**ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

⋅⋅⋅⋅**🙧🕮🙥**⋅⋅⋅⋅



**BÁO CÁO HỌC PHẦN**

**PHÁT TRIỂN ỨNG DỤNG TRÊN THIẾT BỊ DI ĐỘNG**

**ĐỀ TÀI:**

**XÂY DỰNG ỨNG DỤNG QUẢN LÝ CÔNG VIỆC TRÊN ĐIỆN THOẠI DI ĐỘNG**

| **Giảng viên hướng dẫn :** | Đỗ Hữu Công |
| --- | --- |
| **Lớp :** | 20242IT6029001 |
| **Nhóm :** | 05 |
| **Sinh viên thực hiện :** | Nguyễn Tấn Dũng - 2021607770 |
|  | Phạm Thành Đông - 2022600318 |
|  | Trần Đức Hoàng - 2021605654 |
|  | Nguyễn Hữu Kiên - 2021600509 |
|  | Vàng Mạnh Quỳnh - 2021608679 |

**Hà Nội, 2025**

**LỜI CẢM ƠN**

Trong suốt quá trình tiếp cận với học phần *Phát triển ứng dụng trên thiết bị di động* và làm bài tập lớn, nhóm đã nhận được rất nhiều sự hướng dẫn, quan tâm và động viên từ giảng viên phụ trách học phần – thầy Đỗ Hữu Công. Qua đây chúng em xin được gửi lời cảm ơn chân thành và sâu sắc đến thầy. Chính phong cách giảng dạy tận tình, tỉ mỉ của thầy đã khơi gợi trong chúng em nguồn cảm hứng học tập vô hạn, giúp chúng em dễ dàng tiếp thu, lĩnh hội các kiến thức trên lớp và tinh thần đam mê khám phá các nội dung của học phần.

Nhóm cũng xin gửi lời cảm ơn chân thành tới Khoa Công nghệ Thông tin, Trường Công nghệ thông tin và Truyền thông thuộc Đại học Công nghiệp Hà Nội cùng các thầy, cô giảng viên khoa đã cung cấp tài liệu và tạo điều kiện để chúng em được học tập với học phần này. Những kiến thức mà chúng em tiếp thu được trong quá trình học tập với học phần chắc chắn là những hành trang vững chắc giúp chúng em tự tin bước vào thị trường lao động công nghệ hiện nay đòi hỏi ứng viên có trình độ, năng lực và kỹ thuật cao.

Trong toàn bộ thời gian hoàn thành đề tài ***“Xây dựng ứng dụng quản lý công việc trên điện thoại di động”***,chúng em đã cùng nhau tìm hiểu, thảo luận, học hỏi, tích lũy được rất nhiều kiến thức và kinh nghiệm. Trải qua quá trình nỗ lực, nhóm đã đạt được mục tiêu đề ra. Song, chúng em mong muốn nhận được thêm những ý kiến đóng góp quý báu của quý thầy, cô trong khoa để hoàn đề tài hoàn thiện hơn nữa.

Nhóm xin chân thành cảm ơn!

***Nhóm sinh viên thực hiện 05***

# MỤC LỤC

[MỤC LỤC 2](#_heading=h.dsc5furbw6ce)

[DANH MỤC HÌNH ẢNH 5](#_heading=h.n2am8jghi8kz)

[DANH MỤC BẢNG BIỂU 7](#_heading=h.w9x2yi297guv)

[PHẦN 1. MỞ ĐẦU 9](#_heading=h.5f5g7k7jl5ui)

[1.1. Tên đề tài 9](#_heading=h.jovv2iqa3h0b)

[1.2. Lý do chọn đề tài 9](#_heading=h.j7ko8zig81l8)

[1.3. Mục tiêu 10](#_heading=h.1fbz25tlh5v1)

[1.4. Bố cục đề tài 10](#_heading=h.jvh7tjpp5rz3)

[PHẦN 2. NỘI DUNG 11](#_heading=h.jx3o10e3z7jx)

[2.1. Tổng quan về dự án 11](#_heading=h.eevespvjeov)

[2.1.1. Mục đích của dự án 11](#_heading=h.btjryh9fmlxq)

[2.1.2. Mục tiêu của dự án 12](#_heading=h.phxs7pj0uy31)

[2.1.3. Phạm vi của dự án 13](#_heading=h.sjxmmqlyea40)

[2.1.4. Công cụ và môi trường 13](#_heading=h.hz0ahkhrzmkp)

[2.2. Xây dựng ứng dụng 14](#_heading=h.sbc65d6v1s5p)

[2.2.1. Phân tích yêu cầu hệ thống 14](#_heading=h.xclwhbxquze5)

[2.2.1.1. Yêu cầu chức năng 14](#_heading=h.13y72qep49d)

[2.2.1.2. Yêu cầu phi chức năng 17](#_heading=h.c853fy355vx4)

[2.2.1.3. Yêu cầu về dữ liệu 17](#_heading=h.66umacyrsd5i)

[2.2.1.4. Mô hình hóa chức năng 18](#_heading=h.ix7wp26ks86q)

[2.2.1.5. Thiết kế cơ sở dữ liệu mức logic 21](#_heading=h.mpoiockevmya)

[2.2.1.6. Thiết kế cơ sở dữ liệu mức vật lý 22](#_heading=h.xkm5l8ounz4m)

[2.2.2. Thiết kế hệ thống 22](#_heading=h.dvoffoaj6fr9)

[2.2.2.1. Thiết kế kiến trúc chương trình 22](#_heading=h.86fwcysnasfr)

[2.2.2.2. Thống nhất trong thiết kế 23](#_heading=h.2y2bws1ze51e)

[2.2.2.3. Thiết kế cơ sở dữ liệu 25](#_heading=h.qkdrrqyd08m7)

[2.3. Xây dựng cấu trúc chương trình 27](#_heading=h.kkqsebnmtaja)

[2.3.1. Quản lý nhãn 27](#_heading=h.e7i9q71ds7ux)

[2.3.2. Quản lý danh mục 31](#_heading=h.28ohud6plfy5)

[2.3.3. Xem màn hình công việc theo thời gian 37](#_heading=h.h7hlz0ueibcq)

[2.3.4. Quản lý danh sách nhiệm vụ 41](#_heading=h.nlzz4ugygv36)

[2.3.4.1. Xem danh sách nhiệm vụ 41](#_heading=h.gphginoewiv1)

[2.3.4.2. Thêm mới nhiệm vụ 46](#_heading=h.hnnac28mf77p)

[2.3.4.3. Xem chi tiết nhiệm vụ 53](#_heading=h.qtklbr7kkia5)

[2.3.4.4. Tìm kiếm nhiệm vụ 59](#_heading=h.1g00o3sa9ocr)

[2.3.5. Thống Kê 65](#_heading=h.gdtx4zr946mr)

[2.3.6. Đăng nhập và đăng kí 71](#_heading=h.2vh9d7rkext)

[2.3.6.1. Đăng nhập 71](#_heading=h.txpqd4446qrr)

[2.3.6.2. Đăng kí 75](#_heading=h.wgxk9x8op6j2)

[2.3.7. Chức năng Pomodoro Timer 79](#_heading=h.td7f7rvrvn5u)

[2.3.8. Chức năng Habit Tracking 83](#_heading=h.69izeyhue8z2)

[PHẦN 3. KIẾN THỨC LĨNH HỘI VÀ BÀI HỌC KINH NGHIỆM 91](#_heading=h.4tj081im6pm1)

[3.1. Nội dung đã thực hiện 91](#_heading=h.2rvjjetexak7)

[3.2. Hướng phát triển 93](#_heading=h.l6jhndp8ianl)

[TÀI LIỆU THAM KHẢO 94](#_heading=h.238hrayx81j8)

# DANH MỤC HÌNH ẢNH

[Hình 2.1: Biểu đồ use case 19](#_heading=h.tjcd2ess8r8x)

[Hình 2.2: Biểu đồ quan hệ giữa các use case 20](#_heading=h.hlcrdz7viroy)

[Hình 2.3: Thiết kế cơ sở dữ liệu mức logic 21](#_heading=h.9n7so5t2oaij)

[Hình 2.4: Thiết kế cơ sở dữ liệu mức vật lý 22](#_heading=h.4vbq1f54cd4w)

[Hình 2.5: Hình dung màn hình quản lý nhãn 28](#_heading=h.v1x27x3is5c9)

[Hình 2.6: Màn hình android quản lý nhãn 29](#_heading=h.drknk9l7la8q)

[Hình 2.7: Giao diện màn hình quản lý nhãn 29](#_heading=h.s8dvktqihtd9)

[Hình 2.8: Hình dung màn hình quản lý danh mục 33](#_heading=h.s9nh42zfvyuk)

[Hình 2.9: Màn hình android quản lý danh mục 34](#_heading=h.4jcyli20hvao)

[Hình 2.10: Giao diện màn hình quản lý danh mục 35](#_heading=h.d2m7sfvmtisu)

[Hình 2.11: Hình dung màn hình xem công việc theo thời gian 38](#_heading=h.8kf0ytz25hgn)

[Hình 2.12: Màn hình android xem công việc theo thời gian 39](#_heading=h.ndfev2q0w527)

[Hình 2.13: Giao diện xem màn hình công việc theo thời gian 39](#_heading=h.cg1fhb975oky)

[Hình 2.14: Hình dung màn hình xem danh sách nhiệm vụ 42](#_heading=h.ckm2ivgtf1ep)

[Hình 2.15: Màn hình android xem danh sách nhiệm vụ 43](#_heading=h.7naoemw9d9ya)

[Hình 2.16: Màn hình xem danh sách nhiệm vụ 44](#_heading=h.yrn9xageq2jb)

[Hình 2.17: Hình dung màn hình thêm nhiệm vụ mới 47](#_heading=h.f5uf2vrprynm)

[Hình 2.18: Màn hình android thêm mới nhiệm vụ 48](#_heading=h.u7r6sxdeimv2)

[Hình 2.19: Giao diện màn hình thêm mới nhiệm vụ (1) 49](#_heading=h.qu8u6zbyqm27)

[Hình 2.20: Giao diện màn hình thêm mới nhiệm vụ (2) 49](#_heading=h.txrdwygl5awg)

[Hình 2.21: Hình dung màn hình xem chi tiết nhiệm vụ 54](#_heading=h.oca01w33bzuw)

[Hình 2.22 Màn hình android xem chi tiết nhiệm vụ 56](#_heading=h.q6ivjlg42c5l)

[Hình 2.23: Giao diện màn hình xem chi tiết nhiệm vụ 56](#_heading=h.k1glwkro56s)

[Hình 2.24: Giao diện màn hình xem chi tiết nhiệm vụ (2) 57](#_heading=h.czl3pbbg6e8l)

[Hình 2.25: Hình dung màn hình tìm kiếm nhiệm vụ 60](#_heading=h.z7ml2wmx9dcj)

[Hình 2.26: Màn hình android tìm kiếm nhiệm vụ 61](#_heading=h.tjj6qixigk8b)

[Hình 2.27: Giao diện màn hình tìm kiếm nhiệm vụ (1) 62](#_heading=h.hfwc41p16064)

[Hình 2.28: Giao diện màn hình tìm kiếm nhiệm vụ (2) 62](#_heading=h.v7htfujifxud)

[Hình 2.29: Hình dung màn hình thống kê 66](#_heading=h.otv79c7nes3c)

[Hình 2.30: Màn hình android thống kê 67](#_heading=h.gwmi593ypxsw)

[Hình 2.31: Giao diện màn hình thống kê 67](#_heading=h.gmt7c3nsxjyd)

[Hình 2.32: Hình dung màn hình android quản lý đăng nhập 71](#_heading=h.efwrg7rftr6c)

[Hình 2.33: Màn hình android quản lý đăng nhập 72](#_heading=h.gpym7kl56uhh)

[Hình 2.34: Giao diện android quản lý đăng nhập 73](#_heading=h.oy8mbmt6lb7s)

[Hình 2.35: Hình dung màn hình đăng ký 76](#_heading=h.szk2wafit95q)

[Hình 2.36: Màn hình android quản lý đăng kí 76](#_heading=h.9p2741iknrl1)

[Hình 2.37: Giao diện android quản lý đăng kí 77](#_heading=h.z5p475qsmd8y)

[Hình 2.38: Hình dung màn hình pomodoro timer 81](#_heading=h.ce1behd7ud7b)

[Hình 2.39: Màn hình pomodoro timer 81](#_heading=h.kmvqmrxs371i)

[Hình 2.40 Giao diện màn hình pomodoro timer 82](#_heading=h.9az1l0hi7odo)

[Hình 2.41: Hình dung màn hình Habit tracker 85](#_heading=h.r6eiyjy0jz14)

[Hình 2.42: Màn hình hiển thị habit và add habit 86](#_heading=h.jpfy4fhc4lu4)

[Hình 2.43: Giao diện màn hình hiển thị habit và add habit 86](#_heading=h.aidi6yl4yllg)

# DANH MỤC BẢNG BIỂU

[Bảng 2.1: Bảng Users 25](#_heading=h.x88m4bgg1qcs)

[Bảng 2.2: Bảng Categories 26](#_heading=h.vc2omppqqxo9)

[Bảng 2.3: Bảng Tags\_Tasks 26](#_heading=h.70xpt2d69ytf)

[Bảng 2.4: Bảng Tasks 26](#_heading=h.q84hfiip17hk)

[Bảng 2.5: Bảng Tags 27](#_heading=h.54uom9n9nl38)

[Bảng 2.6: Ứng dụng thư viện để chọn màu cho nhãn 30](#_heading=h.xy9b1o4baxri)

[Bảng 2.7: Phương thức addTag() 30](#_heading=h.zfetf2wivcwh)

[Bảng 2.8: Phương thức deleteTag() 30](#_heading=h.bbt4nbzb2u86)

[Bảng 2.9: Phương thức getTagByID() 31](#_heading=h.qzd4w0bbn71q)

[Bảng 2.10: Phương thức addCategory() 35](#_heading=h.c694kr1a0pxk)

[Bảng 2.11: Phương thức updateCategory() 36](#_heading=h.mbkygwtv60me)

[Bảng 2.12: Phương thức removeCategory() 37](#_heading=h.bdied1lbswqi)

[Bảng 2.13: Code lấy tính toán theo ngày 40](#_heading=h.116r7w1y0eq)

[Bảng 2.14: Code xử lý điều chỉnh thời gian 40](#_heading=h.rqkzui382d55)

[Bảng 2.15: Code lọc ra công việc theo từng ngày 41](#_heading=h.sadcuzchf2yh)

[Bảng 2.16: Phương thức lọc nhiệm vụ theo nhóm 44](#_heading=h.liefnlz8z7dl)

[Bảng 2.17: Phương thức lưu nhiệm vụ 50](#_heading=h.6ppta8v60lkh)

[Bảng 2.18: Phương thức đặt nhắc nhở 51](#_heading=h.utqwdxwrenos)

[Bảng 2.19: Phương thức mở dialog và lắng nghe kết quả trả về 57](#_heading=h.uau6m8dcittn)

[Bảng 2.20: Phương thức cập nhật danh mục 58](#_heading=h.3hurb1rzyybx)

[Bảng 2.21: Phương thức lắng nghe thay đổi và thực hiện tìm kiếm 62](#_heading=h.tc3448gcypkl)

[Bảng 2.22: Phương thức lọc nhiệm vụ theo từ khóa và danh sách nhãn 63](#_heading=h.bjtt2h3pgin4)

[Bảng 2.23 Phương thức xử lý hiển thị biểu đồ 68](#_heading=h.tv819x49sh5)

[Bảng 2.24 Phương thức xử lý lọc công việc theo ngày, tháng 69](#_heading=h.zet2m78r0tgo)

[Bảng 2.25 Phương thức handleLogin() 73](#_heading=h.4ski5vtr4bup)

[Bảng 2.26: Phương thức registerButton() 77](#_heading=h.hxbzbwco7vv1)

[Bảng 2.27: Phương thức registerButton() 78](#_heading=h.jr1rie61blq7)

[Bảng 2.28 Phương thức startTimer(), startTimerUpdates(), pauseTimer() và skipSession() 82](#_heading=h.llx337ec5ff0)

[Bảng 2.29: Phương thức addHabit() 86](#_heading=h.rtc1kcz38hxa)

# MỞ ĐẦU

## Tên đề tài

***“Xây dựng ứng dụng quản lý công việc trên điện thoại di động"***.

## Lý do chọn đề tài

Theo báo cáo "Vietnam Mobile App Popularity 2025", người Việt dành trung bình 7.3 giờ/ngày để sử dụng điện thoại thông minh, tăng 1.8 giờ so với năm 2024. Xu hướng này phản ánh sự chuyển dịch mạnh mẽ sang nền tảng di động trong mọi hoạt động, từ giải trí đến quản lý công việc. Đặc biệt, 43% thời gian sử dụng điện thoại được dành cho mạng xã hội, nhưng nhu cầu quản lý công việc hiệu quả vẫn chưa được đáp ứng tương xứng. Thực tế này tạo ra nghịch lý: thiết bị di động trở thành công cụ chính trong đời sống, nhưng các ứng dụng hỗ trợ quản lý công việc lại thiếu tính chuyên biệt và phù hợp với thói quen người Việt.

