TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

**KHOA: ĐIỆN TỬ**

**Bộ môn: Tin Học Công Nghiệp**



**BÀI TẬP LỚN**

MÔN HỌC

**THỰC TẬP CHUYÊN NGÀNH**

**Họ và tên : Kun Veng Ann**

**Mã Số sinh viên : CPC205013**

**Ngành học : Kỹ thuật Máy tính**

**Lớp : K56KMT.01**

**Giảng viên hướng dẫn : Th.S Trần Thị Ngọc Linh**

**Thái nguyên-2024**

|  |  |
| --- | --- |
| **TRƯỜNG ĐHKCN** | **CỘNG HÓA XÃ HỘI CHỦ NGHĨA VIỆT NAM** |
| **KHOA ĐIẸN TỬ** | **Độc lập – Tự do – Hạnh phúc** |

**NHIỆM VU BÀI TẬP LỚN KHOA HỌC DỮ LIỆU**

|  |  |
| --- | --- |
| Sinh viên: **Kun Veng Ann** | Mã số sinh viên: **CPC205013** |
| Lớp: **K56KMT.01** | Ngành: **Tin học công nghiệp** |

Giáo viên hướng dẫn: **Trần Thị Ngọc Linh**

1. **Tên đề tài:**

**Xây dựng ứng dụng ghị lịch nhắc online ( giống Google calendar ) trên thiết bị di động.**

1. **Nội dung thực hiện:**

- Khảo sát, phân tích bài toán

- Phân tích cơ sở dữ liệu cho bài toán

- Xây dựng phần mềm trên thiết bị di động ghị nhắc

1. **Kết quả**

* Sản phẩm phần mềm thêo yêu cầu của đề tài
* Thuyết minh đồ án theo mẫu chung của khóa điện tử

Ngày giao đề: **.../.../2024**. Ngày hoàn thành: **.../.../2024**

**GIÁO VIÊN HƯỚNG DẪN**

(Ký và ghi rõ họ tên)

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

Thái Nguyên, ngày….tháng…..năm 20....

GIÁO VIÊN HƯỚNG DẪN

*(Ký ghi rõ họ tên)*

# **MỤC LỤC**

[**MỤC LỤC** 3](#_Toc169388867)

[**LỜI NÓI ĐẦU** 5](#_Toc169388868)

[**CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI** 6](#_Toc169388869)

[**1.1. Bối cảnh và Lý do chọn đề tài** 6](#_Toc169388870)

[**1.2. Mục tiêu của đề tài** 7](#_Toc169388871)

[**1.3. Cấu trúc của báo cáo** 7](#_Toc169388872)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT** 8](#_Toc169388873)

[**2.1. Kiến thức cơ bản về phát triển ứng dụng di động** 8](#_Toc169388874)

[**2.2. Ứng dụng của lập trình thiết bị di động** 10](#_Toc169388875)

[**CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ** 12](#_Toc169388876)

[**3.1 Phân tích yêu cầu** 12](#_Toc169388877)

[3.1.1 Yêu cầu chức năng 12](#_Toc169388878)

[3.1.2 Yêu cầu phi chức năng 13](#_Toc169388879)

[**3.2 Lựa chọn công cụ** 14](#_Toc169388880)

[3.2.1 Flutter 15](#_Toc169388881)

[3.2.2. MySQL và Laravel 17](#_Toc169388882)

[**3.3 Thiết kế hệ thống** 19](#_Toc169388883)

[3.3.1 Kiến trúc ứng dụng 19](#_Toc169388884)

[3.2.2 Các Thành Phần Chính 19](#_Toc169388885)

[3.3.3 Cơ sở dữ liệu 20](#_Toc169388886)

[3.3.4 Giao diện người dùng 26](#_Toc169388887)

[3.3.5 Quy trình hoạt động 26](#_Toc169388888)

[**CHƯƠNG 4: TRIỂN KHAI VÀ KIỂM THỬ** 30](#_Toc169388889)

[**4.1 Triển khai** 30](#_Toc169388890)

[4.1.1 Chuẩn bị môi trường phát triển (Development Environment) 30](#_Toc169388891)

[4.1.2 Triển khai mã nguồn 30](#_Toc169388892)

[4.1.3 Lập trình logic Ứng dụng ghi nhắc **Error! Bookmark not defined.**](#_Toc169388893)

[4.2 Kiểm thử 31](#_Toc169388894)

[**CHƯƠNG 5: KẾT LUẬN VÀ ĐÁNH GIÁ** 35](#_Toc169388895)

[**5.1 Kết luận** 35](#_Toc169388896)

[5.1.1 Tổng quan về dự án 35](#_Toc169388897)

[5.1.2 Các chức năng chính 35](#_Toc169388898)

[5.1.3 Quá trình phát triển 35](#_Toc169388899)

[**5.2 Đánh giá** 35](#_Toc169388900)

[5.2.1 Điểm mạnh 35](#_Toc169388901)

[5.2.2 Điểm yếu 36](#_Toc169388902)

[5.2.3 Bài học kinh nghiệm 36](#_Toc169388903)

[**5.3 Hướng phát triển tương lai** 36](#_Toc169388904)

[**5.4 Kết luận** 36](#_Toc169388905)

[**TÀI LIỆU THAM KHẢO** 37](#_Toc169388906)

# **LỜI NÓI ĐẦU**

Trong thời kỳ công nghiệp hóa, hiện đại hóa ngày nay, các thành tựu liên quan đến khoa học dữ liệu và công nghệ thông tin đang được con người mở rộng và phát triển mạnh mẽ. Một trong những xu hướng nổi bật nhất là sự phát triển của các ứng dụng di động, giúp con người giải quyết các vấn đề hàng ngày một cách tiện lợi và hiệu quả.

Sự phổ biến của điện thoại thông minh đã thay đổi cách chúng ta quản lý công việc và cuộc sống. Các ứng dụng di động hiện nay không chỉ giúp người dùng kết nối, giải trí, mà còn hỗ trợ quản lý thời gian và công việc một cách khoa học. Việc ghi nhớ các sự kiện, công việc hàng ngày, và nhận thông báo nhắc nhở trở nên vô cùng quan trọng trong cuộc sống bận rộn ngày nay.

Nhận thấy nhu cầu thiết yếu này, chúng em đã chọn đề tài **"Xây dựng ứng dụng ghi lịch nhắc online (giống Google Calendar) trên thiết bị di động"**. Ứng dụng này sẽ giúp người dùng tổ chức và quản lý lịch trình một cách hiệu quả, giúp họ không bỏ lỡ các sự kiện quan trọng và nâng cao hiệu suất công việc.

Trong quá trình thực hiện đề tài, chúng em đã nghiên cứu và áp dụng nhiều công nghệ tiên tiến nhằm phát triển một ứng dụng đa nền tảng, dễ sử dụng và đáp ứng tốt nhu cầu của người dùng. Ứng dụng sẽ hỗ trợ quản lý sự kiện, công việc, và đồng bộ hóa dữ liệu trên nhiều thiết bị, đảm bảo tính tiện lợi và nhất quán trong việc quản lý thời gian.

Để thực hiện được đề tài này, chúng em xin gửi lời cảm ơn chân thành nhất đến tất cả các thầy cô trong bộ môn đã hỗ trợ chúng em suốt thời gian qua. Đặc biệt, chúng em xin chân thành cảm ơn cô giáo hướng dẫn **Trần Thị Ngọc Linh**, đã nhiệt tình hướng dẫn, chỉ bảo và cung cấp cho chúng em nhiều kiến thức cũng như tài liệu trong suốt quá trình làm đề tài.

Em xin chân thành cảm ơn!

# **CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI**

# **1.1. Bối cảnh và Lý do chọn đề tài**

Trong thời đại hiện đại, sự phát triển nhanh chóng của công nghệ thông tin đã mang lại nhiều tiện ích và thay đổi đáng kể cách con người quản lý thời gian và công việc. Đặc biệt là với sự phổ biến của điện thoại di động và các thiết bị thông minh, người dùng ngày càng có nhu cầu cao hơn về các ứng dụng hỗ trợ quản lý lịch trình và nhắc nhở.

Các ứng dụng ghi lịch nhắc nhở đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày, giúp người dùng tổ chức thời gian một cách hiệu quả, từ việc ghi nhớ các sự kiện quan trọng đến quản lý công việc hàng ngày. Việc sử dụng các ứng dụng này không chỉ đơn thuần là giải quyết vấn đề về quản lý thời gian mà còn tạo ra thói quen làm việc khoa học và nâng cao năng suất.

