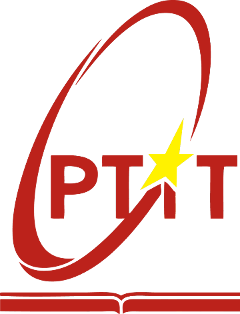
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----



**BÁO CÁO KẾT THÚC MÔN HỌC**

***Phát triển ứng dụng cho các thiết bị di dộng***

***ĐỀ TÀI***

**XÂY DỰNG QUẢN LÝ TÀI CHÍNH**

**FINANCIAL MANAGER**

*Giáo viên hướng dẫn :* **Th. S Trương Bá Thái**

Nhóm sinh viên thực hiện : *Nhóm 11*

|  |  |
| --- | --- |
| **Nguyễn Nhật Trường** | **N14DCCN287** |
| **Nguyễn Hoàng Sang** | **N14DCCN310** |
| **Đào Công Danh** | **N14DCCN219** |
| **Phạm Thanh Sơn** | **N14DCCN311** |
| **Lê Huy Hoàng** | **N14DCCN176** |
| **Đoàn Anh Tuấn** | **N14DCCN245** |
|  |  |

*Hồ Chí Minh, ngày 14 tháng 5 năm 2018*

**MỤC LỤC**

[CHƯƠNG I : MỞ ĐẦU 1](#_Toc514443313)

[1. Giới thiệu môn học và nhóm thực hiện 1](#_Toc514443314)

[2. Mô tả ứng dụng 1](#_Toc514443315)

[3. Ứng dụng trong cuộc sống 1](#_Toc514443316)

[CHƯƠNG II : PHÂN TÍCH CẤU TRÚC HỆ THỐNG VÀ PHÂN CHIA CÔNG VIỆC 3](#_Toc514443317)

[1. Phân tích cấu trúc hệ thống 3](#_Toc514443318)

[a. Màn hình Splash 3](#_Toc514443319)

[b. Màn hình Login 4](#_Toc514443320)

[c. Màn hình SignUp 6](#_Toc514443321)

[d. Màn hình Home 8](#_Toc514443322)

[e. Màn hình Detail 10](#_Toc514443323)

[f. Màn hình Create 12](#_Toc514443324)

[g. Màn hình Profile 14](#_Toc514443325)

[h. Màn hình Chart 16](#_Toc514443326)

[2. Phân chia công việc 17](#_Toc514443327)

[CHƯƠNG III : KẾT QUẢ THU ĐƯỢC 18](#_Toc514443328)

[1. Kết quả đạt được 18](#_Toc514443329)

[2. Kết luận 18](#_Toc514443330)

**PHÂN CHIA HOẠT ĐỘNG NHÓM 11**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | CÔNG VIỆC | TỰ ĐÁNH GIÁ | NHÓM ĐÁNH GIÁ | KÝ TÊN |
| 1 | **Nguyễn Nhật Trường** | * Xây dựng Base theo mô hình MVP với thư viện Dagger * Import và sử dụng các thư viện của Adroid * Custom view, custom Dialog * Màn hình Create, Chart * Push Notification thông qua Firebase và công việc tương lai | 20% | 20% |  |
| 2 | **Nguyễn Hoàng Sang** | * Lấy API, đọc, load JSON * Làm màn hình Home * Báo cáo cuối kỳ * Tạo navigation bar | 16% | 16% |  |
| 3 | **Đào Công Danh** | * Lấy API, đọc, load JSON * Làm màn hình Splash, Setting * Báo cáo cuối kỳ | 16% | 16% |  |
| 4 | **Phạm Thanh Sơn** | * Lấy API, đọc, load JSON * Làm màn hình Profile * Báo cáo cuối kỳ | 16% | 16% |  |
| 5 | **Lê Huy Hoàng** | * Lấy API, đọc, load JSON * Làm màn hình Login, SignUp * Viết báo cáo * Báo cáo cuối kỳ | 16% | 16% |  |
| 6 | **Đoàn Anh Tuấn** | * Lấy API, đọc, load JSON * Làm màn hình Detail, Filter Detail theo Date * Báo cáo cuối kỳ | 16% | 16% |  |