Tình trạng thường xuyên quên công việc ngày càng trở nên nghiêm trọng hơn do ảnh hưởng từ các nền tảng video ngắn (short clip). Theo khảo sát từ VNResource (2023), sinh viên Việt Nam dành trung bình 5–7 giờ/ngày để lướt video ngắn trên các nền tảng như TikTok, YouTube Shorts và Facebook Reels. Đặc điểm nổi bật của short clip là cấu trúc "mì ăn liền" – nội dung thay đổi liên tục sau mỗi 15–60 giây, kích thích giải phóng dopamine và tạo ra vòng lặp "lướt vô thức", khiến cho người xem suy giảm sự tập trung và khả năng ghi nhớ. Song, các phương pháp ghi chú truyền thống như giấy nhớ tỏ ra kém hiệu quả xét cả về khả năng tốc ký, khả năng nhắc nhở và khả năng truy xuất thông tin. Chính vì vậy, phương pháp này đã được thay thế bởi các ứng dụng quản lý công việc hiện đại như Microsoft To do, Google Tasks,… Các ứng dụng hỗ trợ người dùng ở rất nhiều tác vụ đa năng song còn tồn đọng hai vấn đề không thể giải quyết:

* Phần đông người dùng mới chỉ cần một ứng dụng đơn giản thay thế cho giấy nhớ, không cần tích hợp nhiều tính năng phức tạp.
* Dữ liệu cá nhân của người dùng cần được tổ chức lưu trữ cục bộ bảo vệ tính riêng tư.

Nắm bắt được cơ hội này, nhóm đã quyết định cùng nhau xây dựng đề tài “***Xây dựng ứng dụng quản lý công việc trên thiết bị di động*”** với mục tiêu tiếp cận được lớp đối tượng khách hàng mới của thị trường phần mềm quản lý năng suất công việc.

## Mục tiêu

Dự án kỳ vọng đạt được được những mục tiêu sau:

* Xây dựng ứng dụng quản lý công việc đơn giản trên nền tảng Android.
* Xây dựng các module chức năng cơ bản của ứng dụng như đăng nhập, đăng kí tài khoản, thêm mới và quản lý danh mục, công việc,… đạt được MVP.
* Tạo dựng hứng thú của người dùng với các tính năng bổ sung như thống kê hoạt động, đồng hồ Pomodoro,…
* Ứng dụng các kiến thức đã học trong học phần như kiến thức về OOP, xây dựng tương tác với hệ quản trị cơ sở dữ liệu SQLite qua DAO (Data Access Object), xây dựng được các chức năng CRUD cho đối tượng (Create, Read, Update, Delete).

## Bố cục đề tài

Nội dung chính của đề tài gồm 3 phần:

**Phần 1. MỞ ĐẦU**

Xác định bối cảnh xây dựng đề tài, tổng quan về đề tài.

**Phần 2. NỘI DUNG**

Tìm hiểu và xác định hướng giải quyết cho bài toán. Thực hiện các hoạt động nghiệp vụ xây dựng cơ sở dữ liệu và các module chức năng.

**Phần 3: KIẾN THỨC LĨNH HỘI VÀ BÀI HỌC KINH NGHIỆM**

Đánh giá kết quả đạt được so với các mục tiêu ban đầu. Bài học kinh nghiệm về quy trình phát triển phần mềm và các kỹ nghệ nghiệp vụ. Đề xuất các hướng mở rộng, phát triển của dự án.

# NỘI DUNG

## Tổng quan về dự án

### Mục đích của dự án

Tasktory là một ứng dụng được nhóm phát triển trên nền tảng thiết bị di động với mục đích chính là giải quyết các vấn đề quản lý công việc cá nhân trong bối cảnh cuộc sống đô thị hiện đại ngày càng bận rộn và phức tạp. Người dùng hiện đang phải đối mặt với lượng lớn tác vụ, các áp lực về thời hạn và trách nhiệm đan xen dẫn đến tình trạng quá tải, bỏ sót công việc quan trọng, hoặc khó khăn sắp xếp thứ tự ưu tiên.

Tasktory hướng đến cung cấp một giải pháp đơn giản, nhưng hiệu quả, trực quan, giúp người dùng:

* Tổ chức và theo dõi mọi công việc của họ tại một ứng dụng tập trung.
* Phân loại công việc theo danh mục và thẻ tuỷ chỉnh cá nhân hoá, giúp dễ dàng tìm kiếm và sắp xếp các công việc.
* Tạo lập cho đối tượng người dùng mới thói quen quản lý các công việc thông qua các hoạt động đặt mức độ ưu tiên và đặt nhắc nhở cho công việc.
* Đơn giản hoá việc quản lý công việc, từ đó giảm căng thẳng và nâng cao hiệu suất công việc cho người dùng.

Với giao diện thân thiện, dễ dàng tiếp cận người dùng mới, Tasktory hướng đến trở thành trợ lý cá nhân đáng tin cậy, góp phần cải thiện chất lượng công việc của người dùng.



### Mục tiêu của dự án

Dự án tập trung xây dựng các chức năng nền tảng:

1. ***Quản lý tài khoản***
   * *Chức năng:* Cho phép người dùng thực hiện các chức năng Đăng kí, Đăng nhập, Đổi mật khẩu cho tài khoản cá nhân của người dùng
   * *Lợi ích:* Cá nhân hoá các nội dung của người dùng trong quá trình sử dụng ứng dụng
2. ***Quản lý các danh mục, thẻ công việc***
   * *Chức năng:* Cho phép người dùng phân loại các công việc cần làm.
   * *Lợi ích:* Giúp người dùng có thể phân chia các loại công việc một cách rõ ràng để dễ dàng thực hiện.
3. ***Quản lý các công việc theo thời gian, danh mục***
   * *Chức năng:* Cho phép người dùng tạo ra các công việc và sắp xếp chúng theo thời gian đã đề ra và với từng danh mục cụ thể.
   * *Lợi ích:* Giúp người dùng có thể phân chia các công việc một cách rõ ràng để dễ dàng thực hiện.
4. ***Xem và theo dõi tiến độ thực hiện các công việc***
   * *Chức năng:* Cho phép người dùng xem các công việc cần làm theo thời gian và các danh mục công việc.
   * *Lợi ích:* Giúp người dùng theo dõi các công việc của mình, việc nào đã làm, việc nào chưa làm.
5. ***Thống kê các công việc***
   * *Chức năng:* Cho phép người dùng xem thống kê các công việc đã hoàn thành và chưa hoàn thành.
   * *Lợi ích:* Giúp người dùng theo dõi một cách tổng quan về các công việc đã và chưa hoàn thành.
6. ***Các tính năng mở rộng: Pomodoro Timer(đồng hồ pomodoro), Habit Tracker (Trình theo dõi thói quen), Challenge(thử thách hàng ngày)***
   * *Chức năng:* Cho phép người dùng sử dụng các tính năng thú vị được tích hợp trong ứng dụng
   * *Lợi ích:* Giúp người dùng duy trì được sự hứng thú khi sử dụng phần mềm, với những công cụ, kỹ thuật đặc trưng cho các phương pháp xây dựng thói quen công việc tốt.

### Phạm vi của dự án

Tasktory được xây dựng trên nền tảng Android và được thiết kế để hỗ trợ người dùng quản lý công việc cá nhân một cách hiệu quả. Ứng dụng hỗ trợ người dùng quản lý các công việc cá nhân dựa trên các tính năng được cung cấp như:

* Quản lý Tài khoản Người dùng: Đăng ký, đăng nhập, đặt lại mật khẩu.
* Quản lý Tác vụ: Tạo mới, xem danh sách, xem chi tiết, chỉnh sửa, xóa, đánh dấu hoàn thành, đặt mức độ ưu tiên, ngày hết hạn.
* Phân loại và Gắn thẻ: Tạo và quản lý danh mục, gán tác vụ vào danh mục, xem tác vụ theo danh mục. Tạo và quản lý thẻ, gán thẻ cho tác vụ.
* Tìm kiếm Tác vụ: Tìm kiếm công việc dựa trên từ khóa, thuộc tính (ngày, ưu tiên), danh mục, thẻ.
* Hệ thống Nhắc nhở: Thiết lập nhắc nhở cho các tác vụ, nhận thông báo đẩy (push notification) khi đến hạn.
* Giao diện Người dùng: Màn hình chào mừng, màn hình chính hiển thị tổng quan công việc, các màn hình chức năng rõ ràng và dễ sử dụng.

Các nội dung nằm ngoài phạm vi dự án gồm có:

* Chức năng quản lý dự án phức tạp (biểu đồ Gantt, phân công nhóm, theo dõi tài nguyên).
* Tính năng cộng tác thời gian thực hoặc chat nhóm.
* Hỗ trợ đa nền tảng (iOS, Web) – chỉ tập trung vào Android.
* Tích hợp lịch biểu chi tiết với nhiều chế độ xem (ngày/tuần/tháng chuyên sâu).
* Tích hợp sâu với các dịch vụ bên thứ ba (ví dụ: CRM, Email Marketing).
* Phân tích dữ liệu và báo cáo nâng cao về hiệu suất làm việc.
* Và các tính năng khác không được đề cập tới trong mục tiêu của dự án

### Công cụ và môi trường

***Công cụ và môi trường phát triển***

* Ngôn ngữ lập trình: Java
* IDE: Android Studio (Phiên bản mới nhất)
* Hệ thống Build: Gradle
* Thư viện chính: Android Jetpack (ViewModel, LiveData, Room, Navigation)
* Hệ điều hành: Windows 11
* Công cụ quản lý mã nguồn: Github
* Công cụ xây dựng lược đồ hệ cơ sở dữ liệu: Rational Rose
* Công cụ vẽ hình dung màn hình: Balsamiq

***Công cụ và môi trường kiểm thử***

* Công cụ kiểm thử thủ công: LDPlayer 9
* Hệ điều hành: Windows 11

## Xây dựng ứng dụng

### Phân tích yêu cầu hệ thống

#### Yêu cầu chức năng

*Quản lý tài khoản người dùng*

* Đăng ký tài khoản (Register.java)
* Đăng nhập (Login.java)
* Đặt lại mật khẩu (ResetPassword.java)
* Thay đổi mật khẩu (ChangePasswordFragment.java)
* Quản lý thông tin cá nhân (PersonalFragment.java)

*Quản lý tác vụ*

* Tạo tác vụ mới với tiêu đề, ghi chú, thời hạn, thời gian nhắc nhở
* Xem danh sách tác vụ (HomeFragment.java)
* Xem chi tiết tác vụ (TaskDetailActivity.java)
* Chỉnh sửa tác vụ bao gồm tiêu đề, ghi chú (EditTaskTitleFragment.java)
* Cập nhật trạng thái tác vụ (đánh dấu hoàn thành/chưa hoàn thành)
* Xóa tác vụ (DeleteTaskFragment.java)
* Thay đổi danh mục của tác vụ

*Phân loại và gắn thẻ*

* Quản lý danh mục (CategoryFragment.java)
* Chọn danh mục cho tác vụ (ChooseCategoryFragment.java)
* Quản lý thẻ (TagManagement.java)
* Gắn thẻ cho tác vụ (ChooseTagFragment.java)
* Xem tác vụ theo danh mục (TasksByCategory.java)

*Theo dõi thói quen (Habit Tracker)*

* Tạo thói quen mới (AddHabitDialogFragment.java)
* Xem danh sách thói quen (HabitFragment.java)
* Thiết lập tần suất (hàng ngày, hàng tuần, khoảng thời gian)
* Theo dõi tiến độ (đếm số lần hoàn thành)
* Đặt mục tiêu số ngày
* Đặt nhắc nhở thói quen
* Xóa thói quen (DeleteHabitDialogFragment.java)

*Đồng hồ Pomodoro*

* Thiết lập và chạy đồng hồ Pomodoro (PomodoroFragment.java)
* Tùy chỉnh thời gian làm việc/nghỉ ngơi (PomodoroSettingsDialogFragment.java)
* Theo dõi thời gian làm việc cho từng tác vụ (WorkSession)
* Chế độ tự động bắt đầu phiên nghỉ/làm việc

*Hệ thống thử thách (Challenge)*

* Tạo thử thách cá nhân (AddChallengeDialogFragment.java)
* Xem danh sách thử thách (ChallengeFragment.java)
* Thiết lập mục tiêu và thời gian kết thúc
* Xóa thử thách (DeleteChallengeDialogFragment.java)

*Tìm kiếm và lọc*

* Tìm kiếm tác vụ theo từ khóa (SearchActivity.java)
* Lọc tác vụ theo thẻ

*Nhắc nhở và thông báo*

* Đặt thời gian nhắc nhở cho tác vụ
* Xử lý sự kiện nhắc nhở (receiver package)

*Xem lịch*

* Hiển thị tác vụ theo dạng lịch (CalendarFragment.java)

*Chức năng khác*

* Chia sẻ (ShareFragment.java)
* Thông tin ứng dụng (AboutFragment.java)

**Phân tích các tính năng đặc biệt**

*Hệ thống Pomodoro đã được triển khai đầy đủ với các tính năng:*

* Đếm ngược thời gian làm việc (mặc định 25 phút)
* Đếm ngược thời gian nghỉ ngắn (mặc định 5 phút)
* Đếm ngược thời gian nghỉ dài (mặc định 15 phút)
* Tùy chỉnh số phiên làm việc trước khi nghỉ dài
* Lưu trữ lịch sử làm việc

*Theo dõi thói quen có các chức năng nâng cao:*

* Nhiều loại tần suất (hàng ngày, hàng tuần, khoảng thời gian)
* Theo dõi chuỗi ngày hoàn thành
* Hỗ trợ nhiều lần hoàn thành trong một ngày

*Hệ thống thử thách cho phép người dùng:*

* Tạo thử thách cá nhân với thời gian bắt đầu/kết thúc
* Theo dõi tiến độ thử thách
* Đánh dấu hoàn thành/chưa hoàn thành

#### Yêu cầu phi chức năng

* *Hiệu suất*: Ứng dụng phải phản hồi nhanh chóng các thao tác của người dùng. Thời gian phản hồi cho các thao tác người dùng dưới 0.5 giây, thời gian tải danh sách tác vụ dưới 1 giây, thời gian khởi động ứng dụng dưới 3 giây.
* *Tính ổn định*: Ứng dụng phải hoạt động ổn định, không gặp lỗi nghiêm trọng trong quá trình sử dụng. Ứng dụng không bị treo hoặc crash trong điều kiện sử dụng bình thường và dữ liệu người dùng không bị mất mát do lỗi ứng dụng.
* *Bảo mật*: Dữ liệu người dùng phải được bảo vệ, đảm bảo không bị rò rỉ hoặc truy cập trái phép. Mật khẩu phải được băm (hashed) và muối (salted) khi lưu trữ. Sử dụng HTTPS cho tất cả các giao tiếp mạng (nếu có).
* *Tương thích*: Ứng dụng phải tương thích với thiết bị di động sử dụng hệ điều hành Android phiên bản 5.0 (API level 21) trở lên. Hiển thị tốt trên các kích thước màn hình điện thoại phổ biến (4.7 inch - 6.7 inch).
* *Tính khả dụng*: Người dùng mới có thể thực hiện các tác vụ cơ bản trong vòng 5 phút làm quen. Giao diện phải trực quan, dễ sử dụng và nhất quán giữa các màn hình.

