**ỦY BAN NHÂN DÂN TP HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**



**Họ và tên sinh viên : Dương Ngọc Nguyên**

**BÁO CÁO**

**THỰC TẬP TỐT NGHIỆP**

**Công ty thực tập :** **Công ty TMA Solutions**

**Chuyên gia hướng dẫn : Huỳnh Trọng Tuyến**

**Giảng viên hướng dẫn : PGS.TS Nguyễn Tuấn Đăng**

***TP. Hồ Chí Minh, tháng 08 năm 2025***

# MỤC LỤC

[MỤC LỤC I](#_3yk4x4jb5oo3)

[NHẬN XÉT CỦA CHUYÊN GIA DOANH NGHIỆP IV](#_g5pi907thb4z)

[NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN V](#_cofi32uac1px)

[LỜI MỞ ĐẦU V](#_4qsc5r5ckm7f)

[Chương 1:](#_9qsgzfz9aauj) GIỚI THIỆU 3

[**1.1.**](#_jrgeznf0wxmv) **Giới thiệu công ty thực tập 3**

[1.1.1.](#_cjs1kd318u7a) Tổng quan về Công ty Cổ phần Công nghệ AppsCyclone 3

[1.1.2.](#_vq3nktkj1eh2) Lĩnh vực hoạt động và phòng ban thực tập 3

[1.1.3.](#_x5ez8py07fe9) Chuyên gia hướng dẫn tại doanh nghiệp 3

[**1.2.**](#_d97shej0bw24) **Nhiệm vụ thực tập 4**

[1.2.1.](#_prr6v7vjai2y) Tên đề tài 4

[1.2.2.](#_roy54n7lb3ch) Mục tiêu đề tài 4

[1.2.3.](#_6mj89qqle1fz) Phạm vi công việc được giao 4

[**1.3.**](#_ca605d6smt3z) **Kết luận chương 1 4**

[Chương 2:](#_ri83kmq9h1sw) CƠ SỞ LÝ THUYẾT VÀ PHÂN TÍCH HỆ THỐNG 6

[**2.1.**](#_nqgvl2ajr72o) **Giai đoạn Chuẩn bị và Nghiên cứu Nền tảng (Tuần 1-2) 6**

[2.1.1.](#_94g305g83xja) Nghiên cứu Chuyên sâu về Công nghệ và Nền tảng Flutter 6

[2.1.2.](#_y9p3xt30ppsi) Thiết lập Môi trường và Cấu trúc Dự án Chuyên nghiệp 8

[**2.2.**](#_1yj9ibtz01s2) **Phân tích và Thiết kế Hệ thống 9**

[2.2.1.](#_41cb60jp3akz) Phân tích Yêu cầu 9

[2.2.2.](#_fsvi0slou32c) Thiết kế Kiến trúc Tổng quan 10

[2.2.3.](#_m2gxj3zhcs5o) Thiết kế Quản lý Trạng thái Chi tiết với BLoC 11

[2.2.4.](#_eyzhke30ypx4) Thiết kế Cơ sở dữ liệu trên Cloud Firestore 12

[2.2.5.](#_urusvth9ola0) Thiết kế Điều hướng Chi tiết với go\_router 13

[2.2.6.](#_uqn8qhe7rlyl) Thiết kế Giao diện (UI) và Trải nghiệm Người dùng (UX) 13

[**2.3.**](#_etnbsug8wf4h) **Triển khai chi tiết các chức năng và màn hình 14**

[2.3.1.](#_d0qk16hd25jg) Luồng Khởi động và Giới thiệu 14

[2.3.2.](#_ct69tf64q0hx) Hệ thống Xác thực và Quản lý Mật khẩu 17

[2.3.3.](#_eg8daix44kou) Chức năng Trò chuyện và Quản lý 23

[2.3.4.](#_banvjbutwy6w) Cài đặt và Tùy chỉnh 31

[2.3.5.](#_qj4cjio1llnh) Thách thức Kỹ thuật Chuyên sâu và Giải pháp Khắc phục 35

[**2.4.**](#_93udl0c9gdu3) **Phân tích và Nhận định Cá nhân 35**

[**2.5.**](#_mj4m7m8ibuqn) **Kết luận chương 2 36**

[Chương 3:](#_xsk4jsoq36lp) KẾT QUẢ THỰC TẬP 36

[**3.1.**](#_3x17q4ul96nv) **Tổng kết các kết quả cụ thể đã đạt được 37**

[3.1.1.](#_3ngrxs5k6zhm) Về mặt sản phẩm 37

[3.1.2.](#_39ynnm28vr23) Về mặt kiến thức và kỹ năng 38

[3.1.3.](#_dmcsy3f5am1x) Về mặt tài liệu 39

[**3.2.**](#_votqvthucvwr) **Kết luận chương 3 49**

[Chương 4:](#_s6wp9wbxyy8l) KẾT LUẬN VÀ KIẾN NGHỊ 50

[**4.1.**](#_qgbw2ny6bwx4) **Kết luận 50**

[**4.2.**](#_yv2y2jdzxywr) **Kiến nghị 50**

[4.2.1.](#_eeq5a9h9wxm8) Hướng phát triển của đề tài 50

[4.2.2.](#_3gyo1xnjlbl7) Kiến nghị với doanh nghiệp (Công ty Apps Cyclone) 51

[4.2.3.](#_dbekn6exld7b) Kiến nghị với Khoa và Nhà trường (Khoa CNTT - ĐH Sài Gòn) 51

[TÀI LIỆU THAM KHẢO 52](#_w0zr34bvbfrn)

# MỤC LỤC HÌNH ẢNH

[Hình 2.1: BottomInputField khi đang đợi phản hồi từ AI 14](#_71u3z71ds50m)

[Hình 2.2: Màn hình splash 16](#_nar117s0b063)

[Hình 2.3: Màn hình onboarding 17](#_pl2ywvhp3mfy)

[Hình 2.4: Màn hinh Đăng ký 20](#_jt2pjvd9tjll)

[Hình 2.5: Màn hình đăng nhập 21](#_7d8inu540awk)

[Hình 2.6: Widget quên mật khẩu 22](#_g56kbflcodza)

[Hình 2.7: Màn hình thay đổi mật khẩu 23](#_ubeh512tdjpp)

[Hình 2.8: Màn hình initial 26](#_zg56yey22jv0)

[Hình 2.9: Màn hình messageListView 28](#_4vuiziuoy86t)

[Hình 2.10:ConversationDrawer trước khi đăng nhập 30](#_ns7fali31hiw)

[Hình 2.11: ConversationDrawer sau khi đăng nhập 30](#_ypujal9tg7ov)

[Hình 2.12: Widget xác nhận xóa cuộc trò chuyện cụ thể 31](#_okqmclk7e16a)

[Hình 2.13: Màn hình cài đặt 32](#_2kawkkvr5o4)

# NHẬN XÉT CỦA CHUYÊN GIA DOANH NGHIỆP

......................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................... ................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................... ........................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

# NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................... ................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................... ................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................... ..........................................................................................................................................

# LỜI MỞ ĐẦU

Trong bối cảnh cuộc cách mạng công nghiệp 4.0 đang diễn ra mạnh mẽ, công nghệ thông tin đã và đang khẳng định vai trò không thể thiếu trong mọi lĩnh vực của đời sống. Nhận thức được tầm quan trọng của việc kết hợp giữa lý thuyết được trang bị ở nhà trường và kinh nghiệm làm việc thực tiễn, kỳ thực tập tốt nghiệp là cơ hội quý báu để sinh viên chúng em được tiếp cận với môi trường làm việc chuyên nghiệp, áp dụng kiến thức đã học vào các dự án thực tế, từ đó củng cố chuyên môn và định hướng cho sự nghiệp tương lai.

Trước hết, em xin trân trọng gửi lời cảm ơn sâu sắc đến Ban Giám hiệu Trường Đại học Sài Gòn, Ban Chủ nhiệm Khoa Công nghệ thông tin đã tạo điều kiện và giới thiệu chúng em đến với các doanh nghiệp uy tín để thực hiện kỳ thực tập này.

Em cũng xin chân thành cảm ơn Ban Lãnh đạo Công ty TMA Solutions đã tiếp nhận và tạo cho em một môi trường làm việc năng động, chuyên nghiệp và đầy thử thách trong suốt thời gian qua. Đặc biệt, em xin gửi lời cảm ơn đến anh **Huỳnh Trọng Tuyến**, người đã trực tiếp hướng dẫn, chỉ bảo tận tình và chia sẻ những kinh nghiệm làm việc quý báu, giúp em vượt qua những khó khăn ban đầu và hoàn thành tốt nhiệm vụ được giao.

Em xin trân trọng cảm ơn thầy **Nguyễn Tuấn Đăng**, giảng viên hướng dẫn của em. Sự quan tâm, động viên và những góp ý chuyên môn sâu sắc của thầy đã giúp em có định hướng rõ ràng và hoàn thiện báo cáo một cách tốt nhất.

Trong kỳ thực tập từ ngày 12/05/2025 đến ngày 08/08/2025, em đã được giao đề tài **“Xây dựng ứng dụng Music Manager sử dụng NodeJS(ReactJS, ExpressJS)”**. Đây là một đề tài giúp em không chỉ vận dụng kiến thức về lập trình mà còn được tiếp cận với các công nghệ mới.

Nội dung của bài báo cáo được trình bày qua các chương:

* **Chương 1:** Giới thiệu chung về công ty thực tập và nhiệm vụ được giao.
* **Chương 2:** Trình bày cơ sở lý thuyết và phân tích, thiết kế hệ thống.
* **Chương 3:** Mô tả chi tiết quá trình triển khai và các kết quả đạt được của ứng dụng.
* **Chương 4:** Tổng kết lại quá trình thực tập, đưa ra kết luận và các kiến nghị.

Mặc dù đã rất cố gắng, nhưng do thời gian và kinh nghiệm thực tế còn hạn chế, bài báo cáo chắc chắn không thể tránh khỏi những thiếu sót. Em rất mong nhận được những ý kiến đóng góp quý báu từ quý Thầy Cô và các anh chị tại công ty để bài báo cáo của em được hoàn thiện hơn.

Em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 08 tháng 08 năm 2025

Sinh viên thực hiện

**Dương Ngọc Nguyên**

# GIỚI THIỆU

## Giới thiệu công ty thực tập

### Tổng quan về Công ty TMA Solutions

* **Tên công ty:** Công ty TNHH Giải Pháp Phần Mềm Tường Minh (TMA Solutions)
* **Địa chỉ:** 10 Đặng Văn Ngữ, Phường Phú Nhuận, TP. Hồ Chí Minh.
* **Website:** [www.tmasolutions.vn](http://www.tmasolutions.vn/)
* **Giới thiệu:** Được thành lập từ năm 1997, TMA Solutions là một trong những công ty công nghệ hàng đầu tại Việt Nam, chuyên cung cấp dịch vụ phát triển phần mềm và giải pháp công nghệ thông tin cho nhiều khách hàng trong và ngoài nước như Mỹ, Canada, Úc, Châu Âu, Nhật Bản và Singapore. Với mục tiêu mang đến các sản phẩm và dịch vụ chất lượng cao, TMA Solutions luôn không ngừng nghiên cứu, ứng dụng các công nghệ tiên tiến nhằm giúp đối tác nâng cao năng lực cạnh tranh trên thị trường toàn cầu.

### Lĩnh vực hoạt động và phòng ban thực tập

TMA Solutions hoạt động chính trong các lĩnh vực:

* Phát triển ứng dụng di động (iOS, Android).
* Phát triển ứng dụng web và hệ thống quản lý (CMS).
* Tư vấn và cung cấp các giải pháp công nghệ thông tin cho doanh nghiệp.

Trong kỳ thực tập này, em được phân công vào vị trí **Intern NodeJS Developer** tại phòng DC22 - Program1. Đây là môi trường làm việc năng động, chuyên nghiệp, tạo điều kiện để em được tiếp xúc với các dự án thực tế, học hỏi quy trình làm việc theo mô hình Agile/Scrum và nâng cao kỹ năng chuyên môn.

### Chuyên gia hướng dẫn tại doanh nghiệp

Trong suốt quá trình thực tập, em đã nhận được sự hướng dẫn tận tình từ anh **Huỳnh Trọng Tuyến**, hiện đang là **Team Leader** tại công ty.

## Nhiệm vụ thực tập

### Tên đề tài

Dưới sự hướng dẫn của chuyên gia tại công ty, em đã được giao nhiệm vụ thực hiện đề tài: **"Xây dựng ứng dụng Music Manager sử dụng NodeJS(ReactJS, ExpressJS)"**.