CHƯƠNG I : MỞ ĐẦU

1. Giới thiệu môn học và nhóm thực hiện

Môn lập trình di động thiết kế ứng dụng trên nền tảng Android là môn học giúp sinh viên phát triển kỹ năng lập trình, tự tìm kiếm, cập nhật kiến thức mới giúp áp dụng cho các dự án thực tế.

Kiến thức mới: API, JSON, Splash screen, Navigation drawer, Chart, tìm hiểu và sử dụng thư viện Dagger 2, Retrofit 2.0, Glide, MPAndroidChart, CircleImageView, kết nối Firebase, lưu trữ SQLite, Upload File, …

Thành viên thực hiện:

* Nguyễn Nhật Trường
* Nguyễn Hoàng Sang (Leader)
* Phạm Thanh Sơn
* Lê Huy Hoàng
* Đoàn Anh Tuấn
* Đào Công Danh

1. Mô tả ứng dụng

* Tên ứng dụng : Financial Manager
* Ứng dụng cần kết nối mạng : Có
* Phiên bản thích hợp : Android 4+
* Server API sử dụng: <http://sixfinancialbox.azurewebsites.net/>
* Chức năng : Quản lý tài chính theo các khoản mục, lưu lịch sử lại theo từng ngày, tháng, năm, …

1. Ứng dụng trong cuộc sống

Với công việc bộn bề và cuộc sống tấp nập, nhộn nhịp hiện nay thì việc quản lý ngân sách của cá nhân hay gia đình thường mang lại nhiều khó khăn và bất cập vì vấn đề thời gian của mọi người đều hạn hẹp.

Ứng dụng này hỗ trợ được việc cập nhật danh số tài chính hiện tại của mình một cách nhanh nhất và đồng bộ real time trên Firebase. Ứng dụng còn xem được biểu đồ Chart trực quan về số liệu của mình hàng tháng.

CHƯƠNG II : PHÂN TÍCH CẤU TRÚC HỆ THỐNG VÀ PHÂN CHIA CÔNG VIỆC

1. Phân tích cấu trúc hệ thống

Tổng quan: Ứng dụng quản lý tài chính chi tiêu cá nhân

Phạm vi đề tài:

* Lập trình trên Android Studio v3.1.2
* Hệ điều hành Android 4.0+
* Service API : <http://sixfinancialbox.azurewebsites.net/>
* Thiết bị: Virtual Device và Real Device

1. Màn hình Splash

* Giao diện



* Thiết kế màn hình

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu đề** | **Mô tả** | **Quy trình** | **Kết quả** |
| Màn hình Splash | ProgessBar hiển thị loading.  Button Logout và Retry hiện thị khi load không thành công | Hiển thị thanh loading | Hoàn thành |
| Button Logout | Chuyển sang màn hình SignIn | Click | Hoàn thành |
| Button Retry | Thử load lại vào màn hình Home | Click | Hoàn thành |

* Luồng xử lý

Ban đầu App chạy vào màn hình Splash. Kiểm tra xem trước đó có đăng nhập chưa thông qua token được lưu ở SQLite:

MainApplication.*getAppComponent*().getSQLiteManager().isLogin();

Nếu token khác null thì chạy vào màn hình Setting.

