**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

------🙙🕮🙛-------

Ảnh có chứa văn bản, biểu tượng, vòng tròn, ảnh chụp màn hình

Mô tả được tạo tự động

**BÁO CÁO BÀI TẬP LỚN HỌC PHẦN**

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

**ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG QUẢN LÝ THỜI GIAN**

|  |  |  |
| --- | --- | --- |
| **GVHD** | **:** ThS. Đỗ Hữu Công |  |
| **Nhóm – Lớp** | **:** 05 - 2024IT6029002 |  |
| **Sinh viên thực hiện** | **:** Nguyễn Duy Văn | 2022604430 |
|  | Phan Văn Bộ | 2022606440 |
|  | Nguyễn Đức Toàn | 2022605394 |
|  | Trần Văn Cường | 2022606203 |
|  | Nguyễn Thanh Tùng | 2022606334 |
|  |  |  |

**HÀ NỘI – 05/2025LỜI CẢM ƠN**

Lời đầu tiên, nhóm 05 chúng em xin gửi lời cảm ơn chân thành tới thầy hướng dẫn ThS. Đỗ Hữu Công. Trong quá trình học tập và thực hiện đề tài này, chúng em đã nhận được sự quan tâm giúp đỡ, hướng dẫn tận tình, tâm huyết của thầy, đã truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian thực hiện báo cáo thực nghiệm. Nhờ có những hướng dẫn, dạy bảo của các thầy nên đề tài báo cáo của em mới có thể hoàn thiện tốt đẹp.Đây cũng chính là hành trang quan trọng giúp chúng em tự tin bước chân vào môi trường làm việc thực tế.

Dưới sự dẫn dắt của thầy và sự cố gắng, nhiệt huyết sự nỗ lực không ngừng kết hợp kiến thức thuần túy từ giảng đường với những kiến thức của các thành viên trong nhóm, chúng em đã hoàn thành báo cáo này. Chúng em rất trân trọng và mong muốn sẽ nhận được những lời nhận xét, góp ý từ thầy cô và các bạn để bài báo cáo này có thể hoàn thiện hơn nữa.

Chúng em xin chân thành cảm ơn!

***Nhóm sinh viên thực hiện***

***Nhóm 05***

**MỤC LỤC**

[DANH MỤC HÌNH ẢNH 1](#_Toc197976048)

[DANH MỤC BẢNG BIỂU 3](#_Toc197976049)

[MỞ ĐẦU 4](#_Toc197976050)

[Lý do chọn đề tài 4](#_Toc197976051)

[Mục tiêu đề tài 5](#_Toc197976052)

[Đối tượng và phạm vi nghiên cứu 6](#_Toc197976053)

[Kết quả mong muốn 6](#_Toc197976054)

[CHƯƠNG 1. TỔNG QUAN VỀ LẬP TRÌNH ỨNG DỤNG TRÊN THIẾT BỊ DI ĐỘNG 8](#_Toc197976055)

[1.1. Tổng quan về lập trình trên thiết bị di động 8](#_Toc197976056)

[1.2. Tổng quan về công cụ đề tài 8](#_Toc197976057)

[1.2.1 Công cụ Android 8](#_Toc197976058)

[1.2.2 Java Android 10](#_Toc197976059)

[1.2.3 Các công cụ, công nghệ hỗ trợ thực hiện đề tài khác 10](#_Toc197976060)

[1.3. Mô hình MVVM 11](#_Toc197976061)

[CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 12](#_Toc197976062)

[2.1. Tìm hiểu yêu cầu bài toán 12](#_Toc197976063)

[2.2. Phân tích thiết kế bài toán 12](#_Toc197976064)

[2.2.1 Phân tích yêu cầu người dùng 12](#_Toc197976065)

[2.2.2 Biểu đồ thực thể liên kết 14](#_Toc197976066)

[2.2.3 Đặc tả chức năng 18](#_Toc197976067)

[2.2.4 Thiết kế hệ thống 32](#_Toc197976068)

[2.3. Thực hiện bài toán 35](#_Toc197976069)

[2.3.1 Đăng ký 37](#_Toc197976070)

[2.3.2 Đăng nhập 41](#_Toc197976071)

[2.3.3 Quản lý thời gian 44](#_Toc197976072)

[2.3.4 Hẹn giờ 56](#_Toc197976073)

[2.3.5 Bấm giờ 61](#_Toc197976074)

[2.3.6 Xem lịch 66](#_Toc197976075)

[2.3.7 Thống kê 70](#_Toc197976076)

[2.3.8 Quản lý người dùng 75](#_Toc197976077)

[CHƯƠNG 3. KIẾN THỨC LĨNH HỘI 85](#_Toc197976078)

[3.1. Nội dung đã thực hiện 85](#_Toc197976079)

[3.2. Hướng phát triển 87](#_Toc197976080)

[TÀI LIỆU THAM KHẢO 88](#_Toc197976081)

DANH MỤC HÌNH ẢNH

[Hình 1.1. Mô hình Model - View - ViewModel 11](#_Toc197976018)

[Hình 2.1. Biểu đồ lớp 15](#_Toc197976019)

[Hình 2.2. Bảng User 15](#_Toc197976020)

[Hình 2.3. Bảng UserDetail 15](#_Toc197976021)

[Hình 2.4.Bảng TimeTable 16](#_Toc197976022)

[Hình 2.5. Biểu đồ UseCase 17](#_Toc197976023)

[Hình 2.6. UseCase Đăng ký 18](#_Toc197976024)

[Hình 2.7. UseCase Đăng nhập 19](#_Toc197976025)

[Hình 2.8. UseCase Quản lý tài khoản 21](#_Toc197976026)

[Hình 2.9. UseCase Quản lý thời gian 23](#_Toc197976027)

[Hình 2.10. UseCase Xem lịch 25](#_Toc197976028)

[Hình 2.11. UseCase Hẹn giờ 26](#_Toc197976029)

[Hình 2.12. UseCase Bấm giờ 28](#_Toc197976030)

[Hình 2.13. UseCase Sao lưu 29](#_Toc197976031)

[Hình 2.14. UseCase Trợ giúp 31](#_Toc197976032)

[Hình 2.15. Giao diện tổng quan 35](#_Toc197976033)

[Hình 2.16. Cấu trúc tổng quát 36](#_Toc197976034)

[Hình 2.17. Giao diện đăng ký 37](#_Toc197976035)

[Hình 2.18. Giao diện đăng nhập 41](#_Toc197976036)

[Hình 2.19. Giao diện Quản lý thời gian 45](#_Toc197976037)

[Hình 2.20. Giao diện sửa thời gian biểu 51](#_Toc197976038)

[Hình 2.21. Giao diện xóa thời gian biểu 53](#_Toc197976039)

[Hình 2.22. Giao diện Xem chi tiết thời gian biểu 54](#_Toc197976040)

[Hình 2.23. Giao diện Hẹn giờ 56](#_Toc197976041)

[Hình 2.24. Giao diện Bấm giờ 62](#_Toc197976042)

[Hình 2.25. Giao diện Xem lịch 66](#_Toc197976043)

[Hình 2.26. Giao diện Thống kê 70](#_Toc197976044)

[Hình 2.27. Giao diện quản lý người dùng 76](#_Toc197976045)

[Hình 2.28. Giao diện Trợ giúp 76](#_Toc197976046)

[Hình 2.29. Giao diện Sao lưu 78](#_Toc197976047)

DANH MỤC BẢNG BIỂU

[Bảng 2.1. Mô tả UseCase Đăng ký 18](#_Toc197976005)

[Bảng 2.2. Mô tả UseCase Đăng nhập 19](#_Toc197976006)

[Bảng 2.3. Mô tả UseCase Quản lý tài khoản 21](#_Toc197976007)

[Bảng 2.4. Mô tả UseCase Quản lý thời gian 23](#_Toc197976008)

[Bảng 2.5. Mô tả UseCase Xem lịch 25](#_Toc197976009)

[Bảng 2.6. Mô tả UseCase Hẹn giờ 26](#_Toc197976010)

[Bảng 2.7. Mô tả UseCase Bấm giờ 28](#_Toc197976011)

[Bảng 2.8. Mô tả UseCase Sao lưu 30](#_Toc197976012)

[Bảng 2.9. Mô tả UseCase Trợ giúp 31](#_Toc197976013)

MỞ ĐẦU

Lý do chọn đề tài

Trong thời đại công nghệ số phát triển mạnh mẽ hiện nay, quỹ thời gian là một trong những tài sản quý giá nhất đối với mỗi cá nhân. Một lịch trình hợp lý giúp con người làm việc hiệu quả hơn, đạt được mục tiêu học tập, công việc và cân bằng cuộc sống. Tuy nhiên, trước áp lực của nhịp sống hiện đại, con người dễ rơi vào tình trạng quên công việc, trễ hẹn hoặc không sắp xếp thời gian khoa học, gây ảnh hưởng tiêu cực đến hiệu suất làm việc và chất lượng cuộc sống.

Với sự phổ biến rộng rãi của điện thoại thông minh, việc phát triển một ứng dụng di động hỗ trợ người dùng trong việc ghi chú, lên kế hoạch và quản lý công việc hằng ngày là vô cùng cần thiết. Đặc biệt, nền tảng Android với ưu thế về số lượng người dùng lớn và khả năng tùy chỉnh cao là môi trường lý tưởng để triển khai một ứng dụng như vậy.

Chính vì lý do đó, nhóm chúng em quyết định thực hiện đề tài "Xây dựng ứng dụng quản lý thời gian". Thông qua đề tài này, chúng em không chỉ mong muốn giúp người dùng cải thiện kỹ năng quản lý thời gian mà còn rèn luyện khả năng lập trình Android, thiết kế cơ sở dữ liệu, quản lý giao diện người dùng, từ đó phát triển kỹ năng làm việc nhóm và tiếp cận với quy trình xây dựng phần mềm thực tế.

Nhóm chúng em thực hiện nghiêm cứu theo 3 phần chính:

* + **Phần mở đầu:** tổng quan về lập trình trên thiết bị di động, kiến trúc của Android, một số nền tảng lập trình trên thiết bị di động phổ biến hiện nay.
  + **Phần kết quả nghiên cứu:** phân tích yêu cầu của người dùng để đưa ra các yêu cầu chức năng và phi chức năng của hệ thống. Sau đó, thiết kế cơ sở dư liệu và giao diện cho hệ thống.
  + **Phần kiến thức lĩnh hội và bài học kinh nghiệm:** đưa ra những kiến thức đã học được trong quá trình nghiên cứu và các bài học kinh nghiệm.

Mục tiêu đề tài

Mục tiêu chính của đề tài là xây dựng một ứng dụng di động hoạt động trên nền tảng Android, hỗ trợ người dùng quản lý hiệu quả quỹ thời gian của mình. Cụ thể:

* Thiết kế và phát triển một ứng dụng thân thiện với người dùng, cho phép:
  + Tạo mới các công việc hoặc nhiệm vụ cần thực hiện.
  + Chỉnh sửa, cập nhật và xóa công việc khi cần thiết.
  + Thống kê và đánh giá mức độ hoàn thành công việc theo ngày, tuần, tháng.
  + Xem lịch
  + Hẹn giờ
  + Bấm giờ
  + Sao lưu và hỗ trợ người dùng khi cần thiết.
* Áp dụng kiến thức lập trình Android hiện đại, sử dụng mô hình kiến trúc MVVM (Model - View - ViewModel) nhằm:
  + Tách biệt rõ ràng giữa giao diện (View) và xử lý nghiệp vụ (ViewModel).
  + Dễ dàng bảo trì, nâng cấp và mở rộng ứng dụng trong tương lai.
* Quản lý và lưu trữ dữ liệu công việc bằng cơ sở dữ liệu cục bộ SQLite, đảm bảo:
  + Dữ liệu được lưu trữ an toàn ngay trên thiết bị.
  + Người dùng có thể truy cập công việc mọi lúc, mọi nơi ngay.

Hướng đến việc xây dựng một ứng dụng nhẹ, hoạt động ổn định trên nhiều dòng điện thoại Android phổ biến.

Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu:

Người dùng sử dụng điện thoại Android, đặc biệt là:

* + Sinh viên cần quản lý lịch học, bài tập, dự án.
  + Nhân viên văn phòng cần tổ chức công việc, họp hành, dự án cá nhân.
  + Các cá nhân có nhu cầu lập kế hoạch cho công việc, học tập và cuộc sống cá nhân.

Phạm vi nghiên cứu:

* + Nền tảng: Phát triển ứng dụng trên hệ điều hành Android, sử dụng Android Studio làm công cụ lập trình chính.
  + Ngôn ngữ: Java.
  + Kiến trúc phần mềm: Áp dụng mô hình MVVM (Model - View - ViewModel) để quản lý luồng dữ liệu và giao diện.
  + Cơ sở dữ liệu: Sử dụng SQLite để lưu trữ thông tin công việc cá nhân trong ứng dụng.
  + Chức năng hỗ trợ: Ứng dụng tập trung vào các chức năng cơ bản về quản lý thời gian như thêm, sửa, xóa công việc, nhắc nhở nhiệm vụ.
  + Giới hạn: Ứng dụng chỉ hoạt động trên thiết bị Android, chưa hỗ trợ đồng bộ hóa dữ liệu trên nền tảng đám mây hoặc các thiết bị khác.

