**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN TỐT NGHIỆP**

**ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG QUẢN LÝ CÔNG VIỆC BẰNG FLUTTER VÀ FIREBASE**

Giảng viên hướng dẫn: TH.S TRẦN PHONG NHÃ

Sinh viên thực hiện: LÊ CÔNG ĐOAN

Lớp : CÔNG NGHỆ THÔNG TIN K58

Khoá : 58

Tp. Hồ Chí Minh, năm 2025

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN TỐT NGHIỆP**

**ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG QUẢN LÝ CÔNG VIỆC BẰNG FLUTTER VÀ FIREBASE**

Giảng viên hướng dẫn: TH.S TRẦN PHONG NHÃ

Sinh viên thực hiện: LÊ CÔNG ĐOAN

Lớp : CÔNG NGHỆ THÔNG TIN

Khoá : 58

Tp. Hồ Chí Minh, năm 2025

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI **CỘNG HÒA XÃ HỘI CHỦ NGHIÃ VIỆT NAM**

**PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH** Độc lập – Tự do – Hạnh phúc

**NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP**

BỘ MÔN: **CÔNG NGHỆ THÔNG TIN**

-------\*\*\*-------

**Mã sinh viên:**5851071018 **Họ tên SV:** Lê Công Đoan

**Khóa:** 58 **Lớp:** CQ.58.CNTT

1. **Tên đề tài:**

* Xây dựng ứng dụng quản lý công việc bằng Flutter và Firebase

1. **Mục đích, yêu cầu:**
2. **Mục đích:**

- Thiết kế và xây dựng một ứng dụng đa nền tảng (mobile và web) nhằm hỗ trợ việc quản lý công việc trong nhóm hoặc doanh nghiệp.

-Ứng dụng hỗ trợ tạo, phân công, theo dõi và cập nhật tiến độ công việc theo thời gian thực.

- Tăng tính hiệu quả và minh bạch trong việc giao việc và đánh giá tiến độ làm việc của các thành viên.

- Thông qua đề tài này, em muốn vận dụng về Flutter, Dart và Firestore để tạo ra sản phẩm hoàn chỉnh hơn, rèn luyện kỹ năng làm việc với dự án thực tế, chuẩn bị cho công việc sau này.

1. **Yêu cầu:**

* Ứng dụng có các vai trò: Admin, Quản lý và Nhân viên với mỗi vai trò có quyền hạn riêng.
* Admin tạo tài khoản và phân quyền cho người dùng.
* Tạo project, phân công task cho nhân viên và theo dõi tiến độ.
* Nhân viên nhận công việc, cập nhật trạng thái task.
* Các task có thể kéo thả để thay đổi trạng thái.
* Ghi lại lịch sử thay đổi.
* Gửi thông báo đến người có liên quan khi có thay đổi công việc.

1. **Nội dung và phạm vi đề tài:**
2. **Nội dung:**

* Tìm hiểu ngôn ngữ lập trình Flutter và Dart, sử dụng Firestore lưu cơ sở dữ liệu.
* Ứng dụng vào để xây dựng ứng dụng quản lý đồ án tốt nghiệp.

1. **Phạm vi đề tài:**

* Xây dựng ứng dụng quản lý công việc

1. **Công nghệ, công cụ và ngôn ngữ lập trình**
2. **Công nghệ:**

* Flutter
* Firebase: Authentication, **Cloud Firestore,** Cloud Messaging , Firebase Hosting

1. **Công cụ:**

* **Android Studio**
* GitHub
* Flutter SDK
* Dart SDK
* Firebase Console

1. **Ngôn ngữ lập trình:**

* Dart cùng với Flutter

1. **Các kết quả chính dự kiến sẽ đạt được và ứng dụng**

**-** Xây dựng được một ứng dụng quản lý công việc đa nền tảng (Android, iOS, Web) sử dụng Flutter và Firebase với giao diện thân thiện, dễ sử dụng.

- Hệ thống có chức năng phân quyền rõ ràng, hỗ trợ ba vai trò: Admin, Quản lý và Nhân viên, mỗi vai trò có giao diện và quyền truy cập tương ứng.

- Triển khai đầy đủ các chức năng cần thiết trong quản lý công việc.

- Đảm bảo khả năng đồng bộ thời gian thực (realtime) thông qua Cloud Firestore, giúp các thay đổi công việc được cập nhật tức thì trên nhiều thiết bị.

1. **Giáo viên và cán bộ hướng dẫn**

Họ tên: ThS Trần Phong Nhã

Đơn vị công tác: Bộ môn Công Nghệ Thông Tin – Trường Đại Học Giao Thông Vận Tải phân hiệu tại Thành Phố Hồ Chí Minh

Điện thoại: 0906761014 Email: tpnha@utc2.edu.vn

|  |  |
| --- | --- |
| **Ngày tháng 03 năm 2025**  **Trưởng BM Công nghệ Thông tin** | **Đã giao nhiệm vụ TKTN**  **Giáo viên hướng dẫn** |
| **ThS. Trần Phong Nhã** | **ThS. Trần Phong Nhã** |

Đã nhận nhiệm vụ TKTN

Sinh viên: Lê Công Đoan Ký tên:

Điện thoại: 0364602670 Email: 5851071018@st.utc2.edu.vn

**LỜI CẢM ƠN**

Trong suốt quá trình thực hiện đồ án tốt nghiệp em đã nhận được rất nhiều sự giúp đỡ và chỉ bảo tận tình, chi tiết tới từ thầy cô, gia đình và bạn bè.

Trước hết em xin gửi lời cảm ơn chân thành tới thầy Trần Phong Nhã, trưởng Bộ Môn Công Nghệ Thông Tin, Trường Đại Học Giao Thông Vận Tải phân hiệu tại Thành Phố Hồ Chí Minh là người đã trực tiếp hướng dẫn và góp ý để em hoàn thiện tốt đồ án tốt nghiệp của mình.

Đồng thời em cũng chân thành cảm ơn các thầy cô Trường Đại Học Giao Thông Vận Tải phân hiệu tại Thành Phố Hồ Chí Minh nói chung, thầy cô Bộ môn Công nghệ Thông tin nói riêng đã chỉ dạy cho em những kiến thức vững vàng từ cơ sở cho tới chuyên sâu là nền tảng để em học tập tốt và phấn đấu trở thành người có ích cho xã hội.

Cuối cùng em gửi lời cảm ơn sâu sắc tới gia đình, người thân đã luôn ở bên động viên, chăm sóc, giúp đỡ và quan tâm để em hoàn thành tốt quá trình học tập tại trường.

Em xin chân thành cảm ơn !

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

|  |
| --- |
| ***Tp. Hồ Chí Minh, ngày ….… tháng ….… năm ….…***  **Giáo viên hướng dẫn**  **Trần Phong Nhã** |

**MỤC LỤC**

[CHƯƠNG 1: MỞ ĐẦU 2](#_Toc202038110)

[1.1 Lí do chọn đề tài 2](#_Toc202038111)

[1.2 Mục tiêu của đề tài 2](#_Toc202038112)

[1.3 Giới hạn và phạm vi đề tài 3](#_Toc202038113)

[1.3.1 Giới hạn đề tài 3](#_Toc202038114)

[1.3.2 Phạm vi đề tài 3](#_Toc202038115)

[1.4 Kết quả dự kiến đạt được 4](#_Toc202038116)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 5](#_Toc202038117)

[2.1 Lý thuyết về Flutter 5](#_Toc202038118)

[2.1.1 Khái niệm 5](#_Toc202038119)

[2.1.2 Ưu điểm 5](#_Toc202038120)

[2.1.3 Nhược điểm 6](#_Toc202038121)

[2.1.4 Tính chất 7](#_Toc202038122)

[2.1.5 Lý do chọn Flutter cho đề tài 10](#_Toc202038123)

[2.2 Ngôn ngữ lập trình Dart 11](#_Toc202038124)

[2.2.1 Khái niệm 11](#_Toc202038125)

[2.2.2 Tính chất 12](#_Toc202038126)

[2.2.3 Ưu điểm 14](#_Toc202038127)

[2.2.4 Nhược điểm 15](#_Toc202038128)

[2.2.5 So sánh Dart với các ngôn ngữ lập trình phổ biến khác 16](#_Toc202038129)

[2.3 Lý thuyết về Firebase 18](#_Toc202038130)

[2.3.1 Giới thiệu về Firebase 18](#_Toc202038131)

[2.3.2 Lịch sử phát triển Firebase 18](#_Toc202038132)

[2.3.3 Cách thức hoạt động của Firebase 19](#_Toc202038133)

[2.3.4 Các dịch vụ của Firebase 20](#_Toc202038134)

[2.3.5 Ưu điểm của Firebase 22](#_Toc202038135)

[2.3.6 Nhược điểm của Firebase 24](#_Toc202038136)

[2.4 Cơ sở dữ liệu NoSQL 26](#_Toc202038137)

[2.4.1 Khái niệm NoSQL 26](#_Toc202038138)

[2.4.2 Đặc điểm của NoSQL 26](#_Toc202038139)

[2.4.3 Phân loại chính của NoSQL 26](#_Toc202038140)

[2.4.4 So sánh NoSQL và SQL (truyền thống) 27](#_Toc202038141)

[2.4.5 Ứng dụng của NoSQL trong hệ thống quản lý công việc 27](#_Toc202038142)

[2.5 Mô hình quản lý công việc 27](#_Toc202038143)

[2.5.1 Khái niệm quản lý công việc 27](#_Toc202038144)

[2.5.2 Cấu trúc mô hình quản lý công việc 27](#_Toc202038145)

[2.5.3 Luồng xử lý công việc 29](#_Toc202038146)

[2.5.4 Ưu điểm của mô hình 29](#_Toc202038147)

[2.5.5 Mô hình phù hợp với Firebase và Flutter 29](#_Toc202038148)

[2.6 Giao thức Client – Server và REST API 30](#_Toc202038149)

[2.6.1 Mô hình Client – Server 30](#_Toc202038150)

[2.6.2 Kiến trúc RESTful API 30](#_Toc202038151)

[2.6.3 Ưu điểm của REST API 31](#_Toc202038152)

[2.6.4 REST API trong Firebase Authentication 31](#_Toc202038153)

[2.6.5 Vai trò REST API trong ứng dụng 32](#_Toc202038154)

[2.6.6 Minh hoạ mô hình giao tiếp client – server 32](#_Toc202038155)

[CHƯƠNG 3: PHÂN TÍCH HỆ THỐNG 33](#_Toc202038156)

[3.1 Khảo sát hệ thống 33](#_Toc202038157)

[3.1.1 Khảo sát quy trình quản lý công việc 33](#_Toc202038158)

[3.1.2 Phân tích bài toán 34](#_Toc202038159)

[3.2 Biểu đồ phân rã chức năng 36](#_Toc202038160)

[3.2.1 Biểu đồ Use Case 37](#_Toc202038161)

[3.2.2 Biểu đồ tuần tự 45](#_Toc202038162)

[3.2.3 Thiết kế cơ sở dữ liệu sử dụng Firebase Firestore 48](#_Toc202038163)

[3.2.4 Thiết kế chi tiết các collection 48](#_Toc202038164)

[3.3 Thiết kế và xây dựng ứng dụng 52](#_Toc202038165)

[3.3.1 Giao diện trang đăng nhập 52](#_Toc202038166)

[3.3.2 Giao diện danh sách người dùng 52](#_Toc202038167)

[3.3.3 Giao diện form tạo dự án 53](#_Toc202038168)

[3.3.4 Giao diện form tạo công việc 53](#_Toc202038169)

[3.3.5 Giao diện cá nhân và công việc 54](#_Toc202038170)

[3.3.6 Giao diện thông tin cá nhân 54](#_Toc202038171)

[3.3.7 Giao diện quản lý công việc 55](#_Toc202038172)

[3.3.8 Giao diện chi tiết task 55](#_Toc202038173)

[3.3.9 Giao diện tìm kiếm công việc 56](#_Toc202038174)

[3.3.10 Giao diện thông báo 56](#_Toc202038175)

[3.3.11 Giao diện thống kê 57](#_Toc202038176)

[3.4 Kết luận 57](#_Toc202038177)

[3.4.1 Nhận xét 57](#_Toc202038178)

[3.4.2 Ưu điểm 58](#_Toc202038179)

[3.4.3 Hạn chế 58](#_Toc202038180)

[3.4.4 Hướng phát triển 58](#_Toc202038181)

[**PHỤ LỤC** 59](#_Toc202038182)

[**TÀI LIỆU THAM KHẢO** 60](#_Toc202038183)

**DANH MỤC CÁC BẢNG BIỂU**

[Bảng 2.1 So sánh dart và JavaScript 16](#_Toc202038184)

[Bảng 2.2 So sánh Dart và Java 17](#_Toc202038185)

[Bảng 2.3 So sánh Dart và Kotlin 17](#_Toc202038186)

[Bảng 2.4 Phân loại chính của NoSQL 26](#_Toc202038187)

[Bảng 2.5 So sánh NoSQL và SQL (truyền thống) 27](#_Toc202038188)

[Bảng 3.1 Bảng danh sách các cơ sở dữ liệu 48](#_Toc202038189)

[Bảng 3.2 Bảng Collection Users 49](#_Toc202038190)

[Bảng 3.3 Bảng Collection Projects 49](#_Toc202038191)

[Bảng 3.4 Bảng Collection Tasks 50](#_Toc202038192)

[Bảng 3.5 Bảng Collection Notifications 51](#_Toc202038193)

[Bảng 3.6 Bảng Collection Subtasks 51](#_Toc202038194)

**DANH MỤC CÁC HÌNH ẢNH**

[Hình 3.1 Biểu đồ phân rã chức năng của hệ thống 36](#_Toc202038195)

[Hình 3.2 Biểu đồ Use Case tổng quát 37](#_Toc202038196)

[Hình 3.3 Biểu đồ Use Case đăng nhập 37](#_Toc202038197)

[Hình 3.4 Biểu đồ Use Case quản lý dự án 38](#_Toc202038198)

[Hình 3.5 Biểu đồ Use Case quản lý công việc 39](#_Toc202038199)

