BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM
--- 🕮 ---

# ĐỒ ÁN TỐT NGHIỆP

## XÂY DỰNG APP THI TRẮC NGHIỆM TRÊN FLUTTER


**SINH VIÊN THỰC HIỆN : ĐÀO THÁI LAN**

**MÃ SINH VIÊN : 1451020295**

**KHOA : CÔNG NGHỆ THÔNG TIN**


**HÀ NỘI - 2024**

**BỘ GIÁO DỤC ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

---


**ĐÀO THÁI LAN**

# XÂY DỰNG APP THI TRẮC NGHIỆM TRÊN FLUTTER

**CHUYÊN NGÀNH** : **CÔNG NGHỆ THÔNG TIN**

**MÃ SỐ** : **1451020295**

**GIẢNG VIÊN HƯỚNG DẪN** : TS. TRẦN ĐỨC MINH

**HÀ NỘI – 2024**

# LỜI CAM ĐOAN

Em cam kết rằng mọi thông tin được trình bày trong báo cáo đồ án tốt nghiệp của em là hoàn toàn chân thành và dựa trên những kinh nghiệm thực tế của em trong quá trình thực hiện dự án. Em đã tự mình thực hiện và hoàn thành toàn bộ báo cáo này dưới sự hướng dẫn của thầy/cô và không sao chép hay tham khảo từ bất kỳ nguồn nào khác mà không được ghi chú rõ ràng. Em hiểu rằng việc vi phạm quy tắc đạo đức học thuật sẽ gây hậu quả nghiêm trọng và có thể dẫn đến việc bị loại khỏi trường. Do đó, em cam kết tuân thủ mọi quy định và nguyên tắc trong quá trình thực hiện báo cáo này. Xin chân thành cảm ơn!

Hà Nội, ngày 22 tháng 5 năm 2024

Sinh viên thực hiện

**Đào Thái Lan**

# LỜI CẢM ƠN

Trong lời đầu tiên của đồ án "**Xây dựng App thi trắc nghiệm trên flutter**", em muốn gửi những lời cám ơn và biết ơn chân thành nhất của mình tới tất cả những người đã hỗ trợ, giúp đỡ em về kiến thức và tinh thần trong quá trình thực hiện bài làm.

Em xin chân thành gửi lời cảm ơn tới các thầy, cô giáo trong **Trường Đại Học Đại Nam** nói chung và các thầy cô giáo trong **Khoa Công Nghệ Thông Tin** nói riêng đã tận tình giảng dạy, truyền đạt cho chúng em những kiến thức cũng như kinh nghiệm quý báu trong suốt quá trình học tập.

Đặc biệt, em xin gửi lời cảm ơn đến giảng viên – **Thầy Trần Đức Minh**. Thầy đã tận tình theo sát giúp đỡ, trực tiếp chỉ bảo, hướng dẫn trong suốt quá trình nghiên cứu và học tập của chúng em. Trong thời gian học tập với thầy, chúng em không những tiếp thu thêm nhiều kiến thức bổ ích mà còn học tập được tinh thần làm việc, thái độ nghiên cứu khoa học nghiêm túc, hiệu quả.

Do thời gian thực hiện có hạn kiến thức còn nhiều hạn chế nên bài làm của em chắc chắn không tránh khỏi những thiếu sót nhất định. Em rất mong nhận được ý kiến đóng góp của thầy, cô giáo và các bạn để em có thêm kinh nghiệm và tiếp tục hoàn thiện đề tài của mình.

Em xin chân thành cảm ơn!

# NHẬN XÉT

……
……
……
……
……
……
……
……
……
……
……
……

# MỤC LỤC

CHƯƠNG I. TỔNG QUAN VỀ ỨNG DỤNG ..... 1
1.1 Giới thiệu chung. ..... 1
1.2 Mục tiêu nghiên cứu. ..... 1
1.3 Tổng quan về hệ thống ứng dụng ..... 2
CHƯƠNG II CƠ SỞ LÝ THUYẾT ..... 6
2.1 Tìm hiểu về Framework Flutter ..... 6
2.1.1 Flutter là gì ..... 6
2.1.2 Các tính năng của Flutter ..... 7
2.1.3 Kiến trúc của Flutter ..... 8
2.1.4 Một số loại Widgets của Flutter thường gặp ..... 12
2.1.5 Flutter Hot Reload ..... 14
2.1.6 Flutter Packages và Plugins ..... 16
2.1.7 Flutter DevTools ..... 17
2.2 Giới thiệu về ngôn ngữ lập trình Dart ..... 18
2.2.1 Khái niệm ..... 18
2.2.2 Dart được sử dụng cho mục đích gì? ..... 18
2.2.3 Tại sao nên học ngôn ngữ Dart? ..... 19
2.2.4 Ưu – nhược điểm của Dart là gì? ..... 19
2.2.5 Các tính năng của ngôn ngữ Dart ..... 20
2.2.6 Lập trình hướng đối tượng với Dart ..... 20
2.3 Firebase – lưu trữ online ..... 21
2.3.1 Firebase là gì? ..... 21
2.3.2 Lịch sử phát triển của Firebase ..... 22
2.3.3 Những tính năng chính của Firebase ..... 23
CHƯƠNG III. PHÂN TÍCH VÀ THIẾT KẾ ỨNG DỤNG ..... 28
3.1 Mô tả chức năng ứng dụng ..... 28
3.1.1 Chức năng dành cho người dùng. ..... 28

3.1.2 Các chức năng dành cho Admin. ..... 28
3.2 Thiết kế UML. ..... 28
3.2.1 Biểu đồ usecase tổng quát ..... 28
3.2.2 Chức năng đăng nhập của Admin ..... 30
3.2.3 Chức năng thêm bài thi ..... 34
3.2.4 Chức năng thêm câu hỏi ..... 37
3.2.5 Chức năng đăng nhập của người dùng ..... 40
3.2.6 Chức năng chọn bài thi ..... 44
3.2.7 Chức năng làm bài thi ..... 46
3.2.8 Chức năng xem kết quả thi ..... 50
3.2.9 Chức năng xem lịch sử thi ..... 53
3.2.10 Biểu đồ lớp thiết kế ..... 54
3.3 Thiết kế cơ sở dữ liệu ..... 55
3.4 Triển khai mô hình MVC ..... 57
CHƯƠNG IV. KẾT LUẬN VÀ ĐỊNH HƯỚNG ..... 60
TÀI LIỆU THAM KHẢO ..... 61

# MỤC LỤC HÌNH ẢNH

Chương I

Hình 1. 1 Mô hình giao tiếp Client – Server ..... 2
Hình 1. 2 Quy trình hoạt động của hệ thống ..... 4

Chương II

Hình 2. 1 Các tính năng của Flutter ..... 7
Hình 2. 2 Kiến trúc flutter ..... 10
Hình 2. 3 Cây widget ..... 11
Hình 2. 4 Widget cột ..... 14
Hình 2. 5 Google Firebase ..... 23
Hình 2. 6 Realtime Database ..... 24
Hình 2. 7 Authentication ..... 25
Hình 2. 8 Cloud Firestore ..... 26
Hình 2. 9 Cloud Functions ..... 27

Chương III

Hinh 3. 1Biểu đồ usecase tổng quát ..... 29
Hinh 3. 2 Biểu đồ usecase đăng nhập ..... 31
Hinh 3. 3 Giao diện đăng nhập Admin ..... 32
Hinh 3. 4 Giao diện trang Admin ..... 32
Hinh 3. 5 Biểu đồ tuần tự đăng nhập Admin ..... 33
Hinh 3. 6 Biểu đồ hoạt động đăng nhập Admin ..... 33
Hinh 3. 7 Biểu đồ usecase quản lý bài thi ..... 35
Hinh 3. 8 Giao diện form thêm bài thi ..... 36
Hinh 3. 9 Biểu đồ hoạt động thêm bài thi ..... 36
Hinh 3. 10 Biểu đồ usecase quản lý câu hỏi ..... 38
Hinh 3. 11 Giao diện danh sách câu hỏi ..... 39
Hinh 3. 12 Giao diện form thêm câu hỏi ..... 39
Hinh 3. 13 Biểu đồ hoạt động thêm câu hỏi ..... 40
Hinh 3. 14 Biểu đồ usecase đăng nhập của người dùng ..... 41
Hinh 3. 15 Giao diện màn hình đăng nhập ..... 42
Hinh 3. 16 Giao diện màn hình form đăng nhập google ..... 42

Hinh 3. 17 Biểu đồ tuần tự đăng nhập người dùng .....43
Hinh 3. 18 Biểu đồ hoạt động đăng nhập .....43
Hinh 3. 19 Biểu đồ usecase chọn bài thi .....45
Hinh 3. 20 Giao diện màn hình danh sách bài thi .....45
Hinh 3. 21 Biểu đồ tuần tự chọn bài thi .....46
Hinh 3. 22 Biểu đồ usecase làm bài thi .....47
Hinh 3. 23 Giao diện màn hình câu hỏi 1 .....48
Hinh 3. 24 Giao diện màn hình câu hỏi .....48
Hinh 3. 25 Giao diện màn hình nộp bài thi .....49
Hinh 3. 26 Biểu đồ tuần tự làm bài thi .....49
Hinh 3. 27 Biểu đồ hoạt động làm bài thi .....50
Hinh 3. 28 Biểu đồ use case xem kết quả thi .....51
Hinh 3. 29 Giao diện màn hình kết quả thi .....52
Hinh 3. 30 Giao diện màn hình xem đáp án .....52
Hinh 3. 31 Biểu đồ usecase lịch sử thi .....53
Hinh 3. 32 Giao diện màn hình lịch sử thi .....54
Hinh 3. 33Biểu đồ lớp .....54
Hinh 3. 34 Mô hình MVC .....57
Hinh 3. 35 Cấu trúc cây thư mục .....58
Hinh 3. 36 Cấu trúc thư mục models .....58
Hinh 3. 37 Cấu trúc thư mục views .....59
Hinh 3. 38Cấu trúc thư mục controllers .....59

# CHƯƠNG I. TỔNG QUAN VỀ ỨNG DỤNG

## 1.1 Giới thiệu chung.

Ứng dụng Thi Trắc Nghiệm là một ứng dụng di động tiên tiến được phát triển bằng Flutter, với mục đích hỗ trợ người dùng kiểm tra và nâng cao kiến thức của họ trên nhiều lĩnh vực khác nhau. Ứng dụng này không chỉ nổi bật với giao diện người dùng đẹp mắt và dễ sử dụng mà còn tích hợp nhiều tính năng linh hoạt, giúp người học dễ dàng truy cập và ôn tập hiệu quả.

Thi Trắc Nghiệm mang đến trải nghiệm học tập phong phú, từ các bài kiểm tra nhanh đến những đề thi chi tiết, phù hợp với nhiều trình độ và nhu cầu học tập khác nhau. Người dùng có thể theo dõi tiến trình học tập của mình, xem lại kết quả kiểm tra và nhận được các gợi ý, giải thích chi tiết cho từng câu hỏi. Điều này giúp họ không chỉ kiểm tra kiến thức hiện tại mà còn hiểu sâu hơn về những khái niệm mà mình chưa nắm vững.

Ngoài ra, ứng dụng còn cung cấp các tính năng tùy chỉnh như tạo đề thi riêng, chọn lĩnh vực và mức độ khó của câu hỏi, giúp cá nhân hóa quá trình học tập. Với việc sử dụng Flutter, Thi Trắc Nghiệm đảm bảo hoạt động mượt mà trên các hệ điều hành khác nhau, mang lại trải nghiệm liền mạch cho người dùng.

## 1.2 Mục tiêu nghiên cứu.

Để nâng cao kỹ năng phát triển ứng dụng đa nền tảng bằng Flutter, cần tập trung vào việc áp dụng các nguyên tắc phát triển phần mềm, thiết kế giao diện người dùng và quản lý mã nguồn để tạo ra một ứng dụng chất lượng cao và hiệu suất tốt trên nhiều nền tảng khác nhau.

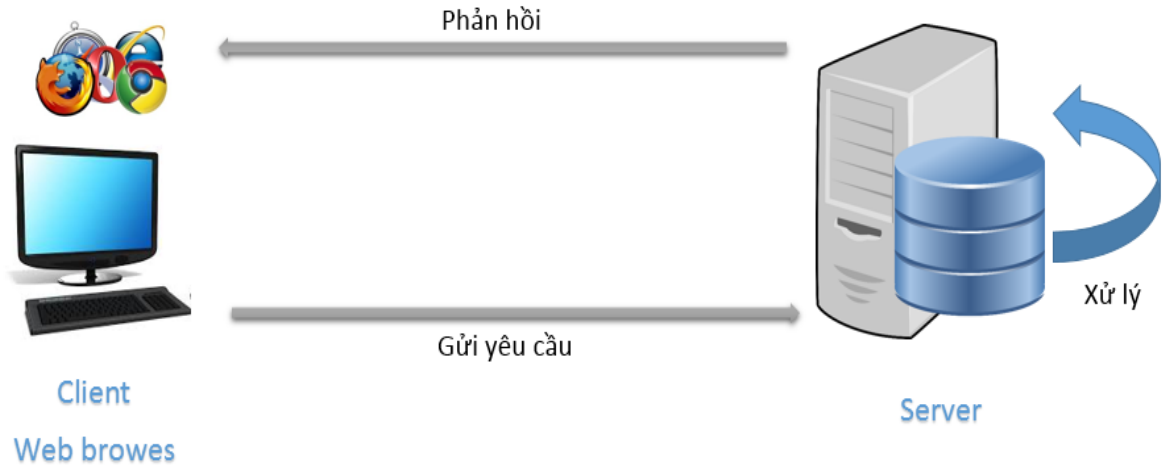
Flutter không chỉ là một công nghệ phát triển ứng dụng, mà còn là một môi trường cho sự sáng tạo và thử nghiệm các tính năng mới. Việc tận dụng widget, plugin và các công cụ hỗ trợ khác trong quá trình phát triển ứng dụng là quan trọng để tối ưu hóa trải nghiệm người dùng và tăng cường hiệu suất.

Mục tiêu cuối cùng của dự án là tạo ra một ứng dụng thi trắc nghiệm đa nền tảng thực tế và có giá trị sử dụng cho người dùng. Việc này không chỉ là một dự án nghiên cứu mà còn là cơ hội để sản xuất một sản phẩm thực tế có thể tiếp cận và sử dụng rộng rãi trên thị trường. Đồng thời, việc áp dụng kiến thức và kỹ năng vào dự án thực tế như vậy

cũng sẽ giúp phát triển và làm chủ các kỹ năng phát triển ứng dụng đa nền tảng một cách linh hoạt và chuyên nghiệp hơn.

## 1.3 Tổng quan về hệ thống ứng dụng

**Mô hình Client – Server** là một mô hình nổi tiếng trong mạng máy tính, được áp dụng rất rộng rãi và là mô hình của mọi trang web hiện nay. Ý tưởng của mô hình này là máy con (client) gửi một yêu cầu (request) đến máy chủ (server), máy chủ sẽ xử lý và trả kết quả về cho máy khách.




*Hình 1. 1 Mô hình giao tiếp Client – Server*

*(Nguồn: Internet)*

**Client (Flutter App)**

- **Giao diện người dùng (UI)**: Ứng dụng di động được phát triển bằng Flutter, cung cấp một giao diện tương tác cho người dùng (học sinh, giáo viên) để đăng nhập, chọn kỳ thi, làm bài thi và xem kết quả.
- **Flutter SDK**: Công cụ phát triển cho phép xây dựng ứng dụng di động cho cả iOS và Android từ một codebase duy nhất.

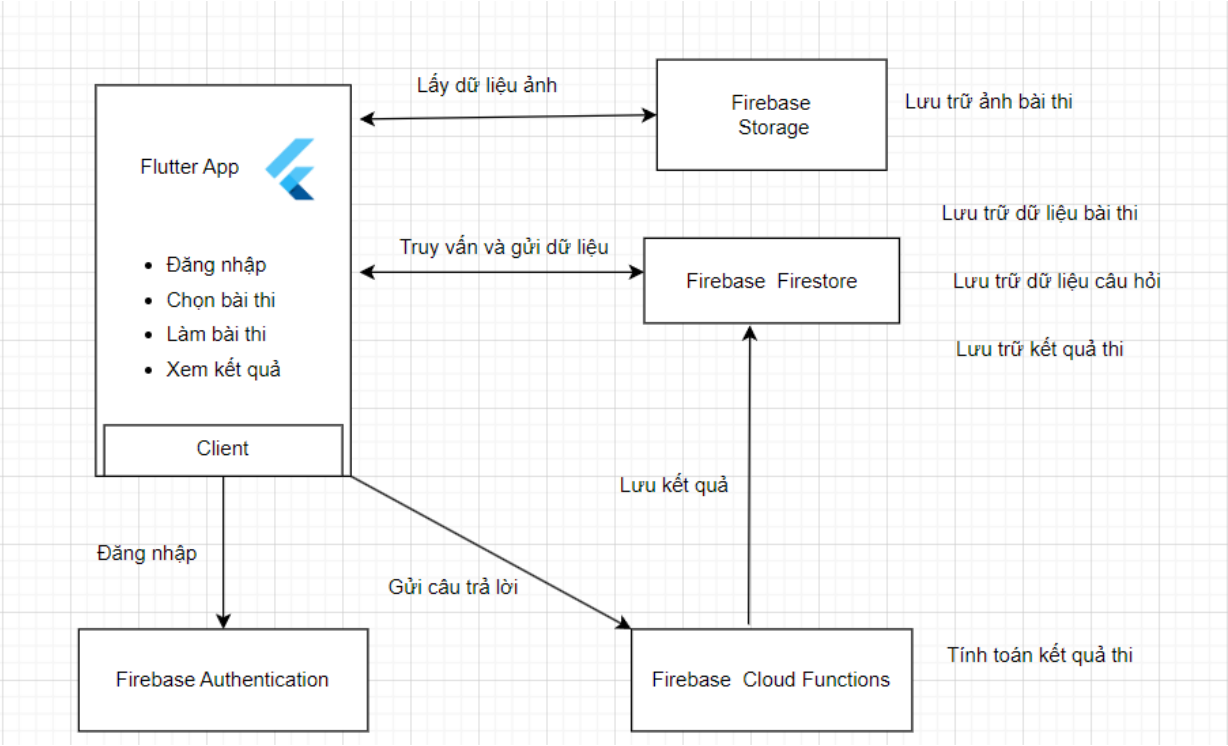
**Backend (Firebase)**

- **Firebase Authentication**: Quản lý xác thực người dùng với các phương thức như email/password, Google, Facebook, và nhiều phương thức khác.

- **Cloud Firestore**: Cơ sở dữ liệu NoSQL của Firebase để lưu trữ thông tin người dùng, kỳ thi, câu hỏi, câu trả lời và kết quả thi. Dữ liệu được đồng bộ hóa thời gian thực với ứng dụng Flutter.
- **Firebase Storage**: Lưu trữ các tệp đính kèm như hình ảnh, âm thanh nếu cần.
- **Firebase Cloud Functions**: Xử lý logic nghiệp vụ trên server, chẳng hạn như tính toán kết quả thi, gửi thông báo hoặc xử lý các sự kiện phức tạp.
- **Firebase Hosting**: Lưu trữ các tệp tĩnh của ứng dụng web (nếu có), hoặc trang quản trị.

**Quy trình hoạt động của hệ thống**

1. Người dùng mở ứng dụng Flutter và đăng nhập.
- Flutter App gửi thông tin đăng nhập đến Firebase Authentication để xác thực.
2. Sau khi đăng nhập thành công, người dùng chọn bài thi muốn tham gia.
- Flutter App truy vấn Cloud Firestore để lấy danh sách các bài thi có sẵn.
3. Người dùng bắt đầu làm bài thi.
- Flutter App tải các câu hỏi từ Cloud Firestore và hiển thị cho người dùng.
- Các câu trả lời của người dùng được lưu trữ tạm thời trên ứng dụng hoặc gửi ngay đến Cloud Firestore.
4. Sau khi hoàn thành bài thi, kết quả được tính toán và lưu trữ.
- Flutter App gửi dữ liệu câu trả lời đến Firebase Cloud Functions, nơi sẽ xử lý và tính toán kết quả.
- Kết quả sau đó được lưu lại trong Cloud Firestore.
5. Người dùng có thể xem kết quả thi ngay lập tức.
- Flutter App truy vấn Cloud Firestore để lấy kết quả và hiển thị cho người dùng.




*Hình 1. 2 Quy trình hoạt động của hệ thống*

**Bảo mật và quản lý**

- **Xác thực và phân quyền**: Firebase Authentication đảm bảo chỉ người dùng hợp lệ mới có thể truy cập hệ thống.
- **Quyền truy cập dữ liệu**: Các quy tắc bảo mật của Firestore (Firestore Security Rules) được cấu hình để kiểm soát quyền truy cập và chỉnh sửa dữ liệu dựa trên vai trò của người dùng (học sinh, giáo viên, quản trị viên).