Ở màn hình Setting goi API getUser :

getPresenter().getUserInfo();

API getUser cần có header là token, truyền token đã đuược lưu ở SQLite vào API getUser. Nếu thành công lưu thông tin User xuống SQLite:

getSQLiteManager().saveUser(**new** Gson().toJson(res.**result**));

rồi chạy vào màn hình Home. Nếu gọi API getUser không thành công thì hiển thị hai button là Logout và Retry. Nếu nhấn button Retry thì gọi lại API getUser. Nếu nhấn Logout thì set lại cho giá trị ở hai cột token và user ở SQLite bằng null:

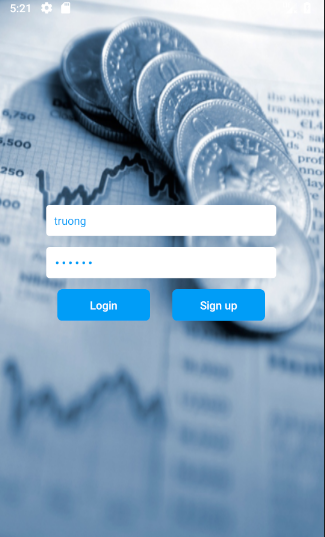
getSQLiteManager().resetUser();

và sau đó chạy vào màn hình Login.

Trường hợp nếu chưa login trước đó, token == null, chạy vào màn hình Login.

1. Màn hình Login

* Giao diện



* Thiết kế màn hình

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu đề** | **Mô tả** | **Quy trình** | **Kết quả** |
| Màn hình Login | Có textfiled nhập user và password  Button Login và SignUp | Thiết kế UI | Hoàn thành |
| Button Login | Chuyển sang màn hình Splash | Click | Hoàn thành |
| Button SignUp | Chuyển sang màn hình SignUp | Click | Hoàn thành |

* Luồng xử lý

Ở màn hình Login có 2 button Login và SignUp.

Khi click vào button Login thì app sẽ lấy user và password được nhập ở textfield.

getPresenter().login(edtUsername.getText().toString().trim(), edtPassword.getText().toString().trim());

rồi gọi API login với body là user và password vừa lấy trên. Nếu thành công server trả về token và thông tin user và được lưu xuống SQLite:

getSQLiteManager().saveToken(res.result.getToken());  
getSQLiteManager().saveUser(new Gson().toJson(res.result));

Sau đó chuyển sang màn hình Setting xử lý và chạy vào màn hình Home.

getView().onSuccess();

public void onSuccess() {  
 Intent intentHome = new Intent(LoginActivity.this, SettingActivity.class);  
 startActivity(intentHome);  
 finish();  
}

Nếu không thành công sẽ hiện dialog báo lỗi.

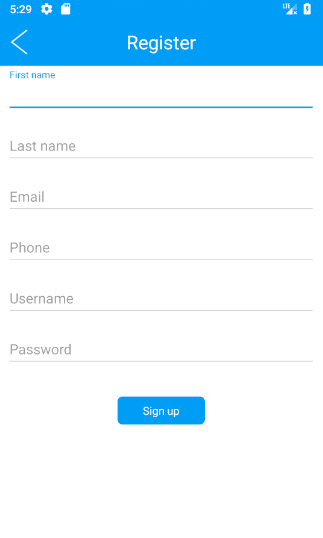
public void failure(RestError error) {  
 getView().hideLoading();  
 getView().onFailure(error.toString());  
}

Khi click vào button SignUp thì chuyển sang màn hình SignUp

Intent intent = new Intent(this, SignUpActivity.class);  
startActivityForResult(intent, *REQUEST\_CODE\_SIGN\_UP*);

1. Màn hình SignUp

* Giao diện



* Thiết kế màn hình

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu đề** | **Mô tả** | **Quy trình** | **Kết quả** |
| Màn hình SignUp | Các textfiled nhập thông tin người dùng  Button SignUp | Thiết kế UI | Hoàn thành |
| Button SignUp | Đăng ký người dùng | Click, call API register | Hoàn thành |

* Luồng xử lý

Ở màn hình này có các textfield nhập thông tin đăng ký và button SignUp. Khi click button SignUp thì sẽ kiểm tra xem thông tin đăng ký ở các textfield có bị bỏ trống không.

