BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA: CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN MÔN**

**LẬP TRÌNH DI ĐỘNG NÂNG CAO**

Nhóm 10 - Hệ đại trà

**ĐỀ TÀI**

**“XÂY DỰNG ỨNG DỤNG QUẢN LÝ CHI TIÊU CÁ NHÂN”**

**GIẢNG VIÊN : ThS Nguyễn Hữu Trung**

**SINH VIÊN THỰC HIỆN**

| **STT** | **Họ và Tên** | **MSSV** |
| --- | --- | --- |
| **1** | **Trần Diệp Phương Vy** | **20110232** |
| **2** | **Phạm Nhật Tiến** | **20110735** |
| **3** | **Đặng Hữu Giàu** | **20110636** |
| **4** | **Trần Văn Quảng** | **20110160** |

**TP.HCM, ngày 20 tháng 5 năm 2024**

# **MỤC LỤC**

[**MỤC LỤC 2**](#_xd1bhkqqhqwa)

[**CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI 5**](#_ov35zaeizcz)

[1. Giới thiệu phần mềm 5](#_igbwzyhgm67m)

[2. Tính cấp thiết của phần mềm 5](#_b35v512b7uli)

[**CHƯƠNG 2: ĐẶC TẢ YÊU CẦU 6**](#_pd3kaq9acvyp)

[2.1. Nhận diện tác nhân và chức năng trong sơ đồ use case 6](#_nvujp4uzomoo)

[2.2. Lược đồ use case 7](#_kfn0hoexat4t)

[2.3. Chi tiết lược đồ use case 8](#_dsh1edvpjnva)

[2.3.1. Use case Đăng ký 8](#_kavc75nvwdao)

[2.3.2. Use case Đăng nhập 9](#_j6au45oojk1m)

[2.3.3. Use case Quên mật khẩu 10](#_3eszputpzv4s)

[2.3.4. Use case Đổi thông tài khoản 11](#_5ugps334qpsf)

[2.3.5. Use case Quản lý nguồn tiền 12](#_obz3zq3q9pkd)

[2.3.5.1. Use case Xem danh sách nguồn tiền 12](#_3gztzakc5u88)

[2.3.5.2. Use case Tạo nguồn tiền 13](#_u0e4voeolmzr)

[2.3.5.3. Use case Sửa Nguồn tiền 14](#_l052iricj16n)

[2.3.5.4. Use case Xoá Nguồn tiền 16](#_2fee8myh0zv7)

[2.3.5.5. Use case Chuyển giao Nguồn tiền cá nhân 17](#_2s6ol7k22m5p)

[2.3.5.6. Use case Xem lịch sử chuyển giao Nguồn tiền cá nhân 18](#_wx3wqvb1iw9t)

[2.3.6. Use case quản lý giao dịch cá nhân 20](#_lq282zlugb8z)

[2.3.6.1. Use case Tạo giao dịch mới 20](#_vybnf2auyypt)

[2.3.6.2. Use case Sửa giao dịch 21](#_8qa33d2bk1xy)

[2.3.6.3. Use case Xoá giao dịch 22](#_rzzj88u17egp)

[2.3.6.4. Use case Xem danh sách giao dịch và lọc theo các tiêu chí 24](#_f24s2dlse5ti)

[2.3.6.5. Use case Xuất báo cáo Excel các giao dịch 25](#_7hyqu0ochsnc)

[2.3.7. Use case Quản lý các chi tiêu thông thường 26](#_n82w0clpd7bh)

[2.3.7.1. Use case Thêm khoản chi tiêu thông thường 26](#_v6p14p79h1d1)

[2.3.7.2. Use case Sửa khoản chi tiêu thông thường 28](#_uwyy2u7vhvgw)

[2.3.7.3. Use case Xoá khoản chi tiêu thông thường 29](#_ah1zjw8yhf9)

[2.3.8. Use case Xem biểu đồ giao dịch cá nhân 30](#_dq5jfq1osoc0)

[2.3.9. Use case Bảng phân tích và báo cáo 31](#_231mf94ci1c1)

[2.3.10. Use case Hỗ trợ người dùng 33](#_bmskoez8onw3)

[2.4. Chi tiết lược đồ sequence 34](#_l826df8bc4oc)

[2.4.1. Chức năng Đăng ký 34](#_pgxlhfaye93c)

[2.4.2. Chức năng Đăng nhập 34](#_e61ageuel4u7)

[2.4.3. Chức năng Quên mật khẩu 35](#_t5zcxv73fy8f)

[2.4.4. Chức năng Đổi thông tài khoản 35](#_gno486lzyeqh)

[2.4.5. Chức năng nguồn tiền 36](#_fhmbuz5knbdj)

[2.4.5.1. Chức năng Xem danh sách nguồn tiền 36](#_mkjeyk8k60xo)

[2.4.5.2. Chức năng Tạo nguồn tiền 36](#_rdbqgzqaki4u)

[2.4.5.3. Chức năng Sửa Nguồn tiền 37](#_7n74d2bkpydu)

[2.4.5.4. Chức năng Xoá Nguồn tiền 37](#_qhvarnofbbva)

[2.4.5.5. Chức năng Chuyển giao Nguồn tiền cá nhân 38](#_dc0dtj1m8bbk)

[2.4.5.6. Chức năng Xem lịch sử chuyển giao Nguồn tiền cá nhân 38](#_urdfe75q8wac)

[2.4.6. Quản lý giao dịch cá nhân 39](#_k7j7r0bgg4f5)

[2.4.6.1. Chức năng Tạo giao dịch mới 39](#_3gdav6v5s8bl)

[2.4.6.2. Chức năng Sửa giao dịch 40](#_3ybiwtwyp196)

[2.4.6.3. Chức năng Xoá giao dịch 41](#_ty1olbwpeiqi)

[2.4.6.4. Chức năng Xem danh sách giao dịch và lọc theo các tiêu chí 42](#_ml1x53hozmwt)

[2.4.6.5. Chức năng Xuất báo cáo Excel các giao dịch 42](#_g9e2pdp05t11)

[2.4.7. Quản lý các chi tiêu thông thường 43](#_q9ljadlu3qxr)

[2.4.7.1. Chức năng Thêm khoản chi tiêu thông thường 43](#_6qo54ymez1q)

[2.4.7.2. Chức năng Sửa khoản chi tiêu thông thường 43](#_j72aqzcp40bn)

[2.4.7.3. Chức năng Xoá khoản chi tiêu thông thường 44](#_b3dcajmh1t)

[2.4.8. Chức năng Xem biểu đồ giao dịch cá nhân 45](#_4eow3r5w9upp)

[2.4.9. Chức năng Bảng phân tích và báo cáo 45](#_kqpw4a549pnp)

[2.4.10. Chức năng Hỗ trợ người dùng 46](#_jhbn1hby6ymx)

[**CHƯƠNG 3: THIẾT KẾ GIAO DIỆN 48**](#_uioi5sltyl4l)

[3.1. Xác thực 48](#_aa05szn85a7l)

[3.2. Trang chủ 49](#_y9o86am9098k)

[3.3. Trang Phân tích 50](#_tj670bfszvao)

[3.4. Trang Transaction 51](#_ywsmjqi5iipp)

[3.5. Trang Profile 52](#_nqoq7edwt37c)

[3.6. Trang nguồn tiền 54](#_h7t1hn72sx7i)

[3.7. Trang Regular Payment 55](#_unal6tgctb5s)

[3.8. Notification 56](#_pxfji1k3ete6)

[**CHƯƠNG 4: API DOCUMENT 57**](#_brixgcmg821b)

[4.1. Tổng quan về Directus 57](#_memgj43kas8)

[4.1.1. Directus là gì 57](#_5hr0pqfkg7kw)

[4.1.2. Cách thức hoạt động của Directus 57](#_xc298k68jag3)

[4.1.3. Tại sao nên sử dụng Directus 58](#_ylji6e6do0l3)

[4.2. Ứng dụng Directus vào đồ án 58](#_q6jzgmo68f10)

[4.2.1. Transaction 58](#_ia2n5g1e8er8)

[4.2.2. Account 59](#_b75s8qm97tgq)

[4.2.3. User 60](#_vyppthf04unu)

[4.2.4. Payment 61](#_tu5mtadxloen)

[4.2.5. Cron Job 62](#_npyyjee2zmyv)

[**CHƯƠNG 5: CODE CÁC CHỨC NĂNG CHÍNH 63**](#_sgjhdj3bm2g)

[5.1. Chức năng đăng nhập , đăng ký 63](#_71xil5b154q2)

[5.2. Chức năng transaction 70](#_fjvftox3sxhl)

[5.3. Chức năng account 87](#_noxaqgjbuag5)

[5.4. Chức năng tranfer history 92](#_x7epsvvc42xo)

[5.5. Chức năng payment 97](#_zf1pggqmlgqg)

[5.6. Chức năng notification 110](#_fzx9b0lo0xqv)

[5.7. Chức năng object hook 111](#_5mrhdloizm9q)

[5.8. Chức năng cron job 116](#_92d8nc8itx91)

[**KẾT LUẬN 119**](#_aujwkufa5zfy)

# 

# **CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI**

## **Giới thiệu phần mềm**

Phần mềm quản lý chi tiêu cá nhân là một công cụ hiệu quả giúp người dùng theo dõi, phân tích và quản lý các khoản chi tiêu của họ. Với giao diện thân thiện và tính năng linh hoạt, phần mềm này đem lại trải nghiệm người dùng tuyệt vời, từ việc nhập dữ liệu đến phân tích và đưa ra báo cáo chi tiết.

## **Tính cấp thiết của phần mềm**

* Tiện lợi và tiết kiệm thời gian: Phần mềm giúp người dùng tiết kiệm thời gian so với việc ghi chép truyền thống. Việc nhập dữ liệu và xem lại thông tin chi tiêu trở nên đơn giản và nhanh chóng hơn bao giờ hết.
* Kiểm soát chi tiêu: Phần mềm cung cấp cái nhìn tổng quan về mẫu chi tiêu cá nhân của người dùng, giúp họ nhận biết và kiểm soát những lãng phí không cần thiết.
* Tạo kế hoạch tài chính: Dựa trên dữ liệu chi tiêu, phần mềm có thể đề xuất các kế hoạch tài chính hợp lý, từ việc tiết kiệm cho mục tiêu cụ thể đến quản lý nợ nần.

# 

# **CHƯƠNG 2: ĐẶC TẢ YÊU CẦU**

## **Nhận diện tác nhân và chức năng trong sơ đồ use case**

| **STT** | **Tên** | **Ý nghĩa** |
| --- | --- | --- |
| 1 | Admin | Người quản trị có quyền cao nhất của hệ thống, có thể thực hiện mọi tác vụ và sử dụng mọi chức năng mà hệ thống cung cấp. |
| 2 | Member | Các thành viên đã đăng nhập vào hệ thống |

**Bảng 1.1**: Các tác nhân và chức năng

## **Lược đồ use case**



**Hình 1.1.**: Lược đồ use case

## **Chi tiết lược đồ use case**

### **Use case Đăng ký**

| **Usecase Id** | UC01 |
| --- | --- |
| **Usecase Name** | **Đăng ký** |
| **Description** | Người dùng đăng ký tài khoản hệ thống để bắt đầu sử dụng |
| **Priority** | **High** |
| **Actor** | User |
| **Trigger** |  |
| **Pre-condition(s)** | Người dùng chưa có tài khoản trong hệ thống. |
| **Post-condition(s)** | Tài khoản mới được tạo thành công và người dùng có thể đăng nhập vào hệ thống. |
| **Basic Flow** | 1. Người dùng truy cập vào trang đăng ký; 2. Người dùng nhập thông tin yêu cầu bao gồm tên, email, và mật khẩu hoặc chọn phương thức đăng ký bằng tài khoản Google; 3. Nếu người dùng chọn đăng ký bằng tài khoản Google: 4. Hệ thống chuyển hướng đến trang đăng nhập Google; 5. Người dùng chọn tài khoản Google để đăng nhập; 6. Hệ thống lấy thông tin từ tài khoản Google và tạo tài khoản mới trong hệ thống với thông tin đã được cung cấp từ Google; 7. Hệ thống chuyển hướng đến trang đăng nhập. 8. Người dùng xác nhận thông tin và bấm nút "Đăng ký"; 9. Hệ thống kiểm tra tính hợp lệ của thông tin nhập vào; 10. Hệ thống tạo tài khoản mới cho người dùng và chuyển hướng đến trang đăng nhập. |
| **Alternative Flow** | Nếu email đã được sử dụng bởi một tài khoản khác, hệ thống sẽ cảnh báo người dùng và yêu cầu nhập lại email khác.  Nếu thông tin không hợp lệ hoặc không đầy đủ, hệ thống sẽ yêu cầu người dùng nhập lại hoặc cung cấp thông tin bổ sung. |
| **Exception Flow** |  |
| **Business Rules** |  |
| **Non-Functional Requirement** |  |

### **Use case Đăng nhập**

| **Usecase Id** | UC02 |
| --- | --- |
| **Usecase Name** | **Đăng nhập** |
| **Description** | Người dùng đăng nhập vào hệ thống để bắt đầu sử dụng |
| **Priority** | **High** |
| **Actor** | User , Admin |
| **Trigger** |  |
| **Pre-condition(s)** | Người dùng phải có một tài khoản của hệ thống. |
| **Post-condition(s)** |  |
| **Basic Flow** | 1. Người dùng nhập tên tài khoản và mật khẩu; 2. Người dùng bấm phím **Enter** hay nhấn nút **Login** trên màn hình; 3. Hệ thống kiểm tra tính hợp lệ của tài khoản; 4. Hệ thống cho phép người dùng truy cập vào hệ thống: redirect đến trang được yêu cầu (ví dụ: nếu là admin thì hệ thống sẽ redirect đến trang quản trị dự án). |
| **Alternative Flow** | Nếu xác định tên tài khoản hoặc mật khẩu không hợp lệ, hệ thống sẽ cảnh báo để người dùng đăng nhập lại hoặc gợi ý tìm lại mật khẩu nếu người dùng quên mật khẩu.  Nếu người dùng đã đăng nhập thành công vào hệ thống mà vẫn gửi request đăng nhập thì hệ thống sẽ redirect đến trang mặc định. |
| **Exception Flow** |  |
| **Business Rules** |  |
| **Non-Functional Requirement** |  |

### **Use case Quên mật khẩu**

| **Usecase Id** | UC03 |
| --- | --- |
| **Usecase Name** | **Quên mật khẩu** |
| **Description** | Người dùng quên mật khẩu và cần thiết lập lại mật khẩu mới để có thể truy cập vào hệ thống. |
| **Priority** | **High** |
| **Actor** | User |
| **Trigger** |  |
| **Pre-condition(s)** | Người dùng quên mật khẩu và yêu cầu đặt lại mật khẩu. |
| **Post-condition(s)** |  |
| **Basic Flow** | 1. Người dùng truy cập vào trang quên mật khẩu; 2. Người dùng nhập địa chỉ email đã đăng ký trong hệ thống; 3. Hệ thống kiểm tra tính hợp lệ của địa chỉ email và gửi một liên kết đặt lại mật khẩu đến địa chỉ email đã cung cấp; 4. Người dùng kiểm tra email và truy cập vào liên kết để thiết lập lại mật khẩu; 5. Người dùng nhập mật khẩu mới và xác nhận mật khẩu mới; 6. Hệ thống lưu mật khẩu mới và chuyển hướng đến trang đăng nhập. |
| **Alternative Flow** | Nếu địa chỉ email không tồn tại trong hệ thống, hệ thống sẽ cảnh báo người dùng và yêu cầu nhập lại địa chỉ email khác.  Nếu người dùng không nhận được email, họ có thể yêu cầu gửi lại email đặt lại mật khẩu.. |
| **Exception Flow** |  |
| **Business Rules** |  |
| **Non-Functional Requirement** |  |

### **Use case Đổi thông tài khoản**

| **Usecase Id** | UC04 |
| --- | --- |
| **Usecase Name** | **Đổi thông tin tài khoản** |
| **Description** | Người dùng muốn thay đổi thông tin cá nhân trong tài khoản của mình trong hệ thống. |
| **Priority** | **High** |
| **Actor** | User |
| **Trigger** | Người dùng muốn thay đổi thông tin cá nhân trong tài khoản. |
| **Pre-condition(s)** | Người dùng đã đăng nhập vào hệ thống và có tài khoản trong hệ thống. |
| **Post-condition(s)** | Thông tin tài khoản được cập nhật thành công và người dùng có thể tiếp tục sử dụng hệ thống. |
| **Basic Flow** | 1. Người dùng truy cập vào trang quản lý thông tin cá nhân; 2. Người dùng chọn chức năng "Chỉnh sửa thông tin"; 3. Người dùng chỉnh sửa các thông tin cá nhân cần thay đổi như tên, email, hoặc mật khẩu; 4. Người dùng xác nhận các thay đổi và bấm nút "Lưu"; 5. Hệ thống kiểm tra tính hợp lệ của các thông tin mới; 6. Hệ thống cập nhật thông tin tài khoản và hiển thị thông báo cập nhật thành công. |
| **Alternative Flow** |  |
| **Exception Flow** | Nếu thông tin nhập vào không hợp lệ, hệ thống sẽ cảnh báo người dùng và yêu cầu nhập lại hoặc cung cấp thông tin bổ sung. |
| **Business Rules** |  |
| **Non-Functional Requirement** |  |

**Bảng 1.3.5**: Use case Đổi thông tin tài khoản

### **Use case Quản lý nguồn tiền**

#### **Use case Xem danh sách nguồn tiền**

| **Usecase Id** | UC05 |
| --- | --- |
| **Usecase Name** | **Xem danh sách nguồn tiền cá nhân** |
| **Description** | Use Case này mô tả tình huống khi người dùng muốn xem tổng quan về danh sách nguồn tiền cá nhân. |
| **Priority** | **High** |
| **Actor** | User |
| **Trigger** |  |
| **Pre-condition(s)** | Người dùng phải có một tài khoản của hệ thống.  Người dùng đã nhập và lưu trữ các nguồn tiền cá nhân trước đó. |
| **Post-condition(s)** |  |
| **Basic Flow** | 1. Người dùng khởi động ứng dụng quản lý tài chính và đăng nhập vào tài khoản của mình. 2. Người dùng truy cập vào menu ứng dụng và chọn tùy chọn "Xem danh sách nguồn tiền". 3. Hệ thống truy xuất và hiển thị danh sách nguồn tiền cá nhân của người dùng trên giao diện ứng dụng. 4. Người dùng xem tổng quan về các nguồn tiền cá nhân gồm tên nguồn tiền, số dư, và thông tin khác liên quan. 5. Người dùng có thể tương tác với danh sách này, chẳng hạn như thêm, chỉnh sửa hoặc xóa nguồn tiền cá nhân. |
| **Alternative Flow** | Nếu không có nguồn tiền cá nhân nào được lưu trữ, hệ thống sẽ hiển thị một thông báo trống rỗng và cung cấp tùy chọn để người dùng thêm mới nguồn tiền cá nhân. |
| **Exception Flow** |  |
| **Business Rules** | * Người dùng chỉ có thể xem danh sách nguồn tiền cá nhân của chính mình. * Các nguồn tiền cá nhân được hiển thị theo thứ tự ưu tiên hoặc mặc định đã được người dùng thiết lập. |
| **Non-Functional Requirement** |  |

#### **Use case Tạo nguồn tiền**

| **Usecase Id** | UC06 |
| --- | --- |
| **Usecase Name** | **Tạo nguồn tiền cá nhân** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn tạo một nguồn tiền mới với các thông tin như tên nguồn tiền, số tiền hiện tại, icon và màu sắc để phân biệt, loại tiền tệ và tùy chọn bật/tắt tính năng cộng nguồn tiền này vào tổng tài sản. |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** | Người dùng chọn tùy chọn "Tạo Nguồn tiền" trong menu ứng dụng. |
| **Pre-condition(s)** | Người dùng phải có một tài khoản của hệ thống.  Người dùng đã truy cập vào màn hình quản lý nguồn tiền hoặc màn hình tạo nguồn tiền mới. |
| **Post-condition(s)** |  |
| **Basic Flow** | 1. Người dùng khởi động ứng dụng quản lý tài chính và đăng nhập vào tài khoản của mình. 2. Người dùng truy cập vào menu ứng dụng và chọn tùy chọn "Tạo Nguồn tiền". 3. Hệ thống hiển thị màn hình tạo nguồn tiền mới với các trường thông tin cần điền. 4. Người dùng nhập tên nguồn tiền vào trường "Tên nguồn tiền". 5. Người dùng nhập số tiền hiện tại của nguồn tiền vào trường "Số tiền hiện tại". 6. Người dùng chọn icon và màu sắc để phân biệt nguồn tiền này với các nguồn tiền khác. 7. Người dùng chọn loại tiền tệ cho nguồn tiền. 8. Người dùng có thể bật hoặc tắt tính năng cộng nguồn tiền này vào tổng tài sản bằng cách chọn tùy chọn "On/Off" tương ứng. 9. Người dùng hoàn tất việc nhập thông tin và chọn tùy chọn "Tạo" để tạo nguồn tiền mới. 10. Hệ thống ghi nhận thông tin nguồn tiền mới và lưu trữ vào cơ sở dữ liệu. |
| **Alternative Flow** | Nếu người dùng không nhập đầy đủ thông tin yêu cầu (ví dụ: bỏ trống trường "Tên nguồn tiền"), hệ thống sẽ hiển thị một thông báo lỗi và yêu cầu người dùng nhập đầy đủ thông tin trước khi tạo nguồn tiền mới. |
| **Exception Flow** |  |
| **Business Rules** | * Người dùng chỉ có thể tạo nguồn tiền mới cho chính mình. * Tên nguồn tiền phải là duy nhất, không trùng lặp với các nguồn tiền đã tồn tại. |
| **Non-Functional Requirement** |  |

#### **Use case Sửa Nguồn tiền**

| **Usecase Id** | UC07 |
| --- | --- |
| **Usecase Name** | **Sửa nguồn tiền cá nhân** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn sửa thông tin của một nguồn tiền đã tồn tại với các thông tin như tên nguồn tiền, số tiền hiện tại, icon và màu sắc để phân biệt, và tùy chọn bật/tắt tính năng cộng nguồn tiền này vào tổng tài sản. |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** |  |
| **Pre-condition(s)** | Người dùng phải có một tài khoản của hệ thống.  Người dùng đã truy cập vào màn hình quản lý nguồn tiền hoặc màn hình chi tiết nguồn tiền đã tồn tại. |
| **Post-condition(s)** |  |
| **Basic Flow** | 1. Người dùng khởi động ứng dụng quản lý tài chính và đăng nhập vào tài khoản của mình. 2. Người dùng truy cập vào menu ứng dụng hoặc màn hình chi tiết nguồn tiền đã tồn tại. 3. Hệ thống hiển thị thông tin chi tiết về nguồn tiền bao gồm tên nguồn tiền, số tiền hiện tại, icon và màu sắc, và tình trạng tính năng cộng nguồn tiền vào tổng tài sản. 4. Người dùng có thể sửa đổi thông tin như tên nguồn tiền, số tiền hiện tại, icon và màu sắc, và tình trạng tính năng cộng nguồn tiền vào tổng tài sản. 5. Người dùng hoàn tất việc sửa đổi thông tin và chọn tùy chọn "Lưu" để cập nhật thông tin nguồn tiền. |
| **Alternative Flow** | Nếu người dùng không thay đổi thông tin nào, người dùng có thể chọn tùy chọn "Hủy" để thoát khỏi màn hình sửa thông tin nguồn tiền mà không có bất kỳ thay đổi nào được áp dụng. |
| **Exception Flow** |  |
| **Business Rules** | * Người dùng chỉ có thể tạo nguồn tiền mới cho chính mình. * Tên nguồn tiền phải là duy nhất, không trùng lặp với các nguồn tiền đã tồn tại. |
| **Non-Functional Requirement** |  |