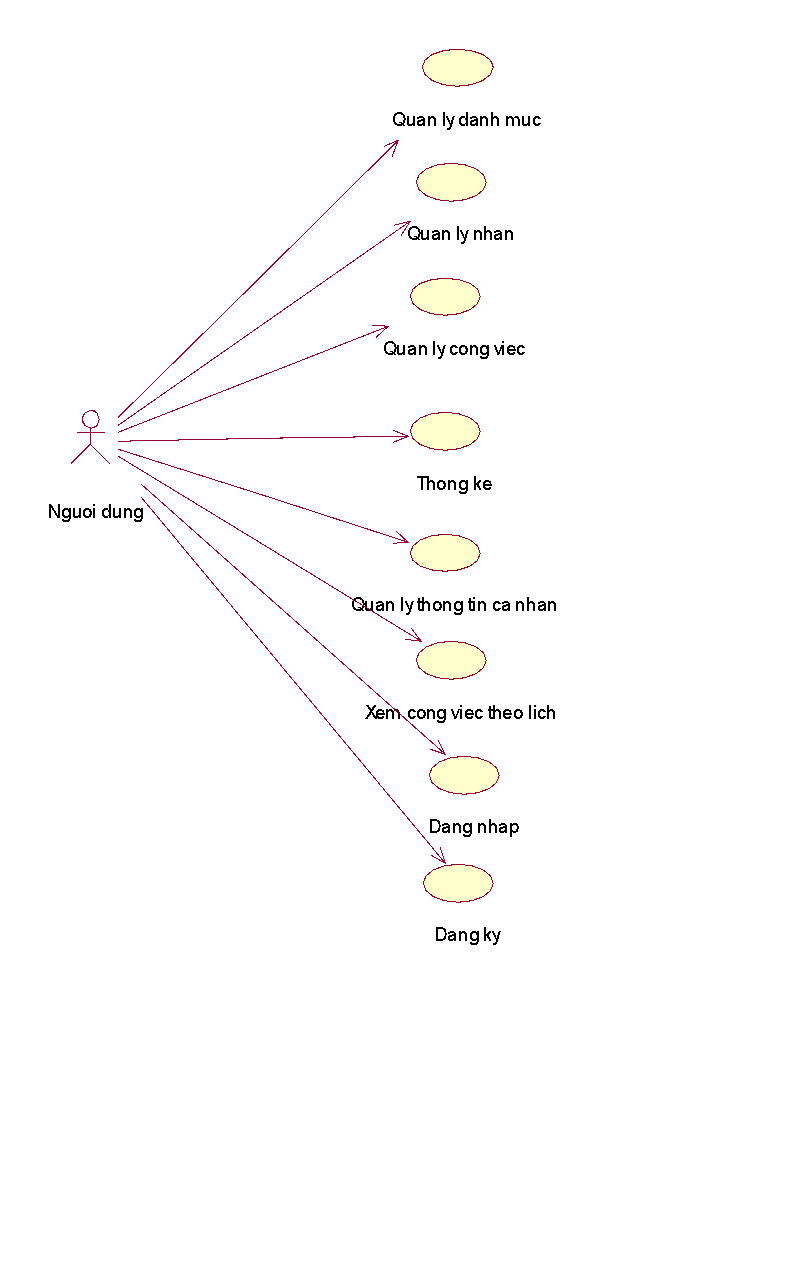
#### Yêu cầu về dữ liệu

Ứng dụng Tasktory là một ứng dụng quản lý công việc cá nhân, cung cấp thông tin chi tiết về các tác vụ, kế hoạch, danh mục và thẻ đã được tạo. Dữ liệu được lưu trữ trong cơ sở dữ liệu cục bộ sử dụng SQLite với Room Persistence Library, với các thực thể chính bao gồm:

* **USER:** Lưu trữ thông tin người dùng (userId, username, email, passwordHash, createdAt)
* **TASK:** Lưu trữ thông tin tác vụ (taskId, userId, title, description, dueDate, dueTime, priority, isCompleted, categoryId, createdAt, updatedAt)
* **CATEGORY:** Lưu trữ các danh mục phân loại (categoryId, userId, name, color)
* **TAG:** Lưu trữ các thẻ/nhãn (tagId, userId, name)
* **TASK\_TAG:** Bảng quan hệ giữa TASK và TAG (taskId, tagId)
* **REMINDER:** Thông tin về các nhắc nhở (reminderId, taskId, reminderDateTime)

#### Mô hình hóa chức năng

1. ***Biểu đồ Use Case***



*Hình 2.1: Biểu đồ use case*

**Đăng ký:** Cho phép người dùng đăng ký tài khoản cho ứng dụng.

**Đăng nhập:** Cho phép người dùng đăng nhập vào ứng dụng.

**Quản lý danh mục:** Cho phép người dùng thực hiện xem, thêm, sửa, xóa các danh mục trong ứng dụng.

**Quản lý công việc:** Cho phép người dùng thực hiện xem, thêm, sửa, xóa các công việc trong ứng dụng.

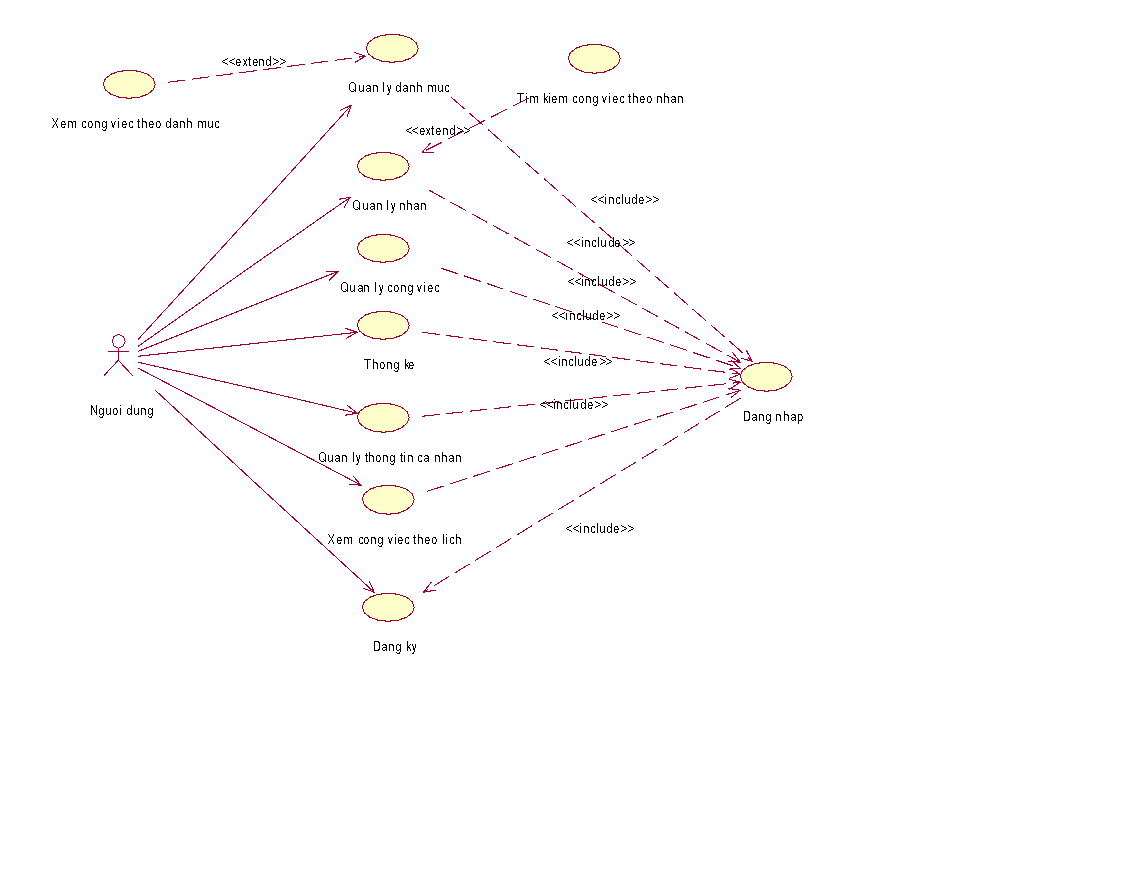
**Quản lý thông tin cá nhân:** Cho phép người dùng thực hiện xem các thông tin về công việc được lưu lại trong ứng dụng.

**Thống kê:** Cho phép người dùng xem biểu đồ thống kê về các công việc đã hoàn thành hoặc chưa hoàn thành trong ứng dụng.

**Quản lý nhãn:** Cho phép người dùng gán nhãn cho công việc.

**Xem công việc theo lịch:** Cho phép người dùng xem công việc theo lịch.

1. ***Biểu đồ quan hệ giữa các Use case***



*Hình 2.2: Biểu đồ quan hệ giữa các use case*

#### Thiết kế cơ sở dữ liệu mức logic

A diagram of a computer

Description automatically generated

*Hình 2.3: Thiết kế cơ sở dữ liệu mức logic*

#### Thiết kế cơ sở dữ liệu mức vật lý

A diagram of a network

Description automatically generated with medium confidence

*Hình 2.4: Thiết kế cơ sở dữ liệu mức vật lý*

### Thiết kế hệ thống

#### Thiết kế kiến trúc chương trình

*Presentation Layer (Tầng giao diện)*

* Activities: Các màn hình chính của ứng dụng (MainActivity, Login, TaskDetail, etc.)
* Fragments: Các thành phần UI có thể tái sử dụng (HomeFragment, HabitFragment, etc.)
* Adapters: Chuyển đổi dữ liệu từ Model thành View hiển thị trong RecyclerView
* UI Utilities: Các tiện ích hỗ trợ giao diện người dùng

*ViewModel/Presenter Layer (Tầng trung gian)*

* Đóng vai trò cầu nối giữa Presentation Layer và Model Layer
* Xử lý logic nghiệp vụ và chuẩn bị dữ liệu cho hiển thị
* Phân chia theo chức năng: Task Logic, Category Logic, Habit Logic, etc.

*Model Layer (Tầng dữ liệu)*

* Data Models: Các lớp đối tượng dữ liệu như Task, Category, Habit, Challenge, User
* Data Access Objects (DAO): Các lớp truy xuất cơ sở dữ liệu SQLite
* Business Logic: Các quy tắc nghiệp vụ độc lập với giao diện

*Database (Cơ sở dữ liệu)*

* SQLite Local Database: Lưu trữ chính cho dữ liệu ứng dụng
* Shared Preferences: Lưu trữ cài đặt và thông tin phiên làm việc

*Background Services (Dịch vụ nền)*

* Notification Service: Quản lý và tạo thông báo
* Timer Service: Dịch vụ đồng hồ Pomodoro chạy ngầm

*Broadcast Receivers*

* Alarm Receiver: Nhận sự kiện báo thức từ hệ thống
* Reminder Receiver: Xử lý sự kiện nhắc nhở cho tác vụ và thói quen

#### Thống nhất trong thiết kế

*Giao diện người dùng*

* Màu sắc:
  + Màu chủ đạo: Lavender (#8687E7 - sắc tím nhạt) và trắng (#FFFFFF)
  + Màu nhấn mạnh: Teal (#FF03DAC5) được sử dụng làm màu thứ cấp (secondary color)
  + Màu văn bản: Chủ yếu là màu đen (#000000) trên nền trắng để tối ưu khả năng đọc
  + Màu đặc biệt: Màu riêng cho các trạng thái như quá hạn (overdue\_color - #f28c8f) và không có hạn chót (no\_deadline\_color - #fcbb6d)
* Bố cục và thiết kế:
  + Thiết kế Material Design: Sử dụng Theme.MaterialComponents làm nền tảng thiết kế
  + Không có Action Bar: Áp dụng NoActionBar trong theme, thay vào đó sử dụng các giải pháp điều hướng hiện đại
  + Hiệu ứng động: Sử dụng các hiệu ứng như slide\_in\_bottom và slide\_out\_bottom cho dialog
  + Bố trí: Sử dụng bố cục đồng nhất cho các màn hình chính như home, calendar, category, challenge, và habit
* Thành phần UI:
* Bottom Sheet: Sử dụng cho form nhập tác vụ (bottom\_sheet\_task\_form.xml)
* Dialog: Nhiều dialog tùy chỉnh cho các tính năng (add\_category, add\_challenge, color\_picker, v.v.)
* Fragment: Tổ chức UI theo fragment để tái sử dụng và quản lý hiệu quả
* Item Layout: Thiết kế nhất quán cho các item trong danh sách (task, category, tag, habit, challenge)

*Quy tắc đặt tên*

* Quy tắc đặt tên cho resources:
  + Layout files: Sử dụng quy ước như activity\_\*, fragment\_\*, dialog\_\*, item\_\*
    - Ví dụ: activity\_login.xml, fragment\_home.xml, dialog\_add\_habit.xml, item\_task.xml
  + ID resources: Tuân theo quy tắc camelCase bắt đầu bằng loại component
    - Ví dụ: @+id/btnLogin, @+id/txtUsername, @+id/rvTasks.
* Quy tắc đặt tên cho code:
  + Class names: Pascal Case (chữ cái đầu viết hoa)
    - Ví dụ: MainActivity, HabitFragment, CategoryDAO
  + Variables và methods: Camel Case (chữ cái đầu viết thường)
    - Ví dụ: getUserId(), parseTask(), findAllByCategoryId()
  + Constants: Toàn bộ viết HOA và phân tách bằng dấu gạch dưới
    - Ví dụ: PREF\_NAME, KEY\_WORK\_DURATION, KEY\_SHORT\_BREAK\_DURATION
* Quy tắc đặt tên cho model:
  + Đặt tên theo chức năng rõ ràng: Task, Category, Tag, Habit, Challenge, PomodoroSettings
  + Thuộc tính: Sử dụng camelCase
    - Ví dụ: taskId, userId, reminderTime, isCompleted

#### Thiết kế cơ sở dữ liệu

Dữ liệu được lưu trữ trong SQLite. Các lớp dữ liệu được ánh xạ thành các bảng trong cơ sở dữ liệu, với mỗi thuộc tính của lớp tương ứng với một trường trong bảng.

