BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI**

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN ĐỒ ÁN PHÁT TRIỂN ỨNNG DỤNG ĐA NỀN TẢNG NÂNG CAO

------🙢⯎🙠*------*



**BÁO CÁO ĐỒ ÁN**

**ỨNG DỤNG QUẢN LÝ CÔNG VIỆC**

**GIẢNG VIÊN HƯỚNG DẪN:**

**GV. Nguyễn Đình Quý**

**Vũ Hoàng Tuấn Kiệt: 0216966**

Ngành: Khoa học máy tính – Khóa: 66

Lời nói đầu

Trong bối cảnh xã hội hiện đại, nhu cầu quản lý công việc một cách khoa học và hiệu quả ngày càng trở nên quan trọng, đặc biệt là đối với các cá nhân, nhóm làm việc hay doanh nghiệp. Việc phân bổ thời gian, theo dõi tiến độ công việc, cũng như quản lý sự phối hợp giữa các thành viên là những yếu tố then chốt quyết định hiệu quả làm việc. Tuy nhiên, trên thực tế, nhiều cá nhân và tổ chức vẫn gặp khó khăn trong việc tổ chức công việc một cách hợp lý, dẫn đến tình trạng trễ tiến độ, công việc bị bỏ sót hoặc phân công không phù hợp.

Sự phát triển của công nghệ thông tin đã mang lại nhiều giải pháp hữu ích cho vấn đề này, đặc biệt là thông qua các ứng dụng quản lý công việc. Những ứng dụng này không chỉ hỗ trợ theo dõi tiến độ, nhắc việc mà còn giúp tối ưu hóa quy trình làm việc, nâng cao tính cộng tác và trách nhiệm của từng thành viên trong nhóm.

Xuất phát từ thực tế đó, em quyết định thực hiện đồ án với đề tài: **“Xây dựng ứng dụng quản lý công việc”**, nhằm thiết kế và phát triển một hệ thống có khả năng hỗ trợ người dùng trong việc lập kế hoạch, quản lý và theo dõi công việc một cách hiệu quả, linh hoạt và thân thiện với người dùng.

Em xin chân thành cảm ơn thầy Nguyễn Đình Quý đã tận tình truyền đạt những kiến thức bổ ích và cần thiết của môn “Đồ án phát triên ứng dụng đa nền tảng nâng cao” thông qua các buổi học trên lớp, các buổi thực hành, cũng như những lời giảng giải tận tình trong những giờ ngoài lớp. Nhờ đó, chúng em đã có cơ hội học tập, rèn luyện và từng bước áp dụng lý thuyết vào thực tế. Để hoàn thành tốt báo cáo này, em đã tiến hành khảo sát thực tế và nỗ lực hết mình để thực hiện đề tài phát triển ứng dụng mạng xã hội. Tuy nhiên, do kiến thức của em về lĩnh vực này còn hạn chế, cùng với việc lần đầu tiếp cận thực tế và xây dựng đồ án trong thời gian có hạn, nên không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự quan tâm, thông cảm và những ý kiến đóng góp quý báu từ các thầy cô để báo cáo này có thể được hoàn thiện hơn.

1. Giới thiệu đề tài

Ứng dụng quản lý công việc là một nền tảng số hiện đại, đóng vai trò quan trọng trong việc hỗ trợ người dùng tổ chức, theo dõi và hoàn thành các nhiệm vụ một cách hiệu quả trong môi trường học tập, làm việc và đời sống cá nhân. Trong thời đại số hóa mạnh mẽ, việc tối ưu hóa quy trình làm việc và nâng cao năng suất đang trở thành yêu cầu cấp thiết đối với cá nhân cũng như tổ chức. Đề tài phát triển ứng dụng quản lý công việc tập trung vào việc thiết kế và triển khai một hệ thống toàn diện, với mục tiêu hỗ trợ người dùng lập kế hoạch, phân bổ thời gian, phối hợp nhóm và kiểm soát tiến độ công việc một cách khoa học và tiện lợi.

Người dùng có thể dễ dàng tạo công việc mới, phân loại chúng theo dự án, mức độ ưu tiên hoặc thời hạn hoàn thành. Ngoài ra, hệ thống còn hỗ trợ tính năng giao việc cho các thành viên khác trong nhóm, theo dõi tiến độ thực hiện, đánh dấu công việc đã hoàn thành, cũng như đưa ra các thông báo và nhắc nhở nhằm đảm bảo không có nhiệm vụ nào bị bỏ sót. Một điểm nổi bật khác là khả năng thống kê hiệu suất làm việc thông qua biểu đồ và báo cáo, giúp người dùng đánh giá quá trình làm việc của bản thân hoặc cả nhóm một cách trực quan và hiệu quả.

Mục tiêu chính của đề tài là phát triển một ứng dụng vừa thân thiện với người dùng, vừa đảm bảo độ tin cậy và tính bảo mật cao, nhằm bảo vệ dữ liệu công việc cũng như thông tin cá nhân của người sử dụng. Hệ thống được xây dựng với kiến trúc tối ưu, đảm bảo hiệu suất hoạt động ổn định ngay cả khi xử lý số lượng lớn công việc và người dùng đồng thời. Ứng dụng không chỉ đáp ứng nhu cầu quản lý công việc đơn thuần mà còn tạo ra một môi trường cộng tác linh hoạt, thúc đẩy sự kết nối giữa các cá nhân trong cùng một tổ chức hay dự án, qua đó nâng cao hiệu quả làm việc nhóm và quản lý dự án.