Kết quả mong muốn

Sau khi hoàn thành đề tài, nhóm mong muốn đạt được những kết quả cụ thể sau:

* + Về sản phẩm:
  + Hoàn thiện một ứng dụng Android quản lý thời gian đơn giản nhưng hiệu quả.
  + Giao diện người dùng thân thiện, bố cục rõ ràng, dễ thao tác đối với người dùng ở mọi lứa tuổi.
  + Ứng dụng vận hành mượt mà, không phát sinh lỗi lớn trong quá trình sử dụng.
  + Về kỹ thuật:
  + Thành thạo kỹ năng lập trình Android cơ bản và nâng cao.
  + Hiểu và vận dụng tốt mô hình kiến trúc MVVM (Model - View - ViewModel) trong thực tế.
  + Quản lý và thao tác với cơ sở dữ liệu SQLite hiệu quả.
  + Rèn luyện khả năng thiết kế giao diện đẹp mắt, phù hợp với trải nghiệm người dùng (UX/UI).
  + Về kỹ năng mềm:
  + Phát triển kỹ năng làm việc nhóm, phân chia công việc hợp lý giữa các thành viên.
  + Nâng cao kỹ năng lập kế hoạch, giải quyết vấn đề và thích ứng với thay đổi trong quá trình thực hiện đề tài.

Qua đề tài này, nhóm mong muốn có thêm kinh nghiệm thực tiễn trong việc xây dựng ứng dụng Android, từ đó chuẩn bị hành trang vững chắc cho công việc và các dự án lớn hơn trong tương lai.

# TỔNG QUAN VỀ LẬP TRÌNH ỨNG DỤNG TRÊN THIẾT BỊ DI ĐỘNG

## Tổng quan về lập trình trên thiết bị di động

Lập trình trên thiết bị di động là quá trình phát triển các ứng dụng phần mềm hoạt động trên các thiết bị di động như điện thoại thông minh, máy tính bảng và đồng hồ thông minh. Với sự phát triển mạnh mẽ của công nghệ và nhu cầu sử dụng thiết bị di động trong đời sống hàng ngày, lập trình di động đã trở thành một lĩnh vực quan trọng và không ngừng mở rộng.Lập trình trên thiết bị di động là quá trình thiết kế, xây dựng và triển khai các phần mềm, ứng dụng chạy trên các thiết bị di động như smartphone, máy tính bảng. Với sự phát triển nhanh chóng của công nghệ, ứng dụng di động đã trở thành một phần không thể thiếu trong đời sống hiện đại, đáp ứng nhu cầu học tập, làm việc và quản lý cá nhân. Trong bối cảnh nhu cầu quản lý thời gian cá nhân và công việc ngày càng trở nên quan trọng, việc ghi chú, theo dõi và tối ưu hóa quỹ thời gian sử dụng mỗi ngày đã trở thành một yêu cầu thiết yếu. Đề tài này tập trung vào việc xây dựng một ứng dụng trên nền tảng Android nhằm hỗ trợ người dùng trong việc lập kế hoạch, quản lý công việc, nhắc nhở các nhiệm vụ quan trọng và đánh giá hiệu quả sử dụng thời gian.

## Tổng quan về công cụ đề tài

### Công cụ Android

Android Studio là môi trường phát triển tích hợp (IDE - Integrated Development Environment) chính thức do Google phát hành dành riêng cho việc phát triển ứng dụng Android. Android Studio được xây dựng dựa trên nền tảng IntelliJ IDEA, cung cấp một hệ sinh thái toàn diện, mạnh mẽ và tối ưu hóa cho việc lập trình Android.

#### Các đặc điểm nổi bật của Android Studio

* + Trình thiết kế giao diện (Layout Editor): Cho phép lập trình viên kéo thả (drag and drop) các thành phần giao diện như Button, TextView, ImageView,... trực quan trên màn hình thiết kế, đồng thời hỗ trợ chỉnh sửa XML nhanh chóng.Trình thiết kế giao diện trực quan (Layout Editor): Cho phép lập trình viên kéo thả các thành phần giao diện như Button, TextView, ImageView,... để thiết kế UI một cách nhanh chóng và trực quan.
  + Trình giả lập thiết bị Android (Emulator): Giúp mô phỏng nhiều loại thiết bị với các phiên bản Android khác nhau, hỗ trợ kiểm thử ứng dụng mà không cần thiết bị thật.Công cụ kiểm thử tích hợp: Android Studio hỗ trợ kiểm thử giao diện người dùng (UI Testing), kiểm thử đơn vị (Unit Testing) và kiểm thử tự động, giúp đảm bảo chất lượng ứng dụng.
  + Quản lý dự án với Gradle: Android Studio sử dụng Gradle để tự động hóa quá trình build, quản lý thư viện bên ngoài, tối ưu hóa việc biên dịch và đóng gói ứng dụng.
  + Công cụ kiểm tra và tối ưu hiệu suất: Android Studio cung cấp các công cụ như Profiler, Logcat giúp theo dõi hiệu suất CPU, bộ nhớ, mạng và phát hiện lỗi khi chạy ứng dụng.
  + Giao diện thân thiện và dễ sử dụng: Android Studio có giao diện được tổ chức hợp lý, hỗ trợ tự động hoàn thành mã (code completion), kiểm tra lỗi thời gian thực (lint checking) và nhiều tính năng tăng năng suất cho lập trình viên.
  + Hệ thống hỗ trợ kiểm thử (Testing): Hỗ trợ các phương pháp kiểm thử đơn vị (Unit Test) và kiểm thử giao diện (UI Test) ngay trong IDE.

#### Vai trò của Android Studio trong đề tài nghiên cứu

Trong đề tài “Xây dựng ứng dụng quản lý thời gian”, Android đóng vai trò là công cụ trung tâm để:

* + Viết mã nguồn ứng dụng
  + Tạo và quản lý cấu trúc dự án
  + Thiết kế giao diện quản lý thời gian
  + Giả lập và kiểm thử ứng dụng
  + Tạo cơ sở dữ liệu cục bộ (SQLite) và kết nối giao diện
  + Biên dịch và kiểm thử ứng dụng trước khi triển khai trên thiết bị Android thực tế
  + Quản lý thư viện và phụ thuộc

### Java Android

#### Khái niệm

Java là một trong những ngôn ngữ lập trình phổ biến nhất thế giới, đặc biệt trong phát triển phần mềm và web. Đối với Android – nền tảng di động được Google phát triển – Java đóng vai trò cực kỳ quan trọng, trở thành ngôn ngữ chính cho việc phát triển ứng dụng.

Hầu hết các thư viện chuẩn Java đều khả dụng trong Android, cùng với những thư viện riêng phục vụ giao diện người dùng và các tính năng di động đặc thù.

Java Android tuy có một số khác biệt so với Java tiêu chuẩn, nhưng vẫn giữ những điểm tương đồng về cú pháp và triết lý lập trình.

Java cũng nổi tiếng với khả năng tương thích cao, tính hướng đối tượng mạnh mẽ và hiệu suất tối ưu nhờ cơ chế thu gom rác (Garbage Collector).

#### Các tính năng nổi bật

Java là ngôn ngữ lập trình độc lập nền tảng, an toàn, thích hợp cho môi trường máy ảo.

Android SDK hỗ trợ nhiều thư viện Java tiêu chuẩn như thư viện cấu trúc dữ liệu, đồ họa, toán học, mạng, cùng với thư viện Android chuyên dụng.

Java kết hợp các tính năng mạnh mẽ từ C và C++, đồng thời khắc phục nhiều hạn chế của các ngôn ngữ này.

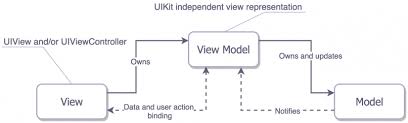
### Các công cụ, công nghệ hỗ trợ thực hiện đề tài khác

* + SQLite: Để lưu trữ dữ liệu cục bộ như danh sách công việc, lịch nhắc nhở..., ứng dụng sử dụng hệ quản trị cơ sở dữ liệu SQLite – nhẹ, đơn giản và tích hợp sẵn trong Android.
  + XML: Ngôn ngữ XML được dùng để thiết kế giao diện người dùng (UI), giúp xây dựng bố cục ứng dụng một cách trực quan, dễ dàng tùy chỉnh và tối ưu.
  + Các công cụ khác: GitHub (quản lý mã nguồn), Figma (thiết kế giao diện người dùng), Firebase (Lưu trữ dữ liệu online), thiết bị di động cá nhân hoặc phần mềm giả lập...

## Mô hình MVVM

**a. Khái niệm**

MVVM (Model - View - ViewModel) là một mô hình kiến trúc phần mềm được phát triển từ MVP (Model - View - Presenter), giúp định hình cách tổ chức mã nguồn trong ứng dụng. MVVM không phải là một thư viện hay framework, mà chỉ là một hướng dẫn kiến trúc.



Hình 1.1. Mô hình Model - View - ViewModel

**b. Các thành phần chính**

* + View: Giao diện người dùng, chịu trách nhiệm hiển thị dữ liệu và nhận sự tương tác.
  + Model: Lớp xử lý dữ liệu thực, tương tự như trong mô hình MVC.
  + ViewModel: Lớp trung gian giữa View và Model. ViewModel thực hiện binding dữ liệu và xử lý logic giao tiếp giữa Model và View, nhưng không trực tiếp biết đến View.

**c. Đặc điểm**

* + Phân tách rõ ràng giữa giao diện và logic nghiệp vụ.
  + Tăng hiệu quả trong việc viết unit test.
  + Phù hợp cho các ứng dụng hỗ trợ nhiều kích cỡ và loại thiết bị khác nhau.
  + Một ViewModel có thể được sử dụng cho nhiều View.
  + ViewModel sử dụng Observer Pattern để đồng bộ hóa dữ liệu với View, hỗ trợ binding dữ liệu một chiều hoặc hai chiều tùy yêu cầu.

# PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## Tìm hiểu yêu cầu bài toán

Nhận thức được tầm quan trọng của việc lập kế hoạch và tổ chức công việc hợp lý, nhóm chúng tôi đã quyết định nghiên cứu và phát triển một “ứng dụng quản lý thời gian” một cách khoa học, tiện lợi và dễ tiếp cận.

Xuất phát từ nhu cầu thực tế đó, nhóm chúng em nảy ra ý tưởng xây dựng một “ứng dụng quản lý thời gian”với các mục tiêu chính:

* + Cho phép người dùng thêm, sửa, xóa, chi tiết các công việc cần thực hiện.
  + Giao diện trực quan, dễ sử dụng, phù hợp cho mọi đối tượng.
  + Nhắc nhở người dùng về các công việc sắp đến hạn.
  + Cung cấp các công cụ thống kê theo ngày, tuần tháng của công việc theo trạng thái (hoàn thành/chưa hoàn thành).
  + Dữ liệu công việc được lưu trữ cục bộ bằng SQLite để đảm bảo an toàn và cho phép hoạt động offline.

Từ đó, ứng dụng sẽ giúp người dùng:

* Có cái nhìn tổng quan về công việc của mình.
* Sắp xếp thời gian hợp lý hơn.
* Nâng cao hiệu quả học tập, làm việc và chất lượng cuộc sống.

## Phân tích thiết kế bài toán

### Phân tích yêu cầu người dùng

**Yêu cầu chức năng:**

* + Đăng nhập/ Đăng ký để sử dụng hệ thống.
  + Quản lý thời gian biểu
  + Thêm, sửa, xóa, xem chi tiết thời gian biểu.
  + Thống kê
* Số công việc thực hiện được/ không thực hiện được.
* Biểu đồ đánh giá công việc
* Thống kê công việc theo ngày, tuần, tháng.
  + Hẹn giờ
  + Xem lịch
  + Bấm giờ
  + Quản lý người dùng
* Thông tin người dùng
* Trợ giúp
* Sao lưu dữ liệu công việc

**Yêu cầu phi chức năng:**

* + Yêu cầu về ngôn ngữ hệ thống: Do hệ thống phục vụ chủ yếu là người Việt Nam nên ngôn ngữ chủ yếu sử dụng tiếng Việt, không yêu cầu cao về đa ngôn ngữ.
  + Hiệu năng:
* Thời gian phản hồi yêu cầu của người dùng nhanh.
* Các thao tác của người dùng trên ứng dụng như kéo thả, trượt phải mượt mà, không bị giật lag.
* Ứng dụng có thể phục vụ cho nhiều người sử dụng trên cùng một ứng dụng được cài đặt trên một máy, thông qua việc đăng nhập tài khoản tương ứng.
  + Bảo mật:
* Đảm bảo thông tin người dùng an toàn, ứng dụng chỉ được mở khóa khi người dùng nhập đúng Tên tài khoản và Mật khẩu.
* Khi đăng ký tài khoản, tài khoản được tạo mới sẽ không trùng với bất kỳ tài khoản nào đã từng được tạo.
  + Khả năng tương thích thiết bị:
* Ứng dụng tương thích với nhiều mẫu máy sử dụng hệ điều hành Android khác nhau.
* Hệ thống phải hoạt động ổn định trên các phiên bản hệ điều hành từ Android 6.0 trở lên.
  + Yêu cầu về dung lượng:
* Ứng dụng cần có dung lượng cài đặt dưới 20 MB để tiết kiệm không gian lưu trữ trên thiết bị của người dùng.
* Dữ liệu người dùng phải được lưu trữ và quản lý một cách hiệu quả, tránh

