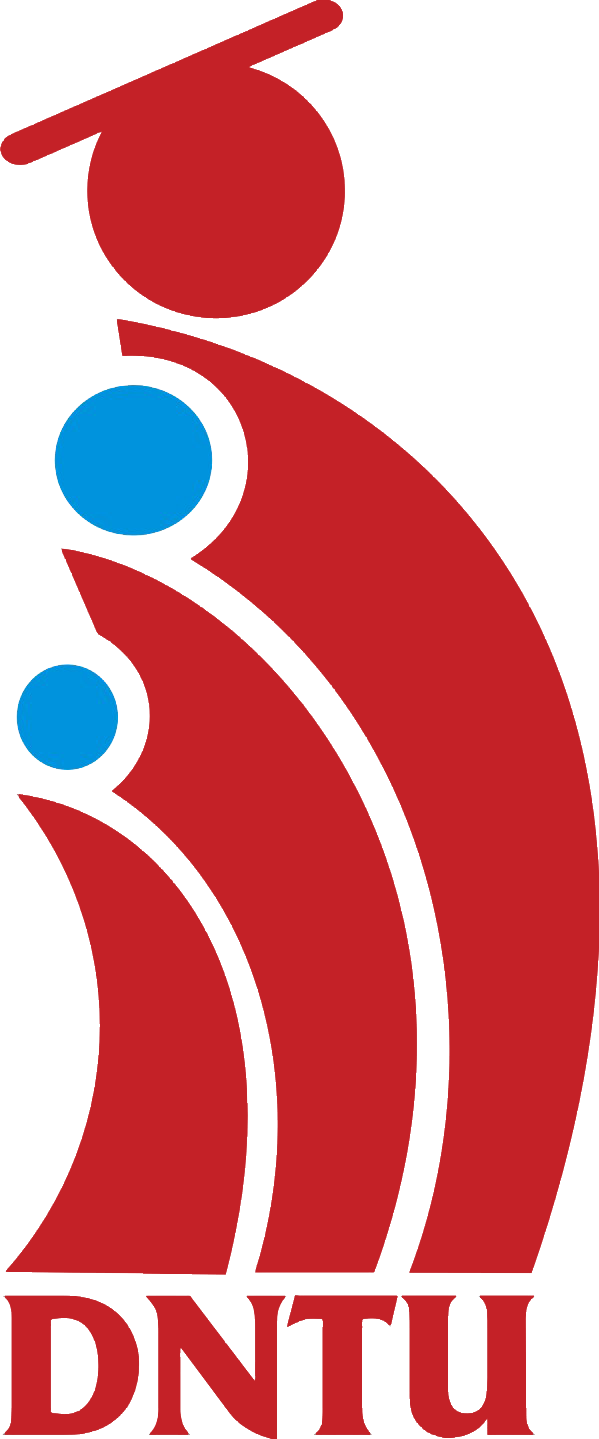
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐỒNG NAI**

**KHOA CÔNG NGHỆ THÔNG TIN**



**KIỂM TRA CUỐI KỲ**

**BÀI TẬP LỚN**

**TÊN ĐỀ TÀI: Task Manager**

**Tên học phần**: Lập Trình Mobile

**Mã học phần:** 0370091

**Chuyên ngành:** Công nghệ thông tin

**Khóa:** 18 (2022 - 2026)

**Sinh viên thực hiện:**

**Hoàng Anh Huy - 1822040269 – 22DTH2**

**Giảng viên: Lê Nhật Tùng**

**Flutter Task Manager**

[**1. Giới thiệu chung 2**](#_Toc197758010)

[**2. Mục tiêu dự án 2**](#_Toc197758011)

[**3. Công nghệ sử dụng 3**](#_Toc197758012)

[**4. Thiết kế của hệ thống 3**](#_Toc197758013)

[**4.1 API 3**](#_Toc197758014)

[**4.2 View 4**](#_Toc197758015)

[**4.3 Model 4**](#_Toc197758016)

[**5. Quy trình xây dựng 5**](#_Toc197758017)

[**6. Tổng quát ứng dụng 8**](#_Toc197758018)

[**7. Các chức năng theo vai trò: 8**](#_Toc197758019)

[**8. Kết quả đạt được 8**](#_Toc197758020)