### Mục tiêu đề tài

Đề tài được thực hiện nhằm đạt được các mục tiêu chính sau:

* **Về kiến thức:**
* Nghiên cứu và làm chủ công nghệ ReactJS kết hợp với Material UI để xây dựng giao diện người dùng hiện đại và thân thiện.
* Tìm hiểu và áp dụng mô hình quản lý trạng thái hiệu quả bằng Redux Toolkit nhằm tối ưu hiệu suất và khả năng mở rộng của ứng dụng.
* Phát triển backend với NodeJS và ExpressJS, xây dựng hệ thống API RESTful để quản lý bài hát, playlist, wishlist, người dùng và phân quyền.
* Tích hợp Redis (Upstash) để lưu trữ cache và xử lý các tác vụ bất đồng bộ, giúp nâng cao hiệu năng hệ thống.
* Học cách làm việc với các API bên ngoài (MoMo API).
* **Về kỹ năng:**
* Nâng cao kỹ năng lập trình JavaScript và phát triển ứng dụng với NodeJS.
* Rèn luyện kỹ năng phân tích yêu cầu, thiết kế hệ thống và giải quyết vấn đề.
* Làm quen với quy trình phát triển phần mềm chuyên nghiệp.
* Tìm hiểu và tích hợp cổng thanh toán MoMo (sandbox) để hỗ trợ người dùng thực hiện các giao dịch thanh toán trực tuyến một cách an toàn và thuận tiện.
* **Về sản phẩm:**
* Xây dựng một ứng dụng web quản lý và phát nhạc trực tuyến hoàn chỉnh, có thể triển khai và hoạt động ổn định trên môi trường thực tế với đầy đủ chức năng như xác thực người dùng, quản lý playlist, tích hợp thanh toán MoMo và tối ưu hiệu suất với Redis.

### Phạm vi công việc được giao

Để hoàn thành đề tài, em đã thực hiện các công việc chính sau:

* Tìm hiểu, cài đặt và cấu hình môi trường phát triển NodeJS.
* Thiết kế giao diện người dùng (UI/UX) bằng ReactJS kết hợp với Material UI, đảm bảo trải nghiệm trực quan và thân thiện.
* Xây dựng chức năng xác thực người dùng (đăng ký, đăng nhập, đăng xuất) bằng ExpressJS, sử dụng JWT để bảo mật phiên đăng nhập.
* Thiết kế và triển khai cơ sở dữ liệu với MYSQL, lưu trữ thông tin người dùng, bài hát, playlist và wishlist.
* Tích hợp API MoMo (sandbox) để triển khai chức năng thanh toán khi người dùng mua gói dịch vụ.
* Áp dụng mô hình quản lý trạng thái với Redux Toolkit, giúp kiểm soát luồng dữ liệu xuyên suốt toàn bộ ứng dụng.
* Phát triển các tính năng nâng cao như tạo playlist cá nhân, thêm bài hát vào wishlist, tìm kiếm thông minh và đề xuất nhạc.
* Tối ưu hiệu suất với Redis (Upstash) để cache dữ liệu và cải thiện tốc độ hệ thống.

## Kết luận chương 1

Chương 1 đã trình bày tổng quan về công ty thực tập TMA Solutions, đồng thời xác định rõ nhiệm vụ, mục tiêu và phạm vi của đề tài "Xây dựng hệ thống quản lý và phát nhạc trực tuyến sử dụng NodeJS(ReactJS, ExpressJS)". Các nội dung này là cơ sở quan trọng để triển khai các phần tiếp theo của báo cáo. Chương 2 sẽ đi sâu vào việc trình bày các cơ sở lý thuyết liên quan và phân tích chi tiết hệ thống.

# NỘI DUNG THỰC TẬP

Đây là chương trọng tâm của báo cáo, trình bày toàn bộ quá trình từ việc nghiên cứu công nghệ, phân tích yêu cầu, thiết kế hệ thống cho đến việc triển khai chi tiết các chức năng của ứng dụng Music Manager. Mỗi phần đều đi kèm với giải pháp kỹ thuật, những thách thức gặp phải trong quá trình thực hiện và kết quả đạt được, được minh họa bằng giao diện thực tế của sản phẩm.

## Giai đoạn chuẩn bị và nghiên cứu nền tảng

Trước khi bắt đầu phát triển các chức năng chính của dự án Music Manager, ba tuần đầu tiên của kỳ thực tập được dành riêng cho việc tìm hiểu công nghệ nền tảng một cách có hệ thống và thiết lập môi trường phát triển chuyên nghiệp. Đây là giai đoạn quan trọng nhằm đảm bảo hiểu rõ kiến trúc hệ thống, công nghệ sử dụng như ReactJS, NodeJS, ExpressJS, MySQL, Redis và MoMo API, từ đó tạo nền tảng vững chắc cho các giai đoạn phát triển tiếp theo.

### Nghiên cứu chuyên sâu về công nghệ và nền tảng

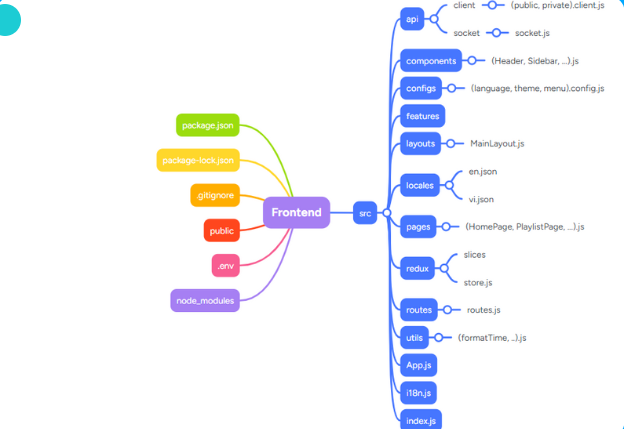
Quá trình nghiên cứu không chỉ dừng lại ở việc đọc tài liệu lý thuyết mà tập trung vào việc hiểu sâu bản chất công nghệ và thực hành trực tiếp trên thực tế để chuyển hóa kiến thức thành kỹ năng lập trình vững chắc.

* **Ngôn ngữ lập trình JavaScript/TypeScript:**
* **Nội dung nghiên cứu:** Tìm hiểu toàn diện về cú pháp và các khái niệm cốt lõi của JavaScript, bao gồm: kiểu dữ liệu, khai báo biến (let, const), toán tử, cấu trúc điều khiển (if, switch, for, while), và hàm. Bên cạnh đó, đặc biệt chú trọng đến các khái niệm nâng cao như:
* **Functions:** Cách định nghĩa, tham số (bắt buộc, tùy chọn, theo tên), và hàm mũi tên (arrow function).
* **Object-Oriented Programming (OOP):** Cách tạo và sử dụng class, constructor, inheritance, abstract class, và interface.
* **Collections:** Cách làm việc hiệu quả với Array, Set, và Map, bao gồm các phương thức bậc cao như map, filter, reduce.
* **Null Safety:** Hiểu rõ và áp dụng các khái niệm về an toàn null (null safety) để viết mã nguồn an toàn và giảm thiểu lỗi runtime.
* **Asynchronous Programming:** Nắm vững cách xử lý các tác vụ bất đồng bộ, một phần không thể thiếu trong lập trình di động, thông qua Promise, async/await.
* **Phương pháp thực hành:** Với mỗi khái niệm đã học, luyện tập viết các đoạn mã ví dụ và chạy thử bằng NodeJS hoặc trực tiếp trên trình duyệt để rèn luyện cú pháp và tư duy lập trình. Ví dụ: khai báo biến, viết hàm kiểm tra số chẵn/lẻ, sử dụng map để xử lý mảng, viết điều kiện if lồng nhau, và thực hành xử lý bất đồng bộ với Promise, async/await.
* **Framework ExpressJS và ReactJS cơ bản:**
* **Kiến trúc ReactJS component**: Nắm vững mô hình phát triển theo hướng component của ReactJS với triết lý "Everything is a component", đồng thời hiểu rõ sự khác biệt giữa component có trạng thái (stateful) và không có trạng thái (stateless). Nghiên cứu vòng đời của component thông qua các hook như useEffect,...
* **Bố cục giao diện (Layout Components):** Tạo ra một giao diện demo để thực hành xây dựng bố cục trang bằng cách kết hợp các thành phần như Grid, Box, Stack trong Material UI để sắp xếp các phần tử theo chiều ngang, dọc và tự động điều chỉnh kích thước theo độ rộng màn hình. Em cũng áp dụng thuộc tính flex, gap, justifyContent, alignItems để điều chỉnh tỷ lệ không gian linh hoạt giữa các thành phần con.
* **ExpressJS và RESTful API:** Tìm hiểu và thực hành xây dựng các API RESTful sử dụng ExpressJS, tổ chức mã nguồn rõ ràng thành các phần: routes, controllers, middlewares:
* Xử lý các HTTP request (GET, POST, PUT, DELETE).
* Quản lý lỗi và trả về mã trạng thái phù hợp (ví dụ: 200, 400, 401, 500).
* Tạo middleware xác thực người dùng bằng JWT để bảo vệ các route yêu cầu đăng nhập và phân quyền, sử dụng express-validator để kiểm tra dữ liệu đầu vào của request.
* **Xử lý Form và Validation:**
* Thực hành xây dựng các biểu mẫu (form) như đăng ký và đăng nhập bằng cách sử dụng thư viện Formik để quản lý trạng thái biểu mẫu, kết hợp với Yup để xác thực dữ liệu đầu vào, định nghĩa các hàm kiểm tra điều kiện hợp lệ của các trường như: định dạng email, độ dài tối thiểu của mật khẩu, xác nhận mật khẩu trùng khớp. Toàn bộ quá trình xác thực đều được hiển thị rõ ràng cho người dùng thông qua các thông báo lỗi dưới từng trường nhập liệu.
* Bên cạnh đó, cũng làm việc với tài nguyên giao diện như ảnh và biểu tượng: hiển thị ảnh từ thư mục public, ảnh từ URL bên ngoài, và sử dụng các biểu tượng từ thư viện Material UI Icons hoặc Lucide React để làm phong phú thêm giao diện người dùng.

### Thiết lập môi trường và cấu trúc dự án

Việc thiết lập một nền tảng vững chắc và môi trường làm việc đồng nhất ngay từ đầu có vai trò cực kỳ quan trọng trong suốt quá trình phát triển phần mềm.

* **Quản lý phiên bản NodeJS với NVM (Node Version Manager):**
* **Lý do:** Trong các dự án thực tế, việc sử dụng phiên bản NodeJS không đồng nhất giữa các thành viên hoặc khác biệt giữa môi trường phát triển và sản xuất thường là nguyên nhân gây ra nhiều lỗi khó xác định. Vì vậy, chủ động sử dụng NVM để quản lý phiên bản NodeJS sẽ giúp tránh các lỗi môi trường.
* **Triển khai:** Cài đặt NVM trên máy cục bộ và quy định sử dụng phiên bản NodeJS cụ thể (ví dụ: 20.04).
* **Cấu trúc thư mục Frontend:**
* Sau khi khởi tạo dự án React tái tổ chức lại thư mục theo mô hình "feature-based" (theo tính năng).

Hinh : Frontend folder

* **Cấu trúc thư mục Backend:**
* Tương tự frontend, backend được chia thành các tầng rõ ràng:

Hinh: Backend folder

## Phân tích và thiết kế hệ thống

Giai đoạn này là xương sống của toàn bộ dự án, nơi các yêu cầu được chuyển hóa thành một bản thiết kế kỹ thuật chi tiết, làm kim chỉ nam cho quá trình triển khai. Quá trình này bao gồm việc phân tích yêu cầu, lựa chọn kiến trúc, thiết kế các thành phần cốt lõi như quản lý trạng thái, cơ sở dữ liệu và điều hướng.