1. Công nghệ và kiến trúc hệ thống
2. Giới thiệu công nghệ sử dụng

Để xây dựng hệ thống quản lý công việc hiện đại, đáp ứng yêu cầu về hiệu suất, khả năng mở rộng và trải nghiệm người dùng, nhóm thực hiện đề tài lựa chọn các công nghệ tiên tiến và phù hợp với kiến trúc phân lớp, cụ thể như sau:

* 1. Flutter – Giao diện người dùng (Frontend)

Flutter là bộ công cụ UI mã nguồn mở do Google phát triển, cho phép xây dựng ứng dụng đa nền tảng từ một codebase duy nhất. Với Flutter, giao diện được xây dựng bằng ngôn ngữ Dart, hỗ trợ thiết kế linh hoạt, mượt mà và dễ mở rộng. Trong đề tài này, Flutter được sử dụng để phát triển ứng dụng trên nền tảng di động (Android/Web) với môi trường phát triển là **Visual Studio Code**.

* 1. ASP.NET Core – Xử lý nghiệp vụ (Backend)

ASP.NET Core là framework hiện đại, mã nguồn mở của Microsoft dùng để phát triển các ứng dụng web và API. Hệ thống backend được xây dựng bằng ASP.NET Core chạy trên môi trường **Visual Studio**, chịu trách nhiệm xử lý logic nghiệp vụ, xác thực người dùng, lưu trữ và truy xuất dữ liệu, đồng thời cung cấp các API RESTful để frontend có thể tương tác thông qua mô hình client-server.

* 1. SQL Server – Quản lý cơ sở dữ liệu

SQL Server là hệ quản trị cơ sở dữ liệu quan hệ mạnh mẽ do Microsoft phát triển. Hệ thống sử dụng SQL Server để lưu trữ thông tin người dùng, công việc, lịch sử hoạt động và các dữ liệu liên quan đến hệ thống. Việc sử dụng SQL Server giúp đảm bảo tính toàn vẹn dữ liệu, khả năng mở rộng và hiệu suất truy vấn cao.

* 1. Clean Architecture – Nguyên lý thiết kế phần mềm

Cả frontend và backend đều được thiết kế theo nguyên lý **Clean Architecture** – một mô hình thiết kế phần mềm chú trọng đến việc phân tách các lớp thành phần (layer) rõ ràng như: Domain, Application, Infrastructure và Presentation. Việc áp dụng Clean Architecture giúp hệ thống dễ bảo trì, dễ mở rộng và tăng tính tái sử dụng mã nguồn.

1. Kiến trúc hệ thống

2.1) Mô hình kiến trúc tổng thể

Hệ thống được xây dựng theo kiến trúc **Client-Server**, trong đó:

* **Client (Frontend)**: Giao diện người dùng được phát triển bằng Flutter, đóng vai trò tiếp nhận thao tác từ người dùng và hiển thị dữ liệu tương tác.
* **Server (Backend)**: Được xây dựng bằng ASP.NET Core, chịu trách nhiệm xử lý các yêu cầu từ client, truy xuất cơ sở dữ liệu, xử lý nghiệp vụ và phản hồi dữ liệu dưới dạng JSON thông qua API RESTful.
* **Database**: SQL Server lưu trữ toàn bộ dữ liệu hệ thống. Backend là thành phần duy nhất có quyền thao tác trực tiếp với cơ sở dữ liệu.

hh

SQL Query

HTTP Request

ASP.NET Core

(Backend)

SqlServer

(Database)

Flutter

(Client)

Resutle

HTTP Respone

2.2) Kiến trúc thiết kế frontend - Clean Architecture (Flutter)

Frontend được chia thành các lớp rõ ràng:

* **Presentation Layer**: Giao diện người dùng (UI), nhận tương tác từ người dùng.
* **Domain Layer**: Chứa các entities, repositories và usecases.
* **Data Layer**: Giao tiếp với các dịch vụ bên ngoài (API, lưu trữ cục bộ...), repositories implement và models .

A screenshot of a computer

AI-generated content may be incorrect.3

2.3) Kiến trúc thiết kế backend(ASP.NET Core)

Backend cũng áp dụng mô hình Clean Architecture, gồm 4 lớp:

* **Domain Layer**: Chứa các thực thể nghiệp vụ (entities) và logic xử lý thuần.
* **Application Layer**: Chứa các service (dịch vụ nghiệp vụ) phục vụ yêu cầu từ phía controller.
* **Infrastructure Layer**: Thực thi các thao tác cụ thể như truy vấn database, gọi API ngoài...
* **Controller Layer**: Các controller nhận yêu cầu từ client và gọi service tương ứng.

A screenshot of a computer program

AI-generated content may be incorrect.