[**9. Kết luận 8**](#_Toc197758021)

**BÁO CÁO DỰ ÁN XÂY DỰNG ỨNG DỤNG FLUTTER TASK MANAGER SỬ DỤNG XAMPP API**

# 1. Giới thiệu chung

Trong thời đại công nghệ số hiện nay, việc quản lý công việc hiệu quả, khoa học là một yêu cầu thiết yếu đối với cá nhân và doanh nghiệp. Ứng dụng Flutter Task Manager được xây dựng nhằm giúc người dùng quản lý nhiệm vụ một cách linh hoạt, hiệu quả. Trong dự án này, API backend được xây dựng với XAMPP, giúp dễ dàng triển khai và tích hợp.

# 2. Mục tiêu dự án

* Xây dựng một ứng dụng Flutter Task Manager có giao diện trực quan, dễ sử dụng.
* Kết nối với backend API sử dụng PHP và MySQL chạy trên XAMPP.
* Cung cấp các chức năng: Thêm, xóa, sửa, tìm kiếm.

# 3. Công nghệ sử dụng

* **Flutter**: Ngôn ngữ lập trình giao diện do Google phát triển.
* **Dart**: Ngôn ngữ backend cho Flutter.
* **PHP**: Ngôn ngữ server-side cho API.
* **MySQL**: Cơ sở dữ liệu quan hệ.
* **XAMPP**: Môi trường giả lập server local (Apache, MySQL, PHP, Perl).

# 4. Thiết kế của hệ thống

## **4.1 API**

**-UserAPIService:**

**Tạo người dùng mới**: Nhập vào các thông tin như tên, mật khẩu, email

**Lấy danh sách người dùng**: Đưa vào danh sách người dùng cho quản trị viên điều hành

**Lấy thông tin người dùng**: Đưa về thông tin đăng ký của người dùng như Tên, email, mật khẩu

**Đăng nhập**: Đưa thông tin để kiểm tra tồn tại của người dùng, trả về đăng nhập thành công hoặc thất bại nếu lỗi thông tin

**Xóa người dùng**: Chỉ dành cho quản trị viên Database

**-TaskAPIService**

**Create:** Tạo công việc mới

**Read:** Đọc hoặc xem các công việc đã tạo, của tất cả mọi người (chỉ dành cho admin)

**Update:** Cập nhật, chỉnh sửa công việc

**Delete:** Xóa công việc của từng người dùng

**Search:** Dựa theo từ khóa Input và lọc ra các công việc liên quan

## **4.2 View**

**-TaskAddScreens:** Màn hình quản lý các chức năng tạo mới công việc

-**TaskDetailScreens:** Màn hình hiển thị đầy đủ chi tiết các thông tin của công việc

**-TaskEditScreens:** Màn hình để quản lý các chức năng chỉnh sửa công việc

**-TaskFormScreens:** Màn hình cho phép điền vào các thông tin cần thiết để tạo hoặc chỉnh sửa các thông tin đã có của công việc

**-TaskListItem:** Màn hình để hiển thị các công việc đã được tạo và sắp xếp theo mức độ ưu tiên của từng công việc

**-TaskListScreens:** Màn hình chính giúp quản lý và hiển thị TaskListScreens theo giao diện và chứa các chức năng như tìm kiếm và đăng xuất

**-TaskLoginScreens:** Màn hình đăng nhập chứa các ô nhập thông tin đăng nhập của tài khoản đã có sẵn

-**TaskRegisterScreens:** Màn hình để người dùng đăng ký tài khoản để sử dụng ứng dụng

## **4.3 Model**

**-Users**

**id**: Mã định danh của từng người dùng.

**username**: Tên đăng nhập, dài từ 6 đến 20 ký tự, giúp người dùng đăng nhập vào ứng dụng.

**password**: Mật khẩu, có thể chứa ký tự đặc biệt.

**email**: Địa chỉ email hợp lệ.

**avatar**: Ảnh đại diện của người dùng.

**createdAt**: Thời gian tạo tài khoản.

**lastActive**: Thời gian hoạt động gần nhất.

**role**: Vai trò của người dùng.

**-Taskdb**

**id**: Mã định danh cho từng công việc.

**title**: Tiêu đề công việc.

**description**: Mô tả chi tiết công việc cần làm.

**status**: Tiến độ hiện tại của công việc, bao gồm các giá trị: Cần làm (to do), đang thực hiện (in progress), đã xong (done), đã hủy (cancelled).