**+ Lý do chọn đề tài**

* Nhu cầu và tiềm năng thị trường: Việc xây dựng một ứng dụng ghi lịch nhắc online trên thiết bị di động mang lại nhiều tiện ích cho người dùng hiện nay. Đây là một lĩnh vực có tiềm năng lớn với số lượng người dùng di động ngày càng tăng cao.
* Áp dụng công nghệ mới: Sử dụng Flutter cho phát triển ứng dụng di động sẽ giúp tối ưu hóa thời gian và chi phí, đồng thời mang lại trải nghiệm người dùng tốt hơn nhờ tính đa nền tảng của nó.
* Sự kết hợp giữa front-end và back-end: Kết hợp Flutter và Laravel cho phần front-end và back-end sẽ giúp đảm bảo tính linh hoạt, đáp ứng hiệu quả các yêu cầu phức tạp từ phía người dùng và quản trị hệ thống.
* Mục tiêu nghiên cứu và học tập: Đề tài cung cấp cơ hội để áp dụng những kiến thức học được trong việc thiết kế và phát triển phần mềm thực tế, từ quy trình khảo sát và phân tích yêu cầu đến triển khai và kiểm thử.
* Đóng góp cá nhân và xã hội: Xây dựng một ứng dụng hữu ích sẽ không chỉ là thành tựu cá nhân mà còn góp phần cải thiện cuộc sống và làm việc của nhiều người, từ việc giảm thiểu quên sót công việc đến tăng cường sự tổ chức và hiệu quả làm việc.

Việc lựa chọn đề tài "Xây dựng ứng dụng ghi lịch nhắc online trên thiết bị di động, sử dụng Flutter và Laravel backend" là một bước đi hợp lý và mang tính thực tiễn cao, đáp ứng nhu cầu của thị trường và học hỏi từ công nghệ mới.

# **1.2. Mục tiêu của đề tài**

Mục tiêu của đề tài này là:

* Nâng cao khả năng lập trình: Phát triển kỹ năng lập trình thông qua việc xây dựng một ứng dụng ghi chú trực tuyến trên thiết bị di động.
* Tìm hiểu và áp dụng công nghệ: Tìm hiểu và áp dụng các công nghệ phát triển ứng dụng di động như Flutter (cho phần front-end) và Laravel (cho phần back-end), nhằm tối ưu hóa quá trình phát triển và trải nghiệm người dùng.
* Tạo ra ứng dụng ghi chú đơn giản và hiệu quả: Xây dựng một ứng dụng ghi chú đơn giản nhưng hiệu quả, cho phép người dùng tạo, sửa và xóa các ghi chú trực tuyến.

# **1.3. Cấu trúc của báo cáo**

Báo cáo này được chia thành các chương sau:

* **Chương 1: Giới thiệu** - Trình bày tổng quan về đề tài, mục tiêu và lý do chọn đề tài.
* **Chương 2: Cơ sở lý thuyết** - Trình bày các kiến thức cơ bản liên quan đến phát triển ứng dụng di động và các công nghệ sử dụng.
* **Chương 3: Phân tích và Thiết kế** - Trình bày quá trình phân tích yêu cầu và thiết kế ứng dụng.
* **Chương 4: Triển khai và Kiểm thử** - Mô tả quá trình triển khai và kiểm thử ứng dụng.
* **Kết luận** - Đánh giá kết quả thực hiện đề tài và những bài học kinh nghiệm.

**Kết luận**

Chương 1 của báo cáo đã trình bày tổng quan về đề tài, bao gồm bối cảnh, mục tiêu và cấu trúc của báo cáo. Các chương tiếp theo sẽ đi vào chi tiết từng phần của quá trình phát triển ứng dụng ghi nhắc trên thiết bị di động.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

# **2.1. Kiến thức cơ bản về phát triển ứng dụng di động**

Phát triển ứng dụng di động là một lĩnh vực thú vị và đầy tiềm năng, cho phép tạo ra các công cụ và dịch vụ hữu ích cho người dùng trên điện thoại thông minh và máy tính bảng. Một số kiến thức cơ bản khi nhắc đến mảng ứng dụng này như sau:

**Nền tảng phát triển:**

Hiện nay, có hai nền tảng di động chính là Android và iOS, mỗi nền tảng có hệ điều hành, ngôn ngữ lập trình và kho ứng dụng riêng:

* Android: Phát triển bằng Java hoặc Kotlin, phổ biến hơn, mã nguồn mở, nhiều tùy chỉnh.
* iOS: Phát triển bằng Swift hoặc Objective-C, giao diện người dùng đẹp mắt, trải nghiệm mượt mà.
* Flutter: Sử dụng ngôn ngữ lập trình Dart, Flutter là một framework đa nền tảng được phát triển bởi Google. Nó cho phép phát triển ứng dụng chất lượng cao trên cả Android và iOS từ một mã nguồn duy nhất. Flutter có những ưu điểm như khả năng hot reload nhanh chóng, giao diện người dùng đẹp và hiệu suất cao, giúp đảm bảo sự đồng nhất và tiết kiệm thời gian phát triển.
* React-Native: Phát triển bằng Javascript hoặc Typescript, dễ dàng nhanh chóng tạo nên các ứng dụng, và có thể chạy trên các nền tảng khác nhau

**Kỹ năng lập trình:**

Tùy thuộc vào nền tảng được chọn, thì chúng ta sẽ có những lựa chọn cho từng dự án khác nhau. Và mỗi sự lựa chọn sẽ có những phụ thuộc đi kèm.

Ngôn ngữ lập trình:

* Android: Java hoặc Kotlin
* iOS: Swift hoặc Objective-C
* React-Native: Javascript hoặc Typescript

Ngoài ra, Những kiến thức nền tảng cần thiết bao gồm:

* Cấu trúc dữ liệu và thuật toán: Nền tảng cơ bản để giải quyết vấn đề lập trình.
* Lập trình hướng đối tượng (OOP): Thiết kế và xây dựng ứng dụng hiệu quả.
* Thiết kế giao diện người dùng (UI): Tạo giao diện đẹp mắt, dễ sử dụng.
* Trải nghiệm người dùng (UX): Mang đến trải nghiệm tốt nhất cho người dùng.
* Kiến thức cơ bản về hệ điều hành di động: Hiểu cách thức hoạt động của hệ điều hành để tối ưu hóa ứng dụng.

**Công cụ phát triển:**

Mỗi nền tảng đều có bộ công cụ phát triển (IDE) riêng:

* Android: Android Studio
* iOS: Xcode
* React-Native: Visual Studio code
* Flutter : Visual Studio code Hoặc Android Studio ...,

Ngoài ra, còn có nhiều công cụ hỗ trợ khác như:

* Kiểm soát phiên bản: Git, SVN
* Công cụ xây dựng: Gradle, Maven
* Công cụ kiểm thử: JUnit, XCTest

**Quy trình phát triển:**

Quy trình phát triển ứng dụng di động thường bao gồm các bước sau:

* Xác định ý tưởng và yêu cầu: Xác định mục đích, tính năng và đối tượng mục tiêu của ứng dụng.
* Thiết kế giao diện: Tạo giao diện người dùng đẹp mắt, dễ sử dụng.
* Lập trình: Viết mã để tạo các chức năng của ứng dụng.
* Kiểm thử: Kiểm tra lỗi và đảm bảo ứng dụng hoạt động tốt.
* Triển khai: Phân phối ứng dụng đến người dùng thông qua cửa hàng ứng dụng (Google Play, App Store).

**Kiến trúc của một ứng dụng di động:**

Các thành phần chính của một ứng dụng di động bao gồm:

* Giao diện người dùng (UI): Phần mà người dùng tương tác trực tiếp.
* Logic ứng dụng: Các quy trình xử lý và luồng điều khiển trong ứng dụng.
* Cơ sở dữ liệu: Lưu trữ dữ liệu của ứng dụng, có thể là cục bộ hoặc trên đám mây.

# **2.2. Ứng dụng của lập trình thiết bị di động**

Ứng dụng của lập trình thiết bị di động mang lại nhiều lợi ích và cơ hội trong các lĩnh vực khác nhau, bao gồm:

* Giao tiếp và kết nối xã hội: Các ứng dụng nhắn tin, mạng xã hội giúp kết nối người dùng trên toàn cầu một cách dễ dàng và tức thì. Ví dụ: Facebook Messenger, WhatsApp.
* Giải trí và truyền thông: Ứng dụng nghe nhạc, xem phim, chơi game giúp người dùng giải trí mọi lúc mọi nơi. Ví dụ: Spotify, Netflix, Candy Crush.
* Giáo dục và học tập: Ứng dụng học tập trực tuyến, từ điển, và các công cụ học tập giúp người dùng học tập hiệu quả. Ví dụ: Duolingo, Coursera.
* Y tế và sức khỏe: Ứng dụng theo dõi sức khỏe, lịch trình tập luyện, và tư vấn y tế giúp người dùng quản lý sức khỏe. Ví dụ: MyFitnessPal, HealthTap.
* Thương mại điện tử và Thanh toán: Ứng dụng mua sắm trực tuyến, thanh toán điện tử giúp người dùng giao dịch thuận tiện. Ví dụ: Amazon, PayPal.
* Du lịch và định vị: Ứng dụng bản đồ, đặt vé, khách sạn giúp người dùng lên kế hoạch và thực hiện các chuyến đi. Ví dụ: Google Maps, Booking.com.
* Năng suất và công việc: Ứng dụng quản lý công việc, ghi chú, lịch biểu giúp người dùng tổ chức và tăng hiệu quả làm việc. Ví dụ: Trello, Evernote.