**Bảng 1.3.5.3**: Use case **Sửa nguồn tiền cá nhân**

#### **Use case Xoá Nguồn tiền**

| **Usecase Id** | UC08 |
| --- | --- |
| **Usecase Name** | **Xoá nguồn tiền cá nhân** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn xoá một nguồn tiền khỏi danh sách nguồn tiền hiện có trong ứng dụng quản lý tài chính. |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** | Người dùng chọn tùy chọn "Xoá nguồn tiền" trong menu ứng dụng hoặc truy cập vào màn hình quản lý nguồn tiền đã tồn tại. |
| **Pre-condition(s)** | Người dùng phải có một tài khoản của hệ thống.  Người dùng đã truy cập vào chi tiết nguồn tiền muốn xoá |
| **Post-condition(s)** |  |
| **Basic Flow** | 1. Người dùng khởi động ứng dụng quản lý tài chính và đăng nhập vào tài khoản của mình. 2. Người dùng truy cập vào menu ứng dụng hoặc màn hình chi tiết nguồn tiền đã tồn tại. 3. Hệ thống hiển thị thông tin chi tiết về nguồn tiền bao gồm tên nguồn tiền, số tiền hiện tại, icon và màu sắc, và tình trạng tính năng cộng nguồn tiền vào tổng tài sản. 4. Người dùng chọn tùy chọn "Xoá" hoặc biểu tượng xoá để tiến hành xoá nguồn tiền. 5. Hệ thống hiển thị cảnh báo xác nhận để đảm bảo người dùng muốn xoá nguồn tiền này. 6. Người dùng xác nhận việc xoá nguồn tiền. 7. Hệ thống xoá nguồn tiền khỏi danh sách nguồn tiền hiện có và cập nhật thông tin tài chính liên quan. |
| **Alternative Flow** | Nếu người dùng không muốn xoá nguồn tiền, người dùng có thể chọn tùy chọn "Hủy" hoặc thoát khỏi màn hình xoá nguồn tiền mà không có bất kỳ thay đổi nào được áp dụng. |
| **Exception Flow** |  |
| **Business Rules** | * Người dùng chỉ có thể xoá nguồn tiền mới cho chính mình. |
| **Non-Functional Requirement** |  |

#### **Use case Chuyển giao Nguồn tiền cá nhân**

| **Usecase Id** | UC09 |
| --- | --- |
| **Usecase Name** | **Chuyển giao nguồn tiền cá nhân** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn chuyển giao số tiền từ một nguồn tiền đã tồn tại sang một nguồn tiền khác trong ứng dụng quản lý tài chính. |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** |  |
| **Pre-condition(s)** | Người dùng đã đăng nhập vào ứng dụng quản lý tài chính.  Người dùng đã truy cập vào màn hình quản lý nguồn tiền hoặc màn hình chi tiết nguồn tiền đã tồn tại. |
| **Post-condition(s)** | Hệ thống ghi nhận và cập nhật số tiền của các nguồn tiền sau khi người dùng chuyển giao. |
| **Basic Flow** | 1. Người dùng khởi động ứng dụng quản lý tài chính và đăng nhập vào tài khoản của mình. 2. Người dùng truy cập vào menu ứng dụng hoặc màn hình chi tiết nguồn tiền đã tồn tại. 3. Hệ thống hiển thị thông tin chi tiết về nguồn tiền, bao gồm tên nguồn tiền, số tiền hiện có, icon và màu sắc. 4. Người dùng chọn tùy chọn "Chuyển giao" hoặc biểu tượng tương ứng để tiến hành chuyển giao nguồn tiền. 5. Hệ thống yêu cầu người dùng chọn nguồn tiền đích (nguồn tiền nhận) và nhập số tiền cần chuyển. 6. Người dùng xác nhận thông tin chuyển giao. 7. Hệ thống cập nhật số tiền của nguồn tiền gốc và nguồn tiền đích theo số tiền chuyển giao. |
| **Alternative Flow** | Nếu người dùng không muốn tiếp tục chuyển giao, người dùng có thể chọn tùy chọn "Hủy" hoặc thoát khỏi màn hình chuyển giao nguồn tiền mà không có bất kỳ thay đổi nào được áp dụng. |
| **Exception Flow** |  |
| **Business Rules** | * Người dùng chỉ có thể chuyển giao số tiền từ nguồn tiền thuộc sở hữu của mình đến nguồn tiền khác trong ứng dụng. * Số tiền chuyển giao không được vượt quá số tiền hiện có trong nguồn tiền gốc. |
| **Non-Functional Requirement** |  |

**Bảng 1.3.5.5**: Use case **Chuyển giao nguồn tiền cá nhân**

#### **Use case Xem lịch sử chuyển giao Nguồn tiền cá nhân**

| **Usecase Id** | UC010 |
| --- | --- |
| **Usecase Name** | **Xem lịch sử chuyển giao nguồn tiền cá nhân** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn xem lịch sử các giao dịch chuyển giao nguồn tiền trong ứng dụng quản lý tài chính. |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** | Người dùng chọn tùy chọn "Xoá nguồn tiền" trong menu ứng dụng hoặc truy cập vào màn hình quản lý nguồn tiền đã tồn tại. |
| **Pre-condition(s)** | Người dùng đã đăng nhập vào ứng dụng quản lý tài chính.  Người dùng đã truy cập vào màn hình lịch sử chuyển giao nguồn tiền hoặc màn hình chuyển giao nguồn tiền. |
| **Post-condition(s)** | Hệ thống hiển thị lịch sử các giao dịch chuyển giao nguồn tiền theo thời gian, bao gồm thông tin về nguồn tiền gốc, nguồn tiền đích, số tiền chuyển giao và thời gian thực hiện giao dịch. |
| **Basic Flow** | 1. Người dùng khởi động ứng dụng quản lý tài chính và đăng nhập vào tài khoản của mình. 2. Người dùng truy cập vào màn hình lịch sử chuyển giao nguồn tiền. 3. Hệ thống hiển thị danh sách các giao dịch chuyển giao nguồn tiền theo thời gian, từ giao dịch mới nhất đến cũ hơn. 4. Người dùng có thể xem chi tiết từng giao dịch chuyển giao bằng cách chọn vào mỗi giao dịch. 5. Hệ thống hiển thị thông tin chi tiết về giao dịch chuyển giao, bao gồm nguồn tiền gốc, nguồn tiền đích, số tiền chuyển giao và thời gian thực hiện. 6. Người dùng có thể quay lại danh sách lịch sử chuyển giao bằng cách chọn nút "Quay lại" hoặc tương tự. |
| **Alternative Flow** | Nếu không có giao dịch chuyển giao nguồn tiền nào trong lịch sử, hệ thống sẽ hiển thị một thông báo tương ứng để thông báo cho người dùng. |
| **Exception Flow** |  |
| **Business Rules** | * Hệ thống phải ghi nhận và hiển thị chính xác các giao dịch chuyển giao nguồn tiền trong lịch sử. * Lịch sử chuyển giao nguồn tiền chỉ hiển thị các giao dịch đã được thực hiện trong ứng dụng quản lý tài chính. * Người dùng chỉ có thể xem lịch sử chuyển giao nguồn tiền của tài khoản của mình. |
| **Non-Functional Requirement** |  |

### **Use case quản lý giao dịch cá nhân**

#### **Use case Tạo giao dịch mới**

| **Usecase Id** | UC011 |
| --- | --- |
| **Usecase Name** | **Tạo giao dịch mới** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn tạo một giao dịch mới trong ứng dụng quản lý tài chính, bao gồm thông tin sau: số tiền giao dịch, loại giao dịch (thu nhập hoặc chi tiêu), nguồn tiền, phân loại giao dịch, thời gian, comment (nếu có) và ảnh chụp (nếu có). |
| **Priority** | **High** |
| **Actor** | User |
| **Trigger** |  |
| **Pre-condition(s)** | * Người dùng đã đăng nhập vào ứng dụng quản lý tài chính. * Người dùng đã truy cập vào màn hình tạo giao dịch mới. |
| **Post-condition(s)** |  |
| **Basic Flow** | 1. Người dùng khởi động ứng dụng quản lý tài chính và đăng nhập vào tài khoản của mình. 2. Người dùng truy cập vào màn hình tạo giao dịch mới. 3. Người dùng nhập số tiền giao dịch. 4. Người dùng chọn loại giao dịch (thu nhập hoặc chi tiêu). 5. Người dùng chọn nguồn tiền cho giao dịch. 6. Người dùng chọn phân loại giao dịch. 7. Người dùng chọn thời gian của giao dịch. 8. Người dùng có thể nhập comment (nếu có) vào giao dịch. 9. Người dùng có thể chụp ảnh (nếu có) liên quan đến giao dịch. 10. Người dùng xác nhận và tạo giao dịch mới. 11. Hệ thống ghi nhận thành công giao dịch mới trong ứng dụng. |
| **Alternative Flow** | Nếu người dùng không nhập đủ thông tin bắt buộc, hệ thống sẽ hiển thị một thông báo lỗi và yêu cầu người dùng cung cấp đầy đủ thông tin cần thiết. |
| **Exception Flow** |  |
| **Business Rules** | * Hệ thống phải ghi nhận và lưu trữ thông tin giao dịch mới một cách chính xác và an toàn. * Giao dịch mới sẽ được tự động cộng hoặc trừ vào tổng tài sản của người dùng, tùy thuộc vào loại giao dịch (thu nhập hoặc chi tiêu). |
| **Non-Functional Requirement** |  |

**Bảng 1.3.6.1**: Use case **Thêm giao dịch mới**

#### **Use case Sửa giao dịch**

| **Usecase Id** | UC012 |
| --- | --- |
| **Usecase Name** | **Sửa giao dịch** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn sửa thông tin của một giao dịch đã được tạo trước đó trong ứng dụng quản lý tài chính. |
| **Priority** | **High** |
| **Actor** | User |
| **Trigger** |  |
| **Pre-condition(s)** | * Người dùng đã đăng nhập vào ứng dụng quản lý tài chính. * Người dùng đã truy cập vào màn hình xem danh sách giao dịch và chọn một giao dịch cần sửa. |
| **Post-condition(s)** | Hệ thống cập nhật thành công thông tin của giao dịch sau khi được sửa đổi. |
| **Basic Flow** | 1. Người dùng truy cập vào màn hình xem danh sách giao dịch. 2. Người dùng chọn một giao dịch cần sửa. 3. Hệ thống hiển thị các thông tin chi tiết của giao dịch cho người dùng. 4. Người dùng chỉnh sửa các thông tin cần thiết của giao dịch, bao gồm số tiền, loại giao dịch, nguồn tiền, phân loại giao dịch, thời gian, comment và ảnh chụp. 5. Người dùng xác nhận và lưu các thay đổi. 6. Hệ thống cập nhật thông tin của giao dịch và hiển thị thông báo thành công cho người dùng |
| **Alternative Flow** | Nếu người dùng không nhập đủ thông tin bắt buộc hoặc thông tin không hợp lệ, hệ thống sẽ hiển thị một thông báo lỗi và yêu cầu người dùng cung cấp thông tin đúng và đủ. |
| **Exception Flow** |  |
| **Business Rules** | * Hệ thống phải cập nhật và lưu trữ thông tin giao dịch sau khi được sửa đổi một cách chính xác và an toàn. |
| **Non-Functional Requirement** |  |

#### **Use case Xoá giao dịch**

| **Usecase Id** | UC013 |
| --- | --- |
| **Usecase Name** | **Xoá giao dịch** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn xoá một giao dịch đã được tạo trước đó trong ứng dụng quản lý tài chính. |
| **Priority** | **High** |
| **Actor** | User |
| **Trigger** |  |
| **Pre-condition(s)** | * Người dùng đã đăng nhập vào ứng dụng quản lý tài chính. * Người dùng đã truy cập vào màn hình xem danh sách giao dịch và chọn một giao dịch cần xoá. |
| **Post-condition(s)** | Hệ thống xoá thành công giao dịch khỏi danh sách. |
| **Basic Flow** | 1. Người dùng truy cập vào màn hình xem danh sách giao dịch và chọn một giao dịch cần xoá. 2. Hệ thống hiển thị các thông tin chi tiết của giao dịch cho người dùng. 3. Người dùng xác nhận muốn xoá giao dịch. 4. Hệ thống yêu cầu xác nhận từ người dùng với một thông báo xác nhận xoá giao dịch. 5. Người dùng xác nhận xoá giao dịch. 6. Hệ thống xoá giao dịch khỏi danh sách và hiển thị thông báo thành công cho người dùng. |
| **Alternative Flow** | 5a. Nếu người dùng không xác nhận xoá giao dịch trong bước 5, hệ thống sẽ không xoá giao dịch và quay lại màn hình xem danh sách giao dịch. |
| **Exception Flow** |  |
| **Business Rules** | * Hệ thống phải xoá giao dịch khỏi danh sách một cách chính xác và an toàn. * Hệ thống có thể cung cấp khả năng khôi phục giao dịch trong một khoảng thời gian nhất định sau khi bị xoá. |
| **Non-Functional Requirement** |  |

**Bảng 1.3.6.3**: Use case **Xoá giao dịch**

#### **Use case Xem danh sách giao dịch và lọc theo các tiêu chí**

| **Usecase Id** | UC014 |
| --- | --- |
| **Usecase Name** | **Xoá giao dịch** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn xem danh sách các giao dịch trong ứng dụng quản lý tài chính và lọc giao dịch theo các tiêu chí như loại (thu nhập, chi tiêu), ngày, danh mục (category) và nhiều tiêu chí khác. |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** | Người dùng truy cập vào màn hình danh sách giao dịch và lựa chọn các tiêu chí lọc. |
| **Pre-condition(s)** | * Người dùng đã đăng nhập vào ứng dụng quản lý tài chính. * Người dùng có ít nhất một giao dịch đã được tạo trong hệ thống.. |
| **Post-condition(s)** | Hệ thống hiển thị danh sách giao dịch phù hợp với các tiêu chí lọc đã được chọn. |
| **Basic Flow** | 1. Người dùng truy cập vào màn hình danh sách giao dịch. 2. Hệ thống hiển thị danh sách tất cả các giao dịch có trong hệ thống. 3. Người dùng lựa chọn các tiêu chí lọc để hiển thị chỉ các giao dịch phù hợp. 4. Hệ thống áp dụng các tiêu chí lọc và hiển thị lại danh sách giao dịch theo các tiêu chí đã chọn. 5. Người dùng có thể tiếp tục thay đổi tiêu chí lọc hoặc xem danh sách giao dịch đã lọc |
| **Alternative Flow** | Nếu không có giao dịch nào phù hợp với các tiêu chí lọc, hệ thống sẽ hiển thị một thông báo rỗng hoặc không có giao dịch phù hợp. |
| **Exception Flow** |  |
| **Business Rules** | * Hệ thống phải hiển thị danh sách giao dịch phù hợp với các tiêu chí lọc đã chọn và đảm bảo tính chính xác của dữ liệu. * Hệ thống có thể cung cấp khả năng sắp xếp danh sách giao dịch theo các tiêu chí khác như thời gian, số tiền, v.v. |
| **Non-Functional Requirement** |  |

#### **Use case Xuất báo cáo Excel các giao dịch**

| **Usecase Id** | UC015 |
| --- | --- |
| **Usecase Name** | **Xoá giao dịch** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn xuất danh sách các giao dịch từ ứng dụng quản lý tài chính ra định dạng báo cáo Excel. |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** | Người dùng truy cập vào màn hình danh sách giao dịch và chọn tùy chọn xuất báo cáo Excel. |
| **Pre-condition(s)** | * Người dùng đã đăng nhập vào ứng dụng quản lý tài chính. * Người dùng có ít nhất một giao dịch đã được tạo trong hệ thống.. |
| **Post-condition(s)** | Hệ thống xuất báo cáo các giao dịch dưới định dạng Excel và cung cấp tệp tin để người dùng tải xuống. |
| **Basic Flow** | 1. Người dùng truy cập vào màn hình danh sách giao dịch. 2. Hệ thống hiển thị danh sách tất cả các giao dịch có trong hệ thống. 3. Người dùng chọn tùy chọn xuất báo cáo Excel. 4. Hệ thống tạo báo cáo Excel từ danh sách giao dịch và lưu tệp tin. 5. Hệ thống cung cấp tệp tin Excel để người dùng tải xuống. |
| **Alternative Flow** | Nếu không có giao dịch nào để xuất báo cáo, hệ thống sẽ hiển thị một thông báo rỗng hoặc không có giao dịch phù hợp để xuất báo cáo. |
| **Exception Flow** |  |
| **Business Rules** | * Báo cáo Excel phải chứa các thông tin cần thiết về các giao dịch như ngày, loại, số tiền, danh mục, v.v. * Báo cáo Excel phải có định dạng dễ đọc và dễ sử dụng cho người dùng. |
| **Non-Functional Requirement** |  |

### **Use case Quản lý các chi tiêu thông thường**

#### **Use case Thêm khoản chi tiêu thông thường**

| **Usecase Id** | UC016 |
| --- | --- |
| **Usecase Name** | **Thêm khoản chi tiêu thông thường** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn thêm một khoản chi tiêu thông thường vào ứng dụng quản lý tài chính với các thông tin như tên khoản chi tiêu, lời nhắc, giao dịch tự động, thời gian bắt đầu và kết thúc, nguồn tiền, số tiền, mô tả và loại chi tiêu. |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** | Người dùng truy cập vào màn hình quản lý khoản chi tiêu thông thường và chọn tùy chọn thêm khoản chi tiêu. |
| **Pre-condition(s)** | * Người dùng đã đăng nhập vào ứng dụng quản lý tài chính. * Người dùng đã truy cập vào màn hình quản lý khoản chi tiêu thông thường. |
| **Post-condition(s)** | Hệ thống thêm thành công khoản chi tiêu mới vào danh sách. |
| **Basic Flow** | 1. Người dùng chọn tùy chọn thêm khoản chi tiêu. 2. Hệ thống hiển thị màn hình thêm khoản chi tiêu với các trường thông tin cần điền. 3. Người dùng điền các thông tin như tên khoản chi tiêu, lời nhắc, giao dịch tự động, thời gian bắt đầu và kết thúc, nguồn tiền, số tiền, comment và category. 4. Người dùng xác nhận để thêm khoản chi tiêu. 5. Hệ thống kiểm tra và lưu khoản chi tiêu mới vào danh sách. |
| **Alternative Flow** | Nếu người dùng không điền đầy đủ các trường thông tin bắt buộc, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu người dùng điền đầy đủ thông tin trước khi thêm khoản chi tiêu. |
| **Exception Flow** |  |
| **Business Rules** | * Hệ thống phải kiểm tra tính hợp lệ của các thông tin cần điền và cung cấp thông báo lỗi nếu có. * Hệ thống phải đảm bảo tính chính xác của dữ liệu khi thêm khoản chi tiêu mới. * Hệ thống có thể cung cấp gợi ý hoặc danh sách các giá trị cho các trường thông tin để giúp người dùng điền dễ dàng hơn. |
| **Non-Functional Requirement** |  |

#### **Use case Sửa khoản chi tiêu thông thường**

| **Usecase Id** | UC017 |
| --- | --- |
| **Usecase Name** | **Sửa khoản chi tiêu thông thường** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn sửa thông tin của một khoản chi tiêu thông thường đã có trong ứng dụng quản lý tài chính, bao gồm tên khoản chi tiêu, lời nhắc, giao dịch tự động, thời gian bắt đầu và kết thúc, nguồn tiền, số tiền, mô tả và loại chi tiêu |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** | Người dùng truy cập vào màn hình quản lý khoản chi tiêu thông thường và chọn tùy chọn sửa khoản chi tiêu. |
| **Pre-condition(s)** | * Người dùng đã đăng nhập vào ứng dụng quản lý tài chính. * Người dùng đã truy cập vào màn hình quản lý khoản chi tiêu thông thường. |
| **Post-condition(s)** | Hệ thống cập nhật thành công thông tin của khoản chi tiêu. |
| **Basic Flow** | 1. Người dùng chọn khoản chi tiêu cần sửa từ danh sách. 2. Hệ thống hiển thị màn hình sửa khoản chi tiêu với các trường thông tin đã được điền sẵn. 3. Người dùng sửa các thông tin cần thiết như tên khoản chi tiêu, lời nhắc, giao dịch tự động, thời gian bắt đầu và kết thúc, nguồn tiền, số tiền, comment và category. 4. Người dùng xác nhận để lưu các thay đổi. 5. Hệ thống kiểm tra và cập nhật thông tin của khoản chi tiêu. |
| **Alternative Flow** | Nếu người dùng không điền đầy đủ các trường thông tin bắt buộc, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu người dùng điền đầy đủ thông tin trước khi lưu thay đổi. |
| **Exception Flow** |  |
| **Business Rules** | * Hệ thống phải kiểm tra tính hợp lệ của các thông tin cần điền và cung cấp thông báo lỗi nếu có. * Hệ thống phải đảm bảo tính chính xác của dữ liệu khi cập nhập khoản chi tiêu. * Hệ thống có thể cung cấp gợi ý hoặc danh sách các giá trị cho các trường thông tin để giúp người dùng điền dễ dàng hơn. |
| **Non-Functional Requirement** |  |

#### **Use case Xoá khoản chi tiêu thông thường**

| **Usecase Id** | UC018 |
| --- | --- |
| **Usecase Name** | **Xoá khoản chi tiêu thông thường** |
| **Description** | Use case này mô tả tình huống khi người dùng muốn xoá một khoản chi tiêu thông thường đã có trong ứng dụng quản lý tài chính. |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** | Người dùng truy cập vào màn hình quản lý khoản chi tiêu thông thường và chọn tùy chọn xoá khoản chi tiêu. |
| **Pre-condition(s)** | * Người dùng đã đăng nhập vào ứng dụng quản lý tài chính. * Người dùng đã truy cập vào màn hình quản lý khoản chi tiêu thông thường. |
| **Post-condition(s)** | Hệ thống xoá thành công thông tin của khoản chi tiêu. |
| **Basic Flow** | 1. Người dùng chọn khoản chi tiêu cần xoá từ danh sách. 2. Hệ thống hiển thị xác nhận xoá khoản chi tiêu và yêu cầu người dùng xác nhận. 3. Người dùng xác nhận để tiếp tục xoá khoản chi tiêu. 4. Hệ thống xoá khoản chi tiêu khỏi danh sách. |
| **Alternative Flow** | Nếu người dùng không điền đầy đủ các trường thông tin bắt buộc, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu người dùng điền đầy đủ thông tin trước khi lưu thay đổi. |
| **Exception Flow** |  |
| **Business Rules** | * Hệ thống phải yêu cầu người dùng xác nhận trước khi xoá khoản chi tiêu để tránh xoá nhầm. * Hệ thống phải xác định và xoá chính xác khoản chi tiêu được chọn. * Hệ thống có thể cung cấp thông báo thành công sau khi xoá thành công khoản chi tiêu. |
| **Non-Functional Requirement** |  |

### **Use case Xem biểu đồ giao dịch cá nhân**

| **Usecase Id** | UC019 |
| --- | --- |
| **Usecase Name** | **Xem biểu đồ giao dịch cá nhân** |
| **Description** | Người dùng có thể xem biểu đồ hoặc biểu đồ thống kê về các giao dịch cá nhân của mình. |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** | Người dùng chọn chức năng "Xem biểu đồ giao dịch". |
| **Pre-condition(s)** | Người dùng đã đăng nhập vào tài khoản cá nhân |
| **Post-condition(s)** | Biểu đồ giao dịch cá nhân được hiển thị trên giao diện người dùng. |
| **Basic Flow** | 1. Người dùng mở ứng dụng quản lý chi tiêu. 2. Người dùng đăng nhập vào tài khoản cá nhân của mình. 3. Người dùng chọn chức năng "Xem biểu đồ giao dịch". 4. Ứng dụng hiển thị biểu đồ giao dịch cá nhân dựa trên dữ liệu đã được nhập hoặc đồng bộ từ nguồn khác. |
| **Alternative Flow** | Không |
| **Exception Flow** | Nếu không có dữ liệu giao dịch, ứng dụng hiển thị thông báo không có dữ liệu để hiển thị. |
| **Business Rules** | Không |
| **Non-Functional Requirement** | Hiệu suất ứng dụng cần đảm bảo hiển thị biểu đồ giao dịch một cách nhanh chóng và mượt mà. |

### 

### **Use case Bảng phân tích và báo cáo**

| **Usecase Id** | UC020 |
| --- | --- |
| **Usecase Name** | **Bảng phân tích và báo cáo** |
| **Description** | Ứng dụng cung cấp bảng phân tích và báo cáo dựa trên thông tin chi tiêu, thu nhập và các thông số khác của người dùng. |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** | Người dùng chọn chức năng "Bảng phân tích và báo cáo". |
| **Pre-condition(s)** | Người dùng đã đăng nhập vào tài khoản cá nhân |
| **Post-condition(s)** | Bảng phân tích và báo cáo được hiển thị trên giao diện người dùng. |
| **Basic Flow** | 1. Người dùng mở ứng dụng quản lý chi tiêu. 2. Người dùng đăng nhập vào tài khoản cá nhân của mình. 3. Người dùng chọn chức năng "Bảng phân tích và báo cáo". 4. Ứng dụng xử lý dữ liệu chi tiêu và thu nhập của người dùng để tạo bảng phân tích và báo cáo. 5. Ứng dụng hiển thị bảng phân tích và báo cáo với các thông số như tổng chi tiêu, tổng thu nhập, biểu đồ phân tích theo danh mục chi tiêu, v.v. |
| **Alternative Flow** | Không |
| **Exception Flow** | Nếu không có dữ liệu giao dịch, ứng dụng hiển thị thông báo không có dữ liệu để hiển thị. |
| **Business Rules** | Không |
| **Non-Functional Requirement** | Hiệu suất ứng dụng cần đảm bảo việc xử lý và hiển thị bảng phân tích và báo cáo nhanh chóng, và giao diện người dùng phải đơn giản và dễ sử dụng. |

