BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC BÀ RỊA-VŨNG TÀU

KHOA CÔNG NGHỆ KỸ THUẬT - NÔNG NGHIỆP CÔNG NGHỆ CAO

BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI

XÂY DỰNG ỨNG DỤNG KHẢO SÁT CÓ THƯỞNG OSURVEY

Giảng viên hướng dẫn	: TS. Phan Ngọc Hoàng
Sinh viên thực hiện	: Nguyễn Hữu Phước
Trình độ đào tạo	: Đại Học Chính Quy
Ngành đào tạo	: Công nghệ thông tin
Chuyên ngành	: Lập trình ứng dụng di động và game
Lớp	: DH18LT
Mã số sinh viên	: 18033458
Niên Khóa	: 2018-2022

BÀ RỊA – VŨNG TÀU, NĂM 2021

LỜI CẢM ƠN

Lời đầu tiên em xin cảm ơn cha, mẹ là những người đã có công sinh thành, nuôi dưỡng,bảo bọc, giúp em có đủ điều kiện để học tập

Em cũng chân thành cảm ơn Ban giám hiệu trường Đại Học Bà Rịa - Vũng Tàu đã tạo điều kiện cho em có môi trường học tập thoãi mái về cơ sở hạ tầng cũng như vật chất

Em xin dành lời cảm ơn đến TS. Phan Ngọc Hoàng, ThS. Nguyễn Văn Trì, TS. Bùi Thị Thu Trang, ThS. Nguyễn Thị Hà, ThS. Nguyễn Thị Minh Nương, ThS. Lê Thị Vĩnh Thanh và các Thầy, Cô trong Bộ môn Công nghệ thông tin Trường Đại học Bà Rịa – Vũng Tàu đã tận tâm hướng dẫn em trong từng buổi học trên lớp

Em chân thành cảm ơn thầy Phan Ngọc Hoàng đã dẫn dắt và hướng dẫn tận tình em phát triển, xây dựng đồ án cũng như hoàn thành bài báo cáo tốt nghiệp một cách tốt nhất

Với điều kiện thời gian cũng như kinh nghiệm còn hạn chế, đề tài đồ án này không thể tránh được những thiếu sót. Rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các quý thầy cô để có điều kiện bổ sung, nâng cao ý thức của mình, phục vụ tốt hơn công tác thực tế sau này

Em xin chân thành cảm ơn!

LỜI NÓI ĐẦU

Hiện nay tại hầu hết các doanh nghiệp, quán ăn, quán cafe, nhà hàng..., khi thực hiện những cuộc điều tra thị trường, nghiên cứu định tính theo phương pháp truyền thống thường mất nhiều thời gian, nhân lực và tài lực cho những cuộc điều tra nhiên cứu đó

Các chiến dịch điều tra, khảo sát truyền thống đòi hỏi cần sự tương tác trực tiếp, đồng nghĩa với việc một nhân sự sẽ tiếp cận với từng khách hàng. Việc này làm mất rất nhiều thời gian trong quá trình khảo sát. Doanh nghiệp cũng sẽ phải đầu tư chi phí đào tạo và thuê nhân sự. Chưa kể đến quá trình thống kê báo cáo cũng cần đầu tư lớn nhưng vẫn không thể đảm bảo chắc chắn về độ chính xác.

Phần lớn người được mời tham gia khảo sát sẽ cảm thấy không thoải mái, khó chịu khi có ai đó muốn mời họ tham gia cuộc phỏng vấn khi họ không có thời gian và thậm chí là không quan tâm đến chủ đề khảo sát. Khi đó vô tình người khảo sát đã tiếp cận sai đối tượng.

Trường hợp này, người được mời tham gia khảo sát hoàn toàn bị động, không có sự chuẩn bị cả về mặt tinh thần. Bởi vậy, thông tin thu thập có thể sẽ không đầy đủ hoạc không chính xác.

"Osurvey" - ứng dụng làm khảo sát có thưởng- là ứng dụng được tạo ra để giải quyết những khó khăn trong việc thu thập khảo sát đóng vai trò là cầu nối giữa những người muốn thu thập thông tin và những người muốn dùng thông tin của bản thân để đổi những phần quà.

LỜI CAM ĐOAN

Tôi xin cam đoan kết quả đạt được trong đồ án là sản phẩm của riêng cá nhân, không sao chép lại của người khác. Trong toàn bộ nội dung của đồ án, những điều được trình bày hoặc là của cá nhân hoặc là được tổng hợp từ nhiều nguồn tài liệu. Tất cả các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn hợp pháp.

Tôi xin hoàn toàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan của mình.

Vũng Tàu, ngày 01 tháng 11 năm 2021

Sinh viên thực hiện

Nguyễn Hữu Phước

NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC HÌNH ẢNH

DANH MỤC SƠ ĐỒ

CHƯƠNG I: GIỚI THIỆU TỔNG QUAN

1. LÝ DO CHỌN ĐỀ TÀI

Thu thập dữ liệu là một phần không thể thiếu trong tất cả các lĩnh vực nghiên cứu bao gồm khoa học vật lý và xã hội, nhân văn, kinh doanh. Lấy ví dụ trong việc kinh doanh, thu thập dữ liệu giúp doanh ngiệp hiểu các vấn đề mà doanh nghiệp đang gặp phải và việc thu thập dữ liệu hiệu quả sẽ giúp doanh nghiệp cải thiện kết quả kinh doanh, đưa ra chiến lược thị trường tốt hơn, giảm chi phí, giúp doanh nghiệp ra quyết định nhanh và chính xác hơn. Tuy nhiên việc thu thập dữ liệu theo cách truyền thống dần lạc hậu, không theo kiệp xu thế thời đại mới.

Theo báo cáo về "Thị trường ứng dụng lao động năm 2021" của Appota Group, trung bình mỗi người Việt Nam dành 6,5 giờ sử dụng Internet mỗi ngày, trong đó khoảng 3 giờ 18 phút là truy cập thông qua điện thoại di động. Điều này giúp cho việc thu thập dữ liệu trên không gian mạng trở nên có nhiều ưu điểm hơn so với việc thu thập dữ liệu theo cách truyền thống như:

- Tiết kiệm thời gian
- Tiết kiệm chi phí
- Tiết kiệm nhân lực
- Không cần quan tâm tới vấn đề thời tiết và địa lý
- Dễ dàng phân loại đối tượng thu thập dữ liệu

Trước thực trạng đó ***Osurvey" - ứng dụng làm khảo sát có thưởng*** ra đời với mong muốn giúp việc thu thập dữ liệu trở nên dễ dàng hơn

2. MỤC TIÊU VÀ CHỨC NĂNG CỦA ỨNG DỤNG

2.1. Mục tiêu

Quản lý khảo sát, sự kiện khảo sát và thống kê, các mục tiêu bao gồm:

- Tính di động và linh hoạt.
- Số hóa việc lưu trữ thông tin khảo sát, sự kiện khảo sát, người tham gia khảo sát.
- Giảm thiểu thời gian cho các sự kiện khảo sát.
- Giảm thiểu chi phí cho các sự kiện khảo sát.
- Theo dõi tình sự kiện khảo sát.
- Hỗ trợ gửi và nhận dữ liệu theo thời gian thực.

2.2. Chức năng

- Quản lý khảo sát.
- Quản lý sự kiện khảo sát.
- Lưu trữ, trích xuất dữ liệu.
- Gửi và nhận thông báo đẩy.

3. MÔ TẢ BÀI TOÁN

Mục đích của web server lả mơi tiếp nhận và quản lý các thông tin như:

- Quản lý khảo sát: Quản lý thông tin khảo sát như mã, tên, câu hỏi, ...
- Quản lý sự kiện khảo sát: Quản lý thông tin sự kiện khảo sát như mã, tên, câu hỏi, câu hỏi ...
- Quản lý người dùng: Quản lý thông tin người dùng như mã, tên, tuổi, giới tính ...

Mục đích của ứng dụng là mơi hiển thị thông tin và tiếp nhận yêu cầu của người dùng và chuyển cho web server xử lý như:

- Tạo khảo sát
- Chỉnh sửa khảo sát
- Xóa khảo sát
- Tạo sự kiện khảo sát
- Xóa sự kiện khảo sát
- Chỉnh sửa sự kiện khảo sát
- Tham gia khảo sát

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

1. CẤU TRÚC CỦA CHƯƠNG TRÌNH

Hệ thống "Osurvey" được phát triển trên nền tảng ứng dụng di động được kết nối với cơ sở dữ liệu SQLite thông qua Web Application Programming Interface (API):

- Hệ thống hỗ trợ:
  - Gửi thông báo đẩy (Push Notification) được cung cấp bởi và Firebase Message Cloud
  - Thanh toán trực tuyến qua Paypal
  - Xác thực chứng minh thư (CMT)/căn cước công dân (CCCD) được cung cấp bởi FPT.AI READER
- Phần ứng dụng di động được xây dựng bằng framework Flutter với ngôn ngữ lập trình Dart
- Phần Web server được xây dựng bằng framework Flask với ngôn ngữ lập trình Python

2. GIỚI THIỆU FLUTTER VÀ DART

2.1. Dart [1]

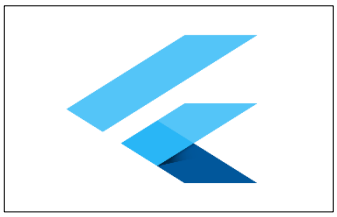
Hình 1: Biểu tượng Dart

Dart là ngôn ngữ lập trình đa mục đích ban đầu được phát triển bởi Google và sau đó được Ecma (ECMA-408) phê chuẩn làm tiêu chuẩn. Nó được sử dụng để xây dựng các ứng dụng web, server, máy tính để bàn và thiết bị di động. Dart là một ngôn ngữ hướng đối tượng, được xác định theo lớp, với cơ chế thu gom rác (garbage-collected), sử dụng cú pháp kiểu C để dịch mã tùy ý sang JavaScript. Nó hỗ trợ giao diện (interface), tái sử dụng (mixin), trừu tượng (abstract), tham số hóa kiểu dữ liệu (generic), static typing và sound type (2 cái cuối có thể hiểu là type-safe). Dart là ngôn ngữ mã nguồn mở và miễn phí, được phát triển trên Github. Hiện nay Dart đã release phiên bản 2.14.1.

Dart có những ưu điểm như:

- Có một hệ sinh thái lên đến hàng ngàn thư viện
- Là ngôn ngữ hướng đối tượng
- Có nhiều nét tương đồng với các ngôn ngữ (C++, C#, Java)
- Hỗ trợ trình biên dịch Ahead Of Time (AOT): Trình biên dịch chuyển ngôn ngữ Dart sang mã gốc (Native Code) giúp hiệu suất tối ưu có khả năng (tức là khi chạy chương trình, nó sẽ biên dịch từ đầu đến cuối)
- Hỗ trợ trình biên dịch Just In Time (JIT):Cho phép hot reloading công việc, giúp phát triển mặt hàng nhanh và tiện dụng hơn (viết đến đâu biên dịch ngay đến đấy)

2.2. Flutter [2]

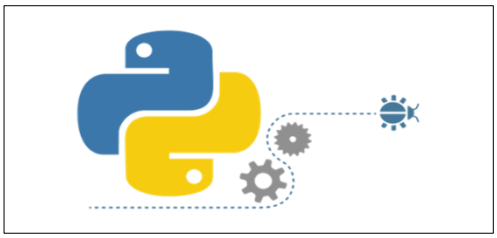


Hình 2: Biểu tượng Flutter

- Flutter là mobile User Interface framework của Google để tạo ra các giao diện chất lượng cao trên iOS và Android trong khoảng thời gian ngắn. Flutter hoạt động với những code sẵn có được sử dụng bởi các lập trình viên, các tổ chức.
- Flutter hoàn toàn miễn phí và cũng là mã nguồn mở.
- Flutter sử dụng Dart, một ngôn ngữ nhanh, hướng đối tượng với nhiều tính năng hữu ích như mixin, generic, isolate, và static type.
- Flutter có các thành phần UI của riêng nó, cùng với một cơ chế để kết xuất chúng trên nền tảng Android và iOS. Hầu hết các thành phần giao diện người dùng, đều sẵn dùng, phù hợp với các nguyên tắc của Material Design.
- Các ứng dụng Flutter có thể được phát triển bằng cách sử dụng IntelliJ IDEA, một IDE rất giống với Android Studio.

3. GIỚI THIỆU FLASK VÀ PYTHON

3.1. Giới thiệu Python



Hình 3: Biểu tượng Python

3.1.1. Tổng quan [3]

Ngôn ngữ Python là một ngôn ngữ lập trình mã nguồn mở, đa nền tảng, dễ học dễ đọc. Python có cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Vì thế nó được sử dụng rộng rãi.

Python là ngôn ngữ hỗ trợ nhiều mẫu đa lập trình khác nhau như: mệnh lệnh, lập trình hướng đối tượng, lập trình hàm, … được dùng đa lĩnh vực: web, 3D CAD,…

3.1.2. Ưu điểm khi lập trình web với python [3]

- **Đơn giản:** Cú pháp đơn giản giúp cho người lập trình dễ dàng đọc và tìm hiểu
- **Tốc độ :** Python có tốc độ xử lý nhanh hơn so với ngôn ngữ PHP.
- **Tương tác:** Chế độ tương tác cho phép người lập trình thử nghiệm tương tác sửa lỗi của các đoạn mã.
- **Chất lượng:** Thư viện có tiêu chuẩn cao, Python có khối cơ sở dữ liệu khá lớn nhằm cung cấp giao diện cho tất cả các CSDL thương mại lớn.
- **Thuận tiện:** Python được biên dịch và chạy trên tất cả các nền tảng lớn hiện nay.

3.2. Giới thiệu Flask

Hình 4: Biểu tượng Flask

3.2.1. Tổng quan [4]

Flask là một web application frameworks, nó thuộc loại micro-framework được xây dựng bằng ngôn ngữ lập trình Python. Flask cho phép các lập trình viên có thể xây dựng các ứng dụng web từ đơn giản tới phức tạp. Nó có thể xây dựng các API nhỏ, ứng dụng web chẳng hạn như các trang web, blog, trang wiki hoặc một website dựa theo thời gian hay thậm chí là một trang web thương mại.

3.2.2. Ưu điểm của Flask [4]

- Xây dựng web application rất giống với việc viết các module Python chuẩn, cấu trúc gọn gàng và rõ ràng.
- Flask cung cấp cho người dùng các thành phần cốt lõi thường được sử dụng nhất của khung ứng dụng web như định tuyến, đối tượng yêu cầu và phản hồi, bản mẫu, ...

4. GIỚI THIỆU FPT.AI READER

Hình 5: Biểu tượng FPT AI

4.1. Tổng quan [5]

FPT.AI Reader cung cấp các dạng và trích xuất chính xác nội dung từ ảnh chụp mẫu văn bản có sẵn như chứng minh thư (CMT)/căn cước công dân (CCCD), giấy phép lái xe, hóa đơn, ...

4.2. Nhận diện CMT/CCCD [6]

Đây là hệ thống tự động hóa quy trình nhận diện và trích xuất thông tin một cách chính xác từ thẻ CMT/CCCD của FPT.AI, ứng dụng những công nghệ mới nhất trong xử lý ảnh, nhận dạng ký tự quang học (Optical Character Recognition) và học sâu (Deep Learning).

Hệ thống nhận diện và trích xuất thông tin chứng minh nhân dân/căn cước công dân Việt Nam của FPT.AI có thể được sử dụng để tự động hóa quy trình nhập liệu trong rất nhiều lĩnh vực bao gồm tài chính ngân hàng, viễn thông, du lịch, lưu trú, ... Cụ thể hơn là trong những tác vụ như mở tài khoản ngân hàng, mở thẻ tín dụng, đăng ký người dùng, xác nhận đặt phòng, check-in, ...

5. GIỚI THIỆU PAYPAL PAYNENT GATEWAY

Hình 6: Biểu tượng Paypal

5.1. Tổng quan [7]

Paypal là một công ty hoạt động trong lĩnh vực thương mại điện tử, chuyên cung cấp các dịch vụ thanh toán và chuyển tiền qua mạng Internet

5.2. Paypal API [7]

Paypal API cung cấp vụ thanh toán và chuyển khoản điện tử thay thế cho các phương thức truyền thống sử dụng giấy tờ như séc và các lệnh chuyển tiền. Hiện nay Paypal được chấp nhận ở 200 quốc gia.

Hiện nay có rất nhiều trang web không thể chuyển khoản trực tiếp về các tài khoản ngân hàng trong nước được vì có quá nhiều dữ liệu cũng như chính sách pháp lý của

từng đất nước. Chính vì điều đó mà Paypal ra đời, Paypal tuân thủ các chính sách pháp lý của từng đất nước khác nhau và Paypal chính là nơi trung gian để người dùng thực hiện các giao dịch tiền tệ.

6. GIỚI THIỆU FIREBASE

Hình 7 Biểu tượng Firebase

6.1. Tổng quan [8]

Firebase là dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây (cloud). Kèm theo đó là hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính là giúp người dùng lập trình ứng dụng bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu.

6.2. Firebase Cloud Messaging (FCM) [8]

Firebase Cloud Messaging (FCM) là một dịch vụ miễn phí của Google. Thông qua FCM, nhà phát triển ứng dụng có thể gửi thông điệp một cách nhanh chóng, an toàn tới các thiết bị cài đặt ứng dụng của họ.

CHƯƠNG III: PHÂN TÍCH HỆ THỐNG

1. MÔ TẢ HỆ THỐNG

1.1. Quản lý khảo sát

- Quản lý khảo sát bao gồm mã người tạo khảo sát, tên, hình ảnh của khảo sát.
- Quản lý trạng thái của khảo sát gồm hai trạng thái đang được sử dụng bởi sự kiện khảo sát và đang rãnh.
- Quản lý các câu hỏi của khảo sát gồm có mã câu hỏi, nội dung câu hỏi và mã khảo sát.
- Quản lý các lựa chọn cho những câu hỏi trong khảo sát gồm có mã lựa chọn và nội dung lựa chọn.

1.2. Quản lý sự kiện khảo sát

- Quản lý sự kiện khảo sát bao gồm mã sự kiện khảo sát, mã khảo sát, thời gian sự kiện bắt đầu, thời gian sự kiện kết thúc, số lượng người tham gia, số tiền thưởng cho người tham gia sự kiện, và số tiền thưởng cho người may mắn.
- Quản lý những người tham gia sự kiện khảo sát bao gồm mã người tham gia sự kiện khảo sát, mã sự kiện khảo sát và số tiền thưởng nhận được trong sự kiện khảo sát.
- Quản lý câu trả lời của những người tham gia sự kiện khảo sát bao gồm mã người tham gia sự kiện khảo sát, mã sự kiện khảo sát, mã câu hỏi của khảo sát trong sự kiện và nội dung câu trả lời.
- Quản lý trạng thái của sự kiện gồm có trạng thái chờ, trạng thái đang diễn ra ra và trạng thái đã hoàn thành.
- Quản lý hóa đơn của sự kiện khảo sát bao gồm mã hóa đơn, tổng tiền, trạng thái thanh toán, mã thanh toán từ Paypal.

1.3. Quản lý tài khoản:

- Quản lý thông tin tài khoản bao gồm tên, địa chỉ, số điện thoại, email cá nhân, giới tính, địa chỉ, ngày sinh, tuổi, số CMT/CCCD.