### Biểu đồ thực thể liên kết

#### Biểu đồ liên kết thực thể

* Thiết kế cơ sở dữ liệu

A screenshot of a computer

Description automatically generated

Hình 2.1. Biểu đồ lớp

* Bảng cơ sở dữ liệu

A yellow box with black text

Description automatically generated

Hình 2.2. Bảng User

A yellow box with black text

Description automatically generated

Hình 2.3. Bảng UserDetail

A screenshot of a computer

Description automatically generated

Hình 2.4.Bảng TimeTable

#### Biểu đồ UseCase

A diagram of a person's relationship

Description automatically generated

Hình 2.5. Biểu đồ UseCase

### Đặc tả chức năng

#### Usecase Đăng ký

A diagram with a diagram and text

Description automatically generated with medium confidence

Hình 2.6. UseCase Đăng ký

Bảng 2.1. Mô tả UseCase Đăng ký

|  |
| --- |
| 1. Tên usecase: Usecase Đăng ký  2. Mô tả vắn tắt  Cho phép người dùng đăng ký tài khoản để đăng nhập vào hệ thống.  3. Luồng sự kiện  a) Luồng cơ bản  Khi người dùng kích vào nút “Đăng ký tài khoản”. Hệ thống sẽ hiển thị thông báo nhập thông tin gồm có email, mật khẩu, họ tên, lớp, số điện thoại. Người dùng kích vào nút “Đăng ký” hệ thống sẽ cập nhật thông tin gồm có email, mật khẩu, họ tên, lớp, số điện thoại vào bảng User của cơ sở dữ liệu và thông báo tạo thành công ra màn hình.  b) Luồng rẽ nhánh  Tại bất kì thời điểm nào nếu không thể kết nối với cơ sở dữ liệu thì usecase kết thúc.  4. Tiền điều kiện  Không có  5. Hậu điều kiện  Không có  6. Điều kiện tiên quyết  Không có  7. Điểm mở rộng  Không có |

#### Usecase Đăng nhập

A diagram with text on it

Description automatically generated

Hình 2.7. UseCase Đăng nhập

Bảng 2.2. Mô tả UseCase Đăng nhập

|  |
| --- |
| 1. Tên usecase: Usecase đăng nhập  2. Mô tả vắn tắt  Cho phép người dùng đăng nhập và sử dụng hệ thống quản lý thời gian.  3. Luồng sự kiện  a) Luồng cơ bản  Người dùng vào hệ thống và thực hiện chức năng đăng nhập. Hệ thống sẽ hiển thị thông báo nhập email và mật khẩu. Người dùng nhập email và mật khẩu sau đó click nút “Đăng nhập”. Hệ thống sẽ chuyển sang trang chính.  b) Luồng rẽ nhánh  Tại bất kì thời điểm nào nếu không thể kết nối với cơ sở dữ liệu thì usecase kết thúc.  4. Tiền điều kiện  Người dùng phải đăng ký tài khoản trước.  5. Hậu điều kiện  Không có  6. Điều kiện tiên quyết  Không có  7. Điểm mở rộng  Không có |

#### Usecase Quản lý tài khoản

A diagram with text on it

Description automatically generated

Hình 2.8. UseCase Quản lý tài khoản

Bảng 2.3. Mô tả UseCase Quản lý tài khoản

|  |
| --- |
| 1. Tên usecase: Usecase quản lý tài khoản  2. Mô tả vắn tắt  Cho phép người dùng xem thông tin chi tiết, trợ giúp và sao lưu dữ liệu của bản thân.  3. Luồng sự kiện  a) Luồng cơ bản  - Xem thông tin  Khi người dùng kích vào nút “Người dùng”. Hệ thống sẽ truy cập vào bảng User và bảng UserDetail sau đó lấy thông tin gồm có email, họ tên, lớp, mã, tình trạng, số điện thoại sau đó hiển thị ra màn hình.  - Trợ giúp  Người dùng kích vào nút “Trợ giúp” trên màn hình. Hệ thống sẽ hiển thị màn hình gồm có 3 loại trợ giúp theo số điện thoại, email và tin nhắn.  Người dùng kích chọn một sau đó hệ thống sẽ chuyển đến cách trợ giúp đó.  - Sao lưu dữ liệu  Người dùng kích vào nút “Sao lưu” trên menu cài đặt. Hệ thống sẽ hiển thị màn hình gồm có trạng thái sao lưu và loại file.  Người dùng kích hoạt trạng thái sao lưu và chọn loại file sau đó kick “Sao lưu”. Hệ thống sẽ tạo file theo yêu cầu.  b) Luồng rẽ nhánh  Tại bất kỳ thời điểm nào nếu không thể kết nối với cơ sở dữ liệu thì usecase kết thúc.  4. Tiền điều kiện  Người dùng phải đăng nhập vào hệ thống trước.  5. Hậu điều kiện  Không có  6. Điều kiện tiên quyết  Không có  7. Điểm mở rộng  Không có |

#### Usecase Quản lý thời gian

A diagram with a black and white text

Description automatically generated with medium confidence

Hình 2.9. UseCase Quản lý thời gian

Bảng 2.4. Mô tả UseCase Quản lý thời gian

|  |
| --- |
| 1. Tên usecase: Usecase quản lý thời gian  2. Mô tả vắn tắt: Cho phép người dùng thêm, sửa, xóa, xem chi tiết lịch thời gian đã cài đặt.  3. Luồng sự kiện  a) Luồng cơ bản  - Thêm thời gian biểu.  Người dùng kích vào dấu “+” trên màn hình chính. Hệ thống sẽ hiển thị thông tin nhập gồm có tên, ngày, ghi chú, thời gian bắt đầu, thời gian kết thúc. Người dùng nhập thông tin sau đó kích vào nút “Thêm thời gian biểu”. Hệ thống sẽ cập nhật thông tin vào bảng Timetable và hiển thị lại danh sách thời gian biểu ra màn hình chính.  - Sửa thời gian biểu.  Người dùng kích nút “Sửa” trên một dòng trong danh sách thời gian biểu ở màn hình chính. Hệ thống sẽ lấy thông tin gồm có mã, tên ghi chú, ngày, ghi chú, thời gian bắt đầu, thời gian kết thúc của bảng “Timetable” sau đó hiển thị thông tin ra màn hình. Người dùng nhập thông tin mới cho các trường tên ghi chú, ngày, ghi chú, thời gian bắt đầu, thời gian kết thúc sau đó nhấn nút “Sửa thời gian biểu”. Hệ thống sẽ cập nhật thông tin vào bảng Timetable trong cơ sở dữ liệu và hiển thị lại danh sách ra màn hình.  - Xoá thời gian biểu.  Người dùng kích nút “Xoá” trên một dòng của danh sách thời gian biểu. Hệ thống sẽ hiển thị thông báo xác nhận xóa. Người dùng kích “Xác nhận”. Hệ thống sẽ xóa thông tin của công việc đó trên bảng Timetable của cơ sở dữ liệu.  - Xem chi tiết.  Người dùng kích nút “Xem chi tiết” trên một dòng của danh sách. Hệ thống sẽ lấy thông tin gồm có mã, tên, ngày, ghi chú, thời gian bắt đầu, thời gian kết thúc trên bảng Timetable của cơ sở dữ liệu sau đó hiển thị chi tiết ra màn hình.  b) Luồng rẽ nhánh  - Tại bước Xoá thời gian biểu nếu khi xác nhận người dùng kích chọn “Huỷ” thì hệ thống sẽ huỷ bỏ thao tác xóa và hiển thị lại danh sách ghi chú công việc ra màn hình.  - Tại bất kỳ thời điểm nào nếu không thể kết nối với cơ sở dữ liệu thì usecase kết thúc.  4. Tiền điều kiện  Phải thực hiện chức năng đăng nhập trước.  5. Hậu điều kiện  Không có  6. Điều kiện tiên quyết  Không có  7. Điểm mở rộng  Hiện thông báo khi đến giờ làm việc và sắp kết thúc công việc |

#### Usecase Xem lịch

A graph paper with a black and white text

Description automatically generated

Hình 2.10. UseCase Xem lịch

Bảng 2.5. Mô tả UseCase Xem lịch

|  |
| --- |
| 1. Tên usecase: Usecase xem lịch  2. Mô tả vắn tắt  Cho phép người dùng xem thông tin chi tiết công việc vào ngày tháng.  3. Luồng sự kiện  a) Luồng cơ bản  Người dùng kick vào nút “Xem lịch”. Hệ thống sẽ hiển thị chi tiết ngày tháng năm và các hoạt động.  b) Luồng rẽ nhánh  Tại bất kỳ thời điểm nào nếu không thể kết nối với cơ sở dữ liệu thì usecase kết thúc.  4. Tiền điều kiện  Không có  5. Hậu điều kiện  Không có  6. Điều kiện tiên quyết  Không có  7. Điểm mở rộng  Không có |

#### Usecase Hẹn giờ

A white background with black text

Description automatically generated

Hình 2.11. UseCase Hẹn giờ

Bảng 2.6. Mô tả UseCase Hẹn giờ

|  |
| --- |
| 1. Tên usecase: Usecase hẹn giờ  2. Mô tả vắn tắt  Cho phép người dùng cài đặt một khoảng thời gian hẹn giờ để thông báo hoặc thực hiện hành động nhắc nhở.  3. Luồng sự kiện  a) Luồng cơ bản  - Người dùng kích vào nút “Hẹn giờ” trên màn hình chính.  - Hệ thống hiển thị giao diện nhập thời gian hẹn (giờ:phút:giây) và nội dung thông báo.  - Người dùng nhập thời gian và nội dung nhắc, sau đó nhấn nút “Đặt hẹn giờ”.  - Hệ thống lưu thông tin hẹn giờ vào cơ sở dữ liệu và bắt đầu đếm ngược.          - Khi hết thời gian, hệ thống sẽ hiển thị thông báo nhắc nhở.  b) Luồng rẽ nhánh  - Nếu thời gian nhập không hợp lệ (ví dụ: âm, quá lớn), hệ thống sẽ báo lỗi và yêu cầu nhập lại.  - Nếu không thể kết nối với cơ sở dữ liệu, usecase kết thúc.  4. Tiền điều kiện  Người dùng đã đăng nhập hệ thống.  5. Hậu điều kiện  Thông tin hẹn giờ được lưu và thực hiện đúng thời điểm.  6. Điều kiện tiên quyết  Không có  7. Điểm mở rộng  Có thể tích hợp tính năng phát âm thanh khi kết thúc thời gian. |

#### Usecase Bấm giờ

A diagram with a black and white text

Description automatically generated with medium confidence

Hình 2.12. UseCase Bấm giờ

Bảng 2.7. Mô tả UseCase Bấm giờ

|  |
| --- |
| 1. Tên usecase: Usecase bấm giờ  2. Mô tả vắn tắt  Cho phép người dùng đo thời gian trôi qua giữa thời điểm bắt đầu và dừng.  3. Luồng sự kiện  a) Luồng cơ bản  - Người dùng kích vào nút “Bấm giờ”.  -  Hệ thống hiển thị giao diện với các nút: “Bắt đầu”, “Tạm dừng”, “Tiếp tục”, “Dừng”, “Đặt lại”.  - Người dùng nhấn “Bắt đầu”, hệ thống bắt đầu đếm thời gian.  - Người dùng có thể tạm dừng, tiếp tục hoặc dừng hoàn toàn quá trình bấm giờ.  - Khi nhấn “Dừng”, hệ thống hiển thị tổng thời gian đã bấm và lưu lại nếu cần.  b) Luồng rẽ nhánh  - Nếu người dùng chưa nhấn “Bắt đầu” mà nhấn “Tạm dừng” hoặc “Dừng” thì hệ thống không thực hiện thao tác.  - Nếu mất kết nối, hệ thống cảnh báo và yêu cầu khởi động lại.  4. Tiền điều kiện  Người dùng đã đăng nhập.  5. Hậu điều kiện  Thời gian bấm giờ có thể lưu lại (nếu người dùng chọn).  6. Điều kiện tiên quyết  Không có  7. Điểm mở rộng  Có thể thêm tính năng lưu lịch sử các phiên bấm giờ. |

#### Usecase Sao lưu

A diagram of a graph

Description automatically generated with medium confidence

Hình 2.13. UseCase Sao lưu

Bảng 2.8. Mô tả UseCase Sao lưu

|  |
| --- |
| 1. Tên usecase: Usecase sao lưu  2. Mô tả vắn tắt  Cho phép người dùng sao lưu toàn bộ dữ liệu tài khoản và thời gian biểu thành file.  3. Luồng sự kiện  a) Luồng cơ bản  - Người dùng vào menu “Cài đặt” và chọn “Sao lưu dữ liệu”.  - Hệ thống hiển thị giao diện chọn loại dữ liệu cần sao lưu (toàn bộ hoặc chỉ thời gian biểu) và định dạng file (.json, .csv, .txt).  - Người dùng chọn xong và nhấn nút “Sao lưu”.  - Hệ thống tạo file, hiển thị thông báo “Sao lưu thành công” và đường dẫn tải file.  b) Luồng rẽ nhánh  - Nếu không chọn loại dữ liệu hoặc định dạng file, hệ thống báo lỗi.  - Nếu mất kết nối, quá trình sao lưu bị hủy.  4. Tiền điều kiện  Người dùng đã đăng nhập.  5. Hậu điều kiện  File sao lưu được tạo và lưu vào bộ nhớ máy hoặc cloud (tuỳ cấu hình).  6. Điều kiện tiên quyết  Không có  7. Điểm mở rộng  Không có |

#### Usecase Trợ giúp

A graph with a white oval with black text

Description automatically generated

Hình 2.14. UseCase Trợ giúp

Bảng 2.9. Mô tả UseCase Trợ giúp

|  |
| --- |
| 1. Tên usecase: Usecase trợ giúp  2. Mô tả vắn tắt  Cung cấp nhiều cách thức hỗ trợ người dùng khi gặp sự cố hoặc thắc mắc.  3. Luồng sự kiện  a) Luồng cơ bản  -  Người dùng vào mục “Trợ giúp” từ menu chính.  -  Hệ thống hiển thị các tuỳ chọn: Gọi điện thoại, Gửi email, Gửi tin nhắn hỗ trợ.  -  Người dùng chọn một trong ba cách:  + Nếu gọi điện: hệ thống hiển thị số điện thoại hỗ trợ.  + Nếu gửi email: hiển thị form nhập nội dung, nhấn “Gửi”.  + Nếu gửi tin nhắn: hiển thị khung chat trực tiếp hoặc mẫu tin nhắn.  -  Hệ thống thông báo “Yêu cầu trợ giúp đã được gửi”.  b) Luồng rẽ nhánh  -  Nếu người dùng không nhập nội dung email/tin nhắn, hệ thống không gửi đi.  -  Nếu mất kết nối, quá trình trợ giúp bị hoãn.  4. Tiền điều kiện  Người dùng đã đăng nhập.  5. Hậu điều kiện  Yêu cầu trợ giúp được ghi nhận hoặc phản hồi.  6. Điều kiện tiên quyết  Không có  7. Điểm mở rộng  Không có |

### Thiết kế hệ thống

Ứng dụng được thiết kế giao diện gồm 2 loại màn hình chính:

* Màn hình đăng ký, đăng nhập
* Màn hình chính:
* Màn hình Quản lý thời gian
* Màn hình Thống kê
* Màn hình Cài đặt
* Màn hình Xem lịch
* Màn hình Hẹn giờ
* Màn hình Bấm giờ

**a. Màn hình đăng ký, đăng nhập**

Màn hình đăng nhập, đăng ký thực hiện chức năng đăng ký tài khoản và đăng nhập vào hệ thống.