2.4) Ưu điểm của kiến trúc lựa chọn

* **Tính phân tách cao**: Giúp dễ bảo trì, nâng cấp, kiểm thử từng phần.
* **Khả năng mở rộng**: Dễ dàng thêm tính năng hoặc thay đổi công nghệ từng phần.
* **Hiệu suất và bảo mật**: Backend xử lý logic, kiểm soát truy cập và bảo vệ dữ liệu an toàn.

1. Các kỹ thuật tối ưu hệ thống

3.1) Tối ưu ở phía Frontend (Flutter)

Để đảm bảo ứng dụng có hiệu năng cao và dễ mở rộng, nhóm đã áp dụng các kỹ thuật tối ưu hóa giao diện và luồng xử lý như sau:

**Quản lý trạng thái bằng Provider và BLoC**

* **Provider**: Được sử dụng để quản lý trạng thái một cách nhẹ nhàng, phù hợp với các phần UI đơn giản cần chia sẻ dữ liệu giữa các widget.
* **BLoC (Business Logic Component)**: Áp dụng mô hình BLoC cho các tính năng phức tạp, giúp tách riêng logic nghiệp vụ khỏi UI. Việc này giúp dễ kiểm thử, tái sử dụng và mở rộng ứng dụng.
* *Lợi ích*: Giảm sự phụ thuộc giữa UI và logic xử lý, tăng tính rõ ràng và kiểm soát được vòng đời dữ liệu.

**Sử dụng PageView thay vì Navigation truyền thống**

* Thay vì sử dụng Navigator.push() hoặc Navigator.pop(), hệ thống sử dụng PageView để điều hướng giữa các màn hình chính.
* Điều này giúp giữ được trạng thái của từng màn hình (stateful), tăng hiệu năng khi chuyển đổi nhanh giữa các tab hoặc các phần trong ứng dụng.
* *Lợi ích*: Tăng tốc độ chuyển trang, giữ lại trạng thái các màn hình trước đó, tối ưu UX (trải nghiệm người dùng).

**Sử dụng ValueNotifier thay cho setState**

* Với các cập nhật UI nhỏ, ứng dụng không sử dụng setState() toàn cục mà thay thế bằng ValueNotifier và ValueListenableBuilder.
* Điều này giúp giảm số lượng widget bị rebuild không cần thiết, giữ hiệu năng ổn định.
* *Lợi ích*: Giảm số lần rebuild không cần thiết, tăng hiệu suất, đặc biệt với các animation hoặc cập nhật dữ liệu thời gian thực.

3.2) Tối ưu ở phía Backend (ASP.NET Core)

Để đảm bảo backend dễ quản lý, phát triển và triển khai, nhóm sử dụng một số kỹ thuật tối ưu như sau:

**Sử dụng Entity Framework Core Migration**

* Entity Framework Core là ORM được sử dụng trong dự án để giao tiếp với SQL Server.
* Việc sử dụng **migration** giúp tự động hóa quá trình tạo và cập nhật cơ sở dữ liệu theo mô hình code-first.
* Khi có thay đổi trong mô hình dữ liệu (entity), chỉ cần tạo migration và cập nhật cơ sở dữ liệu bằng lệnh update-database.
* *Lợi ích*:
* Quản lý lịch sử thay đổi cấu trúc cơ sở dữ liệu.
* Tự động đồng bộ database và mô hình dữ liệu, giảm lỗi do thao tác thủ công.
* Hỗ trợ triển khai nhanh chóng và đồng bộ giữa các môi trường phát triển, test, và production.

3.3) Tổng kết lợi ích tối ưu hóa

|  |  |  |
| --- | --- | --- |
| **Thành phần** | **Kỹ thuật tối ưu** | **Lợi ích mang lại** |
| Flutter (UI) | Provider,BLoC, PageView, ValueNotifier | Hiệu năng cao, dễ bảo trì, UI mượt |
| ASP.NET Core | EF Core Migration | Quản lý DB hiệu quả, triển khai nhanh |
| Toàn hệ thống | CleanArchitecture, Client-Server | Tách biệt rõ ràng, dễ mở rộng |

1. Các thực thể và mối quan hệ trong hệ thống

Hệ thống quản lý công việc được xây dựng dựa trên một số thực thể chính phản ánh các nghiệp vụ cốt lõi như dự án, sprint, công việc (issue), sự kiện, và người dùng tham gia dự án. Mỗi thực thể được thiết kế chi tiết với các thuộc tính và mối quan hệ phù hợp, đảm bảo hỗ trợ đầy đủ các chức năng cần thiết của hệ thống.

4.1) Projectpro (Dự án)

* Là thực thể trung tâm đại diện cho một dự án làm việc.
* Bao gồm các thuộc tính: tên dự án, mô tả, thời gian bắt đầu – kết thúc, ngân sách, tiến độ và mã mời tham gia (invite code).
* **Mối quan hệ:**
* Một dự án có **nhiều** người tham gia (UserProjects).
* Một dự án có thể có **nhiều sprint** (Sprints).
* Một dự án có thể có **nhiều công việc** (Issues).
* Một dự án có thể có **nhiều sự kiện** (Eventpro).