**Thành phần khác**

- **Firebase Analytics**: Theo dõi và phân tích hành vi người dùng, cung cấp thông tin để cải thiện trải nghiệm người dùng.
- **Firebase Crashlytics**: Theo dõi lỗi và sự cố của ứng dụng để cải thiện tính ổn định.
- **Push Notifications**: Sử dụng Firebase Cloud Messaging (FCM) để gửi thông báo đến người dùng về các kỳ thi sắp diễn ra hoặc kết quả thi.

**Lưu đồ hoạt động của hệ thống**

1. Người dùng mở ứng dụng Flutter.
2. Người dùng đăng nhập qua Firebase Authentication.
3. Ứng dụng Flutter lấy dữ liệu kỳ thi từ Cloud Firestore.

4. Người dùng chọn kỳ thi và làm bài thi.
5. Ứng dụng Flutter gửi câu trả lời đến Cloud Firestore và gọi Cloud Functions để tính kết quả.
6. Kết quả thi được lưu trong Cloud Firestore và hiển thị cho người dùng.
7. Hệ thống gửi thông báo cho người dùng qua Firebase Cloud Messaging.

# CHƯƠNG II CƠ SỞ LÝ THUYẾT

## 2.1 Tìm hiểu về Framework Flutter

### 2.1.1 Flutter là gì

Flutter là một bộ SDK đa nền tảng có thể hoạt động trên iOS và Android do Google phát triển được sử dụng để tạo ra các ứng dụng dành cho di động (native app).

Flutter gồm 2 thành phần quan trọng:

- Một SDK (Software Development Kit): Một bộ sưu tập các công cụ sẽ giúp phát triển các ứng dụng của mình.
- Một Framework (UI Library based on widgets): Một tập hợp các thành phần giao diện người dùng (UI) có thể tái sử dụng (button, text inputs, slider, v.v.) có thể cá nhân hóa tùy theo nhu cầu của riêng mình.

Có rất nhiều framework có sẵn, cung cấp các tính năng tuyệt vời để phát triển các ứng dụng di động. Để phát triển các ứng dụng dành cho thiết bị di động, Android cung cấp một framework gốc dựa trên ngôn ngữ Java và Kotlin, trong khi iOS cung cấp một framework dựa trên ngôn ngữ Objective-C/Swift.

Vì vậy, cần hai ngôn ngữ và framework khác nhau để phát triển ứng dụng cho cả hai hệ điều hành. Ngày nay, để khắc phục sự phức tạp này, có một số framework đã được giới thiệu hỗ trợ cả hệ điều hành cùng với các ứng dụng dành cho máy tính để bản. Những loại framework này được gọi là công cụ phát triển đa nền tảng.

Framework phát triển đa nền tảng có khả năng viết một code và có thể triển khai trên nhiều nền tảng khác nhau (Android, iOS và Máy tính để bản). Nó tiết kiệm rất nhiều thời gian và nỗ lực phát triển của các nhà phát triển.

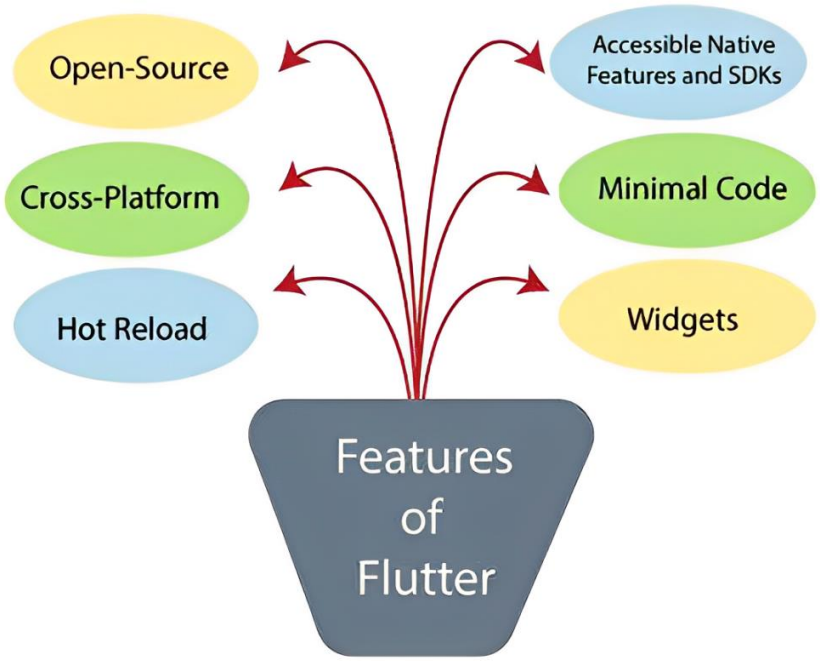
Có một số công cụ có sẵn để phát triển đa nền tảng, bao gồm các công cụ dựa trên web. Mỗi framework này có mức độ thành công khác nhau trong ngành công nghiệp di động. Gần dây, một framework công tác mới đã được giới thiệu trong họ phát triển đa nền tảng có tên là Flutter được phát triển tử Google.

Flutter là một bộ công cụ giao diện người dùng để tạo các ứng dụng nhanh, đẹp, được biên dịch nguyên bản cho thiết bị di động, web và máy tính để bản với một ngôn ngữ lập trình và cơ sở code duy nhất. Nó là miễn phí và code nguồn mở. Ban đầu nó được phát triển từ Google và bây giờ được quản lý theo tiêu chuẩn ECMA. Ứng dụng Flutter sử dụng ngôn ngữ lập trình Dart để tạo ứng dụng.

Flutter chủ yếu được tối ưu hóa cho các ứng dụng di động 2D có thể chạy trên cả nền tảng Android và iOS. Chúng ta cũng có thể sử dụng nó để xây dựng các ứng dụng đầy đủ tính năng, bao gồm máy ảnh, bộ nhớ, vị trí địa lý, mạng, SDK của bên thứ ba, v.v.

### 2.1.2 Các tính năng của Flutter

Flutter cung cấp các phương pháp dễ dàng và đơn giản để bắt đầu xây dựng các ứng dụng dành cho thiết bị di động và máy tính để bàn đẹp mắt với một bộ thiết kế material design và widget phong phú. Ở đây, chúng ta sẽ thảo luận về các tính năng chính của nó để phát triển framework di động.




*Hình 2. 1 Các tính năng của Flutter*

*(Nguồn: Internet)*

**Code nguồn mở(Open-Source:)**: Flutter là một framework code nguồn mở và miễn phí để phát triển các ứng dụng di động.

**Đa nền tảng(Cross-platform)**: Tính năng này cho phép Flutter viết code một lần, duy trì và có thể chạy trên các nền tảng khác nhau. Nó tiết kiệm thời gian, công sức và tiền bạc của các nhà phát triển.

**Hot Reload(Hot Reload)**: Bất cứ khi nào nhà phát triển thực hiện thay đổi trong code, thì những thay đổi này có thể được nhìn thấy ngay lập tức với Hot Reload. Nó có

nghĩa là những thay đổi hiển thị ngay lập tức trong chính ứng dụng. Đây là một tính năng rất tiện dụng, cho phép nhà phát triển sửa các lỗi ngay lập tức.

**Các tính năng và SDK gốc có thể truy cập (Accessible Native Features and SDKs)**: Tính năng này cho phép quá trình phát triển ứng dụng dễ dàng và thú vị thông qua code gốc của Flutter, tích hợp bên thứ ba và các API nền tảng. Do đó, chúng tôi có thể dễ dàng truy cập SDK trên cả hai nền tảng.

**Code tối thiểu (Minimal code)**: Ứng dụng Flutter được phát triển bởi ngôn ngữ lập trình Dart, sử dụng biên dịch JIT và AOT để cải thiện thời gian khởi động tổng thể, hoạt động và tăng tốc hiệu suất. JIT nâng cao hệ thống phát triển và làm mới giao diện người dùng mà không cần nỗ lực thêm vào việc xây dựng hệ thống mới.

**Widget**: framework công tác Flutter cung cấp các widget có khả năng phát triển các thiết kế cụ thể có thể tùy chỉnh. Quan trọng nhất, Flutter có hai bộ widget: Material Design và các widget Cupertino giúp mang lại trải nghiệm không có trục trặc trên tất cả các nền tảng.

### 2.1.3 Kiến trúc của Flutter

Kiến trúc Flutter chủ yếu bao gồm bốn thành phần.

- Động cơ Flutter (Flutter Engine)
- Thư viện nền tảng (Foundation Library)
- Vật dụng (Widgets)
- Thiết kế các widget cụ thể (Design Specific Widgets)
- Cử chỉ(Gestures)
- Quản lý State
- Lớp(Layers)

- **Flutter Engine**

Nó là một cổng để giúp chạy các ứng dụng di động chất lượng cao và cơ bản dựa trên ngôn ngữ C++. Nó triển khai các thư viện lõi Flutter bao gồm animation và đồ họa, tệp và mạng I / O, kiến trúc plugin, hỗ trợ trợ năng và thời gian chạy dart để phát triển, biên dịch và chạy các ứng dụng Flutter. Phải sử dụng thư viện đồ họa mã nguồn mở của Google, Skia, để hiển thị đồ họa cấp thấp.

- **Thư viện nền tảng (Foundation Library)**

Nó chứa tất cả các gói cần thiết cho các khối build cơ bản để viết một ứng dụng Flutter. Các thư viện này được viết bằng ngôn ngữ Dart.

- **Vật dụng (Widget)**

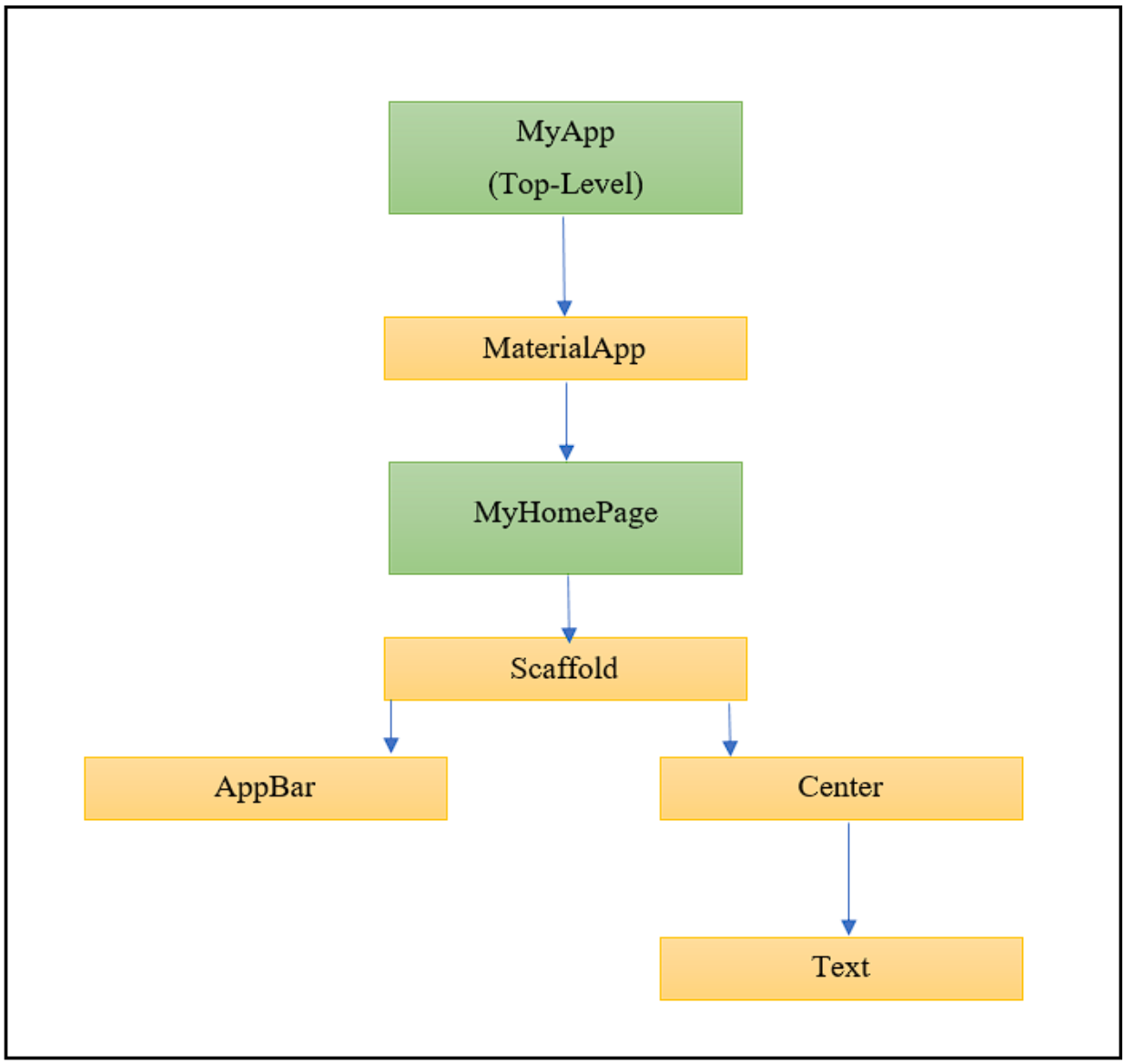
Trong Flutter, mọi thứ đều là một widget, đó là khái niệm cốt lõi của framework. Widget trong Flutter về cơ bản là một thành phần giao diện người dùng ảnh hưởng và kiểm soát chế độ xem và giao diện của ứng dụng. Nó đại diện cho một mô tả bất biến về một phần của giao diện người dùng và bao gồm đồ họa, văn bản, hình dạng và animation được tạo bằng các widget. Có hai loại chính của Widget trong Flutter:

+ **StatelessWidget**: Đây là một widget không thay đổi trạng thái sau khi được xây dựng.
+ **StatefulWidget**: Đây là một widget có thể thay đổi trạng thái sau khi được xây dựng.

**Các Đặc điểm của Widget trong Flutter**

+ **Immutable và Lightweight**: Widget là không thay đổi (immutable), điều này có nghĩa là một khi widget đã được xây dựng, bạn không thể thay đổi nội dung của nó. Điều này làm cho Flutter có thể hiệu quả với việc vẽ giao diện.
+ **Composable**: Widget có thể được kết hợp lại với nhau để tạo thành các giao diện phức tạp. Flutter khuyến khích sử dụng nhiều widget nhỏ hơn thay vì một số lớn để quản lý và tái sử dụng giao diện hiệu quả.
+ **Material Design và Cupertino Style**: Flutter cung cấp các widget thực thi các hướng dẫn thiết kế của Material Design (cho Android) và Cupertino (cho iOS), giúp ứng dụng có giao diện gần giống với các ứng dụng chuẩn trên hai hệ điều hành này.
+ **Thư viện Widget Phong phú**: Flutter đi kèm với một loạt các widget được xây dựng sẵn để giúp bạn xây dựng các thành phần giao diện như nút, danh sách, menu, hộp thoại, v.v.

Trong Flutter, ứng dụng tự nó là một widget chứa nhiều widget con. Điều đó có nghĩa rằng ứng dụng là tiện ích con cấp cao nhất và giao diện người dùng của nó được xây dựng bằng cách sử dụng một hoặc nhiều tiện ích con, bao gồm các tiện ích con phụ. Tính năng này giúp tạo một giao diện người dùng phức tạp rất dễ dàng.



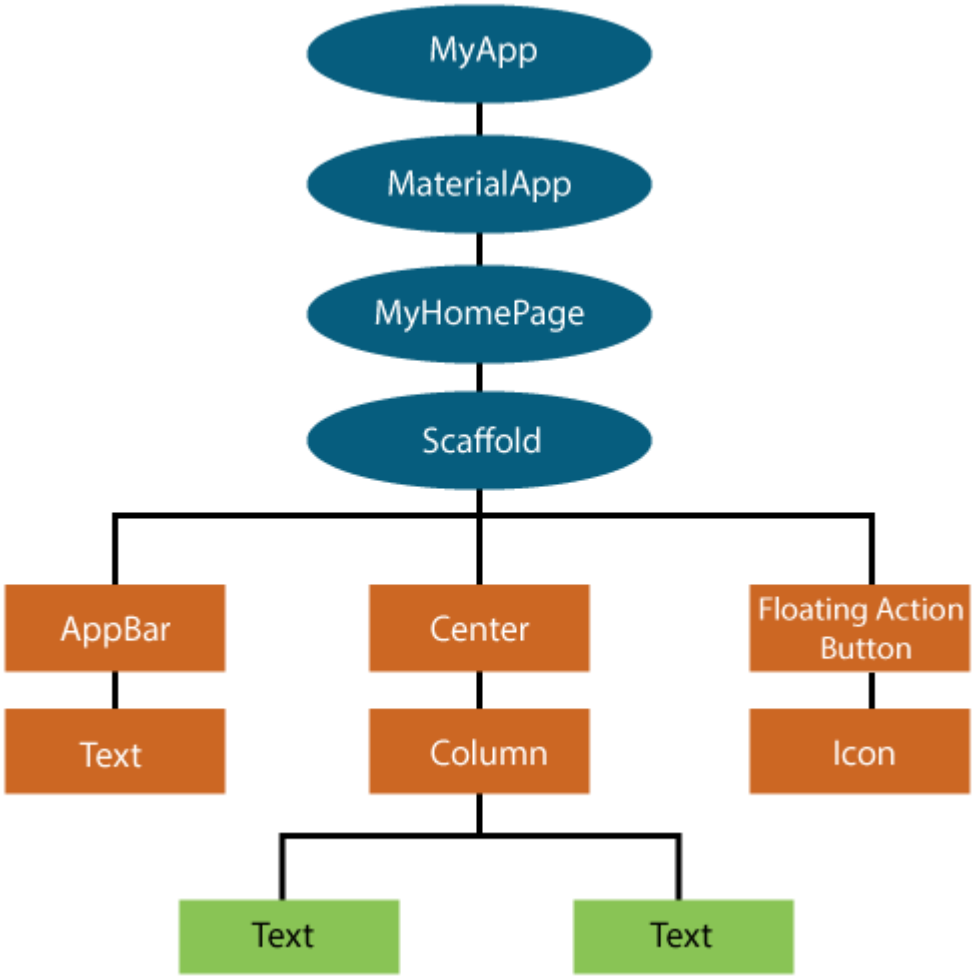

*Hình 2. 2 Kiến trúc flutter*

Có thể thấy rằng tất cả các thành phần đều là các widget có chứa các widget con. Do đó, ứng dụng Flutter tự nó là một widget.

**Cấu trúc cây Widget**

Các widget trong Flutter được tổ chức thành một cây (Widget Tree), trong đó mỗi widget là một nút trong cây. Cây widget này đại diện cho toàn bộ giao diện người dùng của ứng dụng.

Khi widget thay đổi trạng thái, Flutter sẽ xây dựng lại cây widget và cập nhật giao diện người dùng dựa trên các thay đổi.




*Hình 2. 3 Cây widget*

*(Nguồn: cafedev.vn)*

- **Thiết kế các widget cụ thể**

Framework Flutter có hai bộ widget phù hợp với các ngôn ngữ thiết kế cụ thể. Đây là Material Design cho ứng dụng Android và Cupertino Style cho ứng dụng IOS.

- **Cử chỉ(Gestures)**

Cử chỉ (Gestures) là một tính năng thú vị trong Flutter cho phép chúng ta tương tác với ứng dụng di động (hoặc bất kỳ thiết bị dựa trên cảm ứng). Nói chung, cử chỉ xác định bất kỳ hành động hoặc chuyển động vật lý nào của người dùng nhằm mục đích kiểm soát thiết bị di động. Một số ví dụ về cử chỉ là:

- Khi màn hình di động bị khóa, bạn trượt ngón tay trên màn hình để mở khóa.
- Nhấn vào một nút trên màn hình điện thoại di động
- Nhấn và giữ biểu tượng ứng dụng trên thiết bị dựa trên cảm ứng để kéo biểu tượng đó qua các màn hình.

Sử dụng tất cả những cử chỉ này trong cuộc sống hàng ngày để tương tác với điện thoại hoặc thiết bị dựa trên cảm ứng.

- **Trang thái(State)**

Trạng thái là thông tin có thể đọc được khi widget được tạo và có thể thay đổi hoặc sửa đổi trong suốt thời gian tồn tại của ứng dụng. Nếu muốn thay đổi widget của mình, cần cập nhật đối tượng trạng thái, có thể được thực hiện bằng cách sử dụng hàm setState() có sẵn cho các widget Stateful. Hàm setState() cho phép chúng ta thiết lập các thuộc tính của đối tượng trạng thái kích hoạt vẽ lại giao diện người dùng.

