

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**REPORT OF GRADUATION THESIS**

**ỨNG DỤNG QUẢN LÝ TÀI CHÍNH CÁ NHÂN CHO SINH VIÊN TRÊN THIẾT BỊ DI ĐỘNG**

**Chuyên ngành đào tạo: Kỹ thuật phần mềm**

**Giảng viên hướng dẫn: Cô Trần Thị Anh Thi**

**Sinh viên thực hiện 1: Nguyễn Đình Thuận – 16073301**

**Sinh viên thực hiện 2: Bùi Đức Thuận Phát – 16022781**

Thành phố Hồ Chí Minh, 2020



**MINISTRY OF INDUSTRY AND TRADE**

**INDUSTRIAL UNIVERSITY OF HO CHI MINH CITY**

**FALCUTY OF INFORMATION TECHNOLOGY**

**REPORT OF GRADUATION THESIS**

**PERSONAL FINANCIAL MANAGEMENT APPLICATION FOR STUDENT ON MOBILE DEVICES**

**Major: Software Engineering**

**Suppervisor: Mrs Trần Thị Anh Thi**

**Student implementation 1: Nguyễn Đình Thuận – 16073301**

**Student implementation 2: Bùi Đức Thuận Phát – 16022781**

Hồ Chí Minh City, 2020

# LỜI CẢM ƠN

# NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIÁO VIÊN HƯỚNG DẪN

# NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIÁO VIÊN PHẢN BIỆN

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc44352821)

[NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIÁO VIÊN HƯỚNG DẪN 2](#_Toc44352822)

[NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIÁO VIÊN PHẢN BIỆN 3](#_Toc44352823)

[MỤC LỤC 4](#_Toc44352824)

[DANH MỤC CÁC HÌNH ẢNH 7](#_Toc44352825)

[DANH MỤC CÁC BẢNG BIỂU 8](#_Toc44352826)

[DANH MỤC CÁC TỪ VIẾT TẮT 9](#_Toc44352827)

[LỜI MỞ ĐẦU 8](#_Toc44352828)

[CHƯƠNG 1 : GIỚI THIỆU 9](#_Toc44352829)

[1.1 Tổng quan 9](#_Toc44352830)

[1.2 Mục tiêu đề tài 9](#_Toc44352831)

[1.3 Phạm vi đề tài 9](#_Toc44352832)

[1.4 Mô tả yêu cầu chức năng 9](#_Toc44352833)

[CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT 12](#_Toc44352834)

[2.1 Những kiến thức cơ bản và bắt buộc khi học lập trình Android 12](#_Toc44352835)

[2.2 Tại sao phát triển ứng dụng Android là lựa chọn tốt nhất của bạn? 13](#_Toc44352836)

[2.3 Giới thiệu về Activity 16](#_Toc44352837)

[2.4 Giới thiệu về Fragment 20](#_Toc44352838)

[2.5 Cơ sở dữ liệu SQLite 25](#_Toc44352839)

[2.6 Các công cụ hỗ trợ cho việc thực hiện lập trình ứng dụng 29](#_Toc44352840)

[2.6.1 Android Studio – Dùng để lập trình ứng dụng trên hệ điều hành Android 29](#_Toc44352841)

[2.6.2 Github – Dùng để lưu trữ dự án và quản lý việc lập trình của các thành viên trong nhóm 31](#_Toc44352842)

[2.6.3 Enterprise Architect – Dùng để vẽ các mô hình phân tích 35](#_Toc44352843)

[2.6.4 Mockflow – Dùng để thiết kế giao diện ở dạng Mockup 36](#_Toc44352844)

[2.6.5 Genymotion 36](#_Toc44352845)

[CHƯƠNG 3 : PHÂN TÍCH 39](#_Toc44352846)

[3.1 Phân tích yêu cầu bằng UML 39](#_Toc44352847)

[3.1.1 Mô hình Use case tổng quát 39](#_Toc44352848)

[3.1.2 Danh sách tác nhân và mô tả 40](#_Toc44352849)

[3.1.3 Danh sách Use case và mô tả 40](#_Toc44352850)

[3.1.4 Tình huống hoạt động (Use case) 41](#_Toc44352851)

[CHƯƠNG 4 THIẾT KẾ VÀ HIỆN THỰC 86](#_Toc44352852)

[4.1 Class Diagram 87](#_Toc44352853)

[4.2 Mô hình liên kết thực thể 88](#_Toc44352854)

[4.3 Mô hình cơ sở dữ liệu 89](#_Toc44352855)

[4.3.1 Cơ sở dữ liệu quan hệ 89](#_Toc44352856)

[4.3.2 Các ràng buộc toàn vẹn trong cơ sở dữ liệu 90](#_Toc44352857)

[4.4 Thiết kế giao diện dạng Mockup 92](#_Toc44352858)

[4.5 Một số giao diện của ứng dụng 96](#_Toc44352859)

[4.6 Phân luồng màn hình của ứng dụng 96](#_Toc44352860)

[4.7 Cấu hình phần cứng, phần mềm 96](#_Toc44352861)

[4.7.1 Phần cứng 96](#_Toc44352862)

[4.7.2 Phần mềm 96](#_Toc44352863)

[CHƯƠNG 5 : KẾT LUẬN 97](#_Toc44352864)

[5.1 Kết quả đạt được 97](#_Toc44352865)

[5.2 Hạn chế của đồ án 97](#_Toc44352866)

[5.3 Hướng phát triển 97](#_Toc44352867)

[TÀI LIỆU THAM KHẢO 98](#_Toc44352868)

[PHỤ LỤC 99](#_Toc44352869)

# DANH MỤC CÁC HÌNH ẢNH

[*Hình 3.1 Mô hình Usecase tổng quát* 12](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173295)

[*Hình 3.2 Mô hình Activity Usecase 01 – Đăng ký tài khoản* 15](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173296)

[*Hình 3.3 Mô hình Activity Usecase 02 – Đăng nhập* 17](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173297)

[*Hình 3.4 Mô hình Activity Usecase 03 – Thống kê thu chi* 18](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173298)

[*Hình 3.5 Mô hình Activity Usecase 04 – Xem thông tin tài khoản* 20](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173299)

[*Hình 3.6 Mô hình Activity Usecase 05 – Cập nhật thông tin tài khoản* 22](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173300)

[*Hình 3.7 Mô hình Activity Usecase 07 – Tạo ví* 25](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173301)

[*Hình 3.8 Mô hình Activity Usecase 08 – Cập nhật ví* 27](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173302)

[*Hình 3.9 Mô hình Activity Usecase 09 – Xóa ví* 29](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173303)

[*Hình 3.10 Mô hình Activity Usecase 10 – Cập nhật thu chi* 31](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173304)

[*Hình 3.11 Mô hình Activity Usecase 11 – Thêm thu chi* 33](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173305)

[*Hình 3.12 Mô hình Activity Usecase 12 – Xóa thu chi* 35](file:///C:\Users\nguye\OneDrive\Desktop\BaoCaoDoAn\KLTN2020_Nhom02_TaiLieuPhanTich.docx#_Toc39173306)

# DANH MỤC CÁC BẢNG BIỂU

[*Bảng 3.1 Danh sách tác nhân và mô tả* 13](#_Toc39173277)

[*Bảng 3.2 Danh sách Usecase* 14](#_Toc39173278)

[*Bảng 3.3 Đặc tả Usecase 01 – Đăng ký tài khoản* 15](#_Toc39173279)

[*Bảng 3.4 Đặc tả Usecase 02 – Đăng nhập* 16](#_Toc39173280)

[Bản*g 3.5 Đặc tả Usecase 03 – Thống kê thu chi* 18](#_Toc39173281)

[*Bảng 3.6 Đặc tả Usecase 04 – Xem thông tin tài khoản* 19](#_Toc39173282)

[*Bảng 3.7 Đặc tả Usecase 05 – Cập nhật thông tin tài khoản* 21](#_Toc39173283)

[*Bảng 3.8 Đặc tả Usecase 06 – Xem số dư* 23](#_Toc39173284)

[*Bảng 3.9 Đặc tả Usecase 07 – Tạo ví* 24](#_Toc39173285)

[*Bảng 3.10 Đặc tả Usecase 08 – Cập nhật ví* 26](#_Toc39173286)

[*Bảng 3.11 Đặc tả Usecase 09 – Xóa ví* 28](#_Toc39173287)

[*Bảng 3.12 Đặc tả Usecase 10 – Cập nhật thu chi* 30](#_Toc39173288)

[*Bảng 3.13 Đặc tả Usecase 11 – Thêm thu chi* 33](#_Toc39173289)

[*Bảng 3.14 Đặc tả Usecase 12 – Xóa thu chi* 35](#_Toc39173290)

[*Bảng 3.15 Đặc tả Usecase 13 – Thêm kế hoạch* 36](#_Toc39173291)

[*Bảng 3.16 Đặc tả Usecase 14 – Cập nhật kế hoạch* 37](#_Toc39173292)

[*Bảng 3.17 Đặc tả Usecase 15 – Xóa kế hoạch* 38](#_Toc39173293)

[*Bảng 3.18 Đặc tả Usecase 16 – Đổi mật khẩu* 39](#_Toc39173294)

# DANH MỤC CÁC TỪ VIẾT TẮT

# LỜI MỞ ĐẦU

1. Tổng quan tình hình nghiên cứu thuộc lĩnh vực của đề tài

Hiện nay những phần mềm giúp sinh viên quản lý việc chi tiêu của chính mình có rất nhiều và đa dạng, đem lại một lợi ích rất lớn, đáp ứng đầy đủ các nhu cầu quản lý chi tiêu cho cá nhân, gia đình. Sinh viên có thể ghi chép đầy đủ hoạt động chi tiêu, các khoản thu nhập. Quản lý tiền bạc cho từng ngày, từng tháng, từ đó làm chủ ngân quỹ cũng như dành dụm, tích góp tiền. Cho người dùng biết số tiền mình hiện có trong từng tài khoản cá nhân như ví, tài khoản ATM, sổ tiết kiệm,… giúp lên kế hoạch và ghi chép lại chi tiêu cho những chuyến du lịch, những chuyến công tác,…

Nhóm đã tìm hiểu và sử dụng một thời gian các ứng dụng như: phần mềm Sổ thu chi Misa, phần mềm Ví điện tử - Money Manager.

2. Ý nghĩa khoa học và thực tiễn của đề tài

Ứng dụng hỗ trợ sinh viên quản lý việc thu chi cá nhân của mình, sinh viên sẽ đỡ phải ghi chép lại trên mảnh giấy, trong sổ tay những khoản thu chi của mình, việc sử dụng giấy để ghi chép ta thường sẽ để lạc chúng ở đâu đó. Với việc sỡ hữu những chiếc điện thoại thông minh như hiện nay thì không cần những loại giấy tờ dùng để ghi chép nữa.

# : GIỚI THIỆU

## Tổng quan

Ứng dụng quản lý thu chi cá nhân dành cho sinh viên nhóm đã thực hiện là một ứng dụng chạy trên hệ điều hành Android, hỗ trợ các vấn đề sau:

* Giúp sinh viên quan lý những khoản thu, khoản chi của mình.
* Quản lý số tiền hiện có trong các ví của mình.
* Lập kế hoạch tiết kiệm, để dành một số tiền cho một sự kiện nào đó.
* Xem tổng thu, tổng chi hằng ngày, hàng tháng, năm.

## Mục tiêu đề tài

Vận dụng các lý thuyết căn bản đã được học trước đó và các nghiên cứu từ thực tiễn để thiết kế và xây dựng phần mềm quản lý chi tiêu cho sinh viên.

Đem lại sự tiện lợi cho sinh viên trong việc ghi chép lại việc thu chi của mình.

## Phạm vi đề tài

Ứng dụng được xây dựng trên hệ điều hành Android 6.0 trở lên, sử dụng hệ cơ sở dữ liệu SQLite

Ứng dụng được xây dựng dựng dành cho cá nhân như là sinh viên.

## Mô tả yêu cầu chức năng

**Yêu cầu chức năng:**

Đăng nhập, đăng ký tài khoản: người dùng tạo một tài khoản để sử dụng ứng dụng, tránh việc khi người khác mượn điện thoại thì sẽ dễ dàng vào được ứng dụng và xem nội dung của mình.

Quản lý tài khoản: người dùng có thể đổi mật khẩu của tài khoản, nếu quên mật khẩu thì có thể tạo lại mật khẩu thông qua mã bí mật, nhập vào các thông tin cá nhân của mình.

Quản lý ví: với những ví đã có sẵn, người dùng có thể thêm ví cho phù hợp với nhu cầu sử dụng của mình và tiến hành thu chi dựa trên số tiền của những ví này ví này, sinh viên có thể chuyển tiền qua lại giữa các ví với nhau.

Quản lý thu chi: Người dùng thêm mới khoản thu chi, chọn hình thức thu hoặc chi, nhập vào số tiền, thời gian, đối tượng, ghi chú về khoản thu chi, sửa và xóa khoản thu chi.