Giao diện màn hình đăng nhập có 2 thông tin chính: Email và Mật khẩu. Người dùng cần nhập tên tài khoản và mật khẩu.

Nếu người dùng chưa có tài khoản thì cần tiến hành đăng ký, giao diện đăng ký gồm năm thông tin cần nhập như sau:

* + Email
  + Mật khẩu
  + Họ tên
  + Lớp
  + Số điện thoại

Sau khi đăng ký tài khoản, nếu người dùng đăng nhập thành công thì sẽ hiển thị ra màn tra chính bao gồm các chức năng nghiệp vụ.

**b. Màn hình chính**

Sau khi đăng nhập thành công, hệ thống sẽ hiển thị màn hình chính gồm 6 chức năng quản lý cho người quản lý gồm Quản lý thời gian, Thống kê, Cài đặt, Xem lịch, Hẹn giờ, Bấm giờ.

**Quản lý thời gian**

Người dùng sẽ được chuyển tới màn hình Quản lý thời gian, tại đây người dùng có thể theo dõi, quản lý lịch trình và các hoạt động theo thời gian cá nhân. Người dùng có thể thêm, chỉnh sửa, xóa hoặc xem chi tiết các sự kiện trong lịch trình cá nhân.

**Thống kê**

Người dùng sẽ được chuyển tới màn hình Thống kê, tại đây người dùng có thể xem tổng hợp các dữ liệu liên quan đến công việc hoàn thành hoặc chưa thành hoàn thành, và biểu đồ tỷ lệ hoàn thành nhiệm vụ theo ngày, tuần, tháng.

**Người dùng**

Người dùng sẽ được chuyển tới màn hình Người dùng, tại đây người dùng có thể thấy các thông tin cá nhân như họ tên, lớp, số điện thoại,… và các chức năng khác như sao lưu, trợ giúp

**Xem lịch**

Người dùng sẽ được chuyển tới màn hình Xem lịch, tại đây người dùng có thể xem lịch theo ngày, tuần, tháng.

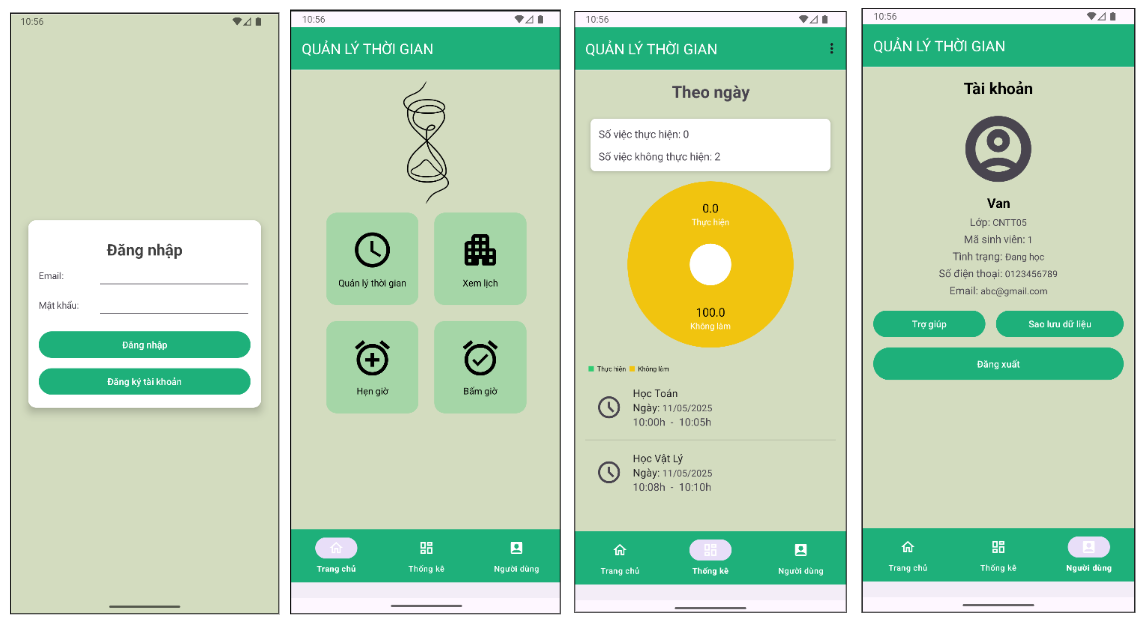
**Hẹn giờ**

Người dùng sẽ được chuyển tới màn hình Hẹn giờ, tại đây người dùng có thể cài đặt một khoảng thời gian hẹn giờ để thông báo hoặc thực hiện hành động nhắc nhở.

**Bấm giờ**

Người dùng sẽ được chuyển tới màn hình Bấm giờ, tại đây người dùng có thể sử dụng chức năng đếm thời gian cho các hoạt động như tập thể dục, làm bài thi, thi đấu,… Người dùng có thể bắt đầu, tạm dừng hoặc đặt lại thời gian bấm giờ.

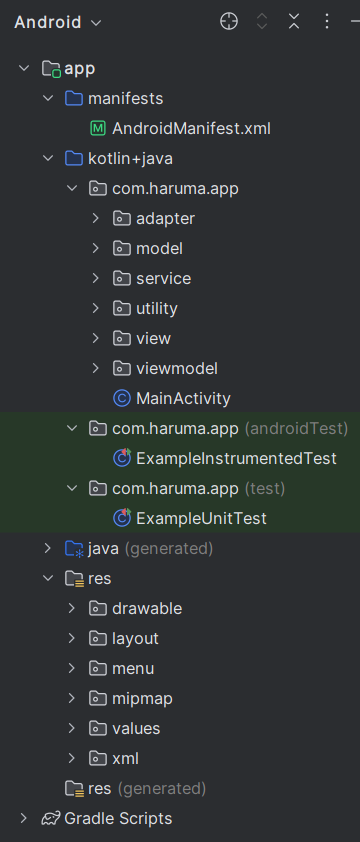
**Tổng quan giao diện**



Hình 2.15. Giao diện tổng quan

## Thực hiện bài toán

**Cấu trúc tổng quát:**

****

Hình 2.16. Cấu trúc tổng quát

### Đăng ký

* Màn hình người dùng:

A screenshot of a screen

Description automatically generated

Hình 2.17. Giao diện đăng ký

**Nhiệm vụ của Activity:**

* Tạo tài khoản mới cho người dùng để họ có thể sử dụng hệ thống, ứng dụng hoặc dịch vụ. Khi người dùng đăng ký, hệ thống sẽ lưu lại thông tin cá nhân cơ bản để xác định danh tính và quản lý hoạt động của họ sau này.

**Thuật toán chính:**

* Kiểm tra ngoại lệ và thông báo lỗi

|  |
| --- |
| private boolean validateEmailAndPassword(String email, String password) throws Exception {  if (password == null || password.isEmpty()) {  throw new Exception("Mật khẩu không được rỗng");  }  if (email == null || email.isEmpty()) {  throw new Exception("Email không được rỗng");  }  if (password.length() < 6) {  throw new Exception("Mật khẩu không được ít hơn 6 kí tự");  }  if (!Pattern.*compile*("^.+@.+\\..+$").matcher(email).matches()) {  throw new Exception("Email không hợp lệ");  }  return true; }  private void showRegistrationError(String errorMessage) {  Toast.*makeText*(this.context, errorMessage, Toast.*LENGTH\_LONG*).show(); } private boolean isEmailExists(String email) {return this.databaseHelper.isEmailExists(email); } private boolean isValidPhoneNumber(String phoneNumber) {  *// Kiểm tra xem số điện thoại có phải là 10 chữ số hay không* if (phoneNumber != null && phoneNumber.length() == 10) {  try {  Long.*parseLong*(phoneNumber); *// Kiểm tra nếu là số* return true;  } catch (NumberFormatException e) {  *// Nếu không phải là số hợp lệ* return false;  }  }  return false; *// Trả về false nếu không phải 10 chữ số* } |

* Xử lý ngoại lệ và đăng ký

|  |
| --- |
| public void onRegister() {  try {  User user = new User(this.getEmail(), this.getPassword());  if (this.getEmail() == null || this.getEmail().isEmpty()) {  showRegistrationError("Email không được rỗng");  return;  }  if (!Pattern.*compile*("^.+@.+\\..+$").matcher(this.getEmail()).matches()) {  showRegistrationError("Email không hợp lệ");  return;  }  if (isEmailExists(this.getEmail())) {  showRegistrationError("Email đã tồn tại");  return;  }  if (this.getPassword() == null || this.getPassword().isEmpty()) {  showRegistrationError("Mật khẩu không được rỗng");  return;  }  if (this.getPassword().length() < 6) {  showRegistrationError("Mật khẩu không được ít hơn 6 kí tự");  return;  }   if (this.getFullName() == null || this.getFullName().isEmpty()) {  showRegistrationError("Họ và tên không được rỗng");  return;}   if (this.getClassName() == null || this.getClassName().isEmpty()) {  showRegistrationError("Lớp không được rỗng");  return;}   if (this.getPhoneNumber() == null || this.getPhoneNumber().isEmpty()) {  showRegistrationError("Số điện thoại không được rỗng");  return;}    if (!isValidPhoneNumber(this.getPhoneNumber())) {  showRegistrationError("Số điện thoại phải có 10 số");  return;  }  if (this.validateEmailAndPassword(user.getEmail(), user.getPassword())) {  makeToast("Đăng ký thành công");  boolean state = this.databaseHelper.registerUser(user.getEmail(), user.getPassword(), this.getFullName(), this.getClassName(), this.getPhoneNumber());  ExceptionHelper.*assertTest*(state, "Đăng ký tài khoản thất bại");  Objects.*requireNonNull*(this.callback.get("onRegister")).run();  }  else {  makeToast("Đăng ký thất bại");  }  } catch (Exception e) {  makeToast("Lỗi xảy ra: " + e.getMessage());  } } |

### Đăng nhập

* Màn hình người dùng

Screens screenshot of a screen

Description automatically generated

Hình 2.18. Giao diện đăng nhập

**Nhiệm vụ của activity:**

* Lấy danh sách trong database và so sánh với email và mật khẩu sau đó cho phép đăng nhập vào hệ thống.
* Người dùng đăng nhập vào hệ thống để sử dụng hệ thống quản lý thời gian với email và mật khẩu đã được đăng ký trước.