**Kết luận**

Phần 2.1 đã trình bày một cách tổng quan về các kiến thức cơ bản và công cụ cần thiết trong quá trình phát triển ứng dụng di động. Hiểu rõ những kiến thức này giúp nhà phát triển xây dựng các ứng dụng hiệu quả và tối ưu hóa trải nghiệm người dùng. Việc nắm vững nền tảng phát triển, ngôn ngữ lập trình, công cụ hỗ trợ và quy trình phát triển là chìa khóa để tạo ra các ứng dụng di động thành công và hữu ích. Phần 2.2 đã nêu bật các ứng dụng của lập trình thiết bị di động trong nhiều lĩnh vực khác nhau, từ giao tiếp xã hội đến y tế và sức khỏe, cho thấy tầm quan trọng và tiềm năng to lớn của lĩnh vực này trong việc cải thiện chất lượng cuộc sống và công việc của người dùng.

# **CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ**

# **3.1 Phân tích yêu cầu**

## **3.1.1 Yêu cầu chức năng**

Yêu cầu chức năng là một phần quan trọng trong quá trình phát triển phần mềm, nó mô tả các hành động và tính năng mà hệ thống phải cung cấp để đáp ứng các yêu cầu và mong đợi của người dùng và các bên liên quan. Các yêu cầu chức năng thường tập trung vào những gì hệ thống phải làm và cung cấp cho người dùng.

Các đặc điểm chính của yêu cầu chức năng bao gồm:

* **Cụ thể và rõ ràng**: Yêu cầu phải được viết một cách chi tiết và dễ hiểu, tránh sự mơ hồ.
* **Có thể kiểm tra được**: Mỗi yêu cầu cần có khả năng kiểm tra để xác nhận rằng nó đã được thực hiện đúng cách.
* **Phù hợp với mục tiêu của người dùng**: Yêu cầu phải đáp ứng được các nhu cầu và mong muốn của người dùng cuối hoặc các bên liên quan.

Với đề tài đưa ra, để xây dựng một ứng dụng ghi lịch nhắc online giống như Google Calendar, các yêu cầu chức năng chính có thể bao gồm:

* **Đăng ký và đăng nhập người dùng: Người dùng có khả năng đăng ký tài khoản mới hoặc đăng nhập bằng Google Authentication để sử dụng ứng dụng.**
* **Tạo, chỉnh sửa và xóa sự kiện: Người dùng có thể thêm, sửa đổi và xóa các sự kiện trên lịch của họ.**
* **Hiển thị và quản lý lịch cá nhân: Các sự kiện được hiển thị rõ ràng trên lịch theo từng ngày, tuần, tháng và năm. Người dùng có thể quản lý lịch của mình dễ dàng.**
* **Nhắc nhở và thông báo: Ứng dụng cung cấp nhắc nhở và thông báo cho người dùng về các sự kiện sắp diễn ra.**
* Đồng bộ hóa và sao lưu dữ liệu: Dữ liệu của người dùng được đồng bộ hóa và sao lưu để đảm bảo an toàn và sẵn sàng sử dụng.

## **3.1.2 Yêu cầu phi chức năng**

Yêu cầu phi chức năng là những yêu cầu không liên quan trực tiếp đến các chức năng cụ thể mà hệ thống phải thực hiện, mà tập trung vào các tiêu chí hoạt động, hiệu suất và các đặc điểm khác của hệ thống. Những yêu cầu này thường đề cập đến chất lượng và cách thức hệ thống hoạt động hơn là những gì hệ thống làm.

Các đặc điểm chính của yêu cầu phi chức năng:

* **Hiệu suất (Performance)**: Tốc độ xử lý, thời gian phản hồi, và thông lượng của hệ thống.
* **Bảo mật (Security)**: Các biện pháp bảo mật, quyền truy cập, và bảo vệ dữ liệu.
* **Khả dụng (Availability)**: Thời gian hệ thống sẵn sàng hoạt động, tỷ lệ uptime.
* **Khả năng mở rộng (Scalability)**: Khả năng của hệ thống để mở rộng và xử lý tải tăng lên.
* **Độ tin cậy (Reliability)**: Khả năng hoạt động liên tục mà không gặp lỗi.
* **Khả năng bảo trì (Maintainability)**: Dễ dàng sửa lỗi, cập nhật và bảo trì hệ thống.

Ngoài các yêu cầu chức năng, ứng dụng cũng cần đáp ứng các yêu cầu phi chức năng sau:

* **Bảo mật thông tin người dùng và tin nhắn:** Đảm bảo rằng thông tin cá nhân và nội dung tin nhắn của người dùng được bảo mật và không bị truy cập trái phép.
* **Hiệu suất cao, phản hồi nhanh chóng:** Ứng dụng cần hoạt động mượt mà, tin nhắn được gửi và nhận gần như ngay lập tức để đảm bảo trải nghiệm người dùng tốt nhất.
* **Khả năng mở rộng dễ dàng:** Ứng dụng cần được thiết kế để có thể mở rộng và thêm các chức năng mới trong tương lai mà không ảnh hưởng đến hệ thống hiện tại.
* **Giao diện thân thiện và dễ sử dụng:** Giao diện người dùng cần được thiết kế đơn giản, trực quan, dễ dàng sử dụng ngay cả đối với những người không rành công nghệ.

# **3.2 Lựa chọn công cụ**

Để xây dựng ứng dụng ghi lịch nhắc online (giống Google Calendar) trên thiết bị di động, việc lựa chọn công nghệ phù hợp là một bước quan trọng. Dưới đây là lý do cho sự lựa chọn Flutter và Laravel:

**Flutter:**

* **Tính di động và đa nền tảng: Flutter là một framework phát triển ứng dụng di động do Google phát triển, cho phép phát triển ứng dụng đồng thời trên cả Android và iOS từ một mã nguồn duy nhất.**
* **Hiệu suất cao: Flutter sử dụng lõi đồ họa riêng (Flutter Engine) để vẽ giao diện người dùng một cách nhanh chóng và mượt mà, mang lại trải nghiệm gần như native.**
* **Phát triển nhanh: Với khả năng hot reload, Flutter cho phép nhà phát triển thấy kết quả ngay lập tức khi thay đổi mã nguồn, giảm thời gian phát triển và kiểm thử.**
* **Cộng đồng và hỗ trợ: Flutter có một cộng đồng lớn và phát triển nhanh chóng, cung cấp nhiều tài liệu, thư viện và công cụ hỗ trợ.**

**Laravel:**

* **Framework phát triển backend mạnh mẽ: Laravel là một framework phát triển backend web mạnh mẽ, được viết bằng PHP. Nó cung cấp các tính năng mạnh mẽ như xử lý yêu cầu HTTP, quản lý session, xác thực người dùng và tạo API.**
* **Đơn giản và dễ sử dụng: Laravel giúp nhà phát triển xây dựng và bảo trì các ứng dụng một cách hiệu quả với cú pháp dễ hiểu và các công cụ mạnh mẽ.**
* **Bảo mật và quản lý dữ liệu: Laravel cung cấp các tính năng bảo mật như xác thực người dùng, middleware để kiểm soát truy cập và ORM (Object Relational Mapping) để quản lý cơ sở dữ liệu.**
* **Cộng đồng và hỗ trợ: Laravel có một cộng đồng lớn và nhiều tài liệu hướng dẫn, giúp nhà phát triển giải quyết các vấn đề phát triển một cách nhanh chóng và hiệu quả.**

## **3.2.1 Flutter**

**Flutter là một framework phát triển ứng dụng di động mã nguồn mở của Google, sử dụng ngôn ngữ lập trình Dart. Flutter cho phép phát triển ứng dụng đa nền tảng từ một mã nguồn duy nhất, bao gồm cả iOS và Android. Dưới đây là các lý do và quy trình phát triển ứng dụng trên Flutter:**

**Lý do chọn Flutter:**

* **Phát triển đa nền tảng: Flutter cho phép viết mã một lần và chạy trên cả iOS và Android, giúp tiết kiệm thời gian và nguồn lực phát triển.**
* **Hiệu suất cao: Flutter sử dụng Flutter Engine để vẽ giao diện người dùng một cách nhanh chóng và mượt mà, mang lại trải nghiệm gần như native.**
* **Hot Reload: Chức năng hot reload của Flutter cho phép nhà phát triển thấy ngay các thay đổi trong ứng dụng mà không cần phải build lại toàn bộ, giúp tăng tốc quá trình phát triển và debug.**
* **Giao diện người dùng tuyệt vời: Flutter cung cấp các widget tùy chỉnh và sẵn có, giúp xây dựng giao diện đẹp và tương tác tốt với người dùng.**
* **Cộng đồng phát triển mạnh mẽ: Flutter có một cộng đồng lớn và hỗ trợ nhiều thư viện, package giúp việc phát triển ứng dụng trở nên dễ dàng và nhanh chóng.**