[Hình 3.6 Biểu đồ Use Case quản lý subtask 41](#_Toc202038200)

[Hình 3.7 Biểu đồ Use Case quản lý thông báo và lịch sử 42](#_Toc202038201)

[Hình 3.8 Biểu đồ Use Case giao diện người dùng 43](#_Toc202038202)

[Hình 3.9 Biểu đồ tuần tự chức năng đăng nhập 45](#_Toc202038203)

[Hình 3.10 Biểu đồ tuần tự chức năng quản lý dự án 45](#_Toc202038204)

[Hình 3.11 Biểu đồ tuần tự chuức năng quản lý công việc 46](#_Toc202038205)

[Hình 3.12 Biểu đồ tuần tự chức năng quản lý subtask 46](#_Toc202038206)

[Hình 3.13 Biểu đồ tuần tự chức năng quản lý thông báo 47](#_Toc202038207)

[Hình 3.14 Biểu đồ tuần tự chức năng quản lý lịch sử 47](#_Toc202038208)

[Hình 3.15 Sơ đồ thiết kế cơ sở dữ liệu Firebase Firestore 48](#_Toc202038209)

[Hình 3.16 Giao diện trang đăng nhập 52](#_Toc202038210)

[Hình 3.17 Giao diện danh sách người dùng 52](#_Toc202038211)

[Hình 3.18 Giao diện form tạo dự án 53](#_Toc202038212)

[Hình 3.19 Giao diện form tạo công việc 53](#_Toc202038213)

[Hình 3.20 Giao diện cá nhân và công việc 54](#_Toc202038214)

[Hình 3.21 Giao diện thông tin cá nhân 54](#_Toc202038215)

[Hình 3.22 Giao diện quản lý công việc 55](#_Toc202038216)

[Hình 3.23 Giao diện chi tiết task 55](#_Toc202038217)

[Hình 3.24 Giao diện tìm kiếm công việc 56](#_Toc202038218)

[Hình 3.25 Giao diện thông báo 56](#_Toc202038219)

[Hình 3.26 Giao diện thông báo 57](#_Toc202038220)

**LỜI MỞ ĐẦU**

Trong thời đại công nghệ phát triển không ngừng như hiện nay, việc tổ chức, theo dõi và quản lý công việc ngày càng đóng vai trò quan trọng, đặc biệt là trong các doanh nghiệp, tổ chức hay thậm chí là trong những nhóm làm việc nhỏ. Nếu trước đây việc giao việc hay báo cáo tiến độ chủ yếu được thực hiện thủ công, gây ra nhiều bất cập như thiếu minh bạch, khó kiểm soát tiến độ hay thông tin bị đứt gãy, thì nay, với sự hỗ trợ của công nghệ, những vấn đề đó hoàn toàn có thể được giải quyết một cách hiệu quả.

Xuất phát từ nhu cầu thực tế đó, em lựa chọn đề tài “Xây dựng ứng dụng quản lý công việc trên nền tảng di động sử dụng Flutter và Firebase” làm đồ án tốt nghiệp. Ứng dụng được xây dựng nhằm hỗ trợ người dùng trong việc phân công, theo dõi tiến độ công việc cũng như phối hợp làm việc nhóm một cách linh hoạt, mọi lúc mọi nơi.

Công nghệ được sử dụng trong đồ án là Flutter – một bộ công cụ phát triển giao diện đa nền tảng do Google phát triển, kết hợp với Firebase – nền tảng backend mạnh mẽ hỗ trợ các tính năng như xác thực người dùng, cơ sở dữ liệu thời gian thực, lưu trữ và gửi thông báo. Việc kết hợp hai công nghệ này không chỉ giúp rút ngắn thời gian phát triển mà còn đảm bảo hiệu năng và khả năng mở rộng của hệ thống.

Thông qua đồ án, em mong muốn áp dụng những kiến thức đã học vào thực tế, đồng thời rèn luyện kỹ năng lập trình, thiết kế hệ thống, giải quyết vấn đề và làm việc độc lập. Đây cũng là cơ hội để em có thêm trải nghiệm trong việc xây dựng một sản phẩm phần mềm hoàn chỉnh, từ khâu lên ý tưởng, thiết kế giao diện, xây dựng chức năng cho đến thử nghiệm và triển khai.

Nội dung đề tài gồm ác chương:

**Chương 1:** Giới thiệu

**Chương 2:** Cơ sở lí thuyết

**Chương 3:** Nội dung thực hiện

# MỞ ĐẦU

## Lí do chọn đề tài

Trong thời đại công nghệ số hiện nay, việc quản lý công việc hiệu quả đóng vai trò quan trọng trong hoạt động của các cá nhân, nhóm làm việc và doanh nghiệp. Tuy nhiên, nhiều tổ chức và cá nhân vẫn gặp khó khăn trong việc theo dõi tiến độ, phân công nhiệm vụ và đánh giá hiệu quả làm việc.

Bên cạnh đó, xu hướng phát triển ứng dụng đa nền tảng (cross-platform) đang ngày càng phổ biến vì tính linh hoạt và tiết kiệm chi phí. Flutter, với khả năng viết một lần và chạy trên nhiều nền tảng (Android, iOS, Web), cùng với Firebase – một nền tảng Backend-as-a-Service mạnh mẽ – là một sự kết hợp lý tưởng để xây dựng các ứng dụng quản lý hiệu quả.

Chính vì những lí do nêu trên em đã quyết định chọn đề tài “Xây dựng ứng dụng quản lý công việc” để làm báo cáo tốt nghiệp giúp công việc quản lí đạt hiệu quả cao.

Thông qua đề tài này, em mong muốn tích lũy kinh nghiệm phát triển ứng dụng thực tế, từ thiết kế giao diện đến tích hợp hệ thống backend, đồng thời thể hiện năng lực tư duy và tổ chức hệ thống phần mềm.

## Mục tiêu của đề tài

Xây dựng một ứng dụng quản lý công việc đa nền tảng, hỗ trợ người dùng trong việc tổ chức, phân công và theo dõi tiến độ công việc theo thời gian thực, góp phần nâng cao hiệu suất làm việc nhóm và cá nhân.

Thiết kế và phát triển giao diện người dùng thân thiện, hiện đại trên nền tảng Flutter.

Triển khai hệ thống xác thực người dùng và phân quyền bằng Firebase Authentication.

Xây dựng cơ sở dữ liệu thời gian thực bằng Firebase Firestore để lưu trữ thông tin người dùng, dự án và công việc.

Phát triển các chức năng chính:

* Quản lý người dùng và phân quyền theo 3 vai trò: Admin, Quản lý, Nhân viên.
* Tạo, phân công, theo dõi và cập nhật trạng thái công việc.
* Hỗ trợ tạo subtask và theo dõi tiến độ chi tiết.
* Gửi thông báo tự động đến người dùng liên quan.
* Thống kê, hiển thị biểu đồ báo cáo tiến độ công việc và dự án.

Áp dụng mô hình Kanban trong quản lý tiến trình công việc.

Đảm bảo tính tương thích ứng dụng trên cả mobile và web.

## Giới hạn và phạm vi đề tài

### Giới hạn đề tài

* Hệ thống không tích hợp các công cụ quản lý tài nguyên nâng cao như chấm công, báo cáo tài chính, lương thưởng, KPI cá nhân.
* Chưa tích hợp AI hoặc machine learning để tự động phân tích hiệu suất hoặc đề xuất tối ưu công việc.
* Tính năng trò chuyện, tương tác trực tiếp (chat realtime) giữa người dùng chưa được triển khai.
* Giao diện được thiết kế ở mức tối ưu hóa cơ bản, chưa áp dụng các nguyên lý thiết kế UI/UX chuyên sâu hoặc animation nâng cao.
* Chưa hỗ trợ phân quyền linh hoạt theo từng chức năng nhỏ (role-based access control chi tiết).

### Phạm vi đề tài

* Đề tài tập trung vào việc xây dựng một ứng dụng quản lý công việc đa nền tảng (web và mobile) sử dụng Flutter làm framework phát triển giao diện và Firebase làm nền tảng backend (Authentication, Firestore, Cloud Messaging).
* Ứng dụng dành cho các nhóm làm việc nhỏ hoặc tổ chức vừa và nhỏ, có nhu cầu quản lý công việc theo nhóm và theo dõi tiến độ dự án.
* Các chức năng chính bao gồm:
* Đăng ký/đăng nhập người dùng qua số điện thoại hoặc email.
* Phân quyền người dùng theo 3 vai trò: Admin, Quản lý, Nhân viên.
* Tạo và quản lý dự án, quy trình (process), công việc (task), công việc con (subtask).
* Giao việc, cập nhật trạng thái, thống kê và ghi log thay đổi.
* Gửi thông báo realtime cho người được giao nhiệm vụ.
* Giao diện kéo thả (kanban) để cập nhật trạng thái công việc.

## Kết quả dự kiến đạt được

Sau khi hoàn thành đề tài, dự kiến sẽ đạt được các kết quả sau:

* Xây dựng thành công một ứng dụng quản lý công việc hoạt động ổn định trên cả nền tảng mobile (Android/iOS) và web, sử dụng Flutter làm công cụ phát triển giao diện và Firebase làm nền tảng backend.
* Ứng dụng đáp ứng đầy đủ các chức năng chính của một hệ thống quản lý công việc:
  + Đăng ký, đăng nhập, phân quyền người dùng (Admin, Quản lý, Nhân viên).
  + Tạo và quản lý dự án, task, subtask theo mô hình Kanban (To Do – Doing – Done – Complete).
  + Giao việc, cập nhật tiến độ, trạng thái task và log lịch sử thay đổi.
  + Gửi thông báo realtime đến người dùng khi có sự kiện liên quan.
* Giao diện thân thiện, dễ sử dụng, tối ưu cho người dùng phổ thông, phù hợp với nhóm làm việc nhỏ hoặc tổ chức quy mô vừa.
* Cơ sở dữ liệu được thiết kế hợp lý và linh hoạt, đáp ứng các truy vấn theo vai trò người dùng và trạng thái công việc.
* Cung cấp thống kê trực quan, giúp người dùng dễ dàng theo dõi tiến độ cá nhân, nhóm và toàn bộ dự án.
* Áp dụng được mô hình phân quyền người dùng linh hoạt, đảm bảo mỗi vai trò trong hệ thống chỉ được thao tác đúng chức năng cho phép, tránh xung đột hoặc sai lệch dữ liệu.
* Ứng dụng hỗ trợ cập nhật trạng thái công việc theo thời gian thực (realtime), giúp các thành viên trong nhóm nắm bắt nhanh tiến độ, hạn chế trễ deadline.

# CƠ SỞ LÝ THUYẾT

## Lý thuyết về Flutter

### Khái niệm

Flutter là một nền tảng hỗ trợ phát triển  cho các ứng dụng đa nền cho hệ iOS và Android do tập đoàn Google phát triển. Nó được sử dụng  vô cùng phổ biến cho nhiệm vụ tạo ra các ứng dụng gốc dành cho Google.

Flutter thường bao gồm 2 thành phần chính quan trọng như sau:

- Một SDK (Software Development Kit): Đây là một bộ sưu tập bao gồm các công cụ có thể hỗ trợ cho người dùng có thể  phát triển được các ứng dụng nền của mình.

- Một Framework (UI Library based on widgets): Mỗi một tập hợp những thành phần giao diện của người dùng đều có thể thực hiện tái sử dụng vô cùng dễ dàng nên người sử dụng có thể cá nhân hóa tùy theo nhu cầu riêng của bản thân mình.

### Ưu điểm

Phát triển đa nền tảng: Một trong những ưu điểm lớn nhất của Flutter là khả năng viết một lần – chạy trên nhiều nền tảng. Thay vì phát triển riêng biệt cho Android (Java/Kotlin) và iOS (Swift/Objective-C), Flutter cho phép xây dựng ứng dụng chỉ với một codebase duy nhất.

Giao diện đẹp, linh hoạt:Flutter cung cấp hệ thống widget phong phú và có thể tùy biến cao, phù hợp với Material Design của Android và Cupertino của iOS. Điều này giúp lập trình viên dễ dàng tạo ra giao diện hiện đại, hấp dẫn, phù hợp với yêu cầu cụ thể của từng ứng dụng.

Hiệu suất cao:Nhờ sử dụng engine đồ họa Skia, Flutter render trực tiếp lên canvas mà không thông qua native component của hệ điều hành. Điều này giúp giao diện chạy nhanh, mượt và tránh bị phụ thuộc vào phiên bản OS.

Hot reload:Cho phép thay đổi mã nguồn và xem kết quả tức thì mà không cần phải khởi động lại ứng dụng. Tính năng này giúp rút ngắn thời gian phát triển và thử nghiệm.

Tài liệu rõ ràng, cộng đồng mạnh:Flutter được hỗ trợ bởi Google và có một cộng đồng phát triển lớn. Tài liệu hướng dẫn chính thức, thư viện và plugin rất phong phú, thuận tiện cho việc học và mở rộng.