Tổng thu chi: ứng dụng sẽ tự động tính tổng thu chi hàng ngày, hàng tháng, năm dựa vào các khoản thu chi đã thêm trước đó.

Lập kế hoạch tiết kiệm: người dùng có thể lập ra một kế hoạch tiết kiệm với một số tiền và mục đích nhất định, nếu thực hiện thu vào kế hoạch đủ số tiền sẽ thông báo cho người dùng, nếu hết hạn của kế hoạch mà người dùng không thu vào đủ số tiền thì kế hoạch sẽ thất bại.

Quản lý danh mục thu chi: với các danh mục thu chi có sẵn như khoản thu cho tiền lương, tiền thưởng, khoản chi cho học tập, ăn uống, người dùng có thể thêm các danh mục khác để phù hợp với nhu cầu sử dụng của mình, mỗi danh mục sẽ có một ví ưu tiên sử dụng tiền cho danh mục này.

**Yêu cầu phi chức năng:**

Ứng dụng có dung lượng không quá lớn, tốc độ xử lý nhanh.

Công việc xử lý thực hiện chính xác, không xảy ra sai sót.

Tránh việc để lộ thông tin nhạy cảm của người dùng.

Đảm bảo an toàn dữ liệu.

Giao diện dễ sử dụng, không quá khó đối với người lần đầu tiếp xúc.

# : CƠ SỞ LÝ THUYẾT

## Những kiến thức cơ bản và bắt buộc khi học lập trình Android

Hiện nay Android là một trong những hệ điều hành di động phổ biến nhất thế giới, rất nhiều công ty đăng tuyển lập trình ứng dụng di động trên Android. Chính vì vậy, cơ hội nghề nghiệp mở ra cho ngành nghề này rất cao, mức thu nhập lại cực kỳ hấp dẫn.

**Lập trình Android là gì ?**

Android là hệ điều hành trên điện thoại di động (và hiện nay là cả trên một số đầu phát HD, HD Player, TV) phát triển bởi Google và dựa trên nền tảng Linux. Lập trình android là một lập trình ứng dụng di động phổ biến. Trước đây, Android được phát triển bởi công ty liên hợp Android (sau đó được Google mua lại vào năm 2005).

Các nhà phát triển viết ứng dụng cho Android dựa trên ngôn ngữ Java. Sự ra mắt của Android vào ngày 5 tháng 11 năm 2007 gắn với sự thành lập của liên minh thiết bị cầm tay mã nguồn mở, bao gồm 78 công ty phần cứng, phần mềm và viễn thông nhằm mục đính tạo nên một chuẩn mở cho điện thoại di động trong tương lai.

**Kiến trúc cơ bản của hệ điều hành Android:**

***Nhân Linux:*** Đây là nhân nền tảng mà hệ điều hành Android dựa vào nó để phát triển. Đây là lớp chứa tất cả các thiết bị giao tiếp ở mức thấp dùng để điều khiển các phần cứng khác trên thiết bị Android.

***Thư viện:*** Chứa tất cả các mã cái mà cung cấp những tính năng chính của hệ điều hành Android, đối với ví dụ này thì SQLite là thư viện cung cấp việc hỗ trợ làm việc với database dùng để chứa dữ liệu. Hoặc Webkit là thư viện cung cấp những tính năng cho trình duyệt Web.

***Android Runtime:*** Là tầng cùng với lớp thư viện, Android runtime cung cấp một tập các thư viện cốt lõi để cho phép các lập trình viên phát triển viết ứng dụng bằng việc sử dụng ngôn ngữ lập trình Java. Android Runtime bao gồm máy ảo Dalvik (ở các version < 4.4, hiện tại là phiên bản máy ảo ART được cho là mạnh mẽ hơn trong việc xử lý biên dịch). Là cái để điều khiển mọi hoạt động của ứng dụng Android chạy trên nó (máy ảo Dalvik sẽ biên dịch ứng dụng để nó có thể chạy, thực thi được, tương tự như các ứng dụng được biên dịch trên máy ảo Java vậy). Ngoài ra máy ảo còn giúp tối ưu năng lượng pin cũng như CPU của thiết bị Android.

***Android Framework:*** Là phần thể hiện các khả năng khác nhau của Android (kết nối, thông báo, truy xuất dữ liệu) cho nhà phát triển ứng dụng, chúng có thể được tạo ra để sử dụng trong các ứng dụng của họ.

***Application:*** Tầng ứng dụng là tầng bạn có thể tìm thấy các thiết bị Android như Contact, trình duyệt…Và mọi ứng dụng bạn viết đều nằm trên tầng này.

## Tại sao phát triển ứng dụng Android là lựa chọn tốt nhất của bạn?

Mỗi doanh nghiệp cần phải cạnh tranh với sự sáng tạo để thu hút sự chú ý của nhiều khách hàng tiềm năng hơn. Đó là bởi vì mục tiêu cuối cùng của bất kỳ doanh nghiệp là tạo ra doanh thu. Hơn nữa, sự phát triển của nền tảng phát triển Android cung cấp cho các doanh nghiệp một cơ hội hiệu quả và có thể mở rộng để phát triển ứng dụng.

Các ứng dụng di động Android đã ảnh hưởng đến hầu hết các ngành công nghiệp như một phần của cuộc cách mạng kỹ thuật số ngày nay. Mặc dù iOS là một nền tảng phổ biến, nhưng được liệt kê dưới đây là một số lý do tại sao phát triển Android là nền tảng tốt nhất và hàng đầu cho các doanh nghiệp.

**Tỷ suất hoàn vốn cao với chi phí thấp:**

Một trong những lợi thế chính của phát triển ứng dụng Android là sự sẵn có dễ dàng của SDK Android. Các nhóm phát triển có thể sử dụng thiết kế vật liệu từ các SDK này để xây dựng các ứng dụng tương tác. Tuy nhiên, các developer/nhóm phát triển được yêu cầu trả phí đăng ký một lần để phân phối ứng dụng. Sau đó, họ có thể tận dụng bất kỳ thiết bị máy tính nào để xây dựng và thử nghiệm sản phẩm cho điện thoại thông minh của họ, đảm bảo đầu tư thấp và tăng sự tham gia của người dùng. Đổi lại, người dùng cuối, được hưởng lợi từ một ứng dụng tương tác và doanh nghiệp có được lợi tức đầu tư cao hơn.

**Triển khai nhanh hơn:**

Các ứng dụng Android dành cho doanh nghiệp có chu kỳ phát triển nhanh chóng kéo dài vài giờ. Nó cung cấp một lợi thế cạnh tranh cho các công ty muốn có một thị trường nhanh hơn cho ý tưởng mới của họ. Giảm thời gian tiếp cận thị trường (TTM), do đó, là một trong những lợi ích tốt nhất của phát triển Android.

**Nhiều nền tảng mục tiêu:**

Việc sử dụng Java làm ngôn ngữ lập trình giúp dễ dàng chuyển ứng dụng sang nhiều hệ điều hành như Symbian và Ubuntu. Do đó, doanh nghiệp có thể nhắm mục tiêu nhiều nền tảng với phát triển ứng dụng Android. Đó là một trong nhiều lý do tại sao các doanh nghiệp chọn phát triển Android.

**Tính linh hoạt và khả năng mở rộng:**

Với sự xuất hiện của Android Studio, hệ điều hành đã mở rộng về tính linh hoạt và khả năng thích ứng. Nó tích hợp với toàn bộ hệ sinh thái Android bao gồm điện thoại thông minh, máy tính bảng, thiết bị đeo và TV Android. Chúng làm cho ứng dụng Android tương thích với các công nghệ mới nổi như IoT, AR và VR. Do đó, nó cũng là một trong những lợi ích đáng kể của ứng dụng Android. Hơn nữa, tính linh hoạt của nền tảng ứng dụng Android cho phép các nhóm phát triển xây dựng các ứng dụng di động động phục vụ nhiều mục đích sau khi được cài đặt trên thiết bị.

**Bảo mật nâng cao:**

Android P đã giới thiệu một số tính năng bảo mật bổ sung và tích hợp. Nó sẽ giúp bảo vệ chống lại phần mềm độc hại và vi-rút. Do đó, an toàn và độ tin cậy là lợi ích đặc biệt của phát triển ứng dụng Android. Đọc về các thực tiễn tốt nhất về bảo mật và quyền riêng tư của Android giúp các tổ chức xác định phương pháp phù hợp trong suốt vòng đời của ứng dụng.

**Tùy chỉnh:**

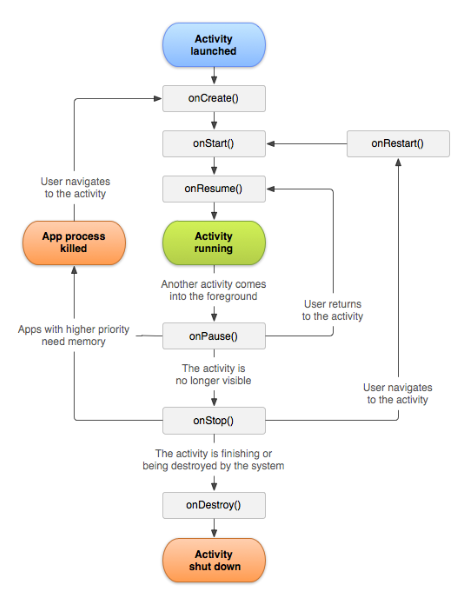
Android là một nền tảng nguồn mở và cung cấp các tính năng tùy chỉnh tối đa cho các nhóm phát triển. Đó là lý do tại sao các ứng dụng Android là một lựa chọn phổ biến. Hơn nữa, hệ điều hành này cho phép tạo ra các ứng dụng Android đa năng có thể dễ dàng tích hợp vào các công cụ đa phương tiện và chức năng quản lý dữ liệu của các quy trình kinh doanh hiện tại của bạn. Do đó, các doanh nghiệp có thể hưởng lợi từ việc tiếp cận cơ sở khách hàng rộng hơn bằng cách đáp ứng các yêu cầu kinh doanh thay đổi.

## Giới thiệu về Activity

Lớp Activity là thành phần quan trọng nhất của ứng dụng Android, cách mà chúng hoạt động tạo thành nền tảng cơ bản của mô hình lập trình ứng dụng. Android khởi chạy một ứng dụng thông thường bằng kích hoạt một Activity tương ứng với vòng đời cụ thể của nó trong quá trình hoạt động.

Thường một Activity cung cấp một của sổ, ở đó ứng dụng sẽ dựng các thành phần UI (User Interface - giao diện người dùng). Mặc định cửa sổ này có thể lấp đầy mà hình thiết bị, nhỏ hơn hoặc nổi phía trên các cửa sổ khác. Hầu hết các ứng dụng đều sử dụng nhiều màn hình khác nhau, có nghĩa nó sẽ phải có nhiều Activity khác nhau. Khi một Activity chỉ định là Main Activity, nó sẽ là màn hình đầu tiên khi khởi chạy ứng dụng. Activity này có thể gọi và kích hoạt một Activity khác.

**Vòng đời của Activity:**



**Mô tả sơ đồ vòng đời của Activity:**

Sơ đồ bắt đầu từ khi Activity launched, tức là khi Activity được kích hoạt, và được hệ thống đẩy vào BackStack. Sau khi kích hoạt, lần lượt các callback onCreate(), onStart(), onResume() sẽ được hệ thống gọi đến.  
Sau khi gọi đến các callback trên, thì Activity mới chính thức được xem là đang chạy (Activity running).

Lúc này, nếu có bất kỳ Activity nào khác chiếm quyền hiển thị, thì Activity hiện tại sẽ rơi vào trạng thái onPause(). Nếu sự hiển thị của Activity khác làm cho Activity mà chúng ta đang nói đến không còn nhìn thấy nữa thì onStop() sẽ được gọi ngay sau đó Nếu Acvitity đã vào onPause() rồi, tức là đang bị Activity khác đè lên, mà người dùng sau đó quay về lại Activity cũ, thì onResume() được gọi. Còn nếu Activity đã vào onStop() rồi, mà người dùng quay về lại Activity cũ thì onRestart() được gọi. Trong cả hai trường hợp Activity rơi vào onPause() hoặc onStop(), nó sẽ rất dễ bị hệ thống thu hồi (tức là bị hủy) để giải phóng tài nguyên, khi này nếu quay lại Activity cũ, onCreate() sẽ được gọi chứ không phải onResume() hay onRestart(). Và cuối cùng, nếu một Activity bị hủy một cách có chủ đích, chẳng hạn như người dùng nhấn nút Back ở System Bar, hay hàm finish() được gọi,… thì onDestroy() sẽ được kích hoạt và Activity kết thúc vòng đời của nó.

**Các trạng thái chính trong vòng đời Activity:**

**Running:** Khi Activity được kích hoạt, và được hệ thống để vào BackStack, nó sẽ bước vào trạng thái active. Với trạng thái active, người dùng hoàn toàn có thể nhìn thấy và tương tác với Activity của ứng dụng.