### 

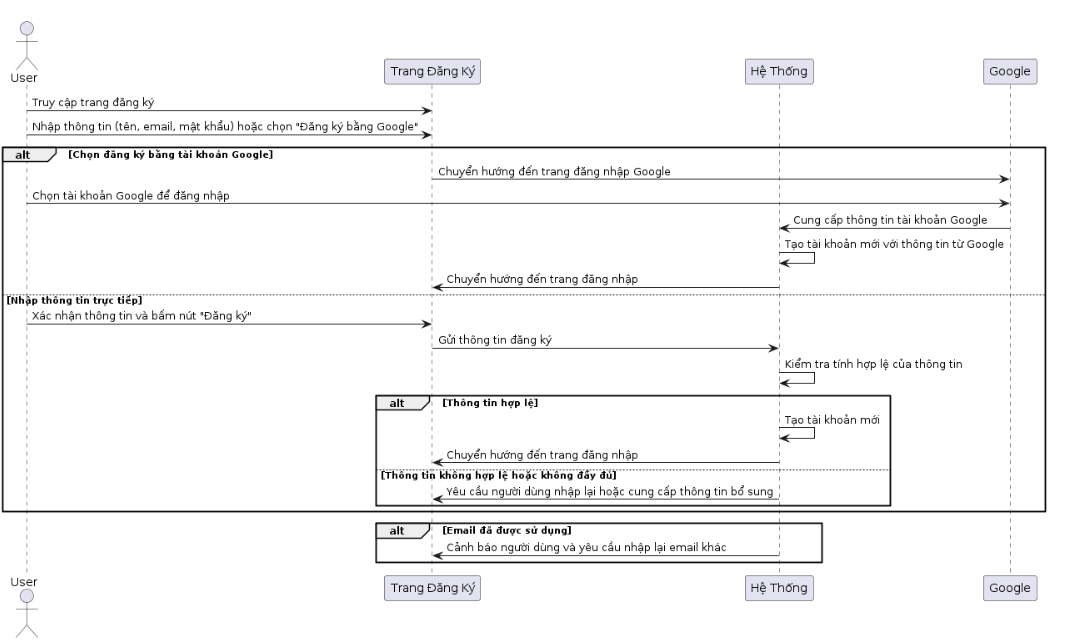
### **Use case Hỗ trợ người dùng**

| **Usecase Id** | UC021 |
| --- | --- |
| **Usecase Name** | **Hỗ trợ người dùng** |
| **Description** | Cung cấp giao diện để người dùng có thể phản hồi các vấn đề cho nhà phát triển ứng dụng |
| **Priority** | **Medium** |
| **Actor** | User |
| **Trigger** | Người dùng chọn mục Hỗ trợ |
| **Pre-condition(s)** | Người dùng đã đăng nhập vào tài khoản cá nhân |
| **Post-condition(s)** | Vấn đề của người dùng được gửi đến nhà phát triển |
| **Basic Flow** | 1. Người dùng mở ứng dụng quản lý chi tiêu. 2. Người dùng đăng nhập vào tài khoản cá nhân của mình. 3. Người dùng chọn chức năng "Hỗ trợ". 4. Người dùng chọn tạo mới một báo cáo 5. Người dùng điền thông tin và ấn gửi 6. Ứng dụng gửi đánh giá của người dùng đi và thông báo gửi thành công |
| **Alternative Flow** | Không |
| **Exception Flow** | Thông tin bị bỏ trống, ứng dụng thông báo lỗi |
| **Business Rules** | Không |
| **Non-Functional Requirement** |  |

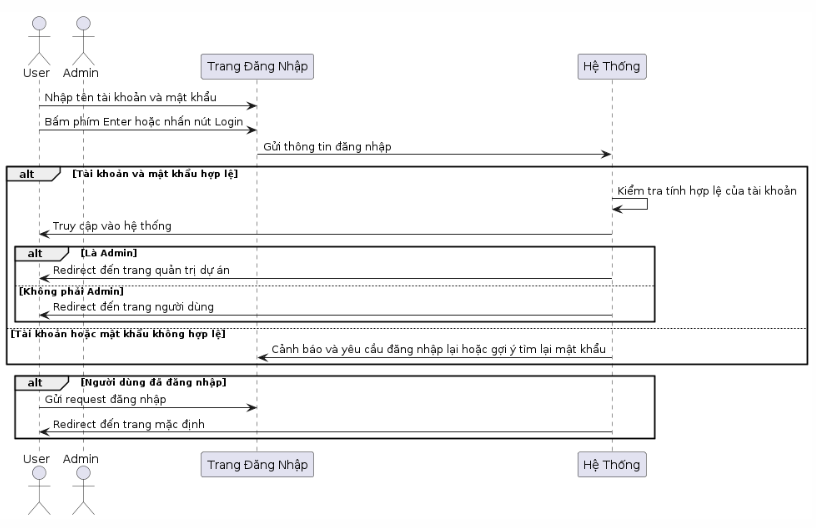
## 

## **Chi tiết lược đồ sequence**

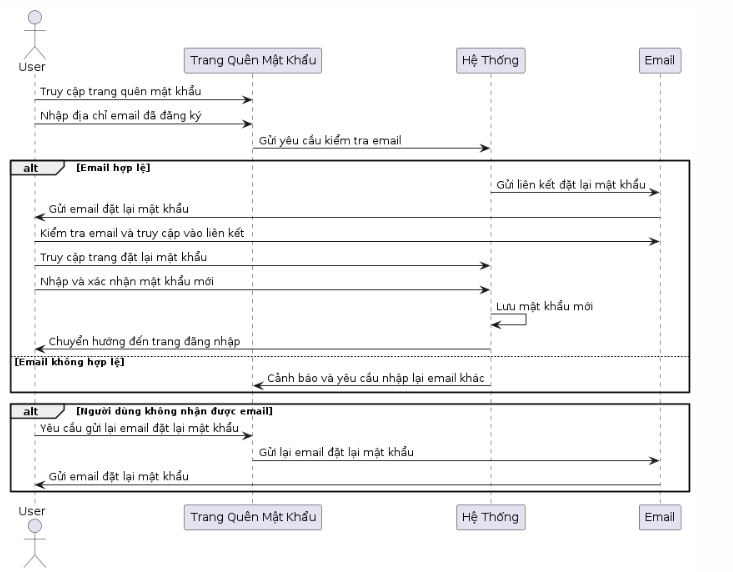
### **Chức năng Đăng ký**



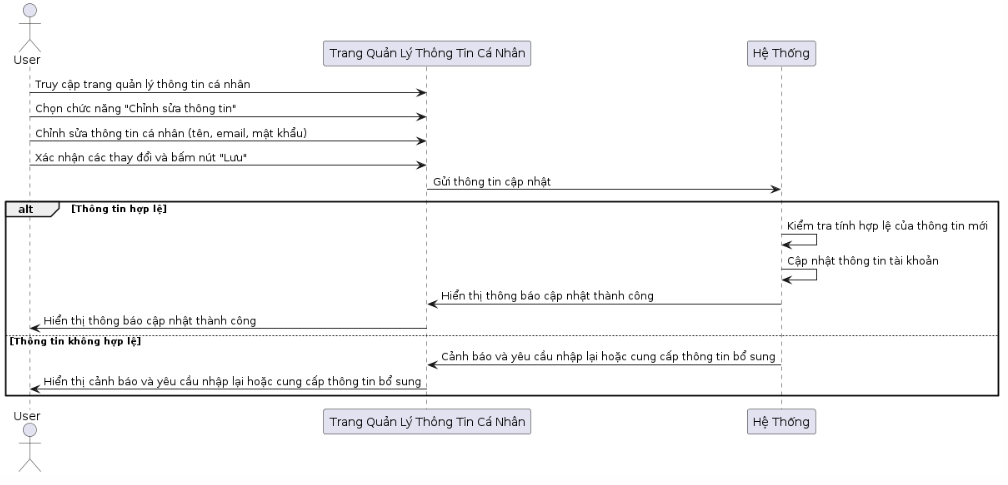
### **Chức năng Đăng nhập**



### **Chức năng Quên mật khẩu**

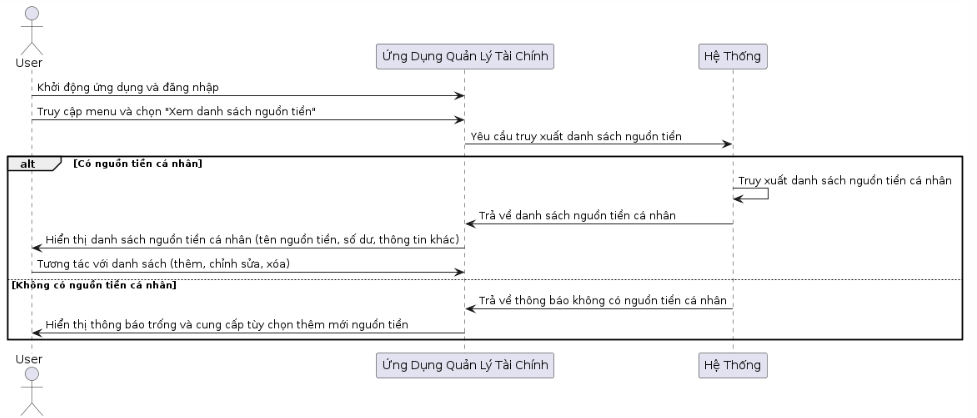


### **Chức năng Đổi thông tài khoản**

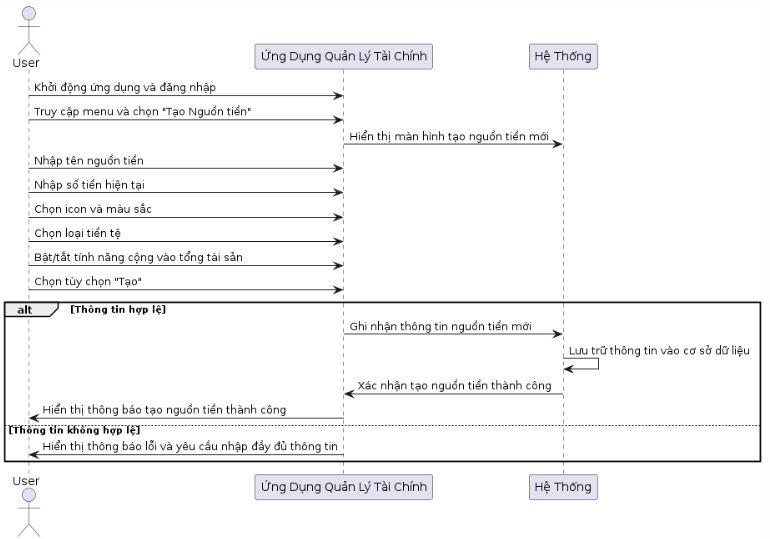


### **Chức năng nguồn tiền**

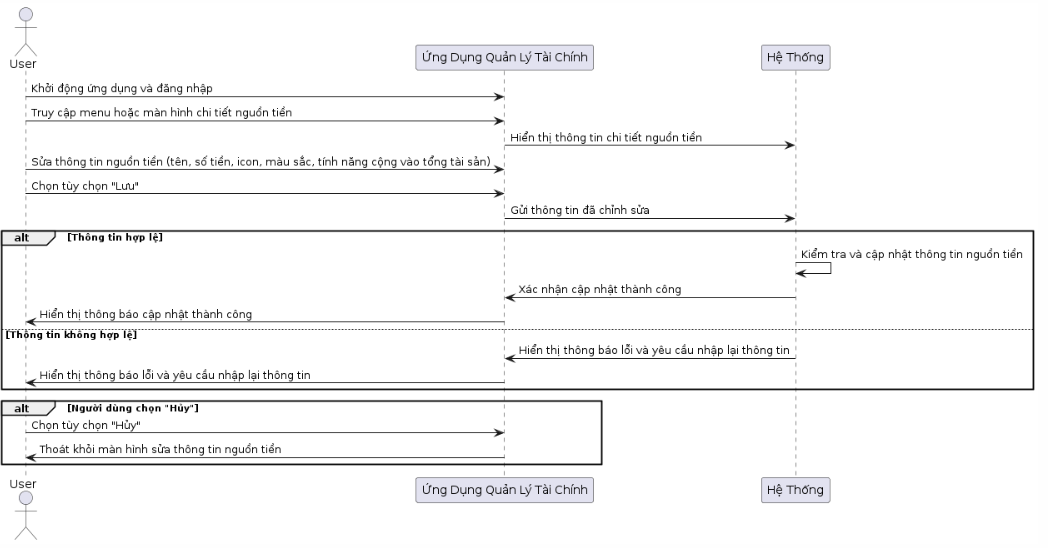
#### **Chức năng Xem danh sách nguồn tiền**

****

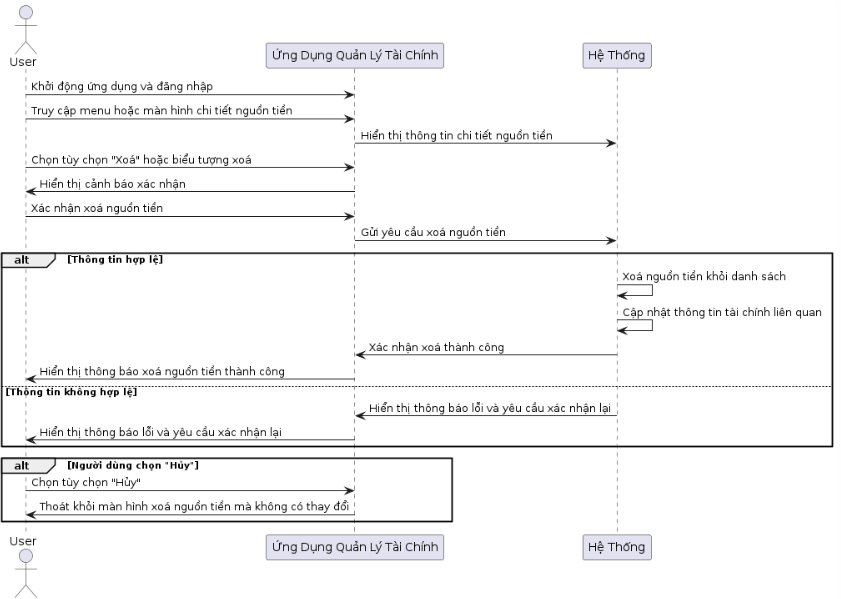
#### **Chức năng Tạo nguồn tiền**

****

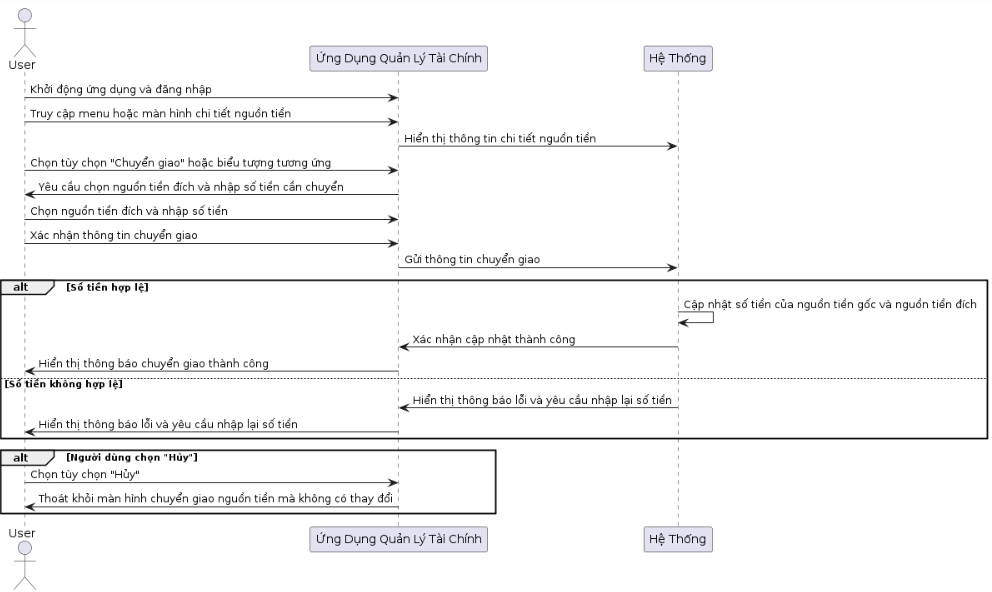
#### **Chức năng Sửa Nguồn tiền**

****

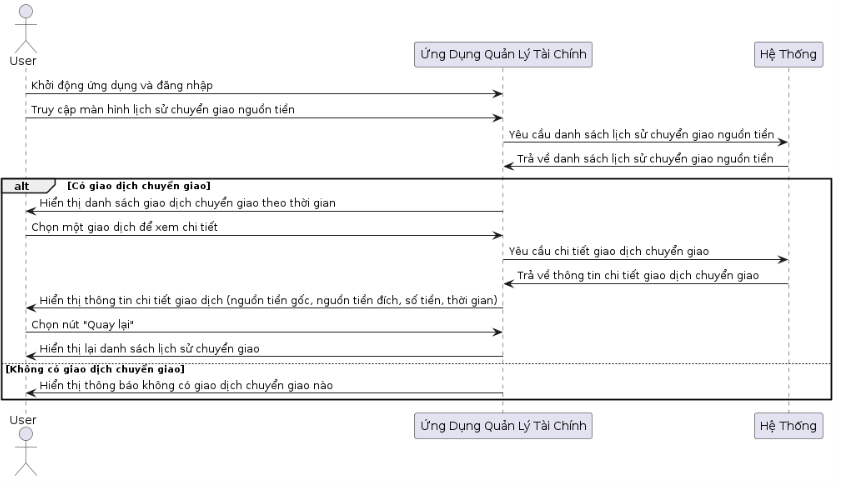
#### **Chức năng Xoá Nguồn tiền**

****

#### **Chức năng Chuyển giao Nguồn tiền cá nhân**

****

#### **Chức năng Xem lịch sử chuyển giao Nguồn tiền cá nhân**

****

### **Quản lý giao dịch cá nhân**

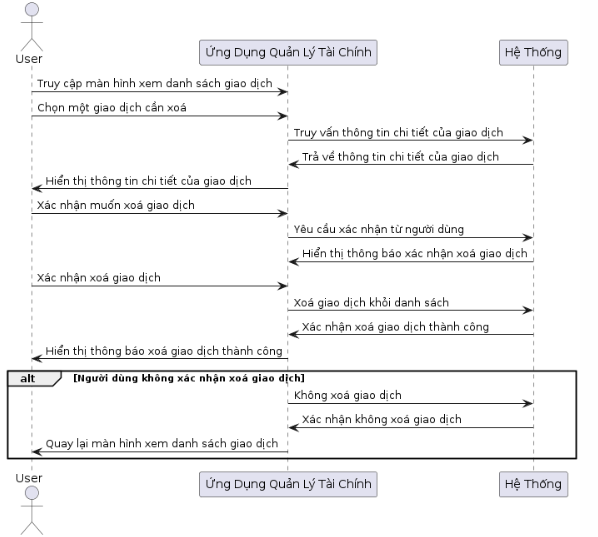
#### **Chức năng Tạo giao dịch mới**

****

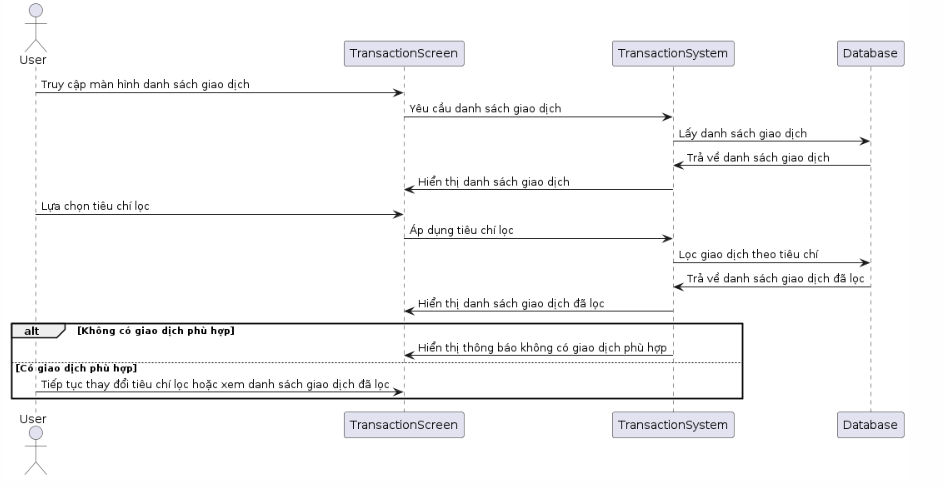
#### **Chức năng Sửa giao dịch**

****

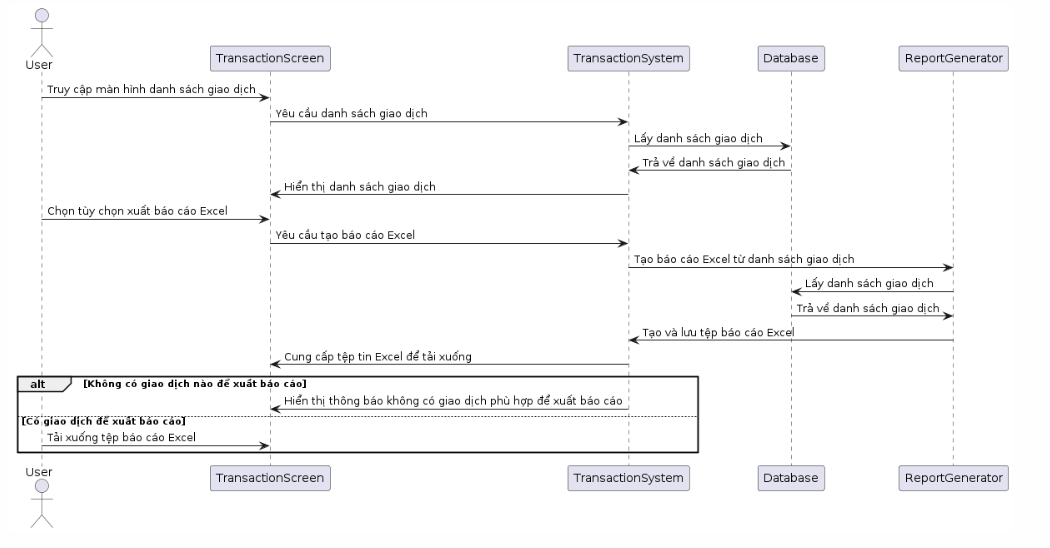
#### **Chức năng Xoá giao dịch**

****

#### **Chức năng Xem danh sách giao dịch và lọc theo các tiêu chí**

****

#### **Chức năng Xuất báo cáo Excel các giao dịch**

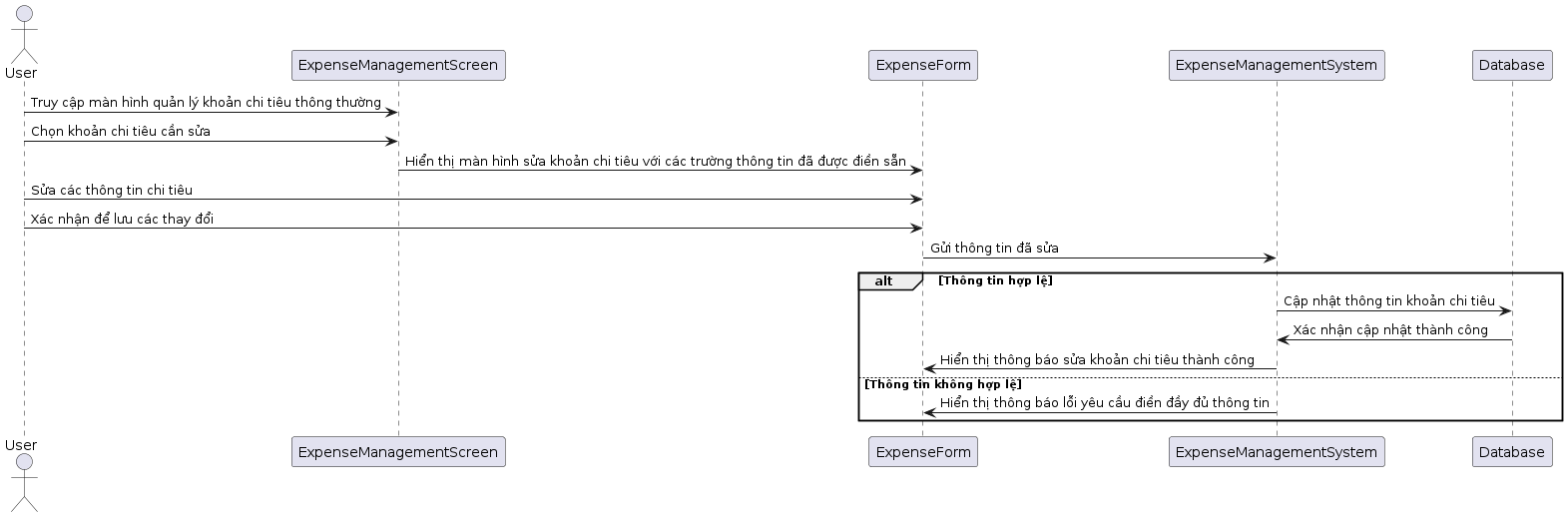
****

### **Quản lý các chi tiêu thông thường**

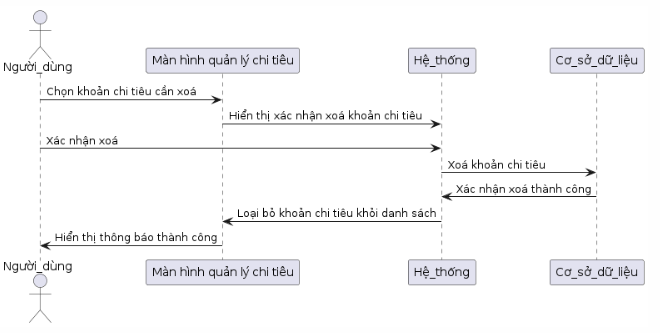
#### **Chức năng Thêm khoản chi tiêu thông thường**