Quản lý state (trạng thái) là một trong những quy trình phổ biến và cần thiết nhất trong vòng đời của một ứng dụng. Theo tài liệu chính thức, Flutter mang tính chất khai báo. Điều đó có nghĩa là Flutter xây dựng giao diện người dùng của mình bằng cách phản ánh trạng thái hiện tại của ứng dụng. Hình sau giải thích rõ hơn về nơi có thể xây dựng giao diện người dùng từ trạng thái ứng dụng.

- **Lớp(layout)**

Khái niệm chính của cơ chế bố trí là widget. Biết rằng sự giả định mọi thứ như một widget Vì vậy, hình ảnh, biểu tượng, văn bản và thậm chí cả bố cục(layout) của ứng dụng đều là widget. Ở đây, một số thứ không thấy trên giao diện người dùng ứng dụng của mình, chẳng hạn như các hàng, cột và lưới sắp xếp, ràng buộc và căn chỉnh các widget hiển thị cũng là các widget.

Flutter cho phép chúng ta tạo bố cục bằng cách soạn nhiều widget để xây dựng các widget phức tạp hơn.

### 2.1.4 Một số loại Widgets của Flutter thường gặp

Bất cứ khi nào viết mã để xây dựng bất cứ thứ gì trong Flutter, nó sẽ nằm trong một widget. Mục đích chính là xây dựng ứng dụng từ các widget. Nó mô tả chế độ xem ứng dụng trông như thế nào với cấu hình và trạng thái hiện tại của chúng.

Khi thực hiện bất kỳ thay đổi nào trong code, widget con sẽ xây dựng lại mô tả của nó bằng cách tính toán sự khác biệt của widget con hiện tại và trước đó để xác định những thay đổi tối thiểu đối với việc hiển thị trong giao diện người dùng của ứng dụng.

Các widget được lồng vào nhau để xây dựng ứng dụng. Nó có nghĩa là thư mục gốc của ứng dụng tự nó là một widget, và tất cả các cách nhìn xuống cũng là một widget.

Ví dụ: một widget có thể hiển thị một thứ gì đó, có thể xác định thiết kế, có thể xử lý tương tác, v.v.

**Widget hiển thị**

Các widget liên quan đến dữ liệu đầu vào và ra của người dùng. Một số loại quan trong của widget con này là:

- **Text(Văn bản)**

Một widget con trong Flutter cho phép chúng ta hiển thị một chuỗi Text với một dòng duy nhất trong ứng dụng của chúng ta . Tùy thuộc vào các ràng buộc về bố cục, chúng ta có thể ngắt chuỗi trên nhiều dòng hoặc tất cả có thể được hiển thị trên cùng một dòng. Nếu chúng ta không chỉ định bất kỳ kiểu nào cho widget Text, nó sẽ sử dụng kiểu lớp DefaultTextStyle gần nhất.

- **Button(Nút)**

Phần tử điều khiển đồ họa cung cấp cho người dùng kích hoạt một sự kiện như thực hiện hành động, lựa chọn, tìm kiếm mọi thứ, v.v. Chúng có thể được đặt ở bất kỳ đâu trong giao diện người dùng như hộp thoại, biểu mẫu, thẻ, thanh công cụ, v.v. Các nút là các widget Flutter, là một phần của thư viện material design. Flutter cung cấp một số loại nút có hình dạng, kiểu dáng và tính năng khác nhau.

- **Image(Ảnh)**

Widget con này giữ hình ảnh có thể tìm nạp hình ảnh từ nhiều nguồn như từ thư mục nội dung hoặc trực tiếp từ URL. Nó cung cấp nhiều hàm tạo để tải hình ảnh.

**Widget ẩn**

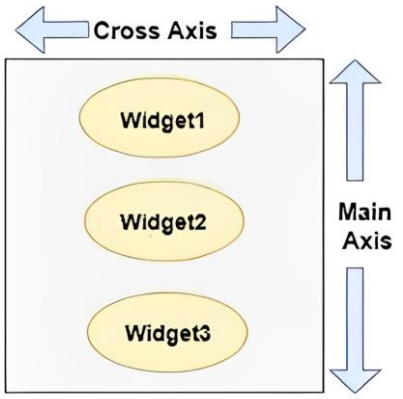
Các widget vô hình có liên quan đến cách bố trí và kiểm soát các widget. Nó cung cấp việc kiểm soát cách các widget thực sự hoạt động và cách chúng sẽ hiển thị trên màn hình. Một số loại widget quan trọng là:

- **Column(Cột)**

Widget này sắp xếp các con của nó theo hướng dọc trên màn hình . Nói cách khác, nó sẽ mong đợi một mảng dọc gồm các widget con. Nếu widget cần lấp đầy không gian dọc có sẵn, chúng ta phải bọc các widget trong widget Mở rộng.

Một widget dạng cột không thể cuộn được vì nó hiển thị các widget trong chế độ xem hiển thị đầy đủ. Vì vậy, sẽ được coi là sai nếu chúng ta có nhiều con hơn trong một cột sẽ không phù hợp với không gian có sẵn. Nếu chúng ta muốn tạo một danh sách các widget cột có thể cuộn được, chúng ta cần sử dụng ListView Widget.

Chúng ta cũng có thể kiểm soát cách một widget cột sắp xếp các con của nó bằng cách sử dụng thuộc tính mainAxisAlignment và crossAxisAlignment. Trục chéo của cột sẽ chạy theo chiều ngang và trục chính sẽ chạy theo chiều dọc . Hình ảnh minh họa dưới đây giải thích rõ ràng hơn.




*Hình 2. 4 Widget cột*

*(Nguồn: Internet)*

- **Stack**

Ngăn xếp (Stack) là một widget trong Flutter chứa danh sách các widget và đặt chúng trên đầu các widget khác. Nói cách khác, ngăn xếp cho phép các nhà phát triển chồng nhiều widget vào một màn hình duy nhất và hiển thị chúng từ dưới lên trên. Do đó, widget đầu tiên là mục dưới cùng và widget cuối cùng là mục trên cùng.

### 2.1.5 Flutter Hot Reload

**Flutter Hot Reload** là một tính năng được tìm kiếm nhiều trong framework của Google, cho phép các nhà phát triển thực hiện thay đổi mã và xem kết quả gần như ngay lập tức – mà không cần khởi động lại ứng dụng. Tính năng này tạo điều kiện cho việc lặp lại và sàng lọc nhanh chóng thiết kế của ứng dụng, thử nghiệm các bố cục và cấu hình giao diện người dùng khác nhau, sửa lỗi và quan trọng nhất là giảm đáng kể thời gian phát triển. Do đó, Hot Reload có thể mang lại lợi ích to lớn cho  công ty phát triển ứng dụng Flutter bằng cách cho phép thay đổi mã theo thời gian thực và cập nhật tức thì, giúp quá trình phát triển hiệu quả và hợp tác hơn.

Tính năng Flutter Hot Reload cho phép các nhà phát triển xem những thay đổi họ thực hiện đối với mã được phản ánh trên trình mô phỏng hoặc thiết bị ngay lập tức mà không cần phải khởi động lại toàn bộ ứng dụng.

Khi mô tả tính năng Hot Reload, điều quan trọng cần nhấn mạnh là Dart VM sử dụng trình biên dịch JIT (Just-in-Time) để chuyển đổi mã thành mã máy gốc, diễn ra ngay trước khi thực thi chương trình. JIT dựa trên dự đoán mã vì nó có quyền truy cập vào thông tin thời gian chạy động, dẫn đến các giải pháp tiết kiệm thời gian, chẳng hạn như thông báo cho nhà phát triển rằng một chức năng cụ thể chưa được sử dụng ở bất kỳ đâu.

Hot Reload sẽ xây dựng lại cây tiện ích nhưng vẫn giữ trạng thái ứng dụng như cũ. Khi sử dụng tính năng Hot Reload, các hàm `main()` và `initState()` không được gọi. Nếu cần build lại các chức năng này thì bạn nên sử dụng Hot restart hoặc Full restart:

- Hot Restart: công cụ kích hoạt mã nguồn của ứng dụng dự án được biên dịch lại, bắt đầu từ trạng thái ban đầu/mặc định, trong đó trạng thái được bảo toàn bị hủy. Công cụ này nhanh hơn Full Reset rất nhiều nhưng tốn nhiều thời gian hơn Hot Reload.
- Full Restart: công cụ xây dựng dự án ứng dụng từ đầu, còn được gọi là "Cold start".

Ngoài ra, đôi khi nhà phát triển phải sử dụng Hot Restart thay vì Hot Reload, ví dụ:

- Nếu ứng dụng ở chế độ nền quá lâu và/hoặc bị tắt,
- Nếu kiểu liệt kê trong tệp Dart được thay đổi thành các lớp bình thường và ngược lại,
- Nếu mã gốc bị thay đổi,
- Sau khi khai báo kiểu chung được thay đổi.

Flutter Hot Reload chỉ có thể được thực hiện ở chế độ gỡ lỗi. Các chế độ xây dựng khác, đó là: chế độ hồ sơ và chế độ phát hành, không hỗ trợ tính năng Hot Reload.

**Quy mô dự án so với hiệu suất Hot Reload của Flutter**

Các dự án Flutter có quy mô khác nhau, dựa trên số lượng thư viện được bao gồm, kiến trúc ứng dụng, tệp phương tiện hoặc tính năng ứng dụng. Cho đến gần đây, Flutter được coi là giải pháp hoàn hảo cho MVP và PoC. Tuy nhiên, khi các dự án Flutter quy mô lớn như Google Pay, eBay, Nubank, Rive hay 47 triệu người dùng Maya Bank đang có đà phát triển, việc khám phá các khả năng của Flutter cho các ứng dụng phức tạp cũng là điều cần thiết.

Tính năng Hot Reload của Flutter có thể được sử dụng cho cả ứng dụng Proof of Concept (PoC) và các sản phẩm kỹ thuật số cấp doanh nghiệp. Tuy nhiên, câu hỏi vẫn là liệu hiệu suất của nó có thỏa đáng đối với các dự án phức tạp hay không và liệu Flutter dành cho ứng dụng doanh nghiệp có phải là một lựa chọn khả thi hay không.

**Flutter Hot Reload: Giải thích về hiệu suất**

Tính năng Flutter Hot Reload là một công cụ mạnh mẽ, hiệu quả, hữu ích trong khi giải quyết các vấn đề liên quan đến giao diện người dùng trong giai đoạn phát triển. Đã được chứng minh trong hầu hết các trường hợp, hiệu suất Hot Reload diễn ra liền mạch – với một thay đổi giao diện người dùng duy nhất chỉ mất chưa đầy một giây và thời gian tải lại trung bình của 1.000 lớp dao động chỉ trong khoảng 1 giây.

Hơn nữa, một thử nghiệm đã chứng minh rằng Flutter có thể tải lại các dự án quy mô doanh nghiệp lớn với hàng nghìn lớp, trong đó thời gian Hot Reload trung bình là dưới 8 giây. Mặc dù hiệu suất Hot Reload có thể không hoàn toàn thỏa đáng trong các dự án khổng lồ (kịch bản 50 000 lớp), Flutter hoàn toàn có thể đối phó với chúng.

Flutter Hot Reload tăng hiệu quả công việc bằng cách xây dựng lại các widget trong cây widget của dự án, giúp dễ dàng đạt được kết quả mong muốn chỉ trong chớp mắt. Nhờ Hot Reload, các nhà phát triển Flutter có khả năng xử lý kịp thời các thay đổi thiết kế phức tạp (ngay cả những thay đổi ảnh hưởng đến toàn bộ ứng dụng).

Không kém phần quan trọng, Hot Reload chỉ là một yếu tố góp phần vào hiệu suất tổng thể của framework (được cộng đồng Flutter liên tục xác minh và được Flutter Dev cải thiện). Khám phá các công cụ phát triển Flutter hàng đầu là điều cần thiết để tạo ra các ứng dụng di động đa nền tảng chất lượng cao một cách hiệu quả.

### 2.1.6 Flutter Packages và Plugins

Gói(Packages) là một không gian tên chứa một nhóm các loại lớp, giao diện và gói con tương tự nhau. Chúng ta có thể coi các gói tương tự như các thư mục khác nhau trên máy tính của mình, nơi chúng ta có thể giữ phim trong một thư mục, hình ảnh trong thư mục khác, phần mềm trong thư mục khác, v.v. Trong Flutter, Dart tổ chức và chia sẻ một bộ chức năng thông qua một gói. Flutter luôn hỗ trợ các gói chia sẻ, được đóng góp bởi các nhà phát triển khác cho hệ sinh thái Flutter và Dart. Các gói cho phép chúng ta xây dựng ứng dụng mà không cần phải phát triển mọi thứ từ đầu.

**Các loại gói(Packages)**

Theo chức năng, chúng ta có thể phân loại gói thành hai loại:

- Gói Dart
- Gói plugin

**Gói Dart**: Là một gói chung, được viết bằng ngôn ngữ Dart, chẳng hạn như gói đường dẫn. Gói này có thể được sử dụng trong cả môi trường, đó là nền tảng web hoặc di động. Nó cũng chứa một số chức năng cụ thể của Flutter và do đó có sự phụ thuộc vào khuôn khổ Flutter, chẳng hạn như gói fluro.

**Gói plugin**: Là một gói Dart chuyên biệt bao gồm một API được viết bằng code Dart và phụ thuộc vào framework Flutter. Nó có thể được kết hợp với triển khai nền tảng cụ thể cho một nền tảng cơ bản như Android (sử dụng Java hoặc Kotlin) và iOS (sử dụng Objective C hoặc Swift). Ví dụ về gói này là gói plugin bộ chọn hình ảnh và pin.

**Phát triển gói hoặc plugin Flutter**

Việc phát triển một plugin hoặc gói Flutter tương tự như tạo một ứng dụng Dart hoặc gói Dart. Tuy nhiên, nó có một số ngoại lệ có nghĩa là plugin luôn sử dụng API hệ thống cụ thể cho một nền tảng như Android hoặc iOS để có được chức năng cần thiết.

### 2.1.7 Flutter DevTools

Flutter DevTools là một bộ công cụ phát triển mạnh mẽ và toàn diện, được thiết kế để hỗ trợ các lập trình viên Flutter trong việc debug, kiểm tra hiệu suất và kiểm tra UI. Các tính năng của Flutter DevTools:

**Debug**

- **Inspector**: Inspector giúp bạn nắm bắt cấu trúc UI của ứng dụng Flutter. Bạn có thể kiểm tra các widget và xem các thuộc tính của chúng.
- **Timeline**: Timeline giúp bạn theo dõi các sự kiện trong quá trình chạy ứng dụng, cho phép bạn phân tích và tìm hiểu các vấn đề liên quan đến hiệu suất.
- **Debugger**: Debugger cho phép bạn tạm dừng thực thi mã, xem giá trị của các biến, và tiếp tục thực thi mã một cách kiểm soát.

**Kiểm Tra Hiệu Suất**

- **Performance**: Performance giúp bạn phân tích hiệu suất của ứng dụng Flutter. Có thể theo dõi sự tiêu thụ CPU, bộ nhớ, và thời gian chạy các frame.
- **Memory**: Memory giúp bạn theo dõi việc sử dụng bộ nhớ của ứng dụng Flutter. Xem biểu đồ sử dụng bộ nhớ, và phân tích các vấn đề liên quan đến bộ nhớ.
- **Network**: Công cụ này giúp theo dõi tất cả các cuộc gọi mạng mà ứng dụng của bạn thực hiện, cho phép bạn phân tích thời gian trả lời, tốc độ truyền, và các vấn đề tiềm ẩn khác.

**Kiểm Tra UI**

- **Layout Explorer**: Công cụ này cho phép bạn kiểm tra và điều chỉnh layout của các widget trong ứng dụng của bạn. Bạn có thể xem thông tin chi tiết về margin, padding, chiều cao, và chiều rộng của các widget.
- **Flutter Widget Inspector**: Công cụ này giúp bạn nắm bắt cấu trúc widget của ứng dụng và kiểm tra các thuộc tính của chúng. Bạn cũng có thể chọn widget trực tiếp trên màn hình và xem thông tin chi tiết về nó.

## 2.2 Giới thiệu về ngôn ngữ lập trình Dart

### 2.2.1 Khái niệm

Dart là một ngôn ngữ lập trình hướng đối tượng mã nguồn mở, có mục đích chung với cú pháp kiểu C do Google phát triển vào năm 2011 . Mục đích của lập trình Dart là tạo giao diện người dùng frontend cho web và ứng dụng dành cho thiết bị di động. Nó đang được phát triển tích cực, được biên dịch sang mã máy gốc để xây dựng ứng dụng di động, lấy cảm hứng từ các ngôn ngữ lập trình khác như Java, JavaScript, C # và Typed mạnh. Vì Dart là một ngôn ngữ biên dịch không thể thực thi code của mình trực tiếp; thay vào đó, trình biên dịch phân tích cú pháp nó và chuyển nó thành code máy.

Nó hỗ trợ hầu hết các khái niệm chung của ngôn ngữ lập trình như lớp, giao diện, hàm, không giống như các ngôn ngữ lập trình khác. Ngôn ngữ Dart không hỗ trợ mảng trực tiếp. Nó hỗ trợ tập hợp, được sử dụng để sao chép cấu trúc dữ liệu như mảng, generic và kiểu tùy chọn.

### 2.2.2 Dart được sử dụng cho mục đích gì?

Như đã giới thiệu ở trên, ngôn ngữ Dart được sử dụng để phát triển, xây dựng các ứng dụng web, di động với hiệu suất cao. Khắc phục những hạn chế của JavaScript như yêu cầu cơ sở mã nguồn mở, không có khả năng mở rộng,… Một số ứng dụng phổ biến và quan trọng của Dart có thể kể đến như:

- Đóng vai trò là ngôn ngữ lập trình cơ bản cho khung Flutter, và được sử dụng để xây dựng các ứng dụng di động có thể mở rộng.
- Do là ngôn ngữ có mục đích chung, nó được sử dụng để xây dựng, lập trình các ứng dụng di động gốc cho iOS, Android hoặc cho máy tính để bàn, máy chủ.

### 2.2.3 Tại sao nên học ngôn ngữ Dart?

Mặc dù so với những loại khác, Dart có tuổi đời "trẻ" hơn, nhưng trong những năm gần đây, nó đã được sử dụng khá phổ biến. Do đó, nếu là newbie đang tìm hiểu về lĩnh vực lập trình, bạn có thể tham khảo một số lý do nên học loại ngôn ngữ này như sau:

- Hiệu quả nhanh: Ngôn ngữ Dart sử dụng trình biên dịch mã nguồn để thực hiện sắp xếp mã nhanh hơn. Do đó, nó sẽ có giao diện người dùng hiệu quả hơn so với những loại ngôn ngữ lập trình khác.
- Dễ sử dụng, có thể làm việc trực tiếp từ trình duyệt, không cần cài đặt thêm bất kỳ phần mềm nào.
- Khung Flutter của Dart được sử dụng để tạo các ứng dụng đa nền tảng và đây là 1 trong 4 khung hàng đầu về phát triển ứng dụng di động hiện nay. Ví dụ như Google Pay, ứng dụng của BMW, Toyota,…
- Tính linh hoạt, khả năng thích ứng cao với JavaScript.
- Nếu đã có kiến thức JavaScript thì việc học Dart tương đối dễ dàng.

### 2.2.4 Ưu – nhược điểm của Dart là gì?

**Ưu điểm của ngôn ngữ Dart**

Dart có những ưu điểm nổi bật như sau:

- Là ngôn ngữ mở rộng, linh hoạt, tạo điều kiện tích cực cho việc biên dịch trở nên nhanh chóng hơn.
- Có tính ổn định tốt, được sử dụng để xây dựng các ứng dụng về thời gian thực với hiệu suất cao.

- Hỗ trợ cả biên dịch Vừa đúng lúc (JIT) và biên dịch Trước thời hạn (AOT).
- Có thể thích ứng nhanh chóng với các quy trình công việc có sự thay đổi.

**Nhược điểm của ngôn ngữ Dart**

Một số hạn chế khi sử dụng Dart mà bạn nên biết như:

- Là ngôn ngữ lập trình mới nên hiện tại cộng đồng người sử dụng, hỗ trợ có quy mô nhỏ, chưa có quá nhiều tài nguyên phục vụ cho công việc học tập.
- Chỉ bao gồm một lớp đối tượng duy nhất, không hỗ trợ quá trình lặp lại mã.
- Không hỗ trợ đổi tên hàm mà không thực hiện viết câu lệnh gán mới.