***Thiết kế bảng***

*Bảng 2.1: Bảng Users*

A screenshot of a computer

Description automatically generated

*Bảng 2.2: Bảng Categories*

A screenshot of a computer

Description automatically generated

*Bảng 2.3: Bảng Tags\_Tasks*

A screenshot of a computer

Description automatically generated

*Bảng 2.4: Bảng Tasks*

A screenshot of a computer

Description automatically generated

*Bảng 2.5: Bảng Tags*

A screenshot of a computer

Description automatically generated

## Xây dựng cấu trúc chương trình

### Quản lý nhãn

***Mô tả***

Đối với màn hình quản lý nhãn sẽ gồm các thành phần chính sau

* Nút ImageButton để quay trở về giao diện trang chủ.
* Ô TextView để hiển thị tên màn hình quản lý nhãn. Danh sách RecycleView để hiển thị danh sách các nhãn công việc, bao gồm:
* View để hiển thị màu sắc của nhãn.
* TextView để hiển thị tên của nhãn.
* Nút FloatingActionButton ở góc màn hình để thêm một nhãn mới.

***Hình dung màn hình***

A screenshot of a cell phone

Description automatically generated

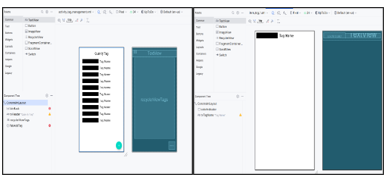
*Hình 2.5: Hình dung màn hình quản lý nhãn*

***Nghiệp vụ***

Người dùng có thể

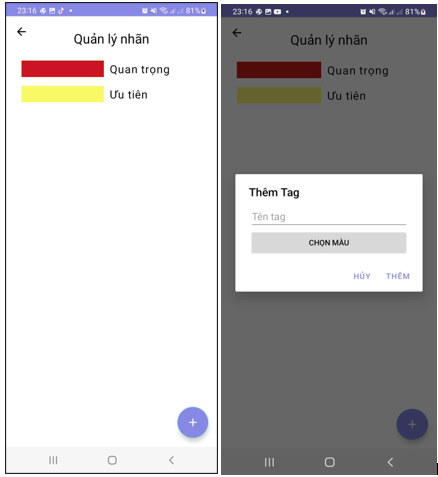
* Quay về giao diện chính bằng cách nhấn vào nút trở về.
* Xem danh sách các nhãn được hiển thị dưới dạng các mục trong RecyclerView.
* Thực hiện việc thêm mới một nhãn bằng cách bấm vào nút thêm mới, màn hình xuất hiện một hộp thông báo để chọn thông tin cần thêm mới.
* Sửa hoặc xóa một nhãn bằng cách lựa chọn chức năng tương ứng khi bấm vào từng nhãn.

***Màn hình android***

******

*Hình 2.6: Màn hình android quản lý nhãn*

***Giao diện khi chạy chương trình***



*Hình 2.7: Giao diện màn hình quản lý nhãn*

***Các phương thức đã sử dụng***

*Bảng 2.6: Ứng dụng thư viện để chọn màu cho nhãn*

| colorPickerButton.setOnClickListener(v -> {  AmbilWarnaDialog colorPicker = new AmbilWarnaDialog(this, selectedColor[0], new AmbilWarnaDialog.OnAmbilWarnaListener() {  @Override  public void onOk(AmbilWarnaDialog dialog, int color) {  // *Chọn màu và gán màu cho nhãn*  selectedColor[0] = color;  colorPickerButton.setBackgroundColor(color);  }  @Override  public void onCancel(AmbilWarnaDialog dialog) {  *// Người dùng hủy chọn màu* }  });  colorPicker.show(); }); |
| --- |

*Bảng 2.7: Phương thức addTag()*

| public long addTag(String name, String color) {  ContentValues values = new ContentValues();  values.put("name", name);  values.put("color", color);  values.put("user\_id", getUserId());  return db.insert("tags", null, values); } |
| --- |

*Bảng 2.8: Phương thức deleteTag()*

| public int deleteTag(int id) {  String whereClause = "tag\_id = ? AND user\_id = ?";  String[] whereArgs = {String.*valueOf*(id), String.*valueOf*(getUserId())};  return db.delete("tags", whereClause, whereArgs); } |
| --- |

*Bảng 2.9: Phương thức getTagByID()*

| public Tag getTagById(int tagId) {  Tag tag = null;  String selectQuery = "SELECT tag\_id, name, color FROM tags WHERE tag\_id = ? AND user\_id = ?";  String[] whereArgs = {String.*valueOf*(tagId), String.*valueOf*(getUserId())};  Cursor cursor = db.rawQuery(selectQuery, whereArgs);  if (cursor != null && cursor.moveToFirst()) {  int id = cursor.getInt(cursor.getColumnIndexOrThrow("tag\_id"));  String name = cursor.getString(cursor.getColumnIndexOrThrow("name"));  String color = cursor.getString(cursor.getColumnIndexOrThrow("color"));  tag = new Tag(id, name, color, getUserId());  cursor.close();  }  return tag; } |
| --- |

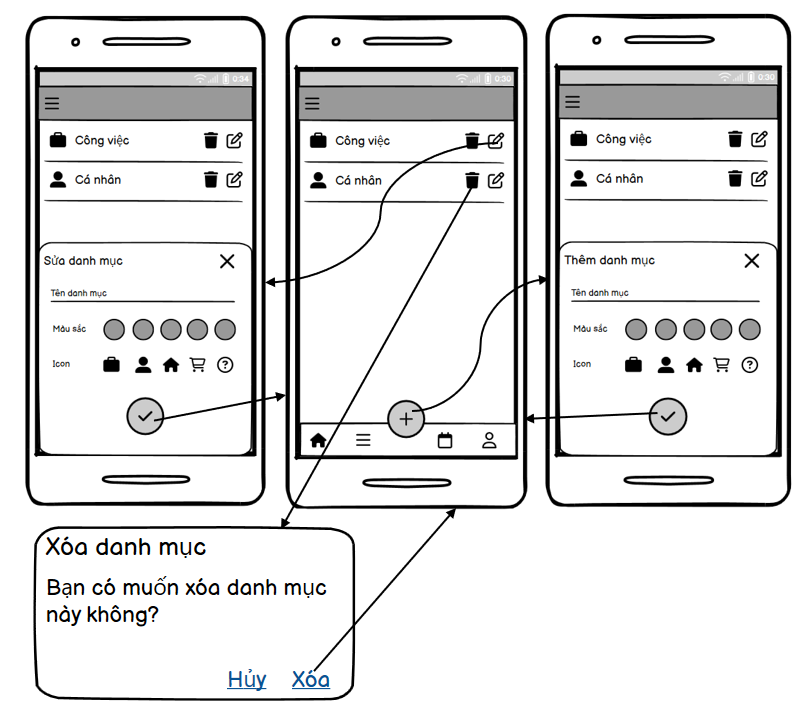
### Quản lý danh mục

***Mô tả***

Đối với màn hình danh sách danh mục sẽ gồm các thành phần chính sau

* Listview để hiển thị danh sách các danh mục lên màn hình
* 2 ImageButton dùng để thực hiện chức năng xóa và sửa danh mục
* 1 FloatingActionButton ở cuối màn hình để hiển thị hộp thoại Dialog thêm danh mục.
* Đối với màn hình thêm và sửa danh mục sẽ gồm các thành phần chính sau:
* TextView hiển thị tên hộp thoại
* EditText để nhập tên danh mục
* TextView “Màu sắc” và 5 ImageView hiển thị 5 màu sắc để người dùng lựa chọn màu sắc cho tên danh mục.
* TextView “Icon” và 5 ImageView hiển thị 5 icon để người dùng lựa chọn icon cho danh mục
* FloatingActionButton ở cuối màn hình để lưu danh mục khi hoàn tất.
* Đối với màn hình xem danh sách công việc theo danh mục sẽ gồm các thành phần chính sau:
* ImageView quản lý nhãn và tìm kiếm nhiệm vụ. RecyclerView hiển thị danh sách các nhóm nhiệm vụ
* Nhóm nhiệm vụ bao gồm: TextView hiển thị tên nhóm và số lượng nhiệm vụ bên trong. RecyclerView hiển thị danh sách nhiệm vụ
* Nhiệm vụ bao gồm: CheckBox hiển thị trạng thái và cập nhật trạng thái khi người dùng ấn, TextView hiển thị tiêu đề, ngày hết hạn, tên danh mục, số lượng nhãn, ImageView hiển thị icon danh mục và nhãn.

***Hình dung màn hình***

****

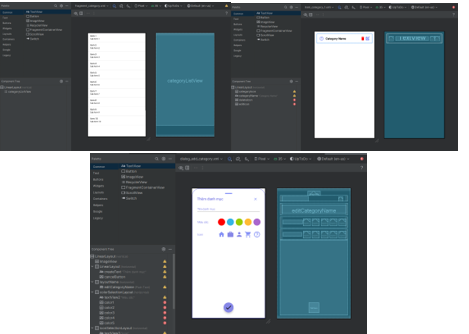
*Hình 2.8: Hình dung màn hình quản lý danh mục*

***Nghiệp vụ***

Người dùng có thể

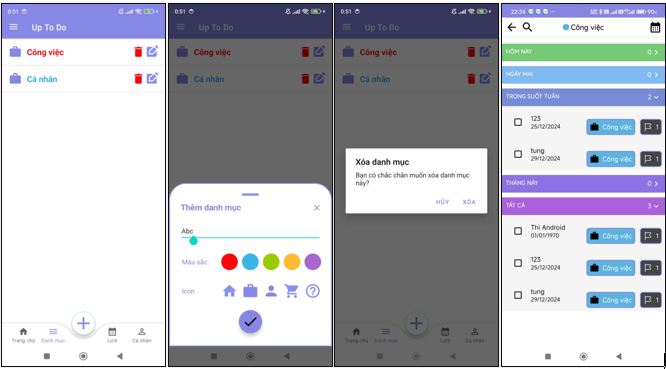
* Quay về giao diện chính bằng cách nhấn vào nút trở về.
* Xem danh sách các danh mục được hiển thị dưới dạng các mục trong ListView.
* Thực hiện việc thêm mới một danh mục bằng cách bấm vào nút thêm mới, màn hình xuất hiện một hộp thoại để nhập thông tin cho danh mục mới.
* Sửa một danh mục bằng cách bấm vào nút sửa ở cuối dòng danh mục, màn hình xuất hiện một hộp thoại để nhập thông tin mới cho danh mục.
* Xóa một danh mục bằng cách bấm vào nút xóa ở cuối dòng danh mục, màn hình sẽ hiển thị hộp thoại xác nhận xóa.
* Xem danh sách các nhiệm vụ theo danh mục bằng cách bấm vào 1 dòng danh mục.

***Màn hình android***

******

*Hình 2.9: Màn hình android quản lý danh mục*

***Giao diện khi chạy chương trình***



*Hình 2.10: Giao diện màn hình quản lý danh mục*

***Các phương thức đã sử dụng***

*Bảng 2.10: Phương thức addCategory()*

| private void addCategory(String categoryName, String selectedIcon, String selectedColor, Dialog dialog) {  if (categoryName.isEmpty()) {  Toast.*makeText*(requireContext(), "Vui lòng nhập tên danh mục", Toast.*LENGTH\_SHORT*).show();  return;  }  Category category = new Category(0, categoryName, selectedIcon, selectedColor, 0);  boolean isInserted = categoryDAO.insertCategory(category);  if (isInserted) {  Toast.*makeText*(requireContext(), "Danh mục đã được thêm", Toast.*LENGTH\_SHORT*).show();  categoryList.clear();  categoryList.addAll(categoryDAO.getAllCategories());  adapter.notifyDataSetChanged(); // Thông báo adapter cập nhật danh sách  } else {  Toast.*makeText*(requireContext(), "Lỗi khi thêm danh mục", Toast.*LENGTH\_SHORT*).show();  }  dialog.dismiss(); } |
| --- |

*Bảng 2.11: Phương thức updateCategory()*

| private void updateCategory(String categoryName, String selectedIcon, String selectedColor, Category category, int position, Dialog dialog) {  if (categoryName.isEmpty()) {  Toast.*makeText*(context, "Vui lòng nhập tên danh mục", Toast.*LENGTH\_SHORT*).show();  return;  }  category.setName(categoryName);  category.setColor(selectedColor);  category.setIcon(selectedIcon);  int result = categoryDAO.updateCategory(category);  if (result > 0) {  Toast.*makeText*(context, "Danh mục đã được cập nhật", Toast.*LENGTH\_SHORT*).show();  categoryList.set(position, category);  notifyDataSetChanged();  } else {  Toast.*makeText*(context, "Cập nhật không thành công", Toast.*LENGTH\_SHORT*).show();  }  dialog.dismiss(); } |
| --- |

*Bảng 2.12: Phương thức removeCategory()*

| private void removeCategory(int position) {  new android.app.AlertDialog.Builder(context)  .setTitle("Xóa danh mục")  .setMessage("Bạn có chắc chắn muốn xóa danh mục này?")  .setPositiveButton("Xóa", (dialog, which) -> {  Category category = categoryList.get(position);  int result = categoryDAO.deleteCategory(category.getCategoryId());  if (result > 0) {  categoryList.remove(position);  notifyDataSetChanged();  Toast.*makeText*(context, "Danh mục đã được xóa", Toast.*LENGTH\_SHORT*).show();  } else {  Toast.*makeText*(context, "Xóa không thành công", Toast.*LENGTH\_SHORT*).show();  }  })  .setNegativeButton("Hủy", (dialog, which) -> {  dialog.dismiss();  })  .show(); } |
| --- |

### Xem màn hình công việc theo thời gian

***Mô tả***

* Button next và previous để điều chỉnh thời gian xem công việc.
* Các TextView để hiển thị thời gian
* RecyclerView để hiện thị ngày trong tuần, bao gồm:
* TextView để hiện thị từng ngày trong tuần.
* ListView để hiển thị danh sách công việc, bao gồm
* CheckBox để hiện thị trạng thái của công việc.
* TextView để hiên thị thông tin công việc.
* ImageView để hiển thị icon của danh mục, nhãn.
* DatePicker để chọn ngày xem công việc.

***Hình dung màn hình***

******

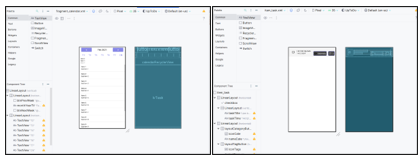
*Hình 2.11: Hình dung màn hình xem công việc theo thời gian*

***Nghiệp vụ***

Người dùng có thể

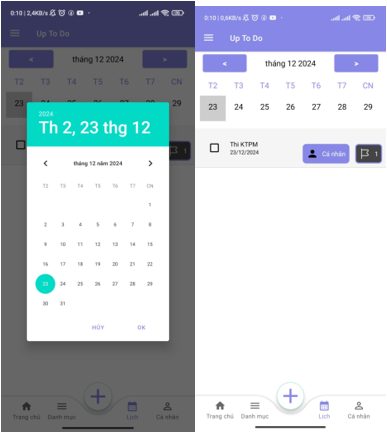
* Chọn từng ngày trong tuần để xem danh sách công việc đó.
* Chọn ngày xem công việc bằng DatePicker.
* Click các nút Next, Previous để điều chỉnh thời gian xem công việc.
* Mở ra màn hình xem chi tiết công việc , điều chỉnh trạng thái, chỉnh sửa công việc khi click vào từng thành phần của ListView.