**priority**: Mức độ ưu tiên của công việc, gồm các giá trị cao (1), trung bình (2), hoặc thấp (3).

**dueDate**: Hạn chót của công việc cần hoàn thành.

**createdAt**: Thời gian tạo công việc, định dạng ngày giờ.

**updatedAt**: Thời gian cập nhật gần nhất, giúp theo dõi lịch sử thay đổi.

**createdBy**: Mã định danh của người tạo công việc.

**assignedTo**: Mã định danh của người được giao công việc.

**category**: Danh mục công việc.

**attachments**: Danh sách đường dẫn đến các tệp đính kèm, như hình ảnh hoặc tài liệu, có thể để trống.

**completed**: Trạng thái hoàn thành, xác định công việc đã xong (true) hay chưa (false).

# 5. Quy trình xây dựng

Để hoàn thiện ứng dụng Flutter Task Manager sử dụng API PHP với XAMPP, nhóm đã triển khai theo một quy trình rõ ràng gồm 5 bước chính. Mỗi bước đều đóng vai trò quan trọng trong việc đảm bảo tính đúng đắn, hiệu quả và khả năng mở rộng của hệ thống.

**Bước 1: Thiết kế cơ sở dữ liệu (CSDL) trong phpMyAdmin sử dụng XAMPP**

* Sử dụng phần mềm XAMPP để tạo môi trường máy chủ ảo (Apache + MySQL).
* Khởi động phpMyAdmin từ trình duyệt để thiết kế CSDL.
* Tạo một database mới (ví dụ: taskdb) và bảng tasks chứa các cột chính như:
  + id (INT, AUTO\_INCREMENT, PRIMARY KEY)
  + title (VARCHAR) – tiêu đề công việc
  + description (TEXT) – mô tả công việc
  + status, priority, category (VARCHAR) – trạng thái, độ ưu tiên, phân loại
  + due\_date, created\_at, updated\_at (DATETIME) – ngày đến hạn, ngày tạo, ngày cập nhật
  + created\_by, assigned\_to (VARCHAR) – người tạo, người được giao
  + attachments (TEXT) – danh sách file đính kèm (mã hóa JSON)
  + completed (TINYINT) – trạng thái hoàn thành (0/1)

Việc thiết kế bảng dữ liệu một cách rõ ràng giúp cho việc thao tác CRUD từ API và ứng dụng Flutter dễ dàng, logic và đồng bộ hơn.

**Bước 2: Xây dựng các API endpoint trong PHP**

* Tạo file tasks.php trong thư mục htdocs/taskdb/ để xử lý các yêu cầu HTTP.
* Sử dụng phương pháp RESTful API với 4 thao tác chính:
  + GET → Lấy danh sách nhiệm vụ (hoặc tìm kiếm theo từ khóa)
  + POST → Tạo nhiệm vụ mới
  + \_method=PUT (dùng kèm POST) → Cập nhật nhiệm vụ
  + \_method=DELETE (dùng kèm POST) → Xóa nhiệm vụ
* Mỗi yêu cầu sẽ nhận và xử lý dữ liệu JSON từ client, tương tác với MySQL thông qua kết nối mysqli hoặc PDO.
* API trả về dữ liệu ở định dạng JSON để Flutter dễ dàng xử lý.

**Bước 3: Tạo và cấu hình Flutter Project, thiết kế giao diện (UI)**

* Tạo project Flutter mới bằng lệnh flutter create task\_manager\_app.
* Cài đặt các package cần thiết như:
  + http – để gọi API
  + intl – xử lý định dạng ngày giờ
  + provider hoặc riverpod – quản lý trạng thái (nếu dùng)
* Thiết kế các màn hình chính:
  + Danh sách công việc (hiển thị từ API)
  + Form thêm/sửa công việc
  + Tìm kiếm
  + Chi tiết công việc
* Giao diện được xây dựng theo nguyên tắc Material Design với bố cục rõ ràng, dễ thao tác trên di động.