2. XÂY DỰNG CÁC CHỨC NĂNG CỦA HỆ THỐNG:

2.1. Mô tả các chức năng

Sau khi đăng ký tài khoản người dùng có thể sử dụng các chức năng:

- Đăng nhập
- Đăng xuất
- Tạo khảo sát
- Chỉnh sửa khảo sát
- Xóa khảo sát
- Tạo sự kiện khảo sát
- Tải xuống thông sự kiện khảo sát
- Xóa sự kiện khảo sát
- Tham gia sự kiện khảo sát
- Chỉnh sửa thông tin cá nhân

Bảng 1: Bảng mô tả chức năng quản lý khảo sát

STT	Chức năng	Diễn giải
1	Đăng ký	Cho phép người dùng đăng ký tài khoản
2	Cập nhật thông tin	Cho phép cập nhật thông tin tài khoản

Bảng 2: Bảng mô tả chức năng quản lý khảo sát

STT	Chức năng	Diễn giải
1	Tạo mới	Cho phép tạo mới khảo sát
2	Chỉnh sửa	Cho phép chỉnh sửa thông tin khảo sát
3	Xóa	Cho phép xóa khảo sát

Bảng 3: Bảng mô tả chức năng quản lý sự kiện khảo sát

STT	Chức năng	Diễn giải
1	Tạo mới	Cho phép tạo mới sự kiện khảo sát
2	Chỉnh sửa	Cho phép chỉnh sửa thông tin sự kiện khảo sát
3	Xóa	Cho phép xóa khảo sát

Bảng 4:Bảng mô tả chức năng quản tham gia sự kiện khảo sát

STT	Chức năng	Diễn giải

1	Tham gia sự kiện khảo sát	Cho phép tham gia sự kiện khảo sát

2.2. Đặc tả user case

2.2.1. Chức năng đăng ký

Chức năng cho phép những người chưa có tài khoản có thể đăng ký tài khoản. Để sử dụng chức năng, người dùng cần có căn cước công dân và email để thực hiện chức năng đăng ký. Sau khi nhập đầy đủ thông tin người dùng nhấn nút "Đăng ký". Nếu thông tin xác thực chính xác, ứng dụng thông báo đăng ký thành công. Nếu thông tin đăng nhập không chính xác, ứng dụng hiện thông báo lỗi cho người dùng và yêu cầu người dùng kiểm tra lại thông tin. Quy trình thực hiện việc đăng ký như sau:

Bước 1: Người dùng nhập thông tin đăng ký và nhấn đăng ký

Bước 2: Ứng dụng kiểm tra định dạng và gửi yêu cầu đến server

Bước 3: Server kiểm tra định dạng và gửi yêu cầu kiểm tra CMT/CCCD đến FPT Vistion API

Bước 4: FPT Vistion API tìm kiếm thông tin CMT/CCCD và gửi phản hồi cho server

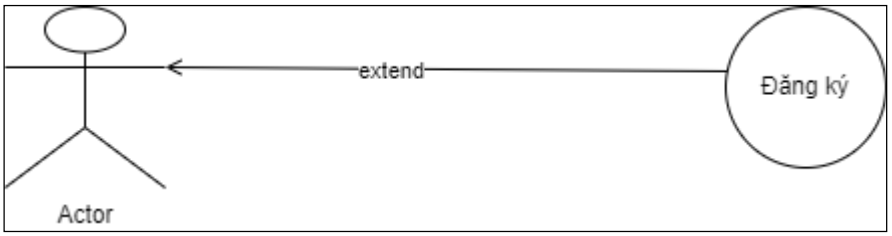
Bước 5: Server xử lý phản hồi và lưu thông tin đăng ký

Bước 6: Server gửi nội dung thông báo và khóa ứng dụng cho firebase cloud message

Bước 7: firebase cloud message gửi phản hồi có nội dung thông báo cho ứng dụng

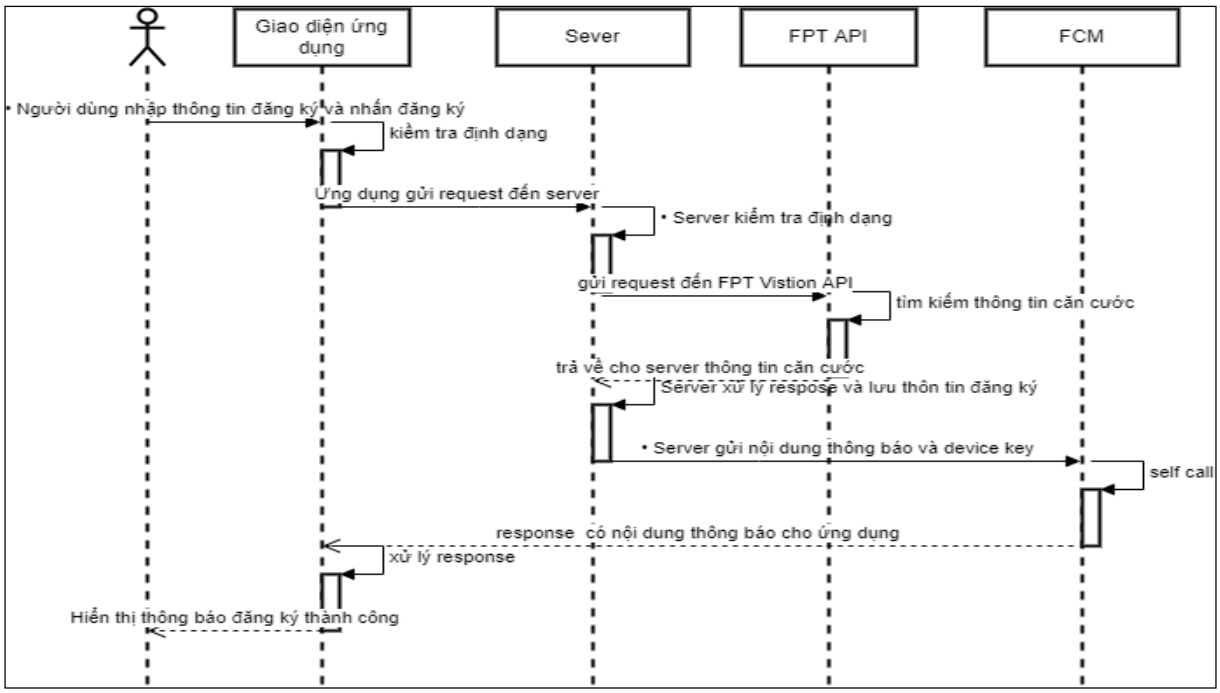
Bước 8: Ứng dụng xử lý phản hồi và hiển thị thông báo đăng ký thành công

Sơ đồ giữa tác nhân và usercase:



Sơ đồ 1:Sơ đồ user-case chức năng đăng ký

Sơ đồ tuần tự



Sơ đồ 2:Sơ đồ tuần tự chức năng đăng ký

2.2.2. Chức năng đăng nhập

Người dùng thực hiện chức năng đăng nhập vào hệ thống bằng cách truy cập vào ứng dụng di động trên thiết bị, sau đó ứng dụng hiển thị giao diện đăng nhập yêu cầu nhập email và mật khẩu. Người dùng nhập đầy đủ thông tin và nhấn nút "Đăng nhập". Nếu thông tin xác thực chính xác, ứng dụng thông báo đăng nhập thành công và chuyển hướng đến giao diện trang chủ, nếu thông tin đăng nhập không chính xác, ứng dụng thông báo lỗi, không chuyển hướng và yêu cầu đăng nhập lại. Quy trình thực hiện đăng nhập như sau:

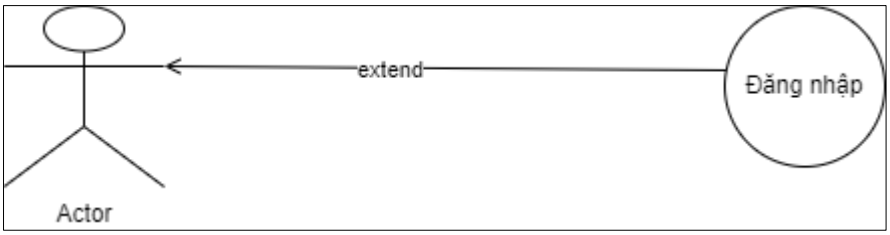
Bước 1: Người dùng nhập thông tin đăng nhập và nhấn nút đăng nhập

Bước 2: Ứng dụng gửi yêu cầu đăng nhập đến server

Bước 3: Server kiểm tra thông tin đăng nhập và gửi thông tin đăng đăng nhập cho ứng dụng

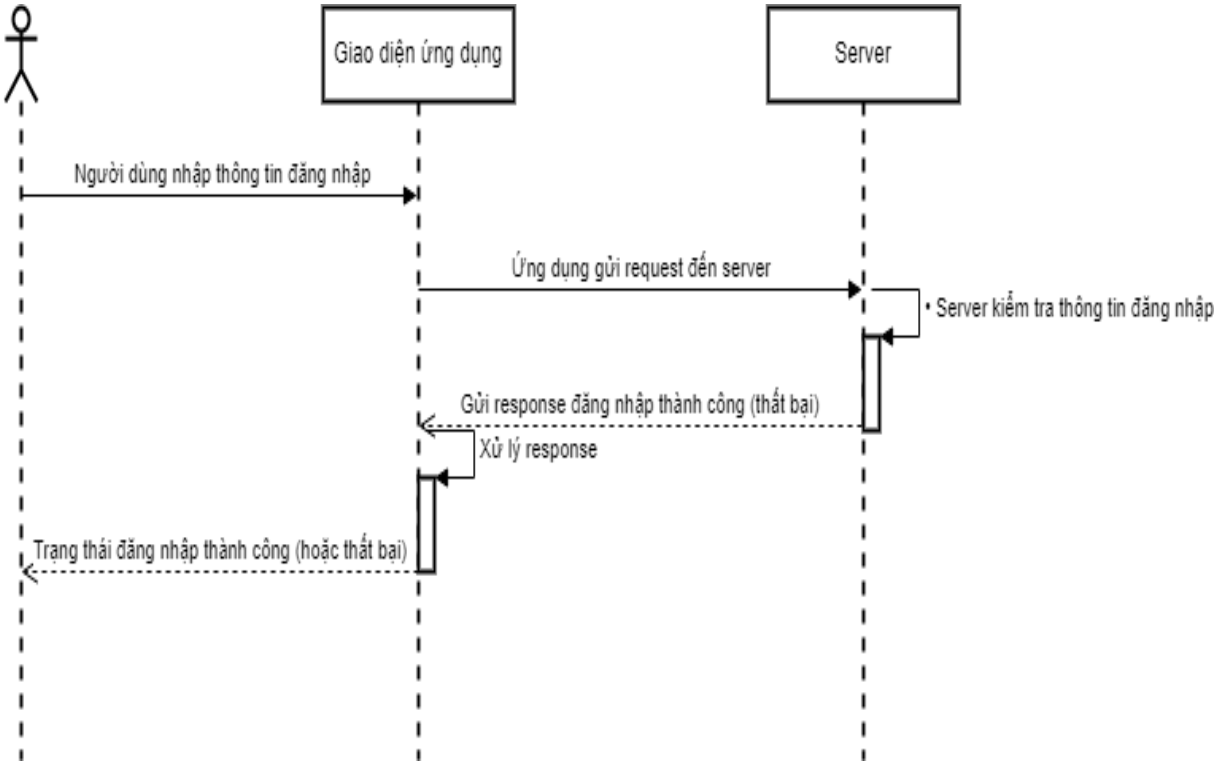
Bước 4: Ứng dụng xử lý phản hồi và hiển thị trạng thái khi đăng nhập thành công (hoặc thất bại)

Sơ đồ giữa tác nhân và usercase:



Sơ đồ 3:Sơ đồ user-case chức năng nhập

Sơ đồ tuần tự:



Sơ đồ 4: Sơ đồ tuần tự chức năng nhập

2.2.3. Chức năng chỉnh sửa thông tin cá nhân

Khi người dùng đăng nhập và chuyển sang tab tài khoản, người dùng có thể nhấn nút " Sửa" để chỉnh sửa thông tin cá nhân. Nếu chỉnh sửa thông tin thành công

ứng dụng làm mới lại tab tài khoản, nếu không ứng dụng sẽ hiện thông báo thay đổi thông tin cá nhân không thành công, Quy trình chỉnh sửa thông tin cá nhân như sau:

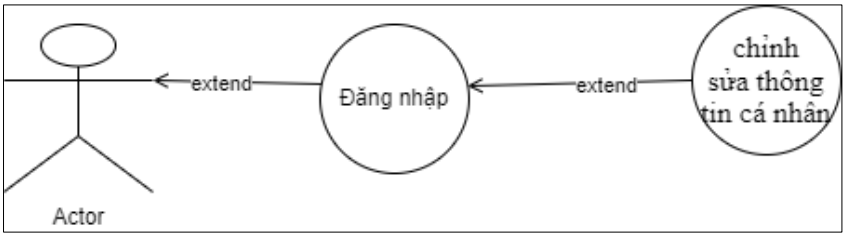
Bước 1: Người dùng nhập thông tin cá nhân cần sửa và nhấn sửa

Bước 2: Ứng dụng gửi yêu cầu chỉnh sửa thông tin cá nhân đến server

Bước 3: Server lưu thông tin đã chỉnh sửa và gửi phản hồi cho ứng dụng

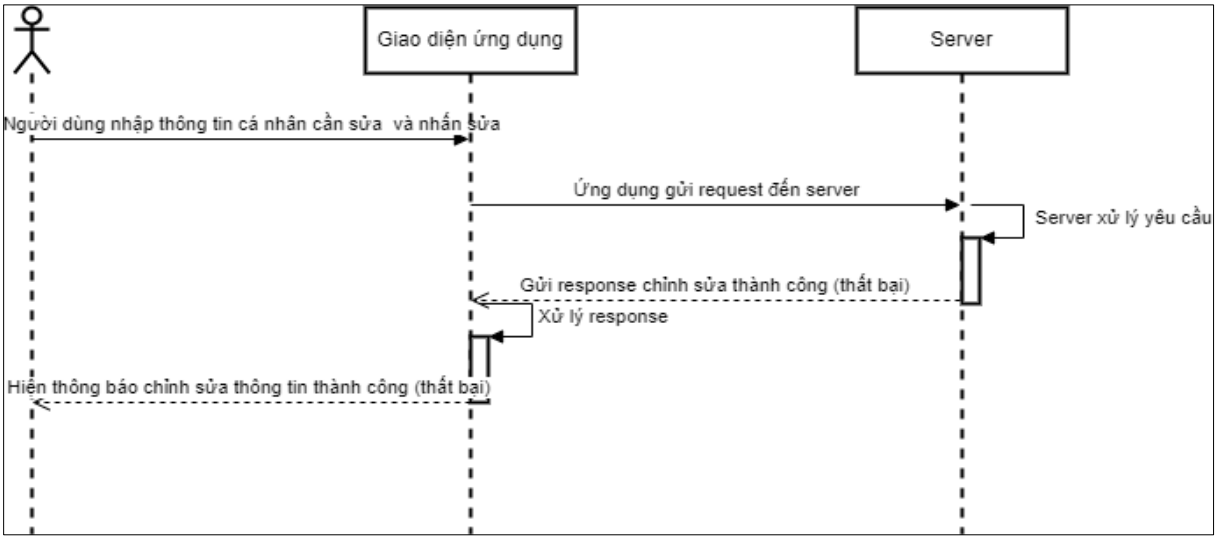
Bước 4: Ứng dụng xử lý phản hồi và hiện thông báo thay đổi thông tin cá nhân thành công

Sơ đồ giữa tác nhân và usercase:



Sơ đồ 5: Sơ đồ user-case chức năng chỉnh sửa thông tin cá nhân

Sơ đồ tuần tự:



Sơ đồ 6: Sơ đồ tuần tự chức năng chỉnh sửa thông tin cá nhân

2.2.4. Chức năng tạo sự kiện khảo sát

Sau khi đăng nhập thành công người dùng có thể tạo sự kiện khảo sát với điều kiện có ít nhất một khảo sát đã được tạo. Sau khi người dùng nhập đầy đủ các thông tin như mã của khảo sát, số lượng người tham gia, số tiền thưởng cho những người tham gia, số tiền thưởng cho người may mắn thì có thể tiến hành nhấn tạo khảo sát. Nếu định dạng dữ liệu chính xác ứng dụng sẽ hiện giao diện thanh toán để người dùng tiến hành thanh toán. Nếu định dạng dữ liệu không chính xác thì sẽ hiện lỗi và yêu cầu người dùng nhập lại. Quy trình tạo sự kiện khảo sát như sau:

Bước 1: Người dùng nhập thông tin sự kiện và nhấn lưu

Bước 2: Ứng dụng kiểm tra định dạng dữ liệu và gửi yêu cầu tạo sự kiện khảo sát đến server

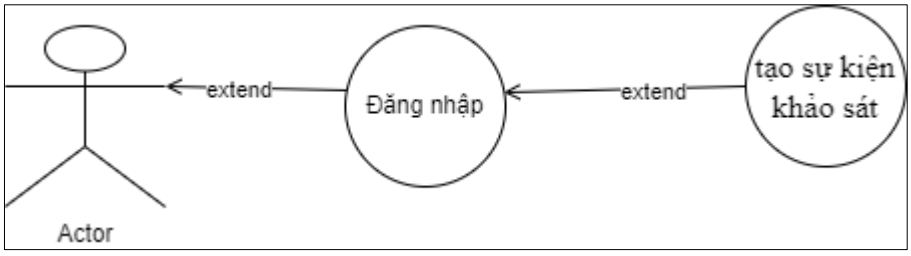
Bước 3: Server xử lý yêu cầu tạo sự kiện khảo sát và gửi yêu cầu tạo thanh toán đến Paypal gateway

Bước 4: Paypal gateway xử lý thông tin thanh toán và gửi phản hồi về server

Bước 5: Server xử lý thông tin thanh toán và gửi phản hồi về ứng dụng

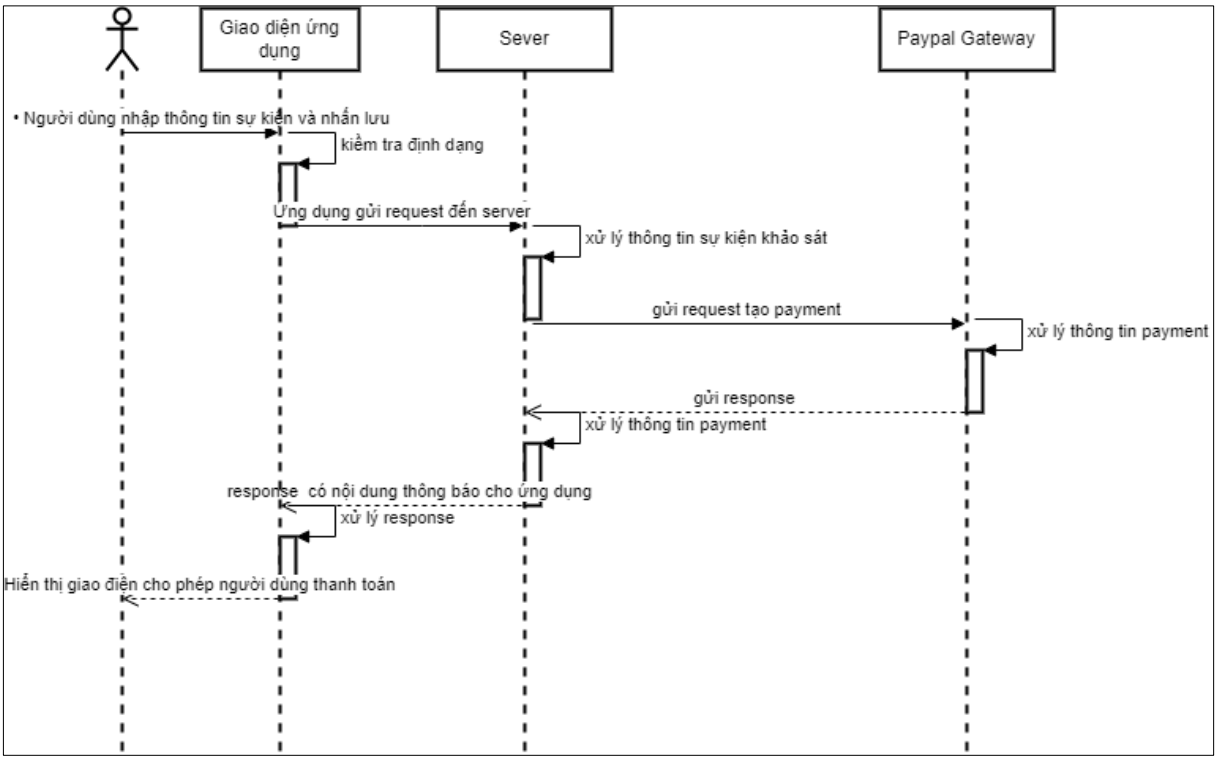
Bước 6: Ứng dụng xử lý thông tin phản hồi và hiển thị giao điện cho phép người dùng thanh toán