#### **Chức năng Sửa khoản chi tiêu thông thường**



#### **Chức năng Xoá khoản chi tiêu thông thường**



### **Chức năng Xem biểu đồ giao dịch cá nhân**

### 

### **Chức năng Bảng phân tích và báo cáo**

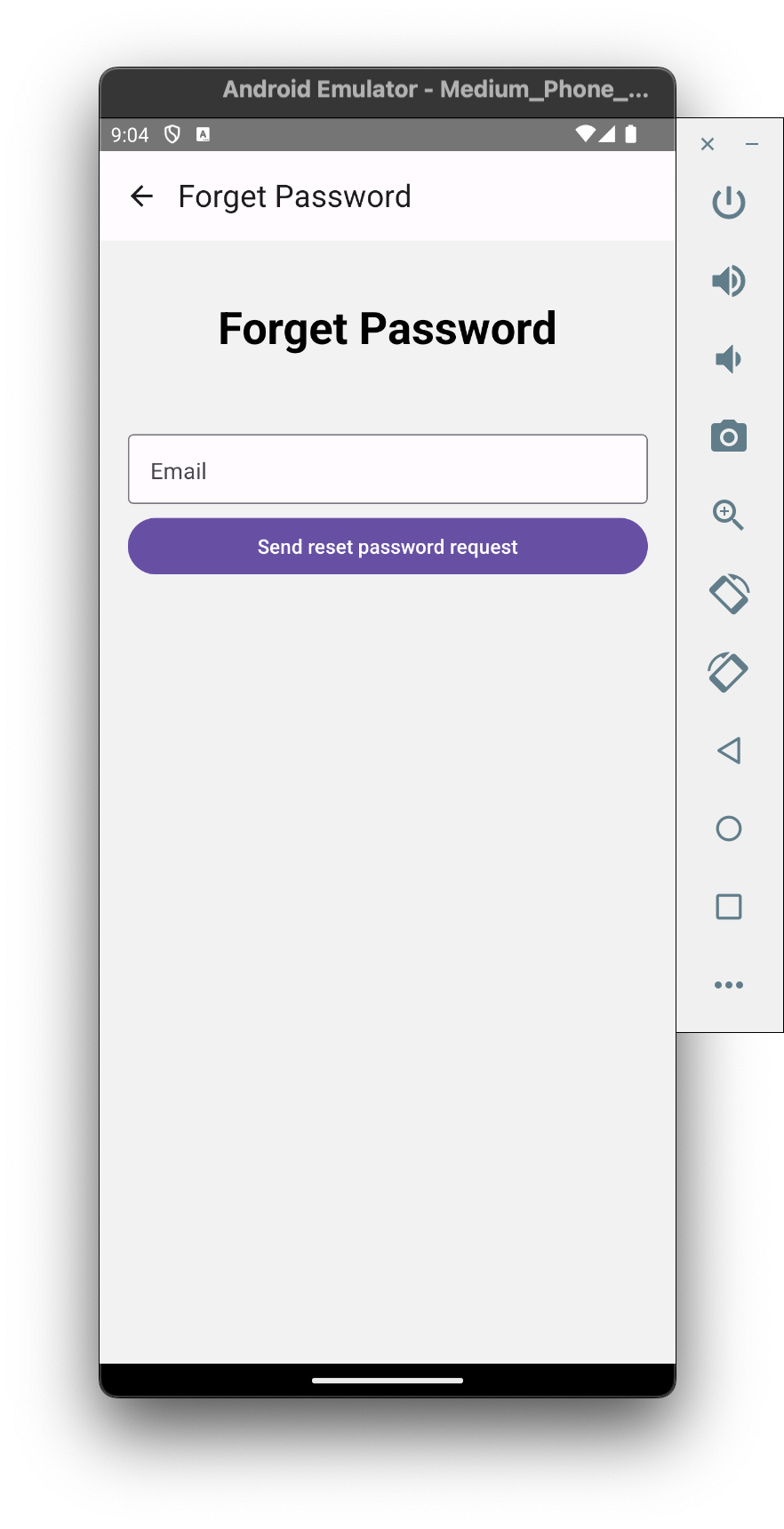
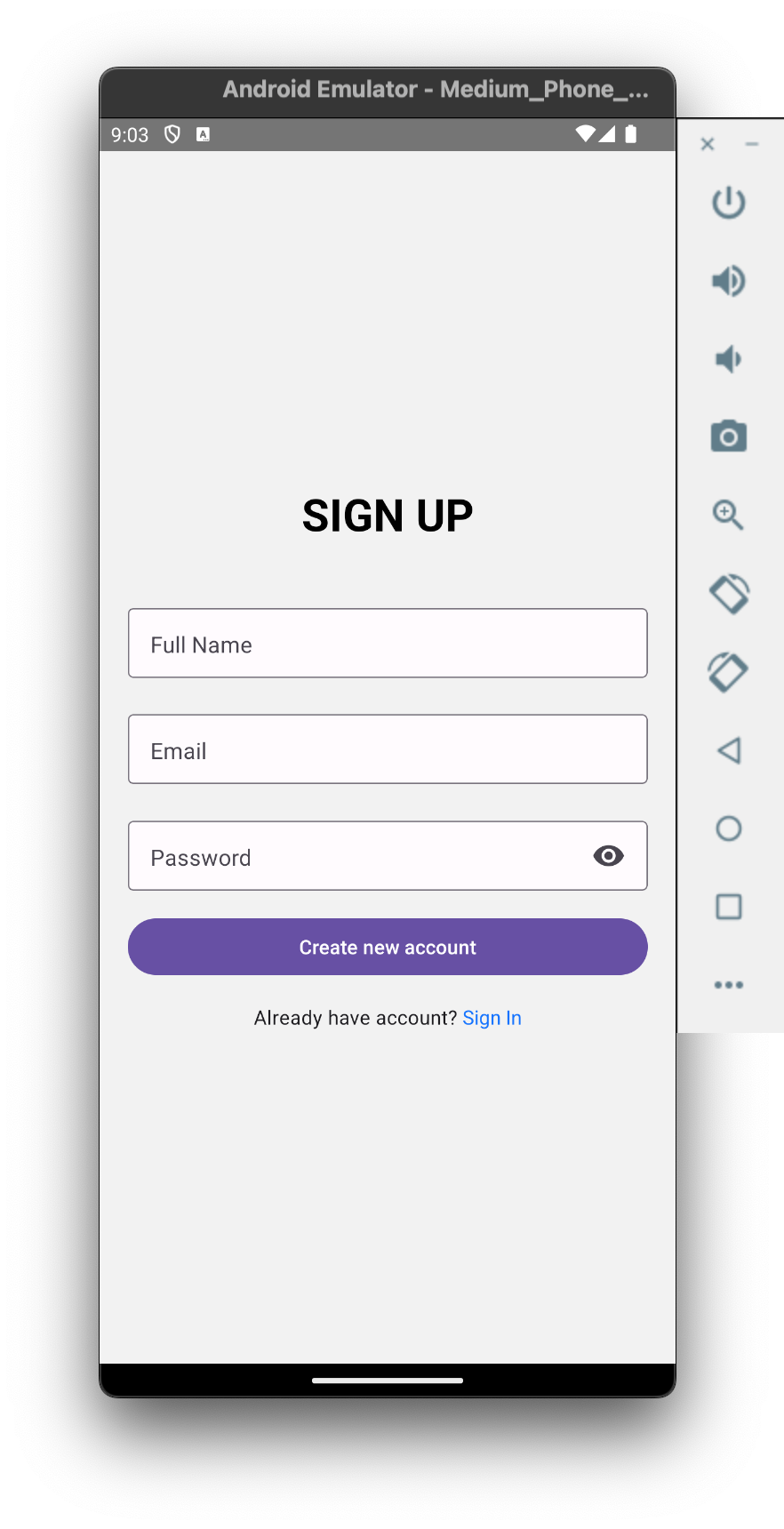
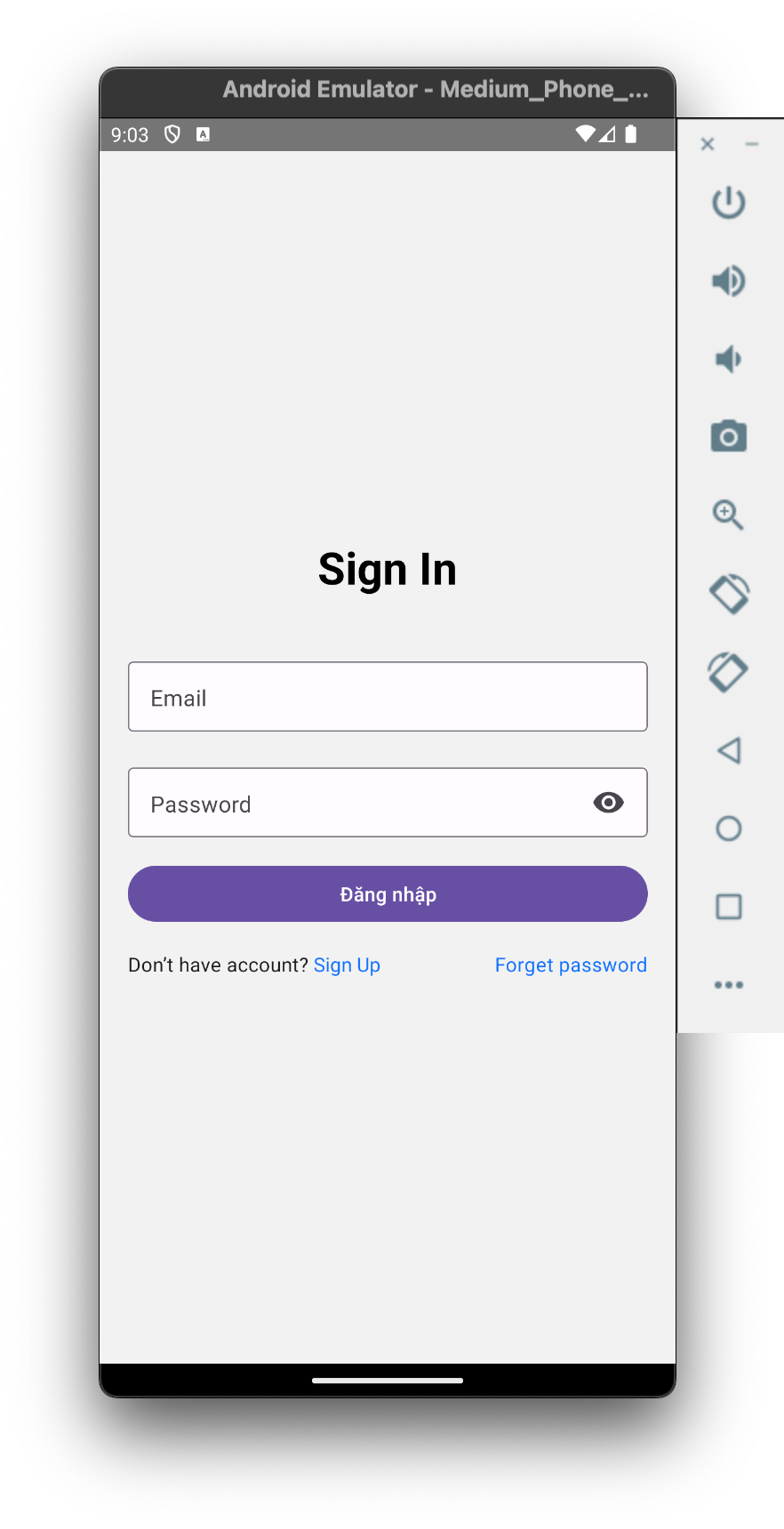
### 

### **Chức năng Hỗ trợ người dùng**

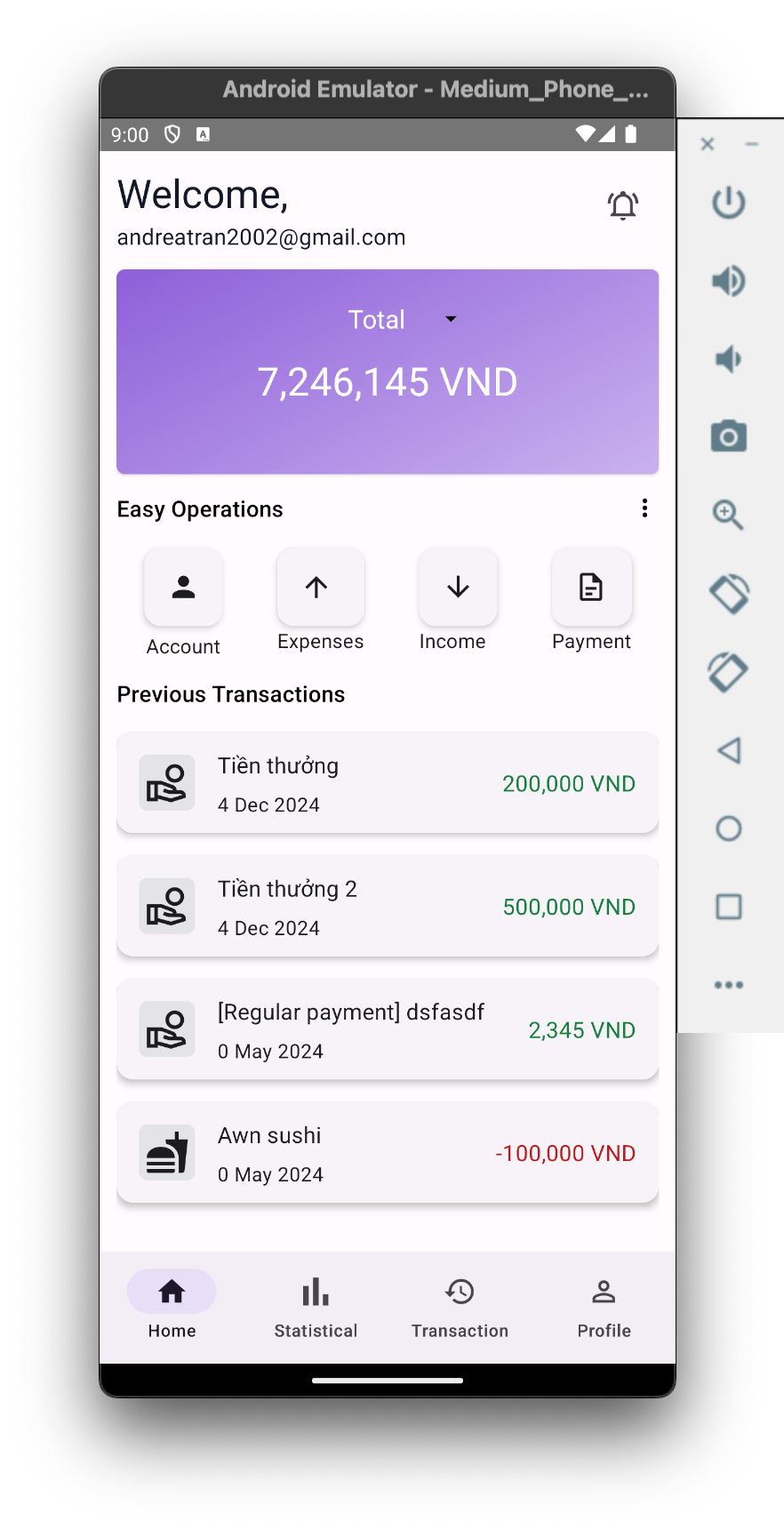
# 

# **CHƯƠNG 3: THIẾT KẾ GIAO DIỆN**

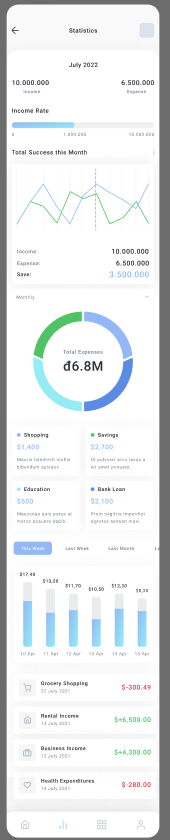
## **Xác thực**



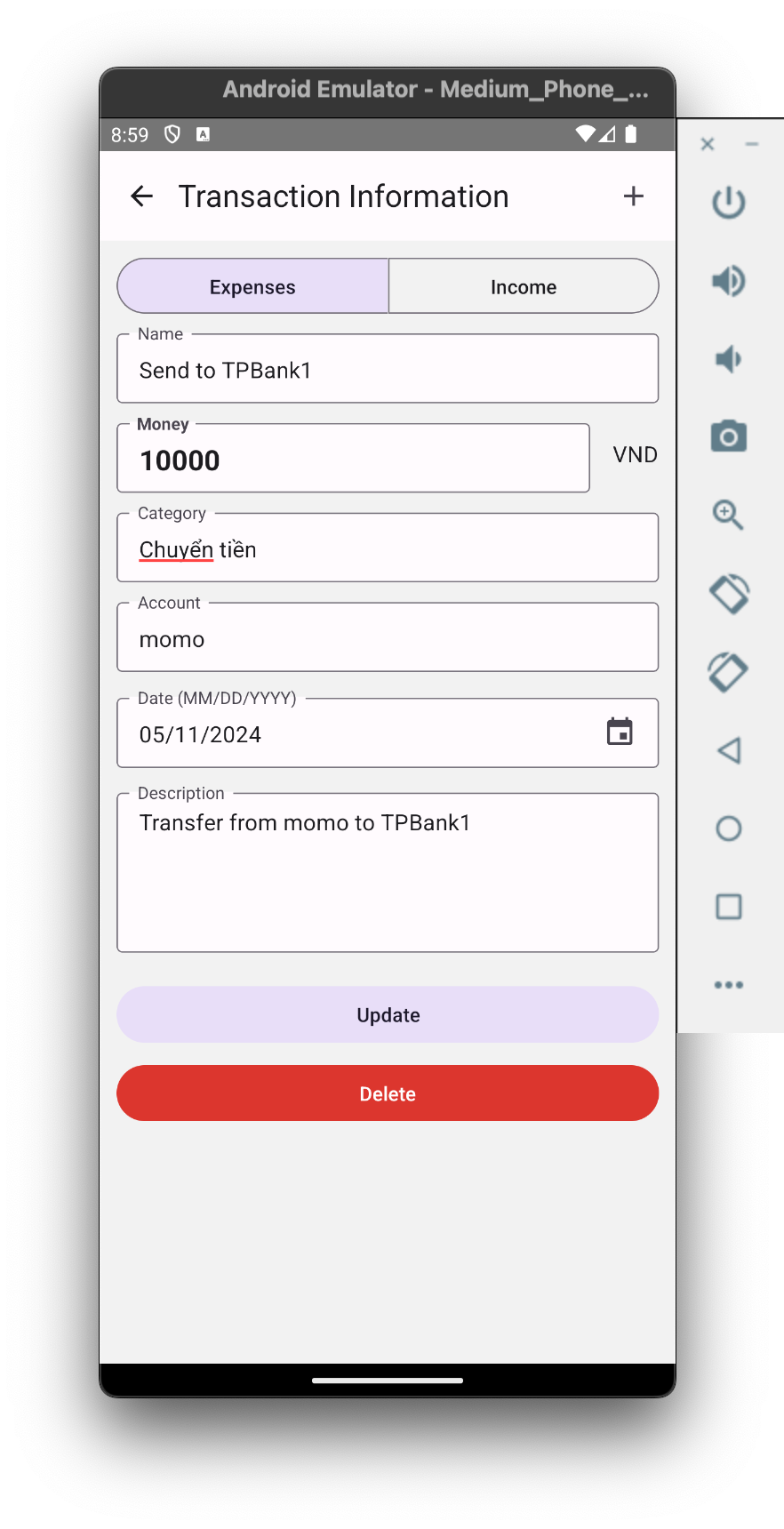
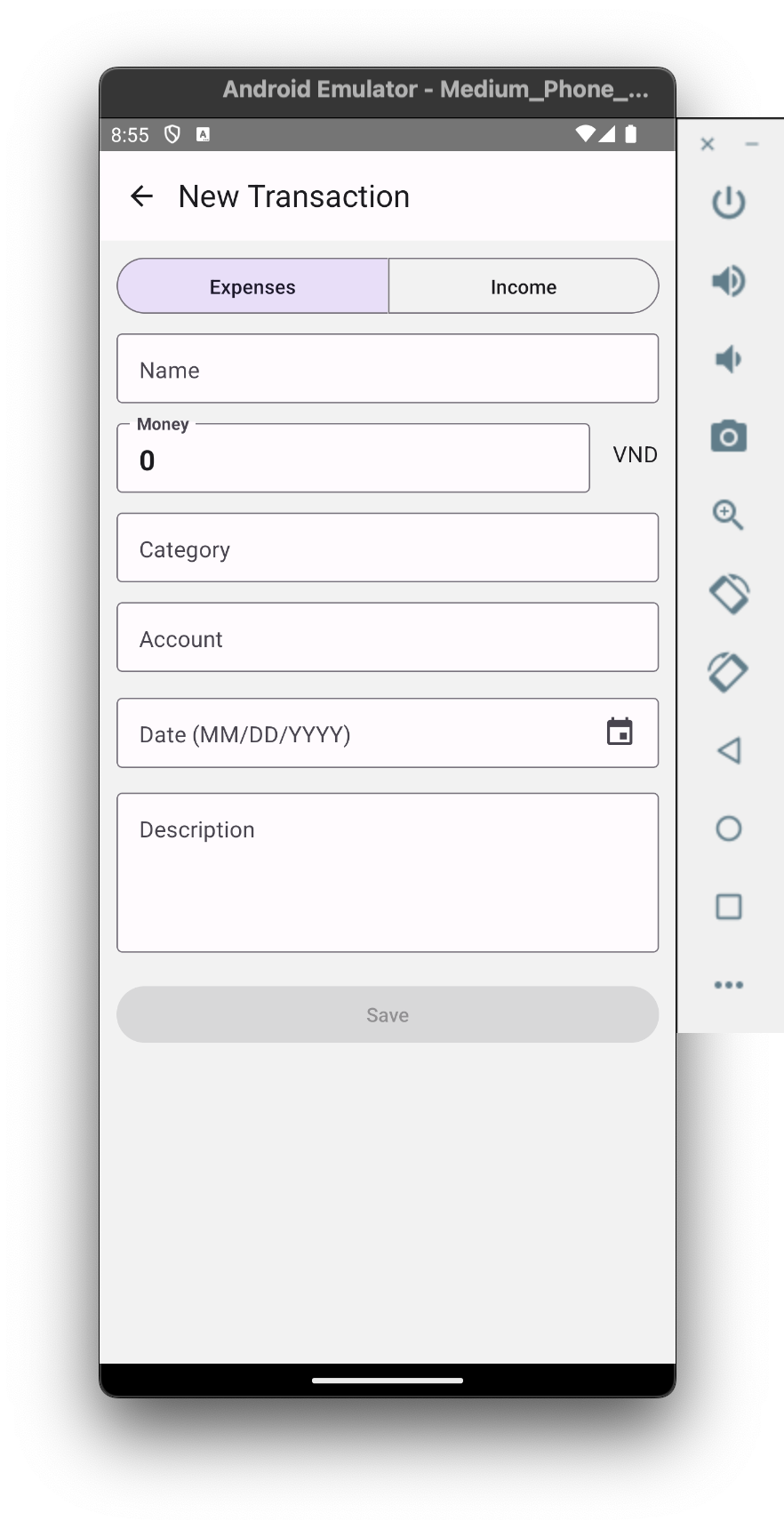
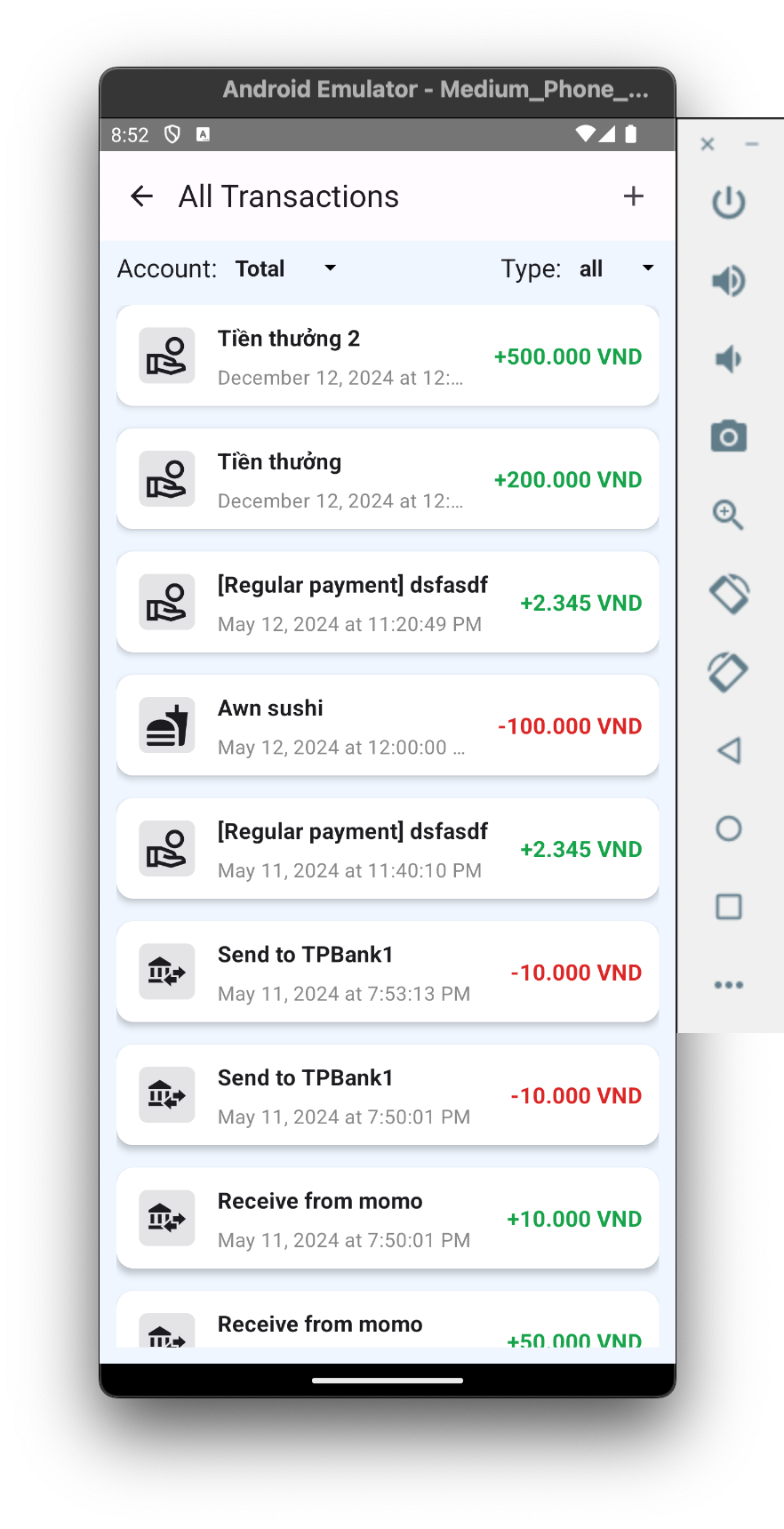
## **Trang chủ**