**Pause:** Trạng thái này khá đặc biệt. Trạng thái tạm dừng. Như bạn đã làm quen trên kia, trạng thái này xảy ra khi mà Activity của bạn vẫn đang chạy, người dùng vẫn nhìn thấy, nhưng Activity khi này lại bị che một phần bởi một thành phần nào đó. Chẳng hạn như khi bị một dialog đè lên. Cái sự che Activity này không phải hoàn toàn. Chính vì vậy mà Activity đó tuy được người dùng nhìn thấy nhưng không tương tác được.

**Stop:** Trạng thái này khá giống với trạng thái tạm dừng trên kia. Nhưng khi này Activity bị che khuất hoàn toàn bởi một thành phần giao diện nào đó, hoặc bởi một ứng dụng khác. Và tất nhiên lúc này người dùng không thể nhìn thấy Activity của bạn được nữa. Hành động mà khi người dùng nhấn nút Home ở System Bar để đưa ứng dụng của bạn về background, cũng khiến Activity đang hiển thị trong ứng dụng rơi vào trạng thái dừng này.

**Dead:** Nếu Activity được lấy ra khỏi BackStack, chúng sẽ bị hủy và rơi vào trạng thái này. Trường hợp này xảy ra khi user nhấn nút Back ở System Bar để thoát một Activity. Hoặc lời gọi hàm finish() từ một Activity để “kill chính nó”. Cũng có khi ứng dụng ở trạng thái background quá lâu, hệ thống có thể sẽ thu hồi tài nguyên bằng cách dừng hẳn các Activity trong ứng dụng, làm cho tất cả các Activity đều vào trạng thái này. Khi vào trạng thái dead, Activity sẽ kết thúc vòng đời của nó.  
Những ý trên giúp bạn nắm được tổng quan các trạng thái mà một Activity có thể trải qua.

**Làm quen với từng callback:**

**onCreate():** hàm này được gọi khá sớm, ngay khi activity được kích hoạt và thậm chí người dùng còn chưa thấy gì cả thì callback này đã đc gọi rồi. Ngoài ra thì bạn nên biết là callback này chỉ được gọi một lần duy nhất khi Activity được khởi tạo. Nó có thể được gọi lại nếu hệ thống xóa Activity này đi để lấy lại tài nguyên của hệ thống, nhưng rất hiếm khi xảy ra. Do đặc tính được gọi khá sớm và chỉ được gọi một lần duy nhất trong vòng đời của nó như vậy, nên bạn sẽ tận dụng để load giao diện cho Activity ở giai đoạn này, thông qua phương thức setContentView(). Ngoài giao diện ra, bạn có thể khởi tạo các logic nào đó chỉ chạy một lần ban đầu, như các lời gọi API, load database, tạo item list, tạo Navigation Drawer, và nhiều logic khác.

**onStart():** sau khi gọi đến onCreate(), hệ thống sẽ gọi đến onStart(). Hoặc hệ thống cũng sẽ gọi lại onStart() sau khi gọi onRestart() nếu trước đó nó bị che khuất bởi Activity nào khác (một màn hình khác hoặc một ứng dụng khác) che hoàn toàn và rơi vào onStop(). Khi hệ thống gọi đến callback này thì Activity được nhìn thấy bởi người dùng và nhưng chưa tương tác được. Bởi đặc tính này mà onStart() ít được dùng đến.

**onResume():** khi hệ thống gọi đến callback này thì bạn yên tâm rằng người dùng đã nhìn thấy và đã tương tác được với giao diện. onResume() được gọi khi Activity được khởi tạo rồi và bước qua onStart(). Hoặc khi Activity bị một giao diện nào khác che đi một phần (hoặc toàn phần), rồi sau đó quay lại Activity hiện tại. Bạn có thể thấy rằng callback này được gọi rất nhiều lần trong một vòng đời của nó.  
Chính đặc điểm này của onResume() mà bạn có thể tận dụng để quay lại tác vụ mà người dùng đang bị dang dở khi onPause()được gọi.

**onPause():** thông thường nếu có một thành phần nào đó che Activity hiện tại mà người dùng vẫn nhìn thấy Activity đó (nhìn thấy chứ không tương tác được). Chẳng hạn một popup hiện lên trên Activity. Thì onPause() của Activity sẽ được gọi. Sau này khi người dùng quay lại Activity thì onResume() sẽ được gọi.  
Bạn có thể tưởng tượng rằng onPause() cũng sẽ được gọi khá nhiều lần trong một vòng đời Activity. Theo như Google thì onPause() được gọi đến khá nhanh, nếu bạn muốn lưu trữ dữ liệu như mình nói trên kia, thì nên lưu những gì nhanh gọn lẹ thôi. Nếu bạn muốn lưu trữ các dữ liệu nặng, hoặc gọi API kết nối server chỗ này, nhiều khả năng ứng dụng sẽ không kịp thực hiện. Do đó, thay vì làm các thao tác nặng nề ở onPause(), bạn có thể cân nhắc gọi chúng ở onStop().

**onStop():** onStop() được gọi khi Activity không còn được nhìn thấy nữa, có thể một màn hình nào khác che lên hoàn toàn, có thể một ứng dụng nào đó vào foreground, hoặc người dùng nhấn nút Home để về màn hình chính. Bạn có thể tận dụng onStop() để lưu trữ dữ liệu ứng dụng. Hoặc để giải phóng các tài nguyên đang dùng. Ngưng các API còn đang gọi dang dở. Tuy nhiên khi onStop() được gọi không phải là lúc chúng ta cũng nói lời tạm biệt Activity, người dùng hoàn toàn có thể quay lại sử dụng Activity sau đó mà không cần phải khởi động lại Activity, khi này thì phương thức onRestart() và onStart() được gọi kế tiếp nhau.

**onDestroy():** có thể tận dụng callback này để giải phóng các tài nguyên hệ thống mà ở onStop() bạn chưa gọi đến. Vòng đời của một Activity kết thúc ở đây.

## Giới thiệu về Fragment

Với việc càng ngày càng nhiều mẫu mã thiết bị Android ra đời thì người ta đã nghĩ ra những cách khác nhau để bố trí giao diện ứng dụng sao cho phù hợp để tận dụng diện tích màn hình. Một trong số đó là **Fragment**. Đây là một thành phần khá quan trọng, được sử dụng rất nhiều trong những ứng dụng chuyên nghiệp, nó khá giống Activity.

Fragment chịu trách nhiệm quản lý một không gian màn hình, nhiều khi không gian này cũng chính là toàn màn hình. Và cái không gian màn hình đó của Fragment phải nằm trong một Activity nào đó. Một Activity có thể có nhiều Fragment, có khi nhiều Fragment của Activity đó cùng nhau hiển thị lên một màn hình, cũng có khi chúng luân phiên hiển thị nếu như mỗi chúng đều chiếm cả không gian màn hình. Và một ý nữa, một Fragment nào đó cũng có thể được khai báo và sử dụng bởi nhiều Activity khác nhau.

**Một số đặc điểm của Fragment:**

* Fragment là một thành phần Android độc lập, được sử dụng bởi một Activity, giống như một sub-activity.
* Fragment có vòng đời và giao diện riêng.
* Các Fragment thường có một file java đi kèm với file giao diện xml. Các fragment không có file giao diện xml thường được gọi là headless fragments.
* Vòng đời của fragment bị ảnh hưởng trực tiếp bởi vòng đời của activity chủ. Ví dụ, khi hoạt động bị tạm dừng, tất cả phân đoạn trong nó cũng vậy, và khi hoạt động bị hủy, tất cả phân đoạn cũng vậy.
* Một Fragment có thể được sử dụng trong nhiều Activitiy.
* Fragment được thêm vào API 11 trở lên.
* Fragment sử dụng phương thức getActivity() để lấy ra Activity cha.
* Fragment được định nghĩa trong file xml của Activity (static definition) hoặc có thể sửa đổi Fragment khi đang chạy (dynamic definition).

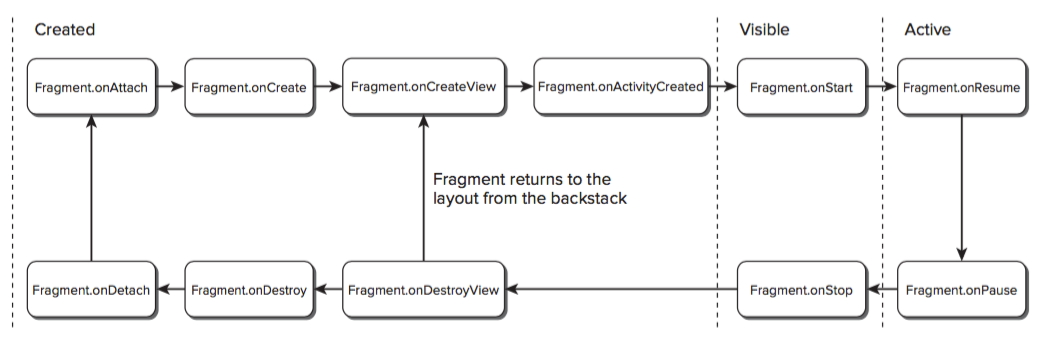
**Các trạng thái chính trong vòng đời Fragment:**

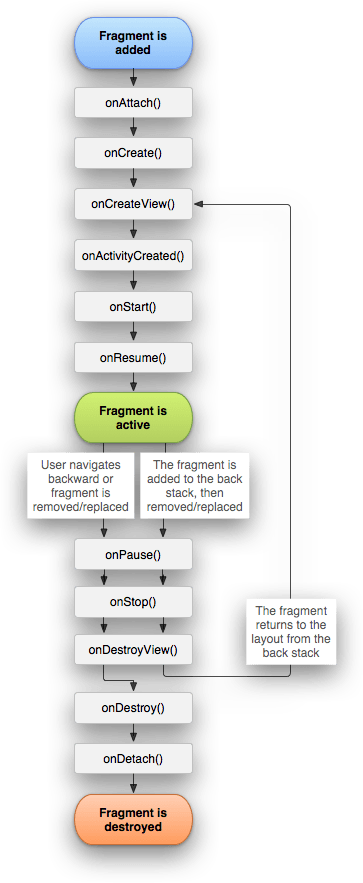
**Resume (Hoạt động):** Fragment hiển thị trong activity đang chạy.

**Pause (Tạm dừng):** một activity ở trong foreground, nhưng activity và fragment này nằm trong màn hình (activity ở foreground mờ một phần hoặc không che phủ toàn bộ màn hình).

**Stop (Dừng lại):** cũng giống với Activity, Fragment bị dừng khi bị thành phần nào đó che khuất hoàn toàn. Hay bị gỡ ra khỏi Activity. Dừng chưa phải là chấm hết cho đời sống của Fragment. Cụ thể là các trạng thái của nó vẫn còn được lưu trữ, để phòng trường hợp Fragment này được trở lại hiển thị cho người dùng.

**Dead (Chết):** nếu Fragment bị gỡ ra khỏi Activity, nhưng không được đưa vào Back Stack trước đó, thì nó sẽ kết thúc vòng đời. Hoặc Activity chứa Fragment này bị gỡ khỏi Back Stack, Fragment cũng sẽ chết theo.

**Vòng đời của Fragment:**

****

**onAttach()**: callback này được gọi khá sớm, ngay khi Activity chứa nó được kích hoạt. Hoặc ngay khi được gắn vào Activity. Callback này được gọi một lần duy nhất trong vòng đời Fragment.

**onCreate()**: callback này được gọi khi Fragment bắt đầu khởi tạo từ các dữ liệu đầu vào. Khác với onCreate() của Activity, bạn có thể tạo giao diện cho màn hình ở callback này, thì với Fragment chúng ta còn phải đợi qua callback tiếp theo mới có thể tạo giao diện được. Callback này cũng được gọi một lần trong vòng đời Fragment. Bạn nên khởi tạo các thành phần cơ bản của Fragment mà bạn muốn duy trì khi Fragment bị dừng hoặc tạm dừng, sau đó được phục hồi lại.

**onCreateView()**: khi Fragment bắt đầu vẽ UI lên màn hình, callback này được gọi. Nên chúng ta sẽ tận dụng callback này cho các thiết lập về giao diện. Bạn thấy rằng, theo như sơ đồ trên, thì callback này sẽ được gọi lại khi mà Fragment được gỡ ra khỏi Activity nhưng được đưa vào Back Stack, và được gọi lại hiển thị sau đó.

**onActivityCreated()**: callback này được gọi ngay sau khi onCreateView() được gọi. Nó báo hiệu trạng thái Activity chứa nó được khởi tạo hoàn toàn. Tuy ít được sử dụng hơn các callback khác, nhưng bạn cũng có thể tận dụng nó để thay đổi giao diện hay các tương tác với Activity chứa Fragment này thoải mái được rồi.