### Nhược điểm

Mặc dù Flutter là một framework mạnh mẽ và ngày càng được ưa chuộng trong phát triển ứng dụng đa nền tảng, nhưng vẫn còn tồn tại một số nhược điểm và hạn chế như sau:

* Dung lượng ứng dụng lớn: Ứng dụng được phát triển bằng Flutter thường có kích thước file cài đặt (APK/IPA) lớn hơn so với ứng dụng native, do phải tích hợp thêm engine Skia và các thư viện Dart runtime. Điều này có thể ảnh hưởng đến trải nghiệm người dùng khi tải ứng dụng hoặc triển khai trên các thiết bị có bộ nhớ hạn chế.
* Hiệu suất chưa tối ưu trong một số trường hợp : Mặc dù Flutter có hiệu năng tốt với giao diện người dùng, nhưng đối với các ứng dụng cần xử lý tính toán phức tạp, đa luồng, hoặc tương tác nhiều với phần cứng (camera, cảm biến, Bluetooth…), Flutter có thể không tối ưu bằng native.
* Chưa hỗ trợ tốt các API nền tảng đặc thù: Một số chức năng đặc biệt của hệ điều hành (ví dụ: trình gọi hệ thống, thông báo nền phức tạp, tích hợp với các phần mềm khác…) đòi hỏi phải viết mã native (Java/Kotlin hoặc Swift/Objective-C), khiến việc phát triển phức tạp hơn nếu cần hỗ trợ sâu theo từng nền tảng.
* Tài nguyên học tập và cộng đồng Dart còn hạn chế: Mặc dù Flutter có cộng đồng lớn, nhưng ngôn ngữ Dart lại không phổ biến như JavaScript hay Python. Điều này khiến số lượng tài liệu chuyên sâu, thư viện hỗ trợ, ví dụ kỹ thuật có thể hạn chế hơn so với các ngôn ngữ phổ biến khác.
* Hạn chế trong khả năng tái sử dụng code giữa web và desktop: Mặc dù Flutter hỗ trợ nhiều nền tảng, nhưng vẫn tồn tại sự khác biệt về hành vi, giao diện và tính năng giữa mobile, web và desktop. Lập trình viên vẫn cần kiểm tra và điều chỉnh riêng cho từng nền tảng, khiến việc bảo trì và phát triển ứng dụng trở nên phức tạp nếu không có kế hoạch rõ ràng.
* Vấn đề tương thích lâu dài: Do Flutter và Dart đều do Google phát triển và chưa trở thành chuẩn công nghiệp lâu dài như Java hay C#, nên vẫn tồn tại một mức độ rủi ro nhất định nếu Google thay đổi chiến lược trong tương lai.

### Tính chất

Flutter là một bộ công cụ phát triển giao diện người dùng (UI toolkit) do Google phát triển, được thiết kế nhằm hỗ trợ lập trình viên tạo ra các ứng dụng đa nền tảng từ một bộ mã nguồn duy nhất. Nhờ vào các đặc tính kỹ thuật nổi bật và tính linh hoạt trong thiết kế giao diện, Flutter đã trở thành một trong những lựa chọn phổ biến nhất trong lĩnh vực phát triển ứng dụng di động và web. Dưới đây là các tính chất đặc trưng của Flutter.

#### Đa nền tảng (Cross-platform)

Một trong những tính chất nổi bật và quan trọng nhất của Flutter là khả năng phát triển ứng dụng đa nền tảng từ một codebase duy nhất. Điều này có nghĩa là lập trình viên chỉ cần viết một lần và có thể triển khai trên nhiều hệ điều hành khác nhau như Android, iOS, Web, Windows, macOS và Linux.

Tính chất này giúp tiết kiệm đáng kể thời gian và chi phí phát triển phần mềm, đồng thời giảm thiểu rủi ro khi phải bảo trì các mã nguồn riêng biệt cho từng nền tảng. Đây chính là yếu tố khiến Flutter trở thành giải pháp lý tưởng cho các startup, doanh nghiệp nhỏ và các đề tài nghiên cứu với nguồn lực hạn chế.

#### Hướng widget (Widget-driven UI)

Toàn bộ giao diện trong Flutter được xây dựng dựa trên hệ thống widget. Widget trong Flutter là những thành phần giao diện nhỏ nhất – có thể là một nút nhấn, một khung văn bản, một biểu tượng, hoặc thậm chí là cả một layout phức tạp.

Flutter cung cấp hai nhóm widget chính: StatelessWidget (không thay đổi theo thời gian) và StatefulWidget (có thể thay đổi trạng thái và tái tạo lại). Việc mọi thứ đều là widget tạo nên một kiến trúc giao diện nhất quán, có thể tái sử dụng và tùy chỉnh mạnh mẽ, đồng thời giúp việc xây dựng giao diện phức tạp trở nên trực quan và dễ kiểm soát.

#### Lập trình phản ứng (Reactive Programming)

Flutter áp dụng mô hình lập trình phản ứng (reactive programming) – một mô hình hiện đại và ngày càng phổ biến trong phát triển giao diện người dùng. Với cách tiếp cận này, khi trạng thái của dữ liệu thay đổi, Flutter sẽ tự động cập nhật lại giao diện tương ứng, mà không cần phải thao tác thủ công hoặc viết các đoạn mã xử lý phức tạp.

Điều này đặc biệt quan trọng trong các ứng dụng có luồng dữ liệu động như ứng dụng quản lý công việc, nơi mà trạng thái task, thông báo hay người dùng thường xuyên thay đổi theo thời gian thực.

#### Hiệu suất cao (High Performance)

Khác với các framework cross-platform khác như React Native, vốn phải dựa vào các component native của hệ điều hành để hiển thị giao diện, Flutter sử dụng engine đồ họa riêng là Skia để vẽ trực tiếp giao diện lên canvas. Điều này giúp Flutter kiểm soát hoàn toàn cách mọi pixel được hiển thị, từ đó mang lại hiệu năng gần như tương đương với native.

Trong thực tế, các ứng dụng Flutter có tốc độ khởi động nhanh, phản hồi mượt mà và có thể xử lý hiệu ứng chuyển động phức tạp mà không gây hiện tượng giật, lag.

#### Tùy biến giao diện mạnh mẽ (Highly Customizable UI)

Flutter cho phép tùy chỉnh sâu từng thành phần giao diện, từ màu sắc, kích thước, font chữ cho đến hiệu ứng, transition và animation. Không giống như một số framework chỉ giới hạn trong bộ giao diện mặc định của hệ điều hành, Flutter cho phép tạo ra UI “theo ý muốn”, không bị ràng buộc bởi giới hạn thiết kế gốc.

Tính chất này rất phù hợp cho các ứng dụng cần giao diện đặc biệt như: bảng quản lý công việc dạng Kanban, biểu đồ động, bảng thống kê... như trong đề tài này.

#### Phát triển nhanh chóng (Fast Development)

Tính năng Hot Reload là một trong những ưu điểm vượt trội nhất của Flutter. Nó cho phép lập trình viên thay đổi mã nguồn và xem kết quả ngay lập tức mà không cần phải biên dịch lại toàn bộ ứng dụng.

Tính chất này không chỉ giúp tăng tốc quá trình phát triển, mà còn nâng cao hiệu quả làm việc nhóm khi thử nghiệm giao diện, tinh chỉnh UI/UX và debug ứng dụng. Điều này đặc biệt quan trọng trong các đề tài có giới hạn thời gian thực hiện như đồ án tốt nghiệp hoặc nghiên cứu học kỳ.

#### Mã nguồn mở và cộng đồng mạnh

Flutter là một dự án mã nguồn mở, được phát hành bởi Google và được cộng đồng đóng góp tích cực. Các bản cập nhật của Flutter được phát hành liên tục, với sự cải tiến về hiệu năng, tính năng mới và sửa lỗi.

Ngoài ra, Flutter cũng sở hữu một hệ sinh thái thư viện (package) rất phong phú, dễ dàng tích hợp các dịch vụ phổ biến như Firebase, Google Maps, Stripe, camera, file, v.v. Điều này giúp tiết kiệm thời gian phát triển và mở rộng ứng dụng một cách linh hoạt.

#### Kiến trúc linh hoạt (Flexible Architecture)

Flutter cung cấp nhiều mô hình kiến trúc hiện đại, phổ biến như: MVVM, Bloc Pattern, Provider, Riverpod,… giúp tổ chức mã nguồn rõ ràng, dễ bảo trì và dễ mở rộng. Việc hỗ trợ nhiều mô hình giúp lập trình viên linh hoạt lựa chọn cách tiếp cận phù hợp với quy mô và tính chất dự án.

**Tổng kết:**

Từ các tính chất trên, có thể thấy Flutter là một công nghệ mang tính cách mạng trong phát triển phần mềm đa nền tảng. Với khả năng linh hoạt, hiệu năng cao và hỗ trợ phát triển nhanh, Flutter không chỉ phù hợp cho các ứng dụng thương mại lớn mà còn đặc biệt hữu ích trong các đề tài nghiên cứu, học thuật hoặc các ứng dụng quản lý công việc như đề tài này.

### Lý do chọn Flutter cho đề tài

Trong quá trình lựa chọn công nghệ để triển khai đề tài “Ứng dụng quản lý công việc”, nhóm tác giả đã cân nhắc nhiều framework phát triển đa nền tảng như React Native, Xamarin, Ionic,... Tuy nhiên, sau khi nghiên cứu và đánh giá thực tế, nhóm quyết định lựa chọn Flutter là công nghệ chính để xây dựng ứng dụng. Quyết định này được đưa ra dựa trên các lý do sau:

* Phát triển đa nền tảng từ một codebase: Một trong những yêu cầu chính của đề tài là xây dựng ứng dụng có thể sử dụng được trên nhiều nền tảng khác nhau, bao gồm Android, iOS và Web. Flutter cho phép viết mã một lần và biên dịch sang nhiều nền tảng, giúp tiết kiệm thời gian phát triển, chi phí và công sức duy trì mã nguồn so với cách phát triển riêng biệt từng nền tảng.
* Giao diện linh hoạt, dễ tùy biến: Ứng dụng quản lý công việc cần có giao diện trực quan, dễ sử dụng, có thể biểu diễn các công việc theo **bảng Kanban**, hỗ trợ kéo thả, cập nhật trạng thái và hiển thị thông tin người thực hiện. Flutter cung cấp hệ thống widget phong phú và hỗ trợ tùy chỉnh mạnh mẽ, cho phép thiết kế giao diện đáp ứng đúng yêu cầu thực tế của đề tài.
* Hiệu năng cao và trải nghiệm người dùng tốt: Flutter sử dụng engine Skia để render giao diện trực tiếp lên canvas mà không cần thông qua thành phần native, giúp ứng dụng chạy mượt mà, phản hồi nhanh. Điều này đặc biệt quan trọng trong các thao tác realtime như kéo thả task, cập nhật trạng thái hay hiển thị danh sách công việc lớn.
* Hỗ trợ tích hợp Firebase dễ dàng: Firebase là một nền tảng backend mạnh mẽ được Google phát triển, bao gồm Authentication, Firestore (cơ sở dữ liệu realtime), Cloud Messaging, v.v. Flutter có hệ sinh thái thư viện tương thích rất tốt với Firebase, hỗ trợ việc xây dựng các chức năng như:
* Đăng nhập bằng số điện thoại/email.
* Phân quyền người dùng theo vai trò (Admin, Quản lý, Nhân viên).
* Lưu trữ và cập nhật trạng thái công việc theo thời gian thực.
* Gửi thông báo đến người dùng liên quan khi có thay đổi.
* Cộng đồng lớn, nhiều tài nguyên hỗ trợ: Flutter có cộng đồng phát triển mạnh mẽ, tài liệu hướng dẫn đầy đủ, ví dụ thực tiễn và thư viện mở phong phú. Điều này giúp nhóm tác giả dễ dàng tra cứu, học hỏi và giải quyết vấn đề trong quá trình triển khai đề tài, đặc biệt phù hợp trong môi trường học thuật có hạn chế về thời gian và kinh nghiệm thực tế.
* Tối ưu cho việc học tập và mở rộng: Flutter là công nghệ hiện đại, được sử dụng rộng rãi trong nhiều công ty công nghệ lớn, phù hợp với xu hướng phát triển ứng dụng hiện nay. Việc lựa chọn Flutter không chỉ giúp đề tài đảm bảo chất lượng, mà còn tạo cơ hội để người thực hiện nâng cao kỹ năng thực tế, dễ dàng tiếp cận các dự án chuyên nghiệp trong tương lai.

**Kết luận:** Với các ưu điểm vượt trội về đa nền tảng, hiệu năng, khả năng tùy biến giao diện, tích hợp Firebase mạnh mẽ và cộng đồng hỗ trợ tốt, Flutter là lựa chọn hoàn toàn phù hợp cho đề tài “Ứng dụng quản lý công việc”. Việc sử dụng Flutter không chỉ đáp ứng tốt các yêu cầu chức năng và phi chức năng, mà còn đảm bảo tính mở rộng và khả năng ứng dụng trong thực tế.

## Ngôn ngữ lập trình Dart

### Khái niệm

Dart là một ngôn ngữ lập trình được phát triển bởi Google, ra mắt lần đầu tiên vào năm 2011. Đây là ngôn ngữ được thiết kế với mục đích xây dựng các ứng dụng client-side hiệu quả, chạy được trên nhiều nền tảng khác nhau như web, mobile và desktop. Dart hiện tại là ngôn ngữ chính được sử dụng trong Flutter – một framework đa nền tảng nổi tiếng của Google.

Với cú pháp đơn giản, gần gũi với Java, JavaScript và C#, Dart phù hợp với cả người mới học lập trình và lập trình viên chuyên nghiệp. Ngôn ngữ này hỗ trợ lập trình hướng đối tượng, lập trình bất đồng bộ và biên dịch linh hoạt, cho phép phát triển các ứng dụng có hiệu suất cao, dễ bảo trì.

### Tính chất

Ngôn ngữ Dart được thiết kế với mục tiêu tối ưu cho việc xây dựng giao diện người dùng (UI) phía client. Dart có cú pháp gọn gàng, dễ đọc, được tối ưu hóa để hoạt động tốt trên nhiều nền tảng như di động, web và desktop. Dưới đây là những tính chất quan trọng thể hiện đặc trưng và ưu thế của Dart trong phát triển ứng dụng, đặc biệt là khi kết hợp với Flutter.

#### Hướng đối tượng hoàn toàn (Purely Object-Oriented)

Dart là một ngôn ngữ lập trình hướng đối tượng thuần túy, có nghĩa là mọi thứ trong Dart đều là một đối tượng, kể cả các kiểu dữ liệu cơ bản như int, double, bool, String. Điều này giúp Dart trở nên nhất quán về mặt cấu trúc và dễ dàng mở rộng thông qua các tính năng như kế thừa (inheritance), đóng gói (encapsulation), và đa hình (polymorphism).

Việc sử dụng mô hình hướng đối tượng giúp mã nguồn dễ tổ chức, tái sử dụng và bảo trì, đặc biệt phù hợp cho các dự án lớn hoặc các ứng dụng có cấu trúc phức tạp như hệ thống quản lý công việc.