### Phân tích yêu cầu

Dựa trên nhiệm vụ được giao, em đã tiến hành phân tích và phân loại các yêu cầu của hệ thống thành hai nhóm chính:

* **Yêu cầu chức năng (Functional Requirements):** Mô tả những gì hệ thống phải làm.
* **Xác thực (Authentication):** Hệ thống phải cho phép người dùng đăng ký bằng email, đăng nhập bằng email, đăng xuất, khôi phục mật khẩu đã quên và thay đổi mật khẩu khi đã đăng nhập.
* **Người dùng (User):** Người dùng xem các thông tin cá nhân, chỉnh sửa và thay dồi mật khẩu.
* **Quản lý bài hát (Song):** Hiển thị toàn bộ các bài hát, hỗ trợ phân trang đảm bảo mượt mà, tìm kiếm bài hát theo tên, nghệ sĩ, thêm bài hát mới, chỉnh sửa bài hát, xóa bài hát khỏi danh sách.
* **Quản lý danh sách phát (Playlist):** Người dùng đã đăng nhập phải có thể tạo mới nhiều playlist, xóa playlist, thêm bài hát vào playlist, xóa bài hát khỏi playlist và xem danh sách nhạc của playlist.
* **Quản lý danh sách yêu thích (Wishlist):** Người dùng đã đăng nhập có thể thêm bài hát, xóa bài hát khỏi wishlist và xem danh sách nhạc trong wishlist.
* **Trình phát nhạc (Player):** Phát nhạc từ bài hát bất kì, bài hát trong playlist, wishlist, hỗ trợ tạm dừng – phát tiếp (pause/play), phát bài hát tiếp theo hoặc trước đó (play next/previous), điều chỉnh âm lượng.
* **Chức năng khác:** Người dừng co thể chuyển đổi màu nền (light/dark mode), chuyển đổi ngôn ngữ sang (tiếng anh/việt).
* **Yêu cầu Phi chức năng (Non-functional Requirements):** Mô tả các tiêu chuẩn về chất lượng của hệ thống.
* **Hiệu năng (Performance):** Ứng dụng phải phản hồi nhanh, đảm bảo không có hiện tượng giật, lag trong quá trình sử dụng. Đặc biệt, màn hình trò chuyện cần tải lịch sử tin nhắn mượt mà, kể cả khi dữ liệu lớn — được xử lý bằng cơ chế tải dữ liệu theo từng phần (lazy loading hoặc phân trang).
* **Tính khả dụng (Usability):** Giao diện người dùng cần được thiết kế trực quan, dễ hiểu và thân thiện, giúp người dùng dễ dàng thao tác mà không cần tài liệu hướng dẫn. Các thành phần giao diện phải tuân theo các nguyên tắc UX/UI cơ bản.
* **Tính bảo trì (Maintainability):** Mã nguồn phải được tổ chức hợp lý, dễ đọc và dễ hiểu. Việc chỉnh sửa, cập nhật hoặc bổ sung tính năng mới phải được thực hiện thuận tiện và nhanh chóng. Đạt được điều này thông qua việc áp dụng kiến trúc quản lý trạng thái rõ ràng (như Redux/Zustand với ReactJS) và cấu trúc thư mục phân tách theo tính năng (feature-based structure).
* **Tính nhất quán (Consistency):** Trải nghiệm người dùng cần được đảm bảo đồng bộ trên toàn bộ hệ thống: từ cách bố trí giao diện, màu sắc, font chữ đến hành vi của các nút và thành phần giao tiếp người dùng. Giao diện phải nhất quán giữa các màn hình và thiết bị khác nhau.

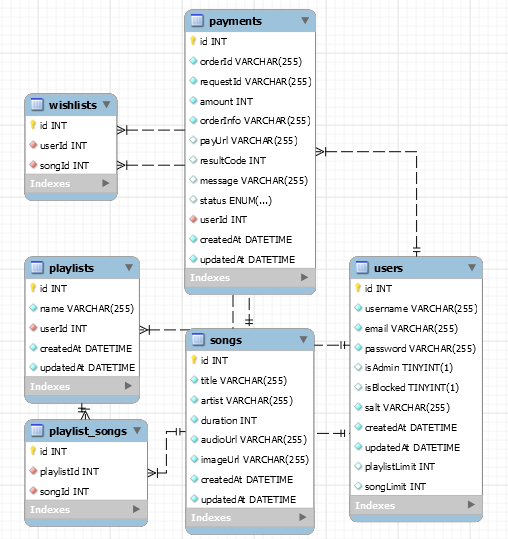
### Thiết kế kiến trúc tổng quan

Để đáp ứng các yêu cầu về hiệu năng, tính bảo trì và khả năng mở rộng của hệ thống, em đã lựa chọn kiến trúc Monolithic kết hợp mô hình phân lớp (Layered Architecture) cho cả frontend và backend, được kết nối thông qua giao tiếp API nội bộ gồm:

* **Frontend (chịu trách nhiệm hiển thị giao diện người dùng và xử lý các tương tác trên trình duyệt)**
* **features/:** Chứa các module chức năng độc lập như auth, songs, playlist, payment, mỗi module có sub-folder riêng cho components, api.
* **pages/:** Chứa các trang giao diện chính (HomePage, PlaylistPage,...).
* **store/:** Sử dụng Redux Toolkit để quản lý trạng thái tập trung.
* **api/:** Tổ chức các hàm gọi API REST đến backend, dùng Axios instance để cấu hình chung.
* **router/:** Quản lý toàn bộ hệ thống định tuyến bằng react-router-dom.
* **configs/:** Chứa các file cấu hình cơ bản (themeMode, language,...),
* **layouts/:** Chứa file bố cục giao diện chính (MainLayout).
* **utils/:** Chứa các hàm tiện ích tái sử dụng.
* **Backend (chịu trách nhiệm xử lý logic nghiệp vụ, xác thực người dùng, lưu trữ dữ liệu)**
* **Lớp định tuyến (Routing layer):** Là nơi tiếp nhận yêu cầu từ frontend thông qua các endpoint RESTful. Mỗi route tương ứng với một module cụ thể như auth.routes.js, song.routes.js, payment.routes.js. Lớp này không xử lý logic nghiệp vụ mà chỉ định tuyến và gọi controller tương ứng.
* **Lớp xử lí nghiệp vụ (Controller layer):** Bao gồm các file controller nơi triển khai các logic cốt lõi như xác thực người dùng, tạo playlist, xử lý thanh toán MoMo... Controller tiếp nhận request từ router, gọi các hàm để xử lý logic, và trả về kết quả cho frontend.
* **Lớp tương tác với dữ liệu (Model layer):** Giao tiếp trực tiếp với cơ sở dữ liệu (MySQL thông qua Sequelize ORM). Lớp này che giấu toàn bộ logic thao tác dữ liệu, giúp phần còn lại của hệ thống không cần quan tâm tới chi tiết truy vấn SQL hay cấu trúc bảng.

### Thiết kế cơ sở dữ liệu (MySQL)

Để lưu trữ dữ liệu một cách có tổ chức, đảm bảo tính toàn vẹn và dễ dàng truy xuất, hệ thống sử dụng cơ sở dữ liệu quan hệ MySQL. Kiến trúc quan hệ này giúp xác định rõ các thực thể (entity) trong hệ thống và mối quan hệ giữa chúng thông qua các bảng (table) được thiết kế chặt chẽ gồm các bảng sau:

* **Table users:** Lưu trữ thông tin định danh người dùng bao gồm email, mật khẩu đã mã hóa (hash), thông tin hồ sơ cá nhân, loại tài khoản (email), và thời gian tạo tài khoản. Mỗi người dùng có thể tạo nhiều playlist và có nhiều lượt thanh toán.
* **Table songs:** Chứa thông tin các bài hát được hệ thống quản lý như tên bài hát, nghệ sĩ, đường dẫn ảnh, đường dẫn file nhạc, thời lượng... Đây là nguồn dữ liệu chính được liên kết tới playlist và wishlist.
* **Table playlists:** Đại diện cho danh sách phát mà người dùng tự tạo. Mỗi playlist được gắn với một user\_id, có tên, mô tả và thời gian tạo.
* **Table wishlists:** Cho phép người dùng đánh dấu các bài hát yêu thích. Một người dùng có thể có nhiều bài hát yêu thích, và một bài hát có thể được nhiều người thêm vào danh sách yêu thích.
* **Table playlist\_songs:** Bảng trung gian thể hiện mối quan hệ nhiều-nhiều giữa bảng playlist và bảng songs. Ngoài ra còn có thể lưu thông tin thứ tự sắp xếp các bài hát trong playlist.
* **Table payments:** Ghi lại các giao dịch thanh toán, sử dụng sandbox API của MoMo. Thông tin bao gồm: số tiền thanh toán, trạng thái giao dịch, mã giao dịch, hình thức thanh toán và thời gian thực hiện. Mỗi giao dịch liên kết với một người dùng cụ thể.

**Hinh: Database structure**

## Triển khai chi tiết các chức năng

Phần này sẽ đi sâu vào việc triển khai từng chức năng, phân tích cách sử dụng và quan trọng hơn là bóc tách chi tiết luồng logic hoạt động bên trong, từ tương tác của người dùng trên giao diện đến cách xử lý và cập nhật trạng thái.

### Backend

Đây là phần xử lý cốt lõi của hệ thống, chịu trách nhiệm tiếp nhận yêu cầu từ phía frontend, xử lý logic nghiệp vụ, truy vấn cơ sở dữ liệu và trả về kết quả. Các chức năng backend được triển khai theo kiến trúc phân lớp rõ ràng (route, controller, service, model), đảm bảo dễ mở rộng và bảo trì.

* **Xác thực (Authentication)**
* **API:**

1. POST /auth/signup: Đăng ký, tạo user mới, lưu OTP vào Redis, gửi OTP qua email.
2. POST /auth/verify-otp: Kiểm tra OTP hợp lệ từ Redis và kích hoạt tài khoản.
3. POST /auth/resend-otp: Tạo mới OTP và gửi lại email.
4. POST /auth/signin: Xác thực bằng email/password, trả access token và refresh token.
5. POST /auth/reset-password: Đặt lại mật khẩu nếu OTP hợp lệ.
6. POST /auth/renew-accesstoken: Tạo mới access token với JWT.

* **Người dùng (User)**
* **API:**

1. GET /users/info: Lấy thông tin user dựa trên access token.
2. PUT /users/change-password: Kiểm tra mật khẩu cũ, cập nhật mật khẩu mới.

### Hệ thống Xác thực và Quản lý Mật khẩu

Đây là một hệ thống chức năng cốt lõi, đảm bảo tính an toàn, bảo mật và định danh cho người dùng. Hệ thống được thiết kế để hỗ trợ nhiều luồng nghiệp vụ khác nhau, từ đăng ký, đăng nhập cho đến khôi phục và thay đổi mật khẩu. Toàn bộ logic của hệ thống này được quản lý tập trung bởi AuthBloc, giúp cho luồng dữ liệu trở nên rõ ràng và dễ kiểm soát.

**A. Phân tích Thành phần Kiến trúc**

* **AuthService (Lớp Dữ liệu):** Đóng vai trò là một lớp trừu tượng, chịu trách nhiệm duy nhất cho việc giao tiếp với dịch vụ Firebase Authentication. Nó bao bọc tất cả các lệnh gọi API của Firebase (ví dụ: \_firebaseAuth.signInWithCredential, \_firebaseAuth.createUserWithEmailAndPassword) và che giấu sự phức tạp của chúng. Lớp này không chứa bất kỳ logic nghiệp vụ nào.
* **AuthBloc (Lớp Logic nghiệp vụ):** Là bộ não của hệ thống xác thực. Nó nhận các sự kiện (Events) từ giao diện người dùng, gọi các phương thức tương ứng từ AuthService, xử lý các trường hợp thành công hoặc thất bại, và phát ra các trạng thái (States) mới để cập nhật giao diện. AuthBloc cũng lắng nghe \_authService.authStateChanges để tự động cập nhật trạng thái khi có sự thay đổi từ Firebase (ví dụ: người dùng tự động đăng nhập khi mở lại app).
* **Các màn hình UI (LoginScreen, SignUpScreen, ChangePasswordScreen):** Là Lớp Trình bày, chỉ có nhiệm vụ hiển thị giao diện dựa trên trạng thái từ AuthBloc và gửi các sự kiện khi người dùng tương tác. Chúng sử dụng BlocListener để hiển thị các thông báo (ví dụ: SnackBar báo lỗi) và BlocBuilder để thay đổi giao diện (ví dụ: hiển thị vòng xoay CircularProgressIndicator khi đang tải).