## **Trang Phân tích**



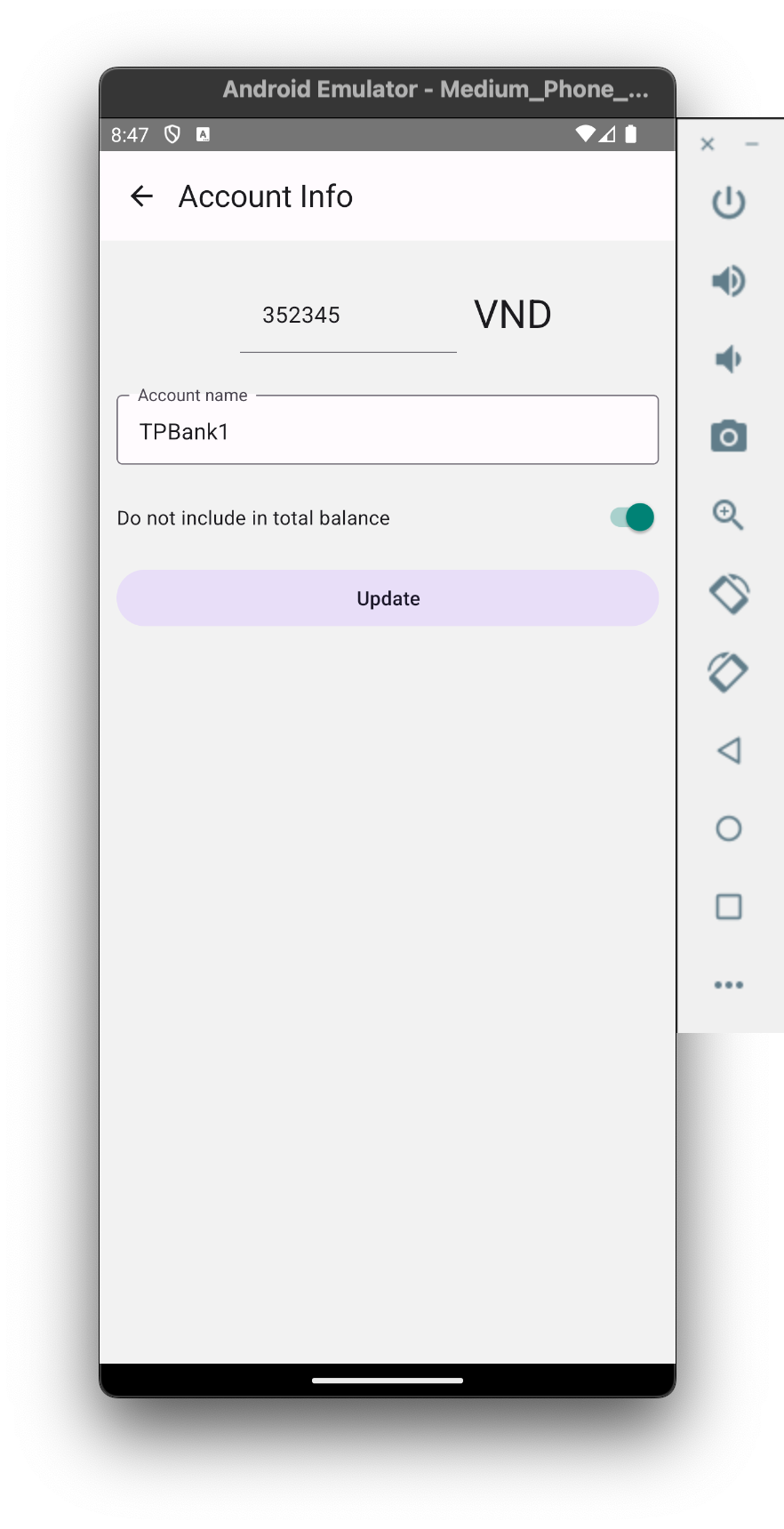
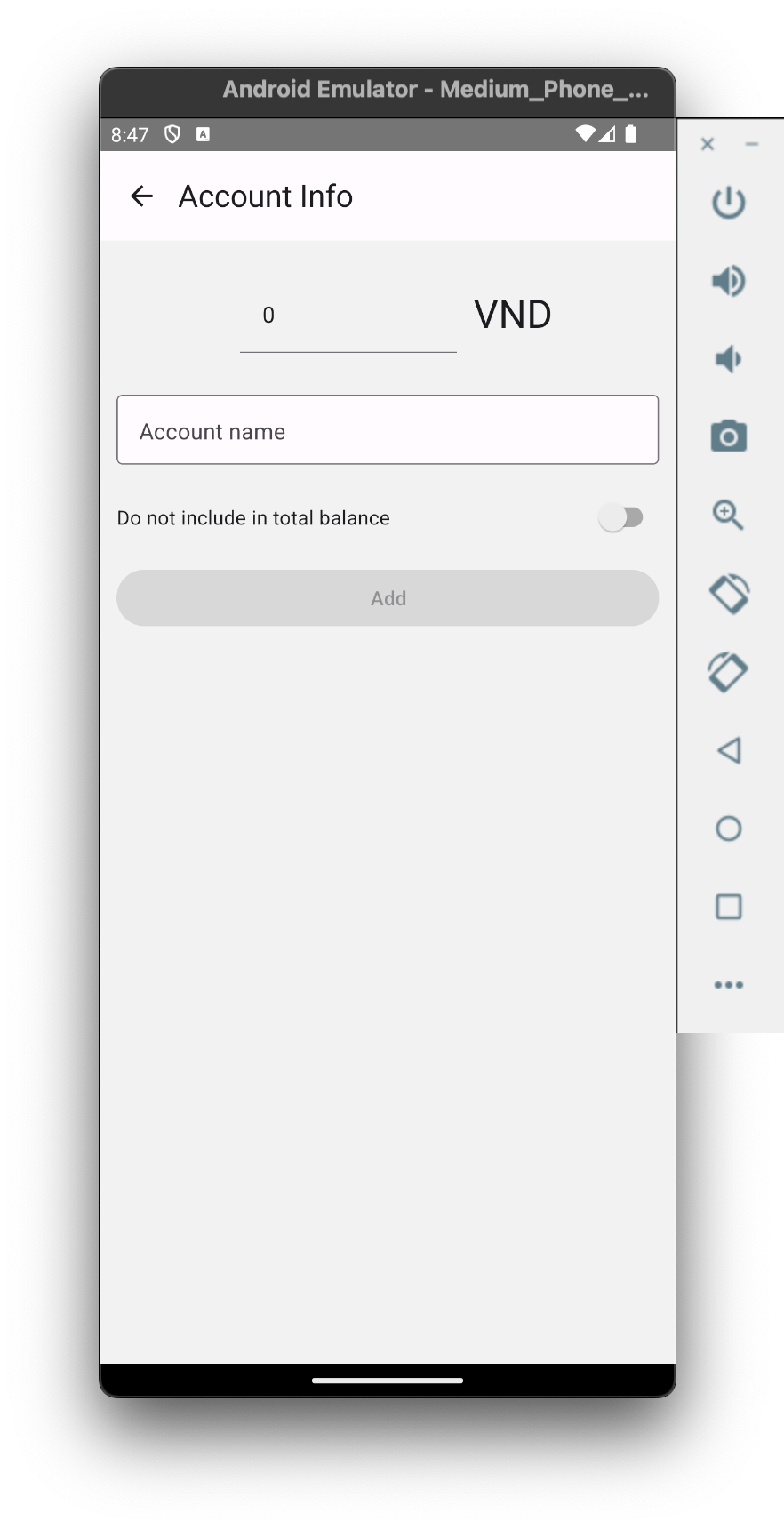
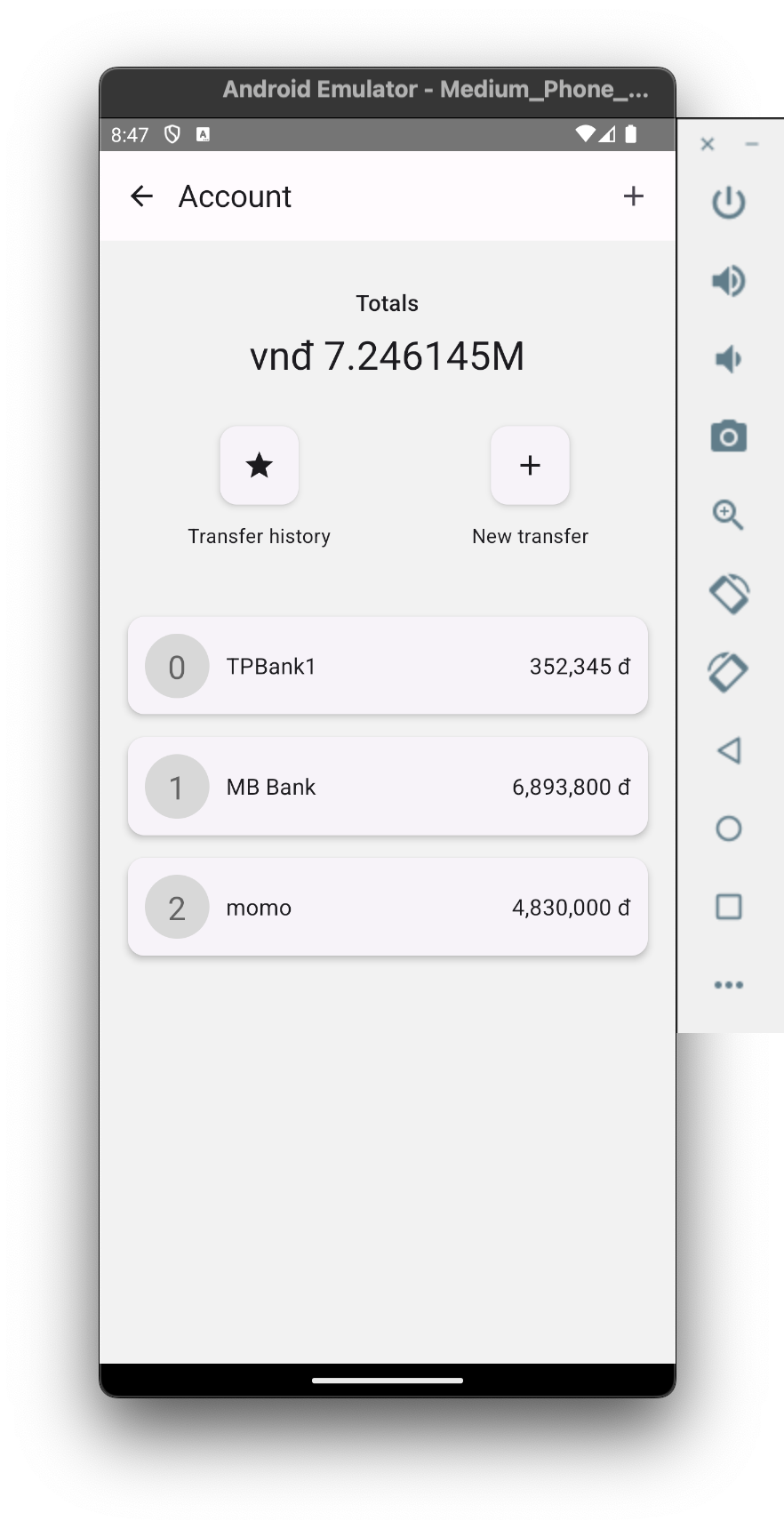
## **Trang Transaction**



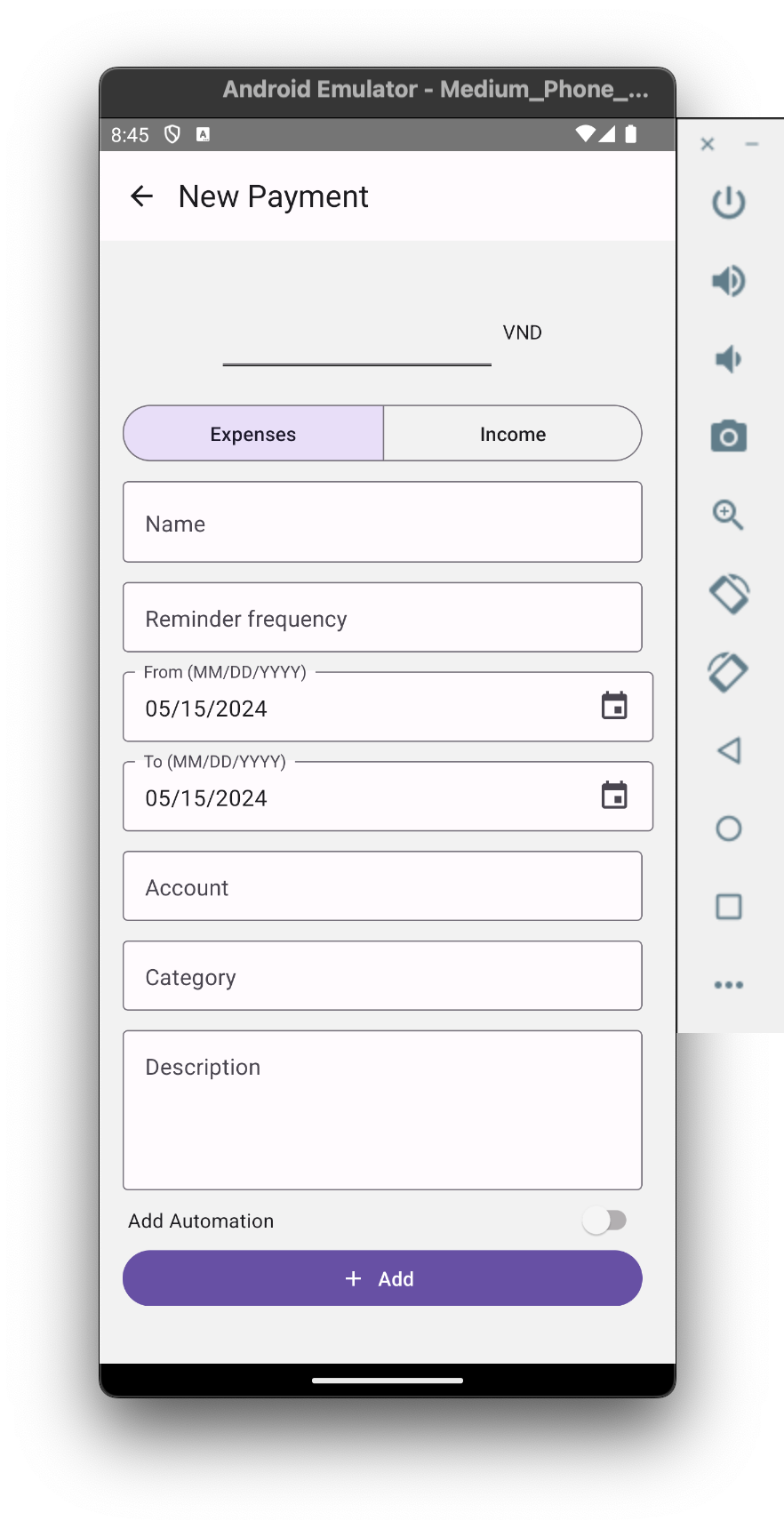
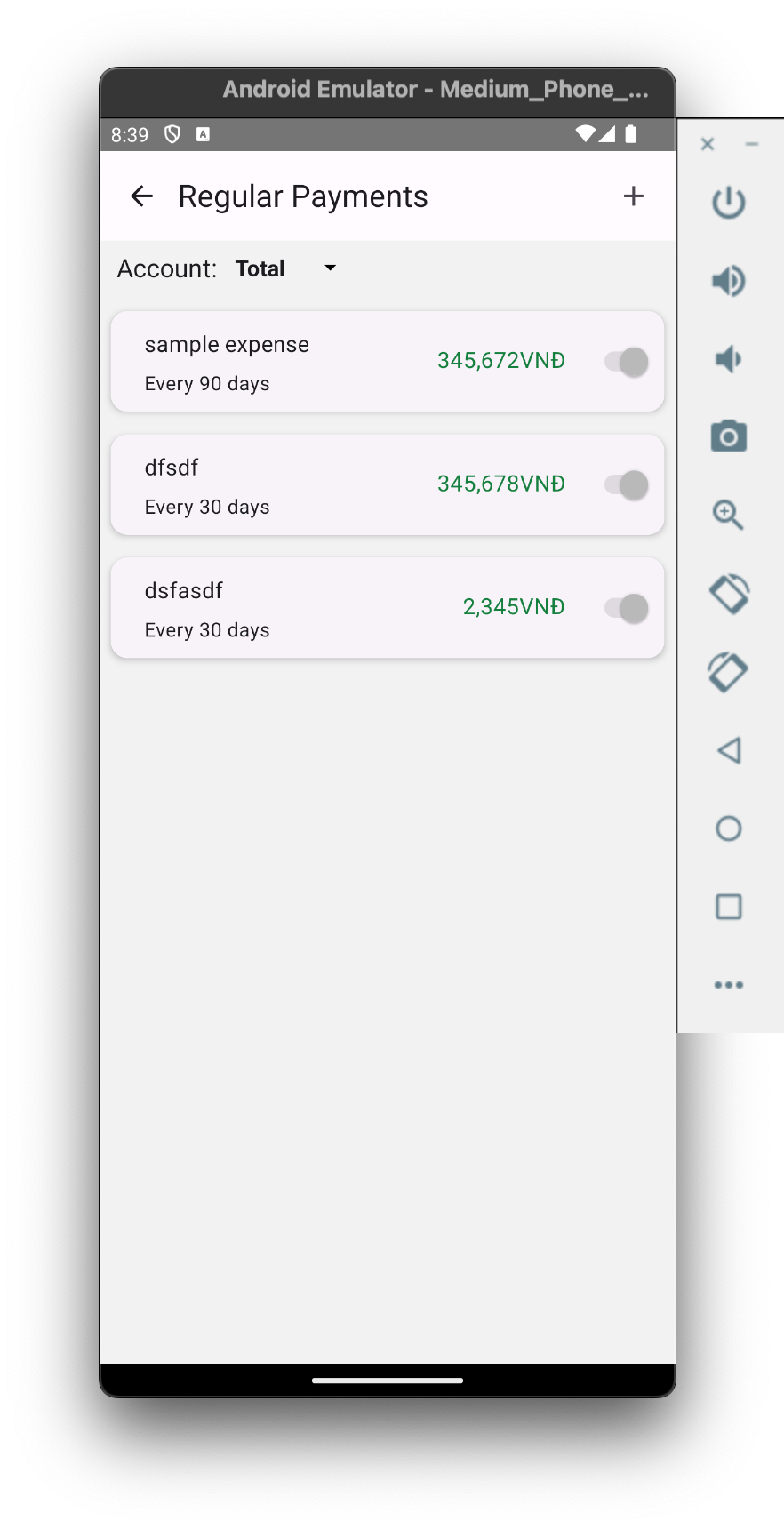
## **Trang Profile**

## 

## **Trang nguồn tiền**



## **Trang Regular Payment**



## **Notification**

## 

# **CHƯƠNG 4: API DOCUMENT**

## **Tổng quan về Directus**

### **Directus là gì**

Directus là một Open Data Platform được xây dựng để dân chủ hóa cơ sở dữ liệu. Nền tảng này cung cấp cho mọi thành viên trong nhóm của bạn, bất kể kỹ năng kỹ thuật, quyền truy cập bình đẳng vào quản lý dữ liệu và tài sản kỹ thuật số, cho bất kỳ mô hình dữ liệu hoặc dự án nào.

Trước tiên, liên kết Directus với cơ sở dữ liệu SQL và file storage adapter. Sau đó, Directus cho phép bạn thực hiện các thao tác CRUD, tạo người dùng, phân quyền với các quyền hoàn toàn có thể cấu hình, xây dựng các truy vấn phức tạp và chi tiết, cấu hình webhooks và tự động hóa tác vụ dựa trên sự kiện...

### **Cách thức hoạt động của Directus**

Directus được cài đặt như một lớp trên cơ sở dữ liệu SQL mới hoặc hiện có của chúng ta.