#### Kiểu dữ liệu tĩnh và an toàn (Statically Typed & Type-safe)

Dart sử dụng hệ thống kiểu dữ liệu tĩnh, nghĩa là các biến phải có kiểu dữ liệu rõ ràng và cụ thể. Điều này giúp phát hiện lỗi ngay từ lúc biên dịch, tránh được nhiều lỗi runtime khó kiểm soát.

Ngoài ra, Dart còn hỗ trợ type inference – cho phép trình biên dịch tự động suy luận kiểu của biến từ giá trị gán ban đầu. Nhờ vậy, mã nguồn vẫn ngắn gọn nhưng vẫn đảm bảo tính an toàn khi biên dịch.

#### Hỗ trợ lập trình bất đồng bộ (Asynchronous Support)

Dart cung cấp hỗ trợ mạnh mẽ cho lập trình bất đồng bộ thông qua các cấu trúc như:

* Future<T>: xử lý giá trị trả về trong tương lai.
* async/await: giúp viết code bất đồng bộ dễ đọc và rõ ràng.
* Stream<T>: xử lý chuỗi dữ liệu bất đồng bộ (ví dụ: dữ liệu realtime từ Firebase).

Tính chất này rất quan trọng trong ứng dụng thực tế khi làm việc với cơ sở dữ liệu từ xa, gọi API, hoặc cập nhật dữ liệu thời gian thực như trong đề tài quản lý công việc.

#### Biên dịch linh hoạt: AOT và JIT (Compilation Modes)

Dart hỗ trợ hai chế độ biên dịch:

* AOT (Ahead-of-Time): Biên dịch mã nguồn thành mã máy trước khi chạy, giúp ứng dụng Flutter hoạt động với hiệu suất cao và tốc độ khởi chạy nhanh.
* JIT (Just-in-Time): Biên dịch ngay tại thời điểm chạy, cho phép sử dụng tính năng hot reload – rất tiện lợi trong giai đoạn phát triển và thử nghiệm.

Việc kết hợp cả hai phương thức biên dịch mang lại cho Dart sự linh hoạt, vừa tối ưu hiệu năng khi triển khai, vừa rút ngắn thời gian phát triển.

#### Cú pháp hiện đại và dễ học

Dart có cú pháp tương tự như Java, C# và JavaScript, nên rất dễ tiếp cận đối với sinh viên và lập trình viên đã quen với các ngôn ngữ phổ biến. Các khái niệm như class, hàm, điều kiện, vòng lặp,... đều rõ ràng, giúp lập trình viên nhanh chóng làm quen và phát triển ứng dụng hiệu quả.

#### Đa nền tảng và khả năng mở rộng cao

Mặc dù Dart được biết đến nhiều nhờ Flutter, nhưng thực tế ngôn ngữ này còn hỗ trợ phát triển trên nhiều nền tảng khác như:

* Web: Biên dịch Dart sang JavaScript để chạy trên trình duyệt.
* Mobile (Android/iOS): Kết hợp với Flutter.
* Desktop (Windows, macOS, Linux): Flutter hiện đã hỗ trợ đầy đủ.
* Command-line / Backend: Dart có thể dùng để viết các công cụ dòng lệnh hoặc backend đơn giản.

Tính chất đa nền tảng và khả năng mở rộng này giúp Dart trở thành một công cụ tiềm năng trong việc phát triển hệ thống phần mềm toàn diện từ frontend đến backend.

#### Tương thích hoàn hảo với Flutter

Dart được thiết kế riêng để hoạt động tối ưu với Flutter. Việc sử dụng Dart giúp Flutter đạt hiệu suất cao, hỗ trợ cơ chế dựng widget nhanh, animation mượt mà và quản lý trạng thái hiệu quả. Mọi thành phần trong ứng dụng Flutter – từ logic xử lý, kết nối dữ liệu đến giao diện người dùng – đều được viết bằng Dart.

Kết luận: Ngôn ngữ lập trình Dart là một công cụ hiện đại, mạnh mẽ và linh hoạt, mang nhiều tính chất nổi bật như hướng đối tượng thuần túy, hỗ trợ bất đồng bộ hiệu quả, cú pháp dễ học, và khả năng biên dịch linh hoạt. Việc sử dụng Dart trong đề tài “Ứng dụng quản lý công việc” không chỉ giúp phát triển ứng dụng nhanh chóng và hiệu quả, mà còn đảm bảo khả năng mở rộng, bảo trì, cũng như tính tương thích cao với Flutter và các dịch vụ như Firebase.

### Ưu điểm

* + Phát triển nhanh với hot reload: Giúp lập trình viên thấy ngay kết quả thay đổi mà không phải khởi động lại ứng dụng.
  + Cú pháp rõ ràng, dễ học: Cú pháp gần gũi với Java, C# nên dễ tiếp cận cho nhiều lập trình viên.
  + Tối ưu hiệu suất: Biên dịch AOT giúp ứng dụng chạy mượt mà và nhanh.
  + Tính linh hoạt trong kiểu dữ liệu: Hỗ trợ cả kiểu tĩnh và kiểu động, thuận tiện cho cả phát triển nhanh và phát triển quy mô lớn.
  + Thư viện chuẩn phong phú: Cung cấp nhiều thư viện sẵn có hỗ trợ đa dạng nhu cầu như UI, mạng, xử lý dữ liệu,...
  + Hỗ trợ lập trình bất đồng bộ tốt: Giúp xử lý các tác vụ cần chờ như truy xuất dữ liệu, gọi API dễ dàng và hiệu quả.

### Nhược điểm

Mặc dù Dart là một ngôn ngữ lập trình hiện đại, mạnh mẽ và có nhiều ưu điểm nổi bật, nhưng giống như bất kỳ công nghệ nào khác, Dart cũng tồn tại một số hạn chế và thách thức nhất định. Việc nhận diện các nhược điểm này giúp đánh giá khách quan hơn khi lựa chọn ngôn ngữ trong các dự án phần mềm.

* Cộng đồng còn hạn chế so với các ngôn ngữ phổ biến khác: Mặc dù Dart đang ngày càng được sử dụng rộng rãi nhờ Flutter, nhưng xét về quy mô cộng đồng, Dart vẫn chưa phổ biến bằng các ngôn ngữ như JavaScript, Python hay Java. Do đó:
* Tài nguyên học tập, thư viện mở (open-source) chưa phong phú bằng.
* Việc tìm kiếm giải pháp cho các vấn đề phức tạp đôi khi khó hơn.
* Số lượng lập trình viên thành thạo Dart ngoài thị trường còn ít.
* Phụ thuộc nhiều vào Flutter: Dart hiện tại được sử dụng gần như chủ yếu trong Flutter. Điều này dẫn đến việc:
* Ít công ty sử dụng Dart trong các lĩnh vực khác ngoài UI (frontend).
* Nếu không dùng Flutter, Dart gần như ít có cơ hội phát triển độc lập.
* Khó tìm việc nếu chỉ biết Dart mà không biết Flutter.
* Chưa phổ biến trong các hệ thống backend: Dart chưa được sử dụng rộng rãi cho các ứng dụng backend như Node.js (JavaScript), Spring Boot (Java) hay Django (Python). Dù Dart **có** khả năng viết backend (qua Dart Frog, shelf,...), nhưng chưa thực sự được lựa chọn nhiều trong môi trường doanh nghiệp. Điều này giới hạn Dart ở các ứng dụng client-side.
* Hạn chế về hiệu năng trên Web: Dù Dart hỗ trợ phát triển ứng dụng web thông qua việc biên dịch sang JavaScript, nhưng:
* Hiệu năng vẫn chưa thể vượt qua native JavaScript.
* Kích thước file biên dịch đôi khi khá lớn, gây chậm khi tải trang lần đầu.
* Một số tính năng của Flutter Web vẫn đang trong quá trình hoàn thiện, chưa ổn định như mobile.
* Công cụ hỗ trợ chưa phong phú như các ngôn ngữ lớn: Dart có IDE chính là Visual Studio Code và Android Studio, tuy nhiên các công cụ hỗ trợ nâng cao như:
* Debugger chuyên sâu
* Profiling tools nâng cao
* Static analysis/coverage cho dự án lớn vẫn còn hạn chế so với Java hoặc JavaScript. Điều này có thể ảnh hưởng đến năng suất trong những dự án lớn.
* Khả năng tương thích với thư viện bên thứ ba còn hạn chế: Một số thư viện bên ngoài được viết cho native Android hoặc iOS không thể sử dụng trực tiếp trong Dart/Flutter, do khác biệt về runtime và kiến trúc. Lúc này, lập trình viên cần tự viết code native (dùng platform channel), gây mất thời gian và phức tạp hơn.

Kết luận: Tuy có nhiều ưu điểm đáng kể, Dart vẫn tồn tại những hạn chế nhất định, đặc biệt là về phạm vi ứng dụng ngoài Flutter, quy mô cộng đồng, và hiệu năng web. Tuy nhiên, với sự phát triển không ngừng của Flutter và hệ sinh thái đi kèm, những nhược điểm này đang dần được cải thiện. Trong phạm vi đề tài này, Dart vẫn là lựa chọn hợp lý nhờ phù hợp với mục tiêu đa nền tảng, realtime và phát triển UI nhanh chóng.

### So sánh Dart với các ngôn ngữ lập trình phổ biến khác

#### So sánh Dart và JavaScript

| **Tiêu chí** | **Dart** | **JavaScript** |
| --- | --- | --- |
| Kiểu dữ liệu | Tĩnh (có type inference) | Động (dynamic) |
| Hướng đối tượng | Hướng đối tượng hoàn toàn | Hướng đối tượng nhưng không thuần túy |
| Bất đồng bộ | Hỗ trợ async/await, Future, Stream tốt | Hỗ trợ Promise, async/await |
| Biên dịch | JIT và AOT, biên dịch sang native hoặc JavaScript | Chạy trực tiếp trong trình duyệt |
| Ứng dụng chính | Flutter (mobile, desktop, web), CLI | Web frontend/backend, Node.js |
| Ưu điểm | Gọn, sạch, hiệu năng cao khi dùng Flutter | Phổ biến, hỗ trợ mọi trình duyệt |
| Nhược điểm | Kém phổ biến hơn, ít dùng ngoài Flutter | Cú pháp phức tạp nếu làm ứng dụng lớn |

Bảng 2.1 So sánh dart và JavaScript

#### So sánh Dart và Java

| **Tiêu chí** | **Dart** | **Java** |
| --- | --- | --- |
| Kiểu dữ liệu | Tĩnh, nhưng hỗ trợ suy luận kiểu (var, final) | Tĩnh, bắt buộc khai báo kiểu |
| Hướng đối tượng | Thuần túy | Hướng đối tượng mạnh |
| Đơn giản cú pháp | Cú pháp hiện đại, ngắn gọn hơn | Dài dòng, cần nhiều boilerplate |
| Bất đồng bộ | async/await, Future, Stream dễ dùng | Dùng Thread, ExecutorService, khó tiếp cận hơn |
| Ứng dụng chính | Flutter app, web, CLI | Backend (Spring), Android native |
| Hiệu năng | Tốt với AOT | Cao nhờ JVM, tối ưu lâu dài |
| Nhược điểm | Ít được dùng trong backend | Cú pháp dài, phức tạp, tốn bộ nhớ |

Bảng 2.2 So sánh Dart và Java

#### So sánh Dart và Kotlin

| **Tiêu chí** | **Dart** | **Kotlin** |
| --- | --- | --- |
| Mục tiêu thiết kế | UI đa nền tảng (Flutter) | Android native, backend (Ktor), đa nền tảng (KMM) |
| Cú pháp | Gần giống JavaScript/C# | Rất hiện đại, gần Scala |
| Hướng đối tượng | Có, đầy đủ tính năng | Có, hỗ trợ OOP + FP (lập trình hàm) |
| Null safety | Có, bắt buộc từ Dart 2.12 trở đi | Có, kiểm tra rất tốt |
| Bất đồng bộ | Future, async/await, Stream | Coroutine, suspend, flow – rất mạnh |
| Ứng dụng phổ biến | Flutter UI | Android, backend, multiplatform |
| Nhược điểm | Chưa mạnh về backend, phụ thuộc Flutter | Phức tạp hơn khi học, khó dùng ngoài Android |

Bảng 2.3 So sánh Dart và Kotlin

## Lý thuyết về Firebase

### Giới thiệu về Firebase

Firebase là một nền tảng phát triển ứng dụng di động và web (Mobile & Web Application Development Platform) cung cấp các dịch vụ backend dưới dạng Backend-as-a-Service (BaaS). Được phát triển ban đầu bởi Firebase Inc. năm 2011, sau đó được Google mua lại vào năm 2014, Firebase hiện nay trở thành một trong những công cụ mạnh mẽ nhất cho lập trình viên trong việc phát triển và triển khai ứng dụng nhanh chóng.

Firebase cho phép lập trình viên:

* Tạo ứng dụng mà không cần tự xây dựng backend truyền thống.
* Đồng bộ dữ liệu thời gian thực giữa các thiết bị.
* Tự động mở rộng (auto-scaling) theo lượng người dùng.
* Quản lý người dùng, gửi thông báo, lưu trữ file, xử lý logic backend, và phân tích hành vi người dùng, tất cả trong một nền tảng duy nhất.

### Lịch sử phát triển Firebase

Gần một thập niên trước, Firebase ra đời với tiền thân là Envolve. Đây là một nền tảng đơn giản chuyên cung cấp những API cần thiết để tích hợp tính năng chat vào trang web. Bên cạnh ứng dụng nhắn tin trực tuyến, Envolve còn được người dùng sử dụng để truyền và đồng bộ hóa dữ liệu cho những ứng dụng khác như các trò chơi trực tuyến,… Do đó, các nhà sáng lập đã tách biệt hệ thống nhắn tin trực tuyến và đồng bộ dữ liệu thời gian thực thành hai phần riêng biệt.

Trên cơ sở đó, năm 2012, Firebase ra đời với sản phẩm cung cấp là dịch vụ Backend-as-a-Service. Tiếp đến, vào năm 2014, Google mua lại Firebase và phát triển nó thành một dịch vụ đa chức năng được hàng triệu người sử dụng cho đến hiện nay.