A screenshot of a project

AI-generated content may be incorrect.

4.2) AuthUser (Người dùng hệ thống)

* AuthUser là thực thể đại diện cho người dùng đăng nhập vào hệ thống. Mỗi người dùng có thể tham gia nhiều dự án, đảm nhận nhiều công việc và tham gia nhiều sự kiện.
* **Thuộc tính chính:**
  + userId: Khóa chính định danh người dùng.
  + username, password: Thông tin xác thực.
  + email, phone: Thông tin liên lạc.
  + name, age, address, gender: Thông tin cá nhân mở rộng.
* **Mối quan hệ:**
* Một người dùng có thể được phân công vào **nhiều dự án** (thông qua bảng trung gian UserProject).
* Một người dùng có thể được **giao nhiều công việc (Issue)** thông qua trường AssigneeId.
* Một người dùng có thể **tham gia nhiều sự kiện** thông qua bảng trung gian EventUser.
* Một người dùng có thể **tạo sự kiện** thông qua trường CreatedBy trong Eventpro.

A screenshot of a computer

AI-generated content may be incorrect.

4.3) UserProject (Người dùng trong dự án)

* Đại diện cho mối liên kết giữa người dùng và dự án.
* Bao gồm: UserId, ProjectId, và cờ IsManager để xác định quyền quản lý.
* **Mối quan hệ:**
* Một người dùng có thể thuộc **nhiều dự án**.
* Một dự án có thể có **nhiều người dùng**.

A screenshot of a project

AI-generated content may be incorrect.

4.4) Issue (Công việc/Task)

* Là đơn vị công việc cơ bản trong dự án.
* Gồm tiêu đề, mô tả, trạng thái, mức độ ưu tiên, người thực hiện, ngày tạo – hoàn thành.
* **Mối quan hệ:**
* Mỗi issue thuộc về **một dự án**.
* Mỗi issue có thể được gán cho **một sprint** (tùy chọn).
* Một issue có thể gán cho **một người phụ trách** (qua AssigneeId).

A screenshot of a project

AI-generated content may be incorrect.

4.5) Eventpro (Sự kiện)

* Đại diện cho các sự kiện liên quan đến dự án (họp, deadline...).
* Bao gồm tiêu đề, thời gian bắt đầu – kết thúc, vị trí, màu sắc, trạng thái cả ngày.
* **Mối quan hệ:**
* Mỗi event thuộc về **một dự án**.
* Một event có thể có **nhiều người tham gia** (EventUsers).

A screenshot of a project

AI-generated content may be incorrect.

4.6) EventUser (Người tham gia sự kiện)

* Đại diện cho người dùng tham gia vào một sự kiện cụ thể.
* Chứa UserId và EventId.
* **Mối quan hệ:**
* Một người có thể tham gia **nhiều sự kiện**.
* Một sự kiện có thể có **nhiều người tham gia**.

A screenshot of a computer

AI-generated content may be incorrect.

4.7) Sprint (Giai đoạn thực thi)

* Đại diện cho một chu kỳ làm việc ngắn trong dự án (thường dùng trong mô hình Agile).
* Bao gồm tên, mô tả, thời gian tạo – kết thúc, mức độ ưu tiên, trạng thái.
* **Mối quan hệ:**
* Mỗi sprint thuộc về **một dự án duy nhất**.
* Một sprint có thể chứa **nhiều công việc (issue)**.

A screenshot of a computer

AI-generated content may be incorrect.

4.8) Tóm tắt các mối quan hệ chính

|  |  |  |  |
| --- | --- | --- | --- |
| **Thực thể** | **Liên kết với** | **Kiểu quan hệ** | **Ghi chú** |
| Projectpro | Userproject | 1 - M | Một dự án có nhiều người dùng |
| Projectpro | Sprint | 1 - M | Một dự án có nhiều sprint |
| Projectpro | Issue | 1 - M | Một dự án có nhiều issue |
| Projectpro | Eventpro | 1 - M | Một dự án có nhiều Event |
| Sprint | Issue | 1 - M | Một sprint có nhiều issue |
| Issue | Sprint(nullabe) | M - 1 | Issue có thể thuộc sprint hoặc không |
| Eventpro | EventUser | 1 - M | Một sự kiện có nhiều người tham gia |

A screenshot of a computer

AI-generated content may be incorrect.

1. Phân tích chi tiết Hệ thống
2. Đăng ký(Login Page)

Trang LoginPage là giao diện đăng nhập của ứng dụng quản lý công việc, được viết bằng **Flutter** và sử dụng kiến trúc **BLoC** (Business Logic Component) để quản lý trạng thái. Đây là điểm khởi đầu cho người dùng để xác thực danh tính trước khi truy cập vào các tính năng chính của ứng dụng (ví dụ: trang ProjectPage).

1. **Mục đích chính**:

* Cho phép người dùng nhập **tên đăng nhập** (username) và **mật khẩu** (password) để đăng nhập.
* Cung cấp tùy chọn chuyển hướng đến trang đăng ký (RegisterPage) hoặc khôi phục mật khẩu (chưa được triển khai).
* Hiển thị thông báo thành công hoặc lỗi dựa trên kết quả xác thực từ AuthBloc.