Ứng dụng và API sẽ "phản chiếu" động lược đồ và nội dung thực tế của ta trong thời gian thực. Điều này tương tự như cách các ứng dụng khách cơ sở dữ liệu kỹ thuật (như phpMyAdmin) hoạt động. Tuy nhiên, Introspection Cơ sở dữ liệu có nhiều ưu điểm độc đáo:

* Kiểm soát tuyệt đối lược đồ cơ sở dữ liệu SQL thuần túy.
* Hoàn toàn minh bạch, di động và an toàn cho dữ liệu.
* Cho phép nhập các cơ sở dữ liệu hiện có mà không bị thay đổi và không cần di chuyển.
* Truy cập trực tiếp cơ sở dữ liệu và toàn bộ sức mạnh của các truy vấn SQL thô, phức tạp.
* Cải thiện hiệu suất đáng kể thông qua tối ưu hóa và lập chỉ mục.

Ngược lại, các nền tảng khác thường sử dụng mô hình dữ liệu định trước hoặc độc quyền một kích cỡ cho tất cả để lưu trữ nội dung. Điều này không xảy ra với Directus.

Directus cung cấp cho chúng ta quyền truy cập trực tiếp vào dữ liệu thuần túy và không bị thay đổi. Điều đó có nghĩa là ta có tùy chọn bỏ qua lớp trung gian của Directus (API, SDK, Ứng dụng) và kết nối với dữ liệu bằng các truy vấn SQL phù hợp. Điều này loại bỏ hiệu quả tất cả các nút thắt cổ chai, độ trễ và hạn chế truy cập độc quyền.

### **Tại sao nên sử dụng Directus**

Directus là một giải pháp đơn giản, ưu tiên dữ liệu cho các vấn đề phức tạp mà không có nhược điểm nào.

Directus được tạo ra vào năm 2004 và đã được cải tiến trong một thời gian dài. Đội ngũ chính đã suy nghĩ cẩn thận về ứng dụng này từ đầu đến cuối và cố gắng làm cho nền tảng này mang lại tất cả lợi ích và không có nhược điểm. Ta có thể liên kết hoặc gỡ bỏ bất kỳ lúc nào mà không ảnh hưởng đến dữ liệu. Ta không bị khóa vào bất kỳ nhà cung cấp nào, dữ liệu là của chúng ta. Không có giới hạn dữ liệu nhân tạo nào (ví dụ: giới hạn người dùng hoặc vai trò).

Xem xét các nền tảng cạnh tranh trong bốn trường hợp sử dụng chung được đề cập ở trên. Việc chọn một giải pháp khả thi từ một trong các danh mục này có thể mất sáu tháng để đảm bảo danh sách tính năng, giá cả, khả năng mở rộng, tùy chọn di chuyển, v.v. đều phù hợp với dự án. Tuy nhiên, với Directus, điều đó không xảy ra! Chúng ta có thể thử ngay trên cơ sở dữ liệu hiện có hoặc xây dựng một mô hình dữ liệu mới từ đầu, mà không ảnh hưởng đến dữ liệu và không cam kết lâu dài.

Những nguyên tắc cốt lõi sau đây hướng dẫn nền tảng này:

* Pure: Không có lược đồ định trước hoặc độc quyền. Tất cả siêu dữ liệu hệ thống được lưu trữ riêng biệt.
* Open: Directus Core là mã nguồn mở, không có code bị che khuất hoặc chỉ dành cho đám mây.
* Portable: Không bị khóa nhà cung cấp. Cơ sở dữ liệu có thể được xuất hoặc di chuyển bất kỳ lúc nào.
* Limitless: Không có giới hạn tùy ý hoặc paywalls về người dùng, vai trò, bản dịch và dữ liệu.
* Extensible: Mọi khía cạnh của nền tảng này đều có thể mở rộng để tránh bất kỳ giới hạn tính năng nào.
* Unopinionated: Chọn stack, cơ sở dữ liệu và kiến trúc theo ý muốn.

## **Ứng dụng Directus vào đồ án**

Nội dung đồ án sử dụng gói **@directus/sdk** của Directus để thao tác với dữ liệu

### **Transaction**

| **STT** | **Chức năng** | **Sử dụng** |
| --- | --- | --- |
| 1 | Tạo mới | await directusInstance.request(  createItems(TRANSACTION\_KEY, {  name: name,  type: type,  total: money,  trading\_date: inputDate,  description: description,  category: category,  account\_id: account,  } as any),  ) |
| 2 | Cập nhật | await directusInstance.request(  updateItem(TRANSACTION\_KEY, props.route?.params?.id, {  type: type,  total: money,  trading\_date: inputDate,  description: description,  category: category,  account\_id: account,  }),  ) |
| 3 | Xoá | await directusInstance.request(  deleteItem(TRANSACTION\_KEY, props.route?.params?.id),  ) |
| 4 | Filter | Không |

### **Account**

| **STT** | **Chức năng** | **Sử dụng** |
| --- | --- | --- |
| 1 | Tạo mới | directusInstance.request(  createItems('account', {  total: money,  name,  include\_in\_balance,  user\_id: {  email: user.email,  },  }),  ) |
| 2 | Cập nhật | await directusInstance.request(  updateItem('account', route.params.id, {  total: money,  name,  include\_in\_balance,  user\_id: {  email: user.email,  },  }),  ) |
| 3 | Xoá | Không |
| 4 | Filter | await directusInstance.request(  readItems('account', {  filter: {  name : name  },  }),  ) |

### **User**

| **STT** | **Chức năng** | **Sử dụng** |
| --- | --- | --- |
| 1 | Tạo mới | directusInstance.request(  createItem('app\_user', {  email: email,  full\_name: username,  }),  ) |
| 2 | Cập nhật | directusInstance.request(  updateItem('app\_user', user.id, {  full\_name : user.full\_name,  phone\_number : user.phone\_number,  gender : user.gender  }),  ) |
| 3 | Xoá | Không |
| 4 | Filter | directusInstance.request(  readItems('app\_user', {  filter: {  email: {  \_eq: email,  },  },  }),  ) |

### **Payment**

| **STT** | **Chức năng** | **Sử dụng** |
| --- | --- | --- |
| 1 | Tạo mới | directusInstance.request(  createItems('payment', {  type: type,  name: name,  total: total,  cycle\_day: cycle\_day,  description: description,  category: category,  from: from,  to: to,  add\_automation : addAutomation,  account\_id: account  }),  ) |
| 2 | Cập nhập | Không |
| 3 | Xoá | Không |
| 4 | Filter | await directusInstance.request(  readItems('payment', {  filter: {  account\_id: {  \_in : accounts.map((acc:any) => acc.id)  }  },  }),  ) |

### **Cron Job**

| **STT** | **Chức năng** | **Sử dụng** |
| --- | --- | --- |
| 1 | Kiểm tra regular payment và tạo mới transaction nếu tới hạn | POST API:  http://0.0.0.0:8055/transaction-operation/payment\_cronjob |

# 

# **CHƯƠNG 5: CODE CÁC CHỨC NĂNG CHÍNH**

## **Chức năng đăng nhập , đăng ký**

| import \* as React from 'react';  import {View, StyleSheet} from 'react-native';  import {TextInput, Button, Avatar, Text} from 'react-native-paper';  import {signInUserPassword, signInWithGoogle} from '../../services/oauth';  import {realm} from '../../services/realm';  import {getProfile, saveProfile} from '../../storages/profile';  import {useEffect} from 'react';  const Login = ({navigation}: any) => {  const [email, setEmail] = React.useState('');  const [password, setPassword] = React.useState('');  const [showPassword, setShowPassword] = React.useState(false);  const handleLogin = () => {  signInUserPassword(email, password).then(() => {  saveProfile(email, password).then(() => {  navigation.push('Home');  });  });  };  const goToForgetPass = () => {  navigation.navigate('ForgetPassword');  };  const goToSignUp = () => {  navigation.navigate('SignUp');  };  const handleSignInWithGooglePress = () => {  signInWithGoogle();  };  useEffect(() => {  getProfile().then(result => {  if (result.length > 0) {  navigation.push('Home');  }  });  }, []);  return (  <View style={styles.container}>  <Text style={styles.title}>Sign In</Text>  <Button  mode='contained'  contentStyle={styles.loginWithGoogle}  style={styles.loginWithGoogleWrap}  onPress={handleSignInWithGooglePress}  icon={() => (  <Avatar.Image  style={styles.loginWithLeft}  size={48}  source={require('../../assets/google.png')}  />  )}  textColor='#000'>  <Text variant='titleSmall' style={styles.loginWithGoogleText}>  Login with Google  </Text>  </Button>  <View style={styles.or}>  <View style={styles.orline} />  <Text style={styles.orText}>OR</Text>  <View style={styles.orline} />  </View>  <TextInput  mode='outlined'  label='Email'  placeholder='Enter your Email'  value={email}  onChangeText={setEmail}  style={styles.input}  />  <TextInput  mode='outlined'  label='Password'  placeholder='Enter your Pass'  value={password}  onChangeText={setPassword}  style={styles.input}  secureTextEntry={!showPassword}  right={  <TextInput.Icon  icon={showPassword ? 'eye-off' : 'eye'}  onPress={() => setShowPassword(!showPassword)}  />  }  />  <Button mode='contained' onPress={handleLogin} style={styles.button}>  Đăng nhập  </Button>  <View style={styles.gotoSignup}>  <Text variant='bodyMedium'>  Don’t have account?{' '}  <Text onPress={goToSignUp} style={styles.link}>  Sign Up  </Text>  </Text>  <Text onPress={goToForgetPass} variant='bodyMedium' style={styles.link}>  Forget password  </Text>  </View>  </View>  );  };  const styles = StyleSheet.create({  container: {  flex: 1,  justifyContent: 'center',  alignItems: 'center',  paddingHorizontal: 20,  gap: 20,  },  title: {  fontSize: 32,  fontWeight: 'bold',  marginBottom: 20,  color: '#000',  },  input: {  width: '100%',  },  button: {  width: '100%',  },  loginWithTitle: {  fontSize: 16,  color: '#000',  fontWeight: 'bold',  },  loginWithLeft: {  backgroundColor: 'rgba(0,0,0,0)',  },  loginWithGoogle: {  backgroundColor: '#E3F4E1',  display: 'flex',  flexDirection: 'row',  alignItems: 'center',  justifyContent: 'flex-start',  gap: 12,  width: '100%',  },  loginWithGoogleWrap: {  width: '100%',  },  loginWithGoogleText: {  lineHeight: 72,  textAlignVertical: 'center',  },  or: {  display: 'flex',  flexDirection: 'row',  justifyContent: 'space-between',  alignItems: 'center',  width: '100%',  },  orText: {  color: '#000',  },  gotoSignup: {  flexDirection: 'row',  justifyContent: 'space-between',  width: '100%',  alignItems: 'flex-start',  },  orline: {  height: 1,  backgroundColor: '',  width: '45%',  },  link: {  color: '#1573FE',  },  });  export default Login; |
| --- |

| import React from 'react';  import {Avatar, Button, Text, TextInput} from 'react-native-paper';  import {signInUserPassword, signUpWithUserPassword} from '../../services/oauth';  import {StyleSheet, View} from 'react-native';  import {createAppUser} from '../../controllers/auth.controller';  const SignUpScreen = ({navigation}: any) => {  const [email, setEmail] = React.useState('');  const [fullName, setFullName] = React.useState('');  const [password, setPassword] = React.useState('');  const [showPassword, setShowPassword] = React.useState(false);  const handleSignUp = () => {  createAppUser(email, password, fullName).then(() => {  navigation.push('Login');  });  };  const goToSignIn = () => {  navigation.navigate('Login');  };  return (  <View style={styles.container}>  <Text style={styles.title}>SIGN UP</Text>  <TextInput  mode="outlined"  label="Full Name"  placeholder="Enter your Full name"  value={fullName}  onChangeText={setFullName}  style={styles.input}  />  <TextInput  mode="outlined"  label="Email"  placeholder="Enter your Email"  value={email}  onChangeText={setEmail}  style={styles.input}  />  <TextInput  mode="outlined"  label="Password"  placeholder="Enter your Pass"  value={password}  onChangeText={setPassword}  style={styles.input}  secureTextEntry={!showPassword}  right={  <TextInput.Icon  icon={showPassword ? 'eye-off' : 'eye'}  onPress={() => setShowPassword(!showPassword)}  />  }  />  <Button mode="contained" onPress={handleSignUp} style={styles.button}>  Create new account  </Button>  <View style={styles.gotoSignIn}>  <Text variant="bodyMedium">  Already have account?{' '}  <Text onPress={goToSignIn} style={styles.link}>  Sign In  </Text>  </Text>  </View>  </View>  );  };  const styles = StyleSheet.create({  container: {  flex: 1,  justifyContent: 'center',  alignItems: 'center',  paddingHorizontal: 20,  gap: 20,  },  title: {  fontSize: 32,  fontWeight: 'bold',  marginBottom: 20,  color: '#000',  },  input: {  width: '100%',  },  button: {  width: '100%',  },  loginWithTitle: {  fontSize: 16,  color: '#000',  fontWeight: 'bold',  },  loginWithLeft: {  backgroundColor: 'rgba(0,0,0,0)',  },  loginWithGoogle: {  backgroundColor: '#E3F4E1',  display: 'flex',  flexDirection: 'row',  alignItems: 'center',  justifyContent: 'flex-start',  gap: 12,  width: '100%',  },  loginWithGoogleWrap: {  width: '100%',  },  loginWithGoogleText: {  lineHeight: 72,  textAlignVertical: 'center',  },  or: {  display: 'flex',  flexDirection: 'row',  justifyContent: 'space-between',  alignItems: 'center',  width: '100%',  },  orText: {  color: '#000',  },  gotoSignIn: {  justifyContent: 'space-between',  width: '100%',  alignItems: 'center',  },  orline: {  height: 1,  backgroundColor: '',  width: '45%',  },  link: {  color: '#1573FE',  },  });  export default SignUpScreen; |
| --- |

## **Chức năng transaction**

| **import { readItems } from '@directus/sdk'**  **import { useIsFocused, useNavigation } from '@react-navigation/native'**  **import { useCallback, useContext, useEffect, useState } from 'react'**  **import { SafeAreaView, View } from 'react-native'**  **import {**  **ScrollView,**  **TouchableWithoutFeedback,**  **} from 'react-native-gesture-handler'**  **import { Button, Icon, IconButton, Menu, SegmentedButtons, Text } from 'react-native-paper'**  **import TransactionCard from '../../components/transaction/TransactionCard'**  **import {**  **ACCOUNT\_KEY,**  **TRANSACTION\_KEY,**  **} from '../../contants/schema-key.constant'**  **import { allAccount } from '../../contants/transaction/empty-account.constant'**  **import { filterItems } from '../../contants/transaction/filter-item'**  **import { GlobalContext } from '../../contexts/context'**  **import { directusInstance } from '../../services/directus'**  **import { Transaction as TTransaction } from '../../types/transaction'**  **import { formatVND } from '../../utils/money'**  **import { filterDates, getFirstDateFilter } from '../../utils/number'**  **export default function Transaction() {**  **const { user, account, setAccount } = useContext(GlobalContext)**  **const navigation = useNavigation()**  **const isFocused = useIsFocused()**  **const [showAccountsDropdown, setShowAccountsDropdown] = useState(false)**  **const [accounts, setAccounts] = useState<any[]>([])**  **const [transactions, setTransactions] = useState<TTransaction[]>([])**  **const [filter, setFilter] = useState(filterItems[0].value)**  **const fetchTransactions = async () => {**  **const response = (await directusInstance.request(**  **readItems(TRANSACTION\_KEY, {**  **sort: ['-trading\_date'],**  **filter: {**  **account\_id: account.id == "-1" ? {**  **user\_id: user?.id**  **} : account.id,**  **trading\_date: {**  **\_gte: getFirstDateFilter(filter),**  **\_lte: new Date()**  **}**  **},**  **}),**  **)) as any**  **setTransactions(response)**  **}**  **const fetchAccounts = useCallback(async () => {**  **const accounts = (await directusInstance.request(**  **readItems(ACCOUNT\_KEY, {**  **filter: {**  **user\_id: user?.id**  **},**  **}),**  **)) as any**  **setAccounts([**  **allAccount,**  **...accounts**  **])**  **}, [])**  **let totalSpending = 0,**  **totalEarning = 0**  **transactions.forEach(item =>**  **item.type === 'expenses'**  **? (totalSpending += item.total)**  **: (totalEarning += item.total),**  **)**  **useEffect(() => {**  **if (isFocused && user.email) {**  **fetchAccounts()**  **}**  **}, [isFocused])**  **useEffect(() => {**  **if (account) {**  **fetchTransactions()**  **}**  **}, [account, filter])**  **return (**  **<SafeAreaView className='flex-1'>**  **<View className='pt-4 pb-2 flex-row items-center border-b border-[#ccc] mx-4 flex justify-between'>**  **<Text className='text-xl font-bold items-center align-middle'>Transactions Manager</Text>**  **<View className='items-center '>**  **<IconButton**  **icon="plus"**  **onPress={() => navigation.navigate('Transaction Info')}>**  **</IconButton>**  **</View>**  **</View>**  **<View className='flex-1 pt-2 pb-4 bg-[#fafafa] flex flex-col'>**  **<ScrollView className='px-4'>**  **<TouchableWithoutFeedback**  **onPress={() => setShowAccountsDropdown(true)}**  **className='mb-2'>**  **<View className='flex flex-row justify-start items-center'>**  **<View className='flex flex-row gap-2'>**  **<Text className='font-bold text-base'>**  **{accounts.find(i => i.id === account.id)?.name}**  **</Text>**  **</View>**  **<Menu**  **visible={showAccountsDropdown}**  **onDismiss={() => setShowAccountsDropdown(false)}**  **anchor={**  **<Button>**  **<Icon source='menu-down' color={'black'} size={20} />**  **</Button>**  **}>**  **{accounts.map((account, index) => (**  **<Menu.Item**  **key={index}**  **onPress={() => {**  **setAccount(account)**  **setShowAccountsDropdown(false)**  **}}**  **title={account.name}**  **/>**  **))}**  **</Menu>**  **<View className='flex-1 flex justify-end flex-row'>**  **<Text className=''>**  **Total:{' '}**  **<Text className='font-bold text-base'>**  **{formatVND(**  **account.id === '-1'**  **? accounts.reduce((acc, item) => acc + item.total, 0)**  **: accounts.find(item => item.id === account.id)?.total ||**  **0,**  **)}**  **</Text>**  **</Text>**  **</View>**  **</View>**  **</TouchableWithoutFeedback>**  **<SegmentedButtons**  **value={filter}**  **onValueChange={setFilter}**  **buttons={filterItems}**  **density='small'**  **/>**  **<View className='mt-4 items-center gap-x-3'>**  **<Text className=''>**  **{filterDates(filter)}**  **</Text>**  **</View>**  **<View className='mt-4 flex flex-row items-center gap-x-3'>**  **<Text className='text-xs text-[#7D8895] mt-1'>**  **My Total Earnings:**  **</Text>**  **<Text className='font-bold text-xl text-green-700'>**  **{formatVND(totalEarning)}**  **</Text>**  **</View>**  **<View className='flex flex-row items-center gap-x-3'>**  **<Text className='text-xs text-[#s7D8895] mt-1'>**  **My Total Spending:**  **</Text>**  **<Text className='font-bold text-xl text-red-700'>**  **{formatVND(totalSpending)}**  **</Text>**  **</View>**  **<View className='mt-4'>**  **<View className='flex mb-2 flex-row items-center justify-between'>**  **<Text className='font-medium'>Recently expenses</Text>**  **<Button onPress={() => navigation.navigate('All Transactions')}>**  **See All {'>'}**  **</Button>**  **</View>**  **<View className='w-full mt-1 flex flex-col'>**  **{transactions**  **.filter((transaction) => transaction.type == "expenses").length > 0 ? (**  **transactions**  **.filter((transaction) => transaction.type == "expenses")**  **.slice(0, 3)**  **.map((transaction, index) => (**  **<TransactionCard key={index} {...transaction} />**  **))**  **) : (**  **<Text className='text-center text-gray-400 mb-2'>**  **The is no expenses here, create one now!**  **</Text>**  **)}**  **</View>**  **</View>**  **<View className='mt-4'>**  **<View className='flex mb-2 flex-row items-center justify-between'>**  **<Text className='font-medium'>Recently income</Text>**  **<Button onPress={() => navigation.navigate('All Transactions')}>**  **See All {'>'}**  **</Button>**  **</View>**  **<View className='w-full mt-1 flex flex-col'>**  **{transactions**  **.filter((transaction) => transaction.type == "income").length > 0 ? (**  **transactions**  **.filter((transaction) => transaction.type == "income")**  **.slice(0, 3)**  **.map((transaction, index) => (**  **<TransactionCard key={index} {...transaction} />**  **))**  **) : (**  **<Text className='text-center text-gray-400 mb-2'>**  **The is no income here, create one now!**  **</Text>**  **)}**  **</View>**  **</View>**  **</ScrollView>**  **</View>**  **</SafeAreaView>**  **)**  **}** |
| --- |