Sơ đồ giữa tác nhân và usercase:



Sơ đồ 7: Sơ đồ user-case chức năng tạo sự kiện

Sơ đồ giữa tuần tự:



Sơ đồ 8: Sơ đồ tuần tự chức năng tạo sự kiện khảo sát

2.2.5. Chức năng chỉnh sửa sự kiện khảo sát

Sau khi người dùng nhập thông tin chỉnh sửa sự kiện thì nhấn nút lưu. Nếu sự kiện khảo sát ở trạng thái chờ hoặc hoàn thành, ứng dụng hiện thông báo chỉnh sửa sự kiện thành công. Nếu sự kiện khảo sát ở trạng thái hoạt động thì người hiện thông báo chỉnh sửa sự kiện khảo sát thất bại. Quy trình chỉnh sửa sự kiện khảo sát như sau:

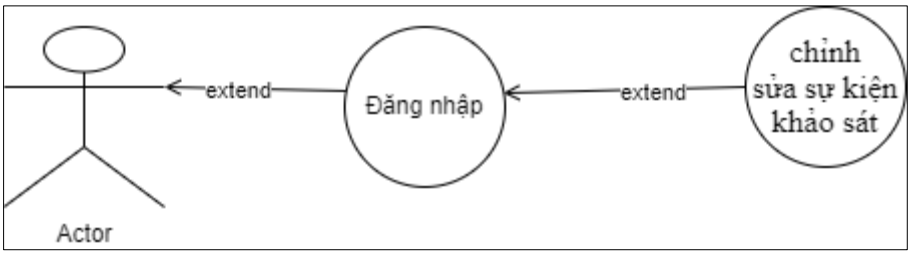
Bước 1: Người dùng nhập thông tin sự kiện khảo sát cần sửa và nhấn sửa

Bước 2: Ứng dụng kiểm tra định dạng dữ liệu và gửi yêu cầu chỉnh sửa sự kiện khảo sát đến server

Bước 3: Server xử lý yêu cầu chỉnh sửa sự kiện khảo sát và gửi phản hồi cho ứng dụng

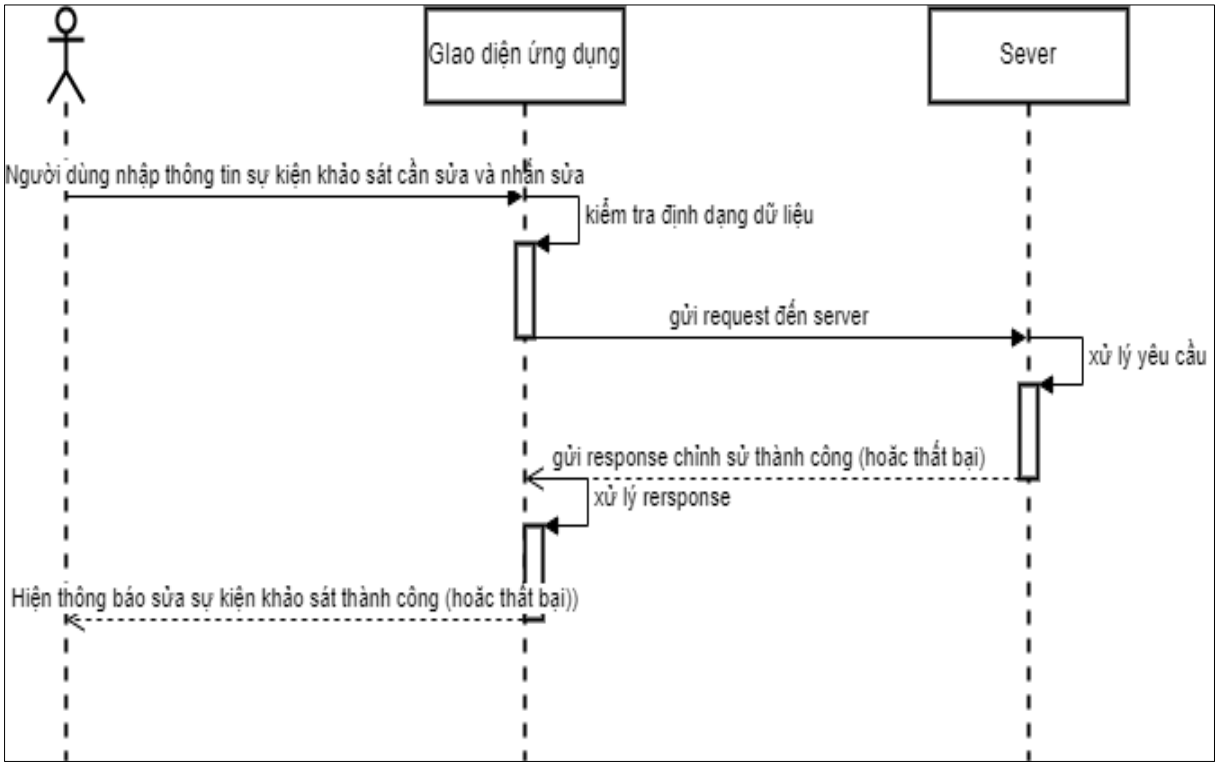
Bước 4: Ứng dụng xử lý phản hồi và hiện thông báo sửa sự kiện khảo sát thành công

Sơ đồ giữa tác nhân và usercase



Sơ đồ 9: Sơ đồ user-case chức năng chỉnh sửa sự kiện khảo sát

Sơ đồ tuần tự



Sơ đồ 10: Sơ đồ tuần tự chức năng chỉnh sửa sự kiện khảo sát

2.2.6. Chức năng xóa sự kiện khảo sát

Người dùng nhấn nút "xóa sự kiện khảo sát". Nếu sự kiện khảo sát ở trạng thái chờ hoặc hoàn thành, ứng dụng hiện thông báo xóa sự kiện thành công. Nếu sự kiện ở trạng thái hoạt động thì người hiện thông báo xóa sự kiện khảo sát thất bại. Quy trình thực hiện xóa khảo sát như sau:

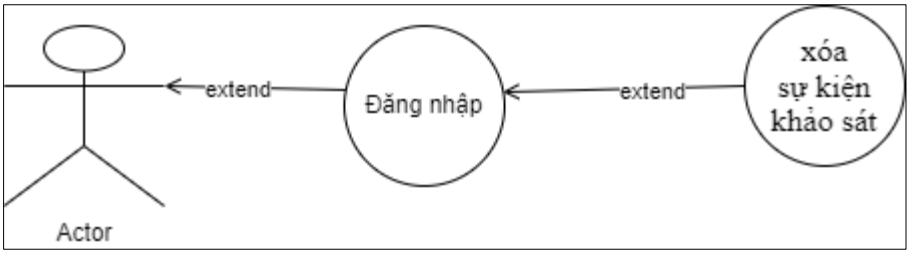
Bước 1: Người dùng nhấn xóa sự kiện khảo sát

Bước 2: Ứng dụng gửi yêu cầu xóa sự kiện khảo sát đến server

Bước 3: Server xử lý yêu cầu xóa sự kiện khảo sát và gửi phản hồi cho ứng dụng

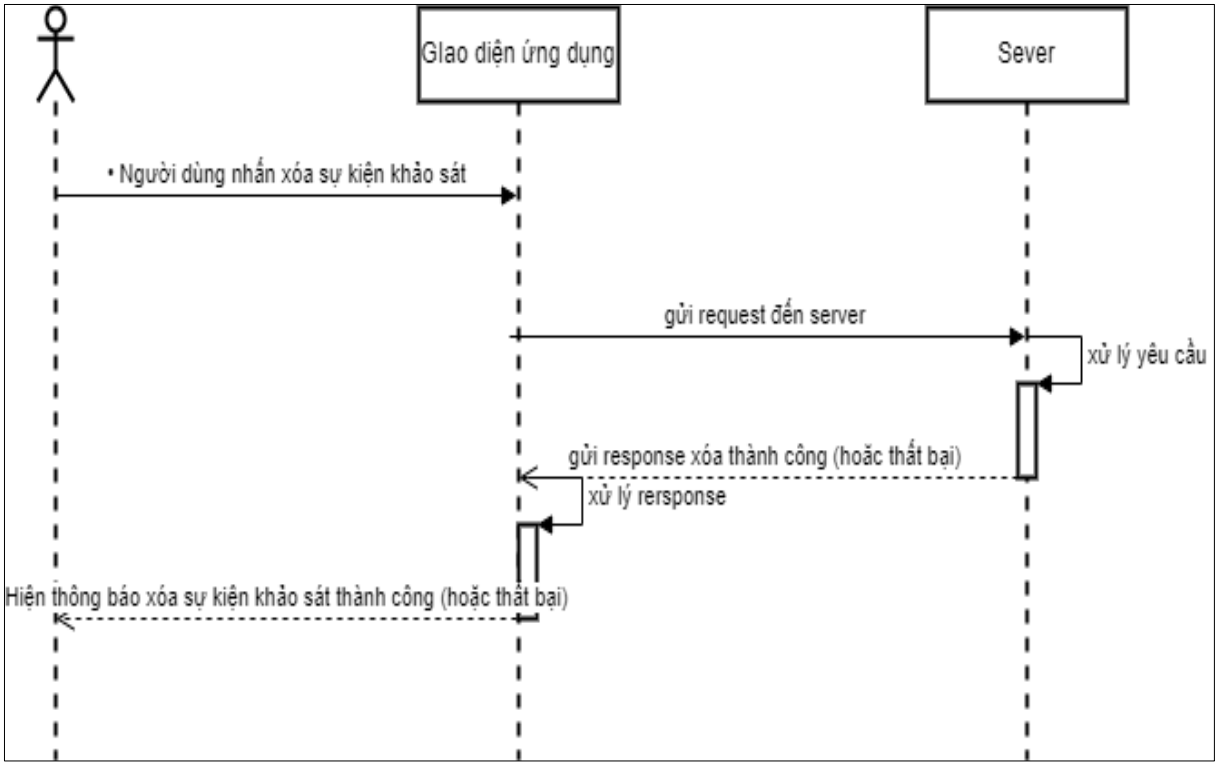
Bước 4: Ứng dụng xử lý phản hồi và hiện thông báo xóa sự kiện khảo sát thành công (hoặc thất bại)

Sơ đồ giữa tác nhân và usercase:



Sơ đồ 11: Sơ đồ user-case chức năng xóa sự kiện khảo sát

Sơ đồ giữa tuần tự:



Sơ đồ 12 Sơ đồ tuần tự chức chỉnh sửa sự kiện khảo sát

2.2.7. Chức năng tải thông tin sự kiện khảo sát

Khi người dùng nhấn nút “Tải”. Nếu ứng dụng tải xuống và lưu dữ liệu thành công, ứng hiện thông báo tải thành công. Nếu ứng dụng tải xuống và lưu dữ liệu thất bại thì sẽ hiện thông báo tải thất bại. Quy trình thực hiện tải thông tin sự kiện khảo sát như sau:

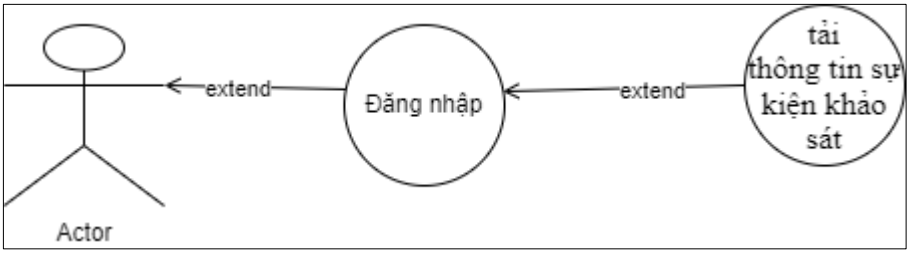
Bước 1: Người dùng nhấn tải thông tin sự kiện khảo sát

Bước 2: Ứng dụng gửi yêu cầu tải thông tin sự kiện khảo sát đến server

Bước 3: Server truy xuất dữ liệu và gửi tệp văn bản được phân tách bởi dấu phẩy (Comma Separated Values) cho ứng dụng

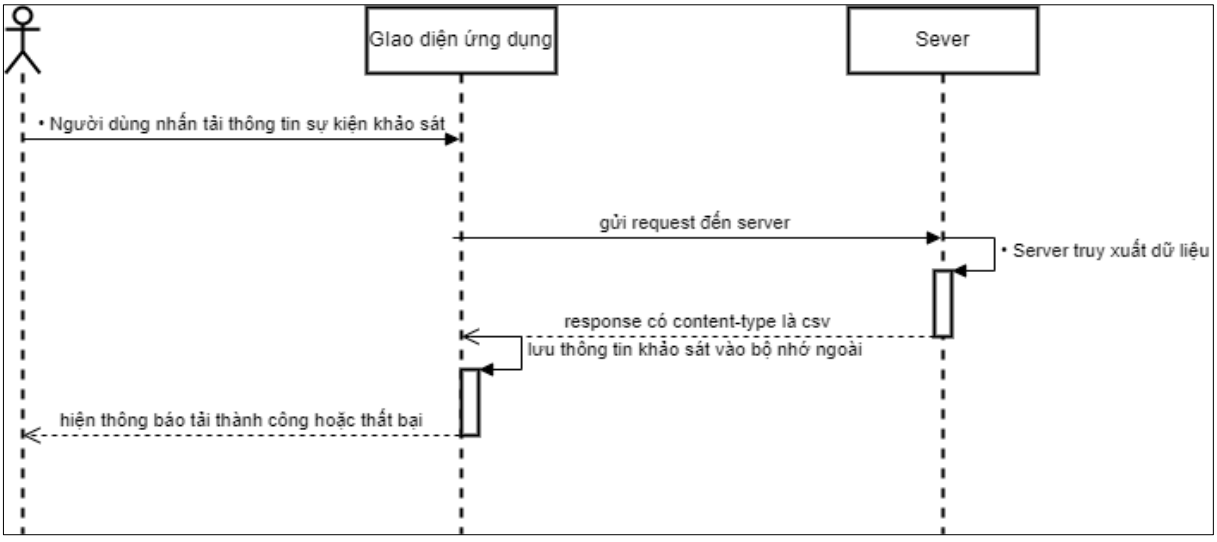
Bước 4: Ứng dụng lưu thông tin khảo sát vào bộ nhớ ngoài, hiện thông báo tải thành công

Sơ đồ giữa tác nhân và usercase:



Sơ đồ 13: Sơ đồ user-case chức năng tải thông tin sự kiện khảo sát

Sơ đồ tuần tự:



Sơ đồ 14 Sơ đồ tuần tự chức năng tải thông tin sự kiện khảo sát

2.2.8. Chức năng tạo khảo sát

Sau khi đăng nhập thành công, người dùng có thể tạo khảo sát bằng cách di chuyển sang tab khảo sát. Tại tab khảo sát người dùng nhấn tạo khảo sát khảo sát. Khi thông tin của khảo sát đã được điền đầy đủ người dùng nhấn "Lưu". Nếu dữ liệu đúng định dạng, ứng dụng sẽ hiện thông báo tạo khảo sát thành công. Quy trình tạo khảo sát như sau:

Bước 1: Người dùng nhập thông tin của khảo sát và nhấn lưu

Bước 2: Ứng dụng kiển tra định dạng và gửi yêu cầu tạo khảo sát đến server

Bước 3: Server lưu thông tin khảo sát và gửi phản hồi đến ứng dụng

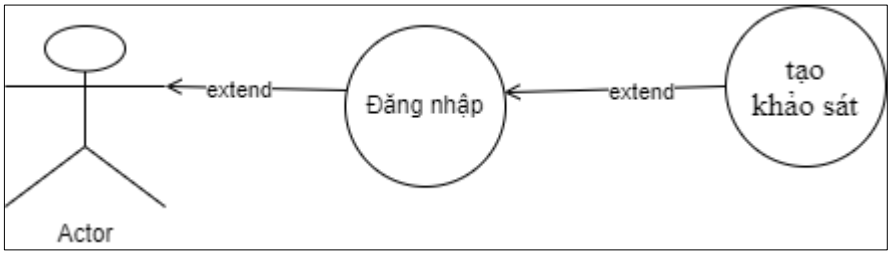
Bước 4: Ứng dụng hiện giao diện lưu thông tin khảo sát thành công

Bước 5: Server gửi nội dung thông báo thành công và mã thiết bị đến Firebase Cloud Messaging

Bước 6: Firebase cloud xử lý thông tin và gửi phản hồi về ứng dụng

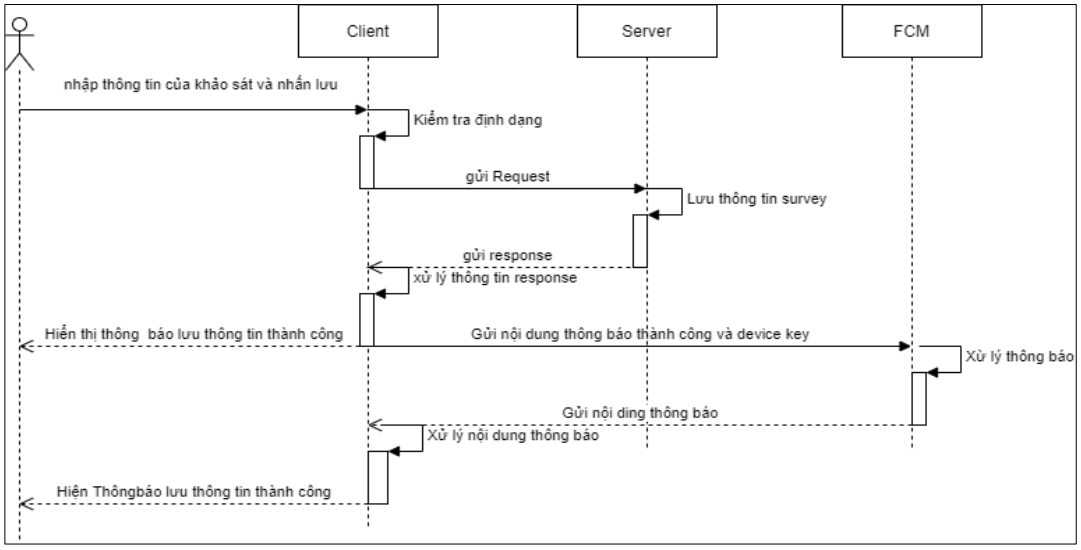
Bước 7: Ứng dụng xử lý thông tin và hiện thông tin lưu khảo sát thành công

Sơ đồ giữa tác nhân và usercase:



Sơ đồ 15: Sơ đồ user-case chức năng tạo khảo sát

Sơ đồ tuần tự:



Sơ đồ 16 Sơ đồ tuần tự chức năng tạo khảo sát

2.2.9. Chức năng xóa khảo sát

Khi người dùng chuyển sang tab khảo sát và nhấn "xóa" một khảo sát có trong danh sách khảo sát. Nếu khảo sát không nằm trong sự kiện nào, ứng dụng thông báo xóa khảo sát thành công và làm mới danh sánh khảo sát hiển thị trên giao diện. Nếu khảo sát nằm trong một sự kiện ứng dụng hiện thông báo không thể xóa khảo sát. Quy trình xóa khảo sát như sau:

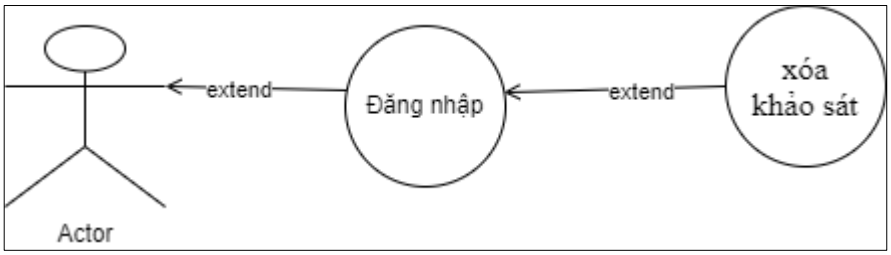
Bước 1: Người dùng nhấn xóa khảo sát

Bước 2: Ứng dụng gửi yêu cầu đến server

Bước 3: Server sử lý yêu cầu xóa khảo sát và gửi phản hồi có nội dung xóa khảo sát thành công đến ứng dụng

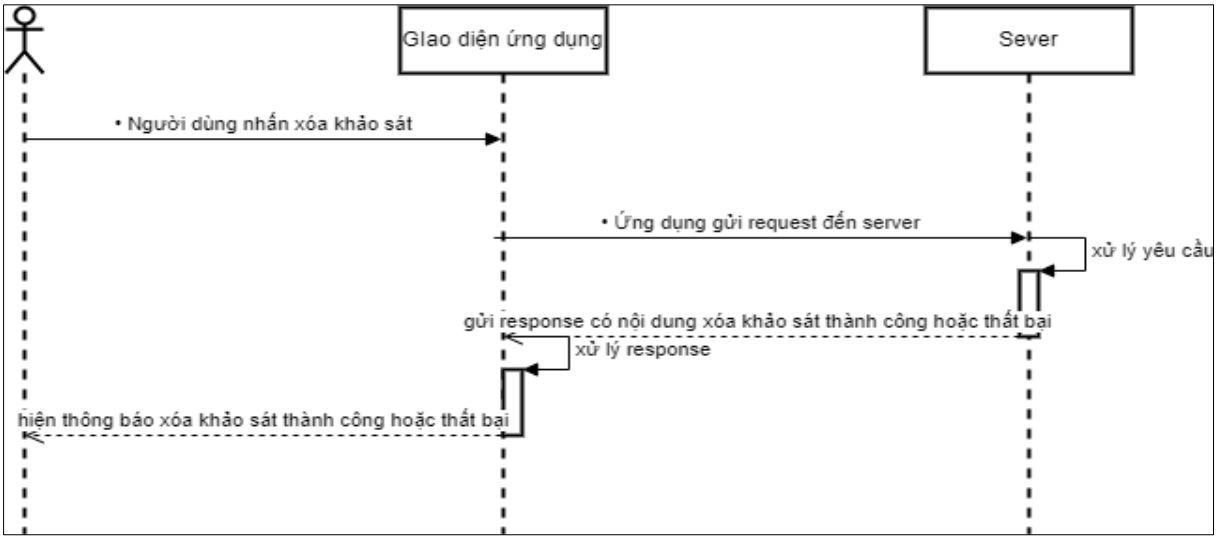
Bước 5: Ứng dụng xử lý phản hồi và hiển thị thông báo xóa khảo sát thành công (hoặc thất bại)

Sơ đồ giữa tác nhân và usercase:



Sơ đồ 17: Sơ đồ user-case chức năng xóa khảo sát

Sơ đồ tuần tự:



Sơ đồ 18 Sơ đồ user-case chức năng xóa khảo sát

2.2.10. Chức năng chỉnh sửa khảo sát

Khi người dùng nhập thông tin khảo sát cần sửa và nhấn sửa, ứng dụng sẽ kiểm tra trạng thái của khảo sát. Nếu khảo sát ở trong một sự kiện đang diễn ra, ứng dụng hiện thông báo chỉnh sửa không thành công. Nếu khảo sát không ở trong một sự kiện đang diễn ra thì ứng dụng hiện thông báo sửa khảo sát thành công. Quy trình chỉnh sửa khảo sát như sau:

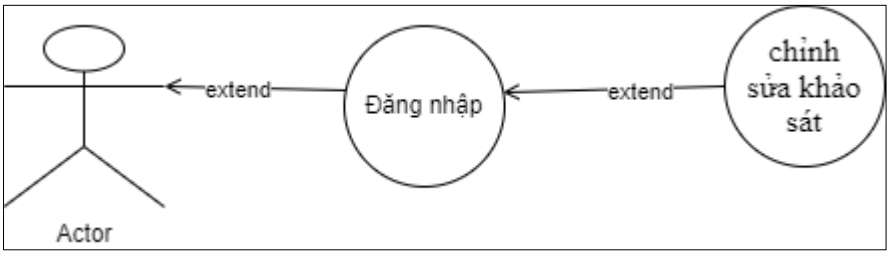
Bước 1: Người dùng nhập thông tin khảo sát cần sửa và nhấn sửa

Bước 2: Ứng dụng gửi yêu cầu chỉnh sửa khảo sát đến server

Bước 3: Server xử lý yêu cầu chỉnh sửa khảo sát và gửi phản hồi cho ứng

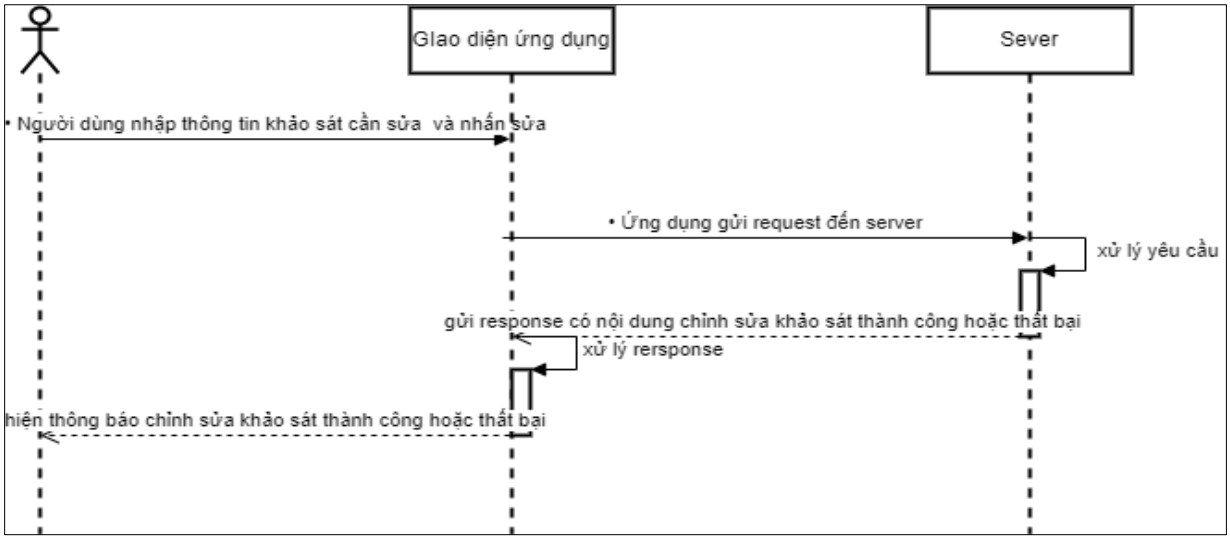
Bước 4: Ứng dụng xử lý phản hồi và hiện thông báo sửa khảo sát thành công (hoặc thất bại)

Sơ đồ giữa tác nhân và usercase:



Sơ đồ 19 Sơ đồ user-case chức năng chỉnh sửa

Sơ đồ tuần tự:



Sơ đồ 20: Sơ đồ tuần tự chức năng xóa khảo sát

2.2.11.Chức năng tham gia sự kiện khảo sát

Khi người dùng nhấn vào sự kiện khảo sát để tham gia, ứng dụng hiện giao diện tham gia sự kiện khảo sát để người dùng tham gia. Sau khi hoàn thành khảo sát, người dùng nhấn "Gửi", ứng dụng hiện thông báo cảm ơn người dùng. Quy trình tham gia khảo sát như sau:

Bước 1: Người dùng nhấn vào tham gia sự kiện đề tham gia sự kiện khảo sát

Bước 2: Giao diện hiển thị giao diện tham gia khảo sát

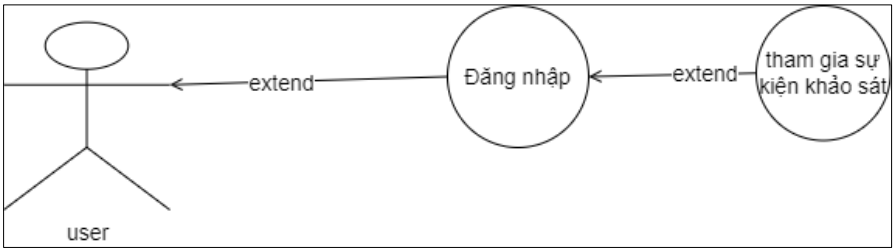
Bước 3: Người dùng hoàn thành khảo sát và nhấn gửi

Bước 4: Ứng dụng gửi yêu cầu đến server

Bước 5: Server lưu thông tin tham gia sự kiện của người dùng và gửi phản hồi cho ứng dụng

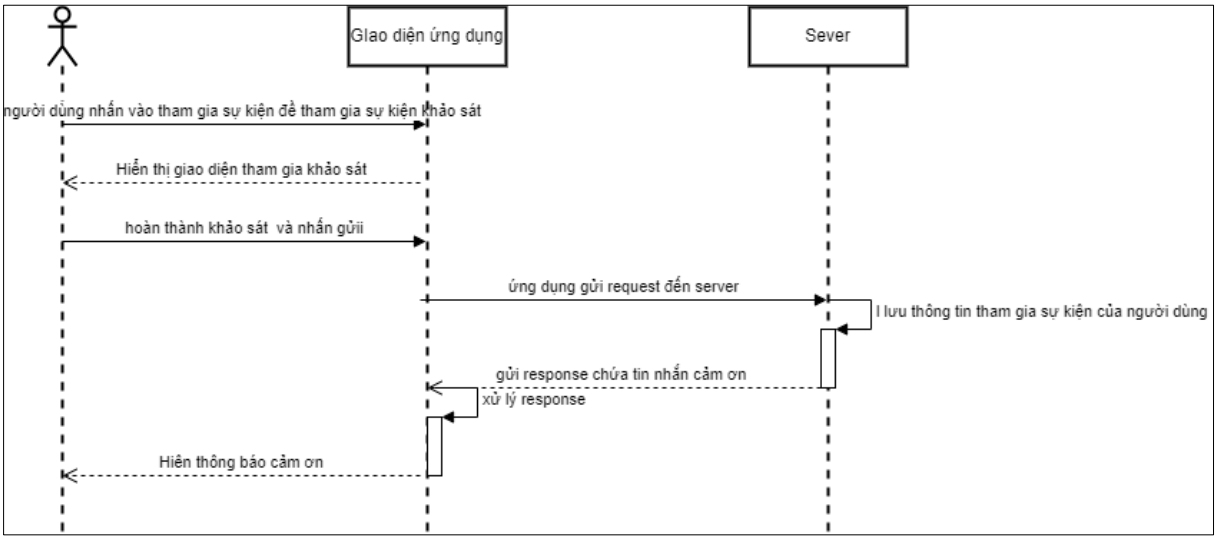
Bước 6: Ứng dụng xử lý phản hồi và hiện thông báo cảm ơn cho người dùng

Sơ đồ giữa tác nhân và usercase:



Sơ đồ 21: Sơ đồ user-case chức năng tham gia sự kiện khảo sát

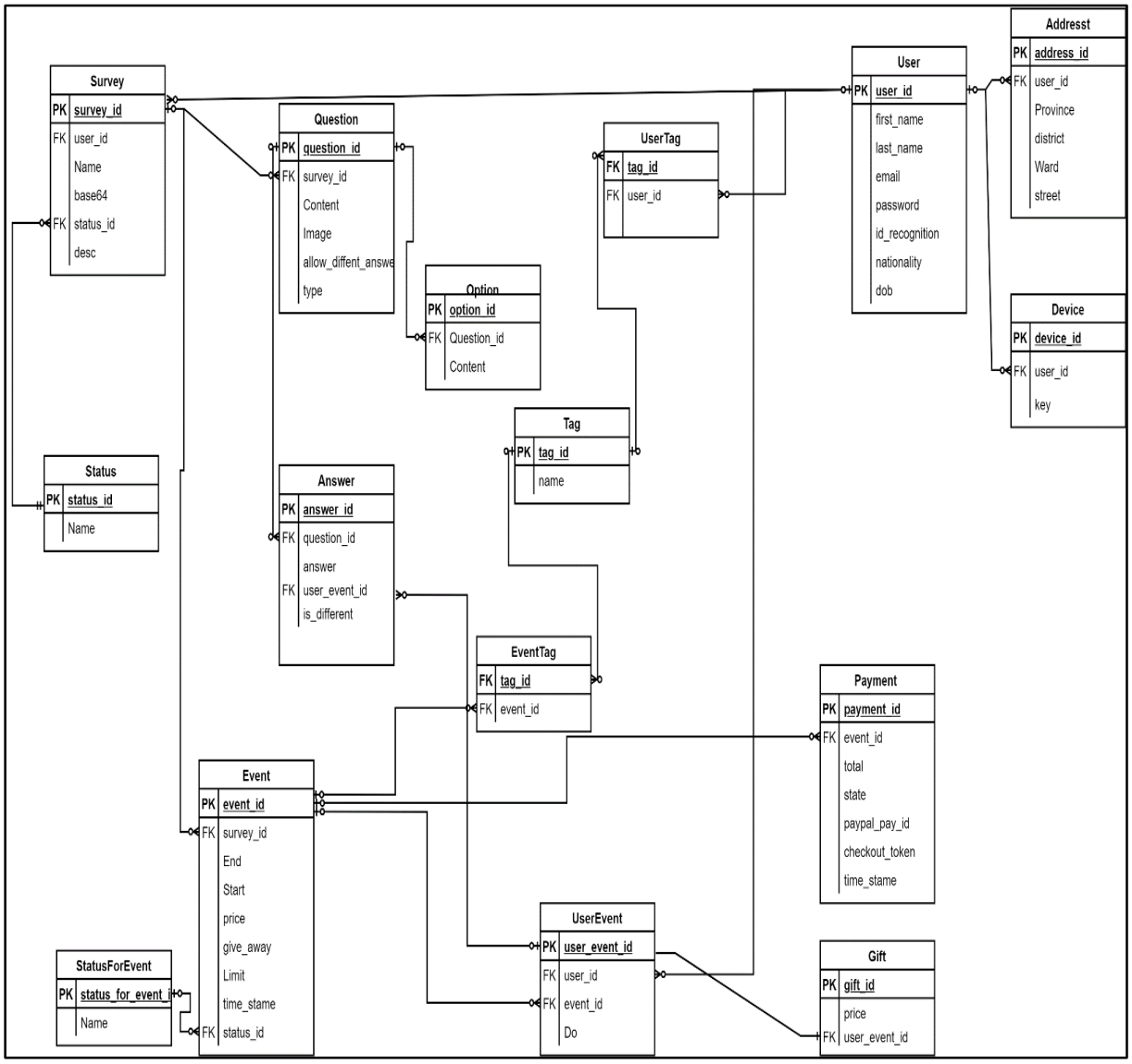
Sơ đồ tuần tự:



Sơ đồ 22: Sơ đồ tuần tự chức năng tham gia sự kiện khảo sát

3. XÂY DỰNG CƠ SỞ DỮ LIỆU

3.1. Sơ đồ cơ sở dữ liệu tổng quát



Sơ đồ 23: Sơ đồ cơ sở dữ liệu

3.2. Chi tiết các bảng

Lưu ý: Các trường **<u>In đậm gạch dưới</u>** – "Là khóa chính của bảng", ***<u>In đậm nghiêng gạch dưới</u>*** – "Là khóa ngoại liên kết tới các bảng dữ liệu khác", Viết thường – "Là các trường dữ liệu cần lưu trong các bảng".

- Quản lý người dùng

Bảng 5: Cơ sở dữ liệu quản lý người dùng

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	User	**user_id**	Integer	Mã người dùng
2		first_name	String	Họ người dùng
2		last_name	String	Tên người dùng
3		email	String	Email của người dùng
4		password	String	Mật khẩu của người dùng
5		id_recognition	String	Số CMT/CCCD của người dùng
6		nationality	String	Quốc tịch
7		dob	Datetime	Ngày sinh

- Quản lý phân loại người dùng

Bảng 6: Cơ sở dữ liệu quản lý phân loại người dùng

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	UserTag	***tag_id***	Integer	Mã loại người dùng
2		***user_id***	Integer	Mã người dùng

- Quản lý loại người dùng

Bảng 7: Cơ sở dữ liệu quản lý loại người dùng

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	Tag	**tag_id**	Integer	Mã loại người dùng
2		name	String	Tên loại người dùng

- Quản lý khảo sát

Bảng 8: Cơ sở dữ liệu quản lý khảo sát

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	Survey	**survey_id**	Integer	Mã khảo sát
2		***user_id***	Integer	Mã người dùng
3		Name	String	Tên sự kiện khảo sát

4		base64	String	Hình ảnh sự kiện khảo sát dưới định dạng chuỗi
5		***status_id***	Integer	Mã trạng thái của khảo sát
6		desc	String	Mô tả của sự kiện khảo sát

- Quản lý trạng thái của khảo sát

Bảng 9: Cơ sở dữ liệu quản lý trạng thái của khảo sát

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	Status	**status_id**	Integer	Mã trạng thái của khảo sát
2		Name	String	Tên trạng thái của khảo sát

- Quản lý câu hỏi

Bảng 10: Cơ sở dữ liệu quản lý câu hỏi

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	Question	**question_id**	Integer	Mã câu hỏi
2		***survey_id***	String	Mã khảo sát
3		Content	String	Nội dung của khảo sát
4		Image	String	Hình ảnh của câu hỏi dưới định dạng chuỗi
5		allow_diffent_answer	Bool	Câu hỏi có cho phép người dùng thêm ý kiến khác không
6		type	Integer	Loại câu hỏi

- Quản lý lựa chọn cho câu hòi

Bảng 11: Cơ sở dữ liệu quản lý lựa chọn cho câu hỏi

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	Option	**option_id**	Integer	Mã lựa chọn trả lời câu hỏi
2		***Question_id***	Integer	Mã câu hỏi

3		Content	String	Nội dung lựa chọn trả lời câu hỏi

- Quản lý sự kiện

Bảng 12: Cơ sở dữ liệu quản lý sự kiện

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	Event	**event_id**	Integer	Mã sự kiện khảo sát
2		***survey_id***	Integer	Mã khảo sát
3		End	Datetime	Thời gian kết thúc sự kiện khảo sát
4		Start	Datetime	Thời gian bắt đầu sự kiện khảo sát
5		price	float	Giá sự kiện khảo sát
6		give_away	float	Phần thưởng cho người may mắn
7		Limit	Integer	Giới hạn lượng người tham gia
8		time_stame	Datetime	Thời gian sự kiện được tạo
9		***status_id***	Integer	Trạng thái của sự kiện khảo sát

- Quản lý người loại người dùng tham gia khảo sát

Bảng 13: Cơ sở dữ liệu quản lý loại người dùng tham gia sự kiện

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	EventTag	***tag_id***	Integer	Mã loại người dùng
2		***event_id***	Integer	Mã sự kiện khảo sát

- Quản lý thiết bị người dùng

Bảng 14: Cơ sở dữ liệu quản lý thiết bị

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	Device	**device_id**	Integer	Mã thiết bị

2		***user_id***	Integer	Mã người dùng
3		key	String	Mã Firebase Cloud Messaging của thiết bị

- Quản lý trạng thái sự kiện

Bảng 15: Cơ sở dữ liệu quản lý trạng thái của sự kiện

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	StatusForEvent	**status_for_event_id**	Integer	Mã trạng thái sự kiện khảo sát
2		Name	String	Tên trạng thái sự kiện khảo sát

- Quản lý câu trả lời

Bảng 16: Cơ sở dữ liệu quản lý câu trả lời

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	Answer	**answer_id**	Integer	Mã câu trả lời
2		answer	String	Nội dung câu trả lời
3		***question_id***	Integer	Mã câu hỏi
4		***user_event_id***	Integer	Mã bảng người dùng tham gia sự kiện
5		is_different	Bool	Có phải câu trả lời khác những lựa chọn cho trước hay không