private boolean missingInfo() {  
 return isTextEmpty(edtFirstName)  
 && isTextEmpty(edtLastName)  
 && isTextEmpty(edtEmail)  
 && isTextEmpty(edtPhone)  
 && isTextEmpty(edtUsername)  
 && isTextEmpty(edtPassword);  
}

Nếu đang thiếu thì hiện thông báo cho người dùng điền đầy đủ.

if (missingInfo()) {  
 showOkDialog("", "Please, fill full information!", null);  
}

Nếu đã điền đủ thì lấy thông tin từ các field và call API signUp với body là các thông tin đã lấy.

getPresenter().signUp(getInfoSignUp());

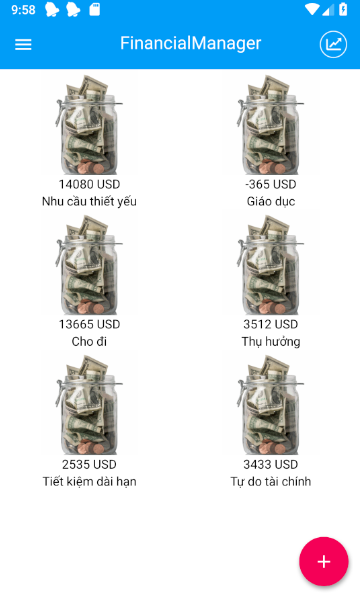
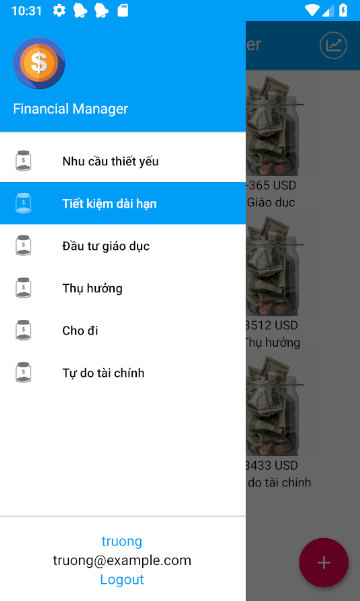
Nếu server trả về thành công thì hiện dialog và chuyển về màn hình Login, đồng thời set các thông tin vừa đăng ký vào textfield ở màn hình Login

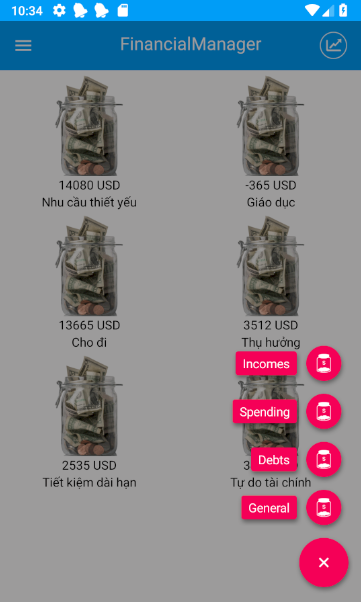
public void onSuccess(final User user) {  
 showOkDialog(getString(R.string.*sign\_up*), "Success!", new DialogOk.IOkDialogListener() {  
 @Override  
 public void onIOkDialogAnswerOk(DialogOk dialog) {  
 dialog.dismiss();  
 Intent intent = new Intent(getApplicationContext(), LoginActivity.class);  
 intent.putExtra(*RESULT\_SIGN\_UP\_DATA*, user);  
 setResult(*RESULT\_CODE\_SIGN\_UP*, intent);  
 finish();  
 }  
 });  
}

Nếu không thành công thì hiện dialog thông báo lỗi.

