPage.
2. Xử lý sự kiện UI:
   * Màn hình BookDetailPage là một StatefulWidget. Biểu tượng trái tim là một IconButton có sự kiện onPressed.
   * Khi được nhấn, sự kiện onPressed được kích hoạt.
3. Gọi lớp Dịch vụ:
   * Bên trong onPressed, code sẽ gọi đến thực thể Singleton của FavoritesManager thông qua lệnh: \_favoritesManager.toggleFavorite(widget.book.id).
4. Xử lý logic nghiệp vụ:
   * Lớp FavoritesManager nhận được book.id, sau đó kiểm tra xem mã sách này đã tồn tại trong Set<String> \_favoriteIds hay chưa.
     + Nếu đã tồn tại, nó sẽ xóa mã sách khỏi Set (hủy yêu thích).
     + Nếu chưa tồn tại, nó sẽ thêm mã sách vào Set (đánh dấu yêu thích).
   * Phương thức này trả về một giá trị boolean cho biết sách vừa được thêm (true) hay xóa (false).
5. Cập nhật giao diện:
   * Ngay sau khi gọi FavoritesManager, hàm onPressed sẽ gọi setState(() {}).
   * Lệnh này thông báo cho Flutter rằng trạng thái của BookDetailPage đã thay đổi và cần được vẽ lại.
6. Vẽ lại Widget:
   * Flutter thực hiện quá trình "rebuild" cho BookDetailPage.
   * Trong quá trình này, thuộc tính icon và color của IconButton được tính toán lại. Chúng dựa trên kết quả của getter isFavorite (getter này cũng gọi đến FavoritesManager để lấy trạng thái mới nhất).
   * Kết quả là biểu tượng trái tim thay đổi giữa:
     + Trạng thái yêu thích: Icons.favorite (đầy) với màu đỏ (Colors.red)
     + Trạng thái chưa yêu thích: Icons.favorite\_border (rỗng) với màu xám (Colors.grey)
7. Phản hồi cho người dùng:
   * Cuối cùng, một SnackBar được hiển thị để thông báo cho người dùng biết hành động (thêm/xóa) đã thành công.
   * Thông báo hiển thị theo ngôn ngữ hiện tại thông qua AppLanguage.get().

### A screenshot of a computer AI-generated content may be incorrect.

*Hình 4. Sơ đồ tuần tự - thêm vào danh sách yêu thích*

### **3.4.2. Chuyển đổi ngôn ngữ**

Đây là chức năng quan trọng cho phép người dùng tùy chỉnh ngôn ngữ giao diện theo sở thích, với thay đổi áp dụng ngay lập tức cho toàn bộ ứng dụng.

Bối cảnh: Người dùng đang ở màn hình ProfilePage và muốn chuyển đổi ngôn ngữ.

Các bước xử lý:

1. Tương tác người dùng: Người dùng nhấn vào mục "Language / Ngôn ngữ" trong danh sách cài đặt trên màn hình ProfilePage.
2. Xử lý sự kiện UI:
   * Màn hình ProfilePage là một StatefulWidget. Mục ngôn ngữ là một ListTile có sự kiện onTap.
   * Khi được nhấn, sự kiện onTap được kích hoạt.
3. Gọi lớp Dịch vụ:
   * Bên trong onTap, code sẽ gọi AppLanguage.toggleLanguage() bên trong hàm setState(() {}).
   * Điều này đảm bảo việc thay đổi ngôn ngữ và cập nhật giao diện xảy ra đồng bộ.
4. Xử lý logic nghiệp vụ:
   * Lớp LanguageService (class AppLanguage) nhận được lệnh toggleLanguage(), sau đó đảo ngược giá trị của biến static isEnglish.
     + Nếu isEnglish = false (đang dùng Tiếng Việt), nó sẽ chuyển thành isEnglish = true (chuyển sang English).
     + Nếu isEnglish = true (đang dùng English), nó sẽ chuyển thành isEnglish = false (chuyển sang Tiếng Việt).
   * Nếu có callback onLanguageChanged được đăng ký, nó cũng được gọi để thông báo cho các thành phần khác.
5. Cập nhật giao diện của ProfilePage:
   * Do toggleLanguage() được gọi trong setState(), ProfilePage được đánh dấu cần rebuild.
   * Flutter thực hiện rebuild cho ProfilePage.
   * Tất cả các widget trong ProfilePage gọi lại AppLanguage.get(key) để lấy text mới.
6. Lan tỏa thay đổi đến toàn bộ ứng dụng:
   * Do ProfilePage nằm trong MainScreen (StatefulWidget chứa Bottom Navigation), khi ProfilePage rebuild, Flutter cũng rebuild MainScreen.
   * Khi MainScreen rebuild, tất cả các tab khác (HomePage, DiscoverPage, LibraryPage, ChallengePage) cũng được rebuild.
   * Mọi widget gọi AppLanguage.get(key) đều nhận được text mới theo ngôn ngữ vừa chuyển.
7. Cập nhật nội dung động từ Data Layer:
   * Các Model có thuộc tính động (sử dụng getter) cũng tự động cập nhật:
     + CategoryModel: getter translatedName gọi lại AppLanguage.get() để lấy tên danh mục mới.
     + ChallengeModel và BadgeModel: do sử dụng getter function thay vì final List, mỗi lần truy cập sẽ tạo danh sách mới với text từ AppLanguage.get().
   * Điều này đảm bảo tất cả nội dung động đều đồng bộ với ngôn ngữ hiện tại.
8. Hiển thị kết quả cuối cùng:
   * Toàn bộ giao diện ứng dụng được cập nhật ngay lập tức:
     + Bottom Navigation Bar: "Trang chủ" ↔ "Home", "Khám phá" ↔ "Discover", v.v.
     + Tiêu đề các trang: "Thư viện" ↔ "Library", "Thử thách" ↔ "Challenge", v.v.
     + Danh mục sách: "Tiểu thuyết" ↔ "Novel", "Khoa học" ↔ "Science", v.v.
     + Thử thách: "Đọc 5 cuốn sách trong tháng" ↔ "Read 5 books this month"
     + Huy hiệu: "Người mới bắt đầu" ↔ "Beginner"
     + Tất cả các nút, label và thông báo đều được chuyển đổi.
9. Phản hồi trực quan cho người dùng:
   * Người dùng thấy toàn bộ giao diện thay đổi ngay lập tức mà không cần restart ứng dụng.
   * Mục "Language" trong ProfilePage cũng cập nhật hiển thị: "Tiếng Việt" ↔ "English".

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 4. Sơ đồ tuần tự - đổi ngôn ngữ*

# **Chương 4: Thiết kế giao diện**

## **4.1. Sơ đồ cấu trúc Widget**

A diagram of a company

AI-generated content may be incorrect.

*Hình 5 Sơ đồ cấu trúc Widget*

## **4.2. Giao diện các màn hình**

**Trang chủ**

**A screenshot of a book

AI-generated content may be incorrect. A screenshot of a book

AI-generated content may be incorrect.**

*Hình 6. Màn hình trang chủ*

## **Trang khám phá**

**A screenshot of a book

AI-generated content may be incorrect. A screenshot of a movie

AI-generated content may be incorrect.**

*Hình 7. Màn hình khám phá*

## **Trang thư viện**

**A screenshot of a chat

AI-generated content may be incorrect. A screenshot of a cell phone

AI-generated content may be incorrect.**

*Hình 8. Màn hình thư viện*

## **Trang thử thách**

**A screenshot of a phone

AI-generated content may be incorrect. A screenshot of a phone

AI-generated content may be incorrect.**

*Hình 9. Màn hình thử thách*

## **Trang cá nhân**

**A screenshot of a phone

AI-generated content may be incorrect.**

*Hình 10. Màn hình trang cá nhân*

## **chi tiết sách**

**A screenshot of a phone

AI-generated content may be incorrect. A screenshot of a phone

AI-generated content may be incorrect.**

*Hình 11. Màn hình chi tiết sách*

## **Trang đọc sách**

**A poster with a person in a red coat

AI-generated content may be incorrect.**

*Hình 12. Màn hình đọc sách*

## **chi tiết danh mục**

**A screenshot of a book

AI-generated content may be incorrect. A screenshot of a phone

AI-generated content may be incorrect.**

*Hình 13. Màn hình chi tiết danh mục*

# **Chương 5: Tổng kết và hướng phát triển**

## **5.1. Kết quả đạt được**

Sau quá trình nghiên cứu và phát triển, ứng dụng **BookReader** đã hoàn thành với các kết quả:

**Chức năng đã thực hiện**:

* Hiển thị danh sách sách theo 20 danh mục khác nhau
* Tìm kiếm và lọc sách theo từ khóa, danh mục
* Xem thông tin chi tiết sách (tiêu đề, tác giả, mô tả)
* Đọc sách PDF với công cụ điều khiển (zoom, chuyển trang, slider)
* Quản lý sách yêu thích (thêm/xóa bằng icon trái tim)
* Lưu tiến độ đọc (bookmark khi đọc PDF)
* Hệ thống thử thách đọc sách với progress tracking
* Hệ thống huy hiệu thành tích
* Chuyển đổi ngôn ngữ toàn bộ ứng dụng (Tiếng Việt ↔ English)

**Kiến trúc và thiết kế**:

* Áp dụng kiến trúc phân lớp với 3 lớp rõ ràng:
  + UI Layer: Giao diện người dùng (lib/ui/)
  + Service Layer: Logic nghiệp vụ (lib/services/)
  + Data Layer: Cấu trúc dữ liệu (lib/data/)
* Sử dụng mẫu thiết kế Singleton cho FavoritesManager và BookmarksManager
* Sử dụng Static Class cho LanguageService
* Giao diện tuân thủ Material Design 3

## **5.2. Hạn chế hiện tại**

### **5.2.1. Về dữ liệu và lưu trữ**

**Không có persistence**:

* Dữ liệu yêu thích và bookmark chỉ lưu trong RAM
* Mất toàn bộ dữ liệu khi tắt ứng dụng
* Chưa tích hợp SharedPreferences hoặc Database local

**Dữ liệu tĩnh**:

* Sách là dữ liệu mẫu cố định trong code
* Không có backend để quản lý dữ liệu động
* Người dùng không thể thêm sách mới

### **5.2.2. Về chức năng**

**Thiếu tính năng nâng cao**:

* Chưa có đăng nhập/đăng ký người dùng
* Chưa có ghi chú (annotation) trên PDF
* Chưa có highlight text
* Chưa có tìm kiếm trong PDF
* Thống kê còn đơn giản, chưa có biểu đồ chi tiết

**Chức năng đơn giản**:

* Bookmark chỉ lưu 1 trang cuối cùng cho mỗi sách
* Thử thách và huy hiệu có dữ liệu tĩnh, không tự động cập nhật
* Chưa có đồng bộ dữ liệu giữa nhiều thiết bị

### **5.2.3. Về nền tảng**

* Chỉ phát triển cho Android
* Chưa build cho iOS, Web, Desktop

## **5.3. Đề xuất cải tiến**

**1. Tích hợp Local Storage**

* Sử dụng SharedPreferences để lưu:
  + Danh sách ID sách yêu thích
  + Map bookmark (bookId → pageNumber)
  + Ngôn ngữ đã chọn

**2. Backend và Database**

* Xây dựng REST API (Node.js + Express hoặc Firebase)
* Database: PostgreSQL hoặc Firebase Firestore
* Quản lý: Sách, người dùng, yêu thích, bookmark, thống kê

**3. Authentication**

* Đăng ký/Đăng nhập (Email + Password)
* Đăng nhập qua Google/Facebook
* Quản lý profile cá nhân

**4. Cải thiện chức năng PDF**

* Thêm nhiều bookmark (đánh dấu nhiều trang quan trọng)
* Ghi chú (annotation) trên PDF
* Highlight text với nhiều màu
* Tìm kiếm từ khóa trong PDF
* Chế độ đọc ban đêm (Night Mode/Dark Mode)

**5. Thống kê nâng cao**

* Biểu đồ thời gian đọc (ngày/tuần/tháng)
* Phân tích thể loại sách yêu thích
* Thống kê số trang đọc theo thời gian
* Mục tiêu đọc cá nhân hóa

**6. Cloud Storage và Sync**

* Upload sách PDF lên cloud
* Đồng bộ yêu thích và bookmark giữa thiết bị
* Backup tự động