***Màn hình android***

******

*Hình 2.12: Màn hình android xem công việc theo thời gian*

***Giao diện khi chạy chương trình***

******

*Hình 2.13: Giao diện xem màn hình công việc theo thời gian*

***Các phương thức đã sử dụng***

*Bảng 2.13: Code lấy tính toán theo ngày*

| private void setWeekView() { monthYearText.setText(*monthYearFromDate*(CalendarUtils.*selectedDate*));  ArrayList<LocalDate> days = *daysInWeekArray*(CalendarUtils.*selectedDate*);  CalendarAdapter calendarAdapter = new CalendarAdapter(days, this);  RecyclerView.LayoutManager layoutManager = new GridLayoutManager(getContext(), 7);  calendarRecyclerView.setLayoutManager(layoutManager);  calendarRecyclerView.setAdapter(calendarAdapter); } |
| --- |

*Bảng 2.14: Code xử lý điều chỉnh thời gian*

| public void previousWeekAction(View view) {  CalendarUtils.*selectedDate* = CalendarUtils.*selectedDate*.minusWeeks(1);  setWeekView();  Date selectedDate = java.sql.Date.*valueOf*(CalendarUtils.*selectedDate*.toString());  setTasksForSelectedDate(selectedDate); } public void nextWeekAction(View view) {  CalendarUtils.*selectedDate* = CalendarUtils.*selectedDate*.plusWeeks(1);  setWeekView();  Date selectedDate = java.sql.Date.*valueOf*(CalendarUtils.*selectedDate*.toString());  setTasksForSelectedDate(selectedDate); } |
| --- |

*Bảng 2.15: Code lọc ra công việc theo từng ngày*

| private void setTasksForSelectedDate(Date selectedDate) {  List<Task> tasks = taskDAO.findAll().stream()  .filter(task -> {  Date taskDueDate = task.getDueDate();  if (taskDueDate != null) {  Calendar taskCalendar = Calendar.*getInstance*();  taskCalendar.setTime(taskDueDate);  taskCalendar.set(Calendar.*HOUR\_OF\_DAY*, 0);  taskCalendar.set(Calendar.*MINUTE*, 0);  taskCalendar.set(Calendar.*SECOND*, 0);  taskCalendar.set(Calendar.*MILLISECOND*, 0);  Calendar selectedCalendar = Calendar.*getInstance*();  selectedCalendar.setTime(selectedDate);  selectedCalendar.set(Calendar.*HOUR\_OF\_DAY*, 0);  selectedCalendar.set(Calendar.*MINUTE*, 0);  selectedCalendar.set(Calendar.*SECOND*, 0);  selectedCalendar.set(Calendar.*MILLISECOND*, 0);  return taskCalendar.getTime().equals(selectedCalendar.getTime());  }  return false;  })  .collect(Collectors.*toList*());  TaskArrayAdapter arrayAdapter = new TaskArrayAdapter(getActivity(), R.layout.*item\_task*, tasks);  lvTask.setAdapter(arrayAdapter); } |
| --- |

### Quản lý danh sách nhiệm vụ

#### Xem danh sách nhiệm vụ

***Mô tả***

* ImageView quản lý nhãn và tìm kiếm nhiệm vụ. RecyclerView hiển thị danh sách các nhóm nhiệm vụ
* Nhóm nhiệm vụ bao gồm: TextView hiển thị tên nhóm và số lượng nhiệm vụ bên trong. RecyclerView hiển thị danh sách nhiệm vụ
* Nhiệm vụ bao gồm: CheckBox hiển thị trạng thái và cập nhật trạng thái khi người dùng ấn, TextView hiển thị tiêu đề, ngày hết hạn, tên danh mục, số lượng nhãn, ImageView hiển thị icon danh mục và nhãn.

***Hình dung màn hình***

A screen shot of a computer

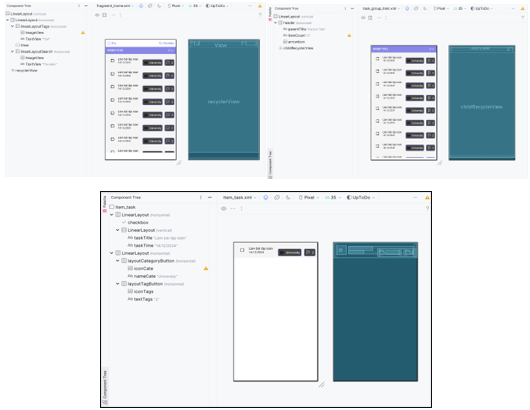
Description automatically generated

*Hình 2.14: Hình dung màn hình xem danh sách nhiệm vụ*

***Nghiệp vụ***

* Hiển thị danh sách nhiệm vụ: Người dùng có thể xem danh sách các nhiệm vụ hiện tại, bao gồm thông tin như tiêu đề, ngày hết hạn, nhãn liên quan, danh mục, trạng thái của nhiệm vụ.
* Đánh dấu trạng thái hoàn thành: Người dùng có thể đánh dấu nhiệm vụ là hoàn thành hoặc chưa hoàn thành bằng cách thay đổi trạng thái của checkbox.
* Tìm kiếm nhiệm vụ: Gọi đến Activity SearchActivity để thực hiện tìm kiếm nhiệm vụ.

***Màn hình android***



*Hình 2.15: Màn hình android xem danh sách nhiệm vụ*

***Giao diện khi chạy chương trình***

A screenshot of a phone

Description automatically generated

*Hình 2.16: Màn hình xem danh sách nhiệm vụ*

***Các phương thức đã sử dụng***

*Bảng 2.16: Phương thức lọc nhiệm vụ theo nhóm*

| //Code lọc nhiệm vụ theo các nhóm  private List<TaskGroup> getTaskGroups() {  List<TaskGroup> parentList = new ArrayList<>();  List<Task> allTasks = taskDAO.findAll();  List<Task> overdueTasks = new ArrayList<>();  List<Task> noDeadlineTasks = new ArrayList<>();  List<Task> completedTasks = new ArrayList<>();  List<Task> todayTasks = new ArrayList<>();  List<Task> tomorrowTasks = new ArrayList<>();  List<Task> thisWeekTasks = new ArrayList<>();  List<Task> thisMonthTasks = new ArrayList<>();  LocalDate today = LocalDate.*now*();  LocalDate tomorrow = today.plusDays(1);  LocalDate weekStart = today.with(ChronoField.*DAY\_OF\_WEEK*, 1);  LocalDate weekEnd = weekStart.plusDays(6);  LocalDate monthEnd = today.withDayOfMonth(today.lengthOfMonth());   for (Task task : allTasks) {  if (task.isCompleted()) {  completedTasks.add(task);  continue;  }  if (task.getDueDate() == null) {  noDeadlineTasks.add(task);  continue;  }  LocalDate taskDueDate = Utils.*convertToLocalDate*(task.getDueDate());  if (taskDueDate.isBefore(today)) {  overdueTasks.add(task);  } else if (taskDueDate.isEqual(today)) {  todayTasks.add(task);  } else if (taskDueDate.isEqual(tomorrow)) {  tomorrowTasks.add(task);  } else if (!taskDueDate.isBefore(today) && taskDueDate.isBefore(weekEnd.plusDays(1))) {  thisWeekTasks.add(task);  } else if (!taskDueDate.isBefore(weekEnd.plusDays(1)) && taskDueDate.isBefore(monthEnd.plusDays(1))) {  thisMonthTasks.add(task);  }  }  parentList.add(createTaskGroup("Quá hạn", R.color.*overdue\_color*, R.color.*white*, overdueTasks));  parentList.add(createTaskGroup("Không có hạn", R.color.*no\_deadline\_color*, R.color.*white*, noDeadlineTasks));  parentList.add(createTaskGroup("Hôm nay", R.color.*today\_color*, R.color.*white*, todayTasks));  parentList.add(createTaskGroup("Đã hoàn thành", R.color.*completed\_color*, R.color.*black*, completedTasks));  parentList.add(createTaskGroup("Ngày mai", R.color.*tomorrow\_color*, R.color.*white*, tomorrowTasks));  parentList.add(createTaskGroup("Trong suốt tuần", R.color.*this\_week\_color*, R.color.*white*, thisWeekTasks));  parentList.add(createTaskGroup("Tháng này", R.color.*this\_month\_color*, R.color.*white*, thisMonthTasks));  parentList.add(createTaskGroup("Tất cả", R.color.*all\_tasks\_color*, R.color.*white*, allTasks));  return parentList; } private TaskGroup createTaskGroup(String title, int bgColorRes, int textColorRes, List<Task> tasks) {  Context context = requireContext();  return new TaskGroup(  title,  ContextCompat.*getColor*(context, bgColorRes),  ContextCompat.*getColor*(context, textColorRes),  tasks  ); } |
| --- |

#### Thêm mới nhiệm vụ

***Mô tả***

* TextView thêm nhiệm vụ và ImageView quay lại. EditText nhập tiêu đề nhiệm vụ. TextView chọn danh mục và hiển thị danh mục được chọn. TextView chọn các nhãn và hiển thị danh sách nhãn được chọn. TextView chọn ngày hoàn thành nhiệm vụ.
* TextView chọn thời gian hoàn thành nhiệm vụ. TextView chọn ngày và thời gian nhắc nhở. EditText nhập ghi chú nhiệm vụ. FloatingActionButton thêm mới.

***Hình dung màn hình***

A drawing of a cell phone

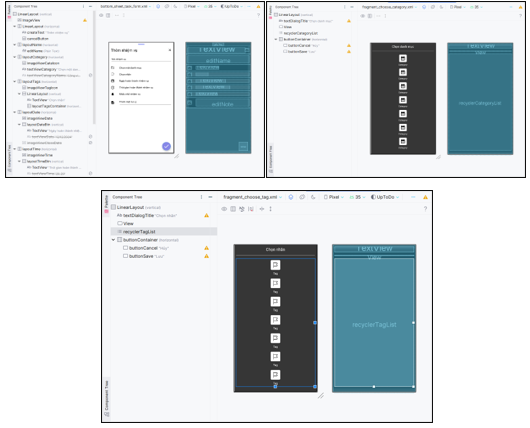
Description automatically generated

*Hình 2.17: Hình dung màn hình thêm nhiệm vụ mới*

***Nghiệp vụ***

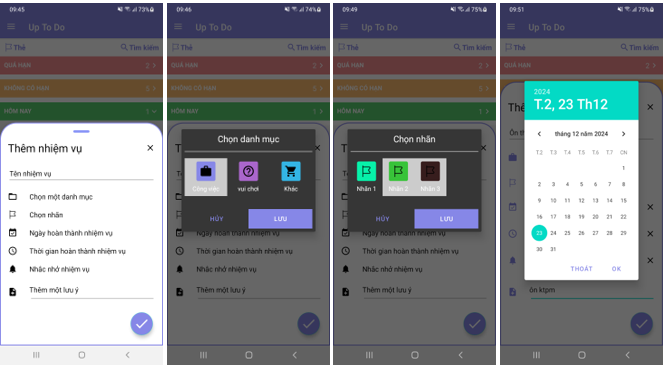
* Người dùng có thể thêm nhiệm vụ mới với các thông tin như tiêu đề, ghi chú, ngày hoàn thành, thời gian nhắc nhở, danh mục, nhãn
* Quay lại màn hình trước đó: Người dùng có thể sử dụng nút ImageView quay lại để hủy bỏ thao tác thêm mới và trở về màn hình trước.

***Màn hình android***

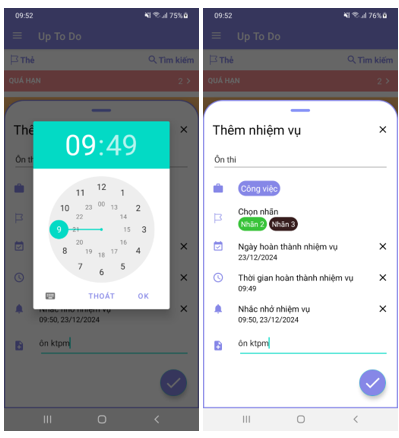
******

*Hình 2.18: Màn hình android thêm mới nhiệm vụ*

***Giao diện khi chạy chương trình***



*Hình 2.19: Giao diện màn hình thêm mới nhiệm vụ (1)*



*Hình 2.20: Giao diện màn hình thêm mới nhiệm vụ (2)*

***Phương thức đã sử dụng***

*Bảng 2.17: Phương thức lưu nhiệm vụ*

| //Code lưu nhiệm vụ  fabSave.setOnClickListener(v -> {  String title = editTextName.getText().toString();  String note = editTextNote.getText().toString();   if (title.isEmpty()) {  editTextName.setError("Vui lòng nhập tiêu đề");  return;  }  Calendar date = selectedDate[0];  Calendar time = selectedTime[0];  Date dueDate = Utils.*combineDateAndTime*(date, time);   Task task = new Task();  task.setTitle(title);  task.setNote(note);  task.setDueDate(dueDate);  task.setCategoryId(categoryId);  long taskId = taskDAO.save(task);  if (taskId != -1) {  task.setTaskId((int) taskId);  taskTagsDAO.create(task.getTaskId(), tagIds.stream().mapToInt(Integer::intValue).toArray());  *//Cài đặt thông báo* if (timeToRemind != -1 && timeToRemind > System.*currentTimeMillis*()) {  taskDAO.updateTaskReminderTime(taskId, new Date(timeToRemind));  ReminderUtils.*setReminder*(requireContext(), timeToRemind, (int) taskId, "Nhắc nhở nhiệm vụ");  }   *//Gửi yêu cầu cập nhật dữ liệu* Bundle result = new Bundle();  result.putBoolean("isTaskAdded", true);  getParentFragmentManager().setFragmentResult("taskAdded", result);  Toast.*makeText*(requireContext(), "Nhiệm vụ đã được lưu", Toast.*LENGTH\_SHORT*).show();  } else {  Toast.*makeText*(requireContext(), "Lỗi khi lưu nhiêm vụ", Toast.*LENGTH\_SHORT*).show();  }  dismiss(); }); |
| --- |

*Bảng 2.18: Phương thức đặt nhắc nhở*

| *// Đặt nhắc nhở* public static void setReminder(Context context, long reminderTimeMillis, int taskId, String title) {  Intent intent = new Intent(context, ReminderReceiver.class);  intent.putExtra("title", title);  intent.putExtra("taskId", taskId);  intent.putExtra("isCancel", false);  PendingIntent pendingIntent = PendingIntent.*getBroadcast*(  context,  taskId,  intent,  PendingIntent.*FLAG\_UPDATE\_CURRENT* | PendingIntent.*FLAG\_IMMUTABLE* );  AlarmManager alarmManager = ContextCompat.*getSystemService*(context, AlarmManager.class);  if (alarmManager != null) {  alarmManager.set(AlarmManager.*RTC\_WAKEUP*, reminderTimeMillis, pendingIntent);  } }  //Phần xử lý nhắc nhở trong ReminderReceiver.class  @Override public void onReceive(Context context, Intent intent) {  String title = intent.getStringExtra("title");  int taskId = intent.getIntExtra("taskId", 1);  boolean isCancel = intent.getBooleanExtra("isCancel", false);  if (isCancel) {  cancelNotification(context, taskId);  } else {  TaskDAO taskDAO = new TaskDAO(context);  Task task = taskDAO.findById(taskId);  if (task == null || task.isCompleted()) {  return;  }  if (title == null) title = "Nhắc nhở nhiệm vụ";  String message = task.getTitle();  showNotification(context, taskId, title, message);  } }  private void showNotification(Context context, int taskId, String title, String message) {  *// Tạo Intent để mở MainActivity khi người dùng nhấp vào thông báo* Intent intent = new Intent(context, MainActivity.class);  intent.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK* | Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*);  *// Tạo PendingIntent để hoạt động khi nhấp vào thông báo* PendingIntent pendingIntent = PendingIntent.*getActivity*(  context,  0,  intent,  PendingIntent.*FLAG\_UPDATE\_CURRENT* | PendingIntent.*FLAG\_IMMUTABLE* );  *// Tạo thông báo* NotificationCompat.Builder builder = new NotificationCompat.Builder(context, "REMINDER\_CHANNEL")  .setSmallIcon(R.drawable.*baseline\_notifications\_24*)  .setContentTitle(title)  .setContentText(message)  .setPriority(NotificationCompat.*PRIORITY\_HIGH*)  .setContentIntent(pendingIntent)  .setAutoCancel(true);  *// Hiển thị thông báo* NotificationManagerCompat notificationManager = NotificationManagerCompat.*from*(context);  notificationManager.notify(taskId, builder.build()); } |
| --- |

#### Xem chi tiết nhiệm vụ

***Mô tả***

ImageButton quay về màn hình trước đó. Checkbox hiển thị và thay đổi trạng thái. TextView hiển thị tiêu đề và ghi chú. ImageView chỉnh sửa tiêu đề và ghi chú. Các TextView hiển thị và chọn: thời gian, danh mục, nhãn và nhắc nhở. TextView xóa nhiệm vụ.