**Thuật toán chính:**

* Hàm đăng nhâp

|  |
| --- |
| public void onLogin() {  try {  User user = new User(this.getEmail(), this.getPassword());  if (this.validateEmailAndPassword(user.getEmail(), user.getPassword())) {  User currentUser = this.databaseHelper.loginUser(user.getEmail(), user.getPassword());  ExceptionHelper.*assertTest*(currentUser != null, "Không tìm thấy tài khoản");  UserSessionManager.*getInstance*().saveUser(currentUser);  makeToast("Đăng nhập thành công");  Objects.*requireNonNull*(this.callback.get("onLogin")).run();  }  else {  makeToast("Đăng nhập thất bại");  }  } catch (Exception e) {  makeToast("Lỗi xảy ra: " + e.getMessage());  }  } |

* Hàm kiểm tra dữ liệu vào

|  |
| --- |
| private boolean validateEmailAndPassword(String email, String password) throws Exception {  if (password == null || password.isEmpty()) {  throw new Exception("Mật khẩu không được rỗng");  }  if (email == null || email.isEmpty()) {  throw new Exception("Email không được rỗng");  }  if (password.length() < 6) {  throw new Exception("Mật khẩu không được ít hơn 6 kí tự");  }  if (!Pattern.*compile*("^.+@.+\\..+$").matcher(email).matches()) {  throw new Exception("Email không hợp lệ");  }  return true;  } |

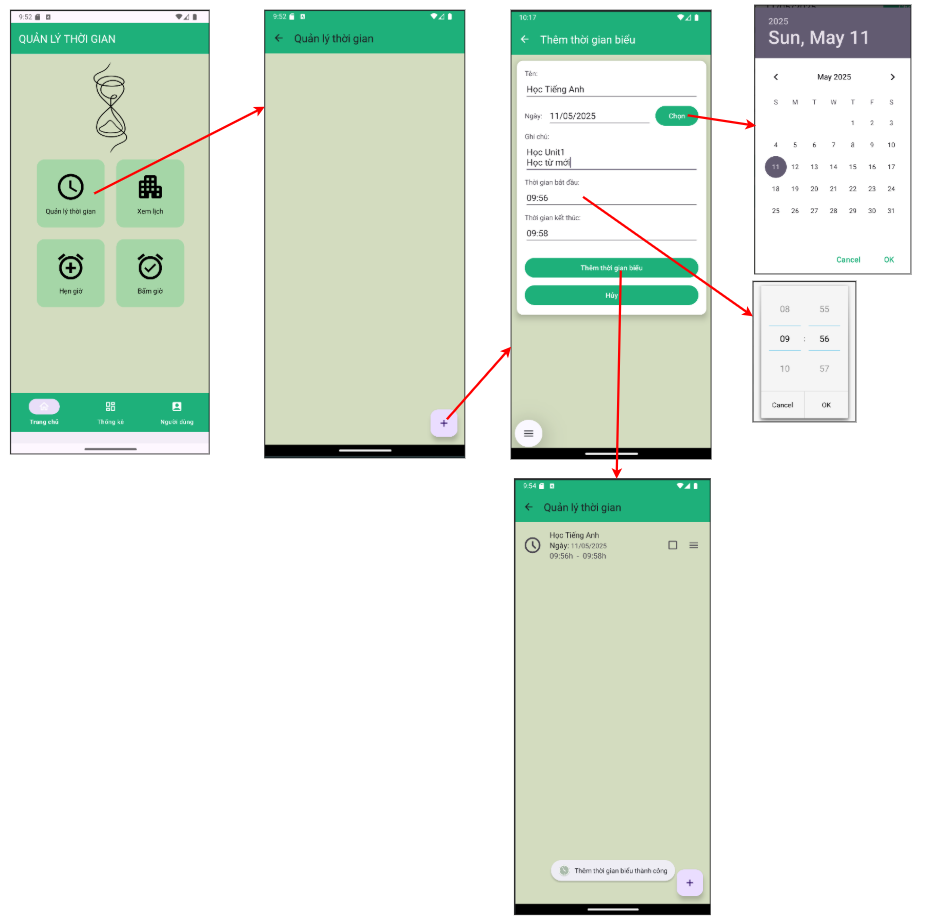
* Xử lý gọi giao diện của đăng nhập và lưu vào Map

|  |
| --- |
| ActivityMainBinding mainBinding = DataBindingUtil.*setContentView*(this,  R.layout.*activity\_main*);  Map<String, Callback> callback = new HashMap<>();  callback.put("onLogin", () -> {  Intent intent = new Intent(MainActivity.this, RootActivity.class);  startActivity(intent);  });  callback.put("onRegister", () -> {  Intent intent = new Intent(MainActivity.this, RegisterActivity.class);  startActivity(intent);  });  LoginViewModel loginViewModel = new LoginViewModel(this, callback);  mainBinding.setLoginViewModel(loginViewModel); |

### Quản lý thời gian

#### Chức năng thêm thời gian biểu

* Màn hình người dùng



Hình 2.19. Giao diện Quản lý thời gian

**Nhiệm vụ Activity:**

* Cho phép người dùng dễ dàng thêm các khoảng thời gian cần ghi nhớ các công việc vào ứng dụng, hỗ trợ việc quản lý thời gian cá nhân một cách trực quan và hiệu quả.

**Thuật toán chính:**

* Thêm thời gian biểu

|  |
| --- |
| @Override  protected void onCreate(Bundle savedInstanceState) {     super.onCreate(savedInstanceState);     setContentView(R.layout.*activity\_add*);     ActivityAddBinding mainBinding = DataBindingUtil.*setContentView*(this,             R.layout.*activity\_add*);     Map<String, Callback> callback = new HashMap<>();     callback.put("onAdd", () -> {         DatabaseHelper db = new DatabaseHelper(this);         List<Timetable> myList = db.getAllTimeTable();         AdapterSessionManager.*getInstance*().getCustomAdapter().setList(myList);         AdapterSessionManager.*getInstance*().getCustomAdapter().notifyDataSetChanged();         finish();     });  Toolbar toolbar = findViewById(R.id.*toolbar*);  setSupportActionBar(toolbar);  ActionBar actionBar = getSupportActionBar();  if (actionBar != null) {     actionBar.setDisplayHomeAsUpEnabled(true);     Drawable backArrow = ContextCompat.*getDrawable*(this, R.drawable.*ic\_back\_arrow*);     if (backArrow != null) {         backArrow.setColorFilter(Color.*WHITE*, PorterDuff.Mode.*SRC\_ATOP*);         actionBar.setHomeAsUpIndicator(backArrow);     }     SpannableString title = new SpannableString("Thêm thời gian biểu");     title.setSpan(new ForegroundColorSpan(Color.*WHITE*), 0, title.length(), Spannable.*SPAN\_INCLUSIVE\_INCLUSIVE*);     actionBar.setTitle(title);  }  callback.put("onCancel", () -> {     finish();  });  } |

* Xử lý sự kiện thêm thời gian biểu:

|  |
| --- |
| public void onAdd() {     try {         if (isValidInput()) {             if (!isStartTimeBeforeEndTime(startTime, endTime)) {                 makeToast("Thời gian bắt đầu phải nhỏ hơn thời gian kết thúc.");                 return;             }             DatabaseHelper databaseHelper = new DatabaseHelper(context);             databaseHelper.addTimeTable(name, date, note, startTime, endTime, false, 0);             makeToast("Thêm thời gian biểu thành công");             Objects.*requireNonNull*(this.callback.get("onAdd")).run();         } else {             makeToast("Vui lòng điền đầy đủ thông tin.");         }     } catch (Exception e) {         makeToast("Lỗi xảy ra: " + e.getMessage());     }  } |

* Xử lý sự kiện thông báo

|  |
| --- |
| public static void showStartingSoonNotification(Context context, String taskName, long timeRemainingMinutes) {     NotificationCompat.Builder builder = new NotificationCompat.Builder(context, *CHANNEL\_ID*)             .setSmallIcon(android.R.drawable.*ic\_dialog\_info*)  .setContentTitle("Sắp đến giờ làm việc")             .setContentText("Công việc " + taskName + " sẽ bắt đầu trong " + timeRemainingMinutes + " phút")             .setPriority(NotificationCompat.*PRIORITY\_HIGH*)             .setAutoCancel(true);     NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.*NOTIFICATION\_SERVICE*);     notificationManager.notify(taskName.hashCode(), builder.build());  }  public static void showEndingSoonNotification(Context context, String taskName, long timeRemainingMinutes) {     NotificationCompat.Builder builder = new NotificationCompat.Builder(context, *CHANNEL\_ID*)             .setSmallIcon(android.R.drawable.*ic\_dialog\_alert*)  .setContentTitle("Sắp hết giờ làm việc")             .setContentText("Công việc " + taskName + " sẽ kết thúc trong " + timeRemainingMinutes + " phút")             .setPriority(NotificationCompat.*PRIORITY\_HIGH*)             .setAutoCancel(true);     NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.*NOTIFICATION\_SERVICE*);     notificationManager.notify((taskName + "\_end").hashCode(), builder.build());  } |

* Phương thức chọn lịch

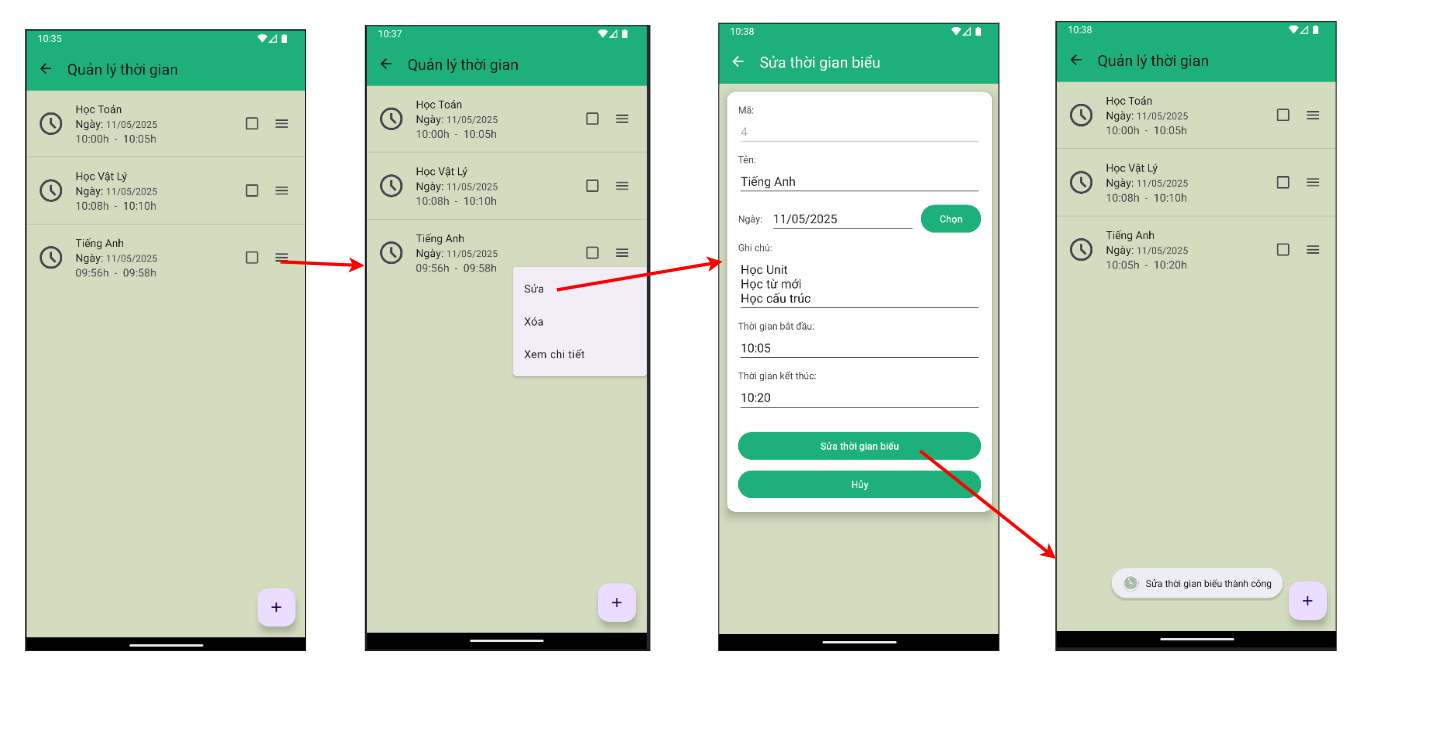
|  |
| --- |
| Button btnPickDate = findViewById(R.id.*btnPickDate*);  EditText edtDate = findViewById(R.id.*edtDate*);  AddViewModel addViewModel = new AddViewModel(this, callback);  btnPickDate.setOnClickListener(v -> {     final Calendar calendar = Calendar.*getInstance*();     int year = calendar.get(Calendar.*YEAR*);     int month = calendar.get(Calendar.*MONTH*);     int day = calendar.get(Calendar.*DAY\_OF\_MONTH*);     DatePickerDialog datePickerDialog = new DatePickerDialog(             AddActivity.this,             (view, selectedYear, selectedMonth, selectedDay) -> {                 String selectedDate = String.*format*("%02d/%02d/%04d", selectedDay, selectedMonth + 1, selectedYear);                 addViewModel.setDate(selectedDate);                 edtDate.setText(selectedDate);             },             year, month, day     );     datePickerDialog.show();  });  mainBinding.setAddViewModel(addViewModel);  mainBinding.setLifecycleOwner(this); |

* Phương thức chọn thời gian:

|  |
| --- |
| private void showTimePickerDialog(EditText targetEditText, boolean isStartTime) {     int hour = 0;     int minute = 0;  String timeStr = targetEditText.getText().toString();     if (timeStr != null && timeStr.matches("\\d{2}:\\d{2}")) {         String[] parts = timeStr.split(":");         hour = Integer.*parseInt*(parts[0]);         minute = Integer.*parseInt*(parts[1]);     } else {  final Calendar calendar = Calendar.*getInstance*();         hour = calendar.get(Calendar.*HOUR\_OF\_DAY*);         minute = calendar.get(Calendar.*MINUTE*);     }     TimePickerDialog timePickerDialog = new TimePickerDialog(             this,             android.R.style.*Theme\_Holo\_Light\_Dialog\_NoActionBar*,             (view, hourOfDay, minuteOfHour) -> {                 String selectedTime = String.*format*(Locale.*getDefault*(), "%02d:%02d", hourOfDay, minuteOfHour);                 targetEditText.setText(selectedTime);  },             hour,             minute,             true     );     timePickerDialog.show();  } |

#### Chức năng sửa thời gian biểu

* Màn hình người dùng

****

Hình 2.20. Giao diện sửa thời gian biểu

**Nhiệm vụ Activity:**

* Cho phép người dùng sửa lại nội dung ghi nhớ, thời gian ngày tháng dễ dàng.