### Cách thức hoạt động của Firebase

#### Firebase Realtime Database

Khi đăng ký một tài khoản trên Firebase để tạo ứng dụng, bạn đã có một cơ sở dữ liệu thời gian thực. Dữ liệu bạn nhận được dưới dạng JSON. Đồng thời nó cũng luôn được đồng bộ thời gian thực đến mọi kết nối client.

Đối với các ứng dụng đa nền tảng, tất cả các client đều sử dụng cùng một cơ sở dữ liệu. Nó được tự động cập nhật dữ liệu mới nhất bất cứ khi nào các lập trình viên phát triển ứng dụng. Cuối cùng, tất cả các dữ liệu này được truyền qua kết nối an toàn SSL có bảo mật với chứng nhận 2048 bit.

Trong trường hợp bị mất mạng, dữ liệu được lưu lại ở local. Vì thế khi có mọi sự thay đổi nào đều được tự động cập nhật lên Server của Firebase. Bên cạnh đó, đối với các dữ liệu ở local cũ hơn với Server thì cũng tự động cập nhật để được dữ liệu mới nhất.

#### Firebase Authentication

Firebase xây dựng hành động tự động đăng nhập cho ứng dụng bằng cách xác thực danh tính.

Hoạt động nổi bật của Firebase là xây dựng các bước xác thực người dùng bằng Email, Facebook, Twitter, GitHub, Google. Đồng thời cũng xác thực nặc danh cho các ứng dụng. Hoạt động xác thực có thể giúp thông tin cá nhân của người sử dụng được an toàn và đảm bảo không bị đánh cắp tài khoản.

#### Firebase Hosting

Firebase cung cấp các hosting được phân phối theo tiêu chuẩn SSL.

Cách thức hoạt động cuối cùng của Firebase được đề cập trong bài viết này là cung cấp các hosting. Hosting được phân phối qua tiêu chuẩn công nghệ bảo mật SSL từ mạng CDN(CDN viết tắt của Content Delivery Network là mạng lưới máy chủ lưu giữ bản sao của các nội dung tĩnh bên trong website và phân phối đến nhiều máy chủ PoP. Mạng lưới máy chủ CDN được đặt ở khắp mọi nơi trên toàn cầu. Từ PoP (Points of Presence), dữ liệu sẽ tiếp tục được gửi đến người dùng cuối. Thông qua CDN, bản sao nội dung trên máy chủ gần nhất sẽ được trả về cho người dùng khi họ truy cập website. )

Hoạt động này giúp lập trình viên tiết kiệm thời gian thiết kế, xây dựng và phát triển ứng dụng.

### Các dịch vụ của Firebase

Các lợi ích mà Google Firebase mang lại cho người dùng xuất phát từ những dịch vụ nổi bật của nó, những dịch vụ này dựa trên hai nhóm công cụ sản phẩm chính, cụ thể như sau:

Nhóm công cụ Firebase Develop & Test Your App: Nhóm công cụ này – hay còn gọi là công cụ phát triển và kiểm thử các ứng dụng được thiết kế, bao gồm các dịch vụ nổi bật sau:

* Realtime Database
* Crashlytics
* Cloud Firestore
* Authentication
* Cloud Functions
* Cloud Storage
* Hosting
* Test Lab for Android
* Performance Monitoring

#### Firebase Realtime Database

Firebase Realtime Database là một cơ sở dữ liệu NoSQL dạng cây JSON, cho phép lưu trữ và đồng bộ dữ liệu theo thời gian thực giữa các thiết bị và server. Nhờ sử dụng WebSocket, mọi thay đổi dữ liệu sẽ được cập nhật tức thì đến tất cả các client đang kết nối.

Dịch vụ này phù hợp với các ứng dụng cần cập nhật nhanh như chat, bảng công việc, giám sát IoT. Tuy dễ dùng và tốc độ nhanh, nhưng Realtime Database hạn chế về truy vấn, khó xử lý dữ liệu quan hệ, và không mở rộng tốt bằng Firestore.

#### Firebase Crashlytics

Firebase Crashlytics là công cụ giúp theo dõi và ghi nhận lỗi (crash) xảy ra trong ứng dụng theo thời gian thực. Khi ứng dụng gặp sự cố, Crashlytics sẽ gửi báo cáo chi tiết gồm: loại lỗi, dòng gây lỗi, thiết bị, hệ điều hành, và hành vi người dùng trước khi lỗi xảy ra.

Công cụ này hỗ trợ lập trình viên phát hiện sớm lỗi, ưu tiên xử lý lỗi nghiêm trọng và cải thiện độ ổn định ứng dụng. Crashlytics tích hợp tốt với Android, iOS, Flutter và cung cấp giao diện trực quan trên Firebase Console.

#### Cloud Firestore

Cloud Firestore là cơ sở dữ liệu NoSQL dạng collection-document, giúp lưu trữ và đồng bộ dữ liệu giữa client và server theo thời gian thực. Firestore hỗ trợ truy vấn linh hoạt, cấu trúc dữ liệu rõ ràng, mở rộng tốt, và hoạt động hiệu quả cả khi offline. Đây là giải pháp mạnh mẽ hơn Realtime Database, phù hợp cho ứng dụng quy mô lớn.

#### Firebase Authentication

Authentication là dịch vụ xác thực người dùng, hỗ trợ nhiều phương thức như email/mật khẩu, số điện thoại, Google, Facebook, Apple, v.v. Firebase giúp triển khai đăng nhập nhanh chóng, bảo mật cao và quản lý người dùng trực tiếp trên Firebase Console.

#### Cloud Functions

Cloud Functions cho phép viết các hàm backend chạy trên nền tảng serverless của Google. Hàm sẽ tự động được kích hoạt khi có sự kiện xảy ra (ví dụ: người dùng đăng ký, thêm dữ liệu, gửi thông báo...). Không cần quản lý máy chủ, dễ tích hợp với Firestore, Auth và các dịch vụ Firebase khác.

#### Cloud Storage

Cloud Storage là dịch vụ lưu trữ file như hình ảnh, video, tài liệu với khả năng phân quyền chi tiết. Dữ liệu được lưu trữ an toàn trên hạ tầng của Google Cloud, hỗ trợ tải lên/xuống nhanh, và tích hợp tốt với cả mobile lẫn web.

#### Firebase Hosting

Firebase Hosting cung cấp dịch vụ lưu trữ và triển khai website tĩnh với tốc độ cao, hỗ trợ HTTPS mặc định và dễ tích hợp CI/CD. Thích hợp cho các ứng dụng web, SPA, hoặc admin dashboard viết bằng HTML, React, Angular, Flutter Web,…

#### Test Lab for Android

Firebase Test Lab là nền tảng kiểm thử ứng dụng Android trên thiết bị thật qua đám mây. Lập trình viên có thể chạy tự động các bài kiểm tra để phát hiện lỗi giao diện, hiệu năng và khả năng tương thích trên nhiều dòng máy mà không cần thiết bị vật lý.

#### Performance Monitoring

Dịch vụ này giúp theo dõi hiệu suất ứng dụng như: thời gian tải trang, độ trễ mạng, hành vi người dùng. Firebase Performance Monitoring hỗ trợ lập trình viên cải thiện tốc độ và trải nghiệm người dùng bằng cách phân tích các điểm nghẽn hiệu năng.

### Ưu điểm của Firebase

Tạo tài khoản và sử dụng dễ dàng: Firebase cho phép lập trình viên bắt đầu chỉ với một tài khoản Google. Không cần cấu hình server hay thiết lập môi trường phức tạp, chỉ với vài bước đơn giản là đã có thể tạo dự án, tích hợp Firebase SDK vào ứng dụng Android, iOS, Flutter hoặc Web. Điều này rất thuận tiện cho sinh viên, lập trình viên cá nhân hoặc nhóm nhỏ bắt đầu xây dựng ứng dụng một cách nhanh chóng.

Tốc độ phát triển nhanh: Firebase cung cấp đầy đủ các dịch vụ backend như cơ sở dữ liệu, xác thực, lưu trữ tệp, gửi thông báo, xử lý backend bằng Cloud Functions,... Tất cả đều hoạt động như một hệ thống "backend-as-a-service" (BaaS), giúp tiết kiệm thời gian xây dựng server truyền thống, từ đó rút ngắn đáng kể thời gian phát triển ứng dụng.

Nhiều dịch vụ trong một nền tảng: Firebase là hệ sinh thái toàn diện với nhiều dịch vụ được tích hợp sẵn:

* Firebase Authentication (xác thực người dùng)
* Firestore / Realtime Database (cơ sở dữ liệu NoSQL)
* Cloud Storage (lưu trữ tệp)
* Firebase Cloud Messaging (thông báo đẩy)
* Hosting (triển khai website)
* Crashlytics, Performance Monitoring, A/B Testing, Remote Config,...  
  Việc này giúp lập trình viên không cần kết nối nhiều bên thứ ba mà vẫn có đầy đủ tính năng.

Được cung cấp bởi Google: Là một nền tảng do Google phát triển và vận hành, Firebase đảm bảo hiệu suất ổn định, độ tin cậy cao và khả năng mở rộng tốt. Google cũng liên tục cập nhật và bổ sung thêm các công cụ hỗ trợ mới như App Check, App Distribution, hay tích hợp với Google Analytics và BigQuery.

Tập trung vào phát triển giao diện người dùng: Firebase lo toàn bộ phần backend, giúp lập trình viên tập trung phát triển phần frontend như thiết kế giao diện, tối ưu trải nghiệm người dùng và nâng cao tính năng mà không cần lo hạ tầng máy chủ, CSDL, bảo mật kết nối,...

Firebase không có máy chủ: Firebase hoạt động theo mô hình "serverless", tức là không cần máy chủ truyền thống. Các hàm backend có thể xử lý bằng Cloud Functions, tự động chạy khi có sự kiện. Tất cả dịch vụ được vận hành bởi hạ tầng đám mây của Google. Điều này giúp giảm chi phí vận hành và tránh lỗi bảo trì máy chủ.

Học máy (Machine Learning): Firebase tích hợp dịch vụ ML Kit – cho phép sử dụng các chức năng học máy như:

* Nhận diện văn bản từ ảnh
* Nhận diện khuôn mặt
* Quét mã vạch
* Dịch ngôn ngữ
* Phân loại hình ảnh ML Kit có thể chạy trên thiết bị (on-device) hoặc trên đám mây, hỗ trợ thêm tính năng thông minh cho ứng dụng mà không cần hiểu sâu về AI.

Tạo lưu lượng truy cập: Firebase hỗ trợ các công cụ tăng trưởng người dùng như:

* Cloud Messaging: gửi thông báo đẩy để giữ chân người dùng.
* Remote Config: cá nhân hóa trải nghiệm người dùng mà không cần cập nhật app.
* A/B Testing: thử nghiệm nhiều phiên bản giao diện/chức năng để chọn bản hiệu quả nhất.
* Analytics: theo dõi hành vi người dùng, đánh giá hiệu quả chiến dịch.

Tất cả giúp tạo tương tác, tăng lượt sử dụng, cải thiện hiệu suất ứng dụng.

Theo dõi lỗi Sao lưu: Firebase Crashlytics giúp tự động ghi lại các lỗi xảy ra trong ứng dụng, bao gồm:

* Loại lỗi
* Thiết bị, hệ điều hành
* Hành vi trước khi lỗi xảy ra
* Mức độ nghiêm trọng thông tin này giúp lập trình viên phát hiện và xử lý lỗi nhanh chóng, đảm bảo ứng dụng ổn định và chuyên nghiệp.

### Nhược điểm của Firebase

* Không phải là mã nguồn mở: Firebase là một nền tảng đóng do Google phát triển, nên người dùng không thể xem, sửa hoặc tùy biến mã nguồn hệ thống. Điều này gây khó khăn nếu muốn tự host, thay đổi hành vi backend hoặc sử dụng trong các môi trường yêu cầu minh bạch mã nguồn.
* Người dùng không có quyền truy cập mã nguồn: Toàn bộ dịch vụ như Firestore, Auth, Cloud Functions,... đều được Google quản lý. Lập trình viên chỉ tương tác qua SDK hoặc API, không thể can thiệp sâu vào cách hoạt động nội bộ của Firebase, gây hạn chế với dự án cần tùy biến đặc biệt.
* Firebase không hoạt động ở nhiều quốc gia: Firebase hoạt động chủ yếu tại các khu vực được Google Cloud hỗ trợ. Một số quốc gia có thể bị giới hạn hoặc truy cập chậm do vị trí máy chủ xa, hoặc do chính sách kiểm duyệt, ví dụ: Trung Quốc, Bắc Triều Tiên, Iran,... Điều này ảnh hưởng đến trải nghiệm người dùng toàn cầu.
* Chỉ hoạt động với Cơ sở dữ liệu NoSQL: Firebase chỉ cung cấp hai loại CSDL chính là Realtime Database và Cloud Firestore, đều là NoSQL. Điều này gây khó khăn cho các dự án quen dùng SQL, cần join nhiều bảng, hoặc yêu cầu dữ liệu quan hệ phức tạp.
* Truy vấn chậm: Các truy vấn trong Firestore hoặc Realtime Database không hỗ trợ các chức năng nâng cao như JOIN, GROUP BY, hoặc sub-query như trong SQL. Nếu thiết kế sai hoặc dữ liệu lớn, việc truy vấn có thể bị chậm và tốn chi phí.
* Không phải tất cả các dịch vụ Firebase đều miễn phí: Firebase chỉ miễn phí trong mức giới hạn (free tier). Khi vượt quá số request, dung lượng lưu trữ hoặc thông báo gửi đi, người dùng phải trả phí. Một số dịch vụ như Test Lab, App Distribution, Hosting lớn cũng tính tiền theo mức độ sử dụng.
* Firebase khá đắt và giá không ổn định: Firebase tính phí theo số lượng request, dữ liệu truyền tải, lưu trữ, thời gian xử lý Cloud Function,... Việc không kiểm soát tốt logic có thể khiến chi phí tăng đột biến. Ngoài ra, Google có thể thay đổi chính sách giá bất kỳ lúc nào, gây khó khăn trong lập kế hoạch tài chính.
* Chỉ chạy trên Google Cloud: Firebase được tích hợp chặt với Google Cloud Platform (GCP) và không thể chạy trên nền tảng đám mây khác như AWS hay Azure. Điều này gây ra hiện tượng vendor lock-in – khó chuyển sang nhà cung cấp khác nếu muốn rời Firebase.
* Thiếu Dedicated Servers và hợp đồng doanh nghiệp: Firebase không hỗ trợ Dedicated Servers (máy chủ riêng) hay tùy chọn hợp đồng theo yêu cầu doanh nghiệp lớn (Enterprise SLA). Điều này hạn chế với công ty cần đảm bảo thời gian hoạt động cao, tuân thủ bảo mật nghiêm ngặt hoặc yêu cầu pháp lý riêng biệt.