### 2.2.5 Các tính năng của ngôn ngữ Dart

Dưới đây là một số tính năng của ngôn ngữ này:

- Hướng đối tượng: Dart sử dụng dữ liệu dưới dạng đối tượng thay vì coi dữ liệu là hàm hoặc logic và hỗ trợ cả các khác niệm lập trình hướng đối tượng cơ bản.
- Không đồng bộ: Dart không có tính đồng bộ nhưng cho phép đồng thời nâng cao. Hiểu đơn giản là bạn có thể chạy đồng thời nhiều tác vụ với Dart bằng cách sử dụng thể độc lập (phân lập).
- Các thư viện tích hợp: Dart bao gồm các thư viện tích hợp mở rộng như Input – Output (IO), Software Development Kit (SDK),… Bạn có thể tìm các đoạn mã code viết sẵn trong những thư viện này và tối ưu theo mục đích của mình.
- Hỗ trợ đa nền tảng: Dart có thể hoạt động trên nhiều hệ điều hành khác nhau Windows, Linux, macOS cũng như nhiều hệ điều hành khác bởi tính năng Máy ảo Dart.

### 2.2.6 Lập trình hướng đối tượng với Dart

Dart là một ngôn ngữ lập trình hướng đối tượng, có nghĩa là mọi giá trị trong Dart đều là một đối tượng. Một số cũng là một đối tượng trong ngôn ngữ Dart. Lập trình Dart hỗ trợ khái niệm về các tính năng OOP như đối tượng, lớp, giao diện,V.V.

**Các đặc tính trong lập trình hướng đối tượng của ngôn ngữ Dart.**

- **Tính trừu tượng (abstraction)**

Tính trừu tượng thể hiện ở việc lựa chọn các thuộc tính và hành vi của đối tượng mà không phải liệt kê hết tất cả các thuộc tính và hành vi của đối tượng đó.Ví dụ: Để mô

tả một người có rất nhiều thuộc tính và hành vi. Nhưng chúng ta chỉ sử dụng các thuộc tính như: tên, năm sinh, quê quán và thuộc tính như: đi, chạy mà không cần liệt kê hết tất cả các thuộc tính và hành vi khác như : tình trạng hôn nhân, lái xe, đá, đấm...

- **Tính đóng gói (Encapsulation)**

Tính đóng gói thể hiện sự che dấu trong đối tượng với mục đích bảo vệ dữ liệu và tăng khả năng mở rộng. Vì vậy khi triển khai một đối tượng, các thuộc tỉnh nên dùng tính năng private.

- **Tính kế thừa**

Trong một phần mềm hay chương trình, được cấu tạo bởi nhiều lớp khác nhau củng các thành phần khác. Mối quan hệ giữa các lớp, có mối quan hệ kế thừa, gồm lớp cha ( super class )và các lớp con(sub class), Các lớp con đó lại có thể là lớp cha của các lớp khác.

Mục đích của kế thừa là tái sử dụng. Lớp con có thể sở hữu các thuộc tỉnh và phương thức public của lớp cha nhưng không được sở hữu các thuộc tính hay phương thức private và các hàm constructor. Biểu diễn kế thừa trong Dart cũng tương tự như trong Java : dùng extends.

Trong ngôn ngữ Dart, không dùng từ khoá interface hay không dùng phương thức có từ khoá abstract ở phía trước

- **Tính đa hình**

Tính đa hình trong ngôn ngữ Dart cũng có ý nghĩa giống như trong các ngôn ngữ khác. Cùng biểu diễn một hành vi nhưng từng lớp có cách biểu diễn khác nhau.

## 2.3 Firebase – lưu trữ online

### 2.3.1 Firebase là gì?

Firebase là một nền tảng sở hữu bởi google giúp chúng ta phát triển các ứng dụng di động và web. Họ cung cấp rất nhiều công cụ và dịch vụ tiện ích để phát triển ứng dụng nên một ứng dụng chất lượng. Điều đó rút ngắn thời gian phát triển và giúp ứng dụng sớm ra mắt với người dùng.

Firebase cung cấp cho người dùng các dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây với hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính của

firebase là giúp người dùng lập trình ứng dụng, phần mềm trên các nền tảng web, di động bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu.

Với firebase, bạn có thể tạo ra những ứng dụng real-time như app chat, cùng nhiều tính năng như xác thực người dùng, Cloud Messaging,... Bạn có thể dùng firebase giống như phần backend của app.

Các dịch vụ của firebase hoàn toàn miễn phí, tuy nhiên bạn cần phải trả thêm tiền nếu muốn nâng cấp lên. Điều này bạn nên cân nhắc nếu muốn xây dựng một ứng dụng lớn sử dụng phần backend là firebase, vì cái giá khi muốn nâng cấp còn khá đắt đỏ so với việc xây dựng backend truyền thống.

### 2.3.2 Lịch sử phát triển của Firebase

Firebase phát triển từ Envolve, một công ty khởi nghiệp trước đó do James Tamplin và Andrew Lee thành lập vào năm 2011. Họ thành lập Firebase như một công ty vào tháng 9 - 2011. Đến tháng 4 năm 2012 Firebase đã lần đầu tiên được ra mắt công chúng.

Sản phẩm đầu tiên của Firebase là Cơ sở dữ liệu thời gian thực (Firebase realtime database), một API đồng bộ hóa dữ liệu ứng dụng trên các thiết bị iOS, Android và Web, đồng thời lưu trữ trên đám mây của Firebase. Sản phẩm hỗ trợ các nhà phát triển phần mềm trong việc xây dựng các ứng dụng cộng tác, theo thời gian thực.

Vào tháng 10 năm 2014, Firebase đã được Google mua lại. Từ đó đến nay, Firebase đã ra mắt thêm nhiều tính năng mới và được nhiều nhà phát triển ưa thích sử dụng trong các dự án của mình.


*Hình 2. 5 Google Firebase*

*(Nguồn: Internet)*

### 2.3.3 Những tính năng chính của Firebase

**Firebase realtime database**

Firebase realtime database là một cơ sở dữ liệu thời gian thực, NoSQL được lưu trữ đám mây cho phép bạn lưu trữ và đồng bộ dữ liệu. Dữ liệu được lưu trữ dưới dạng cây Json, và được đồng bộ theo thời gian thực đối với mọi kết nối.

*Hình 2. 6 Realtime Database*

*(Nguồn: Internet)*

Khi xây dựng những ứng dụng đa nền tảng như Android, IOS và Web App, tất cả các client của bạn sẽ kết nối trên cùng một cơ sở dữ liệu Firebase và tự động cập nhật dữ liệu mới nhất khi có sự thay đổi Cả một cơ sở dữ liệu là một cây json lớn, với độ trễ thấp, Firebase realtime database cho phép bạn xây dựng các ứng dụng cần độ realtime như app chat, hay game online…

Firebase có các tính năng bảo mật hàng đầu Tất cả dữ liệu được truyền qua một kết nối an toàn SSL, việc truy vấn cơ sở dữ liệu truy vấn và việc xác nhận thông tin được điều khiển theo một số các quy tắc security rules language. Các logic bảo mật dữ liệu của bạn được tập trung ở một nơi để dễ dàng cho việc sửa đổi, cập nhật và kiểm thử.

Làm việc offline Ứng dụng của bạn sẽ duy trì tương tác mặc dù có các vấn đề về kết nối internet như mạng chậm chờn, mất mạng hay mạng yếu. Trước khi bất kỳ dữ liệu được ghi đến firebase thì tất  cả dữ liệu lập tức sẽ được ghi tạm vào một cơ sử dữ liệu ở local.

Sau khi có kết nối internet lại, client sẽ nhận bất kỳ thay đổi mà nó thiếu/ bỏ lỡ và đồng bộ hoá nó với cơ sở dữ liệu tại firebase. -Firebase realtime database cho phép nhiều kết nối đồng thời mà bạn không cần tính toán đến vấn đề nâng cấp máy chủ. Tuy nhiên bạn vẫn cần phải trả phí để có thể nâng cấp firebase khi quy mô ứng dụng đủ lớn.

**Firebase Authentication**

Firebase Authentication là chức năng xác thực người dùng. Hiểu một cách đơn giản, app của bạn cần phải đăng nhập/ đăng ký tài khoản để sử dụng, Firebase cung cấp cho chúng ta chức năng xác thực người dùng bằng email, số điện thoại, hay tài khoản Facebook, Google,...

Việc xác thực người dùng là một chức năng vô cùng quan trọng trong việc phát triển ứng dụng. Tuy nhiên, khi bạn muốn xác thực với nhiều phương thức khác nhau như email, số điện thoại, google, facebook sẽ tốn nhiều thời gian và công sức. Firebase Authentication giúp thực hiện việc đó một cách dễ dàng, giúp người dùng nhanh chóng tiếp cận sản phẩm hơn.

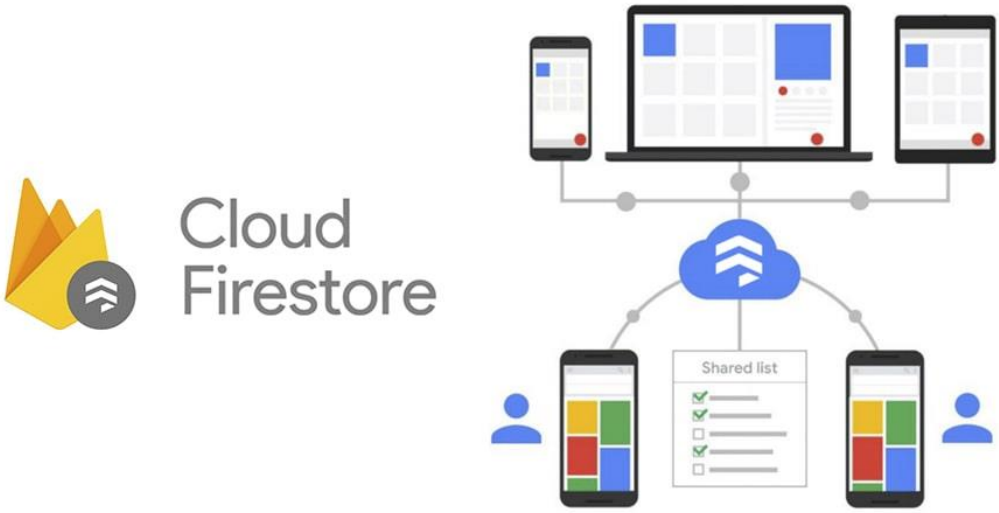

*Hình 2. 7 Authentication*

*(Nguồn: Internet)*

Vì thế, nó là một chức năng vô cùng hữu ích của firebase. Nếu bạn muốn xây dựng sản phẩm một cách nhanh chóng, hay chỉ đơn giản là làm bài tập, đồ án thì việc tích hợp Firebase Authentication và Firebase Realtime Database vào ứng dụng sẽ giúp bạn giảm rất nhiều thời gian so với các cách khác.

**Firebase Cloud Storage**

Firebase Cloud Storage là một không gian lưu trữ dữ liệu, nó giống như một chiếc ổ cứng. Bạn có thể upload và download các loại file bạn muốn. Đó có thể là một file ảnh, hay file văn bản, .zip, …




*Hình 2. 8 Cloud Firestore*

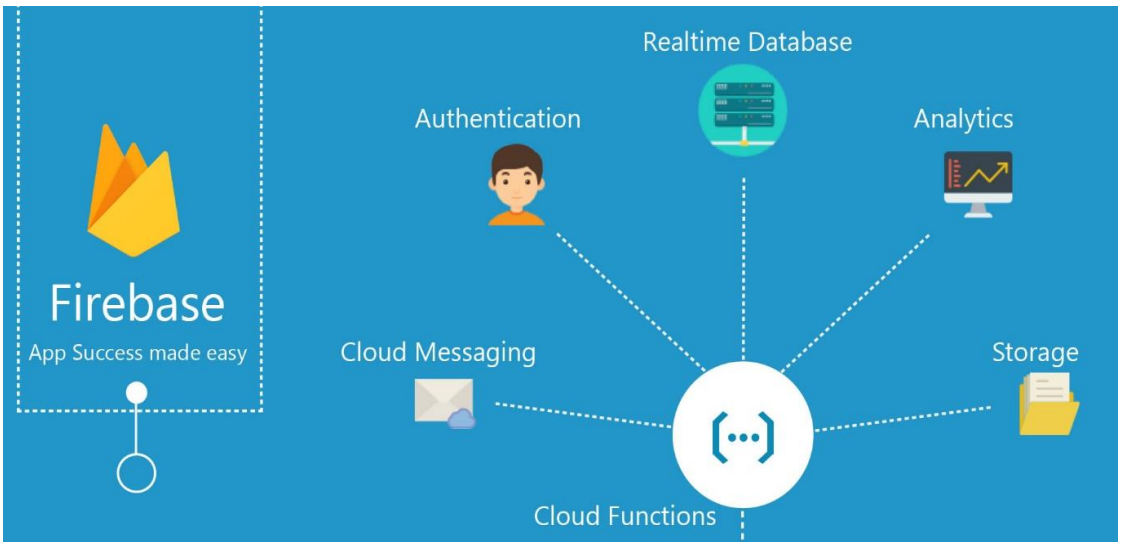
*(Nguồn: Internet)*

Phân biệt Firebase cloud storage với Firebase realtime database. Câu trả lời rất đơn giản, Firebase là một cơ sở dữ liệu- nơi bạn có thể lưu trữ các thông tin về tài khoản người dùng, hay các thông tin về một mặt hàng nếu bạn xây dựng một app bán hàng.

Còn với Firebase cloud storage, chúng là nơi lưu trữ những file, đó có thể là những hình ảnh về một mặt hàng chẳng hạn. Bạn có thể lưu trữ link tới file hình ảnh trong database, còn file ảnh đặt trong cloud storage. Vậy là client có thể dễ dàng truy vấn và sử dụng.

**Firebase Cloud Function**

Cloud Functions Firebase cho phép chúng ta viết những câu truy vấn database lưu trữ trên cloud. Code của bạn được lưu trữ trong cloud của Google và chạy trong một môi trường bảo mật, được quản lý. Bạn không cần quan tâm đến vấn đề mở rộng các máy chủ. Với firebase, khi bạn muốn lấy dữ liệu bạn cần phải viết các câu truy vấn trực tiếp từ client.




*Hình 2. 9 Cloud Functions*

*(Nguồn: Internet)*

Điều này có thể vô tình để lộ một số thông tin nhạy cảm. Để khắc phục vấn đề đó, Cloud Function đã ra đời. Nhiều lúc, các developers muốn kiểm soát logic trên server để tránh giả mạo phía client. Ngoài ra, đôi khi không muốn mã của mình khi bị decode sẽ gây ra các vấn đề về bảo mật. Cloud Functions được tách biệt hoàn toàn với client, vì vậy bạn có thể yên tâm nó bảo mật và luôn thực hiện chính xác những gì bạn muốn.

**Firebase Analytics**

Firebase Analytics là tính năng giúp bạn phân tích hành vi của người sử dụng trên ứng dụng của bạn. Cuối cùng nó sẽ đưa ra lời khuyên về lộ trình xây dựng ứng dụng. Để làm việc này bạn cần cài đặt SDK (Software Development Kit, cụ thể hơn là FirebaseAnalytics.unitypackage), chức năng phân tích sẽ trở nên khả dụng.

Khi đó, không chỉ xem được hành vi của người dùng mà còn có thể biết được thông tin về như hiệu quả quảng cáo, tình trạng trả phí, v.v. Với tính năng này, bạn có thể biết người dùng của bạn thường xuyên truy cập tính năng nào, từ đó bạn có thể đưa ra chiến lược phát triển sản phẩm của mình.

# CHƯƠNG III. PHÂN TÍCH VÀ THIẾT KẾ ỨNG DỤNG

## 3.1 Mô tả chức năng ứng dụng

### 3.1.1 Chức năng dành cho người dùng.

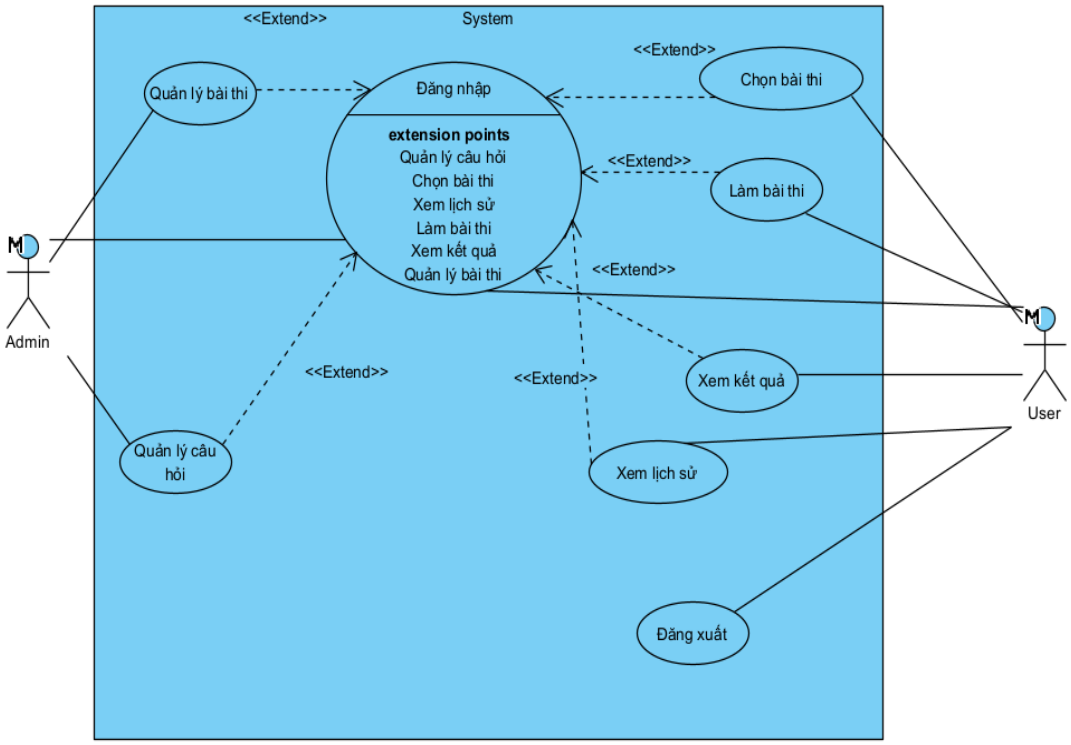
- Chức năng đăng nhập để lưu các trạng thái mà khách hàng đã thao tác.
- Chọn, làm bài thi:
  + Hiển thị danh sách đề thi theo các đề tài hoặc độ khó khác nhau.
  + Chọn 1 đề thi để tiến hành làm bài
  + Cho phép người dùng chọn và trả lời các câu hỏi từ danh sách. Cần có khả năng chọn câu trả lời đúng từ các phương án cho trước.
- Xem và lưu kết quả.
  + Sau khi hoàn thành bài thi, hiển thị kết quả với số điểm và chi tiết câu trả lời đúng/sai của người dùng.
  + Lưu kết quả của người dùng vào cơ sở dữ liệu để có thể xem lại sau này hoặc để so sánh với kết quả của người dùng khác.
- Thống kê và lịch sử.
  + Cung cấp thống kê về số lượng bài thi đã hoàn thành, số câu trả lời đúng/sai, điểm số,…

### 3.1.2 Các chức năng dành cho Admin.

- Chức năng đăng nhập để lưu các trạng thái
- Quản lý bài thi.
  + Tùy chỉnh thêm bài thi.
- Quản lý câu hỏi.
  + Tùy chỉnh thêm câu hỏi, đáp án.