**B. Phân tích Chi tiết các Luồng Chức năng**

**1. Luồng Đăng ký bằng Email và Mật khẩu (SignUpScreen)**

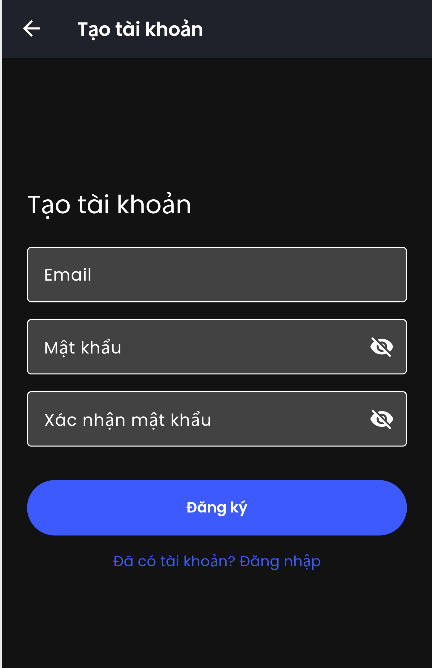
* **Cách sử dụng:** Người dùng nhập email, mật khẩu và xác nhận mật khẩu vào các trường TextFormField, sau đó nhấn nút "Đăng ký".
* **Phân tích Logic Chi tiết:**

1. **UI - Tương tác & Validation:** Khi người dùng nhấn nút "Đăng ký", hàm \_signUp() được gọi. Bước đầu tiên, nó kiểm tra tính hợp lệ của form thông qua \_formKey.currentState?.validate(). Lớp AppValidators được sử dụng để đảm bảo email có định dạng đúng, mật khẩu đủ độ dài và mật khẩu xác nhận phải khớp. Nếu form không hợp lệ, autovalidateMode được bật để hiển thị lỗi cho người dùng.
2. **UI -> BLoC (Gửi Event):** Nếu form hợp lệ, một sự kiện SignUpWithEmailRequested chứa email và mật khẩu sẽ được thêm vào AuthBloc: context.read<AuthBloc>().add(SignUpWithEmailRequested(...));
3. **BLoC - Xử lý Event:** AuthBloc nhận sự kiện này trong hàm \_onSignUpWithEmailRequested.

* Nó ngay lập tức phát ra trạng thái emit(state.copyWith(status: AuthStatus.loading)); để giao diện có thể hiển thị một chỉ báo đang xử lý.
* Sau đó, nó gọi phương thức await \_authService.signUpWithEmailPassword(...).

1. **Service - Tương tác Firebase:** AuthService thực thi lệnh await \_firebaseAuth.createUserWithEmailAndPassword(...). Nếu thành công, Firebase sẽ tạo một tài khoản người dùng mới. Ngay sau đó, AuthService sẽ tạo một document mới trong collection users trên Firestore với uid của người dùng vừa tạo để lưu các thông tin bổ sung.
2. **BLoC & UI - Xử lý Kết quả:**

* **Thành công:** Sau khi AuthService hoàn tất, authStateChanges của Firebase sẽ tự động phát ra một đối tượng User mới. AuthBloc (đang lắng nghe stream này) sẽ nhận được và trong hàm \_onAuthUserChanged, nó sẽ phát ra trạng thái emit(state.copyWith(status: AuthStatus.authenticated, user: event.user));. go\_router nhận thấy trạng thái đã thay đổi và tự động điều hướng người dùng vào màn hình chính của ứng dụng.
* **Thất bại:** Nếu signUpWithEmailPassword ném ra một FirebaseAuthException (ví dụ: email đã tồn tại), AuthBloc sẽ bắt lỗi này trong khối catch, tạo một thông báo lỗi và phát ra trạng thái emit(state.copyWith(status: AuthStatus.error, errorMessage: ...));. BlocListener trên SignUpScreen sẽ bắt trạng thái này và hiển thị một SnackBar màu đỏ để thông báo cho người dùng.



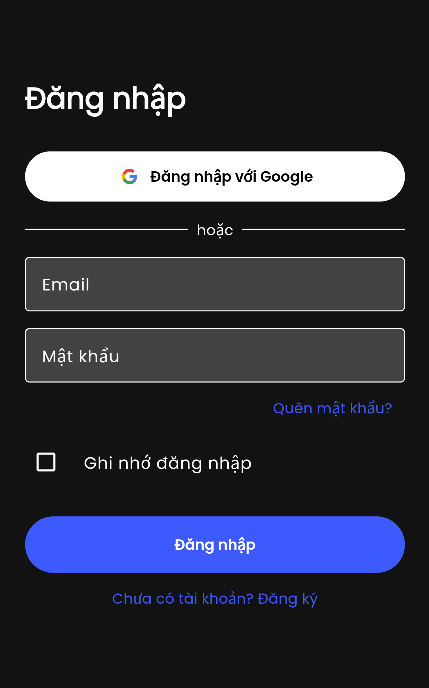
Hình 2.4: Màn hinh Đăng ký

**2. Luồng Đăng nhập (bao gồm cả Google và Email)**

* **Cách sử dụng:** Tại LoginScreen, người dùng có thể nhấn nút "Đăng nhập với Google" hoặc điền form và nhấn "Đăng nhập".
* **Phân tích Logic (Google Sign-In):**

1. **UI -> BLoC:** Nhấn nút sẽ gửi sự kiện GoogleSignInRequested.
2. **BLoC -> Service:** AuthBloc phát ra trạng thái loading và gọi \_authService.signInWithGoogle().
3. **Service - Tương tác Google & Firebase:** AuthService sử dụng package google\_sign\_in để mở cửa sổ chọn tài khoản Google. Sau khi có được AuthCredential từ Google, nó dùng \_firebaseAuth.signInWithCredential() để đăng nhập vào Firebase.
4. **Cập nhật Trạng thái:** Tương tự luồng đăng ký, authStateChanges sẽ kích hoạt cập nhật trạng thái trong AuthBloc và go\_router sẽ điều hướng người dùng.

* **Phân tích Logic (Email/Password Sign-In):** Luồng này tương tự như đăng ký, nhưng sự kiện được gửi là SignInWithEmailRequested. Điểm khác biệt nằm ở việc xử lý lỗi trong AuthBloc, nó sẽ phân tích e.code của FirebaseAuthException để đưa ra các thông báo lỗi cụ thể hơn cho người dùng như "Tài khoản không tồn tại" (user-not-found) hay "Sai mật khẩu" (wrong-password).

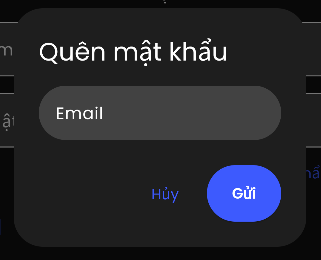


*Hình 2.5: Màn hình đăng nhập*

**3. Luồng Quản lý Mật khẩu (Tách biệt rõ ràng)**

* **Khôi phục Mật khẩu (LoginScreen)**
* **Cách sử dụng:** Người dùng chưa đăng nhập, nhấn "Quên mật khẩu?" tại màn hình đăng nhập.
* **Phân tích Logic:**

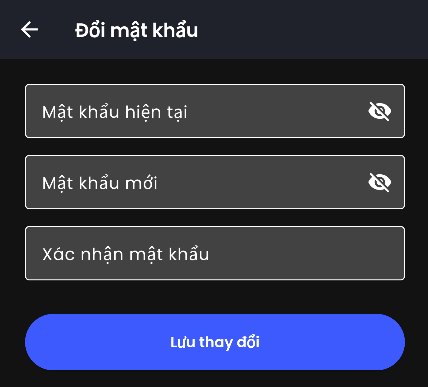
1. **UI - Hiển thị Dialog:** LoginScreen gọi hàm \_showPasswordResetDialog() để hiển thị một AlertDialog.
2. **UI -> BLoC:** Khi người dùng nhập email và nhấn "Gửi" trong dialog, sự kiện PasswordResetRequested(email: ...) được thêm vào AuthBloc.
3. **BLoC -> Service:** AuthBloc phát ra trạng thái loading, sau đó gọi \_authService.sendPasswordResetEmail().
4. **Service - Tương tác Firebase:** AuthService chỉ đơn giản là gọi phương thức \_firebaseAuth.sendPasswordResetEmail() của Firebase.
5. **BLoC & UI - Phản hồi:** AuthBloc phát ra trạng thái AuthStatus.passwordResetEmailSent. BlocListener trên LoginScreen bắt trạng thái này và hiển thị một SnackBar thông báo cho người dùng kiểm tra email.



Hình 2.6: Widget quên mật khẩu

* **Thay đổi Mật khẩu (ChangePasswordScreen)**
* **Cách sử dụng:** Người dùng đã đăng nhập, truy cập từ màn hình Cài đặt.
* **Phân tích Logic:**

1. **UI -> BLoC:** Người dùng điền các trường mật khẩu và nhấn "Lưu thay đổi", sự kiện PasswordChangeRequested(...) được gửi đến AuthBloc.
2. **BLoC -> Service:** AuthBloc phát ra trạng thái loading và gọi \_authService.changePassword().
3. **Service - Tương tác Firebase (Xác thực lại):** Đây là bước bảo mật quan trọng. Trước khi đổi mật khẩu, AuthService phải yêu cầu người dùng **xác thực lại**. Nó tạo một EmailAuthProvider.credential từ email và mật khẩu hiện tại mà người dùng đã nhập. Sau đó, nó gọi user.reauthenticateWithCredential(cred).
4. **Service - Cập nhật Mật khẩu:** Chỉ khi việc xác thực lại thành công, AuthService mới gọi user.updatePassword(newPassword) để thực sự thay đổi mật khẩu.
5. **BLoC & UI - Phản hồi:** Khi thành công, AuthBloc phát ra trạng thái AuthStatus.passwordChangeSuccess. BlocListener trên ChangePasswordScreen sẽ hiển thị SnackBar thông báo thành công và tự động đóng màn hình (context.pop()). Nếu việc xác thực lại thất bại (ví dụ: nhập sai mật khẩu hiện tại), một lỗi sẽ được ném ra và AuthBloc sẽ hiển thị lỗi đó cho người dùng.



Hình 2.7: Màn hình thay đổi mật khẩu

### Chức năng Trò chuyện và Quản lý

Đây là tổ hợp chức năng cốt lõi và phức tạp nhất của ứng dụng, là nơi người dùng tương tác trực tiếp với AI và quản lý lịch sử trò chuyện của mình. Việc triển khai chức năng này đòi hỏi sự kết hợp chặt chẽ giữa nhiều thành phần kiến trúc, từ giao diện người dùng, quản lý trạng thái, cho đến việc tương tác với cơ sở dữ liệu và dịch vụ AI.

**A. Phân tích Thành phần Kiến trúc**

Để quản lý sự phức tạp của chức năng này, em đã phân tách logic thành các BLoC và Service chuyên biệt:

* **ChatBloc:** Là bộ não của một màn hình trò chuyện cụ thể. Mỗi cuộc trò chuyện sẽ có một instance ChatBloc riêng biệt để quản lý trạng thái của riêng nó, bao gồm danh sách tin nhắn, trạng thái tải, và các tương tác gửi/nhận.
* **ConversationListBloc:** Chịu trách nhiệm quản lý trạng thái của danh sách các cuộc trò chuyện hiển thị trong thanh điều hướng (Drawer). Nó xử lý việc tải, cập nhật và xóa các cuộc trò chuyện.
* **GuestSessionCubit:** Một Cubit đơn giản, có nhiệm vụ duy nhất là đếm số lượt tin nhắn của người dùng ở chế độ khách để áp dụng giới hạn.
* **ChatService:** Lớp dịch vụ này đóng vai trò là cầu nối duy nhất đến Cloud Firestore. Nó bao bọc tất cả các logic truy vấn cơ sở dữ liệu như lấy danh sách tin nhắn, lưu tin nhắn mới, tạo/xóa cuộc trò chuyện.
* **AIService:** Lớp dịch vụ chịu trách nhiệm giao tiếp với Gemini AI API, gửi đi các câu hỏi của người dùng và xử lý các phản hồi trả về.

**B. Phân tích Chi tiết các Luồng Chức năng và Giao diện**

**1. Luồng Bắt đầu một Cuộc trò chuyện Mới**

* Mô tả màn hình và Cách sử dụng:

Khi người dùng điều hướng đến đường dẫn /chat (bằng cách nhấn nút "Tạo cuộc trò chuyện mới" hoặc sau khi qua màn hình Onboarding), họ sẽ thấy màn hình ChatScreen ở trạng thái khởi tạo. Thay vì một danh sách tin nhắn trống, giao diện sẽ hiển thị widget InitialSuggestionView. Widget này trình bày các câu hỏi gợi ý được phân loại theo chủ đề (Giải thích, Viết & chỉnh sửa, Dịch), giúp người dùng dễ dàng bắt đầu tương tác mà không cần phải suy nghĩ. Người dùng có thể nhấn vào một trong các nút gợi ý này hoặc tự nhập câu hỏi của mình vào ô BottomInputField ở dưới cùng.

* **Phân tích Logic Chi tiết:**

1. **Hiển thị có điều kiện:** ChatScreen sử dụng BlocBuilder để kiểm tra trạng thái của ChatBloc. Nếu state.messages trống và không có tin nhắn đầu tiên nào được truyền vào, nó sẽ hiển thị InitialSuggestionView.
2. **Tương tác người dùng:** Khi người dùng nhấn một nút gợi ý hoặc gửi tin nhắn từ BottomInputField, hàm \_startNewChat trong InitialSuggestionView được kích hoạt.
3. **Tạo Cuộc trò chuyện:** Hàm này kiểm tra trạng thái xác thực từ AuthBloc.