- Quản lý người dùng tham gia sự kiện

Bảng 17: Cơ sở dữ liệu quản lý người dùng tham gia sự kiện

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	UserEvent	**user_event_id**	Integer	Mã quản lý người dùng tham gia sự kiện
2		***user_id***	Integer	Mã người dùng

3		***event_id***	Integer	Mã sự kiện khảo sát
4		do	Bool	Đã thực hiện khảo sát hay chưa

- Quản lý phần thưởng

Bảng 18: Cơ sở dữ liệu quản lý phần thưởng

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	Gift	**gift_id**	Integer	Mã phần quà
2		price	Float	Giá của phần quà
3		***user_event_id***	Integer	Mã quản lý người dùng tham gia sự kiện

- Quản lý địa chỉ người dùng

Bảng 19: Cơ sở dữ liệu quản lý câu trả lời

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	Address	**address_id**	Integer	Mã địa chỉ
2		***user_id***	Integer	Mã người dùng
3		Province	String	Địa bàn tỉnh
4		district	String	Tên quận/huyện
5		Ward	String	Tên phường
6		street	String	Tên đường

- Quản lý thanh toán

Bảng 20: Cơ sở dữ liệu quản lý thanh toán

STT	Tên bảng	Tên cột	Kiểu	Mô Tả
1	Payment	**payment_id**	Integer	Mã thanh toán
2		***event_id***	String	Mã sự kiện khảo sát
3		total	float	Tổng tiền
4		state	bool	Trạng thái đã thanh toán hay chưa

5		Paypal_pay_id	String	Mã hóa đơn Paypal
6		checkout_token	String	Mã thực thi thanh toán
7		time_stame	Datetime	Thời gian thanh toán được tạo

CHƯƠNG IV: GIAO DIỆN HỆ THỐNG

Ứng dụng cho phép người dùng sử dụng ngôn ngữ tiếng Việt (Hình 8) và tiếng Anh (Hình 9)

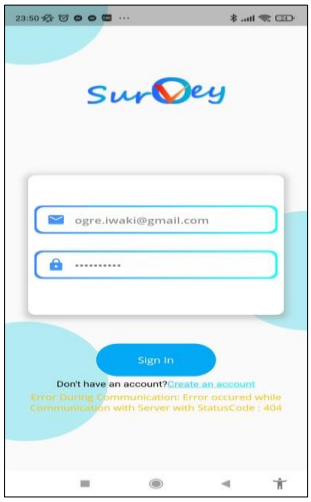
Hình 8 Giao diện tiếng Việt

Hình 9: Giao diện tiếng Anh

1. GIAO DIỆN ĐĂNG NHẬP

Giao diện cho phép người dùng đăng nhập vào hệ thống (Hình 10). Nếu quá trình đăng nhập thất bại giao diện hiện lỗi (Hình 11)

Hình 10: Giao diện đăng nhập



Hình 11: Giao diện đăng nhập lỗi

2. GIAO DIỆN TRANG CHỦ

Sau khi đăng nhập thành công ứng dụng sẽ hiện giao diện trang chủ có thanh điều hướng cho phép người dùng có thể dễ dàng di chuyển giữa các tab màn hình (Hình 12), trong đó:

1. Biểu tượng cho phép người dùng di chuyển đến tab tham gia sự kiện
2. Biểu tượng cho phép người dùng di chuyển đến tab khảo sát
3. Biểu tượng cho phép người dùng di chuyển đến tab sự kiện khảo sát
4. Biểu tượng cho phép người dùng di chuyển đến tab tài khoản

Hình 12: Giao diện trang chủ

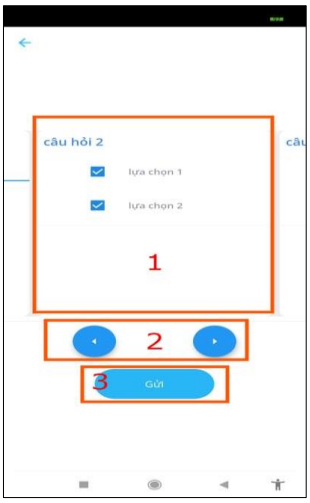
2.1. Giao diện tab tham gia sự kiện

Giao diện hiện danh sách những sự kiện khảo sát mà người dùng được mời tham gia (Hình 13). Người dùng có thể nhấn vào các sự kiện khảo sát để di chuyển sang màn hình thực hiện khảo sát (Hình 14), trong đó:

1. Thẻ câu hỏi cho phép người dùng điền thông tin
2. Các nút cho phép người dùng di chuyển

3. Nút cho phép người dùng gửi thông tin tham gia khảo sát

Hình 13 Giao diện tab tham gia khảo sát



Hình 14: Giao diện thực hiện khảo sát

2.2. Giao diện tab sự kiện khảo sát

Giao diện hiện danh sách các sự kiện khảo sát của người dùng (Hình 15), trong đó:

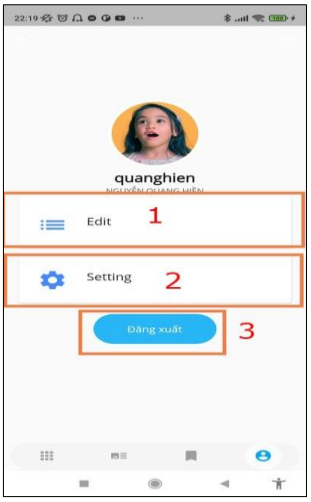
1. Danh sách các sự kiện khảo sát trong thời gian chờ hoặc chưa trả phí
2. Danh sách các sự kiện khảo sát đang diễn ra
3. Danh sách các sự kiện khảo sát đã hoàn thành
4. Nút thêm một sự kiện khảo sát mới

Hình 15: Giao diện tab sự kiện khảo sát

2.3. Giao diện tab tài khoản

Giao điện hiển thị thông tin cơ bản của người dùng (Hình 16), trong đó:

1. Thẻ Edit cho phép di chuyển tới màn hình thông tin chi tiết cho phép chỉnh sửa
2. Thẻ Setting cho phép di chuyển tới trang cài đặt
3. Nút đăng xuất cho phép người dùng đăng xuất khỏi ứng dụng



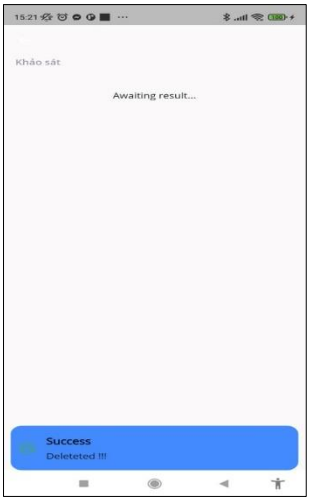
Hình 16: Giao diện tab thông tin tài khoản

2.4. Giao diện tab khảo sát

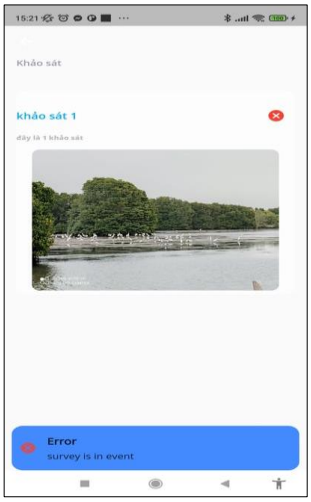
Hiển thị những khảo sát mà người dùng sở hữu (Hình 17), người dùng có thể nhấn vào các thẻ khảo sát để di chuyển tới màn hình chỉnh sửa khảo sát, trong đó:

1. Nút cho phép người dùng xóa khảo sát. Nếu xóa thành công hiện thông báo xóa thành công (Hình 18). Nếu xóa thất bại hiện thông báo lỗi (Hình 19).
2. Nút cho phép người dùng thêm mới khảo sát

Hình 19: Giao diện tab khảo sát



Hình 17: Giao diện thông báo xáo khảo sát thành công



Hình 18: Giao diện thông báo xóa khảo sát thất bại

3. GIAO DIỆN THÊM MỚI(CHỈNH SỬA) KHẢO SÁT

Giao diện cho phép người dùng nhập thông tin khảo sát (Hình 20), trong đó:

1. Chứa ô cho người dùng nập tên khảo sát, ô cho phép người dùng nhập mô tả của khảo sát và nút cho phép người dùng chọn hình ảnh đại diện cho khảo sát.
2. Ô cho phép người dùng nhập câu hỏi.
3. Nút cho phép người dùng thêm lựa chọn cho câu hỏi.
4. Nút cho phép người dùng lựa chọn loại của câu hỏi.
5. Nút cho phép người dùng xóa câu hỏi khỏi danh sách.
6. Nút cho phép câu hỏi có ý kiến khác hay không.
7. Ô cho phép người dùng nhập nội dung lựa chọn.
8. Nút cho phép xóa lựa chọn.
9. Nút cho phép lưu thông tin khảo sát.
10. Nút cho phép người dùng thêm một câu hỏi mới vào danh sách câu hỏi

.

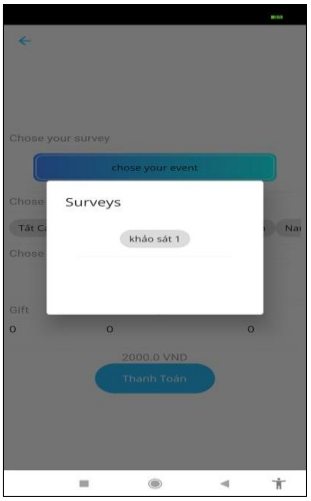
Hình 20: giao diện thêm (chỉnh sửa) khảo sát

4. GIAO DIỆN THÊM SỰ KIỆN KHẢO SÁT

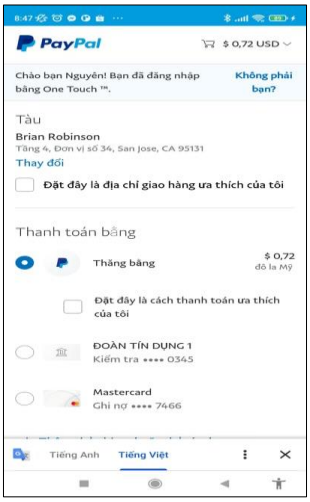
Giao diện cho phép người dùng nhập thông tin sự kiện (Hình 21) khảo sát, trong đó:

1. Nút cho phép chọn khảo sát cho sự kiện khảo sát khi nhấn chọn ứng dụng sẽ hiện một hộp thoại chứa danh sách các khảo sát của người dùng và cho phép người dùng chọn lựa (Hình 22).
2. Là các nút đại diện cho các phân loại người dùng.
3. Cho phép người dùng chọn lựa thời gian bắt đầu và thời gian kết thúc.
4. Ô cho phép người dùng nhập số tiền thưởng cho người tham gia làm khảo sát.
5. Ô cho phép người dùng nhập số tiền thưởng cho người may mắn.
6. Ô cho phép người dùng nhập giới hạn lượng người tham gia.
7. Nút cho phép người dùng thanh toán và lưu thông tin của sự kiện khảo sát, khi người dùng nhấn thanh toán sẽ chuyển sang trình duyệt cho phép người dùng thanh toán qua cổng thanh toán điện tử Paypal (Hình 23).

Hình 23: Giao diện thêm sự kiện khảo sát



Hình 21: Giao diện chọn khảo sát cho sự kiện

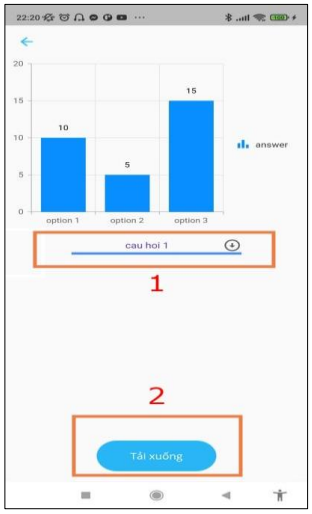


Hình 22: Giao diện thanh toán của Paypal

5. GIAO DIỆN THÔNG TIN SỰ KIỆN KHẢO SÁT

Giao diện cho phép người dùng xem biểu đồ cơ bản của sự kiện khảo sát (Hình 24), trong đó:

1. Thanh xổ cho phép người dùng lựa chọn câu hỏi muốn xem thống kê.
2. Nút cho phép người dùng tải thông tin sự kiện khảo sát



Hình 24 Giao diện thông tin sự kiện khảo sát

6. GIAO DIỆN ĐĂNG KÝ

Giao diện đăng ký cho phép người dùng đăng ký tài khoản (Hình 25), trong đó:

1. Ô cho phép người dùng nhập email.
2. Ô cho phép người dùng nhập tên đăng nhập.
3. Ô cho phép người dùng nhập tên mật khẩu.
4. Ô cho phép người dùng nhập lại mật khẩu.
5. Ô cho phép người dùng chọn hình ảnh.
6. Nút cho phép người dùng gửi thông tin đăng ký tới hệ thống.

Hình 25: Giao diện đăng ký

CHƯƠNG V: KẾT LUẬN

1. KẾT QUẢ ĐẠT ĐƯỢC

Trong quá trình thực hiện đề tài có thể ôn lại được những kiến thức đã được học trên lớp:

- Kiến thức cơ bản về Python và Flask.
- Phân tích thết kế hệ thống.
- Xây dựng cơ sở dữ liệu.
- Làm việc với framework Flask.

Ngoài ra tìm hiểu thêm được nhiều kiến thức mới về hai framework Flutter và Flask như:

- Xây dựng API với framework Flask.
- Xây dựng ứng dụng di động với framewor Flutter.
- Làm việc với API trong cả hai framework Flask và Flutter.

2. HƯỚNG PHÁT TRIỂN

Với việc sử dụng Flutter trong tương lai ứng dụng sẽ được phát triển trên những nền tảng khác như : IOS, WEB, Desktop.

Ứng dụng sẽ được phát triển thêm các tính năng như:

- Thêm phương thức thanh toán.
- Chia sẻ dữ liệu.
- Dánh giá người dùng xấu.

Do thời gian thực hiện còn hạn hẹp, kiến thức và kinh nghiệp còn nhiều hạn chế nên bài báo cáo này không tránh khỏi những sai sót. Rất mong nhận được những đóng góp ý kiến quý báu của thầy cô trường Đại học Bà Rịa – Vũng Tàu.

Một lần nữa em xin chân thành cảm ơn quý thầy cô trong khoa Công nghệ kỹ thuật- Nông nghiệp công nghệ cao và đặc biệt em xin gửi đến thầy Phan Ngọc Hoàng, người đã tận tình hướng dẫn, giúp đỡ em hoàn thành đề tài đồ án tốt nghiệp lời cảm ơn sâu sắc nhất.

TÀI LIỆU THAM KHẢO

[1] Vũ Việt Anh - "Tìm hiểu về ngôn ngữ Dart - Phần I" , 21/04/2019 lấy từ URL:

https://viblo.asia/p/tim-hieu-ve-ngon-ngu-dart-phan-i-bJzKmykwK9N

[2] VU NGOC TUAN – "Giới thiệu về Flutter", 20/03/2018 lấy từ URL: https://viblo.asia/p/gioi-thieu-ve-flutter-bWrZnNxrZxw

[3] Poman (2020) - 'HọcPython Để Làm Gì Cho Đời?', 15/12/2020 Lấy từ URL:

https://codelearn.io/sharing/hoc-python-de-lam-gi-cho-doi

[4] "7 lý do bạn nên chọn Flask Framework", 29-11-2021 lấy từ URL:

https://csc.edu.vn/lap-trinh-va-csdl/tin-tuc/kien-thuc-lap-trinh/7-ly-do-ban-nen-chon-Flask-Framework-4130

[5] FPT.AI Reader "Hướng dẫn sử dụng FPT.AI Reader - phần mềm ocr trích xuất thông tin từ ảnh chụp" – lấy từ URL:

https://fpt.ai/vi/huong-dan-su-dung-fptai-reader-phan-mem-ocr-trich-xuat-thong-tin-tu-anh-chup

[6] FPT.AI Reader – "Nhận diện CMT/CCCD" – lấy từ URL: https://docs.fpt.ai/docs/vi/vision/API/id-recognition

[7] Trang - 2020-07-30 "Paypal là gì?" – lấy từ URL:

https://lagi.wiki/paypal

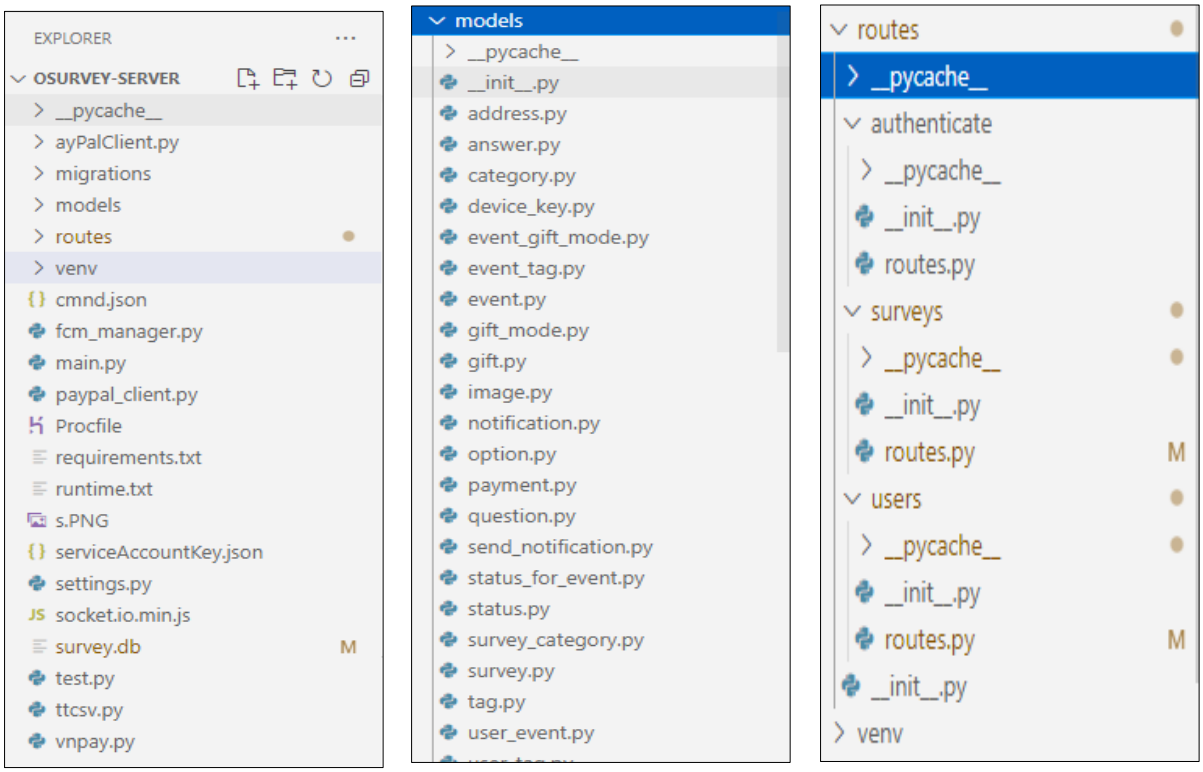
[8] Mắt Bão – "Firebase là gì? Giải pháp lập trình không cần Backend từ Google", 25/02/2021– lấy từ URL

https://wiki.matbao.net/firebase-la-gi-giai-phap-lap-trinh-khong-can-backend-tu-google/#firebase-la-gi

PHỤ LỤC

1. WEB SERVER

1.1. Sơ đồ cấu trúc



Hình 26:Cấu trúc thư mục của web server a, b, c

Bảng 21 :Bảng mô tả cấu trúc thư mục của Web server

STT	Tên thư mục	Mô Tả
1	venv	Thư mục quản lý môi trường, các thư viện
2	migration	Thư mục quản lý các file chuyển đổi dữ liệu
3	models	Thư mục quản lý các file đối tượng
4	routes	Thư mục quản lý cấu hình các route