1. **Cấu trúc**

Giao diện của LoginPage được xây dựng bằng widget Scaffold với các thành phần chính:

* **SafeArea**: Đảm bảo nội dung không bị che khuất bởi các khu vực hệ thống (notch, thanh trạng thái).
* **SingleChildScrollView**: Cho phép cuộn nội dung khi bàn phím xuất hiện, tránh lỗi tràn giao diện.
* **Column**: Sắp xếp các thành phần theo chiều dọc, bao gồm:
* Tiêu đề "Welcome Back" và dòng chữ phụ "Sign in to continue" với định dạng văn bản hấp dẫn.
* Hai trường nhập liệu (TextField) cho **username** và **password**.
* Nút **Login** kiểu ElevatedButton để gửi yêu cầu đăng nhập.
* Liên kết văn bản (TextButton) để chuyển hướng đến **Forgot Password** (chưa triển khai) và **Register**.

1. **Chức năng**

* **Nhập liệu**: Người dùng nhập **username** và **password** qua hai TextField được điều khiển bởi TextEditingController.
* **Xác thực**:
  + Khi người dùng nhấn nút **Login**, sự kiện LoginUserEvent được gửi đến AuthBloc với thông tin username và password.
  + BlocConsumer lắng nghe trạng thái từ AuthBloc:
  + **AuthLoading**: Hiển thị vòng tròn tải (CircularProgressIndicator) và vô hiệu hóa các trường nhập liệu/nút.
  + **AuthSuccess**: Hiển thị thông báo thành công qua SnackBar và chuyển hướng đến ProjectPage.
  + **AuthError**: Hiển thị thông báo lỗi qua SnackBar.
  + Kiểm tra đầu vào: Nếu username hoặc password rỗng, hiển thị thông báo lỗi yêu cầu nhập đầy đủ.
* **Điều hướng**:
* Chuyển hướng đến ProjectPage khi đăng nhập thành công.
* Chuyển hướng đến RegisterPage khi người dùng nhấn liên kết "Don’t have an account?

A screenshot of a login screen

AI-generated content may be incorrect.

1. Trang dự án (Projects Page)

Trang ProjectPage là giao diện chính để người dùng quản lý các dự án trong ứng dụng quản lý công việc, được viết bằng **Flutter** và sử dụng kiến trúc **BLoC** (Business Logic Component) để quản lý trạng thái. Đây là điểm đến sau khi người dùng đăng nhập thành công, cho phép xem, tạo và tham gia các dự án.

**a) Mục đích chính:**

* Hiển thị danh sách các dự án mà người dùng đang tham gia, với phân trang để quản lý số lượng lớn dự án.
* Cung cấp chức năng tạo dự án mới (Create Project) và tham gia dự án hiện có (Join Project) thông qua mã mời.
* Hiển thị thông báo thành công hoặc lỗi dựa trên kết quả từ các thao tác liên quan đến dự án (ProjectBloc và UserProjectBloc).
* Đảm bảo người dùng phải đăng nhập để truy cập các tính năng, nếu không sẽ chuyển hướng về LoginPage.

**b) Cấu trúc**

Giao diện của ProjectPage được xây dựng bằng widget Scaffold với các thành phần chính:

* **MultiBlocProvider**: Cung cấp hai BLoC (ProjectBloc và UserProjectBloc) để quản lý trạng thái dự án và tương tác người dùng-dự án.
* **Column**: Sắp xếp các thành phần giao diện theo chiều dọc, bao gồm:
  + **Tiêu đề và nút hành động**: Phần đầu chứa tiêu đề "Projects" và hai nút TextButton ("Join Project" và "New Project") để mở các dialog tương ứng.
  + **Danh sách dự án**: Sử dụng ListView để hiển thị các dự án dưới dạng các widget ProjectItem.
  + **Thanh phân trang**: Một Row chứa các nút điều hướng (IconButton) và một TextField để nhập số trang, hỗ trợ chuyển đổi giữa các trang dự án.
* **Dialogs**:
  + **Create Project Dialog**: Hiển thị khi nhấn "New Project", chứa các trường nhập liệu cho tên dự án (Project Name) và mô tả (Description).
  + **Join Project Dialog**: Hiển thị khi nhấn "Join Project", chứa trường nhập liệu cho mã mời (Invite Code).