## 3.2 Thiết kế UML.

### 3.2.1 Biểu đồ usecase tổng quát



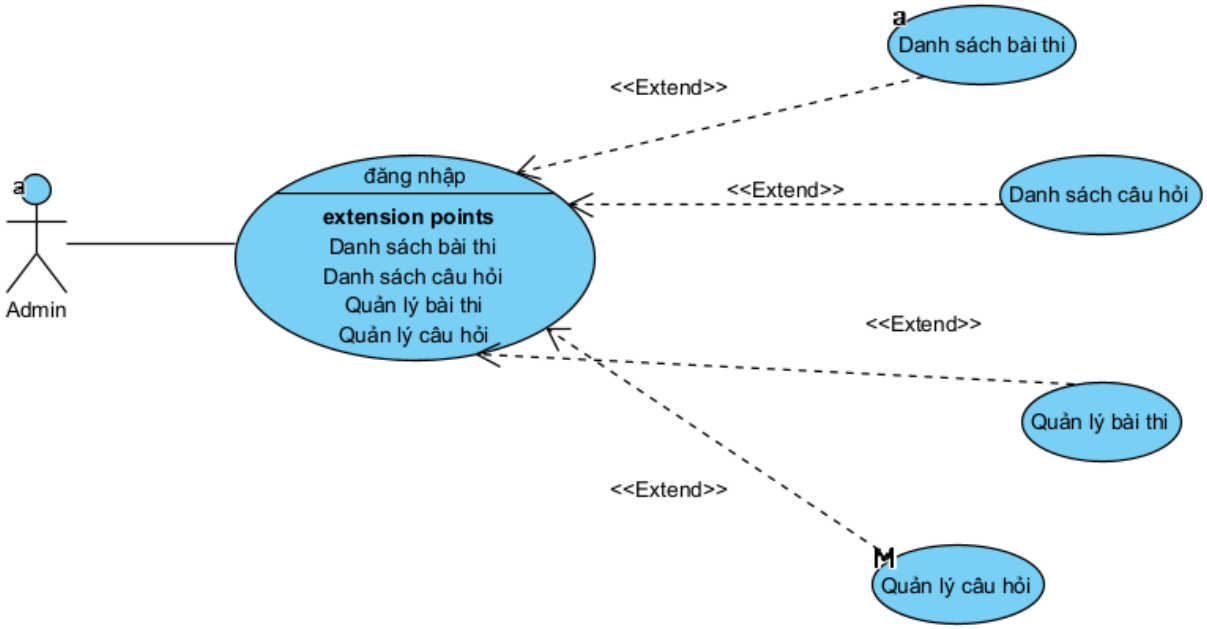

*Hình 3. 1Biểu đồ usecase tổng quát*

| STT | Tên Use case | Ý nghĩa/Ghi Chú |
|---|---|---|
| 1 | Đăng nhập | Use case này giúp người dùng hoặc Admin sử dụng các chức năng của hệ thống cần đến quyền truy cập. |
| 2 | Quản lý bài thi | Use case này mô tả chức năng cập nhật danh sách bài thi của Admin. |
| 3 | Quản lý câu hỏi | Use case này mô tả chức năng cập nhật nội dung và đáp án câu hỏi vào hệ thống của Admin. |
| 4 | Chọn bài thi | Use case này mô tả chức năng hiển thị danh sách đề tài câu hỏi và cho phép chọn 1 đề tài để bắt đầu thi |
| 5 | Làm bài thi | Use case này mô tả chức năng hiển thị nội dung câu hỏi, có thể chọn đáp án và chuyển sang câu hỏi trước hoặc sau, sau khi hoàn thành tất cả câu hỏi có thể nộp bài thi |
| 6 | Xem kết quả thi | Use case này mô tả chức năng xem kết quả sau khi làm xong bài thi bao gồm điểm, các câu hỏi đúng và sai |
| 7 | Xem lịch sử thi | Usecase này mô tả chức năng xem lại kết quả các bài thi đã làm và được lưu trong cơ sở dữ liệu |

### 3.2.2 Chức năng đăng nhập của Admin

<table>
<tr><td colspan="2">UC001</td><td>Đăng nhập</td><td>Độ phức tạp: thấp</td></tr>
<tr><td colspan="2">Mô tả</td><td colspan="2">Chức năng này cho phép người dùng đăng nhập vào hệ thống để sử dụng các chức năng trên hệ thống theo từng quyền hạn cho phép.</td></tr>
<tr><td colspan="2">Tác nhân</td><td colspan="2">Admin</td></tr>
<tr><td colspan="2">Tiền điều kiện</td><td colspan="2">Người dùng sử dụng tài khoản của mình để đăng nhập vào hệ thống.</td></tr>
<tr><td rowspan="2">Hậu điều kiện</td><td>Thành công</td><td colspan="2">Sau khi đăng nhập thành công vào hệ thống, giao diện các chức năng của hệ thống sẽ được hiển thị theo quyền của tài khoản người dùng vừa xác thực.</td></tr>
<tr><td>Lỗi</td><td colspan="2">Hệ thống không xác thực được tài khoản, trạng thái không thay đổi.</td></tr>
<tr><td colspan="4">ĐẶC TẢ CHỨC NĂNG</td></tr>
<tr><td colspan="4">Chức năng Đăng nhập</td></tr>
<tr><td colspan="4">Luồng sự kiện chính/Kịch bản chính</td></tr>
<tr><td colspan="4">Khi hệ thống được kích hoạt, chức năng đăng nhập cho phép xác thực người dùng thông qua tài khoản được cấp phép:<br>• Người dùng kích hoạt hệ thống, hệ thống sẽ hiển thị form đăng nhập yêu cầu nhập vào tên đăng nhập và mật khẩu.<br>• Hệ thống xác nhận tên đăng nhập và mật khẩu, truy vấn đến cơ sở dữ liệu để kiểm tra thông tin tài khoản.<br>• Sau khi thông tin tài khoản được xác thực, hệ thống sẽ đóng form đăng nhập và hiển thị giao diện với danh sách các chức năng của hệ thống tùy theo quyền truy cập của tài khoản.</td></tr>
<tr><td colspan="4">Luồng sự kiện phát sinh/Kịch bản phát sinh</td></tr>
<tr><td colspan="4">Nếu tên đăng nhập và mật khẩu được nhập vào không chính xác với tài khoản người dùng được đăng ký trên hệ thống hoặc tài khoản đã bị khóa trên hệ thống thì:<br>– Hệ thống sẽ hiện thị thông báo không tìm thấy tài khoản người dùng tương ứng với tên đăng nhập và mật khẩu, sau đó hệ thống sẽ hiển thị lại form đăng nhập.<br>– Người dùng nhập lại tên đăng nhập và mật khẩu vào form, hệ thống sẽ thực hiện lại quá trình xác thực.</td></tr>
</table>

**Sở đồ usecase**



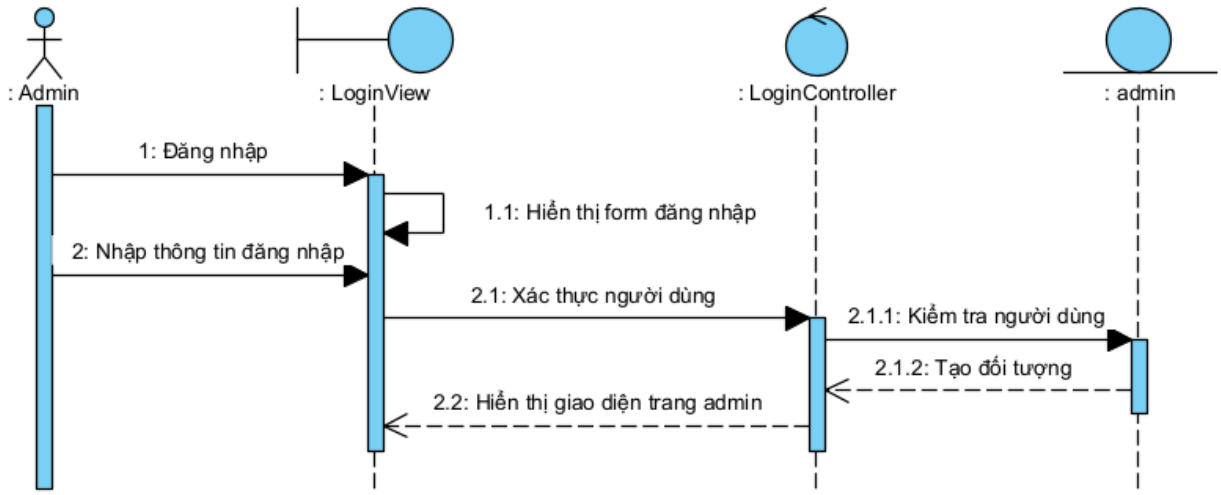

*Hinh 3. 2 Biểu đồ usecase đăng nhập*

**Giao diện minh họa**


*Hình 3. 3 Giao diện đăng nhập Admin*

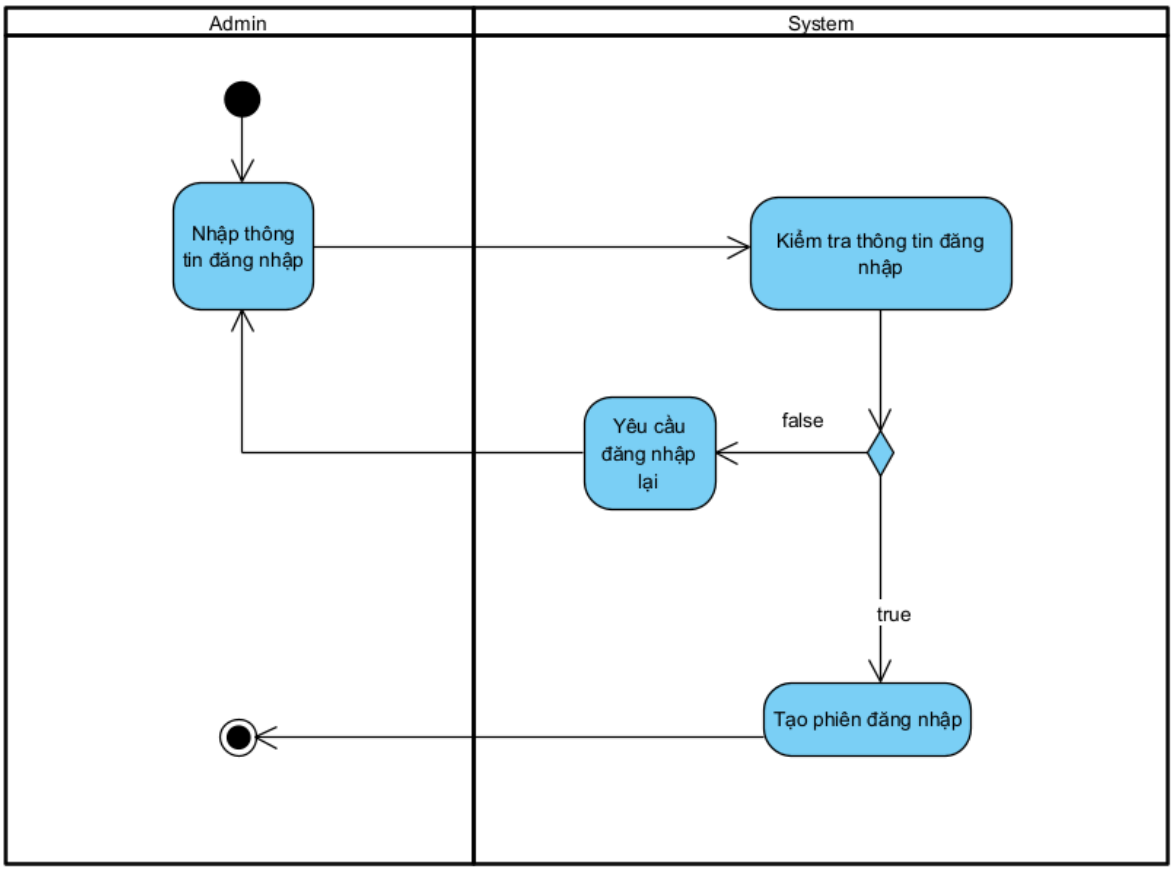

*Hình 3. 4 Giao diện trang Admin*

## Sơ đồ Trình tự (Sequence diagram)




*Hình 3. 5 Biểu đồ tuần tự đăng nhập Admin*

## Sơ đồ hoạt động (Activity diagram)




*Hình 3. 6 Biểu đồ hoạt động đăng nhập Admin*

### 3.2.3 Chức năng thêm bài thi

| | | | |
|---|---|---|---|
| **UC002** | | Thêm bài thi | Độ phức tạp: thấp |
| **Mô tả** | | Người dùng sẽ sử dụng chức năng này để quản lý danh sách bài thi trên hệ thống. Chức năng cho phép thêm mới bài thi. | |
| **Tác nhân** | | Admin | |
| **Tiền điều kiện** | | Người dùng sử dụng tài khoản của mình để đăng nhập vào hệ thống. | |
| **Hậu điều kiện** | **Thành công** | Người dùng có thể xem, thêm thông tin bài thi trên hệ thống. | |
| | **Lỗi** | Trạng thái hệ thống sẽ không thay đổi. | |
| **ĐẶC TẢ CHỨC NĂNG** | | | |
| **Luồng sự kiện chính/Kịch bản chính** | | | |

USECASE này hoạt động khi người dùng hệ thống muốn thêm thông tin bài thi trong hệ thống.

- Hệ thống sẽ lấy ra và hiển thị danh sách các bài thi mà hệ thống lưu trữ.
- Hệ thống cho phép người dùng có thể chọn các chức năng thêm thông tin bài thi.

**1. Thêm mới bài thi**

- Hệ thống sẽ yêu cầu người dùng nhập vào thông tin bài thi. Bao gồm:
  - + Mã bài thi;
  - + Tên bài thi;
  - + Mô tả;
  - + Ảnh nền;
  - + Số câu hỏi;
  - + Thời gian;
- Hệ thống yêu cầu người dùng xác nhận "**Thêm mới**" hoặc "**Huỷ bỏ**".
- Người dùng chọn "**Thêm mới**", bài thi sẽ được thêm vào hệ thống.

**Luồng sự kiện phát sinh/Kịch bản phát sinh**

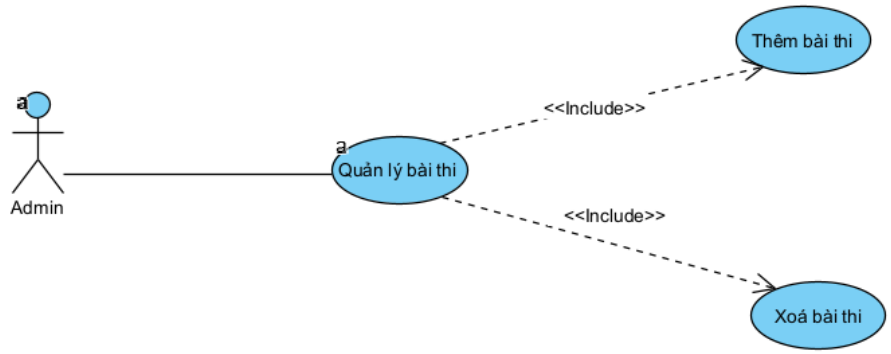
❖ **Lỗi không tìm thấy thông tin nhập**

– Nếu trong khi tìm kiếm, hệ thống không tìm được thông tin liên quan đến thông tin vừa nhập. Hệ thống sẽ hiển thị báo lỗi "**Không có bản ghi nào được tìm thấy**".

❖ **Hủy thêmthông tin bài thi**

– Nếu trong luồng con thêm thông tin giảng viên, người dùng quyết định không thêm hoặc sửa thông tin bài thi, thì hệ thống sẽ tự động hủy tác vụ thêm thông tin bài thi và quay trở về danh sách bài thi.

**Sơ đồ usecase**

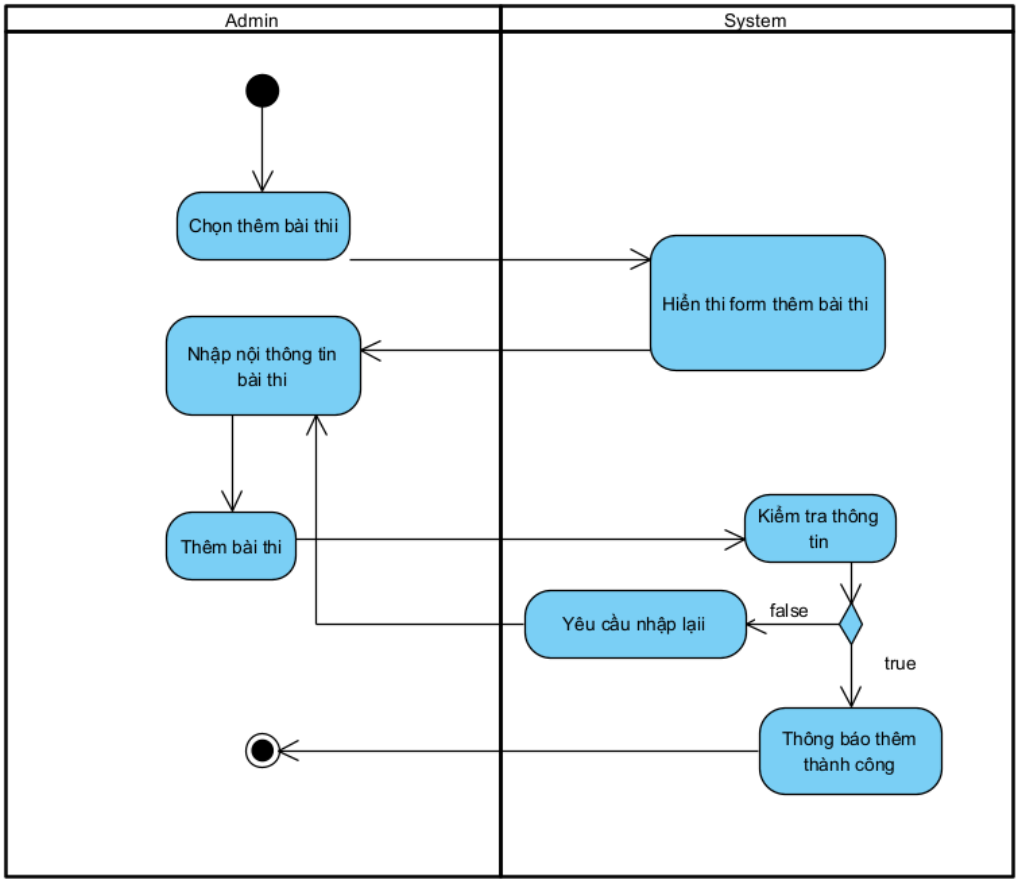



*Hình 3. 7 Biểu đồ usecase quản lý bài thi*

**Giao diện minh họa**


*Hình 3. 8 Giao diện form thêm bài thi*

## Sơ đồ hoạt động (Activity diagram)




*Hình 3. 9 Biểu đồ hoạt động thêm bài thi*

### 3.2.4 Chức năng thêm câu hỏi

<table>
<tr><td colspan="2">UC003</td><td>Thêm câu hỏi</td><td>Độ phức tạp: thấp</td></tr>
<tr><td colspan="2">Mô tả</td><td colspan="2">Người dùng sẽ sử dụng chức năng này để quản lý danh sách câu hỏi trên hệ thống. Chức năng cho phép thêm mới câu hỏi.</td></tr>
<tr><td colspan="2">Tác nhân</td><td colspan="2">Admin</td></tr>
<tr><td colspan="2">Tiền điều kiện</td><td colspan="2">Người dùng sử dụng tài khoản của mình để đăng nhập vào hệ thống.</td></tr>
<tr><td rowspan="2">Hậu điều kiện</td><td>Thành công</td><td colspan="2">Người dùng có thể xem, thêm nội dung câu hỏi trên hệ thống.</td></tr>
<tr><td>Lỗi</td><td colspan="2">Trạng thái hệ thống sẽ không thay đổi.</td></tr>
<tr><td colspan="4">ĐẶC TẢ CHỨC NĂNG</td></tr>
<tr><td colspan="4">Luồng sự kiện chính/Kịch bản chính</td></tr>
<tr><td colspan="4">USECASE này hoạt động khi người dùng hệ thống muốn thêm nội dung câu hỏi trong hệ thống.<br>– Hệ thống sẽ lấy ra và hiển thị danh sách các câu hỏi mà hệ thống lưu trữ.<br>– Hệ thống cho phép người dùng có thể chọn các chức năng thêm nội dung câu hỏi.<br>2. Thêm mới câu hỏi<br>– Hệ thống sẽ yêu cầu người dùng nhập vào nội dung câu hỏi và đáp án. Bao gồm:<br>+ Mã câu hỏi;<br>+ Nội dung câu hỏi;<br>+ Đáp án đúng;<br>+ Tên đáp án;<br>+ Nội dung đáp án;<br>– Hệ thống yêu cầu người dùng xác nhận “Thêm mới” hoặc “Huỷ bỏ”.<br>– Người dùng chọn “Thêm mới”, câu hỏi sẽ được thêm vào hệ thống.</td></tr>
</table>

**Luồng sự kiện phát sinh/Kịch bản phát sinh**

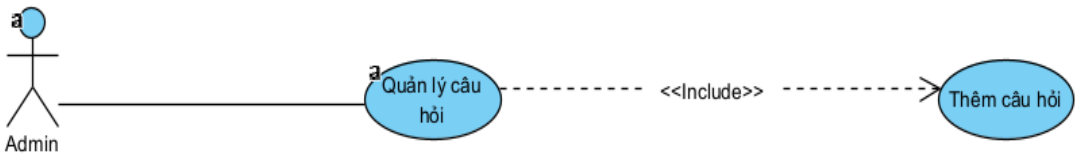
❖ **Lỗi không tìm thấy thông tin nhập**

– Nếu trong khi tìm kiếm, hệ thống không tìm được thông tin liên quan đến thông tin vừa nhập. Hệ thống sẽ hiển thị báo lỗi "**Không có bản ghi nào được tìm thấy**".