**Thuật toán chính:**

* Sửa thời gian biểu:

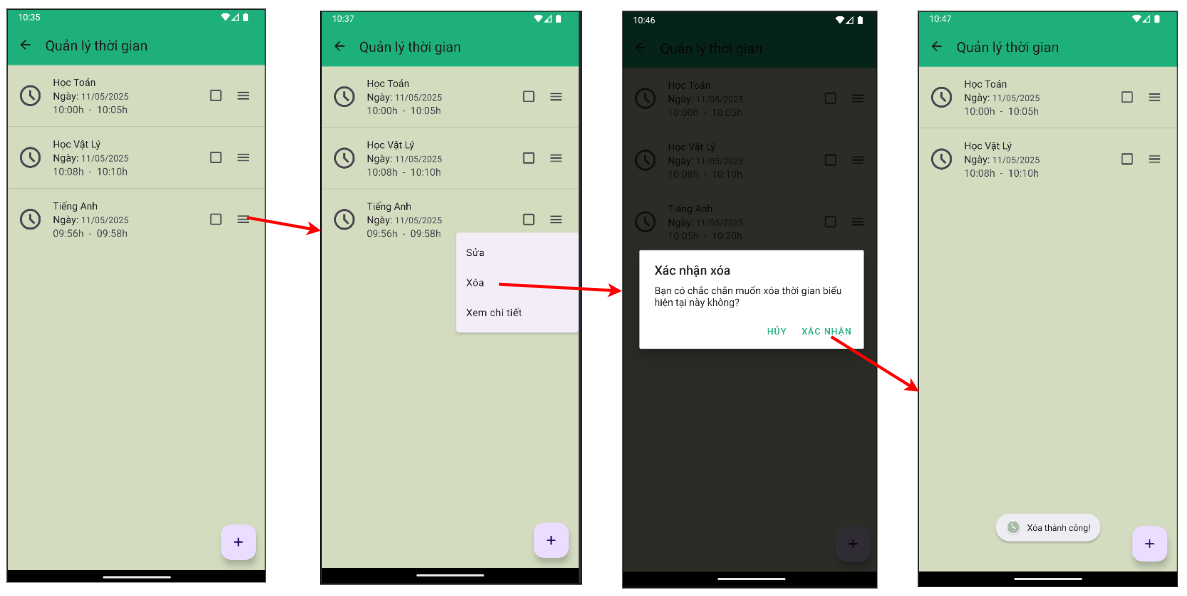
|  |
| --- |
| protected void onCreate(Bundle savedInstanceState) {     super.onCreate(savedInstanceState);     setContentView(R.layout.*activity\_edit*);     ActivityEditBinding mainBinding = DataBindingUtil.*setContentView*(this,             R.layout.*activity\_edit*);     Map<String, Callback> callback = new HashMap<>();     callback.put("onEdit", () -> {         DatabaseHelper db = new DatabaseHelper(this);         List<Timetable> myList = db.getAllTimeTable();         AdapterSessionManager.*getInstance*().getCustomAdapter().setList(myList);         AdapterSessionManager.*getInstance*().getCustomAdapter().notifyDataSetChanged();         finish();  **});**  **}** |

* Xử lý sự kiện bắt lỗi

|  |
| --- |
| public void onEdit() {     try {         if (isValidInput()) {             if (!isStartTimeBeforeEndTime(startTime, endTime)) {                 makeToast("Thời gian bắt đầu phải nhỏ hơn thời gian kết thúc.");                 return;             }             db.updateTimeTable(timeTable.getTimeTableId(), name, date, note, startTime, endTime, timeTable.getStatus());             makeToast("Sửa thời gian biểu thành công");             Objects.*requireNonNull*(this.callback.get("onEdit")).run();         } else {             makeToast("Vui lòng điền đầy đủ thông tin.");         }     } catch (Exception e) {         makeToast("Lỗi xảy ra: " + e.getMessage());     }  } |

#### Chức năng xóa thời gian biểu

* Màn hình người dùng



Hình 2.21. Giao diện xóa thời gian biểu

**Nhiệm vụ Acticity:**

* Cho phép người dùng xóa dữ liệu ghi nhớ không cần thiết.

**Thuật toán chính:**

* Xóa thời gian biểu:

|  |
| --- |
| myCallback.put("onDelete", (id) -> {     new AlertDialog.Builder(getContext())             .setTitle("Xác nhận xóa")             .setMessage("Bạn có chắc chắn muốn xóa thời gian biểu hiện tại này không?")             .setPositiveButton("Xác nhận", (dialog, which) -> {                 try {                     db.deleteTimetable(id);                     List<Timetable> updatedList = db.getAllTimeTable();                     CustomAdapter adapter = AdapterSessionManager.*getInstance*().getCustomAdapter();                     if (adapter != null) {                         adapter.setList(updatedList);                     }                     Toast.*makeText*(getContext(), "Xóa thành công!", Toast.*LENGTH\_SHORT*).show();                 } catch (Exception e) {                     Toast.*makeText*(getContext(), "Lỗi khi xóa: " + e.getMessage(), Toast.*LENGTH\_SHORT*).show();                 }             })             .setNegativeButton("Hủy", (dialog, which) -> {                 dialog.dismiss();             })             .show();  **});** |

#### Chức năng xem chi tiết thời gian biểu

* Màn hình người dùng

Screens screenshot of a phone

Description automatically generated

Hình 2.22. Giao diện Xem chi tiết thời gian biểu

**Nhiệm vụ Activity:**

* Cho phép người dùng xem chi tiết ghi nhớ thời gian biểu.

**Thuật toán chính:**

* Xem chi tiết

|  |
| --- |
| @Override  protected void onCreate(Bundle savedInstanceState) {     super.onCreate(savedInstanceState);     setContentView(R.layout.*activity\_detail*);     ActivityDetailBinding mainBinding = DataBindingUtil.*setContentView*(this,             R.layout.*activity\_detail*);     Intent intent = getIntent();     int id = intent.getIntExtra("id", 0);  DetailViewModel detailActivityViewModel = new DetailViewModel(this.getApplicationContext(), id, () -> {         finish();     });     mainBinding.setDetailViewModel(detailActivityViewModel);     mainBinding.setLifecycleOwner(this);     Toolbar toolbar = findViewById(R.id.*toolbar*);     setSupportActionBar(toolbar);     ActionBar actionBar = getSupportActionBar();     if (actionBar != null) {         actionBar.setDisplayHomeAsUpEnabled(true);         Drawable backArrow = ContextCompat.*getDrawable*(this, R.drawable.*ic\_back\_arrow*);         if (backArrow != null) {             backArrow.setColorFilter(Color.*WHITE*, PorterDuff.Mode.*SRC\_ATOP*);             actionBar.setHomeAsUpIndicator(backArrow);         }         SpannableString title = new SpannableString("Chi tiết thời gian biểu");         title.setSpan(new ForegroundColorSpan(Color.*WHITE*), 0, title.length(), Spannable.*SPAN\_INCLUSIVE\_INCLUSIVE*);         actionBar.setTitle(title);     }  **}** |

|  |
| --- |
| public DetailViewModel(Context context, int id, Callback callback) {     this.context = context;     this.db = new DatabaseHelper(this.context);     this.timetable = db.findTimeTableById(id);     this.callback = callback;  } |

### Hẹn giờ

* Màn hình người dùng

Screens screenshot of a screenshot of a phone

Description automatically generated

Hình 2.23. Giao diện Hẹn giờ

**Nhiệm vụ của Activity:**

* Giúp người dùng thực hiện một hành động trong 1 khoàng thời gian nhất định

**Thuật toán chính:**

* Xử lí sự kiện kích nút bắt đầu/ Dừng lại

|  |
| --- |
| private void setupButtonListeners() {  btnStart.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  if (!isTimerRunning) {  if (timeLeftInMillis == 0L) {  *// Lấy thời gian từ NumberPicker* int hours = hourPicker.getValue();  int minutes = minutePicker.getValue();  int seconds = secondPicker.getValue();   *// Kiểm tra nếu thời gian là 0* if (hours == 0 && minutes == 0 && seconds == 0) {  Toast.*makeText*(context, "Vui lòng chọn thời gian hẹn giờ", Toast.*LENGTH\_SHORT*).show();  return;  }if (etContent.getText().toString().trim().isEmpty()) {  Toast.*makeText*(context, "Vui lòng nhập nội dung hẹn giờ", Toast.*LENGTH\_SHORT*).show();  return;  } timeLeftInMillis = (((hours \* 60 \* 60) + (minutes \* 60) + seconds) \* 1000L);  originalTimeInMillis = timeLeftInMillis;  }  startTimer();  btnStart.setText("Dừng lại");  btnPause.setEnabled(true);  } else {  stopTimer();  resetTimer();  btnStart.setText("Bắt đầu");  btnPause.setEnabled(false);  }  }  });  *// Xử lý sự kiện nút Tạm dừng* btnPause.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  if (isTimerRunning) {  pauseTimer();  btnPause.setText("Tiếp tục"); btnPause.setBackgroundColor(Color.*parseColor*("#4CAF50")); *// màu xanh lá cây* } else {  startTimer();  btnPause.setBackgroundColor(Color.*parseColor*("#4CAF50"));  btnPause.setText("Tạm dừng");  }  }  }); } |

* Xử lí sự kiện khi đang hẹn giờ

|  |
| --- |
| private void startTimer() {  *// Khóa trường nhập nội dung khi đồng hồ đếm ngược bắt đầu* etContent.setEnabled(false);  hourPicker.setEnabled(false);  minutePicker.setEnabled(false);  secondPicker.setEnabled(false);   timer = new CountDownTimer(timeLeftInMillis, 1000) {  @Override  public void onTick(long millisUntilFinished) {  timeLeftInMillis = millisUntilFinished;  updateTimerText();  }   @Override  public void onFinish() {  isTimerRunning = false;  isPaused = false;  btnStart.setText("Bắt đầu");  btnPause.setEnabled(false);  btnPause.setText("Tạm dừng");   *// Mở khóa trường nhập khi đồng hồ kết thúc* etContent.setEnabled(true);  hourPicker.setEnabled(true);  minutePicker.setEnabled(true);  secondPicker.setEnabled(true);  *// Hiển thị thông báo nâng cao khi hẹn giờ kết thúc* showEnhancedNotification();  Log.*d*(*TAG*, "Timer finished, notification should be displayed");  *// Reset timer* resetTimer();  }  }.start();  isTimerRunning = true;  isPaused = false; } |

* Xử lí sự kiện khi kích nút tạm dừng

|  |
| --- |
| Private void pauseTimer() {  if (timer != null) {  timer.cancel();  }  isTimerRunning = false;  isPaused = true;} |

* Xử lí sự kiện thông báo

|  |
| --- |
| private void createNotificationChannel() {  try {  if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  String name = "Timer Notification";  String descriptionText = "Channel for Timer App notifications";  int importance = NotificationManager.*IMPORTANCE\_HIGH*;   *// Tạo kênh với cấu hình cao nhất có thể* NotificationChannel channel = new NotificationChannel(*CHANNEL\_ID*, name, importance);  channel.setDescription(descriptionText);  channel.enableVibration(true);  channel.enableLights(true); channel.setLightColor(android.graphics.Color.*RED*); channel.setLockscreenVisibility(NotificationCompat.*VISIBILITY\_PUBLIC*);  channel.setBypassDnd(true); *// Cho phép thông báo vượt qua chế độ Không làm phiền* channel.setImportance(NotificationManager.*IMPORTANCE\_HIGH*);  channel.setShowBadge(true);  *// Log để debug* Log.*d*(*TAG*, "Đang tạo kênh thông báo: " + *CHANNEL\_ID*);   NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.*NOTIFICATION\_SERVICE*);  if (notificationManager != null) {  notificationManager.createNotificationChannel(channel);  Log.*d*(*TAG*, "Đã tạo kênh thông báo thành công");  } else {  Log.*e*(*TAG*, "NotificationManager là null");  }  }  } catch (Exception e) {  Log.*e*(*TAG*, "Lỗi khi tạo kênh thông báo: " + e.getMessage());  e.printStackTrace();  } } |

### Bấm giờ

* Màn hình người dùng

A screenshot of a cell phone

Description automatically generated

Hình 2.24. Giao diện Bấm giờ

**Nhiệm vụ của activity:**

* Cho phép người dùng đếm thời gian khi thực hiện một hoạt động nào đó. Chức năng này có thể sử dụng để hiển thị theo vòng để ứng dụng trong quá trình sử dụng để đo trong quá trình chạy đua nhiều người.