**Quy trình phát triển ứng dụng trên Flutter:**

**1. Cài đặt môi trường phát triển:**

* **Cài đặt Flutter SDK và cấu hình biến môi trường.**
* **Cài đặt Android Studio hoặc IntelliJ IDEA với plugin Flutter để phát triển ứng dụng Android.**
* **Cài đặt Xcode để phát triển ứng dụng iOS trên macOS.**

**2. Khởi tạo dự án:**

* **Sử dụng lệnh ‘flutter create’ để tạo một dự án mới.**
* **Cấu trúc dự án Flutter bao gồm các thư mục và tệp quan trọng như lib/ (chứa mã nguồn), pubspec.yaml (quản lý dependencies), và thư mục android/ hoặc ios/ (chứa các cấu hình và mã native cho từng nền tảng).**

**3. Phát triển giao diện người dùng:**

* **Sử dụng Flutter Widget để xây dựng giao diện người dùng.**
* **Flutter cung cấp nhiều loại Widget như Text, Image, ListView, và các Widget layout như Row, Column, để xây dựng giao diện phong phú và đa dạng.**

**4. Lập trình chức năng:**

* **Viết mã nguồn bằng Dart để cài đặt các chức năng của ứng dụng như đăng nhập, đăng ký, tạo và quản lý sự kiện trong lịch.**

**5. Kiểm thử và sửa lỗi:**

**Kiểm tra và debug ứng dụng trên môi trường giả lập hoặc thiết bị thực.**

* **Sử dụng chức năng hot reload để thử nghiệm và điều chỉnh nhanh chóng.**

**6. Triển khai ứng dụng:**

* **Đóng gói ứng dụng và triển khai lên Google Play Store cho Android và App Store cho iOS.**
* **Đảm bảo tuân thủ các quy định và hướng dẫn từ Google và Apple để đưa ứng dụng lên cửa hàng.**

## **3.2.2. MySQL và Laravel**

**1. Laravel**

**Laravel là một framework PHP hiện đại và mạnh mẽ, được thiết kế để giúp các nhà phát triển xây dựng các ứng dụng web và API một cách nhanh chóng và dễ dàng.**

****

* **Routing: Laravel cung cấp một hệ thống routing mạnh mẽ và linh hoạt, cho phép bạn định nghĩa các tuyến đường (routes) của API một cách dễ dàng.**
* **Controllers: Bằng cách sử dụng controllers trong Laravel, bạn có thể xử lý các yêu cầu HTTP từ client và trả về các phản hồi dưới dạng JSON hoặc các định dạng khác.**
* **Eloquent ORM: Eloquent là một ORM mạnh mẽ của Laravel, giúp bạn tương tác với cơ sở dữ liệu MySQL một cách dễ dàng thông qua các model và relationships. Điều này giúp cho việc truy vấn và xử lý dữ liệu trở nên đơn giản và hiệu quả.**
* **Migrations: Laravel cung cấp migrations để quản lý và duy trì cấu trúc của cơ sở dữ liệu. Điều này giúp việc phát triển và triển khai cơ sở dữ liệu trở nên dễ dàng và bảo đảm tính nhất quán của dữ liệu.**
* **Middleware: Laravel sử dụng middleware để xử lý các yêu cầu HTTP trước khi chúng được chuyển đến các routes hoặc controllers. Điều này giúp bạn thực hiện các công việc như xác thực người dùng, kiểm tra quyền truy cập, và xử lý ngoại lệ.**
* **Authentication: Laravel cung cấp hệ thống xác thực mạnh mẽ, bao gồm cả xác thực người dùng thông qua API tokens, cookies, session, hoặc các phương thức khác.**

**2. MySQL**

**MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) phổ biến và mạnh mẽ, thích hợp cho việc lưu trữ dữ liệu cho các ứng dụng web và API.**

* **Transaction ACID: MySQL hỗ trợ các giao dịch đảm bảo tính toàn vẹn của dữ liệu, quan trọng trong việc bảo vệ tính nhất quán của dữ liệu trong các ứng dụng có tính chất giao dịch.**
* **Quản lý quyền truy cập: MySQL cung cấp các cơ chế để quản lý quyền truy cập vào dữ liệu, bảo vệ dữ liệu của bạn khỏi các mối đe dọa bảo mật.**
* **Khả năng mở rộng: MySQL có thể mở rộng từ các ứng dụng nhỏ đến các hệ thống lớn với hàng triệu người dùng một cách dễ dàng và hiệu quả.**

**3. Phát triển API**

**Để phát triển API sử dụng Laravel và MySQL, bạn có thể thực hiện các bước sau:**

* **Cài đặt Laravel: Sử dụng Composer để cài đặt Laravel và thiết lập môi trường phát triển.**
* **Thiết kế cơ sở dữ liệu: Sử dụng migrations của Laravel để thiết kế và tạo bảng cho các đối tượng như người dùng, sản phẩm, đơn hàng và các quan hệ liên quan.**
* **Xây dựng API: Sử dụng Laravel để xây dựng các API cho việc xem danh sách, tạo mới, cập nhật và xóa các đối tượng trong cơ sở dữ liệu. Đảm bảo API trả về dữ liệu dưới dạng JSON hoặc các định dạng phổ biến khác.**
* **Bảo mật và quản lý quyền truy cập: Sử dụng Laravel Passport hoặc JWT để xác thực và bảo vệ các API. Quản lý quyền truy cập bằng middleware và các quy tắc xác thực để đảm bảo an toàn cho dữ liệu người dùng.**
* **Kiểm thử và triển khai: Kiểm tra ứng dụng trên các môi trường khác nhau để đảm bảo tính ổn định và bảo mật của ứng dụng. Sau đó, triển khai ứng dụng lên môi trường sản phẩm hoặc một dịch vụ đám mây.**

# **3.3 Thiết kế hệ thống**

## 3.3.1 Kiến trúc ứng dụng

Ứng dụng sẽ được thiết kế theo kiến trúc client-server, trong đó Flutter sẽ đóng vai trò là client (giao diện người dùng) và Laravel API cùng MySQL sẽ là server (quản lý và lưu trữ dữ liệu). Cụ thể, Flutter sẽ xử lý các tương tác người dùng, hiển thị giao diện và gửi/nhận dữ liệu từ Laravel API. Laravel sẽ đảm nhiệm việc quản lý logic xử lý phía server, xác thực người dùng, bảo mật và tương tác với cơ sở dữ liệu MySQL để lưu trữ và quản lý dữ liệu.

## 3.2.2 Các Thành Phần Chính

* **Xác thực người dùng (Authentication)**: Sử dụng Laravel Passport hoặc JWT (JSON Web Tokens) để quản lý đăng ký và đăng nhập người dùng. Laravel hỗ trợ nhiều phương thức xác thực như email/password, Google, Facebook, giúp tăng cường bảo mật và thuận tiện cho người dùng.
* **Hồ sơ người dùng (User Profile)**: Quản lý thông tin cá nhân của người dùng, bao gồm tên, email, avatar, và danh sách liên lạc.
* **Lịch sự kiện (Event Calendar)**: Cho phép người dùng tạo, chỉnh sửa và xóa các sự kiện. Các sự kiện sẽ được lưu trữ trong MySQL và đồng bộ hóa theo thời gian thực thông qua Laravel API.
* **Nhắc nhở (Reminders)**: Cho phép người dùng đặt nhắc nhở cho các sự kiện. Nhắc nhở sẽ được gửi dưới dạng thông báo tới người dùng.
* **Thông báo (Notifications)**: Sử dụng Flutter Awesome Notifications để gửi thông báo cho người dùng khi có sự kiện mới hoặc khi đến thời gian nhắc nhở.

## **3.3.3 Cơ sở dữ liệu**

Cơ sở dữ liệu sẽ bao gồm hai phần chính: lưu trữ thông tin người dùng và lưu trữ dữ liệu sự kiện. Dữ liệu sẽ được lưu trữ dưới dạng các bảng trong MySQL, mỗi bảng chứa các thông tin cần thiết cho từng chức năng của ứng dụng. Việc quản lý cấu trúc cơ sở dữ liệu sẽ được thực hiện thông qua migrations của Laravel.

Migrations trong Laravel

Laravel cung cấp một hệ thống migrations mạnh mẽ để quản lý cấu trúc cơ sở dữ liệu. Migrations cho phép bạn định nghĩa và theo dõi các thay đổi trong cơ sở dữ liệu của mình. Dưới đây là cách sử dụng migrations để tạo các bảng cần thiết cho ứng dụng.