**onStart()**: khi Fragment bắt đầu được nhìn thấy bởi người dùng và chuẩn bị nhận tương tác.

**onResume()**: người dùng hoàn toàn nhìn thấy và tương tác được với Fragment.

**onPause()**: callback này như một dấu hiệu cho thấy rằng người dùng đang rời khỏi Fragment hiện tại.

**onStop()**: Fragment chính thức không còn được nhìn thấy nữa.

**onDestroyView()**: chắc chắn là đối tượng View sẽ bị hủy ở callback này. Và do đó các khởi tạo view của bạn ở onCreateView() sẽ nhanh chóng không còn nữa. Nếu như Fragment được đưa vào Back Stack, thì khi được lấy ra lại sau đó, callback onCreateView() sẽ được gọi lại.

**onDestroy()**: callback này chỉ như một lời “nhắc nhở” về vận mệnh của Fragment mà thôi.

**onDetach()**: callback này gọi đến báo hiệu Fragment sẽ được gỡ khỏi Activity đang chứa nó. Kết thúc vòng đời của Fragment.

## Cơ sở dữ liệu SQLite

SQLite là một cơ sở dữ liệu SQL mã nguồn mở, nó lưu trữ dữ liệu vào một tập tin văn bản trên một thiết bị. Nó mặc định đã được tích hợp trên thiết bị Android. Để truy cập dữ liệu này, bạn không cần phải thiết lập bất kỳ loại kết nối nào cho nó như JDBC, ODBC,... SQLite được Richard Hipp viết dưới dạng thư viện bằng ngôn ngữ lập trình C.

**SQLite có các ưu điểm:**

Tin cậy: các hoạt động transaction (chuyển giao) nội trong cơ sở dữ liệu được thực hiện trọn vẹn, không gây lỗi khi xảy ra sự cố phần cứng.

Tuân theo chuẩn SQL92 (chỉ có một vài đặc điểm không hỗ trợ).

Không cần cài đặt cấu hình.

Kích thước chương trình gọn nhẹ, với cấu hình đầy đủ chỉ không đầy 300 kB.

Thực hiện các thao tác đơn giản nhanh hơn các hệ thống cơ sở dữ liệu khách/chủ khác.

Không cần phần mềm phụ trợ.

Phần mềm tự do với mã nguồn mở, được chú thích rõ ràng.

**Cách sử dụng SQLite trong lập trình Android:**

Tương tự như các hệ quản trị cơ sở dữ liệu khác thì khi thao tác với Database thì bạn sẽ có những hành động cổ điển đó là **CRUD: Create, Read, Update, Delete.**

Đầu tiên, để thao tác với SQLite, ta phải dùng 2 đối tượng:

* **SQLiteOpenHelper**: đối tượng dùng để tạo, nâng cấp, đóng mở kết nối CSDL.
* **SQLiteDatabase**: đối tượng dùng để thực thi các câu lệnh SQL trên một CSDL.

**Đối tượng SQLiteOpenHelper:**

Lớp này có 2 hàm khởi tạo, một hàm 4 tham số: **SQLiteOpenHelper** (Context context, String name, SQLiteDatabase.CursorFactory factory, int version), một hàm 5 tham số: **SQLiteOpenHelper** (Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler), tuy nhiên ta chủ yếu làm việc với hàm 4 tham số.

Context context: Context là một lớp trừu tượng của hệ thống, chứa thông tin môi trường ứng dụng, cung cấp các phương thức để có thể tương tác với hệ điều hành, giúp chúng ta dễ dàng truy cập và tương tác tới các tài nguyên của hệ thống…

String name: Tên database.

CursorFactory factory: thường để null.

int version: version của database.

**Khi khởi tạo một đối tượng của lớp này, ta phải ghi đè 2 phương thức:**

* **onCreate()**: phương thức này được gọi bởi framework, nếu có yêu cầu truy cập database mà lại chưa khởi tạo database, ở đây ta phải viết code khởi tạo database, cụ thể là khởi tạo bảng (chú ý: khi khởi tạo bảng, ta phải đặt tên khóa chính là **\_id**).
* **onUpgrade()**: phương thức này được dùng khi ứng dụng của bạn có nhiều phiên bản database đc thêm vào. Nó sẽ cập nhật database hiện có hoặc khởi tạo lại thông qua onCreate(). Lớp này có 2 phương thức **getReadableDatabase()** (chỉ đọc) và **getWriteableDatabase()** (cho phép ghi đọc). Thông qua 2 phương thức này, ta có thể tạo ra một đối tượng **SQLiteDatabase.**

**Một số hàm có sãn trong gói cơ sở dữ liệu:**

* **openDatabase (String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler):**phương thức này chỉ mở thêm cơ sở dữ liệu hiện có với các chế độ cờ thích hợp. Các chế độ cờ phổ biến có thể là OPEN\_READWRITE OPEN\_READONLY.
* **openDatabase (String path, SQLiteDatabase.CursorFactory factory, int flags):** phương thức này tương tự như phương thức trên vì nó cũng mở ra các cơ sở dữ liệu hiện có, nhưng nó không định nghĩa bất kỳ xử lý để xử lý các lỗi cơ sở dữ liệu.
* **openOrCreateDatabase (String path, SQLiteDatabase.CursorFactory factory):**phương thức này không chỉ mở ra nhưng lại tạo ra cơ sở dữ liệu nếu nó không tồn tại. Phương thức này là tương đương với phương thức OpenDatabase.
* **openOrCreateDatabase (File file, SQLiteDatabase.CursorFactory factory):** phương thức này cũng tương tự như phương thức trên nhưng phải mất File đối tượng như là một con đường mà không phải là một chuỗi. Nó tương đương với file .getPath().

**Các phương thức để làm việc với SQLite:**

* insert().
* update().
* delete().
* execSQL(): thực thi một câu lệnh SQL trực tiếp.
* query(): dùng để truy vấn cơ sở dữ liệu.

**Đối tượng Cursor:**

Đối tượng cursor hiểu đơn giản là một con trỏ, trỏ đến kết quả trả về của câu truy vấn. con trỏ này trỏ đến cái bảng trả về của câu truy vấn.

* **cursor.getCount():**phương thức trả về số dòng của bảng kết quả.
* **cursor.moveToFirst():**phương thức này di chuyển con trỏ này lên đầu bảng.
* **cursor.moveToNext():**phương thức nàyđể di chuyển sang dòng tiếp theo.
* **cursor.isAfterLast():**kiểm tra xem cursor ở cuối bảng hay không, không trỏ vào dòng nào, phương thức này sẽ trả về giá trị true.
* **cursor.getString(), cursor.getInt():**hai phương này**đ**ể lấy ra thông tin cột theo tên cột hoặc index (chỉ mục).

**Đối tượng ContentValues:**

Các đối tượng ContentValues cho phép xác định khóa/giá trị. Các key đại diện nhận dạng cột bảng và value đại diện cho nội dung cho các bảng ghi trong cột này. ContentValues Có thể được sử dụng để chèn và cập nhật các mục cơ sở dữ liệu.

## Các công cụ hỗ trợ cho việc thực hiện lập trình ứng dụng

### Android Studio – Dùng để lập trình ứng dụng trên hệ điều hành Android

Có nhiều công cụ để phát triển Android nhưng đến nay công cụ chính thức và mạnh mẽ nhất là Android Studio. Đây là IDE (Môi trường phát triển tích hợp) chính thức cho nền tảng Android, được phát triển bởi Google và được sử dụng để tạo phần lớn các ứng dụng mà bạn có thể sử dụng hàng ngày.

Android Studio lần đầu tiên được công bố tại hội nghị Google I/O vào năm 2013 và được phát hành cho công chúng vào năm 2014 sau nhiều phiên bản beta khác nhau. Trước khi được phát hành, các nhà phát triển Android thường sử dụng các công cụ như Eclipse IDE, một IDE Java chung cũng hỗ trợ nhiều ngôn ngữ lập trình khác.

Android Studio khiến việc tạo ứng dụng trở nên dễ dàng hơn đáng kể so với phần mềm không chuyên dụng. Đối với người mới bắt đầu, có rất nhiều thứ để học và nhiều thông tin có sẵn, thậm chí thông qua các kênh chính thức nhưng chúng có thể đã lỗi thời hoặc quá nhiều thông tin khiến họ cảm thấy choáng ngợp. Bài viết này sẽ giải thích ngắn gọn nhưng chi tiết về một số chức năng cơ bản của nó để bạn có thể nắm bắt được bước đầu trong công cuộc phát triển Android của mình.

Chức năng của Android Studio là cung cấp giao diện để tạo các ứng dụng và xử lý phần lớn các công cụ quản lý file phức tạp đằng sau hậu trường. Ngôn ngữ lập trình được sử dụng ở đây là Java và được cài đặt riêng trên thiết bị của bạn. Android Studio rất đơn giản, bạn chỉ cần viết, chỉnh sửa và lưu các dự án của mình và các file trong dự án đó. Đồng thời, Android Studio sẽ cấp quyền truy cập vào Android SDK.

Hãy coi đây là đuôi cho code Java cho phép nó chạy trơn tru trên các thiết bị Android và tận dụng lợi thế của phần cứng gốc. Bạn cần sử dụng ngôn ngữ lập trình Java để viết các chương trình, Android SDK có nhiệm vụ kết nối các phần này lại với nhau. Cùng lúc đó Android Studio kích hoạt để chạy code, thông qua trình giả lập hoặc qua một phần cứng kết nối với thiết bị. Sau đó, bạn cũng có thể “gỡ rối” chương trình khi nó chạy và nhận phản hồi giải thích sự cố, v.v… để bạn có thể nhanh chóng giải quyết vấn đề.

Google đã nỗ lực rất nhiều để làm cho Android Studio trở nên mạnh mẽ và hữu ích nhất có thể. Nó cung cấp những gợi ý trực tiếp trong khi viết code và thường đề xuất những thay đổi cần thiết để sửa lỗi hoặc làm code hiệu quả hơn. Ví dụ, nếu không sử dụng biến, biến đó sẽ được tô đậm bằng màu xám. Và khi bắt đầu gõ một dòng code, Android Studio sẽ cung cấp danh sách gợi ý tự hoàn thành để giúp bạn hoàn thiện dòng code đó. Chức năng này rất hữu ích khi bạn không nhớ được chính xác cú pháp hoặc để tiết kiệm thời gian.

Android Studio đã được tích hợp sẵn rất nhiều tính năng như tạo ra các máy ảo để chạy ứng dụng, kết nối với Github, Firebase,…

Có thể tải Android Studio tại đây: <https://developer.android.com/studio/>

### Github – Dùng để lưu trữ dự án và quản lý việc lập trình của các thành viên trong nhóm

**Github** là một hệ thống quản lý dự án và phiên bản code, hoạt động giống như một mạng xã hội cho lập trình viên. Các lập trình viên có thể clone lại mã nguồn từ một repository và Github chính là một dịch vụ máy chủ repository công cộng, mỗi người có thể tạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để có thể làm việc.

Github là một dịch vụ nổi tiếng cung cấp kho lưu trữ mã nguồn Git cho các dự án phần mềm. **Github có đầy đủ những tính năng của Git**, ngoài ra nó còn bổ sung những tính năng về social để các developer tương tác với nhau.

Github có 2 phiên bản: miễn phí và trả phí. Với phiên bản có phí thường được các doanh nghiệp sử dụng để tăng khả năng quản lý team cũng như phân quyền bảo mật dự án. Còn lại thì phần lớn chúng ta đều sử dụng Github với tài khoản miễn phí để lưu trữ source code.

Github cung cấp các tính năng social networking như feeds, followers, và network graph để các developer học hỏi kinh nghiệm của nhau thông qua lịch sử commit.

Nếu một comment để mô tả và giải thích một đoạn code. Thì với Github, commit message chính là phần mô tả hành động mà bạn thực hiện trên source code.

Github trở thành một yếu tố có sức ảnh hưởng lớn trong cộng động nguồn mở. Cùng với Linkedin, Github được coi là một sự thay thế cho CV của bạn. Các nhà tuyển dụng cũng rất hay tham khảo Github profile để hiểu về năng lực coding của ứng viên.

Giờ đây, kỹ năng sử dụng git và Github từ chỗ ưu thích sang bắt buộc phải có đối với các ứng viên đi xin việc.

**Tính năng của Github:**

**Github** được coi là một mạng xã hội dành cho lập trình viên lớn nhất và dễ dùng nhất với các tính năng cốt lõi như:

* Wiki, issue, thống kê, đổi tên project, project được đặt vào namespace là user.
* Watch project: theo dõi hoạt động của project của người khác. Xem quá trình người ta phát triển phầm mềm thế nào, project phát triển ra sao.
* Follow user: theo dõi hoạt động của người khác.

Có 2 cách tiếp cận GitHub: Tạo project của riêng mình Contribute cho project có sẵn: fork project có sẵn của người khác, sửa đổi, sau đó đề nghị họ cập nhật sửa đổi của mình (tạo pull request).

**Lịch sử của Github:**

Github được viết bằng Ruby on Rails và Erlang do Tom Preston-Werner, Chris Wanstrath, và PJ Hyett phát triển trang web được đưa ra và chạy chính thức vào tháng 4 năm 2008.