**Thuật toán chính:**

* Hàm xử lý cập nhật thời gian và format hiển thị

|  |
| --- |
| private Runnable updateTimerRunnable = new Runnable() {  @Override  public void run() {  timeInMillis = System.*currentTimeMillis*() - startTime;  int minutes = (int) (timeInMillis / 1000) / 60;  int seconds = (int) (timeInMillis / 1000) % 60;  int milliseconds = (int) (timeInMillis % 1000) / 10;  String time = String.*format*("%02d:%02d,%02d", minutes, seconds, milliseconds);  timerTextView.setText(time);  handler.postDelayed(this, 10);  }  }; |

* Hàm xử lý sự kiện và thiết lập activity hoạt động

|  |
| --- |
| @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_stopwatch*);  *// Thiết lập Toolbar và tiêu đề*  Toolbar toolbar = findViewById(R.id.*toolbar*); *// Lấy Toolbar từ layout*  setSupportActionBar(toolbar); *// Đặt Toolbar làm ActionBar*  setTitle("Hẹn giờ"); *// Thiết lập tiêu đề cho Toolbar*  *// Xử lý sự kiện quay lại*  if (getSupportActionBar() != null) {  getSupportActionBar().setDisplayHomeAsUpEnabled(true); *// Hiển thị nút quay lại*  getSupportActionBar().setHomeButtonEnabled(true); *// Cho phép nhấn nút quay lại*  }  *// Lắng nghe sự kiện nhấn nút quay lại*  toolbar.setNavigationOnClickListener(view -> onBackPressed());  timerTextView = findViewById(R.id.*timerTextView*);  btVong = findViewById(R.id.*btVong*);  btBatdau = findViewById(R.id.*btBatdau*);  lvVong = findViewById(R.id.*lvVong*);  lapAdapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, lapList);  lvVong.setAdapter(lapAdapter);  btBatdau.setOnClickListener(view -> {  if (!running) {  startTime = System.*currentTimeMillis*() - timeInMillis;  handler.post(updateTimerRunnable);  running = true;  btBatdau.setText("Dừng");  btBatdau.setBackgroundTintList(getResources().getColorStateList(android.R.color.*holo\_red\_dark*));  btVong.setText("Vòng");  } else {  handler.removeCallbacks(updateTimerRunnable);  running = false;  btBatdau.setText("Bắt đầu");  btBatdau.setBackgroundTintList(getResources().getColorStateList(android.R.color.*holo\_blue\_dark*));  btVong.setText("Đặt lại");  }  });  *// Sử dụng ArrayAdapter để hiển thị dữ liệu vào ListView*  btVong.setOnClickListener(view -> {  if (running) {  lapCount++; *// Tăng số vòng lên mỗi lần bấm*  String lapTime = "Vòng " + lapCount + ": " + timerTextView.getText().toString();  lapList.add(0, lapTime); *// Thêm vòng mới vào đầu danh sách*  lapAdapter.notifyDataSetChanged();  } else {  timeInMillis = 0L;  timerTextView.setText("00:00,00");  lapList.clear();  lapCount=0;  lapAdapter.notifyDataSetChanged();  }  });  }  } |

### Xem lịch

A screenshot of a phone

Description automatically generated

Hình 2.25. Giao diện Xem lịch

**Nhiệm vụ của Activity:**

* Hiển thị giao diện xem lịch, cho phép người dùng chọn một ngày bất kỳ từ CalendarView, sau đó hiển thị: Thứ trong tuần (bằng tiếng Việt) Ngày, tháng/năm, Giờ hoàng đạo dựa theo giờ hiện tại, Chỉ số ngày tốt (tính đơn giản từ ngày). Ngoài ra, có nút để quay về ngày hôm nay.

**Thuật toán chính**:

* Thiết lập giao diện, ánh xạ các thành phần, gắn sự kiện chọn ngày và nút quay về hôm nay.

|  |
| --- |
| @Override  protected void onCreate(Bundle savedInstanceState) {     super.onCreate(savedInstanceState);     setContentView(R.layout.*activity\_calendar*);  *// Toolbar*  Toolbar toolbar = findViewById(R.id.*toolbar*);     setSupportActionBar(toolbar);     setTitle("Xem lịch");     if (getSupportActionBar() != null) {         getSupportActionBar().setDisplayHomeAsUpEnabled(true);         getSupportActionBar().setHomeButtonEnabled(true);     }     toolbar.setNavigationOnClickListener(view -> onBackPressed());  *// Ánh xạ view*  calendarView = findViewById(R.id.*calendarView*);     textView2 = findViewById(R.id.*textView2*);     textView3 = findViewById(R.id.*textView3*);     textView4 = findViewById(R.id.*textView4*);     textView5 = findViewById(R.id.*textView5*);     button2 = findViewById(R.id.*button2*);     textView6 = findViewById(R.id.*textView6*);     Calendar today = Calendar.*getInstance*();     updateDateInfo(today);  *// Xử lý chọn ngày*  calendarView.setOnDateChangeListener((view, year, month, dayOfMonth) -> {         Calendar selectedCal = Calendar.*getInstance*();         selectedCal.set(year, month, dayOfMonth);         updateDateInfo(selectedCal);     });  *// Khi bấm nút quay về hôm nay*  button2.setOnClickListener(view -> {         long todayMillis = System.*currentTimeMillis*();         calendarView.setDate(todayMillis, true, true);         Calendar todayCal = Calendar.*getInstance*();         updateDateInfo(todayCal);     });  } |

* Cập nhật thông tin chi tiết cho ngày được chọn.

|  |
| --- |
| private void updateDateInfo(Calendar cal) {     Date date = cal.getTime();  *// Thứ*  String thu = new SimpleDateFormat("EEEE", new Locale("vi")).format(date);     textView2.setText(thu);  *// Ngày*  String ngay = new SimpleDateFormat("dd", Locale.*getDefault*()).format(date);     textView3.setText(ngay);  *// Tháng và Năm*  String thangNam = new SimpleDateFormat("'Tháng' MM 'Năm' yyyy", Locale.*getDefault*()).format(date);     textView4.setText(thangNam);  *// Giờ âm (giờ hoàng đạo)*  String gioAm = getGioAm(Calendar.*getInstance*().get(Calendar.*HOUR\_OF\_DAY*));     textView5.setText("Giờ " + gioAm);  *// Chỉ số ngày tốt (ví dụ đơn giản: chỉ số = (ngày % 9) \* 10 + 20)*  int dayOfMonth = cal.get(Calendar.*DAY\_OF\_MONTH*);     int chiSo = (dayOfMonth % 9) \* 10 + 20; *// Từ 20 đến 100*  if (chiSo > 100) chiSo = 100;     textView6.setText("Chỉ số ngày tốt: " + chiSo + "%");  } |

* Tính giờ hoàng đạo

|  |
| --- |
| private String getGioAm(int hour) {     if (hour >= 23 || hour < 1) return "Tý";     if (hour >= 1 && hour < 3) return "Sửu";     if (hour >= 3 && hour < 5) return "Dần";     if (hour >= 5 && hour < 7) return "Mão";     if (hour >= 7 && hour < 9) return "Thìn";     if (hour >= 9 && hour < 11) return "Tỵ";     if (hour >= 11 && hour < 13) return "Ngọ";     if (hour >= 13 && hour < 15) return "Mùi";     if (hour >= 15 && hour < 17) return "Thân";     if (hour >= 17 && hour < 19) return "Dậu";     if (hour >= 19 && hour < 21) return "Tuất";     if (hour >= 21 && hour < 23) return "Hợi";     return "";  } |

### Thống kê

* Màn hình người dùng

Screens screenshot of a phone

Description automatically generated

Hình 2.26. Giao diện Thống kê

**Nhiệm vụ của Activity:**

* Lấy toàn bộ danh sách công việc từ cơ sở dữ liệu .
* Lọc danh sách dựa theo chế độ người dùng chọn.
* Tính số lượng công việc đã hoàn thành và chưa hoàn thành.
* Hiển thị kết quả thống kê dưới dạng biểu đồ tròn (PieChart) bằng thư viện MPAndroidChart.
* Cập nhật lại danh sách công việc và số liệu thống kê tương ứng.

**Thuật toán chính:**

* Lọc danh sách công việc theo thời gian.

|  |
| --- |
| @SuppressLint("SetTextI18n")  public void showListTimetable(int mode){     List<Timetable> di2 = new ArrayList<>();     @SuppressLint("SimpleDateFormat") SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");     if (mode==0){  *//Lấy ngày hiện tại*  Date current = new Date();  *//Định dạng lại ngày hiện tại thành chuỗi để so sánh*  String date = sdf.format(current);  *//Kiểm tra các đối tượng có ngày trùng với ngày đã định dạng để thêm vào danh sách*  for (int i=0; i<di.size(); i++){             if (di.get(i).getDay().equals(date)){                 di2.add(di.get(i));             }         }     }     else if (mode==1){         for (int i=0; i<di.size(); i++){             try {                 if (*isInCurrentWeek*(Objects.*requireNonNull*(sdf.parse(di.get(i).getDay())))){                     di2.add(di.get(i));                 }             }             catch (Exception e){                 return;             }         }     }     else if (mode==2){         for (int i=0; i<di.size(); i++){             try {                 if (*isInCurrentMonth*(sdf.parse(di.get(i).getDay()))){                     di2.add(di.get(i));                 }             }             catch (Exception e){                 return;             }         }     }     else{         di2=di;     }     DashboardAdapter adapter = new DashboardAdapter(activity, R.layout.*dashboard\_list\_tile*, di2);     ListView lv = root.findViewById(R.id.*listview*);     lv.setAdapter(adapter);     float done=0;     for (int i=0; i<di2.size(); i++){         if (di2.get(i).getStatus()){             done+=1;         }     }     dh.setText("Số việc thực hiện: "+String.*valueOf*((int)done));     ch.setText("Số việc không thực hiện: "+String.*valueOf*(di2.size()-(int)done));     if(di2.isEmpty()){         pieChart.setVisibility(View.*INVISIBLE*);         return;     }     else{         pieChart.setVisibility(View.*VISIBLE*);     }     float tmp = done/di2.size();     showChart(tmp, 1-tmp);  } |

* Tạo biểu đồ tròn với 2 phần:

"Thực hiện" = num1 \* 100

"Không làm" = num2 \* 100

|  |
| --- |
| public void showChart(float num1, float num2){ *//num1 là phân trăm các công việc đã hoàn thành*  *//num2 là phần trăm công việc chưa hoàn thành*  *//Khởi tạo biểu đồ*  *//Tạo biểu đồ*  ArrayList<PieEntry> pieEntries = new ArrayList<>();     pieEntries.add(new PieEntry(num1\*100, "Thực hiện"));     pieEntries.add(new PieEntry(num2\*100, "Không làm"));  *//Tạo định dạng cho biểu đồ*  PieDataSet pieDataSet = new PieDataSet(pieEntries, "");     pieDataSet.setColors(ColorTemplate.*MATERIAL\_COLORS*);  *//Tạo và định dạng đối tượng chứa dữ liệu*  PieData pieData = new PieData(pieDataSet);     pieData.setValueFormatter(new PercentFormatter());  *//Hiển thị dữ liệu*  pieChart.setData(pieData);     pieChart.invalidate();  *//Tùy chỉnh giao diện cho biểu đồ*  pieChart.setHoleRadius(25f);     pieChart.getDescription().setEnabled(false);     pieChart.setDrawHoleEnabled(true);     pieChart.setHoleColor(android.graphics.Color.*WHITE*);     pieChart.setTransparentCircleRadius(20f);     pieDataSet.setValueTextSize(18f);  } |

* Kiểm tra ngày có thuộc tuần hiện tại không, dựa vào Calendar.SUNDAY làm mốc.

|  |
| --- |
| public static boolean isInCurrentWeek(Date dateToCheck) {  *//Lấy ngày đầu tiên của tuần*  Calendar calendar = Calendar.*getInstance*();     calendar.set(Calendar.*DAY\_OF\_WEEK*, Calendar.*SUNDAY*);     calendar.set(Calendar.*HOUR\_OF\_DAY*, 0);     calendar.set(Calendar.*MINUTE*, 0);     calendar.set(Calendar.*SECOND*, 0);     calendar.set(Calendar.*MILLISECOND*, 0);     Date startOfWeek = calendar.getTime();     Calendar today = Calendar.*getInstance*();     today.set(Calendar.*MINUTE*, 0);     today.set(Calendar.*SECOND*, 0);     today.set(Calendar.*HOUR\_OF\_DAY*, 0);     today.set(Calendar.*MILLISECOND*, 0);     Date today\_date=today.getTime();     if (today\_date.equals(startOfWeek)){         calendar.add(Calendar.*DATE*, -7);         startOfWeek = calendar.getTime();     }  *//Lấy ngày cuối cùng của tuần*  calendar.add(Calendar.*DATE*, 1);     startOfWeek=calendar.getTime();     calendar.add(Calendar.*DATE*, 6);     Date endOfWeek = calendar.getTime();  *//Kiểm tra nếu ngày là ngày đầu hoặc cuối hoặc là trong tuần hay không, nếu có sẽ trả về True*  return dateToCheck.equals(startOfWeek) || dateToCheck.equals(endOfWeek)             || (dateToCheck.after(startOfWeek) && dateToCheck.before(endOfWeek));  } |

* Kiểm tra ngày có thuộc tháng hiện tại không

|  |
| --- |
| public static boolean isInCurrentMonth(Date checkDate) {  *//Lấy thông tin tháng và năm hiện tại*  Calendar calendar = Calendar.*getInstance*();     int currentMonth = calendar.get(Calendar.*MONTH*);     int currentYear = calendar.get(Calendar.*YEAR*);  *//Lấy ra thông tin tháng và năm của đối tượng Date được truyền vào để kiểm tra*  calendar.setTime(checkDate);     int checkMonth = calendar.get(Calendar.*MONTH*);     int checkYear = calendar.get(Calendar.*YEAR*);  *//Kiểm tra tháng và năm có trùng với tháng và năm hiện tại không, nếu có sẽ trả về True*  return (currentMonth == checkMonth && currentYear == checkYear);  } |

### Quản lý người dùng

* Màn hình Người dùng

Screens screenshot of a cell phone

AI-generated content may be incorrect.

Hình 2.27. Giao diện quản lý người dùng

#### Trợ giúp

A screenshot of a calculator

AI-generated content may be incorrect.