1. Màn hình Home

* Giao diện



* Thiết kế màn hình

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu đề** | **Mô tả** | **Quy trình** | **Kết quả** |
| Màn hình Home | NavigationView chuyển sang các view  RecyclerView chứa item  FloatingButtonAction thêm chi tiêu  DrawerLayout | Thiết kế UI | Hoàn thành |
| Floating Button Action | Chuyển sang màn hình Create Debts, Spending, Incomes, General | Click | Hoàn thành |
| Recycler  View | Chứa các item. Click vào các item để chuyển sang màn hình Detail | Click | Hoàn thành |
| Navigation  Drawer | Chứa các item của 6 màn hình. Click vào các item chuyển sang các màn hình đó.  Button Logout click để thoát.  Button Profile click chuyển sang màn hình Profile | Click | Hoàn thành |
| Navigation  Button | Chuyển sang màn hình Chart | Click | Hoàn thành |

* Luồng xử lý

Khi chạy vào màn hình Home, app call API getJars với header là userId lấy từ SQLite

User user = getSQLiteManager().getUser();

Nếu server trả về thành công thì sẽ đổ dữ liệu ra List jarList

getJarList().clear();  
jarList.addAll(res.result);

Nếu thất bại thì báo lỗi.

Nhân vào hình ở góc trên bên trái sẽ hiện ra NavigationView chưa list Jar, user name và text Logout. Nế nhấn vào item của list Jar thì sẽ chuyển sang màn hình Detail. Nhấn vào username sẽ chuyển sang màn hình Profile.

Nhấn Logout thì sẽ log out và chuyển sang màn hình Login.

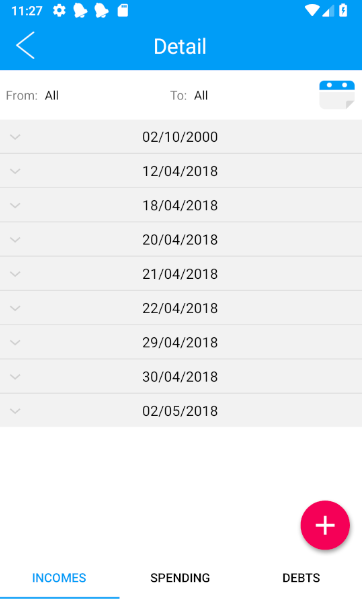
Nhấn vào biểu tưởng chart ở góc trên bên phải sẽ chuyển sang màn hình Chart.

Nhấn vào item của list Jar sẽ chuyển sang màn hình Detail.

Nhấn vào button dấu cộng dưới cùng bên phải sẽ hiện ra các button để chọn tạo các dữ liệu như: Income, Spending, Debt, General. Khi chọn tạo dữ liệu sẽ chuyển sang màn hình Create.

1. Màn hình Detail

* Giao diện



* Thiết kế màn hình

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu đề** | **Mô tả** | **Quy trình** | **Kết quả** |
| Màn hình Detail | Custom View filter Detail theo Date  Floating Button click chuyển sang màn hình Create  ExpandableListView hiển thị chi tiết chi tiêu theo ngày | Thiết kế UI | Hoàn thành |
| Button  Floating | Chuyển sang màn hình Create | Click, get API detail | Hoàn thành |
| Custom View Filter | Hiển thị Date view để chọn thời gian Filter | Click | Hoàn thành |

* Luồng xử lí

Màn hình này sẽ có 3 tab là Income, Spending và Debt. Mỗi tab sẽ hiển thì danh sách các dữ liệu được nhóm theo ngày, tháng năm. Tại màn hình này cho phép người dùng filter theo ngày, tháng, năm bằng việc nhấn vào biểu tưởng lịch ở phía trên bên phải màn hình. Khi nhấn vào biểu tượng lịch, sẽ hiện ra 1 dialog để chọn ngày, tháng, năm. Nhấn Done sẽ tắt dialog và gọi API lấy dữ liệu của tab hiện tại đang đứng theo ngày, tháng, năm.