***Hình dung màn hình***

A screenshot of a cell phone

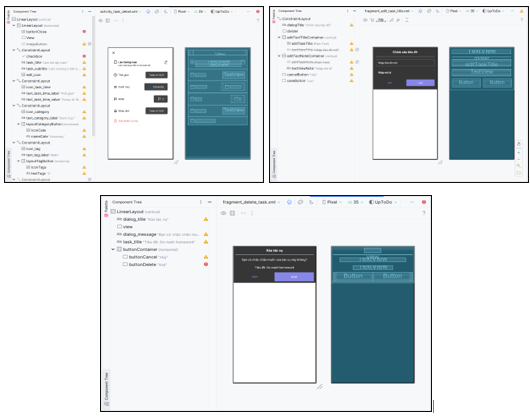
Description automatically generated

*Hình 2.21: Hình dung màn hình xem chi tiết nhiệm vụ*

***Nghiệp vụ***

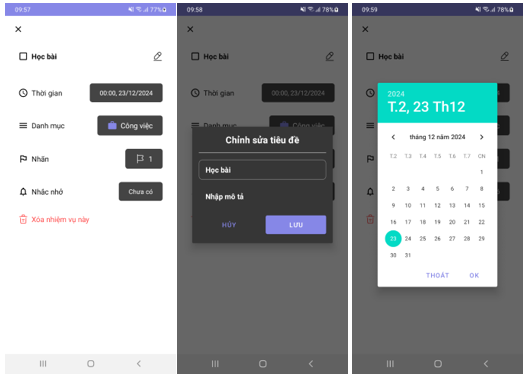
* Hiển thị thông tin nhiệm vụ: Màn hình hiển thị đầy đủ thông tin chi tiết của nhiệm vụ, bao gồm tiêu đề, ghi chú, danh mục, số lượng thẻ, thời hạn, thời gian nhắc nhở, và trạng thái hoàn thành.
* Chỉnh sửa tiêu đề và ghi chú nhiệm vụ: Người dùng có thể chỉnh sửa tiêu đề và ghi chú của nhiệm vụ thông qua hộp thoại EditTaskTitleFragment.
* Thay đổi danh mục nhiệm vụ: Người dùng có thể thay đổi danh mục của nhiệm vụ thông qua hộp thoại ChooseCategoryFragment.
* Quản lý thẻ của nhiệm vụ: Hệ thống cho phép người dùng thêm hoặc xóa thẻ liên quan đến nhiệm vụ thông qua hộp thoại ChooseTagFragment.
* Đánh dấu trạng thái hoàn thành: Người dùng có thể đánh dấu nhiệm vụ là hoàn thành hoặc chưa hoàn thành bằng cách thay đổi trạng thái của checkbox.
* Cập nhật thời hạn nhiệm vụ: Người dùng có thể thêm, thay đổi, hoặc xóa thời hạn của nhiệm vụ thông qua giao diện DatePicker và TimePicker.
* Cập nhật thời gian nhắc nhở: Tương tự thời hạn, người dùng có thể thay đổi hoặc xóa thời gian nhắc nhở nhiệm vụ thông qua DatePicker và TimePicker.
* Xóa nhiệm vụ: Người dùng có thể xóa nhiệm vụ thông qua hộp thoại xác nhận DeleteTaskFragment.
* Đóng màn hình chi tiết: Người dùng có thể quay lại màn hình trước đó bằng cách nhấn vào nút đóng.

***Màn hình android***

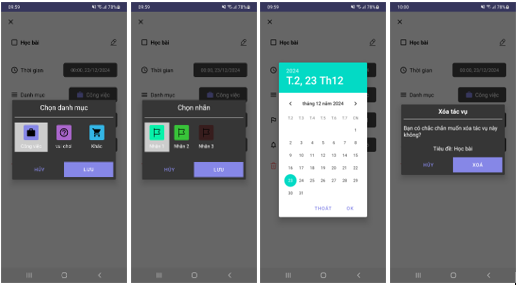
******

*Hình 2.22 Màn hình android xem chi tiết nhiệm vụ*

***Giao diện khi chạy chương trình***

******

*Hình 2.23: Giao diện màn hình xem chi tiết nhiệm vụ*

******

*Hình 2.24: Giao diện màn hình xem chi tiết nhiệm vụ (2)*

***Phương thức đã sử dụng***

*Bảng 2.19: Phương thức mở dialog và lắng nghe kết quả trả về*

| //Code mở dialog và lắng nghe kết quả trả về  getSupportFragmentManager().setFragmentResultListener("UPDATED\_TITLE", this, this::handleUpdatedTitle);  getSupportFragmentManager().setFragmentResultListener("UPDATED\_CATEGORY", this, this::handleUpdatedCategory); getSupportFragmentManager().setFragmentResultListener("UPDATED\_TAG", this, this::handleUpdatedTag); getSupportFragmentManager().setFragmentResultListener("DELETED\_TASK", this, this::handleDeletedTask); imageViewEditTitleTask.setOnClickListener(view -> {  EditTaskTitleFragment dialogFragment = EditTaskTitleFragment.*newInstance*(currentTask.getTitle(), currentTask.getNote());  dialogFragment.show(getSupportFragmentManager(), "EditTaskDialogFragment"); }); layoutCategoryButton.setOnClickListener(view -> {  ChooseCategoryFragment dialogFragment = ChooseCategoryFragment.*newInstance*(currentTask.getCategoryId());  dialogFragment.show(getSupportFragmentManager(), "ChooseCategoryFragment"); }); layoutTagButton.setOnClickListener(view -> {  ChooseTagFragment dialogFragment = ChooseTagFragment.*newInstance*(currentTask.getTaskId());  dialogFragment.show(getSupportFragmentManager(), "ChooseTagFragment"); }); textViewDelete.setOnClickListener(view -> {  DeleteTaskFragment dialogFragment = DeleteTaskFragment.*newInstance*(currentTask.getTitle());  dialogFragment.show(getSupportFragmentManager(), "DeleteTaskFragment"); });  //Cập nhật trạng thái nhiệm vụ  checkBox.setOnCheckedChangeListener((buttonView, isChecked) -> {  taskDAO.updateTaskStatus(currentTask.getTaskId(), isChecked);  currentTask.setCompleted(isChecked); }); |
| --- |

*Bảng 2.20: Phương thức cập nhật danh mục*

| //Cập nhật danh mục  private void handleUpdatedCategory(String s, Bundle bundle) {  int newCategoryId = bundle.getInt("updatedCategoryId");  long isUpdate = taskDAO.updateTaskCategory(currentTask.getTaskId(), newCategoryId);  if (isUpdate > 0) {  currentTask.setCategoryId(newCategoryId);  updateViewCategory();  Toast.*makeText*(this, "Cập nhật thành công", Toast.*LENGTH\_SHORT*).show();  } else {  Toast.*makeText*(this, "Lỗi khi cập nhật", Toast.*LENGTH\_SHORT*).show();  } }  //Cập nhật nhãn private void handleUpdatedTag(String s, Bundle bundle) {  int[] newTagIds = bundle.getIntArray("updatedTagIds");  taskTagsDAO.deleteByTaskId(currentTask.getTaskId());  if (newTagIds != null && newTagIds.length > 0) {  taskTagsDAO.create(currentTask.getTaskId(), newTagIds);  }  updateViewTags(); }  //Xóa nhiệm vụ private void handleDeletedTask(String s, Bundle bundle) {  long isDeleted = taskDAO.delete(currentTask.getTaskId());  if (isDeleted > 0) {  Toast.*makeText*(this, "Đã xóa nhiệm vụ", Toast.*LENGTH\_SHORT*).show();  finish();  } else {  Toast.*makeText*(this, "Lỗi khi xóa nhiệm vụ", Toast.*LENGTH\_SHORT*).show();  } } |
| --- |

#### Tìm kiếm nhiệm vụ

***Mô tả***

ImageView quản lý nút quay lại, EditText để người dùng nhập từ khóa tìm kiếm, RecyclerView hiển thị danh sách các nhiệm vụ sau khi đã lọc theo từ khóa và các thẻ đã chọn. LinearLayout chứa các thẻ đã chọn, TextView hiển thị thông báo khi không có kết quả tìm kiếm.

***Hình dung màn hình***

A screen shot of a cell phone

Description automatically generated

*Hình 2.25: Hình dung màn hình tìm kiếm nhiệm vụ*

***Nghiệp vụ***

* Hiển thị danh sách nhiệm vụ: Người dùng có thể tìm kiếm nhiệm vụ thông qua từ khóa nhập vào trong EditText và các thẻ đã chọn. Các nhiệm vụ sẽ được lọc theo tiêu đề và thẻ. Hệ thống sẽ tự động cập nhật danh sách nhiệm vụ trong RecyclerView sau mỗi lần thay đổi tìm kiếm.
* Đóng màn hình tìm kiếm: Người dùng có thể quay lại màn hình trước thông qua nút quay lại.

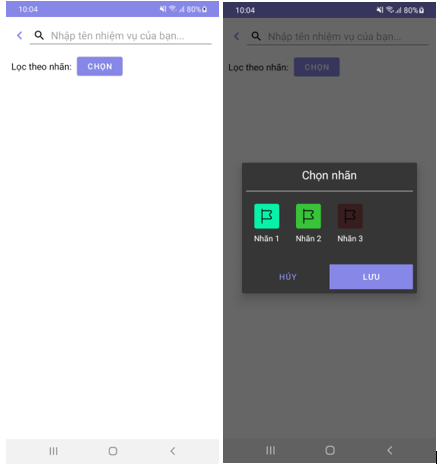
***Màn hình android***

A screenshot of a computer

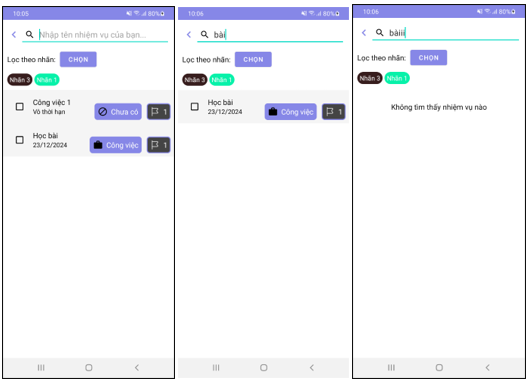
Description automatically generated

*Hình 2.26: Màn hình android tìm kiếm nhiệm vụ*

***Giao diện khi chạy chương trình***

******

*Hình 2.27: Giao diện màn hình tìm kiếm nhiệm vụ (1)*



*Hình 2.28: Giao diện màn hình tìm kiếm nhiệm vụ (2)*

***Phương thức đã sử dụng***

*Bảng 2.21: Phương thức lắng nghe thay đổi và thực hiện tìm kiếm*

| //Lắng nghe sự thay đổi của input và thực hiện tìm kiếm  searchBar.addTextChangedListener(new TextWatcher() {  @Override  public void beforeTextChanged(CharSequence s, int start, int count, int after) {  }  @Override  public void onTextChanged(CharSequence s, int start, int before, int count) {  *// Hủy tìm kiếm trước đó* if (searchRunnable != null) {  searchHandler.removeCallbacks(searchRunnable);  }  *// Lập lịch tìm kiếm sau 300ms* searchRunnable = () -> filterTasks();  searchHandler.postDelayed(searchRunnable, 300);  }  @Override  public void afterTextChanged(Editable s) {  } }); |
| --- |

*Bảng 2.22: Phương thức lọc nhiệm vụ theo từ khóa và danh sách nhãn*

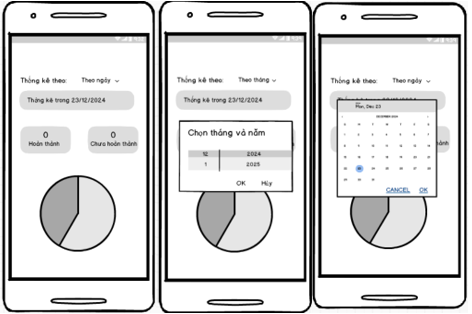
| //Hàm lọc nhiệm vụ theo từ khóa và danh sách nhãn  public List<Task> searchTask(String keyword, ArrayList<Integer> selectedTagIds) {  List<Task> tasks = new ArrayList<>();  StringBuilder sqlQuery = new StringBuilder("SELECT DISTINCT t.\* FROM tasks t ");  sqlQuery.append("LEFT JOIN tasks\_tags tt ON t.task\_id = tt.task\_id ");  sqlQuery.append("LEFT JOIN tags tg ON tt.tag\_id = tg.tag\_id WHERE ");  sqlQuery.append("(LOWER(t.title) LIKE ? OR LOWER(t.note) LIKE ?) ");  if (!selectedTagIds.isEmpty()) {  sqlQuery.append("AND tg.tag\_id IN (");  for (int i = 0; i < selectedTagIds.size(); i++) {  sqlQuery.append("?");  if (i < selectedTagIds.size() - 1) {  sqlQuery.append(", ");  }  }  sqlQuery.append(") ");  }  List<String> selectionArgs = new ArrayList<>();  selectionArgs.add("%" + keyword.toLowerCase() + "%");  selectionArgs.add("%" + keyword.toLowerCase() + "%");  if (!selectedTagIds.isEmpty()) {  for (Integer tagId : selectedTagIds) {  selectionArgs.add(String.*valueOf*(tagId));  }  }  Cursor cursor = db.rawQuery(sqlQuery.toString(), selectionArgs.toArray(new String[0]));  if (cursor != null && cursor.moveToFirst()) {  do {  Task task = parseTask(cursor);  tasks.add(task);  } while (cursor.moveToNext());  cursor.close();  }  return tasks; }  private Task parseTask(Cursor cursor) {  Task task = new Task(); task.setTaskId(cursor.getInt(cursor.getColumnIndexOrThrow("task\_id"))); task.setTitle(cursor.getString(cursor.getColumnIndexOrThrow("title"))); task.setNote(cursor.getString(cursor.getColumnIndexOrThrow("note")));  int dueDateIndex = cursor.getColumnIndexOrThrow("due\_date");  if (!cursor.isNull(dueDateIndex)) {  task.setDueDate(new Date(cursor.getLong(dueDateIndex)));  } else {  task.setDueDate(null);  }  int reminderTimeIndex = cursor.getColumnIndexOrThrow("reminder\_time");  if (!cursor.isNull(reminderTimeIndex)) {  task.setReminderTime(new Date(cursor.getLong(reminderTimeIndex)));  } else {  task.setReminderTime(null);  }  int isCompleted = cursor.getInt(cursor.getColumnIndexOrThrow("is\_completed"));  task.setCompleted(isCompleted == 1); task.setUserId(cursor.getInt(cursor.getColumnIndexOrThrow("user\_id"))); task.setCategoryId(cursor.getInt(cursor.getColumnIndexOrThrow("category\_id")));  return task; } |
| --- |

### Thống Kê

Thống kê

***Mô tả***

* TextView hiển thị thông tin thống kê.
* Spinner để chọn thời gian lọc.
* DatePicker để chọn thời gian lọc.
* MothPicker để chọn tháng lọc.
* PieChart để hiển thị tỉ lệ hoàn thành, chưa hoàn thành.