| **import { useContext, useEffect, useMemo, useState } from 'react'**  **import { View } from 'react-native'**  **import {**  **createItems,**  **deleteItem,**  **readItem,**  **readItems,**  **updateItem,**  **} from '@directus/sdk'**  **import { useNavigation } from '@react-navigation/native'**  **import {**  **Button,**  **Dialog,**  **MD3Colors,**  **Portal,**  **SegmentedButtons,**  **Text,**  **TextInput,**  **} from 'react-native-paper'**  **import { DatePickerInput } from 'react-native-paper-dates'**  **import DropDown from 'react-native-paper-dropdown'**  **import { GlobalContext } from '../../contexts/context'**  **import { SnackBarContext } from '../../hocs/SnackBar'**  **import { CategoryList, TransactionType } from '../../services/const'**  **import { directusInstance } from '../../services/directus'**  **import {**  **ACCOUNT\_KEY,**  **TRANSACTION\_KEY,**  **} from '../../contants/schema-key.constant'**  **import { Account } from '../../types/account'**  **import Loading from '../../components/Loading'**  **export default function CreateTransaction(props: any) {**  **const transactionId = useMemo(() => props.route?.params?.id, [])**  **const mode = transactionId ? 'update' : 'create'**  **const navigation = useNavigation()**  **const [accounts, setAccounts] = useState<Account[]>([])**  **const [accountsOptions, setAccountOptions] = useState([])**  **const { user } = useContext(GlobalContext)**  **const { setData } = useContext(SnackBarContext)**  **const [oldData, setOldData] = useState({} as any)**  **// Form data**  **const [type, setType] = useState(TransactionType[0].value)**  **const [name, setName] = useState('')**  **const [money, setMoney] = useState(0)**  **const [category, setCategory] = useState('')**  **const [account, setAccount] = useState('')**  **const [inputDate, setInputDate] = useState(undefined)**  **const [description, setDescription] = useState('')**  **const [loading , setLoading] = useState(false);**  **const [showCategories, setShowCategories] = useState(false)**  **const [showAccounts, setShowAccounts] = useState(false)**  **const [visible, setVisible] = useState(false)**  **const formReady = useMemo(**  **() => name && money && category && account && inputDate,**  **[money, category, inputDate],**  **)**  **const showDialog = () => setVisible(true)**  **const hideDialog = () => setVisible(false)**  **const createTransaction = async () => {**  **try {**  **await directusInstance.request(**  **createItems(TRANSACTION\_KEY, {**  **name: name,**  **type: type,**  **total: money,**  **trading\_date: inputDate,**  **description: description,**  **category: category,**  **account\_id: account,**  **} as any),**  **)**  **setData({ text: 'Create transaction successfully!' })**  **navigation.goBack()**  **} catch (e) {**  **console.log('error when creating a new transaction: ', e)**  **}**  **}**  **const updateTransaction = async () => {**  **await directusInstance.request(**  **updateItem(TRANSACTION\_KEY, props.route?.params?.id, {**  **type: type,**  **total: money,**  **trading\_date: inputDate,**  **description: description,**  **category: category,**  **account\_id: account,**  **}),**  **)**  **setData({ text: 'Create transaction successful!' })**  **navigation.goBack()**  **}**  **const deleteTransaction = async () => {**  **await directusInstance.request(**  **deleteItem(TRANSACTION\_KEY, props.route?.params?.id),**  **)**  **await directusInstance.request(**  **updateItem(ACCOUNT\_KEY, oldData.account\_id, {**  **total:**  **(accounts.find(i => i.id === oldData.account\_id) as any).total +**  **(oldData.type === 'income' ? -oldData.total : oldData.total),**  **}),**  **)**  **setData({ text: 'Create transaction successful!' })**  **navigation.goBack()**  **}**  **useEffect(() => {**  **if (user) {**  **;(async () => {**  **const res = await directusInstance.request(**  **readItems('account', {**  **filter: {**  **user\_id: user?.id**  **},**  **}),**  **)**  **setAccountOptions(**  **(res as any).map((i: { name: any; id: any }) => ({**  **label: i.name,**  **value: i.id,**  **})),**  **)**  **setAccounts(res as any)**  **})()**  **}**  **}, [user])**  **useEffect(() => {**  **if (mode === 'update') {**  **setLoading(true);**  **(async () => {**  **const transactionData = await directusInstance.request(**  **readItem(TRANSACTION\_KEY, transactionId),**  **)**  **setName(transactionData.name)**  **setType(transactionData.type)**  **setCategory(transactionData.category)**  **const dateData = new Date(transactionData.trading\_date)**  **setInputDate(dateData as any)**  **setDescription(transactionData.description)**  **setAccount(transactionData.account\_id)**  **setMoney(transactionData.total)**  **setOldData(transactionData)**  **setLoading(false);**  **})()**  **}**  **}, [transactionId])**  **return (**  **loading ? <Loading/> : <View className='flex flex-1 flex-col justify-start p-3 gap-2'>**  **<SegmentedButtons**  **value={type}**  **onValueChange={setType}**  **buttons={TransactionType}**  **/>**  **<View>**  **<TextInput**  **label='Name'**  **value={name}**  **onChangeText={value => setName(value)}**  **mode='outlined'**  **/>**  **</View>**  **<View className='flex flex-row justify-around items-center'>**  **<TextInput**  **label='Money'**  **mode='outlined'**  **keyboardType='numeric'**  **className='flex-1 mr-4 font-bold text-xl'**  **value={money.toString()}**  **onChangeText={e => {**  **setMoney(+e)**  **}}**  **/>**  **<Text variant='bodyLarge'>VND</Text>**  **</View>**  **<View className=''>**  **<DropDown**  **label={'Category'}**  **mode='outlined'**  **setValue={setCategory}**  **list={CategoryList}**  **visible={showCategories}**  **showDropDown={() => setShowCategories(true)}**  **onDismiss={() => setShowCategories(false)}**  **value={category}**  **/>**  **</View>**  **<View className=''>**  **<DropDown**  **label={'Account'}**  **mode='outlined'**  **setValue={setAccount}**  **list={accountsOptions}**  **visible={showAccounts}**  **showDropDown={() => setShowAccounts(true)}**  **onDismiss={() => setShowAccounts(false)}**  **value={account}**  **/>**  **</View>**  **<View className='pt-8'>**  **<DatePickerInput**  **locale='en'**  **label='Date'**  **mode='outlined'**  **value={inputDate}**  **onChange={(d: any) => setInputDate(d)}**  **inputMode='start'**  **/>**  **</View>**  **<View className='pt-8'>**  **<TextInput**  **label={'Description'}**  **value={description}**  **onChangeText={e => setDescription(e)}**  **mode='outlined'**  **multiline={true}**  **numberOfLines={5}**  **/>**  **</View>**  **<View className='pt-4'>**  **{transactionId ? (**  **<>**  **<Button**  **mode='contained-tonal'**  **onPress={updateTransaction}**  **// disabled={!formReady}**  **className='mb-4'>**  **Update**  **</Button>**  **<Button**  **mode='contained'**  **buttonColor={MD3Colors.error50}**  **onPress={showDialog}>**  **Delete**  **</Button>**  **<Portal>**  **<Dialog visible={visible} onDismiss={hideDialog}>**  **<Dialog.Title>Alert</Dialog.Title>**  **<Dialog.Content>**  **<Text variant='bodyMedium'>**  **Are you sure you want to delete this transaction ?**  **</Text>**  **</Dialog.Content>**  **<Dialog.Actions>**  **<Button**  **buttonColor={MD3Colors.error50}**  **textColor='#fff'**  **className='px-4'**  **onPress={deleteTransaction}>**  **Yes**  **</Button>**  **<Button**  **mode='contained'**  **className='px-4'**  **onPress={hideDialog}>**  **No**  **</Button>**  **</Dialog.Actions>**  **</Dialog>**  **</Portal>**  **</>**  **) : (**  **<Button**  **mode='contained'**  **onPress={createTransaction}**  **disabled={!formReady}>**  **Save**  **</Button>**  **)}**  **</View>**  **</View>**  **)**  **}** |
| --- |

| **import {**  **useCallback,**  **useContext,**  **useEffect,**  **useState,**  **useTransition,**  **} from 'react'**  **import { Button, Icon, Menu, Text } from 'react-native-paper'**  **import { Account } from '../../../types/account'**  **import { Transaction } from '../../../types/transaction'**  **import { GlobalContext } from '../../../contexts/context'**  **import { useNavigation } from '@react-navigation/native'**  **import { directusInstance } from '../../../services/directus'**  **import { readItems } from '@directus/sdk'**  **import {**  **ACCOUNT\_KEY,**  **TRANSACTION\_KEY,**  **} from '../../../contants/schema-key.constant'**  **import { allAccount } from '../../../contants/transaction/empty-account.constant'**  **import { ActivityIndicator, FlatList, View } from 'react-native'**  **import {**  **ScrollView,**  **TouchableWithoutFeedback,**  **} from 'react-native-gesture-handler'**  **import { formatVND } from '../../../utils/money'**  **import TransactionCard from '../../../components/transaction/TransactionCard'**  **const PAGE\_SIZE = 10**  **export default function AllTransactions() {**  **const { user, account, setAccount } = useContext(GlobalContext)**  **const navigation = useNavigation()**  **const [isLoading, setLoading] = useState(false)**  **const [accounts, setAccounts] = useState<Account[]>([])**  **const [type, setType] = useState('all')**  **const [transactions, setTransactions] = useState<Transaction[]>([])**  **const [showAccountsDropdown, setShowAccountsDropdown] = useState(false)**  **const [showTypeDropdown, setShowTypeDropdown] = useState(false)**  **const [page, setPage] = useState(1)**  **const [isEndOfList, setEndOfList] = useState(false)**  **const fetchAccounts = async () => {**  **const accounts = (await directusInstance.request(**  **readItems(ACCOUNT\_KEY, {**  **filter: {**  **user\_id: user?.id,**  **},**  **sort: ['-name'],**  **}),**  **)) as any**  **setAccounts([allAccount, ...accounts])**  **}**  **const fetchTransactions = async () => {**  **setLoading(true)**  **var filter = {}**  **if (type == 'all') {**  **filter = {**  **account\_id:**  **account.id == '-1'**  **? {**  **user\_id: user?.id,**  **}**  **: account.id,**  **}**  **} else {**  **filter = {**  **type: type,**  **account\_id:**  **account.id == '-1'**  **? {**  **user\_id: user?.id,**  **}**  **: account.id,**  **}**  **}**  **const response = (await directusInstance.request(**  **readItems(TRANSACTION\_KEY, {**  **sort: ['-trading\_date'],**  **filter: filter,**  **limit: PAGE\_SIZE, // Number of items to return**  **offset: (page - 1) \* PAGE\_SIZE, // Number of items to skip**  **}),**  **)) as any**  **if (response.length < PAGE\_SIZE) setEndOfList(true)**  **setTransactions(prev => [...prev, ...response])**  **setLoading(false)**  **}**  **useEffect(() => {**  **user.email && fetchAccounts()**  **}, [])**  **useEffect(() => {**  **setTransactions([])**  **setPage(1)**  **setEndOfList(false)**  **}, [account, type])**  **useEffect(() => {**  **if (account && !isEndOfList) {**  **fetchTransactions()**  **}**  **}, [account, page, isEndOfList, type ])**  **return (**  **<View className='flex-1 bg-blue-50 px-3 pb-3'>**  **<View className='flex flex-row justify-between'>**  **<TouchableWithoutFeedback**  **onPress={() => setShowAccountsDropdown(true)}**  **className='mb-2'>**  **<View className='flex flex-row justify-start items-center'>**  **<View className='flex items-center flex-row gap-2'>**  **<Text className='text-lg'>Account: </Text>**  **<Text className='font-bold text-base'>{account?.name}</Text>**  **</View>**  **<Menu**  **visible={showAccountsDropdown}**  **onDismiss={() => setShowAccountsDropdown(false)}**  **anchor={**  **<Button>**  **<Icon source='menu-down' color={'black'} size={20} />**  **</Button>**  **}>**  **{accounts.map((acc, index) => (**  **<Menu.Item**  **key={index}**  **onPress={() => {**  **setAccount(acc)**  **setShowAccountsDropdown(false)**  **}}**  **title={acc.name}**  **/>**  **))}**  **</Menu>**  **</View>**  **</TouchableWithoutFeedback>**  **<TouchableWithoutFeedback**  **onPress={() => setShowTypeDropdown(true)}**  **className='flex items-end w-fit translate-x-6'>**  **<View className='flex flex-row items-center align-right'>**  **<View className='flex items-center flex-row gap-2'>**  **<Text className='text-lg'>Type: </Text>**  **<Text className='font-bold text-base'>{type}</Text>**  **</View>**  **<Menu**  **visible={showTypeDropdown}**  **onDismiss={() => setShowTypeDropdown(false)}**  **anchor={**  **<Button>**  **<Icon source='menu-down' color={'black'} size={20} />**  **</Button>**  **}>**  **<Menu.Item**  **key={1}**  **onPress={() => {**  **setType('all')**  **setShowTypeDropdown(false)**  **}}**  **title={'all'}**  **/>**  **<Menu.Item**  **key={1}**  **onPress={() => {**  **setType('expenses')**  **setShowTypeDropdown(false)**  **}}**  **title={'expenses'}**  **/>**  **<Menu.Item**  **key={2}**  **onPress={() => {**  **setType('income')**  **setShowTypeDropdown(false)**  **}}**  **title={'income'}**  **/>**  **</Menu>**  **</View>**  **</TouchableWithoutFeedback>**  **</View>**  **<View className='flex-1'>**  **<FlatList**  **data={transactions}**  **renderItem={({ item }) => {**  **return <TransactionCard {...item} />**  **}}**  **keyExtractor={item => item.id}**  **ListFooterComponent={() =>**  **isLoading ? (**  **<View className='my-4 items-center'>**  **<ActivityIndicator size='large' color='#aaa' />**  **</View>**  **) : (**  **<></>**  **)**  **}**  **onEndReached={() =>**  **transactions.length > 0 && !isEndOfList && setPage(prev => prev + 1)**  **}**  **ListEmptyComponent={() => (**  **<View className='flex-1 flex items-center justify-center'>**  **<Text>Empty</Text>**  **</View>**  **)}**  **/>**  **</View>**  **</View>**  **)**  **}** |
| --- |

## **Chức năng account**

| **import { useContext, useEffect, useState } from 'react'**  **import { Switch, View } from 'react-native'**  **import { Button, Text, TextInput } from 'react-native-paper'**  **import { directusInstance } from '../../services/directus'**  **import { createItems, updateItem } from '@directus/sdk'**  **import { GlobalContext } from '../../contexts/context'**  **import { SnackBarContext } from '../../hocs/SnackBar'**  **import { useNavigation } from '@react-navigation/native'**  **export default function AccountNew({ route }) {**  **const [money, setMoney] = useState(0)**  **const [name, setName] = useState('')**  **const [include\_in\_balance, setIncludeInBalnace] = useState(false)**  **const [buttonLabel, setButtonLabel] = useState("Add")**  **const { user } = useContext(GlobalContext)**  **const { setData } = useContext(SnackBarContext)**  **const navigation = useNavigation()**  **const buttonEnable = money && name !== ''**  **const createOrUpdateAccount = async () => {**  **if (route.params) {**  **const res = await directusInstance.request(**  **updateItem('account', route.params.id, {**  **total: money,**  **name,**  **include\_in\_balance,**  **user\_id: user.id**  **}),**  **)**  **if (res) {**  **setData({ text: 'Update account successful!' })**  **navigation.goBack()**  **}**  **} else {**  **const res = await directusInstance.request(**  **createItems('account', {**  **total: money,**  **name,**  **include\_in\_balance,**  **user\_id: user.id**  **}),**  **)**  **if (res) {**  **setData({ text: 'Create account successful!' })**  **navigation.goBack()**  **}**  **}**  **}**  **useEffect(() => {**  **if (route.params) {**  **const data = route.params**  **setMoney(data.total)**  **setName(data.name)**  **setIncludeInBalnace(data.include\_in\_balance == 'true')**  **setButtonLabel("Update")**  **}**  **}, [route])**  **return (**  **<View className='w-full flex flex-col p-3 gap-6'>**  **<View className='w-full flex flex-row justify-center items-center gap-3'>**  **<TextInput**  **keyboardType='number-pad'**  **mode='flat'**  **className='w-[40%] bg-transparent'**  **value={`${money}`}**  **onChangeText={v => setMoney(+v)}**  **/>**  **<Text variant='headlineMedium'>VND</Text>**  **</View>**  **<TextInput**  **className='w-full'**  **mode='outlined'**  **label='Account name'**  **value={name}**  **onChangeText={setName}**  **/>**  **<View className='w-full flex flex-row justify-between items-center'>**  **<Text variant='bodyMedium'>Do not include in total balance {include\_in\_balance}</Text>**  **<Switch**  **value={include\_in\_balance}**  **onValueChange={e => { setIncludeInBalnace(!include\_in\_balance) }}**  **/>**  **</View>**  **<Button**  **mode='contained-tonal'**  **className='w-full'**  **onPress={createOrUpdateAccount}**  **disabled={!buttonEnable}>**  **{buttonLabel}**  **</Button>**  **</View>**  **)**  **}** |
| --- |

| **import { useContext, useEffect, useMemo, useState } from 'react'**  **import { Avatar, Card, Icon, Text } from 'react-native-paper'**  **import { HeaderContext } from '../../contexts/header'**  **import { TouchableOpacity, View } from 'react-native'**  **import { useIsFocused, useNavigation } from '@react-navigation/native'**  **import { directusInstance } from '../../services/directus'**  **import { readItems } from '@directus/sdk'**  **import { GlobalContext } from '../../contexts/context'**  **export default function AccountPage(props: any) {**  **const { subfix, setSubfix } = useContext(HeaderContext)**  **const [accounts, setAccounts] = useState([])**  **const sum = useMemo(() => (accounts.reduce((accumulator, account: any) => accumulator + account.total, 0)) / 1000000**  **, [accounts])**  **const { user } = useContext(GlobalContext)**  **const navigation = useNavigation()**  **const isFocused = useIsFocused()**  **useEffect(() => {**  **if (isFocused) {**  **setSubfix({**  **icon: 'plus',**  **onPress: () => {**  **navigation.navigate('Account Info')**  **setSubfix(null)**  **}**  **})**  **}**  **}, [props, isFocused])**  **useEffect(() => {**  **; (async () => {**  **const res = await directusInstance.request(**  **readItems('account', {**  **filter: {**  **user\_id: user.id**  **},**  **}),**  **)**  **setAccounts(res)**  **})()**  **}, [user, isFocused])**  **return (**  **<View className='w-full flex flex-col items-center p-3 pt-6 gap-y-8'>**  **<View className='w-full flex flex-col items-center gap-2'>**  **<Text variant='titleMedium'>Totals</Text>**  **<Text variant='headlineMedium'>vnđ {sum}M</Text>**  **</View>**  **<View className='w-full flex flex-row items-center justify-around'>**  **<TouchableOpacity**  **onPress={() => navigation.navigate('Transfer History')}>**  **<View className='w-fit flex flex-col items-center gap-3'>**  **<Card>**  **<Card.Content>**  **<Icon source='star' size={24} />**  **</Card.Content>**  **</Card>**  **<Text variant='bodyMedium'>Transfer history</Text>**  **</View>**  **</TouchableOpacity>**  **<TouchableOpacity onPress={() => navigation.navigate('Transfer New')}>**  **<View className='w-fit flex flex-col items-center gap-3'>**  **<Card>**  **<Card.Content>**  **<Icon source='plus' size={24} />**  **</Card.Content>**  **</Card>**  **<Text variant='bodyMedium'>New transfer</Text>**  **</View>**  **</TouchableOpacity>**  **</View>**  **<View className='w-full flex flex-col gap-4 '>**  **{accounts.map((acc: any, index: number) => (**  **<Card id={index.toString()} onPress={() => {**  **navigation.navigate('Account Info', { ...acc, label: "Update" })**  **setSubfix(null)**  **}**  **} >**  **<View className='w-full flex flex-row justify-between p-3 items-center'>**  **<View className='flex flex-row gap-3 items-center'>**  **<Avatar.Text**  **className='bg-[#D8D8D8]'**  **label={`${index}`}**  **size={46}></Avatar.Text>**  **<Text variant='bodyLarge'>{acc.name}</Text>**  **</View>**  **<Text variant='bodyLarge'>{acc.total.toLocaleString()} đ</Text>**  **</View>**  **</Card>**  **))}**  **</View>**  **</View>**  **)**  **}** |
| --- |

## **Chức năng tranfer history**

| **import {**  **Card,**  **Icon,**  **IconButton,**  **Paragraph,**  **Switch,**  **Text,**  **} from 'react-native-paper';**  **import { ScrollView, View } from 'react-native';**  **import { useNavigation } from '@react-navigation/native';**  **import TimeTabs from '../../components/CustomDatetimeTabs';**  **import { useContext, useEffect, useState } from 'react';**  **import { directusInstance } from '../../services/directus';**  **import { readItems } from '@directus/sdk';**  **import { GlobalContext } from '../../contexts/context';**  **import dayjs from 'dayjs';**  **const TransferHistory = () => {**  **const navigation = useNavigation();**  **const [transfers, setTransfers] = useState([])**  **const { user } = useContext(GlobalContext)**  **useEffect(() => {**  **(async () => {**  **const transfersData = await directusInstance.request(readItems('transfer\_history', {**  **filter: {**  **user\_id: user?.id**  **},**  **fields: ['\*', "from\_acc.\*", "to\_acc.\*"]**  **}))**  **console.log(transfersData)**  **const groupedData = {};**  **// Lặp qua từng mục trong dữ liệu và nhóm chúng theo tháng**  **transfersData.forEach(item => {**  **const date = new Date(item.date);**  **const month = date.getMonth() + 1; // Tháng bắt đầu từ 0 nên cần cộng thêm 1**  **const year = date.getFullYear();**  **const key = `${year}-${month}`;**  **// Nếu chưa có mảng cho tháng đó, tạo mới**  **if (!groupedData[key]) {**  **groupedData[key] = [];**  **}**  **// Thêm mục vào mảng của tháng đó**  **groupedData[key].push(item);**  **});**  **setTransfers(groupedData)**  **})()**  **}, [])**  **return (**  **<ScrollView>**  **<View className="flex w-full p-1">**  **<TimeTabs />**  **<View className="flex flex-col w-full gap-4">**  **{Object.keys(transfers).map((key) =>**  **<View className="w-full p-2">**  **<Text className="py-2 text-base font-bold">{dayjs(key).format('MMMM YYYY')}</Text>**  **<View className="flex gap-4">**  **{**  **transfers[key].map((i) =>**  **<Card>**  **<Card.Title**  **title={i.from\_acc.name}**  **subtitle={`Transfer to ${i.to\_acc.name}`}**  **left={props => (**  **<View className="flex items-center justify-center p-1 rounded-md bg-zinc-200">**  **<Icon size={32} source="home-outline" />**  **</View>**  **)}**  **right={props => (**  **<Text variant="bodyLarge" className="mr-4 text-green-700">**  **{i.amount.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ".")}VND**  **</Text>**  **)}**  **/>**  **</Card>**  **)**  **}**  **</View>**  **</View>**  **)}**  **</View>**  **</View>**  **</ScrollView>**  **);**  **};**  **export default TransferHistory;** |
| --- |

| **import {**  **Button,**  **Card,**  **SegmentedButtons,**  **Switch,**  **Text,**  **TextInput,**  **} from 'react-native-paper';**  **import { ScrollView, View } from 'react-native';**  **import { CategoryList, TransactionType } from '../../services/const';**  **import { useContext, useEffect, useMemo, useState } from 'react';**  **import { DatePickerInput } from 'react-native-paper-dates';**  **import DropDown from 'react-native-paper-dropdown';**  **import { directusInstance } from '../../services/directus';**  **import { createItem, readItems, updateItem } from '@directus/sdk';**  **import { GlobalContext } from '../../contexts/context';**  **import { useNavigation } from '@react-navigation/native';**  **const CreateTransfer = () => {**  **const [fromAccount, setFromAccount] = useState('');**  **const [toAccount, setToAccount] = useState('');**  **const [showDropDown1, setShowDropDown1] = useState(false);**  **const [showDropDown2, setShowDropDown2] = useState(false);**  **const [description, setDescription] = useState('');**  **const [money, setMoney] = useState('');**  **const [accounts, setAccounts] = useState([])**  **const { user } = useContext(GlobalContext)**  **useEffect(() => {**  **(async () => {**  **const accountsRes = await directusInstance.request(readItems('account', {**  **filter: {**  **user\_id: user?.id**  **}**  **}))**  **setAccounts(accountsRes || [])**  **})()**  **}, [])**  **const navigation = useNavigation()**  **const fromAccounts = useMemo(() => {**  **return accounts.map(i => ({**  **value: i.id,**  **label: i.name**  **}))**  **}, [accounts])**  **const toAccounts = useMemo(() => {**  **return accounts.map(i => ({**  **value: i.id,**  **label: i.name**  **})).filter(i => i.value != fromAccount)**  **}, [accounts, fromAccount])**  **const addTransfer = async () => {**  **const data = {**  **amount: money,**  **from\_acc: {**  **id: fromAccount**  **},**  **to\_acc: {**  **id: toAccount**  **},**  **user\_id: user?.id,**  **description,**  **date: new Date()**  **}**  **const fromAccountData = accounts.find(i => i.id == fromAccount)**  **const toAccountData = accounts.find(i => i.id == toAccount)**  **try {**  **await directusInstance.request(createItem('transfer\_history', data))**  **navigation.goBack()**  **} catch (error) {**  **console.log(error)**  **}**  **}**  **return (**  **<ScrollView className="w-full h-full">**  **<View className="flex w-full h-full gap-2 p-1 top-1">**  **<View className='flex flex-row justify-around items-center'>**  **<TextInput**  **label={'Money'}**  **value={money}**  **onChangeText={e => setMoney(e)}**  **mode={'outlined'}**  **className='flex-1 mr-4 font-bold '**  **/>**  **<Text variant='bodyLarge'>VND</Text>**  **</View>**  **<View>**  **<DropDown**  **label={'Transfer from account'}**  **mode={'outlined'}**  **setValue={setFromAccount}**  **list={fromAccounts}**  **visible={showDropDown1}**  **showDropDown={() => setShowDropDown1(true)}**  **onDismiss={() => setShowDropDown1(false)}**  **value={fromAccount}**  **/>**  **</View>**  **<View>**  **<DropDown**  **label={'Transfer to account'}**  **mode={'outlined'}**  **setValue={setToAccount}**  **list={toAccounts}**  **visible={showDropDown2}**  **showDropDown={() => setShowDropDown2(true)}**  **onDismiss={() => setShowDropDown2(false)}**  **value={toAccount}**  **/>**  **</View>**  **<TextInput**  **label={'Comment'}**  **value={description}**  **onChangeText={e => setDescription(e)}**  **mode={'outlined'}**  **multiline={true}**  **numberOfLines={5}**  **/>**  **<Button**  **icon="plus"**  **mode="contained"**  **disabled={!(money && description && fromAccount !== '' && toAccount !== '')}**  **onPress={addTransfer}>**  **Add**  **</Button>**  **</View>**  **</ScrollView>**  **);**  **};**  **export default CreateTransfer;** |
| --- |