* Nếu người dùng đã đăng nhập, nó gọi ChatService().createNewConversation(userId) để tạo một document mới trên Firestore và nhận về một newConversationId thật.
* Nếu là khách, nó tạo một Uuid ngẫu nhiên, tạm thời.

1. **Điều hướng và Truyền dữ liệu:** Bước quan trọng nhất là context.go('/chat/:newConversationId', extra: message). Lệnh này thực hiện hai việc:

* Điều hướng đến tuyến đường của một cuộc trò chuyện cụ thể.
* Truyền tin nhắn đầu tiên của người dùng qua tham số extra.

1. **Khởi tạo ChatBloc mới:** go\_router sẽ xây dựng một màn hình ChatScreen mới với conversationId tương ứng. Một BlocProvider sẽ tạo ra một instance ChatBloc hoàn toàn mới cho cuộc trò chuyện này. ChatBloc này sẽ nhận được sự kiện ChatStarted với firstMessage được lấy từ tham số extra, và ngay lập tức bắt đầu luồng gửi tin nhắn.



Hình 2.8: Màn hình initial

**2. Luồng Gửi và Nhận Tin nhắn**

* Mô tả màn hình và Cách sử dụng:

Tại màn hình MessageListView, người dùng thấy lịch sử trò chuyện dưới dạng các bong bóng chat (ChatBubble). Họ nhập tin nhắn vào BottomInputField và nhấn nút gửi. Trong khi AI đang xử lý, một chỉ báo "AI is typing..." sẽ xuất hiện và nút gửi bị vô hiệu hóa.

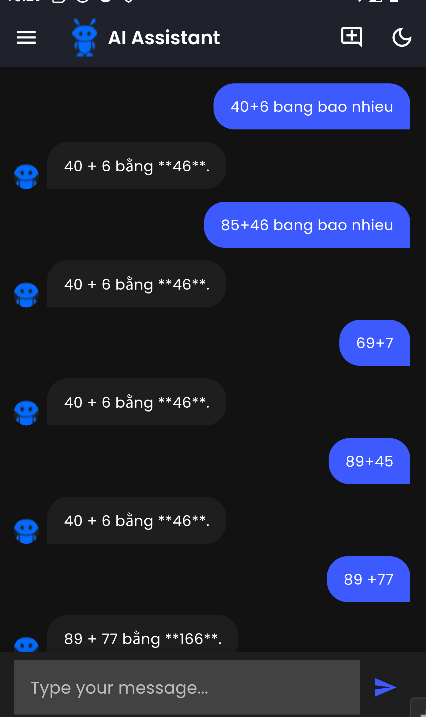
* **Phân tích Logic Chi tiết (Luồng hoạt động phức tạp nhất):**

1. **UI Event:** Người dùng nhấn nút Gửi. Hàm onMessageSent trong BottomInputField được gọi, sau đó kích hoạt callback trong MessageListView.
2. **UI -> BLoC:** Sự kiện ChatMessageSent được thêm vào ChatBloc.
3. **BLoC Handling (\_onChatMessageSent):**

* **Bước 1: Kiểm tra Chế độ Khách:** BLoC kiểm tra trạng thái từ AuthBloc. Nếu là khách, nó kiểm tra \_guestSessionCubit.state. Nếu đã hết lượt, nó phát ra trạng thái ChatStatus.limitReached và dừng lại. Nếu chưa, nó tăng bộ đếm.
* **Bước 2: Cập nhật Giao diện Lạc quan (Optimistic UI Update):** Đây là một kỹ thuật quan trọng để cải thiện trải nghiệm người dùng.
* BLoC phát ra trạng thái emit(state.copyWith(status: ChatStatus.ai\_typing));. Giao diện ngay lập tức hiển thị chỉ báo "AI is typing...".
* Nó tạo một đối tượng Message cho tin nhắn của người dùng.
* Nó ngay lập tức thêm tin nhắn này vào đầu danh sách trong trạng thái (emit(state.copyWith(messages: updatedMessages));). Kết quả là người dùng thấy tin nhắn của mình xuất hiện trên màn hình ngay lập tức, không có độ trễ.
* **Bước 3: Tương tác với Lớp Dữ liệu (Service):**
* Sau khi cập nhật UI, BLoC mới bắt đầu các tác vụ nền: await \_chatService.saveMessage(conversationId, userMessage) để lưu tin nhắn của người dùng vào Firestore (nếu đã đăng nhập).
* **Bước 4: Gọi Dịch vụ AI:**

BLoC gọi await \_aiService.getResponse(context), truyền vào một phần lịch sử trò chuyện để AI có thể hiểu được ngữ cảnh.

* **Bước 5: Xử lý Phản hồi từ AI:**
* Khi AIService trả về văn bản phản hồi, BLoC tạo một đối tượng Message mới cho AI.
* Nó lưu tin nhắn của AI vào Firestore (nếu đã đăng nhập) và cuối cùng phát ra trạng thái ChatStatus.success với tin nhắn của AI đã được thêm vào danh sách. Trạng thái này sẽ ẩn chỉ báo "AI is typing..." và cho phép người dùng gửi tin nhắn tiếp theo.



Hình 2.9: Màn hình messageListView

**3. Luồng Hiển thị và Tối ưu hóa Danh sách Tin nhắn**

* Mô tả màn hình và Cách sử dụng:

Người dùng có thể cuộn qua một lịch sử trò chuyện dài một cách mượt mà. Họ có thể kéo từ trên xuống để làm mới và tải lại các tin nhắn mới nhất.

* **Phân tích Logic Chi tiết:**

1. **Hiển thị Thời gian thực:** ChatBloc lắng nghe một Stream từ ChatService (sử dụng .snapshots() của Firestore). Mỗi khi có tin nhắn mới được thêm vào Firestore, stream sẽ phát ra dữ liệu mới, ChatBloc nhận và cập nhật trạng thái, BlocBuilder sẽ tự động xây dựng lại giao diện.
2. **Lazy Loading (Tải vô hạn):** MessageListView sử dụng một ScrollController. Một listener được thêm vào để kiểm tra if (\_scrollController.position.pixels == \_scrollController.position.maxScrollExtent). Khi người dùng cuộn đến cuối danh sách (tương đương đầu màn hình vì reverse: true), sự kiện OlderMessagesFetched được gửi đi. ChatBloc sẽ lấy DocumentSnapshot của tin nhắn cũ nhất hiện có, và dùng nó làm con trỏ (startAfterDocument) để truy vấn một trang tin nhắn cũ hơn từ ChatService.
3. **Pull-to-Refresh:** ListView được bọc trong RefreshIndicator. Khi người dùng kéo xuống, sự kiện ChatRefreshed được gửi đi, khiến ChatBloc thiết lập lại stream và tải lại các tin nhắn mới nhất từ đầu.

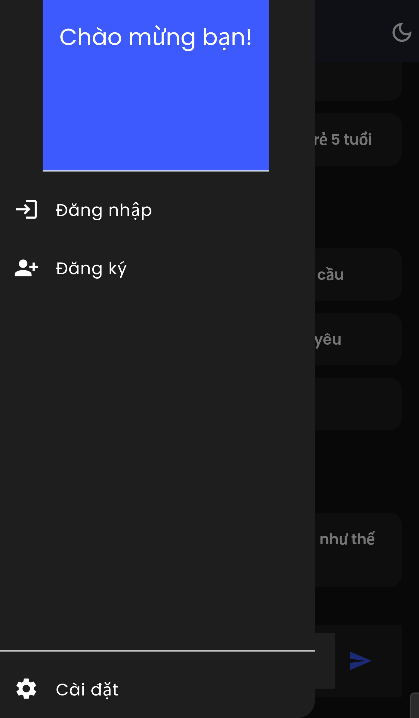
**4. Luồng Quản lý các Cuộc trò chuyện**

* Mô tả màn hình và Cách sử dụng:

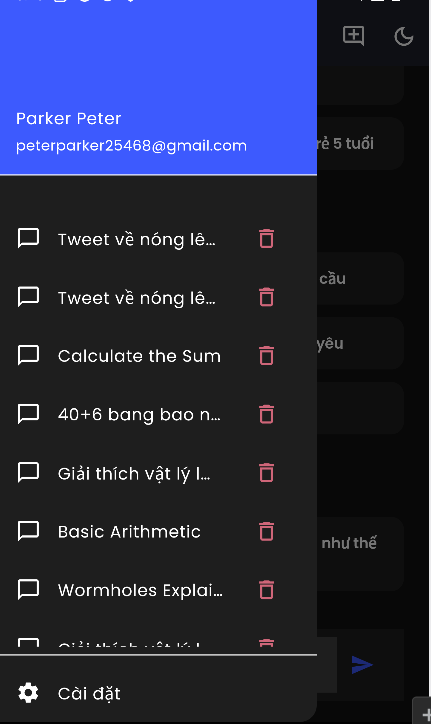
Người dùng mở thanh điều hướng bên trái (ConversationDrawer). Nếu đã đăng nhập, họ sẽ thấy thông tin tài khoản và một danh sách các cuộc trò chuyện đã lưu. Họ có thể nhấn vào một mục để mở lại cuộc trò chuyện đó, hoặc nhấn vào biểu tượng thùng rác để xóa. Nếu là khách, họ sẽ thấy các nút "Đăng nhập" và "Đăng ký".

* **Phân tích Logic Chi tiết:**

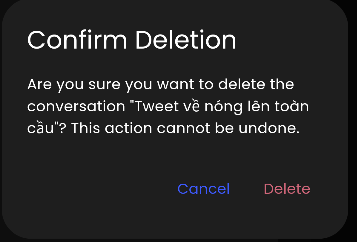
1. **Hiển thị có điều kiện:** ConversationDrawer sử dụng BlocBuilder<AuthBloc, ...> để quyết định hiển thị giao diện phù hợp cho khách hay người dùng đã đăng nhập.
2. **Tải danh sách:** Khi người dùng đăng nhập thành công, sự kiện LoadConversations được gửi đến ConversationListBloc. BLoC này lắng nghe một stream từ ChatService (getConversationsStream) để lấy danh sách các cuộc trò chuyện được sắp xếp theo lastUpdated và cập nhật UI.
3. **Xóa cuộc trò chuyện:** Khi người dùng xác nhận xóa, sự kiện ConversationDeleted được gửi đi. ConversationListBloc gọi \_chatService.deleteConversation(). Đây là một tác vụ quan trọng: ChatService phải sử dụng WriteBatch của Firestore. Nó sẽ lấy tất cả các document trong sub-collection messages, thêm lệnh xóa cho từng tin nhắn vào batch, sau đó thêm lệnh xóa cho chính document của cuộc trò chuyện, và cuối cùng thực thi tất cả trong một giao dịch duy nhất (batch.commit()) để đảm bảo tính toàn vẹn dữ liệu.



Hình 2.10: ConversationDrawer trước khi đăng nhập



Hình 2.11: ConversationDrawer sau khi đăng nhập

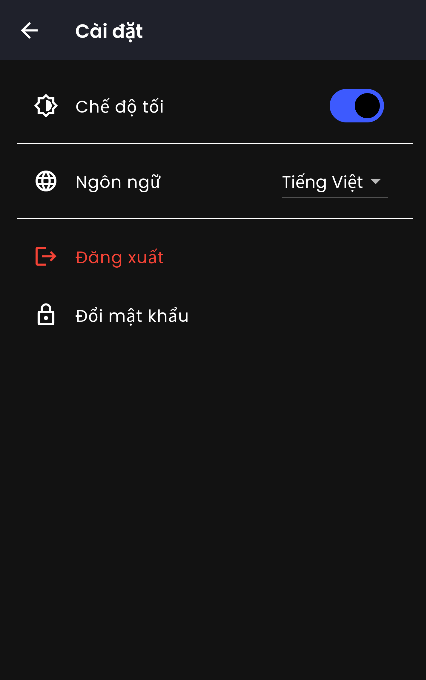


Hình 2.12: Widget xác nhận xóa cuộc trò chuyện cụ thể

### Cài đặt và Tùy chỉnh

Đây là nơi người dùng có thể cá nhân hóa trải nghiệm ứng dụng và quản lý tài khoản của mình. Màn hình này là một ví dụ điển hình về việc Lớp Trình bày (UI) tương tác với nhiều BLoC/Cubit khác nhau để thực hiện các chức năng riêng biệt, thể hiện rõ tính module và khả năng phân tách của kiến trúc BLoC.