❖ **Hủy thêm câu hỏi**

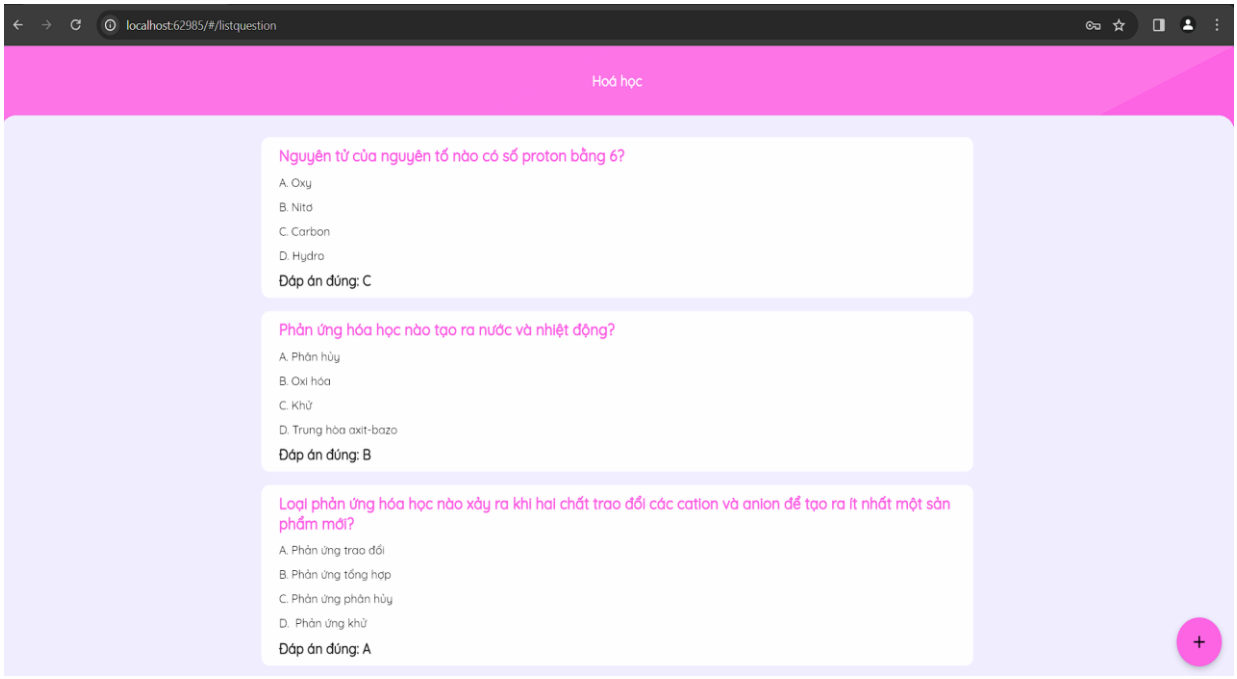
– Nếu trong luồng con thêm nội dung câu hỏi, người dùng quyết định không thêm thông tin câu hỏi, thì hệ thống sẽ tự động hủy tác vụ thêm câu hỏi và quay trở về danh sách câu hỏi.

**Sơ đồ usecase**

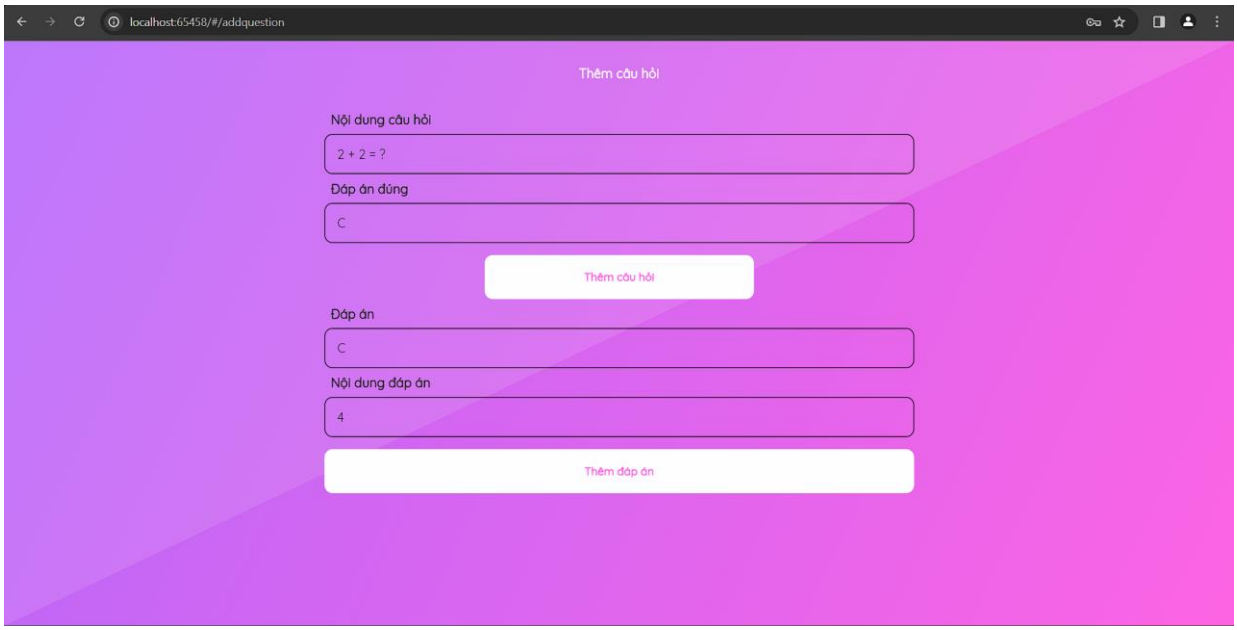



*Hình 3. 10 Biểu đồ usecase quản lý câu hỏi*

**Giao diện minh họa**

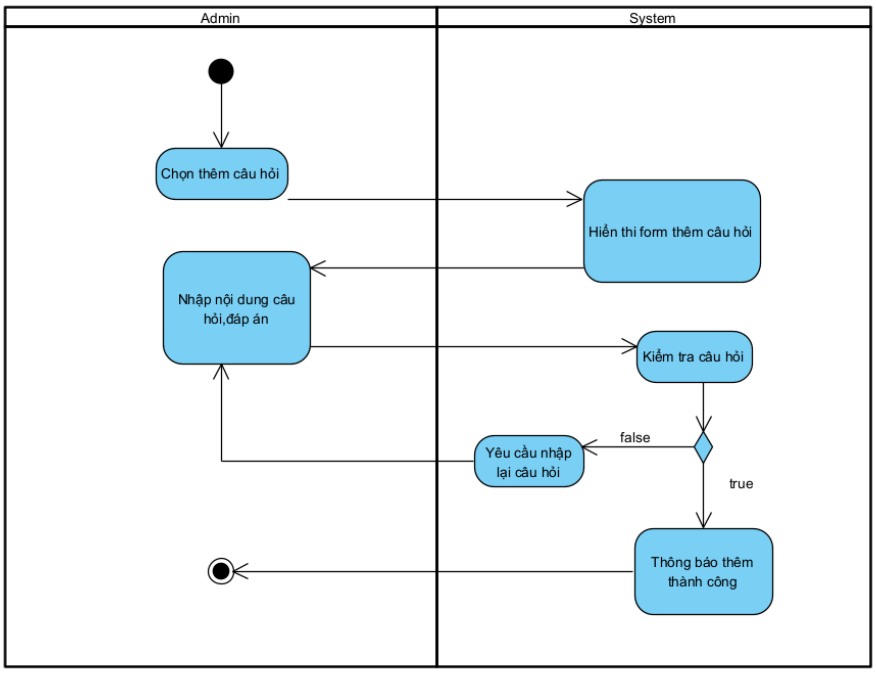



*Hình 3. 11 Giao diện danh sách câu hỏi*




*Hình 3. 12 Giao diện form thêm câu hỏi*

| Sơ đồ hoạt động (Activity diagram) |
|---|




*Hình 3. 13 Biểu đồ hoạt động thêm câu hỏi*

### 3.2.5 Chức năng đăng nhập của người dùng

| | | | |
|---|---|---|---|
| **UC004** | | Đăng nhập | Độ phức tạp: thấp |
| **Mô tả** | | Chức năng này cho phép người dùng đăng nhập vào hệ thống để sử dụng các chức năng trên hệ thống theo từng quyền hạn cho phép. | |
| **Tác nhân** | | User | |
| **Tiền điều kiện** | | Người dùng sử dụng tài khoản của mình để đăng nhập vào hệ thống. | |
| **Hậu điều kiện** | **Thành công** | Sau khi đăng nhập thành công vào hệ thống, giao diện các chức năng của hệ thống sẽ được hiển thị theo quyền của tài khoản người dùng vừa xác thực. | |
| | **Lỗi** | Hệ thống không xác thực được tài khoản, trạng thái không thay đổi. | |
| **ĐẶC TẢ CHỨC NĂNG** | | | |
| **Chức năng Đăng nhập của người dùng** | | | |
| **Luồng sự kiện chính/Kịch bản chính** | | | |

Khi hệ thống được kích hoạt, chức năng đăng nhập cho phép xác thực người dùng thông qua tài khoản được cấp phép:

- Người dùng kích hoạt hệ thống, hệ thống sẽ hiển thị form đăng nhập yêu cầu nhập vào tên đăng nhập và mật khẩu.
- Hệ thống xác nhận tên đăng nhập và mật khẩu, truy vấn đến cơ sở dữ liệu để kiểm tra thông tin tài khoản.
- Sau khi thông tin tài khoản được xác thực, hệ thống sẽ đóng form đăng nhập và hiển thị giao diện với danh sách các chức năng của hệ thống tùy theo quyền truy cập của tài khoản.

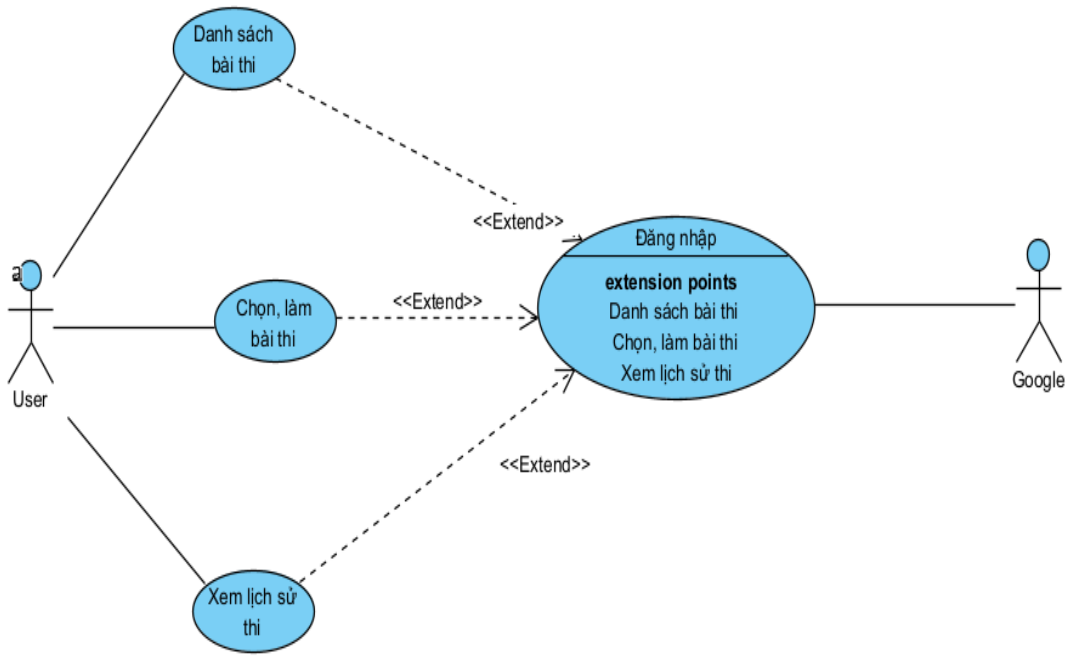
**Luồng sự kiện phát sinh/Kịch bản phát sinh**

Nếu tên đăng nhập và mật khẩu được nhập vào không chính xác với tài khoản người dùng được đăng ký trên hệ thống hoặc tài khoản đã bị khóa trên hệ thống thì:

– Hệ thống sẽ hiện thị thông báo không tìm thấy tài khoản người dùng tương ứng với tên đăng nhập và mật khẩu, sau đó hệ thống sẽ hiển thị lại form đăng nhập.

– Người dùng nhập lại tên đăng nhập và mật khẩu vào form, hệ thống sẽ thực hiện lại quá trình xác thực.

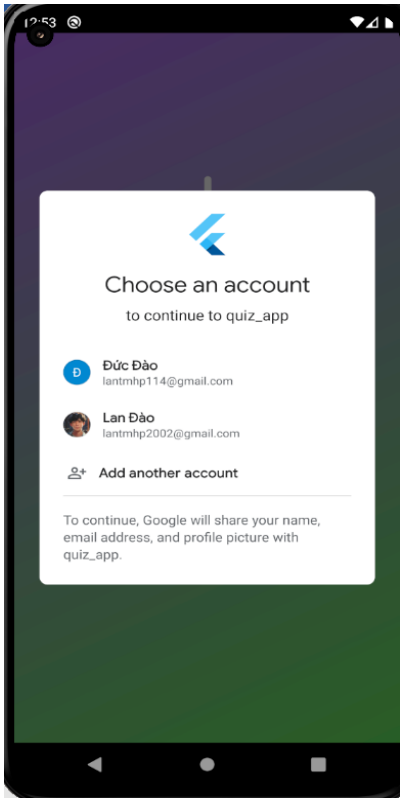
**Sơ đồ usecase**




*Hình 3. 14 Biểu đồ usecase đăng nhập của người dùng*

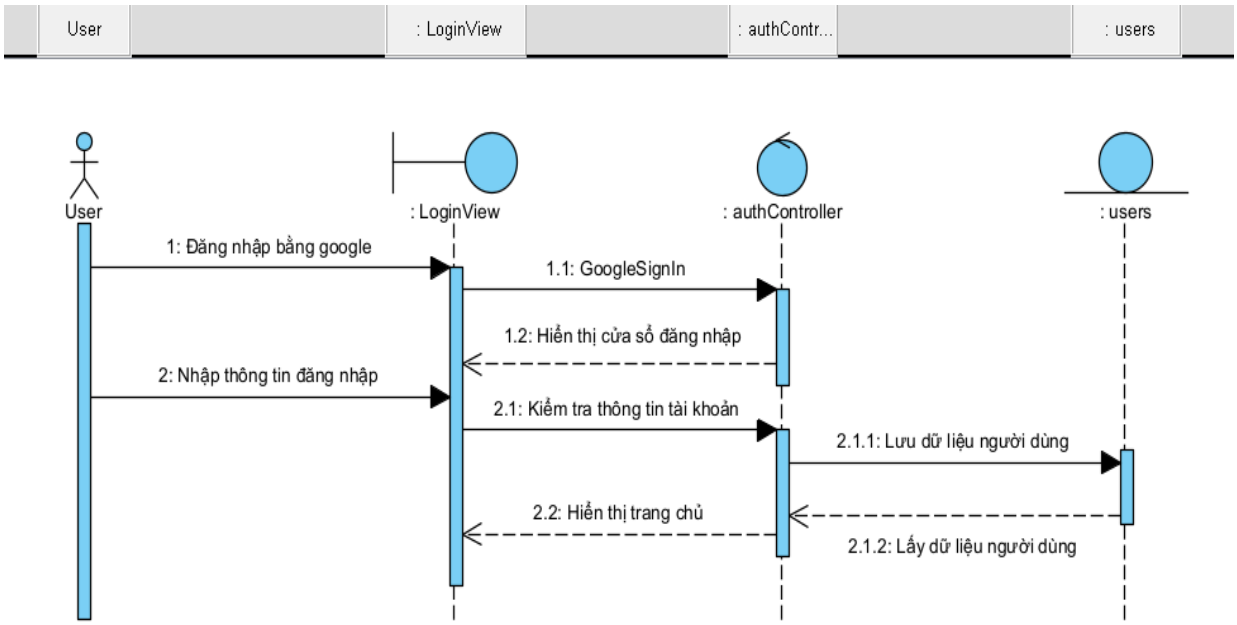
**Giao diện minh họa**


*Hình 3. 15 Giao diện màn hình đăng nhập*



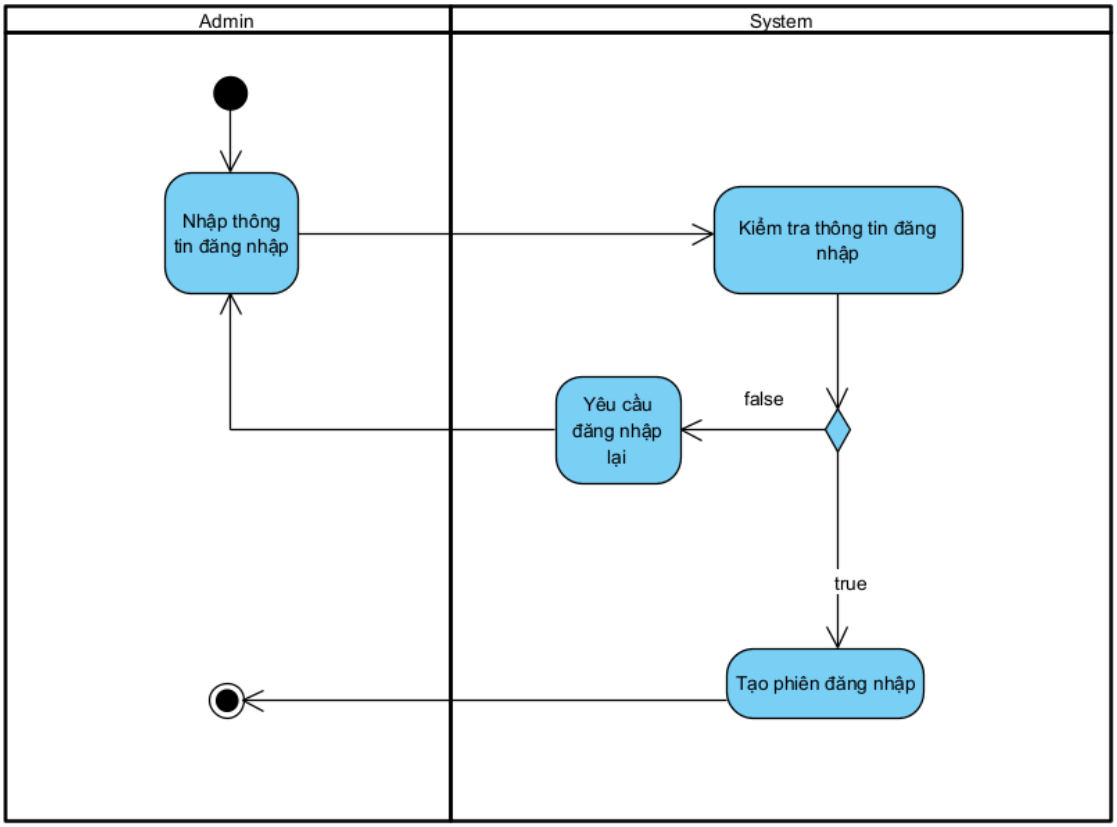

*Hình 3. 16 Giao diện màn hình form đăng nhập google*

**Sơ đồ Trình tự (Sequence diagram)**




*Hình 3. 17 Biểu đồ tuần tự đăng nhập người dùng*

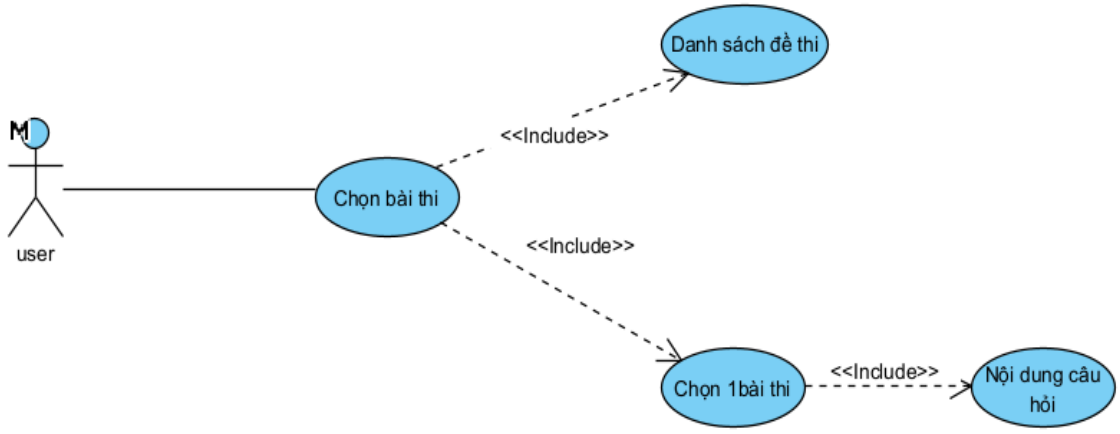
## Sơ đồ hoạt động (Activity diagram)