Tính đến thời điểm tháng 3 năm 2018 Github đang là dịch vụ máy chủ lưu trữ các mã nguồn lập trình lớn nhất thế giới. Với hơn 25 triệu người dùng và hơn 80 triệu mã nguồn dự án, Github đã trở thành một phần không thể thiêu đối với cộng đồng phát triển mã nguồn mở và cộng đồng lập trình viên trên toàn thế giới.

**Lợi ích của Github đối với lập trình viên:**

**Quản lý Sourcode dễ dàng:** khi bạn tạo một repo, toàn bộ source code của repo đó được lưu trên Github. Tại đây, bạn có thể coi lại quá trình mình đã làm việc thông qua các comment sau mỗi lần commit. Và cái hay ở đây, là nhiều người có thể cùng làm một repo. Lợi ích đầu tiên, chính là bạn biết được ai đã commit và commit cái gì. Tiếp theo, source của bạn có thể phát triển theo nhiều nhánh. Nguyên tắc làm việc với các nhánh như thế này: Bạn có thể rẽ nhiều nhánh để phát triển project. Nhưng cuối cùng, bạn phải merge lại vào nhánh MASTER để ra được project hoàn chỉnh.

**Github giúp chứng tỏ bạn là ai:** chẳng thể phủ nhận những lời hay ý đẹp bạn viết trong CV là cần thiết. Nhưng Source code luôn là minh chứng tốt nhất để thể hiện bạn là developer thật thụ. Hãy đầu tư cho mình một tài khoản Github thật ấn tượng và đưa đường dẫn vào trong CV, chẳng nhà tuyển dụng nào lại dại dột mà bỏ qua bạn đâu. Có rất nhiều công ty lớn trên thế giới xem đây là một yêu cầu trong quy trình tuyển dụng của họ. Nếu bạn có nhiều đóng góp cho cộng đồng hoặc có nhiều sản phẩm trên Github, sẽ là một lợi thế rất lớn so với các ứng viên khác. Vì bằng cách đăng tải các project của mình lên đây, bạn đã tạo cho mình một profile cá nhân vô cùng đáng tin cậy. Vì khi nhìn vào đó, nhà tuyển dụng sẽ biết được ngay thế mạnh của bạn là gì, và khả năng coding của bạn thế nào.

**Github giúp cả thiện kỹ năng code:** có hàng ngàn hàng vạn cách để học, học trên Github sẽ là một ý kiến không tồi trong thời đại này. Với hàng vạn open source projects, hàng trăm ngàn contributors, hàng tỉ commit mỗi ngày thì chỉ bằng việc xem. So sánh, học tập từ những thay đổi đó đã đem lại cho bạn hàng tá điều hay để cải thiện kỹ năng code của bản thân mình.

**Github là một kho tài nguyên tuyệt vời:** với chức năng Explore, bạn có thể theo dõi, tìm kiếm những open source projects theo đúng technology pattern mà bạn ưa thích. Đừng ngần ngại mà không tạo ngay cho mình một tài khoản Github. Tạo những project của riêng mình và chia sẻ với mọi người, hoặc bạn có thể thoải mái fork một project của một open source nào đó. Tạo pull request hoặc issues nếu như tìm được lỗi, cần support.

Có thể sử dụng Github tại đây: <https://github.com/>

### Enterprise Architect – Dùng để vẽ các mô hình phân tích

**Lịch sử phát triển:**

Phần mềm Enterprise Architect được phát triển bởi công ty Sparx Systems, được thành lập năm 1996 bởi Geoffrey Sparks, là một công ty của Australian có trụ sở tại Creswick, Victoria.

Sparx Systems có kinh nghiệm với hơn 10 năm phát triển phần mềm Enterprise Architect.

Phiên bản hiện tại của Enterprise Architect là phiên bản 9.1.

Hãng đưa ra khá nhiều phiên bản với tính năng khác nhau giúp người sử dụng có thể lựa chọn dễ dàng, ví dụ phiên bản dành cho doanh nghiệp, phiên bản dành cho cá nhân,…

**Một số tính năng nổi bật:**

Được đánh giá cao về hiệu suất và giá cả.

Cấu hình cài đặt không cao, có thể sử dụng trên những máy tính có cấu hình thấp.

Giao diện đồ họa trực quan, dễ sử dụng.

Được thiết kế theo mô hình chuẩn UML.

Hỗ trợ toàn bộ các biểu đồ trong UML.

Có thể tài và cài đặt Enterprise Architect tại đây: <https://sparxsystems.com/>

### Mockflow – Dùng để thiết kế giao diện ở dạng Mockup

Mockflow là một trang web giúp cho việc lên ý tưởng, thiết kế giao diện ở dạng Mockup, người dùng có thể sử dụng trang web để thiết kế giao diện rất nhanh, dễ dàng, chỉ cần kéo thả các thành phần như button, textview, edittext là có tạo ra một giao diện cơ bản.

Trang web của Mockflow: <https://www.mockflow.com/>

### Genymotion

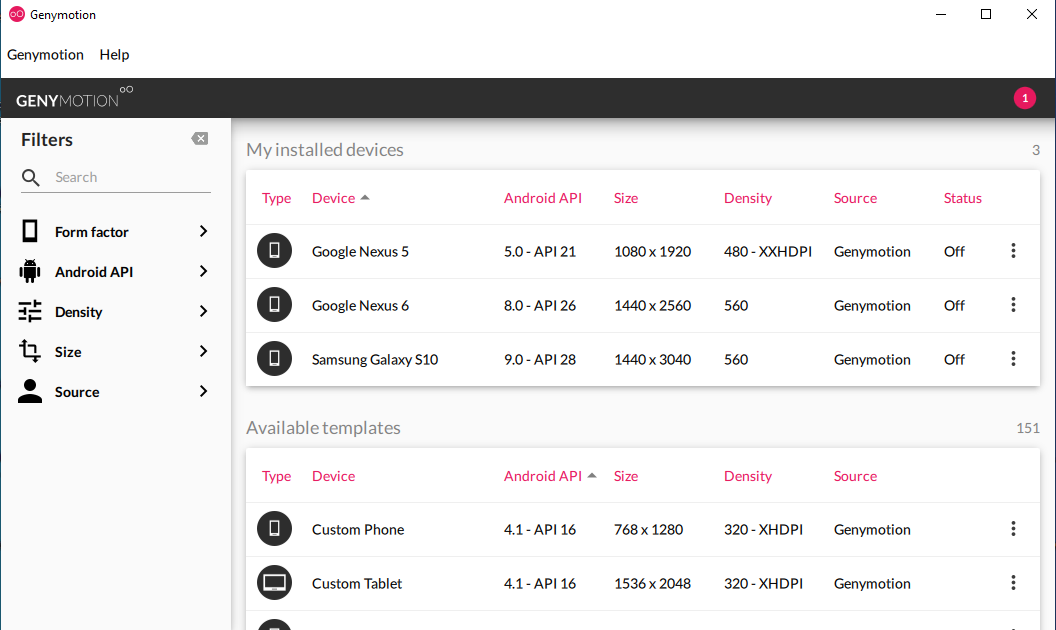
**Genymotion là một trong những trình giả lập Android miễn phí tốt nhất trên thị trường** . Phần mềm này vừa mạnh mẽ vừa dễ sử dụng, sẽ được các nhà phát triển Android tò mò muốn biết.

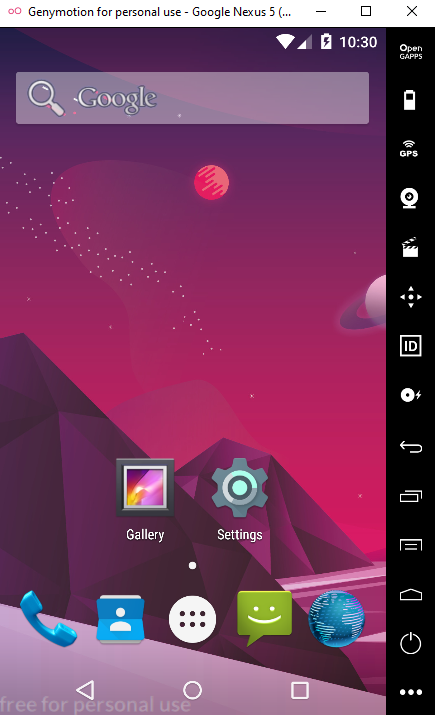
Genymotion rất dễ sử dụng và **có thể mô phỏng hàng chục điện thoại Android** (bao gồm các mẫu phổ biến như Google Nexus và dòng sản phẩm Samsung Galaxy). Bàn phím và chuột của máy tính của bạn được tự động phát hiện và truy cập internet được hỗ trợ mà không có vấn đề gì. Bạn có thể bật các tùy chọn định vị địa lý và thậm chí tăng hoặc giảm kích thước của cửa sổ trình mô phỏng khi các ứng dụng Android được khởi chạy.

Bộ giả lập Genymotion **không có các tùy chọn không cần thiết** khi khởi động, nhưng menu chính cho phép bạn thiết lập **ADB** (Trình gỡ lỗi Android), một tùy chọn chỉ hữu ích cho những người dùng cao cấp nhất. Trình mô phỏng Android khởi chạy trong thời gian kỷ lục mà không cần phải dành thời gian để định cấu hình phần mềm.

Genymotion là một **phần mềm mạnh mẽ** và chất lượng là đáng ngạc nhiên tốt xem xét nó miễn phí, nhưng chương trình vẫn chưa được cuối cùng vì vậy nó vẫn còn một chút lỗi. Xoay màn hình có thể gây ra sự cố trên một số điện thoại được mô phỏng và việc chọn menu thả xuống Android đôi khi khiến trình giả lập gặp sự cố. Ngoài những thiếu sót này, Genymotion tiếp cận sự hoàn hảo như xa như Android giả lập đi.

**Genymotion phù hợp cho tất cả người dùng** : dễ sử dụng nhưng có đủ tính năng nâng cao để đáp ứng các nhà phát triển và chuyên gia Android. Người mới đến thị trường giả lập này sẽ không gặp khó khăn trong việc thiết lập chính nó như là một sản phẩm tham khảo trong thị trường này.

Có thể tải và cài đặt Genymotion tại đây: <https://genymotion.softonic.vn/>



# : PHÂN TÍCH

## Phân tích yêu cầu bằng UML

### Mô hình Use case tổng quát

*Hình 3.1 Mô hình Usecase tổng quát*

### Danh sách tác nhân và mô tả

|  |  |  |
| --- | --- | --- |
| Tác nhân | Mô tả | Ghi chú |
| Sinh viên | Có thể sử dụng toàn bộ chức năng của ứng dụng khi đã đăng ký tài khoản | Có tham gia vào ứng dụng |

*Bảng 3.1 Danh sách tác nhân và mô tả*

### Danh sách Use case và mô tả

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Tên Use case | Mô tả ngắn gọn Use case | Ghi chú |
| UC01 | Đăng ký tài khoản | Mô tả các bước tác nhân đăng ký tài khoản để sử dụng ứng dụng |  |
| UC02 | Đăng nhập | Mô tả các bước tác nhân đăng nhập vào ứng dụng |  |
| UC03 | Quên mật khẩu | Mô tả các bước tác nhân tạo mật khẩu mới khi quên mật khẩu |  |
| UC04 | Thống kê thu chi | Mô tả các bước tác nhân xem thống kê thu chi theo ngày, tháng, năm |  |
| UC05 | Xem thông tin tài khoản | Mô tả các bước tác nhân xem thông tin tài khoản |  |
| UC06 | Cập nhật thông tin tài khoản | Mô tả các bước tác nhân cập nhật thông tin tài khoản |  |
| UC07 | Đổi mật khẩu | Mô tả các bước tác nhân đổi mật khẩu cho tài khoản |  |
| UC08 | Đăng xuất | Mô tả các bước tác nhân đăng xuất khỏi ứng dụng |  |
| UC09 | Thêm ví | Mô tả các bước tác nhân thêm ví để thực hiện việc thu chi |  |
| UC10 | Cập nhật ví | Mô tả các bước tác nhân cập nhật thông tin ví |  |
| UC11 | Xóa ví | Mô tả các bước tác nhân xóa ví khi không sử dụng |  |
| UC12 | Chuyển tiền | Mô tả các bước tác nhân chuyển tiền từ ví này sang ví khác |  |
| UC13 | Xem lịch sử chuyển tiền | Mô tả các bước tác nhân xem lịch sử chuyển tiền giữa các ví |  |
|  | Cập nhật thu chi |  |  |
| UC14 | Thêm thu chi | Mô tả các bước tác nhân thêm các thu chi |  |
| UC15 | Xóa thu chi | Mô tả các bước tác nhân xóa các thu chi khi không cần thiết |  |
| UC16 | Xem lịch sử thu chi | Mô tả các bước tác nhân xem lịch sử thu chi |  |
| UC17 | Thêm kế hoạch tiết kiệm | Mô tả các bước tác nhân |  |
| UC18 | Thu chi cho kế hoạch tiết kiệm | Mô tả các bước tác nhân |  |
| UC19 | Xem chi tiết kế hoạch tiết kiệm | Mô tả các bước tác nhân |  |
| UC20 | Xem lịch sử thu chi cho kế hoạch tiết kiệm | Mô tả các bước tác nhân |  |
| UC21 | Xóa kế hoạch tiết kiệm | Mô tả các bước tác nhân |  |
| UC22 | Thêm danh mục thu chi | Mô tả các bước tác nhân |  |
| UC23 | Xóa danh mục thu chi | Mô tả các bước tác nhân |  |
| UC24 | Cập nhật danh mục thu chi | Mô tả các bước tác nhân |  |