## Cơ sở dữ liệu NoSQL

### Khái niệm NoSQL

NoSQL (Not Only SQL) là một hệ thống cơ sở dữ liệu được thiết kế để lưu trữ và truy vấn dữ liệu có cấu trúc không cố định hoặc bán cấu trúc. Khác với cơ sở dữ liệu quan hệ (SQL) sử dụng bảng và ràng buộc chặt chẽ, NoSQL sử dụng các mô hình linh hoạt như: key-value, document, column, graph.

### Đặc điểm của NoSQL

* Không yêu cầu schema cố định: Các bản ghi (document) có thể có cấu trúc linh hoạt và mở rộng dễ dàng.
* Dễ mở rộng quy mô: Có thể scale-out theo chiều ngang để đáp ứng lượng người dùng lớn.
* Tốc độ truy cập cao: Phù hợp cho ứng dụng cần hiệu suất xử lý nhanh và thời gian thực.
* Dễ dàng tích hợp với các ứng dụng di động, web hiện đại.

### Phân loại chính của NoSQL

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Loại NoSQL** | |  | | --- | |  |  |  | | --- | | **Mô tả** | | |  | | --- | |  |  |  | | --- | | **Ví dụ** | |
| Document-based | Dữ liệu lưu dưới dạng tài liệu JSON | |  | | --- | |  |  |  | | --- | | Firestore, MongoDB | |
| Key-value store | |  | | --- | |  |  |  | | --- | | Lưu trữ đơn giản dạng khóa – giá trị | | |  | | --- | |  |  |  | | --- | | Redis, DynamoDB | |
| Column-based | |  | | --- | |  |  |  | | --- | | Lưu theo cột thay vì dòng | | |  | | --- | |  |  |  | | --- | | Cassandra, HBase | |
| Graph-based | |  | | --- | |  |  |  | | --- | | Lưu trữ dữ liệu có quan hệ phức tạp | | |  | | --- | |  |  |  | | --- | | Neo4j | |

Bảng 2.4 Phân loại chính của NoSQL

### So sánh NoSQL và SQL (truyền thống)

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu chí** | |  | | --- | | **SQL (Quan hệ)** | | **NoSQL** |
| Cấu trúc dữ liệu | Bảng – hàng – cột | Document, key-value,... |
| Schema | Cố định | Linh hoạt |
| Truy vấn | Ngôn ngữ SQL chuẩn | API / DSL riêng từng hệ |
| Quan hệ dữ liệu | Tốt | Kém hơn, cần xử lý thủ công |
| Mở rộng | Dọc (scale-up) | Ngang (scale-out) |
| Tính nhất quán | Cao (ACID) | Tuỳ hệ, thường là BASE |

Bảng 2.5 So sánh NoSQL và SQL (truyền thống)

### Ứng dụng của NoSQL trong hệ thống quản lý công việc

Lưu trữ linh hoạt các project, task, subtask, notification...

Phù hợp với app dùng Firebase Firestore vì:

* Hỗ trợ real-time
* Không cần thiết kế bảng cứng nhắc
* Dễ đồng bộ dữ liệu trên nhiều nền tảng

## Mô hình quản lý công việc

### Khái niệm quản lý công việc

Quản lý công việc (Task Management) là quá trình lập kế hoạch, tổ chức, phân công, theo dõi và kiểm soát quá trình thực hiện các nhiệm vụ trong một nhóm, tổ chức hoặc dự án. Đây là một phần thiết yếu của quản lý dự án, nhằm đảm bảo công việc được thực hiện đúng thời hạn, đúng người và đúng mục tiêu.

Trong thời đại số, việc quản lý công việc bằng các công cụ phần mềm giúp tăng hiệu suất làm việc, giảm sai sót và đảm bảo minh bạch trong phân công và đánh giá kết quả.

### Cấu trúc mô hình quản lý công việc

Một hệ thống quản lý công việc hiện đại thường bao gồm các thành phần sau:

#### Dự án (Project)

* Là tập hợp các công việc có cùng mục tiêu, thời hạn và nhóm thực hiện.
* Có thể có nhiều quản lý và nhân viên tham gia.
* Trạng thái tổng thể có thể là Đang thực hiện, Đã hoàn thành,...

#### Công việc (Task)

* Là đơn vị công việc chính trong một dự án.
* Có thể chứa mô tả, hạn hoàn thành, người thực hiện.
* Mỗi task thường được gắn với một trạng thái trong quy trình (To Do, Doing, Done, Complete).

#### Công việc con (Subtask)

* Là phần việc chi tiết nằm trong một task.
* Được sử dụng khi task phức tạp, cần chia nhỏ cho nhiều người xử lý.
* Có trạng thái riêng biệt như: Bàn giao, Đang làm, Hoàn thành, Tạm dừng, Hủy bỏ.

#### Người dùng và phân quyền

* Admin: Quản trị toàn bộ hệ thống, tạo user, tạo project, phân quyền.
* Quản lý: Tạo và phân công task, theo dõi tiến độ của nhân viên.
* Nhân viên: Nhận task, thực hiện và cập nhật tiến độ subtask.

#### Trạng thái tiến độ

Được sử dụng để biểu diễn tình trạng của task/subtask theo thời gian thực.

Hệ thống sử dụng mô hình Kanban để hiển thị trực quan:

* Task: To Do → Doing → Done → Complete
* Subtask: Bàn giao → Đang làm → Hoàn thành → Tạm dừng → Hủy bỏ

### Luồng xử lý công việc

* Admin tạo tài khoản người dùng (quản lý, nhân viên).
* Admin hoặc Quản lý tạo dự án và thêm thành viên vào dự án.
* Quản lý tạo các task và gán người thực hiện.
* Nhân viên nhận thông báo, xem task, có thể tạo subtask để chia nhỏ phần việc.
* Trạng thái task/subtask được cập nhật theo tiến độ làm việc thực tế.
* Khi tất cả subtask hoàn thành, task chính có thể được chuyển sang trạng thái Done hoặc Complete.
* Thống kê, báo cáo, và log hoạt động sẽ được lưu lại để theo dõi kết quả.

### Ưu điểm của mô hình

* Quản lý tập trung: Tất cả task, subtask, người thực hiện, tiến độ đều tập trung trong một hệ thống.
* Phân quyền rõ ràng: Ai được làm gì được xác định cụ thể theo vai trò.
* Cập nhật real-time: Mỗi thay đổi đều phản ánh ngay trên giao diện các thành viên.
* Tăng hiệu suất nhóm: Nhờ có theo dõi, giao tiếp và phân công hợp lý.
* Lưu vết hoạt động: Lịch sử chỉnh sửa task/subtask giúp đánh giá hiệu quả làm việc.

### Mô hình phù hợp với Firebase và Flutter

* Firebase cho phép lưu task và subtask dạng document–collection.
* Trạng thái, người thực hiện, thời gian... được cập nhật real-time.
* Flutter dễ dàng triển khai giao diện kéo thả Kanban, theo dõi tiến độ sống động.
* Không cần xây dựng backend phức tạp – phù hợp nhóm sinh viên hoặc startup nhỏ.

## Giao thức Client – Server và REST API

### Mô hình Client – Server

Mô hình Client – Server là một kiến trúc phổ biến trong phát triển phần mềm hiện đại. Trong mô hình này, hệ thống được chia thành hai phần chính:

* Client (khách hàng): là các thiết bị đầu cuối hoặc phần mềm mà người dùng tương tác trực tiếp như ứng dụng web, ứng dụng di động.
* Server (máy chủ): là nơi xử lý các yêu cầu, truy xuất dữ liệu, thực hiện các nghiệp vụ và phản hồi lại kết quả cho client.

Mô hình này hoạt động dựa trên nguyên lý:

* Client gửi yêu cầu đến server thông qua mạng (thường là qua giao thức HTTP hoặc HTTPS).
* Server tiếp nhận, xử lý yêu cầu, truy xuất dữ liệu cần thiết (từ cơ sở dữ liệu), và trả về kết quả cho client.

Mô hình Client – Server giúp hệ thống trở nên tách biệt giữa phần giao diện người dùng và phần xử lý logic, từ đó dễ dàng bảo trì, nâng cấp, mở rộng và tái sử dụng.

Trong đề tài này, Flutter đóng vai trò là client, đảm nhiệm phần hiển thị giao diện và tương tác người dùng, còn Firebase đóng vai trò là server, cung cấp dịch vụ backend như xác thực người dùng, lưu trữ dữ liệu, xử lý truy vấn.

### Kiến trúc RESTful API

REST (Representational State Transfer) là một kiểu kiến trúc phần mềm được sử dụng rộng rãi trong việc thiết kế các hệ thống API hiện đại. REST không phải là một giao thức, mà là một quy tắc thiết kế dành cho việc tạo ra các dịch vụ Web sử dụng giao thức HTTP.

Một API tuân thủ kiến trúc REST được gọi là RESTful API. RESTful API thường sử dụng các phương thức HTTP cơ bản:

* GET: Lấy dữ liệu
* POST: Gửi dữ liệu mới
* PUT: Cập nhật toàn bộ dữ liệu
* PATCH: Cập nhật một phần dữ liệu
* DELETE: Xoá dữ liệu

Các thao tác RESTful API luôn hướng đến việc xử lý tài nguyên (resource), ví dụ như: người dùng, công việc, dự án,... Mỗi tài nguyên được biểu diễn bằng một đường dẫn (URL) duy nhất.

### Ưu điểm của REST API

REST API ngày càng được sử dụng phổ biến nhờ những đặc điểm nổi bật như:

* Tính đơn giản: REST sử dụng giao thức HTTP tiêu chuẩn nên rất dễ tích hợp vào các ứng dụng web và mobile.
* Tính mở rộng: REST hỗ trợ việc mở rộng hệ thống dễ dàng, có thể thêm tài nguyên hoặc chức năng mới mà không ảnh hưởng đến phần cũ.
* Khả năng tái sử dụng: Các API RESTful được thiết kế hướng resource nên có thể tái sử dụng ở nhiều nơi.
* Tương thích tốt: REST hoạt động tốt với nhiều nền tảng, công nghệ và ngôn ngữ khác nhau.
* Dễ học và dễ triển khai: So với các giao thức phức tạp như SOAP, REST đơn giản hơn nhiều.

### REST API trong Firebase Authentication

Firebase là một dịch vụ BaaS (Backend-as-a-Service) được cung cấp bởi Google. Một trong những dịch vụ cốt lõi của Firebase là Firebase Authentication, dùng để xác thực và quản lý người dùng. Ngoài việc tích hợp sẵn qua SDK, Firebase còn cung cấp REST API để quản lý người dùng thông qua các yêu cầu HTTP.

Ví dụ, để tạo một người dùng mới bằng REST API, có thể sử dụng endpoint:

https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=API\_KEY

Yêu cầu gửi qua phương thức POST, kèm theo nội dung là email, mật khẩu, và các thông tin khác dưới dạng JSON. Firebase sẽ phản hồi lại dữ liệu người dùng sau khi tạo thành công.

Trong ứng dụng Flutter của đề tài, REST API được sử dụng để tạo tài khoản người dùng (nhân viên, quản lý) từ phía Admin. Điều này giúp hệ thống có thể chủ động tạo user mà không cần qua giao diện mặc định của Firebase.

### Vai trò REST API trong ứng dụng

REST API đóng vai trò là cầu nối giữa giao diện người dùng Flutter và dữ liệu backend Firebase, giúp:

* Gửi yêu cầu xác thực, tạo user từ client đến server.
* Lấy dữ liệu người dùng đã xác thực hoặc thông tin xác thực token.
* Mở rộng hệ thống theo hướng linh hoạt, có thể kết nối thêm các hệ thống bên ngoài trong tương lai.

Mô hình kết hợp giữa Flutter (client) và Firebase (server thông qua REST API) không chỉ đơn giản trong triển khai mà còn đảm bảo tính bảo mật, hiệu suất cao và dễ mở rộng.

### Minh hoạ mô hình giao tiếp client – server

A graph with a black arrow pointing to the side

AI-generated content may be incorrect.

Ví dụ luồng:

- Flutter gửi POST tạo user qua REST API

- Firebase xác thực và trả kết quả JSON

# PHÂN TÍCH HỆ THỐNG

## Khảo sát hệ thống

### Khảo sát quy trình quản lý công việc

Trong môi trường làm việc hiện nay, việc quản lý và phân chia công việc đóng vai trò quan trọng nhằm đảm bảo tiến độ và hiệu quả công việc. Tuy nhiên, nhiều doanh nghiệp nhỏ hoặc nhóm làm việc vẫn đang quản lý công việc qua các công cụ truyền thống như Excel, Zalo, hoặc trao đổi trực tiếp, dẫn đến:

* Khó theo dõi tiến độ công việc.
* Dễ bỏ sót nhiệm vụ.
* Thiếu tính đồng bộ giữa các thành viên.
* Không có hệ thống phân quyền và ghi nhật ký thay đổi.

Dựa trên khảo sát, quy trình làm việc tại một số nhóm làm việc thường được thực hiện như sau:

* Người quản lý thông báo công việc cho từng nhân viên qua tin nhắn hoặc gọi điện.
* Nhân viên tự theo dõi và cập nhật tiến độ, thường không được lưu trữ trên hệ thống chung.
* Các thay đổi về công việc hoặc tiến độ được thông báo miệng hoặc nhắn lại trong nhóm chat.
* Việc tổng hợp tiến độ hoặc kiểm tra hiệu quả công việc được thực hiện thủ công, mất nhiều thời gian và thiếu chính xác.