*Hình 3. 18 Biểu đồ hoạt động đăng nhập*

### 3.2.6 Chức năng chọn bài thi

<table>
<tr><td colspan="2">UC005</td><td>Chọn tài thi</td><td>Độ phức tạp: thấp</td></tr>
<tr><td colspan="2">Mô tả</td><td colspan="2">Chức năng này cho phép người dùng chọn 1 bải thi trong danh sách để bắt đầu làm bài.</td></tr>
<tr><td colspan="2">Tác nhân</td><td colspan="2">User</td></tr>
<tr><td colspan="2">Tiền điều kiện</td><td colspan="2">Người dùng sử dụng tài khoản của mình để đăng nhập vào hệ thống.</td></tr>
<tr><td rowspan="2">Hậu điều kiện</td><td>Thành công</td><td colspan="2">Người dùng có thể chọn 1 bài thi trong danh sách bài thi để bắt đầu làm bài.</td></tr>
<tr><td>Lỗi</td><td colspan="2">Trạng thái hệ thống sẽ không thay đổi.</td></tr>
<tr><td colspan="4">ĐẶC TẢ CHỨC NĂNG</td></tr>
<tr><td colspan="4">Luồng sự kiện chính/Kịch bản chính</td></tr>
<tr><td colspan="4">USECASE này hoạt động khi người dùng hệ thống muốn chọn 1 bài thi<br>– Hệ thống sẽ lấy ra và hiển thị danh sách các bài thi mà hệ thống lưu trữ.<br>– Hệ thống cho phép người dùng có thể chọn 1 bài thi trong danh sách.<br>3. Chọn bàu thi<br>– Nếu người dùng chưa đăng nhập hệ thống sẽ yêu cầu đăng nhập để tiếp tục.</td></tr>
<tr><td colspan="4">Sơ đồ usecase</td></tr>
</table>

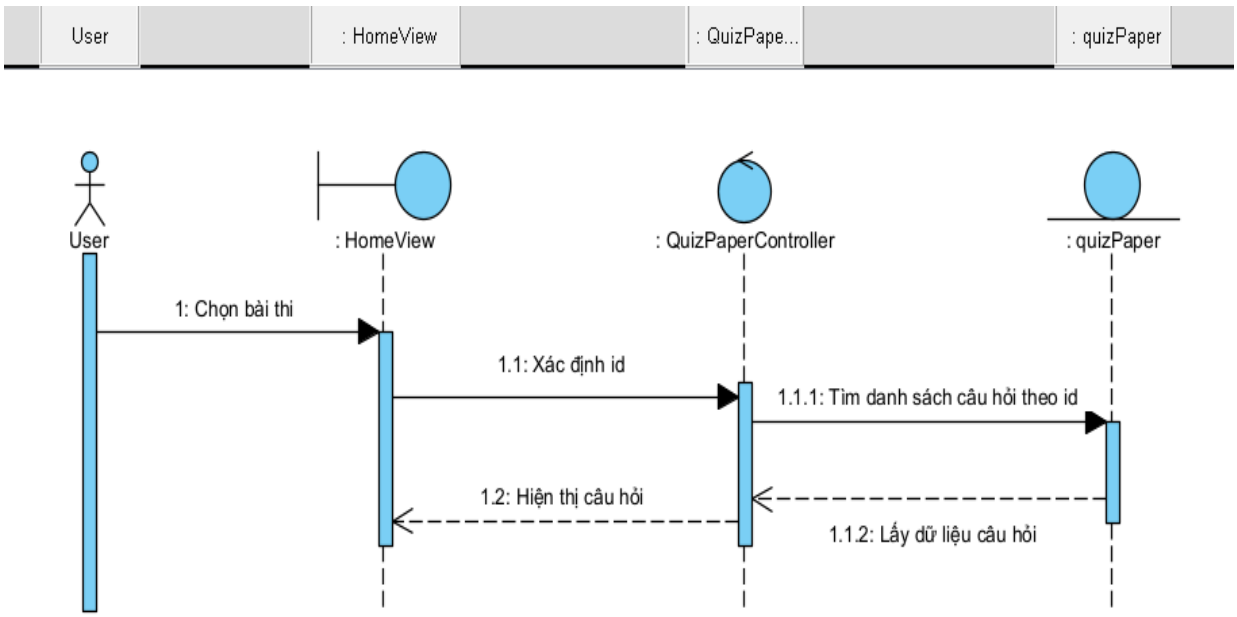



*Hình 3. 19 Biểu đồ usecase chọn bài thi*

**Giao diện minh họa**


*Hình 3. 20 Giao diện màn hình danh sách bài thi*

**Sơ đồ Trình tự (Sequence diagram)**




*Hình 3. 21 Biểu đồ tuần tự chọn bài thi*

### 3.2.7 Chức năng làm bài thi

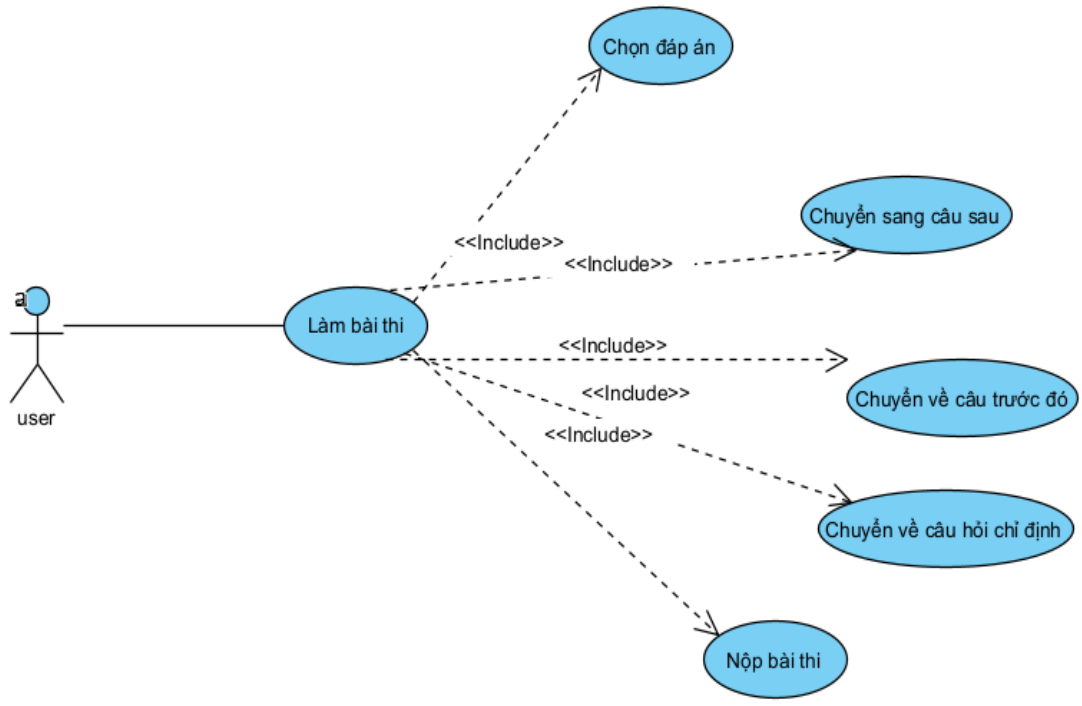
| **UC006** | | Làm bài thi | Độ phức tạp: cao |
|---|---|---|---|
| **Mô tả** | | Chức năng này cho phép người dùng chọn đáp án để trả lời câu hỏi, chuyển câu hỏi và nộp bài thi | |
| **Tác nhân** | | User | |
| **Tiền điều kiện** | | Người dùng đã chọn 1 bài thi. | |
| **Hậu điều kiện** | **Thành công** | Giao diện câu hỏi và đáp án được hiển thị. | |
| | **Lỗi** | Trạng thái không thay đổi. | |
| **ĐẶC TẢ CHỨC NĂNG** | | | |
| **Luồng sự kiện chính/Kịch bản chính** | | | |

USECASE này hoạt động khi người dùng hệ thống đã chọn 1 bài thi và bắt đầu làm bài

- Hệ thống sẽ lấy ra và hiển thị lần lượt nội dung câu hỏi cùng với đáp án tương ứng.
- Hệ thống cho phép người dùng có thể chọn 1 đáp án với mỗi câu hỏi.
- Hệ thống cho phép người dùng chuyển câu hỏi với tháo tác tương ứng.

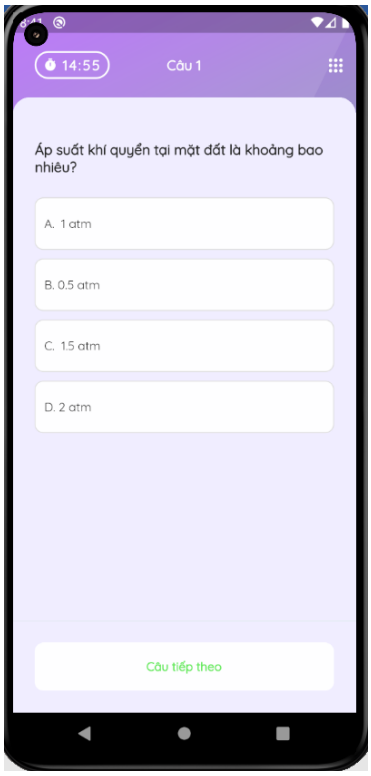
- Làm bài thi
  - Người dùng chọn 1 trong 4 đáp án để trả lời câu hỏi, có thể chọn lại.
  - Người dùng chọn "**câu hỏi tiếp theo"** để chuyển qua câu kế tiếp.
  - Người dùng chọn nút **"<"** để chuyển qua câu trước đó.
  - Hệ thống yêu cầu phải hoàn thành đủ các câu hỏi.
- Người dùng chọn "**Nộp bài**", hệ thống sẽ chấm điểm và đưa ra kết quả thi.

**Sơ đồ usecase**

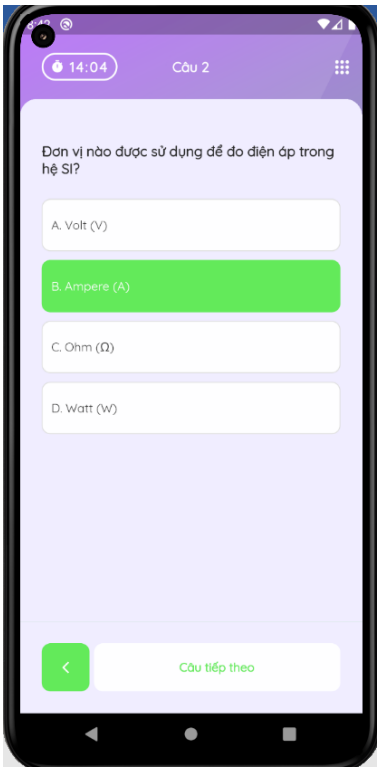



*Hình 3. 22 Biểu đồ usecase làm bài thi*

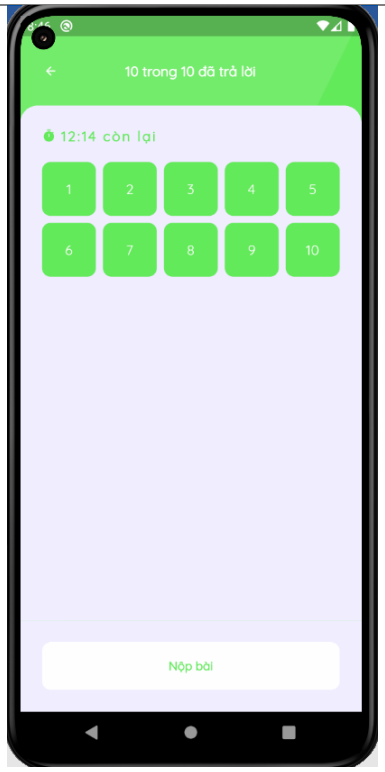
**Giao diện minh họa**




*Hình 3. 23 Giao diện màn hình câu hỏi 1*

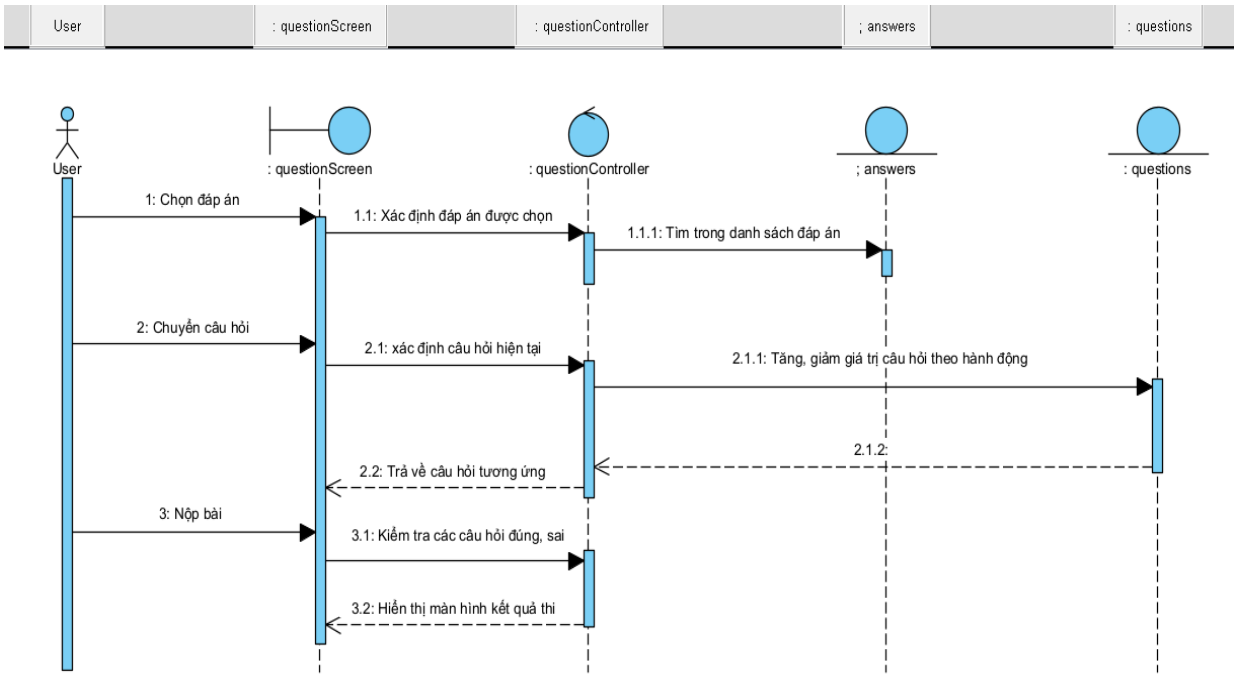



*Hình 3. 24 Giao diện màn hình câu hỏi*



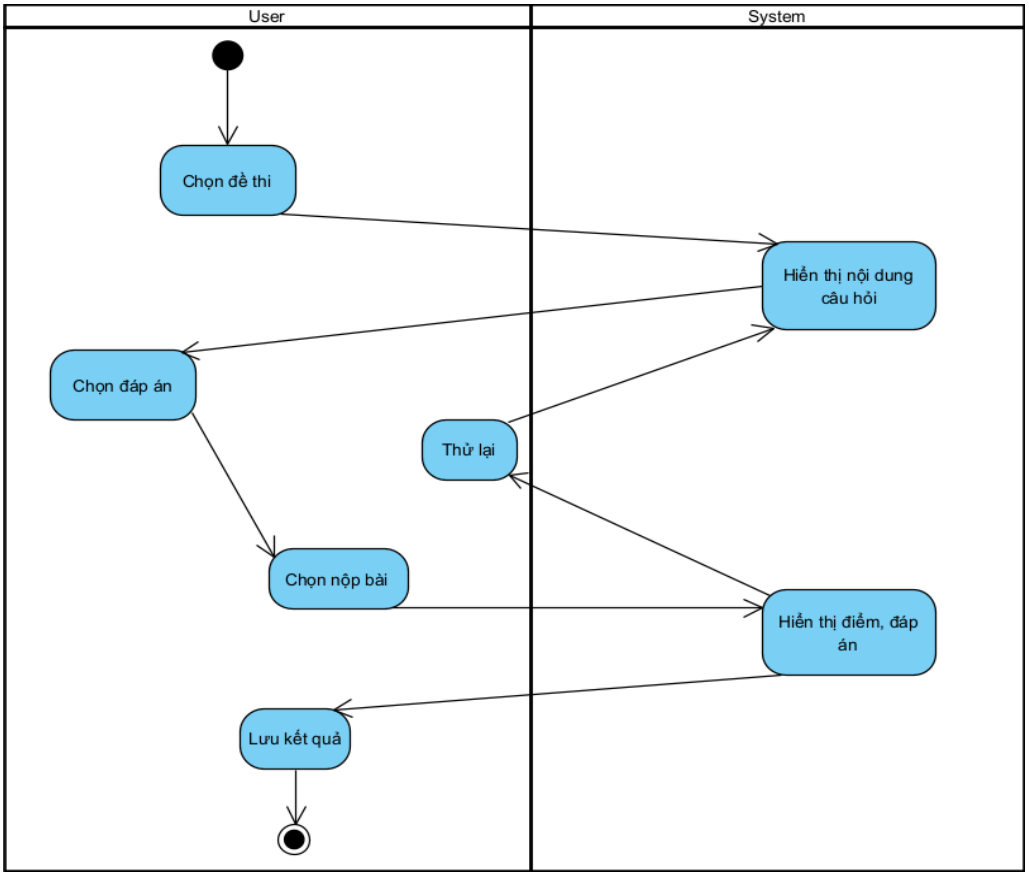

*Hình 3. 25 Giao diện màn hình nộp bài thi*

## Sơ đồ Trình tự (Sequence diagram)




*Hình 3. 26 Biểu đồ tuần tự làm bài thi*

**Sơ đồ hoạt động (Activity diagram)**




*Hình 3. 27 Biểu đồ hoạt động làm bài thi*

### 3.2.8 Chức năng xem kết quả thi

| | | | |
|---|---|---|---|
| **UC007** | | Xem kết quả thi | Độ phức tạp: trung bình |
| **Mô tả** | | Chức năng này cho phép người dùng xem kết quả thi và đáp án sau khi nộp bài thi | |
| **Tác nhân** | | User | |
| **Tiền điều kiện** | | Người dùng hoàn thành bài thi. | |
| **Hậu điều kiện** | **Thành công** | Giao diện kết quả thi được hiển thị. | |
| | **Lỗi** | Trạng thái không thay đổi. | |

**ĐẶC TẢ CHỨC NĂNG**

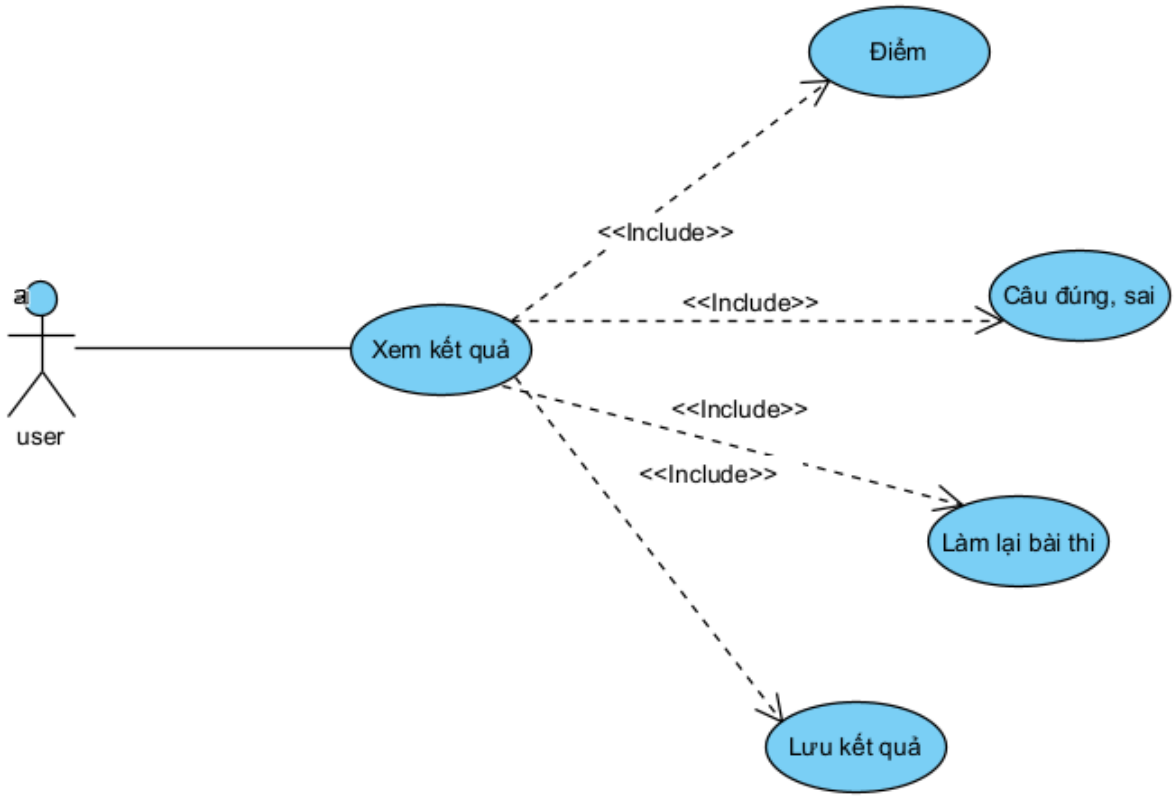
**Luồng sự kiện chính/Kịch bản chính**

USECASE này hoạt động khi người dùng hệ thống đã hoàn thành xong các câu hỏi và nộp bài

- Hệ thống sẽ lấy ra và hiển thị điểm và các câu hỏi đúng, sai của bài thi.
- Hệ thống cho phép người dùng có thể đáp án của các câu hỏi.
- Hệ thống cho phép người dùng làm lại bài thi.
- Hệ thống sẽ tự động lưu lại kết quả thi.