**switch** (**vpDetail**.getCurrentItem()){  
 **case *TYPE\_SPENDING***:  
 ((SpendingFragment) **mFragments**.get(***TYPE\_SPENDING***)).filterSpending(dateFrom, dateTo);  
 **break**;  
 **case *TYPE\_INCOME***:  
 ((IncomeFragment) **mFragments**.get(***TYPE\_INCOME***)).filterIncome(dateFrom, dateTo);  
 **break**;  
 **case *TYPE\_DEBT***:  
 ((DebtFragment) **mFragments**.get(***TYPE\_DEBT***)).filterDebt(dateFrom, dateTo);  
 **break**;  
}

Tại tab spending, sẽ cho xóa khi nhấn vào nút xóa của item. Nhấn vào nút xóa sẽ hiện dialog xác nhận xóa hay không? Nếu xác nhận xóa sẽ gọi API xóa item spending

getPresenter().deleteSpending(positionGroup, positionChild);

Tại tab debt, sẽ cho xóa và chỉnh sửa item. Nếu nhấn vào nút xóa sẽ hiện dialog xác nhận xóa hay không gọi API xóa item debt.

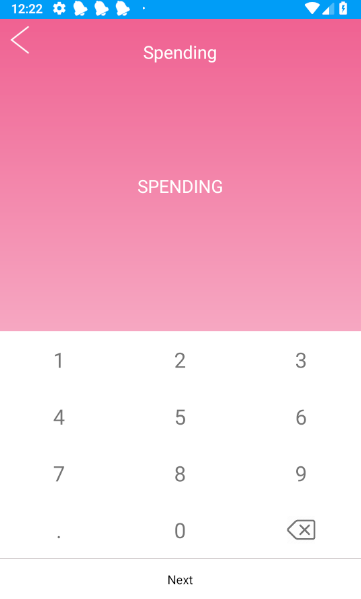
getPresenter().deleteDebt(positionGroup, positionChild);

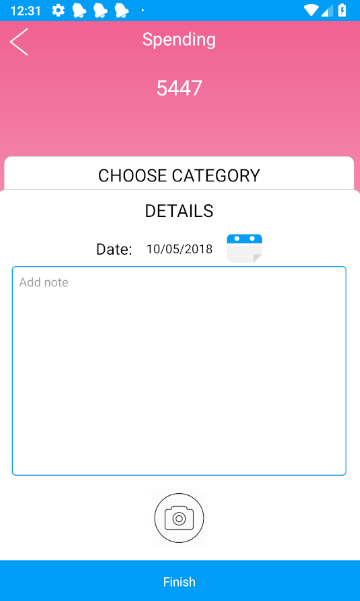
Nếu nhấn vào biểu tượng cây bút thì sẽ hiển thị dialog chỉnh sửa debt. Nếu nhấn biểu tượng back thì sẽ trở tăt dialog không thay đổi gì. Nếu nhấn nút Save thì sẽ tắt dialog, đồng thời trả về lại đối tượng đã thay đổi bằng callback lại màn hình Detail gọi API update Debt.

getPresenter().updateDebt(debt);

1. Màn hình Create

* Giao diện



* Thiết kế màn hình

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu đề** | **Mô tả** | **Quy trình** | **Kết quả** |
| Màn hình Create | LinearLayout để nhập số liệu  Custom View hiển thị các view Choose Category, Details  Button chuyển View | Thiết kế UI | Hoàn thành |
| Button | Chuyển các view | Click | Hoàn thành |
| Custom View | View Choose Category chứa Recycler để chọn loại mục.  View Detail để nhập Date, Content | Hiển thị UI và xử lý chọn các mục | Hoàn thành |
| Button Finish | Chuyển sang màn hình Home | Click, post API create | Hoàn thành |

* Luồng xử lí

Nhập vào số liệu. Sau đó sẽ có 2 TH:

* TH1: Nếu là type General thì sẽ chọn ngày theo năm và nhập note -> Nhấn Finish để tạo.
* TH2: Không phải là type General. Chọn loại jar mà chúng ta sẽ tạo. Nếu là type Debt thì sẽ nhập vào Oringin, chọn State và chọn Negative hoặc Positive. Sau đó, chọn ngày, tháng, năm và nhập note. Nhấn finish để tạo.