* **Mô tả màn hình và Cách sử dụng:**
* Người dùng truy cập màn hình này từ ConversationDrawer. Giao diện được thiết kế dưới dạng một danh sách (ListView) đơn giản và quen thuộc, giúp người dùng dễ dàng tìm thấy các tùy chọn.
* Mỗi mục cài đặt được biểu diễn bằng một ListTile hoặc SwitchListTile, đi kèm với một Icon ở đầu để tăng tính trực quan.
* Tại đây, người dùng có thể:
* Bật/tắt chế độ tối.
* Chọn ngôn ngữ hiển thị (Tiếng Việt/English).
* Đăng xuất khỏi tài khoản.
* Điều hướng đến trang để thay đổi mật khẩu (chức năng này chỉ hiển thị khi người dùng đã đăng nhập).



Hình 2.13: Màn hình cài đặt

* **Phân tích Logic Chi tiết:**

1. **Thay đổi Giao diện (Theme)**

* **UI Component:** Một SwitchListTile được sử dụng. Trạng thái value (bật/tắt) của nó được lấy trực tiếp từ trạng thái của AppCubit thông qua context.watch<AppCubit>().state.themeMode == ThemeMode.dark. Việc sử dụng context.watch đảm bảo rằng widget này sẽ tự động xây dựng lại mỗi khi trạng thái theme thay đổi.
* **Luồng Logic:**

1. **UI Event:** Người dùng gạt Switch.
2. **UI -> Cubit:** Hàm onChanged được kích hoạt, nó thực thi một lệnh duy nhất và trực tiếp: context.read<AppCubit>().toggleTheme().
3. **Cubit Handling:** Bên trong AppCubit, phương thức toggleTheme() thực hiện hai việc:

* **Cập nhật Trạng thái:** Nó tính toán ThemeMode mới (nếu đang là light thì đổi thành dark và ngược lại), sau đó phát ra một trạng thái AppState mới với giá trị theme đã được cập nhật: emit(state.copyWith(themeMode: newThemeMode));.
* **Lưu trữ Bền bỉ:** Ngay sau khi phát trạng thái, nó gọi await prefs.setBool(\_themeKey, ...) để lưu lựa chọn của người dùng vào SharedPreferences. Điều này đảm bảo rằng khi người dùng mở lại ứng dụng, AppCubit sẽ đọc được giá trị này và áp dụng chủ đề chính xác.

1. **Cập nhật Toàn bộ Giao diện:** Widget MaterialApp.router ở gốc của ứng dụng (trong AppView) đang lắng nghe AppCubit (final appState = context.watch<AppCubit>().state;). Khi AppCubit phát ra trạng thái mới, MaterialApp.router sẽ được xây dựng lại với themeMode mới (themeMode: appState.themeMode,), dẫn đến việc toàn bộ giao diện ứng dụng được cập nhật theo chủ đề mới.
2. **Thay đổi Ngôn ngữ**

* **UI Component:** Một ListTile chứa một DropdownButton. Giá trị hiện tại của nó được lấy từ context.locale, một thuộc tính được cung cấp bởi package easy\_localization.
* **Luồng Logic:**

1. **UI Event:** Người dùng chọn một ngôn ngữ mới từ danh sách trong DropdownButton.
2. **Xử lý bởi Package:** Hàm onChanged gọi context.setLocale(locale). Đây là một phương thức mở rộng (extension method) do easy\_localization cung cấp.
3. **Cập nhật Giao diện:** Package easy\_localization sẽ tự động xử lý việc tải file ngôn ngữ tương ứng từ thư mục assets/l10n, xây dựng lại tất cả các widget đang sử dụng các chuỗi văn bản được dịch (ví dụ: LocaleKeys.settings\_title.tr()), và tự động lưu lại lựa chọn ngôn ngữ của người dùng để sử dụng cho các lần sau.
4. **Đăng xuất**

* **UI Component:** Một ListTile với văn bản và biểu tượng được tô màu đỏ để cảnh báo người dùng về một hành động quan trọng.
* **Luồng Logic:**

1. **UI -> BLoC:** Khi người dùng nhấn vào, hàm onTap sẽ thêm sự kiện SignOutRequested() vào AuthBloc.
2. **BLoC -> Service:** AuthBloc nhận sự kiện và gọi phương thức await \_authService.signOut().
3. **Service Handling:** AuthService sẽ thực thi hai lệnh: await \_googleSignIn.signOut() (để đảm bảo đăng xuất hoàn toàn khỏi tài khoản Google) và await \_firebaseAuth.signOut().
4. **Cập nhật Trạng thái và Điều hướng:** Khi \_firebaseAuth.signOut() được gọi, authStateChanges của Firebase sẽ phát ra một giá trị null. AuthBloc (đang lắng nghe stream này) sẽ nhận được giá trị null và trong hàm \_onAuthUserChanged, nó sẽ phát ra trạng thái status: AuthStatus.guest. go\_router (đang lắng nghe AuthBloc) nhận thấy trạng thái đã thay đổi và logic redirect sẽ được kích hoạt, tự động chuyển hướng người dùng ra khỏi màn hình Cài đặt (vốn là một trang được bảo vệ) về trang mặc định cho khách (/chat).
5. **Điều hướng đến Trang Thay đổi Mật khẩu**

* **UI Component:** Một ListTile đơn giản.
* **Luồng Logic:**

1. **UI Event:** Người dùng nhấn vào mục "Đổi mật khẩu".
2. **Điều hướng:** Hàm onTap chỉ thực hiện một việc duy nhất là điều hướng bằng go\_router: context.push('/change-password'). Việc sử dụng .push() thay vì .go() cho phép màn hình ChangePasswordScreen được đẩy lên trên màn hình SettingsScreen như một ngăn xếp, và người dùng có thể quay lại bằng nút back trên AppBar. Toàn bộ logic xử lý việc thay đổi mật khẩu sẽ được thực hiện độc lập bên trong ChangePasswordScreen.

### Thách thức Kỹ thuật Chuyên sâu và Giải pháp Khắc phục

* **Thách thức: Tích hợp Nâng cao với Gemini API**
  + **Vấn đề:** Để AI có thể trả lời một cách có ngữ cảnh, không chỉ tin nhắn cuối cùng mà cả một phần lịch sử trò chuyện cần được gửi đi.
  + **Giải pháp chi tiết:** Trong AIService, phương thức getResponse không chỉ nhận một chuỗi văn bản, mà nhận cả một danh sách List<Message>. Một phương thức private \_mapMessagesToContent được tạo ra để chuyển đổi danh sách này thành định dạng List<Content> mà Gemini API yêu cầu. Nó lặp qua danh sách tin nhắn, gán vai trò user hoặc model cho từng tin nhắn để tạo ra một lịch sử trò chuyện có cấu trúc.
  + **Tạo tiêu đề tự động:** Em cũng đã triển khai một tính năng nâng cao là tự động tạo tiêu đề cho cuộc trò chuyện. Phương thức generateTitle trong AIService gửi một câu lệnh (prompt) đặc biệt đến Gemini, yêu cầu nó tóm tắt tin nhắn đầu tiên của người dùng thành một tiêu đề ngắn gọn.

## Phân tích và Nhận định Cá nhân

Qua quá trình thực hiện dự án, em đã có những quan sát và bài học kinh nghiệm sâu sắc:

* **Về kỹ thuật:** Việc lựa chọn kiến trúc BLoC kết hợp với go\_router đã tạo ra một nền tảng vững chắc, giúp quản lý các luồng dữ liệu và điều hướng phức tạp một cách hiệu quả. Thách thức lớn nhất là việc xử lý các tác vụ bất đồng bộ một cách hiệu quả, đảm bảo giao diện người dùng luôn phản hồi nhanh và không bị "đơ" khi chờ dữ liệu từ Firebase hay Gemini API.
* **Về quy trình:** Em đã học được tầm quan trọng của việc phân tích và thiết kế kỹ lưỡng trước khi bắt tay vào lập trình. Việc vẽ ra các luồng dữ liệu, thiết kế cấu trúc database và phác thảo giao diện đã giúp tiết kiệm rất nhiều thời gian sửa lỗi và tái cấu trúc sau này.
* **Về sản phẩm:** Mặc dù ứng dụng đã đáp ứng được các yêu cầu cơ bản, vẫn còn nhiều tiềm năng để phát triển thêm như: cho phép trò chuyện bằng giọng nói, lưu trữ tin nhắn offline, hoặc tối ưu hóa câu truy vấn Firestore để xử lý lượng dữ liệu lớn hơn.

## Kết luận chương 2

Chương 2 đã trình bày một cách toàn diện quá trình xây dựng ứng dụng Chatbot AI, từ khâu lên ý tưởng, lựa chọn công nghệ, thiết kế hệ thống cho đến triển khai chi tiết từng chức năng. Các kết quả đạt được không chỉ là một sản phẩm phần mềm hoạt động được mà còn là những kiến thức và kinh nghiệm thực tiễn vô cùng quý báu, là nền tảng vững chắc cho sự nghiệp phát triển phần mềm của em trong tương lai.

# KẾT QUẢ THỰC TẬP

Chương này tổng hợp các kết quả cụ thể đã đạt được trong suốt 8 tuần thực tập tại Công ty Cổ phần Công nghệ AppsCyclone. Nội dung chính bao gồm việc tổng kết các thành tựu về mặt sản phẩm và kỹ năng, nhật ký công việc chi tiết hàng tuần, và các biểu mẫu ghi nhận, đánh giá quá trình thực tập từ phía chuyên gia doanh nghiệp và giảng viên hướng dẫn.

## Tổng kết các kết quả cụ thể đã đạt được

Kết thúc kỳ thực tập, với sự hướng dẫn của chuyên gia tại công ty và giảng viên nhà trường, em đã hoàn thành tốt các nhiệm vụ được giao và đạt được những kết quả cụ thể, rõ ràng trên nhiều phương diện:

### Về mặt sản phẩm

Kết quả hữu hình và quan trọng nhất của kỳ thực tập là đã xây dựng thành công một sản phẩm phần mềm hoàn chỉnh từ đầu đến cuối: **ứng dụng Chatbot AI**, có khả năng hoạt động ổn định trên nền tảng Android. Sản phẩm cuối cùng là một file APK (Android Package Kit) đã được build ở chế độ release, sẵn sàng để cài đặt và sử dụng trên thiết bị thật.

Ứng dụng bao gồm đầy đủ các chức năng đã đề ra trong bản thiết kế:

* **Hệ thống xác thực người dùng đa phương thức:** Cho phép người dùng đăng ký tài khoản mới, đăng nhập/đăng xuất bằng Email và Mật khẩu, hoặc đăng nhập nhanh chóng và tiện lợi thông qua tài khoản Google.
* **Chức năng quản lý tài khoản an toàn:** Triển khai đầy đủ các luồng chức năng cần thiết như "Quên mật khẩu" (dành cho người dùng chưa đăng nhập) và "Thay đổi mật khẩu" (dành cho người dùng đã đăng nhập).
* **Chức năng trò chuyện thông minh:** Người dùng có thể tương tác, đặt câu hỏi và nhận phản hồi từ mô hình Gemini AI theo thời gian thực.
* **Quản lý cuộc trò chuyện hiệu quả:** Người dùng đã đăng nhập có thể tạo, lưu trữ nhiều cuộc trò chuyện khác nhau và dễ dàng truy cập, quản lý hoặc xóa chúng thông qua thanh điều hướng (Drawer).
* **Chế độ khách cho phép trải nghiệm:** Người dùng mới có thể sử dụng thử ứng dụng với giới hạn 10 lượt tin nhắn trước khi được khuyến khích đăng ký tài khoản.
* **Giao diện người dùng thân thiện, có khả năng tùy chỉnh:**
* Hỗ trợ chuyển đổi giữa giao diện Sáng (Light Mode) và Tối (Dark Mode).
* Hỗ trợ đa ngôn ngữ (Tiếng Việt và Tiếng Anh).
* **Tối ưu hiệu năng:** Áp dụng kỹ thuật Lazy Loading để tải dần lịch sử tin nhắn, đảm bảo ứng dụng hoạt động mượt mà ngay cả với các cuộc trò chuyện dài.