**+ Migration cho bảng users**

|  |
| --- |
| use Illuminate\Database\Migrations\Migration;  use Illuminate\Database\Schema\Blueprint;  use Illuminate\Support\Facades\Schema;  class CreateUsersTable extends Migration  {  public function up()  {  Schema::create('users', function (Blueprint $table) {  $table->increments('user\_id');  $table->string('name');  $table->string('email')->unique();  $table->string('password');  $table->string('profile\_url')->nullable();  $table->rememberToken();  $table->timestamps();  });  }  public function down()  {  Schema::dropIfExists('users');  }  } |

**Giải thích**

***‘****use****’*** *statements*

* use Illuminate\Database\Migrations\Migration: Import lớp Migration để tạo migration.
* use Illuminate\Database\Schema\Blueprint: Import lớp Blueprint để định nghĩa cấu trúc bảng.
* use Illuminate\Support\Facades\Schema: Import lớp Schema để thao tác với cơ sở dữ liệu.

*‘up’ method*

Phương thức up được sử dụng để định nghĩa các thay đổi cần áp dụng khi chạy migration.

* **Schema::create('users', function (Blueprint $table))**: Tạo bảng users với các cột được định nghĩa trong callback function.
* **$table->increments('user\_id')**: Tạo cột user\_id là khóa chính và tự động tăng.
* **$table->string('name')**: Tạo cột name kiểu chuỗi để lưu tên người dùng.
* **$table->string('email')->unique()**: Tạo cột email kiểu chuỗi, phải là duy nhất trong bảng để tránh trùng lặp.
* **$table->string('password')**: Tạo cột password kiểu chuỗi để lưu mật khẩu đã được mã hóa.
* **$table->string('profile\_url')->nullable()**: Tạo cột profile\_url kiểu chuỗi, có thể chứa giá trị null để lưu URL ảnh đại diện.
* **$table->rememberToken()**: Tạo cột remember\_token để lưu token cho tính năng "ghi nhớ đăng nhập".
* **$table->timestamps()**: Tạo hai cột created\_at và updated\_at để lưu thời gian tạo và cập nhật bản ghi.

*‘down’ method*

Phương thức down được sử dụng để đảo ngược các thay đổi khi rollback migration.

* **Schema::dropIfExists('users')**: Xóa bảng users nếu nó tồn tại.

**+ Migration: CreateEventsTable**

Dưới đây là phần mô tả chi tiết về đoạn mã migration sử dụng Laravel để tạo bảng events trong cơ sở dữ liệu MySQL.

|  |
| --- |
| use Illuminate\Database\Migrations\Migration;  use Illuminate\Database\Schema\Blueprint;  use Illuminate\Support\Facades\Schema;  class CreateEventsTable extends Migration  {  public function up()  {  Schema::create('events', function (Blueprint $table) {  $table->increments('event\_id');  $table->unsignedInteger('user\_id');  $table->string('title');  $table->text('note')->nullable();  $table->date('date');  $table->time('start\_time');  $table->time('end\_time')->nullable();  $table->string('remind')->nullable();  $table->string('repeat')->nullable();  $table->string('color')->nullable();  $table->boolean('status')->default(false);  $table->timestamps();  $table->foreign('user\_id')->references('user\_id')->on('users')->onDelete('cascade');  });  }  public function down()  {  Schema::dropIfExists('events');  }  } |

**Giải thích**

*‘up’ method*

Phương thức up được sử dụng để định nghĩa các thay đổi cần áp dụng khi chạy migration.

* **Schema::create('events', function (Blueprint $table))**: Tạo bảng events với các cột được định nghĩa trong callback function.
* **$table->increments('event\_id')**: Tạo cột event\_id là khóa chính và tự động tăng.
* **$table->unsignedInteger('user\_id')**: Tạo cột user\_id để liên kết với bảng users.
* **$table->string('title')**: Tạo cột title kiểu chuỗi để lưu tiêu đề sự kiện.
* **$table->text('note')->nullable()**: Tạo cột note kiểu text để lưu ghi chú sự kiện, có thể chứa giá trị null.
* **$table->date('date')**: Tạo cột date kiểu date để lưu ngày diễn ra sự kiện.
* **$table->time('start\_time')**: Tạo cột start\_time kiểu time để lưu thời gian bắt đầu sự kiện.
* **$table->time('end\_time')->nullable()**: Tạo cột end\_time kiểu time để lưu thời gian kết thúc sự kiện, có thể chứa giá trị null.
* **$table->string('remind')->nullable()**: Tạo cột remind kiểu chuỗi để lưu thời gian nhắc nhở, có thể chứa giá trị null.
* **$table->string('repeat')->nullable()**: Tạo cột repeat kiểu chuỗi để lưu tần suất lặp lại sự kiện, có thể chứa giá trị null.
* **$table->string('color')->nullable()**: Tạo cột color kiểu chuỗi để lưu màu sắc sự kiện, có thể chứa giá trị null.
* **$table->boolean('status')->default(false)**: Tạo cột status kiểu boolean để lưu trạng thái sự kiện, mặc định là false.
* **$table->timestamps()**: Tạo hai cột created\_at và updated\_at để lưu thời gian tạo và cập nhật bản ghi.

*‘down’ method*

Phương thức down được sử dụng để đảo ngược các thay đổi khi rollback migration.

* **Schema::dropIfExists('events')**: Xóa bảng events nếu nó tồn tại.

**+ Models**

*User Model*

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Foundation\Auth\User as Authenticatable;  use Illuminate\Notifications\Notifiable;  use Laravel\Passport\HasApiTokens;  class User extends Authenticatable  {  use HasApiTokens, HasFactory, Notifiable;  protected $primaryKey = 'user\_id'; // Specify the custom primary key  protected $fillable = [  'name', 'email', 'password', 'profile\_url',  ];  protected $hidden = [  'password', 'remember\_token',  ];  protected $casts = [  'email\_verified\_at' => 'datetime',  ];  public function events()  {  return $this->hasMany(Event::class);  }  }  ?> |

**Giải thích**

Thuộc tính và phương thức

* **use Laravel\Passport\HasApiTokens**: Import trait để sử dụng các tính năng API token của Laravel Passport.
* **protected $primaryKey = 'user\_id'**: Đặt khóa chính của bảng là user\_id thay vì mặc định id.
* **protected $fillable**: Xác định các thuộc tính có thể gán giá trị hàng loạt.
* **protected $hidden**: Xác định các thuộc tính sẽ bị ẩn khi model được chuyển đổi thành mảng hoặc JSON.
* **protected $casts**: Xác định các thuộc tính sẽ được ép kiểu khi truy vấn model.

Phương thức events

* **$this->hasMany(Event::class)**: Xác định mối quan hệ một-nhiều giữa User và Event. Một người dùng có thể có nhiều sự kiện.

*Event Model*

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class Event extends Model  {  use HasFactory;  protected $primaryKey = 'event\_id'; // Đặt khóa chính tùy chỉnh  protected $fillable = [  'user\_id', 'title', 'note', 'date', 'start\_time', 'end\_time', 'remind', 'repeat', 'color', 'status',  ];  public function user()  {  return $this->belongsTo(User::class);  }  }  ?> |

**Giải thích**

Thuộc tính và phương thức

* **use HasFactory**: Import trait để sử dụng factory cho việc tạo dữ liệu giả trong quá trình phát triển và kiểm thử.
* **protected $primaryKey = 'event\_id'**: Đặt khóa chính của bảng là event\_id thay vì mặc định id.
* **protected $fillable**: Xác định các thuộc tính có thể gán giá trị hàng loạt.

Phương thức user

* **$this->belongsTo(User::class)**: Xác định mối quan hệ ngược lại giữa Event và User. Một sự kiện thuộc về một người dùng.

## **3.3.4 Giao diện người dùng**

Thiết kế giao diện người dùng (UI) cho ứng dụng ghi nhớ trực tuyến cần đảm bảo tính đơn giản, trực quan và thân thiện. Giao diện nên tối giản và dễ hiểu, giúp người dùng dễ dàng điều hướng và thực hiện các thao tác chính như tạo, quản lý và nhận thông báo về các sự kiện và ghi chú. Đảm bảo tính nhất quán trong phong cách thiết kế, màu sắc và biểu tượng giúp tạo cảm giác chuyên nghiệp và quen thuộc cho người dùng. Thiết kế cần đẹp mắt và hấp dẫn, mang lại trải nghiệm thú vị với các hiệu ứng mượt mà. Cuối cùng, giao diện cần được tối ưu hóa để tải nhanh, tiết kiệm pin và tài nguyên, đảm bảo hiệu suất cao trên các thiết bị di động.