Khi bám finish, ta sẽ dựa vào type mà gọi API create phù hơp:

* Type Spending: gọi API create spending.

getPresenter().createSpending(

date, note, Double.*parseDouble*(**tvCurrency**.getText().toString()));

* Type Income: gọi API create income.

getPresenter().createIncomeForJar(

date, note, Double.*parseDouble*(**tvCurrency**.getText().toString()));

* Type Debt: gọi API create debt.

getPresenter().createDebt(

date, note, Double.*parseDouble*(**tvCurrency**.getText().toString()), origin, state, isPositive);

* Type General: gọi API create general.

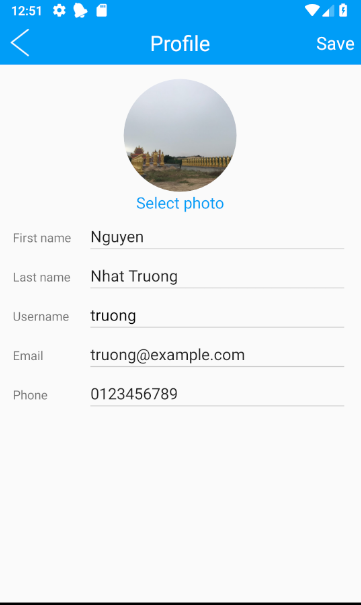
getPresenter().createGeneralIncome(

date, note, Double.*parseDouble*(**tvCurrency**.getText().toString()));

Khi gọi các API create nếu thành công sẽ trở về màn hình home, nếu không thành công thì thông báo lỗi.

1. Màn hình Profile

* Giao diện



* Thiết kế màn hình

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu đề** | **Mô tả** | **Quy trình** | **Kết quả** |
| Màn hình Home | Textfield nhập thông tin  Button Save cập nhật  ImageView hiển thị Avatar | Thiết kế UI | Hoàn thành |
| Button Save | Call API update user | Click | Hoàn thành |
| TextField | Nhập thông tin | Hiển thị UI | Hoàn thành |
| Button SelectPhoto | Chọn ảnh Avatar | Click | Hoàn thành |

* Luồng xử lý

Khi vào màn hình Profile, App call API getUser

getApiManager().getUser(getSQLiteManager().getUser().getId(), new ApiCallback<UserResponse>()

Nếu thành công thì load thông tin trả về của User vào các textfied

public void getUserSuccess(User user) {  
 if (user.getAvatarUrl() !=null){  
 Glide.*with*(this).load(user.getAvatarUrl()).apply(RequestOptions.*circleCropTransform*()).into(ivUserAvatar);  
 }  
 edtFirstName.setText(user.getFirstName());  
 edtLastName.setText(user.getLastName());  
 edtUsername.setText(user.getUserName());  
 edtEmail.setText(user.getEmail());  
 edtPhone.setText(user.getPhone());  
}

Nếu không thành công thì hiện dialog báo lỗi.

Click button Select Photo để đổi avatar. Lúc này app sẽ lấy quyền truy cập vào bộ nhớ thiết bị và chọn file ảnh làm avatar

public void handleOpenLibrary() {  
 String s[] = {android.Manifest.permission.*READ\_EXTERNAL\_STORAGE*};  
 if (checkPermissions(s)) {  
 FileUtils.*startActionGetImage*(  
 this,  
 AppConstants.*REQUEST\_READ\_LIBRARY*,  
 "Select photo"  
 );  
 } else {  
 ActivityCompat.*requestPermissions*(this, s, AppConstants.*REQUEST\_PERMISSION\_READ\_LIBRARY*);  
 }  
}

Khi chọn ảnh thành công, convert ảnh thành Part file và call API uploadImage

File file = FileUtils.*convertUriToFile*(this, data.getData());  
  