*Bảng 3.2 Danh sách Use case*

### Tình huống hoạt động (Use case)

#### *Use case UC01 – Đăng ký tài khoản*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC01 - Đăng ký tài khoản | | |
| Mô tả | Tác nhân muốn sử dụng ứng dụng thì phải có một tài khoản để sử dụng riêng cho bản thân mình, giúp bảo mật thông tin cá nhân và các thông tin khác | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã cài đặt và khởi động thành công ứng dụng | |
| Điều kiện sau | Đăng ký tài khoản khoản thành công cho tác nhân, lưu tài khoản vào cơ sở dữ liệu | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên mở ứng dụng đã được cài đặt trên thiết bị di động |  |
|  | 2. Hệ thống hiển thị giao diện đăng nhập |
| 3. Sinh viên nhấn nút “Đăng ký” |  |
|  | 4. Hệ thống hiển thị giao diện đăng ký |
| 5. Sinh viên nhập vào các thông tin (tên tài khoản, mật khẩu, mã số bí mật) |  |
| 6. Sinh viên nhấn nút “Đăng ký” |  |
|  | 7. Hệ thống kiểm tra dữ liệu nhập và nếu thông tin hợp lệ, thông báo “Tạo tài khoản thành công” |
| **Luồng sự kiện** **phụ** | 6.1 Sinh viên nhấn nút “Hủy” |  |
|  | 6.2 Quay lại giao diện đăng nhập ở bước 2 |
|  | 7.1 Hệ thống kiểm tra dữ liệu nhập và nếu tên đăng nhập hoặc mã số bí mật không hợp lệ, thông báo “Tài khoản này đã tồn tại” hoặc “Hãy chọn mã số bí mật khác” |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.3 Đặc tả Use case 01 – Đăng ký tài khoản*



*Hình 3.2 Mô hình Activity Usecase 01 – Đăng ký tài khoản*



#### *Use case UC02 – Đăng nhập*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC02 - Đăng nhập | | |
| Mô tả | Khi đã đăng ký thành công tài khoản, tác nhân đăng nhập bằng tài khoản vừa đăng ký để sử dụng ứng dụng | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng ký tài khoản | |
| Điều kiện sau | Đăng nhập thành công vào ứng dụng | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên mở ứng dụng đã được cài đặt trên thiết bị di động |  |
|  | 2. Hệ thống hiển thị giao diện đăng nhập |
| 3. Sinh viên nhập tên tài khoản và mật khẩu vào trường tài khoản và mật khẩu trên giao diện |  |
| 4. Có thể tích chọn “Nhớ mật khẩu” hoặc không tích, nhấn nút “Đăng nhập” |  |
|  | 5. Hệ thống kiểm tra dữ liệu nhập và nếu tài khoản hợp lệ, chuyển sang giao diện chính của ứng dụng, thông báo “Đăng nhập thành công” |
| **Luồng sự kiện** **phụ** |  | 5.1 Hệ thống kiểm tra dữ liệu nhập và nếu tài khoản không hợp lệ, thông báo “Tài khoản này không tồn tại” hoặc “Không đúng mật khẩu” |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.4 Đặc tả Use case 02 – Đăng nhập*



*Hình 3.3 Mô hình Activity Usecase 02 – Đăng nhập*



#### *Use case UC03 – Quên mật khẩu*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC03 – Quên mật khẩu | | |
| Mô tả | Khi tác nhân quên mật khẩu tài khoản đang sử dụng, tác nhân có thể tạo lại mật khẩu mới thông qua mã số bí mật lúc đăng ký tài khoản | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng ký tài khoản và có mã số bí mật | |
| Điều kiện sau | Tạo thành công mật khẩu mới cho tài khoản của tác nhân | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên mở ứng dụng đã được cài đặt trên thiết bị di động |  |
|  | 2. Hệ thống hiển thị giao diện đăng nhập |
| 3. Sinh viên nhấn nút “Quên mật khẩu” |  |
|  | 4. Hệ thống hiển thị giao diện đổi mật khẩu |
| 5. Sinh viên nhập vào các thông tin (mã số bí mật, mật khẩu mới, nhập lại mật khẩu) |  |
|  | 6. Sinh viên nhấn nút “Lưu” |  |
|  |  | 7. Hệ thống kiểm tra dữ liệu nhập và nếu thông tin hợp lệ, thông báo “Tạo mật khẩu thành công” |
| **Luồng sự kiện** **phụ** | 6.1 Sinh viên nhấn nút “Hủy” |  |
|  | 6.2 Quay lại giao diện đăng nhập ở bước 2 |
|  | 7.1 Hệ thống kiểm tra dữ liệu nhập và nếu mã số bí mật hoặc mật khẩu mới không hợp lệ, thông báo “Mã số bí mật sai” hoặc “Mật khẩu không khớp” |
| Luồng sự kiện ngoại lệ |  | |

#### *Use case UC04 – Thống kê thu chi*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC04 – Thống kê thu chi | | |
| Mô tả | Khi thực hiện việc thu chi, tác nhân có thể xem lại tổng thu, tổng chi của mình thèo ngày, tháng, năm | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng, có thực hiện việc thu chi | |
| Điều kiện sau | Xem được tổng thu, tổng chi theo ngày, tháng, năm | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng thống kê thu chi |  |
|  | 2. Hệ thống hiển thị giao diện thống kê thu chi |
| 3. Sinh viên chọn xem tổng thu, tổng chi theo ngày, tháng hoặc năm bằng cách chọn vào combobox |  |
|  | 4. Hệ thống hiển thị tổng thu, tổng chi theo ngày, tháng hoặc năm như đã chọn |
| **Luồng sự kiện** **phụ** |  |  |
| Luồng sự kiện ngoại lệ | Nếu chưa thực hiện việc thu chi thì sẽ không hiển thị dữ liệu tổng thu, tổng chi | |



*Hình 3.4 Mô hình Activity Usecase 03 – Thống kê thu chi*

#### *Use case UC05 – Xem thông tin tài khoản*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC05 – Xem thông tin tài khoản | | |
| Mô tả | Giúp tác nhân xem các thông tin của tài khoản đang sử dụng (họ tên, địa chỉ, email, số điện thoại, tổng số tiền đang có trong các ví) | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng | |
| Điều kiện sau | Tác nhân xem được các thông tin của tài khoản đang sử dụng | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý tài khoản |  |
|  | 2. Hệ thống hiển thị giao diện quản lý tài khoản |
| 3. Tại đây sinh viên có thể xem các thông tin của tài khoản mình đang sử dụng |  |
| **Luồng sự kiện phụ** |  |  |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.6 Đặc tả Use case 05 – Xem thông tin tài khoản*



*Hình 3.5 Mô hình Activity Usecase 04 – Xem thông tin tài khoản*

#### *Use case UC06 – Cập nhật thông tin tài khoản*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC06 – Cập nhật thông tin tài khoản | | |
| Mô tả | Giúp tác nhân cập nhật các thông tin của tài khoản mình đang sử dụng (tên, địa chỉ, email, số điện thoại) | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng | |
| Điều kiện sau | Tác nhân cập nhật thành công các thông tin tài khoản của mình, lưu thông tin vào cơ sở dữ liệu | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý tài khoản |  |
|  | 2. Hệ thống hiển thị giao diện quản lý tài khoản |
| 3. Sinh viên nhấn nút “Cập nhật thông tin” |  |
|  | 4. Hệ thống hiển thị giao diện cập nhật thông tin tài khoản |
| 5. Sinh viên nhập các thông tin của tài khoản |  |
| 6. Sinh viên nhấn nút “Lưu” |  |
|  | 7. Hệ thống kiểm tra dữ liệu nhập và nếu thông tin hợp lệ, thông báo “Cập nhật thành công” |
| **Luồng sự kiện phụ** | 6.1 Sinh viên nhấn nút “Hủy” |  |
|  | 6.2 Quay lại giao diện quản lý tài khoản ở bước 2 |
|  | 7.1 Hệ thống kiểm tra dữ liệu nhập và nếu thông tin không hợp lệ, thông báo “Cập nhật không thành công” |
|  | 7.2 Quay lại giao diện thông tin của tài khoản ở bước 4 |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.7 Đặc tả Usecase 05 – Cập nhật thông tin tài khoản*



*Hình 3.6 Mô hình Activity Usecase 05 – Cập nhật thông tin tài khoản*



#### *Use case UC07 – Đổi mật khẩu*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC07 – Đổi mật khẩu | | |
| Mô tả | Khi mật khẩu của tài khoản đã bị người khác biết được hoặc quá khó nhớ thì tác nhân có thể đổi mật khẩu | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng | |
| Điều kiện sau | Đổi thành công mật khẩu mới cho tác nhân | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý tài khoản |  |
|  | 2. Hệ thống hiển thị giao diện quản lý tài khoản |
| 3. Sinh viên nhấn nút “Đổi mật khẩu” |  |
|  | 4. Hệ thống hiển thị giao diện đổi mật khẩu |
| 5. Sinh viên nhập vào các thông tin (mã số bí mật, mật khẩu mới, xác nhận mật khẩu) |  |
| 6. Sinh viên nhấn nút “Lưu” |  |
|  | 7. Hệ thống kiểm tra dữ liệu nhập và nếu thông tin hợp lệ, thông báo “Tạo mật khẩu thành công” |
| **Luồng sự kiện phụ** | 6.1 Sinh viên nhấn nút “Hủy” |  |
|  | 6.2 Quay lại giao diện quản lý tài khoản ở bước 2 |
|  | 7.1 Hệ thống kiểm tra dữ liệu nhập và nếu mã số bí mật và mật khẩu mới không hợp lệ, thông báo “Mã số bí mật sai” hoặc “Mật khẩu không khớp” |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.8 Đặc tả Use case 06 –*

#### *Use case UC08 – Đăng xuất*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC08 – Đăng xuất | | |
| Mô tả | Dùng để đăng xuất khỏi tài khoản đang sử dụng | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng | |
| Điều kiện sau | Đăng xuất khỏi tài khoản đang sử dụng | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý tài khoản |  |
|  | 2. Hệ thống hiển thị giao diện quản lý tài khoản |
| 3. Sinh viên nhấn nút “Đăng xuất” |  |
|  | 4. Hệ thống hiển thị giao diện đăng nhập |
| **Luồng sự kiện phụ** |  |  |
| Luồng sự kiện ngoại lệ |  | |

#### *Usecase UC09 – Thêm ví*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC09 - Thêm ví | | |
| Mô tả | Với các ví có sẵn, tác nhân có thể thêm ví khác phù hợp với nhu cầu sử dụng của mình, số ví tối đa là 5 bao gồm cả ví có sẵn | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng | |
| Điều kiện sau | Thêm thành công ví cho tác nhân sử dụng vào việc thu chi, lưu ví vào cơ sở dữ liệu | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý ví |  |
|  | 2. Hệ thống hiển thị giao diện quản lý ví |
| 3. Sinh viên nhấn nút “Thêm” (nút dấu + màu xanh phía dưới góc phải màn hình) |  |
|  | 4. Hệ thống hiển thị giao diện thêm ví |
| 5. Sinh viên nhập vào các thông tin cần thiết của ví (tên ví, mô tả, số tiền ban đầu) |  |
| 6. Sinh viên nhấn nút “Lưu” |  |
|  | 7. Hệ thống kiểm tra dữ liệu nhập và nếu thông tin hợp lệ, thông báo “Thêm ví thành công” |
| **Luồng sự kiện phụ** | 6.1 Sinh viên nhấn nút “Hủy” |  |
|  | 6.2 Quay lại giao diện quản lý ví ở bước 2 |
|  | 7.1 Hệ thống kiểm tra dữ liệu nhập và nếu tên ví không hợp lệ, thông báo “Tên ví này đã tồn tại” |
| Luồng sự kiện ngoại lệ | Nếu số hiện có đã đạt số lượng tối đa là 5, tại bước 6, khi sinh viên nhấn nút “Lưu” sẽ thông báo “Số ví đã đạt tối đa” | |