Từ thực trạng nêu trên, nhận thấy nhu cầu cấp thiết cần có một hệ thống quản lý công việc chuyên nghiệp, hỗ trợ đa nền tảng, có khả năng đồng bộ dữ liệu theo thời gian thực, phân quyền người dùng rõ ràng và có tính linh hoạt cao trong sử dụng, nên em xây dựng ứng dụng quản lý công việc

### Phân tích bài toán

#### Bài toán đặt ra

Xây dựng một hệ thống quản lý công việc trên nền tảng di động và web, hỗ trợ người dùng có thể tạo, phân chia, theo dõi và cập nhật tiến độ công việc theo từng dự án, đảm bảo thông tin được đồng bộ hóa theo thời gian thực và được phân quyền rõ ràng theo vai trò người dùng. Hệ thống cần có giao diện trực quan, dễ sử dụng và phù hợp với quy mô của nhóm làm việc hoặc doanh nghiệp nhỏ.

#### Mục tiêu bài toán

* Hỗ trợ người dùng đăng nhập và phân quyền (Admin, Quản lý, Nhân viên).
* Quản lý danh sách project, mỗi project gồm nhiều task và subtask.
* Giao diện trực quan theo dạng bảng Kanban, cho phép theo dõi tiến độ dễ dàng.
* Cho phép người dùng thay đổi trạng thái công việc bằng cách kéo-thả giữa các cột.
* Ghi lại nhật ký thay đổi công việc, giúp truy vết người chỉnh sửa.
* Gửi thông báo (notification) đến người liên quan khi có sự thay đổi.
* Đồng bộ dữ liệu real-time giữa các thiết bị.
* Đảm bảo tính bảo mật và phân quyền truy cập.

#### Giao diện người dùng

* Đơn giản – trực quan: Giao diện được thiết kế tối giản, tập trung vào các chức năng chính.
* Phân quyền rõ ràng: Giao diện hiển thị theo vai trò người dùng (Admin, Quản lý, Nhân viên).
* Hiển thị theo Kanban: Công việc được trình bày theo bảng Kanban trực quan giúp dễ dàng quản lý.
* Phản hồi nhanh và realtime: Cập nhật trạng thái, thông báo thay đổi ngay tức thì.
* Đồng bộ đa nền tảng: Thiết kế giao diện phù hợp cho cả điện thoại và web.

#### Các yêu cầu phi chức năng

* Hiệu năng: Phản hồi nhanh, dữ liệu đồng bộ theo thời gian thực với Firebase Firestore.
* Bảo mật: Xác thực người dùng bằng Firebase Authentication, phân quyền rõ ràng, kiểm soát truy cập bằng Firebase Security Rules.
* Khả dụng: Hoạt động ổn định 24/7 khi có kết nối mạng, hỗ trợ nhiều thiết bị.
* Dễ sử dụng: Giao diện trực quan, thao tác đơn giản, phù hợp cho người dùng không chuyên.
* Mở rộng: Dễ tích hợp thêm tính năng (báo cáo, lịch, nhắc nhở…), đáp ứng tốt số lượng người dùng và dự án tăng cao.
* Tương thích: Hỗ trợ đa nền tảng (Android, iOS, Web) nhờ sử dụng Flutter.
* Bảo trì: Mã nguồn rõ ràng, dễ nâng cấp và sửa lỗi trong tương lai.

#### Phân tích và đặc tả yêu cầu hệ thống

Hệ thống cần đáp ứng các chức năng cơ bản sau:

* Đăng nhập, đăng xuất: Cho phép người dùng đăng nhập vào hệ thống thông qua tài khoản xác thực Firebase Authentication.
* Phân quyền người dùng: Hệ thống hỗ trợ ba vai trò: Admin, Quản lý, và Nhân viên. Mỗi vai trò có quyền hạn tương ứng.
* Quản lý tài khoản: Admin có quyền tạo mới người dùng, gán vai trò và ẩn (xóa mềm) tài khoản người dùng.
* Quản lý dự án: Admin và Quản lý có thể tạo, chỉnh sửa, và xem danh sách các dự án.
* Quản lý công việc (Task): Quản lý và Nhân viên có thể tạo công việc, cập nhật thông tin và thay đổi trạng thái công việc thông qua bảng Kanban.
* Quản lý công việc con (Subtask): Nhân viên có thể tạo subtask và cập nhật trạng thái của từng subtask.
* Ghi nhận lịch sử chỉnh sửa: Hệ thống ghi lại lịch sử chỉnh sửa của task, bao gồm: thời gian, người chỉnh sửa, và nội dung thay đổi.
* Thông báo: Khi có thay đổi liên quan đến người dùng, hệ thống gửi thông báo thông qua Firebase Cloud Messaging.
* Giao diện Kanban: Các công việc trong dự án được hiển thị dưới dạng bảng Kanban với các cột trạng thái: To Do, Doing, Done, Complete.
* Tìm kiếm, lọc dữ liệu: Người dùng có thể tìm kiếm và lọc công việc theo trạng thái, người thực hiện hoặc dự án.

## Biểu đồ phân rã chức năng

A diagram of a company

AI-generated content may be incorrect.

Hình 3.1 Biểu đồ phân rã chức năng của hệ thống

### Biểu đồ Use Case

#### Biểu đồ Use Case tổng quát

A diagram of a diagram

AI-generated content may be incorrect.

Hình 3.2 Biểu đồ Use Case tổng quát

#### Biểu đồ Use Case đăng nhập

**A screen shot of a computer screen

AI-generated content may be incorrect.**

Hình 3.3 Biểu đồ Use Case đăng nhập

* Mô tả: Use Case này mô tả quá trình người dùng đăng nhập vào hệ thống bằng tài khoản đã được cấp. Sau khi xác thực thông tin, người dùng sẽ được chuyển đến giao diện phù hợp với vai trò của mình (Admin, Quản lý, Nhân viên).
* Actor: Người dùng
* Tiền điều kiện: Người dùng đã được tạo tài khoản và có thông tin đăng nhập hợp lệ.
* Luồng chính:
  + Người dùng mở ứng dụng và chọn chức năng đăng nhập.
  + Nhập số điện thoại và mật khẩu.
  + Hệ thống kiểm tra thông tin đăng nhập qua Firebase Authentication.
  + Nếu thông tin hợp lệ, người dùng được chuyển đến màn hình chính.
* Luồng phụ:
  + Nếu thông tin không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.
* Hậu điều kiện: Người dùng được đăng nhập thành công vào hệ thống và sử dụng các chức năng theo phân quyền.

#### Biểu đồ Use Case quản lý dự án

A diagram of a chat

AI-generated content may be incorrect.

Hình 3.4 Biểu đồ Use Case quản lý dự án

Tác nhân: Admin  
Mô tả: Use Case cho phép xem, thêm, sửa, xóa người dùng và quản lý người thực hiện trong các dự án.  
Điều kiện trước: Admin đã đăng nhập vào hệ thống.

Dòng sự kiện chính:  
Người sử dụng chọn kiểu tác động: thêm, sửa, xóa dự án.

* Thêm dự án: Chọn “Thêm dự án”, hệ thống hiển thị giao diện nhập thông tin và danh sách dự án. Người dùng nhập thông tin và chọn người thực hiện. Nếu thành công, hệ thống thông báo và lưu dữ liệu. Nếu sai, thực hiện dòng rẽ nhánh A1.
* Sửa dự án: Hệ thống hiển thị danh sách dự án. Chọn dự án cần sửa, nhập thông tin cần thay đổi. Nếu thành công, hệ thống thông báo và lưu dữ liệu. Nếu sai, thực hiện dòng rẽ nhánh A1.
* Xóa dự án: Hệ thống hiển thị danh sách dự án. Chọn dự án cần xóa. Nếu thành công, hệ thống thông báo và cập nhật trạng thái xóa mềm. Nếu sai, thực hiện dòng rẽ nhánh A1.

Dòng rẽ nhánh A1: Hệ thống thông báo dữ liệu không hợp lệ, yêu cầu nhập lại, quay lại bước 1 dòng chính.

Hậu điều kiện: Thông tin dự án được cập nhật trong cơ sở dữ liệu.

#### Biểu đồ Use Case quản lý công việc (task)

**A diagram with text and images

AI-generated content may be incorrect.**

Hình 3.5 Biểu đồ Use Case quản lý công việc

Tác nhân: Quản lý

Mô tả: Use Case cho phép xem, thêm, sửa, xóa, phân công và cập nhật trạng thái công việc trong hệ thống.

Điều kiện trước: Quản lý đã đăng nhập vào hệ thống và đã được phân công vào dự án.

Dòng sự kiện chính:

Người sử dụng chọn kiểu tác động: thêm, sửa, xóa, phân công, cập nhật trạng thái công việc.

* Thêm công việc: Chọn “Thêm công việc”, hệ thống hiển thị giao diện nhập thông tin và danh sách công việc. Người dùng nhập thông tin. Nếu thành công, hệ thống thông báo và lưu dữ liệu. Nếu sai, thực hiện dòng rẽ nhánh A1.
* Sửa công việc: Hệ thống hiển thị danh sách công việc. Chọn công việc cần sửa, nhập thông tin cần thay đổi. Nếu thành công, hệ thống thông báo và lưu dữ liệu. Nếu sai, thực hiện dòng rẽ nhánh A1.
* Xóa công việc: Hệ thống hiển thị danh sách công việc. Chọn công việc cần xóa. Nếu thành công, hệ thống thông báo và cập nhật trạng thái xóa mềm. Nếu sai, thực hiện dòng rẽ nhánh A1.
* Phân công công việc: Chọn công việc, chọn nhân viên từ danh sách, gán cho công việc. Nếu thành công, hệ thống thông báo và lưu dữ liệu. Nếu sai, thực hiện dòng rẽ nhánh A1.
* Cập nhật trạng thái: Chọn công việc cần cập nhật, thay đổi trạng thái (To Do, Doing, Done, Complete). Nếu thành công, hệ thống thông báo và lưu dữ liệu. Nếu sai, thực hiện dòng rẽ nhánh A1.

Dòng rẽ nhánh A1:

Hệ thống thông báo dữ liệu không hợp lệ, yêu cầu nhập lại, quay lại bước 1 dòng chính.

Hậu điều kiện:

Thông tin công việc được cập nhật trong cơ sở dữ liệu, trạng thái và phân công phản ánh đúng trên hệ thống.

#### Biểu đồ Use Case quản lý subtask

**A diagram of a diagram

AI-generated content may be incorrect.**

Hình 3.6 Biểu đồ Use Case quản lý subtask

Tác nhân: Nhân viên

Mô tả: Use Case cho phép xem, tạo và cập nhật trạng thái subtask thuộc công việc được phân công.

Điều kiện trước: Nhân viên đã đăng nhập và được phân công vào công việc.

Dòng sự kiện chính:

Người sử dụng chọn kiểu tác động: tạo mới, xem chi tiết, cập nhật trạng thái subtask.

* Tạo subtask: Chọn công việc đang làm, chọn “Tạo subtask”, hệ thống hiển thị giao diện nhập thông tin. Người dùng nhập tên subtask, mô tả. Trạng thái mặc định là “Bàn giao”. Nếu thành công, hệ thống thông báo và lưu dữ liệu. Nếu sai, thực hiện dòng rẽ nhánh A1.
* Xem subtask: Hệ thống hiển thị danh sách các subtask thuộc công việc. Chọn subtask để xem thông tin chi tiết.
* Cập nhật trạng thái subtask: Chọn subtask, chọn trạng thái mới từ danh sách có sẵn (Đang làm, Hoàn thành, Tạm dừng, Hủy bỏ). Nếu thành công, hệ thống thông báo và lưu dữ liệu. Nếu sai, thực hiện dòng rẽ nhánh A1.

Dòng rẽ nhánh A1: Hệ thống thông báo dữ liệu không hợp lệ, yêu cầu nhập lại, quay lại bước 1 dòng chính.

Hậu điều kiện: Thông tin subtask được cập nhật trong cơ sở dữ liệu; trạng thái hiển thị đúng với tiến độ thực tế.

#### Biểu đồ Use Case quản lý thông báo và lịch sử

**A diagram of a diagram

AI-generated content may be incorrect.**

Hình 3.7 Biểu đồ Use Case quản lý thông báo và lịch sử

* Tác nhân: Quản lý / Nhân viên
* Mô tả: Use Case cho phép người dùng xem danh sách thông báo và lịch sử chỉnh sửa liên quan đến các công việc được phân công.
* Điều kiện trước: Người dùng đã đăng nhập vào hệ thống.
* Dòng sự kiện chính:
* Người dùng chọn tab “Thông báo” hoặc “Lịch sử”.
* Xem thông báo:  
  Hệ thống hiển thị danh sách thông báo (theo assigneeId), gồm:
  + Nội dung thông báo
  + Thời gian
  + Trạng thái (chưa đọc / đã đọc)

Người dùng có thể đánh dấu là đã đọc hoặc xóa thông báo. Nếu thành công, hệ thống cập nhật trạng thái. Nếu lỗi, thực hiện dòng rẽ nhánh A1.

* Xem lịch sử chỉnh sửa:  
  Hệ thống hiển thị log các chỉnh sửa công việc liên quan, bao gồm:
  + Tên công việc
  + Người chỉnh sửa
  + Thời gian
  + Nội dung thay đổi
  + Trạng thái trước và sau

Dòng rẽ nhánh A1:

Nếu không thể tải dữ liệu, hệ thống hiển thị lỗi và cho phép người dùng thử lại.

Hậu điều kiện:

Người dùng đã nắm được các thông báo mới và lịch sử thay đổi liên quan đến công việc của họ.

#### Biểu đồ Use Case giao diện người dùng

A diagram with black ovals and white text

AI-generated content may be incorrect.

Hình 3.8 Biểu đồ Use Case giao diện người dùng

* Tác nhân: Admin
* Mô tả: Use Case cho phép xem, thêm, sửa, xóa (ẩn), và gán vai trò cho người dùng trong hệ thống.
* Điều kiện trước: Admin đã đăng nhập vào hệ thống.