1.2. Code xử lý

1.2.1. Thiết lập Flask App,CORS Origin

```
from flask import Flask
```

```
from flask_cors import CORS
app = Flask(__name__)
CORS(app)
```

Mẫu thiết lập Access-Control-Allow-Origin một route

```
@cross_origin(origin='*')
```

1.2.2. Cấu hình Firebase admin

- File JSON chứa service account key từ Firebase

```
{
  "type": "service_account",
  "project_id": "esurvey-ee4c3",
  "private_key_id": "fe590e2c41d337bea1c84edab045d2f7a329404c",
  "private_key": "-----BEGIN PRIVATE KEY-----
\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDNUmkJpLA0JzHY\nKtIFfHbkrc2lj
CzrJYGaYzcNbRkE0JvZiGwXVJE2Pi7MJKL7twOZ0t7/CmWaEuiv\n5cVkG3l6oAJJBf/duZrdjmZ91vBJ
sfZmFDQbuvt46Pn+naZzg5MGz8YOHSOaNWoK\nQb7mDVWPGvKgxleunVFoQ5lwovhUaSBUJhwxEWKHYNn
X2BDPqz2NiyXTtHD3sluT\nPM06dkPfokZrCB3hP9LlervctP/UvtMZcQfLWCpn9eOaACaGXWf/TJ81H3
Vu7Kau\npl06nQtLiZ/fVAn1TO5CAKcY7dRimwwSQGRBphCUUNTeyfTNLH9iRzQTkeaFyIpR\nxP/pOoq
pAgMBAAECggEAVacA2XyB4yfMo9RtK8u533jf1qVM4cWSNOEZyLiP3NHX\nih5elTb6oum65P+NnDtDPc
rZqwRf0S/9MM/GEhZiDXUtrt/ZVtpEUBPFDs75C405\ncGkSOdU9bbmm/IAwvXSQg6Fuilm+jeildWTq6
ZZ7cEj6GXbUgivvl3LH0lSyCqfX\nAZbVVDA5pyILONUU+XtGu7sY2r7KdbdSNj/4ltxiA7E0XWqAf+sE
fbgjO/572O5l\nqmG/Ch/b/DzmbX0ODHZ3y1cbD3ZObEIb5Y9nrIaybzLSYgj1c7YqAn9eeQ4Hq5Au\nv
rn5ubFxgOMfhclZrlOxHth4tX+G5U98ZbmUVe0zLwKBgQDnUhagnbdC5m+5eAXE\niwxCp2rzoF9JzIwa
rwHCSQL5HqnIyOfc8C/ahlwwhsoYTXrq5BYxrbMVsCkZjYKE\nGdWAeoMYDx7u+gEojJhUNfKJudTeQmX
+aW8pGgMEubTaWfYwDKuEU5qq2WjXMqQs\nrXC/TAMMm1lVuue/ztBP2shy+wKBgQDjOjwtNwU28hrfEp
6IRAArRGuT/gJq8XiI\niZtMgBMOrOIl+TgqLw8+0BF33zFTbqeyD1pYdKKMCNbrVVAPkBgeHHF8fLiNS
XZk\nmxs//RxXbIeVuMimsBpWYbNcgU+tTnAhHO7B+nWSLTtL2LOSjpi6/iT1V1Ls7uf8\nMsFuFrynqw
KBgBwpr12Qth4sBhAzn6glMbHvxiKxNMi8YZZiFjm8P+Nqq0spwqjw\niTL2xsSVtIcu77cnW9hiiHosf
2SACRLiPk8tG0bTmHWJ9JgmPemKw+Omv1bsCJTn\n60O6ygFKRs2KxGFnOX6b2ynP3GbO4JQb7a0sqPZg
94hrgAmnOU9vmJX/AoGBAKf3\ne1Y9DrjAHMb7H39BLRLOv+sk4Cqnlt7vQYI6RlwZxg4l/LOKF+3pppA
Cx5aR/Jpu\nKO2sQh/bxOsvJEBs1rcjdWx9UXr3a/IQigMyGgox6sPtVT2Kfd2O2SQvXnOQwOhp\nV8DC
YLiOgP+fD5btm87WYQGO7HjCXXAbIMs9OXWDAoGACGCU0JCEyjKf6ErFPnNx\npwXrsqVAILwaSCXUu5E
BU+aUgoM/e0WKwudiur4EXxEXi4agys/fZJBHXNxfCvGy\n4vIg7QQ2mY/48VSZvRRi78NEsEgfWiHak3
aOsgxyydlYKwJWlJM3GXS64AcM34Av\nwkpCt+DTKgCS0aPUbVqAoj8=\n-----END PRIVATE KEY---
--\n",
  "ứng dụng_email": "firebase-adminsdk-styad@esurvey-
ee4c3.iam.gserviceaccount.com",
  "ứng dụng_id": "112825693966029201082",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleAPIs.com/token",
  "auth_provider_x509_cert_url": "https://www.googleAPIs.com/oauth2/v1/certs",
  "ứng dụng_x509_cert_url":
"https://www.googleAPIs.com/robot/v1/metadata/x509/firebase-adminsdk-
styad%40esurvey-ee4c3.iam.gserviceaccount.com"
```

```
}
```

- Kết nối với firebase thông qua service account key và Firebase admin SDK

```
cred=credentials.Certificate("./serviceAccountkey.json")
default_app = firebase_admin.initialize_app(cred)
```

- Hàm chức năng đẩy thông báo

```
def sendPush(title,msg,re_tokens,dataObj=None):
    message=messaging.MulticastMessage(
        notification=messaging.Notification(
            title=title,
            body=msg
        ),
        data=dataObj,
        tokens=re_token
    )
    response=messaging.send_multicast(message)
    print(response)
```

- Mẫu sử dụng hàm:

```
fcm_manager.sendPush(title="Wellcome to oSurvey",msg="Hello",re_token=
['fnZaLKjcSA6B8GshGewNM1:APA91bEwxgS5KYYnueFT-
2FjjdK2EYawGkYmHplynQyYUWCyCK15jFvvoNCTj_1lqI1s9cY6R4EF6CTr5_ansgd6D7XtHLNPTn3Ej4
6otmFSnGJvftoFRrqti45e1ptK4Tuu-J4X08ti'])
```

1.2.3. Cấu hình kết nối với PAYPAL API

- Hàm xác thực tài khoản ứng dụng để lấy access token

```
def get_access_token():
    headers={
                        'Content-Type': 'application/x-www-form-urlencoded'
                }
    data={'grant_type': 'ứng dụng_credentials'}
    response = requests.post("https://API.sandbox.paypal.com/v1/oauth2/token",
    auth=('AWclxyD8OVZu-UlkCAOMgFAcep_rbqGL5jhlazEw-HiXqFMpO6P6mOqLyByoC-
BnwmkqZMnQlDT0itxE',
    'ELcrCDJE9TV6gxCRZnS0sWP7116Fa444UnqjBUqMRdzco713VJ9hiI2C67ndfU3GBUz1kUNfLMYO
pcvL'), data=data, headers=headers)

    return response.json()["access_token"]
```

1.2.4. Cấu hình kết nối với FPT.AI Reader - Vietnamese ID Card Recognition

- Url: https://API.fpt.ai/vision/idr/vnm
- Thiết lập header với API key được lấy từ FPT console

```
headers = {
    'API-key': 'wEGzmFlGFGJZVhRiWmbIp77SFu5uvl7S'
}
```

- Method:Post

```
requests.post(url, files=files, headers=headers)
```

- Với file từ định dạng base64 ảnh chứng minh thư của người dùng sau khi gửi yêu cầu đến sever thì được chuyển thành định dạng byte array

```
data= requesJson['image64']
im = Image.open(BytesIO(base64.b64decode(data)))
im.save(byteIO, format='PNG')

byteArr = byteIO.getvalue()

files = {'image': byteArr}
```

- Response có dạng json

```
{
    "errorCode": 0,
    "errorMessage": "",
    "data": [
      {
        "id": "030174005711",
        "id_prob": "98.72",
        "name": "XXXXXXXXXX",
        "name_prob": "89.15",
        "dob": "20/10/1974",
        "dob_prob": "99.89",
        "sex": "NỮ",
        "sex_prob": "93.89",
        "nationality": "VIỆT NAM",
        "nationality_prob": "72.06",
        "home": "KIM TÂN, KIM THÀNH, HẢI DƯƠNG",
```

```
        "home_prob": "99.49",
        "address": "KHU PHỐ 5 PHƯỚC TRUNG TP BÀ RA, BÀ RỊA, VŨNG TÀU",
        "address_prob": "58.76",
        "doe": "20/10/2034",
        "doe_prob": "95.13",
        "overall_score": "71.86",
        "address_entities": {
          "province": "BÀ RỊA VŨNG TÀU",
          "district": "BÀ RỊA",
          "ward": "PHƯỚC TRUNG",
          "street": "KHU PHỐ 5"
        },
        "type_new": "cccd_chip_front",
        "type": "chip_front"
      }
    ] }
```

1.2.5. Xây dựng cơ sở dữ liệu

- Thiết lập đường dẫn CSDL

```
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///survey.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
migrate = Migrate(app, db)
```

- Thiết lập các bảng,ràng buộc và mối quan hệ giữa các bảng trong CSDl

```
import hashlib
import hmac
from  sqlalchemy import  Sequence
from werkzeug.security import generate_password_hash, check_password_hash
from settings import db

class User(db.Model):
    user_id = db.Column(db.Integer,Sequence('user_id_seq'),primary_key=True)
    first_name = db.Column(db.String(64), nullable=False)
    last_name =  db.Column(db.String(64),nullable=False)
    user_name=db.Column(db.String(128),unique=True,nullable=False)
    email =  db.Column(db.String(128),unique=True,nullable=False)
    password =  db.Column(db.String(128), index=True,nullable=False)
    id_recognition=db.Column(db.String(),unique=True)
    sex=db.Column(db.String(64),nullable=False,default="N/A")
    nationality=db.Column(db.String(64),nullable=False,default="N/A")
```

```
    home=db.Column(db.String(64),nullable=False,default="N/A")
    address_entities=db.relationship("Address", back_populates="user")
    dob=db.Column(db.String(64),nullable=False,default="N/A")
    type=db.Column(db.String(64),nullable=False,default="N/A")
    do_surveys =db.relationship('Event', secondary='user_event')
    own_events=db.relationship("Event", back_populates="user")
    owl_surveys = db.relationship("Survey", back_populates="user")
    tags=db.relationship('Tag', secondary='user_tag')
    notifications=db.relationship('Notification', secondary='send_notification')
    payments=db.relationship("Payment", back_populates="user")


    device_keys=db.relationship("DeviceKey", back_populates="user")
    def __repr__(self):
        return  '<User full name {} {} ,email
{}>'.format(self.first_name,self.last_name,self.email)
    def set_password(self, password):
        self.password = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password, password)
```

```
from settings import db
from  sqlalchemy import  Sequence
class DeviceKey(db.Model):
    device_id=db.Column(db.Integer,Sequence('device_id_seq'),primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.user_id'))
    user = db.relationship("User", back_populates="device_keys")
    key=db.Column(db.String(),nullable=False)
```

```
from settings import db
from  sqlalchemy import  Sequence
class Tag(db.Model):
    tag_id=db.Column(db.Integer,Sequence('tag_id_seq'),primary_key=True)
    name=db.Column(db.String(),nullable=False,default="N/A")
    users=db.relationship('User', secondary='user_tag')
    events=db.relationship('Event', secondary='event_tag')
```

```
from settings import db
```

```
class UserTag(db.Model):
    tag_id = db.Column(db.Integer, db.ForeignKey('tag.tag_id'),primary_key=True)
    user_id = db.Column(db.Integer,
db.ForeignKey('user.user_id'),primary_key=True)
```

```
import datetime
from settings import db
from  sqlalchemy import  Sequence
class Event(db.Model):
    event_id=db.Column(db.Integer,Sequence('question_id_seq'),primary_key=True)
    survey_id = db.Column(db.Integer, db.ForeignKey('survey.survey_id'))
    survey = db.relationship("Survey", back_populates="events")
    users =db.relationship('User', secondary='user_event')
    start=db.Column(db.DateTime(timezone=True))
    end=db.Column(db.DateTime(timezone=True))
    user_id=db.Column(db.Integer, db.ForeignKey('user.user_id'))
    user=db.relationship("User", back_populates="own_events")
    price = db.Column(db.Integer,default=0)
    give_away = db.Column(db.Integer,default=0)
    limit=db.Column(db.Integer)
    time_stame=db.Column(db.DateTime(timezone=True),default=datetime.datetime.utc
now())
    tags=db.relationship('Tag', secondary='event_tag')
    status_id= db.Column(db.Integer,
db.ForeignKey('status_for_event.status_for_event_id'),default=2)
    status=db.relationship("StatusForEvent", back_populates="events")
    gift_modes=db.relationship('GiftMode', secondary='event_gift_mode')
    payment=db.relationship("Payment", back_populates="event")
```

```
from settings import db
from  sqlalchemy import  Sequence
class UserEvent(db.Model):
    user_event_id=db.Column(db.Integer,Sequence('user_event_id_seq'),primary_key=
True)
    event_id = db.Column(db.Integer, db.ForeignKey('event.event_id'))
    user_id = db.Column(db.Integer, db.ForeignKey('user.user_id'))
    answers=db.relationship("Answer", back_populates="user_event")
    do=db.Column(db.Boolean,nullable=False,default=False)
    gift=db.relationship("Gift", back_populates="user_event")
```

```
from settings import db
from  sqlalchemy import  Sequence
class StatusForEvent(db.Model):
    status_for_event_id=db.Column(db.Integer,Sequence('status_for_event_id_seq'),
primary_key=True)
    name=db.Column(db.String(),nullable=False,default="N/A")
    events=db.relationship("Event", back_populates="status")
```

```
from settings import db
from  sqlalchemy import  Sequence
class Survey(db.Model):
    survey_id=db.Column(db.Integer,Sequence('survey_id_seq'),primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.user_id'))
    user = db.relationship("User", back_populates="owl_surveys")
    name=db.Column(db.String(300),nullable=False,default="N/A")
    questions=db.relationship("Question", back_populates="survey")
    status_id=db.Column(db.Integer, db.ForeignKey('status.status_id'),default=2)
    events=db.relationship("Event", back_populates="survey")
    base64=db.Column(db.String(),nullable=False,default="N/A")
    desc=db.Column(db.String(),nullable=False,default="N/A")
    status=db.relationship("Status", back_populates="surveys")
    categories=db.relationship('Category', secondary='survey_category')
```

```
from settings import db
from  sqlalchemy import  Sequence
class Status(db.Model):
    status_id=db.Column(db.Integer,Sequence('status_id_seq'),primary_key=True)
    name=db.Column(db.String(),nullable=False,default="N/A")
    surveys=db.relationship("Survey", back_populates="status")
```

```
from settings import db
from  sqlalchemy import  Sequence
class Question(db.Model):
    question_id=db.Column(db.Integer,Sequence('question_id_seq'),primary_key=True
)
    survey_id = db.Column(db.Integer, db.ForeignKey('survey.survey_id'))
    survey = db.relationship("Survey", back_populates="questions")
    content=db.Column(db.String(3000),nullable=False,default="N/A")
```

```
    options=db.relationship("Option", back_populates="question")
    images = db.relationship("Image", back_populates="question")
    allow_diffent_answer=db.Column(db.Boolean,nullable=False,default=False)
    type=db.Column(db.Integer,nullable=False,default=0)
```

```
from settings import db
from  sqlalchemy import  Sequence
class Option(db.Model):
    option_id=db.Column(db.Integer,Sequence('option_id_seq'),primary_key=True)
    question_id = db.Column(db.Integer, db.ForeignKey('question.question_id'))
    question = db.relationship("Question", back_populates="options")
    content=db.Column(db.String(3000),nullable=True)
    type=db.Column(db.String(3000),nullable=False,default="N/A")
```

```
from settings import db
from  sqlalchemy import  Sequence
class Gift(db.Model):
    gift_id=db.Column(db.Integer,Sequence('gift_id_seq'),primary_key=True)
    price=db.Column(db.Float,nullable=True,default=0.0)
    user_event_id=db.Column(db.Integer,
db.ForeignKey('user_event.user_event_id'))
    user_event=db.relationship("UserEvent", back_populates="gift")
```

```
from settings import db
from  sqlalchemy import  Sequence
class Answer(db.Model):
    answer_id=db.Column(db.Integer,Sequence('answer_id_seq'),primary_key=True)
    answer=db.Column(db.String(),nullable=False,default="N/A")
    base64=db.Column(db.String(),nullable=False,default="N/A")
    question_id = db.Column(db.Integer, db.ForeignKey('question.question_id'))
    user_event_id = db.Column(db.Integer,
db.ForeignKey('user_event.user_event_id'))
    user_event=db.relationship("UserEvent", back_populates="answers")
    is_different=db.Column(db.Boolean,nullable=False,default=False)
```

```
import datetime
from settings import db
```

```
from  sqlalchemy import  Sequence
class Payment(db.Model):
    payment_id=db.Column(db.Integer,Sequence('payment_id_seq'),primary_key=True)
    event_id=db.Column(db.Integer, db.ForeignKey('event.event_id'))
    event=db.relationship("Event", back_populates="payment")
    user_id=db.Column(db.Integer, db.ForeignKey('user.user_id'))
    user=db.relationship("User", back_populates="payments")
    total=db.Column(db.Float)
    state=db.Column(db.Boolean,nullable=False,default=False)
    paypal_pay_id=db.Column(db.String(),unique=True,nullable=False)
    checkout_token=db.Column(db.String(),unique=True,nullable=False)
    time_stame=db.Column(db.DateTime(timezone=True),default=datetime.datetime.utc
now())
```

1.2.6. Thiết lập file API controller có nhiệm vụ quả lý việc gọi API

```
class APIController {
  final String baseUrl = Env.baseUrl;
  Future<Map<String, dynamic>> signIn(
      String email, String password, String? key) async {
    APIProvider APIProvider = APIProvider();
    try {
      final String data = json
          .encode({"email": email, "password": password, "device_key": key});
      return await APIProvider.post(baseUrl + '/signIn', data);
    } on Error catch (e) {
      throw Exception('Failed to load post ' + e.toString());
    }
  }
```