String mineType = FileUtils.*getMineType*(this, file);  
if (mineType == null) return;  
MediaType mediaType = MediaType.*parse*(mineType);  
RequestBody requestFile = RequestBody.*create*(mediaType, file);  
MultipartBody.Part filePath = MultipartBody.Part.*createFormData*("file", file.getName(), requestFile);  
  
getPresenter().uploadImage(filePath);

Nếu upload thành công thì set lại Avatar cho user và load lại màn hình

public void success(ImageResponse res) {  
 getUser().setAvatarUrl(res.getImage().getUrl());  
 if (!isViewAttached()) return;  
 getView().hideLoading();  
 getView().onUploadImageSuccess(res.getImage().getUrl());  
}

Nếu không thành công thì hiện dialog báo lỗi.

Khi thay đổi thông tin, click vào button Save. Khi đó app sẽ call API updateUser

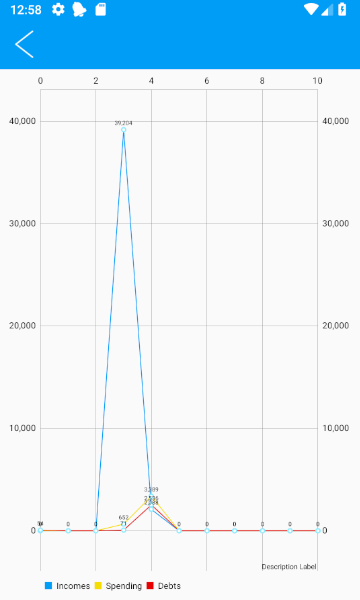
getPresenter().updateUser();

getApiManager().updateUser(getSQLiteManager().getUser().getId(), getUser(), new ApiCallback<BaseResponse>()

Nếu thành công thì load lại thông tin, nếu không thì hiện dialog báo lỗi.

1. Màn hình Chart

* Giao diện



* Thiết kế màn hình

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu đề** | **Mô tả** | **Quy trình** | **Kết quả** |
| Màn hình Chart | LineChart hiển thị biểu đồ chi tiêu | Thiết kế UI | Hoàn thành |

* Luồng xử lí

Hiển thị dữ liệu thống kê thu, chi, nợ bằng 3 đường. Nếu vuốt màn hình theo chiều từ trên xuống sẽ reload lại dữ liệu. Nhấn biểu tưởng back phía trên bên trái sẽ trở về màn hình Home.

1. Phân chia công việc

|  |  |
| --- | --- |
| **Thành viên** | **Màn hình** |
| Nguyễn Nhật Trường | Màn hình Chart, Base, Create |
| Nguyễn Hoàng Sang | Màn hình Home |
| Đào Công Danh | Màn hình Setting, Splash |
| Phạm Thanh Sơn | Màn hình Profile |
| Lê Huy Hoàng | Màn hình SignUp, Login |
| Đoàn Anh Tuấn | Màn hình Detail |

CHƯƠNG III : KẾT QUẢ THU ĐƯỢC

1. Kết quả đạt được

* Ứng dụng hiển thị được các
* Ứng dụng cho phép người dùng tạo, xóa, chỉnh sửa chi tiêu
* Ứng dụng hiển thị được biểu đồ trực quan chi tiêu từng tháng
* Lưu được lịch sử, lọc được chi tiết theo ngày tháng
* Cho phép kiểm soát được chi tiêu hiện tại của mình
* Thông báo Notification mỗi khi có thay đổi
* Đồng bộ được realtime

1. Kết luận

* *Những điều làm được*
* Làm việc với Firebase, các thư viện hỗ trợ
* Lưu dữ liệu local với SQLite
* Lưu được thông tin chi tiêu, nhập, tạo mới được các giao dịch
* Hiển thị được biểu đồ chi tiêu trong tháng
* Lọc được chi tiết theo Date
* Bắn Notification thông báo
* *Những điều chưa làm được*
* Giao diện còn chưa được đẹp mắt
* Chưa có nhiều chức năng