Các màn hình của ứng dụng ghi nhớ bao gồm:

1. Màn hình Đăng nhập/Đăng ký:
   * Giao diện đơn giản và trực quan, yêu cầu người dùng nhập email và mật khẩu để đăng nhập hoặc tạo tài khoản mới.
   * Có thể thêm các lựa chọn đăng nhập qua Google để tăng tính tiện lợi.
2. Màn hình Danh sách Sự kiện/Ghi chú:
   * Hiển thị danh sách các sự kiện và ghi chú của người dùng, cho phép tìm kiếm và thêm sự kiện mới.
   * Mỗi sự kiện/ghi chú hiển thị tiêu đề, ngày, giờ và một phần nội dung ngắn gọn.
3. Màn hình Chi tiết Sự kiện/Ghi chú:
   * Hiển thị chi tiết một sự kiện hoặc ghi chú cụ thể, bao gồm tiêu đề, nội dung, ngày giờ bắt đầu và kết thúc, nhắc nhở và lặp lại.
   * Cho phép chỉnh sửa hoặc xóa sự kiện/ghi chú.
4. Màn hình Thêm Sự kiện/Ghi chú:
   * Giao diện đơn giản cho phép người dùng nhập tiêu đề, nội dung, ngày giờ bắt đầu và kết thúc, nhắc nhở và lặp lại.
5. Màn hình Thông báo:
   * Hiển thị các thông báo về sự kiện hoặc ghi chú mới nhất, giúp người dùng biết khi nào đến thời gian nhắc nhở.
6. Màn hình Cài đặt Người dùng:
   * Cho phép cài đặt ứng dụng, các chức năng tùy chỉnh phía người dùng và thông tin cá nhân.
   * Cho phép đăng xuất tài khoản.

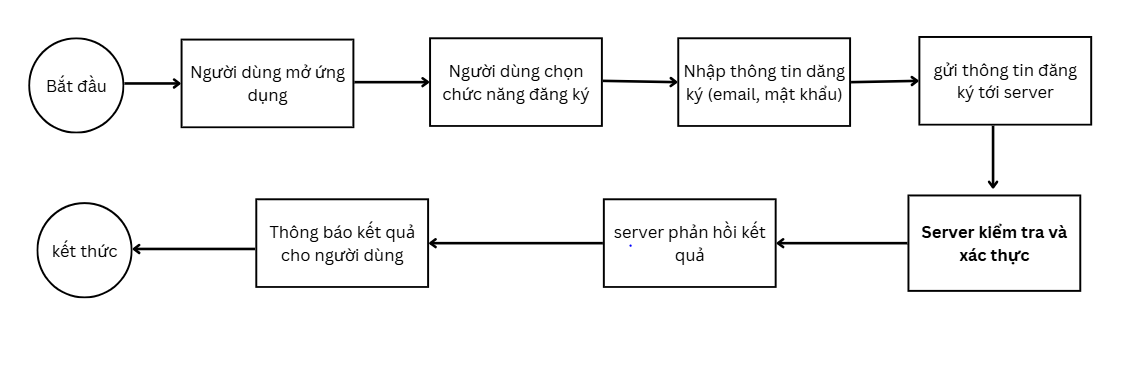
## **3.3.5 Quy trình hoạt động**

Quy trình hoạt động của ứng dụng ghi nhớ trực tuyến sẽ mô tả chi tiết về cách thức hoạt động của các chức năng chính, từ việc đăng ký và đăng nhập, quản lý sự kiện, nhận thông báo, đến việc xem và chỉnh sửa sự kiện. Dưới đây là mô tả chi tiết về quy trình hoạt động của ứng dụng ghi nhớ trực tuyến:

**+ Đăng ký**

Người dùng nhập thông tin và xác thực qua Laravel Passport.

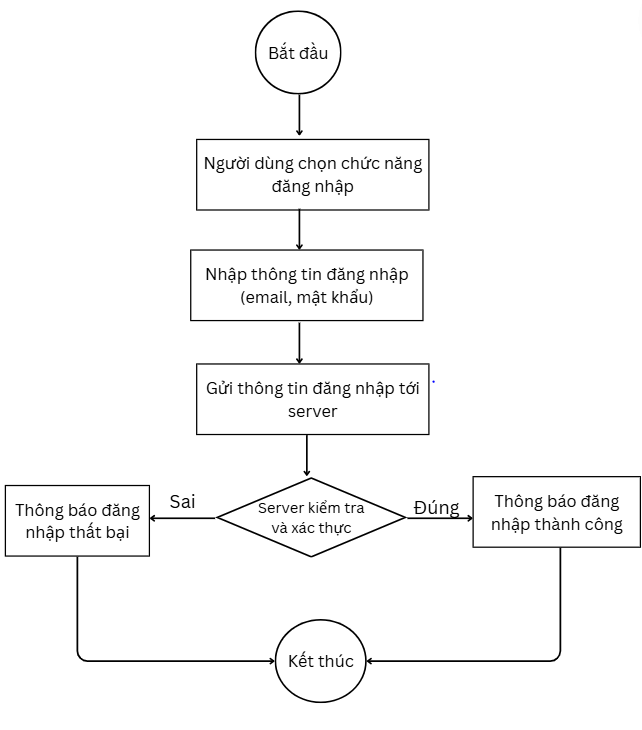
* Người dùng mở ứng dụng ghi nhớ trên thiết bị di động của mình.
* Người dùng chọn tùy chọn đăng ký để tạo tài khoản mới.
* Người dùng nhập thông tin cần thiết như email, mật khẩu, và các thông tin cá nhân khác.
* Thông tin đăng ký được gửi đến server để xử lý qua Laravel Passport.
* Laravel Passport xác thực thông tin đăng ký, kiểm tra email hợp lệ, độ mạnh của mật khẩu và các điều kiện khác.
* Server trả về kết quả xác thực cho ứng dụng.
* Ứng dụng hiển thị thông báo cho người dùng biết đăng ký thành công hay thất bại.



**+ Đăng nhập**

Người dùng đăng nhập bằng thông tin xác thực qua Laravel Passport.

* Người dùng mở ứng dụng và chọn tùy chọn đăng nhập để truy cập vào tài khoản đã tạo.
* Người dùng nhập email và mật khẩu đã đăng ký.
* Thông tin đăng nhập được gửi đến server để xác thực qua Laravel Passport.
* Laravel Passport xác thực thông tin đăng nhập.
* Server trả về kết quả xác thực cho ứng dụng.
* Ứng dụng hiển thị thông báo cho người dùng biết đăng nhập thành công hay thất bại.



**+ Quản lý sự kiện**

Người dùng có thể thêm, xem, chỉnh sửa và xóa các sự kiện.

* Người dùng chọn chức năng thêm sự kiện và nhập các thông tin cần thiết như tiêu đề, ngày, giờ, ghi chú, nhắc nhở và lặp lại.
* Thông tin sự kiện được gửi tới server và lưu trữ trong MySQL database.
* Server lưu trữ thông tin sự kiện và phản hồi kết quả cho ứng dụng.
* Người dùng chọn sự kiện cần chỉnh sửa, thay đổi thông tin và lưu lại.
* Người dùng chọn sự kiện cần xóa và xác nhận.

**+ Nhận thông báo**

Sử dụng Flutter Awesome Notifications để gửi thông báo nhắc nhở.

* Khi tạo hoặc chỉnh sửa sự kiện, ứng dụng thiết lập thời gian nhắc nhở và thông báo.
* Flutter Awesome Notifications quản lý và gửi thông báo đến thiết bị của người dùng theo thời gian đã thiết lập.
* Khi đến thời gian nhắc nhở, thiết bị của người dùng sẽ hiển thị thông báo về sự kiện.

**+ Xem và chỉnh sửa sự kiện**

Người dùng có thể xem và chỉnh sửa chi tiết các sự kiện đã tạo.

* Người dùng chọn sự kiện từ danh sách sự kiện.
* Ứng dụng hiển thị đầy đủ thông tin về sự kiện, bao gồm tiêu đề, nội dung, ngày, giờ, nhắc nhở và lặp lại.
* Người dùng chọn chức năng chỉnh sửa, thay đổi thông tin và lưu lại.

Kết luận

Bằng cách sử dụng Laravel Passport để xác thực và MySQL để lưu trữ dữ liệu, ứng dụng ghi nhớ trực tuyến của bạn sẽ có khả năng quản lý dữ liệu người dùng và sự kiện một cách hiệu quả. Flutter Awesome Notifications sẽ đảm bảo rằng người dùng nhận được thông báo nhắc nhở kịp thời, mang lại trải nghiệm sử dụng tốt nhất.

# **CHƯƠNG 4: TRIỂN KHAI VÀ KIỂM THỬ**

# **4.1 Triển khai**

Triển khai ứng dụng bao gồm việc chuẩn bị môi trường, triển khai mã nguồn và đảm bảo rằng ứng dụng hoạt động ổn định trong môi trường sản xuất.

## 4.1.1 Chuẩn bị môi trường phát triển (Development Environment)

Để bắt đầu phát triển và triển khai ứng dụng, cần phải thiết lập môi trường phát triển cho cả Flutter (frontend) và Laravel (backend).

* **Cài đặt Node.js và npm:**
  + Tải Node.js từ trang web chính thức của Node.js và cài đặt phiên bản LTS mới nhất.
  + Sau khi cài đặt Node.js, npm (Node Package Manager) sẽ được cài đặt cùng.
* **Cài đặt Flutter CLI:**
  + Sử dụng Terminal trên macOS/Linux hoặc Command Prompt trên Windows, cài đặt Flutter CLI bằng lệnh: *‘flutter pub get’*

**Cài đặt Android Studio và thiết lập Android SDK:**

* Tải và cài đặt Android Studio từ trang web chính thức của Android.
* Mở Android Studio và cài đặt các SDK cần thiết thông qua SDK Manager.