**c) Chức năng**

* **Hiển thị danh sách dự án**:
  + Khi người dùng đăng nhập (trạng thái AuthSuccess), ProjectBloc được khởi tạo và gửi sự kiện FetchProjectsByUserEvent để lấy danh sách dự án dựa trên userId.
  + Danh sách dự án được phân trang với 7 dự án mỗi trang (itemsPerPage = 7), hỗ trợ điều hướng qua các nút "trái/phải" hoặc nhập số trang trực tiếp.
  + Nếu không có dự án, hiển thị thông báo "No projects available". Nếu chưa đăng nhập, hiển thị thông báo yêu cầu đăng nhập với nút chuyển hướng đến LoginPage.
* **Tạo dự án mới**:
  + Người dùng nhấn nút "New Project" để mở dialog nhập tên và mô tả dự án.
  + Dữ liệu được gửi qua sự kiện CreateProjectEvent tới ProjectBloc. Dự án mới được tạo với các thuộc tính như managerId (lấy từ AuthBloc), startDate (ngày hiện tại), và trạng thái pending.
  + Kiểm tra đầu vào: Tên dự án không được để trống.
* **Tham gia dự án**:
  + Người dùng nhấn nút "Join Project" để mở dialog nhập mã mời.
  + Dữ liệu được gửi qua sự kiện InviteUserToProjectEvent tới UserProjectBloc, kèm theo userId và vai trò (isManager mặc định là false).
  + Kiểm tra đầu vào: Mã mời không được để trống.
* **Xử lý trạng thái**:
  + MultiBlocListener lắng nghe trạng thái từ ProjectBloc và UserProjectBloc:
* **ProjectLoading**: Hiển thị CircularProgressIndicator khi đang tải dữ liệu.
* **ProjectsLoaded**: Hiển thị danh sách dự án với phân trang.
* **ProjectSuccess** / **UserProjectSuccess**: Hiển thị thông báo thành công qua SnackBar và làm mới danh sách dự án.
* **ProjectError** / **UserProjectError**: Hiển thị thông báo lỗi qua SnackBar

.

* **Điều hướng**:
  + Nếu người dùng chưa đăng nhập, chuyển hướng về LoginPage khi cố gắng tạo hoặc tham gia dự án.
  + Sau khi tạo hoặc tham gia dự án thành công, danh sách dự án được làm mới tự động.
  + Nhấn vào dự án đề điều hướng đên trang dự án chi tiết.

A screenshot of a phone

AI-generated content may be incorrect.

1. Chi tiết dự án(Project Detail Page)

Trang ProjectItemDetail là giao diện hiển thị chi tiết thông tin của một dự án cụ thể trong ứng dụng quản lý công việc. Được viết bằng **Flutter**, trang này sử dụng kiến trúc **BLoC** và **Provider** để quản lý trạng thái, cung cấp thông tin chi tiết về dự án và các tính năng liên quan như backlog và sự kiện.

**a) Mục đích chính:**

* Hiển thị thông tin chi tiết của một dự án được chọn, bao gồm các thông tin như tên, mô tả, và các thuộc tính khác của đối tượng Project.
* Cung cấp giao diện để xem và quản lý **backlog** (danh sách công việc) và **sự kiện** (events) liên quan đến dự án.
* Cho phép người dùng tương tác với dự án thông qua một thanh bên (sidebar) có thể bật/tắt.
* Tích hợp với các thành phần như UserProjectBloc và EventBloc để lấy dữ liệu người dùng và sự kiện liên quan đến dự án.

**b) Cấu trúc**

Giao diện của ProjectItemDetail được xây dựng bằng widget Scaffold với các thành phần chính:

* **MultiProvider**: Cung cấp các provider và BLoC để quản lý trạng thái, bao gồm:
  + BacklogProvider (dùng ChangeNotifierProvider): Quản lý dữ liệu backlog của dự án.
  + UserProjectBloc: Lấy thông tin người dùng tham gia dự án.
  + EventBloc: Lấy danh sách sự kiện liên quan đến dự án.
* **Stack**: Sắp xếp các thành phần giao diện chồng lên nhau, bao gồm:
  + **TopBar**: Một widget tùy chỉnh hiển thị thông tin tổng quan của dự án (ví dụ: tên dự án) và các chức năng liên quan.
  + **SidebarLayout**: Một widget tùy chỉnh hiển thị thanh bên, có thể bật/tắt để cung cấp các tùy chọn quản lý dự án.
  + **SidebarToggleButton**: Một nút để bật/tắt thanh bên, hỗ trợ tương tác người dùng.
* **Background**: Màu nền được đặt là trắng (Colors.white) để tạo giao diện sạch sẽ.

**c) Chức năng**

* **Hiển thị chi tiết dự án**:
  + Trang nhận một đối tượng Project qua tham số widget.project và sử dụng projectId để tải dữ liệu liên quan.
  + Widget TopBar hiển thị thông tin tổng quan của dự án.
* **Quản lý backlog**:
  + Trong initState, BacklogProvider được gọi để lấy dữ liệu backlog của dự án thông qua fetchBacklogData(projectId) ngay sau khi khung giao diện được vẽ xong (WidgetsBinding.instance.addPostFrameCallback).
  + BacklogProvider sử dụng cơ chế ChangeNotifier để thông báo cập nhật giao diện khi dữ liệu backlog thay đổi.
* **Quản lý người dùng và sự kiện**:
  + UserProjectBloc được khởi tạo và gửi sự kiện FetchUserProjectByProjectIdEvent để lấy danh sách người dùng liên quan đến dự án.
  + EventBloc được khởi tạo và gửi sự kiện FetchEventsByProjectEvent để lấy danh sách sự kiện liên quan đến dự án.
* **Thanh bên bật/tắt**:
  + Biến isSidebarOpen kiểm soát trạng thái của thanh bên (mở/đóng).
  + Hàm toggleSidebar chuyển đổi trạng thái của thanh bên và cập nhật giao diện thông qua setState.
  + Widget SidebarToggleButton cung cấp nút để người dùng bật/tắt thanh bên, cải thiện trải nghiệm người dùng khi cần truy cập các chức năng bổ sung.
* **Xử lý trạng thái**:
* Dữ liệu được tải bất đồng bộ thông qua BacklogProvider, UserProjectBloc, và EventBloc, đảm bảo giao diện phản hồi nhanh và dữ liệu được cập nhật chính xác.