Hình 2.28. Giao diện Trợ giúp

**Nhiệm vụ của Activity:**

* Trợ giúp người dùng thực hiện liện hệ với với người phát triển qua các kênh liên hệ: góp ý, hợp tác, trò chuyện, phát triển thêm chức năng,…

**Thuật toán chính:**

* Xử lí sự kiện kích nút Gửi tin nhắn SMS

|  |
| --- |
| public void onSMS() {  Uri smsUri = Uri.*parse*("smsto:0123456789");  Intent intent = new Intent(Intent.*ACTION\_SENDTO*, smsUri);  intent.putExtra("sms\_body", "Xin chào, tôi cần hỗ trợ.");  intent.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  application.startActivity(intent); } |

* Xử lí sự kiện khi Gọi điện thoại

|  |
| --- |
| private void makePhoneCall() {  new AlertDialog.Builder(this)  .setTitle("Bạn có muốn gọi điện thoại?")  .setMessage("Cuộc gọi sẽ tính cước phí đến máy bạn")  .setNegativeButton("Hủy", (dialog, which) -> dialog.dismiss())  .setPositiveButton("Có", (dialog, which) -> {  Uri uri = Uri.*parse*("tel:0123456789");  Intent intent = new Intent(Intent.*ACTION\_DIAL*, uri);  startActivity(intent);  }).show(); } |

* Xử lí sự kiện khi Gửi Email

|  |
| --- |
| public void onEmail() {  Intent intent = new Intent(Intent.*ACTION\_SENDTO*);   intent.putExtra(Intent.*EXTRA\_SUBJECT*, "Yêu cầu hỗ trợ");  intent.putExtra(Intent.*EXTRA\_TEXT*, "Xin chào nhóm phát triển, tôi cần hỗ trợ về...");  intent.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  application.startActivity(intent); } |

#### Sao lưu

A screenshot of a cell phone

AI-generated content may be incorrect.

Hình 2.29. Giao diện Sao lưu

**Nhiệm vụ của Activity:**

* Hỗ trợ người dùng sao lưu dữ liệu. Sử dụng trong mục dích muốn trích xuất dữ liệu, lấy lại dữ liệu cũ trong tương lai.

**Thuật toán chính:**

* Cấp quyền cần thiết để sao lưu lên hệ điều hành

|  |
| --- |
| Button btnRequest = findViewById(R.id.*btnRequest*); btnRequest.setOnClickListener(v -> {  try {  StorageHelper.*requestStoragePermission*(this, storagePermissionLauncher);  } catch (Exception e) {  Toast.*makeText*(this, e.getMessage(), Toast.*LENGTH\_LONG*).show();  } }); storagePermissionLauncher = registerForActivityResult(  new ActivityResultContracts.StartActivityForResult(),  result -> {  try {  if (StorageHelper.*checkStoragePermission*(this)) {  Toast.*makeText*(this, "Đã cấp quyền", Toast.*LENGTH\_SHORT*).show();  } else {  Toast.*makeText*(this, "Quyền đã bị từ chối", Toast.*LENGTH\_SHORT*).show();  }  viewModel.checkPermissions();  } catch (Exception e) {  Toast.*makeText*(this, e.getMessage(), Toast.*LENGTH\_SHORT*).show();  }  });  public void checkPermissions() {  try {   if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*R*) {  if (Environment.*isExternalStorageManager*()) {  permissionStatus.setValue("Cho phép");  } else {  permissionStatus.setValue("Từ chối");  }  } else {  if (ContextCompat.*checkSelfPermission*(context, android.Manifest.permission.*READ\_EXTERNAL\_STORAGE*) == PackageManager.*PERMISSION\_GRANTED* && ContextCompat.*checkSelfPermission*(context, android.Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*) == PackageManager.*PERMISSION\_GRANTED*) {  permissionStatus.setValue("Cho phép");  } else {  permissionStatus.setValue("Từ chối");  }  }  } catch (Exception e) {  Toast.*makeText*(this.context, e.getMessage(), Toast.*LENGTH\_SHORT*).show();  } } |

* Sao lưu tệp JSON

|  |
| --- |
| saveFileLauncher = registerForActivityResult(  new ActivityResultContracts.StartActivityForResult(),  result -> {  if (result.getResultCode() == Activity.*RESULT\_OK*) {  Intent data = result.getData();  if (data != null) {  Uri fileUri = data.getData();  if (fileUri != null) {  viewModel.writeJson(this, fileUri);  }  }  }  });  public void writeJson(Context context, Uri fileUri) {  try {  JsonHelper.*writeJson*(UriHelper.*resolveUri*(context, fileUri), databaseHelper.getAllTimeTable());  Toast.*makeText*(this.context, "Sao lưu dữ liệu thành công", Toast.*LENGTH\_LONG*).show();  } catch (Exception e) {  Toast.*makeText*(this.context, e.getMessage(), Toast.*LENGTH\_LONG*).show();  } } |

* Lấy dữ liệu từ tệp JSON

|  |
| --- |
| pickFileLauncher = registerForActivityResult(  new ActivityResultContracts.StartActivityForResult(),  result -> {  if (result.getResultCode() == Activity.*RESULT\_OK*) {  Intent data = result.getData();  if (data != null) {  new AlertDialog.Builder(this)  .setMessage("Nếu bạn xác nhận, toàn bộ thời gian biểu hiện tại sẽ bị xóa sổ và tạo lại thời gian biểu hiện tại")  .setTitle("Bạn có muốn sửa toàn bộ thời gian biểu hiện tại?")  .setPositiveButton("Có", (dialog, which) -> {  Uri fileUri = data.getData();  if (fileUri != null) {  viewModel.readJson(this, fileUri);  }  dialog.dismiss();  })  .setNegativeButton("Không", (dialog, which) -> {  dialog.dismiss();  })  .show();  }  }  } );  public void readJson(Context context, Uri fileUri) {  try {  String filePath = UriHelper.*resolveUri*(context, fileUri);  List<Map<String, Object>> myList = JsonHelper.*readJson*(filePath);  this.onChangeTimetable(makeList(myList));  Toast.*makeText*(this.context, String.*format*("Đã thêm %d thời gian biểu vào cơ sở dữ liệu", myList.size()), Toast.*LENGTH\_LONG*).show();  } catch (Exception e) {  Toast.*makeText*(this.context, e.getMessage(), Toast.*LENGTH\_LONG*).show();  } }  private List<Timetable> makeList(List<Map<String, Object>> rawList) {  List<Timetable> timetableList = new ArrayList<>();  for (Map<String, Object> rawEntry : rawList) {  Timetable timetable = new Timetable();  timetable.setTimeTableId(((Double) Objects.*requireNonNull*(rawEntry.get("timeTableId"))).intValue());  timetable.setDay((String) rawEntry.get("day"));  timetable.setName((String) rawEntry.get("name"));  timetable.setNote((String) rawEntry.get("note"));  timetable.setStartTime((String) rawEntry.get("startTime"));  timetable.setEndTime((String) rawEntry.get("endTime"));  timetable.setUserId(((Double) Objects.*requireNonNull*(rawEntry.get("userId"))).intValue());  timetableList.add(timetable);  }  return timetableList; }  public void onChangeTimetable(List<Timetable> timetableList) {  this.databaseHelper.clearTimeTable();  for (Timetable t : timetableList) {  this.databaseHelper.addTimeTable(t.getName(), t.getDay(), t.getNote(), t.getStartTime(), t.getEndTime(), t.getStatus(),  UserSessionManager.*getInstance*().getCurrentUser().getUserId());  } } |

# KIẾN THỨC LĨNH HỘI

## Nội dung đã thực hiện

Thông qua bài tập lớn, nhóm 05 chúng em đã thực hiện được một số nội dung như sau:

* + Nắm được kiến thức liên quan đến việc xử lý các sự kiện trên một ứng dụng Android, thiết kế màn hình giao diện, hiểu rõ hơn về mô hình MVC, điều hướng giữa các Activity và Fragment, làm việc với SQLite để xây dựng cơ sở dữ liệu cục bộ cho ứng dụng.
  + Có kỹ năng làm việc nhóm, phân chia công việc dựa trên thế mạnh của từng thành viên và phối hợp hiệu quả để đảm bảo tiến độ.
  + Sử dụng các công cụ quản lý dự án như GitHub và Google Drive để chia sẻ tài liệu, mã nguồn và theo dõi tiến độ.
  + Rèn luyện kỹ năng tìm kiếm thông tin và giải quyết vấn đề khi gặp lỗi hoặc khó khăn trong quá trình phát triển.
  + Nâng cao vốn từ vựng tiếng Anh chuyên ngành công nghệ thông tin.

**Kết quả đạt được:**

Những chuẩn đầu ra của học phần mà nhóm đã đạt được trong quá trình thực hiện:

* + Cài đặt được một bài toán cụ thể sử dụng các thành phần cơ bản trong lập trình di động trên nền tảng Android như View, Adapter, Intent, Service.
  + Vận dụng tốt các nguyên tắc thiết kế và lập trình để giải quyết các bài toán thực tế.
  + Phát triển một ứng dụng hoàn chỉnh với các tính năng quản lý thời gian, bao gồm:
* Quản lý thời gian biểu: Cho phép người dùng tạo, chỉnh sửa, xóa và xem chi tiết các hoạt động trong ngày, trong tuần.
* Thống kê: Cho phép người dùng có thể xem tổng hợp các dữ liệu liên quan đến công việc và biểu đồ tỷ lệ hoàn thành nhiệm vụ theo ngày, tuần, tháng
* Xem lịch: Hiển thị ngày, tháng, năm; hỗ trợ đồng bộ hóa với lịch hệ thống để quản lý dễ dàng hơn.
* Hẹn giờ: Thiết lập thời gian báo thức cho các sự kiện, công việc hoặc thời điểm cần chú ý.
* Bấm giờ: Tính năng đo thời gian thực hiện một công việc hoặc hoạt động, hỗ trợ chia các vòng (lap time) để theo dõi chi tiết hơn.
* Sao lưu: Lưu trữ thông tin thời gian biểu, lịch hẹn lên bộ nhớ thiết bị.
* Trợ giúp: Tiếp nhận phản ánh, hỗ trợ người dùng khi gặp vấn đề qua tin nhắn SMS, số điện thoại, Email.
  + Cài đặt được bài toán trên thiết bị di động chạy Android.
  + Xử lý hiệu quả các lỗi liên quan đến giao diện, chức năng và hiệu năng trong quá trình thử nghiệm, nâng cao trải nghiệm người dùng.
  + Triển khai thành công ứng dụng thực tế trên nền tảng Android.

**Hạn chế:**

Mặc dù đã đạt được nhiều kết quả tích cực, ứng dụng vẫn còn tồn tại một số hạn chế:

* + Phụ thuộc nhập liệu thủ công: người dùng cần tự tạo và nhập thông tin lịch hẹn, thời gian biểu. Nếu không duy trì thường xuyên sẽ dẫn đến thiếu hiệu quả.
  + Chưa hỗ trợ tích hợp với các dịch vụ đám mây như Google Calendar để tự động đồng bộ sự kiện.
  + Một số tính năng còn hạn chế chưa được tối ưu hóa.
  + Ứng dụng chưa áp dụng trí tuệ nhân tạo (AI) để phân tích lịch sử sử dụng và đưa ra khuyến nghị tối ưu hóa thời gian.
  + Các báo cáo và thống kê chỉ dừng lại ở mức cơ bản, chưa có phân tích nâng cao về mức độ hiệu quả sử dụng thời gian.

**Bài học kinh nghiệm**

Do thời gian có hạn, nhóm chưa thể hoàn thiện toàn bộ chức năng như mong muốn và chưa khai thác hết các tình huống sử dụng phức tạp. Tuy nhiên, thông qua quá trình phát triển, nhóm đã rút ra được nhiều bài học quý giá như:

* Cần lên kế hoạch và phân chia thời gian hợp lý hơn.
* Cần tìm hiểu thêm về các thư viện, công nghệ mới để tăng tính năng cho ứng dụng.
* Giao tiếp và làm việc nhóm đóng vai trò quan trọng trong việc đảm bảo tiến độ và chất lượng dự án.

## Hướng phát triển

Dựa trên những kết quả và hạn chế hiện tại, nhóm đề ra các hướng phát triển trong tương lai như sau:

* + Tối ưu hóa trải nghiệm người dùng: Cải thiện tốc độ phản hồi và giao diện trực quan, dễ sử dụng hơn.
  + Tích hợp với các nền tảng thông minh: Hỗ trợ đồng bộ với Google Calendar, đồng hồ thông minh để tự động ghi nhận và nhắc nhở lịch trình.
  + Nâng cao khả năng phân tích và báo cáo: Phát triển tính năng thống kê chi tiết, phân tích hiệu suất sử dụng thời gian theo ngày, tuần, tháng.
  + Ứng dụng công nghệ AI: Đưa ra các gợi ý cá nhân hóa về cách sắp xếp thời gian, cải thiện hiệu quả công việc và sinh hoạt.
  + Mở rộng khả năng sao lưu và chia sẻ: Hỗ trợ sao lưu đám mây và chia sẻ lịch trình với bạn bè, đồng nghiệp để dễ dàng cộng tác.

TÀI LIỆU THAM KHẢO

1. Nguyễn Bá Nghiễn (chủ biên) - Nguyễn Thái Cường - Vũ Thị Dương - Nguyễn Văn Tỉnh, Giáo trình Phát triển ứng dụng cho thiết bị di động, Nhà xuất bản Thống kê.
2. Wei-Meng Lee, Beginning Android 4 application development, 2012.
3. Bùi Duy Thành, Lập trình thiết bị di động với J2ME, 2007.