## **Chức năng payment**

| **import {**  **Button,**  **Card,**  **Icon,**  **IconButton,**  **Menu,**  **Paragraph,**  **Switch,**  **Text,**  **} from 'react-native-paper'**  **import { ScrollView, TouchableWithoutFeedback, View } from 'react-native'**  **import { useIsFocused, useNavigation } from '@react-navigation/native'**  **import { useContext, useEffect, useState } from 'react'**  **import { HeaderContext } from '../../contexts/header'**  **import PaymentCard from '../../components/PaymentCard'**  **import { readItems } from '@directus/sdk'**  **import { directusInstance } from '../../services/directus'**  **import { GlobalContext } from '../../contexts/context'**  **import Loading from '../../components/Loading'**  **import {**  **getAccountByName,**  **getAccountsByEmail,**  **} from '../../controllers/account.controller'**  **import { Account } from '../../types/account'**  **import { ACCOUNT\_KEY, PAYMENT\_KEY } from '../../contants/schema-key.constant'**  **import { allAccount } from '../../contants/transaction/empty-account.constant'**  **const RegularPayments = ({ props }: any) => {**  **const { subfix, setSubfix } = useContext(HeaderContext)**  **const navigation = useNavigation()**  **const { account, setAccount, user } = useContext(GlobalContext)**  **const [accounts, setAccounts] = useState<Account[]>([])**  **const gotoAddPayment = () => {**  **navigation.navigate('Add Payment')**  **setSubfix(null)**  **}**  **const [showAccountsDropdown, setShowAccountsDropdown] = useState(false)**  **const [payments, setPayments] = useState([])**  **const isFocused = useIsFocused()**  **const fetchAccounts = async () => {**  **const accounts = (await directusInstance.request(**  **readItems(ACCOUNT\_KEY, {**  **filter: {**  **user\_id: user?.id,**  **},**  **sort: ['-name'],**  **}),**  **)) as any**  **setAccounts([allAccount, ...accounts])**  **}**  **useEffect(() => {**  **;(async () => {**  **try {**  **var res = await directusInstance.request(**  **readItems(PAYMENT\_KEY, {**  **filter: {**  **account\_id:**  **account.id == '-1'**  **? {**  **user\_id: user?.id,**  **}**  **: account.id,**  **},**  **}),**  **)**  **console.log('do')**  **console.log(JSON.stringify(res, null, 2))**  **setPayments(res)**  **} catch (error) {**  **console.log(error)**  **}**  **})()**  **}, [account,props, isFocused])**  **useEffect(() => {**  **user.email && fetchAccounts()**  **}, [])**  **return (**  **<ScrollView className='flex w-full '>**  **<View className='flex-1 px-3 pb-3'>**  **<TouchableWithoutFeedback**  **onPress={() => setShowAccountsDropdown(true)}**  **className='mb-2'>**  **<View className='flex flex-row justify-start items-center'>**  **<View className='flex items-center flex-row gap-2'>**  **<Text className='text-lg'>Account: </Text>**  **<Text className='font-bold text-base'>{account?.name}</Text>**  **</View>**  **<Menu**  **visible={showAccountsDropdown}**  **onDismiss={() => setShowAccountsDropdown(false)}**  **anchor={**  **<Button>**  **<Icon source='menu-down' color={'black'} size={20} />**  **</Button>**  **}>**  **{accounts.map((acc, index) => (**  **<Menu.Item**  **key={index}**  **onPress={() => {**  **setAccount(acc)**  **setShowAccountsDropdown(false)**  **}}**  **title={acc.name}**  **/>**  **))}**  **</Menu>**  **</View>**  **</TouchableWithoutFeedback>**  **</View>**  **<View className='flex flex-col w-full '>**  **{payments.length > 0 ? (**  **payments.map((payment: any) => (**  **<PaymentCard**  **id={payment.id}**  **type={payment.type}**  **name={payment.name}**  **total={payment.total}**  **add\_automation={payment.add\_automation == 'true'}**  **cycle\_day={payment.cycle\_day}**  **/>**  **))**  **) : (**  **<Text className='w-full h-16 justify-around text-center self-center'>**  **You don't have any regular payment yet!**  **</Text>**  **)}**  **</View>**  **</ScrollView>**  **)**  **}**  **export default RegularPayments** |
| --- |

| **import {**  **Button,**  **Card,**  **SegmentedButtons,**  **Switch,**  **Text,**  **TextInput,**  **} from 'react-native-paper';**  **import {ScrollView, View} from 'react-native';**  **import {CategoryList, ReminderFrequencyList, TransactionType} from '../../services/const';**  **import {useContext, useEffect, useState} from 'react';**  **import {DatePickerInput} from 'react-native-paper-dates';**  **import DropDown from 'react-native-paper-dropdown';**  **import { directusInstance } from '../../services/directus';**  **import { readItems } from '@directus/sdk';**  **import { GlobalContext } from '../../contexts/context';**  **import { createPayment, getPayment, updatePayment } from '../../controllers/payment.controller';**  **import { SnackBarContext } from '../../hocs/SnackBar';**  **import { useIsFocused, useNavigation } from '@react-navigation/native';**  **import Loading from '../../components/Loading';**  **const EditPayment = (props:any) => {**  **const [loading, setLoading] = useState(false)**  **const [type, setType] = useState('expenses');**  **const [total, setTotal] = useState('');**  **const [name, setName] = useState('');**  **const [reminder, setReminder] = useState(0);**  **const [fromDate, setFromDate] = useState(new Date());**  **const [toDate, setToDate] = useState(new Date());**  **const {setData} = useContext(SnackBarContext);**  **const [account, setAccount] = useState('');**  **const [accounts, setAccounts] = useState([])**  **const [addAutomation,setAddAutomation] =useState(false)**  **const [showReminderFrequencyDropDown, setShowReminderFrequencyDropDown] = useState(false);**  **const [showAccountDropDown, setShowAccountDropDown] = useState(false);**  **const [showCategoryDropDown, setShowCategoryDropDown] = useState(false);**  **const [description, setDescription] = useState('');**  **const [category,setCategory] = useState("")**  **const {user} =useContext(GlobalContext)**  **const navigation = useNavigation()**  **const isFocused = useIsFocused()**  **const handleSwitchAddAutomation = (value)=>{**  **console.log({addAutomation,value})**  **setAddAutomation(!addAutomation)**  **}**  **const handleEditPayment = ()=>{**  **(async ()=>{**  **const res = await updatePayment(props.route?.params.id, type,name,Number(total),reminder,fromDate,toDate, category,description,addAutomation, account)**  **if (res){**  **setData({text: 'Update regular payment successful!'});**  **navigation.goBack();**  **}**  **})()**  **}**  **useEffect(() => {**  **if (user) {**  **(async () => {**  **const res =**  **(await directusInstance.request(**  **readItems('account', {**  **filter: {**  **user\_id: user?.id**  **},**  **}),**  **)) || [];**  **setAccounts(**  **res.map(i => ({**  **label: i.name,**  **value: i.id,**  **})),**  **);**  **})();**  **}**  **}, [account , isFocused]);**  **useEffect(()=>{**  **setLoading(true)**  **;(async ()=>{**  **const payment = await getPayment(props.route?.params.id)**  **if (payment) {**  **setName(payment.name)**  **setTotal(payment.total.toString())**  **setCategory(payment.category)**  **setAccount(payment.account\_id)**  **setReminder(payment.cycle\_day)**  **setAddAutomation(payment.add\_automation == 'true')**  **setType(payment.type)**  **setFromDate(new Date(payment.from))**  **setToDate(new Date(payment.to))**  **setDescription(payment.description)**  **}else{**  **setData({text: "Payment not exist!. Please try again"});**  **navigation.goBack()**  **}**  **setLoading(false)**  **})()**  **},[])**  **return loading ? <Loading/> : (**  **<ScrollView className="w-full h-full p-3">**  **<View className="flex w-full h-full gap-2 ">**  **<View className="flex-row items-center justify-center p-5">**  **<TextInput**  **mode="flat"**  **value={total}**  **onChangeText={e => setTotal(e)}**  **className="w-48 h-12 text-lg bg-transparent border-b"**  **/>**  **<Text className="ml-2">VND</Text>**  **</View>**  **<SegmentedButtons**  **value={type}**  **onValueChange={setType}**  **buttons={TransactionType}**  **/>**  **<TextInput**  **className="h-14"**  **label={'Name'}**  **value={name}**  **onChangeText={e => setName(e)}**  **mode={'outlined'}**  **multiline={true}**  **numberOfLines={5}**  **/>**  **<View>**  **<DropDown**  **label={'Reminder frequency'}**  **mode={'outlined'}**  **setValue={setReminder}**  **list={ReminderFrequencyList}**  **visible={showReminderFrequencyDropDown}**  **showDropDown={() => setShowReminderFrequencyDropDown(true)}**  **onDismiss={() => setShowReminderFrequencyDropDown(false)}**  **value={reminder}**  **/>**  **</View>**  **<DatePickerInput**  **locale="en"**  **label="From"**  **mode={'outlined'}**  **value={fromDate}**  **onChange={(d: any) => setFromDate(d)}**  **inputMode="start"**  **/>**  **<DatePickerInput**  **locale="en"**  **label="To"**  **mode={'outlined'}**  **value={toDate}**  **onChange={(d: any) => setToDate(d)}**  **inputMode="start"**  **/>**  **<View>**  **<DropDown**  **label={'Account'}**  **mode={'outlined'}**  **setValue={setAccount}**  **list={accounts}**  **visible={showAccountDropDown}**  **showDropDown={() => setShowAccountDropDown(true)}**  **onDismiss={() => setShowAccountDropDown(false)}**  **value={account}**  **/>**  **</View>**  **<View>**  **<DropDown**  **label={'Category'}**  **mode={'outlined'}**  **setValue={setCategory}**  **list={CategoryList}**  **visible={showCategoryDropDown}**  **showDropDown={() => setShowCategoryDropDown(true)}**  **onDismiss={() => setShowCategoryDropDown(false)}**  **value={category}**  **/>**  **</View>**  **<TextInput**  **label={'Description'}**  **value={description}**  **onChangeText={e => setDescription(e)}**  **mode={'outlined'}**  **multiline={true}**  **numberOfLines={5}**  **/>**  **<View className="flex flex-row items-center justify-between">**  **<Text variant="bodyMedium"> Add Automation {addAutomation}</Text>**    **<Switch value={addAutomation} onValueChange={handleSwitchAddAutomation}></Switch>**  **</View>**  **<Button**  **mode="contained"**  **onPress={handleEditPayment}>**  **Edit**  **</Button>**  **</View>**  **</ScrollView>**  **);**  **};**  **export default EditPayment;** |
| --- |

| **import {**  **Button,**  **Card,**  **SegmentedButtons,**  **Switch,**  **Text,**  **TextInput,**  **} from 'react-native-paper';**  **import {ScrollView, View} from 'react-native';**  **import {CategoryList, ReminderFrequencyList, TransactionType} from '../../services/const';**  **import {useContext, useEffect, useState} from 'react';**  **import {DatePickerInput} from 'react-native-paper-dates';**  **import DropDown from 'react-native-paper-dropdown';**  **import { directusInstance } from '../../services/directus';**  **import { readItems } from '@directus/sdk';**  **import { GlobalContext } from '../../contexts/context';**  **import { createPayment } from '../../controllers/payment.controller';**  **import { SnackBarContext } from '../../hocs/SnackBar';**  **import { useIsFocused, useNavigation } from '@react-navigation/native';**  **const AddPayment = () => {**  **const [type, setType] = useState('expenses');**  **const [total, setTotal] = useState('');**  **const [name, setName] = useState('');**  **const [reminder, setReminder] = useState(0);**  **const [fromDate, setFromDate] = useState(new Date());**  **const [toDate, setToDate] = useState(new Date());**  **const {setData} = useContext(SnackBarContext);**  **const [account, setAccount] = useState('');**  **const [accounts, setAccounts] = useState([])**  **const [addAutomation,setAddAutomation] =useState(false)**  **const [showReminderFrequencyDropDown, setShowReminderFrequencyDropDown] = useState(false);**  **const [showAccountDropDown, setShowAccountDropDown] = useState(false);**  **const [showCategoryDropDown, setShowCategoryDropDown] = useState(false);**  **const [description, setDescription] = useState('');**  **const [category,setCategory] = useState("")**  **const {user} =useContext(GlobalContext)**  **const navigation = useNavigation()**  **const isFocused = useIsFocused()**  **const handleCreatePayment = ()=>{**  **(async ()=>{**  **const res = await createPayment(type,name,Number(total),reminder,fromDate,toDate, category,description,addAutomation, account)**  **if (res){**  **setData({text: 'Create regular payment successful!'});**  **navigation.goBack();**  **}**  **})()**  **}**  **useEffect(() => {**  **if (user) {**  **(async () => {**  **const res =**  **(await directusInstance.request(**  **readItems('account', {**  **filter: {**  **user\_id: user?.id**  **},**  **}),**  **)) || [];**  **setAccounts(**  **res.map(i => ({**  **label: i.name,**  **value: i.id,**  **})),**  **);**  **})();**  **}**  **}, [account , isFocused]);**  **return (**  **<ScrollView className="w-full h-full p-3">**  **<View className="flex w-full h-full gap-2 p-1 top-1">**  **<View className="flex-row items-center justify-center p-5">**  **<TextInput**  **mode="flat"**  **value={total}**  **onChangeText={e => setTotal(e)}**  **className="w-48 h-12 text-lg bg-transparent border-b"**  **/>**  **<Text className="ml-2">VND</Text>**  **</View>**  **<SegmentedButtons**  **value={type}**  **onValueChange={setType}**  **buttons={TransactionType}**  **/>**  **<TextInput**  **className="h-14"**  **label={'Name'}**  **value={name}**  **onChangeText={e => setName(e)}**  **mode={'outlined'}**  **multiline={true}**  **numberOfLines={5}**  **/>**  **<View>**  **<DropDown**  **label={'Reminder frequency'}**  **mode={'outlined'}**  **setValue={setReminder}**  **list={ReminderFrequencyList}**  **visible={showReminderFrequencyDropDown}**  **showDropDown={() => setShowReminderFrequencyDropDown(true)}**  **onDismiss={() => setShowReminderFrequencyDropDown(false)}**  **value={reminder}**  **/>**  **</View>**  **<DatePickerInput**  **locale="en"**  **label="From"**  **mode={'outlined'}**  **value={fromDate}**  **onChange={(d: any) => setFromDate(d)}**  **inputMode="start"**  **/>**  **<DatePickerInput**  **locale="en"**  **label="To"**  **mode={'outlined'}**  **value={toDate}**  **onChange={(d: any) => setToDate(d)}**  **inputMode="start"**  **/>**  **<View>**  **<DropDown**  **label={'Account'}**  **mode={'outlined'}**  **setValue={setAccount}**  **list={accounts}**  **visible={showAccountDropDown}**  **showDropDown={() => setShowAccountDropDown(true)}**  **onDismiss={() => setShowAccountDropDown(false)}**  **value={account}**  **/>**  **</View>**  **<View>**  **<DropDown**  **label={'Category'}**  **mode={'outlined'}**  **setValue={setCategory}**  **list={CategoryList}**  **visible={showCategoryDropDown}**  **showDropDown={() => setShowCategoryDropDown(true)}**  **onDismiss={() => setShowCategoryDropDown(false)}**  **value={category}**  **/>**  **</View>**  **<TextInput**  **label={'Description'}**  **value={description}**  **onChangeText={e => setDescription(e)}**  **mode={'outlined'}**  **multiline={true}**  **numberOfLines={5}**  **/>**  **<View className="flex flex-row items-center justify-between">**  **<Text variant="bodyMedium"> Add Automation</Text>**  **<Switch value={addAutomation} onValueChange={setAddAutomation}></Switch>**  **</View>**  **<Button**  **icon="plus"**  **mode="contained"**  **onPress={handleCreatePayment}>**  **Add**  **</Button>**  **</View>**  **</ScrollView>**  **);**  **};**  **export default AddPayment;** |
| --- |

## **Chức năng notification**

| **import { useContext, useEffect, useState } from 'react'**  **import { RefreshControl, ScrollView, View } from 'react-native'**  **import { Badge, Button, Card, Text } from 'react-native-paper'**  **import { readItems, updateItem } from '@directus/sdk'**  **import { GlobalContext } from '../../contexts/context'**  **import { directusInstance } from '../../services/directus'**  **import { calculateTimeAgo } from '../../utils/number'**  **import { NOTIFICATION\_KEY } from '../../contants/schema-key.constant'**  **export default function Notification() {**  **const [notifications, setNotifications] = useState([])**  **const [refreshing, setRefreshing] = useState(false)**  **const { user } = useContext(GlobalContext)**    **useEffect(() => {**  **; (async () => {**  **try {**  **const res = await directusInstance.request(**  **readItems(NOTIFICATION\_KEY, {**  **sort: ['-date\_created'],**  **filter: {**  **user\_id: user?.id**  **},**  **}),**  **)**  **setNotifications(res)**  **await Promise.all(res.map(async (noti: any) => {**  **noti.is\_read == false && await directusInstance.request(**  **updateItem(NOTIFICATION\_KEY, noti.id, {**  **is\_read: true**  **}),**  **)**  **}))**  **setRefreshing(false)**  **} catch (error) {**  **// console.log(error)**  **}**  **})()**  **}, [refreshing])**  **return (**  **<ScrollView className='h-full w-full' refreshControl={<RefreshControl refreshing={refreshing} onRefresh={() => setRefreshing(true)} />} >**  **<View className='w-full pt-2 gap-y-4 items-center pb-4'>**  **{notifications.map((noti: any, index: number) => (**  **<Card id={index.toString()} className='p-4 w-[90%]' key={noti.id} >**  **<View className='w-full'>**  **<Text variant='titleMedium'>{noti.title}</Text>**  **<Text variant='bodyLarge'>{noti.message}</Text>**  **{noti.is\_read ? <View className='flex flex-row justify-between items-center '>**  **<Text className=' text-right' variant='bodyMedium'>{calculateTimeAgo(new Date(noti.date\_created))}</Text>**  **<Button className='text' icon="chevron-down-circle">Seen</Button>**  **</View> :**  **<Text className=' text-right' variant='bodyMedium'>{calculateTimeAgo(new Date(noti.date\_created))}</Text>**  **}**  **</View>**  **</Card>**  **))}**  **</View>**  **</ScrollView>**  **)**  **}** |
| --- |

## **Chức năng object hook**

| **import { defineHook } from '@directus/extensions-sdk';**  **import axios from 'axios';**  **export default defineHook(({ filter, action, schedule }, { getSchema, services, database }) => {**  **const { ItemsService } = services;**  **filter('trasaction.items.update', async (payload: any, meta, context) => {**  **const trasactionService = new ItemsService('trasaction', {**  **schema: await getSchema(),**  **accountability: context.accountability**  **});**  **const data = await trasactionService.readOne(meta.keys[0]);**  **return {**  **...payload,**  **oldTotal: data.total,**  **oldType: data.type**  **}**  **});**  **action('trasaction.items.update', async (handler, context) => {**  **const trasactionService = new ItemsService('trasaction', {**  **schema: await getSchema(),**  **accountability: context.accountability**  **});**  **const accountService = new ItemsService('account', {**  **schema: await getSchema(),**  **accountability: context.accountability**  **});**  **const transaction = await trasactionService.readOne(handler.keys[0]);**  **if (transaction.total != handler.payload.oldTotal || transaction.type != handler.payload.oldType) {**  **const account = await accountService.readOne(transaction.account\_id);**  **const newMoney = (transaction.type == "expenses") ? -1 \* transaction.total : 1 \* transaction.total**  **const oldMoney = (handler.payload.oldType == "expenses") ? -1 \* handler.payload.oldTotal : 1 \* handler.payload.oldTotal**  **account.total = account.total - oldMoney + newMoney;**  **await accountService.updateOne(transaction.account\_id, account);**  **}**  **});**  **action('trasaction.items.create', async (handler, context) => {**  **const accountService = new ItemsService('account', {**  **schema: await getSchema(),**  **accountability: context.accountability**  **});**  **console.log({ handler })**  **const account = await accountService.readOne(handler.payload.account\_id.id || handler.payload.account\_id);**  **const total = (handler.payload.type == "expenses") ? -1 \* handler.payload.total : 1 \* handler.payload.total**  **account.total = account.total + total**  **await accountService.updateOne(account.id, account);**  **});**  **filter('trasaction.items.delete', async (payload: any, meta, context) => {**  **const accountService = new ItemsService('account', {**  **schema: await getSchema(),**  **accountability: context.accountability**  **});**  **const trasactionService = new ItemsService('trasaction', {**  **schema: await getSchema(),**  **accountability: context.accountability**  **});**  **const transaction = await trasactionService.readOne(payload[0]);**  **const account = await accountService.readOne(transaction.account\_id);**  **const total = (transaction.type == "expenses") ? -1 \* transaction.total : 1 \* transaction.total**  **account.total = account.total - total**  **await accountService.updateOne(account.id, account);**  **});**  **action('transfer\_history.items.create', async (handler, context) => {**  **const accountService = new ItemsService('account', {**  **schema: await getSchema(),**  **accountability: context.accountability**  **});**  **const trasactionService = new ItemsService('trasaction', {**  **schema: await getSchema(),**  **accountability: context.accountability**  **});**  **const fromAccount = await accountService.readOne(handler.payload.from\_acc.id);**  **const toAccount = await accountService.readOne(handler.payload.to\_acc.id);**  **await trasactionService.createOne({**  **name: `[Transfer] Send to ${toAccount.name}`,**  **total: handler.payload.amount,**  **trading\_date: handler.payload.date,**  **account\_id: handler.payload.from\_acc.id,**  **type: 'expenses',**  **category: 'Chuyển tiền',**  **description: `Transfer from ${fromAccount.name} to ${toAccount.name} `,**  **});**  **await trasactionService.createOne({**  **name: `[Transfer] Receive from ${fromAccount.name} `,**  **total: handler.payload.amount,**  **trading\_date: handler.payload.date,**  **category: 'Chuyển tiền',**  **type: 'income',**  **account\_id: handler.payload.to\_acc.id,**  **description: `Transfer from ${fromAccount.name} to ${toAccount.name} `,**  **});**  **});**  **schedule('\* \* 24 \* \* \*', async () => {**  **await axios.post('http://0.0.0.0:8055/transaction-operation/payment\_cronjob', {**  **headers: {**  **Authorization: 'Bearer 5P8aI2ZZN4xnF2i5weQeIk28tR33\_DQD'**  **}**  **});**  **})**  **});** |
| --- |

## **Chức năng cron job**

| **import { defineEndpoint } from '@directus/extensions-sdk';**  **export default defineEndpoint((router, context) => {**  **const { services, getSchema } = context;**  **const { ItemsService } = services;**  **router.post('/payment\_cronjob', async (req, res) => {**  **console.log(req.params);**  **const transactionsService = new ItemsService('trasaction', {**  **schema: await getSchema(),**  **accountability: req?.accountability**  **});**  **const notiService = new ItemsService('notification', {**  **schema: await getSchema(),**  **accountability: req?.accountability**  **});**  **const paymentService = new ItemsService('payment', {**  **schema: await getSchema(),**  **accountability: req?.accountability**  **});**  **const payments : [] = await paymentService.readByQuery({**  **fields: ['\*', "account\_id.\*"],**  **filter: {**  **to: {**  **\_gte: new Date()**  **},**  **from: {**  **\_lte: new Date()**  **}**  **}**  **})**  **await Promise.all(payments.map( async (payment:any)=>{**  **var lastDate = new Date(payment.last\_payment\_date || payment.from)**  **console.log(payment.account\_id.user\_id)**  **if (diffDays(lastDate, new Date()) >= payment.cycle\_day) {**  **await notiService.createOne({**  **title: `Regular payment for ${payment.name}`,**  **message : `Your regular payment cycle (${payment.cycle\_day}) ${payment.name} is comming`,**  **user\_id : payment.account\_id.user\_id**  **});**  **await paymentService.updateOne(payment.id, {**  **last\_payment\_date : new Date()**  **});**    **if (payment.add\_automation){**  **await transactionsService.createOne({**  **name: `[Regular payment] ${payment.name}`,**  **total: payment.total ,**  **trading\_date: new Date(),**  **account\_id: payment.account\_id ,**  **type : payment.type ,**  **category: payment.category,**  **description : `Automation transaction for regular payment ${payment.name}`,**  **});**  **}**  **}**  **}))**  **res.send({ payments: payments })**  **})**  **});**  **function diffDays(date1: Date, date2: Date): number {**  **const time1 = date1.getTime();**  **const time2 = date2.getTime();**  **const diffTime = Math.abs(time2 - time1);**  **const diffDays = Math.ceil(diffTime / (1000 \* 60 \* 60 \* 24));**  **return diffDays;**  **}** |
| --- |

# **KẾT LUẬN**

Dự án xây dựng ứng dụng quản lý chi tiêu cá nhân đã mang lại những giải pháp hiệu quả cho việc quản lý tài chính cá nhân. Bằng cách cung cấp một giao diện thân thiện và các tính năng linh hoạt, dự án đã tạo điều kiện thuận lợi cho người dùng trong việc theo dõi và kiểm soát các khoản chi tiêu hàng ngày. Việc này giúp họ đào sâu vào mối quan hệ giữa thu nhập và chi tiêu, từ đó xây dựng kế hoạch tài chính cá nhân một cách có ý thức hơn. Dự án đã góp phần nâng cao nhận thức về quản lý tài chính cá nhân và tạo ra một cộng đồng người dùng thông minh về tiền bạc.