Dòng sự kiện chính:

Người sử dụng chọn kiểu tác động: thêm, sửa, xóa (ẩn), gán vai trò người dùng.

* Thêm người dùng: Chọn “Thêm người dùng”, hệ thống hiển thị giao diện nhập thông tin và danh sách người dùng. Người sử dụng nhập thông tin (họ tên, số điện thoại, email, v.v.). Nếu thành công, hệ thống thông báo và lưu dữ liệu. Nếu sai, thực hiện dòng rẽ nhánh A1.
* Sửa người dùng: Hệ thống hiển thị danh sách người dùng. Chọn người cần sửa, nhập các thông tin cần thay đổi. Nếu thành công, hệ thống thông báo và lưu dữ liệu. Nếu sai, thực hiện dòng rẽ nhánh A1.
* Xóa (ẩn) người dùng: Hệ thống hiển thị danh sách người dùng. Chọn người cần ẩn, hệ thống đánh dấu là “đã xóa” (xóa mềm). Nếu thành công, hệ thống thông báo. Nếu sai, thực hiện dòng rẽ nhánh A1.
* Gán vai trò: Chọn người dùng, chọn vai trò (Quản lý / Nhân viên). Nếu gán thành công, hệ thống thông báo và cập nhật dữ liệu. Nếu sai, thực hiện dòng rẽ nhánh A1.

Dòng rẽ nhánh A1:

Hệ thống thông báo dữ liệu không hợp lệ, yêu cầu nhập lại, quay lại bước 1 dòng chính.

Hậu điều kiện:

Thông tin người dùng được cập nhật trong cơ sở dữ liệu và hiển thị chính xác trên hệ thống.

### Biểu đồ tuần tự

#### Biểu đồ tuần tự chức năng đăng nhập

**A diagram of a diagram

AI-generated content may be incorrect.**

Hình 3.9 Biểu đồ tuần tự chức năng đăng nhập

#### Biểu đồ tuần tự chức năng quản lý dự án

**A diagram of a project

AI-generated content may be incorrect.**

Hình 3.10 Biểu đồ tuần tự chức năng quản lý dự án

#### Biểu đồ tuần tự chức năng quản lý công việc

**A diagram of a diagram

AI-generated content may be incorrect.**

Hình 3.11 Biểu đồ tuần tự chuức năng quản lý công việc

#### Biểu đồ tuần tự chức năng quản lý subtask

**A diagram of a graph

AI-generated content may be incorrect.**

Hình 3.12 Biểu đồ tuần tự chức năng quản lý subtask

#### Biểu đồ tuần tự chức năng quản lý thông báo

**A diagram of a diagram

AI-generated content may be incorrect.**

Hình 3.13 Biểu đồ tuần tự chức năng quản lý thông báo

#### Biểu đồ tuần tự chức năng quản lý lịch sử

**A diagram of a graph

AI-generated content may be incorrect.**

Hình 3.14 Biểu đồ tuần tự chức năng quản lý lịch sử

### Thiết kế cơ sở dữ liệu sử dụng Firebase Firestore

**A computer screen shot of a computer program

AI-generated content may be incorrect.**

Hình 3.15 Sơ đồ thiết kế cơ sở dữ liệu Firebase Firestore

### Thiết kế chi tiết các collection

#### Danh sách các collection

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên mô tả** | **Chức năng** |
| 1 | users | Lưu trữ thông tin người dùng của hệ thống, gồm quản trị viên, quản lý và nhân viên. |
| 2 | projects | Lưu thông tin các dự án, bao gồm người quản lý, mô tả, tiến độ và danh sách quy trình. |
| 3 | tasks | Lưu trữ các công việc thuộc từng dự án, bao gồm người thực hiện, trạng thái, log thay đổi. |
| 4 | notifications | Gửi thông báo tới người dùng về các sự kiện như được giao việc, cập nhật task. |

Bảng 3.1 Bảng danh sách các cơ sở dữ liệu

#### Collection Users

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Trường** | **Kiểu dữ liệu** | **Mô tả** |
| 1 | uid | string | Mã người dùng |
| 2 | fullName | string | Họ tên đầy đủ |
| 3 | email | string | Email đăng nhập |
| 4 | phone | string | Số điện thoại |
| 5 | address | string | Địa chỉ cá nhân |
| 6 | role | string | Vai trò |
| 7 | isVisible | boolean | Ẩn/hiện người dùng |

Bảng 3.2 Bảng Collection Users

#### Collection Projects

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Trường** | **Kiểu dữ liệu** | **Mô tả** |
| 1 | projectId | string | ID dự án |
| 2 | name | string | Tên dự án |
| 3 | description | string | Mô tả dự án |
| 4 | assignee | string | ID người quản lý |
| 5 | assigneeName | string | Tên người quản lý |
| 6 | createdAt | timestamp | Ngày tạo |
| 7 | deadline | timestamp | Hạn hoàn thành |
| 8 | isCompleted | boolean | Trạng thái hoàn thành |
| 9 | processes | array[string] | Danh sách process: "To Do", "Doing", "Done", "Complete" |

Bảng 3.3 Bảng Collection Projects

#### Collection Tasks

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Trường** | **Kiểu dữ liệu** | **Mô tả** |
| 1 | taskId | string | Mã công việc |
| 2 | name | string | Tên công việc |
| 3 | description | string | Nội dung công việc |
| 4 | priority | number | Mức độ ưu tiên(1: cao nhất) |
| 5 | status | string | Trạng thái hiện tại: To Do, Doing, Done, ... |
| 6 | assigneeId | string | ID người được giao |
| 7 | assigneeName | string | Tên người được giao |
| 8 | projectId | string | ID dự án |
| 9 | processId | string | |  | | --- | |  |   ID của process trong dự án |
| 10 | createdAt | timestamp | Ngày tạo task |
| 11 | dueDate | timestamp | Hạn hoàn thành |
| 12 | completedAt | timestamp | Ngày hoàn thành (nullable) |
| 13 | logs | array[map] | |  |  | | --- | --- | |  | Danh sách log: {action, timestamp, user} | |

Bảng 3.4 Bảng Collection Tasks

#### Collection Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Trường** | **Kiểu dữ liệu** | **Mô tả** |
| 1 | notificationId | string | Mã thông báo |
| 2 | assigneeId | string | ID người nhận |
| 3 | title | string | Tiêu đề thông báo |
| 4 | description | string | Nội dung thông báo |
| 5 | timestamp | timestamp | Thời điểm tạo thông báo |

Bảng 3.5 Bảng Collection Notifications

#### Collection Subtasks

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Trường** | **Kiểu dữ liệu** | **Mô tả** |
| 1 | subtaskId | string | ID subtask |
| 2 | createdAt | timestamp | Ngày giờ tạo |
| 3 | createdBy | string | ID người tạo task |
| 4 | createdByName | string | Email người tạo |
| 5 | status | string | Trạng thái subtask |
| 6 | taskId | string | ID của công việc cha mà subtask thuộc |
| 7 | title | string | Tiêu đề subtask |

Bảng 3.6 Bảng Collection Subtasks

## Thiết kế và xây dựng ứng dụng

### Giao diện trang đăng nhập

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.16 Giao diện trang đăng nhập

### Giao diện danh sách người dùng

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.17 Giao diện danh sách người dùng

### Giao diện form tạo dự án

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.18 Giao diện form tạo dự án

### Giao diện form tạo công việc

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.19 Giao diện form tạo công việc

### Giao diện cá nhân và công việc

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.20 Giao diện cá nhân và công việc

### Giao diện thông tin cá nhân

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.21 Giao diện thông tin cá nhân

### Giao diện quản lý công việc

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.22 Giao diện quản lý công việc

### Giao diện chi tiết task

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.23 Giao diện chi tiết task

### Giao diện tìm kiếm công việc

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.24 Giao diện tìm kiếm công việc

### Giao diện thông báo

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.25 Giao diện thông báo

### Giao diện thống kê

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.26 Giao diện thông báo

## Kết luận

### Nhận xét

Trong quá trình thực hiện đồ án, em đã có cơ hội tiếp cận và áp dụng các công nghệ hiện đại như Flutter và Firebase để xây dựng một ứng dụng quản lý công việc đa nền tảng. Hệ thống được thiết kế với các chức năng phân quyền rõ ràng (Admin, Quản lý, Nhân viên), cho phép theo dõi, phân chia và cập nhật tiến độ công việc một cách trực quan và tiện lợi.

Bên cạnh việc trau dồi kỹ năng lập trình, đồ án cũng giúp em hiểu rõ hơn về quy trình xây dựng một ứng dụng thực tế: từ thiết kế giao diện, tổ chức dữ liệu trong Firestore, cho đến xử lý logic nghiệp vụ và triển khai các chức năng đồng bộ, realtime.

Tuy còn một số điểm cần tối ưu và hoàn thiện thêm, nhưng qua đồ án này, em đã rút ra được nhiều kinh nghiệm quan trọng trong việc phát triển ứng dụng mobile/web, cũng như khả năng tư duy hệ thống và làm việc theo hướng sản phẩm.

### Ưu điểm

Sau khi xây dựng được ứng dụng quản lí công việc em đã đạt được một số ưu điểm sau:

* Ứng dụng hỗ trợ realtime, đa nền tảng (mobile, web) nhờ Flutter và Firebase.
* Phân quyền rõ ràng giữa Admin, Quản lý và Nhân viên.
* Giao diện trực quan, dễ sử dụng, có thể kéo-thả task giữa các trạng thái.
* Dữ liệu được đồng bộ tức thời và bảo mật qua Firebase Rules.
* Hệ thống thông báo và log giúp theo dõi thay đổi công việc hiệu quả.

### Hạn chế

Bên cạnh những kết quả khả quan đã đạt được, em nhận thấy còn tồn tại một số hạn chế như sau:

* Chưa hỗ trợ hoạt động tốt khi mất kết nối mạng (offline).
* UI còn đơn giản, thiếu tuỳ biến (chưa có dark mode, đa ngôn ngữ).
* Chưa có kiểm thử tự động và triển khai CI/CD.
* Một số thao tác xử lý logic vẫn phụ thuộc phía client.
* Khả năng mở rộng lớn cần tối ưu thêm nếu triển khai thực tế.

### Hướng phát triển

Sau khi hoàn thành đề tài và xây dựng được “ứng dụng quản lý công việc” em sẽ tiếp tục nghiên cứu và phát triển ứng dụng này nhằm tăng các tính năng và tối ưu hóa tốc độ xử lý để đem lại hiệu quả cao hơn.

* Cải tiến chức năng cũ
* Thêm một số tính năng mới như thông báo đẩy khi có task mới, bị giao việc hoặc cập nhật trạng thái ngay cả khi không mở ứng dụng; đính kèm file tài liệu, hình ảnh; hiển thị công việc dạng lịch, giúp quản lý theo ngày/tuần/tháng.

**PHỤ LỤC**

* + Link sourcecode: https://github.com/congdoan99/DATN-3-2025
  + Tên WebSite: https://lecongdoan.io.vn/
  + Tài khoản admin: Tên đăng nhập “doan@gmail.com”, mật khẩu “123456”
  + Tài khoản người dùng: Tên đăng nhập “poro@gmail.com”, mật khẩu “654321”

**TÀI LIỆU THAM KHẢO**

1. S. Alessandria, ‘Flutter Projects: A practical, project-based guide to building real-world cross-platform mobile applications and games’. Packt Publishing, 2020.
2. N. Ahmad, ‘Beginning Flutter: A Hands On Guide To App Development’. Apress, 2021.
3. xeladu, ‘The Flutter Firebase Compendium’, 3rd ed., 2024.
4. R. Agarwal and Code01, ‘Firebase for Flutter Developers: Authentication, Database and Storage Mastery’, 2023.
5. F. Hussain and K. Hussain, ‘Building Mobile Magic: Integrating Flutter with Firebase’, 2024.
6. N. Nursobah, M. I. Saad và J. A. J. Kansil, ‘Implementation of the Flutter Framework for Developing an E‑Commerce Application,’ TEPIAN, vol. 5, no. 4, pp. 127–135, 2024.
7. D. A. Alfahri và A. Widarma, ‘Implementation of Flutter and Firebase in Developing a Mobile News Portal Application,’ Bigint Computing Journal, vol. 3, no. 1, pp. 50–57, Jan. 2025.
8. I. K. Wairooy, I. Dillwyn, K. Putra và A. Lay, ‘Development of Mobile QR Warehouse Management Application Based on Flutter and Firebase,’ EMACS, 2025.
9. S. Shrey, ‘Flutter & Firebase BootCamp,’ slide PDF, Feb. 2023.
10. I. Y. Nugraha, ‘Flutter Firebase PDF,’ slide PDF, circa 2021–2022.
11. T. Bailey and A. Biessek, ‘Flutter for Beginners – Third Edition’. Packt Publishing, 2023.
12. S. C. Shukla, ‘Flutter Zero to Hero – 2023’. Independently published, 2023.
13. F. Hussain and K. Hussain, ‘Building Mobile Magic: Integrating Flutter with Firebase’. Sonar Publishing, 2024.
14. S. Alessandria and B. Kayfitz, ‘Flutter Cookbook’. Packt Publishing, 2021.
15. A. Biessek, \*Flutter for Beginners\*. O’Reilly, 2022.
16. R. Rose, ‘Flutter and Dart Cookbook: Developing Full‑Stack Applications for the Cloud’. O’Reilly, 2023.
17. Google. Flutter Documentation. Truy cập tại: https://docs.flutter.dev
18. Google. Firebase Documentation. Truy cập tại: https://firebase.google.com/docs
19. M. Funk and Y. Zhang, ‘Coding Art’, 2nd ed. Apress, 2023.
20. FlutterFire. Official Firebase plugins for Flutter. Truy cập tại: https://firebase.flutter.dev
21. Reso Coder. Flutter Firebase tutorials (YouTube). Truy cập tại: <https://www.youtube.com/c/ResoCoder>
22. Medium. Building Real-Time Apps with Flutter & Firestore. Truy cập tại: <https://medium.com> (từ khóa: Flutter Firestore real-time)
23. Stack Overflow. Hỏi đáp về Flutter và Firebase. Truy cập từ: <https://stackoverflow.com>