### Về mặt kiến thức và kỹ năng

Bên cạnh sản phẩm cụ thể, kỳ thực tập đã giúp em củng cố và tích lũy được nhiều kiến thức, kỹ năng quan trọng:

* **Về kiến thức công nghệ:**
* **Làm chủ Flutter và ngôn ngữ Dart:** Nắm vững cách xây dựng giao diện người dùng, xử lý các tác vụ bất đồng bộ và cấu trúc một dự án Flutter hoàn chỉnh.
* **Áp dụng thành thạo kiến trúc BLoC:** Hiểu sâu sắc và có thể tự triển khai kiến trúc BLoC để quản lý trạng thái cho các ứng dụng phức tạp, đảm bảo mã nguồn sạch sẽ, dễ bảo trì và mở rộng.
* **Sử dụng và tích hợp các dịch vụ Firebase:** Có kinh nghiệm thực tế trong việc tích hợp và sử dụng Firebase Authentication cho việc xác thực và Cloud Firestore cho việc lưu trữ, truy vấn dữ liệu thời gian thực.
* **Làm việc với API bên ngoài:** Nắm được cách gọi và xử lý dữ liệu trả về từ một API của bên thứ ba (Gemini AI API).
* **Sử dụng các package quan trọng:** Thành thạo trong việc sử dụng các thư viện phổ biến trong hệ sinh thái Flutter như go\_router, easy\_localization, shared\_preferences, và equatable.
* **Về kỹ năng mềm:**
* **Kỹ năng giải quyết vấn đề:** Rèn luyện khả năng phân tích một vấn đề kỹ thuật, tìm kiếm các giải pháp khả thi và lựa chọn phương án tối ưu để triển khai.
* **Kỹ năng làm việc theo quy trình:** Làm quen với quy trình phát triển phần mềm cơ bản, từ khâu phân tích yêu cầu, thiết kế, triển khai cho đến kiểm thử.
* **Kỹ năng tự học và nghiên cứu:** Chủ động tìm hiểu tài liệu kỹ thuật, đọc mã nguồn của các thư viện và học hỏi các công nghệ mới để áp dụng vào dự án.

### Về mặt tài liệu

* Hoàn thành quyển báo cáo thực tập tốt nghiệp chi tiết, trình bày toàn bộ quá trình thực hiện dự án từ phân tích, thiết kế, triển khai cho đến kết quả và đánh giá.
* Tạo ra bộ mã nguồn hoàn chỉnh, có cấu trúc tốt, sẵn sàng cho việc bảo trì và phát triển trong tương lai.

Chi tiết quá trình thực tập tại công ty qua các tuần, em xin được trình bày trong các bảng sau:

**BẢNG GHI NHẬN KẾT QUẢ THỰC TẬP HÀNG TUẦN**

|  |  |
| --- | --- |
| TRƯỜNG ĐẠI HỌC SÀI GÒN  **KHOA CÔNG NGHỆ THÔNG TIN** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh phúc**  *Tp. Hồ Chí Minh, ngày … tháng …. năm 20…* |

Mẫu 06\_TTTN\_k.CNTT

**BẢNG GHI NHẬN KẾT QUẢ THỰC TẬP HÀNG TUẦN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Một số thông tin liên hệ** | | | | |
| Họ và tên: Hoàng Ngọc Đại Phước | | |  | |
| Ngày sinh: 03/11/2003 | | |  | |
| Mã số sinh viên: 3121411170 | | |  | |
| Lớp: DCT121C2 | | |  | |
| Ngành học: Công nghệ thông tin | | |  | |
| Email: daiphuoc1497@gmail.com | | |  | |
| Điện thoại: 0984253741 | | |  | |
|  | | |  | |
| Chuyên gia doanh nghiệp: Cao Nhật Toàn | | |  | |
| Email: toan.cao@apps-cyclone.com | | |  | |
| Điện thoại: 0353469594 | | |  | |
|  | | |  | |
| Giảng viên hướng dẫn: Nguyễn Tuấn Đăng | | |  | |
| Email: dangnt@sgu.edu.vn | | |  | |
| Điện thoại: | | |  | |
|  | **Tuần** | **Nội dung thực tập**  **(do chuyên gia của doanh nghiệp giao)** | | **Kết quả thực tập**  **(do chuyên gia của doanh nghiệp đánh giá)** |
|  | 1  Từ ngày 16 /06/ 2025  đến ngày 22/ 06/ 2025 | - **Research Dart syntax**  Đọc kỹ tài liệu về Dart Language tại <https://dart.dev/language>. Ghi chú các cú pháp cơ bản: biến, kiểu dữ liệu, hàm, class, collection, điều kiện, vòng lặp, extension, mixin...  Thực hành trên <https://dartpad.dev/> với từng tính năng vừa học.  **Ví dụ:**   * Viết một class Person có trường tên và tuổi, hàm giới thiệu bản thân. * Tạo 1 danh sách số nguyên, duyệt qua và in ra các số chẵn.   **Báo cáo:**   * Gửi file ghi chú tổng hợp lý thuyết (có ví dụ minh hoạ) * Gửi file code thực hành   - **Project Initial**  Khởi tạo project Flutter mới, cài các công cụ:   * fvm để quản lý phiên bản Flutter (tham khảo <https://fvm.app/documentation/getting-started/installation>) * easy\_localization để hỗ trợ đa ngôn ngữ (<https://pub.dev/packages/easy_localization>) * go\_router cho điều hướng (<https://pub.dev/packages/go_router>)   Thiết lập chức năng đổi theme (light/dark mode).  **Ví dụ thực tế:**   * Có 2 màn hình: Trang chủ (HomePage) và Trang chi tiết (DetailPage), điều hướng qua lại dùng go\_router. * Hiển thị 1 câu chào (Xin chào! hoặc Hello!) và nút đổi ngôn ngữ. * Có nút đổi theme.   **Output:** Source code đầy đủ, mỗi chức năng phải có hướng dẫn/test được.  **- Sample UI:**  So sánh CupertinoApp vs MaterialApp (khi nào dùng, khác gì về giao diện).  Dùng Scaffold tạo layout có AppBar, Body, Drawer, BottomNav.  Thực hành các widget: Row, Column, Wrap, Expanded, Flexible.  **Ví dụ:**   * Tạo màn hình với 3 ô vuông nằm ngang (Row), thử thay bằng Column hoặc Wrap để xem sự khác biệt. * Sử dụng Expanded để phân chia tỉ lệ các ô. * Thêm padding, căn lề cho đẹp mắt.   **Output:** Ảnh chụp màn hình + file code UI demo.  **- List UI:**  Học và trình bày sự khác nhau giữa ListView.builder, ListView.separated, CustomScrollView.  Thêm tính năng pull-to-refresh, load-more khi cuộn cuối danh sách.  **Ví dụ:**   * Tạo danh sách hiển thị tên các quốc gia. * Khi kéo xuống cuối sẽ tự động load thêm 5 quốc gia mới (fake data). * Thêm nút refresh ở đầu danh sách.   **Output:** Video/ảnh minh hoạ + file code widget list.  - **Assets & Image**  Hiển thị ảnh từ asset (ảnh thêm vào project), từ internet (dùng URL), và SVG.  Dùng các package: cached\_network\_image, flutter\_svg, Image widget chuẩn.  **Ví dụ:**   * Hiển thị logo Flutter từ asset. * Hiển thị ảnh nền từ url: '<https://picsum.photos/200>'. * Hiển thị icon vector dạng SVG (có thể dùng svg online hoặc file local). * Viết một widget nhận vào source (asset, url, svg) và hiển thị đúng, có thuộc tính width/height.   **Output:** File code widget, hướng dẫn add asset vào pubspec.yaml. | |  |
|  | 2  Từ ngày 23/ 06/ 2025  đến ngày 29/ 06/ 2025 | - **Form Widget**  Tạo Form Widget có validate dữ liệu từng trường, che/mở password, đổi màu border khi có lỗi.  **Ví dụ:**   * Form đăng ký với trường Email (kiểm tra đúng định dạng), Password (>=6 ký tự), Confirm Password (khớp Password). * Có nút Đăng ký. Khi bấm sẽ hiển thị lỗi bên dưới từng trường nếu nhập sai.   **Output:** Ảnh chụp giao diện + code, ghi chú ý chính về validate form.  - **RxDart**  Tìm hiểu stream, streamcontroller, subject (BehaviorSubject, PublishSubject).  Dùng RxDart để quản lý dữ liệu bất đồng bộ.  **Ví dụ:**   * Tạo một stream count tăng dần khi bấm nút, hiển thị giá trị mới ra màn hình. * Viết demo về việc search list (search có debounce, không query liên tục).   **Output:** File code ví dụ, giải thích các loại subject khác nhau.  - **Bloc - Cubit**  Nắm rõ pattern Bloc, Cubit để quản lý state.  Phân biệt Bloc và Cubit (so sánh qua ví dụ thực tế).  **Ví dụ:**   * Bloc quản lý login (đang login, thành công, thất bại), * Cubit quản lý đổi theme (sáng/tối), đổi ngôn ngữ. * Tách riêng file cho mỗi Bloc/Cubit, không gộp vào 1 file.   **Output:** Code mẫu + sơ đồ/cây state, giải thích điểm mạnh/yếu. | |  |
|  | 3  Từ ngày 30/ 06/ 2025  đến ngày  06/ 07/ 2025 | **- Firebase**  Tích hợp các dịch vụ Firebase: remote config, analytics, crashlytics, auth, messaging, firestore.  **Ví dụ:**   * Đăng nhập Google/Facebook bằng firebase\_auth. * Lưu dữ liệu note đơn giản lên firestore. * Lấy thông tin từ remote config để đổi theme app. * Gửi notification test từ Firebase Console, nhận notification trong app.   **Output:** Source code app, ảnh/video test thực tế.  - **Build APK Release**  Build app Android dạng release.  Tạo file keystore (tham khảo <https://docs.flutter.dev/deployment/android>).  Build APK release, thử cài trên điện thoại thật.  **Hướng dẫn:**   * Thêm file keystore vào android/app, cấu hình build.gradle đúng. * Chạy lệnh: flutter build apk --release   **Output:** File APK, note lại các bước gặp lỗi và cách xử lý.  - **Offline State**  So sánh shared\_preferences và flutter\_secure\_storage (ưu/nhược điểm).  Cấu hình offline mode cho firestore, remote config.  **Ví dụ:**   * Lưu token đăng nhập xuống shared\_preferences, đóng app mở lại vẫn đăng nhập. * Khi tắt wifi, thử đọc dữ liệu firestore/remote config xem app có chạy được không.   **Output:** Bảng so sánh, code demo, ảnh/video minh hoạ test offline.  - **Reuse Widget Workshop**  Refactor lại 1 màn hình code cũ thành widget tái sử dụng, có parameter cho các thuộc tính chính.  **Ví dụ:**   * Widget Button có tham số: text, color, onTap. * Widget thông báo lỗi có thể dùng lại ở mọi màn hình.   Tham gia workshop, trình bày kinh nghiệm cá nhân.  **Output:** File code widget reuse, slide hoặc ghi chú ngắn về kết quả học được. | |  |
|  | 4  Từ ngày 07/ 07/ 2025  đến ngày 13/ 07/ 2025 | **Thiết kế Giao diện Chat:** Xây dựng UI cho màn hình trò chuyện, bao gồm danh sách tin nhắn, khung nhập liệu và các trạng thái tin nhắn.  **Tùy chỉnh Giao diện (App Theme):** Sử dụng BLoC/Cubit để cho phép người dùng chuyển đổi giữa giao diện Sáng (Light) và Tối (Dark).  **Hỗ trợ Đa ngôn ngữ:** Tích hợp package easy\_localization để ứng dụng có thể hiển thị bằng Tiếng Việt và Tiếng Anh. | |  |
|  | 5  Từ ngày 14/ 07/ 2025  đến ngày 20/ 07/ 2025 | **Tích hợp Gemini AI:** Kết nối với API của Gemini để gửi yêu cầu từ người dùng và nhận phản hồi từ AI.  **Chat Thời gian thực:** Sử dụng BLoC và Firestore để quản lý và hiển thị tin nhắn mới ngay lập tức.  **Xây dựng Chức năng Xác thực:** Hoàn thiện các màn hình và chức năng Đăng ký, Đăng nhập, Đăng xuất, Quên mật khẩu, và Thay đổi mật khẩu   * **Tối ưu Hiệu năng (Lazy Loading):** Áp dụng kỹ thuật tải dần (infinite scroll) cho lịch sử trò chuyện để ứng dụng chạy mượt mà.   **Làm mới Cuộc trò chuyện:** Thêm chức năng cho phép người dùng làm mới hoặc xóa nội dung cuộc trò chuyện hiện tại. | |  |
|  | 6  Từ ngày 21/ 07/ 2025  đến ngày 27/ 07/2025 | **Chế độ Khách (Guest Mode):** Triển khai logic giới hạn 10 tin nhắn cho người dùng chưa đăng nhập, sau đó hiển thị yêu cầu đăng nhập.  **Quản lý Nhiều Cuộc trò chuyện:** Cho phép người dùng tạo, lưu trữ và chuyển đổi giữa các cuộc hội thoại khác nhau.  **Tối ưu Hiệu năng (Lazy Loading):** Áp dụng kỹ thuật tải dần (infinite scroll) cho lịch sử trò chuyện để ứng dụng chạy mượt mà.  **Làm mới Cuộc trò chuyện:** Thêm chức năng cho phép người dùng làm mới hoặc xóa nội dung cuộc trò chuyện hiện tại. | |  |
|  | 7  Từ ngày 28/ 07/ 2025  đến ngày 03/ 08/ 2025 | **Refactor Code:** Rà soát và cấu trúc lại code, tách các widget lớn thành các thành phần nhỏ có thể tái sử dụng.  **Kiểm thử Chức năng:** Viết và thực hiện các kịch bản kiểm thử cho tất cả các luồng chính của ứng dụng.  **Sửa lỗi:** Ghi nhận và khắc phục các lỗi phát sinh trong quá trình kiểm thử. | |  |
|  | 8  Từ ngày 4/ 08/ 2025  đến ngày 08/ 08/ 2025 | **Build ứng dụng:** Tạo file cài đặt APK/AAB ở chế độ release, cấu hình keystore.  **Kiểm tra trên Thiết bị thật:** Cài đặt và kiểm tra lần cuối trên thiết bị di động thật để đảm bảo hoạt động ổn định.  **Hoàn thiện Tài liệu và Báo cáo:** Hoàn tất báo cáo thực tập, tổng kết các kiến thức và kinh nghiệm đã học được. | |  |