*Bảng 3.9 Đặc tả Use case 09 – Thêm ví*



*Hình 3.7 Mô hình Activity Usecase 07 – Tạo ví*



#### *Usecase UC10 – Cập nhật ví*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC10 – Cập nhật ví | | |
| Mô tả | Nếu tác nhân có nhu cầu cập nhật thông tin ví đã tạo, tác nhân có thể cập nhật lại thông tin của ví, ví có sẵn chỉ có thể cập nhật mô tả, không được cập nhật tên | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng và đã tạo ví | |
| Điều kiện sau | Cập nhật thành công thông tin của ví, lưu ví vào cơ sở dữ liệu | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý ví |  |
|  | 2. Hệ thống hiển thị giao diện quản lý ví |
| 3. Sinh viên tìm và nhấn giữ vào ví cần cập nhật, nhấn nút “Cập nhật” |  |
|  | 4. Hệ thống hiển thị giao diện thông tin của ví đã chọn |
| 5. Sinh viên cập nhật thông tin ví |  |
| 6. Sinh viên nhấn nút “Lưu” |  |
|  | 7. Hệ thống kiểm tra dữ liệu nhập và nếu thông tin hợp lệ, thông báo “Cập nhật thành công” |
| **Luồng sự kiện phụ** | 6.1 Sinh viên nhấn nút “Hủy” |  |
|  | 6.2 Quay lại giao diện quản lý ví ở bước 2 |
|  | 7.1 Hệ thống kiểm tra dữ liệu nhập và tên ví không hợp lệ, thông báo “Tên ví này đã tồn tại” |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.10 Đặc tả Use case 08 – Cập nhật ví*



*Hình 3.8 Mô hình Activity Usecase 08 – Cập nhật ví*



#### *Usecase UC11 – Xóa ví*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC11 – Xóa ví | | |
| Mô tả | Khi không còn sử dụng một ví nào đó, tác nhân có thể xóa nó khỏi ứng dụng, ví có sẵn không thể xóa | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng và đã tạo ví | |
| Điều kiện sau | Xóa thành công ví không sử dụng của tác nhân, xóa ví khỏi cơ sở dữ liệu | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý ví |  |
|  | 2. Hệ thống hiển thị giao diện quản lý ví |
| 3. Sinh viên tìm và nhấn giữ vào ví cần xóa, nhấn nút “Xóa” |  |
|  | 4. Hệ thống hiển thị thông báo “Bạn có chắc chắn muốn xóa ví này ?” |
| 5. Sinh viên nhấn nút “Có” |  |
|  | 6. Hệ thống hiển thị thông báo “Xóa thành công” |
| **Luồng sự kiện phụ** | 5.1 Sinh viên nhấn nút “Không” |  |
|  | 5.2 Quay lại giao diện quản lý ví ở bước 2 |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.11 Đặc tả Usecase 11 – Xóa ví*



*Hình 3.9 Mô hình Activity Usecase 09 – Xóa ví*



#### *Use case UC12 – Chuyển tiền*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC12 – Chuyển tiền | | |
| Mô tả | Khi có một ví nào đó hết tiền và đang cần tiền thì tác nhân có thể chuyển tiền từ ví khác sang | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng, ví chuyển phải còn tiền | |
| Điều kiện sau | Chuyển tiền thành công cho ví, cập nhật số tiền ở ví chuyển và ví nhận | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý ví |  |
|  | 2. Hệ thống hiển thị giao diện quản lý ví |
| 3. Sinh viên tìm và nhấn giữ vào ví cần chuyển, nhấn nút “Chuyển tiền” |  |
|  |  | 4. Hệ thống hiển thị giao diện chuyển tiền |
| 5. Sinh viên chọn ví cần nhận tiền, nhập số tiền muốn chuyển |  |
| 6. Sinh viên nhấn nút “Chuyển” |  |
|  | 7. Hệ thống kiểm tra dữ liệu nhập và nếu thông tin hợp lệ, thông báo “Chuyển tiền thành công” |
| **Luồng sự kiện phụ** | 6.1 Sinh viên nhấn nút “Hủy” |  |
|  | 6.2 Quay lại giao diện quản lý ví ở bước 2 |
|  | 7.1 Hệ thống kiểm tra dữ liệu nhập và nếu ví nhận và số tiền chuyển không hợp lệ, thông báo “Tên ví chuyển bị trùng” hoặc “Số tiền chuyển vượt quá số tiền trong ví” |
| Luồng sự kiện ngoại lệ |  | |

#### Use case UC13 – Xem lịch sử chuyển tiền

|  |  |  |
| --- | --- | --- |
| **Use case:** UC13 – Xem lịch sử chuyển tiền | | |
| Mô tả | Khi thực hiện việc chuyển tiền, tác nhân có thể xem lại lịch sử chuyển tiền đã thực hiện để có thể dễ dàng quản lý được số tiền đang có | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng | |
| Điều kiện sau | Xem được lịch sử chuyển tiền, lưu vào cơ sở dữ liệu | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý ví |  |
|  | 2. Hệ thống hiển thị giao diện quản lý ví |
| 3. Sinh viên nhấn nút “Lịch sử chuyển tiền” (góc trái bên dưới màn hình) |  |
|  |  | 4. Hệ thống hiển thị giao diện lịch sử chuyển tiền |
| **Luồng sự kiện phụ** |  |  |
| Luồng sự kiện ngoại lệ |  | |

#### *Usecase UC10 – Cập nhật thu chi (xem xét bỏ)*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC10 – Cập nhật thu chi | | |
| Mô tả | Khi đã thêm khoản thu hoặc chi, nếu chưa hài lòng về thông tin khoản thu hoặc chi đã thêm, tác nhân có thể cập nhật lại thông tin của khoản thu hoặc chi | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng và đã thêm khoản thu hoặc chi | |
| Điều kiện sau | Cập nhật thành công khoản thu hoặc chi của tác nhân | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý thu chi |  |
|  | 2. Hệ thống hiển thi giao diện quản lý thu chi |
| 3. Sinh viên tìm và nhấn vào khoản thu hoặc chi cần cập nhật |  |
|  | 4. Hệ thống hiển thị giao diện thông tin của khoản thu hoặc chi đã chọn |
| 5. Sinh viên cập nhật lại thông tin theo ý muốn của mình |  |
| 6. Sinh viên nhấn nút Lưu |  |
|  | 7. Hệ thống kiểm tra dữ liệu nhập và nếu thông tin hợp lệ, thông báo Lưu thành công |
| **Luồng sự kiện phụ** | 6.1 Sinh viên nhấn nút Hủy |  |
|  | 6.2 Quay lại giao diện quản lý thu chi ở bước 2 |
|  | 7.1 Hệ thống kiểm tra dữ liệu nhập và nếu thông tin không hợp lệ, thông báo Lưu không thành công |
|  | 7.2 Quay lại giao diện thông tin của khoản thu chi ở bước 4 |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.12 Đặc tả Usecase 10 – Cập nhật thu chi*



*Hình 3.10 Mô hình Activity Usecase 10 – Cập nhật thu chi*

#### *Use case UC14 – Thêm thu chi*

|  |  |  |
| --- | --- | --- |
| **Use case:** UC11 – Thêm thu chi | | |
| Mô tả | Khi thực hiện một khoản thu hoặc chi ngoài thực tế, tác nhân có thể thêm khoản này vào ứng dụng để có thể dễ dàng quản lý được việc chi tiêu của mình | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng | |
| Điều kiện sau | Thêm thành công khoản thu hoặc chi của tác nhân, lưu vào cơ sở dữ liệu | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý thu chi |  |
|  | 2. Hệ thống hiển thị giao diện quản lý thu chi |
| 3. Sinh viên nhấn vào nút tròn dấu cộng tại góc phải màn hình |  |
|  | 4. Hệ thống hiển thị giao diện thêm thu chi |
| 5. Sinh viên nhập vào các thông tin cần thiết (loại thu hoặc chi, số tiền, danh mục thu chi, chọn ví, ngày giờ sẽ lấy ngay thời điểm hiện tại) |  |
| 6. Sinh viên nhấn nút Thêm |  |
|  | 7. Hệ thống kiểm tra dữ liệu và nếu hợp lệ, thông báo Thêm thành công |
| **Luồng sự kiện phụ** | 6.1 Sinh viên nhấn nút Hủy |  |
|  | 6.2 Quay lại giao diện quản lý thu chi ở bước 2 |
|  | 7.1 Hệ thống kiểm tra dữ liệu và nếu không hợp lệ, thông báo Thêm không thành công |
|  | 7.2 Quay lại giao diện thêm thu chi ở bước 4 |
| Luồng sự kiện ngoại lệ | 5.1 Nếu chưa có ví, sinh viên thực hiện chức năng Tạo ví ở Usecase 07  5.2 Nếu muốn thêm một danh mục thu chi khác, sinh viên thực hiện chức năng Thêm danh mục thu chi ở Usecase 17 | |

*Bảng 3.13 Đặc tả Usecase 11 – Thêm thu chi*

*Hình 3.11 Mô hình Activity Usecase 11 – Thêm thu chi*

******

#### Use case UC15 – Xóa thu chi

|  |  |  |
| --- | --- | --- |
| **Use case:** UC12 – Xóa thu chi | | |
| Mô tả | Khi đã thêm khoản thu hoặc chi, nhưng không muốn tồn tại khoản thu chi này thì tác nhân có thể xóa khoản thu hoặc chi này đi | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng và đã thêm khoản thu hoặc chi | |
| Điều kiện sau | Xóa thành công khoản thu hoặc chi của tác nhân | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý thu chi |  |
|  | 2. Hệ thống hiển thị giao diện quản lý thu chi |
| 3. Sinh viên tìm và nhấn giữ vào khoản thu hoặc chi mà mình muốn xóa |  |
|  | 4. Hệ thống hiển thị thông báo Bạn có muốn xóa hay không ? |
| 5. Sinh viên nhấn nút Có |  |
|  | 6. Hệ thống thông báo Xóa thành công |
| **Luồng sự kiện phụ** | 5.1 Sinh viên nhấn nút Hủy |  |
|  | 5.2 Quay lại giao diện quản lý thu chi ở bước 2 |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.14 Đặc tả Usecase 12 – Xóa thu chi*



*Hình 3.12 Mô hình Activity Usecase 12 – Xóa thu chi*

#### Use case UC16 – Xem lịch sử thu chi

|  |  |  |
| --- | --- | --- |
| **Use case:** UC16 – Lịch sử thu chi | | |
| Mô tả | Khi thực hiện thêm thu hoặc chi, ứng dụng sẽ lưu lại các khoản thu hoặc chi mà tác nhân đã thêm giúp tác nhân xem được các khoản chi tiêu của mình | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng | |
| Điều kiện sau | Tác nhân xem được lịch sử thu chi của mình theo ngày, theo tháng | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý thu chi |  |
|  | 2. Hệ thống hiển thị giao diện quản lý thu chi |
| 3. Sinh viên có thể chọn xem tất cả lịch sử thu chi của mình hoặc xem theo ngày bằng cách chọn nút “Theo ngày”, chọn ngày muốn xem, nhấn nút “Chọn” |  |
|  | 4. Hệ thống hiển thị lịch sử thu chi theo ngày |
| **Luồng sự kiện phụ** |  | 4.1 Hệ thống hiển thị lịch sử thu chi theo ngày nhưng dữ liệu trống nếu không có khoản thu chi nào được thêm |
| Luồng sự kiện ngoại lệ |  | |

#### Use case UC17 – Thêm kế hoạch tiết kiệm

|  |  |  |
| --- | --- | --- |
| **Use case:** UC17 – Thêm kế hoạch tiết kiệm | | |
| Mô tả | Thêm kế hoạch thu chi giúp tác nhân có thể quản lý việc thu chi của mình một cách hợp lý, tránh việc tác nhân chi tiêu quá nhiều sẽ gây mất cân bằng tài chính | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng | |
| Điều kiện sau | Thêm thành công kế hoạch tiết kiệm của tác nhân | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý kế hoạch tiết kiệm |  |
|  | 2. Hệ thống hiển thị giao diện quản lý kế hoạch tiết kiệm |
| 3. Sinh viên nhấn nút Thêm kế hoạch tiết kiệm (nút dấu + màu xanh phía dưới góc màn hình) |  |
|  | 4. Hệ thống hiển thị giao diện thêm kế hoạch tiết kiệm |
| 5. |  |
|  |  |
|  |  |
|  |  |
| **Luồng sự kiện phụ** |  |  |
|  |  |
|  |  |
|  |  |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.15 Đặc tả Usecase 13 – Thêm kế hoạch*

#### Use case UC18 – Thu chi cho kế hoạch tiết kiệm

|  |  |  |
| --- | --- | --- |
| **Use case:** UC18 – Thu chi cho kế hoạch tiết kiệm | | |
| Mô tả |  | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng và đã thêm kế hoạch | |
| Điều kiện sau | Cập nhật thành công kế hoạch của tác nhân | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. |  |
|  | 2. |
| 3. |  |
|  | 4. |
| 5. |  |
|  | 6. |
| 7. |  |
|  | 8. |
| 9. |  |
| 10. |  |
|  | 11. |
| **Luồng sự kiện phụ** | 10.1 |  |
|  | 10.2 |
|  | 11.1 |
|  | 11.2 |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.16 Đặc tả Usecase 14 – Cập nhật kế hoạch*