***Hình dung màn hình***

*Hình 2.29: Hình dung màn hình thống kê*

***Nghiệp vụ***

Người dùng có thể

* Chọn tiêu chuẩn lọc bằng Spinner.
* Chọn thời gian lọc bằng DatePicker, MothPicker.

***Màn hình android***

***:***A screenshot of a computer

Description automatically generated

*Hình 2.30: Màn hình android thống kê*

***Giao diện khi chạy chương trình***



*Hình 2.31: Giao diện màn hình thống kê*

***Các phương thức đã sử dụng***

*Bảng 2.23 Phương thức xử lý hiển thị biểu đồ*

| public void getPieChart(int doneValue,int unfinishValue){  ArrayList<PieEntry> entries = new ArrayList<>();  entries.add(new PieEntry(doneValue, "Hoàn thành"));  entries.add(new PieEntry(unfinishValue, "Chưa hoàn thành"));  *// Create and customize the pie data set* PieDataSet pieDataSet = new PieDataSet(entries, null);  pieDataSet.setColors(ColorTemplate.*MATERIAL\_COLORS*);  pieDataSet.setValueTextSize(16f);  pieDataSet.setValueTextColor(android.graphics.Color.*WHITE*);  pieDataSet.setSliceSpace(3f);  pieDataSet.setSelectionShift(5f);  *// Create and set the pie data* PieData pieData = new PieData(pieDataSet);  pieData.setValueFormatter(new PercentFormatter(chart));  pieData.setValueTextSize(12f);  *// Customize the pie chart* chart.clear();  chart.setData(pieData);  chart.setUsePercentValues(true);  chart.getDescription().setEnabled(false);  chart.setDrawHoleEnabled(true);  chart.setHoleColor(android.graphics.Color.*TRANSPARENT*);  chart.setHoleRadius(58f);  chart.setTransparentCircleRadius(61f);  chart.setDrawCenterText(true);  chart.setCenterTextSize(18f);  chart.setRotationAngle(0);  chart.setRotationEnabled(true);  chart.setHighlightPerTapEnabled(true);  *// Disable the legend* Legend legend = chart.getLegend();  legend.setEnabled(false);  chart.animateY(1400, Easing.*EaseInOutQuad*);  chart.notifyDataSetChanged();  chart.invalidate(); } |
| --- |

*Bảng 2.24 Phương thức xử lý lọc công việc theo ngày, tháng*

| public void getStatisticByDay(Date filterDate) {  int inComp = 1, comp = 1;  Calendar filterCalendar = Calendar.*getInstance*();  filterCalendar.setTime(filterDate);  for (Task task : tasks) {  Calendar taskCalendar = Calendar.*getInstance*();  taskCalendar.setTime(task.getDueDate());  boolean sameDay = taskCalendar.get(Calendar.*YEAR*) == filterCalendar.get(Calendar.*YEAR*) &&  taskCalendar.get(Calendar.*DAY\_OF\_YEAR*) == filterCalendar.get(Calendar.*DAY\_OF\_YEAR*);  if (sameDay) {  if (task.isCompleted()) {  comp++;  } else {  inComp++;  }  } }  txtComp.setText(String.*valueOf*(comp - 1));  txtInComp.setText(String.*valueOf*(inComp - 1));  if(inComp ==1 && comp ==1){  getPieChart(comp, inComp);  }  else {  getPieChart(comp-1, inComp-1);  } } public void getStatisticByMonth(Date filterDate) {  int inComp = 0, comp = 0;  Calendar filterCalendar = Calendar.*getInstance*();  filterCalendar.setTime(filterDate);  for (Task task : tasks) {  Calendar taskCalendar = Calendar.*getInstance*();  taskCalendar.setTime(task.getDueDate());  boolean sameMonth = taskCalendar.get(Calendar.*YEAR*) == filterCalendar.get(Calendar.*YEAR*) &&  taskCalendar.get(Calendar.*MONTH*) == filterCalendar.get(Calendar.*MONTH*);  if (sameMonth) {  if (task.isCompleted()) {  comp++;  } else {  inComp++;  }  }  }  txtComp.setText(String.*valueOf*(comp));  txtInComp.setText(String.*valueOf*(inComp));  getPieChart(comp, inComp);  } |
| --- |

### Đăng nhập và đăng kí

#### 2.3.6.1. Đăng nhập

**Mô tả**

* ImageView banner đăng nhập ở phía trên
* TextView hiển thị tiêu đề "ĐĂNG NHẬP"
* EditText nhập tên đăng nhập, có icon người dùng bên trái
* EditText nhập mật khẩu, có icon khóa bên trái và icon ẩn/hiện mật khẩu bên phải
* Button "Đăng nhập" màu tím lavender (#7F5AF0)

**Nghiệp vụ:**

* Người dùng nhập thông tin đăng nhập (email/tên đăng nhập và mật khẩu)
* Người dùng có thể ẩn/hiện mật khẩu bằng cách nhấn vào icon con mắt
* Khi người dùng nhấn nút đăng nhập:
* Hệ thống kiểm tra thông tin đăng nhập
* Nếu thông tin chính xác, cho phép người dùng truy cập vào ứng dụng
* Nếu thông tin không chính xác, hiển thị thông báo lỗi
* Người dùng có thể chuyển sang màn hình đăng ký nếu chưa có tài khoản
* Người dùng có thể chọn "Quên mật khẩu" để khôi phục mật khẩu thông qua email

**Hình dung màn hình**

****

*Hình 2.32: Hình dung màn hình android quản lý đăng nhập*

A screenshot of a screenshot of a screenshot of a screenshot of a screenshot of a screenshot of a screenshot of a screenshot of a screenshot of a screenshot

Description automatically generated

*Hình 2.33: Màn hình android quản lý đăng nhập*

A screenshot of a phone

Description automatically generated

*Hình 2.34: Giao diện android quản lý đăng nhập*

***Các phương thức đã sử dụng***

*Bảng 2.25 Phương thức handleLogin()*

| private void handleLogin() {  // Lấy thông tin từ các EditText  String username = usernameEditText.getText().toString().trim();  String password = passwordEditText.getText().toString().trim();  // Kiểm tra nếu các trường nhập liệu rỗng  if (username.isEmpty() || password.isEmpty()) {  Toast.makeText(Login.this, "Vui lòng điền đầy đủ thông tin!", Toast.LENGTH\_SHORT).show();  return;  }  // Kiểm tra thông tin đăng nhập từ cơ sở dữ liệu  UserDAO userDAO = new UserDAO(Login.this);  boolean isLoginSuccessful = userDAO.loginUser(username, password);  // Đăng nhập thành công  if (isLoginSuccessful) {  Log.d("Login", "Login successful for user: " + username);  Toast.makeText(Login.this, "Đăng nhập thành công", Toast.LENGTH\_SHORT).show();  // Lưu trạng thái đăng nhập vào SharedPreferences  SharedPreferences preferences = getSharedPreferences("UserPrefs", MODE\_PRIVATE);  SharedPreferences.Editor editor = preferences.edit();  editor.putBoolean("isLoggedIn", true);  editor.putString("username", username); // Lưu tên người dùng nếu cần  editor.apply();  // Chuyển sang màn hình chính  startActivity(new Intent(Login.this, MainActivity.class));  finish(); // Đóng màn hình đăng nhập  } else {  // Đăng nhập thất bại  Log.d("Login", "Invalid login attempt for user: " + username);  Toast.makeText(Login.this, "Tên đăng nhập hoặc mật khẩu không chính xác", Toast.LENGTH\_SHORT).show();  }  } } |
| --- |

#### 2.3.6.2. Đăng kí

**Mô tả**

* ImageView banner đăng ký ở phía trên
* TextView hiển thị tiêu đề "ĐĂNG KÝ"
* EditText nhập tên tài khoản, có icon người dùng bên trái
* EditText nhập email, có icon email bên trái
* EditText nhập mật khẩu, có icon khóa bên trái
* EditText nhập lại mật khẩu để xác nhận
* Button "Đăng ký" để tạo tài khoản mới

**Nghiệp vụ**

* Người dùng nhập đầy đủ thông tin đăng ký
  + Tên tài khoản
  + Email
  + Mật khẩu
  + Nhập lại mật khẩu
* Khi thông tin hợp lệ và người dùng nhấn đăng ký
  + Tạo tài khoản mới trong cơ sở dữ liệu
  + Chuyển người dùng đến màn hình đăng nhập
* Nếu có lỗi trong quá trình đăng ký
* Hiển thị thông báo lỗi cụ thể cho từng trường hợp
* Giữ nguyên thông tin đã nhập để người dùng có thể chỉnh sửa

**Hình dung màn hình**

A black and white sign

Description automatically generated

*Hình 2.35: Hình dung màn hình đăng ký*

Screenshot of a screenshot of a screenshot of a screenshot of a screenshot of a screenshot of a screenshot of a screenshot of a screenshot of a screenshot of

Description automatically generated

*Hình 2.36: Màn hình android quản lý đăng kí*

A screenshot of a phone

Description automatically generated

*Hình 2.37: Giao diện android quản lý đăng kí*

***Các phương thức đã sử dụng***

*Bảng 2.26: Phương thức registerButton()*

| registerButton.setOnClickListener(v -> {  String username = usernameEditText.getText().toString();  String password = passwordEditText.getText().toString();  String confirmPassword = confirmPasswordEditText.getText().toString();  if (username.isEmpty() || password.isEmpty() || confirmPassword.isEmpty()) {  Toast.makeText(Register.this, "Vui lòng điền đầy đủ thông tin!", Toast.LENGTH\_SHORT).show();  } else if (!password.equals(confirmPassword)) {  Toast.makeText(Register.this, "Mật khẩu không khớp!", Toast.LENGTH\_SHORT).show();  } else {  // Thực hiện đăng ký người dùng  UserDAO userDAO = new UserDAO(Register.this);  boolean isRegistered = userDAO.registerUser(username, "email@example.com", password); // email có thể được lấy từ EditText hoặc mặc định  if (isRegistered) {  Toast.makeText(Register.this, "Đăng ký thành công", Toast.LENGTH\_SHORT).show();  startActivity(new Intent(Register.this, Login.class)); // Chuyển đến màn hình Login  } else {  Toast.makeText(Register.this, "Đăng ký thất bại", Toast.LENGTH\_SHORT).show();  }  }  });  // Chuyển sang màn hình Login khi nhấn vào "Đã có tài khoản? Đăng nhập"  loginPrompt.setOnClickListener(v -> {  startActivity(new Intent(Register.this, Login.class));  }); |
| --- |

*Bảng 2.27: Phương thức registerButton()*

*Bảng 2.27**Phương thức registerUser()*

| public boolean registerUser(String username, String email, String password) {  ContentValues values = new ContentValues();  values.put("username", username);  values.put("email", email);  values.put("password", password);  long result = db.insert("Users", null, values); // Thêm dữ liệu vào bảng Users  return result != -1; // Trả về true nếu thêm thành công  } |
| --- |

### Chức năng Pomodoro Timer

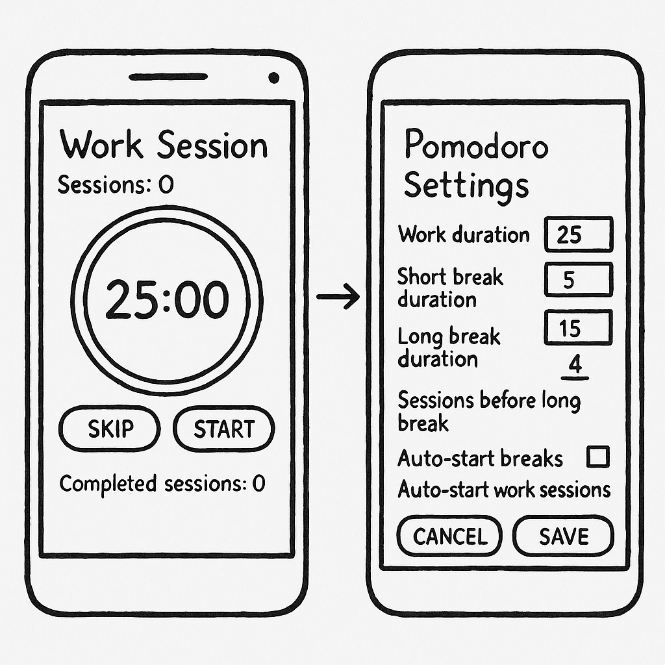
**Mô tả**

* Màn hình chính Pomodoro
* Hiển thị số phiên đã hoàn thành (Sessions: X)
* Hiển thị trạng thái hiện tại (Work Session/Break)
* Vòng tròn tiến độ lớn ở giữa màn hình
* Thời gian còn lại ở trung tâm vòng tròn (ví dụ: 25:00)
* Nút Skip để bỏ qua phiên hiện tại
* Nút Start/Pause để điều khiển đồng hồ
* Dialog cài đặt Pomodoro (mở từ menu)
* Thời gian làm việc (mặc định 25 phút)
* Thời gian nghỉ ngắn (mặc định 5 phút)
* Thời gian nghỉ dài (mặc định 15 phút)
* Số phiên trước khi nghỉ dài
* Mục tiêu số phiên mỗi ngày
* Tùy chọn tự động bắt đầu giờ nghỉ
* Tùy chọn tự động bắt đầu giờ làm việc

**Nghiệp vụ**

* Điều khiển Pomodoro
* Người dùng nhấn Start để bắt đầu đếm ngược
* Có thể Pause để tạm dừng bất cứ lúc nào
* Nút Skip chỉ kích hoạt khi đồng hồ đang chạy
* Vòng tròn tiến độ và thời gian được cập nhật liên tục
* Chu kỳ làm việc
* Work Session -> Short Break
* Short Break -> Work Session
* Sau X phiên -> Long Break (X là số phiên cấu hình)
* Long Break -> Work Session mới
* Tự động hóa (tùy theo cài đặt)
* Tự động bắt đầu giờ nghỉ sau khi kết thúc work session
* Tự động bắt đầu work session sau khi kết thúc giờ nghỉ
* Theo dõi tiến độ
* Đếm số phiên đã hoàn thành
* So sánh với mục tiêu phiên trong ngày
* Hiển thị trạng thái hiện tại rõ ràng