**Bước 4: Kết nối Flutter với API PHP**

* Tạo lớp TaskAPIService để xử lý các yêu cầu HTTP từ Flutter.
* Sử dụng http.post để giao tiếp với tasks.php.
* Gửi dữ liệu dưới dạng JSON, nhận lại danh sách task và chuyển đổi sang model Taskdb.
* Tạo lớp Taskdb để ánh xạ giữa dữ liệu JSON và object Dart.
* Triển khai các chức năng:
  + Tạo mới task (createTask)
  + Lấy tất cả task (getAllTasks)
  + Cập nhật task (updateTask)
  + Xóa task (deleteTask)
  + Tìm kiếm (searchTasks, searchAllTasks)

**Bước 5: Kiểm thử, sửa lỗi và hoàn thiện**

* Kiểm tra toàn bộ các chức năng CRUD:
  + Tạo mới, cập nhật, xóa có hoạt động đúng và phản hồi tức thì không?
  + Dữ liệu có đồng bộ giữa UI và MySQL không?
* Thử với dữ liệu trống, dữ liệu không hợp lệ để xử lý lỗi:
  + Kiểm tra lỗi mạng, lỗi API trả về, validate form trước khi gửi.
* Kiểm thử trên thiết bị thật (Android emulator hoặc máy thật).
* Tối ưu mã nguồn, bổ sung tính năng tìm kiếm nâng cao, phân trang nếu cần.
* Cuối cùng là hoàn thiện tài liệu báo cáo và đóng gói sản phẩm.

# 6. Tổng quát ứng dụng

Ứng dụng được xây dựng theo mô hình MVC được điều chỉnh để phù hợp với ngôn ngữ dart, phần mềm Android Studio và phần mềm giả lập API XAMPP

# 7. Các chức năng theo vai trò:

**- Admin:**

**+** Có thể xem toàn bộ các nhiệm vụ của tất cả các người dùng

+ Có thể Thêm, Xóa, Sửa, Tìm Kiếm và giao nhiệm vụ cho người khác

-**User:**

**+** Có thể Xem, Xóa, Sửa và Tìm Kiếm nhiệm vụ của bản thân.

+ Có thể cập nhật tiến độ nhiệm vụ thông qua ứng dụng

# 8. Kết quả đạt được

* Ứng dụng có thể hoạt động trên Android và iOS.
* API backend hoạt động trên localhost.
* Các chức năng thêm/xóa/sửa/tìm kiếm hoạt động ổn định.

# 9. Kết luận

Việc xây dựng ứng dụng **Flutter Task Manager** kết hợp với backend sử dụng **PHP/MySQL** trên nền tảng **XAMPP** không chỉ là một trải nghiệm kỹ thuật thực tiễn mà còn là một bước tiến quan trọng trong quá trình rèn luyện kỹ năng phát triển phần mềm toàn diện. Qua dự án này, người thực hiện đã có cơ hội tiếp cận, hiểu rõ và áp dụng quy trình phát triển ứng dụng theo mô hình client-server, vốn đang rất phổ biến trong ngành công nghiệp phần mềm hiện đại.

Trong quá trình thực hiện, các kỹ năng về thiết kế giao diện người dùng (UI) với **Flutter**, lập trình logic bằng **Dart**, cùng với kiến thức quản lý cơ sở dữ liệu bằng **MySQL** và xử lý dữ liệu phía máy chủ bằng **PHP** đã được củng cố và nâng cao đáng kể. Việc kết nối giữa ứng dụng di động và server thông qua các API RESTful là một điểm nhấn đặc biệt, cho phép ứng dụng hoạt động linh hoạt, dễ mở rộng, đồng thời chuẩn hóa cách giao tiếp giữa các tầng trong hệ thống.

Việc xử lý CRUD hoàn chỉnh cũng giúp người thực hiện hiểu rõ quy trình lưu trữ, truy xuất và cập nhật thông tin từ hệ quản trị cơ sở dữ liệu – một kỹ năng thiết yếu trong bất kỳ ứng dụng nào.

Quan trọng hơn, ứng dụng được phát triển có tính ứng dụng thực tiễn cao, có thể phục vụ nhu cầu quản lý công việc cá nhân hoặc nhóm nhỏ, và hoàn toàn có thể được mở rộng thành một hệ thống quản lý dự án chuyên nghiệp trong tương lai.