1. Sidebarlayout(Ở trong Project Detail Page)

SidebarLayout là một widget tùy chỉnh trong trang chi tiết dự án (ProjectItemDetail), chịu trách nhiệm cung cấp thanh bên có thể bật/tắt và các chế độ xem khác nhau (như Summary, Timeline, Backlog, v.v.) liên quan đến một dự án cụ thể trong ứng dụng quản lý công việc. Widget này sử dụng **Flutter** và tích hợp với PageView để chuyển đổi giữa các trang con.

**a) Mục đích chính:**

* Cung cấp một thanh bên điều hướng (sidebar) có thể bật/tắt để người dùng truy cập vào các chế độ xem khác nhau của dự án, như tổng quan (Summary), dòng thời gian (Timeline), danh sách công việc (Backlog), lịch (Calendar), danh sách (List), và lịch trình (Schedule).
* Hiển thị thông tin cơ bản của dự án (như tên dự án) và các tùy chọn điều hướng trong thanh bên.
* Cho phép chuyển đổi mượt mà giữa các chế độ xem thông qua PageView mà không cần cuộn thủ công và Navigation
* Cung cấp nút bật/tắt (SidebarToggleButton) để mở/đóng thanh bên, tối ưu hóa không gian giao diện.

**b) Cấu trúc**

Giao diện của SidebarLayout được xây dựng bằng widget Stack với các thành phần chính:

* **PageView**:
* Hiển thị các trang con liên quan đến dự án, bao gồm:
* SummaryPage: Trang tổng quan dự án.
* TimelinePage: Trang hiển thị dòng thời gian.
* BacklogPage: Trang danh sách công việc.
* CalendarPage: Trang lịch dự án.
* ListPage: Trang danh sách (có thể là danh sách nhiệm vụ hoặc thành viên).
* SchedulePage: Trang lịch trình.
* Sử dụng PageController để điều khiển chuyển đổi giữa các trang, với thuộc tính physics: NeverScrollableScrollPhysics() để vô hiệu hóa cuộn thủ công, đảm bảo chỉ chuyển trang qua điều hướng thanh bên.
* **AnimatedPositioned (Sidebar)**:
  + Một Container hiển thị thanh bên với chiều rộng cố định (210px), có thể trượt vào/ra bằng hiệu ứng hoạt hình (duration: 300ms).
  + Thanh bên chứa:
    - Tên dự án với biểu tượng (dùng widget.project.name hoặc "Unnamed Project" nếu tên rỗng).
    - Nhóm "PLANNING" với các SidebarItem tương ứng với các trang (Summary, Timeline, Backlog, Calendar, List, Schedule, Add Views).
* **SidebarToggleButton**:
  + Một widget riêng biệt hiển thị nút bật/tắt thanh bên, sử dụng AnimatedPositioned để di chuyển cùng thanh bên.

**c) Chức năng**

* **Điều hướng giữa các chế độ xem**:
  + Thanh bên chứa các SidebarItem, mỗi mục đại diện cho một trang trong PageView.
  + Khi người dùng nhấn vào một SidebarItem, hàm \_onItemTapped được gọi, cập nhật \_selectedIndex và chuyển PageView đến trang tương ứng thông qua \_pageController.jumpToPage.
  + PageView tự động cập nhật \_selectedIndex khi trang thay đổi, đảm bảo thanh bên và nội dung luôn đồng bộ.
* **Bật/tắt thanh bên**:
  + Biến isSidebarOpen kiểm soát trạng thái mở/đóng của thanh bên.
  + Hàm onToggle (được truyền từ ProjectItemDetail) được gọi khi người dùng nhấn vào SidebarToggleButton, chuyển đổi trạng thái và kích hoạt hiệu ứng hoạt hình trượt.
* **Hiển thị thông tin dự án**:
  + Thanh bên hiển thị tên dự án (widget.project.name) với biểu tượng Icons.build, cung cấp ngữ cảnh rõ ràng về dự án đang xem.
* **Quản lý trạng thái**:
  + PageController được khởi tạo với trang mặc định là BacklogPage (initialPage: 2) và được giải phóng trong dispose để tránh rò rỉ bộ nhớ.
* **Hiệu ứng giao diện**:
* Thanh bên sử dụng AnimatedPositioned để tạo hiệu ứng trượt mượt mà khi mở/đóng.
* SidebarToggleButton có hiệu ứng bóng (BoxShadow) và mũi tên động để cải thiện trải nghiệm người dùng.

A screenshot of a computer

AI-generated content may be incorrect.