**Hình dung màn hình**

****

*Hình 2.38: Hình dung màn hình pomodoro timer*

Screens screenshot of a computer

Description automatically generated

*Hình 2.39: Màn hình pomodoro timer*

**A screenshot of a phone

Description automatically generated**

*Hình 2.40 Giao diện màn hình pomodoro timer*

***Các phương thức đã sử dụng***

*Bảng 2.28 Phương thức startTimer(), startTimerUpdates(), pauseTimer() và skipSession()*

| private void startTimer() {  if (currentSession == null) {  currentSession = new WorkSession();  currentSession.setUserId(UserSession.getInstance().getUserId());  currentSession.setStartTime(System.currentTimeMillis());  currentSession.setSessionType(timerState.isWorkSession() ? "work" : "break");  }  timerState.startTimer();  startTimerUpdates();  updateButtons();  }  private void startTimerUpdates() {  handler.post(new Runnable() {  @Override  public void run() {  if (timerState.isRunning()) {  updateTimerUI();  if (timerState.getTimeLeftInMillis() > 0) {  handler.postDelayed(this, 1000);  } else {  completeSession();  }  }  }  });  }  private void pauseTimer() {  timerState.pauseTimer();  handler.removeCallbacksAndMessages(null);  updateButtons();  }  private void skipSession() {  handler.removeCallbacksAndMessages(null);  timerState.setRunning(false);  updateButtons();  } |
| --- |

### Chức năng Habit Tracking

**Mô tả:**

* Màn hình chính Habits:
  + Tiêu đề "My Habits"
  + ListView hiển thị danh sách thói quen
  + Thông báo "No habits yet" khi chưa có thói quen nào
  + Nút FAB "+" để thêm thói quen mới
* Mỗi item thói quen (CardView) hiển thị:
  + Tên thói quen
  + Mô tả thói quen
  + Thanh tiến độ và phần trăm hoàn thành
  + Checkbox đánh dấu hoàn thành
  + Nút xóa thói quen
* Dialog thêm thói quen mới:
  + Nhập tên thói quen
* Chọn tần suất thực hiện:
  + HÀNG NGÀY: Chọn các ngày cụ thể (T2-CN)
  + HÀNG TUẦN: Chọn số ngày mỗi tuần
  + KHOẢNG THỜI GIAN: Chọn số ngày giữa các lần thực hiện
  + Tùy chọn mục tiêu "Đạt được tất cả"
  + Chọn ngày bắt đầu

**Nghiệp vụ:**

* Quản lý thói quen:
  + Thêm thói quen mới với các thông tin:
  + Tên thói quen
  + Tần suất thực hiện
  + Ngày bắt đầu
  + Mục tiêu hoàn thành
  + Xóa thói quen không cần thiết
  + Xem danh sách tất cả thói quen
* Theo dõi tiến độ:
  + Đánh dấu hoàn thành thói quen hàng ngày
  + Hiển thị tiến độ bằng thanh progress bar
  + Tính toán tỷ lệ hoàn thành
* Tần suất thực hiện:
  + Hàng ngày: Chọn các ngày cụ thể trong tuần
  + Hàng tuần: Đặt mục tiêu số ngày cần thực hiện mỗi tuần
  + Khoảng thời gian: Đặt số ngày giữa các lần thực hiện

**Hình dung màn hình:**

A screenshot of a phone

Description automatically generated

*Hình 2.41: Hình dung màn hình Habit tracker*

**Screens screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

*Hình 2.42: Màn hình hiển thị habit và add habit*

**A screenshot of a computer

Description automatically generated** **A screenshot of a phone

Description automatically generated**

*Hình 2.43: Giao diện màn hình hiển thị habit và add habit*

*:*

***Các phương thức đã sử dụng:***

*chBảng 2.29: Phương thức addHabit()*

| private boolean addHabit(){  try {  // Kiểm tra null cho các view cần thiết  if (editTextName == null || tabLayout == null) {  showError("Lỗi khởi tạo giao diện");  return false;  }  String name = editTextName.getText().toString().trim();  if (name.isEmpty()) {  editTextName.setError("Vui lòng nhập tên");  return false;  }  // Kiểm tra UserSession  UserSession userSession = UserSession.getInstance();  if (userSession == null || userSession.getUserId() <= 0) {  showError("Vui lòng đăng nhập lại");  return false;  }  Habit habit = new Habit();  habit.setName(name);  habit.setStartDate(calendar.getTime());  habit.setTargetDays(selectedTargetDays == -1 ? null : selectedTargetDays);  habit.setAllDayGoal(checkBoxAllDay.isChecked());  habit.setReminderTime(selectedReminderTime);  habit.setTimesPerDay(numberPickerTimesPerDay.getValue());  habit.setLastResetDate(System.currentTimeMillis());    // Xử lý targetDays  if (selectedTargetDays > 0) {  habit.setTargetDays(selectedTargetDays);  }  // Lưu tần suất dựa trên tab được chọn  int selectedTab = tabLayout.getSelectedTabPosition();  switch (selectedTab) {  case 0: // Hàng ngày  if (dayButtons == null || dayButtons.length != 7) {  showError("Lỗi khởi tạo nút chọn ngày");  return false;  }  StringBuilder selectedDays = new StringBuilder();  boolean hasSelectedDay = false;  for (ToggleButton button : dayButtons) {  if (button == null) continue;  boolean isChecked = button.isChecked();  selectedDays.append(isChecked ? "1" : "0");  if (isChecked) hasSelectedDay = true;  }  if (!hasSelectedDay) {  showError("Vui lòng chọn ít nhất một ngày");  return false;  }  habit.setFrequencyData(selectedDays.toString());  habit.setFrequencyType("daily");  break;  case 1: // Hàng tuần  if (daysPerWeekPicker == null) {  showError("Lỗi khởi tạo bộ chọn số ngày");  return false;  }  habit.setFrequencyData(String.valueOf(daysPerWeekPicker.getValue()));  habit.setFrequencyType("weekly");  break;  case 2: // Khoảng thời gian  if (intervalDaysPicker == null) {  showError("Lỗi khởi tạo bộ chọn khoảng thời gian");  return false;  }  habit.setFrequencyData(String.valueOf(intervalDaysPicker.getValue()));  habit.setFrequencyType("interval");  break;  default:  showError("Vui lòng chọn loại tần suất");  return false;  }  habit.setUserId(userSession.getUserId());  habit.setCompleted(false);  habit.setCompletedCount(0);  try {  long result = habitDAO.insert(habit);  if (result > 0) {  if (listener != null) {  listener.onHabitAdded();  }  return true;  } else {  showError("Không thể thêm thói quen");  }  } catch (Exception e) {  showError("Lỗi khi lưu thói quen: " + e.getMessage());  }  } catch (Exception e) {  showError("Lỗi không xác định: " + e.getMessage());  }  return false;  } |
| --- |

# PHẦN 3. KIẾN THỨC LĨNH HỘI VÀ BÀI HỌC KINH NGHIỆM

## 3.1. Nội dung đã thực hiện

***Thành tựu đã đạt được***

Trong quá trình thực hiện đề tài "***Xây dựng ứng dụng quản lý công việc trên điện thoại di động***", nhóm đã thu được nhiều kiến thức và kỹ năng quan trọng. Dưới đây là một số điểm nhấn về những thành tựu và kinh nghiệm mà nhóm đã đạt được:

• **Kiến thức về phát triển ứng dụng di động**: Qua quá trình tìm hiểu và thực hành, nhóm đã nắm được các kiến thức phát triển ứng dụng Android phổ biến như kiến trúc MVVM, quản lý vòng đời Activity/Fragment, quản lý dữ liệu với SQLite/Room, thiết kế giao diện người dùng theo chuẩn Material Design và xử lý các tác vụ bất đồng bộ. Đã hiểu cách ứng dụng những kiến thức này để tạo ra một ứng dụng di động chất lượng cao và thân thiện với người dùng.

• **Sử dụng công cụ phát triển Android Studio và Java**: Nhóm đã học cách sử dụng Android Studio hiệu quả từ việc thiết kế giao diện bằng XML, sử dụng các thư viện hỗ trợ của Android Jetpack, đến việc debug và tối ưu hiệu suất ứng dụng. Đồng thời, nhóm đã củng cố kiến thức về ngôn ngữ lập trình Java và áp dụng các nguyên tắc lập trình hướng đối tượng trong dự án.

• **Kỹ năng giao tiếp và làm việc nhóm**: Qua việc thực hiện các hoạt động nhóm, chúng em đã rèn luyện kỹ năng giao tiếp hiệu quả. Chúng em đã học cách phân công công việc, phối hợp phát triển các module khác nhau (quản lý tác vụ, theo dõi thói quen, đồng hồ Pomodoro, v.v.) và giải quyết xung đột trong nhóm để đạt được mục tiêu chung.

• **Kỹ năng quản lý dự án và kiểm soát phiên bản**: Nhóm đã sử dụng Git/GitHub để quản lý mã nguồn, thiết lập quy trình làm việc phù hợp, và tổ chức các buổi họp định kỳ để theo dõi tiến độ dự án. Kỹ năng này giúp nhóm làm việc nhịp nhàng và hạn chế xung đột khi nhiều thành viên cùng phát triển một dự án.

***Hạn chế***

Mặc dù nhóm đã thu hoạch được nhiều thành tựu và các bài học kinh nghiệm, song đề tài không thể tránh khỏi một số hạn chế như:

• **Hiệu suất ứng dụng**: Một số tính năng như hiển thị danh sách tác vụ lớn hoặc xử lý dữ liệu thói quen có thể chưa được tối ưu hoàn toàn, dẫn đến hiệu suất chưa mượt mà, giao diện tương tác người dùng chưa được đạt chuẩn UX/UI, cần thời gian thu thập thêm các phản hồi của người dùng để nâng cao trải nghiệm mà ứng dụng mang lại

• **Đồng bộ hóa dữ liệu**: Ứng dụng hiện tại chỉ lưu trữ dữ liệu cục bộ trên thiết bị, chưa hỗ trợ đồng bộ hóa dữ liệu lên đám mây hoặc giữa nhiều thiết bị khác nhau.

• **Hỗ trợ đa ngôn ngữ**: Ứng dụng chủ yếu hỗ trợ tiếng Anh, cần bổ sung hỗ trợ đa ngôn ngữ để tiếp cận người dùng quốc tế.

• **Kiểm thử chưa toàn diện**: Mặc dù đã thực hiện kiểm thử cơ bản, nhưng chưa có bộ kiểm thử tự động toàn diện cho mọi tính năng và trường hợp sử dụng.

Song các hạn chế trên là có thể chấp nhận với phạm vi thời gian hoàn thành đề tài. Đây sẽ là những hứa hẹn, động lực to lớn giúp nhóm cải thiện các kỹ năng tích lũy được trong thời gian tới.

***Bài học kinh nghiệm***

Trong quá trình thực hiện đề tài "Xây dựng ứng dụng quản lý công việc trên điện thoại di động Tasktory", nhóm đã có những bài học kinh nghiệm sau:

• **Phát triển ứng dụng Android với nhiều tính năng phức tạp**: Nhóm đã tìm hiểu và triển khai thành công các tính năng quan trọng của ứng dụng quản lý công việc như quản lý tác vụ, phân loại và gắn thẻ, đồng hồ Pomodoro, theo dõi thói quen (Habit Tracker), hệ thống thử thách (Challenge), hệ thống nhắc nhở, và tìm kiếm. Mỗi tính năng đều đòi hỏi những kỹ thuật lập trình khác nhau và cách tiếp cận riêng.

• **Thiết kế cơ sở dữ liệu và quản lý dữ liệu**: Nhóm đã thiết kế và triển khai cơ sở dữ liệu SQLite với nhiều bảng dữ liệu có mối quan hệ phức tạp như Task, Category, Tag, TaskTags, Habit, Challenge, User, WorkSession. Việc này giúp nhóm hiểu sâu về thiết kế cơ sở dữ liệu và cách tương tác hiệu quả với dữ liệu trong ứng dụng Android.

• **Nhận xét và đánh giá**: Qua quá trình làm đề tài, chúng em đã nâng cao được kiến thức và kỹ năng về phát triển ứng dụng Android và ứng dụng vào thực tế. Ngoài ra, chúng em cũng đã có cơ hội làm việc nhóm, trao đổi ý kiến và giải quyết các vấn đề phát sinh. Tuy nhiên, nhóm cũng gặp một số khó khăn như xử lý đồng thời nhiều luồng dữ liệu, tối ưu hiệu suất ứng dụng trên các thiết bị cấu hình thấp, và đảm bảo trải nghiệm người dùng nhất quán. Nhóm đã cố gắng tìm hiểu và giải quyết các khó khăn này bằng cách nghiên cứu tài liệu, học hỏi từ cộng đồng phát triển Android, và thử nghiệm nhiều giải pháp khác nhau.

## 3.2. Hướng phát triển

Trong thời gian tới, nhóm hy vọng sẽ phát triển ứng dụng Tasktory theo các hướng sau:

• **Mở rộng nền tảng**: Phát triển phiên bản cho hệ điều hành iOS và phiên bản web để người dùng có thể truy cập từ nhiều thiết bị khác nhau.

• **Tích hợp đám mây**: Xây dựng hệ thống đồng bộ hóa dữ liệu lên đám mây để bảo vệ dữ liệu người dùng và cho phép truy cập từ nhiều thiết bị.

• **Tính năng nâng cao**: Bổ sung các tính năng nâng cao như báo cáo và phân tích năng suất, tích hợp lịch Google, chia sẻ tác vụ và thử thách với bạn bè.

• **Tối ưu hóa hiệu suất**: Cải thiện hiệu suất ứng dụng, giảm thiểu thời gian phản hồi và sử dụng tài nguyên hệ thống.

• **Kiểm thử toàn diện**: Xây dựng bộ kiểm thử tự động cho mọi tính năng để đảm bảo chất lượng ứng dụng khi phát triển tính năng mới.

Đây sẽ là bước tiến cơ sở để mỗi thành viên trong nhóm tự tin hơn bước vào thị trường lao động công nghệ và góp sức mình vào công cuộc cách mạng số hóa hiện nay.

Một lần nữa, nhóm xin được gửi lời cảm ơn tới giảng viên hướng dẫn cùng toàn thể quý thầy, cô giảng viên khoa CNTT và các bạn độc giả đã quan tâm theo dõi dự án Tasktory của chúng em.

# TÀI LIỆU THAM KHẢO

**Tài liệu tiếng Việt**

[1]. Nguyễn Bá Nghiễn, Nguyễn Thái Cường, Vũ Thị Dương, Nguyễn Văn Tỉnh, Phát triển ứng dụng trên thiết bị di động, 2020

[2]. Phùng Đức Hòa (Chủ biên), Hoàng Quang Huy, Hoàng Văn Hoành, Nguyễn Đức Lưu, Trịnh Bá Quý. Giáo trình Nhập môn công nghệ phần mềm. Thống kê, 2019

**Tài liệu tiếng Anh**

[1]. Google LLC. (2023). Android Developers Documentation. Truy xuất ngày 25 tháng 04 năm 2025, từ <https://developer.android.com/docs>

[2]. SQLite. (2023). *SQLite Documentation*. Truy xuất ngày 21 tháng 04 năm 2025, từ <https://www.sqlite.org/docs.html>