**Cài đặt Visual Studio Code (hoặc trình soạn thảo khác):**

* Tải và cài đặt Visual Studio Code để làm môi trường phát triển.
* Cài đặt các extension cần thiết như Flutter và Dart Plugin để hỗ trợ phát triển Flutter.

## 4.1.2 Triển khai mã nguồn

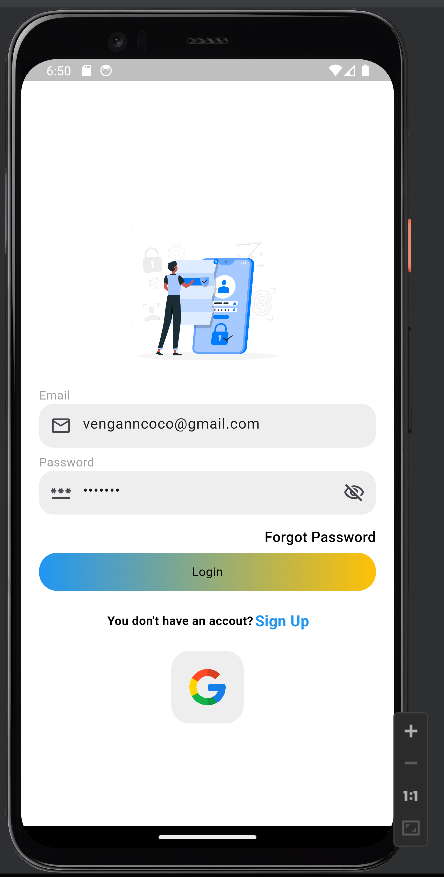
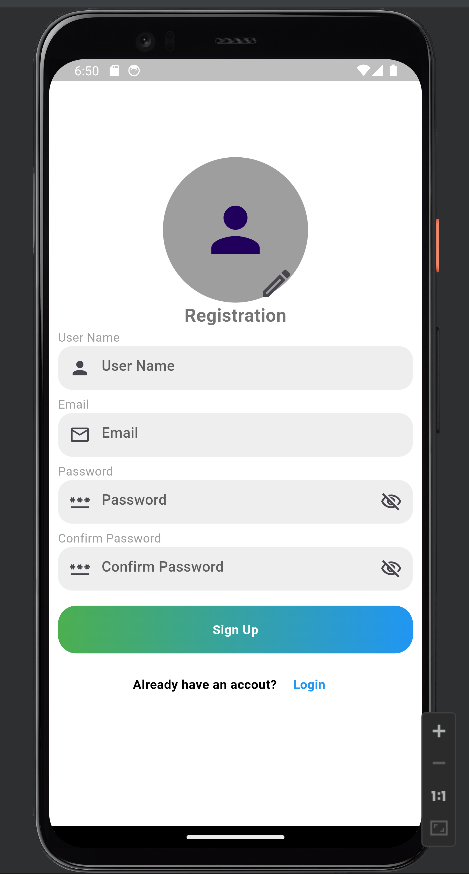
Sau khi môi trường đã được chuẩn bị, tiếp theo là triển khai mã nguồn của ứng dụng.

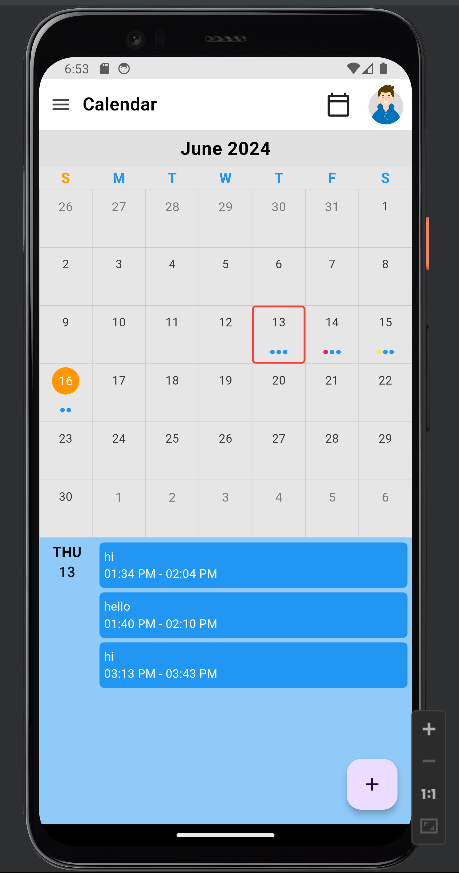
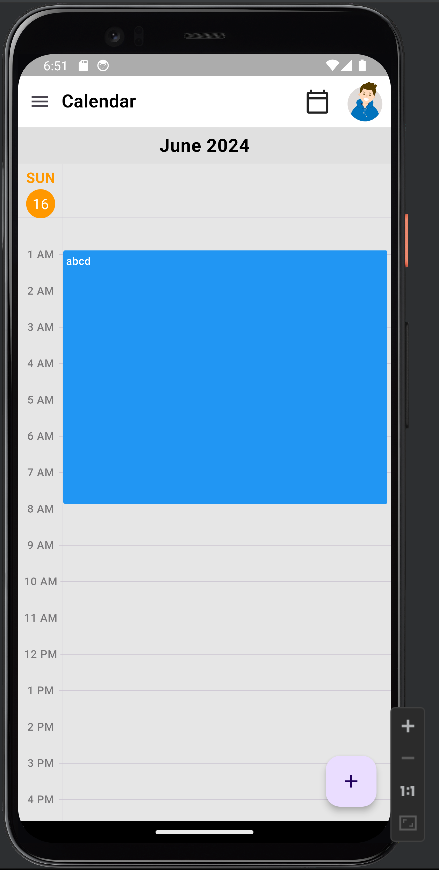
* **Triển khai phần frontend (Flutter):**
  + Chạy lệnh để cài đặt các dependencies của Flutter*: ‘flutter pub get’*
* Tiến hành build ứng dụng Flutter:
  + *‘flutter build apk’* hoăc trên macos *‘flutter build ios’*

**+ Triển khai phần backend (Laravel):**

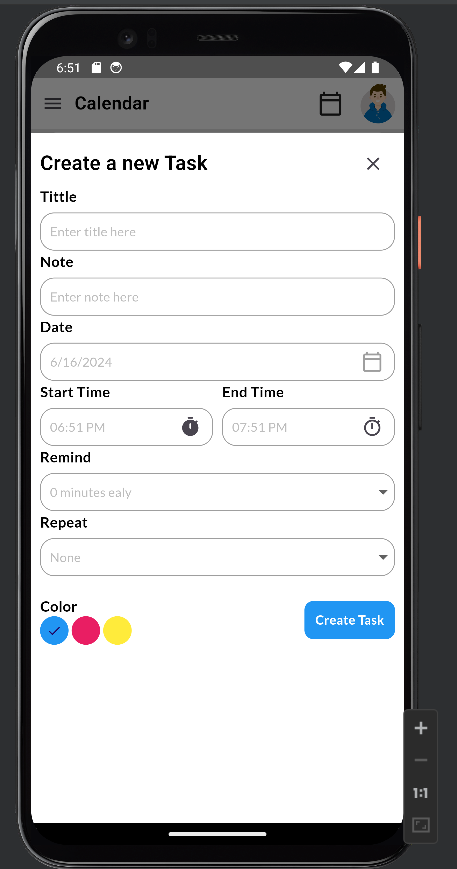
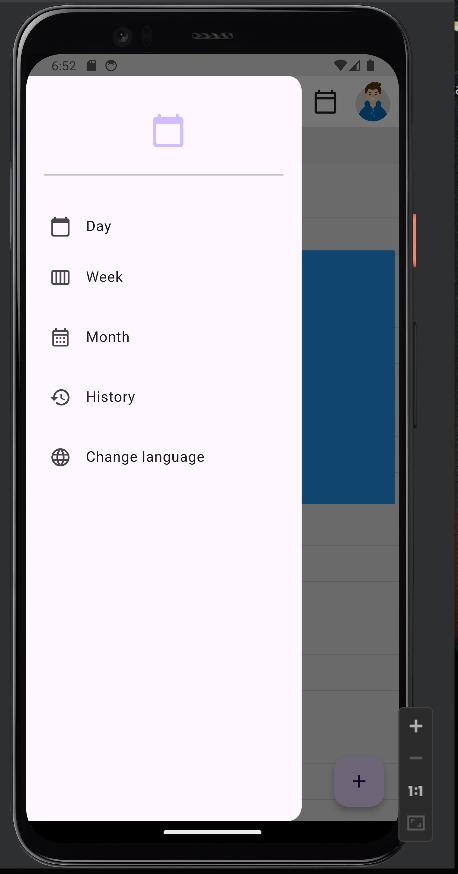
* Đảm bảo mã nguồn Laravel của bạn đã được đẩy lên một repository Git hoặc một nền tảng quản lý mã nguồn.
* Đối với Laravel, bạn cần thiết lập môi trường trên máy chủ sản phẩm, đảm bảo các cấu hình môi trường (như file .env) được cập nhật đúng.