1.2.7. Cấu hình các route

- Route đăng ký

```
@app.route('/signUp',methods=["POST"])
@cross_origin(origin='*')
def signup():
    if request.method=="POST":
        url = 'https://API.fpt.ai/vision/idr/vnm'
        requesJson=request.get_json()

        if 'image64' in requesJson and "email" in requesJson and 'password' in
requesJson and "user_name" in requesJson :
```

```
            data= requesJson['image64']
            tokens=[]
            tokens.append(requesJson['device_key'])
            if data:

                im = Image.open(BytesIO(base64.b64decode(data)))

                byteIO = io.BytesIO()
                im.save(byteIO, format='PNG')


                byteArr = byteIO.getvalue()


                files = {'image': byteArr}
                headers = {
                        'API-key': 'p9ZIXNTiIt03kJLIB2dICFV3QNb9tiMj'
                    }

                response = requests.post(url, files=files, headers=headers)
                values= response.json()



                if "data" in values:
                    if values["data"]:
                        name=values["data"][0]['name'].split()
                        first_name=name[0]

                        name=name[1:]
                        last_name = ' '.join([str(elem) for elem in name])
                        id_recognition=values["data"][0]['id']
                        province=values["data"][0]["address_entities"]['province'
]
                        district=values["data"][0]["address_entities"]['district'
]
                        ward=values["data"][0]["address_entities"]['ward']
                        street=values["data"][0]["address_entities"]['street']
                        dob=values["data"][0]['dob']
                        nationality=values["data"][0]['nationality']
                        sex=values["data"][0]['sex']
                        email=requesJson['email']
                        password=requesJson['password']
                        user_name=requesJson['user_name']
                        n_user=models.User(first_name=first_name,
```

```
                        user_name=user_name,
                        last_name=last_name,
                        id_recognition=id_recognition,
                        dob=dob,nationality=nationality,
                        sex=sex,email=email)
                        n_user.set_password(password)
                        if
(db.session.query(models.User).filter_by(email=n_user.email).count() == 0 and
db.session.query(models.User).filter_by(id_recognition=id_recognition).count()==0
):
                            db.session.add(n_user)
                            db.session.commit()
                            n_address=models.Address(province=province,street=str
eet,ward=ward,district=district,user=n_user)
                            db.session.add(n_address)
                            db.session.commit()
                            n_device=models.DeviceKey(key=requesJson['device_key'
],user=n_user)
                            db.session.add(n_device)
                            db.session.commit()
                            datetime.datetime.now().year

                            t=n_user.dob.replace("/", "-")
                            n_year=datetime.datetime.strptime(t, '%d-%m-%Y')

                            if (datetime.datetime.now().year -
                            n_year.year)>17  and (datetime.datetime.now().year -
                            n_year.year)<40:
                                ntag=
models.UserTag(user_id=n_user.user_id,tag_id=1)
                                db.session.add(ntag)
                                db.session.commit()
                            if (datetime.datetime.now().year -
                            n_year.year)>=40  and (datetime.datetime.now().year -
                            n_year.year)<60:
                                ntag=
models.UserTag(user_id=n_user.user_id,tag_id=2)
                                db.session.add(ntag)
                                db.session.commit()
                            if (datetime.datetime.now().year -
                            n_year.year)>=60:
                                ntag=
models.UserTag(user_id=n_user.user_id,tag_id=3)
                                db.session.add(ntag)
                                db.session.commit()
```

```
                            if n_user.sex=="Nam":
                                ntag=
models.UserTag(user_id=n_user.user_id,tag_id=4)
                                db.session.add(ntag)
                                db.session.commit()
                            if n_user.sex=="Nữ":
                                ntag=
models.UserTag(user_id=n_user.user_id,tag_id=5)
                                db.session.add(ntag)
                                db.session.commit()
                            fcm_manager.sendPush(title="Wellcome to
oSurvey",msg="Hello",re_token= tokens)
                            return json.dumps({"success ":"account is created
!"}),200
                        else:
                            return json.dumps({"error":'Email or Id recognition
is alrealy exsits!'}),401
                    else:

                        if values['errorCode']==3:
                            return json.dumps({"error":"Unable to find ID card in
the image"}),200

                else:
                    return json.dumps({"error":"Invalid Id recognition"}),200
                # im.save("ddd"+".PNG", format='PNG')
                # with open(response.json()['data'][0]['id']+".json", 'w',
encoding='utf-8') as f:
                #                 json.dump(response.json(), f,
ensure_ascii=False, indent=4)

            else: return json.dumps({"error":"Invalid value image64 null"}),200
        else:
            return json.dumps({"error":"Invalid values"}),200
```

- Route đăng nhập

```
@app.route('/signIn',methods=["POST"])
@cross_origin(origin='*')
def signin():
    if request.method=="POST":
        s=request.get_json()
        if s:
```

```
            user=models.User.query.filter_by(email=s["email"]).first()
            if user:
                if user.check_password(s['password']):


                    token=jwt.encode({"email":user.email,"exp":datetime.datetime.
utcnow()+datetime.timedelta(minutes=30)},app.config['SECRET_KEY'],
algorithm="HS256")

                    key=models.DeviceKey.query.filter_by(user_id=user.user_id,key
=s['device_key']).first()
                    if key==None:
                        n_key=models.DeviceKey(key=s['device_key'],user=user)
                        db.session.add(n_key)
                        db.session.commit()
                    user=models.User.query.filter_by(email=s["email"]).first()
                    tokens=[i.key for i in user.device_keys]
                    fcm_manager.sendPush(title="Wellcome to
oSurvey",msg="Hello",re_token= tokens)
                    return jsonify({"token":token})
                else:
                    return make_response('Could not verify password
error',401,{'WWW-Authenticate':'Basic realm="Login required!"'})
            else:
                return make_response('Could not verify user error',401,{'WWW-
Authenticate':'Basic realm="Login required!"'})

    else:
        return make_response('Could not verify user error',401,{'WWW-
Authenticate':'Basic realm="Login required!"'})
```

- Route khảo sát của người dùng

```
@app.route('/ownSurvey',methods=["POST"])
@cross_origin(origin='*')
@token_required
def ownSurvey(current_user):
    if current_user:
        own_surveys=models.Survey.query.filter_by(user_id=current_user.user_id).a
ll()
        ows=None
        if own_surveys:
            ows={"surveys":[
                {
```

```
                    "name":s.name,
                    "id":s.survey_id,
                    "base64":str(s.base64),
                    "description":s.desc,
                    "questions":[
                {
                    "content":i.content,
                    "options":[
                        {
                            "content":o.content,
                            "type":o.type
                            } for o in i.options]
            } for i in s.questions]} for s in own_surveys]}
        else:
            ows={"surveys":[]}
    return jsonify(ows)
```

- Route sự kiện khảo sát của người dùng

```
@app.route('/ownEvent',methods=["POST"])
@cross_origin(origin='*')
@token_required
def ownEvent(current_user):
    if current_user:
        ows=None
        if current_user.own_events:
            ows={"events":
            [ {
                    "id":e.event_id,
                    "start":e.start.date(),
                    "end":e.end.date(),
                    "status":e.status_id,
                    "payment":
                    [{
                            "paypal_payment_id":p.paypal_pay_id,
                            "state":p.state,
                            "checkout_token":p.checkout_token}
                        for p in e.payment],
                    "time_stame":e.time_stame,
                     "survey":
                {
                    "survey_id":e.survey_id,
                    "name":e.survey.name,
                    "description":e.survey.desc,
```

```
                    "questions":[
                        {
                            "id":q.question_id,
                            "type":q.type,
                            "allow_diffrent_answer":q.allow_diffent_answer,
                            "content":q.content,
                            "options":[
                                {
                                    "id":o.option_id,
                                    "content":o.content
                                }  for o in q.options
                                    ] }  for q in e.survey.questions
                        ]},"tags":[{"id":t.tag_id,"name":t.name} for t in
e.tags]} for e in current_user.own_events]}
        else:
            ows={"events":[]
    return jsonify(ows)
```

- Route sự kiện khảo sát người dùng được mời tham gia

```
@app.route('/joinEvent',methods=["POST"])
@cross_origin(origin='*')
@token_required
def joinEvent(current_user):
    if current_user:
        ows=None
        if current_user.do_surveys:
            ows={"events":[
            {
            "id":e.event_id,
            'may_get':e.price/e.limit,
            'give_away':e.give_away,
            "payment":[
                {
                    "paypal_payment_id":p.paypal_pay_id,
                    "state":p.state,
                    "checkout_token":p.checkout_token
                }for p in e.payment
                ],
            "time_stame":e.time_stame,
            "survey":{
                    "survey_id":e.survey_id,
                    "name":e.survey.name,
                    "description":e.survey.desc,
                    "base64":e.survey.base64,
```

```
                    "questions":[{
                            "id":q.question_id,
                            "type":q.type,
                            "allow_diffrent_answer":q.allow_diffent_answer,
                            "content":q.content,
                            "options":[{
                                    "id":o.option_id,
                                    "content":o.content,
                                    "check":False
                                } for o in q.options]}for q in e.survey.questions
                        ]},
            "tags":[    {
                    "id":t.tag_id,
                    "name":t.name
                }for t in e.tags]} for e in current_user.do_surveys]}
        else:
            ows={"events":[]}
    return jsonify(ows)
```

- Route tạo khảo sát

```
@app.route('/createSurvey', methods=["POST"])
@cross_origin(origin='*')
@token_required
def createSurvey(current_user):
    if request.method=="POST":
        if "name" in request.get_json() and "questions"in request.get_json()  and
"description"in request.get_json():
            res=request.get_json()
            if res["name"]:
                byteArr=None
                if res["base64"] !="N/A":
                      data= res["base64"]
                      if data:
                          im = Image.open(BytesIO(base64.b64decode(data)))
                          byteIO = io.BytesIO()
                          im.save(byteIO, format='PNG')
                          byteArr = byteIO.getvalue()
                n_survey=models.Survey(name=res["name"],desc=res["description"],u
ser=current_user)
                n_survey.base64=res["base64"]
                n_survey.status_id=2;
                db.session.add(n_survey)
                db.session.commit()
                questions=res["questions"]
```

```
                for question in questions:
                    typeq=question["type"]
                    content=question["content"]
                    allow_diffrent_answer=question["allow_diffrent_answer"]
                    n_question=models.Question(content=content,survey=n_survey,al
low_diffent_answer=allow_diffrent_answer ,type=typeq)
                    db.session.add(n_question)
                    db.session.commit()
                    for option in question["options"]:

                        n_option=models.Option(content=option["content"],question
=n_question)
                        db.session.add(n_option)
                        db.session.commit()
            else:
                return json.dumps({"error":"data not found"}),200
            return  json.dumps({"message":"create survey completed"}),200
        else:
            return json.dumps({"error":"data not found"}),200
```

- Route sửa khảo sát

```
@app.route('/editSurvey', methods=["POST"])
@cross_origin(origin='*')
@token_required
def editSurveys(current_user):
    if current_user:
        if request.method=="POST":
            if "id" in request.get_json():
                res=request.get_json()
                survey=models.Survey.query.filter_by(survey_id=res["id"]).first()
                if survey.status_id==2:
                    if res["name"]:
                        survey.name=res["name"]
                    if res["description"]:
                        survey.desc=res["description"]
                    db.session.commit()
                    db.session.delete(survey.questions)
                    db.session.commit();
                    questions=res["questions"]
                    for question in questions:
                        contentt=question["content"]
                        n_question=models.Question(content=contentt,survey=survey
)
                        db.session.add(n_question)
```

```
                        db.session.commit()
                        for option in question["options"]:

                            n_option=models.Option(content=option['content'],ques
tion=question)
                            db.session.add(n_option)
                            db.session.commit()
                    return json.dumps({"message":"survey created"}),200
                else:
                    return json.dumps({"error":"survey is in event"}),200
            else:
                return  json.dumps({"error":"required id survey"}),200
    else:
        return  json.dumps({"error":"required user"}),401
```

- Route xóa khảo sát

```
@app.route('/deleteSurvey', methods=["POST"])
@cross_origin(origin='*')
@token_required
def deleteSurvey(current_user):
    if request.method=="POST":
        if "id" in request.get_json():
            res=request.get_json()

            survey=models.Survey.query.filter_by(survey_id=res["id"]).first()
            print(survey.status_id)
            if survey.status_id==2:

                db.session.delete(survey)
                db.session.commit()
                return jsonify({"message":"delete complete"}),200
            else:
                return jsonify({"error":"survey is in event"}),200
        else:
            return jsonify({"error":"no survey"}),200
    return jsonify({"message":"delete complete"}),200
```

- Route tạo sự kiện khảo sát

```
@app.route('/createEvent', methods=["POST"])
@cross_origin(origin='*')
@token_required
def createEvent(current_user):
    if request.method=="POST":
```

```
        if "start" in request.get_json() and "end" in request.get_json() and
"survey" in request.get_json() and "limit" in request.get_json() and "tags" in
request.get_json():
            res=request.get_json()
            survey=models.Survey.query.filter_by(survey_id=res["survey"]["id"],us
er=current_user).first()

            start=datetime.strptime(res["start"].split(".")[0],'%Y-%m-%d
%H:%M:%S')


            end=datetime.strptime(res["end"].split(".")[0],'%Y-%m-%d %H:%M:%S')
            if start.date()>=datetime.now().date() and start.date() < end.date():
              price=res["price"]
              give_away=res["give_away"]
              n_event=models.Event(limit=res["limit"],user=current_user,survey=su
rvey,start=start,end=end,price=price,give_away=give_away)
              survey.status_id=1;
              db.session.add(n_event)
              db.session.commit()
              tags=[e["id"]  for e in res["tags"] if e["selected"]==True]
              if 1 in tags:
                  ev_tag=models.EventTag(tag_id=1,event_id=n_event.event_id)
                  db.session.add(ev_tag)
                  db.session.commit()


              if 2 in tags:
                  ev_tag=models.EventTag(tag_id=2,event_id=n_event.event_id)
                  db.session.add(ev_tag)
                  db.session.commit()

              if 3 in tags:
                  ev_tag=models.EventTag(tag_id=3,event_id=n_event.event_id)
                  db.session.add(ev_tag)
                  db.session.commit()

              if 4 in tags:
                  ev_tag=models.EventTag(tag_id=4,event_id=n_event.event_id)
                  db.session.add(ev_tag)
                  db.session.commit()

              if 5 in tags:
                  ev_tag=models.EventTag(tag_id=5,event_id=n_event.event_id)
```

```
                    db.session.add(ev_tag)

                    db.session.commit()



                headers={
                            'Content-Type': 'application/json',
                            'Authorization':'Bearer '+ get_access_token()
                    }
                x=round(res["total"]*0.000044,2)

                n_pay=models.Payment(user=current_user,event=n_event,total=res["tot
al"])


                year=str(datetime.now().year)

                month=str(datetime.now().month)

                day=str(datetime.now().day)

                hour=str(datetime.now().hour)

                minu=str(datetime.now().minute)

                second=str(datetime.now().second)

                data={
                  "intent": "sale",
                  "payer": {
                    "payment_method": "paypal"
                  },
                  "transactions": [
                    {
                      "amount": {
                        "total": str(round(x+0.05,2)),
                        "currency": "USD",
                        "details": {
                          "subtotal": str(x),
                          "tax": "0.01",
                          "shipping": "0.03",
                          "handling_fee": "1.00",
                          "shipping_discount": "-1.00",
                          "insurance": "0.01"
                        }
```

```
                    },
                    "description": "The payment transaction description.",
                    "custom": "EBAY_EMS_90048630024435",
                    "invoice_number":
str(n_pay.payment_id)+year+month+day+hour+minu+second,
                    "payment_options": {
                      "allowed_payment_method": "INSTANT_FUNDING_SOURCE"
                    },
                    "soft_descriptor": "ECHI5786786",
                    "item_list":
                     {
                      "items": [
                        {
                          "name": "event",
                          "description": "Brown hat.",
                          "quantity": "1",
                          "price": str(x),
                          "tax": "0.01",
                          "sku": "1",
                          "currency": "USD"
                        },

                      ],

                      "shipping_address":
                       {
                        "recipient_name": "Brian Robinson",
                        "line1": "4th Floor",
                        "line2": "Unit #34",
                        "city": "San Jose",
                        "country_code": "US",
                        "postal_code": "95131",
                        "phone": "011862212345678",
                        "state": "CA"
                      }
                    }
                  }
                ],

                "note_to_payer": "Contact us for any questions on your order.",
                "redirect_urls": {
                  "return_url": "https://5266-2001-ee0-5759-ba0-f410-9e54-795f-
1a.ngrok.io/checkout",
                  "cancel_url": "https://example.com/cancel"
                }
```

```
            }

            response =
requests.post("https://API.sandbox.paypal.com/v1/payments/payment",data=json.dump
s(data), headers=headers)

            paypal_pay_id=response.json()["id"]
            s=None

            for link in response.json()["links"]:

              if link["method"]=='REDIRECT':

                    s=link['href'].split("token=")[1]

                    checkout_token=s

            n_pay.paypal_pay_id=paypal_pay_id

            n_pay.checkout_token=checkout_token

            db.session.add(n_pay)

            db.session.commit()

            return  json.dumps({"pay_token":s}),200

          else:

              return  json.dumps({"error":"required id even data"}),200

       else:

           return  json.dumps({"error":"required id event data"}),200
```

- Route tham gia sự kiện khảo sát

```
@app.route('/doSurvey',methods=["POST"])
@cross_origin(origin='*')
@token_required
def doSurvey(current_user):
    if request.method=="POST":
        if current_user:
            event=request.get_json()["event"]
            if event:
```

```
                u_e=models.UserEvent.query.filter_by(user_id=current_user.user_id
,event_id=event["id"]).first()
                if u_e:
                    for q in event["survey"]["questions"]:
                        for a in q["answers"]:

                            answer=
models.Answer(question_id=q["id"],user_event_id=u_e.user_event_id,answer=a["conte
nt"])
                            if a['is_different']==True:
                                answer.is_different=True
                            u_e.do=True
                            db.session.add(answer)
                            db.session.commit()
                else:
                    return json.dumps({"error":"not invited"}),200
            else:
                return json.dumps({"error":"require data"}),200
        else:
            return json.dumps({"error":"require user"}),401
    return json.dumps({"message":"completed"}),200
```

- Route tải thông tin sự kiện khảo sát

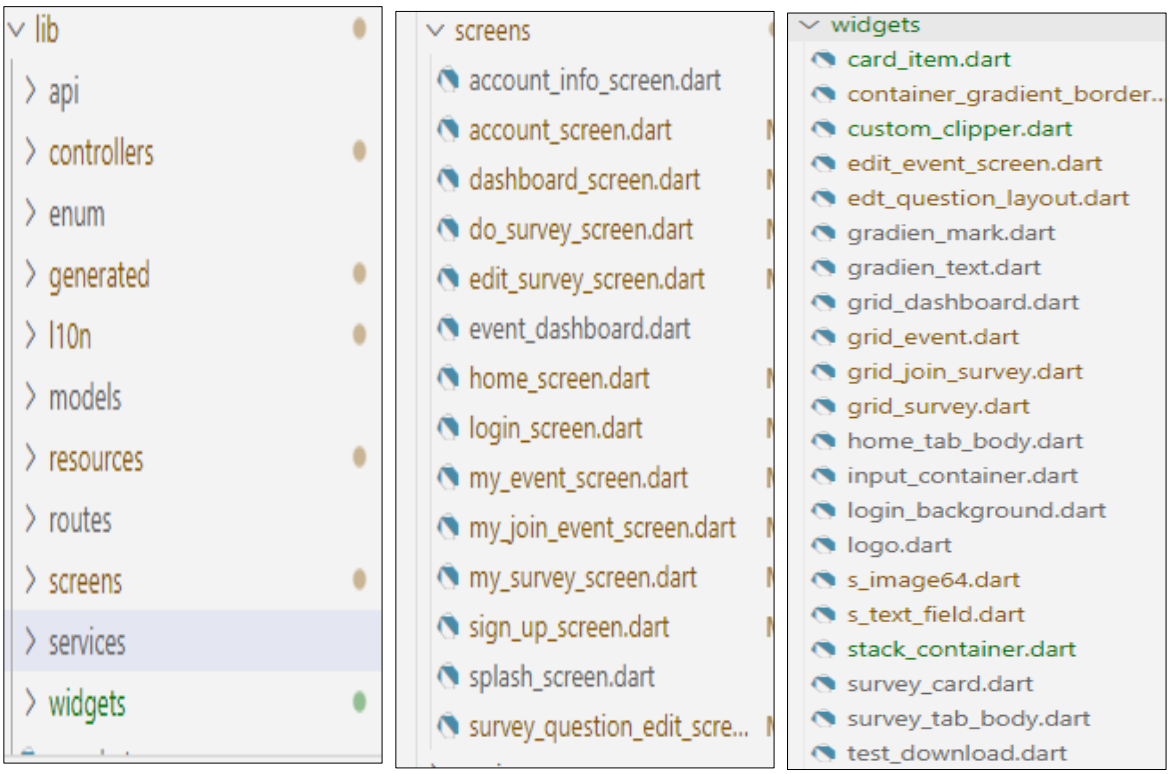
```
@app.route('/download',methods=["POST"])
def downloadFile():
    if request.method== "POST":
        if 'event_id' in request.get_json():
            event_id=request.get_json()["event_id"]
            result = db.session.execute(
            "SELECT user.user_name,user.sex,question.content, answer.question_id"
            +" FROM "+
                "event,user_event,answer,question,user"
            +" WHERE user_event.event_id=event.event_id "
                +" and event.event_id={}".format(event_id)
                +" and answer.user_event_id=user_event.user_event_id"
                +" and question.question_id=answer.question_id"
                +" and user_event.user_id=user.user_id "+
            "group by user_event.event_id,answer.question_id")
            key = ','.join([str(elem) for elem in result.keys()])
            val = ',\n'.join([str(elem).replace(")", "").replace("(", "") for
elem in result])
            data=key+"\n"+val+"\n"
```

```
            s=np.fromstring(data, sep=',')
            return Response(
                data,
                mimetype="text/csv",
                headers={"Content-disposition":
                    "attachment; filename=myplot.csv"}),200
```