- Xem kết quả thi
  - Người dùng chọn câu hỏi muốn xem đáp án để hiển thị giao diện đáp án chính xác.
- Người dùng chọn "**Thử lại"** để bắt đầu lại bài thi.
- Người dùng chọn **"Về trang chủ"** để lưu lại kết quả thi.

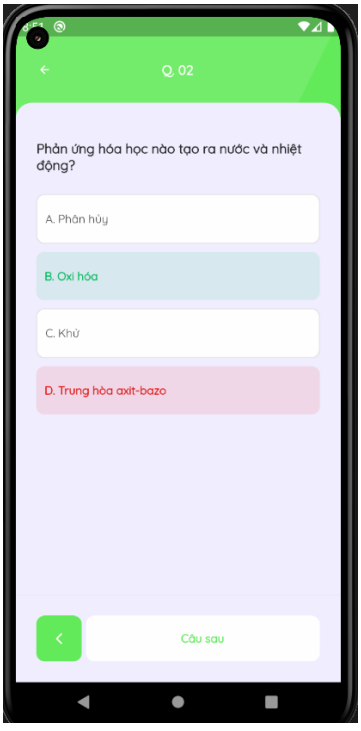
**Sơ đồ usecase**




*Hình 3. 28 Biểu đồ use case xem kết quả thi*

## Giao diện minh họa


*Hình 3. 29 Giao diện màn hình kết quả thi*




*Hình 3. 30 Giao diện màn hình xem đáp án*

### 3.2.9 Chức năng xem lịch sử thi

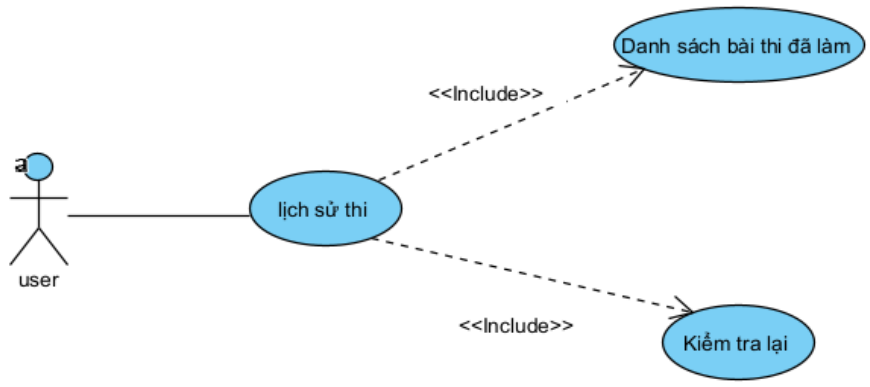
| | | | |
|---|---|---|---|
| **UC008** | | Lịch sử thi | Độ phức tạp: thấp |
| **Mô tả** | | Chức năng này cho phép người dùng xem lại kết quả các bài thi đã thi | |
| **Tác nhân** | | User | |
| **Tiền điều kiện** | | Người dùng đăng nhập thành công vào hệ thống. | |
| **Hậu điều kiện** | **Thành công** | Giao diện lịch sử thi được hiển thị. | |
| | **Lỗi** | Trạng thái không thay đổi. | |
| **ĐẶC TẢ CHỨC NĂNG** | | | |
| **Luồng sự kiện chính/Kịch bản chính** | | | |

USECASE này hoạt động khi người dùng hệ thống muốn xem lại lịch sử thi

- Hệ thống sẽ lấy ra và hiển thị danh sách các bài thi người dùng đã làm.
- Hệ thống cho phép người dùng kiểm tra lại bài thi.

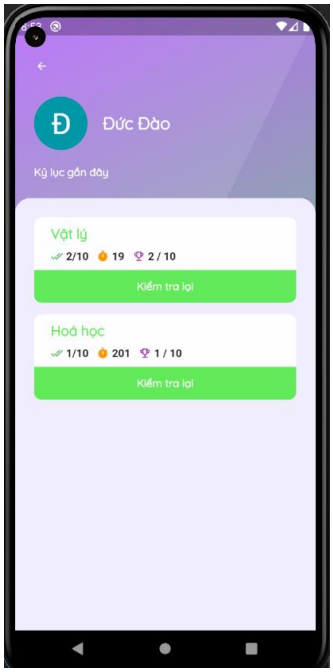
- Xem lịch sử thi
- Người dùng chọn "**Kiểm tra lại"** để bắt đầu lại bài thi.

**Sơ đồ usecase**



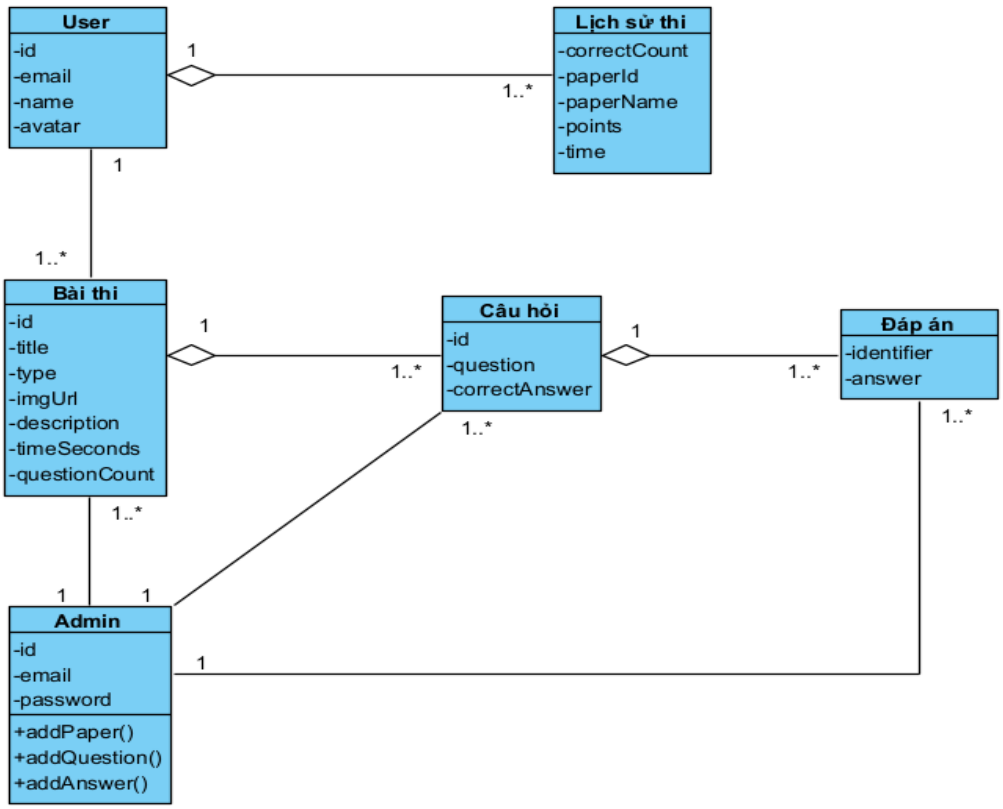

*Hình 3. 31 Biểu đồ usecase lịch sử thi*

**Giao diện minh họa**




*Hình 3. 32 Giao diện màn hình lịch sử thi*

### 3.2.10 Biểu đồ lớp thiết kế




*Hình 3. 33Biểu đồ lớp*

### 3.3 Thiết kế cơ sở dữ liệu

**Phân tích yêu cầu nghiệp vụ**

- Xác định đối tượng người dùng:
  + Học sinh
  + giáo viên
  + quản trị viên
- Xác định các chức năng chính:
  + Đăng nhập
  + Quản lý người dùng
  + Tạo bài thi
  + Làm bài thi
  + Xem kết quả
  + Quản lý câu hỏi

**Thiết kế mô hình dữ liệu**

- Xác định các thực thể:
  + Người dùng
  + Bài thi
  + Câu hỏi
  + Câu trả lời
  + Kết quả thi
- Tạo bảng thực thể

**Bảng questionPapers(Bài thi)**

| Tên trường | Kiểu dữ liệu | Diễn giải |
|---|---|---|
| Titile | string | Tên bài thi |
| Description | string | Mô tả |
| Img_url | string | Link ảnh |
| Question_count | number | Số câu hỏi |
| Time_second | number | Thời gian làm bài(giây) |

**Bảng questions(Câu hỏi)**

| Tên trường | Kiểu dữ liệu | Diễn giải |
|---|---|---|
| Question | string | Nội dung câu hỏi |
| Correct_answer | string | Đáp án đúng |

**Bảng answers(Câu trả lời)**

| Tên trường | Kiểu dữ liệu | Diễn giải |
|---|---|---|
| Identifier | string | Tên đáp án |
| answer | string | Nội dung đáp án |

**Bảng users(Người dùng)**

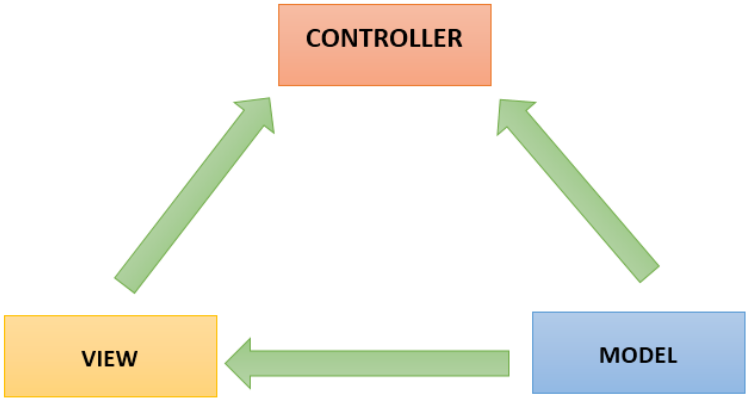
| Tên trường | Kiểu dữ liệu | Diễn giải |
|---|---|---|
| Email | string | Email |
| Name | string | Tên người dùng |
| Profilepic | string | Link avatar |

**Bảng my_quizes(Kết quả thi)**

| Tên trường | Kiểu dữ liệu | Diễn giải |
|---|---|---|
| Paper_id | string | Mã bài thi |
| Paper_name | string | Tên bài thi |
| Correct_count | string | Số câu trả lời đúng |
| Point | string | Điểm thi |
| Times | number | Thời gian hoàn thành(giây) |

### 3.4 Triển khai mô hình MVC

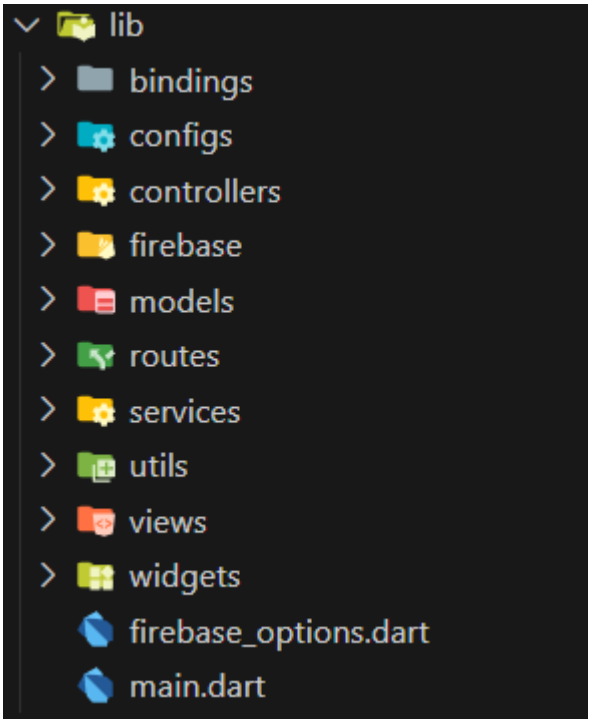
**MVC** là một mô hình (kiến trúc) trong lập trình, cho phép tách biệt các mã nghiệp vụ (business logic) và giao diện (UI) thành các phần riêng biệt, điều này đồng nghĩa với việc ta có thể chỉnh sửa chúng một cách riêng lẻ. MVC là chữ viết tắt của **Model - View - Controller**, đây là một mô hình được tạo ra với mục đích quản lý và xây dựng dự án phần mềm có hệ thống. Mô hình này được dùng khá rộng rãi và đặc biệt là trong các ngôn ngữ lập trình web.




*Hình 3. 34 Mô hình MVC*

*(Nguồn: Internet)*

**Cấu trúc cây thư mục**

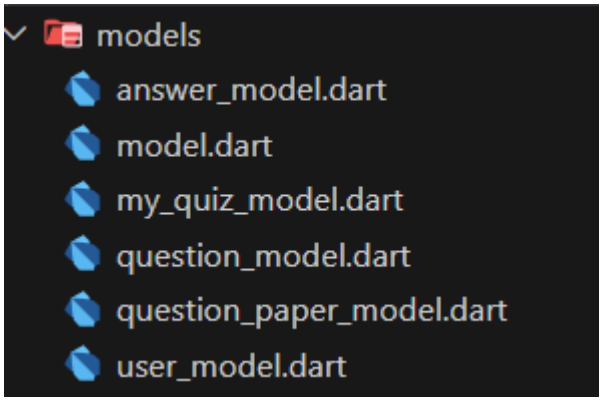


*Hình 3. 35 Cấu trúc cây thư mục*

**Model** đại diện cho dữ liệu và logic nghiệp vụ của ứng dụng. Đối với hệ thống thi trắc nghiệm online, Model sẽ bao gồm các lớp dữ liệu và tương tác với Firebase.

Các lớp Model:

- User: Thông tin về người dùng (học sinh, giáo viên).
- QuizPaper: Thông tin về các bài thi.
- Question: Thông tin về các câu hỏi trong kỳ thi.
- Answer: Thông tin về câu trả lời của người dùng.
- My_quizes: Thông tin kết quả các bài thi

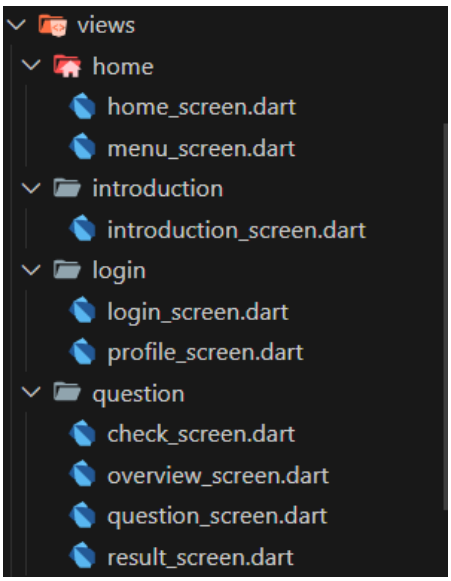



*Hình 3. 36 Cấu trúc thư mục models*

**View** đại diện cho giao diện người dùng và là nơi hiển thị dữ liệu. Đối với Flutter, View được xây dựng bằng cách sử dụng các Widget.

Các lớp View:

- LoginScreen: Màn hình đăng nhập.
- IntroductionScreen: Màn hình intro.
- HomeScreen: Màn hình trang chủ.
- MenuScreen: Màn hình menu.
- ProfileScreen: Màn hình thông tin người dùng.
- QuestiontScreen: Màn hình làm bài thi.
- CheckScreen: Màn hình xem đáp án.
- ResultScreen: Màn hình hiển thị kết quả thi.

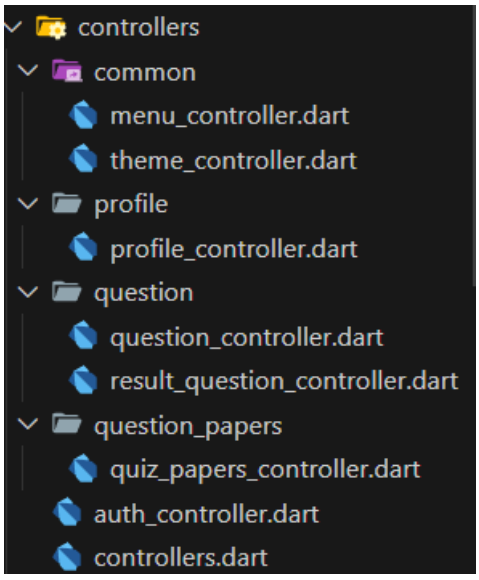



*Hình 3. 37 Cấu trúc thư mục views*

**Controller** xử lý các logic điều khiển, tương tác giữa Model và View. Controller nhận đầu vào từ người dùng qua View, xử lý logic nghiệp vụ qua Model, và cập nhật lại View.

Các lớp Controller:

- AuthController: Xử lý logic đăng nhập và xác thực.
- MenuController: Xử lý logic liên quan đến menu.
- ThemeController: Xử lý logic liên quan đến màu nền ứng dụng.
- QuizpaperController: Xử lý logic liên quan đến bài thi.
- QuestionController: Xử lý logic làm bài thi.
- ResultController: Xử lý logic liên quan đến kết quả thi.
- ProfileController: Xử lý logic liên quan đến trang cá nhân.




*Hình 3. 38Cấu trúc thư mục controllers*

# CHƯƠNG IV. KẾT LUẬN VÀ ĐỊNH HƯỚNG

Trong suốt quá trình thực hiện đề tài **"Xây dựng App thi trắc nghiệm trên Flutter",** em đã tiến hành phân tích, thiết kế và phát triển một ứng dụng thi trắc nghiệm trực tuyến trên smartphone. Ứng dụng đáp ứng được các yêu cầu cơ bản của việc tổ chức thi trắc nghiệm và mang lại nhiều tiện ích vượt trội như tính năng chấm điểm tự động, cung cấp đáp án chính xác và quản lý dữ liệu thi hiệu quả.

Sau quá trình nghiên cứu, tìm hiểu và thực hiện nhóm đã xây dựng thành công ứng dụng hệ thống quản lý giáo vụ tại trường "**Đại học Đại Nam"**. Hệ thống đã đáp ứng được những yêu cầu nghiệp vụ cơ bản của bài bài toán.

Về thực nghiệm, ứng dụng đã có khả năng áp dụng trong thực tế, mang lại lợi ích cho người dùng.

Sau khi hoàn thành đề tài này, em đã rút ra được rất nhiều kinh nghiệm thực tế và em hy vọng nó có thể góp phần trong công việc xây dựng và phát triển tại các trường đại học.

- Kết quả thu được:
  - Ứng dụng đã hoàn thành được các yêu cầu đặt ra.
  - Ứng dụng cung cấp tương đối đầy đủ của một ứng dụng thi trắc nghiêm.
  - Ứng dụng có giao diện thân thiện, dễ sử dụng.
  - Ứng dụng được xây dựng trên Flutter Framework nên có thể dễ dàng trong việc nâng cấp và phát triển.
- Hạn chế:
  - Giao diện chưa thực sự phù hợp chạy trên hệ điều hành IOS.
  - Tốc độ chương trình chưa được tối ưu.
  - Kho câu hỏi chưa phong phú
- Hướng phát triển:
  - Ứng dụng có thể phát triển để phù hợp với nhiều đối tượng hơn.
  - Phát triển thêm các tính năng cần thiết nhằm nâng cao chất lượng phần mềm.

# TÀI LIỆU THAM KHẢO

## Danh mục tài liệu tham khảo

[1] Flutter for Beginners: Build Cross-Platform Mobile Apps with Flutter 2.5 and Dart 2nd ed.

[2] Learn Google Flutter Fast: 65 Example Apps

## Danh mục website tham khảo

[1] https://www.youtube.com/@flutterdev

[2] https://github.com/flutter/gallery#flutter-gallery

[3] https://flutter.github.io/samples/#

[4] https://www.youtube.com/watch?v=fjqkuwgVTkc

[5] https://www.youtube.com/watch?v=BpGa9mhgL0U