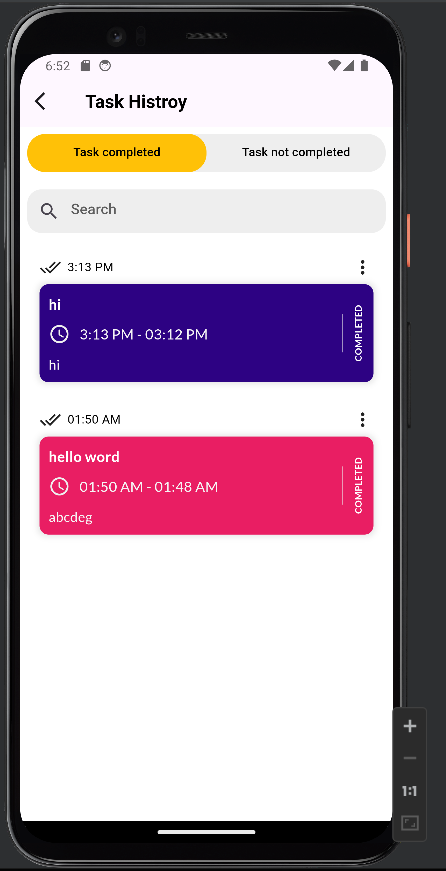
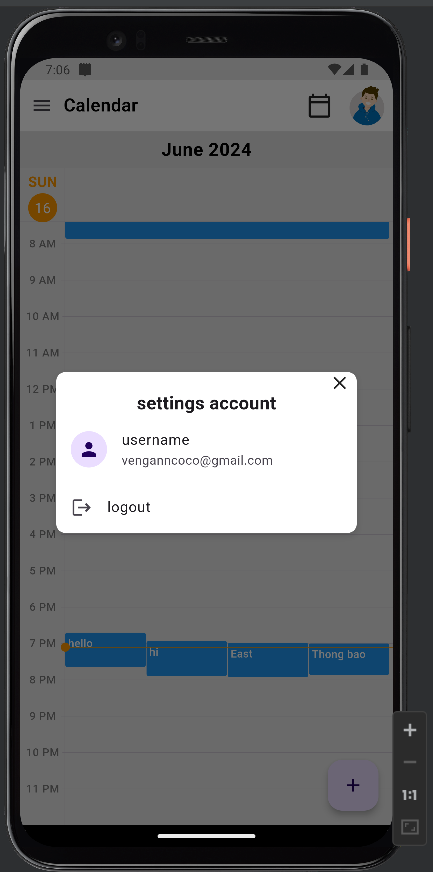
# 4.2 Kiểm thử

Kiểm thử là một phần quan trọng trong quá trình phát triển phần mềm để đảm bảo rằng ứng dụng hoạt động đúng như mong đợi và không có lỗi. Trong phần này, chúng tôi sẽ trình bày chi tiết về quy trình kiểm thử ứng dụng ghi nhắc, bao gồm các phương pháp kiểm thử, môi trường kiểm thử, và các trường hợp kiểm thử cụ thể.

+ **Xem lịch:** Người dùng có thể xem lịch dưới các dạng khác nhau như ngày, tuần, tháng để dễ dàng theo dõi và quản lý các sự kiện.

**+ Tạo sự kiện:** Người dùng có thể tạo sự kiện mới, đặt tên, thời gian bắt đầu và kết thúc cho sự kiện.

****

**+ Xem lịch sử sự kiện:** Cho phép người dùng xem lại các sự kiện đã diễn ra, giúp quản lý và theo dõi công việc hiệu quả hơn.

# **CHƯƠNG 5: KẾT LUẬN VÀ ĐÁNH GIÁ**

Phần này sẽ trình bày các kết luận chính rút ra từ quá trình phát triển và kiểm thử ứng dụng ghi nhớ, cũng như đánh giá về hiệu quả và những điểm mạnh, điểm yếu của ứng dụng.

# **5.1 Kết luận**

## 5.1.1 Tổng quan về dự án

Ứng dụng ghi nhớ được phát triển nhằm cung cấp cho người dùng một công cụ quản lý thời gian và nhắc nhở hiệu quả, tương tự như Google Calendar. Sử dụng các công nghệ hiện đại như Flutter và Laravel, ứng dụng không chỉ đáp ứng được các yêu cầu cơ bản của một ứng dụng ghi nhớ mà còn mang lại trải nghiệm người dùng mượt mà và nhất quán trên cả hai hệ điều hành iOS và Android.

## 5.1.2 Các chức năng chính

* **Tạo sự kiện:** Người dùng có thể tạo sự kiện mới, đặt tên, thời gian bắt đầu và kết thúc cho sự kiện.
* **Nhắc nhở:** Cung cấp tính năng nhắc nhở trước khi sự kiện diễn ra, giúp người dùng không bỏ lỡ các công việc quan trọng.
* **Quản lý sự kiện:** Cho phép người dùng chỉnh sửa, xóa các sự kiện đã lưu và xem lịch sử các sự kiện đã tạo.
* **Giao diện thân thiện:** Thiết kế giao diện người dùng đơn giản, dễ sử dụng trên cả nền tảng iOS và Android.

## 5.1.3 Quá trình phát triển

Quá trình phát triển ứng dụng đã trải qua các giai đoạn chính sau:

* **Thiết kế và Lên kế hoạch:** Xác định yêu cầu và thiết kế kiến trúc hệ thống.
* **Phát triển:** Viết mã và triển khai các chức năng chính.
* **Kiểm thử:** Thực hiện kiểm thử để đảm bảo ứng dụng hoạt động đúng như mong đợi.
* **Triển khai:** Đưa ứng dụng lên môi trường sản xuất.

# **5.2 Đánh giá**

## 5.2.1 Điểm mạnh

* **Khả năng tương thích đa nền tảng:** Sử dụng Flutter giúp ứng dụng hoạt động trên cả iOS và Android với mã nguồn chung.
* **Quản lý thời gian hiệu quả:** Tính năng nhắc nhở và tạo sự kiện giúp người dùng quản lý thời gian và công việc một cách hiệu quả.
* **Bảo mật và ổn định:** Laravel cung cấp các giải pháp bảo mật và quản lý dữ liệu mạnh mẽ, đảm bảo tính ổn định và bảo mật của ứng dụng.

## 5.2.2 Điểm yếu

* **Phụ thuộc vào internet:** Ứng dụng yêu cầu kết nối internet liên tục để hoạt động, có thể không hoạt động tốt trong điều kiện mạng kém.
* **Giới hạn của Laravel và Flutter:** Một số tính năng có thể gặp khó khăn khi mở rộng quy mô hoặc tích hợp các công nghệ mới.

## 5.2.3 Bài học kinh nghiệm

* **Quản lý dự án:** Việc lập kế hoạch và theo dõi tiến độ dự án giúp đảm bảo rằng các nhiệm vụ được hoàn thành đúng hạn.
* **Kiểm thử toàn diện:** Kiểm thử trên nhiều thiết bị và hệ điều hành giúp phát hiện sớm các lỗi và cải thiện chất lượng ứng dụng.
* **Cập nhật và bảo trì:** Cần thiết lập quy trình cập nhật và bảo trì ứng dụng để đảm bảo rằng nó luôn hoạt động ổn định và an toàn.

# **5.3 Hướng phát triển tương lai**

* **Tính năng mở rộng:** Thêm các tính năng mới như chia sẻ sự kiện, tích hợp định vị, và gửi thông báo qua nhiều kênh khác nhau để nâng cao trải nghiệm người dùng.
* **Tối ưu hóa hiệu suất:** Cải thiện hiệu suất ứng dụng để hoạt động mượt mà hơn, đặc biệt trên các thiết bị cũ.
* **Hỗ trợ offline:** Phát triển tính năng lưu trữ và đồng bộ dữ liệu offline để người dùng có thể sử dụng ứng dụng ngay cả khi không có kết nối internet.

# **5.4 Kết luận**

Dự án phát triển ứng dụng ghi nhớ đã đạt được các mục tiêu đề ra ban đầu và cung cấp một công cụ quản lý thời gian hiệu quả cho người dùng. Qua quá trình phát triển và kiểm thử, chúng tôi đã học được nhiều bài học quý giá và xác định được các hướng phát triển tiếp theo để cải thiện và mở rộng ứng dụng. Chúng tôi tin rằng với những cải tiến liên tục, ứng dụng sẽ ngày càng hoàn thiện và đáp ứng tốt hơn nhu cầu của người dùng.

# **TÀI LIỆU THAM KHẢO**

1. **Chat** **openAI:** hỗ trợ tạo gaio diện sửa lỗi
2. **Wikepedia**: các khái niệm về học máy học sâu ....