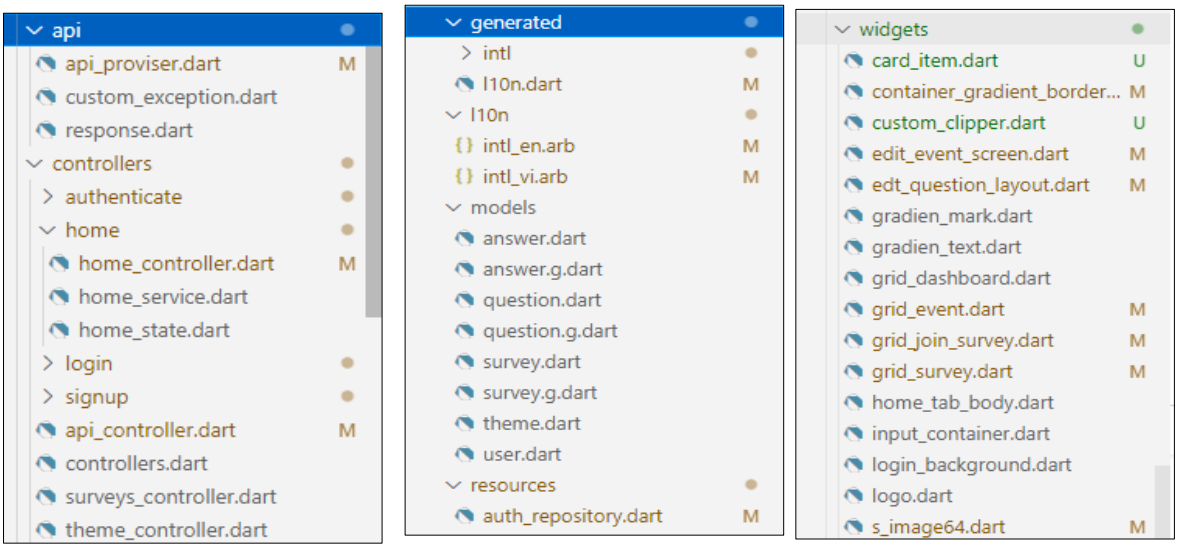
- Route nhận Incoming Webhooks từ Paypal

```
@app.route('/checkout',methods=["GET"])
@cross_origin(origin='*')
def checkout():
    payment_id = request.args.get('paymentId')
    payer_id=request.args.get('PayerID')
    token=request.args.get('token')
    headers={
                        'Content-Type': 'application/json',
                        'Authorization':'Bearer '+ get_access_token()
                }
    data={
        "payer_id":payer_id
    }
    payment=models.Payment.query.filter_by(paypal_pay_id=payment_id).first()
    if payment:
        response =
requests.post("https://API.sandbox.paypal.com/v1/payments/payment/"+payment_id+"/
execute",data=json.dumps(data), headers=headers)
        if "id" in response.json():
            payment.state=True
            db.session.commit()
        return  json.dumps(response.json()),200
    else:
       return json.dumps({"error":"payment error"}),200
```

2. ỨNG DỤNG DI ĐỘNG



Hình 27: Cấu trúc thư mục ứng dụng di động a,b,c



Hình 28: Cấu trúc thư mục ứng dụng di động d, e, f

2.1. Sơ đồ cấu trúc

Bảng 22: Bảng mô tả cấu trúc thư mục ứng dụng

STT	Tên thư mục	Mô Tả
1	API	Thư mục quản lý file cấu hình kết nối với API
2	controller	Thư mục quản lý các file quản lý trạng thái ứng dụng,giao tiếp với API
3	models	Thư mục quản lý các file đối tượng
5	screens	Thư mục quản lý file giao diện màn hình
6	widgets	Thư mục quản lý file widget
7	Resources	Thư mục quản lý file quản lý lưu trữ
10	generated	Thư mục quản lý các file xử lý đa ngôn ngữ
11	L10n	Thư mục quản lý các file ngôn ngữ

2.2. Code xử lý

2.2.1. Thiết lập các quyền cần thiết của ứng dụng

```
<uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
android:maxSdkVersion="29"/>
  <uses-permission android:name="android.permission.QUERY_ALL_PACKAGES"/>
  <uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    android:maxSdkVersion="29"/>
    <uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.USE_FINGERPRINT"/>
```

2.2.2. Thiết lập việc gọi API

- Thiết lập file API_prosiver làm việc trực tiếp với API thông qua phương thức chính là POST

```
Future<Map<String, dynamic>> post(String url, dynamic body,
      {String token = ''}) async {
    dynamic responseJson;
    AuthRepository authRepository = AuthRepository();
    var token = await authRepository.fetchToken();
```

```
    try {
      final dynamic response =
          await http.post(Uri.parse(url), body: body, headers: {
        'content-type': 'application/json',
        "Access-Control_Allow_Origin": "*",
        "Authorization": "basic",
        "X-Authorization": token
      });

      responseJson = _response(response);
    } on SocketException {
      throw FetchDataException(S.current.domain_does_not_exist);
    }
    return responseJson;
  }
```

2.2.2. Thiết bản địa hoá

```
localizationsDelegates: [
          S.delegate,
          GlobalMaterialLocalizations.delegate,
          GlobalWidgetsLocalizations.delegate,
          GlobalCupertinoLocalizations.delegate,
        ]
```

2.2.3. Thiết lập đa ngôn ngữ với thư viện và extension của intl

- Pubspec.yaml

```
intl: ^0.17.0
```

- File ngôn ngữ tiếng anh intl_en.arb

```
{
    "@@locale": "en",
    "add_contact_success":"Add contact successfully",
    "retry":"Retry",
    "no_internet":"No internet connection, please try again",
    "close_app":"Close app",
    "add_contact":"Add Contact",
    "domain_does_not_exist":"Domain does not exist",
    "database_does_not_exist":"Database does not exist",
    "username_or_password_is_incorrect":"Username or Password is incorrect",
```

```
    "check_domain":"Check Domain",
    "does_not_have_account":"Does not have account ?",
    "already_user":"Already a user",
    "email_hint":"Email",
    "error":"Error, close the app and try again",
    "name_hint":"Name",
    "new_user":"A new user?",
    "no_results":"No results",
    "ok":"Ok",
    "password_hint":"Password",
    "please_wait":"Please Wait...",
    "sign_in":"Sign In",
    "sign_up":"Sign Up",
    "signup_success":"Successfully signed up, please Sign In",
    "user_name_hint":"User Name",
    "chose_id_recognition":"chose your id recognition",
    "password_again":"Enter the password again",
    "save":"Save",
  "add_to_event":"Add to event",
  "download":"Download",
  "create_new":"Create new",
  "event":"Event",
  "question":"Question",
  "adding":"Add",
  "your":"Your",
  "survey":"Survey",
  "sign_out":"Sign Out",
  "account":"Account",
  "allow_diffrent_answer":"Allow different answer",
  "option":"option"
  ,"add_survey_name":"Add survey name"
  ,"description":"Add description"
  ,"limit":"Limit",
  "checkout":"Checkout",
  "youth":"Youth",
  "middle_age":"Middle Age",
  "elder":"Elder",
  "male":"Male",
  "female":"Female",
  "all":"All",
  "dont_have_an_account":"Don't have an account?",
  "create_an_account":"Create an account" }
```

- File ngôn ngữ tiếng việt intl_vi.arb

```
{
```

```
  "@@locale": "vi",
  "add_contact_success":"Thêm liên hệ thành công",
  "retry": "Thử lại",
   "no_internet":"Không có kết nối,vui lòng thử lại",
   "close_app": "Đóng ứng dụng",
   "add_contact": "Thêm Liên Hệ",
   "database_does_not_exist":"Database không tồn tại",
   "domain_does_not_exist":"Tên miền không tồn tại",
   "username_or_password_is_incorrect":"Tên đăng nhập hoặc mật khẩu không hợp
lệ",
   "check_domain": "Kiểm Tra",
   "does_not_have_account":"Chưa có tài Khoản ?",
   "already_user":"Người dùng đã tồn tại",
   "email_hint": "Email",
   "error": "Đã xảy ra lỗi,vui lòng đóng ứng dụng và thử lại sau",
   "name_hint": "Tên",
   "new_user": "Người dùng mới?",
   "no_results": "Không có kết quả",
   "ok": "Ok",
   "password_hint": "Mật khẩu",
   "please_wait":"Vui lòng chờ trong giây lát",
   "sign_in": "Đăng Nhập",
   "sign_up": "Đăng Ký",
   "signup_success": "Đăng ký thành công, vui lòng Đăng nhập ",
   "user_name_hint": "Tên Đăng Nhập",
   "chose_id_recognition":"Chọn ảnh căn cước công dân",
   "password_again":"Nhập lại mật khẩu",
   "save":"Lưu",
   "add_to_event":"Thêm vào sự kiện",
   "download":"Tải xuống",
   "create_new":"Tạo mới",
   "event":"Sự kiện",
   "question":"câu hỏi",
   "adding":"Thêm",
   "your":"Của Bạn",
   "survey":"Khảo sát",
   "sign_out":"Đăng xuất",
   "account":"Tài khoản",
   "allow_diffrent_answer":"chấp thuận ý kiến khác",
   "option":"Lựa chọn"
   ,"add_survey_name":"Thêm tên khảo sát"
   ,"description":"Thêm mô tả"
   ,"limit":"Giới Hạn",
   "checkout":"Thanh Toán",
   "youth":"Thanh Niên",
```

```
    "middle_age":"Trung Niên",
    "elder":"Cao niên",
    "male":"Nam",
    "female":"Nữ",
    "all":"Tất Cả",
    "dont_have_an_account":"không có tài khoản?",
    "create_an_account":" Tạo mới"
}
```

2.2.4. Quản lý trạng thái của ứng dụng và của màn hình bằng thư viện GetX

-Gồm 3 file authenticate_controller.dart ,authenticate_service.dart và authenticate_sate.dart

- File authenticate_sate.dart định nghĩ các trạng thái xác thực

```
import 'package:equatable/equatable.dart';
import 'package:survey_app/models/user.dart';

class AuthenticationState extends Equatable {
  const AuthenticationState();

  @override
  List<Object> get props => [];
}

class AuthenticationLoading extends AuthenticationState {}

class UnAuthenticated extends AuthenticationState {}

class Authenticated extends AuthenticationState {
  final User user;

  Authenticated({required this.user});

  @override
  List<Object> get props => [user];
}

class AuthenticationFailure extends AuthenticationState {
  final String message;

  AuthenticationFailure({required this.message});
```

```
  @override
  List<Object> get props => [message];
}
```

➢ File authenticate_service.dart thiết lập các chức năng cho việc xác thực

```
import 'package:get/get.dart';
import 'package:survey_app/API/response.dart';
import 'package:survey_app/resources/auth_repository.dart';

abstract class AuthenticateService extends GetxService {
  Future<String?> getCurrentToken();
  Future<DataResponse<String>> signInWithEmailAndPassword(
      String email, String password, String? key);
  Future<void> signOut();
}

class FAuthenticateService extends AuthenticateService {
  AuthRepository authRepository = new AuthRepository();
  @override
  Future<String?> getCurrentToken() async {
    // simulated delay
    var token = await authRepository.fetchToken();
    return token;
  }

  @override
  Future<DataResponse<String>> signInWithEmailAndPassword(
      String email, String password, String? key) async {
    var token = await authRepository.signIn(email, password, key);

    return token;
  }

  @override
  Future<void> signOut() async {}
}

class AuthenticationException implements Exception {
  final String message;

  AuthenticationException({this.message = 'Unknown error occurred. '});
}
```

➢ File authenticate_controller.dart quả lý trạng thái xác thực ứng dụng

```
import 'package:get/get.dart';
import 'package:survey_app/controllers/authenticate/authenticate_service.dart';
import 'package:survey_app/models/user.dart';
import 'package:survey_app/resources/auth_repository.dart';

import 'authentication.dart';

class InitialAuthenticateStateBinding implements Bindings {
  @override
  void dependencies() {
    Get.lazyPut(() => AuthenticationState(), fenix: true);
  }
}

class AuthenticateController extends GetxController {
  final AuthenticateService authenticateService;
  final _authenticationStateStream = AuthenticationState().obs;
  AuthRepository authRepository = new AuthRepository();

  AuthenticationState get state => _authenticationStateStream.value;
  AuthenticateController(this.authenticateService);
  @override
  void onInit() async {
    InitialAuthenticateStateBinding().dependencies();
    _getAuthenticatedUser();
    super.onInit();
  }

  Future<void> signIn(String email, String password, String? key) async {
    final data = await authenticateService.signInWithEmailAndPassword(
        email, password, key);

    if (data.data.toString() == "null" || data.data.toString().isEmpty) {
      _authenticationStateStream.value =
          AuthenticationFailure(message: data.message.toString());
    } else {
      User user = User(name: "curUs", email: data.data.toString());
      _authenticationStateStream.value = Authenticated(user: user);
    }
  }

  void signOut() async {
    AuthRepository authRepository = AuthRepository();
    await authRepository.persistUser("");
```

```
    _authenticationStateStream.value = UnAuthenticated();
  }

  void _getAuthenticatedUser() async {
    _authenticationStateStream.value = AuthenticationLoading();

    var data = await authenticateService.getCurrentToken();

    if (data == "") {
      _authenticationStateStream.value = UnAuthenticated();
    } else {
      User user = User(name: "curUs", email: data.toString());
      _authenticationStateStream.value = Authenticated(user: user);
    }
  }
}
```

- File main khởi tạo các route, trạng thái ứng dụng,binding app state, location,background notification và foreground notification..

```
import 'package:flutter/material.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
import 'package:get/get.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:survey_app/controllers/authenticate/authenticate_service.dart';
import 'package:survey_app/controllers/authenticate/authenticate_state.dart';

import 'package:survey_app/routes/routes_generator.dart';
import 'package:survey_app/screens/dashboard_screen.dart';

import 'package:survey_app/screens/do_survey_screen.dart';
import 'package:survey_app/screens/event_dashboard.dart';

import 'package:survey_app/screens/home_screen.dart';
import 'package:survey_app/screens/login_screen.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_messaging/firebase_messaging.dart';

import 'package:flutter/foundation.dart';
import 'package:flutter_local_notifications/flutter_local_notifications.dart';
import 'package:survey_app/screens/my_event_screen.dart';
import 'package:survey_app/screens/my_join_event_screen.dart';
import 'package:survey_app/screens/my_survey_screen.dart';
import 'package:survey_app/screens/sign_up_screen.dart';
```

```dart
import 'package:survey_app/screens/splash_screen.dart';
import 'package:survey_app/screens/survey_question_edit_screen.dart';
import 'package:survey_app/widgets/edit_event_screen.dart';
import 'package:survey_app/widgets/test_download.dart';

import 'controllers/authenticate/authenticate_controller.dart';
import 'generated/l10n.dart';

Future<void> _firebaseMessagingBackgroundHandler(RemoteMessage message) async {
  // If you're going to use other Firebase services in the background, such as
Firestore,
  // make sure you call `initializeApp` before using other Firebase services.
  await Firebase.initializeApp();
  print('Handling a background message ${message.messageId}');
  print(message.data.length);
}
const AndroidInitializationSettings initializationSettingsAndroid =
    AndroidInitializationSettings('app_icon');
AndroidNotificationChannel? channel;
/// Initialize the [FlutterLocalNotificationsPlugin] package.
FlutterLocalNotificationsPlugin? flutterLocalNotificationsPlugin;
RouteGenerator routeGenerator = RouteGenerator();
void initialize() async {
  Get.lazyPut(() => AuthenticateController(Get.put(FAuthenticateService())),
      fenix: true);
}
void main() async {
  InitialBinding().dependencies();
  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp();

  // Set the background messaging handler early on, as a named top-level function
  FirebaseMessaging.onBackgroundMessage(_firebaseMessagingBackgroundHandler);

  if (!kIsWeb) {
    channel = const AndroidNotificationChannel(
      'high_importance_channel', // id
      'High Importance Notifications', // title
      'This channel is used for important notifications.', // description
      importance: Importance.high,
    );

    flutterLocalNotificationsPlugin = FlutterLocalNotificationsPlugin();
```

```
  /// Create an Android Notification Channel.
  ///
  /// We use this channel in the `AndroidManifest.xml` file to override the
  /// default FCM channel to enable heads up notifications.
  FirebaseMessaging _firebaseMessaging = FirebaseMessaging.instance;
  await flutterLocalNotificationsPlugin!
      .resolvePlatformSpecificImplementation<
          AndroidFlutterLocalNotificationsPlugin>()
      ?.createNotificationChannel(channel!);

  /// Update the iOS foreground notification presentation options to allow
  /// heads up notifications.

  await FirebaseMessaging.instance
      .setForegroundNotificationPresentationOptions(
    alert: true,
    badge: true,
    sound: true,
  );
}

String? token = await FirebaseMessaging.instance.getToken();

print(token);
if (!kIsWeb) {
  FirebaseMessaging.onMessage.listen((RemoteMessage message) async {
    RemoteNotification? notification = message.notification;
    AndroidNotification? android = message.notification?.android;
    if (notification != null && android != null) {
      flutterLocalNotificationsPlugin!.show(
          notification.hashCode,
          notification.title,
          notification.body,
          NotificationDetails(
            android: AndroidNotificationDetails(
              channel!.id,
              channel!.name,
              channel!.description,
              icon: 'ic_launcher',
            ),
          ));
    }
  });
}
// Get a specific camera from the list of available cameras.
```

```
  runApp(App());
}

class App extends GetWidget<AuthenticateController> {
  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
        onInit: () {
          InitialBinding().dependencies();
        },
        getPages: [
          GetPage(name: '/', page: () => HomeScreen()),
          GetPage(name: '/login', page: () => LogInScreen()),
          GetPage(name: '/signUp', page: () => SignUpScreen()),
          GetPage(name: '/mySurveys', page: () => MySurveyScreen()),
          GetPage(
              name: '/createSurvey', page: () => SurveyQuestionEditScreen()),
          GetPage(name: '/createEvent', page: () => EditEventScreen()),
          GetPage(name: '/myEvents', page: () => MyEventScreen()),
          GetPage(name: '/joinEvents', page: () => MyJoinEventScreen()),
          GetPage(name: '/doSurvey', page: () => DoSurveyScreen()),
          GetPage(name: '/splash', page: () => SplashScreen()),
          GetPage(name: '/dashboard', page: () => DashboardScreen())
        ],
        localizationsDelegates: [
          S.delegate,
          GlobalMaterialLocalizations.delegate,
          GlobalWidgetsLocalizations.delegate,
          GlobalCupertinoLocalizations.delegate,
        ],
        initialBinding: InitialBinding(),
        supportedLocales: S.delegate.supportedLocales,
        debugShowCheckedModeBanner: false,
        theme: ThemeData(
          fontFamily: GoogleFonts.openSans().fontFamily,
        ),
        defaultTransition: Transition.rightToLeft,
        home: GetX<AuthenticateController>(
          init: AuthenticateController(Get.put(FAuthenticateService())),
          builder: (controller) {
            if (controller.state is UnAuthenticated ||
                controller.state is AuthenticationFailure) {
              return LogInScreen();
            }
```

```
            if (controller.state is Authenticated) {
              return HomeScreen();
            }
            return SplashScreen();
          },
        ));
  }
}
class InitialBinding implements Bindings {
  @override
  void dependencies() {
    Get.lazyPut(() => AuthenticateController(Get.put(FAuthenticateService())),
        fenix: true);
  }}
```