#### Use case UC19 – Xem chi tiết kế hoạch tiết kiệm

|  |  |  |
| --- | --- | --- |
| **Use case:** UC19 – Xem chi tiết kế hoạch tiết kiệm | | |
| Mô tả |  | |
| Tác nhân | Sinh viên | |
| Điều kiện trước |  | |
| Điều kiện sau |  | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. |  |
|  | 2. |
| 3. |  |
| **Luồng sự kiện phụ** |  |  |
| Luồng sự kiện ngoại lệ |  | |

#### Use case UC20 – Xem lịch sử thu chi cho kế hoạch tiết kiệm

|  |  |  |
| --- | --- | --- |
| **Use case:** UC20 – Xem lịch sử thu chi cho kế hoạch tiết kiệm | | |
| Mô tả | Khi đã thêm kế hoạch, nhưng không muốn tồn kế hoạch này thì tác nhân có thể xóa kế hoạch này đi | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng và đã thêm kế hoạch | |
| Điều kiện sau | Xóa thành công kế hoạch của tác nhân | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý kế hoạch |  |
|  | 2. Hệ thống hiển thị giao diện quản lý kế hoạch tiết kiệm |
| 3. |  |
|  | 4. |
| 5. |  |
|  | 6. |
| 7. |  |
|  | 8. |
| 9. |  |
|  | 10. |
| **Luồng sự kiện phụ** | 9.1 |  |
|  | 9.2 |
| Luồng sự kiện ngoại lệ |  | |

#### Usecase UC21 – Xóa kế hoạch tiết kiệm

|  |  |  |
| --- | --- | --- |
| **Use case:** UC21 – Xóa kế hoạch tiết kiệm | | |
| Mô tả | Khi đã thêm kế hoạch, nhưng không muốn tồn kế hoạch này thì tác nhân có thể xóa kế hoạch này đi | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng và đã thêm kế hoạch | |
| Điều kiện sau | Xóa thành công kế hoạch của tác nhân | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý kế hoạch |  |
|  | 2. Hệ thống hiển thị giao diện quản lý kế hoạch tiết kiệm |
| 3. |  |
|  | 4. |
| 5. |  |
|  | 6. |
| 7. |  |
|  | 8. |
| 9. |  |
|  | 10. |
| **Luồng sự kiện phụ** | 9.1 |  |
|  | 9.2 |
| Luồng sự kiện ngoại lệ |  | |

*Bảng 3.17 Đặc tả Usecase 15 – Xóa kế hoạch*

#### Usecase UC22 – Thêm danh mục thu chi

|  |  |  |
| --- | --- | --- |
| **Use case:** UC22 – Thêm danh mục thu chi | | |
| Mô tả | Thêm danh mục thu chi giúp người dùng có thể chọn cho việc thu chi, ví dụ chi về việc ăn uống, học tập, … | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng | |
| Điều kiện sau | Thêm thành công danh mục thu chi, lưu danh mục vào cơ sở dữ liệu | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý danh mục thu chi |  |
|  | 2. Hệ thống hiển thị giao diện quản lý danh mục thu chi |
| 3. Ở phía dưới giao diện, sinh viên nhập vào tên danh mục, chọn loại khoản (thu hoặc chi) |  |
| 4. Sinh viên nhấn nút Thêm (nút dấu + màu xanh phía dưới góc màn hình) |  |
|  | 5. Hệ thống hiển thị thông báo Bạn có muốn thêm danh mục này ? |
| 6. Sinh viên nhấn nút Có |  |
|  | 7. Hệ thống kiểm tra dữ liệu và nếu hợp lệ, thông báo Thêm danh mục thành công |
| **Luồng sự kiện phụ** | 6.1 Sinh viên nhấn nút Không |  |
|  | 6.2 Quay lại giao diện quản lý danh mục thu chi ở bước 2 |
|  | 7.1 Hệ thống kiểm tra dữ liệu và nếu không hợp lệ, thông báo “Thêm danh mục thất bại” |
| Luồng sự kiện ngoại lệ |  | |





#### Usecase UC23 – Xóa danh mục thu chi

|  |  |  |
| --- | --- | --- |
| **Use case:** UC23 – Xóa danh mục thu chi | | |
| Mô tả | Khi những danh mục thu chi không phù hợp với mục đích sử dụng của tác nhân thì có thể xóa danh mục đó đi | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng và đã thêm danh mục | |
| Điều kiện sau | Xóa thành công danh mục thu chi | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. Sinh viên chọn chức năng quản lý danh mục thu chi |  |
|  | 2. Hệ thống hiển thị giao diện quản lý danh mục thu chi |
| 3. Sinh viên tìm và nhấn giữ vào danh mục cần xóa |  |
|  | 4. Hệ thống hiển thị thông báo Bạn có chắc chắn muốn xóa danh mục này ? |
| 5. Sinh viên nhấn nút Có |  |
|  | 6. Hệ thống hiển thị thông báo Xóa thành công |
| **Luồng sự kiện phụ** | 5.1 Sinh viên nhấn nút Không |  |
|  | 5.2 Quay lại giao diện quản lý danh mục thu chi ở bước 2 |
| Luồng sự kiện ngoại lệ |  | |





#### Use case UC24 – Cập nhật danh mục thu chi

|  |  |  |
| --- | --- | --- |
| **Use case:** UC24 – Cập nhật danh mục thu chi | | |
| Mô tả |  | |
| Tác nhân | Sinh viên | |
| Điều kiện trước | Tác nhân đã đăng nhập thành công vào ứng dụng | |
| Điều kiện sau |  | |
| Luồng sự kiện chính | Tác nhân | Hệ thống |
| 1. |  |
|  | 2. |
| 3. |  |
|  | 4. |
| 5. |  |
| 6. |  |
|  | 7. |
| **Luồng sự kiện phụ** | 6.1 |  |
|  | 6.2 |
|  | 7.1 |
|  | 7.2 |
| Luồng sự kiện ngoại lệ |  | |

# THIẾT KẾ VÀ HIỆN THỰC

## Class Diagram

## Mô hình liên kết thực thể

## Mô hình cơ sở dữ liệu

### Cơ sở dữ liệu quan hệ

TaiKhoan (tenTaiKhoan, matKhau, maSoBiMat, hoTen, diaChi, soDienThoai, email, ngaySinh)

KhoanThuChi (maThuChi, loaiThuChi, moTaKhoanThuChi, soTienThuChi, ngayThucHien, tenTaiKhoan, maVi, maDanhMuc)

DanhMucThuChi (maDanhMuc, tenDanhMuc, loaiDanhMuc, tenTaiKhoan)

Vi (maVi, tenVi, moTaVi, soTien, tenTaiKhoan)

KeHoachTietKiem (maKeHoach, tenKeHoach, soTienKeHoach, thoiGianKeHoach, tenTaiKhoan)

### Các ràng buộc toàn vẹn trong cơ sở dữ liệu

**TaiKhoan**

tenTaiKhoan: PRIMARY KEY, UNIQUE

matKhau: NOT NULL

maSoBiMat: NOT NULL, UNIQUE

hoTen: NULL

diaChi: NULL

soDienThoai: NULL

email: NULL

ngaySinh: NULL

**KhoanThuChi**

maThuChi: PRIMARY KEY, UNIQUE

loaiThuChi: NOT NULL

soTienThuChi: NOT NULL

moTaKhoanThuChi: NOT NULL

ngayThucHien: NOT NULL

tenTaiKhoan: FOREIGN KEY

maVi: FOREIGN KEY

maDanhMuc: FOREIGN KEY

**DanhMucThuChi**

maDanhMuc: PRIMARY KEY, UNIQUE

tenDanhMuc: NOT NULL

loaiDanhMuc: NOT NULL

tenTaiKhoan: FOREIGN KEY

**Vi**

maVi: PRIMARY KEY, UNIQUE

tenVi: NOT NULL, UNIQUE

moTaVi: NOT NULL

soTien: NOT NULL

tenTaiKhoan: FOREIGN KEY

**KeHoachTietKiem**

maKeHoach: PRIMARY KEY, UNIQUE

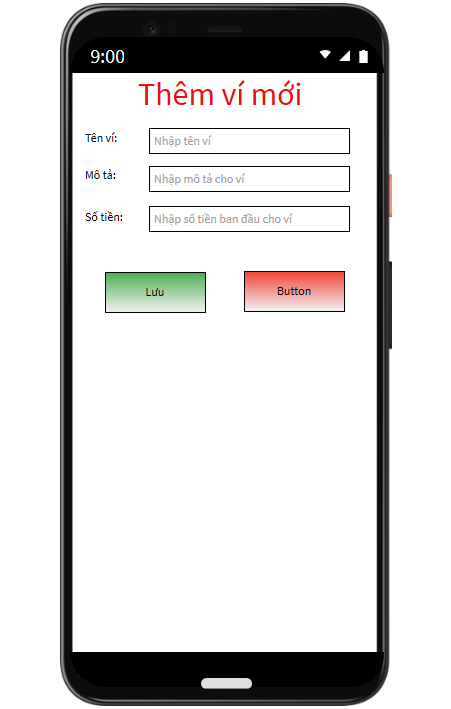
tenKeHoach: NOT NULL, UNIQUE

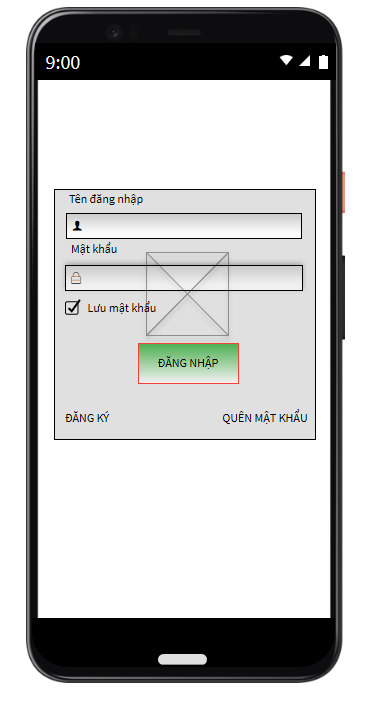
soTienKeHoach: NOT NULL

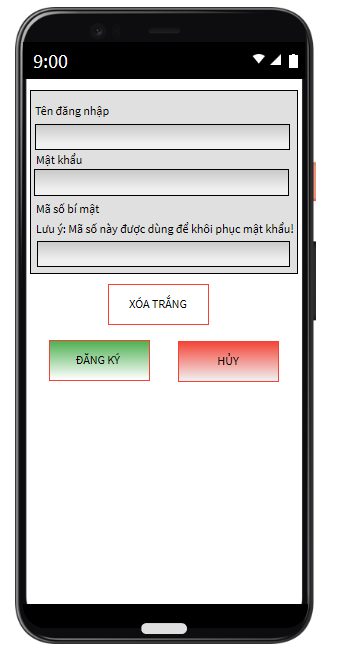
thoiGianKeHoach: NOT NULL

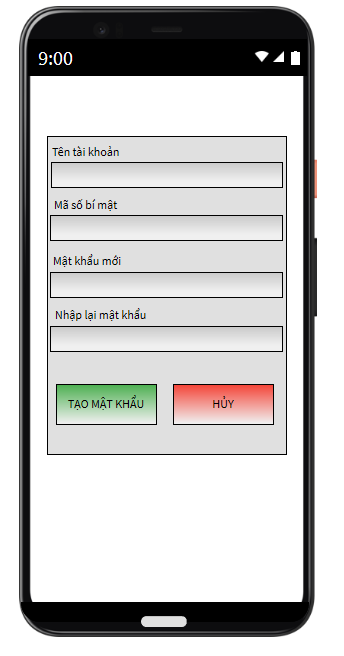
tenTaiKhoan: FOREIGN KEY

## Thiết kế giao diện dạng Mockup









## Một số giao diện của ứng dụng

## Phân luồng màn hình của ứng dụng

## Cấu hình phần cứng, phần mềm

### Phần cứng

### Phần mềm

# : KẾT LUẬN

## Kết quả đạt được

Trình bày các chức năng đã hoàn thành trong đồ án hoặc kết quả nghiên cứu.

## Hạn chế của đồ án

Trình bày những hạn chế, công việc chưa hoàn tất trong đồ án

## Hướng phát triển

Trình bày các định hướng phát triển cho hệ thống hoặc hướng nghiên cứu trong tương lai.

# TÀI LIỆU THAM KHẢO

Các tài liệu Tiếng Việt

1. Họ và Tên tác giả (Thứ tự theo Tên). Tên sách. Tên nhà xuất bản, năm xuất bản.

Các tài liệu Tiếng Anh

Các tài liệu từ Internet

1. Website: <https://stackoverflow.com/>
2. Website: <https://developer.android.com/>

# PHỤ LỤC