**Chuyên gia doanh nghiệp hướng dẫn thực tập**

(Ký tên và ghi họ tên)

**Ghi chú:**

-**Chuyên gia doanh nghiệp** ghi nhận kết quả thực tập của **sinh viên** theo tuần và gởi qua email cho **giảng viên hướng dẫn** khi kết thúc các tuần 3,6 của đợt thực tập. BẢNG GHI NHẬN KẾT QUẢ THỰC TẬP TỐT NGHIỆP HÀNG TUẦN này là một trong những hồ sơ kèm theo quyển báo cáo thực tập tốt nghiệp.

-Cột **Kết quả thực tập**, chuyên gia doanh nghiệp có thể ghi *Hoàn thành tốt*, *Hoàn thành*, *Không đạt* hoặc có thể ghi nhận chi tiết hơn.

**BẢNG ĐÁNH GIÁ QUÁ TRÌNH THỰC TẬP TỐT NGHIỆP**

|  |  |
| --- | --- |
| **ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH** **TRƯỜNG ĐẠI HỌC SÀI GÒN**  **KHOA CNTT** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh phúc** |

Mẫu 07\_TTTN\_k.CNTT

**BẢNG ĐÁNH GIÁ QUÁ TRÌNH THỰC TẬP TỐT NGHIỆP**

**(do chuyên gia doanh nghiệp đánh giá).**

Họ và tên sinh viên: Hoàng Ngọc Đại Phước

Ngày sinh: 03/11/2003

Mã số sinh viên: 3121411170

Lớp: DCT121C2

Thời gian thực tập: 16/6/2025 – 9/8/2025

Doanh nghiệp thực tập: Apps Cyclone Technology JSC

Địa chỉ doanh nghiệp: 168/6 Bùi Thị Xuân, Phường 3, Tân Bình, Hồ Chí Minh

Chuyên gia doanh nghiệp hướng dẫn: Cao Nhật Toàn

**I. ĐÁNH GIÁ VỀ QUÁ TRÌNH THỰC TẬP**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Nội dung đánh giá** | **ĐIỂM** | | |
| **0** | **0.5** | **1** |
| 1 | Khả năng thực hành |  |  |  |
| 2 | Khả năng làm việc nhóm |  |  |  |
| 3 | Tính thân thiện |  |  |  |
| 4 | Tính năng động |  |  |  |
| 5 | Tính thần sáng tạo |  |  |  |
| 6 | Chấp hành nội quy cơ quan |  |  |  |
| 7 | Giờ giấc làm việc |  |  |  |
| 8 | Phương pháp làm việc |  |  |  |
| 9 | Khối lượng công việc |  |  |  |
| 10 | Báo cáo thực tập tốt nghiệp |  |  |  |

(theo thang điểm 10).

**II. CÁC ĐÁNH GIÁ KHÁC:** .............................................................................................................................................................................................................................................................................................................................................................................................................................................**III. KẾT QUẢ TỔNG HỢP:**

**Điểm tổng cộng :**

|  |  |
| --- | --- |
| **XÁC NHẬN CỦA DOANH NGHIỆP**  (đóng mộc tròn của doanh nghiệp, họ tên, ký tên) | **Chuyên gia hướng dẫn**  *(Ký và ghi họ tên)* |

**PHIẾU ĐÁNH GIÁ KẾT QUẢ THỰC TẬP TỐT NGHIỆP**

Mẫu 08\_TTTN\_k.CNTT

|  |  |
| --- | --- |
| **ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH** **TRƯỜNG ĐẠI HỌC SÀI GÒN**  **KHOA CNTT** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh phúc** |

**PHIẾU ĐÁNH GIÁ KẾT QUẢ THỰC TẬP TỐT NGHIỆP**

**(do giảng viên hướng dẫn đánh giá)**

Họ và tên sinh viên: Hoàng Ngọc Đại Phước

Ngày sinh: 03/11/2003

Mã số sinh viên: 3121411170

Lớp: DCT121C2

Thời gian thực tập: 16/6/2025 – 9/8/2025

Doanh nghiệp thực tập: Apps Cyclone Technology JSC

Địa chỉ doanh nghiệp: 168/6 Bùi Thị Xuân, Phường 3, Tân Bình, Hồ Chí Minh

Chuyên gia doanh nghiệp hướng dẫn: Cao Nhật Toàn

**I. ĐIỂM CỦA CHUYÊN GIA DOANH NGHIỆP :**

(thang điểm 10)

**II. ĐIỂM CỦA GIẢNG VIÊN HƯỚNG DẪN:**

(thang điểm 10)

**II. ĐIỂM TỔNG KẾT:**

(trung bình cộng 2 cột điểm trên, thang điểm 10)

**Xếp loại :**

*TP Hồ Chí Minh ngày tháng năm*

**Giảng viên hướng dẫn**

***GHI CHÚ :***

*Xếp loại được tính như sau:*

*Xếp loại Xuất sắc nếu điểm tổng hợp từ 9 đến 10*

*Xếp loại Giỏi nếu điểm tổng hợp từ 8 đến cận 9*

*Xếp loại Khá nếu điểm tổng hợp từ 7 đến cận 8*

*Xếp loại TB Khá nếu điểm tổng hợp từ 6 đến cận 7*

*Xếp loại TB nếu điểm tổng hợp từ 5 đến cận 6*

*Xếp loại Không đạt nếu điểm tổng hợp từ 0 đến cận 5*

## Kết luận chương 3

Chương 3 đã tổng hợp một cách chi tiết các công việc và kết quả đạt được trong suốt kỳ thực tập. Nhật ký công việc hàng tuần cho thấy sự tiến bộ rõ rệt trong việc áp dụng kiến thức và kỹ năng để hoàn thành một dự án phần mềm hoàn chỉnh. Các kết quả cụ thể về sản phẩm, kiến thức, kỹ năng cùng các biểu mẫu đánh giá đính kèm là minh chứng khách quan cho sự nỗ lực và thành quả mà em đã đạt được dưới sự hướng dẫn của công ty và nhà trường.

# KẾT LUẬN VÀ KIẾN NGHỊ

## Kết luận

Kỳ thực tập tốt nghiệp tại Công ty Apps Cyclone Technology JSC là một giai đoạn học hỏi và trải nghiệm vô cùng quý báu, là cầu nối quan trọng giúp em liên kết giữa những kiến thức lý thuyết đã được trang bị tại nhà trường với môi trường làm việc thực tế đầy năng động và chuyên nghiệp.

Trong suốt 8 tuần thực tập, dưới sự hướng dẫn tận tình của chuyên gia tại công ty và giảng viên hướng dẫn, em đã hoàn thành tốt nhiệm vụ được giao là xây dựng ứng dụng "Chatbot AI sử dụng Flutter". Qua đó, em không chỉ đạt được mục tiêu về mặt sản phẩm là một ứng dụng di động hoàn chỉnh, mà quan trọng hơn là đã tích lũy được những kiến thức chuyên sâu và kỹ năng thực tiễn giá trị.

Em đã có cơ hội áp dụng và làm chủ các công nghệ hiện đại như Flutter và Dart để xây dựng giao diện, áp dụng thành thạo kiến trúc BLoC để quản lý trạng thái phức tạp, và tích hợp các dịch vụ backend mạnh mẽ từ Firebase. Đặc biệt, việc làm việc với Gemini AI API đã cho em một cái nhìn sâu sắc hơn về lĩnh vực Trí tuệ nhân tạo và cách tích hợp nó vào các sản phẩm phần mềm.

Bên cạnh kiến thức chuyên môn, em còn rèn luyện được các kỹ năng mềm quan trọng như kỹ năng phân tích vấn đề, tự nghiên cứu tài liệu, làm việc theo quy trình và quản lý thời gian. Kỳ thực tập đã thực sự giúp em trưởng thành hơn, tự tin hơn và có một định hướng rõ ràng hơn cho con đường sự nghiệp sau khi tốt nghiệp.

## Kiến nghị

Để quá trình thực tập và phát triển dự án được tốt hơn trong tương lai, em xin đưa ra một số kiến nghị như sau:

### Hướng phát triển của đề tài

Sản phẩm ứng dụng Chatbot AI đã hoàn thành các chức năng cơ bản, tuy nhiên vẫn còn nhiều tiềm năng để cải tiến và phát triển thêm các tính năng nâng cao, bao gồm:

* **Hỗ trợ Offline:** Xây dựng cơ chế lưu trữ dữ liệu tạm thời trên thiết bị, cho phép người dùng xem lại các cuộc trò chuyện cũ ngay cả khi không có kết nối Internet.
* **Tương tác bằng giọng nói:** Tích hợp các công nghệ nhận dạng giọng nói (Speech-to-Text) và tổng hợp giọng nói (Text-to-Speech) để người dùng có thể trò chuyện với AI một cách tự nhiên hơn mà không cần gõ phím.
* **Tìm kiếm trong cuộc trò chuyện:** Xây dựng chức năng tìm kiếm, cho phép người dùng dễ dàng tìm lại các tin nhắn cũ dựa trên từ khóa.
* **Mở rộng nền tảng:** Tận dụng thế mạnh đa nền tảng của Flutter để triển khai ứng dụng trên các nền tảng khác như iOS hoặc Web.

### Kiến nghị với doanh nghiệp (Công ty Apps Cyclone)

Trước hết, em xin một lần nữa cảm ơn công ty đã tạo điều kiện tốt nhất cho em trong suốt kỳ thực tập. Để công tác hướng dẫn sinh viên thực tập trong các đợt tới được hiệu quả hơn nữa, em xin có một vài đề xuất nhỏ:

* Xem xét xây dựng một chương trình "onboarding" hoặc một dự án khởi động nhỏ dành cho thực tập sinh mới, giúp các bạn nhanh chóng làm quen với quy trình làm việc và bộ công cụ của công ty.
* Tổ chức thêm các buổi chia sẻ kỹ thuật (technical sharing) định kỳ, nơi các thực tập sinh có thể trình bày về những khó khăn, giải pháp của mình và học hỏi kinh nghiệm từ các lập trình viên có kinh nghiệm.

### Kiến nghị với Khoa và Nhà trường (Khoa CNTT - ĐH Sài Gòn)

Em xin chân thành cảm ơn Khoa và Nhà trường đã tạo ra chương trình thực tập rất hữu ích này. Để nâng cao hơn nữa chất lượng đào tạo và giúp sinh viên có sự chuẩn bị tốt hơn, em xin kiến nghị:

* Tăng cường các buổi workshop, seminar về các công nghệ và framework mới đang có nhuệ cao trên thị trường như Flutter, React Native và các kiến trúc quản lý trạng thái hiện đại như BLoC, Redux.
* Tiếp tục đẩy mạnh mối quan hệ hợp tác với các doanh nghiệp phần mềm để tạo ra nhiều cơ hội thực tập chất lượng hơn cho sinh viên, đồng thời giúp nhà trường nắm bắt được các yêu cầu thực tế của ngành để kịp thời điều chỉnh chương trình đào tạo.

# TÀI LIỆU THAM KHẢO

1. Learn Google Flutter Fast: 65 Example Apps by Mark Clow
2. Beginning Flutter: A Hands On Guide to App Development by Marco L. Napoli