1. Backlog Page

BacklogPage là giao diện hiển thị và quản lý danh sách công việc (backlog) và các sprint của một dự án trong ứng dụng quản lý công việc. Trang này tích hợp với BacklogProvider để quản lý trạng thái và hỗ trợ các tính năng như tìm kiếm, tạo sprint, và quản lý công việc.

**a) Mục đích chính:**

* Hiển thị danh sách các sprint và công việc (issues) trong backlog của dự án, với khả năng tìm kiếm và lọc.
* Cho phép người quản lý (manager) tạo sprint mới, thêm công việc vào backlog hoặc sprint, và chỉnh sửa trạng thái công việc.
* Cung cấp giao diện kéo-thả để di chuyển công việc giữa backlog và sprint.

**b) Cấu trúc**

Giao diện được xây dựng bằng widget Scaffold với các thành phần chính:

* **Stack**:
  + **Thanh tìm kiếm**: Một TextField với biểu tượng tìm kiếm và nút xóa.
  + **Breadcrumb**: Hiển thị đường dẫn "Projects / Tên dự án / Backlog" và nút "Create Sprint" (dành cho manager).
  + **Danh sách sprint**: Một ListView hiển thị các SprintSection chứa danh sách công việc, sử dụng Consumer<BacklogProvider> để cập nhật theo trạng thái.
  + **BacklogSection**: Một DraggableScrollableSheet hiển thị danh sách công việc trong backlog, có thể kéo mở rộng từ dưới lên.
* **Overlay**: Sử dụng OverlayEntry để hiển thị giao diện tạo sprint (createSprintOverlayEntry) khi người quản lý nhấn "Create Sprint".

**c) Chức năng**

* **Tìm kiếm và lọc**:
  + TextEditingController theo dõi đầu vào tìm kiếm và cập nhật BacklogProvider để lọc danh sách sprint và công việc theo searchQuery.
  + Hiển thị thông báo "No issues found" nếu không có kết quả phù hợp.
* **Quản lý sprint và công việc**:
  + Hiển thị các sprint thông qua SprintSection, với khả năng mở rộng/thu gọn để xem danh sách công việc.
  + Người quản lý có thể tạo sprint mới hoặc thêm công việc vào sprint/backlog thông qua các overlay (createSprintOverlayEntry, createIssueOverlayEntry).
  + Hỗ trợ kéo-thả (DragTarget) để di chuyển công việc giữa backlog và sprint.
* **Xử lý trạng thái**:
  + Consumer<BacklogProvider> hiển thị CircularProgressIndicator khi tải dữ liệu, thông báo lỗi nếu có, hoặc danh sách sprint/backlog khi tải thành công.
  + Chỉ người quản lý (isManager) được phép tạo sprint, thêm công việc, hoặc chỉnh sửa.
* **Giao diện động**:
  + DraggableScrollableSheet cho phép điều chỉnh kích thước vùng backlog (từ 10% đến 40% chiều cao màn hình).
  + Các nút hành động (tạo sprint, thêm công việc) chỉ hiển thị với người quản lý.

A screenshot of a chat

AI-generated content may be incorrect.

Kết luận

Trong bối cảnh công nghệ thông tin ngày càng phát triển mạnh mẽ, việc ứng dụng các nền tảng kỹ thuật số vào quản lý công việc đã trở thành một xu hướng tất yếu nhằm nâng cao hiệu quả làm việc, tối ưu hóa nguồn lực và hỗ trợ cộng tác nhóm một cách khoa học. Với mục tiêu đó, đề tài **“Xây dựng hệ thống quản lý công việc”** đã được thực hiện, sử dụng kết hợp giữa các công nghệ hiện đại như **Flutter** cho giao diện người dùng, **ASP.NET Core** cho xử lý backend, cùng với **SQL Server** trong việc quản lý dữ liệu.

Quá trình phát triển hệ thống tuân thủ nguyên lý **Clean Architecture**, giúp tách biệt rõ ràng giữa các tầng ứng dụng, nâng cao khả năng mở rộng và bảo trì về sau. Đồng thời, việc tối ưu hóa hiệu năng thông qua các kỹ thuật như **Provider**, **BLoC**, **ValueNotifier**, và **EF Core Migration** đã góp phần mang lại trải nghiệm mượt mà và ổn định cho người dùng.

Thông qua đồ án này, em không chỉ củng cố kiến thức nền tảng về lập trình giao diện và phát triển hệ thống mà còn rèn luyện kỹ năng làm việc nhóm, giải quyết vấn đề, và tư duy thiết kế phần mềm theo hướng hiện đại. Mặc dù vẫn còn một số hạn chế và tiềm năng mở rộng trong tương lai như tích hợp thông báo đẩy, quản lý thời gian thực, hay áp dụng AI vào phân tích công việc, em tin rằng hệ thống hiện tại đã đáp ứng được các yêu cầu cốt lõi về chức năng và hiệu quả hoạt động.

Em xin chân thành cảm ơn quý thầy cô đã hướng dẫn và hỗ trợ trong suốt quá trình thực hiện đề tài. Đó là động lực để chúng em không ngừng nỗ lực học hỏi và hoàn thiện hơn trong các dự án tương lai.